HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

GIÁO TRÌNH

Kỹ thuật vi xử lý

TẬP 2

NHÀ XUẤT BẢN BƯU ĐIỆN

GIÁO TRÌNH

Kỹ thuật vi xử lý

(TẬP 2)

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

GIÁO TRÌNH

Kỹ thuật vi xử lý

(TẬP 2)

Biên soạn: **TS. Hồ Khánh Lâm**

NHÀ XUẤT BẢN BƯU ĐIỆN
Hà Nội - 2007

Mã số: GT 02 HM 07

LỜI GIỚI THIỆU

Chiến lược phát triển công nghệ thông tin và truyền thông (ICT) của đất nước ta coi đào tạo nguồn nhân lực, đặc biệt là đội ngũ kỹ sư hệ thống công nghệ điện tử - viễn thông - tin học là một yêu cầu cấp bách đối với công tác nghiên cứu và giáo dục đào tạo ở hệ cao đẳng và đại học. Kỹ thuật vi xử lý là kiến trúc lõi cơ bản nhất trong hầu hết các hệ thống thiết bị điện tử, viễn thông và tự động hóa, cũng như mạng lưới và các hệ thống xử lý thông tin hiện đại khác. Do vậy, hiểu biết "Kỹ thuật vi xử lý" trở thành nhu cầu không thể thiếu của các kỹ thuật viên, cán bộ, chuyên gia trong lĩnh vực Điện tử - Viễn thông - Tin học.

Cùng với nhiều tài liệu, sách tham khảo "Kỹ thuật vi xử lý" đang lưu hành, Học viện Công nghệ Bưu chính Viễn thông và tác giả cho ra đời quyển giáo trình "Kỹ thuật vi xử lý" này trên cơ sở tuân thủ nội dung của một giáo trình, được Hội đồng nghiệm thu phê duyệt, với nhiều nội dung tham khảo phong phú nhằm mở rộng kiến thức cho nhiều loại đối tượng sử dụng trong giảng dạy, học tập và nghiên cứu: giảng viên, sinh viên các trường Đại học, Cao đẳng, các trường Kỹ thuật dạy nghề, các học viên cao học và nói chung những ai khi học tập nghiên cứu về kỹ thuật vi xử lý, máy vi tính nói riêng và kỹ thuật tin học nói chung.

Nội dung của giáo trình đề cập những khái niệm cơ bản về vi xử lý, về công nghệ của một số vi xử lý thông dụng từ 8-bit đến 64-bit, từ kiến trúc tập lệnh phức tạp (CISC) đến kiến trúc tập lệnh giảm thiểu (RISC), được các nhà sản xuất hàng đầu thế giới như Intel, Zilog, Motorola, HP, IBM, Sun,... tạo ra; diễn giải các tập lệnh của các vi xử lý thông qua lập trình hợp ngữ, các nguyên tắc phối ghép và trao đổi thông tin giữa vi xử lý với

thế giới vật lý bên ngoài (bộ nhớ, thiết bị ngoại vi,...); trình bày các loại thiết bị nhớ bằng vật liệu bán dẫn, vật liệu từ, quang học. Giáo trình tập trung trình bày kỹ công nghệ vi xử lý Intel từ 808x đến Pentium M công nghệ Centrino Mobile do sự ứng dụng phổ biến của chúng trong các thế hệ máy vi tính từ trước đến nay và cũng để đảm bảo thỏa mãn đa số nội dung giảng dạy và thực tập môn kỹ thuật vi xử lý trong các cơ sở đào tạo. Các loại vi điều khiển và xử lý tín hiệu cũng được trình bày một cách cơ bản với những ví dụ ứng dụng đơn giản của chúng. Giáo trình có phần bài tập giúp cho người đọc ôn luyện kiến thức dễ dàng.

Nội dung của giáo trình thể hiện sự đầu tư sức lao động nghiêm túc, công phu cùng với kinh nghiệm thực tế và đào tạo của tác giả trong lĩnh vực "Kỹ thuật vi xử lý" và máy tính điện tử nhiều năm với mong muốn đóng góp cho sự nghiệp đào tạo của ngành Bưu chính Viễn thông và công nghệ thông tin nói riêng và giáo dục đào tạo nói chung trong sự nghiệp phát triển nguồn nhân lực trí tuệ cho đất nước.

Xin trân trọng giới thiệu.

GS. TSKH. ĐỖ TRUNG TÁ

LỜI NÓI ĐẦU

Ngày nay, bộ vi xử lý được sử dụng rộng rãi không chỉ trong lĩnh vực máy tính điện tử mà còn trong rất nhiều hệ thống điều khiển khác. Với sự phát triển của công nghệ mạch tích hợp một bộ vi xử lý có thể được chế tạo từ một hoặc vài vi mạch tích hợp cỡ lớn, chứa hàng ngàn hoặc hàng triệu transistor. Dưới sự điều khiển của các chương trình chứa trong bộ nhớ, bộ vi xử lý thực hiện các phép tính số học, logic, các phép toán khác và kết nối, trao đổi dữ liệu với các thiết bị bên ngoài thông qua các cổng vào/ra. Do vậy, vấn đề đặt ra là phải hiểu về kỹ thuật vi xử lý và lập trình cho vi xử lý như thế nào để tự động điều khiển mang lại hiệu quả cao.

Để đáp ứng yêu cầu trên, Học viện Công nghệ Bưu chính Viễn thông phối hợp với Nhà xuất bản Bưu điện xuất bản cuốn sách ***"Giáo trình Kỹ thuật Vi xử lý"*** *do TS. Hồ Khánh Lâm chủ biên. Giáo trình gồm 8 chương, chia làm 02 tập. Tập 1 từ chương 1 đến chương 4 giới thiệu những khái niệm cơ bản về hệ thống vi xử lý như kiến trúc Von Neumann, kiến trúc Harvard; đặc điểm, cấu trúc và đặc tính nâng cao tốc độ của bộ vi xử lý; cấu trúc và hoạt động của họ vi xử lý từ 8085, 8086/8088, 8087, 80x86, Z80, đến Pentium III, IV, Motorola68000,... Bên cạnh đó cuốn sách không chỉ giới thiệu về lệnh, các dạng lệnh, cách mã hóa lệnh, các mã lệnh và tập lệnh của vi xử lý mà còn giới thiệu cụ thể về các khái niệm, cách thức lập trình hợp ngữ, cơ chế làm việc, cấu trúc của chương trình con, macro và các vấn đề liên giúp cho người lập trình tự tạo được những lệnh điều khiển*

Chương 5
GHÉP NỐI CPU VỚI BỘ NHỚ VÀ THIẾT BỊ NGOẠI VI

5.1. CÁC TÍN HIỆU CỦA BỘ VI XỬ LÝ VÀ CÁC MẠCH PHỤ TRỢ

Để đánh giá tính chất phức tạp của vấn đề ghép nối CPU với bộ nhớ và thiết bị ngoại vi phụ thuộc vào mức độ công nghệ của chip vi xử lý, ta chọn công nghệ Intel 16-bit 8086/8088 và công nghệ 32-bit Pentium để khảo sát.

5.1.1. Các tín hiệu của vi xử lý 8086/8088 và các mạch phụ trợ

a) Các tín hiệu

Hình 2.75 phân loại các nhóm tín hiệu theo nhóm chức năng và ký hiệu ở chế độ tối đa để trong ( ) như sau:

(1) Điều khiển hệ thống:

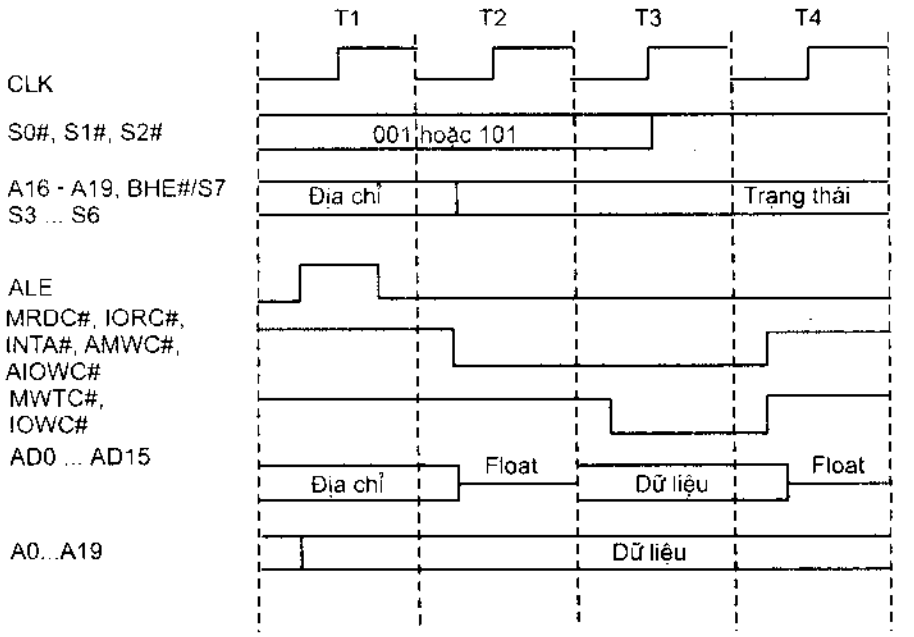
DEN# (S0#), DT/R# (S1#), IO/M# (S2#), RD#, WR# (LOCK#), SS0#, READY: là các tín hiệu điều khiển vào\ra với bộ nhớ và ngoại vi.

S2, S1, và S0 trong chế độ tối đa, được vi xử lý tạo ra ở đầu chu kỳ nhịp đồng hồ T1, là 3 tín hiệu đầu vào của mạch điều khiển bus 8288 để giải mã tạo các tín hiệu lệnh cho điều khiển hệ thống ở bắt đầu của chu kỳ nhịp đồng hồ T3 và duy trì trong chu kỳ T3: đọc mã lệnh từ bộ nhớ (MRDC#), ghi ra bộ nhớ (MWTC#), ghi nâng cấp ra bộ nhớ (AMWC#), đọc từ cổng ngoại vi (IORC#), ghi ra cổng ngoại vi (IOWC#), ghi nâng cấp ra ngoại (AIOWC#), và chấp nhận yêu cầu ngắt INTA# (bảng 5.1). AMWC# và AIOWC# tích cực sớm hơn trong chu kỳ ghi để báo sớm cho các thiết bị ngoại vi biết về thao tác ghi.

Bảng 5.1: Tín hiệu lệnh điều khiển của 8288

S2#	S1#	S0#	Tín hiệu lệnh của 8288	Trạng thái của vi xử lý
0	0	0	INTA#	Chấp nhận yêu cầu ngắt INTR
0	0	1	IORC#	Đọc từ cổng ngoại vi
0	1	0	IOWC#, AIOWC#	Ghi ra cổng ngoại vi
0	1	1	-	Dừng (HALT)
1	0	0	MRDC#	Đọc mã lệnh từ bộ nhớ
1	0	1	MRDC#	Đọc từ bộ nhớ
1	1	0	MWTC#, AMWC#	Ghi ra bộ nhớ
1	1	1	-	Không tác dụng gì cả

Các tín hiệu MRDC#, IORC#, INTA#, AMWC#, AIOWC# xuất hiện trong chu kỳ nhịp T2 và kéo dài đến hết chu kỳ nhịp T3 đầu T4. Các tín hiệu MWTC#, và IOWC# xuất hiện trong chu kỳ nhịp T3 (hình 5.1).



Hình 5.1: Các tín hiệu điều khiển hệ thống được 8288 tạo ra nhờ giải mã S0#, S1#, S2#

IO/M#: Chỉ ra có địa chỉ tham chiếu bộ nhớ hoặc tham chiếu ngoại vi, có sự khác nhau giữ 8086 và 8088:

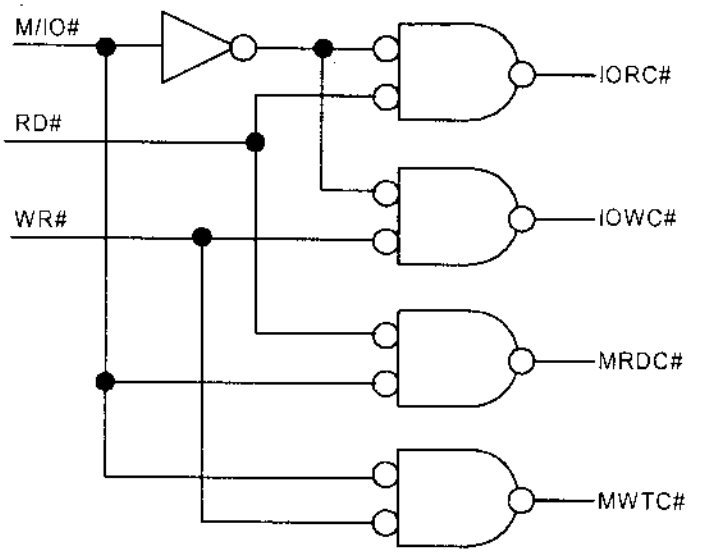
8086: M/IO#: Tham chiếu bộ nhớ M/IO# = 1, tham chiếu I/O M/IO# = 0

8088: IO/M#: Tham chiếu bộ nhớ IO/M# = 0, tham chiếu I/O IO/M# = 1

RD#: Đọc dữ liệu từ bus dữ liệu hệ thống ở mức tích cực thấp.

WR#: Ghi dữ liệu ra bus dữ liệu hệ thống ở mức tích cực thấp.

Trong chế độ tối thiểu các tín hiệu M/IO#, RD# và WR# thường được giải mã để tạo các tín hiệu điều khiển ghi đọc ngoại vi và bộ nhớ (hình 5.2).



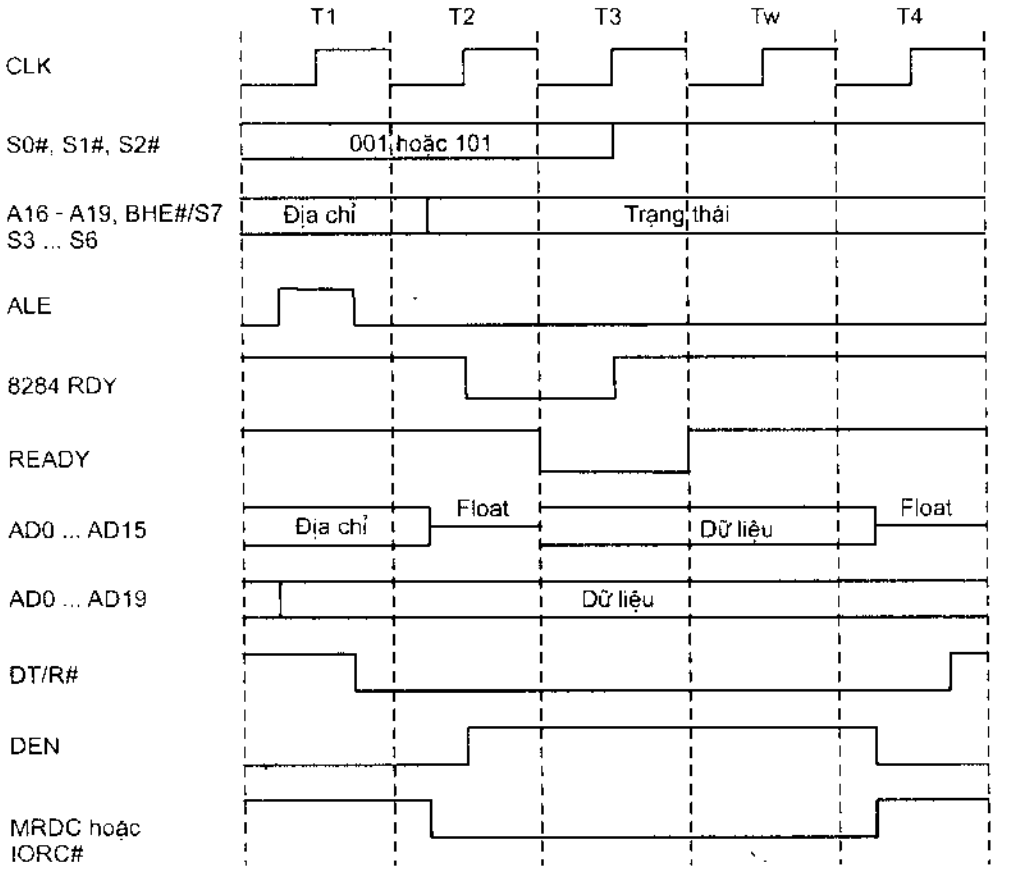
Hình 5.2: Giải mã tạo điều khiển ghi đọc ngoại vi và bộ nhớ trong chế độ tối thiểu

Trong chế độ tối đa WR# là tín hiệu LOCK#. LOCK# = 0 báo cho thiết bị khác biết rằng thiết bị đó không thể chiếm đoạt điều khiển bus hệ thống được. LOCK# tích cực ở mức 0 nhờ mào đầu của lệnh LOCK với mã là F0H. LOCK# duy trì tích cực 0 cho đến khi có lệnh tiếp theo.

DEN# (Data bus ENable: Cho phép bus dữ liệu): được sử dụng để kích hoạt các mạch thu phát dữ liệu và đệm dữ liệu cho bus dữ liệu

hệ thống. Trong chu kỳ đọc hoặc ghi (hình 2.81) DEN# = 0, ALE = 0, và RD# = 0 hoặc WR# = 0 đảm bảo đưa dữ liệu lên bus dữ liệu (cuối chu kỳ nhịp đồng hồ T2 đối với ghi và cuối chu kỳ nhịp T3 đối với đọc).

DT/R# (Data Transmit/Receiver (Phát/thu dữ liệu): để điều khiển các mạch lái bus 2 chiều (bit-directional bus drivers) kết hợp với DEN#. Ví dụ DT/R# và DEN# điều khiển các mạch thu phát dữ liệu bus 74HC245. Khi DT/R# = 0 tức là nhận dữ liệu, DT/R# = 1 tức là phát dữ liệu.



Hình 5.3: Chu kỳ đọc có chờ đợi của 8086

READY: Tín hiệu vào báo sẵn sàng: chỉ trạng thái sẵn sàng đọc hoặc ghi dữ liệu từ bộ nhớ hay thiết bị ngoại vi. Ở sườn lên của nhịp đồng hồ T3 vi xử lý nhận READY. Với READY =.0, không sẵn sàng,

vi xử lý phát sinh thời gian chờ bằng sự chèn thêm chu kỳ nhịp Tw vào tiếp sau T3. Trong trạng thái chờ, READY được vi xử lý lấy mẫu lần nữa bằng sườn lên của nhịp đồng hồ, nếu READY = 0 thì tiếp tục chèn thêm một chu kỳ nhịp chờ Tw nữa.

Trong những trường hợp kết nối với RAM chậm thì đôi khi cần phải có một số chu kỳ nhịp chờ Tw (có thể xem sơ đồ hình 2.61 xét cho vi xử lý 8085). Hình 5.3 mô tả một chu kỳ đọc bộ nhớ chậm phải thêm 1 trạng thái chờ Tw giữa T3 và T4.

SS0# (8088) - là tín hiệu trạng thái trong chế độ tối thiểu và về logic tương tự như S0# trong chế độ tối đa.

BHE#/S7 (8086) (Bus High Enable BHE# - Cho phép bus cao): được sử dụng với bit địa chỉ A0 để chọn dữ liệu byte cao/thấp/hay toàn bộ 16-bit (D0-D15) của bus dữ liệu (S7 luôn bằng 1, không sử dụng). Đôi khi gọi A0 là BLE# (Bus Low Enable - Cho phép bus thấp). Cũng như các đường địa chỉ, BHE#/S7 được vi xử lý tạo ra ở đầu thời gian của chu kỳ T1 như sau:

BHE#	A0 (BLE#)	Chọn dữ liệu
0	0	Toàn bộ 16 bit dữ liệu
0	1	Các byte cao của dữ liệu (địa chỉ lẻ)
1	0	Các byte thấp của dữ liệu (địa chỉ chẵn)
1	1	-

(2) Điều khiển bus:

HOLD (RQ#/GT0#), HLDA (RQ#/GT1#), INTA#(QS1), ALE (QS0): là các tín hiệu điều khiển bus hệ thống.

ALE (Address Latch Enable - Cho phép chốt địa chỉ): trong chế độ tối thiểu. Xung ALE (lên 1 và xuống 0) trong chu kỳ nhịp T1 thực hiện chốt các đường địa chỉ thấp A0-A7 vào bộ đệm địa chỉ từ các đường chung AD0-AD7 trong 8088 và chốt A0-A15 trong 8086 từ các đường chung AD0-AD15. ALE = 0 (từ chu kỳ nhịp T2) để đảm bảo đọc/ghi dữ liệu trên bus dữ liệu D0-D7 trên các đường chung AD0-

AD7 (8088) hay D0-D15 trên các đường chung AD0-AD15 (8086). Các mạch chốt địa chỉ thường dùng là 74HC373 hoặc tương đương. Trong chế độ tối đa ALE được 8288 tạo ra để điểm nhịp (strobe) các trạng thái S0#, S1#, S3# vào trong mạch chốt để mã hóa trạng thái dừng (Halt), còn bản thân nó là QS0 - bit trạng thái hàng lệnh (queue status) nối với đồng xử lý 8087 (xem hình 2.86).

INTA#: chấp nhận yêu cầu ngắt cho tín hiệu vào INTR khi đã thực hiện xong lệnh hiện hành và cất giữ vào ngăn xếp. Trong chế độ tối đa nó là QS1- bit trạng thái của hàng lệnh nối với 8087, còn INTA# được 8288 tạo ra nhờ S0#, S1# và S2#.

HOLD (HOLD Request - Yêu cầu chiếm giữ bus): là tín hiệu vào, kết hợp với HLDA. Khi HOLD = 1 nó chỉ ra rằng một chủ khác (vi xử lý khác hay bộ điều khiển DMA) có yêu cầu sử dụng bus của 8086/8088. Trong chế độ tối đa HOLD trở thành tín hiệu yêu cầu/cho phép bus (RQ#/GT0#) nối với vi xử lý khác.

HLDA (HoLD Acknowledge - Chấp nhận chiếm giữ bus): HLDA = 1 chỉ 8086/8088 chấp nhận yêu cầu sử dụng bus hệ thống của nó. Trong chế độ tối đa HLDA là RQ#/GT1# nối với đồng xử lý 8087 (xem hình 2.86).

Các tín hiệu QS0, QS1, RQ#/GT0#, RQ#/GT1# và TEST# dùng cho kết nối 8086/8088 với các vi xử lý khác trong hệ thống đa xử lý hoặc với đồng xử lý 8087 với phân biệt 8086/8088 là chủ (master) còn 8087 là thợ (slave).

(3) Các đường địa chỉ cao nhất và trạng thái: A19/S6 - A16/S3

Trong chu kỳ nhịp T1, cũng như S2#, S1#, S0#, AD0 - AD15, BHE#/S7, các đường địa chỉ A19/S6 - A16/S3 được tạo ra để chọn bộ nhớ trong các chu kỳ đọc hoặc ghi (xem sơ đồ 2.80). Trong đó, BHE#/S7 và A0 để chọn băng nhớ lẻ hay chẵn, còn A1 - A19 để chọn ngăn nhớ trong băng. Xung ALE trong chu kỳ T1 chốt A16 - A19 vào mạch chốt địa chỉ từ các đường chung A16 - A19. Trong chu kỳ nhịp T2, khi đó ALE = 0, các đường chung A19/S6 - A16/S3 là các tín hiệu

trạng thái S3 - S6. Trong đó: S6 luôn bằng 0 (không sử dụng), và S5 trùng với trạng thái của bit cờ ngắt I trong thanh ghi cờ. S4 và S3 là 2 bit để giải mã chọn một trong các thanh ghi đoạn CS, DS, SS và ES, để địa chỉ chọn các băng 1 MB riêng biệt của bộ nhớ:

S4	S3	Chọn thanh ghi đoạn
0	0	ES (Extra segment - Đoạn mở rộng)
0	1	SS (Stack segment - Đoạn ngăn xếp)
1	0	CS (Code segment - Đoạn mã)
1	1	DS (Data segment - Đoạn dữ liệu)

(4) 08 đường địa chỉ cao: AD15 - AD8 (8086)

Được tạo ra trong chu kỳ nhịp T1 và được chốt địa chỉ bởi xung ALE để tách với D15-D8 trong thời gian từ chu kỳ nhịp T2.

(5) 08 đường địa chỉ và dữ liệu thấp: AD0 - AD7

Được tạo ra trong chu kỳ nhịp T1 và được chốt địa chỉ bởi xung ALE để tách với D7-D0 trong thời gian từ chu kỳ nhịp T2.

(6) Điều kiển CPU: NMI, INTR, RESET, NM/MX#, TEST#

NMI (Non Mask Interrupt): Tín hiệu vào ngắt không che được. Sườn xung NMI từ 0 đến 1 gây ra ngắt mà CPU phải thực hiện xử lý ngắt ngay sau khi kết thúc lệnh đang thực hiện. Bit cờ ngắt I không có ý nghĩa đối với NMI. Vì vậy, NMI thường dành cho những trường hợp sự cố khẩn cấp cần CPU phải xử lý ngay.

INTR (Interrupt request): Tín hiệu yêu cầu ngắt từ các thiết bị ngoại vi (yêu cầu được phục vụ), nhưng là che được nhờ cờ ngắt I. Khi cờ I = 0 (xóa nhờ lệnh CLI) thì INTR không được chấp nhận và phải chờ đợi. Khi cờ I = 1 (thiết lập nhờ lệnh STI), thì CPU sau khi thực hiện xong chu kỳ cuối cùng của lệnh đang thực thì nó sẽ đưa ra tín hiệu INTA# = 0 (trong chế độ tối thiểu) hoặc thông qua S0#, S1#, S2# và mạch điều khiển bus 8288 (trong chế độ tối đa).

MN/MX# (Min/Max): Thiết lập chế độ làm việc tối thiểu/tối đa, (MN/MX# = 1) hay tối đa (MN/MX# = 0).

TEST#: Tín hiệu vào kiểm tra, lệnh WAIT (chờ để kiểm tra) của 8086/8088 kiểm tra tín hiệu TEST#, nếu TEST# = 0 thì 8086/8088 tiếp tục công việc của mình, nếu TEST# = 1 thì 8086/8088 phải chờ đợi (xem hình 2.86).

RESET: Xung xóa vi xử lý, được lấy từ mạch tạo nhịp đồng hồ bên ngoài (8284) để xóa các thanh ghi bên trong vi xử lý, đặt địa chỉ lệnh đầu tiên phải thực hiện ở 0FFFF0h (IP = 0FFF0h, CS = 0F000h) sau khi kết thúc xung xóa.

(7) Đồng hồ: CLK

Nhịp đồng được tạo ra từ mạch tạo nhịp bên ngoài (8284) với dao động thạch anh. IBM PC/XT với CPU là 8086/8088 có nhịp đồng hồ với tần số 4,77 MHz.

b) Các mạch phụ trợ cho 8086/8088

Để kết nối thành một hệ thống vi xử lý, các bộ vi xử lý phải có những vi mạch phụ trợ (các vi mạch lập trình và không lập trình được) thì mới có thể thiết kế được một bộ xử lý trung tâm (CPU) hoàn chỉnh trong một máy vi tính hay một hệ thống máy tính nào đó. Sự liên kết giữa bộ vi xử lý với các khối bên ngoài thông qua bus hệ thống cần phải có các vi mạch giao tiếp, như: phát/thu dữ liệu 2 chiều, đệm dữ liệu, chốt địa chỉ, điều khiển bus, điều khiển trao đổi thông tin với thiết bị ngoại vi và bộ nhớ vào/ra, điều khiển ngắt, điều khiển truy nhập trực tiếp bộ nhớ chính DMA. Các chip vi mạch phụ trợ đi kèm với chip vi xử lý tạo thành một họ (family). Bảng 5.2 liệt kê một số mạch phụ trợ dùng cho các vi xử lý Intel, đặc biệt thông dụng cho 8080/8085/8086/8088/80286. Các bộ vi xử lý công nghệ cao hiện nay (từ 80386 đến Pentiums) đã có những vi mạch VLSI (Very Large Scale Integration) phụ trợ gộp nhiều chức năng điều khiển khác nhau tạo thành các Chipset.

Bảng 5.2 Các vi mạch hỗ trợ cho các bộ vi xử lý 8080/8085/8086/8088/80286

8087	Đồng xử lý toán học	40-pin DIP
8116	Đồng hồ kép	18-pin DIP
8202, 8203	Mạch điều khiển RAM động	40-pin DIP
8224/8284	Mạch tạo nhịp đồng hồ	16-pin DIP
8250, 8251 (UART)	Mạch điều khiển vào/ra (I/O) không đồng bộ	40-pin DIP
8253 (TIMER/CLOCK)	Mạch lập trình đếm thời thời gian/nhịp	24-pin DIP
8255 (PIO)	Mạch điều khiển vào/ra (I/O) 3 cổng song song	40-pin DIP
8259 (PIC)	Mạch điều khiển ngắt (Interrupt controller)	40-pin DIP
8237 (DMA)	Mạch điều khiển truy nhập trực tiếp bộ nhớ (DMA)	40-pin DIP
6845 (CRT)	Mạch điều khiển màn hình	40-pin DIP
PD765	Mạch điều khiển vào/ra (I/O) đĩa mềm	40-pin DIP
8755	Mạch điều khiển ghép nối BUS	40-pin DIP
8156	Mạch định thời gian lập trình	40-pin DIP
8288	Điều khiển Bus	20-pin DIP
8286	Thu/phát dữ liệu 8-bit hai chiều, 3 trạng thái	20-pin DIP
8282, 8212	Chốt 8-bit/lái bus, 3 trạng thái	20-pin DIP
74HC373, 74LS373	Chốt kiểu Flip-flop D 8-bit, 3 trạng thái	20-pin DIP
74HC245, 74LS245	Thu/phát bus dữ liệu 8-bit 2 chiều, 3 trạng thái	20-pin DIP
74HC244, 74LS244	Đệm 8-bit/lái bus, 3 trạng thái	20-pin DIP

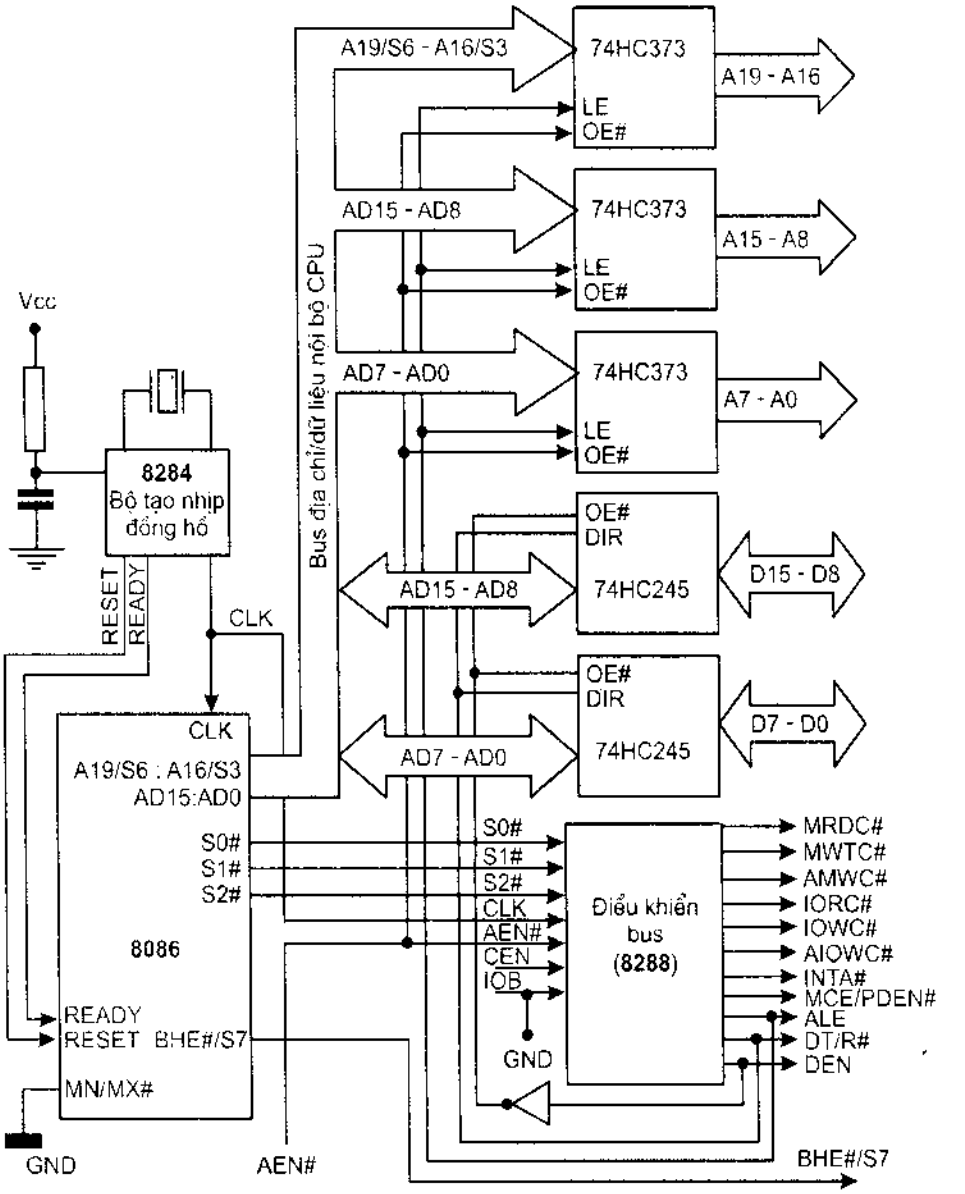
Các vi mạch hỗ trợ có thể lập trình cho điều khiển ghép nối I/O (điều khiển ngắt, điều khiển DMA, định thời gian và nhịp, điều khiển truyền thông, điều khiển đĩa, vào/ra với bàn phím...) và các thanh ghi trạng thái, điều khiển, thu phát dữ liệu của chúng đều được gán các địa chỉ I/O và được truy nhập, lập trình theo các địa chỉ I/O.

5.1.2. Kết nối các mạch phụ trợ với CPU 8086/8088

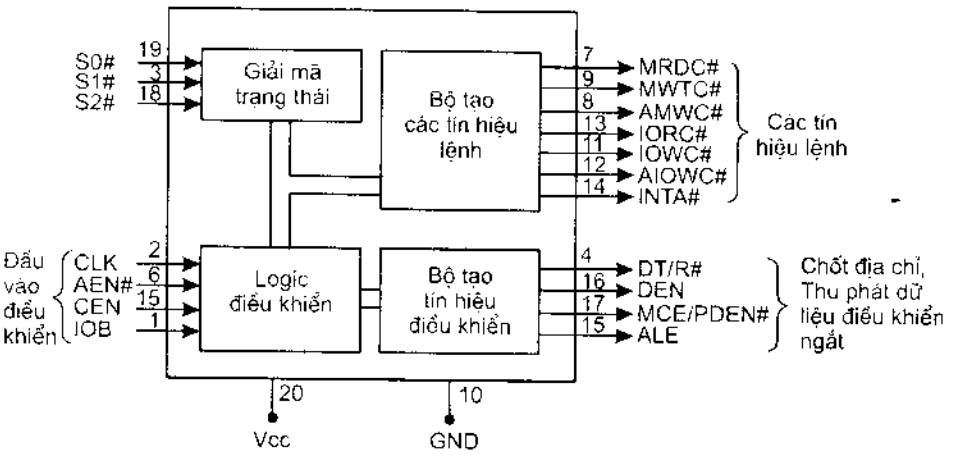
a) Trong chế độ tối đa

Trong chế độ tối đa, để tạo ra bus hệ thống cho kết nối các thiết bị ngoại vi và bộ nhớ (ROM, RAM) phải sử dụng các mạch phụ trợ: điều khiển bus 8288 và các mạch có 3 trạng thái: chốt địa chỉ (74HC373, 74LS373, hoặc 8282), thu phát dữ liệu 2 chiều (74HC245,

74LS245, hoặc 8286), đệm dữ liệu (74HC244, 74LS244). Hình 5.4 là sơ đồ tạo bus hệ thống nhờ các mạch 8288, 74HC245 và 74HC373. Hình 5.5 là sơ đồ khối chức năng của 8288.



Hình 5.4: Kết nối CPU 8086 và các mạch phụ trợ tạo bus hệ thống



Hình 5.5: Sơ đồ khối chức năng của điều khiển bus 8288

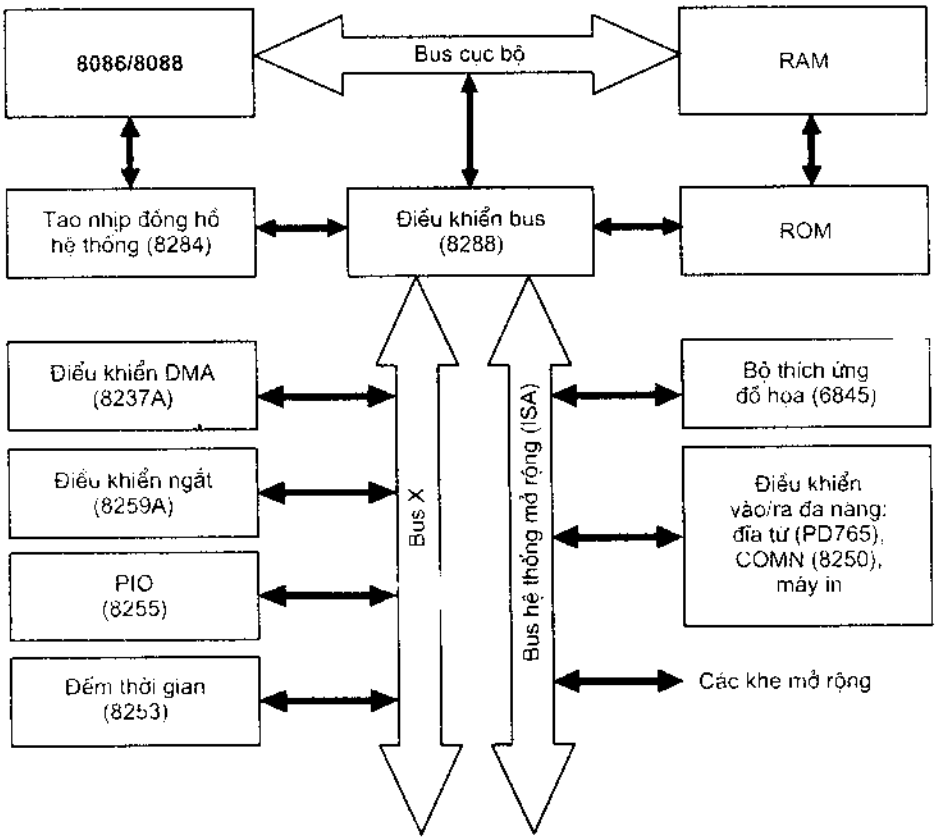
Mạch điều khiển bus 8288 có nhiều ứng dụng trong các hệ thống vi xử lý. Nó có sơ đồ khối chức năng cho trong hình 5.5. Trong đó:

AEN# (chân 6) - (cho phép địa chỉ), cho phép được ra các tín hiệu của 8288 chỉ sau khoảng từ 110 ns đến 250 ns để chờ các địa chỉ đã chốt ra ngoài bus địa chỉ.

CEN (chân 15) - (cho phép lệnh), khi CEN = 0, thì tất cả các đầu ra tín hiệu và các đầu ra điều khiển DEN, PDEN của 8288 trở về trạng thái không tích cực. Khi CEN = 1 thì các tín hiệu ra của 8288 được phép tích cực.

IOB (chân 1) - (chế độ vào/ra bus), khi IOB = 1, 8288 trong chế độ bus vào/ra với ngoại vi. Trong chế độ bus vào/ra tất cả các đường tín hiệu lệnh của 8288 điều khiển vào/ra với thiết bị ngoại vi (bảng 5.1): IORC#, IOWC#, AIOWC# và INTA# luôn được phép (nghĩa là không phụ thuộc vào AEN#). Khi vi xử lý khởi tạo một lệnh vào/ra, 8288 ngay lập tức tích cực các đường tín hiệu lệnh sử dụng MCE/PDEN# và DT/R# để điều khiển mạch thu phát vào/ra bus. Chế độ vào/ra bus tiện lợi để sử dụng trong hệ thống đa xử lý mà các thiết bị ngoại vi kết nối với một bộ xử lý. Khi IOB = 0, 8288 làm việc trong chế độ bus hệ thống thì không tín hiệu lệnh nào của 8288 được đưa ra cho đến khi AEN# tích cực thấp (= 0) sau một quãng thời gian xác

định. Chế độ này được dùng trong hệ thống chỉ có một bus hệ thống mà trên đó kết nối các thiết bị ngoại vi và bộ nhớ. Vì vậy, trong IBM PC XT IOB được đặt luôn = 0 (nối đất).



Hình 5.6: Kết nối CPU và các mạch phụ trợ chính trong IBM PC XT

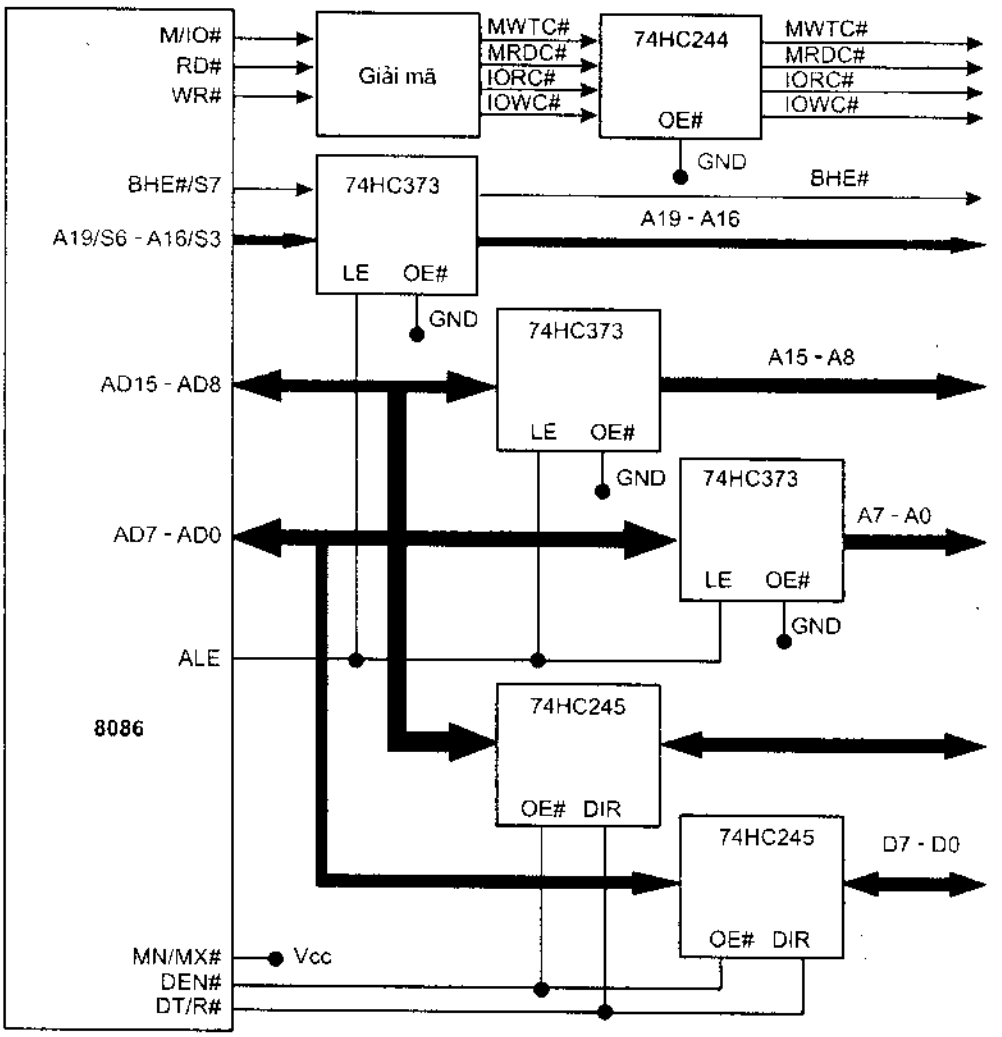
Các tín hiệu ra điều khiển cho phép chuỗi chủ/cho phép dữ liệu ngoại vi MCE/PDEN# (Master Cascade Enable/Peripheral Data Enable) thay đổi chức năng nhờ IOB. Khi IOB = 1 (chế độ vào/ra bus), thì MCE/PDEN# = 0 cho phép dữ liệu cho bus vào/ra hoặc bus hệ thống ngoại vi. Khi IOB = 0 (chế độ bus hệ thống), thì MCE/PDEN# = 1 phục vụ cho chấp nhận yêu cầu ngắt trong hệ thống chuỗi ngắt gồm các mạch PIC (có master và các slaves). Hệ thống ngắt chỉ có một PIC thì MCE không có ý nghĩa. Các chip 80x86 cũng có những mạch điều

khiển bus tương tự. Ví dụ 80286 có mạch 82288 mà ta đã xét trong chương 2, hình 2.93.

Sơ đồ khối kết nối các mạch phụ trợ chính trong IBM PC XT 8086/8088 cho trong hình 5.6.

Sơ đồ chi tiết bo mạch chủ của IBM PC/XT cho trong phần phụ lục.

b) Trong chế độ tối thiểu



Hình 5.7: Kết nối CPU8086 với các mạch phụ trợ để tạo bus hệ thống

Hệ thống vi xử lý 8086/8088 trong chế độ tối thiểu có kết nối đơn giản vì bus hệ thống được tạo ra nhờ các mạch phụ trợ đơn giản hơn và không cần có mạch điều khiển bus. Hình 5.7 là sơ đồ kết nối các CPU 8086 và các mạch phụ trợ 3 trạng thái để tạo ra bus hệ thống. Các tín hiệu của bus điều khiển hệ thống được lấy trực tiếp từ CPU, trong đó, M/IO#, RD#, và WR# đưa vào giải mã để tạo ra các tín hiệu của bus điều khiển hệ thống: MWTC#, MRDC#, IORC#, IOWC# và được chốt bằng mạch 74HC244. Cấu trúc mạch giải mã cho ở hình 5.2.

Bus hệ thống của vi xử lý 8-bit như 8085 được thiết kế đơn giản hơn. Ta có thể tham khảo lại sơ đồ cho ở hình 2.65.

5.1.3. Kết nối các mạch phụ trợ trong các hệ thống 80x86 và Pentium

Các Chipset phân ra các loại: Chipset hỗ trợ bộ xử lý, Chipset hỗ trợ cache, Chipset hỗ trợ bộ nhớ, Chipset cho định thời và điều khiển luồng, Chipset điều khiển bus vào/ra và thiết bị ngoại vi, Chipset hỗ trợ quản lý nguồn. Có một số loại Chipset thông dụng: Các Chipset thế hệ thứ tư (dùng cho 80486), Intel Chipset và Non-Intel Chipset thế hệ thứ năm (dùng cho các loại Pentium P5), Chipset thế hệ thứ sáu (dùng cho Pentium Pro, Pentium II), Chipset cho Pentium III, IV, và M.

a) Chipset thế hệ thứ tư

Có nhiều loại Chipset cho các Intel 80486 của một số nhà sản xuất như: Contaq Chipset 82C599 (82C596 và 82C597 điều khiển hệ thống, mạch 82C599 điều khiển giao tiếp bus PCI), Aser Labs FINALI-486 Chipset (M1487 và M1489), UMC Chipset (8881 và 8886), VIA GMC Chipset (82C486A - bộ điều khiển bộ nhớ/cache + cầu nối VLB - ISA, 82C482 - cầu nối VLB - ISA, 82C483 - điều khiển DRAM, VT82C505 - cầu nối PCI - VLB). Họ Chipset Intel Saturn hỗ trợ đến 486/DX4), OPTi Chipset (82C822 và 82C895 và SiS Chipset. SiS và). Bus nội bộ VESA - VESA Local Bus được đưa ra vào thời gian của 80486, do đó, phần lớn các bo mạch chủ 486 đều là thiết kế có VLB. Một loại bo mạch chủ đáng chú ý có tên VIP sử dụng các loại bus VLB, ISA và PCI.

b) Các Chipset thế hệ thứ năm

Các Chipset của Pentium khác nhiều so với 80486. Có các loại Chipset của Intel và không của Intel. Các loại Chipset của Intel được bắt đầu bằng 430LX "Mercury" hỗ trợ cho các Pentiums 60-66 MHz, hỗ trợ PCI, nhưng không hỗ trợ bộ nhớ EDO, và chỉ hỗ trợ 128 MB RAM. Tiếp theo là 430NX, "Neptune" hỗ trợ các Pentium 75-133 MHz, 512 MB RAM. Chipset 430FX, "Triton" (bộ điều khiển hệ thống, 82437FX đường dữ liệu 82438FX, bộ tăng tốc PCI ISA IDE 82371FB) bắt đầu loạt các Chipset Triton. Nó hỗ trợ bộ nhớ EDO, cache đường ống chế độ bùng phát và PCI 2.0. Chipset 430HX, "Triton II" (bộ điều khiển hệ thống 82439HX và bộ điều khiển IDE 82371SB) có một số đặc tính như: hỗ trợ bộ nhớ có kiểm tra theo chẵn lẻ và ECC, hỗ trợ kết nối 2 vi xử lý, 512 MB RAM, PCI 2.1, định thời cache là 3-1-1-1. Chipset 430VX "Triton III" (bộ điều khiển hệ thống 82437VX, bộ điều khiển IDE 82371SB, đơn vị đường dữ liệu 82438HX) chủ định cho sử dụng gia đình với hỗ trợ SDRAM giá rẻ, định thời của cache là 2-1-1-1 nhanh hơn của Triton II Chipset, hỗ trợ 256 MB RAM, Ultra-DMA và có nhiều khe SIMM hơn loại 430VX. Bảng 5.3 liệt kê một số loại Chipset thế hệ thứ năm của Intel.

Các Chipset không của Intel dùng cho Pentiums không ít. Ví dụ, Chipset Aladdin (Acer Labs, gồm M1511/12/13) hỗ trợ các vi xử lý Intel, AMD, và Cyrix. 3), Genie ALI Chipset (Acer Labs). Nó hỗ trợ để sử dụng trong các hệ thống đa môi trường đa xử lý, và 4 loại vi xử P54C, P55C, Cyrix M1 hoặc AMD K5. OPTi Chipset (82C596, 82C597, 82C599). Chipset OPTI "Viper" gồm các chip bộ điều khiển bộ đệm dữ liệu 82C556, bộ điều khiển hệ thống 82C557 và bộ điều khiển ngoại vi 82C558. Nó hỗ trợ làm chủ bus IDE và kiểu F DMA, và xử lý tín hiệu gốc NSP (Native Signal Processing), nó cho phép Pentium thực hiện các chức năng DSP. Nó cũng hỗ trợ quản lý nguồn và chạy bus PCI ở tốc độ 33 MHz. Nó cũng hỗ trợ các vi xử lý và tương tự chạy tối đa ở 100 MHz, cũng như bộ nhớ DRAM bình thường, SDRAM, EDO, Burst EDO (BEDO) với 512 MB, và cache

đường ống chế độ bùng phát dung lượng tới 2 MB. Nó hỗ trợ tốc độ vận chuyển dữ liệu của bus PCI là 132 MB/s.

Bảng 5.3: Các Chipset của bộ xử lý Pentium

	430VX	430TX	430HX	430FX	430MX
HOST					
Vi xử lý	Pentium	Pentium	Pentium	Pentium	Pentium
Điện áp	3,3 V (I/O)	3,3 V (I/O)	3,3 V (I/O)	3,3 V (I/O)	3,3 V (I/O)
Hỗ trợ xử lý kép	Không	Không	Có	Không	Không
DRAM					
Làm tươi DRAM	CAS - trước - RAS	CAS - trước - RAS	CAS - trước - RAS	Duy nhất RAS	CAS - trước - RAS
Các đường RAS	5	6	8	5	4
Hỗ trợ 64 Mbit	Không	Có	Có	Không	Không
Dung lượng RAM	128 MB	256 MB	512 MB	128 MB	128 MB
Loại bộ nhớ	SDRAM/EDO/ FPM	SDRAM/EDO /FPM	EDO/FPM	EDO/SPM	EDO/SPM
SDRAM (CL=2)	6-1-1-1	6-1-1-1	NA	NA	NA
EDO (66 MHz)	6-2-2-2	5-2-2-2	5-2-2-2	7-2-2-2	7-2-2-2
Các bộ đệm MA	Tích hợp	Tích hợp	Tích hợp	Mở rộng	Mở rộng
ECC/chẵn lẻ	Không	Không	Có	Không	Không
L2					
Loại Cache	Không đồng bộ, DRAM, Pburst	Pburst	Pburst	Không đồng bộ, Burst, Pburst	Không đồng bộ, Burst, Pburst
Khả năng cache	64 MB	64 MB	512 MB	64 MB	64 MB
Giao diện PCI					
Hỗ trợ PCI	PCI 2.1	PCI 2.1	PCI 2.1	PCI 2.0	PCI 2.0
PCI tương đồng	Có	Có	Có	Không	Không
Giám sát (ARBITRATION)					
MTT	Có	Có	Có	Không	Không
Đồ họa (GRAPHICS)					
Hỗ trợ SMBA	Có	Không	Không	Không	Không
Cầu nam (SOUTH BRIDGE)					
Loại	PIIX3	PIIX4	PIIX3	PIIX	MPIIX
Hỗ trợ USB	Có	Có	Có	Không	Không
IDE	BMIDE	Ultra DMA	BMIDE	BMIDE	IDE bình thường
RTC	Mở rộng	Tích hợp	Mở rộng	Mở rộng	Mở rộng
Khả năng quản lý (MANAGEABILITY)					
Quản lý nguồn	N/A	ACPI	N/A	N/A	SMI, APM
Quản lý vào/ra	N/A	SM Bus/GPIO	N/A	N/A	N/A

c) Các Chipset thế hệ thứ sáu

Vi kiến trúc P6 bắt đầu được sử dụng Intel Pentium Pro và sau đó là cho Pentium II, Pentium III. Có một số Chipset của Intel thông dụng cho Pentium Pro/Pentium II như sau:

450GX/KX "Orion": là loại Chipset đầu tiên cho Pentium Pro. Nó có 2 phiên bản: GX và KX. Phiên bản GX là Chipset cho các máy chủ dịch vụ, hỗ trợ bộ nhớ 4 GB tổ chức kiểu 4-đường xen kẽ và 4 vi xử lý. Nó cũng hỗ trợ bus 2 PCI riêng biệt. Phiên bản KX dùng cho các máy tính trạm, nhưng cũng đủ mạnh để có thể dùng cho máy tính dịch vụ. Nó hỗ trợ bộ nhớ 1 GB tổ chức kiểu 2-đường xen kẽ và 2 vi xử lý. Cả 2 phiên bản làm việc ở bus 66 MHz và hỗ trợ kết nối các mô-đun nhớ FPM và EDO. Các Chipset giá cao nên không được sử dụng nhiều trong PC.

440FX "Natoma": dùng cho Pentium Pro, và có thể cho Pentium II. Nó có một số đặc tính của Chipset GX/KX, nhưng hiệu suất cao hơn và hỗ trợ Ultra-DMA, SDRAM, quản lý nguồn. Nó hỗ trợ 2 vi xử lý với SMP, BEDO RAM. Chipset 440LX: chuyên dùng cho Pentium II, hỗ trợ Ultra-DMA và SDRAM, hỗ trợ USB và AGP. Nó kết hợp những ưu điểm của các Chipset 430TX và 440FX. Nó hỗ trợ SRRAM 512 MB và 1 GB EDO RAM. Nó sử dụng EEPROM để truyền thông giữa Chipset và RAM. Chipset 440BX: tương tự như 440LX, nhưng thực sự hỗ trợ bus tốc độ 100 MHz. Chipset 440GX: tương tự như BX, nhưng dùng cho cả Pentium II và Pentium Xeon. Nó hỗ trợ SDRAM 2 GB.

Các Chipset không phải của Intel cũng có một số lại phổ biến, ví dụ như: SiS 620 cho Pentium II, 1.5 GB SDRAM, bus 100 MHz (tốc độ đôi như MVP3), UltraATA-33/66, và UMA, tích hợp bộ tăng tốc 2D/3D; ALi Alladin Pro II cho Pentium và Celeron, 100 MHZ FSB, AGP 2X, 1GB of SDRAM (or 2 GB of EDO).

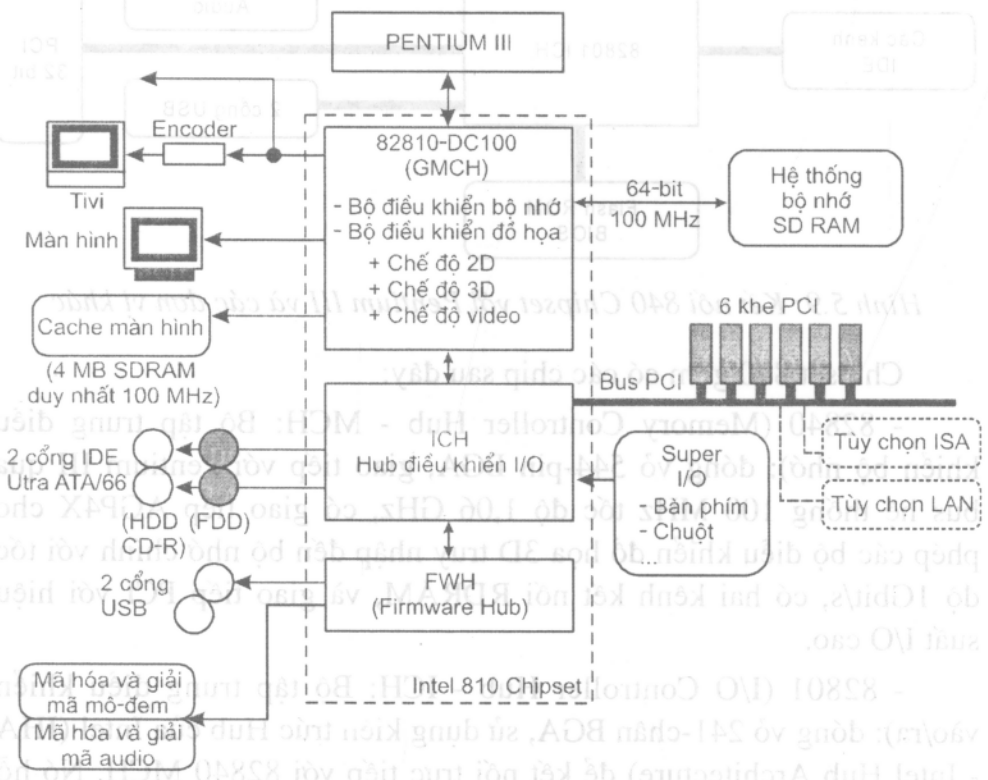
d) Các Chipset cho Pentium III

Các Chipset của Intel dùng cho các loại Pentium III, Celeron, Xeon phổ biến là họ Intel 800 (Bảng 5.4). Các Chipset 810, 820 có thể được dùng cho cả Pentium II.

Bảng 5.4: Các Chipset của Intel dùng cho Pentium III, Celeron, Xeon

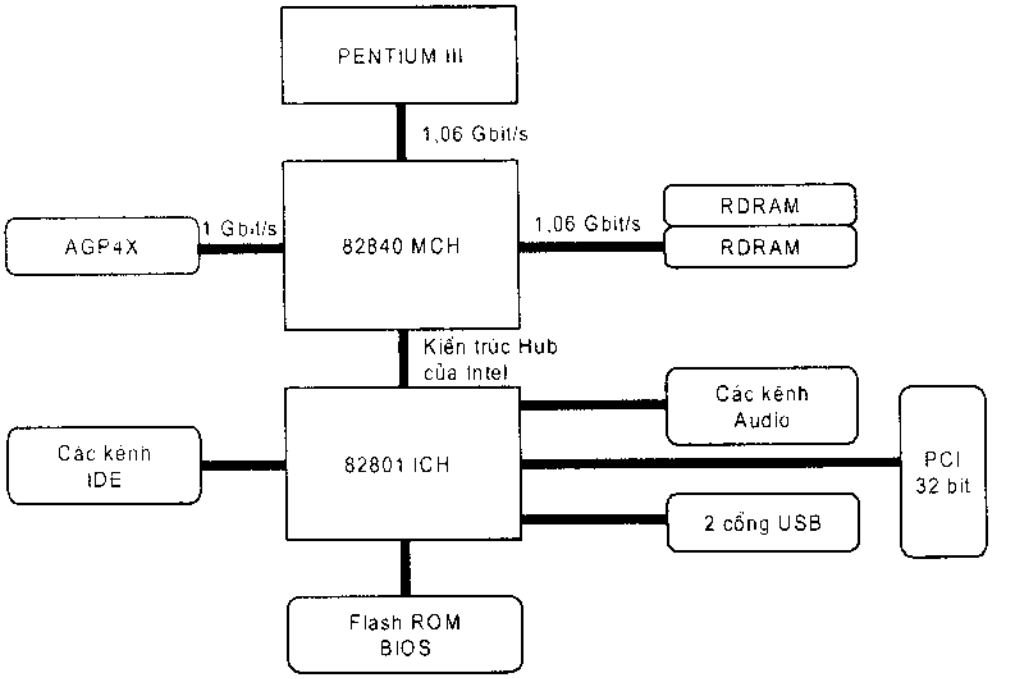
Chipset	840	820E	820	815P	815EG	815G	810E2	810E	810
Vi xử lý	Pentium® III Pentium® III Xeon™	Pentium® III Pentium® II	Pentium® III Pentium® II	Celeron®, Pentium® III (.13μ, .18 μ)	Celeron® , Pentium® III (.13 μ, .18 μ)	Celeron®, Pentium® III (.13 μ, .18 μ)	Celeron®, Pentium® III (.13 μ, .18 μ)	Celeron®, Pentium® III (.13 μ, .18 μ)	Celeron®, Pentium® III, Pentium® II
Số vi xử lý	1-2	1-2	1-2	1	1	1	1	1	1
Bus hệ thống	133/100 MHz	133/100 MHz	133/100 MHz	133/100/66 MHz	133/100/66 MHz	133/100/66 MHz	133/100/66 MHz	133/100/66 MHz	100/66 MHz
Loại MCH	82840 MCH	82820 MCH	82820 MCH	82815EP MCH	82815G GMCH	82810E GMCH	82810E GMCH	82815G GMCH	82810 GMCH
Đóng vỏ MCH	544 BGA	324 BGA	324 BGA	544 BGA	544 BGA	544 BGA	421 BGA	421 BGA	421 BGA
Loại Mô-đun nhớ	2 RIMMs, up to 4 RIMMs with MRH-R	2 RIMMs	2 RIMMs	3 DIMMs	3 DIMMs	3 DIMMs	2 DIMMs	2 DIMMs	2 DIMMs
Loại bộ nhớ	PC800/600 RDRAM	PC800/700/600 RDRAM	PC800/700/600 RDRAM	PC133/100 SDRAM	PC133/100/66 SDRAM	PC133/100/66 SDRAM	PC100/66 SDRAM	PC100/66 SDRAM	PC100/66 SDRAM
Dung lượng nhớ tối đa	4 GB (w/ 2 repeaters)	1 GB	1 GB	512 MB	512 MB	512 MB	512 MB	512 MB	512 MB
Hỗ trợ	256/128/64 Mbit	256/128/64 Mbit	256/128/64 Mbit	256/128/64/16 Mbit	256/128/64/16 Mbit	256/128/64/16 Mbit	128/64/16 Mbit	128/64/16 Mbit	128/64/16 Mbit
ECC/Parity	Yes	Yes	Yes	N/A	No	No	N/A	N/A	N/A
Giao tiếp đồ họa	AGP 4X/2X/1X	AGP 4X/2X/1X	AGP 4X/2X/1X	AGP 4X/2X	Tích hợp 3D với AGP trực tiếp, 133/100 MHz 4MB display cache	Tích hợp 3D với AGP trực tiếp, 133/100 MHz 4MB display cache	Tích hợp 3D với AGP trực tiếp	Tích hợp 3D với AGP trực tiếp	Tích hợp 3D với AGP trực tiếp
Loại ICH	ICH	ICH2	ICH	ICH	ICH2	ICH	ICH2	ICH	ICH
Đóng vỏ ICH	241 BGA	360 EBGA	241 BGA	241 BGA	360 EBGA	241 BGA	360 EBGA	241 BGA	241 BGA
PCI Masters	6	6	6	6	6	6	6	6	6
Hỗ trợ PCI	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2	PCI 2.2
IDE	ATA/66	ATA/100	ATA/66	ATA/66	ATA/100	ATA/66	ATA/100 IAA[1]	ATA/66	ATA/66
USB	1 controller: 2 ports	2 controllers: 4 ports	1 controller: 2 ports	1 controller: 2 ports	2 controllers: 4 ports	1 controller: 2 ports	2 controllers: 4 ports	1 controller: 2 ports	1 controller: 2 ports
LAN MAC/PNA	No	Yes	No	No	Yes	No	Yes	No	No
AC'97 Digital Circuits	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem	Audio/Modem
Quản lý I/O	SMBus / GPIO	SMBus / GPIO	SMBus / GPIO	SMBus / GPIO	SMBus/ GPIO	SMBus/GPIO	SMBus/GPIO	SMBus/GPIO	SMBus/GPIO

Sơ đồ đấu nối Chipset Intel 810 với Pentium III cho ở hình 5.8. Chipset 810 có thể dùng cho các vi xử lý Celeron, Pentium II. Nó gồm có 2 vi mạch: 82810 GMCH đóng vỏ 421-chân BGA và ICH đóng vỏ 241-chân BGA. Chip 82810 GMCH (Graphics and Memory Controller Hub) gồm có: bộ điều khiển bộ nhớ (Memory Controller), bộ điều khiển đồ họa (Graphics Controller) với các chế độ 2D, 3D và video. Nó hỗ trợ kết nối RAM dung lượng tối đa 512MB. Các mô-đun nhớ dùng loại SDRAM kết nối qua bus PC66/100 MHz. Chip ICH (I/O Controller Hub) hỗ trợ kết nối PCI 2.2 (6 khe mở rộng Bus PCI), kết nối IDE (2 cổng cho các ổ đĩa cứng), Ultra ATA/66, USB (2 cổng và 1 bộ điều khiển USB), con chuột, bàn phím, v.v... Chipset 810 chỉ hỗ trợ hệ thống 1 vi xử lý.



Hình 5.8: Kết nối 810 Chipset với Pentium Pro và các thiết bị ngoại vi, hệ thống nhớ

Chipset 840 có tốc độ dữ liệu với Pentium III cao: 1,0 Gbit/s. Sơ đồ khối chức năng kết nối CPU Pentium III với Chipset 840 trên bảng mạch mẹ được mô tả trong hình 5.9.



Hình 5.9: Kết nối 840 Chipset với Pentium III và các đơn vị khác

Chipset 840 gồm có các chip sau đây:

- 82840 (Memory Controller Hub - MCH: Bộ tập trung điều khiển bộ nhớ): đóng vỏ 544-pin BGA, giao tiếp với Pentium III qua bus hệ thống 100 MHz tốc độ 1,06 GHz, có giao tiếp AGP4X cho phép các bộ điều khiển đồ họa 3D truy nhập đến bộ nhớ chính với tốc độ 1Gbit/s, có hai kênh kết nối RDRAM, và giao tiếp PCI với hiệu suất I/O cao.

- 82801 (I/O Controller Hub - ICH: Bộ tập trung điều khiển vào/ra): đóng vỏ 241-chân BGA, sử dụng kiến trúc Hub của Intel (IHA - Intel Hub Architecture) để kết nối trực tiếp với 82840 MCH. Nó hỗ trợ kết nối PCI 32-bit, có bộ điều khiển IDE và bộ điều khiển USB với hai cổng để kết nối với các thiết bị ngoại vi qua giao tiếp USB, hỗ trợ

vi xử lý sử dụng bus hệ thống tốc độ 133 MHz. Để nâng cao hiệu suất, 2 chip này có thể được kết nối với 2 chip cơ bản sau đây:

+ Bộ tập trung điều khiển PCI 82806 64-bit: hỗ trợ các khe PCI 64-bit ở tốc độ 33 hoặc 66MHz. P64H kết nối trực tiếp với 82840 MCH sử dụng kiến trúc IHA.

+ Bộ tập trung lặp bộ nhớ dựa trên RDRAM 82803 dùng để mở rộng dung lượng của RDRAM. Nó chuyển đổi từng kênh nhớ thành 2 kênh nhớ cho kết nối.

e) Các Chipset cho Pentium IV

Các chip Pentium IV sử dụng công nghệ siêu luồng (Hyper-Threading Technology) nên các Chipset của Intel liệt kê trong bảng 5.5 đều hỗ trợ công nghệ này. Các Chipset 865G, 865GV và 845G hỗ trợ kết nối 64 MB bộ nhớ video động với mở rộng đồ họa (Intel Extreme Graphics 2 Driver) phiên bản 11.0 (cho 845G) và 12.0 hoặc muộn hơn (865G và 865V) khi dung lượng RAM >128 MB, và 32 MB bộ nhớ video động khi dung lượng RAM ≤ 128 MB. Giao tiếp DVO (Intel Digital Video Output Interface: giao tiếp đầu ra video số của Intel) mà các Chipset 865G, 865GV hỗ trợ cho phép kết nối với TV hoặc màn hình phẳng (flat-panel display). Chức năng tăng tốc ứng dụng IAA (Intel Application Accelerator) của các giao tiếp IDE trong các Chipset 845, 848, và 850 thực hiện tăng tốc khởi động (boot time) và vào/ra với thiết bị đĩa (disk I/O). Mỗi Chipset chứa 2 thành phần chính: MCH hay GMCH (Graphics and Memory Controller Hub) để làm cầu nối CPU và ICH (I/O Controller Hub) cho hệ thống vào/ra. Bảng 5.5 và 5.6 liệt kê một số Chipset của Intel và của một số nhà sản xuất khác dùng cho Pentium IV. Bảng 5.7 so sánh một số đặc điểm của loại Chipset thông dụng Intel 845, 850 với VIA P4X266.

Bảng 5.5: Các Chipset cho Pentium IV

Chipset	Chipset 875P	Chipset 865G	Chipset 865PE	Chipset 865P	Chipset 865GV	Chipset 850E	Chipset 848P	Chipset 845G
Bộ xử lý	Pentium IV	Pentium IV	Pentium IV	Pentium IV	Pentium IV	Pentium IV, Celeron	Pentium IV	Pentium IV, Celeron

Công nghệ siêu luồng (HHT)	Được tối ưu cho HTT	được tối ưu cho HTT	Được tối ưu cho HTT	Hỗ trợ HTT	Được tối ưu cho HTT	Hỗ trợ HTT	Hỗ trợ HTT	Hỗ trợ HTT
Bus hệ thống	800/533 MHz	800/533/400 MHz	800/533/400 MHz	533/400 MHz	800/533/400 MHz	533/400 MHz	800/533/400 MHz	533/400 MHz
Đóng vỉ xử lý	mPGA 478	mPGA 478	mPGA 478	mPGA 478	mPGA 478	mPGA 478	mPGA 478	mPGA 478
Số lượng vi xử lý	1	1	1	1	1	1	1	1
Chip MCH (cầu bắc)	82875P MCH	82865G GMCH	82865PE MCH	82865P MCH	82865G GMCH	82850E MCH	82848P MCH	82845G GMCH
Đóng vỏ MCH	1005 FC-BGA	932 FC-BGA	932 FC-BGA	932 FC-BGA	932 FC-BGA	615 OLGA	932 FC-BGA	760 FC-BGA
Kết nối các loại mô-đun nhớ	2 DIMMs/2 kênh	2 DIMMs/2 kênh	2 DIMMs/2 kênh	2 DIMMs/2 kênh	2 DIMMs/2 kênh	4 RIMMs	2 DIMMs	2 DDR , 2 SDR DIMMs
Loại bộ nhớ	Hai kênh DDR 400/333/266 SDRAM	Hai kênh DDR 400/333/266 SDRAM	Hai kênh DDR 400/333/ 266 SDRAM	Hai kênh DDR 333/266 SDRAM	Hai kênh DDR 400/333/ 266 SDRAM	PC1066-32 PC800-40 RDRAM	DDR 400/333/ 266 SDRAM	DDR 266/200 SDRAM, PC133 SDRAM
Cấu hình bộ nhớ/FSB	800/400 800/333 533/333 533/266	800/400 800/333 533/333 533/266 400/333 400/266	800/400 800/333 533/333 533/266 400/333 400/266	533/333 533/266 400/333 400/266	800/400 800/333 533/333 533/266 400/333 400/266	533/PC1066-32 533/PC800-40 400/PC800-40	800/400 800/333 533/333 533/266 400/333 400/266	533/266 533/200 400/266 400/200 400/133
Dung lượng bộ nhớ tối đa	4GB	4GB	4GB	4GB	4GB	2GB (1,5GB-PC1066)	2GB	2GB
Hỗ trợ Mbit	512/256/128 Mbit	512/256/128 Mbit	512/256/128 Mbit	512/256/128 Mbit	512/256/128 Mbit	288/256 Mbit, 144/128 Mbit	512/256/128 Mbit	512/256/128 Mbit
Sửa lỗi	ECC/không -ECC	Không - ECC	Không - ECC	Không - ECC	Không - ECC	ECC/ không -ECC	Không - ECC	Không -ECC
Giao tiếp đồ họa	AGP8X (1,5V)	AGP8X (1,5V)	AGP8X (1,5V)	AGP8X (1,5V)	N/A	AGP4X (1,5V)	AGP8X (1,5V)	AGP4X (1,5V)
Loại đồ họa	N/A	Intel Extreme Graphics 2	N/A	N/A	Intel Extreme Graphics 2	N/A	N/A	Intel Extreme Graphics
Tốc độ lõi	N/A	266 MHz	N/A	N/A	266 MHz	N/A	N/A	200 MHz
Max Dynamic Video Memory	N/A	64 MB nếu >128 MB RAM 32 MB nếu <=128 MB RAM	N/A	N/A	64 MB nếu >128 MB RAM 32 MB nếu <=128 MB RAM	N/A	N/A	64 MB nếu >=256 MB RAM, 32 MB nếu từ 128 MB 255 MB RAM
Video / Display	N/A	350 MHz DAC 2x12bit DVO[3]	N/A	N/A	350 MHz DAC 2x12bit DVO	N/A	N/A	350 MHz DAC 2x12bit DVO
Chip ICH (cầu nam)	ICH5 / ICH5R	ICH5 / ICH5R	ICH5 / ICH5R	ICH5 / ICH5R	ICH5 / ICH5R	ICH2	ICH5 / ICH5R	ICH4

Đóng vỏ ICH	460 mGBA	460 mGBA	460 mGBA	460 mGBA	460 mGBA	360 EBGA	460 mGBA	421 mBGA
Hỗ trợ PCI	PCI 2.3	PCI 2.3	PCI 2.3	PCI 2.3	PCI 2.3	PCI 2.2	PCI 2.3	PCI 2.2
PCI Masters	6	6	6	6	6	6	6	6
IDE	Ultra ATA/100	Ultra ATA/100	Ultra ATA/100	Ultra ATA/100	Ultra ATA/100	ATA/100 IAA	ATA/100 IAA	ATA/100 IAA
Serial ATA/Ports	SATA 150/2	SATA 150/2	SATA 150/2	SATA 150/2	SATA 150/2	N/A	SATA 150/2	N/A
Công nghệ Intel RAID	RAID w/ICH5R	RAID w/ICH5R	RAID w/ICH5R	RAID w/ICH5R	RAID w/ICH5R	N/A	RAID w/ICH5R	N/A
USB Ports/Controllers	8 ports, USB 2.0	8 ports, USB 2.0	8 ports, USB 2.0	8 ports, USB 2.0	8 ports, USB 2.0	4 ports, USB 1.1	8 ports, USB 2.0	6 ports, USB 2.0
LAN MAC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GbE Dedicated Network Bus	Yes	Yes	Yes	Yes	Yes	No	Yes	No
AC'97	Enhanced 20-bit Audio	Enhanced 20-bit Audio	Enhanced 20-bit Audio	Enhanced 20-bit Audio	Enhanced 20-bit Audio	Yes	Enhanced 20-bit Audio	Enhanced 20-bit Audio
Quản lý I/O	SMBus 2.0 / GPIO	SMBus 2.0 / GPIO	SMBus 2.0 / GPIO	SMBus 2.0 / GPIO	SMBus 2.0 / GPIO	SMBus / GPIO	SMBus 2.0 / GPIO	SMBus 2.0/ GPIO

Bảng 5.6: Chipset của một số nhà sản xuất

Chipset	VIA P4X266A	VIA P4X266	SiS 645
Tháng-năm đưa ra	12/2001	8/2002	11/2001
CPU Platform	Socket 478	Socket 423/478	Socket 478
Hỗ trợ vi xử lý	Intel Pentium IV	Intel Pentium IV	Intel Pentium IV
Hỗ trợ đa xử lý	Không	Không	Không
Cầu bắc của Chipset (MCH)	VIA P4X266A	VIA VT8753	SiS 645
Cầu nam của chipset (ICH)	VIA VT8233CE	VIA VT8233	SiS 961
Đồng hồ bus bên ngoài đằng trước (FSB)	66/100/133 MHz	100 MHz	100/133 MHz
Nhịp đồng hồ cho bộ nhớ	100/133 MHz DDR-SDRAM	100/133 MHz SDR/DDR	100/133 /166 MHz DDR-SDRAM
Đồng hồ bộ nhớ không đồng bộ	Có	Có	Có
FSB Overclocking	Lên đến 180 MHz	Lên đến 200 MHz	Lên đến 180 MHz
Số khe cho DIMM lớn nhất	3	4	3
Max. Memory Expansion	3072 MB	4096 MB	3072 MB
Hỗ trợ SDRAM	Không	Không	Không
Hỗ trợ DDR SDRAM	Có	Có	Có
Hỗ trợ VC SDRAM	Không	Có	Không

Chipset	VIA P4X266A	VIA P4X266	SiS 645
Hỗ trợ RIMM- (Rambus)	Không	Không	Không
Hỗ trợ Dual-RIMM- (Rambus)	Không	Không	Không
Ultra-DMA/33/66/100	Có/có/có	Có/có/có	Có/có/có
Ultra-DMA/133	Có	Không	Không
Số cổng USB	6	6	6
USB 2.0	Có	Không	Có
Số khe PCI (Slots) max	6	5	6
Đồ họa tích hợp	Không	Không	Không
AGP 1x / 2x / 4x	Có/có/có	Có/có/có	Có/có/có
Đặc tính ACPI	Có	Có	Có

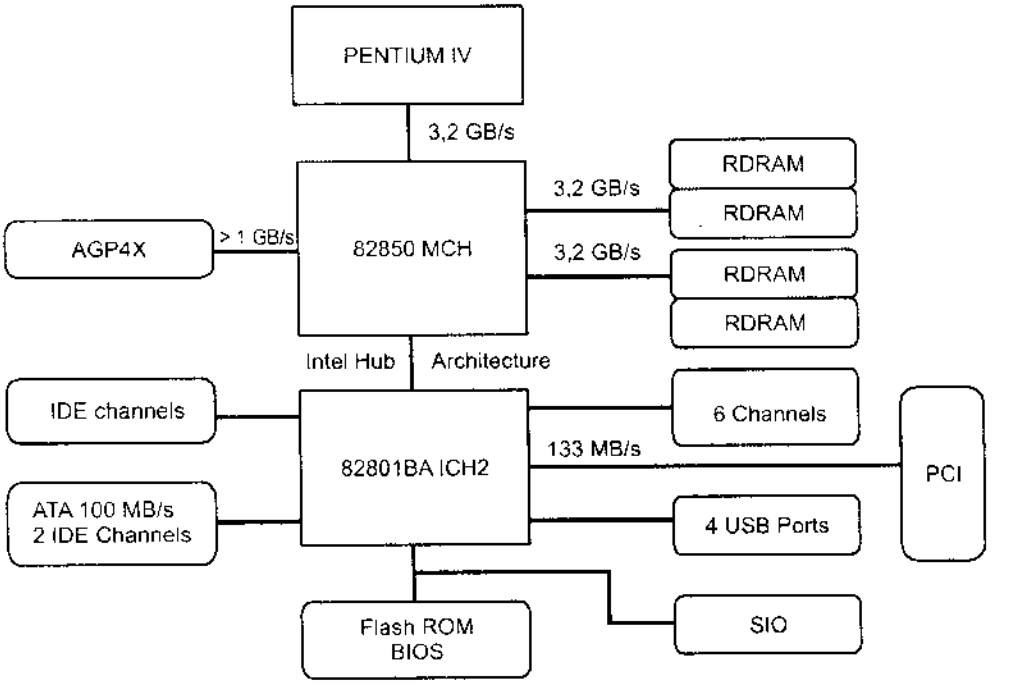
Bảng 5.7: Đặc điểm của Chipset Intel 850, Intel 845 và VIA P4×200

	Intel 850	Intel 845	VIA P4X266
Cầu Bắc (MCH)	i82850	i82845	VIA VT8753
Bus xử lý	400MHz Quad Pumped Bus (3.2GB/sec)	400MHz Quad Pumped Bus (3.2GB/sec)	400MHz Quad Pumped Bus (3.2GB/sec)
Giao tiếp vi xử lý	Socket423, Socket478	Socket478	Socket423, Socket478
Loại bộ nhớ	Dual-channel PC800 RDRAM	PC133/PC100 SDRAM*	PC2100/PC1600 DDR SDRAM PC133/PC100 SDRAM
Băng thông của memory bus (max)	3.2GB/sec	1.06GB/sec	2.1GB/sec
Dung lượng bộ nhớ (max)	2 GB 4 RIMM slots	3 GB 3 DIMM slots	4 GB 4 DDR DIMM slots
Hỗ trợ sửa lỗi ECC	+	+	+
Giao tiếp AGP 4x	+	+	+
Bus Inter-Bridge	Hub Link (266MB/sec)	Hub Link (266MB/sec)	V-Link (266MB/sec)
Cầu Nam (ICH)	i82801BA	i82801BA	VIA VT8233(A/C)
Số lượng max PCI Master	5	5	5
Hỗ trợ ATA/100	+	+	+ (optional ATA/133 support)
AC'97	+	+	+
Hỗ trợ CNR/ACR	CNR	CNR	ACR
10/100Mbit LAN	+	+	+
Số cổng USB	4	4	6
Giá bán (USD)	47	42	34

	Intel 850	Intel 845	VIA P4X266
Với CPU	Intel Pentium 4 1.7GHz		
Bảng chính (Mainboard)	ASUS P4T	ASUS P4B	VIA Demo Board
Bộ nhớ	256MB PC800 RDRAM	256MB PC133 CL2 SDRAM	256MB PC2100 CL2 DDR SDRAM
Bảng mạch đồ họa	Gigabyte GV-GF3000DF (NVIDIA GeForce3)		
Đĩa cứng (HDD)	IBM DTLA 307015		

Các Chipset không phải Intel như VIA P4X266, SiS 645, v.v... (bảng 5.6) về cơ bản tương tự như các Chipset của Intel, nhưng cũng có một số khác biệt. Bảng 5.7 lấy ví dụ so sánh các Chipset Intel 845 và 850 so với Chipset VIA P4X266. Chúng giống nhau ở một số tính năng như: bus Inter-bridge 266 MB/s, giao tiếp đồ họa AGP4x, giao tiếp vi xử lý là Socket 423 và Socket 478, bus xử lý (400 MHz Quad pumped bus 3,2GB/s), số lượng max PCI master (5), hỗ trợ sửa lỗi bộ nhớ theo ECC, hỗ trợ ATA/100. Chúng có sự khác biệt về hỗ trợ kết nối các loại bộ nhớ, băng thông bus nhớ, dung lượng bộ nhớ tối đa. Với các thông số khác nhau này thì Chipset VIA P4X266 có ưu điểm hơn.

Intel 850: Cầu Bắc của Chipset 850 là MCH (Memory Controller Hub) 82850 đóng vỏ OLGA-615 (Organic Land Grrid Array), hỗ trợ nối kênh đôi 2 GB PC800 RDRAM có hoặc sửa lỗi ECC, có giao tiếp đồ họa AGP4x tốc độ > 1 GB/s và bus hệ thống 400 MHz tốc độ đẩy gấp 4 nối (400 MHz Quad Pumped Pentium IV CPU bus). Cầu Nam (south bridge) là vi mạch ICH2 82801BA đóng vỏ EBGA-360 (Enhanced Ball Grid Array), có giao tiếp giao tiếp mạng LAN (LCI) và truyền thông CNR (Communication and Networking Riser) cho phép kết nối mạng một cách linh hoạt với đường dây điện thoại bổ sung bảng điều khiển audio, bảng modem, Ethernet 10/100 Mbit/s, với khả năng quản lý LAN, các điều khiển Dual Ultra ATA/100 hỗ trợ giao tiếp IDE với các thiết bị nhớ ngoài, điều khiển AC'97 kết nối 6 kênh audio chất lượng cao, 2 điều khiển USB hỗ trợ kết nối 24 Mbit/s các ngoại vi với 4 cổng USB 1.1, các kết nối PCI tốc độ 133 MB/s. Sơ đồ kết nối Chipset 850 cho ở hình 5.10.



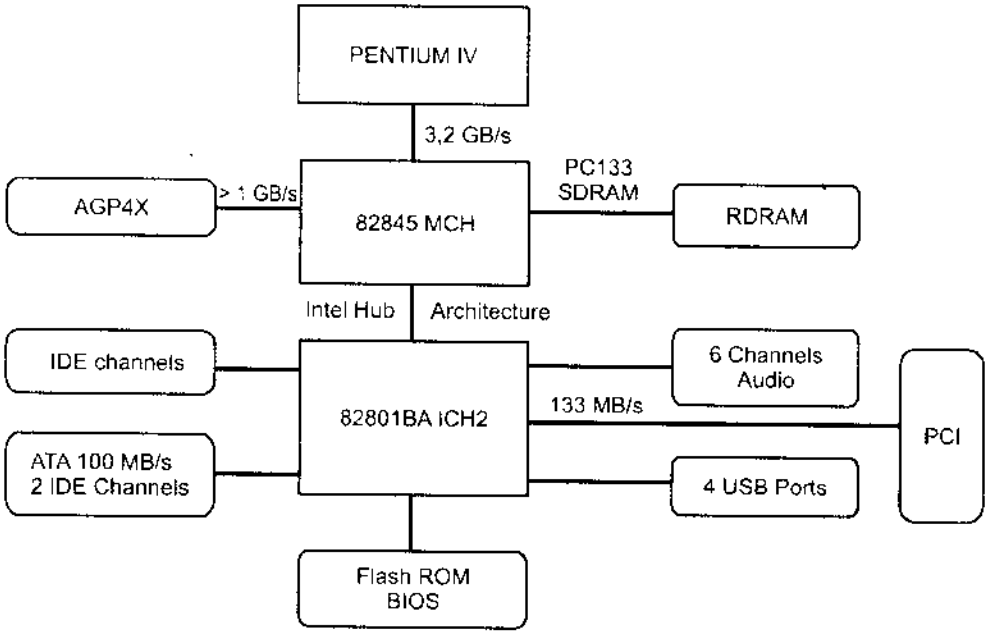
Hình 5.10: Kết nối Chipset 850 với Pentium IV và các đơn vị khác

Intel 845: Hỗ trợ kết nối PC133 SDRAM dung lượng lớn và rẻ tiền với tốc độ băng thông là 1,06 GB/s so với 3,2 GB/s của Chipset 850. Cầu Bắc là chip 82845 MCH đóng vỏ FCPGA-593, hỗ trợ Pentium IV với 400 MHz Quad Pumped Pentium IV CPU bus (dựa trên lõi Willamette và Northwood), giao tiếp đồ họa AGP 4x và tới 3 GB PC133 SDRAM có hoặc không có ECC. Chipset 845 hỗ trợ kết nối 2 GB PC1600/PC2100 DDR SDRAM. Cầu Bắc là vi mạch 282801BA ICH2 (đóng vỏ EBGA-360) có chức năng tương tự như của chipset 850. Hình 5.11 là sơ đồ kết nối Chipset 845 với Pentium IV và các đơn vị khác.

Intel 845G là phiên bản khác của Chipset 845. Nó gồm các khối chức năng:

• 82845G (GMCH): tạo bus hệ thống (kết nối với Pentium IV) tốc độ 4,2 hoặc 3,2 GB/s; tạo bus kết nối hệ thống nhớ dùng SDRAM

PC133, DDR200/266; kết nối đồ họa AGP4X 1 GB/s; đường ra video số (Digital Video Out); với kiến trúc Intel hub kết nối với ICH4



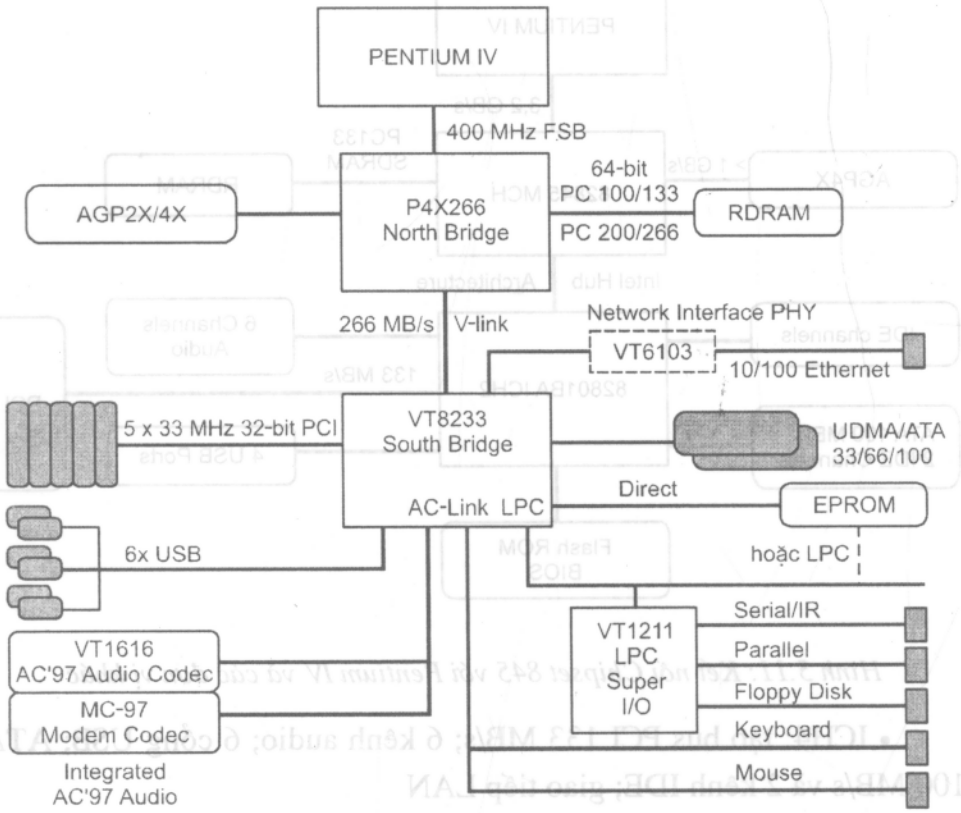
Hình 5.11: Kết nối Chipset 845 với Pentium IV và các đơn vị khác

• ICH4: tạo bus PCI 133 MB/s; 6 kênh audio; 6 cổng USB; ATA 100 MB/s và 2 kênh IDE; giao tiếp LAN

• Flash ROM BIOS.

VIA P4×266: Chipset VIA P4X266 cho phép sử dụng các Pentium IV với PC2100 DDR SDRAM với bus băng thông 2,1 GB/s (của Chipset 850 là 3,2 GB/s và của Chipset 845 là 1,06 GB/s). Nó cũng hỗ trợ PC100/133/200/266 SDRAM và cho phép các bus nhớ và CPU làm việc đồng bộ để tương thích với PC1600 DDR SDRAM và PC100 SDRAM. Nó cho phép kết nối tối đa 4 GB DDR SDRAM (của Chipset 850 là 2 GB, và của Chipset 845 là 3 GB). Chipset 850 kết nối với bộ nhớ qua 2 kênh trong khi Chipset 845 và Chipset VIA P4Z266 kết qua một kênh. Nếu kết nối với Pentium IV 1,7 GHz và dùng các

bo mạch chủ cho trong bo 5.7 thì Chipset VIA P4X266 cho cấu hình kết nối tương tự như các Chipset 845 và 850. Sơ đồ đấu nối Chipset VIA P4X266 với Pentium IV và các giao tiếp cho trong hình 5.12.



Hình 5.12: Kết nối chipset VIA P4X266

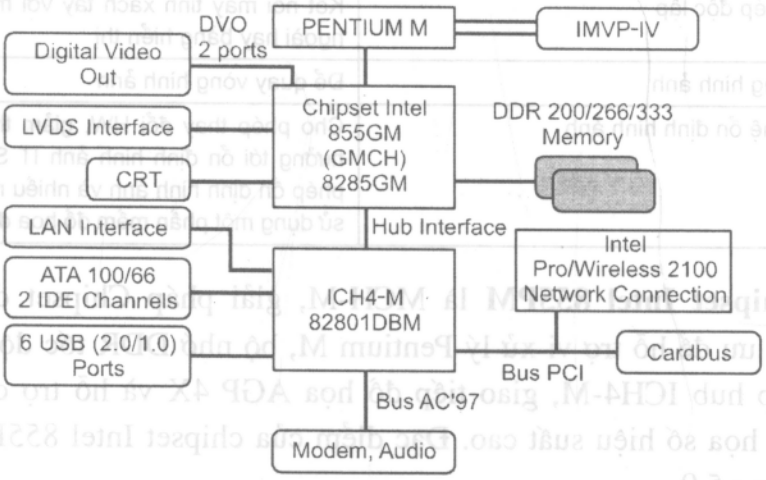
Các Chipset trên đây đều có chế độ tiết kiệm nguồn tiêu thụ khi vi xử lý ở trạng thái nghỉ.

f) Các Chipset cho Pentium M

Họ Chipset Intel 855 là một thành phần của công nghệ Centrino Mobile. Nó gồm có: Chipset Intel 855GM Chipset Intel 855PM, và chipset Intel 855GME.

Chipset Intel 855GM: Là bộ tập trung điều khiển đồ họa và bộ nhớ GMCH-M (Graphics Memory Controller Hub), giải pháp Chipset di động tích hợp cao được tối ưu để hỗ trợ vi xử lý Pentium M, bộ nhớ

DDR tốc độ cao và giao tiếp hub với ICH4-M. Ngoài ra Chipset Intel 855GM còn tích hợp các khả năng đồ họa. Sơ đồ đấu nối chipset 855GM cho trong hình 5.13.



Hình 5.13: Kết nối Chipset Intel 855GM

Chipset 855GM có 2 chip: 82855GM (MCH) đóng vỏ 732 pin Micro-FCBGA và 82801DBM (ICH4-M) đóng vỏ 421-chân Micro-BGA. Đặc điểm của chipset Intel 855GM cho trong bảng 5.8.

Bảng 5.8: Đặc điểm của chipset Intel 855GM

Đặc điểm	Lợi ích
Bus hệ thống 400 MHz nguồn thấp	Hỗ trợ cấu hình 1 vi xử lý với bus hệ thống 400 MHz
Hỗ trợ tối đa bộ nhớ 2GB DDR 333/266/200	Hiệu suất cao và tính linh hoạt
Công nghệ đồ họa Intel Extreme Graphics 2	Độ tương phản cao, hình ảnh nét đồ họa 3D và cho phép bộ nhớ cân bằng được dùng giữa đồ họa và hệ thống để tối ưu hiệu suất
Có USB 2.0 tốc độ cao	Hỗ trợ USB 2.0 tốc độ dữ liệu 40X và các thiết bị USB 1.0
Bộ đệm vào/ra linh hoạt để tách bộ nhớ và bus hệ thống xử lý	Giảm tiêu thụ nguồn của Chipset nhờ kích hoạt thông minh và giảm nguồn của bus hệ thống vi xử lý và bộ nhớ.
Chốt cổng nhịp đồng hồ bên trong tối ưu cho các động cơ 3D và hiển thị	Giảm tiêu thụ nguồn của Chipset

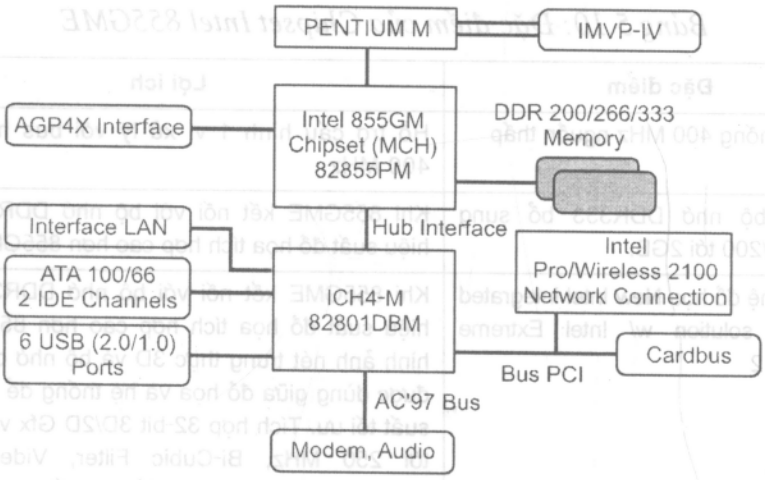
Đặc điểm	Lợi ích
Giao tiếp nguồn tín hiệu vi phân tích hợp nguồn thấp LVDS (Low Voltage Differential Signal)	Tích hợp cao để thiết kế các máy xách tay nhỏ
Hiển thị kép độc lập	Kết nối máy tính xách tay với màn hình ngoài hay bảng hiển thị
Quay vòng hình ảnh	Để quay vòng hình ảnh
Công nghệ ổn định hình ảnh	Cho phép thay đổi HW, giảm thiểu ảnh hưởng tới ổn định hình ảnh IT SW. Cho phép ổn định hình ảnh và nhiều người chỉ sử dụng một phần mềm đồ họa duy nhất

Chipset Intel 855PM là MCH-M, giải pháp Chipset di động được tối ưu để hỗ trợ vi xử lý Pentium M, bộ nhớ DDR tốc độ cao và giao tiếp hub ICH4-M, giao tiếp đồ họa AGP 4X và hỗ trợ các giải pháp đồ họa số hiệu suất cao. Đặc điểm của chipset Intel 855PM cho trong bảng 5.9.

Bảng 5.9: Đặc điểm của Chipset Intel 855 PM

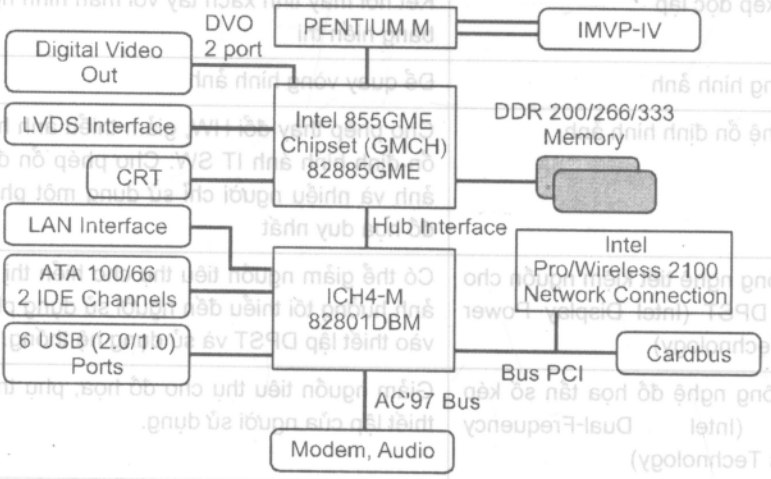
Đặc điểm	Lợi ích
Bus hệ thống 400MHz nguồn thấp	Hỗ trợ cấu hình 1 vi xử lý với bus hệ thống 400 MHz
Hỗ trợ tối đa bộ nhớ 2GB DDR 333/266/200	Hiệu suất cao và tính linh hoạt
Có USB 2.0 tốc độ cao	Hỗ trợ USB 2.0 tốc độ dữ liệu 40X và các thiết bị USB 1.0
Giao tiếp đồ họa AGP4X	Giao tiếp băng thông cao đảm bảo hỗ trợ linh hoạt các giải pháp đồ họa di động hiệu suất cao.
Công nghệ ổn định hình ảnh	Cho phép thay đổi HW, giảm thiểu ảnh hưởng tới ổn định hình ảnh IT SW
Bộ đệm vào/ra linh hoạt để tách bộ nhớ và bus hệ thống xử lý	Giảm tiêu thụ nguồn của chipset nhờ kích hoạt thông minh và giảm nguồn của bus hệ thống vi xử lý và bộ nhớ

Sơ đồ kết nối Chipset 855PM cho trong hình 5.14. Chipset 855GM có 2 chip: 82855PM (MCH) đóng vỏ 593-chân Micro-FCBGA và 82801DBM (ICH4-M) đóng vỏ 421-chân Micro-BGA.



Hình 5.14: Kết nối Chipset Intel 855PM

Chipset Intel 855GME: Là bộ tập trung điều khiển đồ họa và bộ nhớ GMCH-M, hỗ trợ bộ nhớ DDR tốc độ 333/266/200 và giao tiếp hub với ICH4-M. Ngoài ra chipset 855GME đảm bảo các khả năng tích hợp đồ họa (250 MHz) và tiết kiệm nguồn. Sơ đồ kết nối Chipset 855GME cho trong hình 5.15. Chipset 855GME có 2 chip: 82855GME (MCH) đóng vỏ 732 chân Micro-FCBGA, và 82801DBM (ICH4-M) đóng vỏ 421 chân Micro-BGA. Đặc điểm của Chipset Intel 855GME cho trong bảng 5.10.



Hình 5.15: Kết nối Chipset Intel 855GME

Bảng 5.10: Đặc điểm của Chipset Intel 855GME

Đặc điểm	Lợi ích
Bus hệ thống 400 MHz nguồn thấp	Hỗ trợ cấu hình 1 vi xử lý với bus hệ thống 400 MHz
Hỗ trợ bộ nhớ DDR333 bổ sung DDR266/200 tới 2GB.	Khi 855GME kết nối với bộ nhớ DDR333 cho hiệu suất đồ họa tích hợp cao hơn 855GM.
Công nghệ đồ họa New Intel integrated graphics solution w/ Intel Extreme Graphics 2	Khi 855GME kết nối với bộ nhớ DDR333, cho hiệu suất đồ họa tích hợp cao hơn 855GM với hình ảnh nét trung thực 3D và bộ nhớ cân bằng được dùng giữa đồ họa và hệ thống để cho hiệu suất tối ưu. Tích hợp 32-bit 3D/2D Gfx với tần số tới 250 MHz, Bi-Cubic Filter, Video Mixer Rendering hoặc AGP4x (đồ họa số)
USB 2.0 tốc độ cao	Hỗ trợ USB 2.0 tốc độ dữ liệu 40X và các thiết bị USB 1.0
Bộ đệm vào/ra linh hoạt để tách bộ nhớ và bus hệ thống xử lý	Giảm tiêu thụ nguồn của Chipset nhờ kích hoạt thông minh và giảm nguồn của bus hệ thống vi xử lý và bộ nhớ
Chốt cổng nhịp đồng hồ bên trong tối ưu cho các chế độ 3D và hiển thị	Giảm tiêu thụ nguồn của Chipset
Giao tiếp nguồn tín hiệu vi phân tích hợp nguồn thấp LVDS (Low Voltage Differential Signal)	Tích hợp cao để thiết kế các máy xách tay nhỏ
Hiển thị kép độc lập	Kết nối máy tính xách tay với màn hình ngoài hay bảng hiển thị
Quay vòng hình ảnh	Để quay vòng hình ảnh
Công nghệ ổn định hình ảnh	Cho phép thay đổi HW, giảm thiểu ảnh hưởng tới ổn định hình ảnh IT SW. Cho phép ổn định hình ảnh và nhiều người chỉ sử dụng một phần mềm đồ họa duy nhất
Hỗ trợ công nghệ tiết kiệm nguồn cho hiển thị DPST (Intel Display Power Saving Technology)	Có thể giảm nguồn tiêu thụ cho hiển thị 25% với ảnh hưởng tối thiểu đến người sử dụng phụ thuộc vào thiết lập DPST và sử dụng hệ thống.
Hỗ trợ công nghệ đồ họa tần số kép DFGT (Intel Dual-Frequency Graphics Technology)	Giảm nguồn tiêu thụ cho đồ họa, phụ thuộc vào thiết lập của người sử dụng.
Hỗ trợ tự làm tươi bộ nhớ	Tiết kiệm tiêu thụ nguồn cho Chipset và DIMM.

Core-voltage Controller (điều khiển nguồn lõi): để đáp ứng hiệu năng và công nghệ tiết kiệm nguồn tiêu thụ, Intel đưa vào chip điều khiển nguồn nuôi hai pha với phiên bản mới SC1476 IMVP-IV (các phiên bản cũ: IVMP-II, IVMP-III), thỏa mãn các đặc tính của điện áp di động của Intel IMVP-IV (Intel Mobile Voltage Positioning specification) cũng như công nghệ bậc tốc độ (SpeedStep) tân tiến để tăng thời gian nuôi của pin trong khi tối ưu hiệu năng. Chip SC1476 cung cấp cả nguồn nuôi lõi (Vcore) và VID cho các chip vi xử lý Mobile Intel Pentium. Tích hợp bên trong chip SC1476 là mạch DAC 6-bit với độ chính xác ±0,85% có thể thiết lập nguồn nuôi cho lõi vi xử lý từ 0,700 V đến 1,708 V với tối đa 40 A, và 1,2 V, 300 mA VID.

5.1.4. Các loại bus của máy tính

Trong IBM PC hoặc các máy tính tương thích sử dụng các loại bus như sau:

• Bus ISA (Industry Standard Architecture - Kiến trúc chuẩn công nghiệp): được sử dụng cho 8086, 8088, 80286, 80386 và 80486 với bus địa chỉ 24-bit và bus dữ liệu 16-bit.

• Bus MCA (Micro Channel Architecture - Kiến trúc kênh nhỏ): được IBM phát triển năm 1980 với bus địa chỉ 32-bit, bus dữ liệu 32-bit.

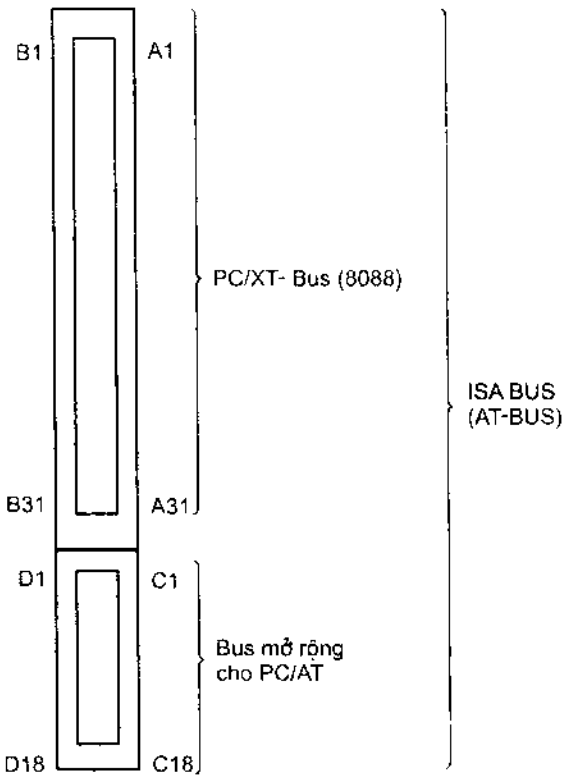
• Bus EISA (Extended Industry Standard Architecture - Kiến trúc chuẩn công nghiệp mở rộng): tương tự như MCA với bus địa chỉ 32-bit và bus dữ liệu 32-bit.

• Bus PCI (Peripheral Component Interface Bus - Bus ghép nối ngoại vi): được Intel phát triển cho bus cục bộ (local bus) dữ liệu 64-bit và tốc độ bus 266 MB/s.

• Bus VESA (Video Electronics Standards Association - Hiệp hội các chuẩn điện tử video): Với bus dữ liệu 64-bit, tốc độ bus 266 MB/s, còn được gọi là bus cục bộ VESA.

a) Bus IBM PC/XT- và Bus ISA

Trong các hệ thống PC 16-bit (IBM PC XT và PC AT) bus hệ thống và các khe mở rộng thực hiện theo chuẩn bus ISA, để mở rộng kết nối bộ nhớ và các thiết bị ngoại vi. Các tín hiệu của các khe mở rộng của bus hệ thống chuẩn ISA được liệt kê trong bảng 5.11. Khe mở rộng bus hệ thống của PC/XT-BUS gồm hai hàng, mỗi hàng có 31 chân tiếp xúc (A1-A31, B1-B31), trong đó có 20 đường địa chỉ và 8 đường dữ liệu. Vì PC/AT dùng CPU là 80286 với 24 bit địa chỉ, do đó bus hệ thống của nó phải có 24 đường bus địa chỉ và 16 đường bus dữ liệu (thường được gọi là e-sah). Như vậy bus ISA dùng trong PC/XT (8086/8088) phải mở rộng thêm để dùng cho PC/AT (hình 5.16).



Hình 5.16: Khe mở rộng của bus ISA (AT-BUS)

Bảng 5.11: Các tín hiệu của các khe mở rộng của bus hệ thống chuẩn ISA

Pin	Signal Name	I/O	Pin	Signal Name	I/O
A1	-I/O CH CK	I	B1	GND	Ground
A2	SD7	I/O	B2	RESET DRV	O
A3	SD6	I/O	B3	+5VDC	Power
A4	SD5	I/O	B4	IRQ9	I
A5	SD4	I/O	B5	-5VDC	Power
A6	SD3	I/O	B6	DRQ2	I
A7	SD2	I/O	B7	-12VDC	Power
A8	SD1	I/O	B8	0 WS	I
A9	SD0	I/O	B9	+12VDC	Power
A10	-I/O CH RDY	I	B10	GND	Ground
A11	AEN	O	B11	-SMEMW	O
A12	SA19	I/O	B12	-SMEMR	O
A13	SA18	I/O	B13	-IOW	I/O
A14	SA17	I/O	B14	-IOR	I/O
A15	SA16	I/O	B15	-DACK3	O
A16	SA15	I/O	B16	DRQ3	I
A17	SA14	I/O	B17	-DACK1	O
A18	SA13	I/O	B18	DRQ1	I
A19	SA12	I/O	B19	-Refresh	I/O
A20	SA11	I/O	B20	CLK	O
A21	SA10	I/O	B21	IRQ7	I
A22	SA9	I/O	B22	IRQ6	I
A23	SA8	I/O	B23	IRQ5	I
A24	SA7	I/O	B24	IRQ4	I
A25	SA6	I/O	B25	IRQ3	I
A26	SA5	I/O	B26	-DACK2	O
A27	SA4	I/O	B27	T/C	O
A28	SA3	I/O	B28	ALE	O
A29	SA2	I/O	B29	+5VDC	Power
A30	SA1	I/O	B30	OSC	O
A31	SA0	I/O	B31	GND	Ground

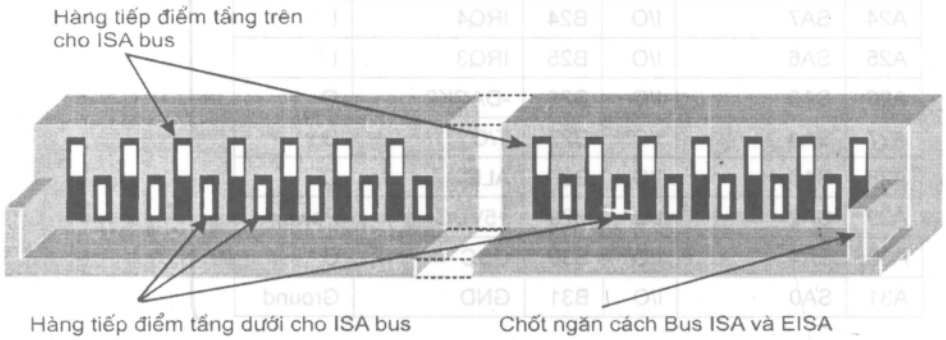
BUS PC/XT (8086/8088)

C1	-SBHE	I/O	D1	-MEM CS16	I
C2	LA23	I/O	D2	-I/O CS16	I
C3	LA22	I/O	D3	IRQ10	I
C4	LA21	I/O	D4	IRQ11	I
C5	LA20	I/O	D5	IRQ12	I
C6	LA19	I/O	D6	IRQ13	I
C7	LA18	I/O	D7	IRQ14	I
C8	LA17	I/O	D8	-DACK0	O
C9	-MEMR	I/O	D9	DRQ0	I
C10	-MEMW	I/O	D10	-DACK5	O
C11	SD08	I/O	D11	DRQ5	I
C12	SD09	I/O	D12	-DACK6	O
C13	SD10	I/O	D13	DRQ6	I
C14	SD11	I/O	D14	-DACK7	O
C15	SD12	I/O	D15	DRQ7	I
C16	SD13	I/O	D16	+5VDC	Power
C17	SD14	I/O	D17	-Master	I
C18	SD15	I/O	D18	GND	Power

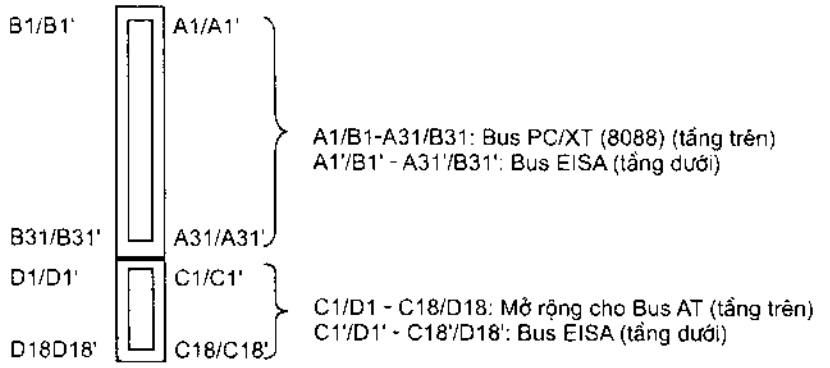
Bus mở rộng cho PC/AT-286

b) Bus EISA

Kiến trúc của 80386 và 80486 có 32 đường địa chỉ và 16 đường dữ liệu, do đó, bo mạch chủ của thế hệ PC386/486 không dùng loại bus ISA được. Một loại bus mở rộng thỏa mãn các kết nối của các hệ thống máy tính dịch vụ mạng (network servers), truyền thông (communication servers), và các hệ thống máy tính có hệ điều hành đa nhiệm (multitasking) và đa sử dụng (multiuser) với các bộ vi xử lý 80386 và 80486 là bus EISA (Extended Industry Standard Architecture, thường được gọi là eye-sah) (hình 5.17). Như vậy, với bus EISA, sự trao đổi dữ liệu giữa bộ nhớ và CPU đạt đủ đầy đủ tốc độ của 80386.



Hình 5.17a: Mặt trong của khe mở rộng Bus EISA



Hình 5.17b: Khe mở rộng của bus EISA

Bus EISA sử dụng các khe kết nối có kích thước giống như của bus ISA. Như vậy cả hai loại bảng mạch có giao tiếp ISA và EISA đều có thể cắm được vào các khe của bus EISA. Sự khác giữa bus ISA và bus EISA là khe mở rộng của bus EISA chia thành 2 tầng (level) (hình 5.17 a, b), tầng trên dành cho các tín hiệu thuộc bus ISA (bus PC/AT), tầng dưới là các tín hiệu bổ sung cho bus EISA cho các hệ thống 386/486. Nếu bảng mạch có giao tiếp ISA cắm vào khe EISA thì sẽ được kết nối ở tầng trên (top level), nơi có đầy đủ các tín hiệu của bus ISA. Bảng mạch có giao tiếp EISA có một rãnh để định vị cắm vào đúng vị trí khe kết nối cả phần trên và phần dưới mở rộng cho EISA. Các tín hiệu có kèm theo dấu -, là tích cực âm, cũng như là biểu diễn kèm theo dấu #. Các chân có tên (KEY) biểu hiện một khe cắt trên bảng các mạch đáp ứng hoặc điều khiển nhằm để chúng được cắm vào đúng vị trí của khe cắm. Các chân có tên MFG SPEC được sử dụng để đưa ra một mã xác định chủng loại của bảng mạch cắm vào khe và bổ sung vào cấu hình của hệ thống khi khởi động máy tính.

Một đặc điểm của bus EISA là các ngắt có thể được lập trình riêng biệt để thích ứng với các bảng mạch có giao tiếp EISA hoặc ISA, và xử lý ngắt từ một số nguồn khác nhau bằng cách thăm dò từng bảng mạch cắm trên khe mở rộng để xác định nguồn gây ra ngắt. Để hỗ trợ thực hiện một bus EISA, Intel có đưa ra các chip vi mạch phụ trợ: Bộ điều khiển bus 82358, hệ thống tích hợp ngoại vi 82357 và bộ điều

khiển giao tiếp chủ bus. Bảng 5.12 liệt kê các tín hiệu của khe mở rộng của bus EISA.

Bảng 5.12: Các tín hiệu của khe mở rộng của Bus EISA

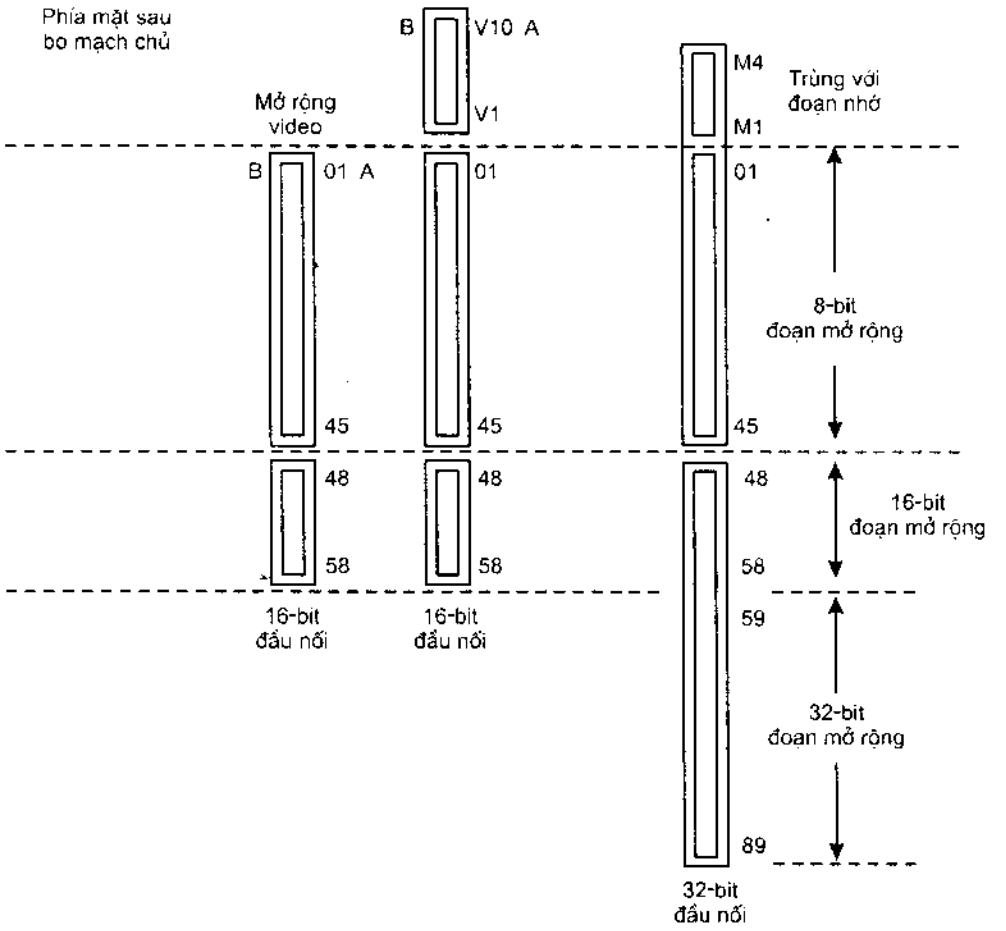
Pin	Tên tín hiệu	I/O	Pin	Tên tín hiệu	I/O
A1'	GND	Ground	B1'	GND	Ground
A2'	GND	Ground	B2'	+5VDC	Power
A3'	EXRDY	I	B3'	-5VDC	Power
A4'	-EXRDY	I	B4'	MFG SPEC	O
A5'	-EX32	I	B5'	MFG SPEC	O
A6'	GND	I/O	B6'	(KEY)	
A7'	Chốt		B7'	MFG SPEC	O
A8'	-EX16	I/O	B8'	MFG SPEC	O
A9'	SLBURST	O	B9'	+12VDC	Power
A10'	-MSBURST	O	B10'	M-IO	Ground
A11'	W-R	I/O	B11'	LOCK	O
A12'	GND	I/O	B12'	RESERVED	
A13'	RESERVED	I/O	B13'	GND	Ground
A14'	RESERVED	I/O	B14'	RESERVED	
A15'	RESERVED	I/O	B15'	BE3	O
A16'	(KEY)		B16'	(KEY)	
A17'	-BE1	I/O	B17'	-BE2	O
A18'	LA31	I/O	B18'	-BE0	O
A19'	GND	Ground	B19'	GND	I/O
A20'	LA30	I/O	B20'	+5VDC	O
A21'	LA28	I/O	B21'	LA29	I/O
A22'	LA27	I/O	B22'	GND	Ground
A23'	LA25	I/O	B23'	LA26	I/O
A24'	GND	Ground	B24'	LA24	I/O
A25'	(KEY)		B25'	(KEY)	
A26'	LA15	I/O	B26'	LA16	I/O
A27'	LA13	I/O	B27'	LA14	I/O
A28'	LA12	I/O	B28'	+5VDC	Power
A29'	LA11	I/O	B29'	+5VDC	Power
A30'	GND	Ground	B30'	GND	Ground
A31'	LA9	I/O	B31'	LA10	Ground
Mở rộng EISA BUS cho AT-BUS					
C1'	LA7	I/O	D1'	LA8	I/O

Pin	Tên tín hiệu	I/O	Pin	Tên tín hiệu	I/O
C2'	GND	I/O	D2'	LA6	I/O
C3'	LA4	I/O	D3'	LA5	I/O
C4'	LA3	I/O	D4'	+5VDC	Power
C5'	GND	Ground	D5'	LA2	I/O
C6'	(KEY)		D6'	(KEY)	
C7'	D17	I/O	D7'	D16	I/O
C8'	D19	I/O	D8'	D18	I/O
C9'	D20	I/O	D9'	GND	Ground
C10'	D22	I/O	D10'	D21	I/O
C11'	GND	Ground	D11'	D23	I/O
C12'	D25	I/O	D12'	D24	I/O
C13'	D26	I/O	D13'	GND	Ground
C14'	D28	I/O	D14'	D27	I/O
C15'	(KEY)		D15'	(KEY)	
C16'	GND	Ground	D16'	D29	I/O
C17'	D30	I/O	D17'	+5VDC	Power
C18'	D31	I/O	D18'	+5VDC	Power
C19'	-MREQn	O	D19'	-MACKn	I

c) Bus MCA

Tương tự như bus EISA, bus MCA (Micro Channel Architecture bus đảm bảo ứng dụng hiệu suất cao trong các loại máy tính dịch vụ, các hệ thống đa nhiệm, đa sử dụng, và cũng có cùng chủng loại tín hiệu như trên bus EISA. Tuy nhiên, bus MCA có sự khác biệt: các bảng mạch có kết nối MCA bus nhỏ và sử dụng các khe cắm khác loại so với bus EISA. Hình 5.18 cho ta các kiểu khe cắm MCA bus sử dụng trong IBM PS/2 Model 80. Bus MCA được thiết kế để kết nối với các bảng điều khiển ngoại vi trao đổi dữ liệu 8-bit, 16-bit và 32-bit. Với ngoại vi trao đổi dữ liệu 8-bit thì sử dụng đầu nối 46-chân, với ngoại vi trao đổi dữ liệu 16-bit thì sử dụng thêm đầu nối 12-chân. Một trong số các khe 16-bit có thêm đầu nối mở rộng 20-chân để kết nối video với sử dụng các bảng mạch điều khiển màn hình mật độ cao. Cho ngoại vi trao đổi 32-bit dữ liệu có mở rộng thêm đầu nối 44-chân thay cho đầu nối 12-chân trên khe 16-bit. Mỗi một khe 32-bit cũng có đầu nối mở rộng để kết nối trao đổi dữ liệu 8-bit. IBM PS/2 Model 80 có 5 khe cắm MCA 16-bit và 3 khe cắm 32-bit. Một trong các khe 16-bit được

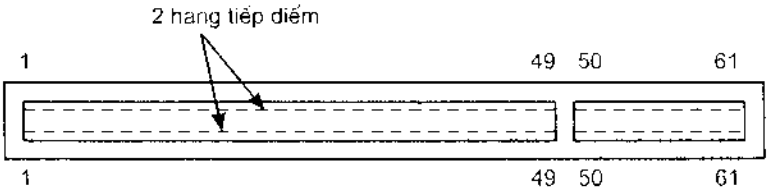
sử dụng để cho các điều khiển đĩa cứng ESDI. Các khe 32-bit có thể được sử dụng cho các bảng mạch nhớ 32-bit hoặc các thiết bị ngoại vi trao đổi dữ liệu 32-bit khác. MCA bus sử dụng một mạch logic phân phối để xét xử các yêu cầu bus từ các thiết bị chủ 16-bit và từ các nguồn có yêu cầu DMA.



Hình 5.18: Các khe mở rộng của bus MCA

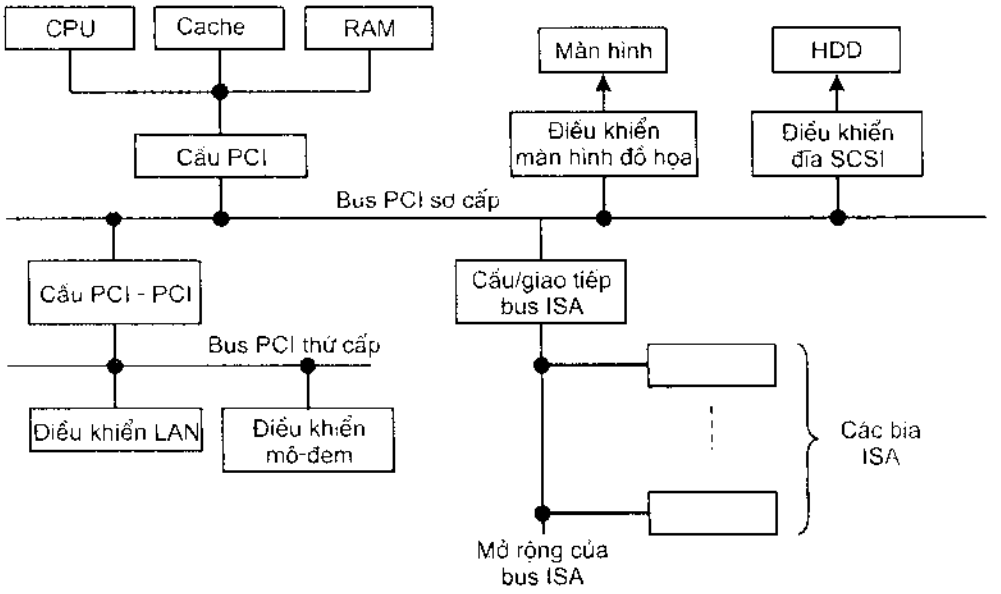
d) Bus PCI

Các khe mở rộng bus PCI (Peripheral Component Interconnect) để cắm các loại bảng mạch điều khiển I/O 32-bit với thiết bị ngoại vi (hình 5.19).



Hình 5.19: Khe mở rộng của bus PCI

Các bo mạch chủ của IBM PC 486 và Pentium có các khe mở rộng của bus ISA và của bus PCI để có thể ghép nối được với đơn vị điều khiển vào/ra 8-bit, 16-bit và 32-bit với các thiết bị ngoại vi. Sơ đồ ứng dụng đặc trưng của bus PCI hiện nay trong các máy tính cá nhân cho trong hình 5.20.



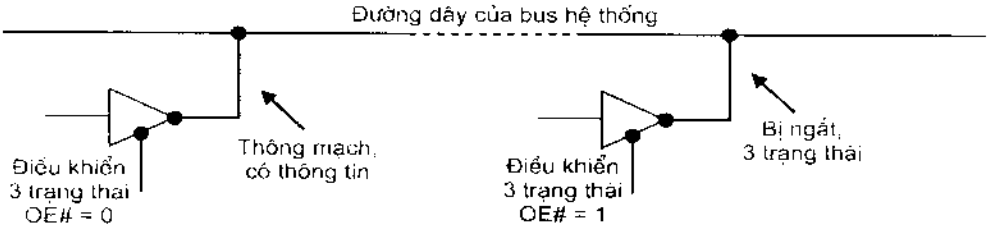
Hình 5.20: Kết nối đặc trưng bus PCI

e) Đặc điểm của các kết nối trên bus hệ thống

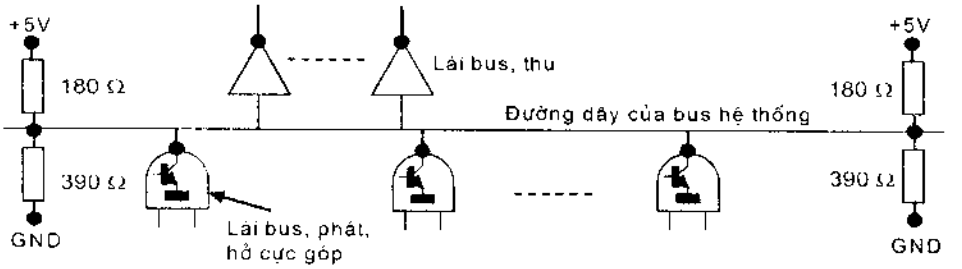
Vì các thiết bị kết nối trên bus đều có chung các đường địa chỉ, dữ liệu của bus hệ thống, mà trong một thời điểm bus hệ thống chỉ trao đổi thông tin với một thiết bị (bộ nhớ, thiết bị ngoại vi), các thiết bị còn lại phải được khóa kết nối với bus (không được chọn). Trạng thái không được chọn của thiết đặt trạng thái trở kháng cao (tri-state) ở lối ra của các mạch lái (bus driver) hay thu phát dữ liệu bus (transceiver).

Các mạch thu phát dữ liệu gồm mạch lái bus thu và lái bus phát. Do vậy phải dùng các mạch đệm/lái bus và thu phát dữ liệu bus 2 chiều có trạng thái trở kháng cao (khi các mạch này không được chọn và bị khóa khi pin OE# = 1) để tạo bus hệ thống cũng như trên các bìa điều khiển của các thiết bị (bộ nhớ, thiết bị ngoại vi). Các mạch lái bus thường có 2 loại: loại có trạng thái trở kháng cao và loại hở cực góp của tranzistor. Trong các mạch có trạng thái trở kháng cao, mỗi một mạch con (phụ trách 1 bit dữ liệu hay địa chỉ) có đầu ra trở kháng cao và đầu điều khiển trạng thái trở kháng cao (chân OE#). Trong các mạch hở cực góp thì khi cực gốc (bazơ) có tín hiệu ở mức 0 thì tranzistor đóng lại và các đường vào bị tách khỏi các đường ra.

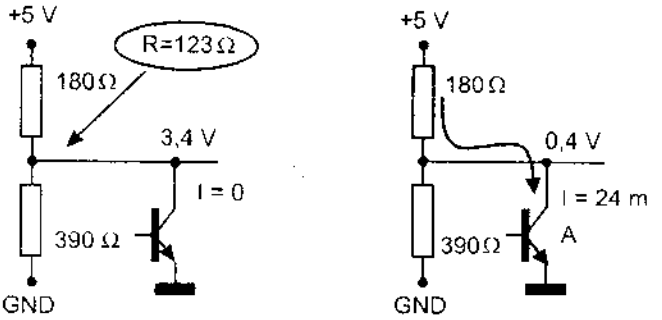
Hình 5.21 là kết nối tượng trưng của các mạch lái có trở kháng cao, hình 5.22 là kết nối tượng trưng của các mạch lái có hở cực góp trên một đường dây tín hiệu của bus hệ thống. Trên mỗi đường dây của bus hệ thống có các trở kháng kết thúc bus. Khi chỉ có một mạch lái bus hay thu phát dữ liệu bus thông thì dòng đi qua +5 V, trở kháng 180 Ω, tranzistor thông, và đất khoảng 25 mA, thiên áp đặt trên cực góp (collector) khoảng 0,4 V. Khi tất cả các mạch kết nối trên bus bị khóa, không được chọn thì các điện trở 180 Ω và 390 Ω làm thành bộ chia điện áp với giá trị trở kháng tổng là R = 123 Ω (hình 5.23).



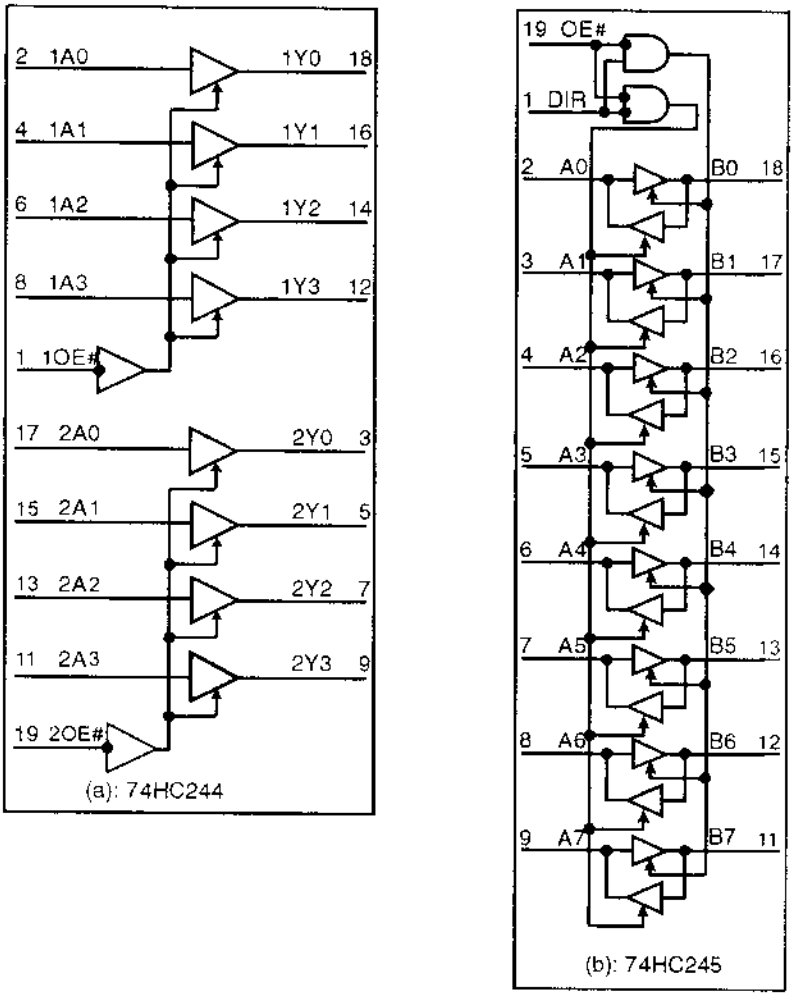
Hình 5.21: Kết nối các mạch lái bus có trạng thái trở kháng cao



Hình 5.22: Kết nối các mạch lái bus với bus hệ thống

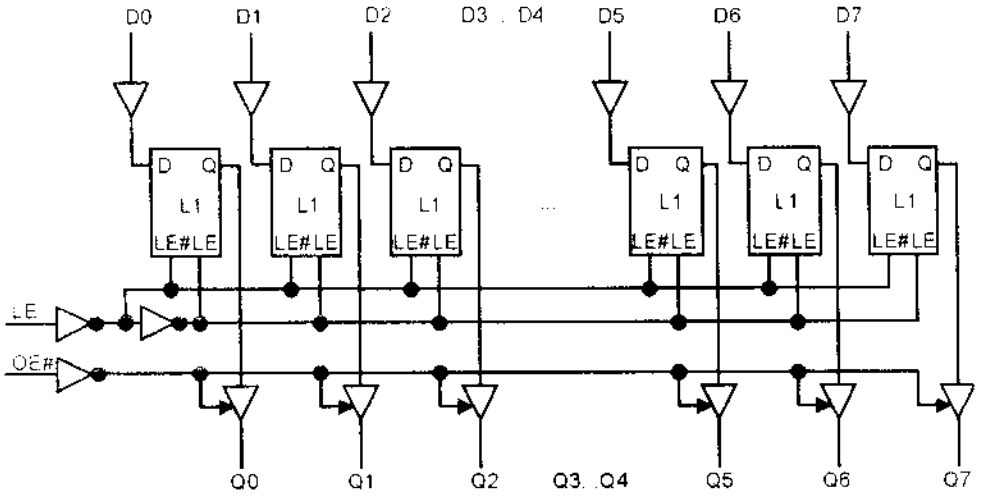


Hình 5.23: Đường dây của bus hệ thống khi có mạch lái bus thông và khi tắt



Hình 5.24: Sơ đồ logic các mạch
(a): đệm 8-bit/lái bus, 3 trạng thái; (b): thu phát 8-bit, 3 trạng thái

Hình 5.24 mô tả sơ đồ logic của hai mạch: đệm 8-bit/lái bus 3 trạng thái 74HC244 và thu phát 8-bit 3 trạng thái 74HC245 để minh họa cho kết nối các mạch này để tạo bus hệ thống. Hình 5.25 là sơ đồ logic của mạch chốt trong suốt 8-bit 3 trạng thái 74HC373.



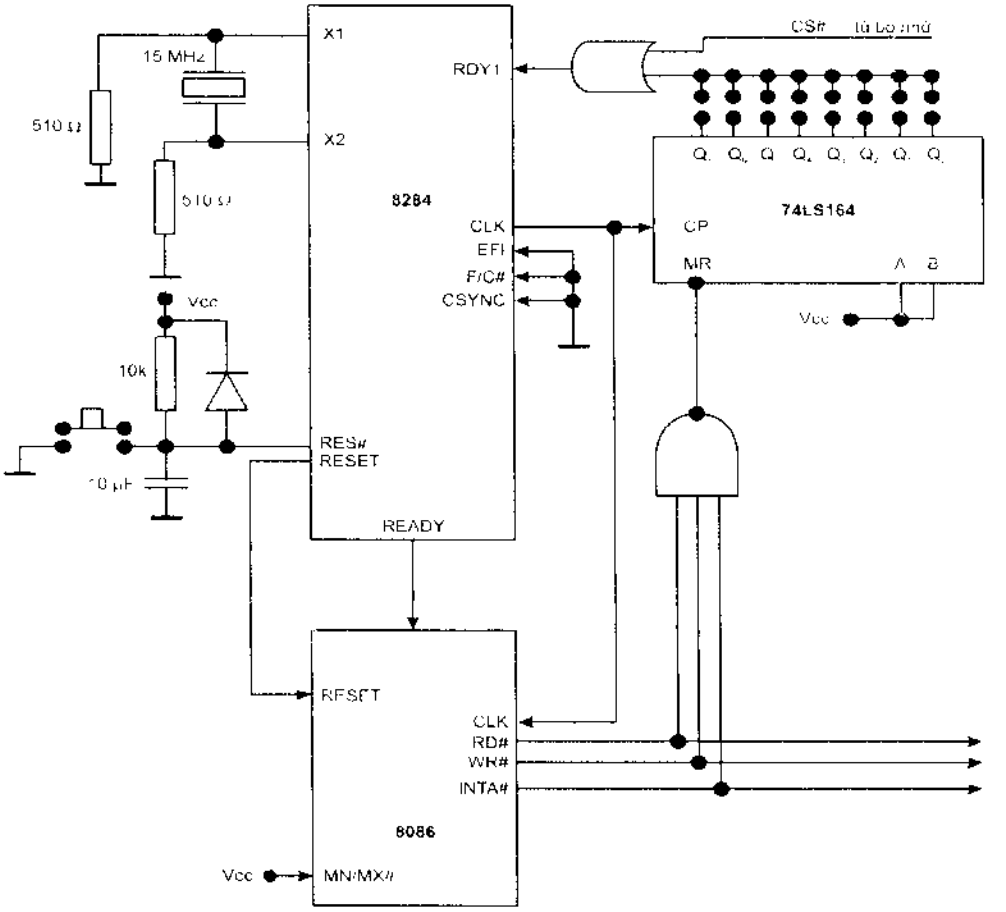
Hình 5.25: Mạch logic chốt trong suốt 8-bit 3 trạng thái 74HC373

f) Nhịp đồng của vi xử lý, định thời của bus và các trạng thái chờ với mạch phụ trợ tạo nhịp đồng hồ 8284A

Mạch tạo nhịp đồng hồ cho vi xử lý 8086/8088 thường dùng là 8284A. Chức năng chính của 8284A là tạo nhịp đồng hồ hệ thống CLK và xóa hệ thống bằng RESET, nhưng nó cũng tạo ra trạng thái chờ đợi cho vi xử lý khi kết nối với DRAM chậm nhờ tín hiệu sẵn sàng RDY và READY.

Hình 5.26 mô tả logic tạo nhịp CLK, RESET và tạo trạng thái chờ đợi đơn giản khi kết nối với DRAM chậm. Thanh ghi dịch 74LS164 nhờ xung nhịp CLK và giá trị 1 ở đầu vào dịch (A.B = 1) tạo ra ở các đầu ra Q0 - Q7 các xung với chu kỳ thời gian tính theo chu kỳ CLK (T1, T2,...). Tùy thuộc vào thời gian truy nhập của DRAM mà ta tính lấy ở đầu ra nào (Qi) xung để đưa vào mạch hoặc và kết hợp với tín hiệu CS# chọn DRAM tạo xung âm RDY vào 8284A. Xung âm 8284RDY tác động tạo xung âm READY. Xung âm 8284READY đi đến đầu vào xung READY của 8086/8088 đặt vi xử lý vào trạng thái

chờ đợi (xem đồ thị xung hình 5.3). Các tín hiệu RD#, WR#, và INTA# khi không tích cực (đều = 1) sẽ xóa thanh ghi dịch. Nghĩa là khi không có tham chiếu bộ nhớ thì thanh ghi dịch không hoạt động. Thời gian truy nhập bình thường vào bộ nhớ là 400 ns đối với 8086/8088. Một DRAM chậm cần ít nhất một trạng thái chờ khoảng 660ns. Một chu kỳ nhịp chờ đợi là Tw bằng 1 chu kỳ nhịp đồng hồ CLK. Nếu tần số nhịp CLK là 5 MHz thì chu kỳ nhịp của nó là T = 1/f = 1/5 MHz = 200 ns. Như vậy Tw cũng bằng 200 ns. Một chu kỳ bus (đọc hoặc ghi với thiết bị ngoài hay bộ nhớ) không có Tw gồm bốn chu kỳ nhịp T1, T2, T3, T4 (hình 5.1) dài 800 ns.



Hình 5.26: Kết nối mạch tạo nhịp đồng hồ 8284 với CPU 8086 trong chế độ tối thiểu với các chức năng tạo nhịp CLK, xóa hệ thống (RESET) và tạo trạng thái chờ (READY) cho RAM chậm

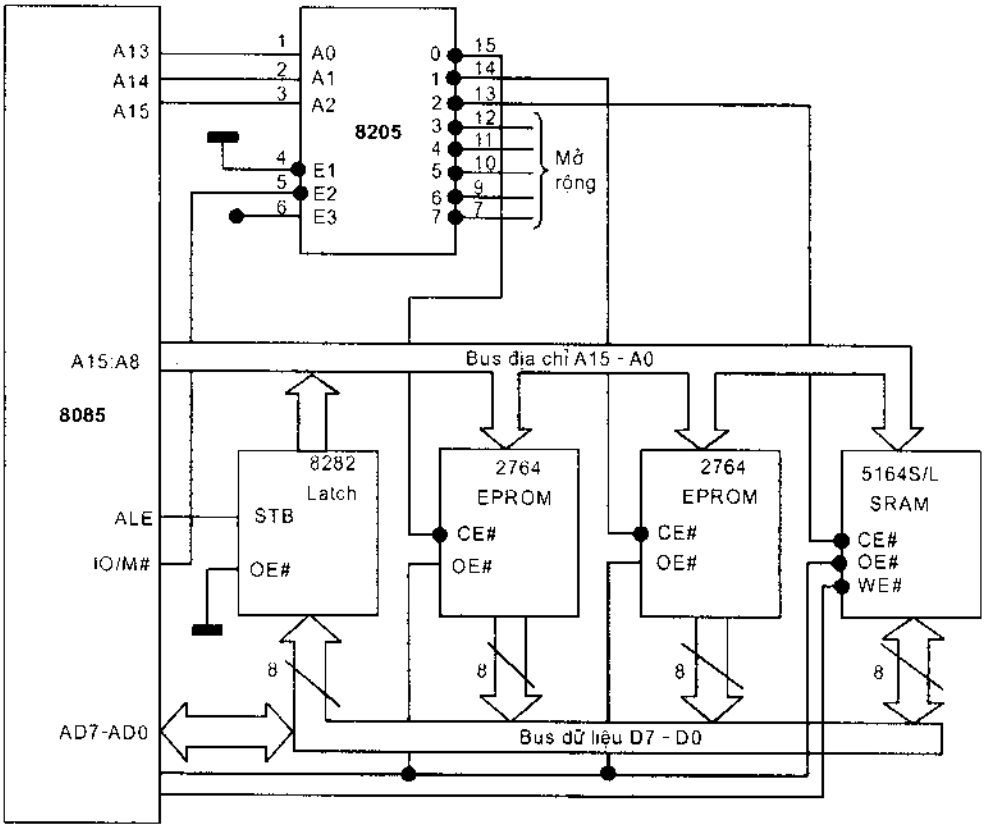
Các mạch phụ trợ tạo nhịp đồng hồ cho 80x86 cũng có những chức năng tương tự như 8284. Ví dụ, mạch tạo nhịp đồng hồ 82284 kết nối với 80286 mà ta đã xét ở chương 2 (hình 2.91 và 2.92).

5.2. PHỐI GHÉP CPU VỚI BỘ NHỚ

5.2.1. Kết nối RAM/ROM trong hệ thống 8085

Bộ nhớ được thiết kế theo mô-đun, hay những mảng nhớ. Ví dụ một mô-đun nhớ dung lượng 64 kB gồm 8 vi mạch nhớ DRAM 8 kx8. Bộ nhớ RAM với dung lượng 128 kB sẽ gồm 2 mô-đun nhớ 64 kB. Như vậy, trong thực tế để thiết kế RAM hay mở rộng RAM chúng ta chỉ cần chọn các mô-đun RAM phù hợp ghép vào bus của CPU (bus cục bộ). Chúng ta xét các trường hợp phối ghép ROM/RAM từ đơn giản đến phức tạp sau đây.

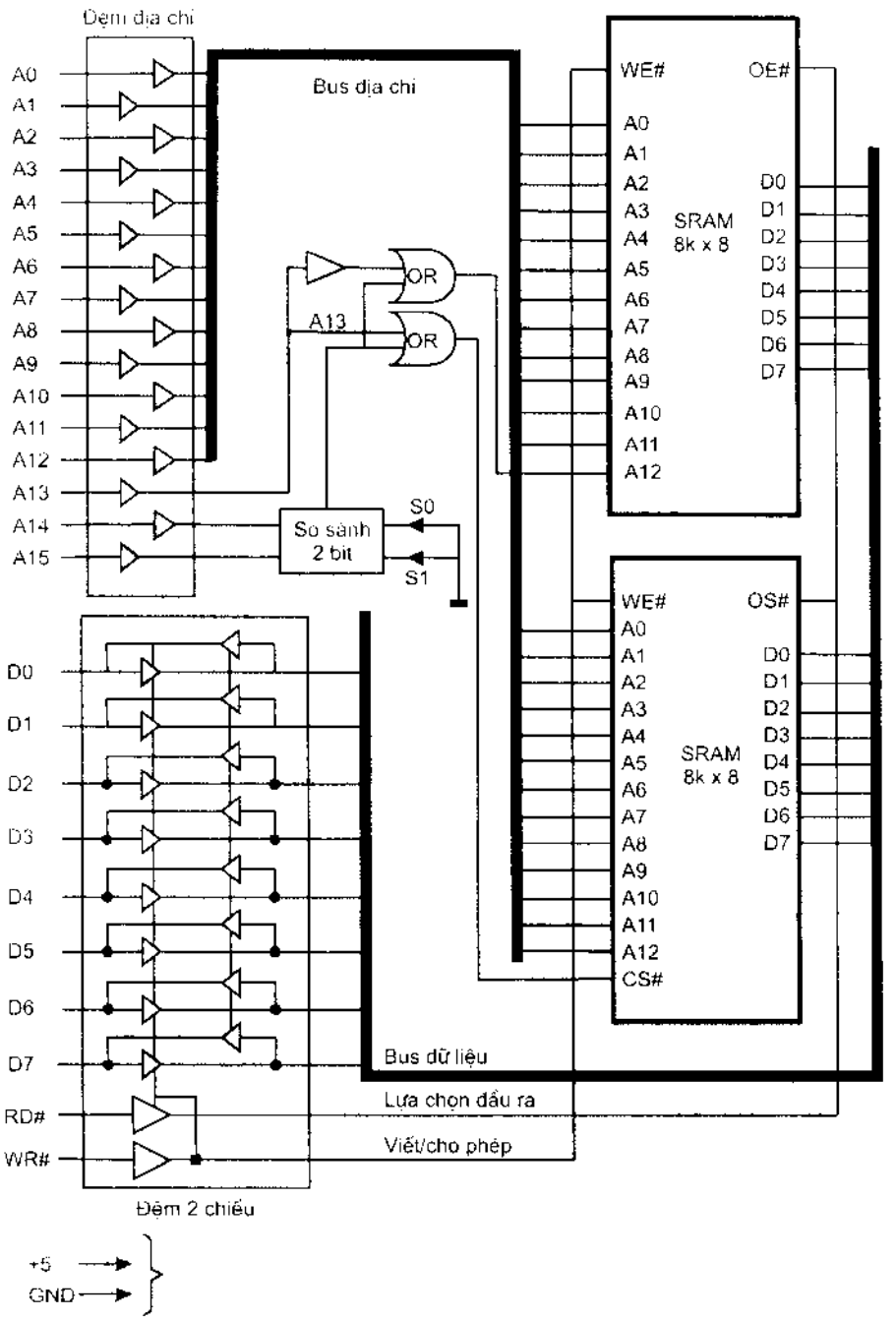
Hình 5.27 là mô tả phối ghép đơn giản các vi mạch nhớ RAM/ROM với vi xử lý 8085, trong đó tổng dung lượng RAM/ROM là 24 kx8: 2 vi mạch ROM 8 kx8 và 1 vi mạch SRAM 8 kx8. RAM/ROM được vi xử lý truy nhập tới thông qua bus hệ thống. Các tín hiệu điều khiển ghi/đọc dữ liệu (RD#, WR#, IO/M#,...) cho RAM/ROM được bộ vi xử lý tạo ra trong các trường hợp có các lệnh máy ghi/đọc bộ nhớ. RAM/ROM nằm ở các vùng địa chỉ khác nhau, tùy thuộc vào nhà thiết kế hệ thống qui định. Trong một thời điểm chỉ có một byte nhớ của RAM hoặc ROM được chọn. Trong khi một vi mạch RAM hoặc ROM được chọn thì các mạch RAM, ROM khác phải bị khóa (không được chọn). Để thực hiện được điều này, các đầu kết nối với hệ thống bus dữ liệu của các vi mạch RAM/ROM phải ở trạng thái trở kháng cao khi không được chọn (tín hiệu chọn chip CE# ở mức cao). Dữ liệu của các vi mạch ROM được đọc ra bus khi CE# = 0 và RD# = 0. Vi mạch SRAM được ghi hoặc đọc nhờ CE# = 0, WR# = 0 hoặc RD# = 0. Do đặc điểm công nghệ như vậy nên chúng ta rất dễ dàng thiết kế một hệ thống kết nối vi xử lý và RAM/ROM với dung lượng tùy ý.



Hình 5.27: Kết nối ROM/SRAM với CPU8085

Vi mạch giải mã 8205 (74LS138) có 8 đầu ra chọn ở mức tích cực thấp để chọn chip nhớ. Các đầu vào giải mã địa chỉ dùng 3 đường địa chỉ cao A15-A13. Với 3 đường vào giải mã cho phép chọn đến 8 vi mạch nhớ. Các đường địa chỉ còn lại A12-A0 được dùng để chọn byte nhớ bên trong từng vi mạch nhớ. Với 13 đường địa chỉ cho phép chọn đến 8 kB = 2^{13}. Mạch giải mã chỉ được tích cực (được chọn) khi tín hiệu vào điều khiển IO/M# = 0 (tức là chỉ khi có tham chiếu bộ nhớ).

Vi mạch chốt 3 trạng thái 8282 chốt các địa chỉ A0-A7, nhờ xung ALE (cho phép chốt địa chỉ) từ 8085, cho bus địa chỉ vì chip vi xử lý 8085 dùng chung các đường địa chỉ A0-A7 với các đường dữ liệu D0-D7, (AD0-AD7). Ban đầu vi xử lý đưa ra địa chỉ chọn ô nhớ A0-A7 và các địa chỉ này được chốt lại ở mạch chốt, sau đó, vi xử lý ghi ra hay đọc vào dữ liệu D0-D7 theo các đường chung AD0-AD7 này.



Hình 5.28: Bộ nhớ 16 kB SRAM kết nối với CPU 8085

Hình 5.28 là một sơ đồ đơn giản của một hệ thống nhớ của hệ thống vi xử lý 8085. Trong sơ đồ này, chúng ta sử dụng hai chip nhớ tĩnh 8k × 8 SRAM để được một bộ nhớ 16 kB. Để địa chỉ tới 16 kB SRAM cần phải có 16 đường địa chỉ (A0-A15). Có 8 đường dữ liệu hai chiều từ D0-D2. Đường RD# cho phép lấy dữ liệu từ chip nhớ. WR# cho phép ghi dữ liệu vào chip nhớ, hai đường này khi tích cực thì có mức điện áp thấp. Ngoài ra còn có đường cung cấp nguồn và đường nối đất.

Bộ đệm địa chỉ (A12-A0) được nối trực tiếp với IC. Địa chỉ A13 chọn 1 trong 2 chip SRAM 8k × 8. Đường địa chỉ của bộ nhớ A0-A12 có thể địa chỉ tới 8192 Byte. Cổng OR không cho phép chọn cả hai chip cùng một lúc. Địa chỉ A14 và A15 được nối đến mạch so sánh 2 bit. A14 và A15 so sánh với S0, S1 để đưa ra địa chỉ khối 16 kB của bộ nhớ, chuyển mạch S0, S1 có mức logic 0. Ví dụ nếu A14 và A15 cùng ở mức logic 0 thì bộ so sánh (magnitude compator) đưa mức 0 tới cổng gate. Nếu địa chỉ A14 và A15 cùng ở mức logic 1 thì mạch so sánh đưa ra mức logic 1.

Bus dữ liệu (D1-D7) hai chiều của hệ thống máy tính được nối với các đường dữ liệu của các chip vi mạch đệm điều khiển hướng. Các chip IC này được điều khiển bởi các tín hiệu đọc RD# và ghi WR#. Tín hiệu WE# và OS# được điều khiển bởi tín hiệu WR# và RD#.

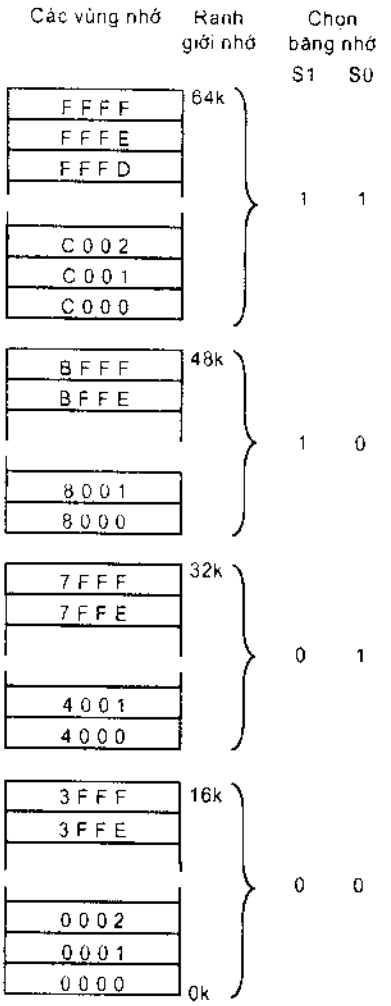
Có thể thiết kế bộ nhớ thành những khối nhớ 16 kByte. S0, S1 sẽ giúp ta chọn khối, ví dụ như ở hình 5.29.

Một bộ nhớ dung lượng 64 kB = 4 × 16 kB được chia ra 4 phân đoạn (segment), và mỗi phân đoạn được chọn nhờ tổ hợp tín hiệu S0, S1 thông qua hai đường địa chỉ A14, A15 như thế nào.

Xét quá trình đọc dữ liệu từ bộ nhớ với hai chip nhớ (hình 5.29):

1. Bộ vi xử lý đưa ra địa chỉ của byte nhớ cần đọc lên bus địa chỉ (A0-A15). Các đường địa chỉ A0-A15 ở mức tích cực.

2. Từ đơn vị logic điều khiển của bộ vi xử lý sinh ra tín hiệu đọc dữ liệu RD#.



Hình 5.29: Bộ nhớ gồm 4 đoạn 16 kB

3. Logic chọn địa chỉ bên trong bảng mạch của bộ nhớ chọn một byte nhớ trong một chip nhớ. Tín hiệu RD# được đưa vào, từ đây sinh ra tín hiệu OS#. Chip được chọn gửi dữ liệu ra bus dữ liệu bên trong của bộ nhớ. Tín hiệu RD# mở thông mạch điều khiển hướng cho 8 đường dữ liệu từ bus dữ liệu bên trong của bộ nhớ đi qua tới bus dữ liệu hệ thống của máy tính.

4. Bộ vi xử lý thông qua cổng bus dữ liệu nhận nội dung dữ liệu của byte nhớ từ bus dữ liệu hệ thống và chuyển vào đích.

Xét quá trình ghi dữ liệu vào một vùng nhớ:

1. Bộ vi xử lí đưa ra địa chỉ vùng nhớ lên bus địa chỉ (A0-A15).

2. Logic điều khiển của bộ vi xử lí sinh ra tín hiệu ghi WR#.

3. Bộ vi xử lí đưa dữ liệu thông qua cổng bus dữ liệu ra bus dữ liệu.

4. Bus dữ liệu của máy tính lưu trữ tạm thời (vào bộ đệm) dữ liệu và nối với đầu vào dữ liệu của chip nhớ .

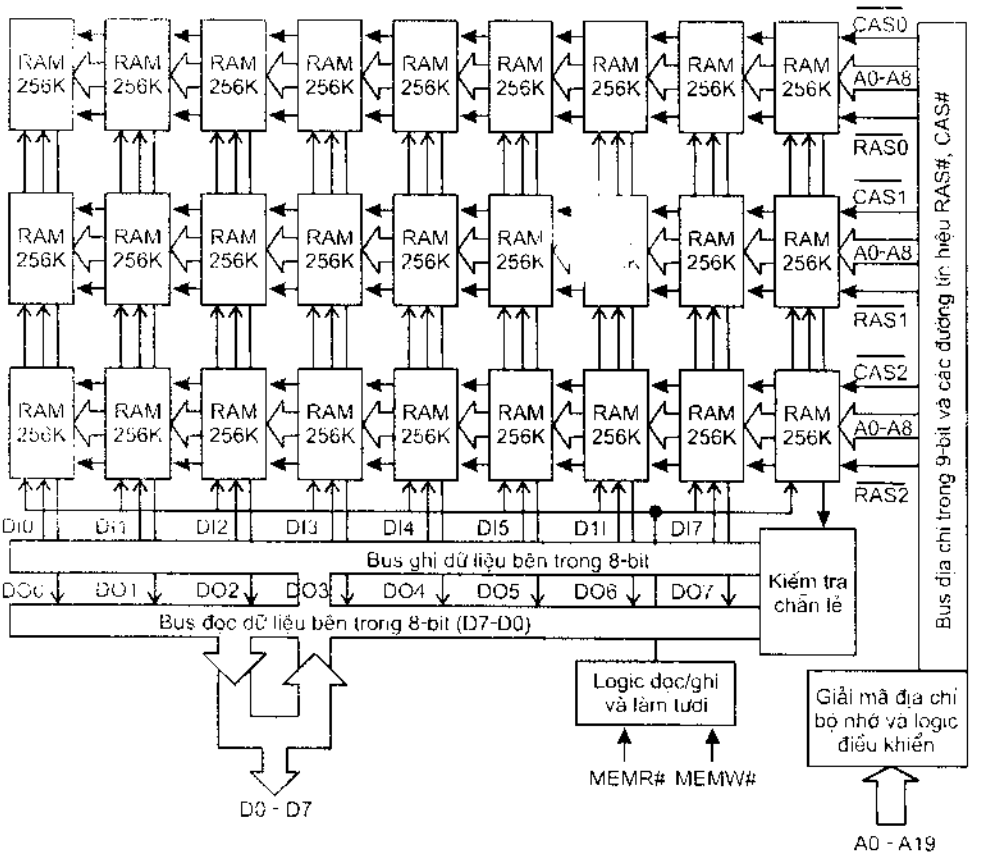
5. Dữ liệu được ghi vào các vùng nhớ đã được chọn.

5.2.2. Kết nối RAM/ROM trong hệ thống 8086/8088/80286/80386SX

Hình 5.30 là sơ đồ của bộ nhớ DRAM được sử dụng cho loại máy tính cá nhân IBM PC/XT 8088. Để thiết kế được bộ nhớ dùng DRAM có đặc tính không cố định phải cần thêm một số mạch logic khác. Đó là logic làm tươi DRAM (khoảng 300 μs). Bộ nhớ RAM tổng dung lượng 768 kB, dùng các vi mạch DRAM 256k x 1 sắp xếp theo 3 băng nhớ, mỗi băng có 9 vi mạch: trong đó 8 vi mạch dành cho các byte dữ liệu và 1 vi mạch (1-bit) dùng để tạo bit chẵn lẻ (P) cho dữ liệu. Bộ nhớ này có thêm một số khối logic có chức năng chọn hàng và cột (chọn băng nhớ), làm tươi, mạch tạo bit chẵn lẻ (EVEN, ODD) khi ghi vào và kiểm tra chẵn lẻ khi đọc dữ liệu ra.

Hình 5.31 là kết nối các tín hiệu của một chip DRAM 256k x 1 và các tín hiệu của chip vi mạch kiểm tra và tạo lập bit chẵn lẻ 74LS280 cho DRAM. Để tạo bit P khi ghi, 8-bit dữ liệu (D7-D0) đưa vào mạch tạo/kiểm tra chẵn lẻ, bit P là chẵn (EVEN), chân 5, hay khi P là lẻ (ODD), chân 6, tùy theo số lượng các bit bằng 1 trong byte dữ liệu là số chẵn hay số lẻ. Tuy vậy chỉ có thể dùng một đầu ra chân 5 hoặc chân 6 để làm bit P ghi vào DRAM mà thôi. Để kiểm tra chẵn lẻ khi đọc, dữ liệu D7-D0 và bit chẵn lẻ (P) được đưa vào mạch kiểm tra chẵn lẻ. Giả sử rằng P chọn theo chẵn (EVEN) khi ghi, tức là chân 5, thì đầu ra ODD, chân 6 được dùng làm đầu ra kiểm ra khi đọc. Nếu khi ghi P = 1, tức là số bit bằng 1 trong byte dữ liệu D7-D0 đọc ra phải là số chẵn (0, 2, 4, 6, 8), ngược lại, khi ghi P = 0 thì tổng số bit của byte dữ liệu đọc ra phải là

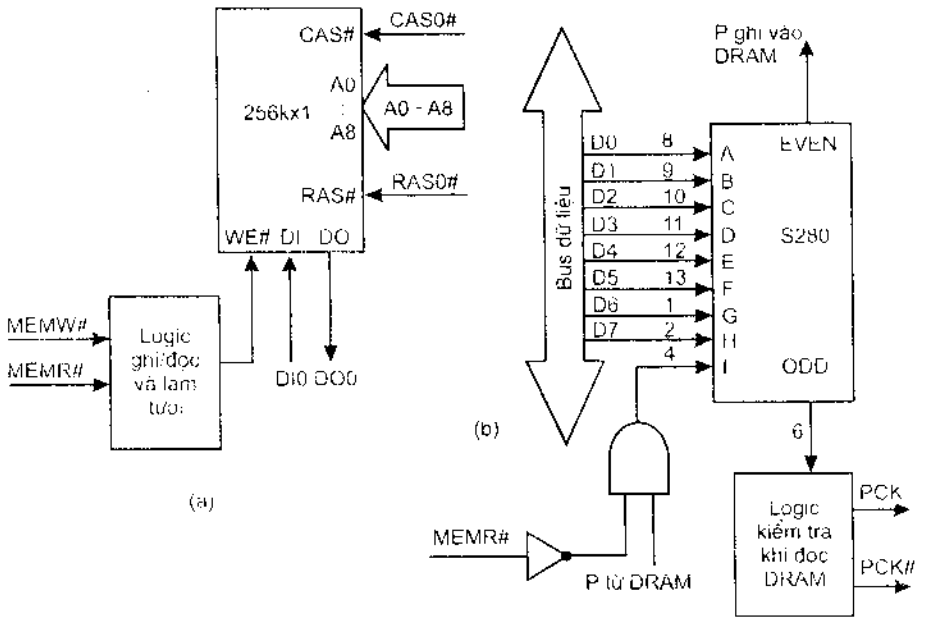
số lẻ (1, 3, 5, 7). Như vậy khi kiểm tra với P = 1 mà byte dữ liệu đọc ra không có lỗi thì đầu ra ODD, chân 6, phải = 1. Với P = 0 và dữ liệu đọc ra đúng thì chân 6 cũng = 1. Trường hợp dữ liệu sai, chân 6 = 0. Có thể tham khảo sơ đồ chi tiết phần này ở bo mạch chủ IBM PC/XT trong phần phụ lục.



Hình 5.30: Kết nối bộ nhớ DRAM dung lượng 468 kB trong IBM PC/XT 8088

Vì các vi mạch DRAM thường có dung lượng tính theo bit, ví dụ: 16 k × 1, 64 k × 1, 128 k × 1, 256 k × 1, và chúng tổ chức theo ma trận 2 chiều hàng và cột, do đó 1 bit trong ma trận được chọn khi có cả địa chỉ hàng và cột giao điểm của bit đó cùng được tích cực (xem chương 2). Do đó các bit địa chỉ của bus địa chỉ được phân chia thành 2 nhóm:

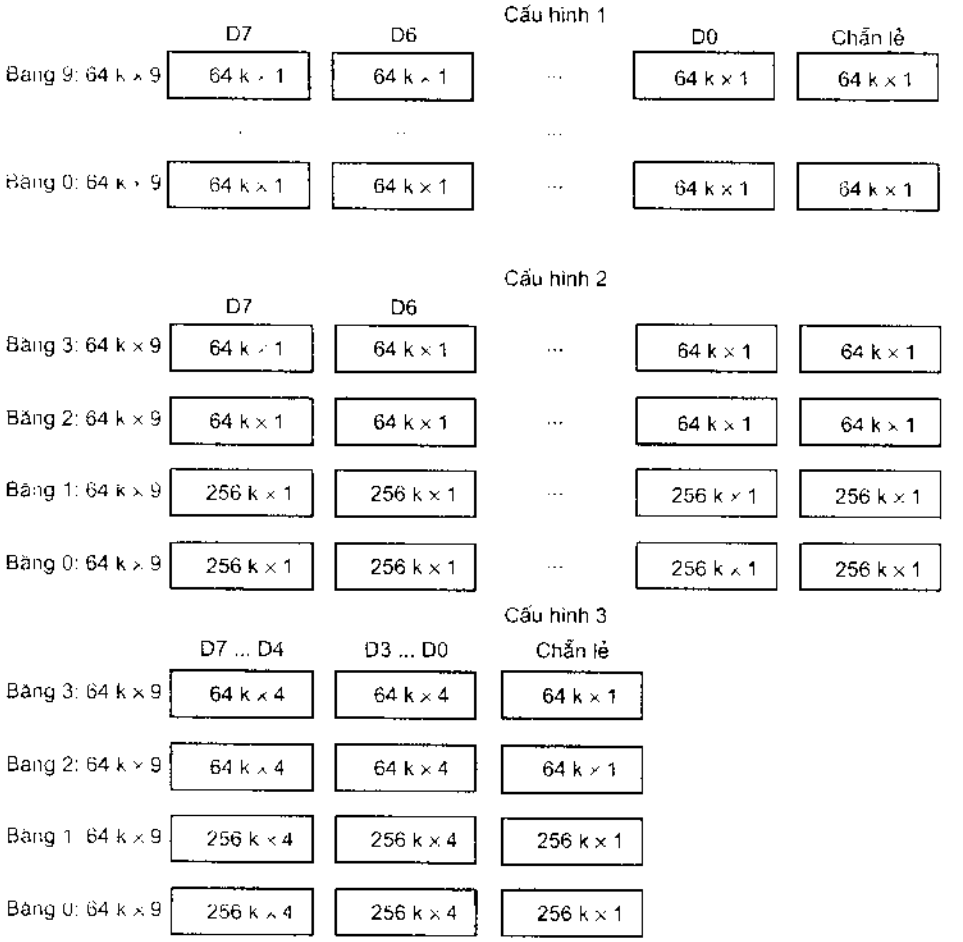
nhóm các bit cao của địa chỉ được đưa vào chọn cột, nhóm các bit thấp còn lại của địa chỉ được đưa vào chọn hàng. Để định thời cho giải mã chọn hàng cột cho các vi mạch DRAM cần có các xung RAS# và CAS#. Tương ứng với mỗi băng nhớ có các cặp tín hiệu: CAS0# và RAS0# (băng 0), CAS1# và RAS1# (băng 1), CAS2# và RAS2# (băng 2). Địa chỉ 20-bit (A19-A0) được đưa vào giải mã chọn băng nhớ và các byte nhớ trong từng băng (A8-A0).



Hình 5.32: (a): Kết nối DRAM 256k × 1, (b) Chip tạo bit chẵn lẻ cho DRAM

Có thể chọn các vi mạch DRAM dung lượng 64 k × 1, 128 k × 1 hoặc 256 k × 1 để thiết kế RAM dung lượng tùy ý. Ví dụ, trong IBM PC/XT nếu muốn có RAM dung lượng cơ sở 640 kB (chiếm dải địa chỉ 00000h - 9FFFFh) thì có thể dùng 90 vi mạch DRAM 64 k × 1 với 10 băng, mỗi băng 9 vi mạch (8 vi mạch cho dữ liệu và 1 vi mạch cho chẵn lẻ) (cấu hình 1, hình 5.32); hoặc dùng 2 băng dung lượng 512 kB dùng 18 vi mạch DRAM 256 k × 1 và 2 băng dung lượng 128 kB dùng 18 vi mạch DRAM dung lượng 64 k × 1 (cấu hình 2, hình 5.32); hoặc dùng 2 băng dung lượng 512 kB dùng 16 vi mạch DRAM 256 k × 4,

2 DRAM 256 k × 1 và dùng 2 băng dung lượng 128 kB dùng 16 vi mạch DRAM 64 k × 4, 2 DRAM 64 k × 1 (cấu hình 3, hình 5.32).

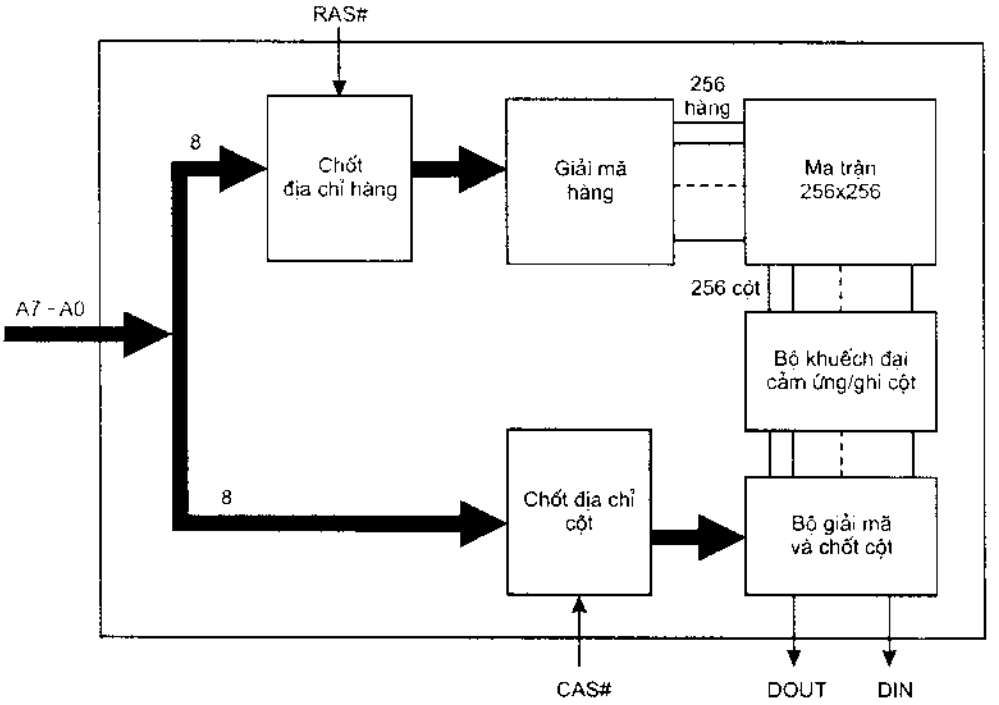


Hình 5.32: Bộ nhớ 640 kB dùng các vi mạch DRAM dung lượng 64 k × 1, 256 k × 1, 64 k × 4, và 256 k × 4

Thực chất vi mạch DRAM 64 k × 4 là 256 k × 1 và 256 k × 4 là 1 M × 1. Nếu chọn cấu hình 1 gồm toàn vi mạch DRAM 64 k × 1 thì cần giải mã tạo 10 cặp tín hiệu CASi# và RASi# (trong đó i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) tương ứng cho 10 băng nhớ. Giải địa chỉ của từng băng của bộ nhớ dung lượng cơ sở 640 kB và các giá trị địa chỉ của bus địa chỉ của CPU được cho trong bảng 5.13.

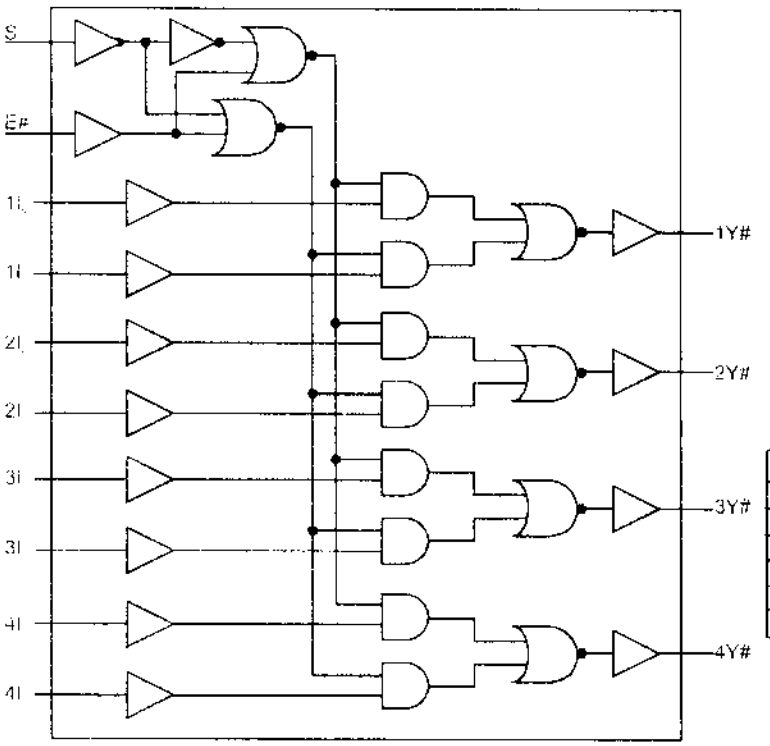
Bảng 5.13: Chọn băng (64 k × 9) cấu hình 1

A19	A18	A17	A16	A15..A12	A11..A8	A7..A4	A3..A0	Băng N	Địa chỉ Hex
0	0	0	0	0000	0000	0000	0000	0	00000
0	0	0	0	1111	1111	1111	1111	0	0FFFF
0	0	0	1	0000	0000	0000	0000	1	10000
0	0	0	1	1111	1111	1111	1111	1	1FFFF
0	0	1	0	0000	0000	0000	0000	2	20000
0	0	1	0	1111	1111	1111	1111	2	2FFFF
0	0	1	1	0000	0000	0000	0000	3	30000
0	0	1	1	1111	1111	1111	1111	3	3FFFF
0	1	0	0	0000	0000	0000	0000	4	40000
0	1	0	0	1111	1111	1111	1111	4	4FFFF
...	...	...	...	...	...	...	...	...	...
1	0	0	1	0000	0000	0000	0000	9	90000
1	0	0	1	1111	1111	1111	1111	9	9FFFF



Hình 5.33: Sơ đồ khối bên trong DRAM 64 k × 1

Như vậy để giải mã chọn băng (tức là để tạo ra các cặp tín hiệu CASi# và RASi# chỉ cần dùng đến các bit địa chỉ A19 - A16, các bit địa chỉ A15 - A0 dùng để chọn byte nhớ trong băng (tức là để chọn từng bit trong các vi mạch DRAM 64 k x 1). Vì mỗi vi mạch DRAM 64 k x 1, ví dụ 65536, có tổ chức ma trận 256 hàng và 256 cột (hình 5.33), với 8 bit địa chỉ A7-A0.



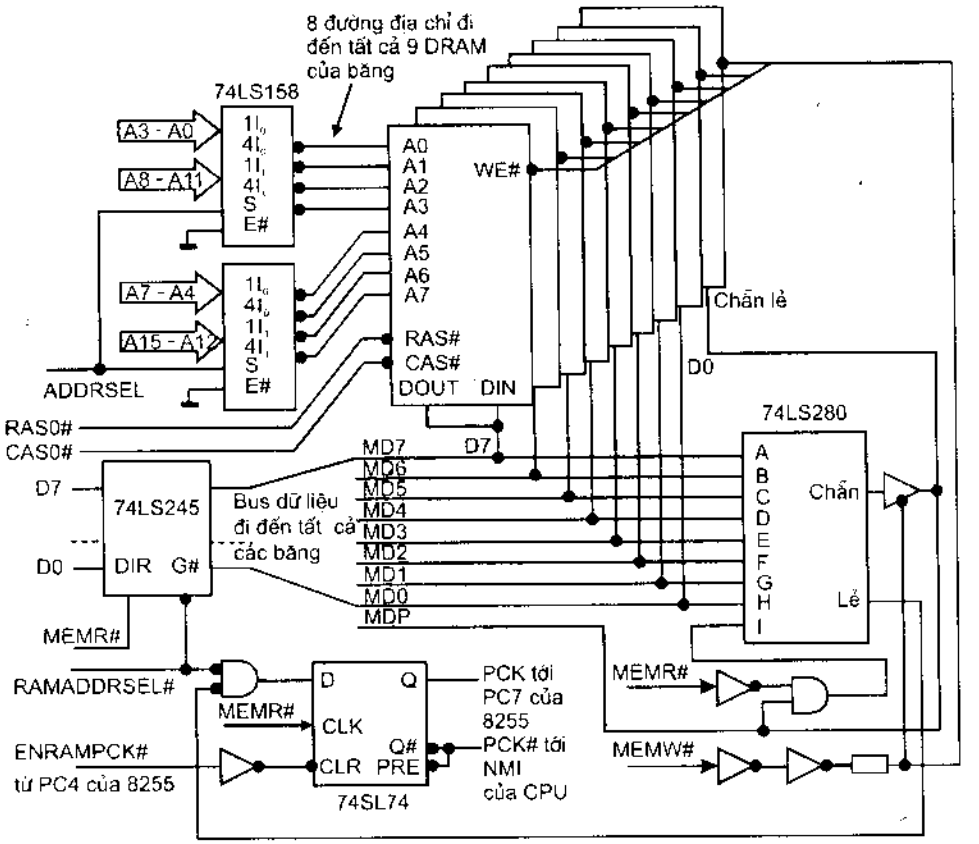
Vào				Ra
E#	S	nI	nI	nY#
H	X	X	X	H
L	L	L	X	H
L	L	H	X	L
L	H	L	L	H
L	H	H	H	L

Hình 5.34: Sơ đồ khối và bảng chân lý của mạch ghép 74LS158

Phải cần địa chỉ 16-bit (A15 - A0) để địa chỉ từng ô bit trong ma trận của vi mạch DRAM 64 k x 1, trong đó 8 bit địa chỉ A15-A8 dùng để giải mã chọn cột và 8 bit địa chỉ A7-A0 dùng để giải mã chọn hàng của ma trận. Vì vi mạch 64 k x 1 chỉ có 8 đường địa chỉ do vậy cần phải lần lượt đưa 8 bit địa chỉ thấp A7-A0 của bus địa chỉ được đưa vào chốt trong thanh ghi chốt địa chỉ hàng bên trong vi mạch DRAM nhờ xung RAS# = 0 (CAS# = 1), sau đó khoảng 75 ns 8 bit cao A15 - A8 của bus địa chỉ được chốt trong thanh ghi chốt địa chỉ cột nhờ xung

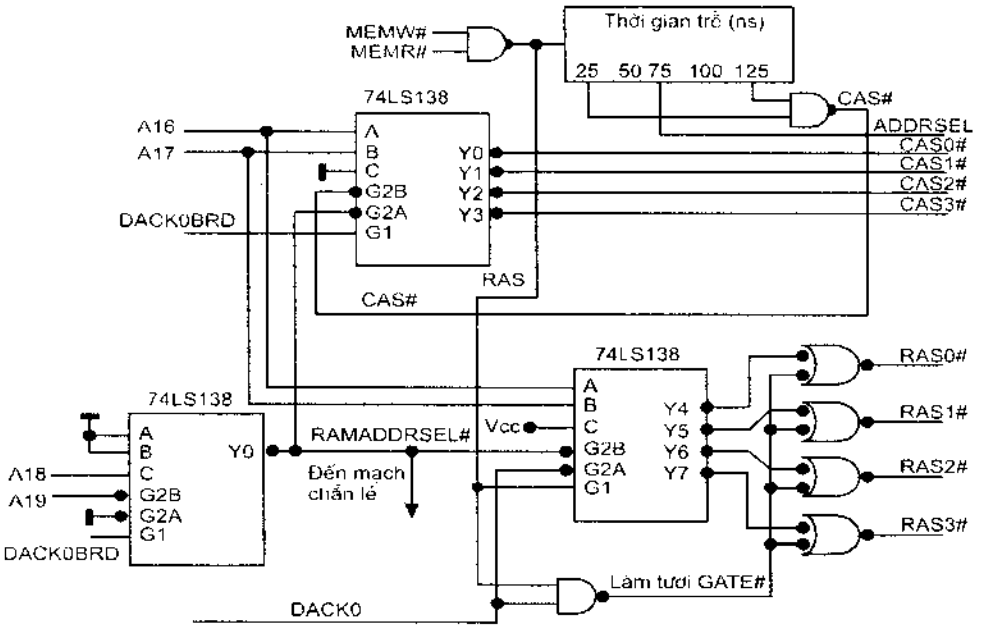
CAS# = 0 (RAS# = 1). Như vậy, để lần lượt đưa 16 bit địa chỉ (A15-A0) vào 8 chân địa chỉ của vi mạch DRAM 64 k x 1 chúng ta phải dùng đến hai mạch ghép đường 74LS158 (Quad 2-input Multiplexer) (hình 5.34). Trong đó, ở mạch 74LS158 thứ nhất: 4 đường vào nI_0 (A3-A0) dùng để tạo 4 đường địa chỉ thấp của hàng (A3-A0) và 4 đường vào nI_1 (A11-A8) dùng cho để tạo 4 đường địa chỉ thấp của cột (A3-A0). Mạch 74LS158 thứ hai: 4 đường vào nI_0 (A7-A4) dùng để tạo 4 đường địa chỉ cao của hàng (A7-A4) và 4 đường vào nI_1 (A15-A12) dùng cho để tạo 4 đường địa chỉ cao của cột (A7-A4).

Sơ đồ khối phối ghép một băng nhớ (băng 0) 64 k x 9 với giải mã chọn hàng và cột trong bảng cho trong hình 5.35.



Hình 5.35: Phối ghép DRAM 64 k x 1 trong IBM PC/XT 8088

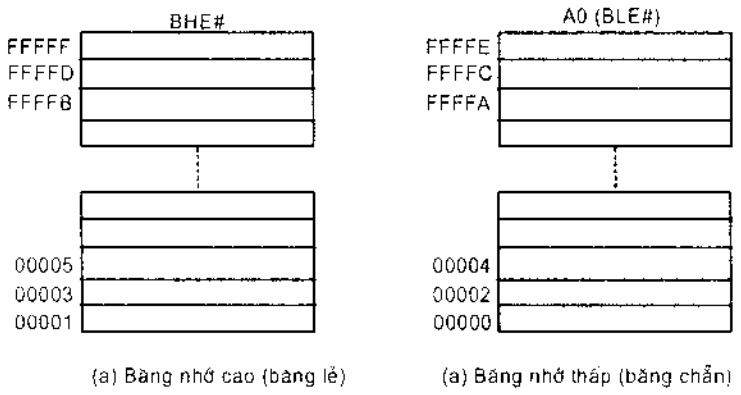
Trong hình 5.36 là sơ đồ tạo các tín hiệu RASi#, CASi# cho 4 băng (i = 0, 1, 2, 3) với dải địa chỉ 00000h - 3FFFFh. Để tạo tạo đủ các tín hiệu cho 10 băng thì mạch này phức tạp hơn nhưng tương tự. Các bit địa chỉ A19, A18, A17, A6 của bus địa chỉ đưa vào các mạch giải mã 74LS138 tạo ra các tín hiệu CASi# và RASi# (bảng 5.13). Khi chưa xuất hiệu các xung ghi (MEMW# =1) hoặc đọc (MEMR# = 1), tín hiệu CAS# = 1 cấm mạch giải mã 74LS138 tạo các tín hiệu CASi#, và tín hiệu RAS = 0 cũng khóa mạch giải mã 74LS138 tạo các xung RASi#. Nhưng với RAS = 0 thì tín hiệu ADDRSEL = 0 nhận các địa chỉ hàng A7-A0 qua các mạch ghép 74LS158 và đưa vào mạch chốt địa chỉ hàng trong các vi mạch DRAM. Khi xuất hiện xung MEMR# = 0 hoặc MEMW# = 0, tín hiệu RAS = 1 mở mạch giải mã 74LS138 tạo tín hiệu RASi# = 0 đi tới các mạch DRAM để thực hiện chốt địa chỉ hàng. Sau khoảng 75 ns, tín hiệu ADDR SEL = 1 đi đến mở các mạnh ghép địa chỉ 74LS158 (hình 5.35) để nhận 8 bit địa chỉ cột (A15-A8) từ bus vào mạch chốt địa chỉ cột trong các mạch DRAM. Sau khoảng 125 ns, tín hiệu CAS# = 0 mở mạch giải mã tạo các tín hiệu CASi# = 0 đi đến các mạch DRAM thực hiện chốt địa chỉ cột.



Hình 5.36: Chọn 4 băng nhớ DRAM: 00000h - 3FFFFh trong IBM PC/XT 8088

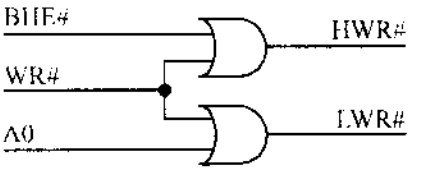
Sự phức tạp của logic điều khiển DRAM được giảm thiểu nếu chúng ta sử dụng các vi mạch điều khiển DRAM. Những mạch điều khiển DRAM thường chịu trách nhiệm ghép các đường địa chỉ và tạo các tín hiệu điều khiển DRAM. Ví dụ sử dụng mạch điều khiển 8203 các DRAM dung lượng 16 k x 1 và 64 k x 1, hoặc 82C08 điều khiển DRAM tổng dung lượng 1MB với 2 băng, mỗi băng dung lượng 256 k x 16. Với sự phát triển của công nghệ DRAM, ví dụ EDO, và SDRAM (Synchronous Dynamic RAM - RAM đồng bộ), sự phối ghép đòi hỏi phải có những logic điều khiển và làm tươi phức tạp và những logic này nằm ngay trên các mô-đun SDRAM.

Khác với 8088, để chọn băng nhớ, các vi xử lý 8086, 80286, 80386SX có tín hiệu BHE# (bus high enable). BHE# kết hợp với bit địa chỉ A0 (gọi là BLE#), dùng để chọn các băng nhớ lẻ (địa chỉ cao), chẵn (địa chỉ thấp) hoặc cả một từ dữ liệu nhớ 16-bit. Bus dữ liệu 16-bit của 8086, 80286 và 80386SX phải được chia thành hai nửa, mỗi nửa có độ rộng 8-bit, do đó có thể ghi/đọc theo byte thấp (địa chỉ chẵn, băng thấp), byte cao (địa chỉ lẻ, băng thấp) và cả 16-bit (2 băng). Hình 5.37 mô tả hai băng nhớ. Một băng chứa tất cả các ngăn nhớ có giá trị địa chỉ chẵn (băng thấp), băng còn lại (băng cao) chứa tất cả các ngăn nhớ có giá trị địa chỉ lẻ. 80286/80386SX có các tín hiệu MRDC# và MWTC# thay cho RD# và WR# như trong 8086.



Hình 5.37: Phân chia bộ nhớ thành 2 băng (chẵn, lẻ) nhờ BHE# và A0

Mạch logic tạo các tín hiệu ghi vào băng nhớ cao (HWR#) và băng nhớ thấp (LWR#) có thể được tạo ra nhờ BHE#, A0, và WR# như hình 5.38.



Hình 5.38: Mạch chọn để ghi băng nhớ

Có thể có các cách chọn băng nhớ: tạo các xung ghi băng nhớ riêng: xung ghi WR# đưa vào mạch chọn cùng các tín hiệu BHE# và A0 để tạo ra các xung ghi băng cao HWR# và ghi băng thấp LWR# (hình 5.33); và giải mã riêng cho từng băng nhớ.

Cách (1): Xung ghi băng nhớ riêng: Cách này đơn giản và chi phí thấp, chỉ cần một bộ giải mã để chọn băng nhớ. Chỉ dùng xung ghi (WR#) mà không dùng xung đọc (RD#) để chọn băng nhớ vì 8086 chỉ đọc các byte dữ liệu cần thiết thông qua một nửa bus dữ liệu 16-bit. Hình 5.39 là bộ nhớ gồm 64 kB EPROM và 128 kKB SRAM sử dụng cách chọn ghi băng bằng xung WR# cho 8086. Bộ nhớ này có 4 chip SRAM 62256 (32 k x 8) chiếm 64 k x 16-bit ở vùng địa chỉ 00000h-1FFFFh. Bốn chip SRAM được chia thành 2 băng, mỗi băng 2 chip. Băng thấp được chọn ghi bằng LWR# và băng cao được chọn ghi bằng HWR#. Các tín hiệu này đưa vào chân WE# của SRAM. Có 4 EPROM 27128 (16 k x 8) tạo ra 32 k x 16-bit EPROM chiếm vùng địa chỉ F0000h-FFFFFh (vùng cao nhất của hệ thống nhớ), và cũng được chia thành 2 băng nhớ, mỗi băng 2 chip EPROM. Mạch giải mã LS139 để chọn băng nhớ. Vi mạch LS139 có 2 mạch giải mã, một mạch dùng để chọn các băng SRAM, mạch còn lại để chọn các băng EPROM. Các mạch giải mã của LS139 được điều khiển (mở hoặc đóng) nhờ chân G#. Tín hiệu đưa vào chân G# được đưa từ các mạch logic chọn vùng địa chỉ. Ví dụ, để chọn EPROM thì phải thỏa mãn điều kiện:

(M/IO#=1)(A19=1)(A18=1)(A17=1)(A16=1) →G# = 0 → chọn EPROM.

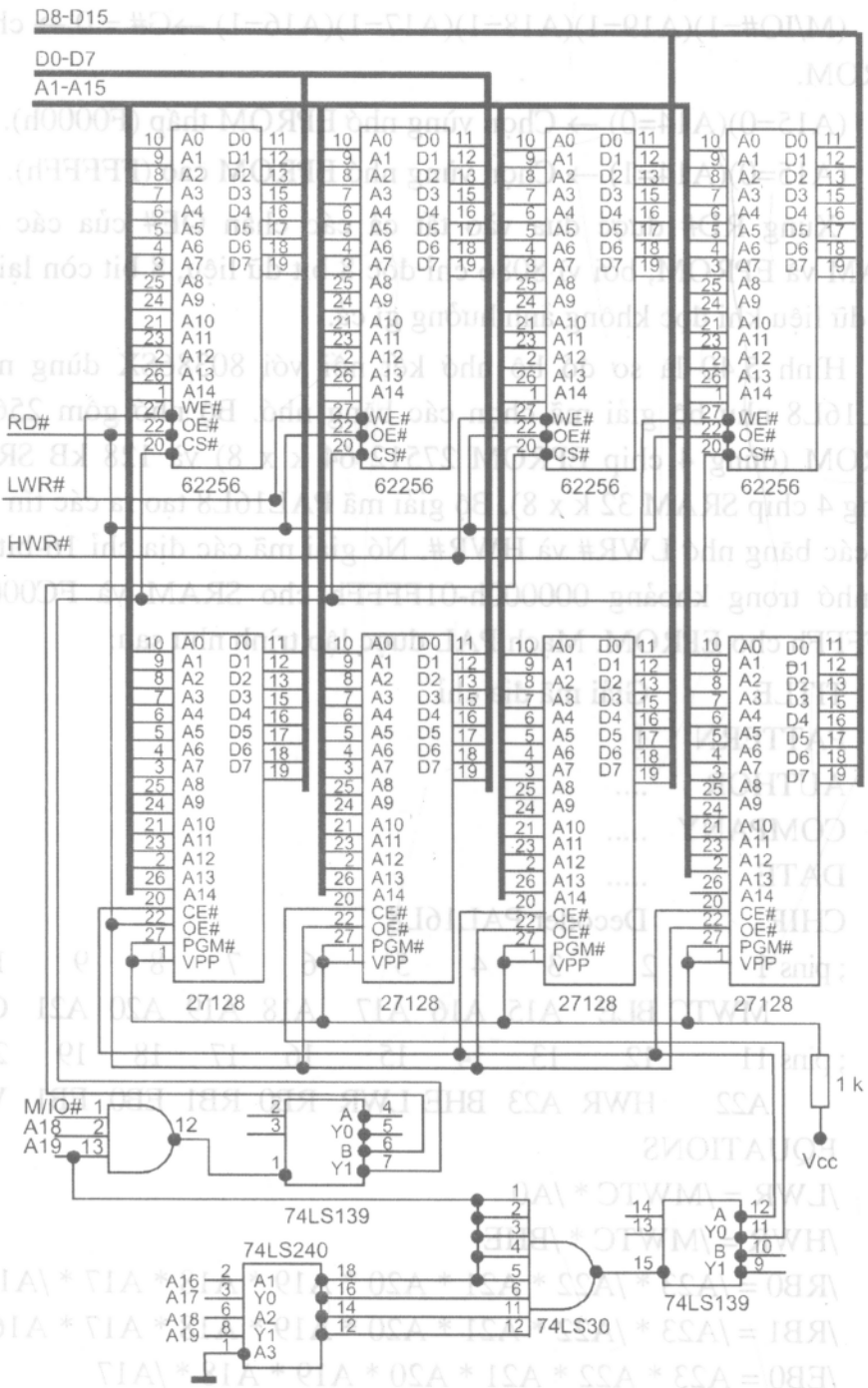
(A15=0)(A14=0) → Chọn vùng nhớ EPROM thấp (F0000h).

(A15=0)(A14=1) → Chọn vùng nhớ EPROM cao (FFFFFh).

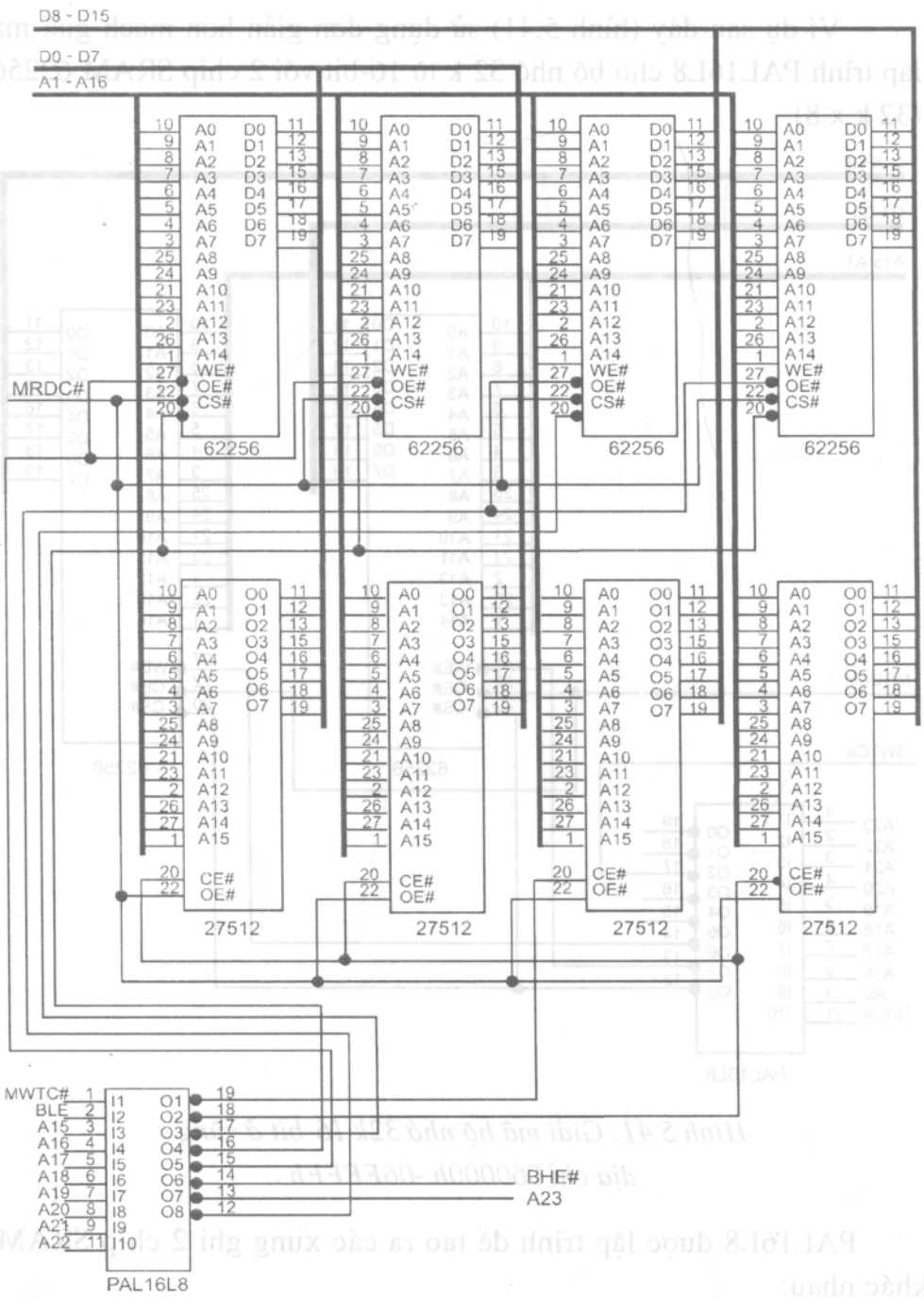
Xung RD# được đưa vào tất cả các chân OE# của các chip SRAM và EPROM, bởi vì 8086 chỉ đọc 8 bit dữ liệu, 8 bit còn lại của bus dữ liệu khi đọc không ảnh hưởng gì cả.

Hình 5.40 là sơ đồ bộ nhớ kết nối với 80386SX dùng mạch PAL16L8 như bộ giải mã chọn các băng nhớ. Bộ nhớ gồm 256 kB EPROM (dùng 4 chip EPROM 27512 64 k x 8) và 128 kB SRAM (dùng 4 chip SRAM 32 k x 8). Bộ giải mã PAL16L8 tạo ra các tín hiệu ghi các băng nhớ LWR# và HWR#. Nó giải mã các địa chỉ 16-bit của bộ nhớ trong khoảng 000000h-01FFFFh cho SRAM và FC0000h-FFFFFFh cho EPROM. Mạch PAL được lập trình như sau:

```
TITLE       Giải mã địa chỉ
PATTERN     1
AUTHOR      ....
COMPANY     .....
DATE        .....
CHIP        Decoder PAL16L8
; pins 1     2     3     4     5     6     7     8     9     10
      MWTC  BLE   A15   A16   A17   A18   A19   A20   A21   GND
; pins 11    12    13    14    15    16    17    18    19    20
      A22   HWR   A23   BHE   LWR   RB0   RB1   EB0   EB1   Vcc
EQUATIONS
/LWR = /MWTC * /A0
/HWR = /MWTC * /BHE
/RB0 = /A23 * /A22 * A21 * A20 * A19 * A18 * A17 * /A16
/RB1 = /A23 * /A22 * A21 * A20 * A19 * A18 * A17 * A16
/EB0 = A23 * A22 * A21 * A20 * A19 * A18 * /A17
/EB1 = A23 * A22 * A21 * A20 * A19 * A18 * A17
```

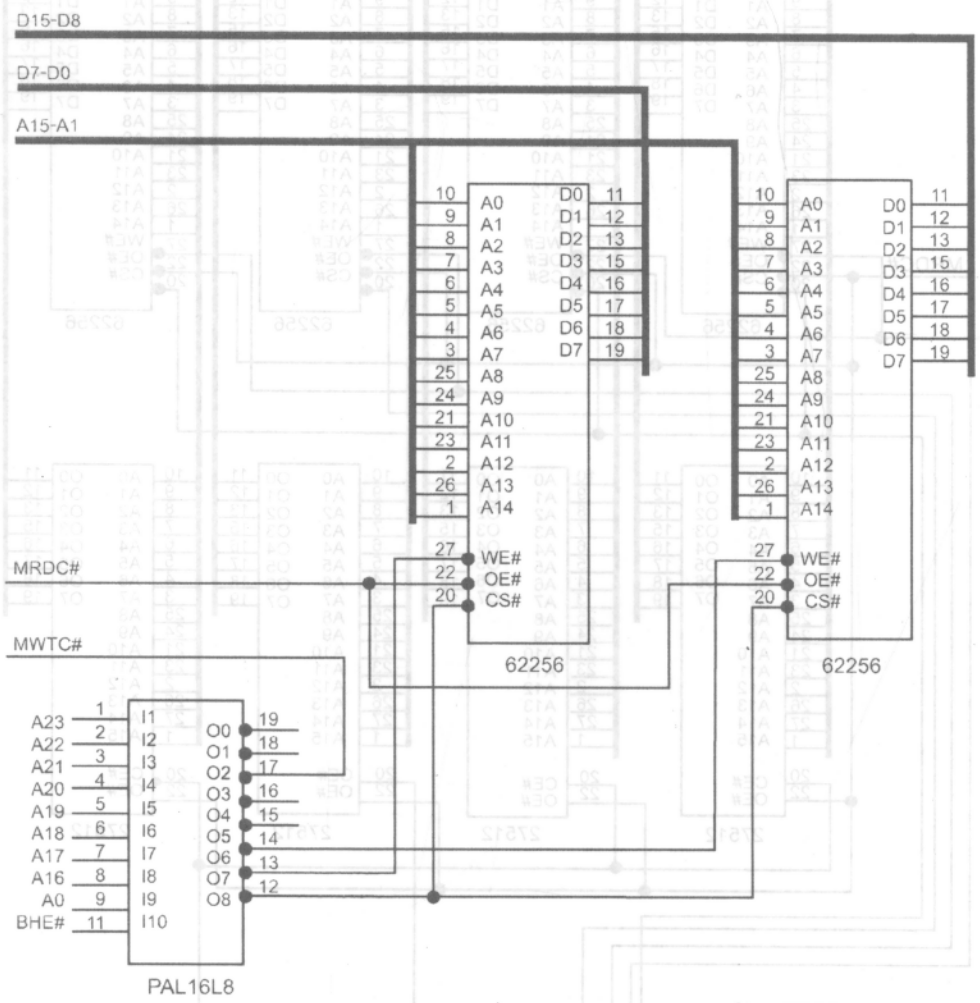


Hình 5.39: Bộ nhớ kết nối 8086 có 64 kB EPROM và 128 kB SRAM



Hình 5.40: Hệ thống nhớ cho 80386SX có 256 kB EPROM và 128 kB SRAM

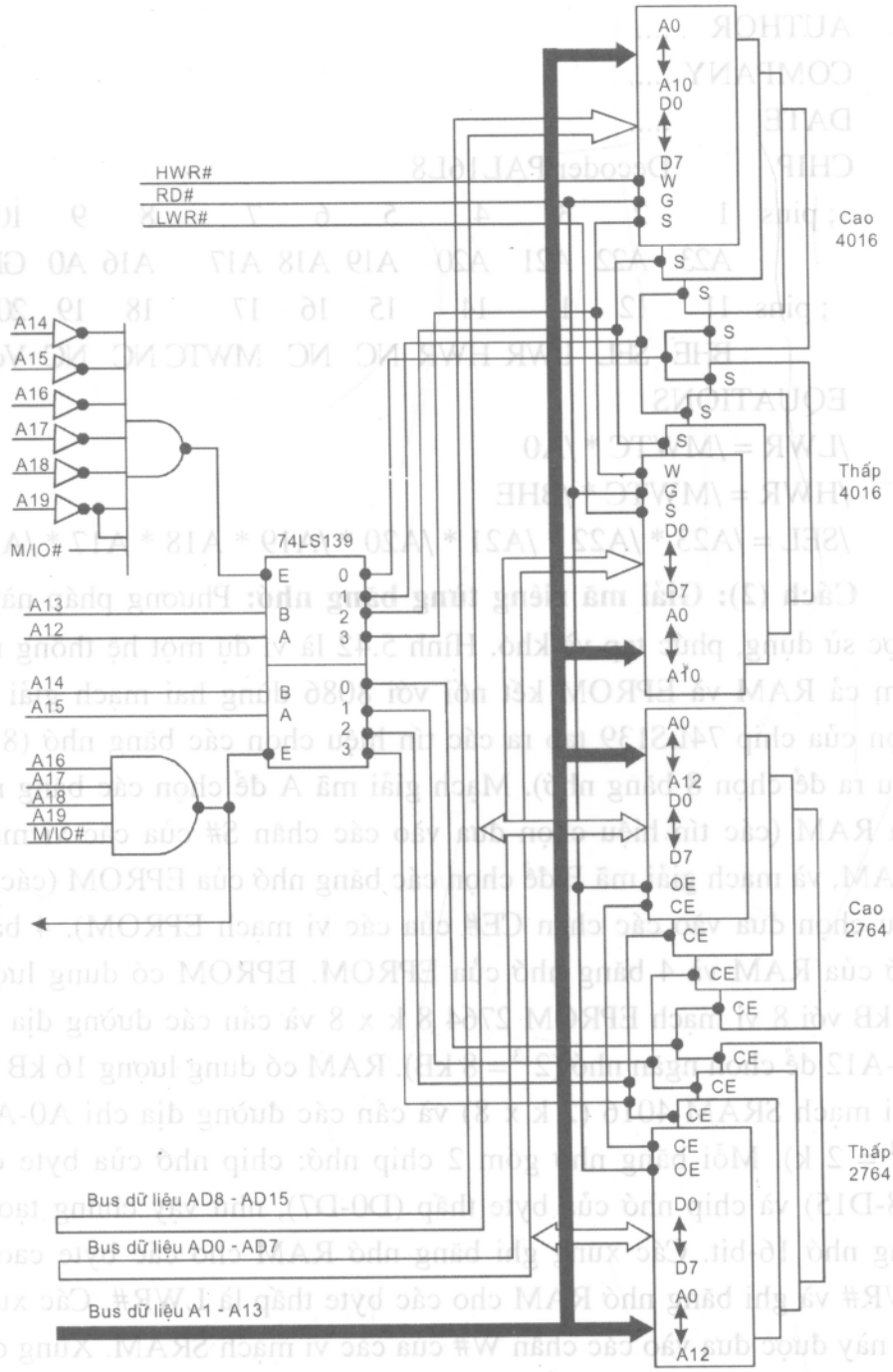
Ví dụ sau đây (hình 5.41) sử dụng đơn giản hơn mạch giải mã lập trình PAL16L8 cho bộ nhớ 32 k từ 16-bit với 2 chip SRAM 62256 (32 k x 8).



Hình 5.41: Giải mã bộ nhớ 32k 16-bit ở vùng địa chỉ 060000h -06FFFFh

PAL16L8 được lập trình để tạo ra các xung ghi 2 chip SRAM khác nhau:

```
TITLE    Giải mã địa chỉ
PATTERN  2
```



Hình 5.42: Kết nối hệ thống nhớ với 8086 dùng giải mã cho từng băng nhớ

```
AUTHOR    ....
COMPANY   .....
DATE      .....
CHIP      Decoder PAL16L8
 ; pins   1    2    3    4    5    6    7    8    9    10
          A23  A22  A21  A20  A19  A18  A17  A16  A0   GND
 ; pins   11   12   13   14   15   16   17   18   19   20
          BHE  SEL  LWR  HWR  NC   NC   MWTC NC   NC   Vcc
EQUATIONS
/LWR = /MWTC * /A0
/HWR = /MWTC * /BHE
/SEL = /A23 * /A22 * /A21 * /A20 * /A19 * A18 * A17 * /A16
```

Cách (2): Giải mã riêng từng băng nhớ: Phương pháp này ít được sử dụng, phức tạp và khó. Hình 5.42 là ví dụ một hệ thống nhớ gồm cả RAM và EPROM kết nối với 8086 dùng hai mạch giải mã chọn của chip 74LS139 tạo ra các tín hiệu chọn các băng nhớ (8 tín hiệu ra để chọn 8 băng nhớ). Mạch giải mã A để chọn các băng nhớ của RAM (các tín hiệu chọn đưa vào các chân S# của các vi mạch SRAM, và mạch giải mã B để chọn các băng nhớ của EPROM (các tín hiệu chọn đưa vào các chân CE# của các vi mạch EPROM). 4 băng nhớ của RAM và 4 băng nhớ của EPROM. EPROM có dung lượng 64 kB với 8 vi mạch EPROM 2764 8 k x 8 và cần các đường địa chỉ A0-A12 để chọn ngăn nhớ (2^{13} = 8 kB). RAM có dung lượng 16 kB với 8 vi mạch SRAM 4016 (2 k x 8) và cần các đường địa chỉ A0-A10 (2^{11} = 2 k). Mỗi băng nhớ gồm 2 chip nhớ: chip nhớ của byte cao (D8-D15) và chip nhớ của byte thấp (D0-D7), như vậy chúng tạo ra vùng nhớ 16-bit. Các xung ghi băng nhớ RAM cho các byte cao là HWR# và ghi băng nhớ RAM cho các byte thấp là LWR#. Các xung ghi này được đưa vào các chân W# của các vi mạch SRAM. Xung đọc dữ liệu từ bộ nhớ RD# được đưa tới các chân G# của các vi mạch SRAM và các chân OE# của các vi mạch EPROM. Tín hiệu RDY

thông báo cho CPU 8086 biết sự sẵn sàng của hệ thống khi đã có dữ liệu tích cực cùng với xung M/IO#.

5.2.3. Kết nối bộ nhớ trong hệ thống 80386DX/80486

Hệ thống nhớ cho 80386DX và 80486 có tổ chức khác so với 8086/80286/80386SX. Các vi xử lý 80386DX và 80486 có dữ liệu 32-bit và do đó bộ nhớ phối ghép với chúng không chỉ có 2 băng thấp và cao (chẵn và lẻ) như trong 8086/80286/80386SX mà có 4 băng 8 bit (chương 2), và 4 băng nhớ được chọn bởi các tín hiệu của vi xử lý: BE0#, BE1#, BE2#, BE3# (chương 2) được tạo ra nhờ giải mã 2 bit địa chỉ A1, A0 bên trong vi xử lý. Với địa chỉ địa chỉ 32 bit thì khoảng đánh địa chỉ của 80486 từ 00000000h đến FFFFFFFFh. Trong khi đó 80386 có 24 bit địa chỉ.

Do có 4 băng nhớ (mỗi băng độ rộng 8-bit) nên cũng phải có logic tạo/kiểm tra chẵn lẻ theo 4 băng. Các bit chẵn lẻ xuất hiện ở các chân DP0-DP3 cho mỗi băng. Chúng vừa là đầu vào vừa là đầu ra. Khi đọc, 80386DX/80486 logic kiểm tra chẵn lẻ sẽ tạo ra tín hiệu PCHK# khi có lỗi chẵn lẻ của dữ liệu đọc ra (qua tín hiệu ODD của mạch kiểm tra chẵn lẻ, tương tự như trong hệ thống 8088/8086). PCHK# được đưa tới đầu vào ngắt để thực hiện xử lý lỗi (NMI). Hình 5.43 là tổ chức của hệ thống nhớ phối ghép với các CPU 80386DX/80486 bao gồm cả mạch lưu giữ bit chẵn lẻ (DP0-DP3). Các tín hiệu chọn băng (BE3#-BE0#) đưa vào các băng nhớ với các byte dữ liệu như sau:

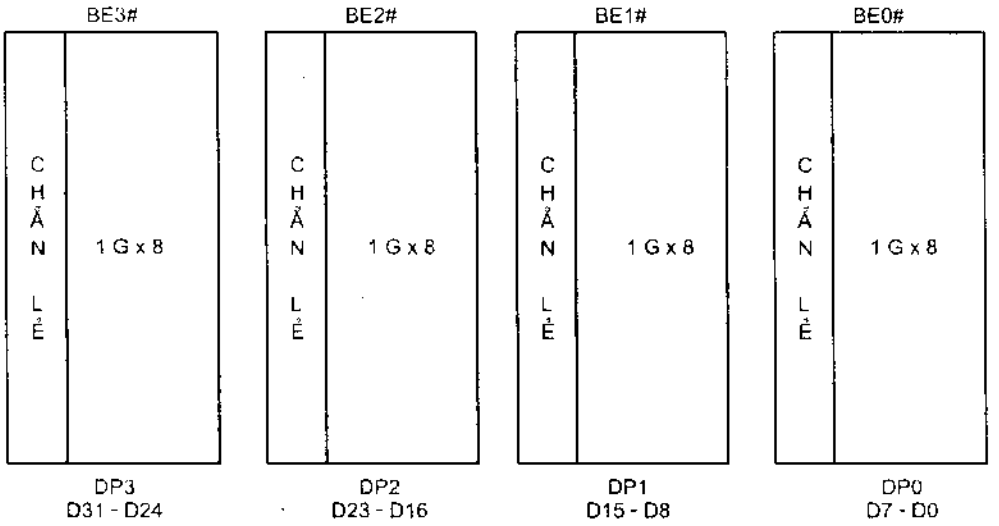
BE3#: chọn băng cao nhất: D31 - D24

BE2#: chọn băng D23 - D16

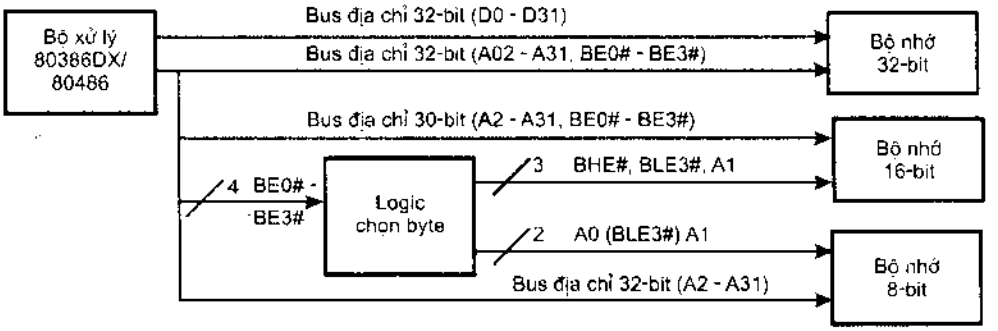
BE1#: chọn băng D15 - D8

BE0#: chọn băng thấp nhất: D7 - D0

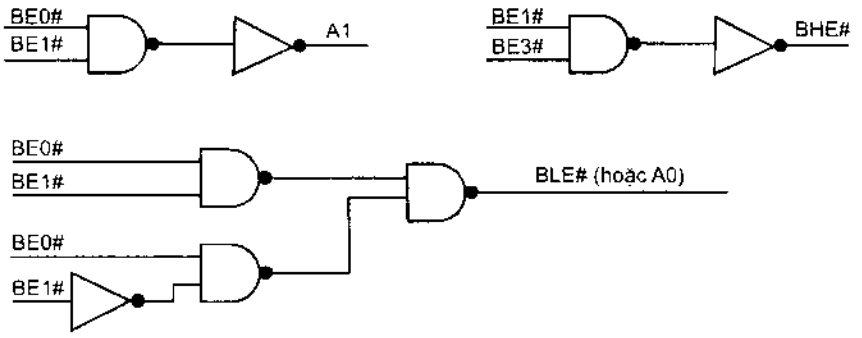
80386DX/80486 kết nối với hệ thống nhớ theo byte, từ, từ kép. Các tín hiệu bus địa chỉ và BE3#-BE0# tham gia chọn các khối nhớ theo byte, từ, và từ kép được thực hiện như mạch trong hình 5.44(a). Thông qua logic chọn byte cho (phép byte thấp) các tín hiệu A1, BLE# và BHE# (cho phép byte cao) được tạo ra (hình 5.44b).



Hình 5.43: Tổ chức bộ nhớ của 80386DX/80486



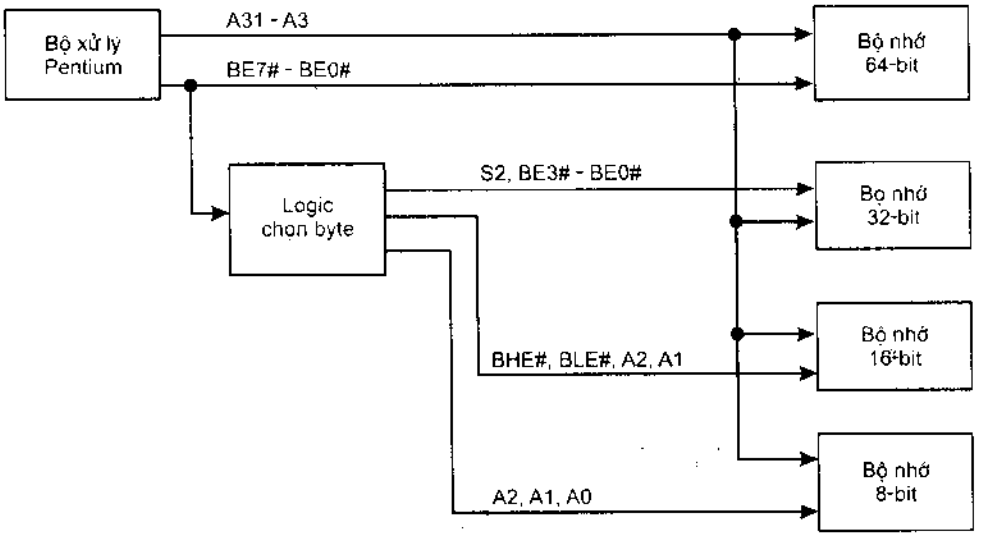
Hình 5.44(a): Tạo địa chỉ kết nối các loại bộ nhớ 32-bit, 16-bit và 8-bit



Hình 5.44(b): Logic chọn byte: tạo A1, BHE#, A0 (BLE#) cho bus 16-bit

5.2.4. Kết nối bộ nhớ trong các hệ thống Pentiums

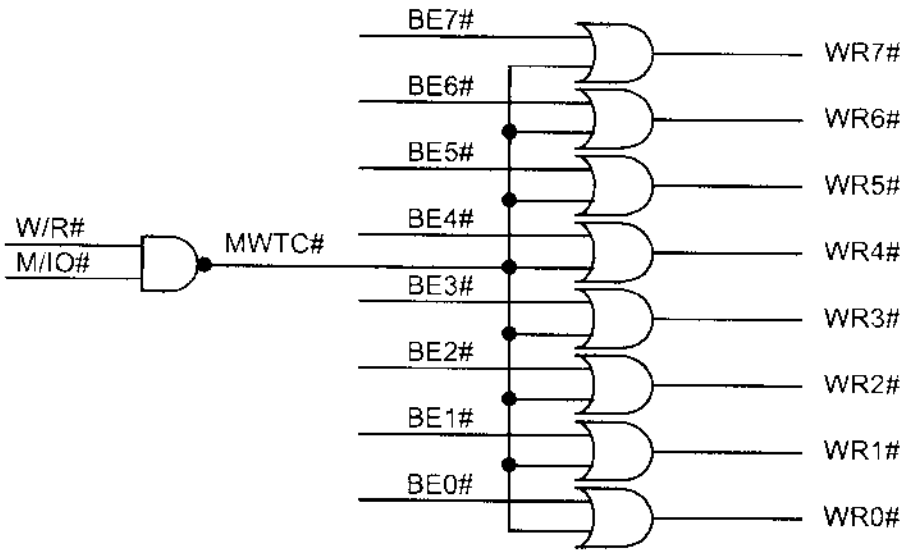
Các vi xử lý Pentium có bus dữ liệu 64-bit và cũng tương tự như 80386DX/80486, chúng có cơ chế chuyển đổi các chế độ làm việc với 8-bit bộ nhớ, bộ nhớ 16-bit, bộ nhớ 32-bit và bộ nhớ 64-bit. Cơ chế địa chỉ này được thể hiện trong hình 5.45. Cần phải lưu ý rằng, Pentium chỉ đưa ra các bit địa chỉ A31-A3 để trỏ tới ngăn nhớ độ dài 64-bit (4 từ, hay 8 byte). 8 byte liên tiếp được tham chiếu bên ngoài nhờ 8 đường tín hiệu cho chọn 8 băng RAM: BE7#-BE0. Tín hiệu BE0# chọn băng RAM thấp: D7-D0, BE1# chọn băng RAM D15-D8, v.v... và BE7# chọn băng RAM cao nhất D31-D24.



Hình 5.45: Đánh địa chỉ cho các bộ nhớ 32-bit, 16-bit, 8-bit

Để tạo ra các tín hiệu ghi các băng RAM trong các hệ thống Pentium, có logic như hình 5.46.

Hình 5.47 là phối ghép bộ nhớ RAM dung lượng 512 kB trong hệ thống vi xử lý Pentium dùng các vi mạch SRAM 27512 (64 k x 8), trong đó các xung ghi WR0# - WR7# được tạo ra từ mạch logic cho ở hình 5.46. Bộ nhớ này lấy dải địa chỉ: FFF80000h - FFFFFFFFh. Như vậy bộ nhớ này được chọn khi tất cả các bit địa chỉ A31-A19 đều bằng 1. Để giải mã chọn bộ nhớ này, ta sử dụng mạch giải mã PAL16L8.



Hình 5.46: Tạo các tín hiệu ghi các băng nhớ trong các hệ thống Pentiums

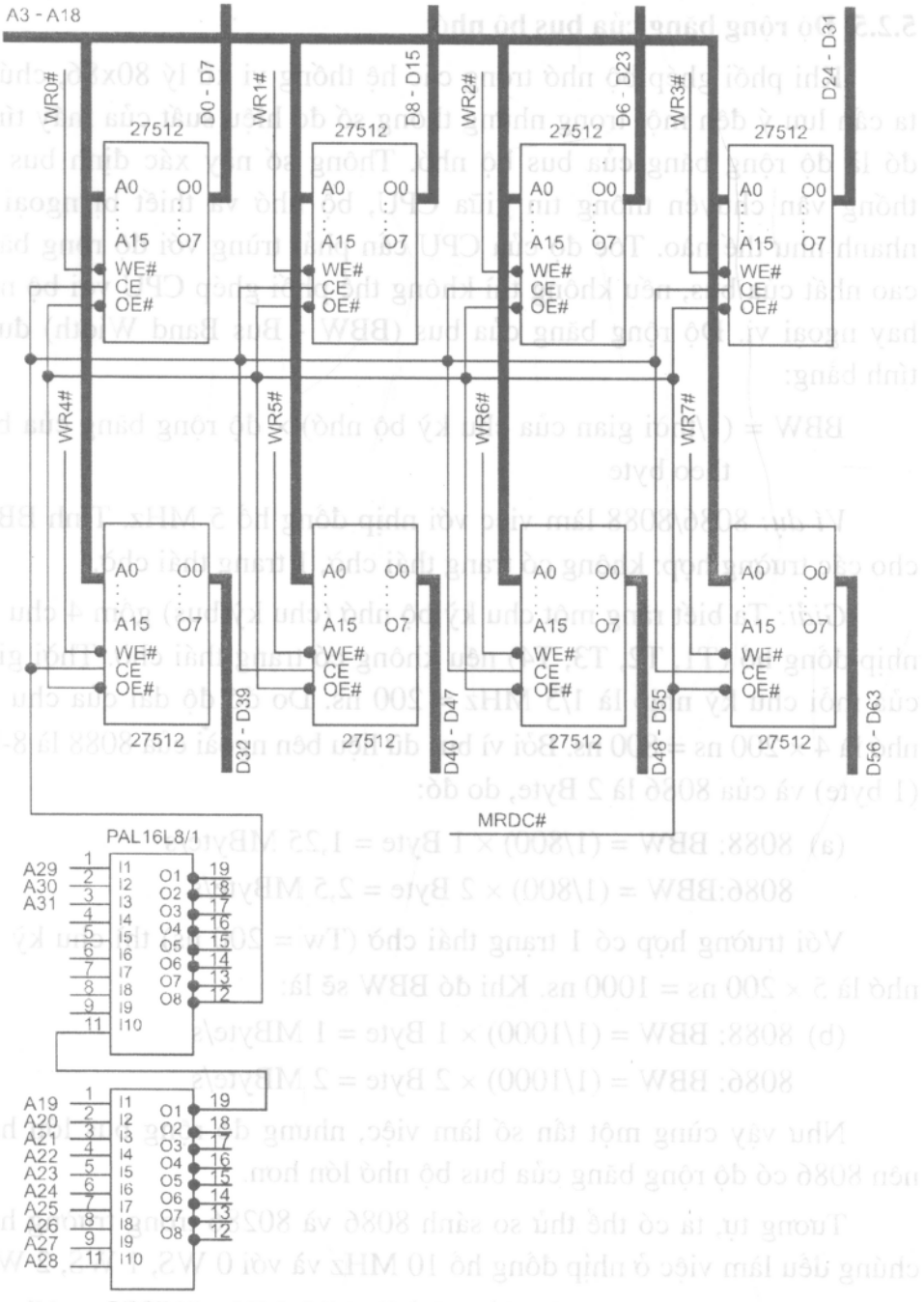
Trong đó mạch PAL16L8/2 được lập trình để:

U2# = A19.A20.A21.A22.A23.A24.A25.A26.A27.A28

Và PAL16L8/1 được lập trình để:

CE# = U2#. A29.A30.A31 (tín hiệu chọn các vi mạch SRAM)

Phối ghép DRAM với CPU đòi hỏi có logic điều khiển DRAM và logic làm tươi DRAM khi đọc. Quá trình làm tươi DRAM đã được xét ở chương 2, và được biểu diễn qua đồ thị xung ở hình 2.24. Để thực hiện làm tươi chỉ cần đưa ra địa chỉ hàng RASi# và không cần có CASi#. Từng ô bit trong hàng được chọn và được làm tươi, còn lối ra dữ liệu của DRAM duy trì ở trạng thái trở kháng cao. Trong IBM PC/XT, để thực hiện làm tươi DRAM phải dùng điều khiển truy nhập trực tiếp bộ nhớ nhờ vi mạch điều khiển DMA 8237A. Vi xử lý 80186, mặc dù không thành công trong ứng dụng cho máy vi tính, nhưng phù hợp cho các hệ thống vi xử lý nhỏ và có cấy bên trong chip khối logic làm tươi bộ nhớ động.



Hình 5.47: Phối ghép với CPU Pentium Bộ nhớ ROM dung lượng 512 kB dùng các vi mạch EPROM 27512 (64 k × 8)

5.2.5. Độ rộng băng của bus bộ nhớ

Khi phối ghép bộ nhớ trong các hệ thống vi xử lý 80x86, chúng ta cần lưu ý đến một trong những thông số đo hiệu suất của máy tính, đó là độ rộng băng của bus bộ nhớ. Thông số này xác định bus hệ thống vận chuyển thông tin giữa CPU, bộ nhớ và thiết bị ngoại vi nhanh như thế nào. Tốc độ của CPU cần phải trùng với độ rộng băng cao nhất của bus, nếu không thì không thể phối ghép CPU với bộ nhớ hay ngoại vi. Độ rộng băng của bus (BBW - Bus Band Width) được tính bằng:

BBW = (1/thời gian của chu kỳ bộ nhớ) × độ rộng băng của bus theo byte

Ví dụ: 8086/8088 làm việc với nhịp đồng hồ 5 MHz. Tính BBW cho các trường hợp: không có trạng thái chờ, 1 trạng thái chờ.

Giải: Ta biết rằng một chu kỳ bộ nhớ (chu kỳ bus) gồm 4 chu kỳ nhịp đồng hồ (T1, T2, T3, T4) nếu không có trạng thái chờ. Thời gian của mỗi chu kỳ nhịp là 1/5 MHz = 200 ns. Do đó độ dài của chu kỳ nhớ là 4 × 200 ns = 800 ns. Bởi vì bus dữ liệu bên ngoài của 8088 là 8-bit (1 byte) và của 8086 là 2 Byte, do đó:

(a) 8088: BBW = (1/800) × 1 Byte = 1,25 MByte/s
8086:BBW = (1/800) × 2 Byte = 2,5 MByte/s

Với trường hợp có 1 trạng thái chờ (Tw = 200 ns) thì chu kỳ bộ nhớ là 5 × 200 ns = 1000 ns. Khi đó BBW sẽ là:

(b) 8088: BBW = (1/1000) × 1 Byte = 1 MByte/s
8086: BBW = (1/1000) × 2 Byte = 2 MByte/s

Như vậy cùng một tần số làm việc, nhưng độ rộng bus lớn hơn nên 8086 có độ rộng băng của bus bộ nhớ lớn hơn.

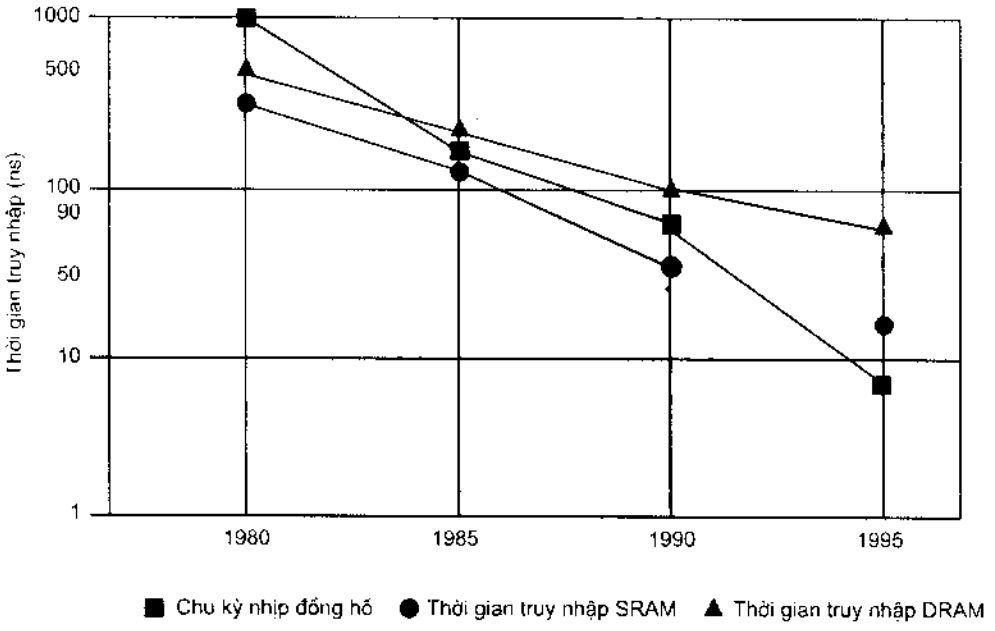
Tương tự, ta có thể thử so sánh 8086 và 80286 trong trường hợp chúng đều làm việc ở nhịp đồng hồ 10 MHz và với 0 WS, 1 WS, 2 WS.

Thời gian chu kỳ nhịp đồng hồ là 1/10 MHz = 100 ns. Nhưng 8086 có chu kỳ bộ nhớ gồm 4 chu kỳ nhịp đồng hồ, còn 80286 có chu kỳ bộ nhớ gồm 2 chu kỳ nhịp đồng hồ (xem chương 2).

WS	8086, 10 MHz	80286, 10 MHz
Thời gian chu kỳ bộ nhớ khi có 0 WS	400 ns	200 ns
Thời gian chu kỳ bộ nhớ khi có 1 WS	500 ns	300 ns
Thời gian chu kỳ bộ nhớ khi có 2 WS	600 ns	400 ns

BBW	8086, 10 MHz	80286, 10 MHz
0 WS	(1/400 ns) × 2 = 5 MByte/s	(1/200ns) × 2 = 10 MByte/s
1 WS	(1/500 ns) × 2 = 4 MByte/s	(1/300ns) × 2 = 6,66 MByte/s
2 WS	(1/600 ns) × 2 = 3,33 MByte/s	(1/400ns) × 2 = 5 MByte/s

Công nghệ bộ nhớ bán dẫn mặc dù đã tiến bộ rất nhiều nhưng về mặt tốc độ vẫn kém hơn so với công nghệ vi xử lý. Đồ thị ở hình 5.48 cho thấy thời gian truy nhập của SRAM và DRAM luôn là nhiều hơn so với thời gian chu kỳ nhịp đồng hồ của vi xử lý. Do đó, trong phối ghép bộ nhớ bán dẫn, chúng ta cần tính toán thời gian cho chu kỳ bộ nhớ phù hợp với tốc độ của vi xử lý.



Hình 5.48: So sánh thời gian truy nhập SRAM, DRAM và chu kỳ nhịp đồng hồ

5.3. PHỐI GHÉP CPU VỚI THIẾT BỊ NGOẠI VI

5.3.1. Các hình thức phối ghép vào/ra

Khi phối ghép các thiết bị ngoại vi với CPU trong hệ thống vi xử lý, chúng ta cần phải xác định những vấn đề sau đây:

1. Dải địa chỉ của thiết bị ngoại vi cần phải có để được CPU chọn đến khi tham chiếu.

2. Thông tin trao đổi giữa thiết bị ngoại vi và CPU được đồng bộ hay không đồng bộ.

Thiết bị ngoại vi theo nghĩa rộng là những thiết bị không nằm bên trong CPU và kết nối với CPU qua bus hệ thống. Nếu là hệ thống vi xử lý theo kiến trúc Von Neumann thì trong một thời điểm bus hệ thống chỉ dành cho vào/ra dữ liệu giữa một thiết bị ngoại vi và CPU mà thôi, tức là chỉ có một thiết bị ngoại vi được chọn. Muốn vậy, mỗi thiết bị ngoại vi kết nối trên bus hệ thống phải có một dải địa chỉ vào/ra (I/O address) duy nhất, không trùng với địa chỉ vào/ra của các thiết bị khác. Trong IBM PC đã có dành riêng các dải địa chỉ mặc định cho các thiết bị ngoại vi cơ bản như bàn phím, màn hình, máy in, vào/ra truyền thông, thiết bị nhớ đĩa từ (HDD, FDD, CD-ROM), con chuột, v.v,... Mỗi thiết bị ngoại vi thường gồm có các thành phần như sau:

1. Khối điện tử công suất và cơ khí, mà bản chất của chúng quyết định tên gọi của thiết bị ngoại vi.

2. Khối điện tử điều khiển nằm cục bộ trên thiết bị ngoại vi

3. Khối điện tử điều khiển phối ghép thiết bị ngoại vi với CPU, hay còn gọi là điều khiển vào/ra bộ thích ứng (I/O controller, Adaptor) là một bìa riêng cắm trên khe mở rộng của bus hệ thống hoặc nằm ngay trên bảng mạch mẹ của máy tính.

Trong khối điện tử điều khiển vào/ra thường có một số thanh ghi điều khiển, trạng thái, đếm, dịch, đệm dữ liệu vào/ra được cả thiết bị ngoại vi và CPU truy nhập tới được trong quá trình vào/ra dữ liệu. Muốn vậy, các thanh ghi này phải có địa chỉ vào/ra của hệ thống. Và để chọn được chúng phải có các mạch giải mã chọn địa chỉ vào/ra.

Đồng bộ và không đồng bộ là hai hình thức trao đổi dữ liệu giữa CPU và thiết bị ngoại vi.

+ Truyền dữ liệu đồng bộ là dữ liệu được truyền đi cố định theo nhịp đồng hồ. Tức là có thể dự đoán trước được quá trình truyền dữ liệu.

+ Truyền dữ liệu không đồng bộ là dữ liệu không được định theo nhịp đồng hồ. Tức là không thể dự đoán trước được quá trình truyền dữ liệu.

+ Truyền dữ liệu hay vào/ra dữ liệu đồng bộ giữa CPU và thiết bị ngoại vi thường được điều khiển bởi CPU và đều liên hệ trực tiếp với nhịp đồng hồ của hệ thống vi xử lý.

+ Nếu truyền dữ liệu hay vào/ra dữ liệu giữa CPU và thiết bị ngoại vi là không đồng bộ thì dữ liệu không có liên hệ trực tiếp với nhịp đồng hồ của hệ thống vi xử lý nhưng có thể đến vào bất kỳ lúc nào (từ thiết bị ngoại vi) và ở các khoảng thời gian và tốc độ không đồng nhất. Truyền dữ liệu không đồng bộ thường được khởi động ban đầu từ thiết bị ngoại vi.

Khi thiết kế phối ghép thiết bị ngoại vi với CPU trong hệ thống vi xử lý, người thiết kế cần phải chọn lựa hình thức truyền dữ liệu. Sự chọn lựa này phụ thuộc vào chủng loại thiết bị ngoại vi dự định phối ghép.

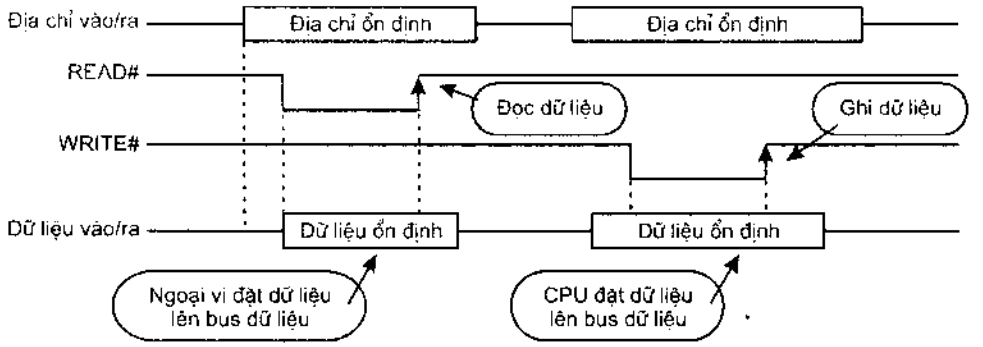
Nếu ta coi RAM/ROM là các thiết bị ngoại vi (bên ngoài vi xử lý) thì trao đổi dữ liệu giữa CPU và RAM/ROM là đồng bộ theo nhịp đồng hồ của hệ thống và song song theo một số bit (8-bit, 16-bit, 32-bit, và 64-bit). Nhưng nếu thiết bị nhớ ngoài như ổ đĩa cứng HDD (Hard Disk Drive), ổ đĩa mềm FDD (Floppy Disk Drive) hay ổ đĩa quang CD-ROM/CD-RW thì lại không đồng bộ theo nhịp của hệ thống vi xử lý mà lại theo sự điều khiển của khối điều khiển DMA và dữ liệu trao đổi giữa bộ nhớ chính bên trong (RAM) và thiết bị đĩa.

Nếu thiết ngoại vi là máy in, trong đó có nhiều khối điện tử và cơ khí thì sự trao đổi dữ liệu giữa máy in và CPU lại là không đồng bộ và song song theo bit (8-bit) cho từng mã ký tự in.

Nếu kết nối các máy tính với nhau thực hiện qua mạng diện rộng (chuyển mạch gói, chuyển mạch kênh, Internet, v.v...) thì phải phối ghép máy tính với mô-đem - một loại thiết bị ngoại vi cho truyền dẫn xa. Mô-đem đòi hỏi phải thực hiện truyền dữ liệu đồng bộ theo bit (nếu truyền tốc độ cao) hoặc không đồng bộ (nếu truyền tốc độ thấp) nối tiếp.

Cũng có thể có những thiết bị ngoại vi có thể có các phối ghép với CPU cả song song và nối tiếp, nhưng phải có một bộ nhớ đệm trung gian nhằm đảm bảo thích ứng các hình thức phối ghép và tốc độ dữ liệu ở phía CPU và cả phía phần điện tử và cơ khí của thiết bị ngoại vi. Chẳng hạn máy in có thể có phối ghép song song và nối tiếp không đồng bộ, hoặc phối ghép mạng cục bộ nhờ cạc giao tiếp mạng NIC (Network Interface Card) hỗ trợ trao đổi dữ liệu đồng bộ theo bit.

Định thời của quá trình vào/ra đồng bộ giữa thiết bị ngoại vi và CPU được mô tả bằng đồ thị xung ở hình 5.49.



Hình 5.49: Vào ra đồng bộ giữa CPU và thiết bị ngoại vi

+ Đọc đồng bộ từ ngoại vi:

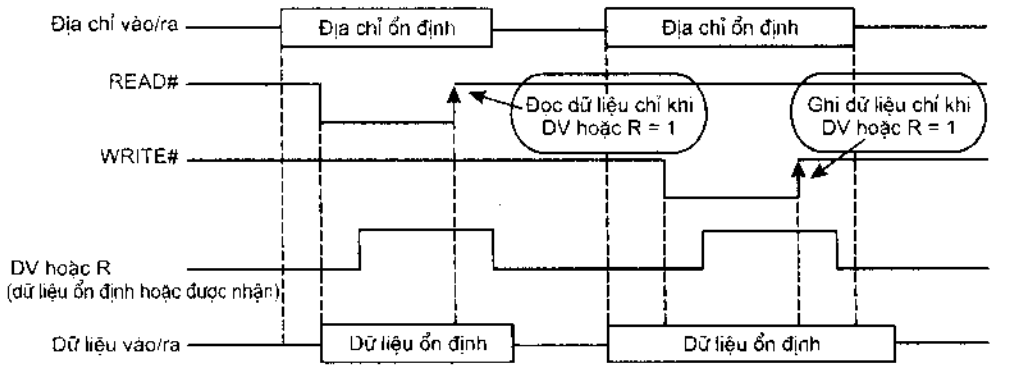
1. CPU gửi ra địa chỉ vào/ra của ngoại vi tham chiếu;
2. CPU đặt xung đọc READ# = 0;
3. Các cổng vào/ra 3 trạng thái của thiết bị ngoại vi (nằm trên khối điều khiển vào/ra) đặt dữ liệu lên bus dữ liệu;
4. CPU chuyển xung READ# = 1 để đặt các cổng vào/ra của thiết bị ngoại vi lên trạng thái trở kháng cao, và CPU lấy dữ liệu trên bus dữ liệu.

+ Ghi đồng bộ dữ liệu ra thiết bị ngoại vi:

1. CPU gửi ra địa chỉ vào/ra của ngoại vi tham chiếu;
2. CPU đặt dữ liệu lên bus dữ liệu;
3. CPU đặt xung ghi WRITE# = 0;
4. CPU đặt xung ghi WRITE# = 1 để dữ liệu được ghi ra các cổng vào/ra của thiết bị ngoại vi từ đó ra thiết bị ngoại vi.

Quá trình vào/ra này được đồng bộ bởi các xung ghi đọc của CPU, mà những xung này được đồng bộ theo nhịp đồng hồ của CPU, và CPU không quan tâm tới sự sẵn sàng của dữ liệu ra ghi thiết bị ngoại vi và dữ liệu từ ngoại vi. Quá trình vào/ra đồng bộ có ưu điểm là nhanh nhưng có nhược điểm là tốc độ bị cố định, do vậy nó thường được dùng khi phối ghép với các thiết ngoại vi có tốc độ truy nhập nhanh và cố định.

Định thời của quá trình vào/ra không đồng bộ giữa thiết bị ngoại vi và CPU được mô tả bằng đồ thị xung ở hình 5.50.



Hình 5.50: Vào ra không đồng bộ giữa CPU và thiết bị ngoại vi

+ Đọc không đồng bộ dữ liệu từ thiết bị ngoại vi:

1. CPU gửi ra địa chỉ vào/ra của ngoại vi tham chiếu;
2. CPU đặt xung đọc READ# = 0;
3. Thiết bị ngoại vi đặt dữ liệu lên bus dữ liệu;

4. Thiết bị ngoại vi chuyển tín hiệu DV hoặc R = 1 dữ liệu ổn định hoặc được nhận báo cho CPU biết dữ liệu đã ổn định hoặc nhận (đã sẵn sàng) để CPU có thể đọc - CPU đang chờ đợi có tín hiệu này lên mức 1;
5. CPU đọc dữ liệu nhờ chuyển xung READ# = 1;
6. Tín hiệu DV hoặc R chuyển xuống mức 0.

+ Ghi đồng bộ dữ liệu ra thiết bị ngoại vi:

1. CPU gửi ra địa chỉ vào/ra của ngoại vi tham chiếu;
2. CPU đặt dữ liệu lên bus dữ liệu;
3. CPU đặt xung ghi WRITE# = 0
4. Thiết bị ngoại vi chuyển tín hiệu VDC hoặc R = 1 báo cho CPU biết nó đã nhận dữ liệu trước đó;
5. CPU chuyển xung WRITE# = 1 để ghi dữ liệu ra thiết bị ngoại vi;
6. Tín hiệu DV hoặc R chuyển xuống mức 0.

Vào/ra không đồng bộ cho phép phối ghép CPU với các thiết bị ngoại vi có tốc độ chậm, không đồng nhất về thời gian và tốc độ. Bởi vì tín hiệu DV hoặc R đảm bảo làm trễ chậm lại quá trình trao đổi dữ liệu. Trong thực tế, tín hiệu DV hoặc R có thể có những tên gọi khác nhau nhưng về bản chất là để thông báo cho CPU sự sẵn sàng của dữ liệu để CPU đọc. Đây là hình thức phối ghép thông dụng nhất trong các hệ thống vi xử lý hiện nay, vì ngoài vấn đề tốc độ chậm của đa số các thiết bị ngoại vi còn có vấn đề đơn giản cho thiết kế các bộ điều khiển vào/ra.

Cũng có những thiết bị ngoại vi phối ghép với CPU theo hình thức bán đồng bộ (Semi-synchronous). Bán đồng bộ cũng tương tự như không đồng bộ, nhưng có một điểm khác là thiết bị ngoại vi không yêu cầu một sự làm trễ nào, mà thay vào đó, có logic tạo ra tín hiệu WAIT# (hoặc READY) để báo cho CPU cần có sự chờ đợi khi thực hiện vào/ra với thiết bị ngoại vi chậm và CPU kéo dài trạng thái chờ

bằng cách đưa thêm vào một số nhịp đồng hồ của hệ thống (Tw). Như vậy, quá trình này có sự tham gia của nhịp đồng hồ hệ thống. Ví dụ cho trường hợp này là phối ghép CPU với RAM/ROM chậm mà ta đã xét đến.

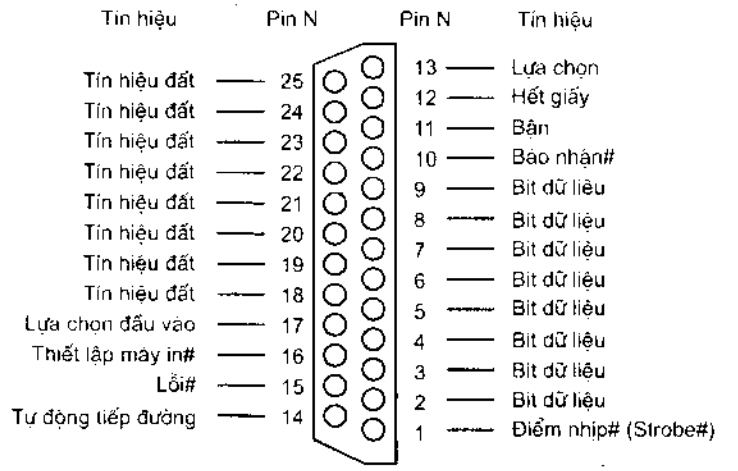
Do có những hình thức trao đổi thông tin giữa thiết bị ngoại vi và CPU như trên nên bus hệ thống phải đảm bảo được những hình thức đó. Tương ứng với những hình thức trao đổi thông tin này có những bus đồng bộ, bus không đồng bộ và bus bán đồng bộ. Các loại bus mà ta đã xét như ISA, MCA, EISA, VESA và PCI đều hỗ trợ cho các hình thức vào/ra ra dữ liệu không đồng bộ và đồng bộ.

Để đảm bảo phối ghép các thiết bị ngoại vi với CPU qua bus cần phải có các mạch lái bus/đệm bus 3 trạng thái, các mạch chốt 3 trạng thái, các mạch thu phát 3 trạng thái, tương tự như những mạch đã xét trong mục phối ghép bộ nhớ với CPU.

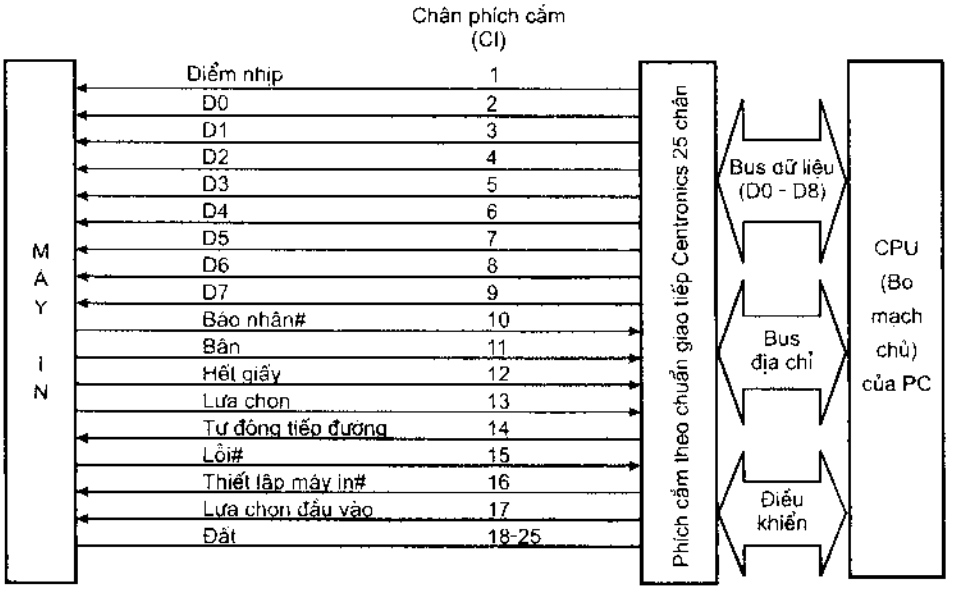
Vào/ra đồng bộ và không đồng bộ giữa CPU và thiết bị ngoại vi có thể là song song hay nối tiếp. Ngày nay các hệ thống máy vi tính đều có các hỗ trợ phối ghép với các thiết bị ngoại vi thực hiện vào/ra dữ liệu song song và nối tiếp đồng bộ và không đồng bộ nhờ một số vi mạch lập trình phụ trợ mà ta đã nhắc đến trong mục 5.1.

5.3.2. Vào ra song song

Máy in là một trong những thiết bị ngoại vi kết nối trong hệ thống vi xử lý đòi hỏi vào/ra dữ liệu song song với CPU. Máy in nhận dữ liệu đưa ra từ CPU bằng các lệnh OUT n, trong đó n là địa chỉ của cổng thanh ghi đệm dữ liệu. Dữ liệu đưa ra máy in là dữ liệu song song 8-bit (D0-D7). Vì vậy, tốc độ đưa ra dữ liệu để in là nhanh. Điều khiển đưa ra máy in cần một số tín hiệu điều khiển, kiểm tra trạng thái và báo lỗi như: máy in sẵn sàng, lỗi hết giấy. Các phối ghép dữ liệu song song thường dùng chuẩn centronics. Phích cắm (đầu nối) theo chuẩn centronics có bố trí các chân tín hiệu dạng D-shell 25-pin. Các tín hiệu điều khiển và dữ liệu song song ra máy in được sắp xếp như mô tả ở hình 5.51.



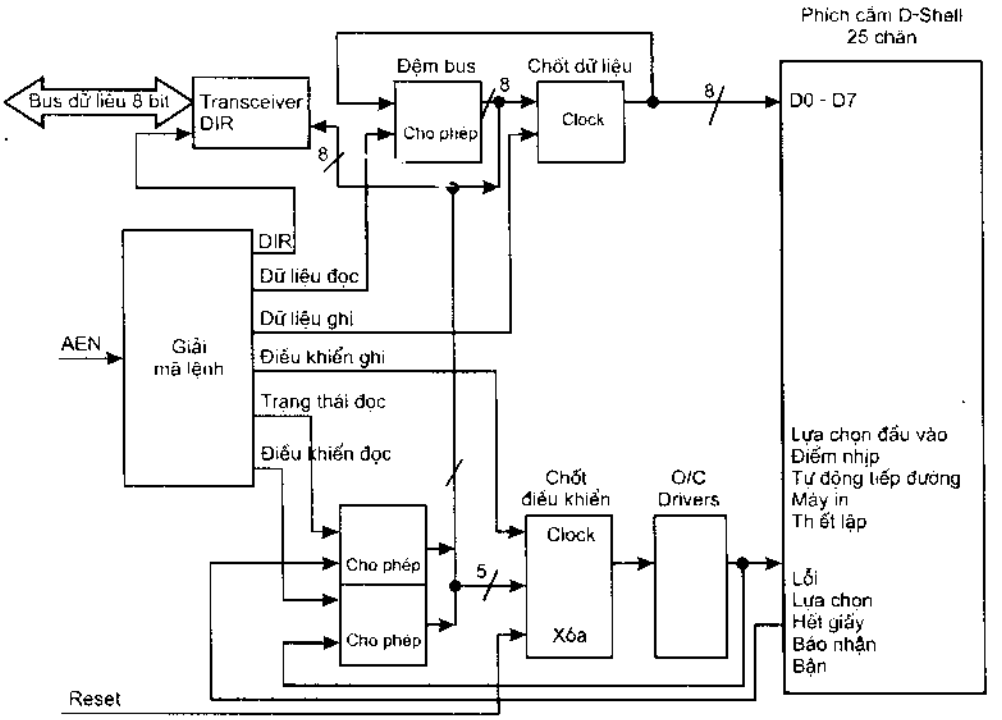
Hình 5.51: Phích cắm D-Shell CI 25 chân



Hình 5.52: Phối ghép song song máy in và CPU

Hình 5.52 là sơ đồ phối ghép máy in và CPU, trong đó thể hiện các tín hiệu điều khiển ra máy in dữ liệu song song. Khối điều khiển vào/ra song song giữa máy in và CPU (Printer adaptor, controller) trong thế hệ vi xử lý 8-bit và 16-bit thường nằm trên bảng mạch riêng hay trong bảng mạch điều khiển vào/ra đa năng (multi I/O controller)

cắm trên khe mở rộng của bus hệ thống của hệ thống vi xử lý. Sơ đồ khối chức năng của logic điều khiển đưa dữ liệu song song (D0-D7) ra máy in mô tả trong hình 5.53. Nó có thể được tham khảo mẫu cho các loại đưa ra song song tương tự. Công nghệ máy tính hiện nay đã cấy các chip vào/ra đa năng: máy in, truyền thông ngay trong một chip điều khiển vào/ra và tích hợp ngay trên bảng mạch mẹ với CPU.



Hình 5.53: Sơ đồ khối điều khiển đưa ra máy in khối nối với IBM PC/ XT (8088)

Loại máy in này là loại máy in ma trận kim có đầu in, được điều khiển chạy dọc bàn trượt cố định. Dữ liệu đưa ra (D0-D7) là mã ASCII của ký tự. Tại máy in mã ASCII sẽ tương ứng với mẫu ma trận kim tác động lên rubăng mực tạo ra ký tự in. Trong hệ thống địa chỉ vào/ra của IBM PC/XT, AT, khối chức năng điều khiển đưa ra máy in dùng 3 địa chỉ I/O cho 3 cổng: Cổng dữ liệu I/O (378h), cổng trạng thái I/O (379h) và cổng điều khiển I/O (37Ah) để điều khiển đưa ra máy in 1 (LPT1).

• Cổng điều khiển I/O (37Ah)

Bit 0: Một giá trị "0" của bit này làm cho máy in đọc dữ liệu. Trong quãng thời gian này (5 µs) dữ liệu được in ra.

Bit 1: Tiếp đường tự động (LF# - Auto line Feed) (tự động lùi trục và chuyển dòng): Bit này bằng "1" sẽ làm cho máy in đưa chuyển một dòng sau khi lùi trục (lùi đầu in).

Bit 2: Khởi động máy in (INIT# - Init printer#): một giá trị "0" của bit này làm xóa máy in (khởi động máy in).

Bit 3: Select (SEL - Lựa chọn): Bit này bằng "1" sẽ chọn máy in và cho phép chuyển dữ liệu ra máy in liên tục.

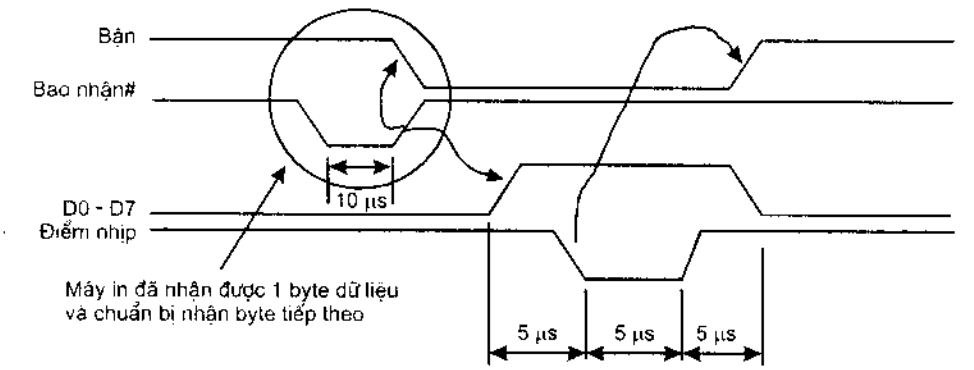
• Status port (379h)

Bit 3: Error# (E# - Lỗi): Bit này được thiết lập khi máy in không có lỗi.

Bit 4: Select Status - Trạng thái lựa chọn: Bit này được thiết lập khi máy in được chọn và sẵn sàng nhận dữ liệu đưa ra

Bit 5: Paper End (PE - Hết giấy): Thiết lập khi trên máy in không có giấy.

Bit 6: Acknowledge# (ACK# - Báo nhận): Bit này được xóa trong khoảng 10 µs khi máy in sẵn sàng nhận byte dữ liệu tiếp theo.

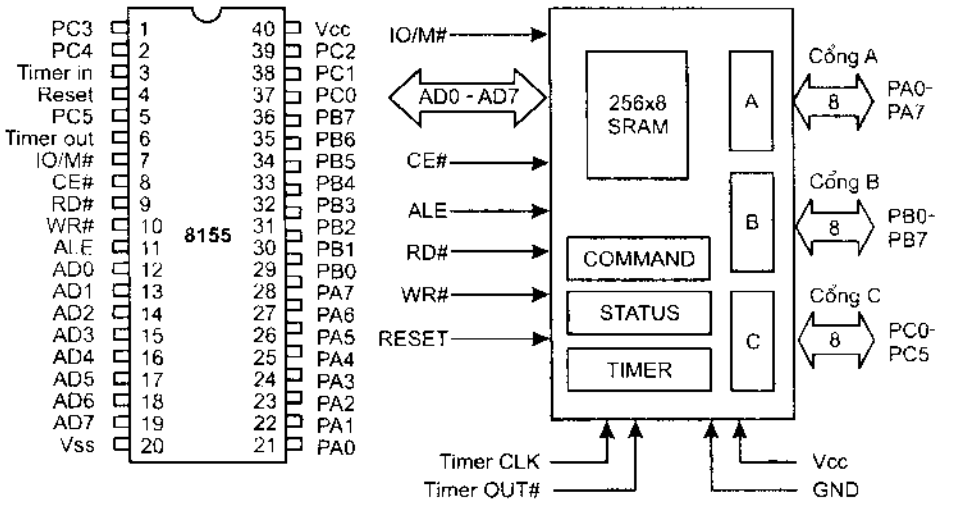


Hình 5.54: Đồ thị xung điều khiển đưa ra máy in

Bit 7: Busy end# (BUSY# - Kết thúc bận): Bit này được thiết lập khi máy in sẵn sàng nhận dữ liệu. Nếu bit này bị xoá trong quá trình máy in nhận dữ liệu thì sẽ có lỗi. Bit này được xóa khi máy in không nối với máy tính.

Quá trình đưa ra máy in ký tự được diễn giải bằng đồ thị xung hình 5.54.

Hiện nay có nhiều loại máy in tốc độ cao, như máy in laser, máy in dòng (line printer) đòi hỏi phải có các điều khiển ghép nối với CPU khác nhau, tuy nhiên về cơ bản vào/ra song song theo chuẩn centronics vẫn là giao tiếp cơ bản.



Hình 5.55: Đóng vỏ, các chân tín hiệu, và sơ đồ khối chức năng bên trong chip 8155/8156

Sơ đồ khối điều khiển vào/ra song song với máy in có thể đỡ phức tạp hơn nếu sử dụng một số mạch phụ trợ lập trình vão/ra song song, ví dụ như sử dụng mạch thích ứng ngoại vi (peripheral interface adaptor) thông dụng 8155. 8155 có bên trong: SRAM dung lượng 256 byte, 2 cổng vào/ra 8 bit (gọi là cổng A, B), 1 cổng vào/ra 6 bit (gọi là cổng C), một bộ đếm thời gian (timer), thanh ghi lệnh (command register), thanh ghi trạng thái (status register). 8155 đóng vỏ gồm dạng DIP 40 chân.

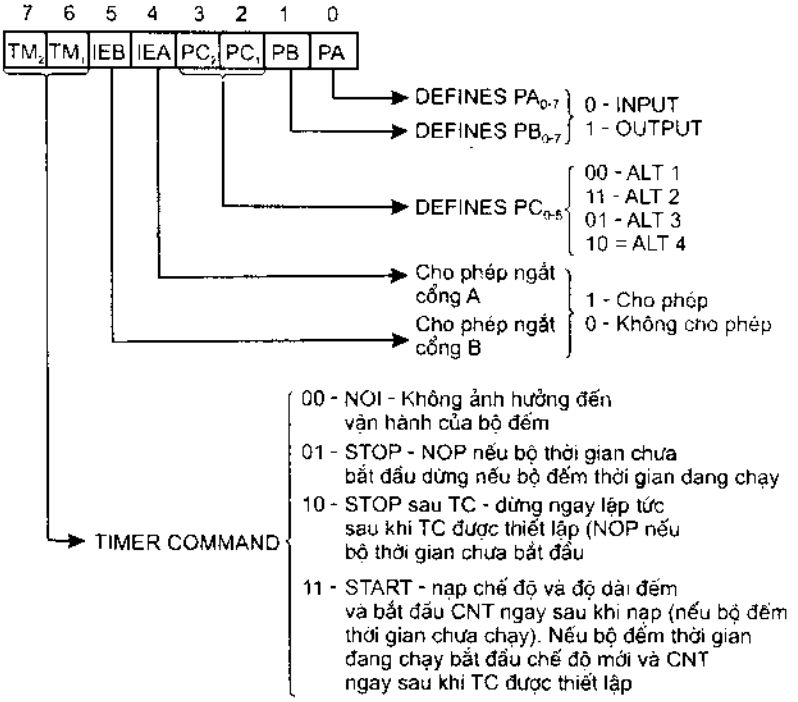
Hình 5.55 là đóng vỏ, các chân tín hiệu và cấu trúc chức năng bên trong của chip 8155. Hình 5.56 là 256B RAM, các cổng vào/ra, bộ đếm thời gian, khuôn dạng các thanh ghi lệnh và trạng thái của 8155. Bảng 5.14 liệt kê chức năng của các tín hiệu.

Bảng 5.14: Chức năng tín hiệu chip 8155

Tín hiệu	Vào/ra	Chức năng
RESET	IN	Xung xóa (tích cực "1") từ bus hệ thống (kết nối với RESET OUT của 8085A) để khởi tạo trạng thái ban đầu của chip 8155 và 3 cổng vào/ra (A,B,C). Độ rộng của xung RESET phải tối thiểu bằng 2 chu kỳ nhịp đồng hồ của 8085A.
AD0-AD7	IN	Các đường địa chỉ/dữ liệu 3 trạng thái từ bus hệ thống (từ các đường địa chỉ/dữ liệu của vi xử lý), trong đó 8 bit địa chỉ thấp (A0-A7) được chốt nhờ xung ALE từ 8 đường chung AD0-AD7.
CE#	IN	Cho phép chọn chip (Chip enable). Xung CE# tích cực "0" từ mạch giải mã chọn các chip ngoại vi (trên bảng mạch hệ thống) theo địa chỉ vào/ra gán cho chip 8155. *Chú ý:* Chip 8156 tương tự như 8155 về cấu trúc và tín hiệu, nhưng khác với 8155 là 8156 dùng CE (tích cực "1")
RD#	IN	Xung điều khiển đọc từ bus hệ thống. Khi CE# = "0", IO/M# = "0", RD# ="0" thì nội dung của SRAM trong 8155 được đọc ra AD0-AD7 Khi CE# = "0", IO/M# = "1", ED = "0" thì dữ liệu của cổng vào/ra (A, B, C) hoặc của các thanh ghi COMMAND/STATU được chọn đọc về AD0-AD7.
WR#	IN	Xung điều khiển ghi từ bus hệ thống. Khi CE# = "0", IO/M# = "0", WR# = "0" thì dữ liệu từ AD0-AD7 ghi vào SRAM trong chip 8155.
ALE	IN	Xung chốt địa chỉ từ bus hệ thống. ALE ở mức tích cực "1" là xung nhịp chốt Thanh ghi chốt 8 bit địa chỉ thấp A0-A7 cho bus địa chỉ.
IO/M#	IN	Xung điều khiển chế độ truy nhập với bộ nhớ hay thiết bị vào/ra ngoại vi. IO/M# = "1": trao đổi dữ liệu với ngoại vi. IO/M# = "0": trao đổi dữ liệu với Bộ nhớ.
PA0-PA7	IN/OUT	8 bit dữ liệu 2 chiều của cổng A. Hướng của chúng có thể lập trình được nhờ Thanh ghi COMMAND
PB0-PB7	IN/OUT	8 bit dữ liệu 2 chiều của cổng B. Hướng của chúng có thể lập trình được nhờ thanh ghi COMMAND
PC0-PC5	IN/OUT	6 bit dữ liệu 2 chiều của cổng C. Cổng C có thể được dùng như cổng vào, cổng ra hoặc cổng điều khiển cho các cổng A, B. Khi các bit của cổng C được dùng để điều khiển thì chúng có ý nghĩa như sau:

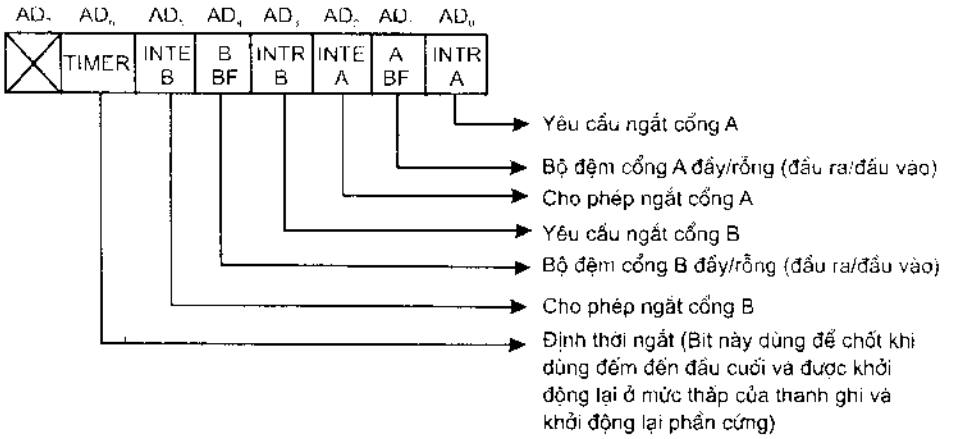
Tín hiệu	Vào/ra	Chức năng
		PC0: A INTR, yêu cầu ngắt cổng A PC1: ABF, cổng A đầy bộ đệm PC2: ASTB#, xung nhịp cho cổng A PC3: B INTR, yêu cầu ngắt cổng B PC4: BBF, cổng B bộ đệm đầy PC5: BSTB#, xung nhịp cho cổng B
Vcc	IN	+5V
Vss	IN	Ground

8155A 256 MB x 8 RAM với các cổng vào/ra và bộ định thời gian



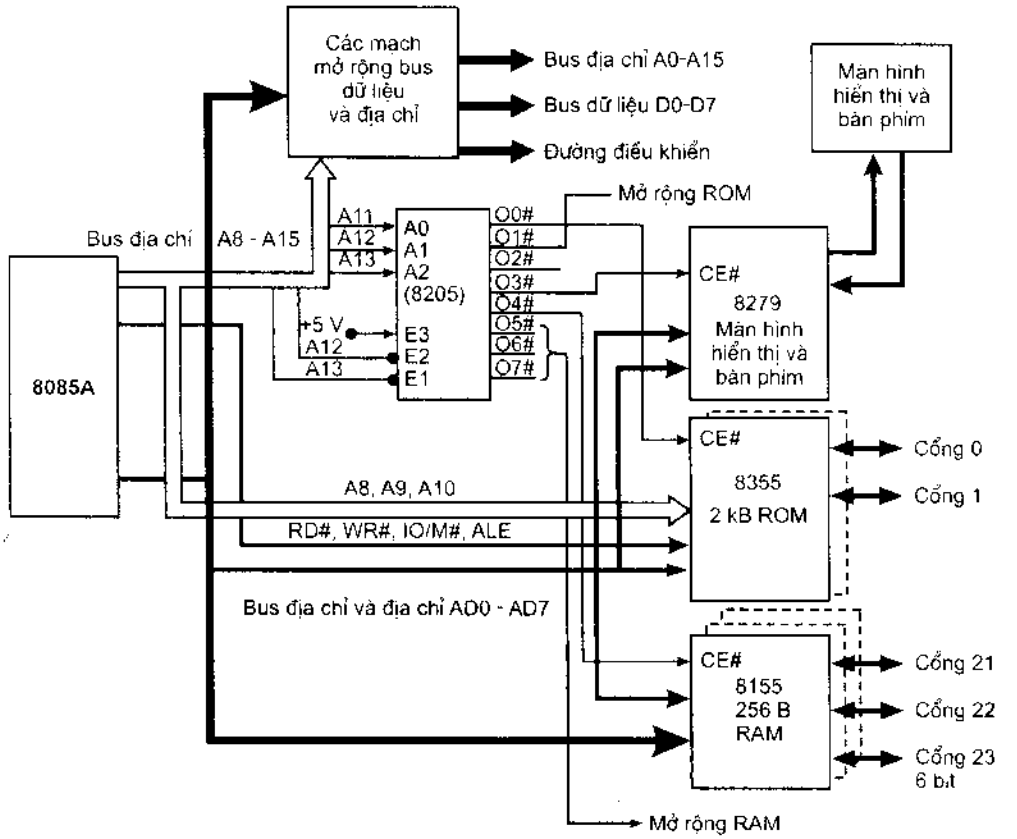
Chân	ALT 1	ALT 2	ALT 3	ALT 4
PC0	Cổng vào	Cổng ra	AINTR (Ngắt cổng A)	AINTR (Ngắt cổng A)
PC1	Cổng vào	Cổng ra	ABF (Đồng bộ đếm cổng A)	ABF (Đồng bộ đếm cổng A)
PC2	Cổng vào	Cổng ra	ASTB (Điểm nhịp cổng A)	ASTM (Điểm nhịp cổng A)
PC3	Cổng vào	Cổng ra	Cổng ra	BTNTR (Ngắt cổng B)
PC4	Cổng vào	Cổng ra	Cổng ra	BBF (Đồng bộ đếm cổng B)
PC5	Cổng vào	Công ra	Cổng ra	BSTB (Điểm nhịp cổng B)

Hình 5.56a: Định dạng thanh ghi lệnh của 8155



Hình 5.56(b): Định dạng thanh ghi lệnh của 8155

Ví dụ kết nối 8155 trong hệ vi xử lý 8085 tối thiểu (hình 5.57):



Hình 5.57: Sơ đồ khối hệ thống 8085 tối thiểu

Để mở rộng bus địa chỉ (A0-A15) sử dụng 2 vi mạch chốt một chiều 8212, mỗi vi mạch đảm nhiệm 8 bit bus địa chỉ. Để mở rộng bus dữ liệu (D0-D7) sử dụng 2 vi mạch chốt dữ liệu 2 chiều 8216, và để mở rộng bus điều khiển sử dụng 3 mạch chốt 2 chiều 8216. Các tín hiệu của bus điều khiển có thể có tới 15, bởi vì không chỉ có các tín hiệu từ 8085 sản sinh ra mà còn một số tín hiệu từ các đơn vị điều khiển I/O đưa về. Do vậy có thể cần tới 3 vi mạch đệm chốt 2 chiều 8216.

Để có thể tạo ra một hệ thống điều khiển hay một máy tính tối thiểu, cần phải có thêm các chip hỗ trợ lập trình được để thiết kế máy tính cá nhân như: điều khiển ngắt 8259A (Bộ điều khiển ngắt có thể lập trình được, 28 chân DIP), định quãng thời gian 8253 (Bộ định quãng thời gian có thể lập trình được, 24 chân DIP), bộ thích ứng ngoại vi và 256 MB RAM 8155A (Bộ thích ứng giao tiếp ngoại vi có thể lập trình, 40 chân DIP) với điều khiển ghép nối bàn phím và hiển thị 8279 (Bộ điều khiển màn hình và bàn phím) và chip ROM 8355 dung lượng 2 kB. Vi mạch 8279 điều khiển ghép nối với bàn phím và chỉ thị của bàn phím đơn giản. Chỉ thị có thể là một số các đèn LED báo hiệu giá trị địa chỉ và báo hiệu giá trị dữ liệu đang thực hiện hoặc kết quả thực hiện. Số lượng bảng phím không nhiều. Với 16 bit địa chỉ có thể dùng 4 đèn LED (7 đoạn), mỗi đèn cho một chữ số hệ Hex. Với 8 bit dữ liệu cần 2 đèn LED là đủ. Để hiển thị các giá trị nhị phân cần phải được chuyển đổi thành mã Hex. Vi mạch 8155 kết nối trên hệ thống bus của 8085 và 3 cổng vào/ra của nó A, B, C được gán các địa chỉ vào/ra tương ứng 21h, 22h, 23h. Có thể xác định cụ thể các giá trị địa chỉ vào/ra cho tất cả các thành phần của 8155 của hệ vi xử lý cho trong hình 5.57 và bảng 5.15.

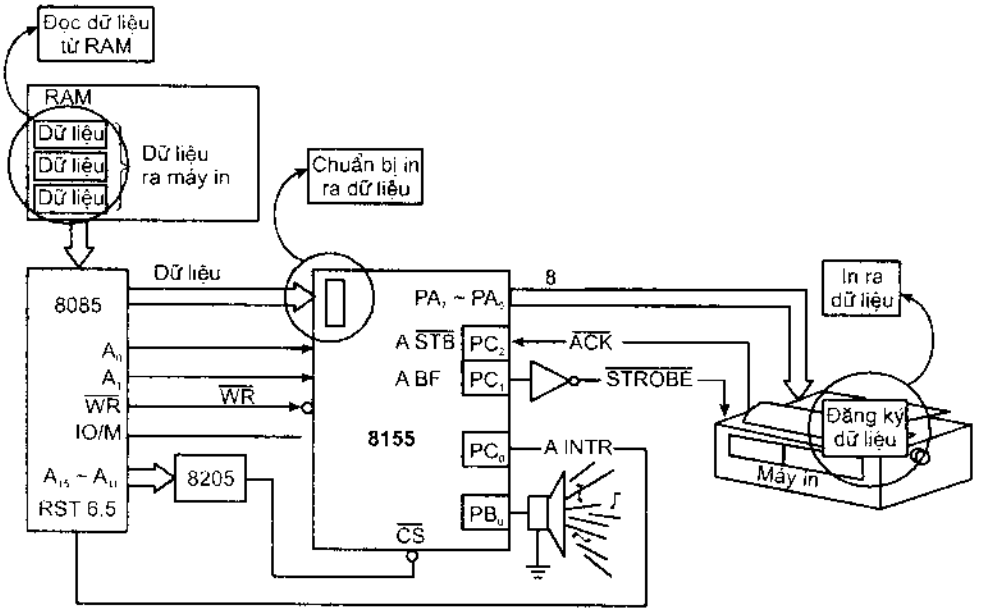
Bảng 5.15: Giá trị địa chỉ vào/ra của các thành phần của 8155

I/O ADDRESS								Chọn
A7	A6	A5	A4	A3	A2	A1	A0	
0	0	1	0	0	0	0	0	Các thanh ghi COMMAND/STATUS
0	0	1	0	0	0	0	1	Chọn cổng vào/ra A
0	0	1	0	0	0	1	0	Chọn cổng vào/ra B

I/O ADDRESS								Chọn
A7	A6	A5	A4	A3	A2	A1	A0	
0	0	1	0	0	0	1	1	Chọn cổng vào/ra C (và/ra dữ liệu hay điều khiển)
0	0	1	0	0	1	0	0	Chọn 8 bit thấp của bộ đếm thời gian TIMER
0	0	1	0	0	1	0	1	Chọn 6 bit cao của bộ đếm thời gian TIMER

Tuy vậy để chọn các cổng vào/ra, các thanh ghi COMMAND/STATUS, TIMER và SRAM của 8155 cần phải đặt tín hiệu CE# (chọn chip) của 8155 ở mức tích cực "0". Mạch giải mã 8205 (74LS138) đặt:

CE# (8155) = "0" khi: (A15 = "0")(A14 = "0")(A13 = "1") và (A12 = "0")(A11 = "0").



Hình 5.58: Sơ đồ kết nối 8155 với máy in trong hệ thống vi xử lý 8085 tối thiểu

Các tín hiệu điều khiển cho 8155: WR#, RD#, IO/M# và ALE được đưa trực tiếp từ vi xử lý 8085. Các đường dữ liệu AD0-AD7 và 16 đường địa chỉ A15-A8, AD7-AD0 cũng lấy trực tiếp từ vi xử lý

8085. Kết nối điều khiển đưa ra máy in là một trong những thí dụ thông dụng của 8155. Hình 5.58 là sơ đồ kết nối 8155 với một loại máy in có hỗ trợ nhận dữ liệu không đồng bộ theo từng byte với nhịp xung điểm nhịp.

Quá trình đưa ra máy in thực hiện theo một trình tự như sau:

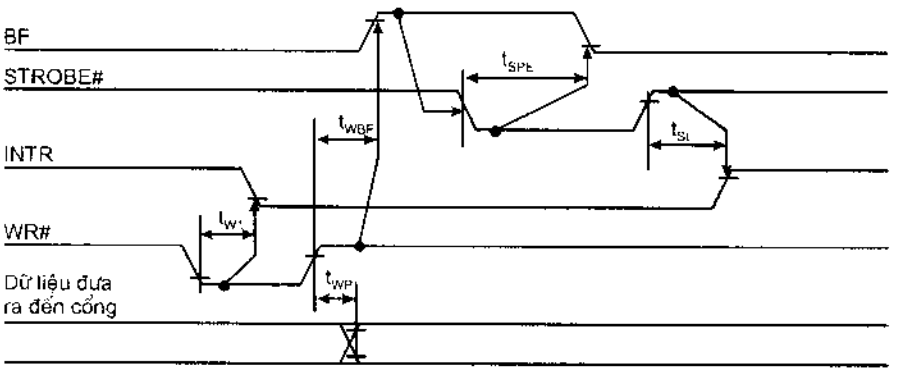
1. Máy in gửi xung tín hiệu ACK# = "0" về chân ASTB# (PC2) của cổng C của 8155, để thông báo rằng nó có đã nhận xong 1 byte dữ liệu (mã ký tự in) và sẵn sàng tiếp nhận byte dữ liệu tiếp theo.

2. Xung ACK# = "0" có 2 tác dụng: nó làm cho tín hiệu đầy bộ đệm ABF = "0" (PC1) và gây ra một yêu cầu ngắt AINTR = "1" thông qua bit PC0 đưa đến chân INTR của vi xử lý 8085.

3. CPU nhận yêu cầu ngắt này và ghi từ dữ liệu tiếp theo ra bộ đệm của 8155. Xung ghi WR# từ vi xử lý có 2 tác dụng: đưa đường tín hiệu yêu cầu ngắt AINTR trở về "0" và tín hiệu ABF chuyển lên "1". Tín hiệu ABF tạo ra STROBE# = "0".

4. Máy in đọc byte dữ liệu, tạo ra xung ACK# = "0" đưa về ASTB# (PC2) và lặp lại bước 1 của quá trình.

Định thời quá trình in trong chế độ điểm nhịp (strobe) cho trong hình 5.59.



Hình 5.59: *Đồ thị thời gian đưa ra máy in trong chế độ điểm nhịp*

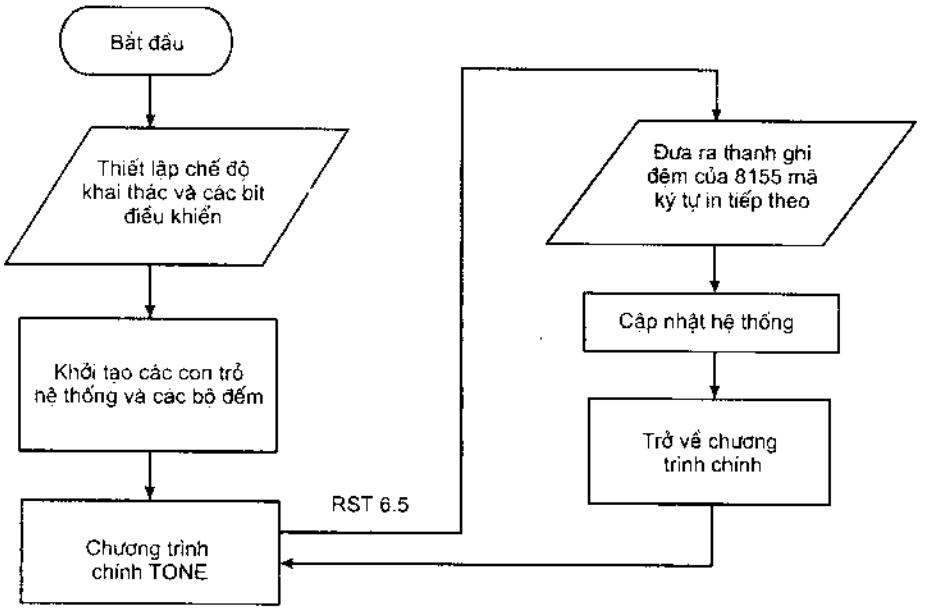
Dưới đây là chương trình mã ngữ của tập lệnh của 8085 điều khiển đưa ra máy in. Chương trình được dịch, liên kết và sắp xếp trong

bộ nhớ chính của hệ thống bắt đầu ở địa chỉ lệnh đầu tiên 2000h. Nó thực hiện đưa ra máy in một chuỗi ký tự toàn chữ 'A' và điều khiển tiếng loa. Các cổng A, B, C có các địa chỉ vào/ra tương ứng: 21h, 22h, và 23h, và địa chỉ thanh ghi COMMAND/STATUS là 20h.

```
LOC   OBJ      LINE      SOURCE STATEMENT
                 1            NAME HARDWAREHANDSHAKING
                 2    ; Đây là chương trình điều khiển vào/ra theo ngắt trong chế độ điểm
                 3    ; nhịp của 8155A có tự động xử lý bắt tay với máy in. Chương trình
                 4    ; đưa ra một chuỗi ký tự 'A', và để xác định hiệu quả của các ngắt
                 5    ; đưa ra, trong chương trình này có tạo ra tone cho loa (sử dụng PC3)
                 7    ; Chương trình viết ở mã của 8155 và được nạp vào địa chỉ 2000h
                 8    ; bộ nhớ chính (RAM) bắt đầu ở địa chỉ 2000h
                 9    ;
                10    ;       MAIN PROGRAM
                11    ;
2000             12                ORG   2000h
2000  3E15       13                MVI   A, 15h        ; Tạo mã lệnh
2002  D320       14                OUT   20h           ; Ghi ra thanh ghi COMMAND của 8155
2004  31C220     15                LXI   SP, 20C2h     ; Lập con trỏ ngăn xếp
2007  FB         16                EI                  ; Cho phép ngắt từ ngoại vi
2008  3E0D       17                MVI   A, 0Dh        ; Nạp mặt nạ ngắt
200A  30         18                SIM                 ; Bỏ mặt nạ ngắt RST 6.5
200B  0C         19          LOOP: INR   C             ; Tăng nội dung thanh ghi C của 8085
200C  79         20                MOV   A, C          ;
200D  D323       21                OUT   23h           ; Đưa ra cổng C giá trị PC1="1" để ngắt
200F  0618       22                MVI   B, 18h        ; Nạp giá trị đếm tre vào thanh ghi B
2011  05         23          TONE: DCR   B             ; Giảm giá trị đếm làm tre
2012  C21120     24                JNZ   TONE          ; Nếu <>0 thì nhắc lại
2015  C30B20     25                JMP   LOOP          ; Quay vòng
                26    ;
                27    ;   Tạo vector ngắt RST 6.5
                28    ;
20C8            29    ;         ORG   20C8h
20C8  C34020    30    ;         JMP   SEND          ; nhảy đến đoạn đưa ra chữ 'A'
                31    ;
                32    ;   Chương trình con phục vụ ngắt đưa ra máy in
                33    ;
2040            34              ORG   2040h
```

```
2040   3E41'  35      SEND:  MVI   A, 'A'    ; Nạp mã chữ A vào thanh ghi A của 8155
2042   D321   36             OUT   21h       ; Đưa ra cổng A của 8155 chữ A
2044   FB     37             EI
2045   C9     38             RET
              39             END
ASSEMBLY COMPLETE, NO ERRORS
```

Giải thuật của trình đưa ra máy in trong chế độ điểm nhịp cho trong hình 5.60.



Hình 5.60: Giải thuật trình đưa ra máy in với xử lý ngắt

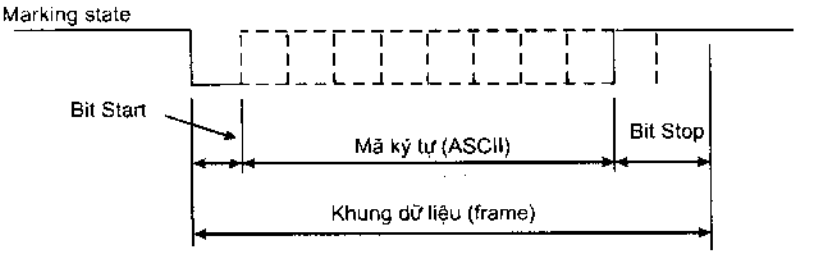
5.3.3. Vào ra nối tiếp

Trong thực tế, hai máy tính khi nằm ở khoảng cách xa sử dụng phương pháp truyền nối tiếp để trao đổi dữ liệu. Để đảm bảo cho các máy tính "nói chuyện" được với nhau qua mạng truyền thông, có cài đặt khối phối ghép truyền thông (Communication), với các cổng vào/ra nối tiếp (COM1-COM4) ngay trên bảng mạch hệ thống của những máy vi tính thế hệ X86, hoặc một bảng vào/ra truyền thông cắm trên khe mở rộng (slot) của bus hệ thống của IBM PC/XT, AT.

Có hai phương pháp truyền dữ liệu giữa các máy tính, đó là truyền dữ liệu.không đồng bộ và truyền dữ liệu đồng bộ.

Truyền dữ liệu nối tiếp không đồng bộ

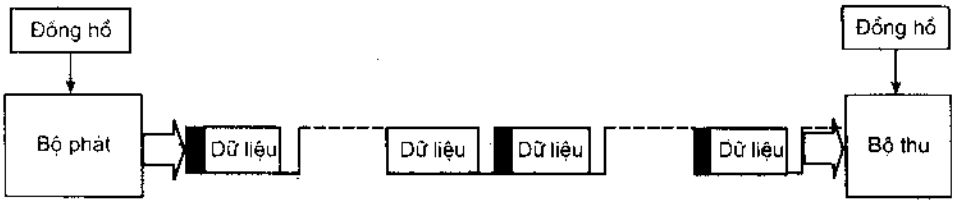
Truyền không đồng bộ là truyền bắt đầu/kết thúc, bởi vì mỗi ký tự truyền trên đường dây nối 2 đầu cuối (máy tính, mã hóa) được mã hóa và đóng khung với các bit liên tiếp nhau, bắt đầu bằng bit start và kết thúc bằng bit stop (hình 5.61).



Hình 5.61: Khuôn dạng mã ký tự truyền nối tiếp không đồng bộ

Bit Start được truyền đi đầu tiên luôn có logic 0 để cho đầu thiết bị thu chuẩn bị (phần logic điện tử) đưa vào trạng thái sẵn sàng đón nhận mã ký tự (bit Start) chính là để đồng bộ quá trình phát và nhận một mã ký tự. Tiếp sau bit Start là khung dữ liệu của mã ký tự (D0-D7). Có hai loại mã chuẩn dùng cho máy tính đó là mã ASCII và EBCDIC. Ta xét mã ASCII để minh họa cho khái niệm truyền dữ liệu không đồng bộ. Mã ASCII (American Standard Code for information Interchange - mã tiêu chuẩn của Mỹ dùng cho trao đổi thông tin) có 8 bit dữ liệu, trong đó, 7 bit được sử dụng cho mã hóa 128 kí tự khác nhau (2^7 = 128), bit thứ 8 được sử dụng cho kiểm tra chẵn lẻ. Nếu sử dụng kiểm tra chẵn (even parity) thì bit 8 luôn = 0. Khi số lượng các bit bằng 1 trong mã ký tự ASCII là giá trị chẵn, và bit 8 = 1 khi số lượng bit bằng 1 trong mã ký tự ASCII là giá trị lẻ. Bit 8 có thể được sử dụng để chỉ ra một số mục đích khác tùy thuộc loại mã sử dụng (mã telex: 5bit,...). Bit Stop có độ dài thời gian tùy chọn, phụ thuộc chủ yếu vào tốc độ truyền, độ dài này có thể là 1 hoặc 2 bit. Thường đặt chế độ

1 Bit Stop. Bit Stop cũng là để điều khiển kết thúc thời gian nhận một mã ký tự và tạo cho đầu thu chuẩn bị nhận ký tự tiếp theo. Sau Bit Stop là trạng thái đường dây nghỉ (marking state). Trạng thái này kéo dài hay ngắt phụ thuộc vào tốc độ xử lý ở đầu phát. Người lập trình thường lợi dụng thời gian này để thực hiện kiểm tra trạng thái đường dây. Khi truyền nối tiếp không đồng bộ, đầu phát và đầu thu đều có xung đồng bộ riêng (hình 5.62), không phụ thuộc vào nhau, chính vì vậy, sự có mặt của bit Start và bit Stop và thủ tục bắt tay (handshaking) là rất quan trọng.



Hình 5.62: Truyền nối tiếp không đồng bộ chuỗi các ký tự

Handshaking - là thủ tục bắt tay trong kết nối mạng máy tính theo chế độ nối tiếp không đồng bộ để đảm bảo phối ghép quá trình vào/ra dữ liệu ở cả hai đầu của mạch truyền. Ví dụ: bộ vi xử lý nhận sử dụng tín hiệu CLEAR-TO-SEND để điều khiển sao cho dòng dữ liệu không bị quá tải. Còn bộ vi xử lý phát dữ liệu thì giám sát đường dây và không gửi dữ liệu lên đường dây khi chưa được phép.

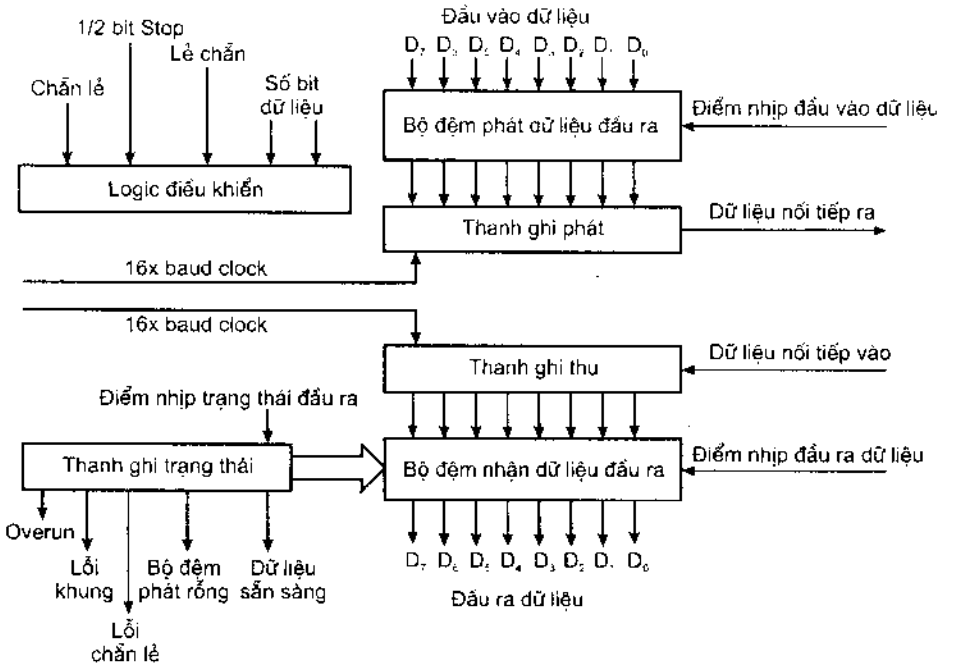
Bảng mã ASCII có hai ký tự đặc biệt đó là DC1(11h) và DC3(13h), chúng được hiểu như là điều khiển Q và điều khiển S hoặc XON và XOFF. Những tín hiệu này được sinh ra bởi hệ thống vi xử lý nhận và chúng phải được hệ thống vi xử lý phát dữ liệu công nhận.

5.3.4. UART - Mạch thu phát không đồng bộ đa năng

Có một số UART thường dùng cho đến nay được liệt kê trong bảng 5.16.

Bảng 5.16: Một số UART hiện dùng

UART	Đặc điểm
8250	Loại đầu tiên, dùng IBM PC/XT, AT, tốc độ truyền tối đa 9600 bit/s, có đệm 1 byte FIFO
8250A	Nhanh hơn 8250 ở tốc độ vận hành bus, có thêm một thanh ghi
16450	Sử dụng trong IBM PC/AT, tốc độ nhanh hơn 8250 chút ít
16550	Tương tự 8250
16550A	Có đệm 16 byte FIFO, kết nối với các mô-đem tốc độ 14,4 kbit/s và 28,8 kbit/s
16650	Có đệm 32 byte FIFO, thanh ghi lập trình các ký tự X-On/X-Off và hỗ trợ quản lý nguồn
16750	Có đệm 64 byte FIFO
16950	Có đệm 128 byte FIFO



Hình 5.63: Sơ đồ khối của mạch UART

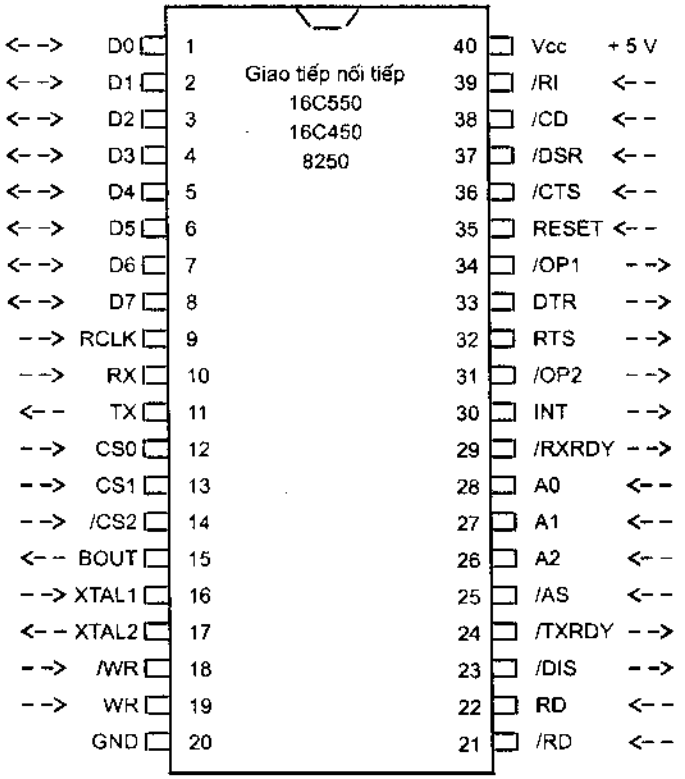
Các máy vi tính đều được trang bị đơn vị điều khiển I/O truyền thông với hạt nhân là vi mạch lập trình UART hoặc tương đương. Trong các thế hệ IBM PC/XT, AT chức năng điều khiển truyền thông được thực hiện bằng bảng mạch điều khiển vào/ra đa năng với các khối điều khiển vào/ra với các thiết bị nhớ FDD, HDD, đưa ra máy in và truyền thông (COM) nối tiếp không đồng bộ và hạt nhân là vi mạch lập trình UART 8250. Hình 5.63 là sơ đồ khối chức của mạch UART

đơn giản. UART được chia thành 4 khối chức năng: phát, thu, thanh ghi trạng thái và logic điều khiển.

Khối chức năng phát gồm: bộ đệm phát dữ liệu đầu ra (transmitted-data output buffer) để nhận dữ liệu song song 8 bit từ CPU để đưa ra thanh ghi phát (transmitted register) với khả năng nhận dữ liệu song song 8 bit và dịch phát ra dữ liệu nối tiếp.

Khối chức năng thu gồm: thanh ghi thu (receive register) với khả năng dịch nhận dữ liệu nối tiếp từ đường truyền và đầu ra dữ liệu song song 8 bit, bộ đệm nhận dữ liệu đầu ra (Received-data output buffer) với khả năng nhận dữ liệu 8 bit từ thanh ghi thu và gửi về CPU dữ liệu song song 8 bit (thông qua bus dữ liệu). Cả thanh ghi dịch nhận và phát đều được điều khiển bởi tín hiệu đồng bộ.

Đóng vỏ và các chân tín hiệu của 8250, 16450, 16550 cho trong hình 5.64.



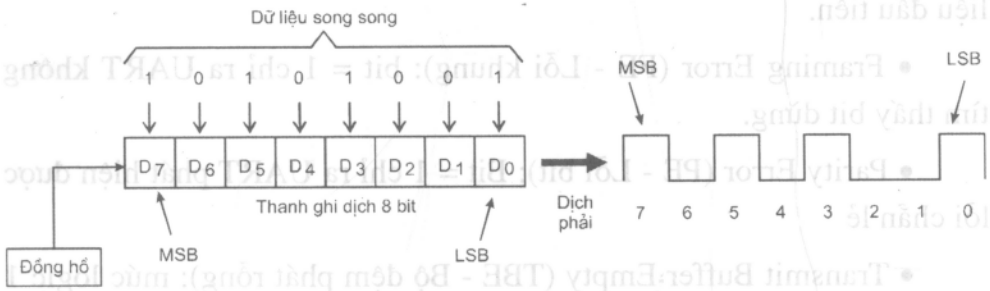
Hình 5.64: Đóng vỏ và các chân tín hiệu của 8250, 16450, 16550UART

Các tín hiệu có ý nghĩa như sau:

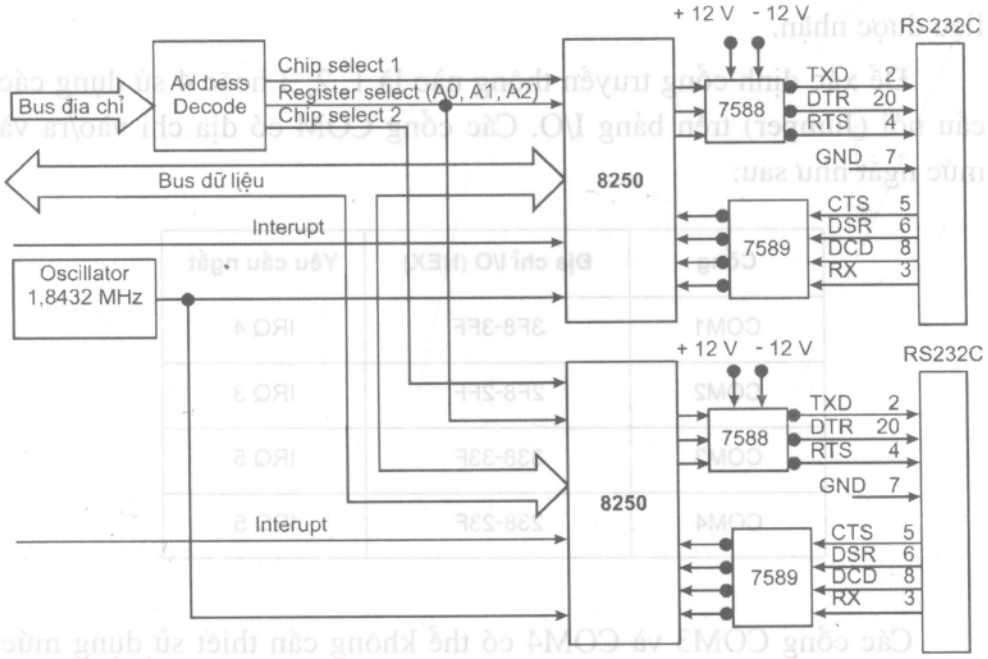
D0-D7	Bus dữ liệu vào/ra.
RCLK	Nhịp đồng hồ vào nhận (normaly connected to BOUT)
RX	Đường vào dữ liệu nối tiếp (mark = high = 1 and space = low = 0)
TX	Đường ra dữ liệu nối tiếp.
CS0	Vào chọn chip 0 (tích cực mức cao)
CS1	Vào chọn chip 1 (tích cực mức cao)
CS2	Vào chọn chip 2 (tích cực mức thấp) (CS0 = 1 and CS1 = 1 and CS2 = 0 cho phép phát dữ liệu)
BOUT	Đầu ra đồng hồ tạo tốc độ baud (16 x baudrate)
XTAL1	Đầu vào tinh thể 1 hoặc đầu vào đồng hồ mở rộng
XTAL2	Đầu vào tinh thể 2 hoặc đầu vào đồng hồ mở rộng
/WR	Nhịp ghi (Nhịp ghi tích cực mức thấp)
WR	Nhịp ghi (Nhịp ghi tích cực mức cao)
GND	Đất
/RD	Nhịp đọc (Nhịp đọc tích cực mức thấp)
RD	Nhịp đọc (Nhịp ghi tích cực mức cao)
/DDIS	Đầu ra không cho phép điều khiển (tích cực mức thấp)
/TXRDY	Đầu ra sẵn sàng phát (mức thấp tích cực) khi bộ đệm phát TX-FIFO bị đầy
/AS	Đầu ra nhịp địa chỉ (tích cực mức thấp) nếu mức thấp không sử dụng.
A2	Lựa chọn địa chỉ đầu ra đường 2, để lựa chọn thanh ghi nội bộ
A1	Lựa chọn địa chỉ đầu ra đường 1, để lựa chọn thanh ghi nội bộ
A0	Lựa chọn địa chỉ đầu ra đường 0, để lựa chọn thanh ghi nội bộ
/RXRDY	Đầu ra sẵn sàng thu (tích cực mức thấp) khi bộ đệm thu RX-FIFO bị đầy
INT	Ra yêu cầu ngắt (tích cực mức cao) được lập trình lên mức cao theo điều kiện ngắt
/OP2	Đầu ra kết quả chung 2 (tích cực mức thấp)
/RTS	Đầu ra yêu cầu gửi (tích cực mức thấp)
/DTR	Đầu ra sẵn sàng đầu cuối dữ liệu (tích cực mức thấp)
/OP1	Đầu ra kết quả chung 1 (tích cực mức thấp)
RESET	Đầu vào khởi động lại máy chủ (tích cực mức thấp)
/CTS	Xóa gửi (tích cực mức thấp)
/DSR	Đầu vào sẵn sàng thiết lập dữ liệu (tích cực mức thấp)
/CD	Đầu vào phát hiện sóng mang (tích cực mức thấp)
/RI	Vào chỉ thị chuông (tích cực mức thấp)
VCC	Nguồn nuôi +5V.

Các mạch UART có các bộ đệm FIFO (vào trước ra trước). Tất cả 8 thanh ghi của UART đều được gán địa chỉ vào/ra (I/O address) thuộc không gian địa chỉ vào/ra của máy vi tính. Bằng các lệnh máy người lập trình có thể truy nhập trực tiếp đến các thanh ghi của UART để ghi hoặc đọc nội dung của chúng. Các thanh ghi phát và thu dịch

chuyển thành dữ liệu tuần tự theo bit nhờ nhịp dịch là 16 x baud clock. Nhịp dịch này phụ thuộc vào tốc độ truyền thông tính bằng baud (bit/sec). Hình 5.65 là sơ đồ chuyển đổi dữ liệu từ song song thành nối tiếp trong các thanh ghi này.



Hình 5.65: Chuyển đổi dữ liệu song song thành nối tiếp trong UART



Hình 5.66: Sơ đồ khối khối điều khiển vào/ra truyền thông

Trong IBM PC/XT, AT, khối điều khiển truyền thông có hạt nhân là 2 chip UART 8250 40 chân (hình 5.66). Mỗi vi mạch 8250 có

8 thanh ghi được truy nhập bằng lập trình ngôn ngữ máy thông qua giải địa chỉ vào/ra: COM1: 3F8h-3FFh và COM2: 2F8h-3FFh. Thanh ghi trạng thái của UART cung cấp một số thông tin sau:

• Overrun (Vượt quá): bit = 1 cho phép từ dữ liệu vượt quá từ dữ liệu đầu tiên.

• Framing Error (FE - Lỗi khung): bit = 1 chỉ ra UART không tìm thấy bit dừng.

• Parity Error (PE - Lỗi bít): Bit = 1 chỉ ra UART phát hiện được lỗi chẵn lẻ

• Transmit Buffer Empty (TBE - Bộ đệm phát rỗng): mức logic 1 chỉ ra từ dữ liệu tiếp theo được nạp vào bộ đệm lấy dữ liệu ra.

• Data Available (DA - Sẵn có dữ liệu): logic 1 cho biết từ dữ liệu được nhận.

Để xác định cổng truyền thông nào là 1, 2, 3 hoặc 4 sử dụng các cầu nối (Jumper) trên bảng I/O. Các cổng COM có địa chỉ vào/ra và mức ngắt như sau:

Cổng	Địa chỉ I/O (HEX)	Yêu cầu ngắt
COM1	3F8-3FF	IRQ 4
COM2	2F8-2FF	IRQ 3
COM3	338-33F	IRQ 5
COM4	238-23F	IRQ 5

Các cổng COM3 và COM4 có thể không cần thiết sử dụng mức ngắt như trên. Chúng thường sử dụng một trong các mức ngắt: IRQ2, IRQ5, IRQ7 hoặc IRQ9. ROM-BIOS của IBM PC/XT, AT hỗ trợ các dịch vụ truy xuất với cổng COM (RS232) qua lệnh ngắt INT 14h. (bảng 5.17).

Bảng 5.17: Ngắt phục vụ I/O với các cổng COM (INT 14h)

AH	Các chức năng
0	Khởi tạo cổng COM:
	1. Nạp vào: AH = 0, DX = số hiệu cổng COM (0-3), AL = tham số khởi tạo, trong đó: Các bit 765 (tốc độ truyền, bit/s): 000 = 110 bit/s, 001 = 150 bit/s, 010 = 300 bit/s, 011 = 600 bit/s, 100 = 1200 bit/s, 101 = 2400 bit/s, 110 = 4800 bit/s, 111 = 9600 bit/s. Các bit 43 (chẵn lẻ): x0 = không chẵn lẻ, 01 = lẻ, 11 = chẵn Bit 2 (đối dài của bit Stop): 0 = 1 bit, 1 = 2 bit Các bit 10 (độ dài mã ký tự): 10 = 7 bit, 11 = 8 bit 2. Kết quả: AH = trạng thái của cổng COM vừa khởi tạo
1	Gửi một ký tự ra cổng COM:
	1. Nạp vào: AH = 1, DX = số hiệu COM (0-3), AL = mã ASCII của ký tự gửi ra cổng COM 2. Kết quả: AH = trạng thái lỗi, Nếu bit 7 = 1: lỗi quá thời gian (timeout), bit 7 = 0: không có lỗi
2	Nhận ký tự từ cổng COM:
	1. Nạp vào: AH = 2, DX = số hiệu cổng COM (0-3) 2. Kết quả: AL = mã ký tự nhận, AH = trang thái lỗi, bit 7 = 1: có lỗi. Bit 7 = 0: không có lỗi
3	Đọc trạng thái cổng COM:
	1. Nạp vào: AH = 3, DX = số hiệu cổng COM 2. Kết quả: AH = trạng thái cổng COM, trong đó:
	Bit 7: Lỗi quá thời gian Bit 6: thanh ghi phát rỗng Bit 5: thanh ghi nhận rỗng Bit 4: Cho phép ngắt Bit 3: Lỗi khung Bit 2: Lỗi chẵn lẻ Bit 1: Lỗi đường Bit 0: Dữ liệu đã trong bộ đệm.
	AL = Trạng thái của mô-đem (nội dung của thanh ghi trạng thái mô-đem MSR của 8250, trong đó:
	Bit 7: Phát hiện vật mang dữ liệu (DCD - Data Carrier Detect) Bit 6: Chỉ thị chuông (RI - Ringng Indicator) Bit 5: Sẵn sàng thiết lập dẽ liệu (DSR - Data Set Ready) Bit 4: Xóa gửi (CTS - Clear to Send) Bit 3: Phát hiện sóng mang dữ liệu Delta (DDCD - Data Carrier Detect) Bit 2: Chỉ thị chuông ở đuôi (TERI - Trailing Edge Ring Indicator) Bit 1: Sẵn sàng thiết lập dữ liệu Delta (DDSR - Data Set Ready) Bit 0: Xóa gửi Delta (DCTS - Delta Data Set Ready)

Bằng cách nạp mã chức năng vào AH, lệnh INT 14h sẽ thực hiện các thao tác vào/ra với các cổng COM theo mong muốn của người lập trình. Ngắt chức năng của DOS: INT 21h với mã chức năng Ah = 3 hoặc 4 cũng cho khả năng điều khiển I/O với cổng COM chuẩn (bảng 5.18). Để khởi tạo cổng COM chuẩn, sử dụng lệnh MODE của DOS.

Bảng 5.18: Ngắt chức năng DOS: INT 21h phục vụ I/O với cổng COM chuẩn

AH	Chức năng
3	Đọc mã ký tự từ cổng COM chuẩn:
	1. Nạp vào: AH = 3, 2. Kết quả: AL = mã ký tự
4	Gửi mã ký tự ra cổng COM chuẩn:
	1. Nạp vào: AH = 4, 2. DL = mã ASCII của ký tự gửi ra.

8250 có một số thanh ghi bên trong và chúng cần phải được lập trình trước khi chúng có thể được sử dụng. Các thanh ghi của 8250 như sau:

Chip 8250 sử dụng nhịp đồng hồ riêng với tần số 1,8432 MHz. Với nhịp đồng hồ này nó cho phép lập trình truyền và nhận dữ liệu với đường truyền thông tốc độ 50 baud đến 9600 baud. Chip 8250 và các thanh ghi của nó được chọn nhờ tín hiệu lựa chọn chip 1 (hoặc 2) thông qua giải mã 8 địa chỉ I/O (Address Bus) gán cho cổng COM định chọn. Các thanh ghi bên trong chip được chọn nhờ 3 bit địa chỉ của bus là A2, A1, A0 (Register select). Byte dữ liệu trao đổi giữa 8 thanh ghi của chip và CPU thông qua bus dữ liệu của PC (D0-D7) và được chuyển thành khuôn dữ liệu truyền thông liên tiếp không đồng bộ (start, stop) và nhờ các thanh ghi dữ liệu của 8250 có đơn vị chức năng dịch liên tiếp để thu hoặc phát trên đường truyền thông với tốc độ được lập trình sẵn. Vi mạch 7588 biến đổi các tín hiệu mức TTL (±5 V) thành mức ±12 V cho đường truyền. Vi mạch 7589 nhận tín hiệu từ đường truyền biến đổi về mức TTL cho 8250. phích cắm COM có 2 loại: DB9 (9 chân) và DB25 (25 chân) theo chuẩn RS-232C.

Các thanh của 8250 đều là 8 bit và có thể được truy nhập trực tiếp thông qua các lệnh cổng IN và cổng OUT, trong đó cổng là địa chỉ vào/ra của thanh ghi.

• *3F8 (DR): Thanh ghi dữ liệu (Data Register)*

Chứa dữ liệu khi cần đưa ra cổng COM, hay dữ liệu khi nhận từ cổng COM. Khi dùng lệnh IN hoặc OUT ta có thể đọc hoặc gửi dữ liệu từ thanh ghi này, ví dụ:

```
MOV DX, 3F8 ; đặt DX địa chỉ thanh ghi dữ liệu của COM1
IN DX       ; nội dung của thanh ghi dữ liệu được ghi vào AL
AND 0Fh     ; nội dung AL nhân lôgic với 0Fh (loại 4 bit cao
              của giá trị AL)
OUT DX      ; kết quả AL đưa thanh ghi dữ liệu 3F8h của
              COM1
```

Thanh ghi này cũng là byte thấp của biến chia đồng hồ, byte cao của biến chia là nội dung của thanh ghi 3F9, tạo nên giá trị biến 16 bit định ra tốc độ baud (tốc độ truyền dữ liệu - tính bit/s). Tuy nhiên, 3F8 và 3F9 chỉ được hiểu là biến chia tốc độ khi trước đó có lệnh đặt tốc độ: OUT 3FBh, 80h. (tạo bit DLAB = 1 của thanh ghi 3FBh).

• *3F9 (IER): Thanh ghi cho phép ngắt (Interrupt Enable Register)*

Byte cao của biến chia định tốc độ baud khi bit DLAB = 1 của 3FBh, trong đó:

Bit 7-4: = 0

Bit 3: = 1, cho phép ngắt theo tình trạng mô-đem (CTS, DSR, RI, RLSD)

Bit 2: = 1, cho phép ngắt theo tình trạng của đường dây (lỗi đường dây)

Bit 1: = 1, cho phép ngắt khi bộ đệm phát dữ liệu rỗng

Bit 0: = 1, cho phép ngắt khi bộ đệm nhận dữ liệu đang có dữ liệu

• *3FA (IR): Thanh ghi ngắt (Interrupt Register)*

Bit 0: = 1, không ghi nhận ngắt (có thể dùng để thăm dò)

Bit 2-1: = 00, ngắt do lỗi đường dây (lỗi chẵn lẻ, lỗi khung,...)

= 01, ngắt khi đã có dữ liệu sẵn sàng trong bộ đệm dữ liệu nhận

= 10, ngắt khi bộ đệm dữ liệu phát rỗng

= 11, ngắt khi có lỗi mô-đem (CTS, DSR, RI, RLSD - Record Line Signal Setect: Phát tín hiệu dòng ghi)

• *3FB (LSR): Thanh ghi điều khiển đường dây (Line Control Register)*

Bit 1-0 : chiều dài mã ký tự: 00 = 5 bit, 01 = 6 bit, 10 = 7 bit, 11 = 8 bit

Bit 2: độ dài bit Stop: 0 = 1 bit Stop, 1 = 2 bit Stop

Bit 4-3 : kiểm tra chẵn lẻ: x0 = không kiểm tra chẵn lẻ, 01 = lẻ, 11 = chẵn

Bit 5 := 1, lỗi chẵn lẻ

Bit 6 := 1, cho phép kiểm tra dừng, bắt đầu gửi các số 0 (khoảng trống)

Bit 7 : BLAB (Divisor Latch Access Bit - Bit truy nhập chốt số chia)

= 0, không có nghĩa

= 1, thực hiện nạp số chia tốc độ vào 3F8, 3F9

• *3FC (MCR): Thanh ghi điều khiển mô-đem (Modem Control Register)*

Bit 0:= 1, thiết lập tín hiệu DTR (Data terminal Ready - Sẵn sàng đầu cuối số liệu)

= 0, xóa DTR

Bit 1: = 1, thiết lập tín hiệu RTS (Request to Send - Yêu cầu gửi)

= 0, bỏ thiết lập RTS

Bit 2: = 1, thiết lập tín hiệu OUT1 (dành cho người thiết kế)

= 0, bỏ thiết lập OUT1

Bit 3: = 1, thiết lập tín hiệu OUT2 (dành cho người thiết kế)

= 0, bỏ thiết lập OUT2

Bit 4: = 1, thiết lập quay vòng thanh ghi dữ liệu (không ra COM, để kiểm tra)

• *3FD (LSR): Thanh ghi trạng thái đường dây (Line Status Register)*

Bit 0: = 1, (DR): đã có dữ liệu nhận trong thanh ghi dữ liệu

Bit 1: = 1, (OE - Overrun Error): lỗi chạy trước, ký tự trước bị mất

Bit 2: = 1, (PE - Parity Error): lỗi chẵn lẻ

Bit 3: = 1, (FE - Frame Eerror): lỗi khung dữ liệu

Bit 4: = 1, (BE - Break Indicated): đã có đứt đoạn đường dây (có khoảng trống)

Các bit 0-4 có thể gây ngắt (ngắt vì có lỗi đường dây)

Bit 5: = 1, thanh ghi phát dữ liệu rỗng

Bit 6: = 1, không có dữ liệu để phát

Bit 7: = 0

• *3FC (MSR): thanh ghi trạng thái mô-đem (Modem Status Register)*

Bit 0: = 1, xóa gửi Delta (DCTS - Delta Clear to Send): vừa đổi trạng thái

Bit 1: = 1, sẵn sàng thiết lập dữ liệu Delta (DDSR - Delta Data Set Ready): vừa đổi trạng thái

Bit 2: = 1, chỉ thị chuông biên đuôi (TERI - Trailing Edge Ring Indicator): đang tích cực

Bit 3: = 1, phát hiện sóng mang dữ liệu Delta (DDCD - Delta Data Carrier Detect): vừa đổi trạng thái

Bit 4: = 1, Xóa gửi (CTS - Clear to Send): đang tích cực

Bit 5: = 1, Sẵn sàng thiết lập dữ liệu (DSR - Data Set Ready): đang tích cực

Bit 6: = 1, Chỉ thị chuông (RI - Ring Indicator): đang tích cực

Bit 7: = 1, Phát hiện sóng mang dữ liệu (DCD - Data Carrier Detect): đang tích cực.

Trước khi thực hiện kết nối truyền thông qua mạng điện thoại công cộng giữa hai đầu cuối là hai máy tính, ta phải khởi tạo cổng COM cho cả hai máy tính ở hai đầu với các thông số giống nhau: tốc độ (baud), kiểm tra chẵn lẻ, độ dài mã ký tự (7 - 8 bit). Để khởi tạo cổng, có thể sử dụng INT 14h, hay trực tiếp can thiệp vào các thanh ghi tương ứng của UART đã nêu trên bằng các lệnh IN hoặc OUT của mã hợp ngữ. Các dịch vụ với cổng COM có thể thực hiện như sau:

INT 14h:

- *AH = 00, khởi tạo cổng COM:*

  DX = địa chỉ I/O của cổng - (tương ứng cho COM1, COM2, COM3, COM4)

  AL = tham số khởi tạo:

  Bit 765 = tốc độ (bit/s), 000 = tốc độ 110 bit/s, 001 = 150 bit/s, 010 = 300 bit/s, 011 = 600 bit/s, 100 = 1200 bit/s, 101 = 2400 bit/s, 110 = 4800 bit/s,111 = 9600 bit/s

  Trả về: AH = trạng thái của đường dây

- *AH = 01, phát một ký tự ra cổng COM (RS232)*

  DX = địa chỉ vào/ra của cổng

  AL = mã ký tự để phát (mã ASCII)

  Trả về: AL = giữ nguyên mã ký tự

  AH = Bit 6-0: trạng thái đường dây, bit 7 = 1, có lỗi

- *AH = 02, nhận một ký tự từ cổng COM*

  DX = địa chỉ I/O của cổng

  Trả về: AL = mã ASCII của ký tự nhận từ COM

  AH, khác 0, nếu có lỗi

- *AH = 03, nhận giá trị trạng thái của cổng COM (đường dây)*

DX = địa chỉ I/O của cổng

Trả về: AX = trạng thái của cổng, trong đó AH:

Bit 7 = 1, lỗi thời gian

Bit 6 = 1, thanh ghi dịch chuyển đổi rỗng

Bit 5 = 1, thanh ghi dữ liệu phát rỗng (TE)

Bit 4 = 1, chỉ thị dừng (BI)

Bit 3 = 1, lỗi khung (FE)

Bit 2 = 1, lỗi chẵn lẻ (PE)

Bit 1 = 1, lỗi chạy trước (OE-Overrun)

Bit 0 = 1, dữ liệu đã sẵn sàng trong thanh ghi dữ liệu nhận (DR)

AL:

Bit 7 = 1, DCD

Bit 6 = 1, RI

Bit 5 = 1, DSR

Bit 4 = 1, CTS

Bit 3 = 1, DDCD

Bit 2 = 1, TERI

Bit 1 = 1, DDSR

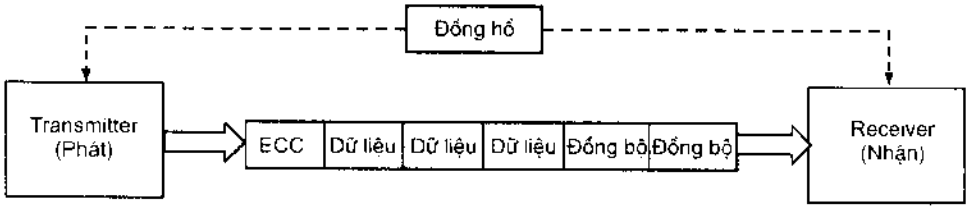
Bit 0 = 1, DCTS

Truyền nối tiếp dữ liệu đồng bộ

Truyền dữ liệu đồng bộ phân chia theo hai loại đồng bộ: đồng bộ theo bit (bit synchronous) và đồng bộ theo ký tự (character synchronous).

Tín hiệu đồng bộ theo từng bit (đồng bộ theo bit), tín hiệu đồng bộ theo ký tự (đồng bộ theo ký tự) được nguồn phát kèm dữ liệu lên đường dây và nơi nhận dựa vào tín hiệu đồng bộ này để xác định thời

điểm tiếp nhận từng dữ liệu. Hình 5.67 là một ví dụ hình thức truyền đồng bộ theo ký tự, trong đó tín hiệu đồng bộ phát kèm với dữ liệu trên cùng một đường dây (đây là phương thức phổ biến). Tuy nhiên, cũng có thể dùng một đường dây đồng bộ riêng với nguồn đồng hồ ngoài chung cho cả thu và phát (đường nét đứt), phương pháp này ít được sử dụng trong kết nối mạng máy tính vì tốn kém và bản thân những thiết bị truyền dẫn phục vụ kết nối mạng máy tính hiện nay là những thiết bị thông minh với các bộ vi xử lý có thể đảm bảo đồng bộ tốt hơn.



Hình 5.67: Truyền nối tiếp dữ liệu đồng bộ theo ký tự

Kỹ thuật truyền dữ liệu đồng bộ cho phép đóng gói dữ liệu khi truyền trên mạng, điều này nảy sinh kỹ thuật chuyển mạch gói thông tin. Các mạng máy tính ứng dụng các kỹ thuật chuyển mạch gói cho phép kết nối nhiều máy tính, nhiều mạng LAN trên diện rộng với các giao tiếp truyền dữ liệu đồng bộ và chuyển mạch gói và định tuyến các gói trong mạng. Các giao tiếp loại này có kết cấu phức tạp và phụ thuộc vào giao thức kết nối mạng máy tính nên chúng không cấy ghép trên bảng mạch hệ thống của các máy vi tính mà thường được thiết kế trên thiết bị hoặc bảng mạch riêng cắm trên khe mở rộng của máy vi tính.

Có một vài giao thức truyền dữ liệu đồng bộ đã được sử dụng cho máy chủ IBM được gọi là điều khiển kết nối dữ liệu đồng bộ SDLC (Synchronous Data Link Control). Giao thức đồng bộ SDLC ra đời sớm hơn là phương pháp truyền dữ liệu đồng bộ theo bit BSC (Bit SynChronous). SDLC và BSC thường được sử dụng khi thực hiện kết nối mạng các máy tính nhỏ hoặc máy tính lớn.

Giao thức điều khiển liên kết dữ liệu mức cao (HDLC - High Level Data-Link Control) và giao thức X.25 của mạng chuyển mạch gói là các chuẩn do ISO thiết lập, chúng là những hệ thống đồng bộ độc lập với dữ liệu, do đó chúng có thể thực hiện gửi dữ liệu trên bất kỳ khoảng cách nào. HDLC và X.25 phần lớn được sử dụng để kết nối các LAN và các hệ thống truyền thông vệ tinh.

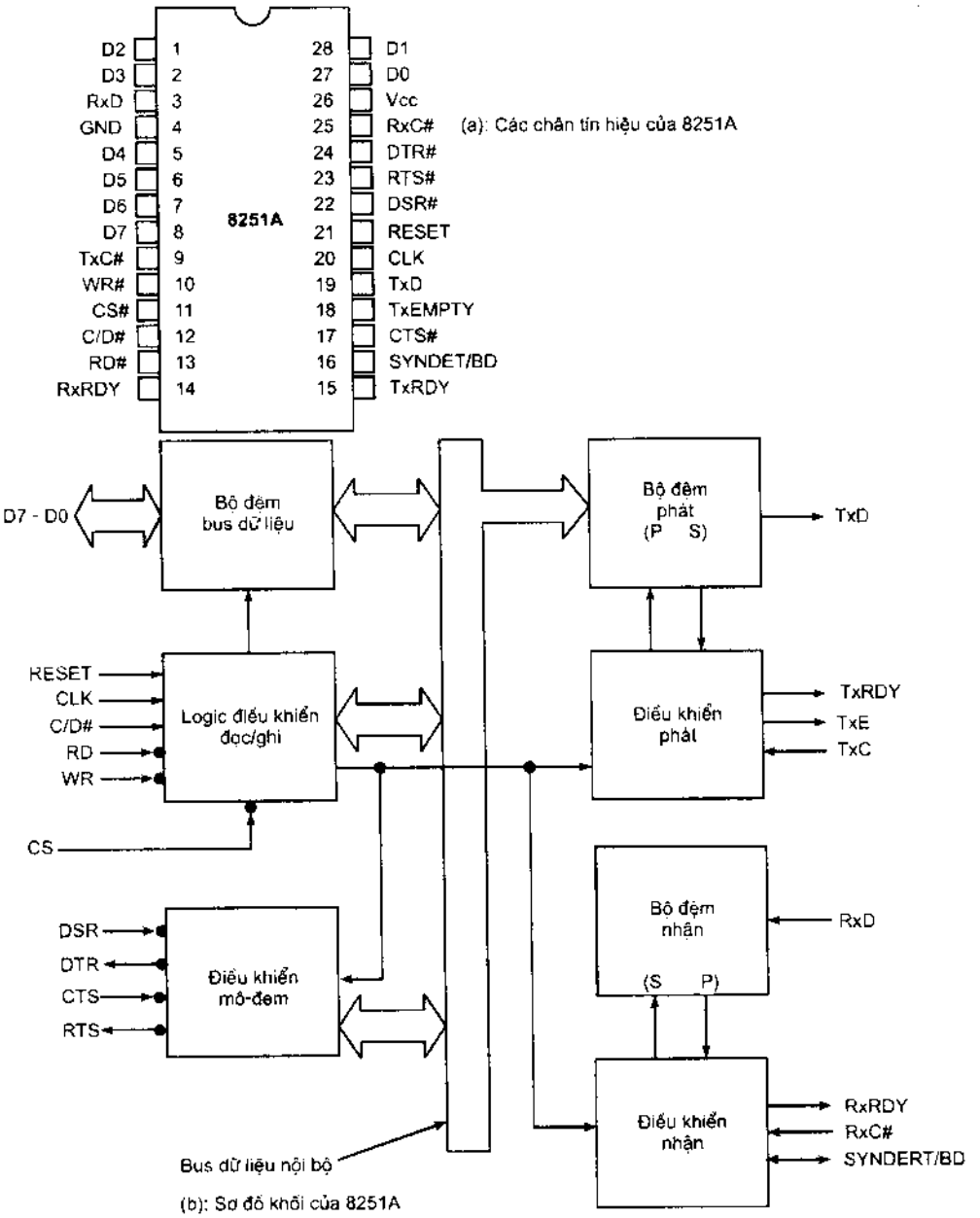
Trong tầng liên kết dữ liệu (Data link) của mô hình tham chiếu các hệ thống mở (OSI model), liên kết dữ liệu làm nhiệm vụ đảm bảo đóng và gỡ gói dữ liệu (framing), đồng bộ (synchronouns) và phát hiện lỗi dữ liệu (error detection) truyền trên môi trường vật lý. Đơn vị dữ liệu xử lý ở lớp này là khung. Một khung thể hiện một cấu trúc dữ liệu. Dữ liệu truyền đi được kiểm tra theo tổng kiểm tra (CRC). Lớp này có một số giao thức:

- ISO Basic Mode hay Truyền thông đồng bộ nhị phân (BSC - Binary Synchronous communication).
- Điều khiển liên kết dữ liệu mức cao (HDLC - ISO High - level Data Link Control).
- Thủ tục cân bằng truy nhập liên kết của CCITT (LAPB Link Access Procedure Balanced) của X.25.
- Điều khiển kết nối dữ liệu đồng bộ của IBM (SDLC IBM-Synchronouns Data Link Control).

Khi đó, lớp mạng của mô hình OSI thực hiện kết nối hai hệ thống qua mạng truyền thông bằng truyền các gói dữ liệu. Đơn vị dữ liệu ở lớp này là gói tin (Datagram). Lớp này đảm bảo sự định tuyến và chuyển tiếp dữ liệu bằng các mạch kết nối ảo và có sự kiểm soát nghẽn lưu lượng. Đối với các kết nối mạng cục bộ (LAN) các máy tính trong một không gian nhỏ, tổ chức IEEE ở Mỹ cũng đã phát triển các chuẩn 802 cho cấu trúc mạng LAN. Các chuẩn này liên quan đến hai nút dưới cùng của mô hình OSI. Kèm theo các chuẩn này có định nghĩa dịch vụ mới là MAC (Media Access Control) (lớp liên kết dữ liệu), phù hợp cho công nghệ LAN. Điều quan trọng trong mô hình tham chiếu mọi tầng trên đều trong suốt đối với tầng dưới, nói cách khác, nếu ta có kết nối truyền dữ liệu giữa hai máy tính thì trước hết phải

đảm bảo tầng thấp nhất - tầng vật lý phải có liên kết cùng một loại tín hiệu điện ở cả hai đầu cuối.

5.3.5. USART- Mạch thu phát đồng bộ/không đồng bộ đa năng



Hình 5.68: Sơ đồ khối chức năngvà đóng vỏ của USART 8251A

Mạch USART (Universal Synchronous/Asynchronous Reveiver/ Transmitter) lập trình được để ghép nối nối với hệ thống vi xử lý điều khiển vào/ra truyền thông đồng bộ và không đồng bộ. Chip Intel 8251 là sự nâng cấp của chip 8250 bằng bổ sung chức năng truyền dữ liệu đồng bộ. Hình 5.68 là sơ đồ chân tín hiệu và sơ dồ khối chức năng của 8251.

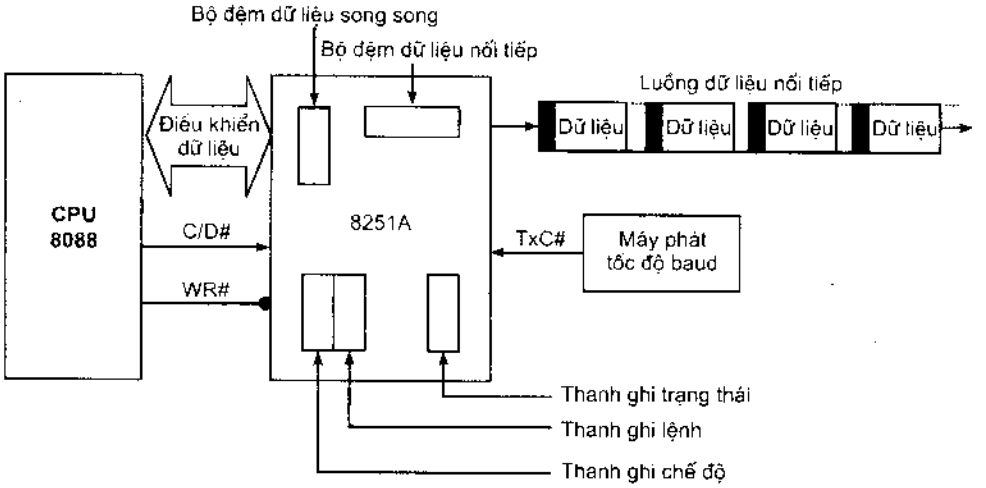
Khi dùng 8251 như là UART (truyền dữ liệu không đồng bộ) trong hệ thống vi xử lý thì lập trình điều khiển 8251 tương tự như 8250. Khi dùng 8251 cho truyền dữ liệu đồng bộ thì khi đó 8251 là thành phần chủ yếu của bảng mạch điều khiển vào ra đồng bộ cho giao tiếp kết nối mạng nào đó, ví dụ, bảng mạch kết nối LAN (NIC), hoặc giao tiếp chuyển mạch gói X.25, v.v... Khi đó lập trình cho 8251 phức tạp hơn. Các tín hiệu của 8251 có ý nghĩa cụ thể trong bảng 5.19.

Bảng 5.19: Các tín hiệu của 8251

D7-D0	Bus dữ liệu 8 bit	RxC#	Nhịp nhận dữ liệu từ đường truyền
C/D#	Điều khiển/dữ liệu để ghi hoặc đọc	RxRDY	Dữ liệu nhận đã sẵn sàng
RD#	Xung đọc dữ liệu	TxRDY	Dữ liệu phát đã sẵn sang
WR#	Xung ghi dữ liệu	DSR#	Sẵn sàng thiết lập dữ liệu
CS#	Chọn chip	DTR#	Sẵn sàng đầu cuối dữ liệu
CLK	Xung nhịp đồng hồ hệ thống	SYNDET/BD	Phát hiện đồng bộ/phát hiện đường
RESET	Xóa	RTS#	Yêu cầu gửi dữ liệu
TxC#	Nhịp phát dữ liệu ra đường truyền	CTS#	Xóa gửi dữ liệu
TxD	Dữ liệu phát	TxE	Phát rỗng
Vcc	Nguồn nuôi +5V	GND	Đất

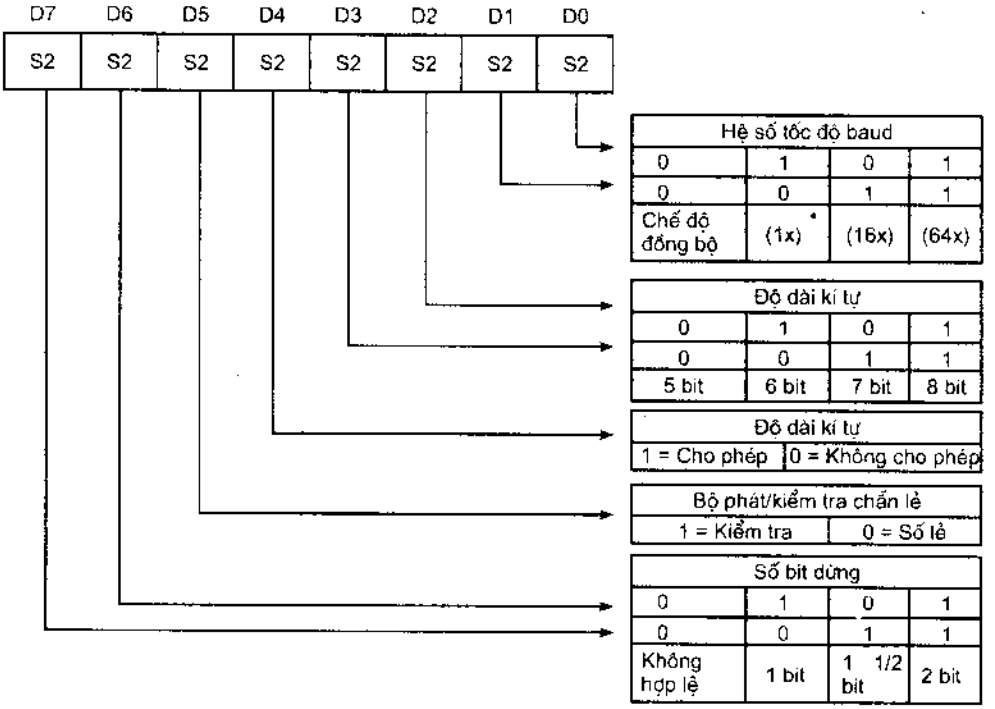
Hình 5.69 là sơ đối khối kết nối truyền thông cơ bản sử dụng USART 8251 trong hệ vi xử lý 8086/8088. Nó cũng tương tự như kết nối truyền thông trong máy vi tính dùng UART 8250.

Để lập trình điều khiển USART 8251A chúng ta phải thiết lập các chế độ khai thác cho 8251A:



Hình 5.69: Sơ đồ kết nối truyền thông của hệ thống vi xử lý 8088 bằng USART 8251

- *Khuôn dạng từ chế độ không đồng bộ của 8251A:*



Hình 5.70: Khuôn dạng chế độ đồng bộ của 8251

Thanh ghi chế độ (mode register) chứa từ chế độ 8-bit với ý nghĩa của từng bit mô tả trong hình 5.70. Ví dụ, nếu đưa ra từ thiết lập chế độ cho 8251A có nội dung là: 01111101 thì có nghĩa là ta thiết lập:

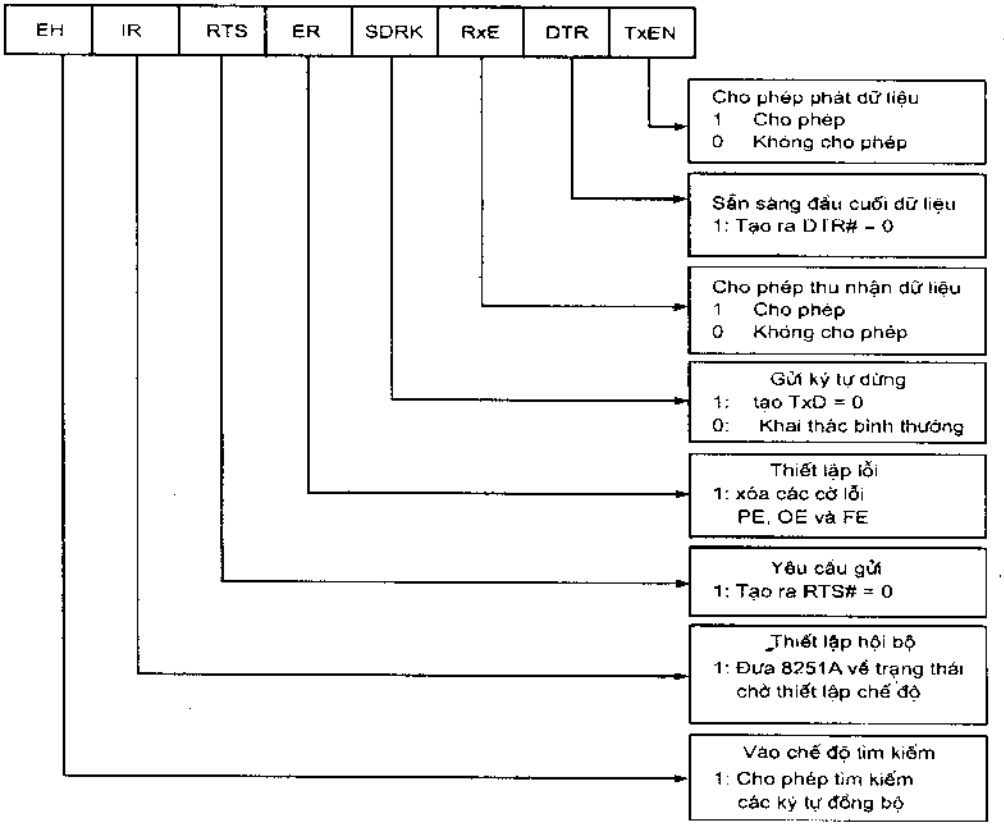
- Tốc độ baud bằng tốc độ nhịp đồng hồ (1x),
- Dữ liệu được mã hóa 8-bit, và có chẵn lẻ chẵn
- Có 1 bit Stop.

Nếu ta đưa ra từ lệnh cho 8251A có nội dung là: XXXXX01X thì có nghĩa là:

- Cho phép phát dữ liệu (enable transmit)
- Cấm thu nhận dữ liệu (disable receive).

• *Khuôn dạng từ lệnh:*

Thanh ghi lệnh 8-bit mô tả chứa từ lệnh với ý nghĩa của từng bit cho trong hình 5.71.



Hình 5.71: Khuôn dạng từ lệnh của USART 8251

• *Từ trạng thái (thanh ghi trạng thái):*

DSR	SYNDET	FE	OE	PE	TxEMPTY	RxRDY	TxRDY

Hình 5.72: Từ trạng thái của USART 8251

Trong quá trình điều khiển vào/ra truyền thông, chúng ta cần phải kiểm soát trạng thái của đường truyền thông qua trạng thái từ của 8251A. Các bit của từ trạng thái có ý nghĩa như sau:

- Bit 0: TxRDY: Sẵn sàng phát dữ liệu

- Bit 1: RxRDY: Sẵn sàng nhận dữ liệu

- Bit 2: TxEMPTY: Bộ đệm phát dữ liệu đã rỗng

- Bit 3: PE: Lỗi chẵn lẻ

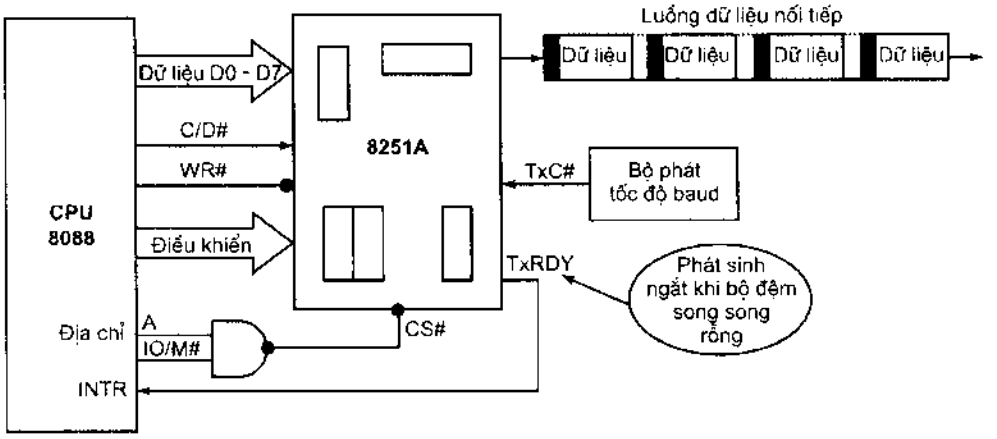
- Bit 4: OE: Lỗi chạy trước (Overun Error): CPU chưa kịp đọc ký tự đã nhận trước, trong khi đó ký tự tiếp theo đã được nhận vào).

- Bit 5: FE Lỗi khung dữ liệu (Framing Error): khi không có bit Stop.

- Bit 6: SYNDET: Xác định tín hiệu đồng bộ

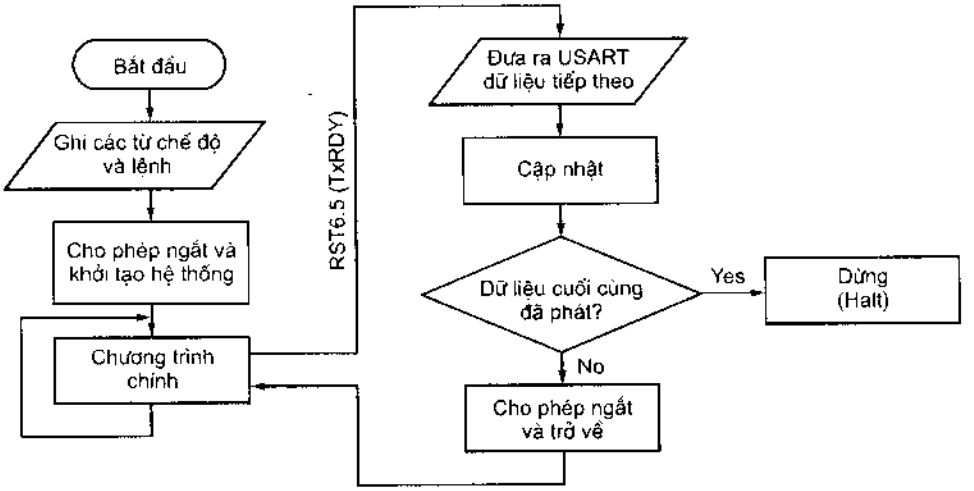
- Bit 7: DSR: Sẵn sàng thiết lập dữ liệu (Data Set Ready).

Các bit D6, D2 và D1 có giá trị tương ứng với các chân tín hiệu cùng tên của 8251A.



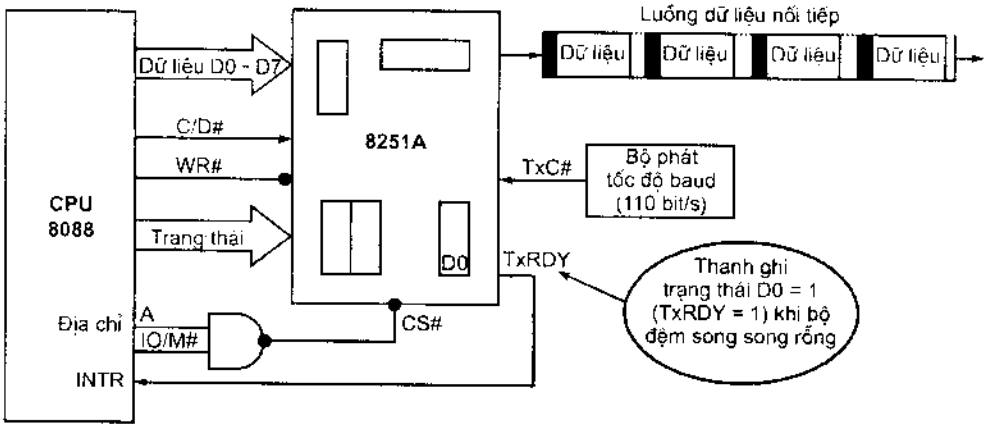
Hình 5.73: Phát không đồng bộ bằng phương pháp ngắt nhờ USART 8251

Để truyền nối tiếp không đồng bộ với mạch USART 8251, cũng như UART 8250 có thể sử dụng các chế độ điều khiển theo ngắt hay thăm dò (polling). Cả hai trường hợp đều cần có sự khởi tạo ban đầu các từ chế độ và từ lệnh bằng lệnh ghi từ CPU ra 8251. Đó là khởi tạo tốc độ truyền, khuôn dạng mã ký tự: độ dài mã ký tự, độ dài bit stop, bit parity; chế độ truyền (ngắt hay chọn). Trong điều khiển phát bằng phương pháp ngắt, một ngắt được tự động sinh ra nhờ tín hiệu sẵn sàng phát TxRDY (transmit ready) (hình 5.73), và CPU ghi một mã ký tự tiếp theo vào bộ đệm song song đang rỗng trong USART. Lưu đồ thao tác điều khiển phát không đồng bộ theo ngắt mô tả trong hình 5.74.

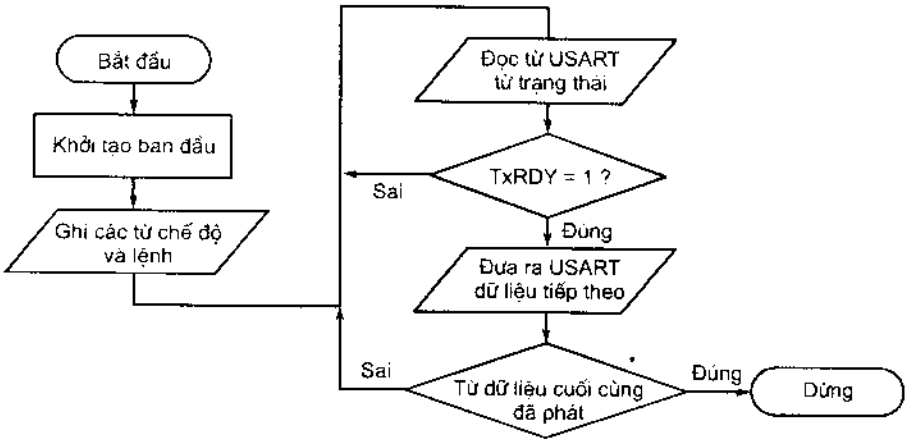


Hình 5.74: Phát nối tiếp không đồng bộ bằng phương pháp ngắt nhờ USART

Trong chế độ thăm dò nhờ USART 8251, CPU liên tục đọc trạng thái của TxRDY để xét điều kiện "bộ đệm song song rỗng". Bit D0 = 1 (TxRDY) của thanh ghi trạng thái của 8251 khi bộ đệm song song rỗng và 8251 sẵn sàng cho phát dữ liệu. Sơ đồ phối ghép 8251 với CPU và lưu đồ điều khiển chọn để phát dữ liệu nối tiếp không đồng bộ cho ở hình 5.75 và hình 5.76.



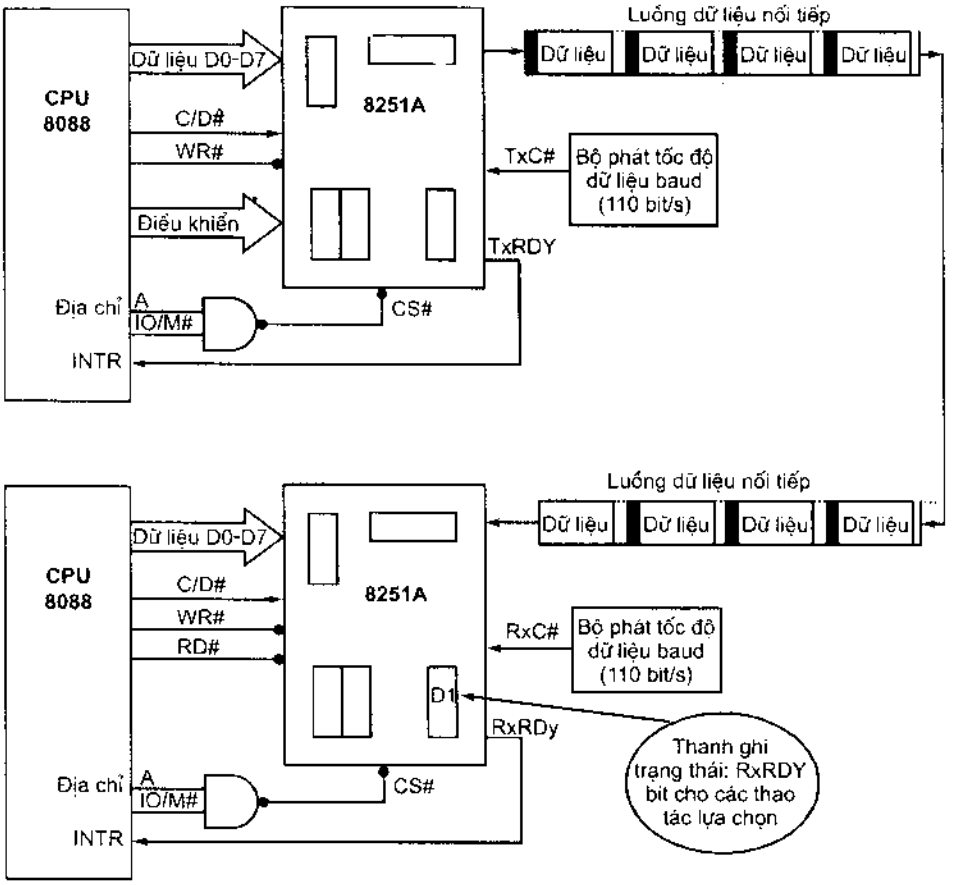
Hình 5.75: Phát không đồng bộ theo phương pháp thăm dò



Hình 5.76: Phát nối tiếp không đồng bộ bằng phương pháp thăm dò nhờ USART

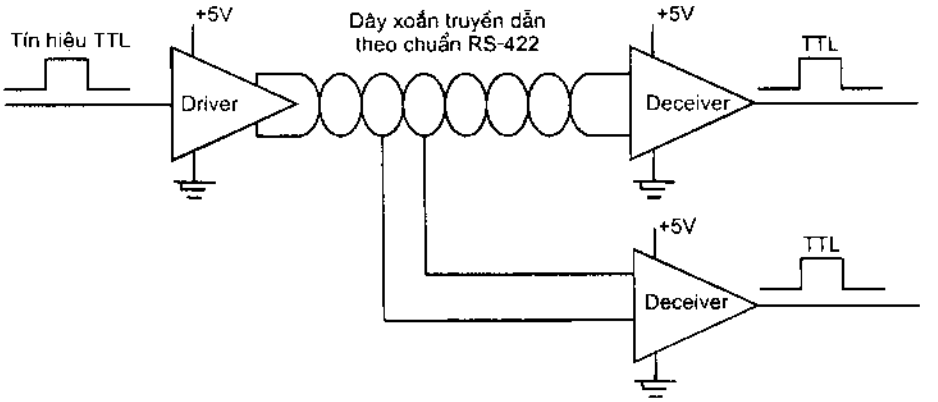
Khi kết nối 2 hệ thống đầu cuối nhờ USART 8251 (hình 5.77), tại phía thu, dữ liệu bit D1 của thanh ghi trạng thái của 8251 được kiểm soát. Nếu D1 = 1, tức là tín hiệu RxRDY = 1, báo rằng 8251 sẵn sàng nhận dữ liệu từ đường truyền. Như vậy trong chế độ thăm dò, CPU của bên nhận luôn luôn kiểm tra D1 của thanh ghi trạng thái của 8251. Trong chế độ ngắt, RxRDY = 1 đưa vào chân yêu cầu ngắt INTR của CPU, và nếu CPU chấp nhận ngắt nó sẽ nhận số hiệu ngắt từ

ngoại vi để nhảy tới thủ tục phục vụ ngắt truyền thông của BIOS. Khuôn dạng dữ liệu, độ dài bit stop, chẵn lẻ, và tốc độ truyền thông của cả hai đầu cuối phải giống nhau. Sự khởi tạo các thông số này sai ở cả 2 USART 8251 sẽ làm cho truyền thông bị lỗi và không thể thực hiện được. Môi trường truyền dẫn có thể là mạng điện thoại công cộng và phải có thêm các mô-đem kết nối qua giao tiếp RS-232C ở mỗi đầu cuối. Nếu dùng đôi cáp nối trực tiếp 2 đầu cuối thì phải sử dụng Null mô-đem.



Hình 5.77: Truyền thông nối tiếp không đồng bộ bằng phương pháp ngắt và thăm dò giữa 2 đầu cuối máy tính sử dụng USART 8251

Các cổng nối tiếp của PC được nối với nhau sử dụng các chuẩn: RS-232C, RS-422, dòng vòng 20mA. RS-232C có mức logic "1" hay trạng thái "Mark" từ -3V đến - 25V, logic "0" hay trạng thái "Space" là từ +3V đến +25V. Trong máy vi tính, sử dụng nguồn cung cấp + 12V/- 12V để thiết lập tín hiệu cho RS-232C. RS-232C sử dụng 1 dây đơn để truyền dữ liệu 1 dây khác để nhận. Khoảng cách cho phép đối với RS-232C là 15 m với mức 9600 bit/s. Đường truyền càng xa thì tốc độ dữ liệu truyền được càng thấp và ngược lại. Không giống như RS-232C, RS-422 có mức logic ở 2 dây khác nhau, tín hiệu biến đổi trong khoảng từ 0-5V. Sơ đồ trong hình 5.78 là mạch vi sai RS-422 đặc trưng. RS-422 không yêu cầu cung cấp nguồn vi sai ngay lập tức trong hệ thống của bộ vi xử lý và các mạch logic. Kết thúc qúa trình nhận dữ liệu trên 2 sợi dây cũng khác nhau. Giá trị điện áp trên đôi sợi dây về khía cạnh nối đất là không có vấn đề gì trừ khi nó lớn hơn giá trị mà bộ phận nhận có thể điều khiển. RS-422 không có đường điều khiển và bộ phận nối thông thường.



Hình 5.78: Phát (driver) và thu (receiver) dữ liệu theo chuẩn RS-422

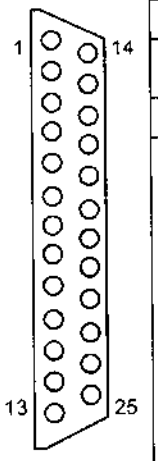
Tín hiệu vi sai của chuẩn RS-422 cho phép truyền nhanh hơn (38,4 kbit/s) và xa hơn (1200 m). RS-422 phổ biến trong nhiều ứng dụng kỹ thuật song nó không được dùng phổ biến như RS-232C cho kết nối truyền thông cục bộ. RS-422 không cung cấp đường truyền cho máy in, mô-đem, các thiết bị nối tiếp khác. Hình 5.79 là dòng mạch

vòng 20 mA đặc trưng. Khác với RS-232C và RS-422 mạch này sử dụng nguồn dòng không đổi 20 mA. Điều đó cho phép loại bỏ những thay đổi bởi tín hiệu từ bên ngoài. Nó có khả năng gửi dữ liệu đi xa hơn RS-232C và tính khử nhiễu lớn hơn RS-422.

Bảng 5.20 mô tả các chân tín hiệu của các đầu nối loại DB25 và loại DB9 theo chuẩn RS-232C và ITU (V.24).

Bảng 5.20: RS-232C phích cắm

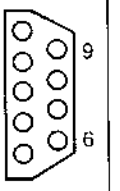
25 chân	Tín hiệu
1	Protective Ground
2	Transmit Data
3	Receive Data
4	Request To Sen
5	Clear To Sen
6	Data Set Ready
7	Signal Ground
8	Receive Live signal detector
9	
10	
11	
12	
13	
14	Lựa chọn dự phòng
15	Thành phần tín hiệu phát
16	
17	Thành phần tín hiệu nhận
18	Thử
19	
20	Sẵn sàng đầu cuối dữ liệu
21	
22	Bộ chỉ thị chuông
23	Lựa chọn tốc độ
24	
25	

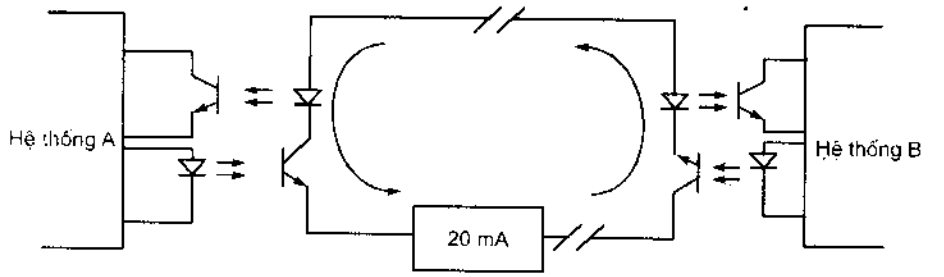


DB25

9 chân	Tín hiệu
1	Carrier Detect
2	Receive Data
3	Transmit Data
4	Data Terminal
5	Ground
6	Data Set Ready
7	Request To Send
8	Clear to Send
9	Ring Indicator

DB9





Hình 5.79: Vòng dòng 20 mA

5.3.6. Phối ghép với bàn phím

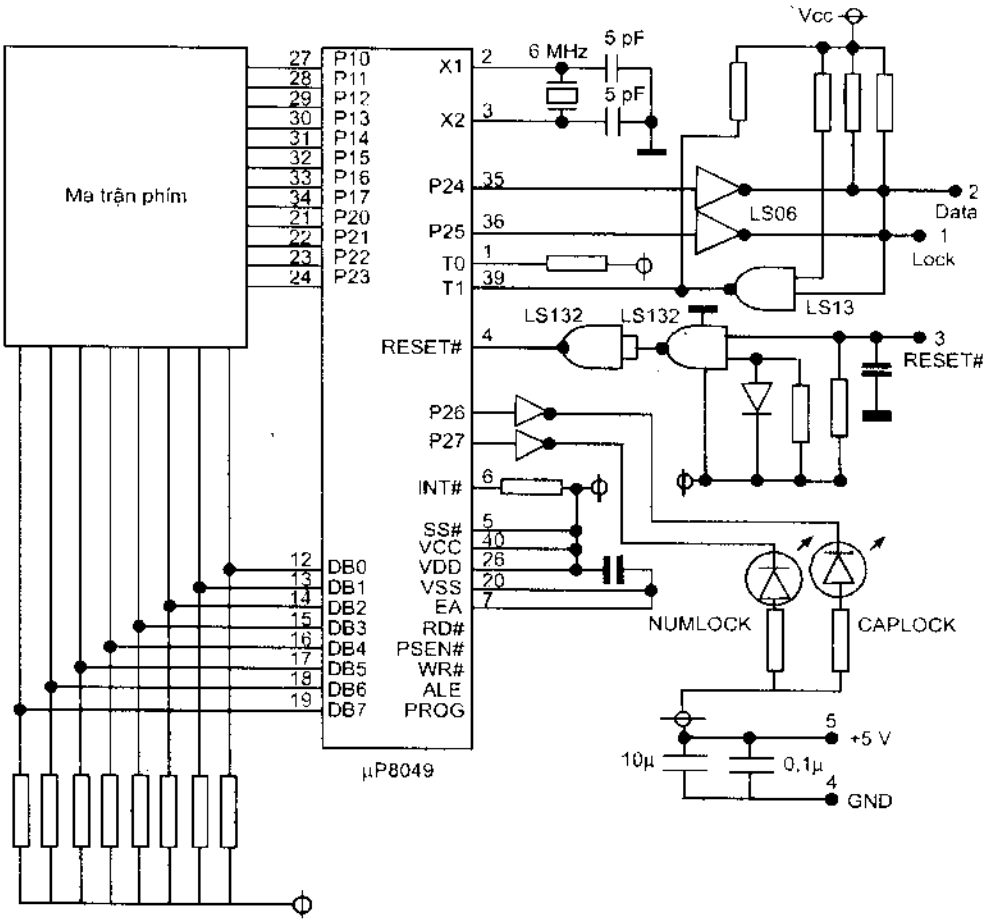
(1) Bàn phím

Nếu như màn hình là thiết bị ra chuẩn, thì bàn phím là thiết bị vào chuẩn của máy tính cá nhân. Bàn phím là thực chất là một hệ thống con, vì nó có một vi xử lý chuyên dụng cho điều khiển giao tiếp bàn phím với CPU (hình 5.80).

Bàn phím dùng cho IBM PC/XT, AT có 83 phím và phân chia thành 3 nhóm chính. Nhóm phím trung tâm được sắp xếp theo chuẩn của bàn phím máy chữ (QWERTY). Nhóm bên trái bàn phím có 10 phím chức năng (F1 đến F10), nhóm bên phải gồm 15 phím, gọi là các phím số (Keypad). Các phím của Keypad cũng được xác định bằng phần mềm nhưng cũng được ghi chú thêm các chức năng như: Num Lock, Scroll Lock, Home, End, PgUp, Pg Down, Enter, Del, các mũi tên chỉ chiều dịch chuyển con trỏ trên màn hình, bàn phím của máy tính bỏ túi (calculator pad). Bảng 83 phím sử dụng công nghệ điện dung có CPU là chip vi xử lý Intel 8049 (hay 8048). Intel 8049 thực hiện một số chúc năng, bao gồm: tự kiểm tra khi bật nguồn, quét bàn phím, kiểm tra các phím được ấn, ghi nhớ đệm 20 mã quét (scan cod) của các phím vào trong bộ đệm mã quét (scan code buffer), duy trì kết nối truyền thông liên tiếp 2 chiều với hệ thống máy tính, và thực hiện thủ tục bắt tay cho mỗi lần truyền mã quét.

Mỗi phím trong bàn phím đều được gán một con số, gọi là mã quét (scan code), từ 1 đến 83. Khi một phím được ấn, bộ xử lý của bàn

phím là chip vi điều khiển 8049 (8048) gửi mã quét của phím được ấn đó cho CPU (8088). Nếu có thể được, CPU sẽ đổi mã quét này thành mã ASCII mở rộng "extended ASCII" và gửi kết quả trả lại bộ đệm của bàn phím. Mã ASCII mở rộng gồm một byte và có thể có các giá trị từ 0 đến 255. Tuy nhiên, một số phím và tổ hợp phím không có mã ASCII mở rộng. Đối với các phím này, CPU sẽ gửi 2 byte vào bộ đệm bàn phím. Byte đầu tiên là 0, byte thứ 2 là mã ASCII mở rộng. Một mã ASCII mở rộng từ 15-83 cũng chính là mã quét.



Hình 5.80: Bàn phím của IBM PC/XT

Lấy ví dụ, khí ấn phím chức năng F1 thì CPU sẽ trả lại byte đầu tiên = 0 và byte thứ 2 có giá trị 59. Như vậy mã quét của phím F1 là

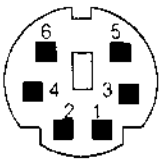
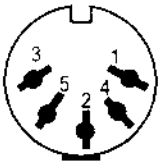
59. Nếu ta ấn tổ hợp các phím: Alt-Q thì CPU trả lại byte 0 và 16. Như vậy mã quét của Alt-Q là 16. Tuy nhiên, một số tổ hợp phím cho mã mở rộng lớn hơn 83, ví dụ, các tổ hợp các phím Shift và Alt với các phím khác. Riêng ký tự NULL có mã mở rộng là 0, và nó không thể gửi đi được vì số 0 được dùng để báo hiệu có mã mở rộng ở byte thứ 2 do CPU trả về bộ đệm bàn phím. Do đó NULL có mã mở rộng là 3 chứ không phải là 0 như bình thường. NULL được sinh ra bởi tổ hợp phím Ctrl-2. Các phím đều tạo ra mã quét khi ấn và khi nhả. Ví dụ: phím số 1 khi ấn xuống tạo ra mã quét = 01 (make scan code) và tạo ra mã 81 khi nhả phím (break scan code). Mã nhả phím = mã quét + 80h.

Trong khi bàn phím của IBM PC/XT có 83, thì của IBM PC hiện nay đã có 104 phím. Các phím được sắp xếp trong một ma trận, mỗi phím được địa chỉ theo cột và hàng. Bộ vi xử lý chuyên dụng 8049 thực hiện quét liên tục ma trận phím để xác định phím đang tích cực (ấn xuống hay nhả lên). Thông tin về từng phím tích cực được đưa vào bộ đệm mã quét của 8049 thông qua các đường dây của bus dữ liệu DB0-DB7.

(2) Phối ghép bàn phím với bảng mạch hệ thống của IBM PC

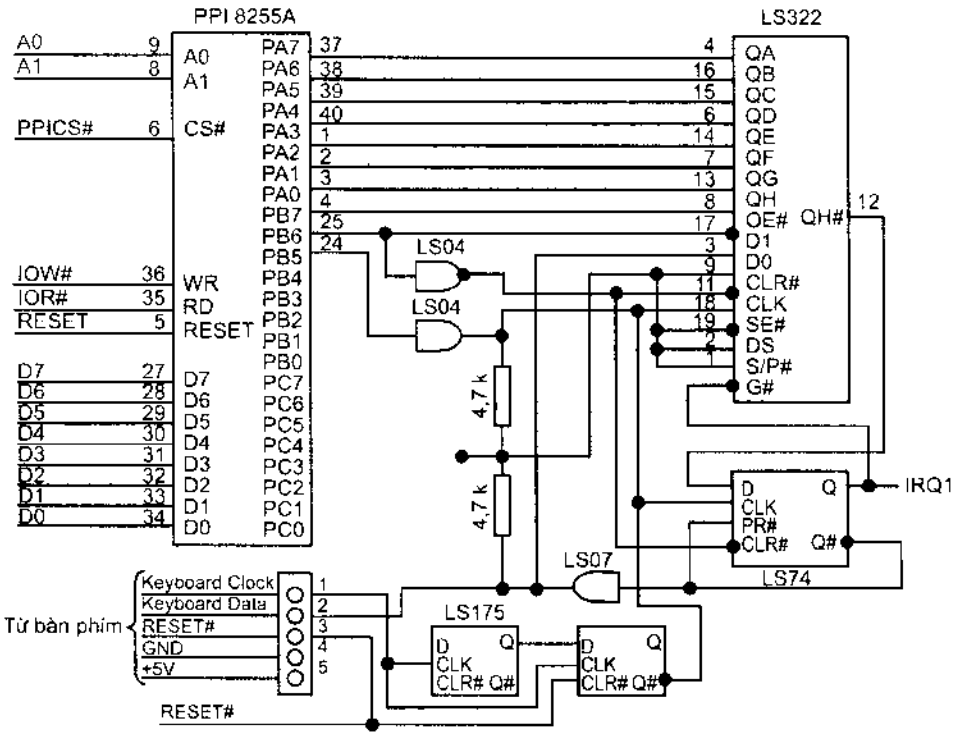
Bàn phím được nối với bảng mạch hệ thống bằng cáp và phích cắm 6 chân (bảng 5.21).

Bảng 5.21: Giao diện bàn phím

6-pin DIN đầu nối	Pin N.	Signal	5-pin DIN đầu nối	Pin N.	Signal
	1	Data		1	Clock
	2	Không dùng		2	Data
	3	GND		3	RESET#
	4	+5V		4	GND
	5	Clock		5	+5V
	6	Không dùng			

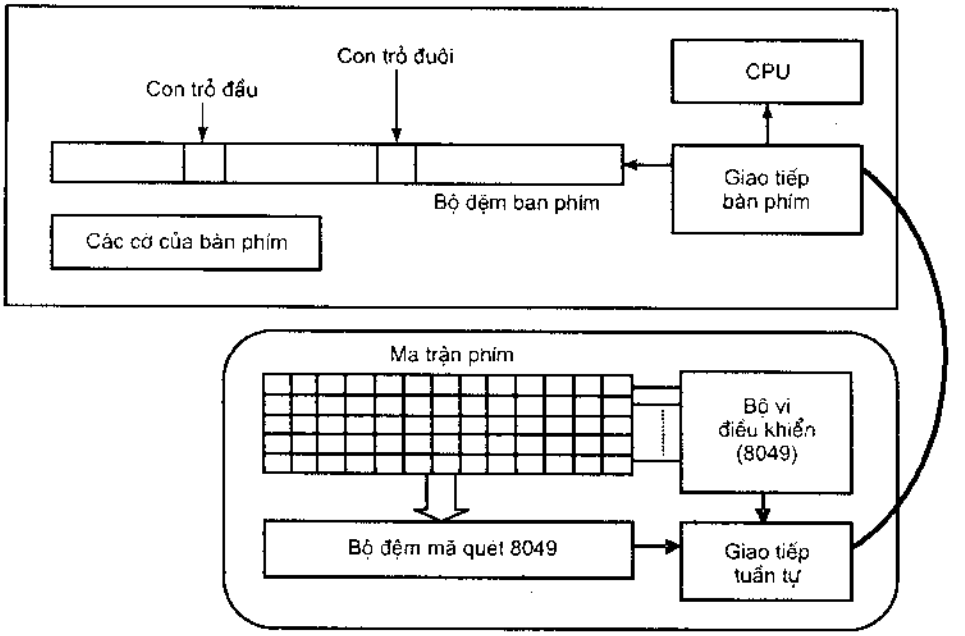
Trên bảng hệ thống của IBM PC/XT, AT khối logic điều khiển vào/ra với bàn phím có hạt nhân là vi mạch lập trình 40 chân DIP PPI 8255.

Hình 5.81 là logic kết nối và bàn phím và PPI 8255. Tín hiệu PPICS# là tín hiệu chọn chip PPI, và được tạo ra từ mạch giải mã chọn các thiết bị ngoại vi: PPICS# = [(AEN = 1)(A9 = 0)(A8 = 0)(A7 = 0)(A6 = 1)(A5 = 1)]#. Dữ liệu (D0-D7) là dữ liệu của bus dữ liệu và địa chỉ (A0, A1) từ bus địa dcó địa chỉ là 60H = (PPICS# = 0)(A1 = 0)(A0 = 0). Cổng B có địa chỉ là 61h = (PPICS# = 0)(A1 = 0)(A0 = 1) và cổng C là 62h = (PPICS# = 0)(A1 = 1)(A0 = 0). Từ bộ đệm này, mã quét được gửi liên tiếp theo đường dây (DATA, CLOCK) đến đơn vị giao tiếp với bàn phím nằm ở trên bảng hệ thống của máy tính. Một ngắt yêu cầu phục vụ vào/ra với bàn phím được tự động sinh ra, nó khởi động thủ tục phục vụ bàn phím trong ROM-BIOS, trong đó mã quét của phím được chuyển thành ký tự ASCII và đưa vào bộ đệm bàn phím trên bảng hệ thống.



Hình 5.81: Khối giao tiếp bàn phím trên bảng mạch hệ thống của IBM PC/XT

Bộ đệm bàn phím được sắp xếp theo nguyên tắc xếp hàng: dữ liệu mới được nạp vào từ bên phải (ở đuôi) vào vị trí mà con trỏ đuôi của bộ đệm chỉ ra, dữ liệu tiếp theo được đọc ra được lấy từ hàng ở phía trái mà con trỏ đầu chỉ ra. Sau mỗi lần thực hiện xử lý phím ấn (input hay output), con trỏ được tự động thay đổi và trỏ đến đầu và đuôi của dòng dữ liệu. Đặc tính này tạo cho việc thao tác bàn phím độc lập. Hình 5.82 là sơ đồ kết nối bàn phím và máy vi tính.



Hình 5.82: Kết nối bàn phím với máy vi tính

Các trình phục vụ của ROM-BIOS được kích hoạt (thông qua cổng và ngắt của bàn phím) có nhiệm vụ đổi các tác động phím thành thông tin có nghĩa cho hiển thị cũng như để xử lý được trong máy tính. Xét trình tự khi thực hiện ấn một phím nào đó, ví dụ phím A:

• Ấn Shift - A: từ ma trận phím một mã quét của A được tạo ra và ghi vào bộ đệm mã quét trong 8049 qua các đường DB0-DB7.

• Mã quét của A được gửi theo đường dây tuần tự (DATA, CLOCK) tới đơn vị giao tiếp bàn phím nằm trên bảng mẹ.

• Khi toàn bộ 1 byte của mã quét phím A (ấn) nhận đủ, đơn vị giao tiếp bàn phím sinh ra ngắt INT 9h đi tới CPU, ROM-BIOS thực hiện chương trình chuyển đổi mã quét thành mã ASCII (41h) và cất mã ký tự ASCII này vào bộ đệm bàn phím ở vị trí con trỏ đuôi. Con trỏ đuôi được tăng lên để chỉ tới vị trí cho đưa vào tiếp theo. Bộ đệm bàn phím được truy nhập ở vị trí con trỏ đầu nhờ chương trình ứng dụng.

• Mã ASCII của phím A (41h) sẽ được chuyển đến vị trí con trỏ đầu.

Khi lập trình, khi muốn truy nhập bàn phím cần phải thực hiện với 3 mức sau đây:

• Mức bộ nhớ/cổng:

Cổng 60h - bộ đệm mã quét

Vùng nhớ 40: 1Ah - Head pointer (trỏ đầu)

Vùng nhớ 40: 1Ch - Tail pointer (trỏ đuôi)

• Mức BIOS:

INT 16h, mã chức năng 00h, đọc bàn phím
Trả về: AL = mã ASCII, AH = mã quét
INT 16h, mã chức năng 01h, đọc trạng thái bàn phím
Trả về: AL = mã ASCII, AH = scan code, ZF = 1 (buffer rỗng)
INT 16h, mã chức năng 02h, đọc trạng thái phím Shift
Trả về: AL = trạng thái của phím Shift
INT 16h, mã chức năng 05h: ghi vào từ bàn phím
CL = mã ASCII của phím, CH = mã quét của phím

• Mức DOS:

INT 21h, mã chức năng 1: ghi vào từ bàn phím mã ASCII của phím ấn vàothanh ghi AL, hiển thị lên màn hình ký tự của phím.

AH = 1, (AH = 8, không có hiển thị)

INT 21h, mã chức năng 11: kiểm tra phím được ấn?

AH = 0BH. Trả về: AL=0 nếu không có phím ấn, #0 nếu có phím ấn

5.3.7. Phối ghép với màn hình

a) Hệ thống hiển thị

Hệ thống hiện thị trong máy vi tính bao gồm: màn hình (đơn sắc, màu/đồ họa), bộ thích ứng màn hình màu/đồ họa (color/graphics adapter), các cáp nối. Bộ điều khiển có thể là bảng mạch riêng hoặc nằm trên bảng mẹ. Màn hình có thể là đơn sắc (dùng cho IBM PC/X, AT) và màu/đồ họa mật độ cao. Các màn hình dùng cho máy tính cá nhân thường cho các chế độ hiển thị trong bảng 5.22.

Phân loại các bộ thích ứng màn hình như sau:

• MDA (Monochrome/graphics Display Adapter) - Bộ thích ứng hiển thị đồ họa đơn sắc

• CGA (Color-Graphics Adapter) - Bộ thích ứng màu/đồ họa.

• EGA (Enhanced-Graphics Adapter) - Bộ thích ứng đồ họa công nghệ cao

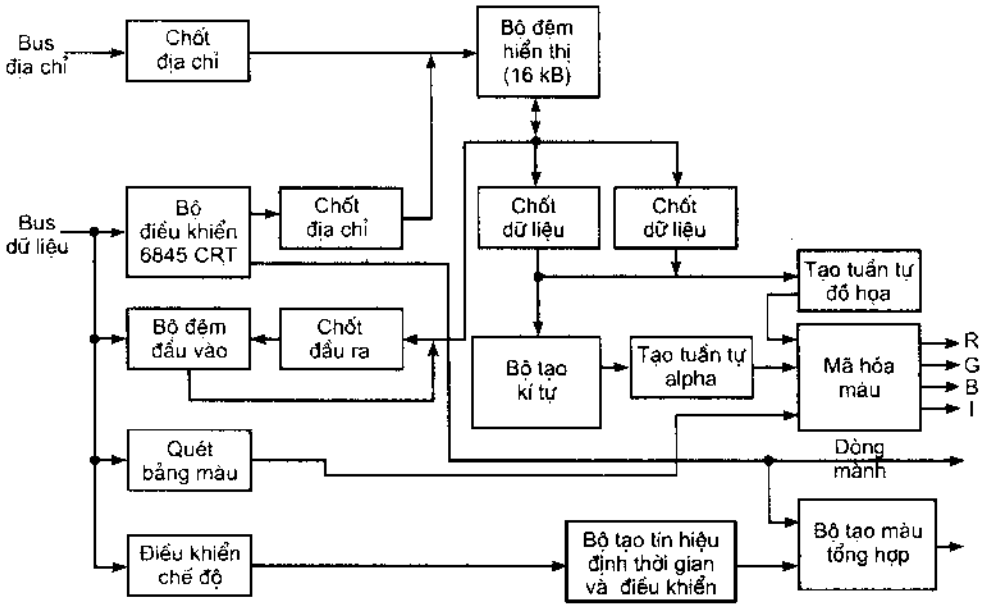
• VGA (Video-Graphics Array) - Bộ thích ứng mảng đồ họa video.

b) Bộ thích ứng màu - đồ họa CGA

Trong các hệ thống IBM PC/XT, AT bảng mạch này có bộ nhớ đệm cho màn hình (sử dụng các chip DRAM) và có hạt nhân là chip điều khiển màn hình, ví dụ chip lập trình điều khiển CRT 40-pin NEC 6845 (hoặc các chip tương tự như: 68845, 6847, HD465055P, MC1372, 8275, 7220, CRT5027, CRT5037, TMS9918A) có tới 19 thanh ghi bên trong.

Hình 5.83 là sơ đồ khối của bảng mạch thích ứng màn hình đồ họa màu (bảng điều khiển màn hình đồ họa màu) có chip 6845 CRT phổ biến dùng cho IBM PC/XT, AT. Bảng mạch này cắm trên khe mở rộng của PC và có phích cắm kết nối với màn hình màu của IBM PC, với các màn hình tần số truyền hình theo chuẩn NTSC (U.S. National Television System Committee) hoặc tivi dân dụng chuẩn NTSC. Bảng mạch này cho phép kết nối cả màn hình đơn sắc và màu. Nó có 3 giao

tiếp video: tổng hợp video (Composite Video), điều khiển trực tiếp (direct drive) và đầu nối cho điều khiển bộ điều biến tần số vô tuyến (RF modulation). Ngoài ra nó có cổng kết nối với bút sáng (light pen).



Hình 5.83: Sơ đồ khối của CGA sử dụng bộ điều khiển Motorola 6845 CRT

Bảng 5.22: Các chế độ hiển thị của màn hình

Chế độ	Loại định dạng	Định dạng Alpha	Độ phân giải (PELs)/cột	Kích thước kí tự	Bộ thích ứng	Bộ đệm bắt đầu
0	text	40x25	320x200/16 xám	8x8	CGA,EGA	B8000
1	text	40x25	320x200/16 màu chữ, 8 màu nền	8x8	CGA,EGA 64 màu	B8000
3	text	80x25	640x200/16 màu chữ, 8 màu nền	8x8	CGA,EGA 64 màu	B8000
4	graphics	40x25	320x200/4	8x8	CGA,EGA	B8000
5	graphics	40x25	320x200/4 xám	8x8	CGA,EGA	B8000
6	graphics	80x25	640x200/2	8x8	CGA,EGA	B8000
7	text	80x25	720x350 mono	9x14	EGA	B8000
8	graphics	80x25	640x200/16	8x8	CGA,EGA	B8000

Chế độ	Loại định dạng	Định dạng Alpha	Độ phân giải (PELs)/cột	Kích thước kí tự	Bộ thích ứng	Bộ đệm bắt đầu
9	graphics	40x25	320x200/16	8x8	CGA,EGA	B8000
A	graphics	80x25	640x200/4	8x8	CGA,EGA	B8000
D	graphics	40x25	320x200/16	8x8	EGA	A0000
E	graphics	80x25	640x200/16	8x8	EGA	A0000
F	graphics	80x25	640x350-mono	8x14	EGA	A0000
10	graphics	80x25	640x350/16-64	8x14	EGA	A0000
11	graphics	80x30	640x480/2	8x16	VGA	A0000
12	graphics	80x30	640x480/16	8x16	VGA	A0000
13	graphics	40x25	320x200/256	8x8	VGA	A0000
50	graphics	80x30	640x480/16	8x16	VGA	B8000
51	graphics	80x43	640x473/16	8x11	VGA	B8000
52	graphics	80x60	640x480/16	8x8	VGA	B8000
53	graphics	132x25	1056x359/16	8x14	VGA	B8000
54	graphics	132x30	1056x480/16	8x14	VGA	B8000
55	graphics	132x43	1056x473/16	8x11	VGA	B8000
56	graphics	132x60	1056x480/16	8x8	VGA	B8000
57	graphics	132x25	1188x350/16	9x14	VGA	B8000
58	graphics	132x30	1188x480/16	9x16	VGA	B8000
59	graphics	132x60	1188x473/16	9x11	VGA	B8000
5A	graphics	132x60	1188x480/16	9x8	VGA	B8000
5B	graphics	100x75	800x600/16	8x8	VGA/VESA	A0000
5C	graphics	80x25	640x480/256	8x16	VGA/VESA	A0000
5D	graphics	80x25	640x480/256	8x16	VGA/VESA	A0000
5E	graphics	100x25	800x600/256	8x8	VGA/VESA	A0000
5F	graphics	128x48	1024x768/16	8x16	VGA/VESA	A0000
60	graphics	128x48	1024x768/4	8x16	VGA/VESA	A0000
61	graphics	96x64	768x1024/16	8x16	VGA/VESA	A0000

Xét bảng thích ứng màn hình màu/đồ họa (CGA) dùng chip CRT 6845.

- Thanh ghi thiết lập chế độ

Chip 6845 có thanh ghi thiết lập chế độ (Mode Set Register), đó là thanh ghi dùng chung, vào/ra và lập trình được. Nó có các cổng I/O có thể được lập trình riêng biệt. Chức năng của nó là để chọn chế độ và chọn màu trong chế độ màu/đồ họa phân giải trung bình.

- Bộ đệm hiển thị

Bộ đệm hiển thị (Display buffer) là bộ nhớ ghi/đọc (DRAM) nằm trên bảng thích ứng màn hình, nhưng chiếm một không gian địa chỉ 16 kB của hệ thống nhớ, bắt đầu từ B8000h. Tức là CPU đánh địa chỉ trực tiếp bộ nhớ hiển thị. Vì là DRAM nên phải được làm tươi, và bộ đệm hiển thị cũng được gọi là bộ đệm làm tươi (refresh buffer). CPU và bộ thích ứng màn hình đều có quyền truy nhập vào bộ đệm hiển thị này trong tất cả các chế độ hiển thị, ngoại trừ chế độ ký tự văn bản phân giải cao. Vì trong chế độ này, chỉ có CPU truy nhập tới trong các quãng thời gian làm tươi bộ đệm hiển thị theo hàng.

- Bộ tạo ký tự

Bộ tạo ký tự chứa trong ROM (ROM Character Generator) có 8 kB nhớ. Nó không thể ghi hoặc đọc thông qua phần mềm được. Nó có 3 bộ phông chữ khác nhau. Hai bộ phông chữ sử dụng trên bộ thích ứng màn hình màu/đồ họa: phông 7×7 điểm kép và 5×7 phông điểm đơn. Các bộ phông chữ được chọn nhờ cầu nối.

- Bộ tạo tín hiệu định thời gian

Bộ tạo tín hiệu định thời gian (Timing Generator) tạo ra các tín hiệu định thời gian cho 6845 và cho cả CPU. Nó cũng giải quyết cả yêu cầu của bộ thích ứng màn hình để truy nhập bộ đệm hiển thị.

- Bộ tạo màu tổng hợp

Bộ tạo thông tin màu hình ảnh video giải cơ bản.

- Bộ điều khiểnCRT 6845

Chip CRT 6845 có 19 thanh ghi bên trong để định nghĩa và điều khiển hiển thị quét mành màn hình (raste scan). Một trong các thanh

ghi này được sử dụng như là thanh ghi chỉ số (index register) với địa chỉ I/O 3D4h để trỏ tới 18 thanh ghi còn lại. Chỉ có thể ghi vào thanh ghi chỉ số, mà không thể đọc được. Bằng lệnh OUT 3D4h, nạp nội dung thanh ghi AL, trong đó chứa số hiệu thanh ghi cần chọn, vào thanh ghi này. Phần cần 5 bit thấp của bus dữ liệu 8-bit ($2^5 = 32$) để ghi số hiệu của 18 thanh ghi. Sau đó, để ghi dữ liệu vào các thanh ghi đã chọn bên trong CRT 6845, thực hiện lệnh OUT 3D5h. Giá trị 3D5h là địa chỉ I/O của thanh ghi dữ liệu của CRT 6845. Bảng 5.23 liệt kê các địa chỉ I/O để lập trình *thanh ghi chế độ và trạng thái* hiển thị cho CGA. Bảng 5.24 các định các giá trị phải được nạp vào các thanh ghi bên trong CRT 6845 để điều khiển các chế độ khai thác của CGA.

Bảng 5.23: Lập trình thanh ghi điều khiển chế độ hiển thị và trạng thái

Địa chỉ	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Chức năng của thanh ghi
3D8H	1	1	1	1	0	1	1	0	0	0	Thanh ghi điều khiển chế độ, D0
3D9H	1	1	1	1	0	1	1	0	0	1	Thanh ghi chọn màu, D0
3DAH	1	1	1	1	0	1	1	0	1	0	Thanh ghi trạng thái, D0
3DBH	1	1	1	1	0	1	1	0	1	1	Chốt xóa bút quang
3DCH	1	1	1	1	0	1	1	1	0	0	Chốt đặt bút quang
3D4H	1	1	1	1	0	1	0	x	x	0	Thanh ghi chỉ số 6845
3D5H	1	1	1	1	0	1	0	x	x	1	Thanh ghi dữ liệu 6845
3D0H	1	1	1	1	0	1	0	x	x	0	Các thanh ghi 6845
3D1H	1	1	1	1	0	1	0	x	x	1	Các thanh ghi 6845

(Các giá trị x trong bảng là giá trị không có nghĩa, bất kỳ).

Bảng 5.24: 18 thanh ghi bên trong của chip CRT 6845

Đ/chỉ	Thanh ghi	Loại thanh ghi	Đơn vị	I/O	40x25	80x25	Đồ họa
0	R0	Tổng số dòng	Kí tự	Chỉ ghi	38	71	38
1	R1	Hiển thị số dòng	Kí tự	Chỉ ghi	28	50	28
2	R2	Đồng bộ dòng	Kí tự	Chỉ ghi	2D	5A	2D
3	R3	Chiều rộng đồng bộ dòng	Kí tự	Chỉ ghi	0A	0A	0A
4	R4	Tổng số mành	Dòng kí tự	Chỉ ghi	1F	1F	1F

Đ/chỉ	Thanh ghi	Loại thanh ghi	Đơn vị	I/O	40x25	80x25	Đồ họa
5	R5	Điều chỉnh tổng số mành	Dòng quét	Chỉ ghi	06	06	06
6	R6	Mành	Dòng kí tự	Chỉ ghi	19	19	64
7	R7	Vị trí đồng bộ mành	Dòng kí tự	Chỉ ghi	1C	1C	70
8	R8	Chế độ xen kẽ	-	Chỉ ghi	02	02	02
9	R9	Các địa chỉ dòng quét lớn nhất	Dòng quét	Chỉ ghi	07	07	01
A	R10	Vị trí đầu của con trỏ	Dòng quét	Chỉ ghi	06	06	06
B	R11	Vị trí cuối của con trỏ	Dòng quét	Chỉ ghi	07	07	07
C	R12	Địa chỉ bắt đầu (H) - byte cao	-	Chỉ ghi	00	00	00
D	R13	Địa chỉ bắt đầu (L) - byte thấp	-	Chỉ ghi	00	00	00
E	R14	Địa chỉ con trỏ (H) - byte cao	-	Ghi/đọc	XX	XX	XX
F	R15	Địa chỉ con trỏ (L) - byte thấp	-	Ghi/đọc	XX	XX	XX
10	R16	Bút quang (H) - byte cao	-	Chỉ đọc	XX	XX	XX
11	R17	Bút quang (L) - byte thấp	-	Chỉ đọc	XX	XX	XX

Chip CRT 6845 có 19 thanh ghi bên trong để định nghĩa và điều khiển hiển thị quét mành màn hình (raste scan). Một trong các thanh ghi này được sử dụng như là thanh ghi chỉ số (index register) với địa chỉ I/O là 3D4h để trỏ tới 18 thanh ghi còn lại. Chỉ có thể ghi vào thanh ghi chỉ số, mà không thể đọc được. Bằng lệnh OUT 3D4h, nạp nội dung của thanh ghi AL, đó là số hiệu thanh ghi cần chọn, vào thanh ghi chỉ số 3D4h. Phần cần 5 bit thấp của Data bus 8-bit ($2^5 = 32$) để ghi số hiệu của 18 thanh ghi. Sau đó, để ghi dữ liệu vào các thanh ghi đã chọn bên trong CRT 6845, thực hiện lệnh OUT 3D5h. Giá trị 3D5h là địa chỉ I/O của thanh ghi dữ liệu của CRT 6845. Bảng 5.23. liệt kê các địa chỉ I/O để lập trình *thanh ghi điều khiển chế độ và trạng thái* hiển thị cho CGA. Bảng 5.24 tập hợp các định các giá trị phải

được nạp vào các thanh ghi bên trong CRT 6845 để điều khiển các chế độ khai thác của CGA.

Thanh ghi chọn màu (color-select register) của CGA là thanh ghi 6-bit và chỉ được ghi/không đọc được. Nó có địa chỉ 3D9h. Các bit của thanh ghi này có ý nghĩa cho trong bảng 5.25. Lệnh OUT 3D9h thực hiện ghi nội dung thanh ghi AL vào thanh ghi này.

Bảng 5.25: Các bit của thanh ghi chọn màu (3D9h)

Bit	Chức năng
0	Chọn màu xanh da trời B (Blue) cho viền ngoài (border) trong chế độ ký tự 40x25
	Chọn màu xanh da trời B (Blue) cho nền (background) trong chế độ đồ họa 320 x 200
	Chọn màu xanh da trời B (Blue) cho phần nổi (foreground) trong chế độ đồ họa 640x200
1	Chọn màu xanh lá cây G (Green) cho viền ngoài (border) trong chế độ ký tự 40x25
	Chọn màu xanh lá cây G (Green) cho nền (background) trong chế độ đồ họa 320 x 200
	Chọn màu đỏ R (Red) cho phần nổi (foreground) trong chế độ đồ họa 640x200
2	Chọn màu đỏ R (Red) cho viền ngoài (border) trong chế độ ký tự 40x25
	Chọn màu đỏ R (Red) cho nền (background) trong chế độ đồ họa 320 x 200
	Chọn màu đỏ R (Red) cho phần nổi (foreground) trong chế độ đồ họa 640x200
3	Chọn I (Intensified) cho viền ngoài (border) trong chế độ ký tự 40x25
	Chọn I (Intensified) cho nền (background) trong chế độ đồ họa 320 x 200
	Chọn I (Intensified) cho phần nổi (foreground) trong chế độ đồ họa 640x200
4	Chọn một trong hai tập màu với I trong chế độ đồ họa, chọn các màu cho nền trong chế độ ký tự
5	Chọn tập màu kích hoạt trong chế độ đồ họa 320x200
6,7	Không sử dụng

Các bit 0, 1, 2, 3 của thanh ghi 3D9h chọn màu viền của màn hình trong chế độ ký tự 40×25. Chúng chọn màu nền cho màn hình (C0-C1) trong chế độ màu/đồ họa phân giải trung bình (320×200). Bit 4 khi được thiết lập, sẽ chọn một trong hai tập màu với I (Intensified). Nó cũng chọn các màu cho nền trong chế độ ký tự. Bit 5 chỉ được sử dụng trong chế độ màu/đồ họa phân giải trung bình (320×200). Nó

được sử dụng để chọn tập màu kích hoạt cho hiển thị. Khi bit 5 thiết lập bằng "1" thì các màu được xác định như sau:

C1	C0	Màu được chọn
0	0	Màu nền được xác định bằng các bit 0-3 của cổng I/O (3D9h)
0	1	Xanh tím (Cyan)
1	0	Đỏ tươi (Magenta)
1	1	Trắng (White)

Khi 5 thiết lập bằng "0" thì các màu được xác định như sau:

C1	C0	Màu được chọn
0	0	Màu nền được xác định bằng các bit 0-3 của cổng I/O (3D9h)
0	1	Xanh lá cây (Green)
1	0	Đỏ (Red)
1	1	Nâu (Brown)

Thanh ghi chọn chế độ (MSR - Mode Select Register) của CGA là thanh ghi 6-bit và chỉ ghi được/không thể đọc được. Nó được truy nhập để ghi bằng lệnh OUT 3D8h. Các bit của thanh ghi này (3D8h) có ý nghĩa và chức năng như sau:

Bit 0: = 1: chọn chế độ ký tự 80 × 25.
= 0: chọn chế độ ký tự 40 × 25.

Bit 1: = 1: chọn chế độ đồ họa 320 × 200.
= 0: chọn chế độ ký tự.

Bit 2: = 1: chọn chế độ hiển thị đen trắng.
= 0: chọn chế độ hiển thị màu

Bit 3: = 1: cho phép tín hiệu video ở thời gian nào đó khi thay đổi chế độ. Tín hiệu video phải bị cấm khi thay đổi chế độ.

Bit 4: = 1: chọn chế độ đồ họa đen trắng phân giải cao (640×200). Một trong 8 màu có thể được chọn nhờ sử dụng thanh ghi 3D9h.

Bit 5: = 1: sẽ thay đổi cường độ sáng (intensity) của nền ký tự thành thuộc tính nhấp nháy cho các chế độ ký tự. Khi bit cao của byte thuộc tính không được chọn, thì 16 màu cho nền được sử dụng. Khi hiển thị bình thường bit 5 phải được thiết lập bằng "1" để cho nhấp nháy.

Thanh ghi trạng thái (SR - Status Register) của CGA là thanh ghi 4 bit và được truy nhập chỉ để đọc/không ghi được qua địa chỉ I/O 3DAh. Các bit của nó có ý như sau:

Bit 0: = 1: chỉ ra rằng bộ đệm hiển thị có thể được truy nhập mà không có gây nhiễu hiển thị.

Bit 1: = 1: chỉ ra rằng có sự thiết lập trigger của bút sáng (light pen trigger). Trigger này được xóa khi thiết lập lại (hoặc bật nguồn) hoặc bằng lệnh đưa ra OUT 3DBh.

Bit 2: = 0: chỉ ra rằng bút sáng đã bật.

Bit 3: Đồng bộ mành = 1: chỉ ra rằng mành (raster) của màn hình đang trong chế độ dò lại mành (vertical retrace mode). Trong thời gian này có thể thực hiện bổ sung bộ đệm hiển thị.

(1) Chế độ hiển thị ký tự

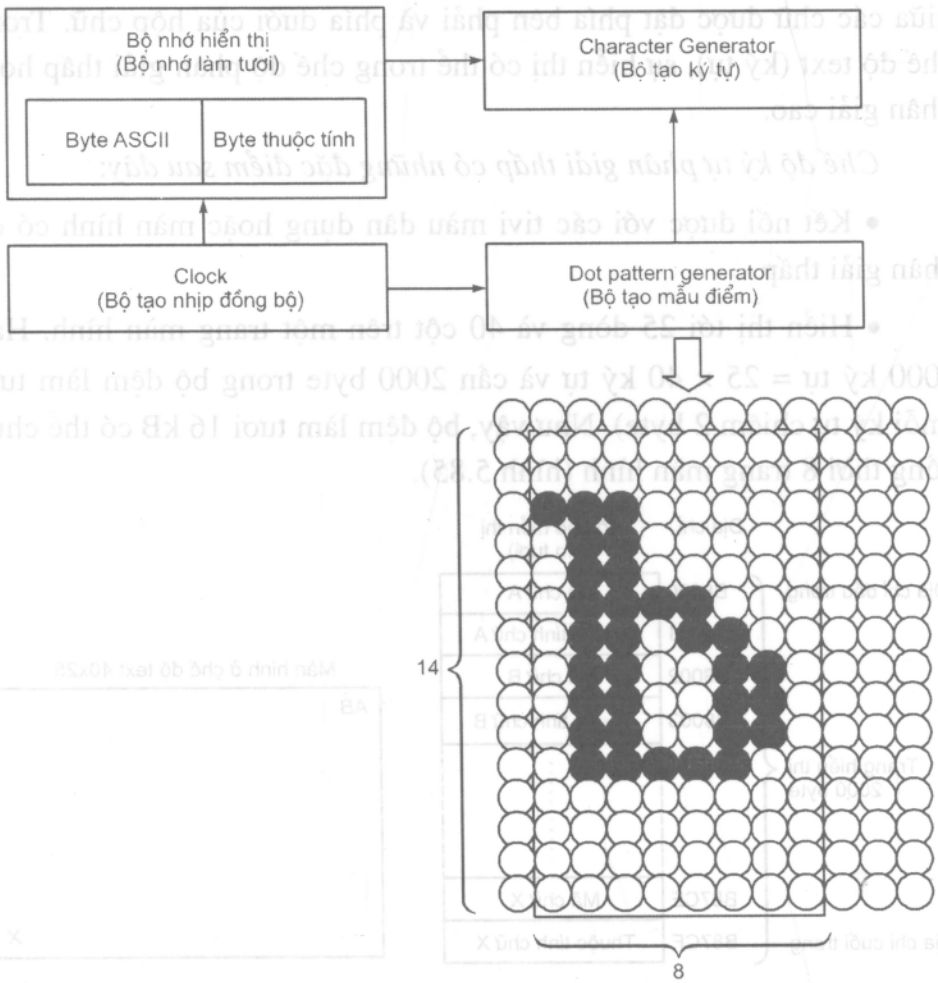
Chế độ này được đề xuất đầu tiên và rất đơn giản. Màn hình được chia thành mảng 25 dòng và 80 cột hộp chữ. Mỗi một hộp chữ chứa một ký tự dựa trên mã ASCII. Một ký tự lưu giữ trong bộ nhớ hiển thị là hai byte dữ liệu liền nhau. Byte thứ nhất là mã ký tự ASCII, byte thứ hai là giá trị thuộc tính của ký tự:

Và ý nghĩa từng bit cho trong bảng 5.26.

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte mã kí tự hiển thị								Byte thuộc tính của kí tự							

Bảng 5.26: Ý nghĩa các bit byte thuộc tính

Chức năng của thuộc tính	Byte thuộc tính của kí tự							
	7	6	5	4	3	2	1	0
	B	R	G	B	I	R	G	B
	FG	Background			Foreground			
Bình thường	B	0	0	0	I	1	1	1
Video đảo	B	1	1	1	I	0	0	0
Không hiển thị (đen)	B	0	0	0	I	0	0	0
Không hiển thị (trắng)	B	1	1	1	1	1	1	1



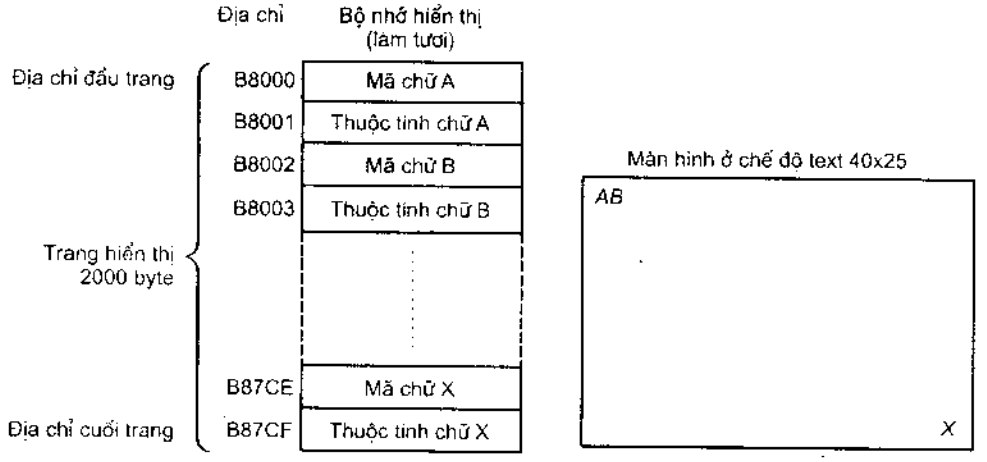
Hình 5.84: Hiển thị ở chế độ ký tự

Hai byte này được gửi tới bộ tạo ký tự (character generator ROM) thông qua các mạch chốt dữ liệu (Data Latches) và bộ tạo ký tự bật điều khiển bộ tạo mẫu điểm cho ký tự cần hiển thị. Toàn bộ quá trình đó có thể được mô tả trong hình 5.84, trong đó kích thước chữ là 8×14 điểm. Vị trí của ký tự trong bộ nhớ hiển thị xác định vị trí hiển thị của nó trên màn hình. Khoảng 30 lần trong một giây, nội dung cần hiển thị được đọc từ bộ nhớ hiển thị (đệm làm tươi), và nhờ bộ tạo ký tự và bộ tạo mẫu điểm, các ký tự được chuyển đổi thành các ký tự trong khuôn dạng ma trận các điểm để đưa lên màn hình. Khoảng cách giữa các chữ được đặt phía bên phải và phía dưới của hộp chữ. Trong chế độ text (ký tự), sự hiển thị có thể trong chế độ phân giải thấp hoặc phân giải cao.

Chế độ ký tự phân giải thấp có những đặc điểm sau đây:

• Kết nối được với các tivi màu dân dụng hoặc màn hình có độ phân giải thấp.

• Hiển thị tới 25 dòng và 40 cột trên một trang màn hình. Hay 1000 ký tự = 25 × 40 ký tự và cần 2000 byte trong bộ đệm làm tươi (mỗi ký tự chiếm 2 byte). Như vậy, bộ đệm làm tươi 16 kB có thể chứa đồng thời 8 trang màn hình (hình 5.85).



Hình 5.85: Vị trí hiển thị của các ký tự trong chế độ ký tự 40 × 25

• Bộ tạo ký tự (ROM character generator) chứa các mẫu điểm cho tối đa 256 ký tự (character set).

• Một hộp biểu diễn ký tự có kích thước 8 × 8 điểm.

• Có 2 bộ phông ký tự: phông 5×7 điểm đơn (single-dot) với một điểm kết thúc sau, và phông 7×7 điểm kép (double-dot) với một điểm kết thúc sau. Mỗi ký tự có một thuộc tính.

Chế độ phân ký tự giải cao có những đặc điểm sau đây:

• Kết nối trực tiếp các loại màn hình màu và đen trắng tích hợp tín hiệu video (black white composite video monitor).

• Hiển thị tới 25 dòng và 80 cột. Hay 2000 ký tự và cần tới 4000 byte trong bộ nhớ hiển thị. Như vậy bộ nhớ hiển thị có thể chứa đồng thời 4 trang màn hình.

• Bộ tạo ký tự chứa các mẫu điểm cho tối đa 256 ký tự.

• Một hộp ký tự có kích thước 8 × 8 điểm và cũng có 2 bộ phông ký tự giống như trong chế độ ký tự phân giải giải thấp.

• Mỗi ký tự có một thuộc tính.

Các bảng mạch phối ghép màn hình đồ họa màu (CGA/EGA/VGA) đều hỗ trợ cho hiển thị ở chế độ ký tự (alphnumeric modes). Các tập mẫu ký tự (character sets) cho CGA/EGA/VGA chứa trong ROM-BIOS. Mỗi một tập có các mẫu điểm cho 256 ký tự. Trong một thời điểm có thể có hai tập mẫu ký tự hoạt động để kết hợp tạo thành một tập với 512 mẫu ký tự. Vì những tập mẫu ký tự này là những phông ký tự có thể nạp, nên người lập trình có thể bổ sung thêm các phông chữ của mình (user fonts).

(2) Chế độ đồ họa

Chế độ đồ họa là chế độ mà trong đó màn hình được chia thành các dòng/cột điểm. Mỗi điểm là một phần tử ảnh (picture element), gọi là PEL. Các PEL được tổ chức thành các dòng quét (scan lines). Một dòng quét gồm một số PEL và tạo nên mật độ theo chiều ngang và

chiều đứng của màn hình. Những bảng mạch CGA thế hệ IBM PC/XT, AT sử dụng chip CRT 6845 (hoặc tương đương) có 3 chế độ đồ họa: chế độ màu/đồ họa phân giải thấp, chế độ màu/đồ họa phân giải trung bình, và chế độ màu/đồ họa phân giải cao. Chỉ có các chế độ đồ họa phân giải trung bình và cao được hỗ trợ trong ROM. Ba chế độ này có đặc tính cho trong bảng 5.27.

Bảng 5.27:

Chế độ (Mode)	**Chiều ngang** (PELs)	**Chiều dọc** (hàng)	**Số lượng màu** (kể cả màu nền)
Phân giải thấp	160	100	16 (kể cả đen trắng)
Phân giải trung bình	320	200	4 màu tổng thể: 1 trong 16 cho nền và 1 trong các màu: Xanh lá cây, đỏ, nâu 1 trong các màu: Xanh tím. đỏ tươi, trắng
Phân giải cao	640	320	Chỉ màu đen trắng (Black-and-white)

Để lập trình hiển thị ở các chế độ này, chúng ta có thể tra theo bảng 5.27 để chọn giá trị chế độ cho thanh ghi AL khi sử dụng ngắt điều khiển màn hình của ROM-BIOS (INT 10H) hoặc ngắt của DOS (INT 21H).

Chế độ màu/đồ họa phân giải thấp (160 × 100)

• Chứa tối đa 100 hàng, mỗi hàng có 160 PEL, mỗi PEL có kích thước: cao 2 (2 bit), và rộng 2 (2 bit).

• Có 1 trong 16 màu cho mỗi PEL với các bit thuộc tính I, R, G và B.

• Chiếm 16000 byte của bộ nhớ hiển thị (làm tươi) trên bảng điều khiển.

• Sử dụng đồ họa bộ nhớ sắp xếp.

Chế độ màu/đồ họa phân giải thấp (320 × 200)

Chế độ này cho phép kết nối với các home tivi hoặc các màn hình màu (color monitor). Nó có các đặc điểm sau đây:

• Chứa tối đa 200 hàng, mỗi hàng có 320 PEL, mỗi PEL có kích thước: cao 1 (1 bit), và rộng 1 (1 bit).

• Có 1 trong 4 màu cho mỗi PEL.

• Chiếm 16000 byte của bộ nhớ hiển thị trên bảng điều khiển.

• Sử dụng đồ họa bộ nhớ sắp xếp.

• Khuôn dạng 4 PEL cho một byte như sau:

7	6	5	4	3	2	1	0
C1	C0	C1	C0	C1	C0	C1	C0
PEL hiển thị đầu tiên		PEL hiển thị thứ 2		PEL hiển thị thứ 3		PEL hiển thị thứ 4	

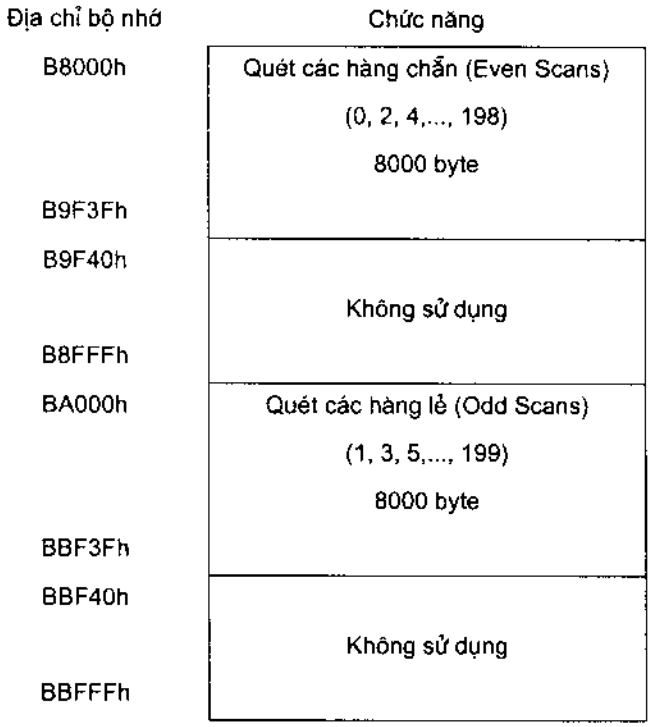
• Hai bit C1 C0 của byte chọn màu được xác định theo logic sau đây:

C1	C0	Chức năng
0	0	Chọn màu trong 16 màu cho nền
0	1	Chọn màu đầu tiên của tập hợp màu 1 hoặc của tập hợp màu 2
1	0	Chọn màu thứ 2 của tập màu 1 hoặc tập màu 2
1	1	Chọn màu thứ 3 của tập màu 1 hoặc tập màu 2

C1 và C0 sẽ chọn 4 trong 16 màu. Sự lựa chọn màu trong bảng màu (color palette) được nạp trước qua cổng I/O của bảng điều khiển màn hình màu/đồ họa. Các màu của nền giống như 8 màu cơ bản và được xác định như chế độ đồ họa phân giải thấp cộng với 8 mức cường độ sáng (Intensities) thông qua các bit cường độ sáng, để có tổng thể 16 màu, kể cả màu đen và trắng. Có 2 tập màu: Set 1 và Set 2:

Color Set 1	Color Set 2
Màu 1 là xanh lá cây (Green)	Màu 1 là xanh tím (Cyan)
Màu 2 là đỏ (Red)	Màu 2 là đỏ tươi (Magenta)
Màu 3 là nâu (Brown)	Màu 3 là trắng (White)

• Bộ nhớ hiển thị tổ chức thành 2 băng, mỗi băng 8000 byte như sau:

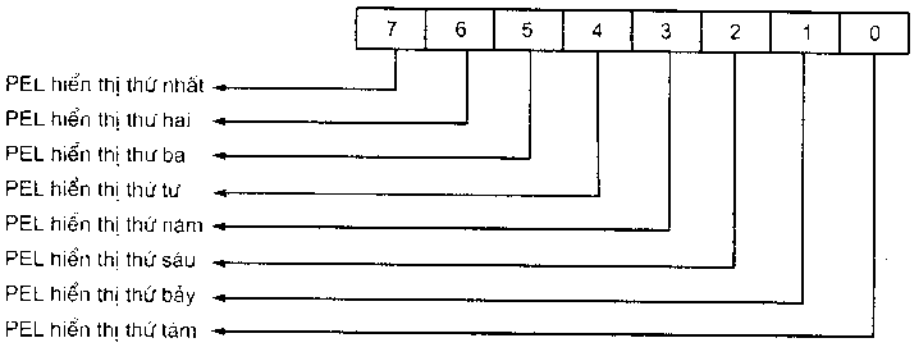


Địa chỉ B8000h chứa lệnh hiển thị PEL ở góc trái trên của vùng hiển thị.

Chế độ đồ họa đen trắng phân giải cao (640 × 200)

Chế độ đồ họa phân giải cao này dùng cho màn hình màu khi hiển thị đen trắng. Nó có những đặc điểm như sau:

- Chứa tối đa 200 hàng, mỗi hàng 640 PEL, mỗi PEL có kích thước: cao 1, rộng 1.
- Chỉ hiển thị ở màu đen trắng.
- Chiếm 16000 byte của bộ nhớ hiển thị trên bảng điều khiển CGA.
- Các thủ tục đánh địa chỉ và sắp xếp như ở chế độ màu/đồ họa phân giải trung bình, nhưng khuôn dạng dữ liệu khác. Trong chế độ này mỗi bit của bộ nhớ được sắp xếp cho từng PEL trên màn hình.
- Khuôn dạng 8 PEL cho 1 byte như sau:



Thuộc tính màu của ký tự hay của PEL được xác định bằng 4 bit của byte thuộc tính: I, R, G, B. Có tới 16 màu khác nhau có thể chọn lựa tùy theo giá trị các bit này. Bảng 5.28 liệt kê các tổ hợp chọn màu.

Bảng 5.28: Tổ hợp màu IRGB

I	R	G	B	Màu	I	R	G	B	Màu
0	0	0	0	Đen (Black)	1	0	0	0	Xám (Gray)
0	0	0	1	Xanh da trời (Blue)	1	0	0	1	Xánh da trời sáng (Light Blue)
0	0	1	0	Xanh lá cây (Green)	1	0	1	0	Xanh lá cây sáng (Light Green)
0	0	1	1	Xanh tím (Cyan)	1	0	1	1	Xanh tím sáng (Light Cyan)
0	1	0	0	Đỏ (Red)	1	1	0	0	Đỏ sáng (Light Red)
0	1	0	1	Đỏ tươi (Magenta)	1	1	0	1	Đỏ tươi sáng (Light Magenta)
0	1	1	0	Nâu (Brown)	1	1	1	0	Vàng (Yellow)
0	1	1	1	Trắng (White)	1	1	1	1	Trắng cường độ cao (High Intensity)

Các máy tính cá nhân cho phép truy nhập tới hệ thống hiển thị ở 3 mức:

- Mức vùng nhớ/cổng:

  Địa chỉ đầu của vùng nhớ hiển thị là 0B8000h hay 0A0000h

- Mức BIOS:

  INT 10h : mã chức năng 2: thiết lập vị trí con trỏ

  AH = 2h

DH = số dòng (0-79 nếu là chế độ 80×25)

DL = số cột (0-24)

BH = 0 (cho là trang 0)

INT 10h: mã chức năng 0Ch: thiết lập chế độ video

AH = 0h

AL = PEL value

BH = 0 (cho là trang 0)

CX = cột (tọa độ x)

DX = dòng (tọa độ y)

INT 10h: mã chức năng 0Eh: hiển thị một ký tự lên màn hình ở chế độ text

AH = 0Eh

AL = mã ký tự (ASCII)

BH = 0 (cho là trang 0)

- Mức DOS:

INT 21h: mã chức năng 2: hiển thị ký tự lên màn hình

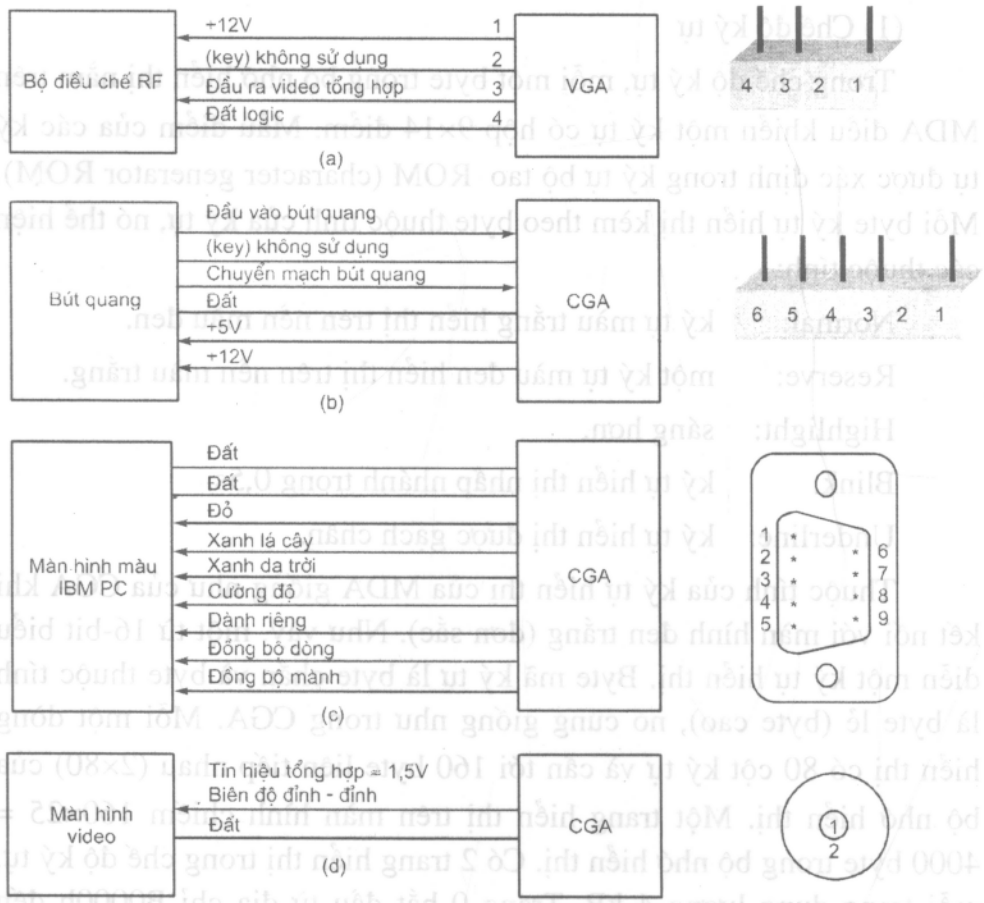
AH = 2

DL = mã ký tự (ASCII)

Các dịch vụ (chức năng) của BIOS và DOS cho phép chúng ta thực hiện được sự can thiệp trực tiếp đến bộ đệm hiển thị, thay đổi chế độ hiển thị, thay đổi màu sắc, kiểm tra trạng thái hiển thị, hiển thị ký tự, hình ảnh,... rất dễ dàng. Tuy nhiên, ở một số bảng ghép nối (như Hercules Graphics Card - Cạc đồ họa Hercules) không phải tất cả những khả năng thực hiện được.

(3) Định nghĩa các chân của đầu nối CGA

Bảng mạch CGA cắm trên khe mở rộng của IBM PC và có các đầu nối (hình 5.86) như sau:



Hình 5.86: Các đầu nối của CGA

• Phích cắm 4 pin Berp Strip để nối với thiết bị điều chế tần số radio (RF modulator) (hình 5.86 a)

• Phích cắm 6 pin Berp Strip để nối với bút quang (light pen) (hình 5.86 b).

• Phích cắm 9 chân D-Shell connector (hình 5.86 c).

• NTSC giắc cắm phono tín hiệu màu tổng hợp màu tương thích với NTSC (hình 5.86 d).

c) Bộ thích ứng hiển thị đồ họa đơn sắc MDA

MDA có 2 chế độ hiển thị: Chế độ ký tự (text mode): 80 cột × 25 hàng, chế độ đồ họa: 720 cột × 348 điểm.

(1) Chế độ ký tự

Trong chế độ ký tự, mỗi một byte trong bộ nhớ hiển thị nằm trên MDA điều khiển một ký tự có hộp 9×14 điểm. Mẫu điểm của các ký tự được xác định trong ký tự bộ tao ROM (character generator ROM). Mỗi byte ký tự hiển thị kèm theo byte thuộc tính của ký tự, nó thể hiện các thuộc tính:

Normal: ký tự màu trắng hiển thị trên nền màu đen.

Reserve: một ký tự màu đen hiển thị trên nền màu trắng.

Highlight: sáng hơn.

Blink: ký tự hiển thị nhấp nhánh trong 0,5s.

Underline: ký tự hiển thị được gạch chân.

Thuộc tính của ký tự hiển thị của MDA giống như của CGA khi kết nối với màn hình đen trắng (đơn sắc). Như vậy, một từ 16-bit biểu diễn một ký tự hiển thị. Byte mã ký tự là byte chẵn và byte thuộc tính là byte lẻ (byte cao), nó cũng giống như trong CGA. Mỗi một dòng hiển thị có 80 cột ký tự và cần tới 160 byte liên tiếp nhau (2×80) của bộ nhớ hiển thị. Một trang hiển thị trên màn hình chiếm 160×25 = 4000 byte trong bộ nhớ hiển thị. Có 2 trang hiển thị trong chế độ ký tự, mỗi trang dung lượng 4 kB. Trang 0 bắt đầu từ địa chỉ B0000h đến B0FFFh và trang 1 bắt đầu từ địa chỉ B800h đến B8FFFh. Bằng cách ghi giá trị "0" vào bit 7 ra cổng điều khiển hiển thị ở địa chỉ I/O 3B8h ta sẽ chọn trang 0 cho hiển thị và ghi giá trị "1" cho bit 7 cho cổng 3B8h sẽ chọn trang 1 cho hiển thị. Thuộc tính của ký tự hiển thị được điều khiển bằng byte thuộc tính như sau:

7	6	5	4	3	2	1	0	Thuộc tính
B	0	0	0	I	0	0	1	Gạch chân
B	0	0	0	I	1	1	1	Video bình thường
B	1	1	1	I	0	0	0	Video đảo

Trong đó: I = 0, độ sáng bình thường

I = 1, độ sáng tăng.

Nếu Blink Enable = 0:

B = 0, độ sáng của nền bình thường.

B = 1, độ sáng của nền tăng.

Nếu Blink Enable = 1:

B = 0, không nhấp nhánh.

B = 1, nhấp nháy.

Blink Enable được điều khiển bởi bit 5 của cổng điều khiển chế độ hiển thị (3B8h).

(2) Chế độ đồ họa

Trong chế độ đồ họa các bit trong bộ nhớ hiển thị điều khiển các điểm trên màn hình. Bộ nhớ hiển thị được tổ chức thành hai màn hình để có thể thay nhau hiển thị. Đặc tính này giúp cho kết hợp thành một mẫu trong khi mẫu khác được hiển thị, và phục vụ cho kỹ thuật hoạt hình.

Mỗi một dòng ngang 720 điểm được cất trong bộ nhớ hiển thị như là 90 byte liên tiếp nhau (vì 720:8, và một bit của bộ nhớ hiển thị tương ứng với một điểm trên màn hình), với 7 bit trái nhất trên màn hình. Bộ nhớ hiển thị chứa 2 trang hiển thị đầy đủ, mỗi trang có dung lượng 32 kB. Trang 0 bắt đầu từ B0000h đến B7FFFh và trang 1 bắt đầu từ B8000h đến BFFFFh. Mỗi một trang được chia thành 4 trường (field 0, 1, 2, 3), mỗi trường có dung lượng 8 kB liên tục nhau. Sự hiển thị lấy một dòng của trường 0, và sau đó đến các trường 1, 2, 3. Trình tự này tiếp tục cho đến hết toàn bộ màn hình sao cho các dòng kề cận nhau trong cùng một trường được hiển thị trên 4 dòng riêng biệt trên màn hình.

MDA sử dụng chip điều khiển màn hình CRT 6845 với các thanh ghi bên trong được truy nhập thông qua các địa chỉ I/O của CPU. Dải địa chỉ I/O của MDA phục vụ hiển thị màn hình như sau:

3B4h 6845 Index register (thanh ghi chỉ số của CRT 6845)

3B5h 6845 Data register (thanh ghi dữ liệu của CRT 6845)

3B8h Display mode control port (cổng điều khiển chế độ hiển thị)

3BAh Display status port (cổng trạng thái hiển thị)

Cổng điều khiển chế độ hiển thị (3B8h):

Bit 1: Graphics Enable (cho phép đồ họa)

Khi bit 1 bị xóa, thông tin hiển thị đi từ character generator ROM và logic giải mã thuộc tính. Các mã ký tự được đọc từ các Byte chẵn trong bộ nhớ hiển thị. Các mã thuộc tính tương ứng được đọc từ các Byte lẻ của bộ nhớ hiển thị.

Bit 3: Video Enable (cho phép video)

Bit này chốt cổng thông tin hiển thị đi tới màn hình.

Bit 5: Blink Enable (cho phép nhấp nháy)

Bit này được sử dụng chỉ khi Graphics Enable Bit bị xóa. Nó làm cho cho ký tự nhấp nhánh nếu bit 7 của byte thuộc tính của ký tự đó được thiết lập.

Bit 7: Page Select (chọn trang)

Bit này chọn trang hiển thị cho cả hai chế độ (ký tự và đồ họa).

Cổng trạng thái hiển thị (3BAH)

Bit 0: Đồng bộ dòng:

Bit này được thiết lập trong chu kỳ quét dòng ngược (horizontal retrace).

Bit 3: Đầu ra Video

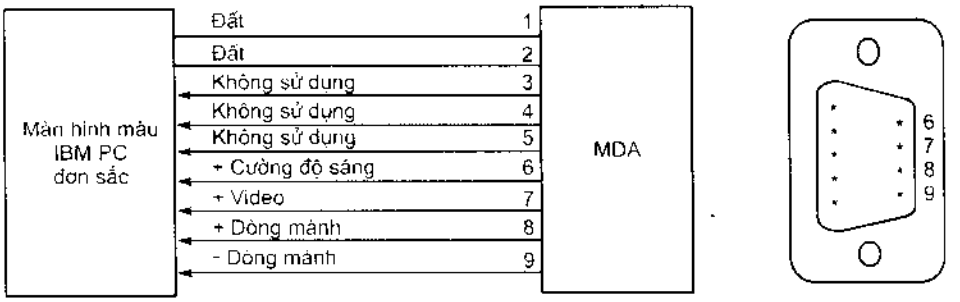
Bit này lặp lại tín hiệu video khi gửi ra màn hình.

Bit 7: Đồng bộ mành

Bit này bị xóa trong chu kỳ quét mành ngược (vertical retrace).

(3) Định nghĩa các chân tín hiệu của đầu nối MGA

Đầu nối của MDA giống như của CGA nhưng không có các tín hiệu màu, do vậy không dùng hết các chân (hình 5.87).



Hình 5.87: Đầu nối MDA

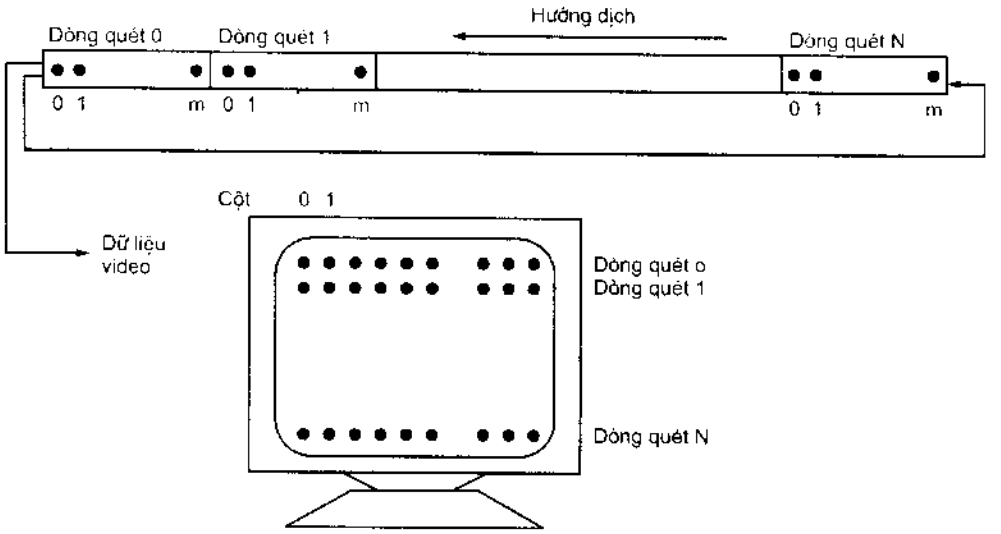
d) Các bảng mạch thích ứng đồ họa tiên tiến EGA (Enhanced Graphics Adapter) và mảng đồ họa video VGA (Video Graphics Array):

Chúng ta đã xét đến MDA và CGAvới các chế độ đồ họa và màu nghèo nàn, độ phân giải thấp không thỏa mãn được nhu cầu của nhiều ứng dụng cần có kỹ thuật đồ họa độ phân giải cao. Tiêu chuẩn EGA được phát triển vào năm 1984 đã phần nào giải quyết vấn đề này. Năm 1987 kỹ thuật VGA được IBM giới thiệu. Nó đã được cấy ngay trên bo mạch chu của các hệ thống IBM PS/2 models 50, 60, và 80. EGA và VGA được thực hiện bằng các chip vi mạch VLSI (Very Large Scale Integrated - Tích hợp mật độ cao). Các bảng mạch EGA và VGA cho IBM PC có tốc độ cao, dung lượng của bộ nhớ hiển thị lớn, gọn nhỏ và chi phí thấp. EGA và VGA có BIOS riêng trên bảng mạch. Kỹ thuật này là sự nâng cấp của EGA với những đặc tính mới:

- *EGA/VGA có bus 32 bit, do đó CPU có thể thực hiện trao đổi dữ liệu 32 bit (ghi, đọc) với EGA/VGA:*

Tốc độ của EGA/VGA tăng gấp 4 lần so với MDA và CGA. Người lập trình có thể ghi và đọc bộ nhớ hiển thị không cần phải có thời gian chờ các chu kỳ quét ngược của dòng ngang và đường dọc (điều này khác với MDA và CGA). CPU có thể thực hiện chuyển dữ liệu 32 bit ra bộ nhớ hiển thị bằng một lệnh đơn, nhưng cũng có thể thực hiện chuyển dữ liệu 32 bit từ một vùng nhớ hiển thị đến vùng nhớ hiển thị khác cũng bằng một lệnh đơn.

• *Bộ nhớ hiển thị*



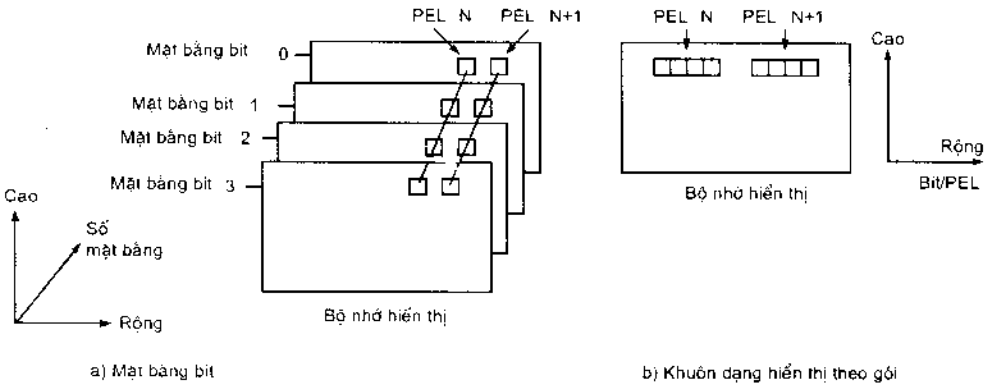
Hình 5.88: Bộ nhớ hiển thị như là thanh ghi dịch lớn

Bộ nhớ hiển thị có dung lượng đến 256 kB và được tổ chức khác nhau cho các phân giải khác nhau phụ thuộc vào chế độ hiển thị. Nó sử dụng khuôn dạng được đóng gói hay khuôn dạng mat băng bit (bit-plan). Có thể tổ chức bộ nhớ hiển thị thành một số trang phân giải thấp hoặc phân giải cao. Bộ nhớ hiển thị có thể được xem như một thanh ghi dịch lớn, thanh ghi này liên tục gửi một chuỗi dữ liệu liên tiếp nhau đưa lên hiển thị màn hình. Nó là sự hiển thị kiểu quét mành (raster scan display). Sở dĩ dùng chuỗi dữ liệu liên tục để hiển thị vì cần phải làm tươi màn hình bằng dữ liệu trong bộ nhớ hiển thị (hình 5.88). Thực sự thì không có thanh ghi dịch cụ thể nào, mà thay vào đó là CPU của hệ thống máy tính và cả logic điều khiển màn hình trên các bảng thích ứng phải thực hiện quá trình này bằng cách truy nhập trực tiếp vào bộ nhớ hiển thị.

• *Kỹ thuật mặt bằng bit và định khuôn dạng hiển thị theo gói (packed display format):*

Có hai cách để phân đoạn bộ nhớ hiển thị. Cách thứ nhất là phân đoạn bộ nhớ hiển thị thành các mặt bằng bit (bit-planes). Một bit-plane

chứa một 1 bit cho một phần tử ảnh PEL (pixel). Số lượng các bit-plane phải bằng số lượng tối đa các bit cho một PEL, bởi vì mỗi bit plane chỉ có 1 bit.. Kỹ thuật thứ hai là định dạng hiển thị theo gói, trong đó, chỉ có một mặt bằng của bộ nhớ, và mặt bằng này được phân chia thành các thành phần, mỗi thành phần là số bit cho một PEL. Cả hai cách này sẽ là giống nhau nếu chỉ gán một bit cho một PEL (hình 5.89).



Hình 5.89: Tổ chức bộ nhớ hiển thị theo Bit và khuôn dạng hiển thị theo gói

EGA/VGA sử dụng góc trái trên cùng của màn hình hiển thị như là địa chỉ thấp nhất của bộ nhớ hiển thị. Ký tự đầu tiên của trang hiển thị phải được hiển thị ở góc trái trên cùng của trang. Tương tự, trong chế độ đồ họa, nếu địa chỉ thấp nhất tương ứng với góc trái trên cùng thì khi vẽ một đường thẳng từ gia trị nhỏ đến giá trị lớn theo chiều đứng thì chiều vẽ sẽ đi từ trên xuống dưới như tự nhiên.

- *Sự sắp xếp dữ liệu từ bộ nhớ chính của máy tính (trên bo mạch chủ) đến bộ nhớ hiển thị (trên EGA/VGA):*

Coi bộ nhớ chính của PC là bộ nhớ hai chiều. Chiều rộng (Width) là số lượng bit của từng thành phần dữ liệu (từ, byte), và chiều dài (length) là số lượng các thành phần dữ liệu (hay dung lượng bộ nhớ). Nếu thành phần nhớ là byte thì chiều rộng chỉ có 8 bit, còn chiều dài có thể tính bằng 1 MB (dung lượng nhớ của 8088/8086).

Bộ nhớ hiển thị có thể coi là bộ nhớ 3 chiều: chiều cao (hight) là số lượng dòng (rows), chiều rộng (width) là số lượng cột (columns) và chiều sâu (depth) là số lượng bit cho một PEL). Như vậy cần phải phân đoạn bộ nhớ 2 chiều của PC sao cho nó sắp xếp vào được bộ nhớ hiển thị 3 chiều của EGA/VGA.

Nếu chỉ có 1 bit gán cho 1 PEL thì một byte (chiều rộng) của bộ nhớ chính xác định cho 8 PEL. Chiều dài của bộ nhớ chính của PC chứa dữ liệu hiển thị được xác định bằng công thức:

$$\text{Số lượng phần tử} = \frac{\text{Số lượng cột}}{\text{Số PEL/phần tử}} \times \text{Số lượng dòng} \qquad (5.1)$$

Bởi vì một phần tử (element) là một byte nên từ công thức tổng quát này ta tính được độ dài (hay dung lượng) của bộ nhớ chính dành cho hiển thị là:

$$\text{Số lượng byte} = \frac{\text{Số lượng cột}}{8} \times \text{Số lượng dòng} \qquad (5.2)$$

Sử dụng các lệnh vận chuyển chuỗi dữ liệu theo byte (MOVSB) và REP (lặp lại lệnh MOVSB cho kết khi nào CX = 0) với cặp thanh ghi địa chỉ nguồn DS:SI (địa chỉ đầu vùng nhớ của bộ nhớ chính chứa dữ liệu hiển thị) và cặp thanh ghi địa chỉ đích ES:DI (địa chỉ đầu của vùng hiển thị trong bộ nhớ hiển thị trên EGA/VGA), và thanh ghi CX chứa số lượng byte (số lượng thành phần hiển thị) có thể chuyển nhanh chóng mảng dữ liệu hiển thị ra màn hình. Đối với trường hợp bộ nhớ hiển thị tổ chức thành 4 bit mặt bằng thì cần phải có 4 mảng dữ liệu riêng biệt cho 4 bit mặt bằng tương ứng. Trong trường hợp bộ nhớ hiển thị có khuôn dạng hiển thị theo gói thì mảng dữ liệu trong bộ nhớ chính của PC phải chứa các byte được đóng gói. Trình tự chuyển dữ liệu từ bộ nhớ hiển thị của EGA/VGA về bộ nhớ chính của PC cũng tương tự.

Dữ liệu trong bộ nhớ hiển thị gọi là hình ảnh ảo (virtual image), dữ liệu hiển thị trên màn hình gọi là hình ảnh nhìn thấy (viewed image). Khi đó công thức (5.2) được biểu diễn như sau:

$$\text{Số lượng byte} = \frac{\text{Số lượng cột ảo}}{8} \times \text{Số lượng dòng ảo} \quad (5.3)$$

Khi số lượng dòng ảo (virtual rows) bằng chiều cao hiển thị và số lượng cột ảo (virtual columns) bằng chiều rộng hiển thị của màn hình thì hình ảnh ảo sẽ được nhìn thấy đủ kín trên màn hình hiển thị. Khi số lượng hàng và cột ảo nhỏ hơn chiều cao và chiều rộng hiển thị của màn hình thì toàn bộ hình ảnh ảo sẽ là hình nhìn thấy ở góc trái trên của màn hình.

Khái niệm hình ảnh ảo có thể ứng dụng cho xử lý cửa sổ. Một cửa sổ trên màn hình thường có kích thước nhỏ hơn kích thước hiển thị tối đa của màn hình. Tức là nó có chiều cao và chiều rộng nhỏ hơn kích thước hiển thị tối đa của màn hình. Có thể hiển thị nhiều cửa sổ trên màn hình. Nhưng cửa sổ không thể được chuyển từ bộ nhớ chính của PC đến bộ nhớ hiển thị của EGA/VGA bằng một lệnh MOVSB đơn, bởi vì cửa sổ không nằm trong vùng nhớ liên tục của bộ nhớ hiển thị.

Bảng 5.29 là sự so sánh các loại bảng mạch thích ứng về dung lượng của bộ nhớ hiển thị và BIOS phụ thuộc vào độ phân giải và số lượng màu. Nếu là MDA thì chỉ có 2 màu (ví dụ, đen và trắng, một màu cho nền màu khác cho chữ).

Bảng 5.29: Các loại bảng mạch thích ứng về dung lượng của bộ nhớ hiển thị và BIOS

Loại bảng	Phân giải theo PELs	Số lượng màu	Bộ nhớ hiển thị	BIOS
MDA	720 × 348	2/2	4 kB	Hệ thống
CGA	320 × 200	4/16	16 kB	Hệ thống
CGA	640 × 200	2/16	16 kB	Hệ thống
EGA	640 × 480	16/64	256 kB	Cục bộ
VGA	640 × 480	16/256	256 kB	Cục bộ
VGA	320 × 200	256/256	256 kB	Cục bộ

• *Chế độ hiển thị ký tự:*

Bộ nhớ hiển thị của EGA/VGA trong chế độ hiển thị được tổ chức tương thích với MDA và CGA. Nghĩa là, mỗi một ký tự được

biểu diễn trong bộ nhớ hiển thị bằng 2 Byte: byte mã ASCII của ký tự (byte chẵn) và byte thuộc tính của ký tự (byte lẻ). Trong bộ nhớ chính của máy tính (trên bo mạch chủ) các byte ký tự và thuộc tính được cất giữ trong các vùng nhớ liên tiếp nhau. Một chế độ đánh địa chỉ đặc biệt có trong EGA/VGA để sao cho các byte ký tự và thuộc tính xuất hiện liền nhau bên trong bộ nhớ hiển thị. Điều này tạo cho người lập trình sử dụng các lệnh vận chuyển chuỗi (move string) để chuyển nhanh dữ liệu hiển thị từ bộ nhớ chính của máy tính ra màn hình. Chúng ta có thể tính ngay được một trang hiển thị ở chế độ ký tự có dung lượng bao nhiêu, tùy thuộc vào dung lượng bộ nhớ hiển thị và chế độ phân giải. Nguyên tắc xác định dung lượng của trang màn hình cũng giống như trong MDA và CGA.

• *Chế độ đồ họa*

Trong chế độ đồ họa, bộ nhớ hiển thị được tổ chức sao cho một số chế độ phân giải phải giống như bộ nhớ hiển thị của CGA. Các chế độ hiển thị của EGA/VGA đã liệt kê trong bảng 5.22.

EGA/VGA có thể sử dụng các tập mẫu ký tự do người sử dụng tạo nên. Một tập mẫu ký tự có thể dài 256 ký tự. Mỗi một ký tự có thể có mẫu rộng 9 PEL và 32 dòng quét chiều cao.

EGA BIOS ROM (nằm trên EGA) có 3 tập mẫu ký tự: mẫu ký tự 8×8, 8×14 và 9×14 PEL. Tập mẫu ký tự 9×14 là tập mẫu bổ sung. Nó chỉ chứa các ký tự khác về hình dáng so với tập mẫu 8×14.

VGA BIOS ROM có 5 tập mẫu ký tự: các tập 8×8, 8×14, 9×14, 8×16, và 9×16. Các phông 9×14 và 9×16 là những phông bổ sung. Chúng chỉ chứa những mẫu dạng khác so với các tập mẫu 8x14 và 8×26.

Có tới 4 tập mẫu ký tự, mỗi tập có độ dài 256 ký tự. Cả 4 tập mẫu đều có thể đồng thời thường trú trong bộ nhớ hiển thị EGA, và bất kỳ 2 trong số 4 tập có thể được kích hoạt để tạo cho người lập trình một tập ký tự có độ dài 512 byte.

Có đến 8 tập mẫu ký tự, mỗi phông có độ dài 256 ký tự, có thể được nạp và đồng thời thường trú trong bộ nhớ hiển thị của VGA. Bất kỳ 2 trong số 8 tập ký tự có thể được đồng thời kích hoạt để tạo ra tập ký tự độ dài 512 ký tự cho người lập trình.

• *Các thanh ghi của các bộ thích ứng EGA/VGA*

Tất cả các chức năng đồ họa của EGA/VGA được điều khiển qua tập các thanh ghi. Những thanh ghi này được gán các địa chỉ cổng I/O của máy tính và được truy nhập qua các lệnh assembly IN XX, OUT XX (XX là địa chỉ I/O của các thanh ghi). Các thanh ghi này có độ dài 1 byte, và mỗi một byte được phân thành 1 đến 8 trường độc lập với nhau. Các chức năng đồ họa được điều khiển bằng các trường này. Để xác định địa chỉ đầu của vùng hiển thị cần phải nạp giá trị 16-bit vào các thanh ghi địa chỉ. Giá trị 16-bit địa chỉ tới 64 kB dung lượng nhớ. Như vậy cần phải có 2 thanh ghi để chứa giá trị địa chỉ 16-bit. Các thanh ghi của EGA/VGA được phân thành các nhóm liệt kê ở bảng 5.30.

Bảng 5.30: Các thanh ghi của EGA/VGA

Nhóm thanh ghi	Địa chỉ I/O của các thanh ghi (Hex)
Các thanh ghi chung	3BA hoặc 3DA, 3CA, 3C2, 3CC
Thanh ghi chuỗi	3C4, 3C5
Thanh ghi CRTC	3B4, 3B5, hoặc 3D4, 3D5
Thanh ghi đồ họa	3CE, 3CF
Thanh ghi thuộc tính	3CO, 3C1

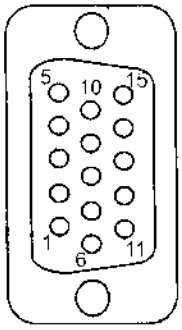
Trong EGA, cũng tương tự như trong CGA, hầu hết các thanh ghi đều chỉ ghi được/không đọc được, do đó hạn chế khả năng lập trình can thiệp hỏi đáp EGA/CGA. Nhưng trong VGA, tất cả các thanh ghi (ngoại trừ Flip-flop điều khiển thuộc tính) đều ghi/đọc được bằng lập trình. Tất cả các chức năng đồ họa cơ bản có thể được lập trình sử dụng các lệnh gọi (CALL) đến BIOS và chúng có thể truy nhập, thay đổi nội dung của các thanh ghi EGA/VGA.

• *EGA/VGA BIOS:*

BIOS thực hiện các chức năng hiển thị cao cấp và phức tạp hơn. Local BIOS được nạp vào bộ nhớ của máy tính từ địa chỉ C0000H đến C4000H. Nó chứa phần khởi động (được gọi ra trong khi khởi động hệ thống điều hành DOS bằng bootstrap), mã để điều khiển EGA/VGA và một số bộ tạo ký tự. Mã nguồn của EGA BIOS có thể xem trong các tài liệu kỹ thuật của IBM PC, nhưng cho VGA thì không có. Khi khởi động hệ thống máy tính, BIOS của hệ thống quét tìm xem có EGA hay không. Nếu có, thì thực hiện nhảy xa tới địa chỉ C0003H bằng lệnh gọi xa (FAR CALL). Đây là vùng bắt đầu của mã khởi tạo EGA BIOS. Ba byte đầu tiên bắt đầu từ C0000h, chứa nhận dạng. Tại địa chỉ C0003h chứa lệnh nhảy JMP SHORT tới chương trình EGA BIOS. Trong EGA/VGA BIOS sử dụng ngắt INT 10h với các thanh ghi như AH (chứa mã chức năng), AL, BX, DX, CL thực hiện xử lý hiển thị. Thanh ghi AH chứa mã chức năng từ 0 đến 13 cho EGA và từ 0 đến 1C cho VGA.

• *Định nghĩa các chân tín hiệu của đầu nối EGA/VGA*

Bảng 5.31: Các chân tín hiệu của đầu nối EGA/VGA

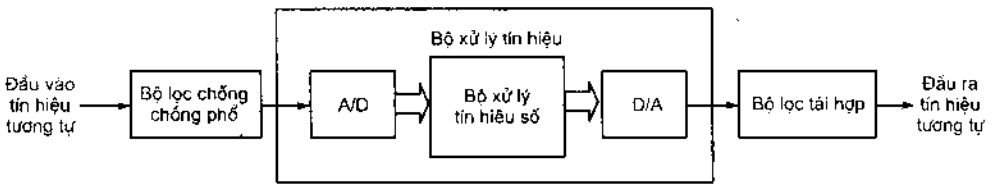
Pin N	EGA signals	VGA signals	VGA connector
1	Ground	Red	
2	S.Red	Green	
3	Red	Blue	
4	Green	Not used	
5	Blue	Ground	
6	S.Green	Red Ground	
7	S.Blue	Green Ground	
8	Đồng bộ dòng	Blue Ground	
9	Đồng bộ mành	Key (no pin)	
10		Sync Ground	
11		Ground	
12		Monitor ID	
13		Đồng bộ dòng	
14		Đồng bộ mành	
15		Không sử dụng	

Bảng EGA có đầu nối 9 chân để nối đến màn hình giống như của đầu nối CGA/MDA, nhưng định nghĩa các tín hiệu khác nhau. Bảng 5.3.1 VGA có đầu nối 15-pin để nối màn hình. Chân 12 của đầu nối CGA (bảng 5.31) sử dụng để xác định loại màn hình hiển thị. Trong một số loại VGA chân 12 và chân 15 được dùng cho DDC giao thức kênh hiển thị dữ liệu DDC (Display Data Channel) và DDC Clock để PC có thể tự động cấu hình và tối ưu các thiết lập của màn hình.

5.3.8. Phối ghép với thế giới tương tự

a) Hệ thống xử lý tín hiệu

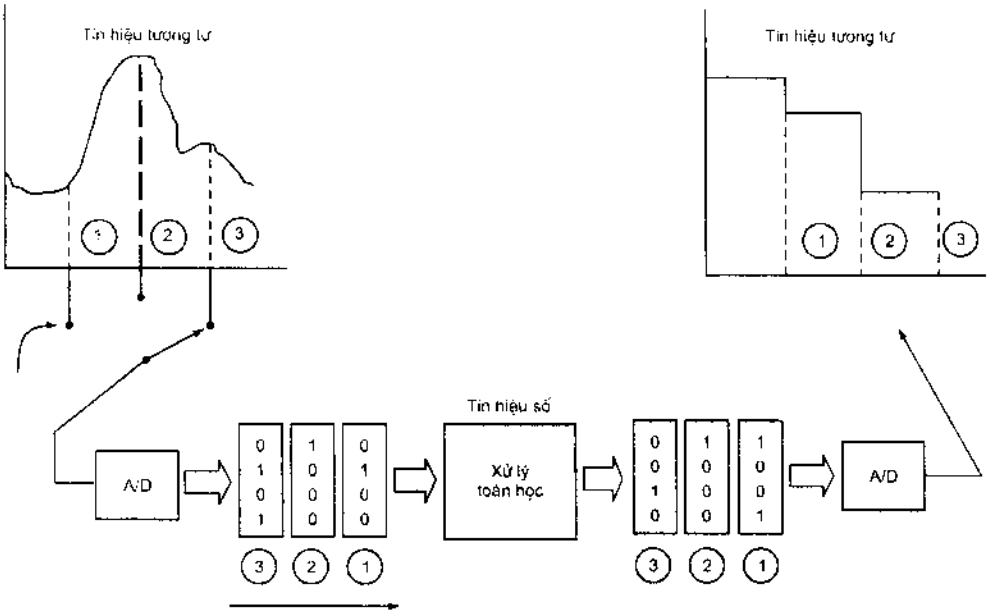
Hệ thống xử lý tín hiệu có sơ đồ khối cho trong hình 5.90, trong đó, bộ xử lý tín hiệu (signal processor) được chia thành 3 thành phần: phần đầu vào: là bộ chuyển đổi tín hiệu tương tự thành tín hiệu số (A/D), phần trung tâm: là bộ xử lý tín hiệu số DSP (Digital Signal Processor), và phần đầu ra: là bộ chuyển đổi tín hiệu số thành tín hiệu tương tự (D/A). Trong thực tế, 3 thành phần này được tích hợp chung trong một chip đơn. Nhưng cũng có thể áp dụng công nghệ máy vi tính trong trường hợp này bằng cách dùng một máy vi tính (IBM PC) làm chức năng của phần trung tâm: đó là bộ xử lý số, còn các phần đầu vào và phần đầu ra là các vi mạch ADC và DAC rời. Như vậy, với một máy vi tính, ta có thể đồng thời xử lý một số đầu vào tín hiệu khác nhau và đồng thời có các tín hiệu kết quả ở một số đầu ra.



Hình 5.90: Hệ thống xử lý tín hiệu

Chức năng của bộ lọc chống chồng phổ (antialiasing filter) của hệ thống xử lý tín hiệu là hạn chế hoặc giảm thiểu độ nhiễu của phổ gây ra khi tín hiệu tương tự được phân đoạn ra thành các điện thế rời rạc theo mẫu. Bộ lọc tái tạo ở đầu ra hệ thống xử lý tín hiệu (reconstruction

filter) làm phẳng lại tín hiệu ra thành tín hiệu tương tự. Hình 5.91 là hệ thống xử lý dữ liệu đã được lấy mẫu. Tín hiệu tương tự đưa vào được lấy mẫu ở từng đoạn đều đặn. Mỗi một điện thế lấy mẫu nhờ mạch A/D được bộ chuyển đổi thành tín hiệu số. Tín hiệu số được xử lý, rồi lại được chuyển đổi thành tín hiệu tương tự nhờ mạch D/A. Sản phẩm đầu ra là tín hiệu ở dạng sóng vuông.



Hình 5.91: Hệ thống xử lý tín hiệu được lấy mẫu

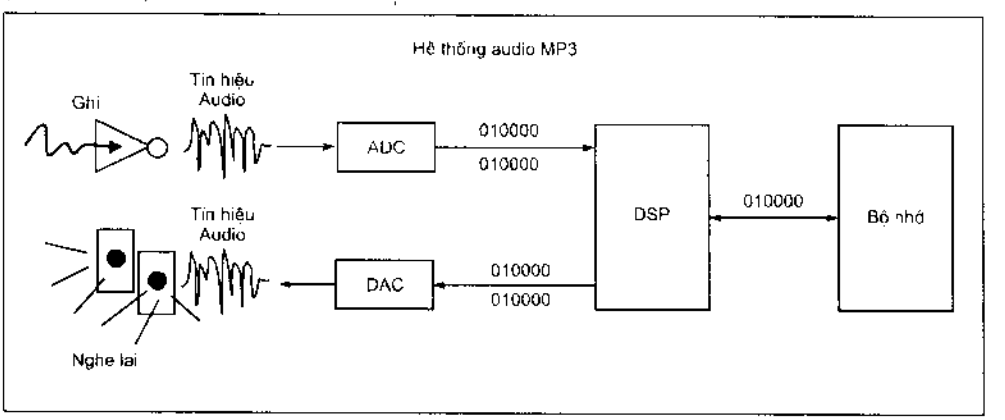
Quá trình xử lý toán học (math processing) tín hiệu số xảy ra trong bộ xử lý tín hiệu (signal processor) là quá trình vừa thông thường vừa đặc biệt. Thông thường là vì nó sử dụng các lệnh tính toán và các lệnh logic thông thường; đặc biệt là vì để đảm bảo tốc độ lấy mẫu không đổi thì không được phép thực hiện bất kỳ một rẽ nhánh có điều kiện nào. Cơ sở để xây dựng các mạch DAC và ADC là các mạch khuếch đại thuật toán.

Bộ xử lý tín hiệu bao gồm các mạch chuyển đổi tín hiệu tương tự thành tín hiệu số (ADC), mạch chuyển đổi tín hiệu số thành tín hiệu tương tự (DAC), và bộ xử lý số DP (Digital Processor).

b) Bộ xử lý tín hiệu số DSP (Digital Signal Processor)

Thế giới thực là thế giới các tín hiệu tương tự rất đa dạng trong thực tế, chúng có thể là các giá trị như : tiếng nói (voice), âm thanh (audio), hình ảnh (video), nhiệt độ, áp suất, độ ẩm môi trường không khí, tốc độ, khối lượng, trọng lượng vật thể hoặc chất nào đó, và chúng cần được kiểm soát, xác định giá trị trong quãng thời gian định trước hoặc được liên tục xử lý trong thời gian thực.

DSP được thiết kế chính là để thực hiện các hàm toán học thông thường trên các tín hiệu của thế giới thực đã được số hóa với tốc độ rất nhanh, vì vậy, để DSP nói chuyện với thế giới tín hiệu tương tự cần phải có ADC. ADC nhận tín hiệu tương tự của thế giới thực ở đầu vào và chuyển đổi thành tín hiệu số hóa ở đầu ra với chuỗi các giá trị 1 và 0. DSP nhận tín hiệu đã số hóa này và xử lý nó. Tín hiệu số hóa đã được DSP xử lý lại có thể đưa trở lại để dùng cho thế giới thực bằng cách: mạch DAC nhận tín hiệu đã số hóa từ DSP và chuyển đổi thành tín hiệu tương tự cho thế giới thực. Để minh họa cho khái niệm này ta hãy xét một DSP được sử dụng trong hệ thống thiết bị nghe audio MP3 (hình 5.92).



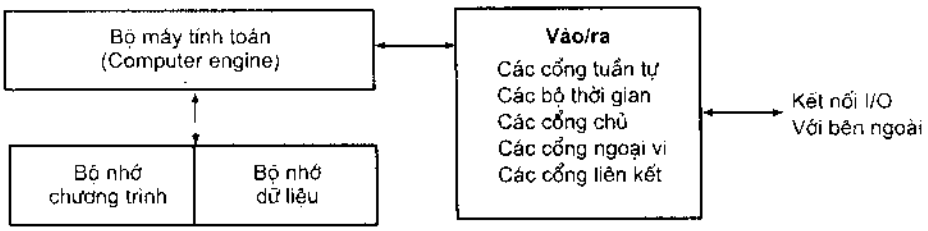
Hình 5.92: Xử lý tín hiệu Audio nhờ DSP

Trong giai đoạn ghi, tín hiệu audio được đưa vào đầu vào của bộ thu. Tín hiệu audio được chuyển đổi thành tín hiệu số nhờ mạch ADC và chuyển tới DSP. DSP thực hiện mã hóa theo MP3 và lưu giữ kết quả

mã hóa như là một tệp tin trong bộ nhớ. Trong giai đoạn nghe lại (playback), tệp tin được đọc từ bộ nhớ ra DSP và được DSP giải mã, sau đó lại được chuyển đổi trở lại thành tín hiệu tương tự nhờ mạch DAC. Tín hiệu tương tự đưa ra loa. Những DSP phức tạp thực hiện cả các chức năng khác như điều khiển âm lượng (volume), chất lượng âm thanh (equalization) và giao tiếp với người sử dụng.

Một máy tính số như hệ thống vi xử lý, PC có thể dùng làm DSP. DSP có thể là vi xử lý chuyên dụng có thể lập trình được. Thông tin của DSP được máy tính sử dụng để điều khiển các hệ thống an ninh, điện thoại, ca nhạc gia đình, và nén hình ảnh (video). Các tín hiệu có thể được nén sao cho chúng có thể truyền đi một cách nhanh chóng và hiệu quả từ nơi này đến nơi khác (ví dụ, kỹ thuật hội thoại từ xa có thể cho phép truyền tín hiệu tiếng nói và video qua đường dây điện thoại). Các tín hiệu có thể cũng được xử lý và nâng cấp để cải thiện chất lượng hoặc đảm bảo thông tin mà con người không cảm nhận được (ví dụ: xóa tiếng vọng đối với các cuộc gọi điện thoại hoặc đối với các hình ảnh y tế xử lý bằng máy tính công nghệ cao).

Chúng ta có thể tạo ra phần mềm riêng hoặc sử dụng phần mềm do nhà sản xuất để thiết kế một giải pháp DSP cho một ứng dụng xử lý tín hiệu nào đó. DSP được chế tạo như một đơn chip bằng công nghệ tích hợp cao.



Hình 5.93: Sơ đồ khối chức năng bên trong DSP

DSP gồm có các thành phần chính sau đây (hình 5.93):

• Bộ nhớ chương trình (program memory): thực hiện cất giữ các chương trình mà DSP sẽ sử dụng để xử lý dữ liệu.

• Bộ nhớ dữ liệu (data memory): cất giữ thông tin được xử lý

• Bộ máy tính toán (compute engine): thực hiện truy nhập chương trình trong bộ nhớ chương trình, dữ liệu trong bộ nhớ dữ liệu và xử lý toán học.

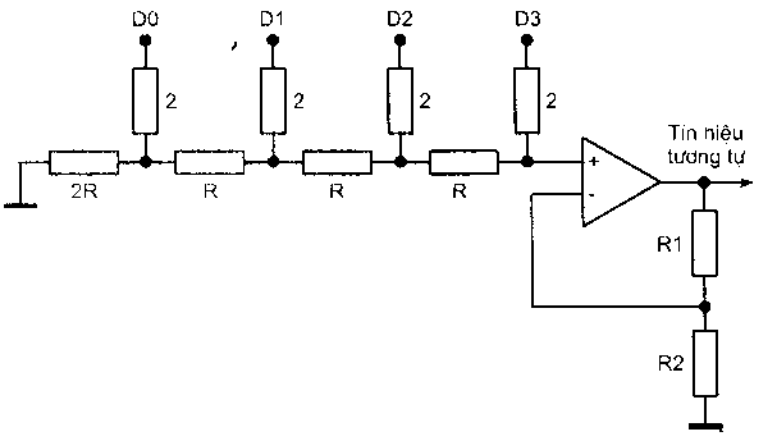
• Vào/ra (Input/Output): có một số loại giao tiếp cho thực hiện kết nối với các thiết bị bên ngoài.

Các bộ vi điều khiển (microcontroller) là các vi xử lý có thể ứng dụng để thiết kế các hệ thống xử lý các tín hiệu tương tự và số. Ví dụ Intel 8048, 8048, 8051, v.v..., mà chúng ta sẽ xét chúng ở chương 7.

c) Phối ghép vi xử lý với các mạch DAC và ADC

(1) Phối ghép với DAC

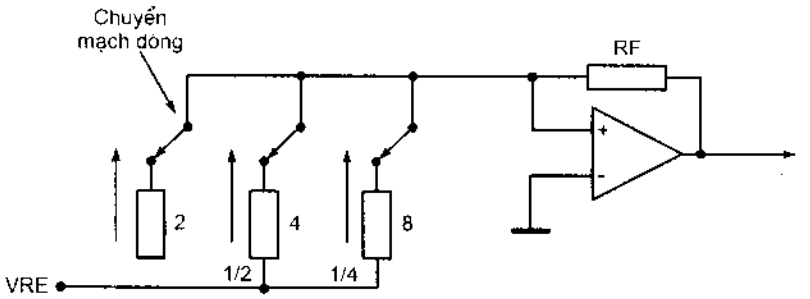
Mạch DAC đơn giản cho trong hình 5.94 là mạng bậc thang các điện trở vào, các điện trở vào có giá trị tỷ lệ tạo nên từng lối vào của khuếch đại thuật toán có trọng số 8-4-2-1 từ các bit dữ liệu vào song song D3-D0. Khuếch đại thuật toán chia độ và đệm cho tín hiệu ra tương tự.



Hình 5.94: DAC - mạch bậc thang điện trở (resister ladder)

Một mạch tương tự cũng đảm bảo các lối vào có trọng, đệm, và khuếch đại là mạch cộng dòng (hình 5.95). Dòng ở đây có trọng số

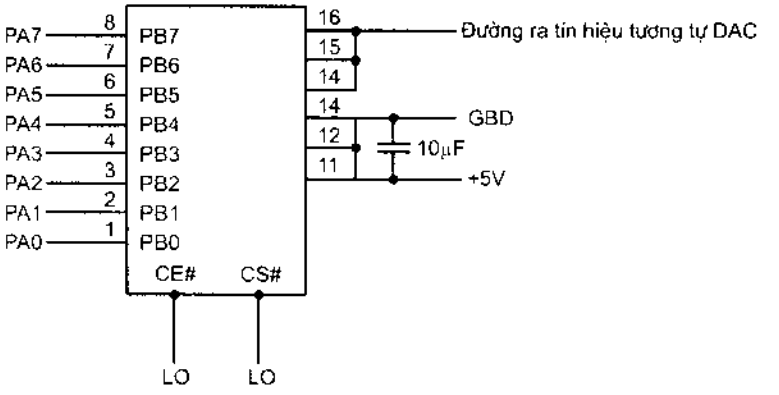
8-4-2-1 nhờ sự chọn lựa thích hợp các giá trị điện trở (bằng các nút chuyển mạch). Trở khách hồi tiếp âm đảm bảo hệ số chia cho tín hiệu ra.



Hình 5.95: DAC - Bộ cộng dòng dùng chuyển mạch dòng bằng cơ khí

Trong thực tế, các mạch DAC và ADC được chế tạo có các giao tiếp dữ liệu song song với bộ xử lý số do đó sự ghép nối giữa chúng với bộ xử lý rất dễ dàng.

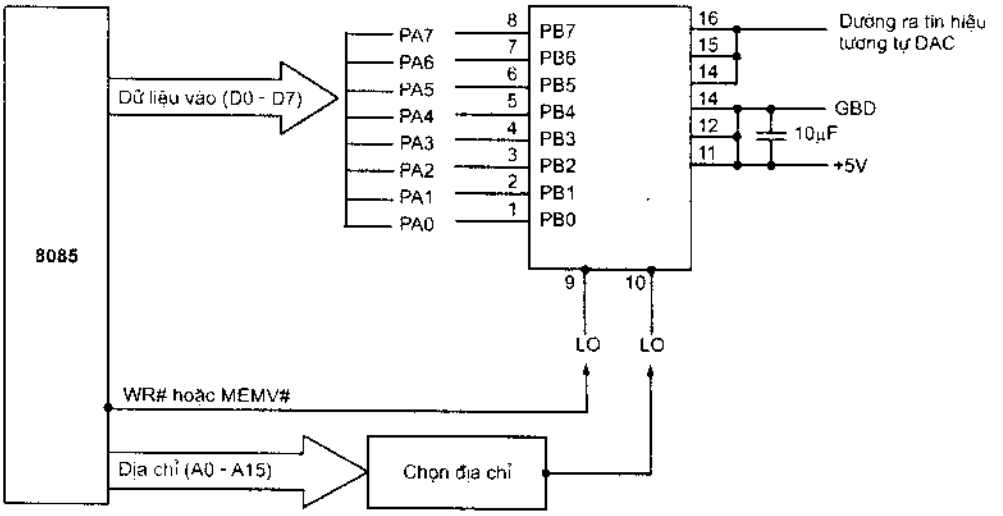
Một trong những mạch A/D thông dụng là AD558 (hình 5.96). AD558 là mạch DAC có 8 đường vào dữ liệu song song (PA0-PA7) nối với vi xử lý, 1 đường ra tín hiệu tương tự (DAC analog output), cổng vào chọn chip CS#, cổng vào cho phép ghi CE#.



Hình 5.96: vi mạch chuyển đổi số - tương tự AD558 DAC

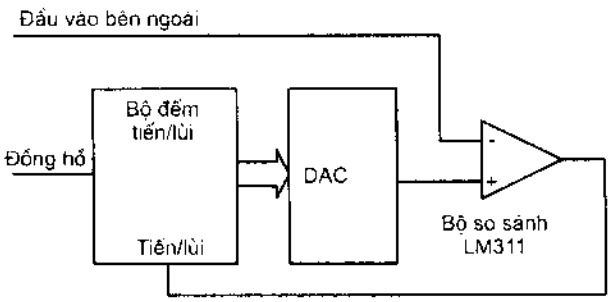
Các tín hiệu vào PA0-PA7 đi vào mạch chốt dữ liệu và đẩy tới mạch 8-bit bậc thang các điện trở và chuyển mạch điện áp bên trong

chip AD558 (8-bit resistor ladder & voltage switches). Bằng kỹ thuật chuyển mạch điện áp mạch AD558 cho phép sau khi nhận được một từ song song (PA7-PA0) thì sau 1μs có một giá trị tương đương ở đầu ra tương tự. Các chân 13, 14, 15 là các chọn lựa các điện trở hồi tiếp âm cho các hệ số khuếch đại điện áp tương ứng. Bằng chương trình điều khiển ghép nối đơn giản trong hệ thống vi xử lý ta có thể có một sơ đồ kết nối DAC AD558 với vi xử lý 8085 (hình 5.97).

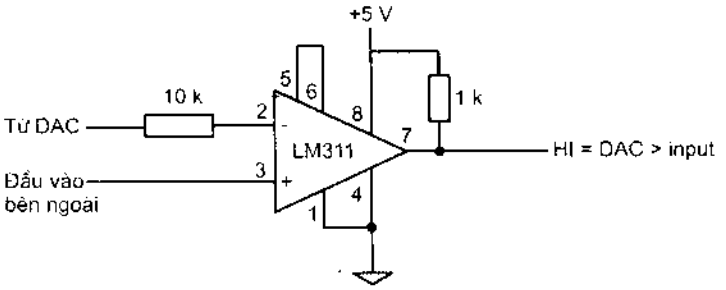


Hình 5.97: Phối ghép 8085 với AD558 DAC

Có thể lập trình trong hệ thống vi xử lý là một bộ đếm tiến/lùi (Up/down counter) (bộ đếm có thể là một vi mạch đếm) và sau khoảng thời gian đếm nào đó đưa giá trị số ra DAC AD558, giá trị đếm tương đương ở đầu ra DAC AD558 được so sánh với giá trị bên ngoài (external input). Kết quả so sánh được đưa trở lại bộ đếm để kiểm tra. Khi đó sơ đồ khối có thể như mô tả trong hình 5.98. Mạch so sánh có thể dùng vi mạch LM311 (hình 5.99). Giá trị ra của mạch so sánh là giá trị logic (0, 1) nên có thể kiểm soát nhờ chương trình được. Tất nhiên sự phối ghép của mạch so sánh với vi xử lý đòi hỏi phải mạch phụ trợ vào ra.



Hình 5.98: Kết nối bộ đếm với DAC AD558 và đưa ra so sánh.



Hình 5.99: vi mạch so sánh LM311

(2) Phối ghép với ADC

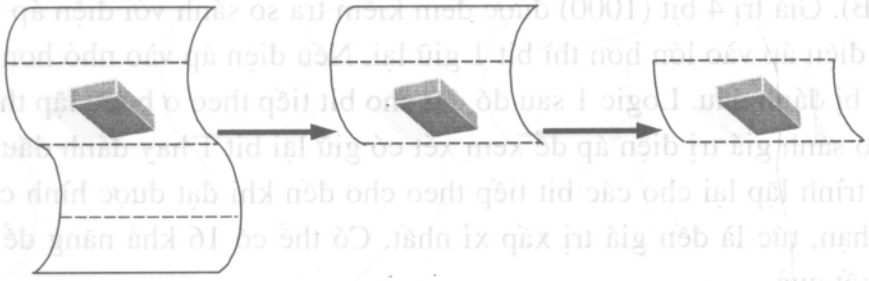
Chuyển đổi tín hiệu số sang tương tự và phối ghép giữa vi xử lý và DAC đơn giản hơn so với chuyển đổi tín hiệu từ tương tự sang số và phối ghép giữa vi xử lý và ADC.

Có một phương pháp thông dụng trong chuyển đổi tín hiệu từ tương tự sang số là phương pháp xấp xỉ liên tiếp. Phương pháp xấp xỉ liên tiếp được sử dụng phổ biến trong biến đổi tương tự-số là vì nó cho ra kết quả rất nhanh. Để hiểu rõ phương pháp này, ta hãy xét một ví dụ sau đây: có một kho báu được một trong bốn băng cướp chôn dấu trên một hòn đảo. Có thể xác định được băng cướp chiếm giữ kho báu bằng cách đặt ra 2 câu hỏi và giải đáp như sau:

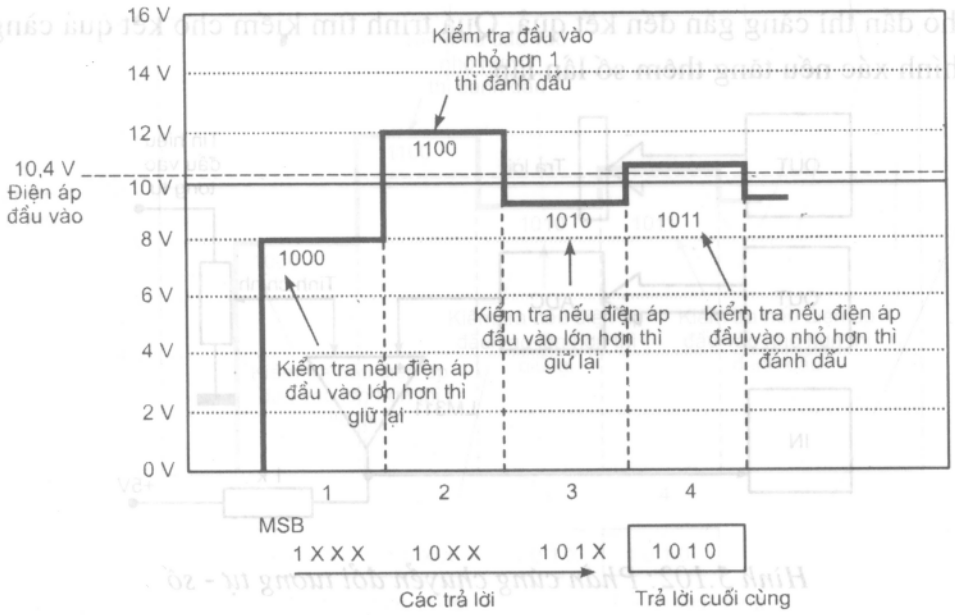
Câu hỏi 1: Kho báu chôn cất ở nửa phía Bắc hay nửa phía Nam đảo? Trả lời: ở nửa phía Bắc đảo.

Câu hỏi 2: Kho báu ở phần trên cao hay ở phần thấp của nửa phía Bắc đảo. Trả lời: ở phần thấp của nửa phía Bắc đảo.

Như thế ta xác định được đúng băng cướp bằng cách phân chia không gian đảo nơi cất dấu kho báu ra thành 2 nửa (hình 5.100).



Hình 5.100: Bài toán tìm kho báu trên đảo

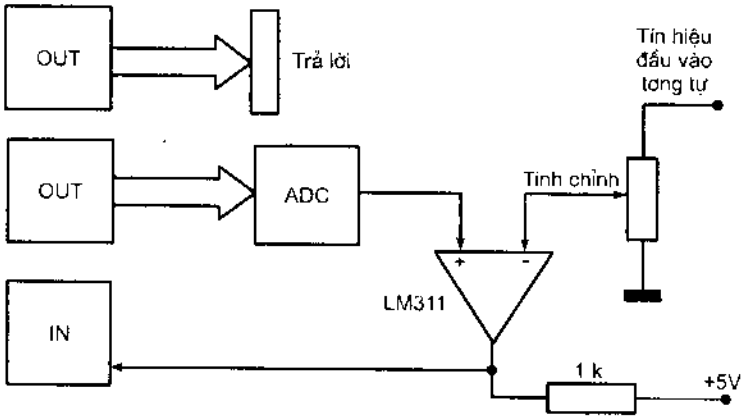


Hình 5.101: Phương pháp gần đúng liên tiếp trong biến đổi A/D giá trị điện áp vào tương tự 10,4V

Bây giờ xét máy tính sử dụng kỹ thuật này trong chính xác điện áp vào tương tự như thế nào. Cho rằng điện áp tương tự vào có giá trị 10,4V. Hình 5.101 mô tả từng bước lặp xấp xỉ giá trị điện áp tương tự bằng các bit tiếp theo, bắt đầu từ bit cao nhất (MSB). Để cho chính xác

cao, quá trình xấp xỉ dài 4 lần lặp. Phương pháp xấp xỉ liên tiếp là quá trình thử và lỗi. Trong bước đầu tiên, đặt giá trị logic 1 ở bit cao nhất (MSB). Giá trị 4 bit (1000) được đem kiểm tra so sánh với điện áp vào. Nếu điện áp vào lớn hơn thì bit 1 giữ lại. Nếu điện áp vào nhỏ hơn, thì bit 1 bị đánh dấu. Logic 1 sau đó đặt cho bit tiếp theo ở bước lặp thứ 2, rồi so sánh giá trị điện áp để xem xét có giữ lại bit 1 hay đánh dấu bỏ. Quá trình lặp lại cho các bit tiếp theo cho đến khi đạt được hình chóp giới hạn, tức là đến giá trị xấp xỉ nhất. Có thể có 16 khả năng để đạt đến kết quả.

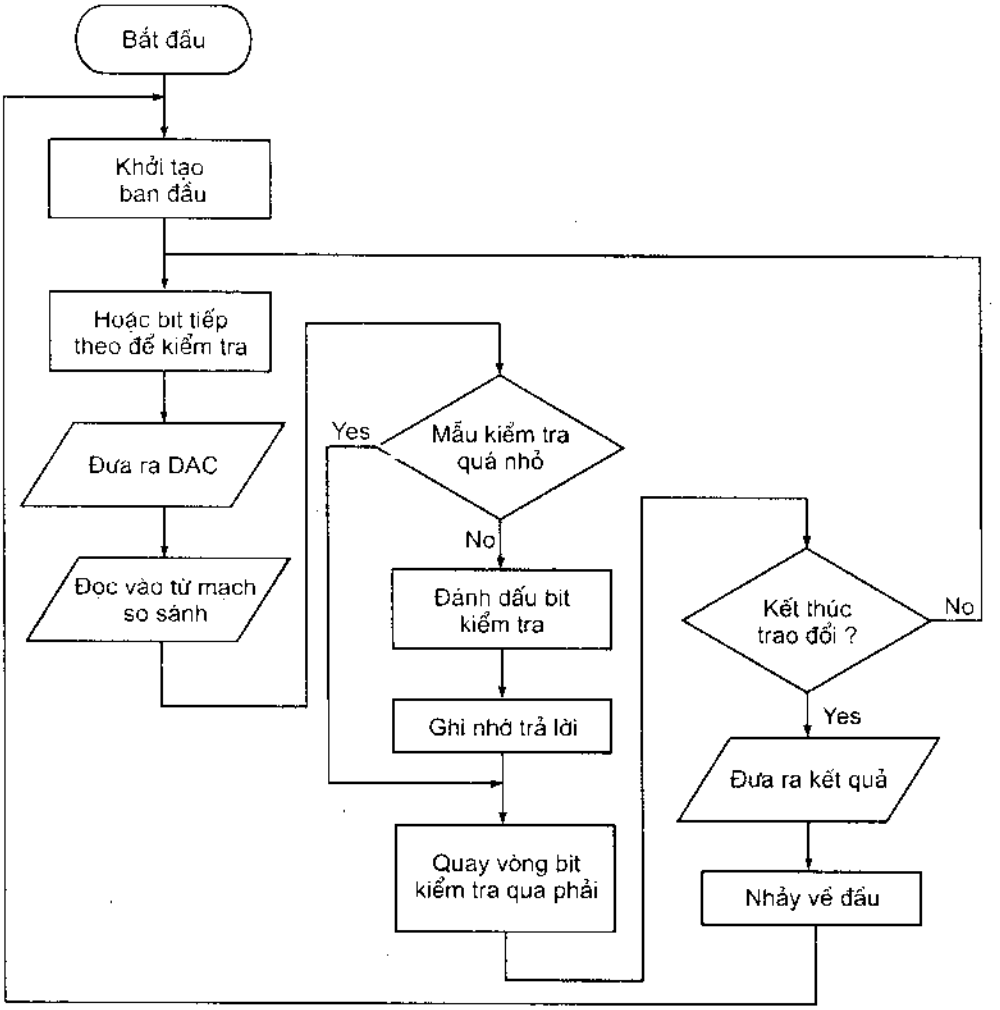
Cũng giống như bài toán tìm kho báu, giá trị điện áp vào ban đầu được chia thành 2 nửa trên và dưới của đường kết quả đúng. Càng chia nhỏ dần thì càng gần đến kết quả. Quá trình tìm kiếm cho kết quả càng chính xác nếu tăng thêm số lần lặp.



Hình 5.102: Phần cứng chuyển đổi tương tự - số

Như vậy phần cứng của chuyển đổi A/D và kiểm tra so sánh cần phải có DAC gửi các giá trị điện áp để kiểm tra và mạch so sánh để xác định điện áp vào lớn hơn hay nhỏ hơn điện áp vào (hình 5.102).

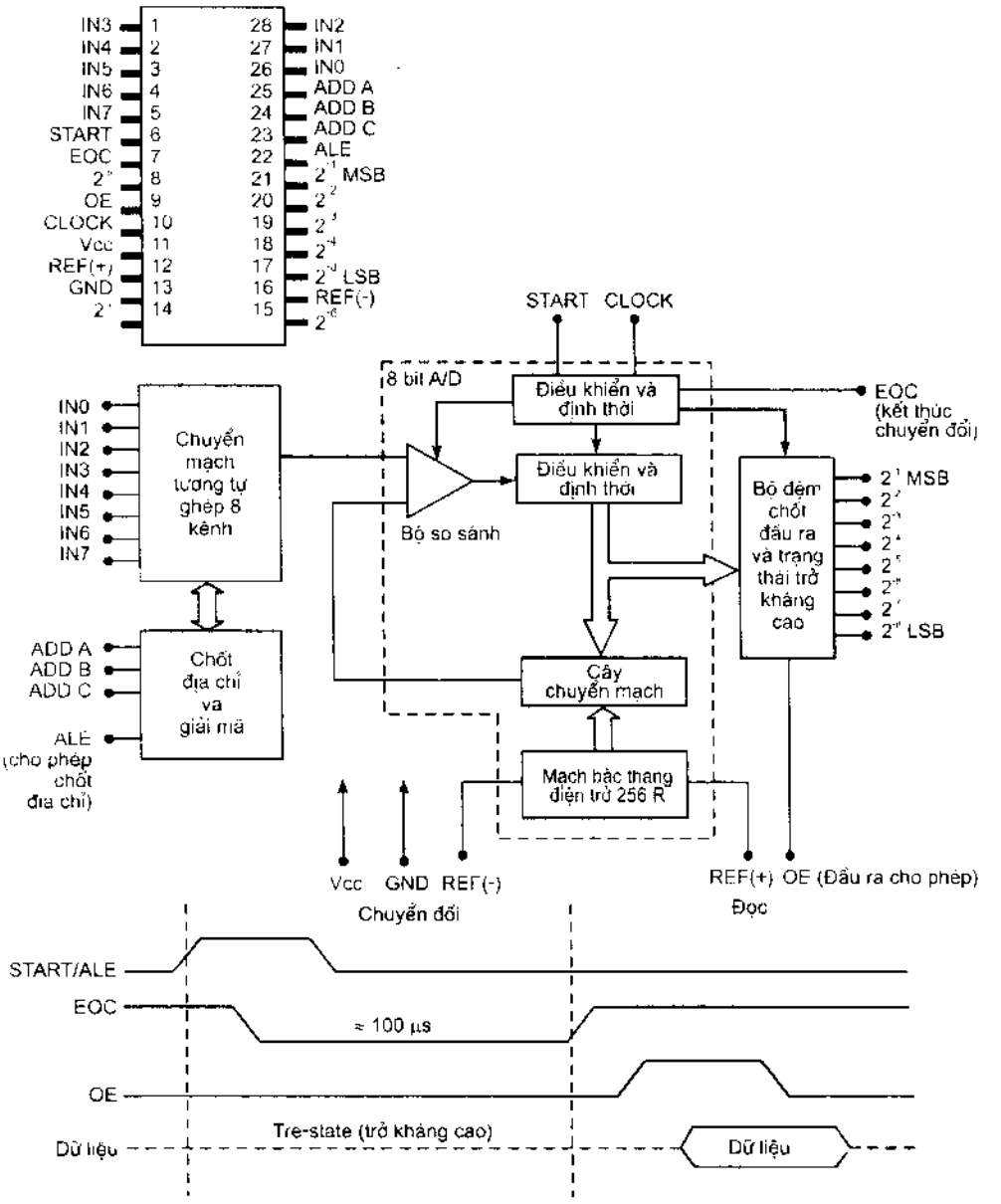
Chương trình (hình 5.103) gửi ra DAC qua bus dữ liệu (D0-D7) một chuỗi các điện áp kiểm tra và nhận về từ mạch so sánh (LM311) kết quả kiểm tra. Quá trình trao đổi tương tự - số kết thúc khi số lượng bước lặp khởi tạo ban đầu thực hiện xong.



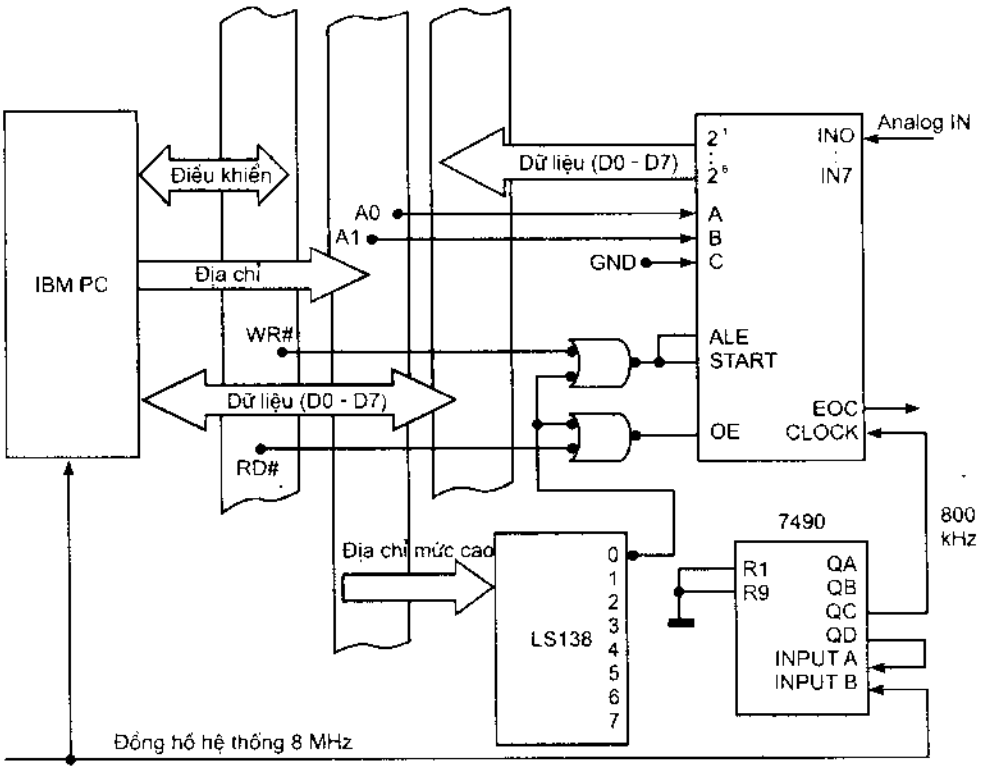
Hình 5.103: Lưu đồ thuật toán chương trình đưa ra DAC mẫu kiểm tra điện áp

Khác với các DAC, các vi mạch ADC có các đường ra dữ liệu song song phối ghép với vi xử lý và đường tín hiệu vào tín hiệu tương tự. Ví dụ vi mạch ADC0808 (hình 5.104), thực hiện toàn bộ sự trao đổi tương tự - số trong 100 μs - tương ứng với tốc độ lấy mẫu 10 kHz. Nó có một bộ ghép 8 kênh vào tương tự và 8-bit dữ liệu ra. Toàn bộ quá trình trao đổi A/D của ADC0808 được thực hiện theo 2 bước: chuyển đổi A/D (Conversion) và vi xử lý đọc (READ) giá trị số hóa được

chuyển đổi. Chu kỳ thời gian cả hai bước này được mô tả trong hình 5.104. Hình 105 là sơ đồ phối ghép ADC0808 như là RAM, ROM, hoặc bộ nhớ chậm với IBM PC qua bus hệ thống.



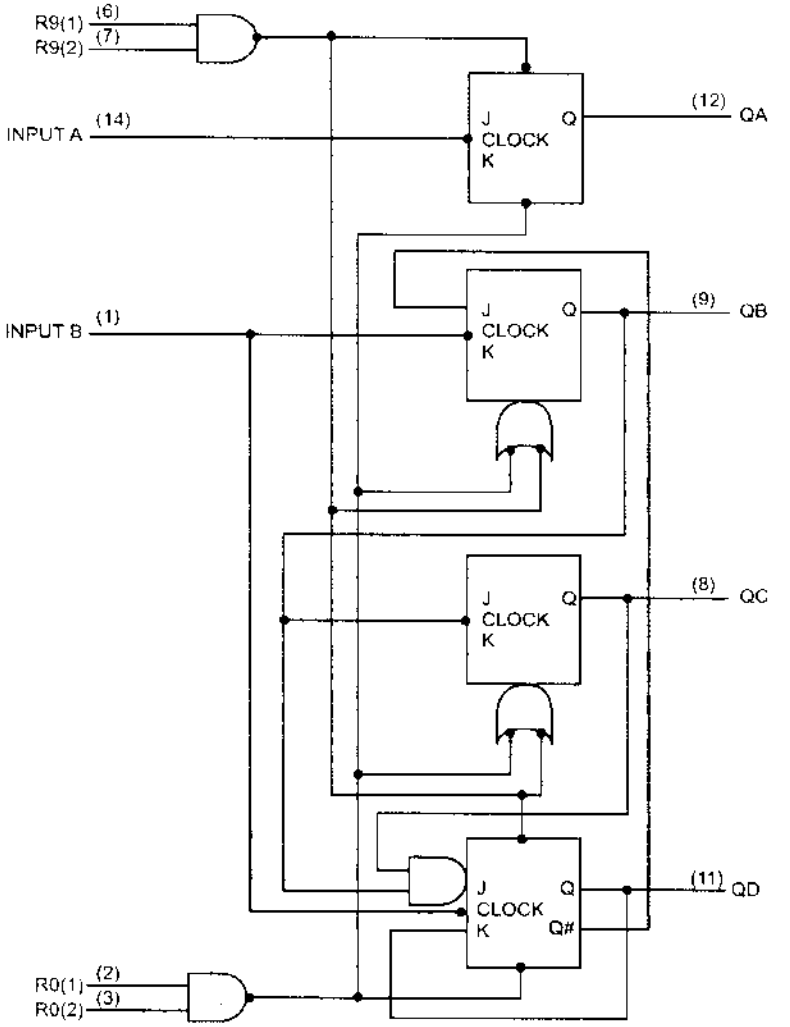
Hình 5.104: Đóng vỏ, các tín hiệu, sơ đồ khối, và đồ thị thời gian trao đổi A/D và đọc của ADC0808



Hình 5.105: Phối ghép ADC0808 với IBM PC qua bus hệ thống

Bước chuyển đổi A/D được bắt đầu bằng thao tác ghi vào ADC 0808 giá trị tương tự ở 1 trong 8 kênh vào tương tự (IN0-IN7). Địa chỉ của kênh vào tương tự được chọn qua các đường địa chỉ (ADD A, ADD B, ADD C) đưa ra từ vi xử lý (A0, A1, A2). Mạch giải mã LS138 (hay 8205) tạo ra tín hiệu chọn các cổng vào/ra của ADC0808 từ các đường địa chỉ cao của vi xử lý. Vùng địa chỉ vào/ra của ADC0808 phải là duy nhất trong hệ thống các địa chỉ vào/ra của IBM PC. Tín hiệu START/ALE (được tạo ra từ xung ghi WR# của vi xử lý) chuyển lên mức logic 1 để chốt địa chỉ chọn và bắt đầu ghi nhận giá trị tương tự. Tín hiệu EOC (end of conversion - kết thúc chuyển đổi) chuyển xuống mức logic thấp. Sau khoảng 100 µs (phụ thuộc vào nhịp đồng hồ CLOCK), EOC chuyển lên mức cao để kết thúc bước chuyển đổi A/D. Bước vi xử lý đọc kết quả số hóa được thực khi từ vi xử lý đưa tín hiệu đọc (khi ADC0808 được chọn) để tạo OE chuyển lên mức logic 1.

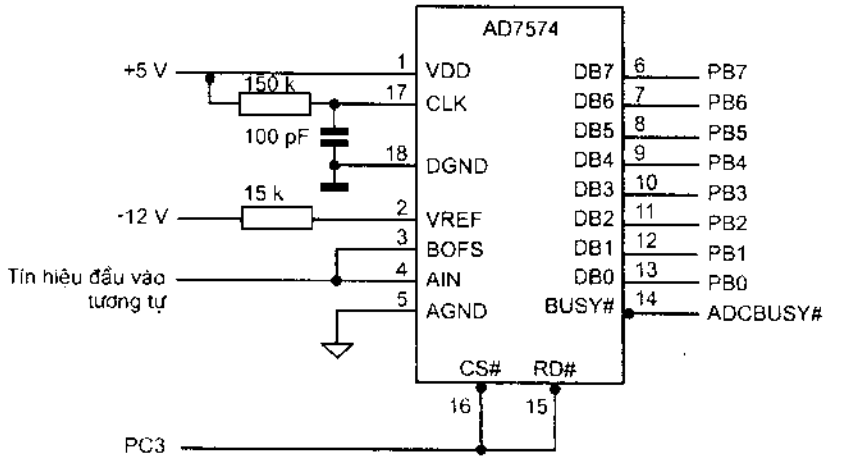
Xung OE kéo dài theo độ rộng của xung đọc của vi xử lý (RD#) và đủ để giá trị số hóa ở các lối ra 2^{-1} ... 2^{-8} đưa về vi xử lý qua bus dữ liệu. ADC0808 làm việc với đồng hồ khoảng 800 kHz, và có thể được tạo ra từ nhịp đồng hồ của hệ thống vi xử lý. IBM PC/XT 8086/8088 với nhịp đồng hồ hệ thống là 8 MHz, thì có thể dùng mạch đếm nhị phân để chia tần số nhỏ hơn, ví dụ mạch đếm nhị phân và thập phân 7490. Mạch 7490 có 4 đầu ra QA, QB, QC, QD, trong đó QA là bit thấp nhất. Có 2 đầu vào nhịp đếm INPUT A, INPUT B, 2 đầu vào xóa bộ đếm R0(1), R0(2), 2 đầu vào xóa 9: R9(1), R9(2) (hình 5.106).



Hình 5.106: Bộ đếm nhị phân và decade 7490

Với các giá trị logic = 0 của các tín hiệu xóa thì là 7574 làm việc như bộ đếm nhị phân các xung nhịp đồng hồ của hệ thống. Đầu vào nhịp đếm là INPUT B. Đầu vào nhịp đếm INPUT A nối với đầu ra của bit cao nhất (QD).

Vi mạch ADC7574 cũng thông dụng, đơn giản, tốc độ chuyển đổi A/D nhanh trong khoảng 15 µs, và phối ghép với vi xử lý như là RAM/ROM hay bộ nhớ chậm (hình 5.107). Vi mạch đóng vỏ DIP 18-pin và tiêu thụ nguồn thấp: +5V, 30 mW.



Hình 5.107: ADC7574

Các tín hiệu vào chọn chip CS# và đọc RD# được tạo ra từ giải mã các đường địa chỉ của bus hệ thống vi xử lý và từ WRITE#/READ# của vi xử lý. Quá trình chuyển A/D bắt đầu bằng vi xử lý đưa ra xung ghi bộ nhớ MEMW# = 0 (hay IO/M# = 0 và WR# = 0), nó tạo xung CS# = 0 để chọn ghi tín hiệu tương tự vào trong ADC7574. Vi xử lý đọc kết quả chuyển đổi từ ADC7574 bằng xung đọc bộ nhớ MEMR# = 0 (hay IO/M# = 0 và RD# = 0) từ các đường ra dữ liệu PB0-PB7 của AD7574. Tín hiệu BUSY# để làm trễ sự bắt đầu của bước đọc dữ liệu. BUSY# = 0 báo cho vi xử lý biết rằng AD7574 chưa sẵn sàng kết quả

chuyển đổi để đọc trong khi đang chuyển đổi A/D. BUSY# chuyển lên mức logic 1 để báo cho vi xử lý biết rằng bước chuyển đổi đã kết thúc và đã sẵn sàng kết quả chuyển đổi để đọc. Chương trình trong hệ thống vi xử lý phải kiểm tra mức logic của BUSY#. Kết thúc đọc, BUSY# trở về mức 0 cho lần ghi nhận tín hiệu tương tự tiếp theo vào AD7574.

Khi phối ghép ADC trong hệ thống vi xử lý như là bộ nhớ chậm thì quá trình tạo ra trạng thái chờ, ví dụ của 8085, 8086/8088 sẽ đơn giản (chúng ta đã xét vấn đề này). Cũng có thể làm chậm bằng cách đưa ra một số lệnh NOP hoặc tạo một chu trình đếm trong chương trình điều khiển với thời gian đủ vượt quá thời gian chuyển đổi A/D của ADC trước khi đưa ra lệnh đọc kết quả chuyển đổi từ ADC.

5.3.9. Một số phối ghép khác

Các mạch phụ trợ lập trình cho phép vi xử lý phối ghép với nhiều loại thiết bị ngoài vi có các kiểu vào/ra dữ liệu khác nhau như song song, nối tiếp, ngắt, thăm dò, truy nhập trực tiếp bộ nhớ. Một hệ thống vi xử lý phức tạp phải có các mạch định thời gian để điều khiển quá trình vào/ra dữ liệu giữa vi xử lý và các thiết bị ngoài. Do vậy, ta xét qua một số phối ghép khác thông qua khảo sát một số mạch phụ trợ lập trình.

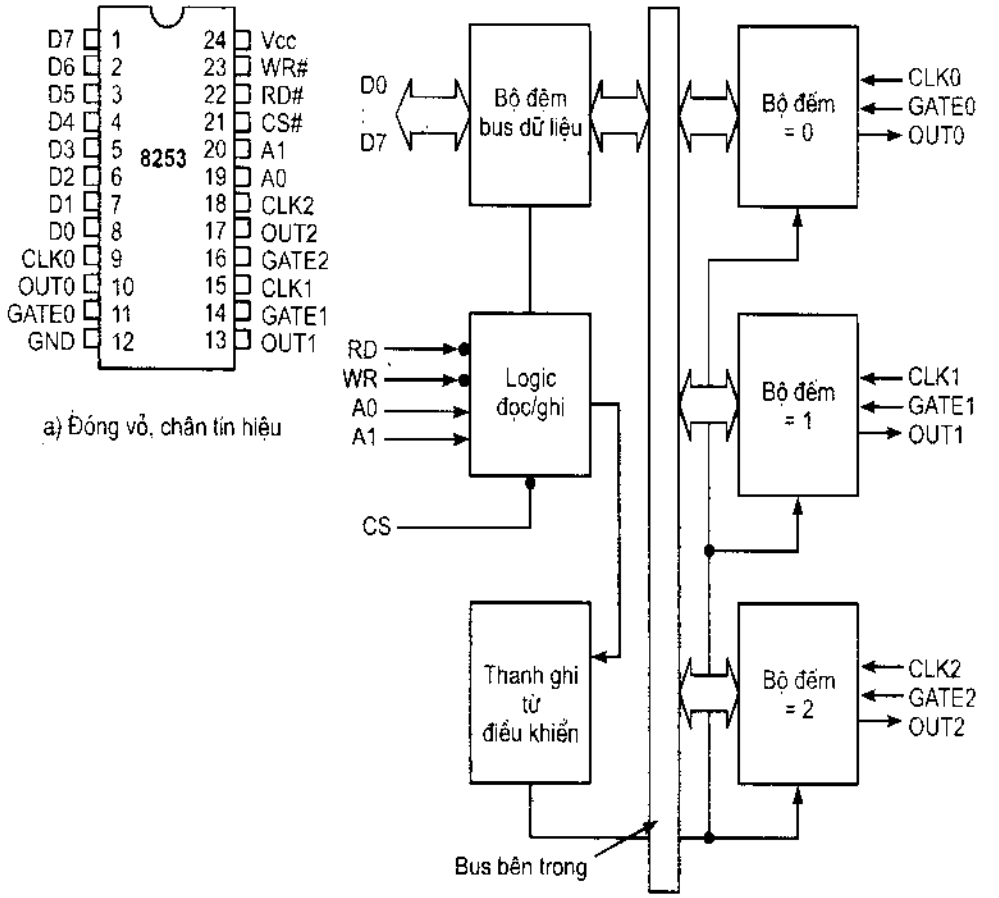
a) Phối ghép vi mạch lập trình thời gian/bộ đếm 8253 với vi xử lý

8253 là bộ lập trình thời gian dùng cho các bộ vi xử lý Intel 8-bit như 8080, 8085, 8086/8088. Đóng vỏ, các chân tín hiệu và sơ đồ khối chức năng bên trong 8253 cho ở hình 5.108.

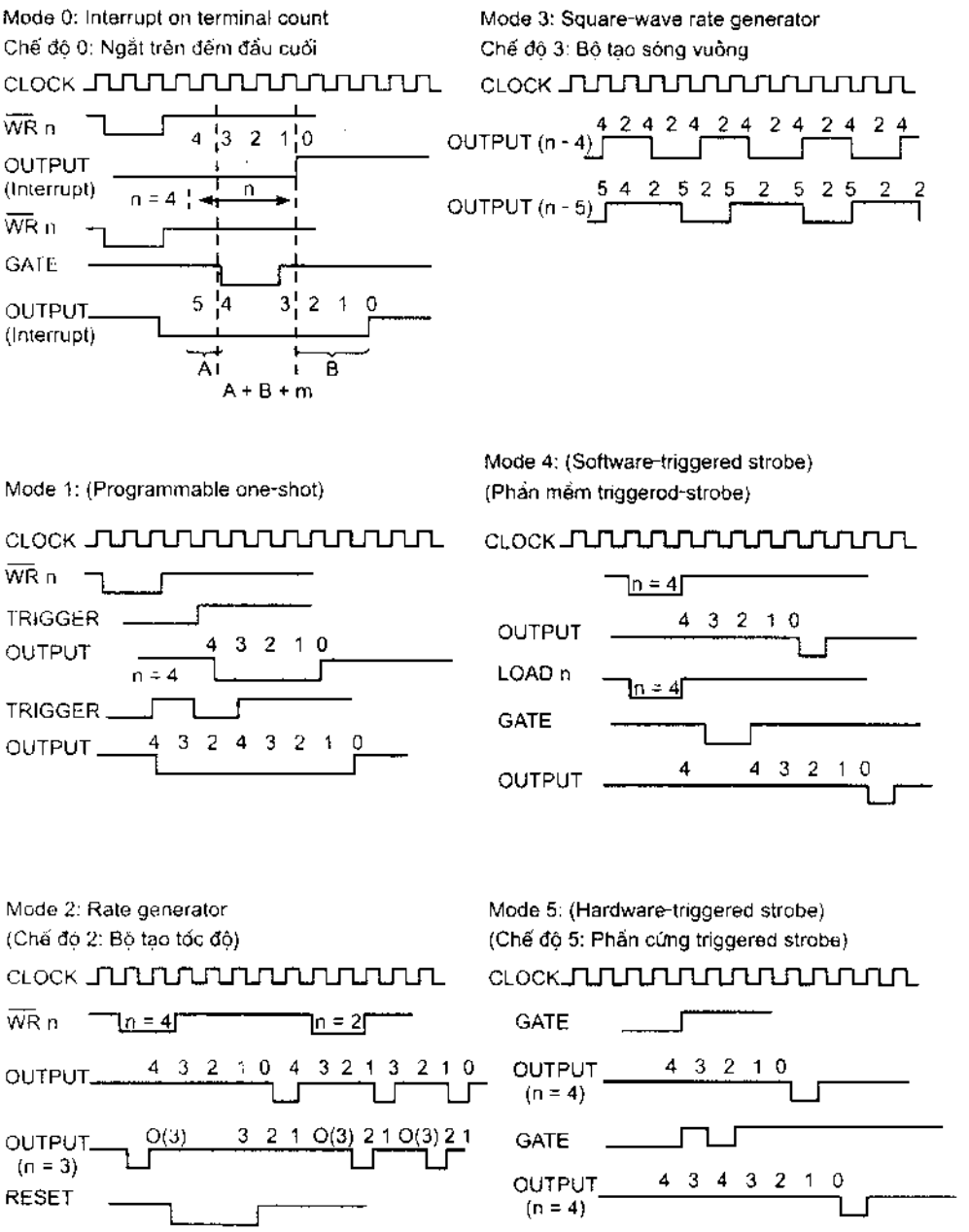
8253 có 3 bộ đếm/thời gian 16-bit và 3 thanh ghi điều khiển để thiết lập chế độ khai thác cho 3 bộ đếm/thời gian tương ứng. Mỗi một bộ đếm/thời gian 16-bit có thể được lập trình độc lập với 6 chế độ khai thác như mô tả trong hình 5.109.

Chế độ khai thác cho từng bộ đếm thời gian và giá trị cần phải thiết lập trong từng bộ đếm được xác định bằng cách lập trình qua hai

bước: ghi từ điều khiển chế độ vào thanh ghi điều khiển chế độ và ghi giá trị định thời gian vào trong thanh ghi bên trong được chọn. Để cấu hình tất cả 3 bộ thời gian, phải cần đến cả 6 chế độ lập trình khai thác.

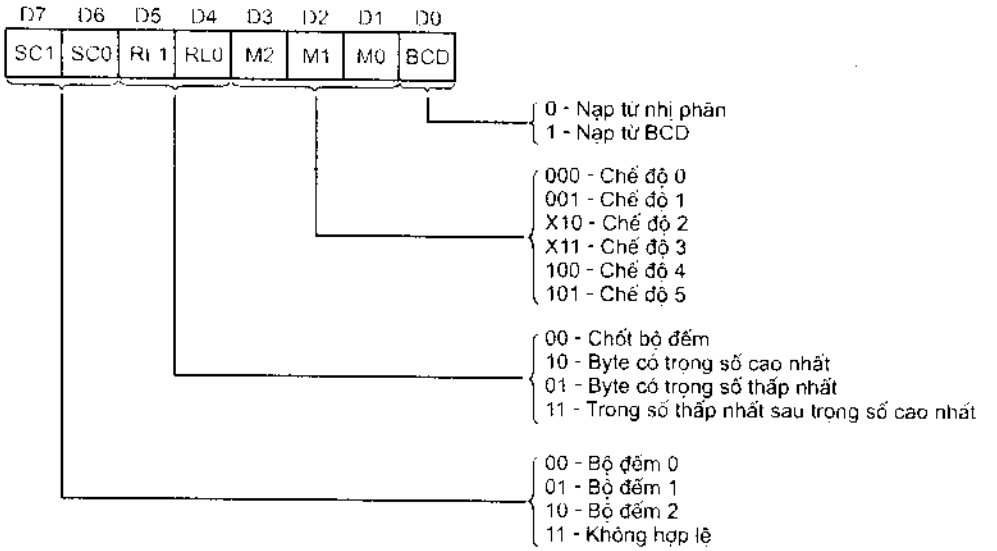


Hình 5.108: Đóng vỏ, chân tín hiệu và sơ đồ khối chức năng của 8253



Hình 5.109: Các đồ thị thời gian các chế độ khai thác 8253

Khuôn dạng từ điều khiển chế độ nạp trong thanh ghi điều khiển chế độ của 8253 như sau hình 5.110.



Hình 5.110: Lập trình điều khiển chế độ 8253

Mỗi một bộ đếm có 2 tín hiệu vào (CLK và GATE) và 1 tín hiệu ra (OUT). Tín hiệu GATE = "1" cho phép thực hiện quá trình đếm, và GATE = "0" cấm đếm. Nhịp đồng hồ (CLOCK) cho 8253 được lấy từ nhịp đồng hồ hệ thống.

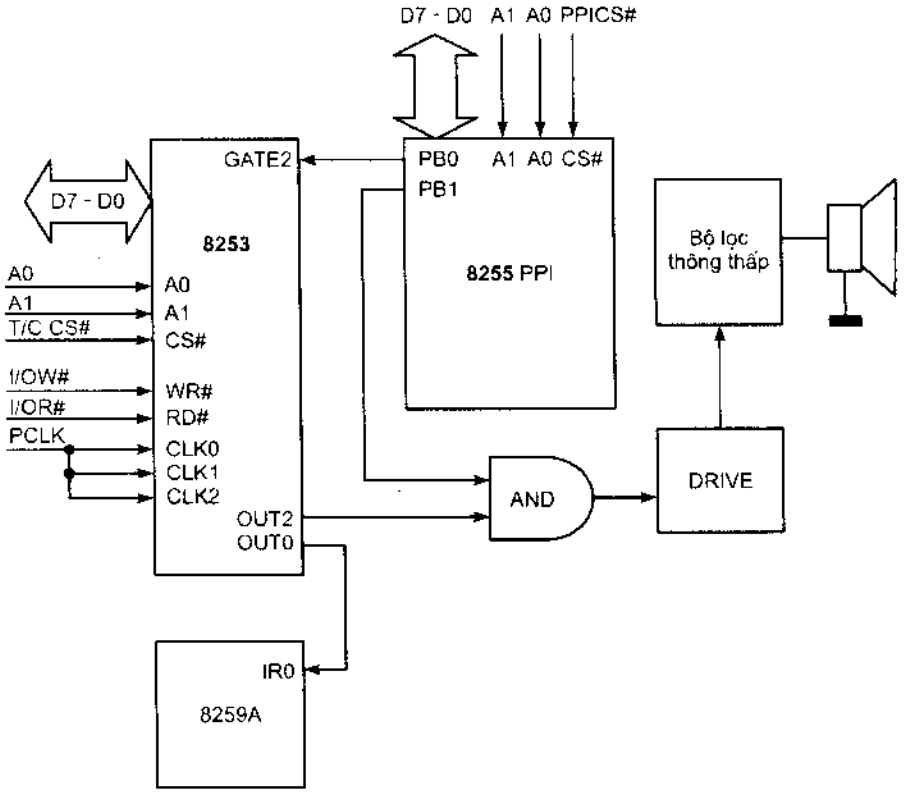
Các đường địa chỉ A1, A0 dùng để chọn các bộ đếm và thanh ghi điều khiển chế độ trong 8253 như sau:

A1	A0	
0	0	Bộ đếm 0
0	1	Bộ đếm 1
1	0	Bộ đếm 2
1	1	Thanh ghi điều khiển chế độ

Trong hệ thống IBM PC/XT, trên bảng mạch hệ thống, chip 8253 được dùng để điều khiển âm thanh ra loa cùng với chip PPI 8255. Phần kết nối 8253 và 8255 để điều khiển ra loa có thể được mô tả đơn giản theo sơ đồ khối ở hình 5.111.

Các đường dây dữ liệu D7-D0 và địa chỉ A1, A0 của 8253 nối với bus hệ thống (bus dữ liệu và address bus) của hệ vi xử lý 8-bit. Các

tín hiệu WR# (ghi dữ liệu) và RD# (đọc dữ liệu) được nối với các xung ghi và đọc từ hệ vi xử lý. Khi nối trong hệ 8086/8088, thì các tín hiệu này được tạo ra từ chip điều khiển bus (8255).



Hình 5.111: Kết nối chức năng điều khiển loa bằng 8253 và 8255 trong IBM PC/XT

Trên bảng mạch chính của IBM PC/XT (xem phụ lục), chip 8253 được chọn nhờ T/C CS# = 0 trong vùng địa chỉ từ 040h đến 043h:

T/C CS# = 0 = (A19=...=A7=0)(A6=1)(A5=...=A0=X)

Nếu kết hợp với 2 đường địa chỉ A0, A1 thì địa chỉ vào/ra của 5253 như sau:

040h chọn count 0

041h chọn count 1

042h chọn count 2, được chọn để điều khiển âm than ra loa (OUT2).

043h chọn Mode-control register.

Chip 8255 PPI được chọn nhờ PPICS# = 0 trong vùng địa chỉ từ 060h đến 063h:

PPICS# = 0 (A19=... A7=0)(A6=1)(A5=1)(A4=0)(A3=...=A0=X)

Với các đường địa chỉ A1, A0, các cổng vào/ra của 8255 PPI có địa chỉ được chọn trong IBM PC/XT như sau:

060h chọn port A

061h chọn port B, bit PB1 đi tới mạch AND cùng OUT2 của 8253 để điều khiển loa.

062h chọn port C

Các tín hiệu PB1, PB0 của 8255 PPI bằng "0" thì mạch AND khóa tín hiệu ra loa OUT2 từ 8253, loa sẽ không có tiếng. Nhịp PCLK có 1,19318 MHz được tạo ra từ bộ dao động U97, U21 với nhịp gốc của hệ thống CLK (4,77 MHz). Các đường dữ liệu 2 chiều (D0-D7) nối với bus hệ thống của PC/XT, các xung ghi IOW# và IOR# cho 8253 được lấy từ vi mạch điều khiển bus 8288 chế độ hoạt động tối da của 8086/8088.

Điều khiển âm thanh loa kêu trong IBM PC/XT thường nhằm mục đích báo hiệu trạng thái máy, như khởi động ban đồng, có các lỗi phần cứng. Âm thanh loa được tạo ra bằng cách gửi ra loa một tín hiệu xung vuông. Dựa vào lập trình chế độ 8253 ta có thể lập trình can thiệp trực tiếp đến các cổng của 8253 và 8255 để điều khiển âm thanh tùy chọn ra loa mà không sử dụng đến các lời gọi (CALL) đến các chương trình con trong ROM-BIOS hay DOS.

Để khởi tạo chế độ cho 8253, chúng ta cần phải ghi từ điều khiển chế độ vào thanh ghi điều khiển chế độ của 8253 thông qua địa chỉ cổng 43h. Phải ghi giá trị đếm vào bộ đếm được chọn thông qua địa chỉ cổng chọn bộ đếm. Để bật /tắt âm thanh loa, ta sử dụng địa chỉ cổng 61h của 8255, trong đó các bit PB1, PB0 có giá trị "1" để bật âm thanh loa, và ngược lại, PB1, PB0 bằng "0" làm tắt âm thanh loa.

Trong sơ đồ bảng mạch chính của IBM PC/XT, bộ đếm 2 (count 2) được chọn để điều khiển loa (GATE2, OUT2). Để tạo tone cho loa, cần đưa ra loa chuỗi xung vuông liên tục (OUT2). Như vậy phải đặt chế độ 3 (Mode 3) cho 8253. Khi lập trình ở chế độ 3, tần số của xung vuông liên tục (OUT2) bằng tần số nhịp đồng hồ của 8253 (là nhịp đồng hồ hệ thống vi xử lý chia cho 4: PCLK/4) chia cho giá trị nạp vào bộ đếm (giá trị đếm). Như vậy, nếu giả sử cần đưa tone âm thanh loa có tần số 440 Hz (nốt "La") thì cần phải nạp giá trị đếm là 2727 vào bộ đếm 2 (địa chỉ 42h) của 8253 khi PCLK = 4,77 MHz:

(PCLK/4)/X = 440 Hz, X = $(4{,}77.10^6/4)/440$ = 2727. (X là giá trị đếm).

Từ điều khiển phải đưa ra 8253 sẽ là: B6h (mã Hex): chọn bộ đếm 2, mode 3, nạp giá trị đếm dạng nhị phân 16-bit, nạp byte thấp trước, byte cao sau. Để tắt âm thanh loa ban đầu ta thiết lập giá trị các bit PB1, PB0 = 00:

OUT 61h, 0; đưa ra cổng B giá trị 0 để tắt âm thanh loa.

Để bật loa, ta thiết lập PB1, PB0 = 11:

OUT 61h, 3;

Như vậy bằng cách làm trễ quá trình bật tắt âm thanh loa với thời gian khác nhau ta có thể định ra được các quãng âm thanh dài ngắn khác nhau tạo qui định thông báo trạng thái của máy tính. Bằng kết hợp điều khiển thời gian làm trễ qua 8255 PPI và chế độ, ton của loa qua 8253 chúng ta có thể tạo âm thanh mong muốn. Thí dụ sau đây giới thiệu cách tạo âm thanh sử dụng bộ hẹn giờ của IBM PC/XT. Đây là chương trình tạo âm thanh tần số 440 Hz, tức là nốt "La" bình thường, viết ở hợp ngữ và để dịch ra dạng .COM.

```
TONSEG SEGMENT "CODE"
ASSUME CS: TONSEG
TONPROC FAR
MOV AL, 0B6h      ; nạp từ điều khiển vào thanh ghi AL
                    (mode 3, count 2)
```

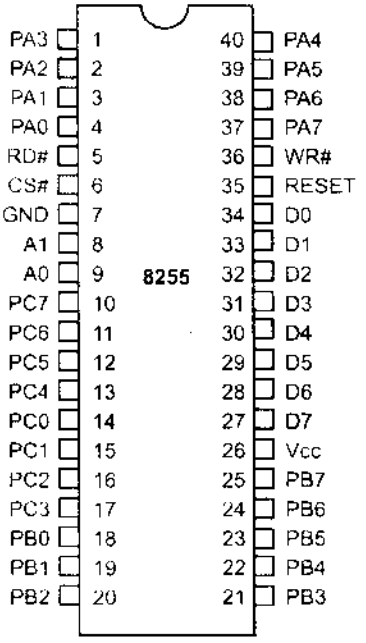
```
OUT 43h, AL      ; đưa ra thanh ghi điều khiển chế độ của 8253
MOV AL, 2727     ; giá trị đếm ứng với tần số 440 Hz
OUT 42h, AL      ; đưa ra bộ đếm 2 byte thấp của số đếm
MOV AL, AH       ; chuyển byte cao của số đếm sang AL
OUT 42h, AL      ; chuyển byte cao của số đếm ra bộ đếm 2
IN AL, 61h       ; đọc giá trị hiện tại từ cổng cổng B (61h) của 8255 PPI
OR AL, 3         ; đặt 2 bit PB1, PB0 bằng 11
OUT 61h, AL      ; cho loa phát âm thanh
INT 20h          ; trở về DOS
TONENDP
TONSEG ENDS
END
```

Chương trình BEEP.ASM dưới đây thực hiện chuyển một dãy xung ra điều khiển loa bằng vòng lặp. Cứ 50 chu kỳ vòng lặp thì hết thời gian ra loa, và chỉ giới hạn trong 3000 chu kỳ.

```
BEEPSEG          SEGMENT "CODE"
        ASSUME CS: BEEPSEG
BEEP    PROC FAR
        CLI              ; cấm ngắt từ bên ngoài
        MOV BX, 3000     ; số chu kỳ
        IN AL, 61h       ; đọc trạng thái từ cổng B (61h) của 8255 PPI.
        PUSH AX          ; cất giá trị trạng thái của cổng B
LAP:    AND AL, 0FCH     ; xóa các bit PB1, PB0 để tạo tắt âm thanh loa
        OUT 61h, AL      ; tắt âm thanh loa
        MOV CX, 50       ; đếm nửa chu kỳ âm thanh
T1:     LOOP T1          ; làm trễ thời gian giữa 2 chu kỳ âm thanh
        OR AL, 2         ; tạo bit PB1=1 cho cổng B của 8255 PPI
        OUT 61h, AL      ; bật âm thanh loa
```

```
                MOV CX, 50
                T2: LOOP T2      ; làm trễ thời gian giữ 2 chu kỳ âm thanh loa
                DEC BX           ; giảm số chu kỳ
                JNZ LAP          ; nhảy tới đầu chu kỳ loa
                POP AX           ; lấy từ ngăn xếp trạng thái cũ của cổng B của
                                   8255 PPI
                OUT 61h, AL      ; trả lại trạng thái cũ cho cổng B
                STI              ; cho phép ngắt từ bên ngoài.
                INT 20h          ; trở về DOS
BEEP            ENDP
BEEPSEG                          ENDS
                END
```

2. Vi mạch lập trình ghép nối ngoại vi 8255

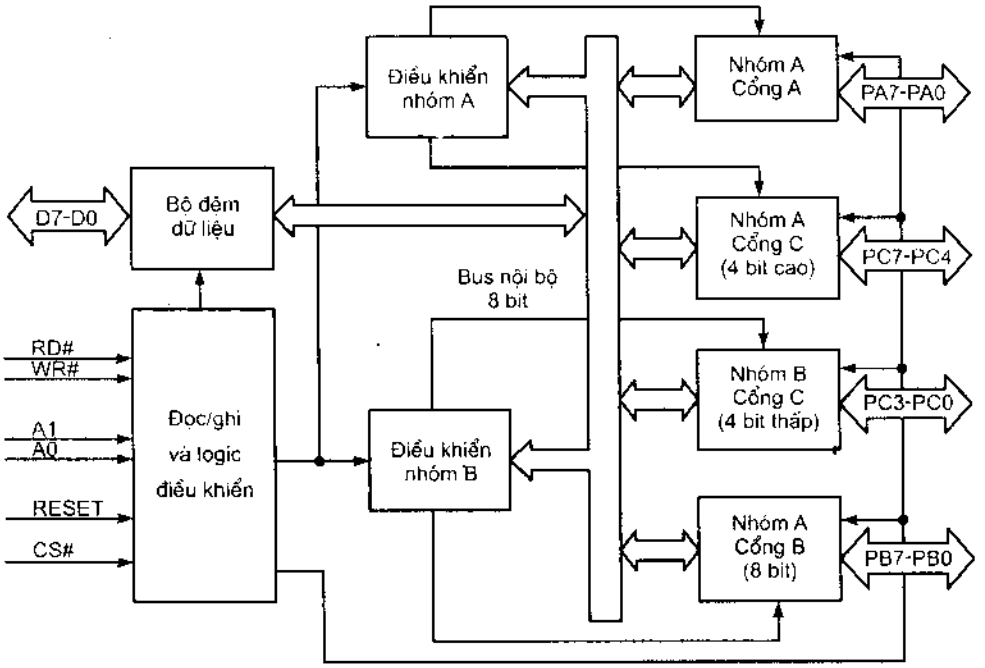


Tên tín hiệu	Chức năng
D7-D0	Bus dữ liệu, hai chiều
RESET	Thiết lập lại đầu vào
CS#	Chọn chíp đầu vào
RD#	Đầu vào đọc
WR#	Đầu vào ghi
A1, A0	Các đầu vào địa chỉ cổng
PA7-PA0	Các bit cổng A, 2 chiều
PB7-PB0	Các bit cổng B, 2 chiều
PC7-PC0	Các bit cổng C, 2 chiều
Vcc	+5 V
GND	0 V

Hình 5.112: Đóng vỏ và các chân tín hiệu của 8255 PPI

8255 PPI (programmable peripheral interface) là chip điều khiển phối ghép với các thiết bị ngoại vi trao đổi dữ liệu với vi xử lý Intel

8-bit, như 8080, 8085, 8086/8088. Nó đóng vỏ DIP 40 chân (hình 5.112). Hình 5.113 là sơ đồ chức năng của 8255 PPI. D7-D0: bus dữ liệu, nối với bus dữ liệu của hệ vi xử lý 8 bit. Dữ liệu đọc từ vi xử lý ra bộ đệm dữ liệu (DATA BUS BUFFER) nhờ lệnh OUT, và từ bộ đệm dữ liệu ghi vào vi xử lý nhờ lệnh IN. Các từ điều khiển từ vi xử lý đọc ra 8255 và từ trạng thái của 8255 ghi vào vi xử lý đều được lưu chuyển thông qua bus dữ liệu này.



Hình 5.113: Sơ đồ khối chức năng của 8255 PPI

1. READ/WRITE AND CONTROL LOGIC: Khối chức năng ghi đọc và logic điều khiển, thực hiện điều khiển tất cả các vận chuyển bên trong và bên ngoài của các từ dữ liệu, điều khiển và trạng thái. Nó tiếp nhận các tín hiệu điều khiển từ bên ngoài (từ vi xử lý) và tạo ra các lệnh (command) cho cả 2 nhóm điều khiển (nhóm A, B).

2. CS#: chọn chip, khi CS#=0 thì chip 8255 mới được chọn kết nối với hệ vi xử lý. Tín hiệu này ở trạng thái tích cực "0" khi địa chỉ từ vi xử lý đưa ra đúng là địa chỉ vào/ra dành cho chip 8255 qui định trong

hệ thống vi xử lý. Trong IBM PC/XT chip 8255 được dành cho giải địa chỉ vào/ra là 60h-63h, và tín hiệu chọn chip CS# = PPICS# được xác định như là:

CS# = PPICS# = 0 (A19=... A7=0)(A6=1)(A5=1)(A4=0)(A3=...=A0=X)

(xem bảng mạch mẹ IBM PC/XT và hình 5.81).

3. A1, A0, địa chỉ chọn các cổng vào/ra của 8255 PPI có địa chỉ được chọn trong IBM PC/XT như sau:

A1	A0	PC/XT I/O ADDR.	Chọn
0	**0**	**60h**	Cổng A
0	1	61h	Cổng B
1	0	62h	Cổng C
1	1	63h	Thanh ghi tư điều khiển

4. RESET: Khi bằng "1", xóa tất cả các thanh ghi bên trong (kể cả thanh ghi điều khiển), và tất cả các cổng A, B, C được xóa về chế độ đưa vào (input mode) từ ngoại vi.

5. GROUP A, GROUP B CONTROL: điều khiển nhóm A, nhóm B. Từng cổng A, B, C được lập trình theo ý muốn thông qua việc gửi ra 8255 từ điều khiển từ vi xử lý. Từ điều khiển chứa các thông tin như "mode" (chế độ), "bit set" (thiết lập bit), "bit reset" (khởi tạo lại bit), v.v... Mỗi một khối điều khiển nhóm (nhóm A, B) tiếp nhận các lệnh từ khối READ/WRITE AND LOGIC CONTROL và tiếp nhận các từ điều khiển từ vi xử lý gửi ra thông qua bus nội bộ 8-bit (8-bit internal bus) của 8255. Từ đó chúng sẽ tạo ra các lệnh cho các cổng tương ứng.

6. Thanh ghi từ điều khiển, là thanh ghi chỉ có thể nhận từ điều khiển gửi từ vi xử lý ra thông qua địa chỉ vào/ra (IBM PC/XT: 63h). Nội dung của nó không thể đọc được.

7. Cổng A: cổng vào/ra dữ liệu 8 bit (PA7-PA0) và có chốt/đệm dữ liệu vào/ra (input/output latch/buffer).

8. Cổng B: cổng vào/ra dữ liệu 8 bit (PB7-PB0) và có chốt/đệm dữ liệu vào/ra (input/output latch/buffer).

9. Cổng C: cổng vào/ra dữ liệu 8 bit, nhưng chỉ có chốt/đệm đưa ra (output latch/buffer) mà không có chốt đệm cho đưa vào. Cổng C được phân thành 2 cổng nhỏ: với mỗi nhóm có 4 bit (4 bit cao và 4 bit thấp). Một nhóm dùng để điều khiển đưa ra và nhóm còn lại dùng để nhận trạng thái từ ngoại vi (4 bit trạng thái). Cổng C thường dành để phục vụ điều khiển và kiểm trạng thái phục vụ vào/ra cho 2 cổng A, B.

Bảng 5.32 là các tổ hợp các tín hiệu điều khiển cơ bản của 8255.

Bảng 5.32: Các chân tín hiệu điều khiển

A1	A0	RD#	WR#	CS#	Thao tác
0	0	0	1	0	Đọc dữ liệu từ cổng A về bus dữ liệu
0	1	0	1	0	Đọc dữ liệu từ cổng B về bus dữ liệu
1	0	0	1	0	Đọc dữ liệu từ cổng C về bus dữ liệu
0	0	1	0	0	Ghi dữ liệu từ bus dữ liệu (CPU) ra cổng A
0	1	1	0	0	Ghi dữ liệu từ bus dữ liệu (CPU) ra cổng B
1	0	1	0	0	Ghi dữ liệu từ bus dữ liệu (CPU) ra cổng C
1	1	1	0	0	Ghi từ điều khiển từ bus dữ liệu (CPU) ra thanh ghi từ điều khiển
x	x	x	x	1	Bus dữ liệu ở trạng thái trở kháng cao: tri-state
1	1	0	1	0	Điều kiện sai
x	x	1	1	0	Bus dữ liệu ở trạng thái trở kháng cao: tri-state

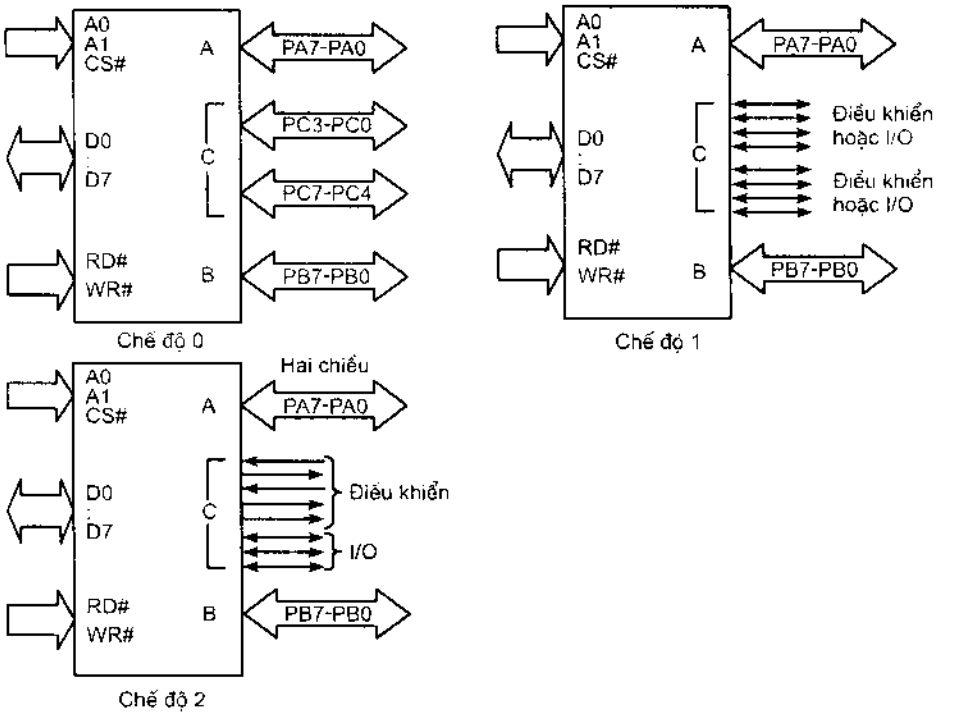
Mạch 8255 có 3 chế độ làm việc (hình 5.114) và được thiết lập nhờ các lệnh (chương trình), đó là các chế độ:

Chế độ 0: vào/ra cơ bản (basic input/output)

Chế độ 1: vào/ra có điểm nhịp (strobed input/output)

Chế độ 2: bus 2 chiều (Bi-directional bus).

Khi xóa bằng RESET="1", tất cả các cổng (A, B, C) được đưa về chế độ đưa vào (input mode). Khi đó, tất cả 24 đường dữ liệu của các cổng ở trạng thái trở kháng cao. Khi RESSET trở về "0", trang thái đưa vào của các cổng tiếp tục được duy trì.

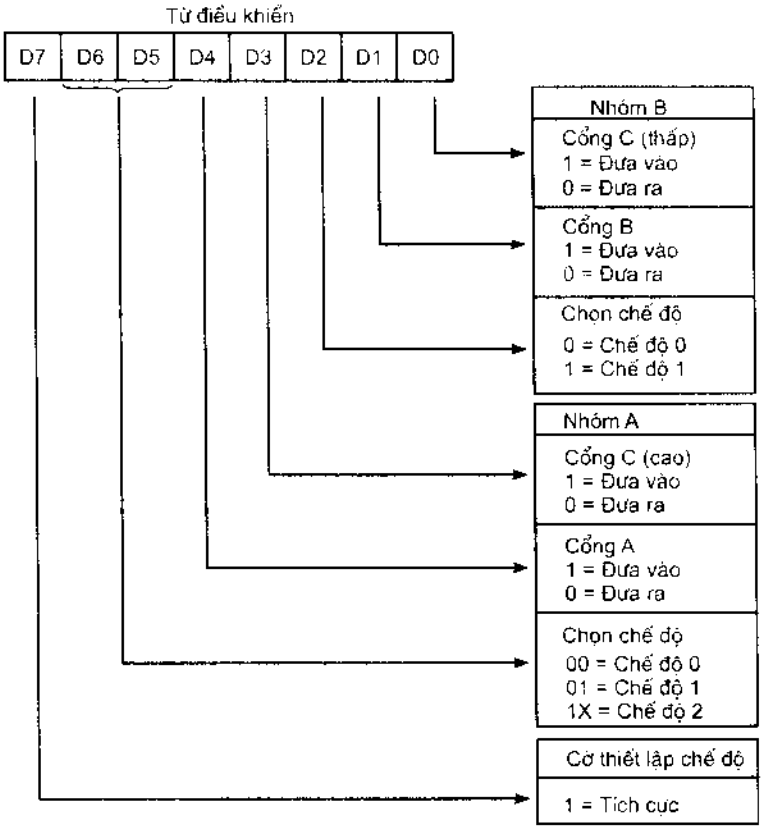


Hình 5.114: Các chế độ làm việc và giao tiếp của 8255

Khi chuyển đổi chế độ, tất cả các thanh ghi đưa ra (output registers), các Flip-Flop trạng thái sẽ được xóa. Từ điều khiển chế độ của 8255 có khuôn dạng cho trong hình 5.115.

1. Mode 0 (basic input/output): đảm bảo vào/ra dữ liệu 8 bit cho từng cổng (A, B, C). Khi đưa ra, các cổng được chốt dữ liệu, nhưng khi đưa ra không chốt dữ liệu. Có thể thực hiện vào/ra dữ liệu 16 bit nhờ kết hợp 2 cổng. Cổng C có thể phân thành 2 nhóm, mỗi nhóm 4 bit để thực hiện vào ra riêng từng nhóm.

2. Chế độ 1(strobed input/output): trong chế độ 1, cổng A và cổng B sử dụng các đường dây của cổng C để tạo hoặc tiếp nhận các tín hiệu "handshaking" (bắt tay) với thiết bị ngoại vi. Nghĩa là, sự vận chuyển dữ liệu vào/ra ở các cổng A và B phải được xác định (strobe) bằng tín hiệu "handshaking" với ngoại vi. Các cổng A, B là các cổng vào/ra dữ liệu 8-bit. Mỗi cổng cần 4 đường tín hiệu của cổng C cho điều khiển:



Hình 5.115: Khuôn dạng từ điều khiển của 8255

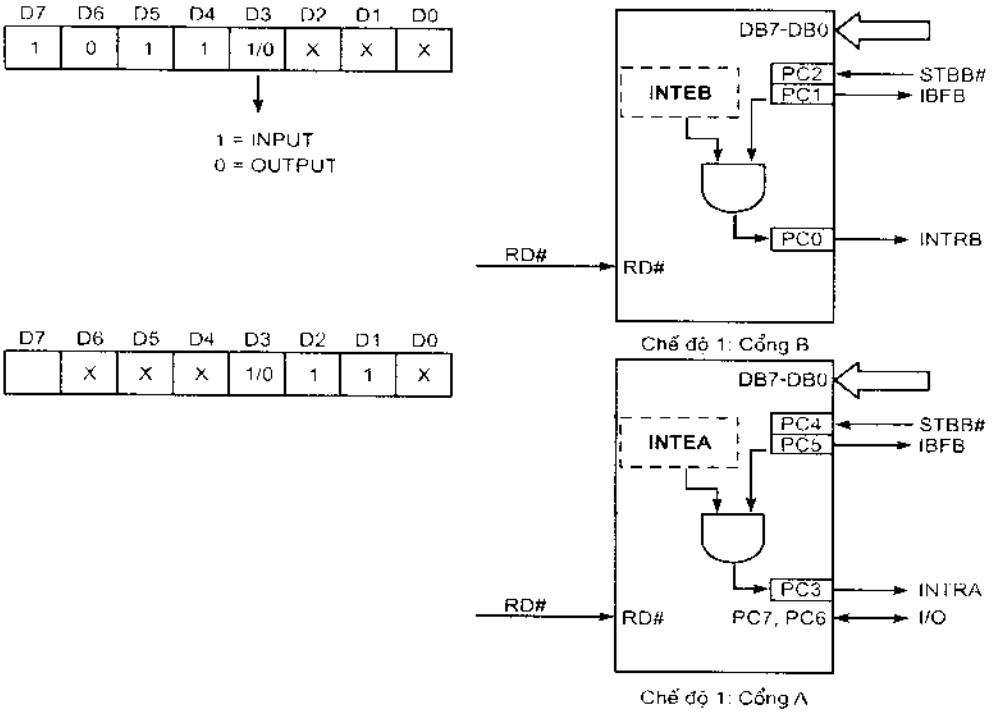
• Strobed Input chế độ 1 (chế độ 1 cho đưa vào): từ điều khiển và mô hình giao tiếp 8255 thể hiện trong hình 5.116.

STB# (Strobe input - Đầu vào nhấp nháy): xuống mức "0" sẽ nạp dữ liệu vào bộ đệm chốt dữ liệu của cổng (A, hoặc B).

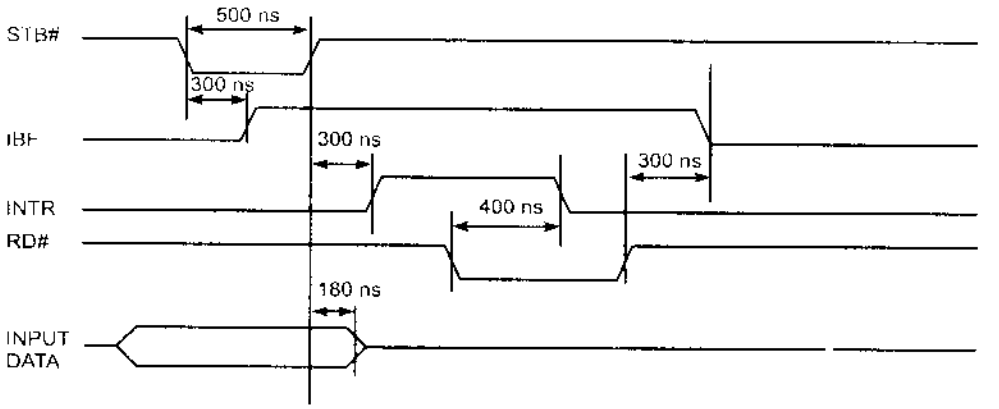
IBF (Input buffer full Flip-Flop - Flip-Flop - Bộ đệm đầu vào đầy): lên mức cao "1" chỉ ra rằng dữ liệu đã được nạp vào trong bộ đệm chốt của cổng vào/ra (A, hoặc B). Flip-Flop IBF được thiết lập "1" khi STB# ở mức "0" và bị xóa về "0" nhờ sườn trước của xung RD#.

INTR (Interrupt request - Yêu cầu ngắt): lên mức cao "1" về CPU để yêu cầu ngắt khi thiết bị ngoại vi có nhu cầu vào/ra. INTR = "1" khi STB# = "1" và Flip-Flop ngắt INTE thiết lập trạng thái = "1".

Flip-Flop INTE được xóa về "0" bằng sườn sau của xung RD#. Đồ thị xung của chế độ 1 cho đưa vào cho trong hình 5.117 (Thời gian tính bằng nanôgiây(ns) và là thời gian tối thiểu).



Hình 5.116: Input Mode 1 cho cổng A, cổng B với điều khiển của cổng C



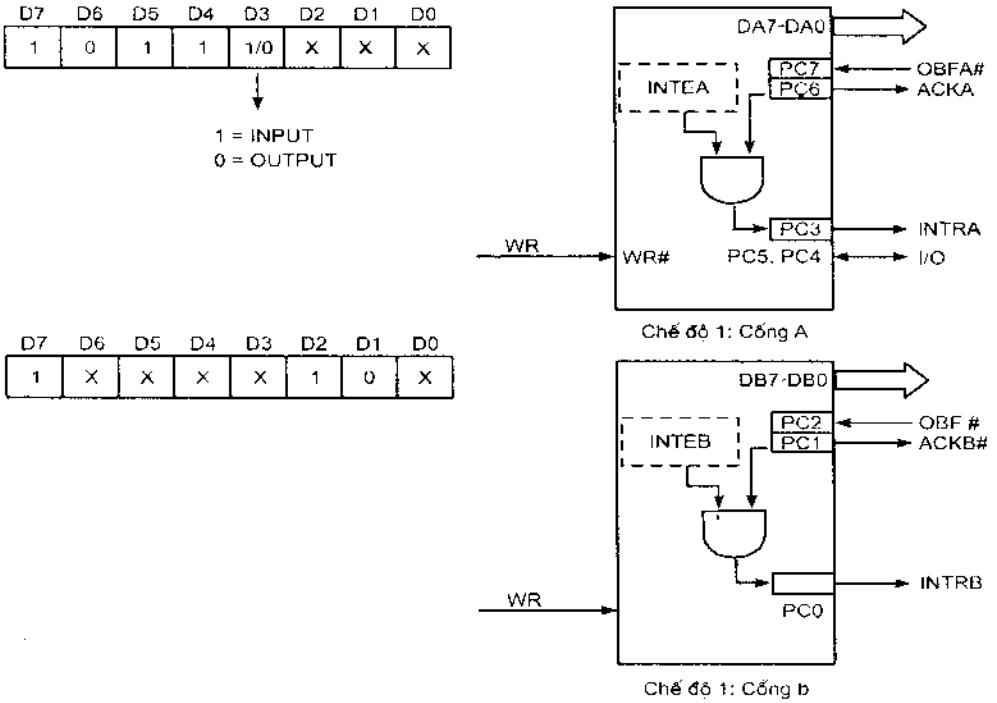
Hình 5.117: Đồ thị xung đưa vào dữ liệu ở chế độ 1
(input mode 1 - Chế độ đưa vào 1)

• Strobed Output Mode 1 (Chế độ 1 cho đưa ra dữ liệu): cần phải có các tín hiệu điều khiển khác so với trường hợp nhận dữ liệu từ ngoại vi, đó là các tín hiệu:

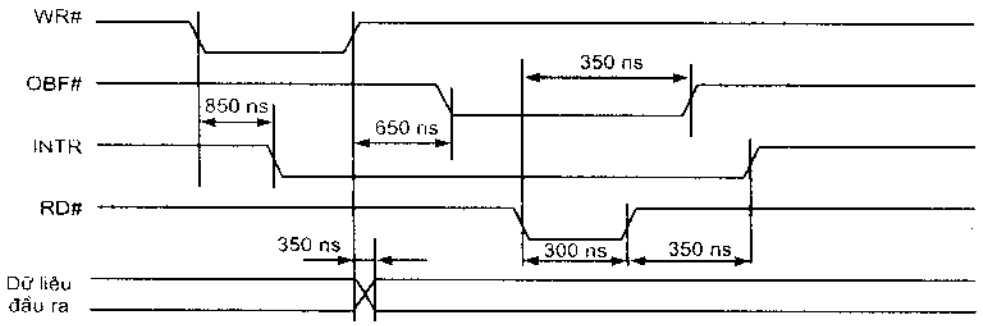
OBF# (Output buffer full Flip-Flop): xuống mức thấp "0" thông báo rằng CPU đã đưa ra cổng (A, hay B) dữ liệu. Flip-Flop OBF được thiết lập "1" nhờ sườn trước của xung WR# và bị xóa về "0" nhờ tín hiệu ACK = "0".

ACK# (Acknowledge): xuống mức thấp "0" thông báo cho 8255 rằng dữ liệu từ cổng A (hoặc B) đã được ngoại vi nhận được.

INTR (Interrupt request): lên mức cao "1" để yêu cầu ngắt (được phục vụ) CPU khi thiết bị ngoại vi đã nhận được dữ liệu từ CPU. INTR được thiết lập "1" nhờ ACK# = "1", OBF# = "1" và Flip-Flop INTE ở trạng thái "1" (bật). INTR được xóa về "0" nhờ sườn sau của xung WR#. Hình 5.118 là sơ đồ chế độ 1 cho đưa ra các cổng A, B. Hình 5.119 là đồ thị xung của chế độ này.



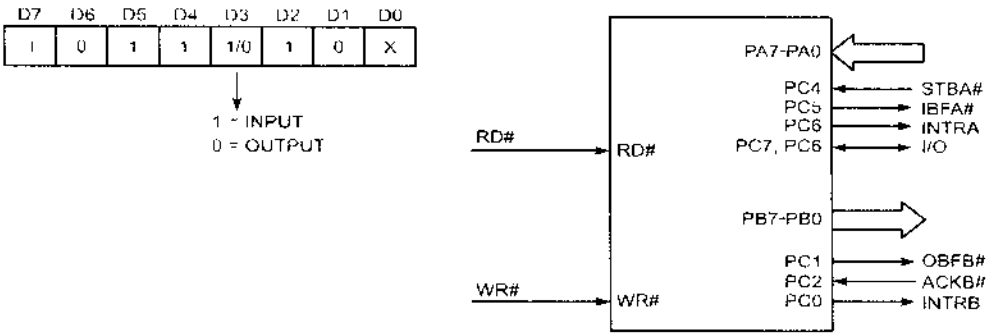
Hình 5.118: Output Mode 1 cho cổng A, cổng B với điều khiển của cổng C



Hình 5.119: Đồ thị xung đưa ra dữ liệu ở chế độ 1 (Output mode 1)

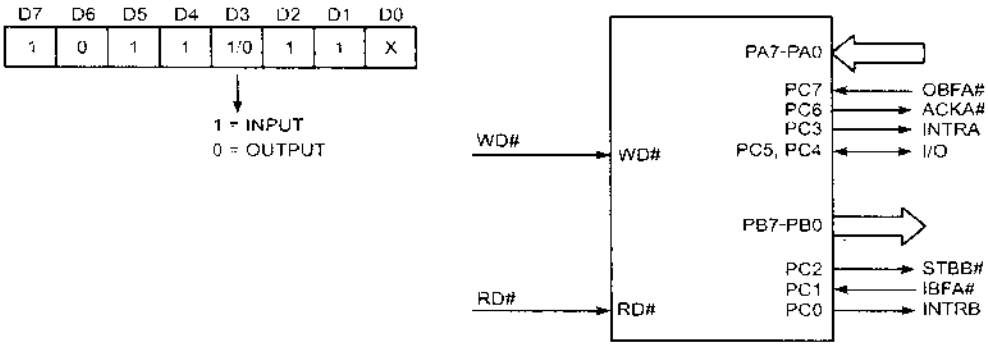
• Chế độ phối hợp nhấp nháy 1: chế độ 1 hỗn hợp cả vào và ra: trong chế độ này, cổng A và cổng B có thể được thiết lập riêng biệt: 1 cổng dùng để nhận dữ liệu, 1 cổng dùng đưa ra dữ liệu. Như vậy, có thể có 2 trường hợp:

- Cổng A: Strobed Input, Cổng B: Strobed Output:



Hình 5.120: Chế độ 1: Cổng A nhận dữ liệu, cổng B phát dữ liệu

- Port A: Strobed Output, cổng B: Strobed Input:



Hình 5.121 Chế độ 1: Cổng A phát dữ liệu, cổng B nhận dữ liệu

• Chế độ 2 (Strobed Bi-directional bus I/O): chế độ 2, với bus vào/ra 2 chiều. Trong chế độ này, kết nối với thiết bị ngoại vi thực hiện thông qua bus 8-bit 2 chiều. Các tín hiệu bắt tay đảm bảo duy trì dòng dữ liệu qua bus tương tự như chế độ 1. Ngắt cũng được tạo ra trong chế độ này. Các chức năng cơ bản của chế độ này như sau:

- Chỉ dùng cho cổng A, và cổng A là bus vào/ra 2 chiều 8-bit.

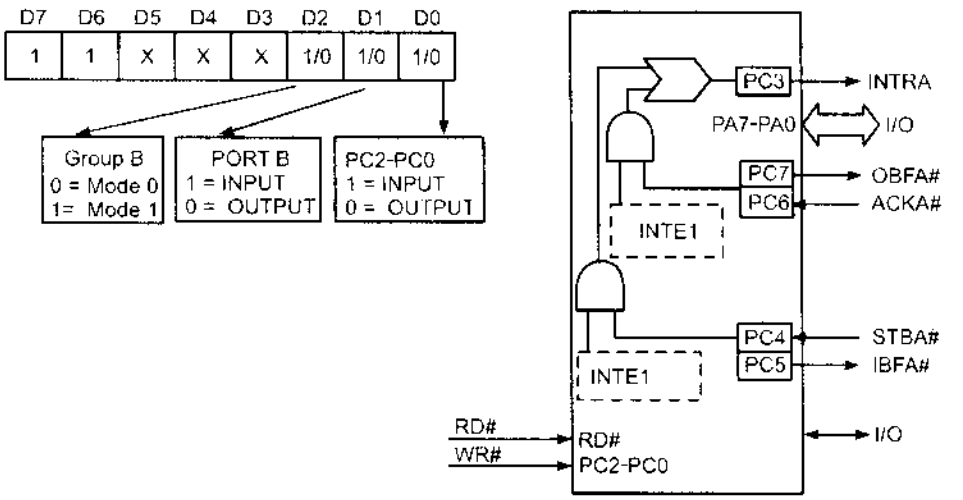
- Dùng 5 bit của cổng C cho điều khiển, trạng thái vào/ra của cổng A.

- Cả 2 chiều vào/ra đều có chốt dữ liệu

• Thao tác đưa ra gồm:

OBF#: xuống mức thấp "0" chỉ ra rằng CPU đã đưa dữ liệu 8-bit ra cổng A.ACK#: xuống mức thấp "0" cho phép bộ đệm 3 trạng thái của cổng A gửi dữ liệu ra thiết bị ngoại vi, ngược lại, nếu lên mức cao "1" thì các đầu ra của bộ đệm cổng A sẽ ở trạng thái trở kháng cao.

INTE1 (INTE Flip-Flop liên hệ với OBF#): được điều khiển bởi bit PC6.



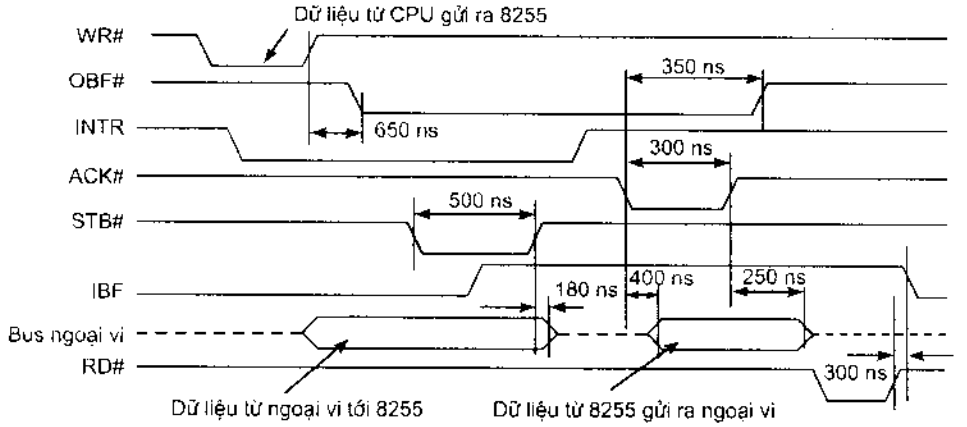
Hình 5.122: Mode 2: Cổng A: bus dữ liệu, cổng C : điều khiển

• Thao tác ghi vào gồm:

STB#: xuống mức thấp chỉ ra rằng dữ liệu đã nạp vào bộ đệm chốt đưa vào (input buffer latch).

IBF: lên mức cao "1" chỉ ra rằng dữ liệu đã được nạp vào bộ đệm chốt đưa vào.

INTE2: được điều khiển bởi bit PC4.



Hình 5.123: Đồ thị xung của chế độ 2 của 8255

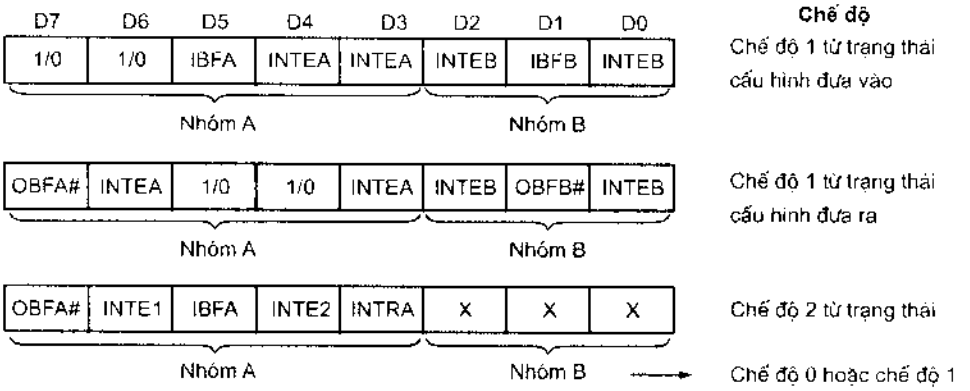
Từ điều khiển và kết nối của 8255 ở chế độ 2 cho ở hình 5.122. Đồ thị xung chế độ 2 cho ở hình 5.123. Do chế độ 2 chỉ thiết lập được cho cổng A nên có thể kết hợp thiết lập các chế độ 0 và 1 cho cổng B. Ta có bảng như sau:

Bảng 5.33: Thiết lập chế độ cổng B

Từ điều khiển								Chế độ của cổng và các tín hiệu
D7	D6	D5	D4	D3	D2	D1	D0	
1	1	X	X	X	0	1	1/0	Cổng A: Chế độ 2; PA7-PA0: bus dữ liệu I/O Cổng C: INTRA=PC3, OBFA# = PC7, PC6 = ACKA# PC4 = STBA# , IBFA = PC5, PC2-PC0 = I/O Cổng B: Chế độ đưa vào 0 (dữ liệu đưa vào - 8 bit)
1	1	X	X	X	0	0	1/0	Cổng A: Chế độ 2; PA7-PA0: I/O bus dữ liệu Cổng C: INTRA=PC3, OBFA# = PC7, PC6 = ACKA# PC4 = STBA# , IBFA = PC5, PC2-PC0 = I/O Cổng B: Chế độ đưa ra 0 (dữ liệu đưa ra - 8 bit)
1	1	X	X	X	1	0	X	Cổng A: Chế độ 2; PA7-PA0: bus dữ liệu I/O Cổng C: INTRA=PC3, OBFA# = PC7, PC6 = ACKA# PC4 = STBA# , IBFA = PC5

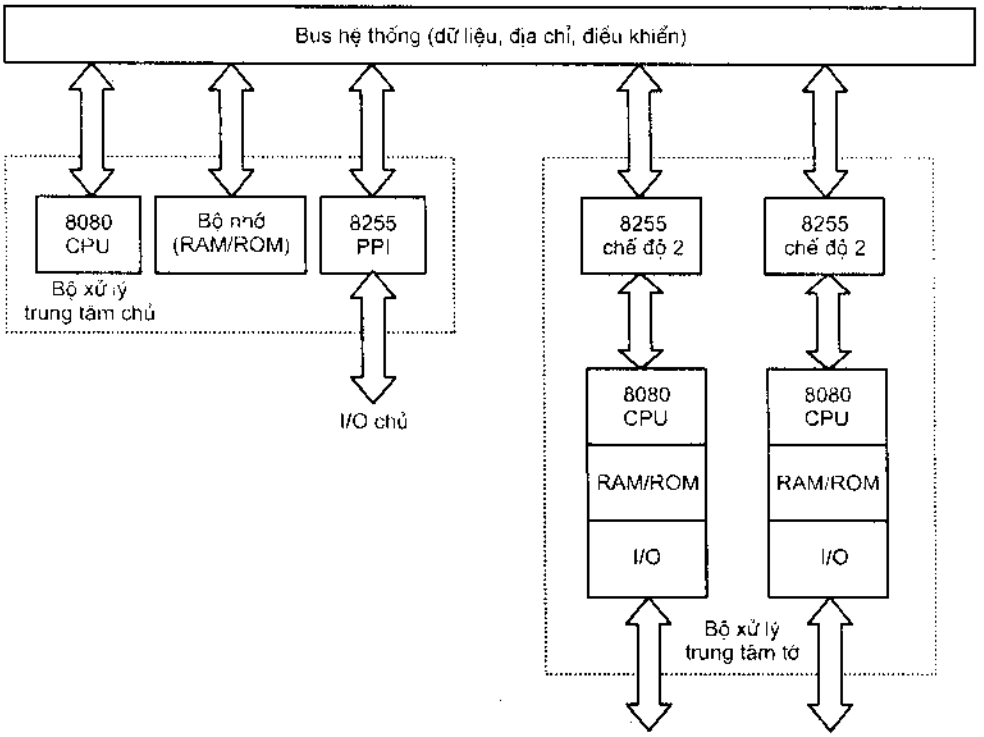
								Cổng B: Chế độ đưa ra 1 (dữ liệu đưa ra - 8 bit) Cổng C: OBFB# = PC1, PC2 = ACKB#, INTRB = PC0
1	1	X	X	X	1	1	X	Cổng A: Chế độ 2; PA7-PA0: bus dữ liệu I/O Cổng C: INTRA=PC3, OBFA# = PC7, PC6 = ACKA# PC4 = STBA# , IBFA = PC5 Cổng B: Chế độ đưa vào 1 (dữ liệu đưa vào 8 - bit) Cổng C: PC2 = STBB#, IBFB = PC1, INTRB = PC0

Trong Mode 0, cổng C chuyển dữ liệu từ ngoại vi hoặc từ 8255 đến ngoại vi, cũng giống như các cổng A và B. Trong Mode 1 và 2, cổng C tạo ra hoặc tiếp nhận các tín hiệu để bắt tay (handshaking) với ngoại vi. Do đó, sự đọc nội dung của cổng C cho phép người lập trình kiểm tra hoặc so sánh trạng thái của từng thiết bị ngoại vi để thay đổi nhánh chương trình điều khiển vào/ra với ngoại vi. Từ trạng thái của 8255 được đọc về CPU thông qua bus dữ liệu (D7-D0) có khuôn dạng như sau:



Trong hệ vi xử lý 8-bit, chip 8255 được sử dụng phần lớn cho ghép nối vào/ra giữa CPU và thiết bị ngoại vi. Trong hệ thống đa xử lý, 8255 thiết lập ở chế độ 2 có thể nối với các bộ xử lý trung tâm tớ (slave CPU), và ở chế độ 0, hoặc 1 trong sơ đồ bộ xử lý trung tâm chủ (Master CPU). Hình 5.124 là một ví dụ kết nối đa xử lý (chip 8080) có sử dụng 8255.

Vì chế độ 2 cho phép cổng A của 8255 có bus dữ liệu 8-bit 2 chiều và chốt cả vào/ra nên 8255 có thể nối trực tiếp với 8080 CPU. Như vậy sẽ tiết kiệm nhiều mạch bổ trợ cho kết nối 8255 với vi xử lý.



Hình 5.124: Hệ đa xử lý 8080 với 8255 PPI

Trong IBM PC/XT, chúng ta cũng nhận thấy 8255 được dùng cho điều khiển loa cùng với 8253, đồng thời để nhận dữ liệu từ bàn phím. Trong trường hợp này, cổng A và B được thiết lập ở chế độ 1 để thực hiện vào ra với bàn phím. Ngoài ra, 8255A được dùng cho giao tiếp vào/ra với băng từ cassette ở một số cấu hình của IBM PC/XT. Cụ thể, có thể xem chi tiết chức năng từng cổng vào/ra của 8255A với các bit của chúng sử dụng trong IBM PC/XT trong bảng mạch mẹ.

Như vậy, 8255 PPI là chip lập trình tương đối vạn năng cho các giao tiếp vào/ra song song và liên tiếp với các chế độ lập trình đơn giản và linh hoạt. Nó chuyên dùng cho các hệ thống vi xử lý Intel 8-bit, vì nó sẽ rất thuận tiện khi kết nối trực tiếp vào bus hệ thống của hệ vi xử lý nếu nó được lập trình ở chế độ 2 vào/ra cổng A. Nhưng cũng có thể cho phép phối ghép vào/ra dữ liệu 16-bit nếu kết hợp cả 2 cổng A và B, và cổng C làm cổng điều khiển phối ghép với thiết bị ngoại.

BÀI TẬP ÔN LUYỆN

1. Dung lượng ngăn xếp trong hệ thống 8085 tối đa là bao nhiêu.
2. Trong hệ thống IBM PC tín hiệu NMI dùng để làm gì.
3. Trong hệ thống đa xử lý 8086, sự khác nhau như thế nào giữa các bus public, local và private.
4. Tất cả các lệnh nhảy ngoại trừ JMP sử dụng:
   a: đánh địa chỉ gián tiếp; b: đánh địa chỉ trực tiếp
   c: đánh địa chỉ gián tiếp thanh ghi; b: tất cả a, b, c.
5. Các lệnh IN, OUT và XLAT có một đặc điểm chung là:
   a: Chúng sử dụng thanh ghi AX như là một phần cho vận chuyển dữ liệu.
   b: Chúng chỉ giới hạn chuyển dữ liệu 8-bit.
   c: Chúng đều có thể làm việc hoặc với μP hoặc với bộ nhớ.
   d: Không phải tất cả trên.
6. Hãy thiết kế chi tiết logic giải mã chọn các chip trong sơ đồ cho ở hình 6.27, trong đó các cổng vào/ra của 8155 PPI.
7. Trong 8155 PPI có SRAM 256B. Vậy chức năng của SRAM này như thế nào.
8. Chip 8155 PPI có bao nhiêu lệnh để lập trình. Hãy liệt kê các lệnh đó. Làm thế nào để thực hiện các lệnh này (thứ tự thực hiện, lệnh được gọi ra như thế nào).

9. Hãy viết đoạn chương trình cho 8085 (hợp ngữ) điều khiển đưa ra máy in một dòng ký tự: gồm toàn bộ các ký tự chữ cái và số của bảng mã ASCII, và kết thúc dòng là hai ký tự CR LF.

10. Các chế độ vận hành nào của 8253 được sử dụng để khởi tạo ngắt.

11. Giá trị đếm tối đa của 8253 là bao nhiêu, vì sao?

12. Điều gì sẽ xảy ra đối với tín hiệu xung vuông tương ứng với chế độ 3 của 8253 nếu cổng vào GATE của 8253 bị nối đất.

13. Kỹ thuật nào được 8253 sử dụng để phân biệt 3 thanh ghi điều khiển chế độ.

14. Cho tần số nhịp đồng hồ của vi xử lý (PCLK) là 8 MHz, hãy lập trình tạo âm thanh nốt "Son" bình thường bằng hợp ngữ dịch ra .com.

15. Cho tần số nhịp đồng hồ của vi xử lý (PCLK) là 8 MHz, hãy lập trình tạo âm thanh đưa ra tập hợp các nút nhạc "Đồ = 296,66 Hz; Rê = 329,63 Hz; Mi = 349,23 Hz; Pha = 392 Hz; Son = 415,30 Hz; La = 440 Hz; Si = 466,16 Hz; Đô = 493,88 Hz" bình thường bằng hợp ngữ dịch ra .com.

16. Vẽ sơ đồ giải mã chọn chip 8253.

17. Vì sao phải sử dụng các địa chỉ vào/ra để truy nhập 8253 mà không dùng BIOS ROM.

18. Thử thiết kế đồng hồ điện tử (chỉ thị LED 7 thanh) sử dụng 8253 kết nối với hệ vi xử lý 8086/8088 ở chế độ tối thiểu, hoặc 8085, 8051. Viết chương trình điều khiển.

19. Hãy so sánh 8155 PPI và 8255 PPI.

20. Thay 8255 vào 8155 vào sơ đồ kết nối máy in ở hình 5.58. Viết lại chương trình điều khiển đưa ra máy in qua 8255.

21. Trong thanh ghi dịch vụ ISR của 8259A sự thiết lập các bit của nó để làm gì.

22. Khi sử dụng 8259A, địa chỉ véc-tơ ngắt được tạo ra như thế nào.

23. Làm thế nào để phân biệt 4 từ lệnh khởi tạo của 8259A.

24. Dựa theo sơ đồ trình tự khởi tạo ban đầu hãy viết chương trình khởi tạo ban đầu cho 8259A kết nối trong IBM PC/XT.

25. Hãy thiết kế hệ thống xử lý ngắt nối tầng gồm 2 chip 8259A với 8086/8088 trong chế độ tối thiểu.

26. Những ưu điểm của truyền dữ liệu nối tiếp so với truyền dữ liệu song song.

27. Những ưu điểm của mạch USART 8251 so với UART 8250.

28. Phân biệt các từ chế độ (mode word) và từ lệnh (command word).

29. Theo chuẩn RS 232C, cái gì xác định logic "1".

30. Thiết kế sơ đồ kết nối 2 chip UART 8250 kết nối 8086/8088 trong chế độ tối thiểu, trong đó một chip UART 8250 làm cổng ra truyền thông (Port A), chip còn lại dùng làm cổng vào (Port B). Nối 2 cổng lại với nhau như thế nào đó, và lập trình đưa ra Port A một chuỗi ký tự A,B,... Z, a, b,... z, đồng thời nhận về từ Port B chuỗi ký tự này.

31. Tương tự như câu 30 nhưng dùng USART 8251.

32. Thiết kế sơ đồ kết nối trực tiếp (2 dây) hai hệ thống 8086/8088 + UART 8250 và 8086/8088 + USART 8251 và lập trình kết nối truyền dữ liệu không đồng bộ giữa hai hệ thống này.

33. Thiết kế hệ 8086/8088 kết nối USART 8251 truyền đồng bộ, lập trình thiết lập chế độ truyền đồng bộ cho hệ thống này.

34. Hãy dùng UART 8250 hoặc USART 8251 kết nối với 8086/8086 hoặc 8085 để thiết kế hệ thống điều khiển một hệ thống điện tử nào đó, ví dụ: đồng hồ Von-Amper, chỉ thị LED quảng cáo, cân điện tử, đo nhiệt độ, đo tốc độ động cơ điện, ... tùy theo khả năng mà bạn lựa chọn.

35. Chuẩn RS-232 xác định điện thế của tín hiệu sẽ thay đổi tối thiểu ... đối với logic 1 đến logic 0.

    a: ...3 V; b: ...20m A; c: 5V vi phân; d: ±12 V.

36. Mô tả ngắn gọn các đặc tính của đường truyền thông tuần tự sử dụng chuẩn RS-232 và sử dụng chuẩn RS-422.

37. Cổng I/O sử dụng logic giải mã địa chỉ để:

    a: Chọn cổng được địa chỉ; b: Chọn một vùng nhớ để vận chuyển dữ liệu;

    c: Sắp xếp cổng I/O trong bản đồ nhớ (Memory map);

    d: Để nhớ đệm các đường địa chỉ.

38. Trong giao tiếp I/O song song, đường tín hiệu DATA OUT AVAILABLE ở mức logic 1 khi:

    a: Một thiết bị ngoài có thể đưa vào dữ liệu;

    b: Vi xử lý đã nhận được dữ liệu;

    c: Một thiết bị ngoài đã nhận được dữ liệu từ sự đưa ra;

    d: Một thiết bị ngoài đã có dữ liệu cho vi xử lý.

39. Các tín hiệu I/O WRITE# và I/O READ# có giống các tín hiệu MEM WRITE# và MEM READ# hay không? giải thích.

40. Các ký tự ở mã ASCII 7-bit, nhưng bình thường chúng ta sử dụng 8 bit cho truyền dữ liệu. Một 1 thêm để:

    a: Chỉ ra sự tràn; b: Bổ sung dữ liệu khi 7 bit không đủ;

    c: Cho bit chẵn lẻ (kiểm tra chẵn lẽ); d: Tất cả a, b, c.

41. Chuẩn EIA cho truyền thông tuần tự có mức tín hiệu giao động từ +3 V đến -3 V trên từng đường dây riêng thu và phát và có thể gồm cả một số đường bắt tay khác là:

    a: RS-422; b: IEEE-488;

    c: RS-232; d: Vòng dòng 20 mA.

Chương 6

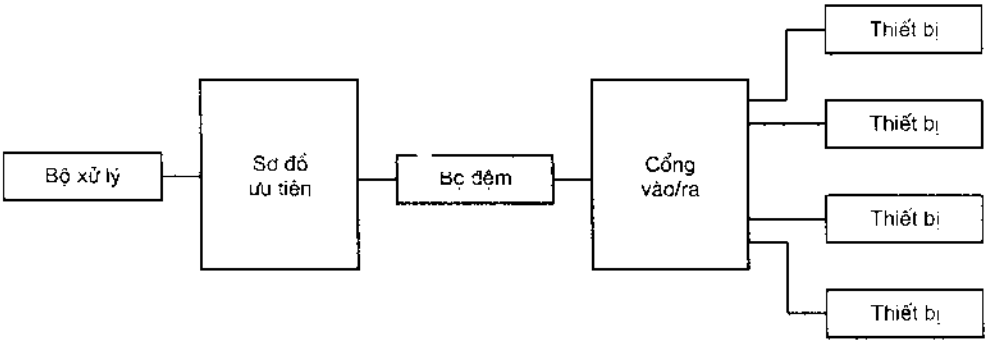
CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN VÀO/RA DỮ LIỆU

6.1. GIỚI THIỆU CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN VÀO/RA DỮ LIỆU

Chúng ta đã xét ở chương 5 vấn đề ghép nối các thiết bị ngoại vi với CPU trong hệ thống vi xử lý để vào ra dữ liệu song song hay nối tiếp. Mỗi một thiết bị ngoại vi được gán cho một dải địa chỉ vào/ra của hệ thống vi xử lý, mà ta vẫn gọi là các cổng vào/ra (I/O ports). Nhưng trong hệ thống vi xử lý không chỉ có một mà là một số thiết bị ngoại vi cùng loại, khác loại. Các thiết bị ngoại vi lại có yêu cầu phục vụ giống nhau, phương pháp vào/ra dữ liệu cũng giống nhau (song song hoặc nối tiếp, đồng bộ hoặc không đồng bộ), và cũng có thể là cùng một thời điểm có nhu cầu trao đổi dữ liệu với CPU. Hệ thống bus chỉ có một và là môi trường duy nhất để CPU và chỉ một thiết bị trao đổi dữ liệu trong một thời điểm, hay nói cách khác, trong một thời điểm CPU chỉ có thể phục vụ được một yêu cầu vào/ra với một thiết bị ngoại vi. Do đó, cần phải có một cơ chế hay là phương pháp điều khiển vào/ra thế nào đó để CPU có thể phục vụ được tất cả các yêu cầu vào/ra của thiết bị ngoại vi. Một cơ chế thường dùng là cơ chế xét ưu tiên, gọi là sơ đồ ưu tiên (priority scheme) (hình 6.1). Thiết bị có ưu tiên cao nhất thì được phục vụ trước. Thiết bị có mức ưu tiên thấp nhất thì được phục vụ sau cùng. Như vậy phải gán trước mức ưu tiên cho từng thiết bị phối ghép trong hệ thống vi xử lý.

Các hệ thống vi xử lý thường áp dụng 3 phương pháp điều khiển vào/ra với các thiết bị ngoại vi qua các cổng vào/ra song song và nối

tiếp, đó là: thăm dò (polling), ngắt và DMA. Mỗi phương pháp có ưu nhược điểm của nó và phù hợp với các đặc tính của các thiết bị ngoại vi cũng như cấu trúc kết nối các thiết bị ngoại vi. Những đặc tính đó là: tốc độ trao đổi dữ liệu, dạng dữ liệu, vào/ra song song hay nối tiếp, dung lượng dữ liệu nhiều hay ít khi trao đổi, khoảng cách kết nối (cục bộ hay trên diện rộng), v.v...



Hình 6.1: Phối ghép các thiết bị ngoại vi với CPU theo ưu tiên

6.1.1. Đặc điểm điều khiển vào/ra bằng phương pháp thăm dò

Đặc điểm của điều khiển vào/ra bằng phương pháp thăm dò (polling):

- Là phương pháp điều khiển vào/ra đơn giản và bằng phần mềm;
- Các thiết bị ngoại vi phải đưa ra các bit cờ trạng thái sẵn sàng của thiết bị (trong thanh ghi trạng thái);
- Phần mềm thực hiện kiểm tra (thăm dò) có chu kỳ các bit trạng thái sẵn sàng của tất cả các thiết bị có trong hệ thống vi xử lý bằng các vòng lặp. Thiết bị nào có trạng thái sẵn sàng thì được phục vụ;
- Có thể có thăm dò tùy ý (không theo ưu tiên) nhưng phải thăm dò hết thiết bị có trong hệ thống, và có thể có thăm dò theo ưu tiên phục vụ. Thiết bị có ưu tiên cao nhất được thăm dò trước, thiết bị có ưu tiên thấp nhất được thăm dò sau cùng;
- Dễ dàng thực hiện các sơ đồ ưu tiên (priority schemes) phục vụ các thiết bị;

• Thời gian của CPU bị chiếm nhiều để thực hiện chương trình lặp thăm dò, vì CPU đảm nhiệm toàn bộ điều khiển vào/ra dữ liệu với thiết bị ngoại vi.

• Chỉ nên dùng cho một số ít thiết bị. Có nhiều thiết bị sử dụng phương pháp thăm dò sẽ làm cho tốc độ xử lý của hệ thống bị suy giảm lớn và do đó giảm hiệu xuất của hệ thống.

6.1.2. Đặc điểm điều khiển vào/ra bằng phương pháp ngắt

Đặc điểm của điều khiển vào/ra bằng phương pháp ngắt là:

• Điều khiển chủ yếu bằng phần cứng. Phải có thêm phần cứng cho điều khiển ngắt;

• Thiết bị có yêu cầu được phục vụ phát sinh ra một yêu cầu cần được phục vụ (INTR) gửi về CPU;

• Tiếp nhận tín hiệu yêu cầu ngắt (INTR), CPU thực hiện xong các chu kỳ dang dở của lệnh máy đang thực hiện, cất giữ nội dung của các thanh ghi cần thiết cho sử tiếp tục thực hiện trở lại chương trình dang dở, vào ngăn xếp, và gửi ra thiết bị ngoại vi có yêu cầu ngắt sự chấp nhận ngắt (tín hiệu ACK);

• Nhận ACK tích cực, thiết bị có yêu cầu ngắt gửi về CPU véc-tơ ngắt (số hiệu ngắt) về CPU để lấy địa chỉ của chương trình con phục vụ ngắt;

• Qua địa chỉ, chương trình phục vụ ngắt để vào/ra dữ liệu với thiết bị có yêu vầu ngắt được thực;

• Kết thúc phục vụ ngắt (vào/ra dữ liệu xong) cho thiết bị trở về lại điểm bị ngắt của chương trình dang dở để tiếp tục thực hiện;

• Các thiết bị được gán mức ưu tiên cho yêu cầu ngắt để CPU thực hiện các yêu cầu ngắt theo ưu tiên;

• Giảm thiểu thời gian của CPU dành cho điều khiển vào/ra thiết bị, vì CPU không bận tâm đến thiết bị ngoại vi. Chỉ khi nào thiết bị sẵn

sàng có nhu cầu được phục vụ thì mới tác động đến CPU, và khi đó CPU mới thực hiện các thao tác cho điều khiển vào/ra;

• Có thể gắn các yêu cầu ngắt với các sự kiện để xử lý như: vào/ra dữ liệu, các lỗi thiết bị, lỗi nguồn,...

• Sự gắn kết các sự kiện ngắt với các số hiệu ngắt (véc-tơ ngắt) làm cho quá trình thực hiện điều khiển vào/ra dữ liệu có thể được chuẩn hóa về địa chỉ và là cơ sở cho hệ thống điều khiển vào/ra cho các hệ điều hành;

• Có hiệu quả với những thiết bị có khối lượng dữ liệu vào/ra theo các khối (block device).

Cả 2 phương pháp: thăm dò và ngắt đều cần có sự can thiệp của CPU, nhưng phương pháp ngắt hiệu quả hơn phương pháp thăm dò.

6.1.3. Đặc điểm điều khiển vào/ra bằng phương pháp DMA

Đặc điểm của điều khiển vào/ra bằng phương pháp DMA là:

• Thiết bị có yêu cầu truy nhập trực tiếp bộ nhớ phải gửi về CPU yêu cầu DMA.

• Các yêu cầu DMA cũng có mức yêu tiên phục vụ khác nhau.

• Tiếp nhận yêu cầu DMA, CPU giải phóng bus hệ thống, trao cho thiết bị có yêu cầu DMA sử dụng, và treo các cửa ra bus hệ thống của mình bằng cách đưa về trạng thái trở kháng cao. CPU còn có thể tiếp tục quá trình xử lý với các thanh ghi bên trong mà thôi.

• Tiếp nhận sự chấp nhận yêu cầu DMA của CPU, thiết bị có yêu cầu DMA bắt đầu quá trình vào/ra dữ liệu với bộ nhớ chính (RAM) với vai trò là chủ (master) và bộ nhớ RAM là thợ (slave) đối với bus hệ thống.

• Kết thúc quá trình DMA, thiết bị trả lại bus cho CPU.

• Phù hợp cho các thiết bị có dung lượng dữ liệu vào/ra lớn theo khối tốc độ cao.

• Sự can thiệp của CPU là rất ít

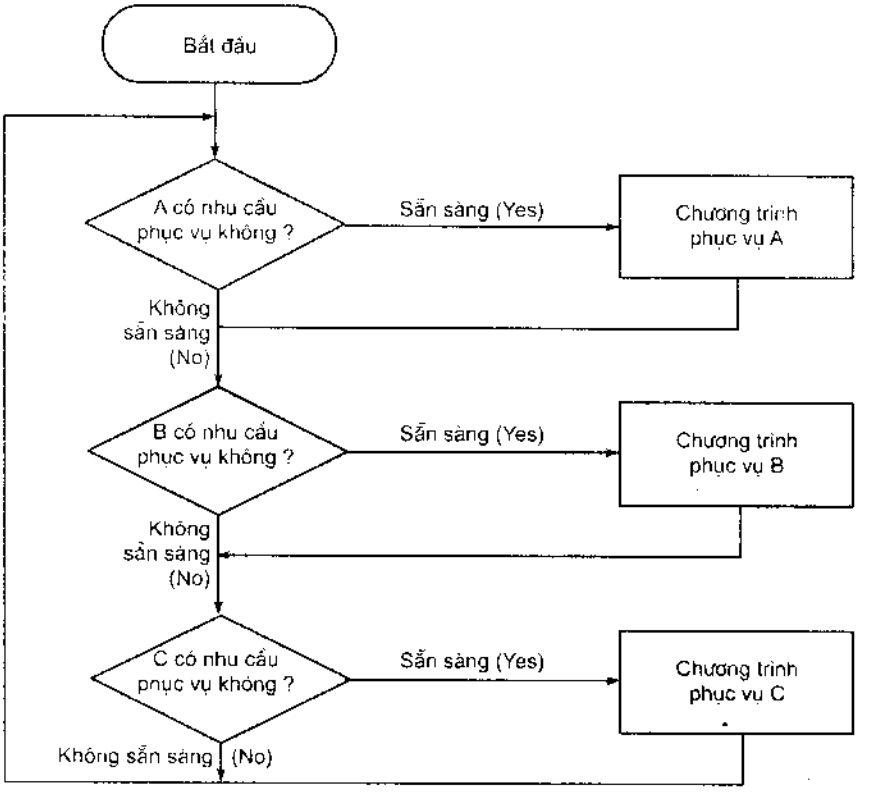
• Khối phần cứng điều khiển DMA rất phức tạp.

6.2. VÀO/RA DỮ LIỆU BẰNG PHƯƠNG PHÁP THĂM DÒ

Trạng thái sẵn sàng của thiết bị nói lên yêu cầu của thiết bị muốn được bắt đầu phục vụ vào/ra dữ liệu hay được tiếp tục quá trình vào/ra dữ liệu đang dang dở. Ví dụ, đĩa mềm đã ở trong ổ FDD, cần khóa của FDD đã nhả.

6.2.1. Vào/ra dữ liệu bằng phương pháp thăm dò không theo ưu tiên

Hình 6.2 mô tả quá trình điều khiển vào/ra dữ liệu bằng thăm dò trong trường hợp thăm dò không theo ưu tiên.



Hình 6.2: Điều khiển vào/ra bằng thăm dò không theo ưu tiên

Sự thăm dò của chương trình được bắt đầu bằng xem thiết bị A có nhu cầu được phục vụ hay không (A need service). Mỗi một thiết bị đều có khối logic điều khiển vào/ra, mà trong đó thường phải có thanh

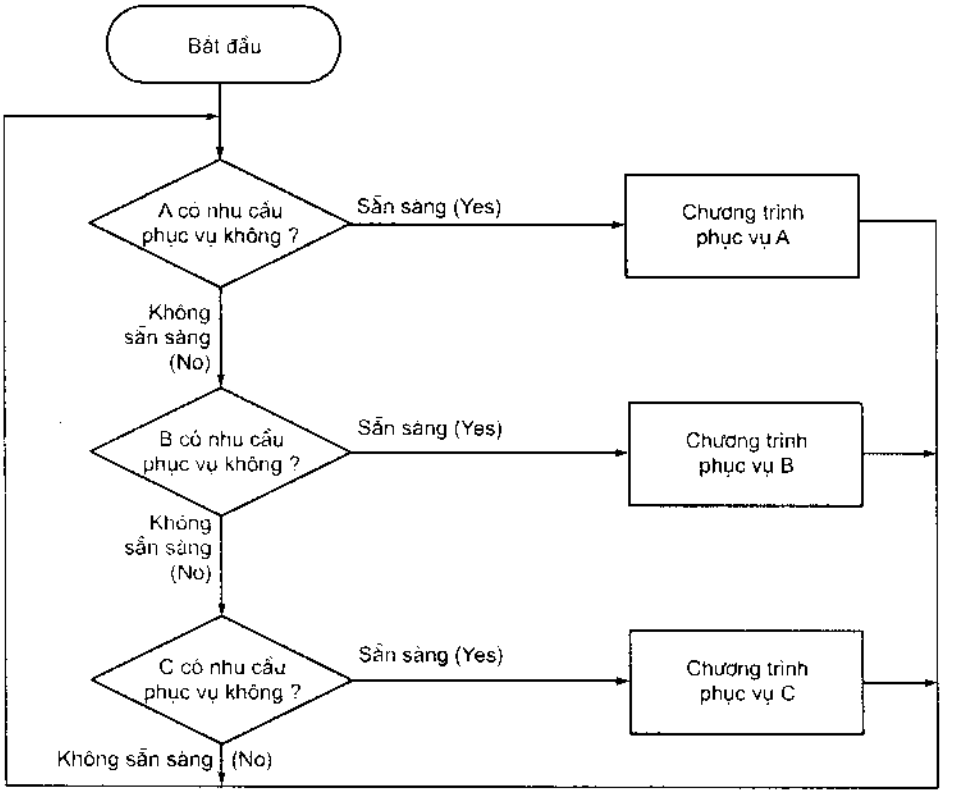
ghi trạng thái với các bit cờ mà giá trị của chúng trỏ các trạng thái của thiết bị. Thanh ghi trạng thái có địa chỉ vào/ra và chương trình thăm dò phải đọc nội dung của thanh ghi trạng thái để kiểm tra. Sự sẵn sàng (Yes) hay không sẵn sàng (No) của thiết bị được đánh giá qua các giá trị các bit này. Nếu Yes thì chuyển đến chương trình phục vụ vào/ra dữ liệu cho thiết bị A (service routine). Nếu thiết bị A không có nhu cầu phục vụ (No) hoặc kết thúc phục vụ thì chuyển qua thăm dò tiếp thiết bị B, quá trình tương tự với các thiết bị còn lại. Chỉ khi thăm dò hết các thiết bị thì vòng thăm dò mới quay trở lại để bắt đầu vòng thăm dò tiếp theo. Thiết bị A là thiết bị đầu và thiết bị C là thiết bị cuối cùng của một vòng thăm dò. Phụ thuộc vào đặc điểm của thiết bị ngoại vi mà số vòng (số lần) thăm dò nhiều hay hết. Ví dụ với con chuột thì số lần thăm dò trong 1 giây ít hơn nhiều so với thiết bị đĩa mềm (FDD) và thiết bị đĩa cứng (HDD). Phương pháp thăm dò này phù hợp trong các hệ thống vi xử lý nhỏ chỉ có một số rất ít thiết bị ngoại vi và thường chúng tương tự nhau về các đặc điểm vào/ra dữ liệu như: song song hay nối tiếp, tốc độ vào/ra, v.v... để đơn giản và tiết kiệm cho thiết kế phần mềm điều khiển vào/ra dữ liệu chung bằng phương pháp này.

Trong vào/ra ra dữ liệu nối tiếp, ví dụ nếu sử dụng mạch UART 8250 để điều khiển vào/ra truyền thông với máy tính khác qua cổng COM1 (3F8h-3FFh), chúng ta có thể kiểm tra nội dung các thanh ghi điều khiển và trạng thái mô-đem (3FC), thanh ghi trạng thái đường dây (3FD), để thăm dò trạng thái đường truyền có sẵn sàng cho truyền thông hay không (xem mục 5.3.4).

6.2.2 Vào/ra dữ liệu bằng phương pháp thăm dò theo ưu tiên

Khi các thiết bị được gán mức ưu tiên phục vụ, thì thiết bị có mức ưu tiên cao hơn được thăm dò trước (giả định là thiết bị A). Nếu thiết bị A sẵn sàng cần được phục vụ (Yes) thì phải chuyển đến thực hiện chương trình gọi phục vụ vào/ra thiết bị A, nếu chưa sẵn sàng (No) thì thăm do thiết bị B có mức ưu tiên phục vụ thấp hơn. Thiết bị C có mức ưu tiên phục vụ thấp nhất. Nếu phục vụ vào/ra xong thiết bị A, thì

quá trình thăm dò lại bắt đầu với thiết bị có mức ưu tiên cao nhất, như vậy, nếu thiết bị A lại có yêu cầu được phục vụ thì nó sẽ được tiếp tục phục vụ, mặc dù thiết bị B đang sẵn sàng và có yêu cầu phục vụ (hình 6.3). Bắt cứ một thiết bị nào được phục vụ và/ra dữ liệu xong đều quay trở lại để bắt đầu vòng thăm dò mới từ thiết bị có mức ưu tiên cao nhất.



Hình 6.3: Điều khiển vào/ra bằng thăm dò theo ưu tiên, thiết bị A có ưu tiên cao nhất, thiết bị C có ưu tiên thấp nhất

Cho rằng vi xử lý có nhịp đồng hồ là 50 MHz, số lượng các chu kỳ nhịp đồng hồ cho thao tác một vòng lặp (một lần) thăm dò của chương trình là 100. Ta xét thử đối với một số thiết bị ngoại vi như sau:

a) Con chuột (mouse)

Đối với con chuột cần thực hiện thăm dò 30 lần trong 1 giây, như vậy:

- Số chu kỳ nhịp đồng hồ cho thăm dò con chuột: $30 \times 100 = 300$.

- Tỷ lệ sử dụng chu kỳ nhịp đồng hồ để thăm dò: $3000/(50 \times 10^6) = 0{,}006\%$.

b) Ổ đĩa mềm (FDD)

Đối với ổ đĩa mềm truyền dữ liệu 16-bit đến vi xử lý với tốc độ là 50 kbit/s, thì:

- Số lượng truy nhập thăm dò (polling access) trong 1 giây (nếu không muốn mất dữ liệu) sẽ là: (50 kbit/s)/(2 byte/truy nhập) = 25 kbyte truy nhập thăm dò/s.

- Số nhịp đồng hồ để thăm dò: $25\ k \times 100 = 25 \times 1024 \times 100 = 25{,}6 \times 10^5$.

- Tỷ lệ sử dụng nhịp đồng hồ cho thăm dò: $(25{,}6 \times 10^5)/(50 \times 10^6) = 5\%$.

c) Ổ đĩa cứng (HDD)

Đối với ổ đĩa cứng truyền dữ liệu 16-bit tốc độ 2 MB/s thì:

- Số lượng truy nhập thăm dò:

(2 MB/s)/(4 byte/truy nhập) = 500 k truy nhập thăm dò/s

- Số nhịp đồng hồ để thăm dò: $500\ k \times 100 = 51{,}2 \times 10^6$

- Tỷ lệ sử dụng nhịp đồng hồ cho thăm dò: $(51{,}2 \times 10^6)/(50 \times 10^6) = 100\%$

6.3. VÀO/RA DỮ LIỆU BẰNG PHƯƠNG PHÁP NGẮT

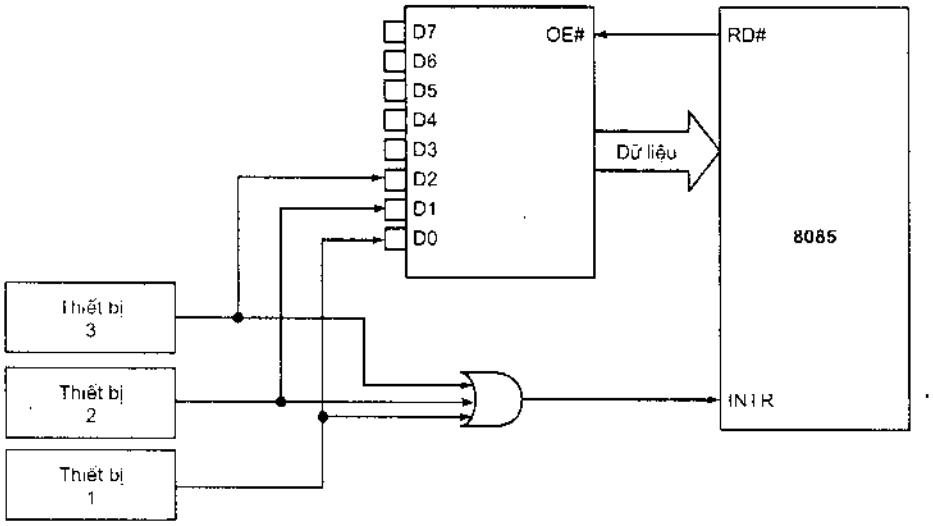
Trong các hệ thống vi xử lý, ngắt là một sự kiện gây tác động đến CPU để CPU dừng quá trình xử lý lệnh đang diễn ra bình thường và chuyển sang xử lý ngắt. Nhưng ngắt không chỉ do thiết bị bên ngoài tác động vào nhằm chủ yếu để thực hiện vào/ra dữ liệu mà còn có thể phát sinh nội tại trong quá trình thực hiện các lệnh máy của CPU, ví dụ: các phát sinh ngoại lệ như lỗi chia cho 0, mã lệnh sai, lỗi chẵn lẻ của DRAM, và các lệnh ngắt mềm (INT X), v.v...

Phân biệt ngắt cứng và ngắt mềm. Ngắt mềm được phát sinh do lệnh ngắt INT X và các phát sinh ngoại lệ. Ngắt cứng là do tác động của các thiết bị ngoại vi sinh ra trong các trường hợp như: sẵn sàng để vào/ra dữ liệu, lỗi vào/ra hay bản thân thiết bị có sự cố phần cứng, không sẵn sàng. Tất cả các phương pháp điều khiển ngắt của hệ thống vi xử lý làm thành hệ thống ngắt của vi xử lý.

Hệ thống ngắt của vi xử lý có 3 loại cơ bản: ngắt theo quét, ngắt theo véc-tơ, và ngắt theo chuỗi. Chúng ta khảo sát các loại điều khiển ngắt này thông qua hệ thống ngắt của 8085.

6.3.1. Hệ thống ngắt theo quét

Trong hệ thống ngắt này, các thiết bị ngoại vi được kết hợp vào mạch "hoặc" (OR) cho một đường dây ngắt dẫn đến chân tín hiệu ngắt (INTR) của CPU. Hình 6.4 mô tả kỹ thuật ngắt này của 8085.



Hình 6.4: Xử lý ngắt bằng phương pháp quét

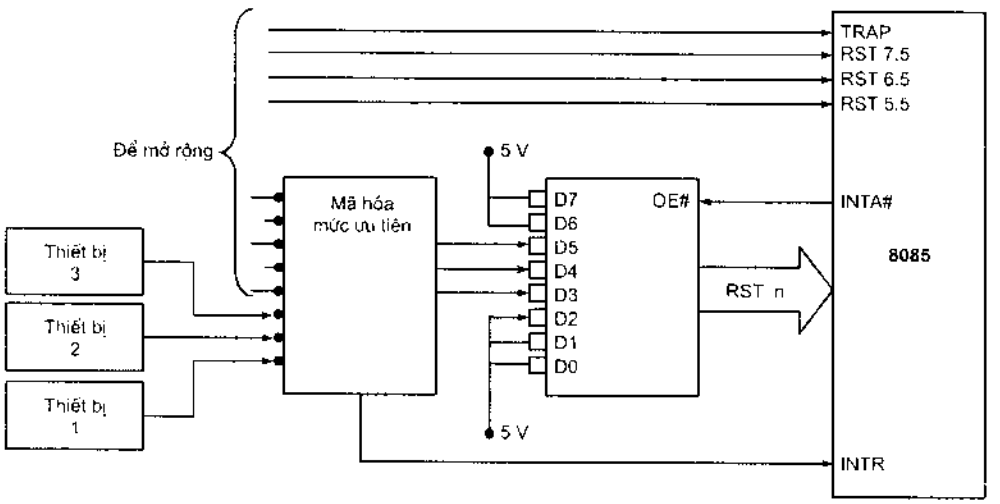
Khi một hoặc một số thiết bị này đưa ra ngắt, máy tính thực hiện quét (scan) các thiết bị để xác định ngắt xuất phát từ đâu. Mức ưu tiên được tự động gán cho các thiết bị theo thứ tự quét. Một từ trạng thái ngắt thông qua mạch 8282 được đọc vào CPU. Bằng phương pháp mặt

nạ hoặc sử dụng các lệnh quay vòng, thiết bị ngắt có mức ưu tiên cao nhất được chọn (polling) và một địa chỉ của chương trình phục vụ ngắt đúng cho thiết bị được chọn được tạo ra.

Kỹ thuật này thường được sử dụng cho những thiết bị tốc độ trao đổi dữ liệu chậm và trong hệ thống ngắt đơn, chỉ có một véc-tơ ngắt được sử dụng.

6.3.2. Hệ thống ngắt theo véc-tơ

Xử lý ngắt theo véc-tơ là phương pháp mà trong đó thiết bị yêu cầu được phục vụ có mức ưu tiên cao nhất đưa ra một véc-tơ ngắt để trỏ tới chương trình phục vụ ngắt tương ứng. Bộ vi xử lý không cần thiết phải mất thời gian quét các thiết bị để tìm kiếm nguồn gây ra ngắt. Mỗi một thiết bị có đường ngắt riêng và mức ưu tiên được xác định bằng mạch phần cứng hay bằng phần mềm. Số lượng các thiết bị thực hiện ngắt theo véc-tơ bị hạn chế bởi các mức ngắt. Hệ thống xử lý ngắt theo véc-tơ được áp dụng trong những hệ thống máy tính có nhiều chủng loại thiết bị I/O ghép nối và thường cho các thiết bị trao đổi dữ liệu nhanh. Hình 6.5 là ví dụ của hệ thống ngắt theo véc-tơ của 8085.



Hình 6.5: Xử lý ngắt bằng phương pháp véc-tơ

Như đã trình bày, để thực hiện xử lý ngắt theo véc-tơ, 8085 có các lệnh RST n (restart) và các đường tín hiệu vào đặc biệt RST 7.5, RST 6.5 và RST 5.5. Chúng ta đã xét sự trỏ véc-tơ tới chương trình con phục vụ ngắt của các đường vào ngắt RST 7.5, RST 6.5 và RST 5.5.

Khi một hoặc nhiều hơn thiết bị đưa ra yêu cầu phục vụ, chúng làm cho các đường vào của mạch mã hóa mức ưu tiên xuống mức thấp, thì một cửa ra của mạch mã hóa này chuyển lên mức cao, đồng thời, mạch mã hóa cũng tạo ra tín hiệu yêu cầu ngắt INTR. Tổ hợp vào 3 bit của mạch chốt đệm 8282 tương ứng với yêu cầu ngắt có mức ưu tiên cao nhất kết hợp với các giá trị bit còn lại của 8282 (tất cả = 1) tạo ra mã lệnh RST n (restat) ở các cửa ra.

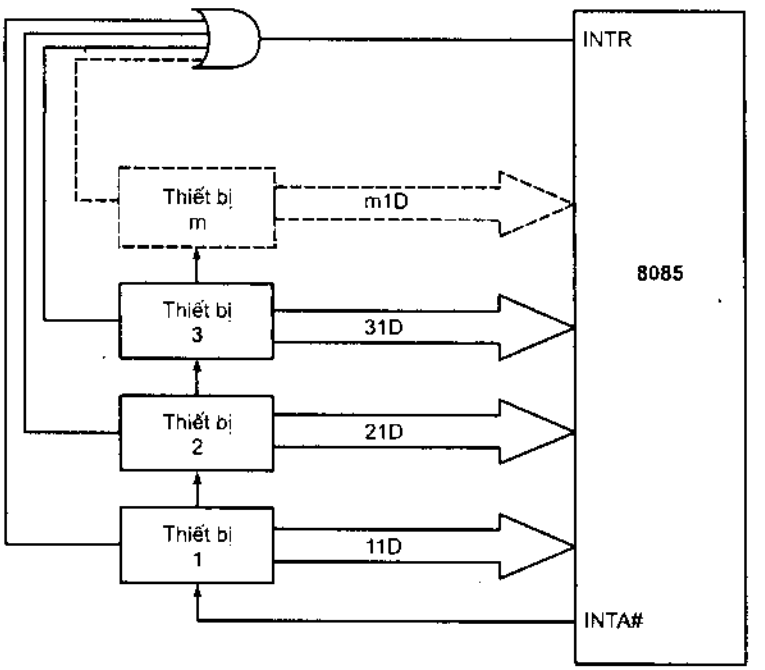
Trong thời gian đọc mã lệnh ngắt, INTA# đọc mã lệnh RST n vào thanh ghi lệnh, và CPU trỏ véc-tơ tới chương trình con phục vụ ngắt tương ứng. Hệ thống ngắt theo véc-tơ có 5 cửa vào ngắt theo mức ưu tiên (xem hình 6.46), trong đó TRAP có mức ưu tiên cao nhất và INTR có mức ưu tiên thấp nhất.

Hệ thống ngắt theo véc-tơ có thể được tổ chức đơn mức với một véc-tơ ngắt, hoặc nhiều mức và có tới 12 véc-tơ ngắt.

6.3.3. Hệ thống ngắt theo chuỗi liên tiếp

Trong hệ thống này, các thiết bị ngoài kết nối trên một chuỗi liên tiếp sao cho yêu cầu ngắt (INTR) từ một hoặc nhiều thiết bị trong chuỗi gây cho CPU gọi tới một chương trình phục vụ ngắt. Chương trình phục vụ ngắt tạo ra tín hiệu chấp nhận ngắt (INTA#) lan truyền qua chuỗi đến khi gặp thiết bị gây ngắt đầu tiên thì dừng lại. Tại đây, thiết bị này đưa lên bus dữ liệu một số xác nhận. Số xác nhận này được CPU đọc vào và sử dụng để tạo địa chỉ của chương trình con phục vụ ngắt thích ứng cho thiết bị ngắt. Mức ưu tiên ngắt được tự động gán cho các thiết bị theo thứ tự. Một thí dụ về hệ thống ngắt theo chuỗi cho trong hình 6.6. Trong đó, thiết bị 1 có mức ưu tiên cao nhất theo thứ tự

kết nối cứng trên chuỗi. Khi thiết bị 1 đưa ra yêu cầu ngắt, thì chuỗi cho tín hiệu INTA# bị chặn tại nó, không thể đi tiếp đến thiết bị 2 được. Tương tự, khi thiết bị 1 và thiết bị 2 không có yêu cầu ngắt, còn thiết bị 3 có ngắt, thì thiết bị 3 mới có thể tiếp nhận được tín hiệu INTA# đi qua thiết bị 1 và thiết bị 2. Các đường dây ngắt của từng thiết bị kết nối qua mạch "OR" để tạo ra tín hiệu INTR cho CPU.



Hình 6.6: Hệ thống ngắt theo chuỗi liên tiếp

Hệ thống ngắt theo chuỗi có tốc độ xử lý trung bình, cao hơn hệ thống ngắt theo quét và chậm hơn hệ thống ngắt theo véc-tơ. Những thiết bị có tốc độ cao thường được đặt ở đầu chuỗi, nơi có mức ưu tiên ngắt cao nhất.

6.3.4.Các bước xử lý ngắt và cấu trúc ngắt của 8085

Chúng ta đã xét đến quá trình gọi tới chương trình con. Đó là một quá trình chủ động của người lập trình, hay người sử dụng mong muốn

máy tính phải thực hiện. Mỗi chương trình con thường thực hiện một chức năng hay nhiệm vụ cố định nào đó. Nhưng do hệ thống máy tính gồm nhiều chủng loại đơn vị logic và thiết bị I/O khác nhau cùng hoạt động và chúng có nhu cầu được phục vụ không định (không đồng bộ) trước cùng một lúc hoặc không cùng lúc, nên CPU cần phải có sự phân phối thời gian phục vụ chúng bất kể lúc nào. Một cơ chế phục vụ tiết kiệm thời gian và hiệu quả chính là cơ chế xử lý ngắt.

Mỗi sự kiện (như: lỗi xảy ra của thiết bị I/O hay trong quá trình tính toán, yêu cầu phục vụ của các đơn vị thiết bị I/O) được gán cho một ngắt, hay còn gọi là yêu cầu phục vụ. Mỗi một sự kiện ngắt được một chương trình con phục vụ ngắt thực hiện giải quyết. Như vậy, khi xuất hiện một sự kiện ngắt, CPU phải gọi tới chương trình con định trước cho sự kiện ngắt này để thực hiện. Để phục vụ ngắt theo thứ tự cần phải có cơ chế xử lý mức ưu tiên phục vụ. Do đó mỗi một ngắt cần phải được gán một mức ưu tiên. Khi có nhiều thiết bị kết nối với CPU thông qua các ngắt thì thiết bị gây ra ngắt có mức ưu tiên cao nhất sẽ được phục vụ trước tiên. Để tiếp nhận yêu cầu ngắt từ thiết bị I/O, 8085 có chân vào tín hiệu INTR (chân 10). Tín hiệu INTR là yêu cầu ngắt che được.Sử dụng tín hiệu này, quá trình xử lý ngắt của 8085 được thực hiện qua 4 bước:

Bước 1: Cho phép và cấm ngắt

Sau khi được khởi động, chân tín hiệu INTR được tự động cách ly với bên trong (bị che) nhằm ngăn chặn ngắt không mong muốn ban đầu. Để người lập trình có thể kiểm soát được ngắt, 8085 có các lệnh cho phép ngắt (EI - Enable Interrupt) và cấm ngắt (DI - Disable Interrupt). Lệnh EI thiết lập trạng thái bật cho trigger EI bên trong 8085, còn lệnh DI xóa trigger EI. Để có thể thực hiện được ngắt (ngắt không bị cấm) trước hết phải thực hiện lệnh EI. Lệnh EI phải được đặt trước lệnh ngắt trong chương trình. Nếu muốn cấm ngắt thì phải thực hiện lệnh DI. Mã lệnh EI và DI như sau:

EI (Enable interrupt)

1	1	1	1	1	1	0	1

Bật trigger EI bên trong 8085

Số chu kỳ may: 1, số trạng thái: 4

DI (Disable interrupt)

1	1	1	1	1	1	0	1

Xóa trigger EI bên trong 8085

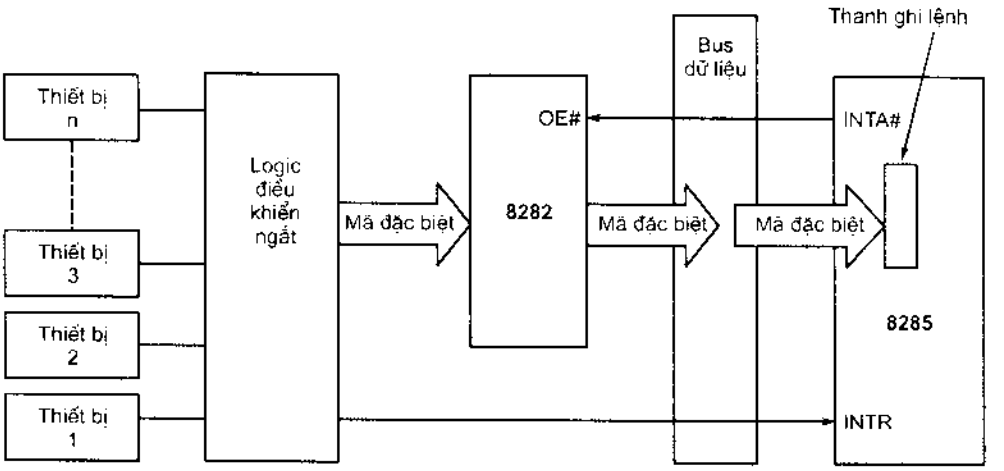
Số chu kỳ máy: 1, số trạng thái: 4

Bước 2: Yêu cầu ngắt

Thiết bị ngoài đưa yêu cầu ngắt chân tín hiệu INTR của 8085 khi nó cần được phục vụ. INTR xuất hiện vào bất kỳ thời điểm nào, nó không thể đồng bộ theo đồng hồ nhịp được mà được 8085 chốt giữ bên trong bằng trigger ngắt ở các giai đoạn cuối cùng của chu kỳ máy của lệnh đang thực hiện dang dở, vì vào thời điểm INTR chuyển lên mức cao (xuất hiện yêu cầu ngắt) có một lệnh máy đang thực hiện. INTR phải duy trì mức tích cực cao trong suốt các giai đoạn cuối cùng của chu kỳ máy của lệnh đang thực hiện.

Bước 3: Gọi và thực hiện chương trình con phục vụ ngắt

Khi lệnh máy dang dở được thực hiện xong, CPU kiểm tra trạng thái của trigger ngắt. Nếu trigger này được bật (vì có INTR) thì CPU tạm dừng chương trình đang thực hiện và khởi động một chu kỳ máy chấp nhận ngắt (interrupt acknowledge machine cycle). Chu kỳ máy này rất giống với chu kỳ đọc mã lệnh, nhưng thay vì RD# thì nó sinh ra INTA# = 0 (chấp nhận ngắt). Sự chuyển INTA# xuống mức thấp điều khiển mạch logic ngắt bên ngoài đưa lên bus dữ liệu một mã lệnh đặc biệt. Mã đặc biệt này được CPU đọc vào thanh ghi lệnh, và tại thanh ghi lệnh nó được giải mã, rồi được thực hiện như những lệnh bình thường khác. Toàn bộ mạch logic xử lý ngắt cho trong hình 6.7.



Hình 6.7: Hệ thống ngắt chung của 8085

Trong hệ thống này chỉ có duy nhất một cửa vào yêu cầu ngắt (INTR), do đó không cần phải có địa chỉ cho cổng ngắt. Mã đặc biệt được tạo ra chỉ có độ dài 1 byte, nhưng nhờ nó vẫn nhảy tới được chương trình con phục vụ ngắt và sau khi phục vụ xong trở lại được chương trình chính. Đây là một khác biệt với lệnh gọi bình thường với 3 bytes (CALL addr). Mã đặc biệt này là lệnh CALL chỉ có 1 byte. Nó được ký hiệu là RST n (restart) và có cấu trúc như sau:

RST n

1	1	N	N	N	1	1	1

(SP): đỉnh ngăn xếp xếp trước khi thực hiện lệnh

(SP) - 1 ← (PCH), (SP) - 2 ← (PCL)

(SP) ← (SP) - 2: đỉnh ngăn xếp sau khi thực hiện lệnh

(PC) ← 8 x (NNN): địa chỉ của chương trình con

Số chu kỳ máy: 3, số trạng thái: 12

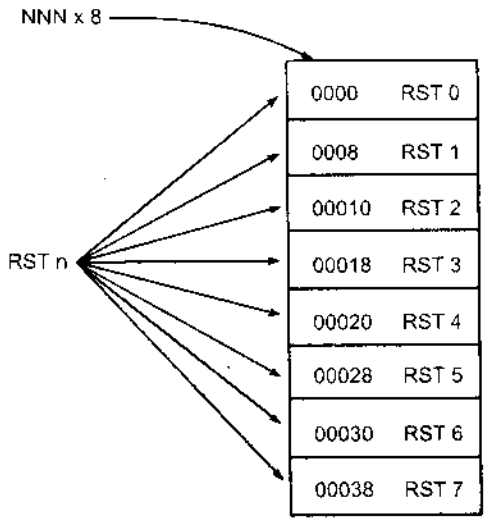
Đánh địa chỉ: gián tiếp thanh ghi

Tổ hợp 3 bit NNN trong mã lệnh RST n được nhân với 8 để xác định địa chỉ đầu của chương trình phục vụ ngắt. Như vậy có 8 điểm của vùng nhớ được trỏ trực tiếp tới (véc-tơ trỏ tới). Vùng nhớ được trỏ tới gọi là vùng véc-tơ ngắt (hình 6.8), còn giá trị NNN x 8 gọi là giá trị

véc-tơ ngắt. Khi thực hiện RST n, thanh đếm lệnh (PC) nhận giá trị NNN x 8 sẽ có nội dung như sau:

Nội dung PC sau khi thực hiện RST n (restart):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0



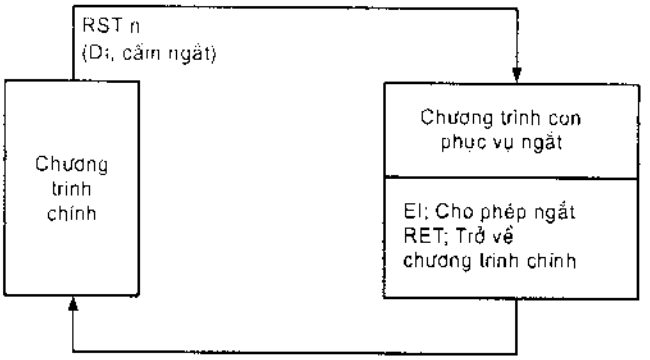
Hình 6.8: 8 vùng nhớ được trỏ véctơ bằng NNN x 8 trong lệnh RST n

Cũng giống như các lệnh CALL bình thường khác, lệnh RST n cần cất giữ địa chỉ trở về từ chương trình phục vụ ngắt, và trong chương trình con phục vụ ngắt có các lệnh PUSH/POP với ngăn xếp. Khi CPU nhảy tới chương trình phục vụ ngắt, hệ thống ngắt bên ngoài phải được cách ly bằng lệnh DI. Điều này ngăn chặn thiết bị bên ngoài (có thể là cùng một thiết bị đã ngắt) gây ra ngắt CPU đang thực hiện chương trình phục vụ ngắt.

Bước 4: Trở về chương trình chính

Bởi vì lệnh RST n thực hiện giống như lệnh CALL nên cũng sử dụng lệnh RET ở cuối chương trình con phục vụ ngắt để trở về chương

trình chính, và trước lệnh RET phải có lệnh EI để cho phép ngắt trở lại (hình 6.9).



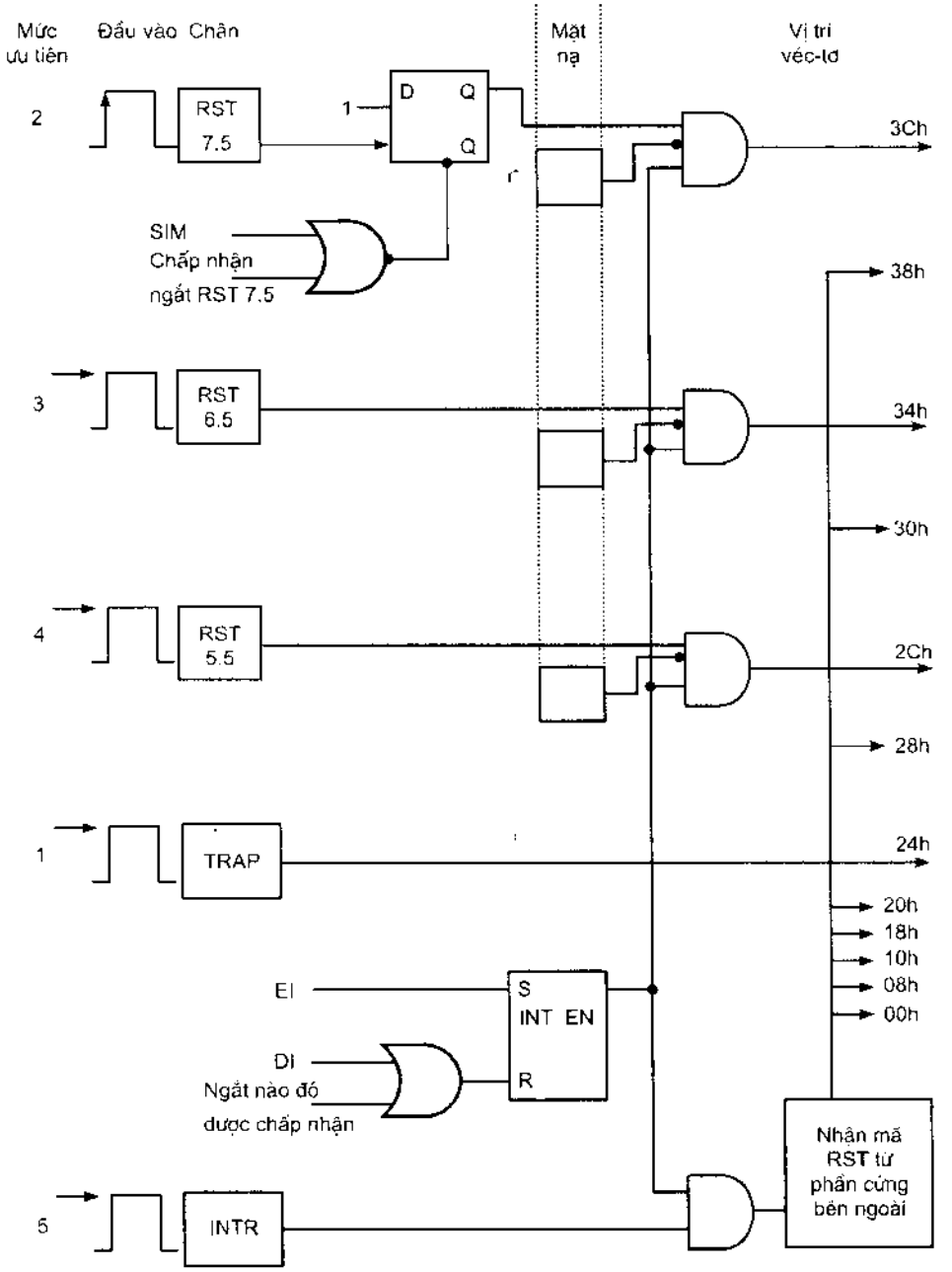
Hình 6.9: Trả lời ngắt là lời gọi đến chương trình con

Đối với các hệ thống máy tính 8085 nhỏ, ngoài chân tín hiệu INTR, 8085 còn có thêm 4 chân vào ngắt bổ sung: RST 7.5, RST 6.5, RST 5.5, và TRAP (RST 4.5). Những tín hiệu này là những tín hiệu yêu cầu ngắt đặc biệt. Chúng tự động đưa các lệnh RST vào trong thanh ghi lệnh. Chúng đặc biệt vì, không giống như yêu cầu ngắt INTR, chúng không đòi hỏi phần cứng bổ sung, không cần phải chấp nhận ngắt (INTA#), bởi vì tất cả đều tích hợp bên trong chip 8085. Kể cả INTR (pin 10) và INTA# (pin 11), toàn bộ hệ thống ngắt của 8085 có 6 chân tín hiệu. Các chân tín hiệu ngắt RST và giá trị véc-tơ (địa chỉ-chương trình được trỏ tới) như sau:

TRAP (RST 4.5):	dóng thẳng hàng chân 6, địa chỉ khởi động lại (hex): 0024 (4.5 × 8)
RST 5.5:	chân 9, địa chỉ khởi động lại: 002C (5.5 × 8)
RST 6.5:	chân 8, địa chỉ khởi động lại: 0034 (6.5 × 8)
RST 7.5:	chân 7, địa chỉ khởi động lại: 003C (7.5 × 8)
INTR:	chân 10, yêu cầu ngắt
INTA#:	chân 11, chấp nhận ngắt

Khi một trong những chân tín hiệu này chuyển lên mức tích cực "high" thì nó sẽ khởi động một chu kỳ máy RST, và gọi tới một vùng nhớ mà địa chỉ đầu của vùng nhớ này bằng NNN x 8. Ví dụ, nếu tín

hiệu RST 6.5 chuyển lên mức "1" thì 8085 sẽ tự động gọi tới vùng nhớ có địa chỉ là 52 hay 34h.



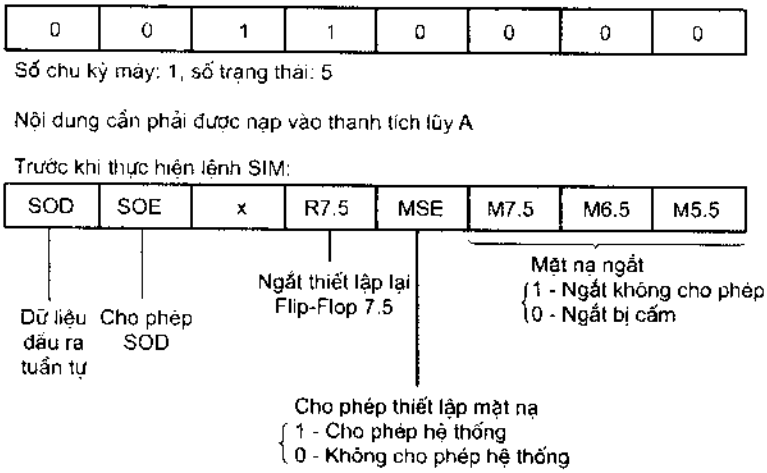
Hình 6.10: Toàn bộ cấu trúc ngắt của 8085

Toàn bộ cấu trúc ngắt của 8085 cho trong hình 6.10.

Đặc tính của cấu trúc ngắt theo các mức ngắt này như sau:

• Mỗi một đầu vào: RST 5.5, RST 6.5, và RST 7.5 có một mặt nạ (cấm) có thể lập trình được thông qua lệnh SIM (thiết lập mặt nạ ngắt) và thông qua lệnh RIM (đọc mặt nạ ngắt) để xác định trạng thái ngắt.

SIM (mã 30h)

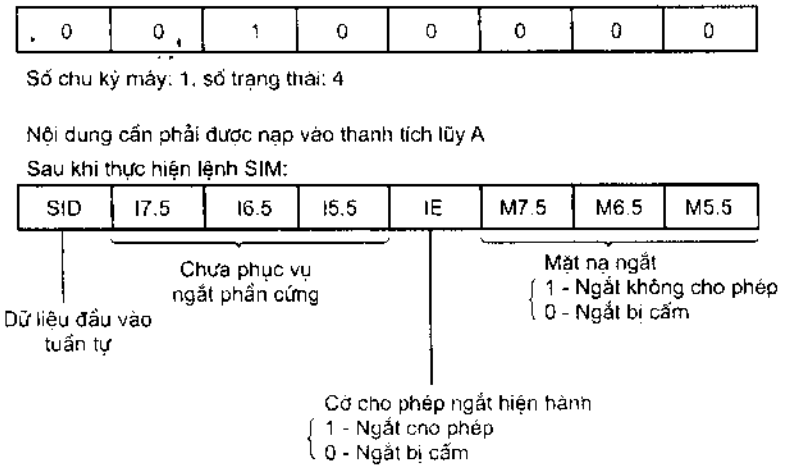


Hình 6.11: Mã lệnh SIM và nội dung nạp vào thanh tổng trước khi thực hiện lệnh SIM

• Để lập trình các mặt nạ ngắt, trước hết đặt bit 3 (MSE) của thanh tích luỹ A bằng 1, và 1 cho bất kỳ bit nào trong các bit 0 (M5.5), bit 1 (M6.5), và 2 (M7.5) để cách ly các ngắt tương ứng. Sau đó thực hiện một lệnh SIM. Nếu bit 3 (MSE) của thanh tích lũy A có giá trị 0 khi SIM được thực hiện thì thanh ghi mặt nạ ngắt không bị thay đổi. Nếu bit 4 (R7.5) của thanh tích luỹ A bằng 1 khi SIM được thực hiện thì mạch chốt tín hiệu RST 7.5 (D trigger) bị xóa. Tín hiệu vào RESETIN# của 8085 khi xuống mức tích cực "thấp" luôn thiết lập các bit mặt nạ M5.5, M6.5, và M7.5 bằng 1. Nếu bit 6 (SOE) của thanh tích luỹ A đặt ở mức 1 khi SIM được thực hiện thì giá trị bit 7 (SOD) được nạp vào mạch chốt SOD và như vậy nó trở thành giao tiếp cho

thiết bị ngoài. Mạch chốt SOD không bị ảnh hưởng bởi lệnh SIM nếu bit 6 (SOE) được đặt bằng 0. SOD luôn bị xóa khi có RESETIN# = 0.

RIM (mã 20h)



Hình 6.12: Mã lệnh RIM và nội dung của thanh tích luỹ A sau khi thực hiện lệnh RIM

• Lệnh RIM (Read Interrupt Mask) nạp vào thanh tích luỹ A dữ liệu có cấu trúc như trong hình 6.12, trong đó: Các bit trạng thái mặt nạ ngắt (M7.5, M6.5, M5.5) có giá trị bằng 0 - ngắt cho phép, bằng 1- ngắt bị cấm; bit cờ cho phép ngắt hiện hành (IE) bằng 1- ngắt đang được phép, bằng 0- ngắt bị cấm, ngoại trừ TRAP (vì TRAP là ngắt không che được bằng mặt nạ); các bit I7.5, I6.5, và I5.5 nếu bằng 1 thì chúng báo rằng, các tín hiệu ngắt RST 7.5, RST 6.5 và RST 5.5 đã được CPU nhận (Interrupt pending) nhưng chưa được phục vụ. Bit I7.5 (bit 6) được nạp bởi trạng thái của trigger RST 7.5. Triger này luôn được bật bằng nhờ tín hiệu ở chân vào RST 7.5 ngay cả khi ngắt đã bị che mặt nạ trước đó.

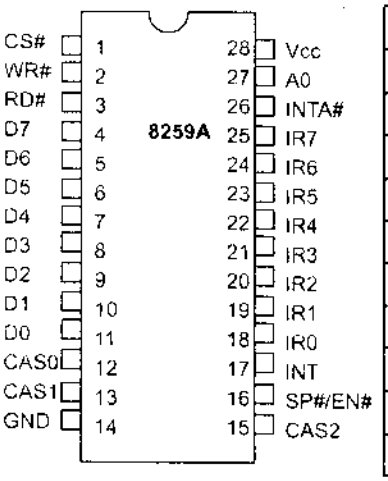
• Một khi chu kỳ ngắt được khởi động, tất cả các ngắt (ngoại trừ TRAP) bị cách ly cho đến khi lệnh EI được thực hiện.

• Các tín hiệu RST 5.5 và RST 6.5 được cảm nhận bằng điện thế cao và phải được duy trì ở mức tích cực "cao" trong cả quá trình của chu kỳ nhịp cuối cùng của lệnh máy đang thực hiện.

• Tín hiệu RST 7.5 được cảm nhận bằng sườn xung lên cao và một xung ngắn cao đủ để thiết lập trigger bên trong, mà trigger này nhờ đó tạo ra một yêu cầu ngắt. Trigger duy trì trạng thái bật cho đến khi yêu cầu được phục vụ. Khi đó trigger được tự động xóa. Nó cung có thể được xóa nhờ lệnh RIM.

• TRAP không bị ảnh hưởng bởi mặt nạ ngắt và lệnh EI, DI. TRAP được coi là ngắt không bị che (NMI). Nó được cảm nhận cả theo sườn xung và điện thế cao. Nó phải được duy trì ở mức điện thế cao để được CPU có thể chấp nhận được. Nó chỉ được chấp nhận 1 lần trong khi duy trì ở mức cao. Sự trở về mức điện thế thấp và sau đó trở lại bằng sườn đi lên được coi lần ngắt tiếp theo.

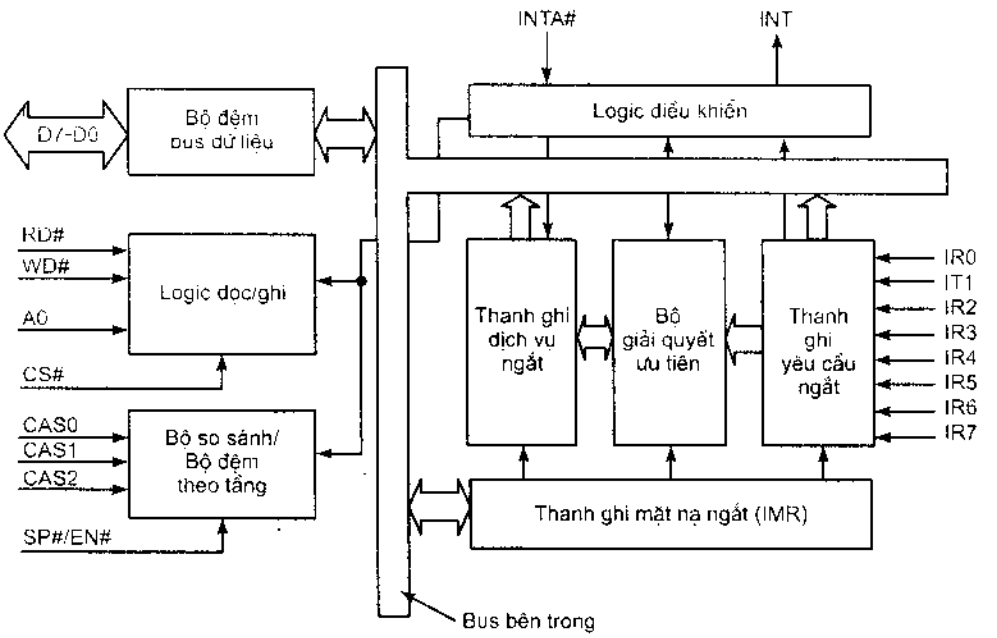
6.3.5. Vi mạch điều khiển ngắt 8259A



Tên tín hiệu	Chức năng
D0 - D7	Bus dữ liệu, hai chiều
CS#	Đầu vào chọn chip
RD#	Đầu vào đọc
WR#	Đầu vào ghi
A0	Địa chỉ chọn lệnh
CAS2-CAS0	Các đường chọn cascade
SP#/EN#	Chương trình thợ/Cho phép bộ đệm
INT	Đầu ra ngắt
INTA#	Đầu vào chấp nhận ngắt
IR0-IR7	Các đầu vào yêu cầu ngắt

Hình 6.13: Đóng vỏ, các tín hiệu của 8259A

Trong một hệ thống vi xử lý thường phải cần khối chức năng điều khiển ngắt, phục vụ yêu cầu vào/ra của thiết bị ngoại vi. Chúng ta đã nói đến vấn đề xử lý khi xét đến các bộ vi xử lý Intel 8-bit (8080, 8085) và 16-bit (8086, 8088, 80286). Để đảm nhiệm chức năng điều khiển ngắt trong hệ thống vi xử lý, thuận tiện hơn cả là có một hoặc vài chip vi mạch chuyên dụng. Chip 8259A được thiết kế chuyên dụng cho chức năng này (hình 6.13, 6.14).



Hình 6.14: Sơ đồ khối chức năng của 8259A

a. Các khối chức năng và tín hiệu

(1) Thanh ghi mặt nạ ngắt IMR (Interrupt Mask Register)

Tám yêu cầu ngắt (yêu cầu được phục vụ) từ ngoại vi trên tám đường IR7-IR0. Tất cả các đường này tương ứng với các bit trong IMR. Nếu bit tương ứng với đường dây IRn (n = 0 ... 7) trong IMR thiết lập "0" thì yêu cầu ngắt tương ứng được cho qua để xử lý, ngược lại nếu bit đó = 1 thì yêu cầu ngắt tương ứng bị bỏ qua.

(2) Thanh ghi yêu cầu ngắt IRR (Interrupt Request Register) và thanh ghi phục vụ ngắt ISR (Interrupt Service Register)

Các yêu cầu ngắt ở các đường dây IR7-IR0 không bị che mặt nạ được bởi 2 thanh ghi IRR và ISR. Các thanh ghi này xử lý lần lượt theo tầng (cascade). IRR được sử dụng để cất giữ tất các yêu cầu phục vụ (không bị che mặt nạ) và ISR được sử dụng để cất giữ tất cả các yêu cầu ngắt đang được phục vụ. Tương ứng với một yêu cầu ngắt có 1 bit

của IRR. Khi có một đường IRX (X = 0,..., 7) nào đó đưa lên "1" thì 1 bit IRR tương ứng được thiết lập và tín hiệu INT chuyển lên mức "1". Tín hiệu IRX phải duy trì ở trạng thái "1" cho đến khi 8259A nhận 1 xung INTA# từ CPU đưa ra. Đồng thời 1 bit tương ứng cho yêu cầu ngắt IRX đó được thiết lập trong ISR. Bit ISR tương ứng duy trì trạng thái "1" trong khi thực hiện chương trình con phục vụ yêu cầu ngắt IRX đó cho đến khi 8259A nhận được lệnh kết thúc ngắt EOI (End Of Interrupt).

(3) Bộ giải quyết ưu tiên (Priority Resolver)

Là khối logic có chức năng xác định các mức ưu tiên của các bit thiết lập trong IRR. Bit có ưu tiên cao nhất được chọn và được định (strobed) vào trong bit tương ứng của ISR trong quá trình xung INTA# ở trạng thái "0", còn bit IRR tương ứng bị xóa đi. Đồng thời 8259 tạo ra mã lệnh gọi (CALL) tới chương trình con phục vụ ngắt này đưa lên bus dữ liệu D7-D0. CPU sẽ tạo ra 2 xung INTA# tiếp theo gửi ra 8259A để nhận từ 8259 16-bit địa chỉ cho lệnh gọi tới chương trình con.

(4) Ngắt INT

Tín hiệu ra này là tín hiệu ngắt để đưa đến chân tín hiệu ngắt INTR của vi xử lý.

(5) Báo nhận ngắt INTA# (INTerrupt Acknowledge)

Tín hiệu INTA# = "0" thứ nhất là tín hiệu chấp nhận ngắt từ CPU như là sự trả lời cho INT của 8259A. Sau đó 8259A nhận 2 xung INTA# tiếp theo sẽ tạo ra địa chỉ 16-bit của lệnh gọi (CALL) tới chương trình con phục vụ ngắt và gửi lên bus dữ liệu D7-D0 (byte địa chỉ thấp lên trước byte địa chỉ cao lên sau). Sau đó bit ISR sẽ tự động bị xóa đi, và quá trình nhắc lại cho chu kỳ ngắt tiếp theo.

(6) Logic đọc/ghi và Logic điều khiển (Control Logic)

Chức năng của các khối này là tiếp nhận các tín hiệu đưa ra từ CPU như CS#, RD#, WR#, A0, và INTA# để thực hiện các thao tác điều khiển 8259A (Bảng 6.1).

Bảng 6.1: Tổ hợp các tín hiệu điều khiển 8259A

Tổ hợp các tín hiệu						Thao tác cơ bản
A0	D4	D3	RD#	WR#	CS#	
0			0	1	0	Nội dung IRR, ISR, hoặc mã BCD của mức ngắt → bus dữ liệu
1			0	1	0	Nội dung của IMR → bus dữ liệu
0	0	0	1	0	0	Từ lệnh thao tác OCW2 từ bus dữ liệu (CPU) > 8259A
0	0	1	1	0	0	Từ lệnh thao tác OCW3 từ bus dữ liệu (CPU) > 8259A
0	1	X	1	0	0	Từ lệnh thao tác OCW1 từ bus dữ liệu (CPU) > 8259A
1	X	X	1	0	0	Từ lệnh thao tác OCW1, OCW2, OCW# từ bus dữ liệu → 8259A
X	X	X	1	1	0	Bus dữ liệu → Trạng thái trở kháng cao
X	X	X	X	X	1	Bus dữ liệu → Trạng thái trở kháng cao

(7) Bộ đệm bus dữ liệu

Đây là bộ đệm 3 trạng thái, 2 hướng, 8-bit phối ghép với bus hệ thống của hệ vi xử lý (8080, 8085, 8086/8088). Các từ điều khiển từ CPU và từ trạng thái của 8259A được truyền qua bộ đệm này.

(8) SP#/EN#

Nếu có hơn 1 chip 8259A được sử dụng trong hệ thống vi xử lý để mở rộng phục vụ ngắt. Có thể thiết lập hệ thống xử lý ngắt lên tới 64 mức ngắt (yêu cầu ngắt), tức là có thể có tới 8 chip 8259 kết nếu và xử lý theo tầng (cascade), và để chọn tầng (hay chọn chip 8259) có các các đường dây CAS2-CAS0. Trong trường hợp như vậy một chip 8259 vận hành như một chủ (master), còn các chip 8259 khác thực hiện như là các thợ (slave). Khi đó tín hiệu SP#/EN# ghim bằng = "1" ở chip 8259 chủ, còn ở các chip 8259 thợ thì SP#/EN# ghim bằng "0".

(9) Bộ so sánh/Bộ đệm theo tầng

Khối chức năng này cất giữ và so sánh các xác thực (ID) tầng của tất cả các chip 8259 có trong hệ vi xử lý. Các tín hiệu CAS2-CAS0

(mã xác thực tầng) là các tín hiệu ra khi chip 8259 được sử dụng như là chủ (SP#/EN# = "1"), và là các tín hiệu vào khi chip 8259 là thợ (SP#/EN# = "0"). Khi là chủ, 8259 gửi một ID của một thiết bị ngắt thợ lên các đường dây CAS2-CAS0. Chip 8259 thợ được chọn theo mã CAS2-CAS0 sẽ gửi địa chỉ 16-bit của chương trình con phục vụ yêu cầu ngắt của nó lên bus dữ liệu D7-D0 trong 2 khoảng thời gian xung INTA# liên tiếp gửi ra từ CPU.

b. Quá trình điều khiển ngắt của 8259A

Quá trình điều khiển và xử lý xử lý ngắt của 8259 như sau:

1. Một hoặc một số đường dây yêu cầu ngắt IRX (X = 0,..., 7) chuyển lên mức "1" (có yêu cầu được phục vụ từ ngoại vi);

2. 8259 tiếp nhận những yêu cầu ngắt và giải quyết theo ưu tiên phục vụ, gửi một INT = "1" về CPU, và chờ nhận trả lời chấp nhận ngắt từ CPU (INTA#);

3. Khi nhận được 1 xung INTA# = "0" từ CPU, 8259 sẽ tạo ra một mã lệnh CALL (11001101) đưa lên bus dữ liệu D7-D0;

4. Lệnh CALL sẽ làm cho CPU khởi tạo thêm 2 xung INTA# nữa (gọi là xung thứ 2, thứ 3) gửi tiếp về 8259;

5. Tiếp nhận hai xung INTA# liên tiếp nhau này từ CPU, 8259 sẽ gửi địa chỉ 16-bit của chương trình con phục vụ ngắt lên bus dữ liệu D7-D0. Byte thấp của địa chỉ được tạo ra ở thời gian xung INTA# thứ 2, và byte địa chỉ cao được tạo ra trong thời gian xung INTA# thứ 3;

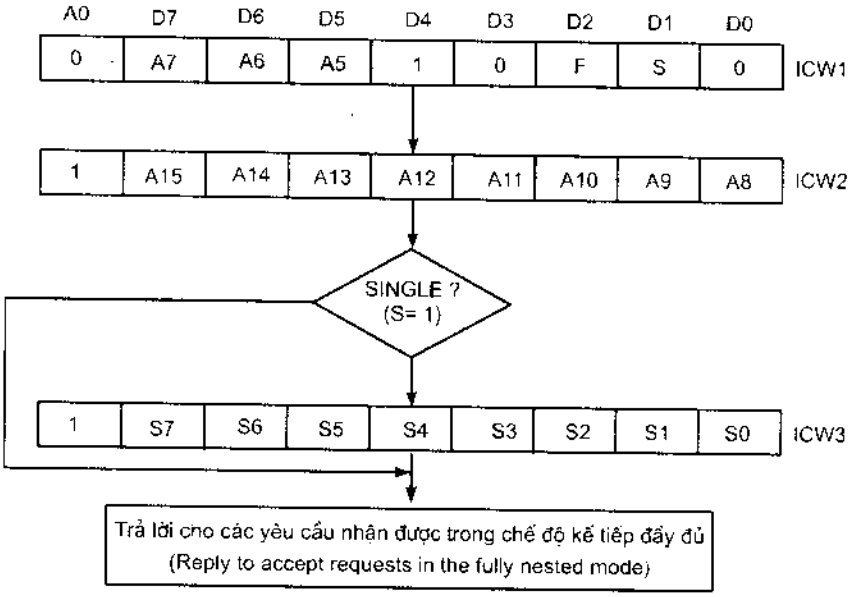
6. Như vậy toàn bộ lệnh CALL tới chương trình con gồm 3 byte được 8259 tạo ra với khoảng thời gian 3 xung INTA#;

7. Trong khi chương trình con phục vụ ngắt thực hiện thì bit ISR tương ứng không bị xóa. Chỉ khi cuối chương trình phục vụ ngắt có lệnh EOI đưa ra 8259 thì ISR sẽ bị xóa.

c. Lập trình 8259A

Chip 8259A chấp nhận 2 kiểu từ lệnh được tạo ra từ CPU, đó là:

(1) Các từ lệnh khởi tạo ban đầu ICW (Initialization Command Word)



Hình 6.15: Trình tự khởi tạo ban đầu 8259A

Trước khi thao tác bình thường bắt đầu, mỗi một chip 8259A trong hệ thống vi xử lý cần phải được đưa về trạng thái ban đầu nhờ một chuỗi xung WR# có độ dài 2 hoặc 3 byte. Chuỗi xung này được mô tả trong hình 6.15.

(2) Các từ lệnh khai thác OCW (Operation Command Word)

Những từ lệnh này là những lệnh vận hành 8259 trong những chế độ ngắt khác nhau: chế độ kế tiếp đầy đủ (Fully nested mode).

Để lập trình chip 8259A và đặc biệt là để xác định vị trí của tám véc-tơ ngắt (những địa chỉ gọi tới), ta xem xét khuôn dạng của từ lệnh ICW (hình 6.15) và khuôn dạng từ lệnh OCW. Để khởi tạo trạng thái ban đầu cho 8259A, có thể có tới 7 từ điều khiển. Trong trường hợp các từ điều khiển là các từ lệnh ICW, thì vị trí từ lệnh đầu tiên được xác định bởi thứ tự mà chúng được ghi (bit vị trí bit bên trái nhất là bit A0, nó xác định trạng thái của chân tín hiệu vào A0 khi đánh địa chỉ thanh ghi bên trong 8259).

Tám thiết bị ngoại vi có yêu cầu ngắt có tám địa chỉ véc-tơ ngắt chiếm không gian như nhau trong bộ nhớ. Các địa chỉ này có thể được lập trình với các khoảng cách 4 hoặc 8 byte. Khi ghi các từ ICW1 và ICW2 (hình 6.15), các bit của từ ICW2 (D7-D0) xác định byte cao (A15-A8) của địa chỉ véc-tơ ngắt (địa chỉ gọi đến), các bit của từ ICW1 (D7-D5) xác định các bit địa chỉ A7-A5 của địa chỉ véc-tơ ngắt. Các bit D4-D0 của từ ICW1 được 8259 tự động đưa vào các giá trị A4-A0 của địa chỉ véc-tơ ngắt. Có thể chọn khoảng cách cho địa chỉ véc-tơ ngắt 4 hoặc 8 byte nhờ bit D2 của ICW1. Các địa chỉ véc-tơ ngắt được đọc từ 8259A về CPU trong chuỗi INTA# theo khuôn dạng cho ở hình 6.16.

Nội dung của byte đầu tiên của véc-tơ ngắt

	D7	D6	D5	D4	D3	D2	D1	D0
Mã gọi:	1	1	0	0	1	1	0	1

Nội dung của byte thứ 2 của véc-tơ ngắt: byte thấp của địa chỉ véc-tơ ngắt

IR	INTERVAL = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	INTERVAL = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

Nội dung của byte thứ 3 của véc-tơ ngắt: byte cao của địa chỉ véc-tơ ngắt

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

Hình 6.16: Lập trình véc-tơ ngắt của 8259A: cấu trúc lệnh CALL (3 byte)-gọi tới chương trình con phục vụ ngắt

Khi khoảng cách chọn là 8 (bit D2 = 0 của ICW1) thì A5 được 8259 cố định. Khi khoảng cách chọn là 4 (bit D2 = 1 của ICW1) thì A5 được lập trình trong ICW1. Như vậy, chương trình con phục vụ ngắt có thể nằm ở bất cứ vùng nào của bộ nhớ.

Các bit F và S của ICW1 có ý nghĩa là:

F: Call address interval. F =1, thì khoảng cách (INTERVAL) bằng 4, F = 0, thì khoảng cách bằng 8.

S: Single, S = 1 có nghĩa là chỉ có duy nhất một chip 8259A (không cần phải lập trình từ ICW3 nữa). S = 0, có một số chip 8259A (chế độ kết nối tầng: cascade mode).

	D7	D6	D5	D4	D3	D2	D1	D0	
INTA# thứ nhất	1	1	0	0	1	1	0	1	Mã gọi
INTA# thứ hai	A7	A6	A5	1	0	1	0	0	Byte thấp của địa chỉ
INTA# thứ ba	A15	A14	A13	A12	A11	A10	A9	A8	Byte cao của địa chỉ

Hình 6.17: Ví dụ lập một khuôn dạng lệnh gọi tới chương trình con phục vụ ngắt

Ví dụ 8259A được lập trình với F = 1, và có yêu cầu ngắt mức IR5 thì cấu trúc lệnh CALL như cho trong hình 6.17.

(3) Từ lệnh khởi tạo ban đầu ICW3 (Initialization Commands Word)

Khi có chuỗi các chip 8259A kết nối theo tầng (cascade), trong đó có 1 chip là chủ (master) và các chip còn lại là thợ (slave) thì từ ICW3 mới sử dụng. Từ ICW3 sẽ được nạp vào thanh ghi thợ 8-bit (slave register). Thanh ghi này có các chức năng sau đây:

- Nếu chip 8259A là chủ, thì trong ICW, từng bit D7-D0 được thiết lập giá trị "1" nếu có kết nối với chip 8259A tớ. Chip chủ sẽ đưa ra byte đầu tiên-mã lệnh gọi (CALL CODE) và cho phép chip thợ tương ứng tạo ra các byte thứ 2 thứ 3 của lệnh CALL, thông qua việc gửi mã xác định chip thợ lên các đường tín hiệu CAS2-CAS0.

- Nếu chip 8259A là thợ thì các bit D2-D0 của ICW3 xác định số hiệu của chip thợ. Chip thợ so sánh giá trị các đường tín hiệu vào của

nó CAS2-CAS0 gửi từ chip chủ tới. Với mã xác nhận của nó (D2-D0 của ICW3), nếu trùng, thì chip thợ đó sẽ tạo ra byte thứ 2 và byte thứ ba của lệnh CALL.

(4) Lập trình các từ lệnh OCW

Sau khi lập trình các từ lệnh ICW vào trong 8259A, chip 8259A sẵn sàng cho tiếp nhận các yêu cầu ngắt ở các đầu vào IR7-IR0. Tuy nhiên, trong quá trình vận hành 8259A, một sự chọn lựa các thuật toán có thể lệnh cho 8259A vận hành trong các chế độ khác nhau thông qua OCW. Những chế độ này như sau:

- Các mặt nạ ngắt (Interrupt masks)

Mỗi một yêu cầu ngắt có thể được che mặt nạ một cách riêng biệt nhờ thanh ghi IMR được lập trình thông qua từ lệnh OCW1. IMR sẽ vận hành dựa trên cả thanh ghi yêu cầu ngắt IRR và thanh ghi phục vụ ngắt ISR. Nếu một yêu cầu ngắt đã được 8259A xác nhận (dựa vào xung INTA# xuất hiện), thì yêu cầu ngắt mặc dù đã bị che mặt nạ, sẽ cấm các yêu cầu ngắt có mức ưu tiên thấp hơn. Để cho phép các ngắt có mức ưu tiên thấp hơn thì phải thực hiện một trong các bước sau đây:

+ Ghi lệnh EOI (End of Interrupt) (từ OCW2) để xóa bit ISR, hoặc
+ Thiết lập chế độ mặt nạ đặc biệt nhờ OCW3 (Special masked mode).

- Chế độ kế tiếp đầy đủ (Fully nested mode):

8259A sẽ làm việc trong chế độ kế tiếp đầy đủ sau khi thực hiện chuỗi khởi tạo ban đầu nhờ các ICW, khi chưa ghi bất kỳ một từ lệnh OCW nào. Trong chế độ này, các yêu cầu ngắt được sắp xếp theo thứ tự mức ưu tiên từ 0 đến 7. Khi một yêu cầu ngắt được chấp nhận, mức ưu tiên cao nhất được nhận và địa chỉ của chương trình con phục ngắt đó đặt lên bus dữ liệu (D7-D0). Ngoài ra một bit của ISR (ISR7-ISR0) được thiết lập. Bit này duy trì trạng thái bật cho đến khi CPU đưa ra lệnh EOI (ở cuối chương trình phục vụ ngắt trước khi nhảy trở lại chương trình chính). Trong khi bit ISRX (X = 0,.., 7) được thiết lập, tất

cả các ngắt mới có mức ưu tiên thấp hơn bị cấm. Sau khi khởi tạo ban đầu bằng các ICW, thì IR0 có mức ưu tiên cao nhất, còn IR7 có mức ưu tiên thấp nhất. Các mức ưu tiên có thể thay đổi trong chế độ quay vòng mức ưu tiên (rotaing priority mode).

- Các chế độ quay vòng mức ưu tiên

Các chế độ quay vòng mức ưu tiên của 8259A dùng trong hệ thống ngắt gồm các thiết bị có mức ưu tiên ngắt bằng nhau, ví dụ như các thiết bị kết nối vào các kênh truyền thông. Có hai chế độ quay vòng mức ưu tiên ngắt: chế độ tự động và chế độ đặc biệt.

+ Chế độ tự động (Auto mode)

Trong chế độ này, thiết bị đang được phục vụ có mức ưu tiên cao nhất, các thiết bị có yêu cầu ngắt phải chờ đợi được phục vụ. Trường hợp xấu nhất (phải chờ lâu nhất) là có tới 7 thiết bị đang được phục vụ, và thiết bị còn lại có yêu cầu ngắt. Trong trường hợp này, mức ưu tiên và trạng thái "đang phục vụ" (nội dung thanh ghi ISR) có nội dung như sau:

a) Trước quay vòng:

Trạng thái của ISR

ISR7	ISR6	ISR5	ISR4	ISR3	ISR2	ISR1	ISR0
0	1	0	1	0	0	0	0

Mức ưu tiến thấp nhất — Mức ưu tiên cao nhất

Trạng thái ưu tiên

7	6	5	4	3	2	1	0

b) Sau quay vòng:

Trạng thái của ISR

ISR7	ISR6	ISR5	ISR4	ISR3	ISR2	ISR1	ISR0
0	1	0	0	0	0	0	0

Mức ưu tiến thấp nhất — Mức ưu tiên cao nhất

Trạng thái ưu tiên

4	3	2	1	0	7	6	5

Trong ví dụ này, bit ISR4 = 1 (trước khi quay vòng) tương ứng với yêu cầu ngắt IR4 có mức ưu tiên cao nhất (= 4). Sau khi quay vòng mức ưu tiên, mức ưu tiên thấp hơn sẽ được đưa vào phục vụ, đó là mức ưu tiên 5, còn mức ưu tiên 4 đã được phục vụ thì bit ISR4 được xóa đi sau khi quay vòng. Lệnh quay vòng được đưa ra trong từ lệnh OCW2, trong đó, R = 1, SEOI = 0, và EOI = 1.

+ Chế độ đặc biệt (Special mode)

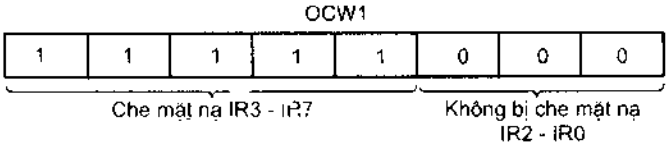
Trong chế độ này, người lập trình có thể thay đổi mức ưu tiên nhờ lập trình mức ưu tiên đáy để cố định mức ưu tiên cao nhất. Nghĩa là, nếu IR5 được lập trình như một thiết bị có mức ưu tiên đáy thì IR6 sẽ có mức ưu tiên cao nhất. Trong trường hợp này lệnh quay vòng đưa ra trong từ lệnh OCW2, trong đó, R = 1, SEOI = 1, và L2, L1, L0 là các bit mã BCD của mức ưu tiên đáy của thiết bị. Chế độ này có thể thực hiện độc lập với lệnh EOI.

- Kết thúc ngắt EOI và kết thúc ngắt đặc biệt SEOI

Mã lệnh EOI cần phải được CPU đưa ra (trong cuối chương trình phục vụ ngắt, trước khi trở về chương trình chính) cho 8259A để xóa bit ISR tương ứng với yêu cầu ngắt đã được phục vụ. Có hai khuôn dạng của lệnh EOI: đặc biệt và không đặc biệt. Khi 8259 được vận hành trong các chế độ kế tiếp đầy đủ, thì nó có thể xác định bit nào của thanh ghi ISR cần phải xóa khi có EOI. Khi có lệnh EOI không đặc biệt, thì 8259 sẽ tự động xóa bit ISR tương ứng với mức ưu tiên cao nhất vừa được phục vụ, bởi vì trong chế độ kế tiếp đầy đủ, mức ưu tiên cao nhất tiếp theo mức ưu tiên vừa được phục vụ sẽ được vào phục vụ. Tuy nhiên, khi một chế độ khác được thiết lập không phải là chế độ kế tiếp đầy đủ, như chế độ quay vòng mức ưu tiên thì 8259A không thể xác định được mức ưu tiên cuối cùng được chấp nhận. Trong trường hợp này, một lệnh EOI đặc biệt (SEOI) phải được đưa ra, trong đó một phần của lệnh có yêu cầu bit ISR của mức ưu tiên cần được xóa. Kết thúc ngắt được đưa ra khi EOI = 1 trong từ lệnh OCW2. Đối với SEOI, thì bit SEOI (hay SI) = 1, và EOI = 1, còn các bit L2, L1, L0 là các bit của mã BCD của giá trị mức ưu tiên cần được xóa.

- Chế độ mặt nạ đặc biệt SMM (Special mask mode)

Chế độ này được sử dụng khi một số bit được thiết lập (che mặt nạ) nhờ thanh ghi mặt nạ ngắt IMR thông qua từ lệnh OCW1. Mỗi một bit của từ OCW1 ứng với một yêu cầu ngắt IRX (X = 0,...,7). Những bit bằng 1 thì các yêu cầu ngắt tương ứng được che mặt nạ (masked), những bit bằng 0 thì những yêu cầu ngắt tương ứng không bị che mặt nạ (unmasked):



Chế độ mặt nạ đặc biệt được thiết lập trong OCW3 hờ bit SMM = 1, ESMM = 1. Chế độ này được xóa khi SMM = 0, ESMM = 1. Trong chế độ mặt nạ đặc biệt, các đường yêu cầu ngắt có mức ưu tiên thấp có thể được phép trong khi SMM được xóa đi. Các mức tiên cao hơn không bị ảnh hưởng.

- Chế độ chọn lựa (Polled mode)

Trong chế độ này, bộ vi xử lý phải cấm đầu vào ngắt của nó (INTR). Như vậy, tín hiệu INTA# cũng không cần tới nữa. Sự phục vụ các thiết bị ngoại vi được lập trình nhờ lệnh chọn trong từ lệnh OCW3. Lệnh chọn được đưa ra nhờ thiết lập bit P = 1 trong từ lệnh OCW3 trong quãng thời gian xung WR# = 0. Chip 8259A tiếp nhận xung RD# như là sự chấp nhận ngắt của CPU, nó thiết lập bit ISR tương ứng nếu có một yêu cầu ngắt., và nó đọc mức ưu tiên ngắt. Xung RD# của CPU sẽ đọc dữ liệu từ 8259A thông qua bus dữ liệu D7-D0 như sau:

D7	D6	D5	D4	D3	D2	D1	D0
1	-	-	-	-	W2	W1	W0

Trong đó, W2-W0: mã BCD của mức ưu tiên cao nhất yêu cầu được phục vụ. Giá trị D7 = 1, có nghĩa là có một yêu cầu ngắt. Chế độ

này thường dùng trong hệ thống có ít thiết bị ngoại vi, có cùng một mức ngắt.

(5) Đọc trạng thái của 8259A

Trạng thái của một số thanh ghi bên trong 8259A có thể được đọc về CPU. Nội dung những thanh ghi sau đây có thể được đọc lên bus dữ liệu D7-D0 nhờ sử dụng các từ lệnh OCW phù hợp và xung đọc RD# của CPU:

- Thanh ghi IRR: Thanh ghi 8-bit này chứa các mức ưu tiên của các yêu cầu ngắt. Mức ưu tiên cao nhất được xóa khỏi IRR khi một yêu cầu ngắt được chấp nhận phục vụ.

- Thanh ghi ISR: Thanh ghi 8-bit này chứa các mức ưu tiên của các yêu cầu ngắt đang được phục vụ. Nội dung của ISR được thay đổi khi có EOI.

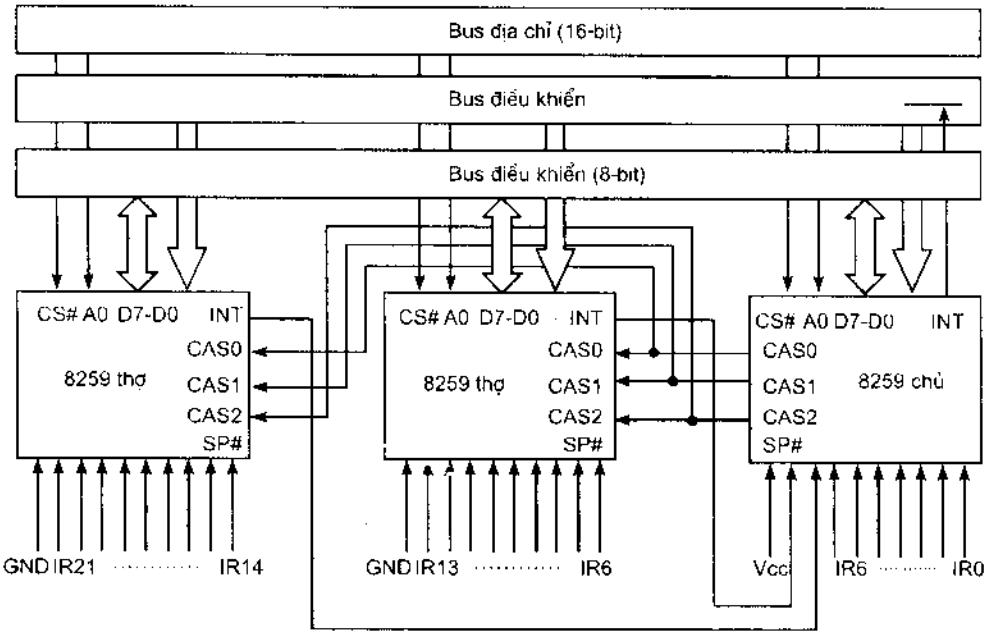
- Thanh ghi IMR: Thanh ghi 8-bit chứa các bit tương ứng với các yêu cầu ngắt. Giá trị từng bit = 1 thì yêu cầu ngắt tương ứng được che mặt nạ (masked), giá trị từng bit = 0 thì yêu cầu ngắt tương ứng không bị che mặt nạ (unmasked).

Thanh ghi IRR có thể được đọc khi trong từ OCW3 giá trị các bit của lệnh đọc: RIS = 0 hoặc 1, RIR = 1 và với xung RD# tiếp theo. Không cần thiết phải ghi một từ OCW3 trước mỗi lần đọc trạng thái, bởi vì ta cần đọc trạng thái của lần ghi OCW3 trước. Nói một cách khác, cần phải thực hiện ghi từ lệnh OCW3 trước khi muốn đọc trạng thái của thanh ghi IRR.

Để đọc thanh ghi IMR, không cần phải có sự ghi (WR#) trước, mà cần có xung RD#, A0 = 1, và trong từ lệnh OCW3 bit P =1 (lệnh lựa chọn) và RIS = 1. Nội dung IMR có thể được tải vào thông qua bus dữ liệu khi A0= 1 và WR# = 0.

(6) Nối tầng (cascading)

Các chip 8259A có thể dễ dàng kết nối với nhau trong một hệ thống xử lý ngắt với một chip 8259A là một chip chủ (hình 6.18).



Hình 6.18: Hệ thống xử lý ngắt gồm các chip 8259 nối tầng

Chip 8259A chủ thông qua các đường CAS2-CAS0 đưa ra các địa chỉ của các chip 8259 tớ. Các đầu ra ngắt (INT) của các chip 8259A thợ đưa tới đầu vào yêu cầu ngắt của 8259A chủ. Chỉ có INT của chip chủ nối với CPU. Khi một đầu ra INT của chip 8259A thợ ở mức tích cực "1" và nó được chấp nhận, chip 8259A chủ sẽ đưa về CPU mã lệnh CALL (byte thứ nhất) trong thời gian INTA# thứ nhất, và nó tạo cho chip 8259 thợ (có yêu cầu ngắt) đưa về CPU hai byte còn lại của lệnh CALL (hai byte địa chỉ nhảy tới chương trình con phục vụ ngắt) với hai xung INTA# tiếp theo. Các byte của lệnh CALL được đọc về CPU thông qua bus dữ liệu. Tín hiệu chọn chip CS# của các chip 8259A là các đầu ra của các logic giải mã địa chỉ vào/ra của hệ thống vi xử lý.

Tín hiệu SP#/EN# = 1 (Vcc) ghim cho chip 8259A chủ, và SP#/EN# = 0 (GND) gim cho các chip 8259A thợ.

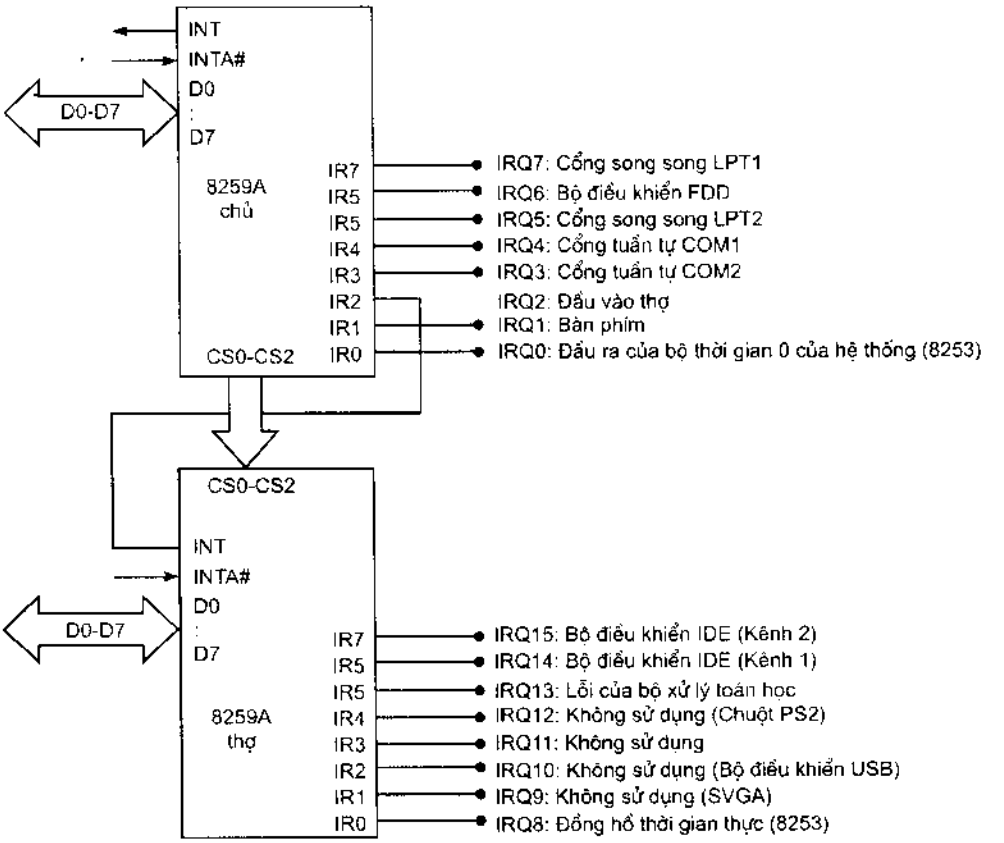
d. Kết nối 8259A trong hệ thống IBM PC/XT và AT

Để minh họa cho ứng dụng của 8259A trong hệ thống vi xử lý, ta lấy ví dụ hệ thống IBM PC/XT. Trên bo mạch chủ (xem Phụ lục), chân IR0 (mức ưu tiên cao nhất) của chip 8259A nối từ đầu ra OUT0 của chip lập trình đếm thời gian 8253. Chân IR1 nối từ logic tạo yêu cầu ngắt của bảng phím (Output Buffer Full). Các đầu vào yêu cầu ngắt IR7-IR2 dành cho các yêu cầu phục vụ của thiết bị ngoại vi khác như máy in LPT1: IRQ7, thiết bị nhớ trên đĩa mềm: IRQ6, thiết bị trên đĩa cứng: IRQ5, các cổng vào/ra truyền thông: COM1: IRQ3, COM2: IRQ3, IRQ2: dự phòng. Chân tín hiệu ngắt INT của 8259A nối trực tiếp đến chân vào yêu cầu ngắt INTR của vi xử lý 8086/8088. Như vậy, sự phân mức ưu tiên ngắt đã được thiết kế cứng cho các thiết bị ngoại vi và địa chỉ các chương trình phục vụ ngắt cũng cố định ở trong ROM BIOS. Trong IBM PC/XT, chip 8259A có địa chỉ vào/ra trong khoảng 020h - 021Fh. Bằng dải địa chỉ 020h-021Fh thực hiện giải mã để tạo tín hiệu chọn chip CS# cho 8259A, đó là tín hiệu INTRCS#. Tín hiệu INTRCS# được tạo ra bằng mạch giải mã 74LS138 (U31) với các đầu vào giải mã là các đường dây địa chỉ lấy từ bus địa chỉ của hệ thống. Tổ hợp của chúng phải là:

INTRCS# = 0 khi (AEN# = 1)(A9 = 0)(A8 = 0)(A7= 0)(A6 = 0)(A5 = 1).

Đường địa chỉ A0 dùng để phân biệt các thanh ghi điều khiển khác nhau trong chip 8259A.

Trong các hệ thống IBM PC/AT 286, số lượng các yêu cầu ngắt tăng gấp đôi so với IBM PC/XT (IR0 - IR15), vì vậy, trên bảng mạch hệ thống của IBM PC/AT phải dùng tới hai chip 8259 kết nối theo tầng, trong đó một chip 8259 chủ bảo đảm các yêu cầu ngắt IR0 - IR7 (giống như IBM PC/XT) và 8259 thợ bảo đảm các yêu cầu ngắt IR8 - IR15 (các yêu cầu ngắt của IBM PC/AT) (hình 6.19).



Hình 6.19: Kết nối theo tầng 2 vi mạch 8259A trong IBM PC/AT

6.3.6 Ngắt không che được

Ngắt không che được (non-maskable interrupt) có đặc điểm như sau:

• Có mức ưu tiên cao nhất trong các ngắt cứng;

• Tác động bằng phần cứng với mức xung logic cao vào chân NMI của vi xử lý (không dùng INTR). Không thể bị che mặt nạ bằng lệnh CLI của vi xử lý;

• Không được điều khiển bởi 8259A (trong IBM PC/XT, AT);

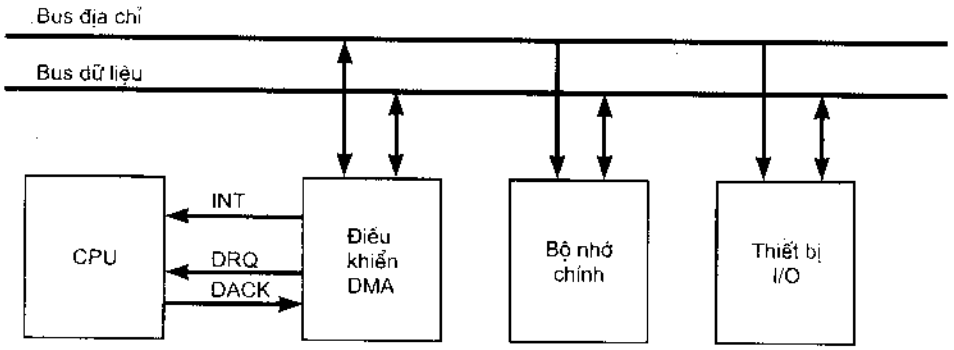
• Được tạo ra khi xuất hiện các sự kiện: lỗi chẵn lẻ bộ nhớ RAM (DRAM), lỗi chẵn lẻ ở kênh vào/ra, lỗi bộ đồng xử lý (trong IBM PC/XT, IRQ13 trong AT);

• Có thể bị cấm khi thiết lập bit 7 của thanh ghi địa chỉ 070h trong các hệ thống từ AT 286 trở lên.

6.4 VÀO/RA DỮ LIỆU BẰNG DMA

6.4.1 Quá trình điều khiển DMA

DMA sử dụng phần cứng để điều khiển vào/ra dữ liệu giữa bộ nhớ chính bên trong (RAM) và thiết bị ngoại vi bên ngoài mà không có sự can thiệp của CPU. Khi đó, CPU tách khỏi bus hệ thống, còn khối logic điều khiển DMA (DMAC) làm chủ bus hệ thống điều khiển vào/ra dữ liệu giữa thiết ngoại vi và bộ nhớ (hình 6.20).



Hình 6.20: Điều khiển DMA

Để điều khiển DMA, trước hết, đối với CPU phải lập chương trình thiết lập các thông số sau:

• Địa chỉ đầu của vùng nhớ mà khối dữ liệu được cất giữ;

• Địa chỉ của các cổng vào/ra mà qua đó dữ liệu vào/ra đi qua (từ/đến thiết bị ngoại vi - thế giới thực);

• Số đếm các byte hoặc từ của các khối dữ liệu cần phải chuyển giữa thiết bị ngoại vi và bộ nhớ;

• Hướng chuyển dữ liệu (từ bộ nhớ ra ngoại vi hay ngược lại).

Sau đó chúng ta khởi động chương trình để gửi các thông số trên ra DMAC.

• Nhận được các thông số này, DMAC gửi đến CPU yêu cầu DMA (DRQ);

• Nhận được từ CPU tín hiệu chấp nhận yêu cầu DMA (DACK), DMAC tiếp nhận quyền điều khiển (làm chủ) bus hệ thống (CPU tách khỏi bus hệ thống;

• DMAC đưa ra bus hệ thống địa chỉ đầu của vùng nhớ mà khối dữ liệu được cất giữ;

• DMAC đưa ra bus hệ thống địa chỉ của các cổng vào/ra mà qua đó dữ liệu vào/ra đi qua (từ/đến thiết bị ngoại vi - thế giới thực);

• Đồng thời đọc từ thiết bị ngoại vi và ghi vào bộ nhớ, hoặc ngược lại đọc từ bộ nhớ và ghi ra thiết bị ngoại vi để truyền 1 byte (hoặc 1 từ) trong một chu kỳ bus (bắt đầu truyền). Mỗi chu kỳ bus truyền 1 byte dữ liệu;

• Giảm số đếm byte hoặc từ truyền;

• Kết thúc truyền (khi số đếm bằng 0) DMAC gửi tín hiệu ngắt (INT) về CPU để báo rằng quá trình truyền dữ liệu bằng DMA đã kết thúc.

DMAC phải có ít nhất 4 thanh ghi bên trong để có thể nạp từ CPU:

• Thanh ghi địa chỉ bộ nhớ (memory address register) để ghi địa chỉ vùng nhớ mà khối dữ liệu được cất giữ;

• Thanh ghi địa chỉ vào/ra (I/O address register) để chứa địa chỉ của các cổng vào/ra mà qua đó dữ liệu vào/ra đi qua (từ/đến thiết bị ngoại vi - thế giới thực);

• Thanh ghi đếm (count register) để chứa số đếm các byte hoặc từ của các khối dữ liệu cần phải chuyển giữa thiết bị ngoại vi và bộ nhớ;

• Thanh ghi hướng (direction register) để ghi giá trị bit hướng chuyển dữ liệu (từ bộ nhớ ra ngoại vi hay ngược lại).

DMAC có thể:

• Truyền khối lớn dữ liệu trong một thao tác cực nhanh, sử dụng chế độ truyền cực nhanh (burst mode);

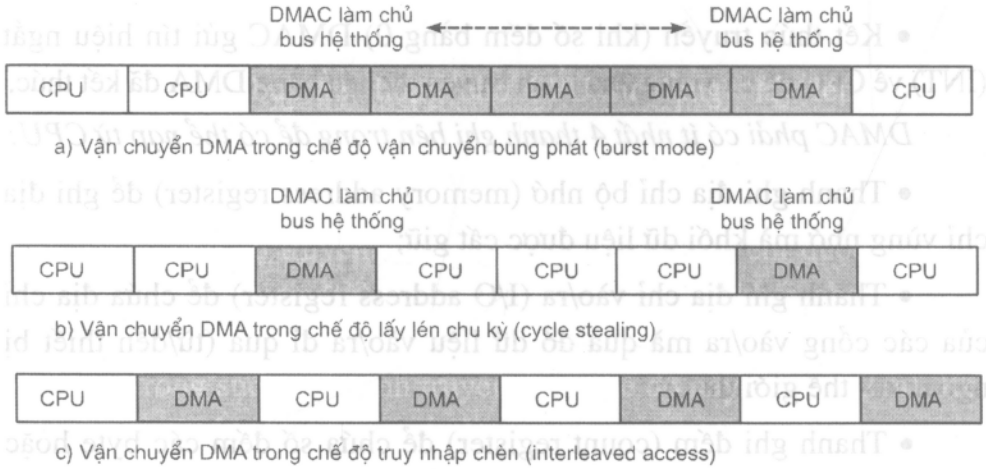
• Truyền 1 hoặc 2 byte (hay từ) dữ liệu trước khi trả quyền điều khiển (làm chủ) bus hệ thống cho CPU sử dụng "lấy lén chu kỳ" (cycle stealing);

• Truyền dữ liệu sử dụng "truy nhập chèn" (interleaved access).

Trong chế độ truyền cực nhanh, thanh ghi địa chỉ được tự động tăng để trỏ đến byte (từ) dữ liệu tiếp theo trong khối cho đến khi truyền xong toàn bộ khối dữ liệu (bộ đếm bằng 0). DMAC là chủ bus hệ thống trong suốt quá trình truyền theo chế độ này (hình 6.21a). Kết thúc truyền DMAC trả lại quyền làm chủ bus cho CPU.

Trong chế độ truyền lấy lén chu kỳ, DMAC chỉ làm chủ bus hệ thống ở một vài chu kỳ bus để truyền byte dữ liệu ở những chu kỳ đó mà thôi (hình 6.21b).

Trong chế độ truyền chèn, DMAC và CPU luân phiên làm chủ bus hệ thống để truyền các byte dữ liệu (hình 6.21c).



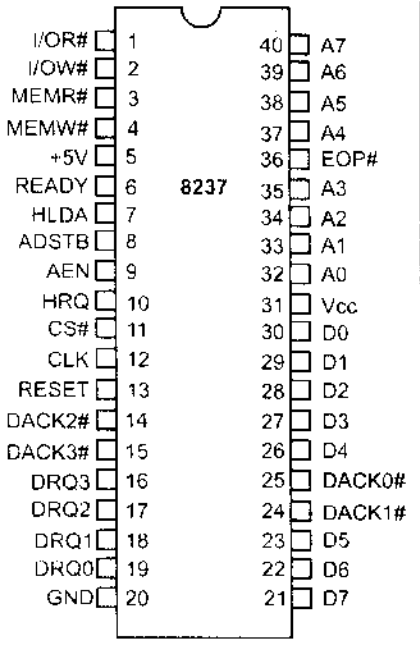
Hình 6.21: Các phương pháp truyền dữ liệu trong DMA

6.4.2 Vi mạch DMAC 8237

DMAC là logic phần cứng phức tạp, do vậy để đơn giản trong phối ghép, có các vi mạch lập trình điều khiển DMA dùng tương thích với các vi xử lý được chế tạo sẵn.

a) Đóng vỏ, các tín hiệu và sơ khối chức năng của DMAC 8237

DMAC 8237 (programmable DMA Controller) là vi mạch lập trình điều khiển truy nhập trực tiếp bộ nhớ sử dụng trong các hệ thống vi xử lý 8-bit và 16-bit của Intel. Các thiết bị nhớ ngoài, ví dụ, đĩa mềm thông qua điều khiển của 8237 có thể thực hiện trao đổi các khối dữ liệu với bộ nhớ RAM của hệ thống máy tính với tốc độ nhanh nhất mà không có sự can thiệp của vi xử lý. Hình 6.22 là đóng vỏ, các chân tín hiệu và hình 6.23 là sơ đồ khối của 8237.



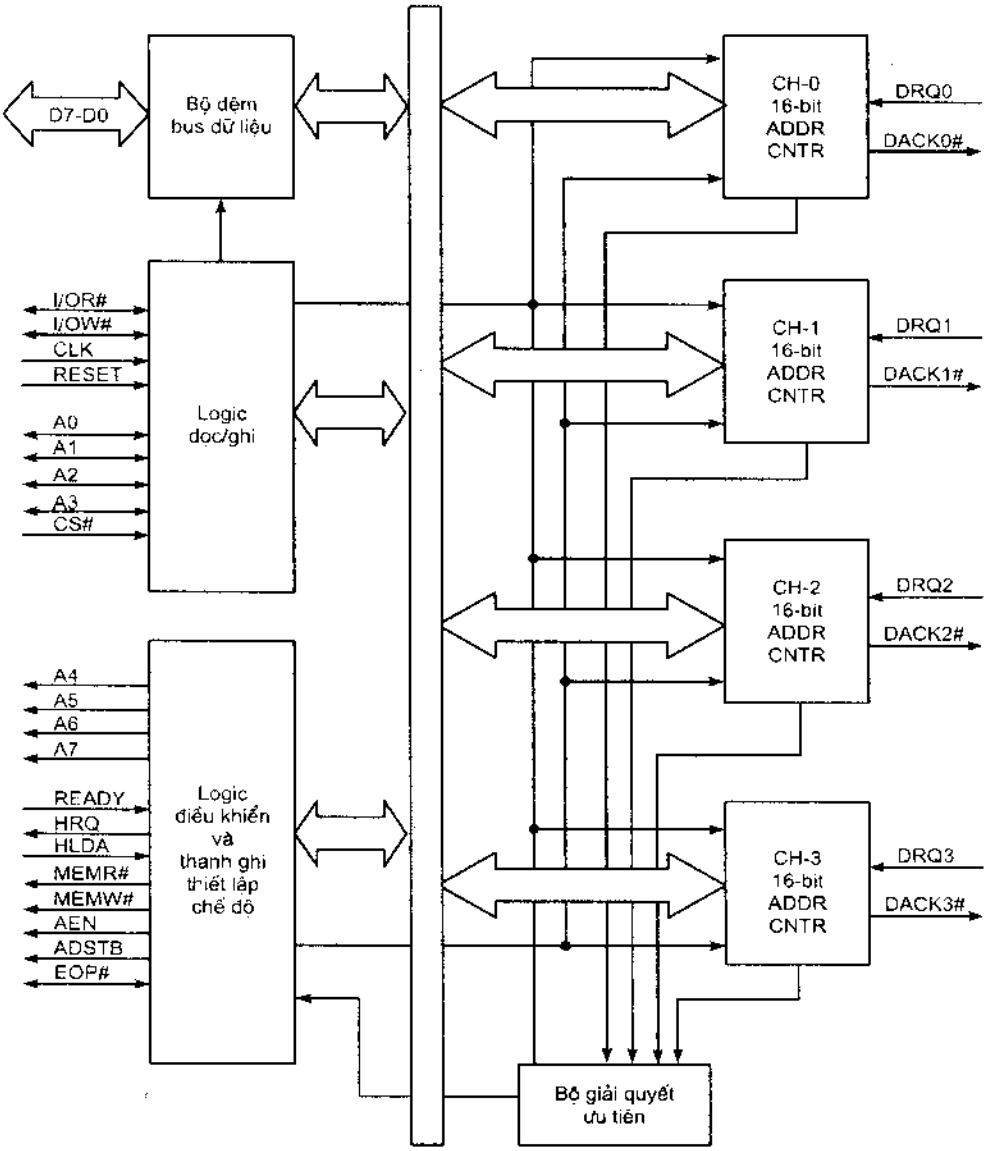
Tên tín hiệu	Chức năng
D7 - D0	Bus dữ liệu, hai chiều, 3 trạng thái
RESET	Đầu vào thiết lập lại
CS#	Đầu vào lựa chọn chip
MEMR#	Đọc bộ nhớ
MEMW#	Ghi bộ nhớ
A7-A4	Đầu ra bus địa chỉ, 3 trạng thái
A3-A0	Bus địa chỉ, 2 chiều, 3 trạng thái
I/OR#	Đọc I/O 2 chiều, 3 trạng thái
I/OW#	Ghi I/O
READY	Đầu vào sẵn sàng
CLK	Đầu vào CLK
HRQ	Đầu ra yêu cầu chờ
HLDA	Đầu vao chấp nhận chờ
AEN	Đầu ra cho phép địa chỉ
ADSTB	Đầu ra định nhịp địa chỉ
EOP#	Kết thúc quá trình, 2 chiều
MARK	Đầu ra mặt nạ modulo 128
DRQ3 - DRQ0	Đầu vào yêu cầu DMA
DACK3# - DACK0#	Đầu ra chấp nhận DMA
Vcc	+5V
GND	Nối đất

Hình 6.22: Đóng và và các chân tín hiệu của 8237

8237 là bộ điều khiển với 4 kênh truy nhập trực tiếp bộ nhớ. Nó là sản phẩm của Intel và chuyên dùng để trao đổi dữ liệu tốc độ cao giữa các thiết bị nhớ ngoài có khả năng truyền dữ liệu theo khối (đĩa từ, băng từ) trong các hệ vi xử lý 8-bit của Intel như 8080, 8085, 8086/8088.

Sự trao đổi dữ liệu giữa bộ nhớ ngoài và bộ nhớ trong (RAM) sẽ là cực kỳ chậm nếu phải thông qua trung gian là bộ vi xử lý, hay chính xác hơn là thông qua các thanh ghi của vi xử lý. Điều khiển truy nhập trực tiếp bộ nhớ RAM có nhiệm vụ khắc phục nhược điểm này bằng

cách tạo khả năng cho thiết bị nhớ ngoài kết nối trực tiếp với RAM thông qua hệ thống bus với điều kiện khóa sự can thiệp của vi xử lý đối với hệ thống bus. Thiết bị nhớ ngoài phải có bộ điều khiển vào/ra dữ liệu với RAM tương đối thông minh để có thể thay vi xử lý thực hiện các cuộc truyền các khối dữ liệu giữa chúng với RAM.



(1): HRQ được thiết lập nếu DRQn tích cực, = "1"; (2): HRQ bị xóa nếu DRQn không tích cực,= "0"

Hình 6.23: Sơ đồ khối chức năng của 8237

b) Các khối chức năng của DMAC 8237

(1) Bộ đệm dữ liệu (Bus dữ liệu buffer), D7-D0

Đây là bộ đệm dữ liệu 8-bit, hai chiều, 3 trạng thái, giao tiếp với bus hệ thống của hệ vi xử lý (D7-D0). Khi lập trình 8237, nội dung dữ liệu 8-bit từ CPU thông qua bus dữ liệu và bộ đệm này có thể ghi vào thanh ghi địa chỉ DMA (DMA address register), thanh ghi đếm kết thúc (Terminal count register), hoặc thanh ghi thiết lập chế độ (Mode set register). Ngược lại, khi CPU đọc dữ liệu từ: thanh ghi địa chỉ DMA, thanh ghi đếm kết thúc, hoặc thanh ghi trạng thái (Status register), thì nội dung các thanh ghi này cũng thông qua bộ đệm dữ liệu và bus dữ liệu về CPU. Trong các chu kỳ DMA (khi 8237 là chủ, CPU treo bus của mình), 8237 sẽ đưa ra 8-bit cao của địa chỉ ngăn nhớ (từ một trong các thanh ghi địa chỉ DMA) tới mạch chốt địa chỉ, ví dụ, 8212, thông qua bus dữ liệu. Những bit cao của địa chỉ sẽ được chuyển ở đầu chu kỳ DMA, sau đó bus sẽ được giải phóng để thực hiện truyền dữ liệu nhớ trong chu kỳ DMA.

(2) Các kênh DMA

Mỗi một kênh DMA (CH0 - CH3) gồm có hai thanh ghi 16-bit: thanh ghi địa chỉ DMA (DMA address register), và thanh ghi đếm kết thúc (Terminal count register). Cả hai thanh ghi này phải được khởi tạo trước khi một kênh được phép làm việc (enable). Thanh ghi địa chỉ DMA được nạp địa chỉ của vùng nhớ đầu tiên phải truy nhập. Giá trị nạp vào 14 bit thấp của thanh ghi đếm kết thúc là giá trị xác định số lượng các chu kỳ DMA trừ 1 trước khi đầu ra đếm kết thúc TC chuyển lên mức tích cực. Hai bit cao còn lại của thanh ghi đếm kết thúc xác định kiểu của thao tác DMA cho kênh.

(3) Khối đọc/ghi (Read/write logic)

Khi lập trình CPU đọc hoặc ghi nội dung các thanh ghi của 8237 thì là đọc/ghi, tức là khi đó CPU là chủ và 8237 là tớ, thì logic của 8237 tiếp nhận các tín hiệu I/OR# và I/OW# từ CPU (từ mạch điều khiển bus 8288 trong IBM PC/XT, xem hình 7.22a), nếu có chọn chip

8237 (CS# = "0"). Nghĩa là các tín hiệu I/OR# và I/OW# là các tín hiệu vào của 8237. Nhưng trong quá trình DMA (hay trong các chu kỳ DMA) thì 8237 là chủ và bộ nhớ là tớ, thì logic đọc/ghi tạo ra các tín hiệu I/OR# và I/OW# để điều khiển quá trình chuyển dữ liệu giữa bộ nhớ trong (RAM) và thiết bị nhớ ngoài. Nghĩa là trong trường hợp này, các tín hiệu I/OR# và I/OW# là các tín hiệu ra của 8237.

(4) A3-A0

Là các đường địa chỉ 2 chiều, 3 trạng thái. Khi 8237 là thợ và CPU là chủ thì các đường địa chỉ này là các đường vào của 8237. Chúng được dùng để đánh địa chỉ các thanh ghi bên trong 8237 khi lập trình CPU ghi hoặc đọc nội dung các thanh ghi của 8237. Khi 8237 là chủ trong các chu kỳ DMA thì A3-A0 là 4 bit thấp nhất của địa chỉ 16-bit của bộ nhớ được 8237 tạo ra. Tức là các đường địa chỉ này là các tín hiệu ra của 8237.

(5) A7-A4

Là các đường ra địa chỉ, 3 trạng thái. Khi 8237 là chủ thì A3-A0 là 4 bit thấp nhất và A7-A4 là 4 bit tiếp theo của byte thấp của địa chỉ 16-bit của bộ nhớ được 8237 tạo ra trong các chu kỳ DMA. Byte cao của địa chỉ 16-bit của bộ nhớ được tạo ra nhờ các đường dữ liệu D7-D0 của 8237 và xung ADSTB.

(6) Cho phép địa chỉ AEN

8237 tạo ra tín hiệu này để cấm (treo) bus dữ liệu và bus điều khiển của hệ thống bus nhằm tách vi xử lý và các thiết bị khác khỏi kênh vào/ra nhằm phục vụ cho truyền dữ liệu DMA. Khi AEN = "1" (tích cực), 8237 giành lấy quyền điều khiển bus địa chỉ, bus dữ liệu của bus hệ thống để ghi/đọc RAM trực tiếp và thiết bị nhớ ngoài.

(7) Chọn chip CS#

Khi 8237 là thợ và CPU là chủ thì tín hiệu CS# = "0" khi lập trình CPU vào/ra dữ liệu với các thanh ghi bên trong 8237.

(8) READY

Là tín hiệu vào của 8237 và xuất hiện không đồng bộ. Nó được sử dụng để kéo dài các chu kỳ đọc và ghi dữ liệu bộ nhớ RAM cho 8237 với các trạng thái chờ nếu bộ nhớ RAM yêu cầu các chu kỳ dài.

(9) Yêu cầu DMA DRQn

Là 4 tín hiệu vào yêu cầu DMA từ thiết bị nhớ ngoài. Khi có một trong các tín hiệu này chuyển lên mức tích cực "1" thì 8237 sẽ phải chuyển yêu cầu này thông qua sinh ra tín hiệu HRQ chuyển về CPU.

(10) Yêu cầu treo bus HRQ

Là tín hiệu được 8237 tạo ra để yêu cầu có được quyền điều khiển bus hệ thống của hệ vi xử lý. Bình thường, HRQ sẽ được đưa tới đầu vào HOLD của chip vi xử lý và chờ đợi sự chấp nhận của vi xử lý yêu cầu này.

(11) Chấp nhận treo bus HLDA

Khi nhận được tín hiệu HRQ ở mức tích cực "1", vi xử lý phải tạo ra HLDA để thực trao quyền điều khiển bus hệ thống cho 8237, nó treo các của ra bus hệ thống của nó (địa chỉ, dữ liệu) lên trạng thái trở kháng cao, và chỉ thực hiện xử lý nội bộ bên trong chip.

(12) Chấp nhận DMA DACKn

Khi nhận được HLDA từ vi xử lý, 8237 chuyển tín hiệu chấp nhận DMA DACKn# xuống mức tích cực "0" đến cho thiết bị nhớ ngoài đã gửi yêu cầu DMA (DRQn) trước đó (có mức yêu tiên phục vụ cao nhất).

(13) Đọc bộ nhớ MEMR#

Là tín hiệu ra, 3 trạng thái. Khi 8237 là chủ trong các chu kỳ DMA thí nó tạo ra MEMR# = "0" đưa tới bộ nhớ RAM để thực hiện chu kỳ đọc DMA (đọc dữ liệu từ RAM). Khi 8237 là thợ thì MEMR# ở trạng thái trở kháng cao (không tích cực).

(14) Ghi vào bộ nhớ MEMW#

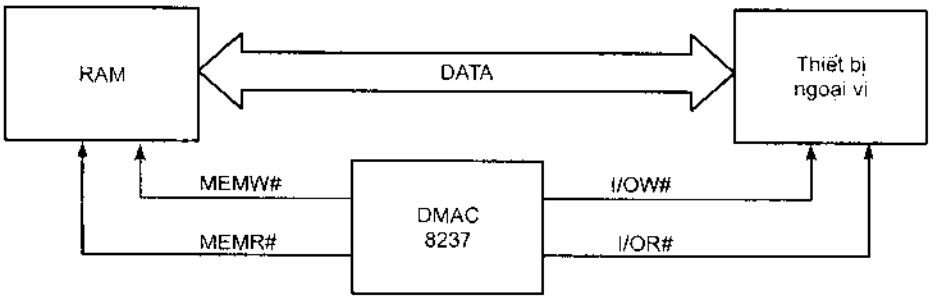
Là tín hiệu ra, 3 trạng thái. Khi 8237 là chủ trong các chu kỳ DMA thì nó tạo ra MEMW# = "0" đưa tới bộ nhớ RAM để thực hiện chu kỳ ghi DMA (ghi dữ liệu vào RAM). Khi 8237 là thợ thì MEMW# ở trạng thái trở kháng cao (không tích cực).

(15) Đọc vào/ra, I/OR#

Là tín hiệu 2 chiều, 3 trạng thái. Khi là tín hiệu vào và ở mức tích cực thấp thì nó được sản sinh từ CPU trong chu kỳ "nghỉ" của 8237 để đọc trạng thái của 8237. Khi nó là tín hiệu ra và ở mức tích cực thấp thì đó là 8237 thực hiện đọc dữ liệu ra thiết bị ngoại vi trong chu kỳ DMA.

(16) Ghi vào/ra, I/OW#

Là tín hiệu 2 chiều, 3 trạng thái. Khi là tín hiệu vào và ở mức tích cực thấp thì nó được sản sinh từ CPU trong chu kỳ "nghỉ" của 8237 để ghi lệnh điều khiển, thiết lập chế độ vào các thanh ghi của 8237. Khi nó là tín hiệu ra và ở mức tích cực thấp thì đó là 8237 thực hiện ghi dữ liệu từ thiết bị ngoại vi trong chu kỳ DMA. Sơ đồ truyền dữ liệu trực tiếp giữa thiết bị ngoại vi và bộ nhớ RAM cho trong hình 6.24.



Hình 6.24: Trao đổi dữ liệu giữa thiết bị ngoại vi và RAM

(17) Kết thúc quá trình, EOP# (End Of Process)

Là tín hiệu 2 chiều, tích cực thấp. Khi kết thúc quá trình DMA thì nó là tín hiệu ra và tích cực thấp đưa về CPU gây ngắt để báo sự kết

thúc DMA. Khi là tín hiệu vào, thì nó là tín hiệu từ ngoại vi dùng để báo sự kết thúc truyền một mảng dữ liệu và có một yêu cầu mới DRQ.

(18) Xóa, RESET

Là tín hiệu xóa toàn bộ của hệ thống vi xử lý. Nó xóa các thanh ghi, các Fflips-flop bên trong 8237 và các tín hiệu yêu cầu DMA sau khi kết thúc xung xóa 8237 được đưa đến trạng thái "nghỉ" sẵn sàng nhận các lệnh và dữ liệu điều khiển từ CPU. Ta có thể lập trình xóa DMAC bằng lệnh đưa ra từ CPU.

(19) Nhịp địa chỉ ADSTB

Xung ADSTB được 8237 tạo ra để định byte cao của địa chỉ vùng nhớ RAM vào bộ đệm địa chỉ từ bus dữ liệu. (D7-D0).

c) Các thanh ghi bên trong DMAC 8237

Tên gọi, số lượng, chức năng của các thanh ghi bên trong 8237 được liệt kê trong bảng bảng 6.2.

Bảng 6.2: Các thanh ghi bên trong 8237

Tên gọi	Số lượng x độ dài	Chức năng
Thanh ghi địa chỉ hiện hành CAR (Current Address Reg.)	4 x 16-bit	Chứa địa chỉ của vùng nhớ chứa dữ liệu truyền. Mỗi một kênh DMA có 1 CAR
Thanh ghi số từ hiện hành CWR (Current Word Reg.)	4 x 16-bit	Chứa số đếm lượng từ (byte) hiện đã được truyền. Mỗi một kênh DMA có 1 CWR
Thanh ghi địa chỉ cơ sở BAR (Base Address Reg.)	4 x 16-bit	Chứa địa chỉ đầu của vùng nhớ dữ liệu truyền Mỗi một kênh DMA có 1 BAR.
Thanh ghi cơ sở đếm từ BWR (Base Word Reg.)	4 x 16-bit	Chứa giá trị ban đầu của số đếm lượng từ truyền. Mỗi một kênh DMA có 1 BWR
Thanh ghi lệnh CR (Command Register)	1 x 8-bit	Chứa giá trị để điều khiển DMA, cho cả 4 kênh
Thanh ghi thiết lập chế độ MSR (Mode Set Register)	4 x 6-bit	Chứa giá trị chế độ truyền dữ liệu và đếm từ
Thanh ghi yêu cầu DMA DRR (DMA Request Reg.)	1 x 4-bit	Chứa số hiệu kênh DMA có yêu cầu và bit xác nhận yêu cầu

Tên gọi	Số lượng x độ dài	Chức năng
Thanh ghi mặt nạ MR (Mask Register)	1 x 4-bit	Chứa các bit mặt nạ cho từng kênh DMA dùng để vô hiệu hóa các tín hiệu DRQn
Thanh ghi tạm thời TR (Temp Register)	1 x 8-bit	Dùng chứa dữ liệu trung gian khi thực hiện chuyển dữ liệu từ vùng nhớ này đến vùng nhớ khác.
Thanh ghi trạng thái SR (Status Register)	1 x 8-bit	Trạng thái làm việc của DMAC trong SR. CPU có thể đọc nội dung SR

Các thanh ghi lệnh CR, thanh ghi thiết lập chế độ MRS, thanh ghi yêu cầu DMA DRR, và thanh ghi mặt nạ MR có nội dung với chức năng từng bit như sau:

(1) Thanh ghi lệnh (CR)

D0: Cho phép chuyển dữ liệu giữa các vùng nhớ. D0 = 0: cấm, D0 = 1: cho phép

D1: Cho phép địa chỉ kênh 0. D1 = 0: cấm, D1 = X: nếu D0 = 0, D1 = 1: cho phép.

D2: Cho phép chọn DMAC. D2 = 0: cho phép, D2 = 1: cấm.

D3: Đọc/ghi bình thường. D3 = 0: đọc/ghi bình thường, D3 = 1: đọc/ghi nhanh. D3 = X: nếu D0 = 1.

D4: Chế độ ưu tiên. D4 = 0: Ưu tiên cố định, D4 = 1: Ưu tiên vòng.

D5: Kiểu xung WR. D5 = 0: xung WR trễ, D5 = 1: xung WR kéo dài, D5 = X: nếu D0 = 0.

D6: DRQ. D6 = 0: DRQ tích cực cao, D6 = 1: DRQ tích cực thấp.

D7: DACK. D7 = 0: DACK tích cực thấp, D7 = 1: DACK tích cực cao.

(2) Thanh ghi lập chế độ (MSR)

D1, D0: Chọn kênh DMA. 00: kênh 0, 01: kênh 1, 10: kênh 2, 11: kênh 3.

D3, D2: Chế độ thao tác: 00: kiểm tra, 01: ghi, 10: đọc, 11: không hợp lệ, xx: nếu D7, D6 = 11.

D4: Cho phép tự khởi động. D4 = 0: cấm, D4 = 1: cho phép.

D5: Chiều đếm địa chỉ. D5 = 0: đếm tăng, D5 = 1: đếm giảm.

D7, D6: Chế độ chuyển dữ liệu: 00: chuyển theo nhu cầu, 01: chuyển từng mảng, 10: chuyển mảng, 11: chuyển dữ liệu trong ghép tầng các DMAC.

(3) Thanh ghi yêu cầu DMA (DRR)

D1, D0: Chọn kênh DMA. 00: kênh 0, 01: kênh 1, 10: kênh 2, 11: kênh 3.

D2: Lập/xóa bit yêu cầu DMA. D2 = 0: xóa bit yêu cầu, D2 = 1: lập bit yêu cầu

D7-D3: Không dùng.

(4) Thanh ghi mặt nạ (MR)

D0: Lập/xóa mặt nạ kênh DMA0. D0 = 0: xóa mặt nạ, D0 = 1: lập mặt nạ.

D1: Lập/xóa mặt nạ kênh DMA1. D1 = 0: xóa mặt nạ, D1 = 1: lập mặt nạ.

D2: Lập/xóa mặt nạ kênh DMA2. D2 = 0: xóa mặt nạ, D2 = 1: lập mặt nạ.

D3: Lập/xóa mặt nạ kênh DMA3. D3 = 0: xóa mặt nạ, D3 = 1: lập mặt nạ.

D7-D4: Không dùng.

d. Các lệnh của DMAC 8237

DMAC 8237 có tập lệnh riêng mà CPU đưa ra qua các đường tín hiệu A3, A2, A1, A0, và I/OR, I/OW (bảng 6.3).

Bảng 6.3: Tập lệnh riêng của DMAC 8237

Tín hiệu						Lệnh của DMAC 8237
A3	A2	A1	A0	I/OR#	I/OW#	
1	0	0	0	0	1	Đọc thanh ghi trạng thái
1	0	0	0	1	0	Nạp vào thanh ghi lệnh
1	0	0	1	0	1	Không hợp lệ
1	0	0	1	1	0	Nạp thanh ghi yêu cầy DMA
1	0	1	0	0	1	Không hợp lệ
1	0	1	0	1	0	Ghi từng bit cho thanh ghi mặt nạ
1	0	1	1	0	1	Không hợp lệ
1	0	1	1	1	0	Nạp thanh ghi chế độ
1	1	0	0	0	1	Không hợp lệ
1	1	0	0	1	0	Xóa mạch Flip-Flop bên trong
1	1	0	1	0	1	Đọc thanh ghi tạm thời
1	1	0	1	1	0	Xóa toàn bộ DMAC (RESET)
1	1	1	0	0	1	
1	1	1	0	1	0	Không hợp lệ
1	1	1	1	0	1	
1	1	1	1	1	0	Ghi toàn bộ thanh ghi mặt nạ

Đối với từng kênh DMA, để truy nhập đến các thanh ghi địa chỉ và số từ thì trong mã lệnh đưa ra từ CPU tín hiệu A3 = 0. Khi đó, bus dữ liệu D7-D0 của DMAC 8237 sẽ có những giá trị khác nhau tùy thuộc vào mã lệnh (bảng 6.4).

Bảng 6.4: Giá trị bus dữ liệu D7-D0 của DMAC 8237

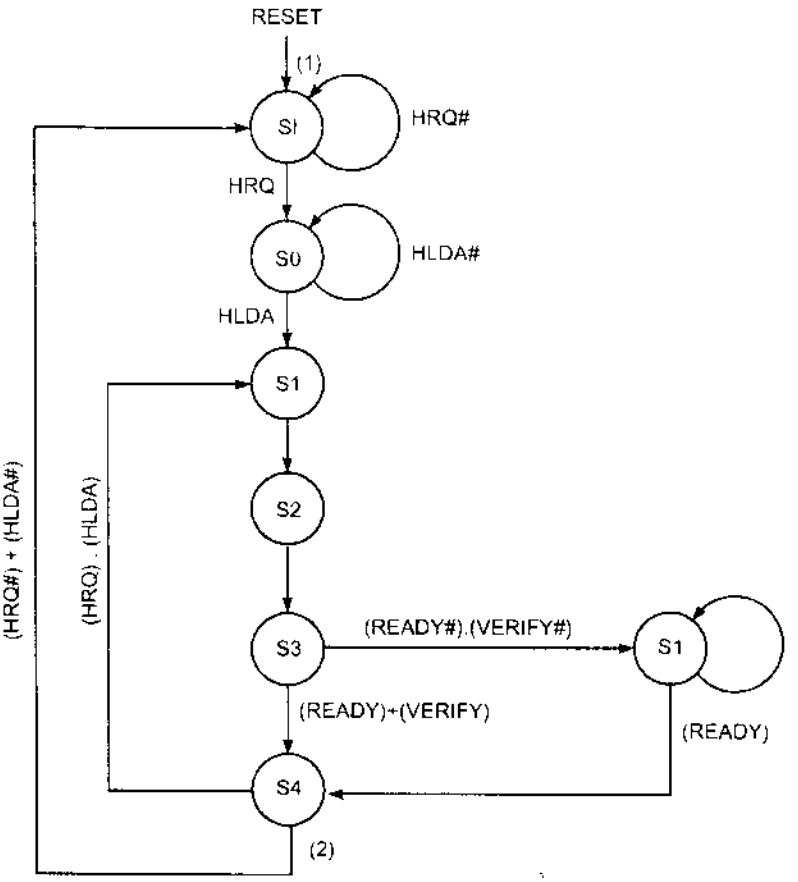
Tín hiệu							Lệnh	Thanh ghi Của DMAC	Kênh	Mạch FF trong	Bus dữ liệu (D7-D0)
A3	A2	A1	A0	CS#	I/OR#	I/OW#					
0	0	0	0	0	1	0	Ghi	BAR và CAR	0	0	A7-A0
0	0	0	0	0	1	0	Ghi	BAR và CAR	0	1	A15-A8
0	0	0	0	0	0	1	Đọc	CAR	0	0	A7-A0
0	0	0	0	0	0	1	Đọc	CAR	0	1	A15-A8
0	0	0	1	0	1	0	Ghi	BWR và CWR	0	0	W7-W0

Tín hiệu							Lệnh	Thanh ghi Của DMAC	Kênh	Mạch FF trong	Bus dữ liệu (D7-D0)
A3	A2	A1	A0	CS#	I/OR#	I/OW#					
0	0	0	1	0	1	0	Ghi	BWR và CWR	0	1	W15-W0
0	0	0	0	0	0	1	Đọc	CWR	0	0	W7-W0
0	0	0	0	0	0	1	Đọc	CWR	0	1	W15-W8
0	0	1	0	0	1	0	Ghi	BAR và CAR	1	0	A7-A0
0	0	1	0	0	1	0	Ghi	BAR và CAR	1	1	A15-A8
0	0	1	0	0	0	1	Đọc	CAR	1	0	A7-A0
0	0	1	0	0	0	1	Đọc	CAR	1	1	A15-A8
0	0	1	1	0	1	0	Ghi	BWR và CWR	1	0	W7-W0
0	0	1	1	0	1	0	Ghi	BWR và CWR	1	1	W15-W0
0	0	1	1	0	0	1	Đọc	CWR	1	0	W7-W0
0	0	1	1	0	0	1	Đọc	CWR	1	1	W15-W8
0	1	0	0	0	1	0	Ghi	BAR và CAR	2	0	A7-A0
0	1	0	0	0	1	0	Ghi	BAR và CAR	2	1	A15-A8
0	1	0	0	0	0	1	Đọc	CAR	2	0	A7-A0
0	1	0	0	0	0	1	Đọc	CAR	2	1	A15-A8
0	1	0	1	0	1	0	Ghi	BWR và CWR	2	0	W7-W0
0	1	0	1	0	1	0	Ghi	BWR và CWR	2	1	W15-W0
0	1	0	1	0	0	1	Đọc	CWR	2	0	W7-W0
0	1	0	1	0	0	1	Đọc	CWR	2	1	W15-W8
0	1	1	0	0	1	0	Ghi	BAR và CAR	3	0	A7-A0
0	1	1	0	0	1	0	Ghi	BAR và CAR	3	1	A15-A8
0	1	1	0	0	0	1	Đọc	CAR	3	0	A7-A0
0	1	1	0	0	0	1	Đọc	CAR	3	1	A15-A8

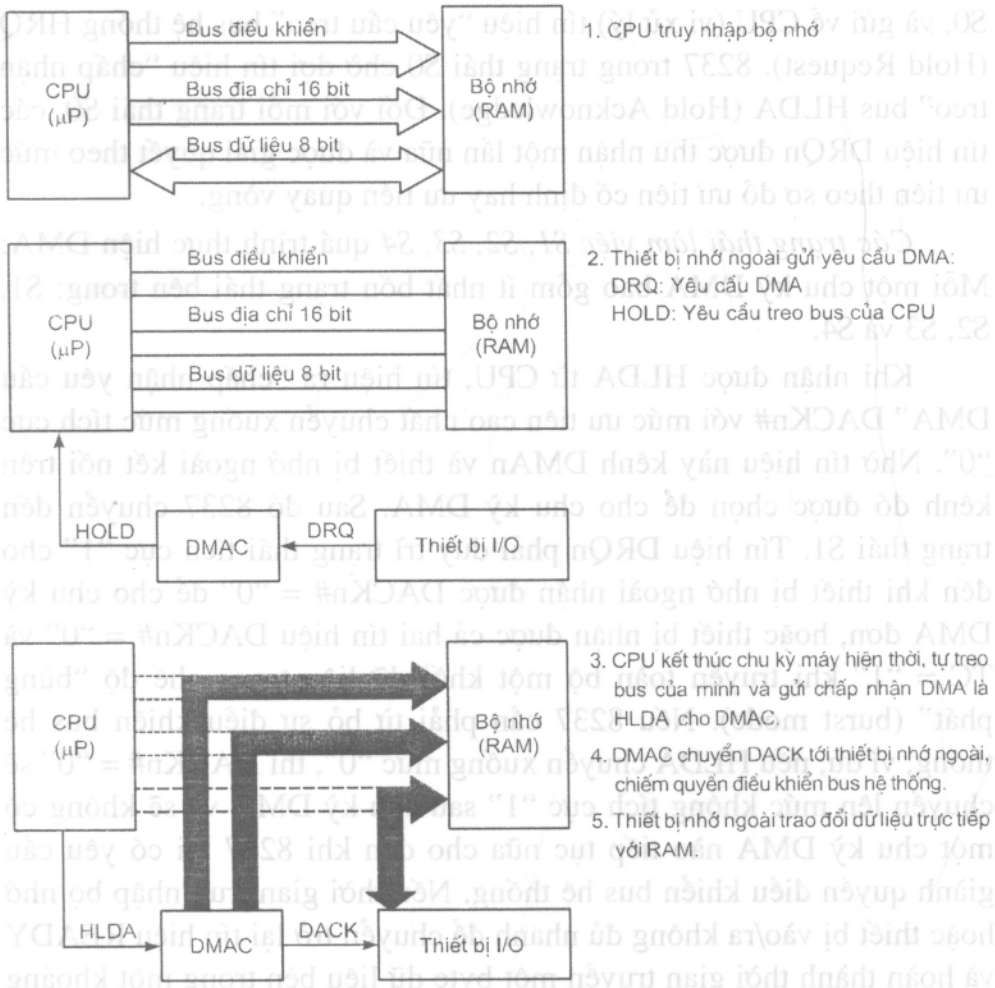
Tín hiệu							Lệnh	Thanh ghi Của DMAC	Kênh	Mạch FF trong	Bus dữ liệu (D7-D0)
A3	A2	A1	A0	CS#	I/OR#	I/OW#					
0	1	1	1	0	1	0	Ghi	BWR và CWR	3	0	W7-W0
0	1	1	1	0	1	0	Ghi	BWR và CWR	3	1	W15-W0
0	1	1	1	0	0	1	Đọc	CWR	3	0	W7-W0
0	1	1	1	0	0	1	Đọc	CWR	3	1	W15-W8

e. Các thao tác bên trong DMAC 8237 và quá trình cơ bản của DMA

Các thao tác bên trong 8237 có thể thực hiện thông qua 7 trạng thái của quá trình DMA (hình 6.25), và quá trình cơ bản của DMA được minh họa qua hình 6.26.



Hình 6.25: Đồ thị quá trình DMA



Hình 6.26: Quá trình cơ bản của DMA

Khoảng thời gian của mỗi trạng thái xác định nhờ nhịp đồng hồ CLK.

Trạng thái SI (Idle state), trạng thái nghỉ: Khi 8237 không thực hiện một chu kỳ DMA nào thì nó ở trong trạng thái nghỉ SI, nó chỉ nhận lệnh từ CPU.

Trạng thái S0: Là trạng thái đầu tiên của quá trình DMA. Một chu kỳ DMA bắt đầu khi một hay nhiều tín hiệu yêu cầu DMA (DRQn) chuyển lên mức tích cực "1". Khi đó, 8237 sẽ chuyển vào trạng thái

S0, và gửi về CPU (vi xử lý) tín hiệu "yêu cầu treo" bus hệ thống HRQ (Hold Request). 8237 trong trạng thái S0 chờ đợi tín hiệu "chấp nhận treo" bus HLDA (Hold Acknowledge). Đối với mỗi trạng thái S0, các tín hiệu DRQn được thu nhận một lần nữa và được giải quyết theo mức ưu tiên theo sơ đồ ưu tiên cố định hay ưu tiên quay vòng.

Các trạng thái làm việc S1, S2, S3, S4 quá trình thực hiện DMA: Mỗi một chu kỳ DMA bao gồm ít nhất bốn trạng thái bên trong: S1, S2, S3 và S4.

Khi nhận được HLDA từ CPU, tín hiệu ra "chấp nhận yêu cầu DMA" DACKn# với mức ưu tiên cao nhất chuyển xuống mức tích cực "0". Nhờ tín hiệu này kênh DMAn và thiết bị nhớ ngoài kết nối trên kênh đó được chọn để cho chu kỳ DMA. Sau đó 8237 chuyển đến trạng thái S1. Tín hiệu DRQn phải duy trì trạng thái tích cực "1" cho đến khi thiết bị nhớ ngoài nhận được DACKn# = "0" để cho chu kỳ DMA đơn, hoặc thiết bị nhận được cả hai tín hiệu DACKn# = "0" và TC = "1" khi truyền toàn bộ một khối dữ liệu trong chế độ "bùng phát" (burst mode). Nếu 8237 cần phải từ bỏ sự điều khiển bus hệ thống, ví dụ, nếu HLDA chuyển xuống mức "0", thì DACKn# = "0" sẽ chuyển lên mức không tích cực "1" sau chu kỳ DMA và sẽ không có một chu kỳ DMA nào tiếp tục nữa cho đến khi 8237 lại có yêu cầu giành quyền điều khiển bus hệ thống. Nếu thời gian truy nhập bộ nhớ hoặc thiết bị vào/ra không đủ nhanh để chuyển trở lại tín hiệu READY và hoàn thành thời gian truyền một byte dữ liệu bên trong một khoảng thời gian xác định nào đó, thì một hoặc một số trạng thái chờ SW (State Wait) được chèn thêm vào giữa trạng thái S3 và S4. Tín hiệu READY không cần thiết trong các chu kỳ so sánh của DMA. Các giá trị tối thiểu và tối đa của thời gian thiết lập READY (Trs), của thời gian thiết lập ghi dữ liệu (Tdw), của thời gian truy nhập đọc dữ liệu (Trd) và của thời gian thiết lập HLDA (Tqs) được liệt kê thông qua các đồ thị thời gian các chu kỳ ghi/đọc DMA.

Trong các chu kỳ ghi DMA, dữ liệu truyền từ thiết bị nhớ ngoài ghi thẳng vào RAM thông qua bus dữ liệu (D7-D0), DMAC 8237 tạo ra I/OR# = 0 ở đầu trạng thái S2 để đọc dữ liệu từ thiết bị nhớ ngoài (thông qua bảng điều khiển thiết bị nhớ ngoài) về bus dữ liệu (D7-D0)

và tạo ra MEMW# = 0 đầu trạng thái S3 để ghi dữ liệu đọc từ bus dữ liệu (D7-D0) ngoài vào RAM.

Trong các chu kỳ đọc DMA, dữ liệu đọc từ RAM ghi thẳng tới thiết bị nhớ ngoài thông qua bus dữ liệu (D7-D0), DMAC 8237 tạo ra MEMR# = 0 ở đầu trạng thái S2 để đọc dữ liệu từ RAM ra bus dữ liệu (D7-D0), tạo ra tín hiệu I/OW# = 0 ở đầu trạng thái S3 để ghi dữ liệu từ bus dữ liệu vào thiết bị nhớ ngoài (thông qua bộ điều khiển thiết bị nhớ ngoài).

Trong chu kỳ DMA không có sự điều khiển nào có thể được thực hiện từ CPU thông qua chọn chip DMAC (CS# = 1). Để thực hiện điều này, tín hiệu HOLDA = 1 (U63, 74LS175) sẽ tạo ra AEN# = 0 khóa bộ giải mã chọn các chip ngoại vi (U31, 74LS138).

Trước khi thực hiện quá trình DMA, chúng ta phải thực hiện đọc trạng thái của DMAC và đưa ra các mã lệnh điều khiển DMAC 8237. Đó là các quá trình đọc và ghi trước DMA. Trong các chu kỳ này tín hiệu chọn chip CS# = 0 và DMAC 8237 trong trạng thái nghỉ (SI).

Khi một thiết bị nhớ ngoài có yêu cầu DMA, nó chuyển tín hiệu DRQn lên mức tích cực "1", thì để thực hiện một quá trình DMA, mọi DMAC phải giải quyết những vấn đề sau đây:

1. Gửi yêu cầu DMA tới CPU.

2. Điều khiển bus hệ thống mà không làm ảnh hưởng đến hoạt động bình thường của CPU hoặc không gây xung đột trên bus hệ thống.

3. Xác định và tạo địa chỉ đầu của vùng nhớ chính, tính số lượng từ (theo byte) cần phải truyền (từ thiết bị nhớ ngoài đến vùng nhớ RAM, hay từ vùng nhớ RAM ra thiết bị nhớ ngoài).

4. Thông báo kết thúc quá trình DMA.

f. Các chu kỳ làm việc của DMAC 8237

(1) Chu kỳ "nghỉ"

Khi không có yêu cầu DMA từ thiết bị ngoại vi thì DMAC 8237 ở trong chu kỳ nghỉ, trong trạng thái SI. Trong chu kỳ này DMAC kiểm tra các đường tín hiệu DRQn theo từng chu kỳ nhịp đồng hồ CLK của hệ thống, đồng thời kiểm tra chân tín hiệu vào chọn chip CS#

để xem CPU có ý định gửi lệnh đến DMAC để đọc/ghi nội dung các thanh ghi, trạng thái của DMAC hay không. Nếu CS# = "0" và DMAC không có yêu cầu treo bus, tức là HRQ = "0", thì DMAC tuân theo sự điều khiển chương trình của CPU. Khi đó A3-A0 là các đầu vào địa chỉ mà CPU đưa ra để chọn các thanh ghi bên trong DMAC. Nội dung các thanh ghi của DMAC được đọc về CPU hay được gửi từ CPU qua bus dữ liệu D7-D0 được thực hiện bởi các xung I/OR#, I/OW# được tạo từ CPU. Lúc này I/OR# và I/OW# là các tín hiệu vào của DMAC (trong khi CS# = "0" phải được duy trì). Như vậy, trong chu kỳ nghỉ các thanh ghi của DMAC có thể được truy nhập tới qua bus địa chỉ và dữ liệu nhờ chương trình của CPU.

(2) Chu kỳ DMA, hay "hoạt động"

Khi thiết bị nhớ ngoài có yêu cầu DMA, nó chuyển tín hiệu DRQn lên mức tích cực "1". Nhận được DRQn = "1", DMAC chuyển sang từ chu kỳ "nghỉ" sang chu kỳ DMA, hay chu kỳ "hoạt động" và phát yêu cầu treo về CPU bằng tín hiệu HRQ = "1". Chu kỳ DMA gồm tối thiểu 4 trạng thái S1, S2, S3, và có thể thêm một số trạng thái chờ SW, như hình 6.25 đã trình bày. Chu kỳ DMA có thể là chu kỳ đọc dữ liệu từ RAM về thiết bị nhớ ngoài hay là chu kỳ ghi dữ liệu từ thiết bị nhớ ngoài về RAM.

Trong chu kỳ DMA, các xung I/OR#, và I/OW# là các tín hiệu ra của DMAC (được DMAC tạo ra).

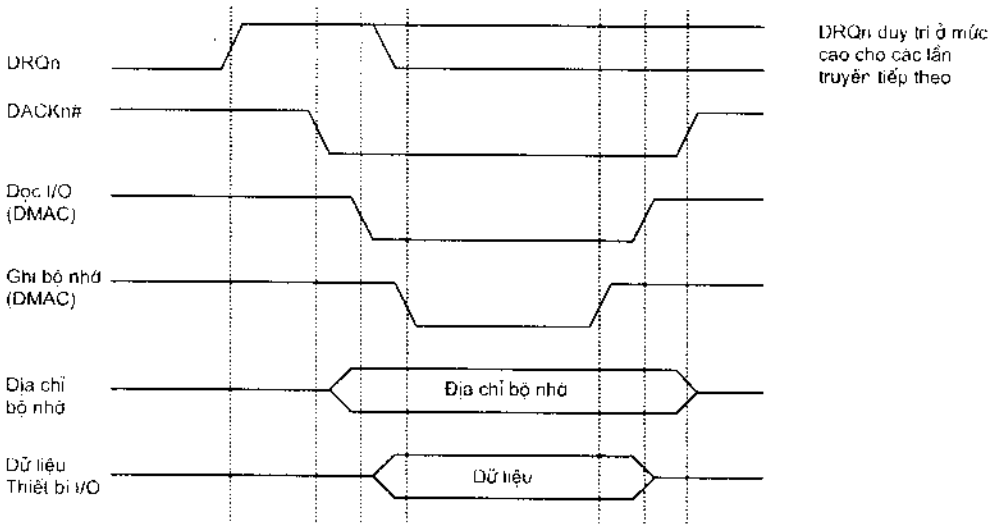
g. Các kiểu truyền và các chế độ DMA

DMAC có 2 kiểu truyền DMA cho từng kênh. Đó là truyền nhanh (flyby) và truyền đọc-và-ủy thác (fetch-and-deposit).

(1) Truyền nhanh flyby

Kiểu truyền flyby chỉ cần một chu kỳ bus để thực hiện truyền nhanh dữ liệu. Trong kiểu truyền này, dữ liệu được đồng thời đọc từ nguồn và ghi ra đích. Thiết bị ngoại vi có yêu cầu DMA gửi DRQn tới DMAC. DMAC trả lời bằng cách giành lấy quyền điều khiển bus hệ thống từ CPU và đưa ra địa chỉ bộ nhớ mà từ CPU đã lập trình cho DMAC trước đó. Đồng thời, DMAC gửi một xác nhận DMA

(DACKn#) cho thiết bị có yêu cầu DMA. DACKn# báo cho thiết bị yêu cầu điều khiển đưa dữ liệu lên bus hệ thống hoặc chốt dữ liệu từ bus hệ thống (từ bộ nhớ) phụ thuộc vào hướng truyền dữ liệu. Nói một cách khác, truyền nhanh flyby DMA giống như một chu kỳ đơn ghi hay đọc bộ nhớ nhưng nhờ DMAC điều khiển đưa ra địa chỉ và thiết bị ngoại vi đọc hoặc ghi dữ liệu. Truyền nhanh flyby DMA chỉ cần một chu kỳ bus đơn để truyền hết bộ nhớ, nên có hiệu suất cao, nhưng truyền giữa bộ nhớ và bộ nhớ là không thể được. Hình 6.27 mô tả các tín hiệu trong truyền flyby DMA.



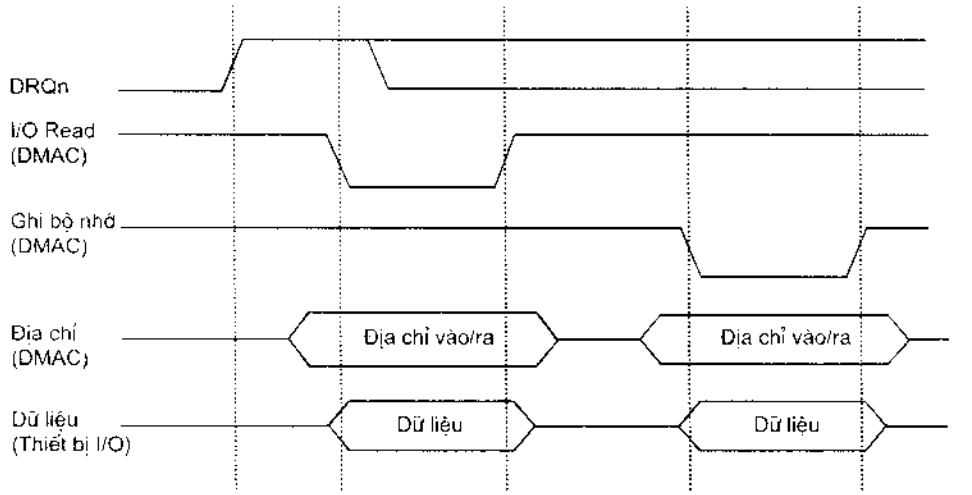
Hình 6.27: Kiểu truyền nhanh flyby DMA

(2) Truyền đọc-và-ủy thác (fetch-and-deposit)

Kiểu truyền đọc và ký thác gồm 2 chu kỳ nhớ hay vào/ra, 2 địa chỉ. Trước hết, dữ liệu được đọc từ bộ nhớ hay thiết bị vào/ra vào thanh ghi tạm thời (TR) bên trong DMAC trong chu kỳ thứ nhất. Sau đó dữ liệu mới được ghi từ DMAC ra bộ nhớ hay thiết bị ngoại vi trong chu kỳ thứ 2 (hình 6.28).

Mặc dù kiểu truyền đọc-và-ủy thác không hiệu quả so với kiểu truyền flyby vì bus hệ thống bị DMAC chiếm dữ lâu hơn (2 chu kỳ),

nhưng nó thuận tiện để phối ghép với các thiết bị ngoại vi có kích thước dữ liệu khác với bus dữ liệu của hệ thống. Ví dụ, DMAC có thể thực hiện 2 phép đọc dữ liệu 16-bit từ một vùng (bộ nhớ hay thiết bị) để sau đó thực hiện một phép ghi dữ liệu 32-bit đến một vùng khác. Và kiểu truyền này cũng có thể thực hiện được cho các tham chiếu bộ nhớ-đến-bộ nhớ. Như vậy, ngoài những thanh ghi bên trong như đã trình bày ở trên, DMAC phải có 2 thanh ghi địa chỉ cho từng kênh DMA (thanh ghi địa chỉ nguồn và thanh ghi địa chỉ đích) và thanh ghi đệm dữ liệu chu bus.



Hình 6.28: Kiểu truyền đọc-và-ủy thác

Bổ sung cho các kiểu truyền DMA trên đây, DMAC còn có các chế độ truyền như sau:

(3) Chế độ truyền đơn (single transfer mode)

DMAC chỉ gửi yêu cầu HRQ cho từng lần chuyển một byte lẻ. Địa chỉ cơ sở của vùng nhớ chứa khối dữ liệu được tăng (incremented) hay giảm (decremented), và thanh ghi đếm được giảm đi một lần. Trong khi DRQn duy trì tích cực, DMAC giải phóng bus hệ thống để cho phép CPU thực hiện một chu kỳ bus, sau đó DMAC lại tiếp nhận quyền làm chủ bus để thực hiện một chu kỳ bus truyền byte dữ liệu

tiếp theo. Khi nào thanh ghi đếm đếm đủ thì quá trình DMA kết thúc. Chuyển từng byte đơn lẻ có thể sử dụng lấy lén hay truy nhập chèn (hình 6.21 b, c)

(4) Chế độ chuyển theo yêu cầu (Demand transfer mode)

Trong chế độ này DMAC chuyển liên tục cả mảng dữ liệu cho đến khi TC chuyển lên mức "1" (thanh ghi đếm số từ trở về 0) hoặc cho đến khi không có DRQn nào nữa (DRQn không còn tích cực). Tức là thiết bị ngoài đã trao đổi hết dữ liệu với RAM và không gửi tiếp HRQn nữa. Nếu thiết bị ngoài có nhu cầu chuyển mảng dữ liệu tiếp theo thì nó gửi tiếp HRQn về DMAC, và quá trình DMA lại tiếp tục. Giữa 2 quá trình DMA, CPU có thể hoạt động bình thường. Nếu thiết bị ngoại vi không đủ nhanh để đọc/ghi dữ liệu thì nó chuyển tín hiệu DRQn không tích cực nữa. Khi thiết bị ngoại vi sẵn sàng nhận/gửi thêm dữ liệu thì nó chuyển DRQn lên tích cực và tiếp tục truyền DMA.

(5) Chế độ truyền khối (block transfer mode)

Quá trình truyền DMA chỉ kết thúc khi TC đếm đủ số byte của khối được truyền. Trong chế độ này, một khối dữ liệu dung lượng tối đa 64 kB có thể được truyền ở các kênh 0-3 hoặc 128 kB ở các kênh 5-7 (trong AT). Nếu kết thúc truyền một khối mà DRQn còn tích cực thì thực hiện truyền khối tiếp theo. Chế độ này đảm bảo truyền nhanh dữ liệu (burst mode, hình 6.21a).

(6) Chế độ ưu tiên

DMAC có thể làm việc ở hai chế độ ưu tiên khác nhau theo quy định bằng phần mềm từ CPU: chế độ ưu tiên cố định, chế độ ưu tiên vòng.

Trong chế độ ưu tiên cố định: kênh 0 có mức ưu tiên cao nhất và theo thứ tự giảm dần. Kênh 3 có mức ưu tiên thấp nhất khi chỉ có một DMAC. Nếu có nối tầng các DMAC thì số kênh càng lớn thì mức ưu tiên càng thấp. Trong chế độ ưu tiên vòng, DMAC đảm bảo bốn thiết bị ngoài nối với bốn kênh DRQn (n = 0, 1, 2, 3) có mức ưu tiên được phục vụ như nhau, không yêu cầu nào có mức ưu tiên lớn hơn yêu cầu khác. Nghĩa là yêu cầu DRQ0 được phục vụ xong sẽ quay về cuối vòng để yêu cầu DRQ1 được phục vụ.

Sau khi phục vụ xong DRQ1 lại xếp sau DRQ0 cho lần 2. Tiếp theo yêu cầu DRQ2 được phục vụ. Phục vụ xong nó quay về xếp sau DRQ1 cho lần 2. Tiếp theo yêu cầu DRQ3 được phục vụ. Được phục vụ xong DRQ3 lại xếp sau DRQ2 cho lần 2. Lần 2 (hay vòng 2), DRQ0 lại được phục vụ đầu tiên. Chế độ ưu tiên quay vòng này tương tư như trường hợp kết nối tầng các chip 8259A mà ta đã xét. Rõ ràng, trường hợp xấu nhất đối với yêu cầu DRQ3 là trước đó có 3 yêu cầu DRQ0, DRQ1, và DRQ2, bởi vì khi đó, DRQ3 phải đợi ba yêu cầu có mức ưu tiên cao hơn được phục vụ xong.

Trước tiên CPU phải lập thứ tự ưu tiên, mặt nạ cho từng kênh DMA. Chế độ nối tầng cho phép tự khởi động, định địa chỉ cho từng kênh, qui định độ dài mảng dữ liệu (theo từ) cần trao đổi bằng cách gửi lệnh và mã điều khiển vào các thanh ghi bên trong các DMAC liên quan đến các kênh.

Trong quá trình hoạt động DMAC luôn cập nhật trạng thái của mình vào thanh ghi trạng thái để CPU có thể đọc bất cứ lúc nào. Thông tin trạng thái cho biết kênh DMA nào đạt đến số đếm TC, tức là đã chuyển xong mảng dữ liệu có độ dài xác định trong thanh ghi số đếm từ gốc, hoặc bị bắt buộc kết thúc DMA do tác động của tín hiệu EOP từ ngoài vào. DMAC có thể nhận lệnh xóa từ CPU: xóa toàn bộ: tất cả các thanh ghi, Flip-flop bên trong DMAC đều được xóa và sau khi kết thúc lệnh xóa toàn bộ này (tương tự như xóa cứng) DMAC chuyển đến trạng thái "nghỉ" SI. Phương pháp thông dụng và đơn giản đối với hệ vi xử lý là bắt buộc vi xử lý tự treo các đường bus địa chỉ, dữ liệu của mình (đưa lên trạng thái trở kháng cao) và nhường quyền điều khiển bus hệ thống cho DMAC. Khi DMAC nhận lấy quyền điều khiển bus hệ thống thì nó là chủ của hệ thống và điều khiển sự trao đổi dữ liệu giữa thiết bị nhớ ngoài và bộ nhớ RAM.

(7) Chế độ nối tầng (cascade mode)

Để mở rộng các yêu cầu DMA có thể kết nối tầng các chip DMAC (cascade connection). Khi đó các tín hiệu HRQ và HLDA từ các DMAC ở tầng 2 được nối với DRQn và DACKn# ở tầng 1. Hình 6.27 cho ví dụ kết nối tầng các DMAC để mở rộng phục vụ DMA cho nhiều thiết bị ngoài. Có 4 mạch DMAC 8237 ở tầng 2 và một DMA ở

tầng 1. Như thế có thể có tới 16 yêu cầu DMA được phục vụ, và được đánh số từ nhỏ đến lớn. Những thiết bị ngoài có tốc độ truy nhập DMA nhanh được gán mức ưu tiên cao hơn. Mạch DMAC nằm ở tầng 1 nơi kết nối với CPU được gọi là DMAC chủ, còn những DMAC khác gọi là các DMA thợ. DMAC ở tầng 1 có nhiệm vụ giải quyết các yêu cầu DMA của tầng 2 theo ưu tiên và chuyển yêu cầu có mức ưu tiên cao nhất đến CPU mà không tạo địa chỉ và các tín hiệu điều khiển khác với bình thường khi chỉ có một DMAC.

(8) Chế độ so sánh

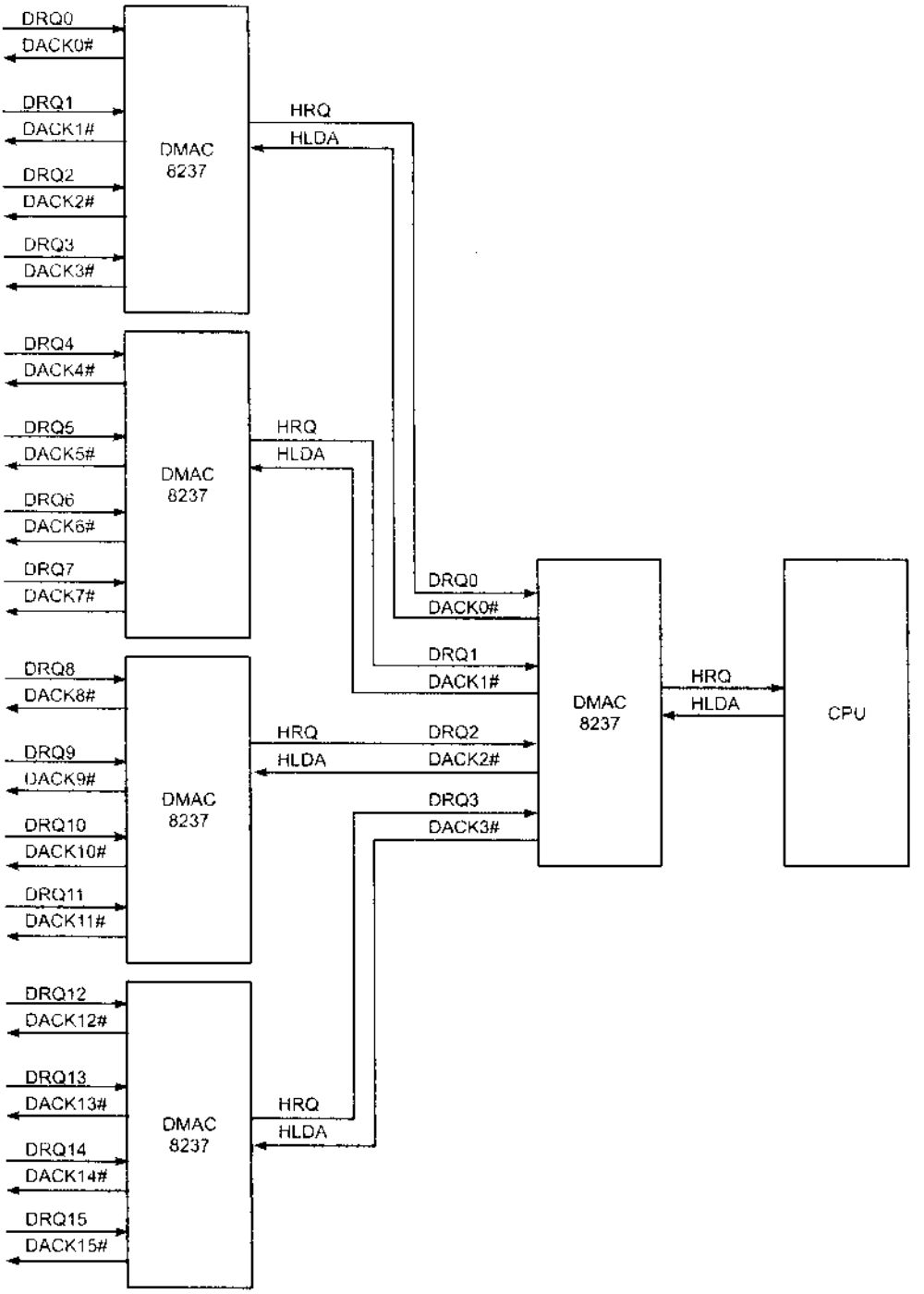
Chế độ này dự phòng cho kiểm tra và dự báo của DMAC khi DRQn tích cực, nhưng không có truyền dữ liệu nào vì không có các tín hiệu truy nhập bộ nhớ và thiết bị vào/ra (nhớ ngoài) sinh ra.

h. Kết nối DMAC 8237 trong IBM PC/XT/AT

Chúng ta xét vận hành của 8237 trong liên kết với 8086/8088 trong hệ thống máy tính IBM PC/XT/AT như là một ví dụ minh họa cụ thể điều khiển DMA.

IBM PC/XT chỉ sử dụng 1 DMAC 8237A cho 4 kênh DMA dữ liệu 8-bit. Trong đó, kênh 2 dùng cho truyền DMA giữa FDD và bộ nhớ (DRQ2 và DACK2#), kênh 3 dùng cho vận chuyển DMA giữa HDD và bộ nhớ (DRQ3 và DACK3#), và kênh 0 cho các chu kỳ làm tươi DRAM (DRQ0 và DACK0#). Ngoài ra có bổ sung thêm bên ngoài DMAC thanh ghi trang (page register) 8-bit để mở rộng địa chỉ bộ nhớ 16-bit mà DMAC đưa ra thành 24-bit cho hệ thống nhớ của PC. Thanh ghi trang này không tự động tăng lên giá trị tiếp theo khi thanh ghi địa chỉ của DMAC đạt được giá trị cuối 0xFFFFh, tức là đạt đến ranh giới trang 64 kB.

Để lập trình DMAC 8237, từ 8086/8088 phải đưa ra lệnh vào/ra (IN, OUT) với dải địa chỉ vào/ra là 000-00Fh (phụ lục sơ đồ bảng mạch chính IBM PC/XT). Khi đó, tín hiệu I/OR#, hay I/OW# được tạo ra từ lệnh IN, hay OUT của 8086/8088 và thông qua giải mã tổ hợp trạng thái S0#, S1#, S2# nhờ mạch điều khiển bus 8288 (mạch U12). DMAC 8237 được chọn nhờ tín hiệu CS# = 0 thông qua địa chỉ vào/ra trong lệnh IN, hay OUT. Vì khi đó, tín hiệu CS# = DMACS# = 0 (mạch U31):



Hình 6.29: Kết nối tầng các mạch DMAC để mở rộng DMA

(AEN# = 1)(A9 = 0)(A8 = 0)(A7 = 0)(A6 = 0)(A5 = 0)

Các đường địa chỉ A3 - A0 như là mã lệnh điều khiển cho DMAC 8237 cùng được đưa ra trong lệnh IN, OUT từ 8086/8088 thông qua bus hệ thống.

Như vậy, khi đưa lệnh điều khiển ra DMAC 8237, các tín hiệu CS# = 0, A3 - A0, I/OR#, I/OW# là các tín hiệu vào của DMAC 8237, trong điều kiện chưa có quá trình DMA, tức là DMAAEN# = 1 (mạch U93).

Khi AEN = 1 (hay AEN# = 0) thì khóa chọn DMAC 8237 từ CPU (DMACS# = 1). Lúc này DMAC 8237 là chủ điều khiển bus địa chỉ, bus dữ liệu và sản sinh ra các tín hiệu lệnh I/OR#, I/OW#, MEMR#, MEMW# điều khiển quá trình DMA giữa thiết bị nhớ ngoài và RAM.

Các tín hiệu DRQ1 - DRQ3 và DACK1# - DACK3#, T/C của DMAC 8237 đưa đến các khe mở rộng của bus hệ thống để ghép nối với các bộ thích ứng (adaptor) của thiết bị ngoại vi. Ví dụ:

- Bảng điều khiển thiết bị nhớ trên đĩa mềm (FDC) trong hệ thống IBM PC/XT sử dụng các tín hiệu DRQ2 và DACK2#, T/C để thực hiện quá trình DMA. Xung T/C ra của 8237 (EOP) chuyển lên mức tích cực "1" khi byte đầu tiên của dữ liệu được chuyển khi bắt đầu quá trình DMA.

- Bảng điều khiển thiết bị nhớ trên đĩa cứng (HDD) trong hệ thống IBM PC/XT sử dụng các tín hiệu DRQ3 và DACK3# để thực hiện quá trình DMA. Các tín hiệu I/OR# và I/OW# trên bảng điều khiển đĩa cứng và đĩa mềm được lấy từ CPU khi từ CPU có lệnh đưa ra điều khiển hay đọc trạng thái của thiết bị đĩa, hoặc được tạo ra từ DMAC 8237 khi có quá trình DMA giữa thiết bị đĩa cứng/đĩa mềm và RAM.

Tín hiệu DRQ0 và DACK0# dùng cho lập trình làm tươi bộ nhớ DRAM. Quá trình làm tươi DRAM được thực hiện nhờ lập trình kênh thời gian 1 (DRQ0 = OUT1) của bộ đếm thời gian 8253 (mạch U76) theo chu kỳ: cứ 15 μs 8253 sinh ra DRQ0 = OUT1 để thực hiện một chu kỳ làm tươi DRAM. Quá trình này tạo ra một chu kỳ đọc DRAM

(DACK0 = 1, XMEMR# = 0) ở trên bảng mạch hệ thống cả ở trên bảng mạch DRAM mở rộng. Các chu kỳ DMA làm tươi DRAM kéo dài trong 4 chu kỳ nhịp CLK (khoảng 218ns với CLK = 4,77 MHz). Tất cả các chu kỳ DMA truyền dữ liệu với bộ nhớ, ngoại trừ DMA làm tươi DRAM kéo dài trong 5 chu kỳ nhịp CLK với 273ns (CLK = 4,77 MHz). DRAM sử dụng trong IBM PC/XT 8086/8088 có thời gian truy nhập là 150 ns và thời gian của một chu kỳ truy nhập bộ nhớ là 410 ns.

Với 8086/8088 làm việc với nhịp 4,77MHz trong IBM PC/XT thì truyền DMA trong chế độ chuyển theo yêu cầu có thể đạt tốc độ 900 kbit/s.

Trong IBM PC/AT và sau này, có 2 DMAC 8237 nối tầng. Trong đó, ở DMAC chủ (thứ nhất), kênh 2 cho truyền dữ liệu giữa FDD (đĩa mềm) và bộ nhớ RAM, kênh 0 cho các chu kỳ làm tươi DRAM, và kênh 4 được dùng để nối cứng với DMAC 8237 thứ 2. Như vậy IBM PC/AT có 7 kênh DMA.

Khi bật nguồn, các DMAC được khởi tạo và BIOS thiết lập kênh 4 kết nối tầng. Sự thiết lập này không bao giờ được thay đổi, nếu không các kênh DMA 5, 6 và 7 sẽ không có tác dụng. Các kênh 0 - 3 của DMAC thứ nhất thực hiện các truyền DMA dữ liệu 8-bit giữa các mạch thích ứng vào/ra (I/O adaptors) và bộ nhớ RAM 8-bit hay 16-bit. Các kênh DMA 6 - 7 thực hiện truyền DMA dữ liệu giữa các mạch thích ứng vào/ra 16-bit và bộ nhớ 16-bit. Các thanh ghi đếm từ trong DMAC thứ 2 hành động như là thanh ghi đếm từ 16-bit. IBM PC/AT cũng bổ sung thêm bên ngoài các DMAC thanh ghi trang (page register) 8-bit để mở rộng địa chỉ bộ nhớ 16-bit (từ các DMAC) lên thành 24-bit cho hệ thống nhớ của 80286. Với truyền DMA dữ liệu 16-bit, thì ranh giới của khối dữ liệu lên đến 128 kB. Với 8 MHz ISA bus, và DMAC vận hành ở 4 MHz sẽ đảm bảo tốc độ truyền dữ liệu 16-bit trong chế độ chuyển theo yêu cầu là 1,6 Mbit/s. Một chu kỳ làm tươi DRAM trong IBM PC/AT cần phải thực hiện cứ từng 15 μs một. Tức là trong các IBM PC/XT/AT, cứ sau 15 μs bus vào/ra của hệ thống cần phải được dành cho chu kỳ làm tươi DRAM. Do vậy, chế độ truyền khối trong DMA trong IBM PC/XT/AT được sử dụng không bình thường.

BÀI TẬP ÔN LUYỆN

1. Phân biệt các kỹ thuật ngắt của 8085 (kỹ thuật quét, véc-tơ, chuỗi), ứng dụng của chúng cho các loại thiết bị ngoại vi như thế nào.
2. Phân biệt ngắt cứng và ngắt mềm.
3. Lệnh RIM cung cấp thông tin gì liên quan đến ngắt.
4. Thế nào là ngắt bị che.
5. Ngắt không bị che dùng trong trường hợp nào.
6. Thế nào gọi là véc-tơ ngắt.
7. Làm thế nào để cấm và cho phép ngắt INTR từ bên ngoài.
8. Trong các hệ thống 8086/8088 địa chỉ bắt đầu của chương trình phục vụ ngắt được cất giữ. Làm thế nào để đọc địa chỉ đầu này ra.
9. Đặc điểm của cơ chế xử lý ngắt trong 8086/8088 có gì khác so với 8085.
10. Trong hệ thống đa xử lý 8086, sự khác nhau như thế nào giữa các bus public, local và private.
11. Một yêu cầu ngắt được chấp nhận gọi tới chương trình con (chương trình phục vụ ngắt) để thực hiện. Sau khi thực hiện phục vụ ngắt ta cần phải thực hiện lệnh... để trở về chương trình chính.
    a) JMP; b) IRET; c) RET; d) bất kỳ a, b, c.
12. Hãy tìm trong các ngắt ROM-BIOS và ngắt chức năng DOS nào có xử lý với 8253.
13. Hãy tìm trong các ngắt ROM-BIOS và ngắt chức năng DOS nào có xử lý với 8255.
14. Trong thanh ghi dịch ISR của 8254A sự thiết lập các bit của nó để làm gì?

15. Hãy tìm trong các ngắt ROM-BIOS và ngắt chức năng DOS nào có xử lý với 8259A.
16. Dựa vào sơ đồ IBM PC/XT hãy giải thích chức năng của chip DMA 8257A.
17. Có thể thay vào vị trí DMA 8257A bằng chip DMA 8237A được không.
18. Các thiết bị ngoại vi nào cần phải và có thể thực hiện được DMA, tại sao.
19. Hãy tìm trong các ngắt ROM-BIOS và ngắt chức năng DOS nào có xử lý với 8257A.
20. Hãy tìm trong các ngắt ROM-BIOS và ngắt chức năng DOS ngắt nào có xử lý với 8250.
21. Khi nào thì sử dụng phương pháp I/O bằng DMA.
22. Khi nào thì sử dụng phương pháp I/O bằng ngắt.
23. Khi nào thì sử dụng phương pháp I/O bằng bầu chọn (polling).
24. Phương pháp I/O giữa bảng phím và CPU là gì.
25. Các thiết bị có cùng một phương pháp I/O thì được kết nối theo thứ tự như thế nào.
26. Vẽ chuỗi ngắt và giải thích cơ chế điều khiển ngắt của các thiết bị theo thứ tự chuỗi.
27. Những trường hợp trạng thái nào của hệ thống máy tính có thể gây ra ngắt không che được.
28. Địa chỉ của chương trình xử lý ngắt được tính như thế nào.
29. Sự khác nhau giữa hệ thống video kiểu bit map và character-map.
30. Vì sao chúng ta sử dụng các cổng bên trong (hơn là dùng các lời gọi BIOS và DOS) để truy nhập đến 8253.
31. Khi chúng ta tạo bản nhạc trên máy tính, ưu điểm gì nếu sử dụng 8253.
32. Sử dụng phương pháp bầu chọn, liệt kê các bước mà 8251 thực hiện để gửi đi một byte (chú ý minh họa thêm bằng lệnh máy).

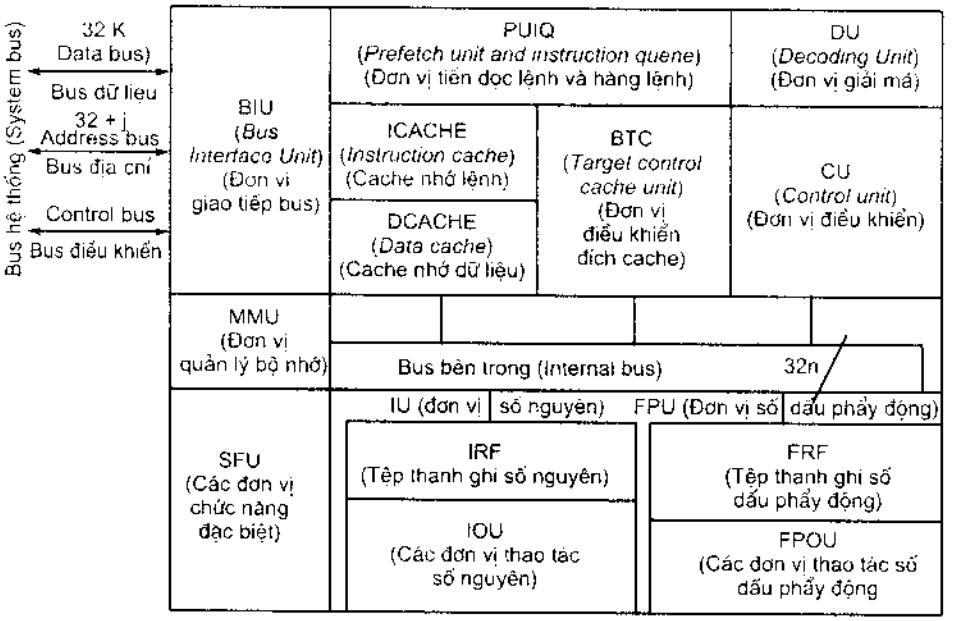
Chương 7
GIỚI THIỆU CÁC BỘ VI XỬ LÝ CÔNG NGHỆ TIÊN TIẾN

7.1. SƠ ĐỒ KHỐI CẤU TRÚC CHUNG CỦA CÁC BỘ VI XỬ LÝ CÔNG NGHỆ TIÊN TIẾN

Các bộ vi xử lý công nghệ tiên tiến được gọi chung cho các bộ vi xử lý được sản xuất vào giữa những năm 1990 của nhiều nhà chế tạo hàng đầu thế giới như Intel, Motorola, Sun, IBM, DEC, AMD... Cấu trúc chung có nghĩa là không thể hiện cụ thể bất kỳ một bộ vi xử lý nào. Nó chứa đựng các đặc điểm của đa số các loại vi xử lý hiện nay (hình 7.1).

Đặc điểm chung của các loại vi xử lý tiên tiến là đa số chúng là các loại vi xử lý 32-bit. Phần lớn các bộ vi xử lý hiện nay có bus dữ liệu bên trong và ngoài 32-bit hay 64-bit (internal/external bus dữ liệu). Trong các bộ vi xử lý 32-bit đơn vị số nguyên (IU) và các thanh ghi trong FPR là 32-bit. Bus dữ liệu có thể là 32-bit. Một số họ vi xử lý có thể có bus dữ liệu với 32x2 (64), 32x3 (96), hoặc 32x4 (128) bit. Trong các bộ vi xử lý 64-bit IU và các thanh ghi trong FPR là 64-bit, bus dữ liệu thường là 64-bit, nhưng có thể là 64x2 (128) hoặc 64x4 (256) bit. Do vậy trong sơ đồ khối ghi giá trị nhân với bus dữ liệu bên trong/bên ngoài là số nguyên lần (32k, 32n). Bus địa chỉ trong hầu hết các họ 32-bit có thể đánh địa chỉ tới 4 GB (2^{32}). Một số họ có thể có khoảng đánh địa chỉ rộng hơn, do vậy trong sơ đồ khối có ghi cho bus địa chỉ là 32+j (j là số nguyên). Cần phải ghi nhớ rằng 32n-bit bus bên trong có phân biệt hai phần: phần địa chỉ và phần dữ liệu. Bus bên trong địa chỉ phân biệt ra hai phần cho các họ vi xử lý có bộ nhớ cache

kép bên trong (dual internal cache memory): phần nhớ lệnh (Icache) và phần nhớ dữ liệu (Dcache).



Hình 7.1: Sơ đồ khối bộ vi xử lý công nghệ cao

7.1.1. Đơn vị giao tiếp bus (BIU)

Đây là bộ đệm giữa các khối bên trong và các hệ thống bên ngoài. Nó được kết nối với bus hệ thống. BIU phân chia ra làm 3 phần chính:

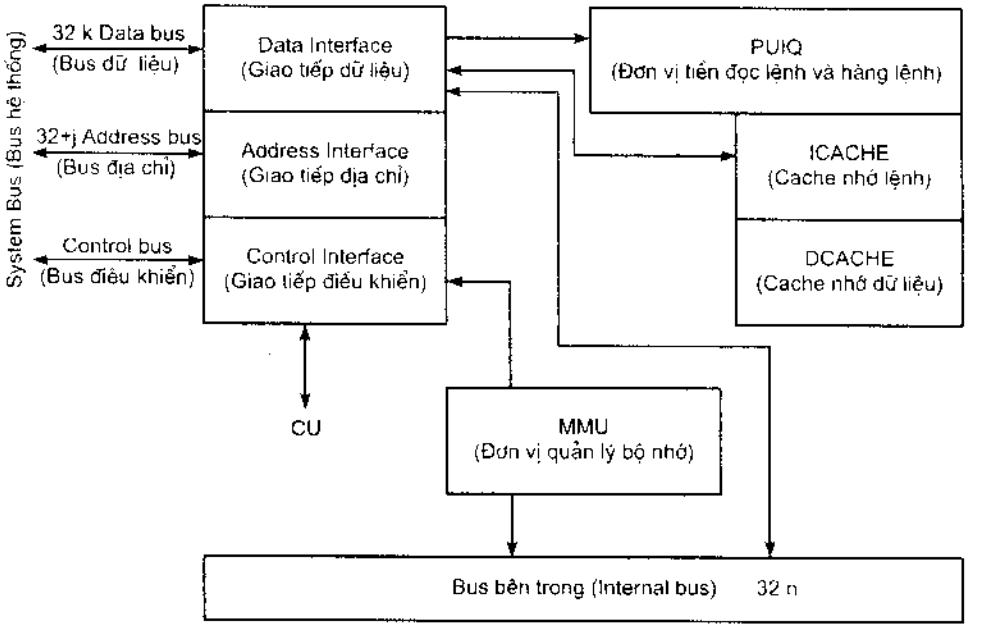
- Giao tiếp dữ liệu (Data Interface);

- Giao tiếp địa chỉ (Address Interface);

- Giao tiếp điều khiển (Control Interface);

- Sơ đồ chi tiết của BIU thể hiện ở hình 7.2.

Bình thường sự trao đổi dữ liệu giữa giao tiếp dữ liệu với các khối bên trong bộ vi xử lý được thực hiện qua bus bên trong, nhưng cũng có các đường liên kết khác như trong hình 7.2 chỉ ra. Thường thì khối giao tiếp dữ liệu có kết nối trực tiếp với PUIQ và các cache.



Hình 7.2: Đơn vị giao tiếp bus (BIU)

Giao tiếp địa chỉ gửi các địa chỉ của lệnh và của dữ liệu được tạo ra bên trong bộ vi xử lý nhờ đơn vị quản lý bộ nhớ MMU ra bus địa chỉ của hệ thống. MMU kết nối trực tiếp với giao tiếp địa chỉ.

Giao tiếp điều khiển (Control interface) gửi và nhận một số các tín hiệu điều khiển và trạng thái từ bộ vi xử lý đến các hệ thống bên ngoài và từ các hệ thống bên ngoài vào bộ vi xử lý. Hầu hết các đường dây điều khiển của giao tiếp điều khiển đều kết nối với đơn vị điều khiển CU và CU có các kết nối với các khối khác trong bộ vi xử lý. Các điều khiển từ ngoài vào là các tín hiệu trạng thái hệ thống bên ngoài, các trạng thái lỗi, các tín hiệu gây ngắt...

7.1.2 Khối tiền đọc lệnh và hàng lệnh PUIQ

Khối này chứa các mạch logic để đọc trước các lệnh từ Icache và đặt lệnh vào hàng xếp các lệnh theo nguyên tắc vào trước-ra trước FIFO (First-In First-Out). Một hàng lệnh (IQ) đặc trưng có từ 8 byte đến 32 byte. Các lệnh xếp hàng được chuyển đến khối giải mã lệnh

DU (Decoding Unit). Phần lớn các họ vi xử lý ngày nay đều cho phép một số lệnh đồng thời được giải mã (xử lý song song).

7.1.3 Khối chức năng đặc biệt SFU

Một số bộ vi xử lý có đơn vị chức năng đặc biệt SFU (Special Function Unit) bổ sung cho IU và FPU. Một SFU có thể là một trong những đơn vị sau đây:

Đơn vị đồ họa (Graphics unit), đơn vị xử lý tín hiệu (Signal processing unit), đơn vị xử lý ảnh (Image processing unit), đơn vị xử lý ma trận và véc-tơ (Vector and matrix processor).

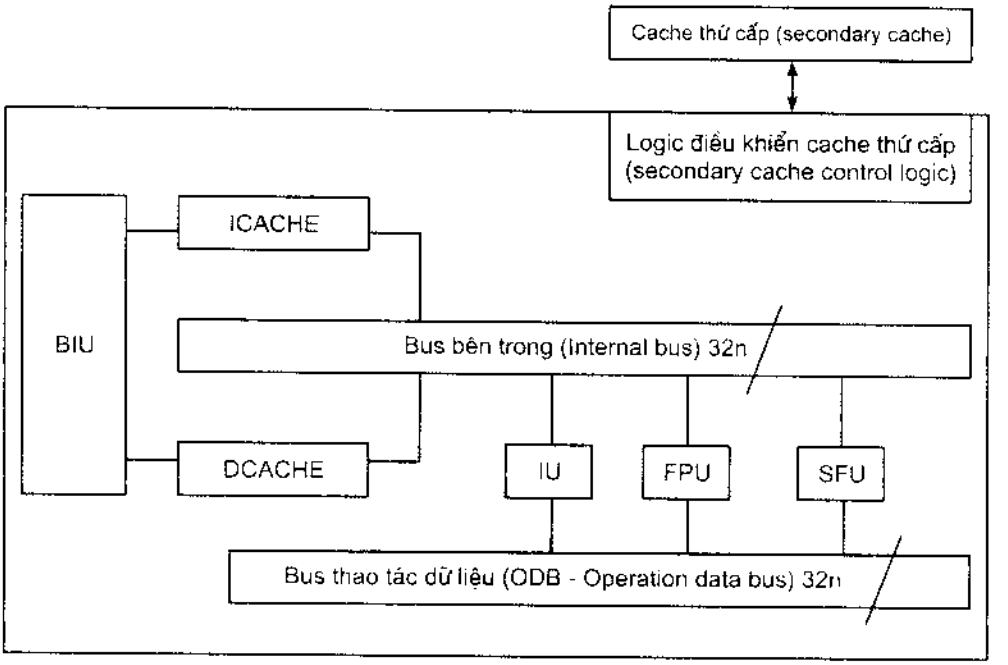
Một bộ vi xử lý có thể có một vài SFU. Việc cấy các SFU vào bên trong chip bộ vi xử lý giảm tối thiểu thời gian trễ do giao tiếp với SFU và như vậy làm tăng tốc độ thực hiện các phép xử lý trên đây.

7.1.4 Bộ nhớ cache

Cache là bộ nhớ trung gian tốc độ nhanh nằm giữa bộ vi xử lý và bộ nhớ chính (main memory). Nó được coi như là bộ nhớ đệm của bộ nhớ chính. Sự tồn tại bộ nhớ cache với kích thước đủ lớn và tối ưu làm tăng hiệu suất của bộ vi xử lý vì nó cho phép bộ vi xử lý truy nhập thông tin nhanh hơn nhiều so với truy nhập vào bộ nhớ chính.

Kỹ thuật cache đã bắt đầu áp dụng trong các chip Intel 80386. Ngày nay một số họ vi xử lý đã cấy bên trong cache dung lượng ≥ 32 kB. Để tăng hiệu suất, các cache thường phân ra cache chứa các lệnh (Icache) bộ nhớ cache lưu dữ liệu (Dcache). Chúng nhận thông tin từ bộ nhớ chính thông qua bus dữ liệu và BIU (dữ liệu 16-bit và 32-bit). Icache kết nối trực tiếp với PUIQ. Nó chuyển một hay một số lệnh vào PUIQ trong một chu kỳ. Cả hai loại Icache và Dcache đều kết nối với bus trong. Thông qua bus dữ liệu bên trong của Dcache và bus dữ liệu điều hành (ODB) giao tiếp với các đơn vị chức năng khác như mô tả ở hình 7.3. ODB thường có độ rộng 128-bit hay 256-bit đủ để tải một số toán hạng cùng một lúc. Hệ thống nhớ của hệ thống vi xử lý được tổ chức với 5 cấp: cấp 1 (L0) là tệp các thanh ghi bên trong chip vi xử lý;

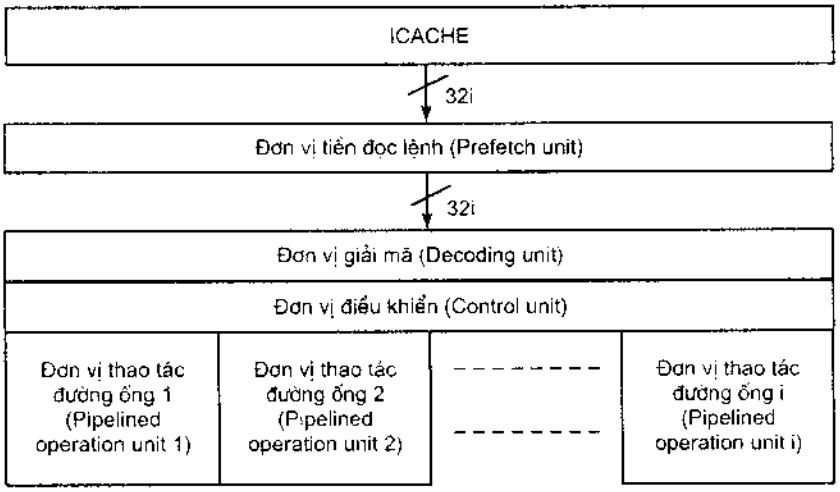
cấp hai (L1) là cache sơ cấp (primary cache) nằm bên trong chip; cấp 3 là cache thứ cấp (secondary cache) nằm giữa cache sơ cấp và bộ nhớ chính (main memory). Các cache thứ cấp kết nối trực tiếp với vi xử lý không thông qua system bus như bộ nhớ chính và chúng được thực hiện bằng công nghệ nhớ tĩnh nên có tốc độ truy nhập nhanh hơn so với truy nhập bộ nhớ chính và có thể có dung lượng lớn hơn cache sơ cấp. Trong bộ vi xử lý có cấy phần điều khiển và giao tiếp logic với cache thứ cấp bên ngoài. Dung lượng của cache thứ cấp có thể lên đến vài MB. Loại vi xử lý Pentium III cho phép mở rộng cache ngoài tới 4 MB.



Hình 7.3: Kết nối CACHES trong vi xử lý

Trong các chip vi xử lý có kiến trúc siêu hướng (superscalar), liên kết của ICACHE cho thực hiện các lệnh song song được thể hiện trong hình 7.4. Khối tiền đọc lệnh (prefetch unit) gửi ra i lệnh cùng một thời điểm tới khối giải mã. Khối điều khiển lần lượt tạo ra các chỉ thị cho một số ống thực hiện. Số lượng lệnh là i không nhất thiết phải

bằng số ống, tuy nhiên nếu số i lệnh bằng số ống sẽ là hiệu quả nhất, vì như vậy cả i lệnh cùng thực hiện song song.



Hình 7.4: Xử lý lệnh trong kiến trúc siêu hướng

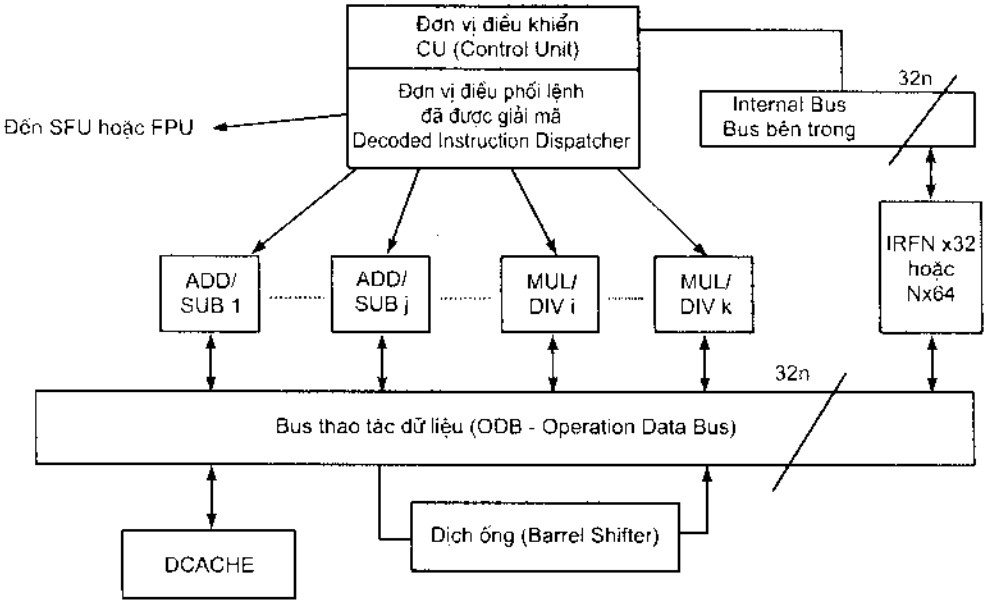
Tuy vậy, khi có lệnh rẽ nhánh, các lệnh tiếp sau lệnh rẽ nhánh phải được lấy ra khỏi đường ống lệnh, và lệnh đích rẽ nhánh (phải rẽ nhánh đến để thực hiện) cần phải được đẩy vào đường ống lệnh. Nếu các lệnh đích rẽ nhánh phải đọc từ bộ nhớ, thì có thời gian trễ thêm trong trình tự thực hiện lệnh. Trong một số chip vi xử lý công nghệ cao có cấy thêm cache đích rẽ nhánh BTC (Branche Target cache). Khi đó, các lệnh đích đầu tiên mà rẽ nhánh phải thực hiện được đẩy vào BTC, như thế sẽ làm giảm thời gian trễ do đọc các lệnh đích rẽ nhánh từ bộ nhớ. Trong một số chip vi xử lý, BTC chỉ chứa địa chỉ của các lệnh đích rẽ nhánh mà không chứa các lệnh đích rẽ nhánh. Sự hiện diện của BTC làm tăng hiệu suất của vi xử lý lên.

7.1.5 Đơn vị điều khiển (CU)

Đơn vị điều khiển CU của bộ vi xử lý có thể là hoàn toàn cứng hoặc được lập trình. Trong hầu hết các bộ vi xử lý công nghệ CISC (Intel X86 và Motorola) CU thường được lập trình. Với xu hướng phát triển công nghệ RISC, CU được hoàn toàn cứng hóa nhằm tăng tốc độ

thực hiện hầu hết các lệnh trong một chu kỳ đơn. Hầu hết các đường tín hiệu điều khiển của giao tiếp điều khiển (control interface) đều kết nối với CU. Các tín hiệu tạo ra từ đơn vị giải mã lệnh được chuyển về CU. Ngoài ra CU cũng có kết nối với nhiều khối chức năng khác. Đơn vị giải mã lệnh.

7.1.6 Đơn vị nguyên (IU)



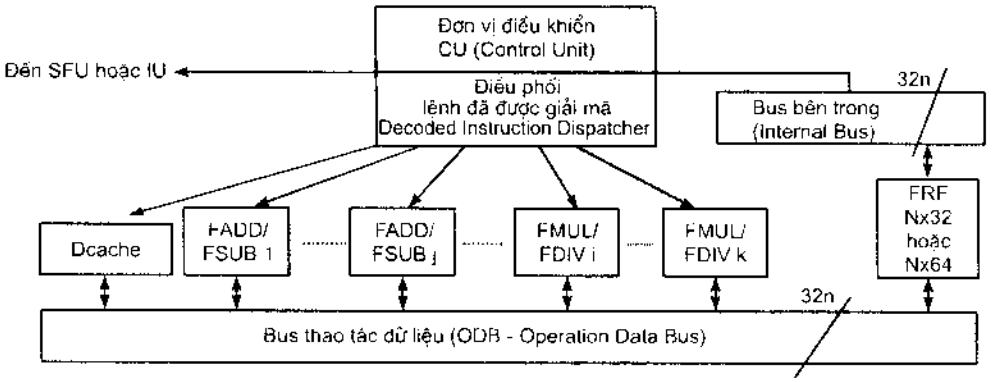
Hình 7.5: Đơn vị nguyên (IU)

IU của bộ vi xử lý được mô tả ở hình 7.5 có đặc tính cấu trúc phục vụ cho xử lý song song. Nó có các đơn vị thao tác thực hiện các phép số học nguyên cộng/trừ và nhân/chia, có tệp các thanh ghi (IRF) 32-bit hoặc 64-bit (phụ thuộc vào các bộ vi xử lý). Các bộ vi xử lý CISC thường có từ 8 đến 16 thanh ghi, còn vi xử lý RISC có tới 32 thanh ghi, đôi khi có tới hơn 100 thanh ghi. Dòng dữ liệu đi theo hai đường để tới các đơn vị thao tác các phép tính số học.

Đơn vị điều phối lệnh đã được giải mã (Decoded instruction dispatcher) nhận các chỉ thị đã được giải mã của CU và gửi chúng tới các đơn vị xử lý các phép tính thích ứng. Các phép tính với số nguyên

chuyển tới các đơn vị xử lý các số nguyên của IU, các phép tính dấu phẩy động chuyển tới đơn vị xử lý dấu phẩy động FPU, các lệnh đặc biệt thì chuyển tới đơn vị các chức năng đặc biệt SFU. Dữ liệu từ Dcache thông qua Bus thao tác dữ liệu ODB (Operation Data Bus) chuyển tới các đơn vị xử lý. Bộ dịch ống (barrel shifter) cho phép thực hiện các lệnh dịch nhiều bit nhanh trong một chu kỳ đơn.

7.1.7. Đơn vị dấu phẩy động FPU



Hình 7.6: Đơn vị dấu phẩy động FPU

Đơn vị điều phối lệnh đã được giải mã nhận các lệnh đã được giải mã của CU và gửi chúng tới các đơn vị xử lý các phép tính thích ứng. Các phép tính với số nguyên chuyển tới các đơn vị xử lý các số nguyên của IU, các phép tính dấu phẩy động chuyển tới đơn vị xử lý dấu phẩy động FPU, các lệnh đặc biệt thì chuyển tới đơn vị các chức năng đặc biệt SFU. Dữ liệu từ Dcache thông qua Bus thao tác dữ liệu ODB (Operation Bus dữ liệu) chuyển tới các đơn vị xử lý. Đơn vị dịch ống (barrel shifter) cho phép thực hiện các lệnh dịch nhiều bit nhanh trong một chu kỳ đơn.

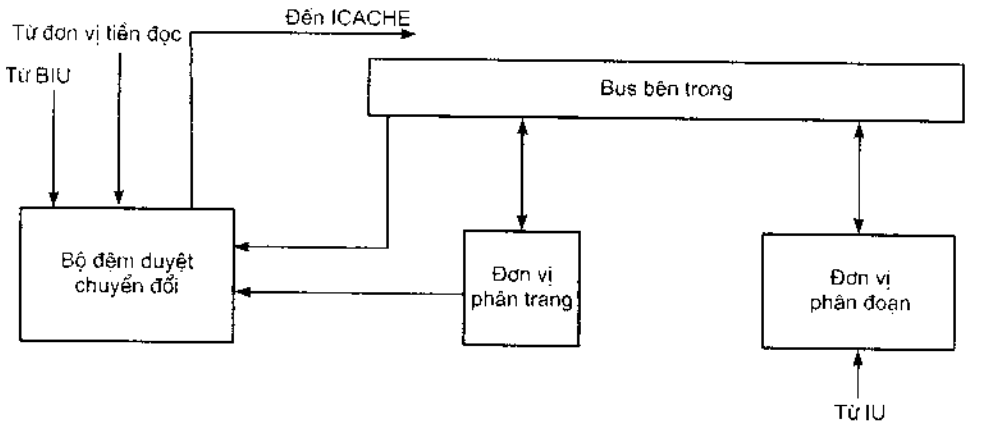
FPU cũng có cấu trúc tương tự như IU (hình 7.6). Các tệp các thanh ghi FRF có N thanh ghi. Trong các họ vi xử lý CISC thường N = 8, trong khi ở các loại vi xử lý RISC có thể N = 32. Hầu hết các thiết kế FPU đều theo tiêu chuẩn dấu phẩy động của Viện IEEE

754-1985 (Institute of Electrical and Electronic Engineers). Tiêu chuẩn này bao gồm quy định các số dấu phẩy động chính xác với 32-bit và 64-bit và số chính xác mở rộng tới 80-bit. Chuẩn IEEE 754-1985 đối với số dấu phẩy động tương đương với chuẩn của DEC Alpha và VAX.

7.1.8 Đơn vị quản lý bộ nhớ (MMU)

MMU (Memory Management Unit) của bộ VXL có các chức năng như sau:

- Chuyển đổi địa chỉ ảo hay địa chỉ logic thành địa chỉ vật lý, tức là địa chỉ thực. Địa chỉ vật lý được chuyển tới cache, hoặc thông qua BIU và Bus địa chỉ tới bộ nhớ bên ngoài bộ vi xử lý để chọn vùng nhớ.
- Đảm bảo cơ chế phân trang trong tổ chức bộ nhớ ảo;
- Đảm bảo cơ chế phân đoạn (segment) cho bộ nhớ;
- Bảo vệ bộ nhớ cho cả hai cơ chế phân trang và phân đoạn;
- Quản lý bộ đệm duyệt chuyển đổi TLB (Translation Lookaside Buffer), hay cache chuyển đổi địa chỉ ATC (Address Translation cache) để chuyển đổi địa chỉ ảo thành địa chỉ vật lý.



Hình 7.7: Đơn vị quản lý bộ nhớ (MMU)

Sơ đồ khối của MMU cho trong hình 7.7. Hầu hết tất cả các chip vi xử lý công nghệ cao có đơn vị chức năng phân trang (paging unit)

và TLB (hoặc ATC). Chỉ trong các chip X86 có thêm đơn vị phân đoạn (segmentation unit).

MMU có logic thực hiện giám sát các truy nhập đến các thư mục trang và bảng tương ứng trong bộ nhớ chính, và nạp vào TLB số hiệu trang nếu trước đó trang không tìm thấy trong TLB (chưa nạp số hiệu trang).

7.2 CÁC ĐẶC ĐIỂM KIẾN TRÚC CỦA VI XỬ LÝ CÔNG NGHỆ TIÊN TIẾN

7.2.1 Khái niệm về xử lý song song

Trong chương 2, có nêu những đặc điểm nâng cao tốc độ của bộ vi xử lý, trong đó công nghệ xử lý song song với các kiến trúc đường ống (pipeline), kiến trúc siêu hướng (superscalar) và đa xử lý (multiprocessor) được trang bị cho các loại máy tính, các loại vi xử lý. Khi một nhiệm vụ T (task) được xử lý tuần tự theo từ, bit theo bit trong bộ xử lý hay máy tính, thì bộ xử lý hay máy tính đó gọi là bộ xử lý hay máy tính tuần tự (sequential) (hay uniprocessor). Nếu nhiệm vụ T được phân chia thành những phần con có kích thước xấp xỉ nhau: T_1, T_2,...,T_n, và tất cả các thành phần con này được xử lý đồng thời (thực hiện song song) thì tốc độ thực hiện nhiệm vụ T nhanh gấp n lần và bộ xử lý (máy tính) có chức năng thực hiện như vậy gọi là bộ xử lý song song (máy tính song song).

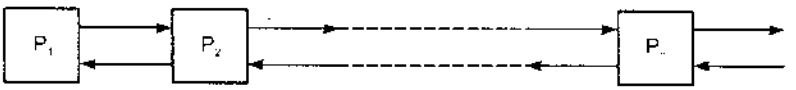
Ví dụ, một nhiệm vụ T là tính tổng (SUM) của N số (hằng số): b_1, b_2,..., b_N. Thuật toán tính tổng N hằng số có thể đơn giản là:

SUM <--- 0;

for i =1 to N do SUM <--- SUM + b[i];

Nếu thuật toán này được thực hiện tuần tự trong máy tính tuần tự thì phải có N phép cộng (add), mỗi phép cộng chiếm một quãng thời gian là T_{add}, nghĩa là để có được kết quả cuối cùng của SUM cần phải mất khoảng thời gian là NT_{add}.

Giả sử ta có một máy tính với N bộ xử lý giống nhau P_i (i = 1, 2,..., N) và chúng kết nối với nhau trong một cấu trúc mảng tuyến tính hai chiều: mỗi một bộ xử lý đồng thời tiếp nhận dữ liệu (receive) và gửi dữ liệu với bộ xử lý lân cận (send) (hình 7.8).



Hình 7.8: Mảng tuyến tính hai chiều gồm n bộ xử lý

Bằng mô hình máy tính kiểu này có thể giải bài toán tính tổng SUM một cách song song như sau: cho N = kn, trong đó: k-một số nguyên; n-số lượng bộ xử lý. Mỗi bộ xử lý xử lý k số và tất cả n bộ xử lý đồng thời thi hành nhiệm vụ. N = nk phép cộng được phân ra n phép cộng (n bộ xử lý), mỗi phép cộng với k số. Bộ xử lý P_i chuyển kết quả phép cộng của nó cho bộ xử lý tiếp thep P_{i+1}, và P_{i+1} cộng kết quả của chính nó với kết quả từ P_i gửi vào. Kết quả phép cộng của P_{i+1} được chuyển tới P_{i+2}, như vậy sau n-1 phép cộng và chuyển dữ liệu, kết quả cuối cùng được cất trong P_n. Chương trình thể hiện quá trình xử lý song song phép tính SUM được biểu diễn bằng đoạn chương trình sau đây:

```
{Mỗi một bộ xử lý Pi tính SUM với các số cục bộ của nó b(1:k)}
    SUM <--- 0;
    for i =1 to k do SUM <--- SUM + b(i);
{Bộ xử lý P1 gửi kết quả phép cộng cục bộ tới P2}
    if INDEX = 1 then
    begin
    if n > 1 then send (RIGHT, SUM); {RIGHT chỉ bộ xử lý
    Pi+1}
    end else
    {Mỗi một bộ xử lý còn lại Pi chờ đợi nhận một kết quả từ Pi-1 }
    begin
```

```
receive(LEFT, LEFTSUM);
SUM <-- SUM + LEFTSUM;
```

{Mỗi một bộ xử lý P_i ngoại trừ P_n gửi kết quả phép cộng cục bộ tới P_{i+1}}

```
if INDEX < n then send (RIGHT, SUM);
end;
```

Thời gian T(n) cần thiết để thực hiện tính tổng SUM theo thuật toán song song trên n bộ xử lý gồm có hai thành phần chính: thời gian tính cục bộ T_L để thực hiện k = N/n phép cộng đồng thời trên n bộ xử lý. Có thể viết $T_L = K_1N/n$, trong đó, K_1 là một hằng số phụ thuộc vào thời gian cộng và lưu giữ trên một bộ xử lý. Thành phần thứ hai của T(n) là thời gian TC (thời gian truyền thông), gửi dữ liệu trung gian qua n-1 phép cộng. Có thể viết $T_C = K_2(n-1)$, trong đó: K_2 là hằng số phụ thuộc vào độ trễ truyền thông giữa các bộ xử lý. Như vậy, thời gian T(n) sẽ là:

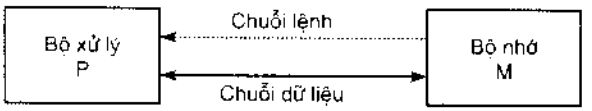
$$T(n) = T_L + T_C = \frac{K_1N}{n} + K_2(n-1) \qquad (7.1)$$

Bởi vì K_2 liên quan đến thời gian truyền dữ liệu (kết quả trung gian) giữa các bộ xử lý, nó lớn hơn nhiều so với K_1, nên sự tăng số lượng các bộ xử lý (n) lên sẽ kéo theo tăng thời gian K_2. Nếu n = 1 thì công thức (7.1) trở thành:

$$T(1) = T_L = K_1N \qquad (7.2)$$

7.2.2 Các kiểu của bộ xử lý song song

Có nhiều cách phân loại các bộ xử lý song song dựa trên cấu trúc hoặc hành vi của chúng. Những phương pháp phân loại chính thường dựa trên số lượng các lệnh và/hoặc số lượng các toán hạng có thể được xử lý đồng thời, tổ chức bên trong của các bộ xử lý, cấu trúc kết nối các bộ xử lý, hoặc các phương pháp điều khiển các dòng lệnh và dữ liệu bên trong hệ thống các bộ xử lý.



Hình 7.9: Trao đổi chuỗi lệnh, chuỗi dữ liệu trong một máy tính đơn giản

Đơn vị xử lý trung tâm (CPU) đọc các lệnh (fetching instructions) và các toán hạng từ bộ nhớ, thực hiện các lệnh (executing instructions), và chuyển kết quả vào lại bộ nhớ chính (main memory). Các bước thực hiện lệnh này gộp lại thành một chu kỳ lệnh (instruction cycle). Các lệnh có thể hình thành một chuỗi các lệnh liên tiếp nhau (instruction stream) đọc từ bộ nhớ vào bộ xử lý, trong khi các toán hạng cũng hợp thành một chuỗi dữ liệu (data stream) theo sau đi tới và từ bộ xử lý. Sự trao đổi các dòng lệnh và các dòng dữ liệu giữa bộ xử lý và bộ nhớ được mô tả đơn giản trong hình 7.9.

a) Phân loại theo Flynn

Nếu đặt M_I và M_D là số lượng các chuỗi lệnh và các chuỗi dữ liệu tương ứng, thì theo Michael J. Flynn, các máy tính được phân loại thành 4 nhóm dựa trên số lượng M_I và M_D được CPU xử lý như sau:

(1) Một chuỗi lệnh một chuỗi dữ liệu SISD (Single Instruction Stream Single Data stream)

$M_I = M_D = 7$. Hầu hết các máy tính và máy vi tính sử dụng với một chip vi xử lý các loại là những máy tính kiểu SISD (Single Instruction stream Single Data stream)

(2) Một chuỗi lệnh nhiều chuỗi dữ liệu SIMD (Single Instruction stream Multiple Data stream)

$M_I = 1$, $M_D > 7$. Máy tính ILLIAC IV, bộ xử lý máy tính DAP (Distributed Array Processor) của International Computers Ltd, thuộc kiểu SIMD. Trong các hệ thống máy tính SIMD có nhiều bộ xử lý làm việc song song với nhau, thực hiện một mệnh lệnh giống nhau nhưng với những dữ liệu khác nhau. Những dạng máy tính chuyên dụng cho

xử lý véc-tơ và mảng (array) là những máy tính SIMD (Single Instruction stream Multiple Data stream).

(3) Nhiều chuỗi lệnh một chuỗi dữ liệu MISD

$M_I > 1$, $M_D = 7$. Không có nhiều bộ xử lý song song phù hợp với nhóm này. Có thể các máy tính như Cray-1 và CYBER 205 của Control Data Corporation có tính năng xử lý đường ống (pipeline processing) được liệt vào nhóm MISD. Trong các máy tính MISD có nhiều bộ xử lý làm việc song song và xử lý chức năng riêng (theo những mệnh lệnh khác nhau) nhưng với dữ liệu giống nhau.

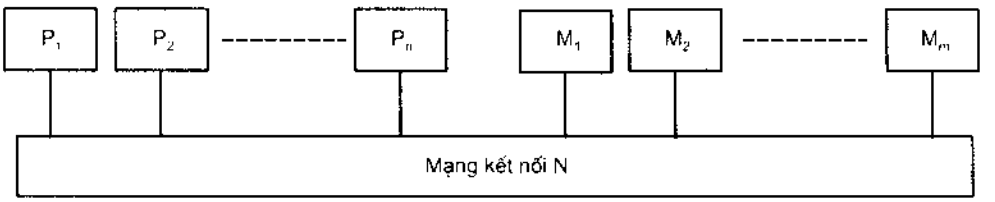
(4) Nhiều chuỗi lệnh nhiều chuỗi dữ liệu MIMD

$M_I > 1$, $M_D > 7$. Thuộc vào nhóm này là các máy tính có kiến trúc đa xử lý, một số bộ vi xử lý làm việc song song độc lập với nhau, thi hành theo các mệnh lệnh khác nhau với các dòng dữ liệu khác nhau.

b) Phân loại theo cấu trúc

(1) Phân loại máy tính

Một hệ thống máy tính có thể được xem như một tập hợp $n \geq 1$ bộ xử lý hay đơn vị xử lý trung tâm (CPU): P_1, P_2,..., P_n và $m \geq 0$ đơn vị nhớ chia sẻ (shared memory unit) hoặc mô-đun nhớ: M_1, M_2,..., M_m kết nối trong mạng N bên trong của hệ thống máy tính như chỉ ra trong hình 7.10.



Hình 7.10: Hệ thống máy tính với n ≥ 1 đơn vị xử lý và m ≥ 0 đơn vị nhớ

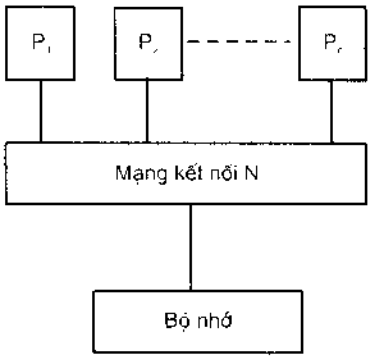
Trong hệ thống máy tính tuần tự, n = m = 1 và N chỉ là một bus đơn (single bus). Ví dụ: các máy vi tính có một CPU và một bus hệ thống mà trên đó các thiết bị ngoại vi, bộ nhớ kết nối.

• *Máy tính chia sẻ bộ nhớ (shared-memory computer)*

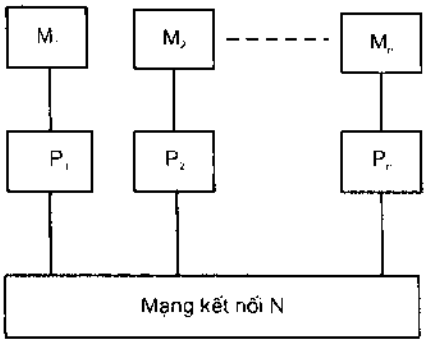
Hệ thống máy tính với tổ chức, mà trong đó các đơn vị nhớ kết nối trên mạng N làm thành một bộ nhớ chính tổng thể đảm bảo chia sẻ cho tất cả các đơn vị xử lý (hình 7.11a), thì gọi là máy tính có bộ nhớ chia sẻ.

• *Máy tính phân tán bộ nhớ (distributed-memory computer)*

Hệ thống máy tính, mà trong đó mỗi đơn vị xử lý nối với một đơn vị nhớ tạo thành một khối xử lý riêng (với tài nguyên riêng) và kết nối với nhau trên mạng kết nối N để trao đổi tài nguyên, thì gọi là máy tính có phân phối tài nguyên (hình 7.11b).



Hình 7.11a: Máy tính chia sẻ bộ nhớ

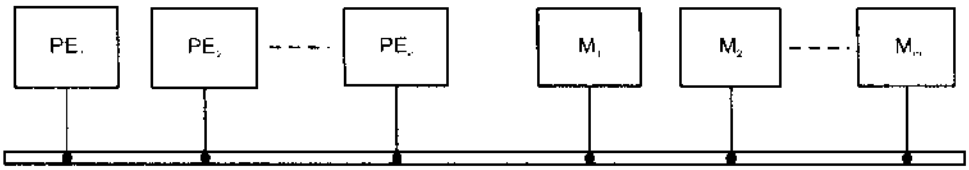


Hình 7.11b: Máy tính bộ nhớ phân tán

(2) Phân loại mạng kết nối N

Cấu trúc kết nối mạng N cũng được phân loại. Nó có thể có những dạng như sau:

• *Bus chia sẻ đơn* (single shared bus) (hình 7.12)

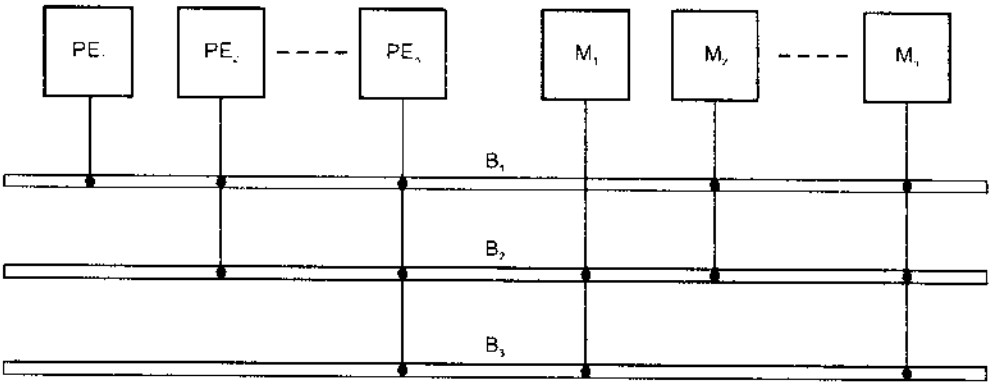


Hình 7.12: Bus đơn (single bus)

Kiểu bus đơn này được sử dụng nhiều trong các hệ thống máy tính tuần tự cũng như máy tính song song, bởi sự đơn giản trong kết nối và điều khiển. Nhưng một nhược điểm lớn là khi số lượng n (các đơn vị xử lý) và m (số lượng đơn vị nhớ) tăng lên thì tốc độ của Bus đơn phải rất nhanh, điều này sẽ dẫn tới nhiều sự đụng độ cạnh tranh truy nhập bus là tăng thời gian chờ đợi được phục vụ trên bus đối với các đơn vị xử lý và đơn vị nhớ.

• *Nhiều Bus (multi-bus)*

Mạng nhiều bus khắc phục nhược điểm của bus đơn, trong đó, một số đơn vị xử lý và đơn vị nhớ được kết nối với một bus, những đơn vị xử lý và đơn vị nhớ khác lại kết nối với một bus khác, hoặc có những đơn vị xử lý và đơn vị nhớ có thể kết nối cùng một lúc với một số bus, như vậy sẽ giảm quá tải cho các bus, sự đụng độ tranh truy nhập bus giảm tối thiểu. Nhưng lại có một nhược điểm ở lại mạng kết nối này là, khi có sự cố xảy ra đối với một bus nào đó, thì hiệu suất mạng giảm đi rõ rệt cũng như gia tăng lỗi.

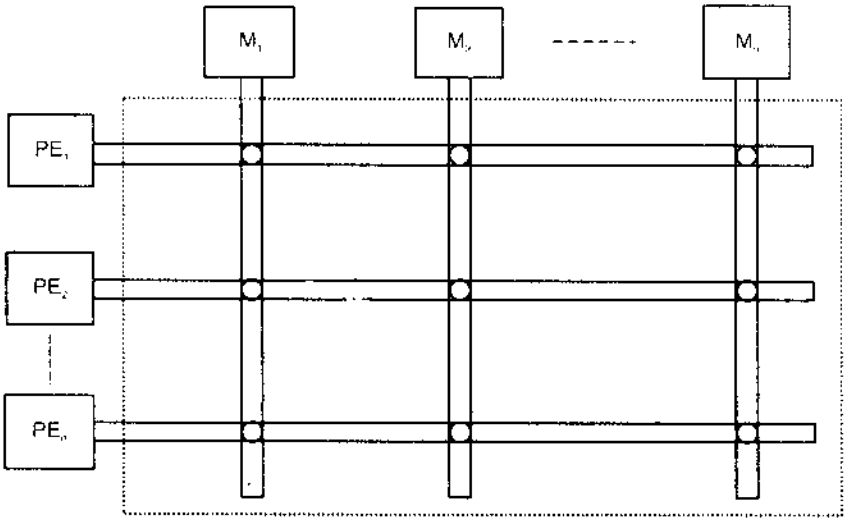


Hình 7.13: Nhiều bus (multiple buses)

• *Bus giao nhau* (crossbar buses) (hình 7.14)

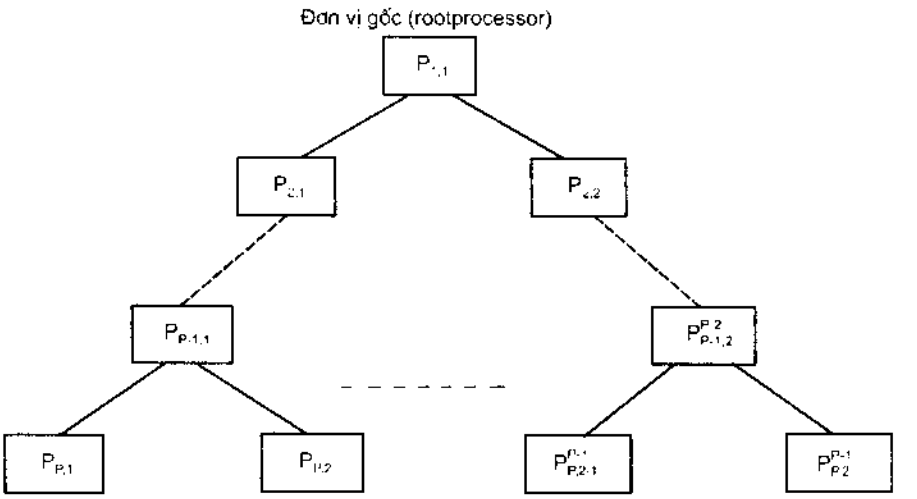
Trong cấu trúc kết nối bus giao nhau này, mỗi một đơn vị xử lý kết nối với tất cả các đơn vị nhớ và tương tự, mỗi một đơn vị nhớ kết nối với tất cả các đơn vị xử lý. Như vậy ta có một kết nối kiểu ma trận hai chiều n x m. Cấu trúc này khắc phục nhược điểm của cấu trúc

nhiều bus. Nhưng có trường hợp xấu nhất có thể xảy ra là: nếu tất cả các đơn vị xử lý cùng truy nhập vào một đơn vị nhớ. Loại kết nối này đã được áp dụng một số hệ thống máy tính lớn.



Hình 7.14: Bus giao nhau (crossbar buses)

• *Kết nối hình cây* (Tree) (hình 7.15)



Hình 7.15: Mạng kết nối hình cây (tree network)

Mạng kết nối hình cây này còn gọi là cây nhị phân, tức là với gốc (các đơn vị xử lý mẹ) chỉ có hai đơn vị xử lý con nối vào. Như vậy nếu có số lượng hàng (tầng cây) là P thì sẽ có tổng số các đơn vị xử lý là $n = 2^P-1$, tức là số tầng cây P xấp xỉ bằng $P = \log_2 n$. Các đơn vị xử lý trao đổi dữ liệu với nhau thông qua các lệnh vào/ra: gửi (send) và nhận (receive). Dạng kết nối hình cây không phức tạp nhưng có tốc độ trao đổi chậm, khi có sự cố xảy ra ở các đơn vị xử lý mẹ thì sẽ làm mất đi liên hệ với các đơn vị xử lý con, dẫn tới sự loại bỏ nhiều đơn vị xử lý trong nhánh. Tuy vậy, dạng cấu trúc kết nối này vẫn được sử dụng ở một số hệ thống máy tính. Giống như một số loại đa xử lý, máy tính có cấu trúc kết nối bên trong hình cây phù hợp với một số bài toán xử lý song song, ví dụ: quay lại với bài toán tính SUM đã đề cập.

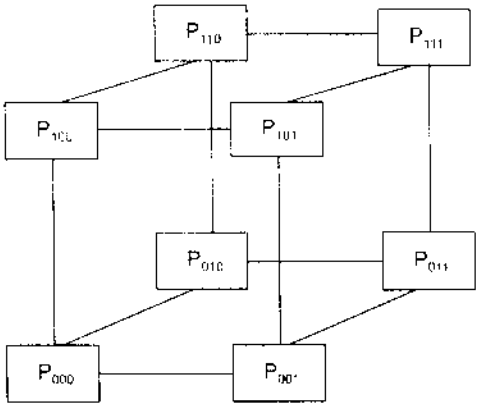
$$SUM = b_1 + b_2 + b_3 + ... + b_N, \text{ trong đó: } N = 2^P - 7.$$

Quá trình giải phép tính SUM này như sau:

Nạp tất cả các toán hạng $b_1, b_2, b_3, ..., b_N$ vào $2^P - 1$ đơn vị xử lý lá (leaf processors) của cây nhị phân (tầng P). Sau đó mỗi một cặp toán hạng b_j, b_{j+1} chuyển từ các đơn vị xử lý tầng P vào đơn vị xử lý $P_{P-1,j}$, ở tầng P-1 để tính tổng cục bộ: $y_j = b_j + b_{j+1}$ và kết quả này nằm lại trong $P_{P-1,j}$. Như vậy sẽ giảm số lượng toán hạng đi một nửa, tất cả kết quả cục bộ này nằm ở trong các đơn vị xử lý ở tầng P-7. Từ tầng P-1, tất cả N/2 toán hạng được đẩy vào các đơn vị xử lý ở tầng P-2. Tại đây, ta có các kết quả cộng cục bộ, và số lượng toán hạng giảm xuống còn N/4. Cuối cùng đạt tới đơn vị xử lý gốc $P_{1,1}$, ta nhận được kết quả của phép cộng SUM. Như vậy, để đạt được kết quả cuối cùng SUM cần phải có P-1 hay $\log_2 N$ lần cộng.

• *Mạng kết nối dạng siêu lập thể n chiều (n-dimensional supercube)*

Dạng kết nối này còn gọi là kết nối lập phương-n nhị phân (binary n-cube), vì nếu có n chiều thì có 2^n đơn vị xử lý kết nối trong khối lập lập phương. Nếu n = 3 thì có 8 đơn vị xử lý (hình 7.16).

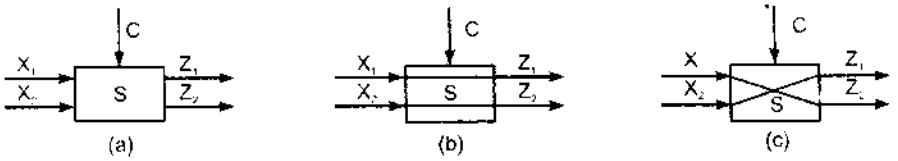


Hình 7.16: Kết nối mạng kiểu lập thể n chiều (3-cube)

Kiến trúc kết nối lập thể bên trong các hệ thống máy tính và các bộ xử lý lớn được đề xuất từ đầu những năm 60, nhưng mãi cho tới khoảng năm 1985 mới bắt đầu có những ứng dụng thực tế.

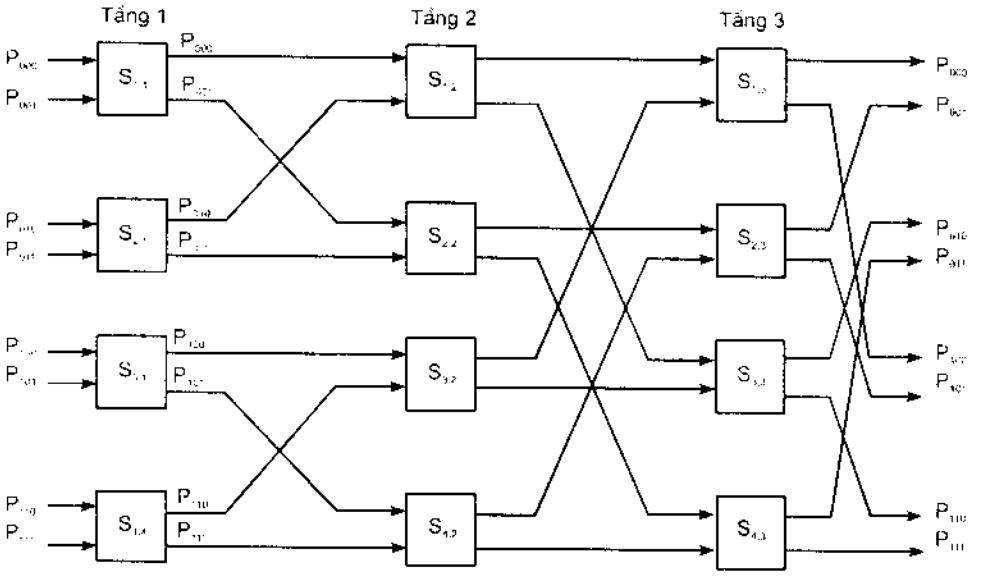
Còn có nhiều kiểu dạng kết nối mạng bên trong khác nữa, nhưng dạng cấu trúc trên đây thường được áp dụng để thiết kế các hệ thống máy tính hơn cả. Các cấu trúc kết nối bên trong dạng đơn bus, nhiều bus, kết nối giao nhau là những ví dụ về cấu trúc kết nối động (dynamic), trong khi cấu trúc kết nối dạng cây, dạng lập phương là cấu trúc tĩnh (cố định).

Một dạng cấu trúc kết nối động bên trong có thể được xây dựng bằng sử dụng các thành phần chuyển mạch 2-trạng thái (hình 7.17). Mỗi một thành phần chuyển mạch có 2 đường vào và 2 đường ra để kết nối với các thành phần chuyển mạch lân cận. Tất cả 4 đường (buses) giống nhau và có thể làm việc như là các liên kết processor-processor hay procesor-memory.



Hình 7.17: (a): Thành phần chuyển mạch
(b): Trạng thái thông suốt (trực tiếp), (c): Trạng thái giao nhau

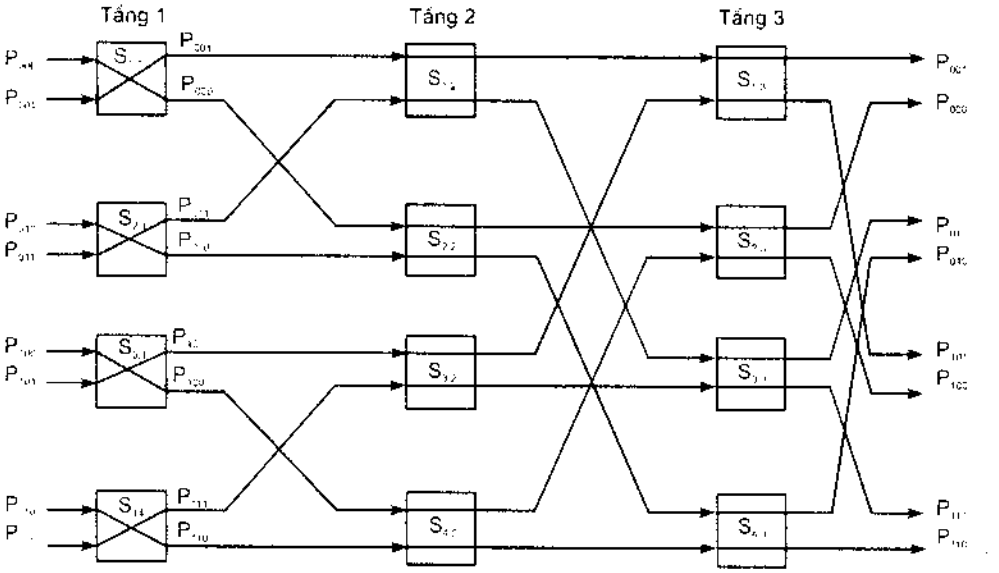
Trạng thái của thành phần chuyển mạch được điều khiển bởi tín hiệu C, nó có hai trạng thái: trạng thái đi qua (through) hay trực tiếp (direct), trong đó: $Z_1 = X_1$, $Z_2 = X_2$ và trạng thái giao nhau (cross), trong đó: $Z_1 = X_2$, $Z_2 = X_7$. Như vậy, với các thành phần chuyển mạch hai trạng thái này có thể thiết kế mạng kết nối bên trong của hệ thống máy tính xử lý song song hay đa xử lý nhiều dạng khác nhau. Hình 7.18 là một ví dụ cấu trúc kết nối mạng thực tế dùng các thành phần chuyển mạch S hai đầu vào và hai đầu ra. Mạng có 12 thành phần chuyển mạch S sắp xếp theo 3 tầng (cột), và dùng để kết nối động giữa 8 thành phần xử lý P_0:P_7. Bằng các cách thiết lập các tín hiệu điều khiển cho các thành phần chuyển mạch S, ta có thể có một số lượng lớn các mẫu kết nối mạng (mẫu kết nối động). Số lượng mẫu kết nối động phụ thuộc vào số tầng (cột) kết nối và trạng thái của các thành phần chuyển mạch.



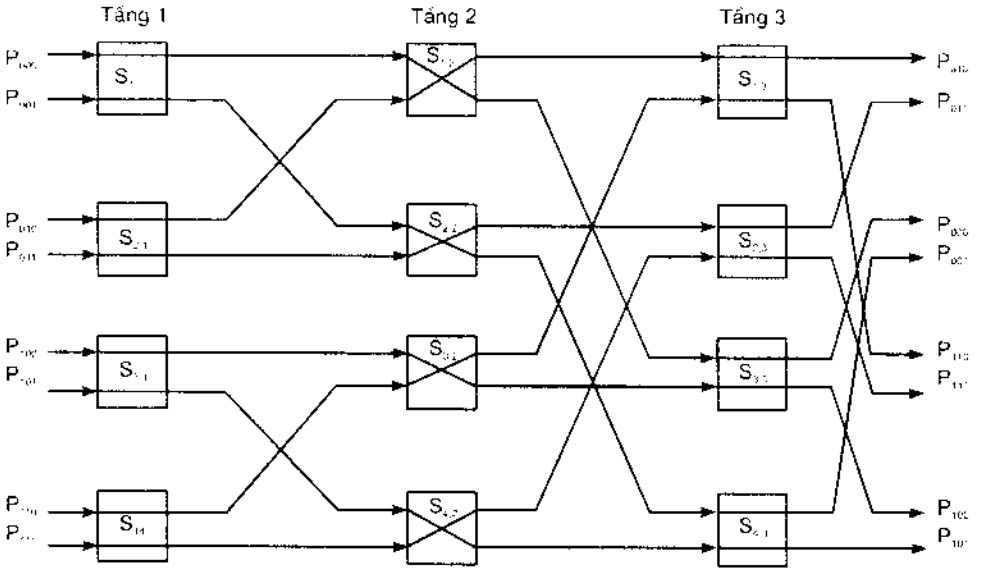
Hình 7.18: Mạng chuyển mạch 3 tầng

Hình 7.19a là một trong nhiều khả năng kết nối mạng phụ thuộc vào sự thiết lập trạng thái của thành phần chuyển mạch. Mạng ở hình 7.19a có 4 thành phần chuyển mạch ở tầng 1 được thiết lập trạng thái

giao nhau (cross), còn lại các thành phần chuyển mạch ở các tầng 2, 3 đều được thiết lập về trạng thái thông suốt.



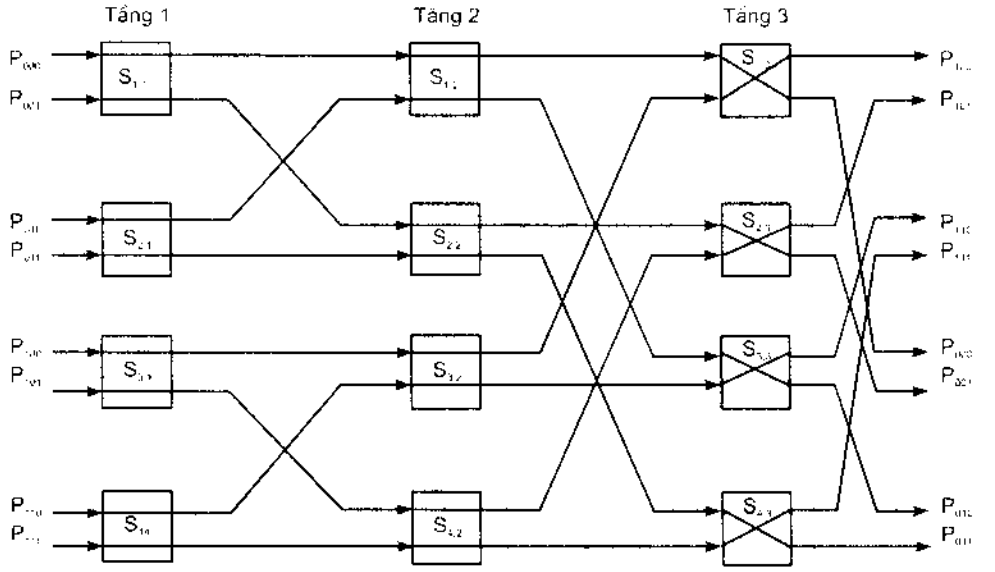
Hình 7.19a: Tầng 1 gồm các thành phần chuyển mạch giao nhau



Hình 7.19b: Tầng 2 gồm các thành phần chuyển mạch giao nhau

Trong hình 7.19b, các thành phần chuyển mạch của tầng 2 ($S_{1,2}$, $S_{2,2}$, $S_{3,2}$, $S_{4,2}$) được thiết lập trạng thái chéo, còn các thành phần chuyển mạch của các tầng 1, 3 đều được thiết lập ở trạng thái thông suốt. Chúng ta có thể thấy rằng đầu ra của mạng kết nối với P_{000} được dẫn từ thành phần xử lý P_{010}, còn đầu ra P_{001} lại kết nối với thành phần xử lý P_{017}.

Trong hình 7.19c, chỉ có tầng 3 là thiết lập ở trạng thái đấu chéo, hai tầng còn lại ở trong trạng thái thông suốt. Như vậy, số lượng cột càng nhiều thì số tổ hợp kết nối càng lớn, nhưng độ trễ truyền thông trong mạng càng cao. So sánh với kết nối tĩnh thì sử dụng các thành phần chuyển mạch để xây dựng các mẫu kết nối động cho các thành phần xử lý bên trong hệ thống máy tính có ưu việt hơn bởi tính linh hoạt và nhiều khả năng chọn lựa tổ chức kết nối hệ thống máy tính đa xử lý.



Hình 7.19c: Tầng 3 gồm các thành phần chuyển mạch giao nhau

7.2.3 Hiệu suất của xử lý song song

Hiệu suất của bộ xử lý song song phụ thuộc vào sự phức tạp của kiến trúc hệ thống và mức độ song song của chương trình thực hiện. Kiến trúc kết nối của các bộ nhớ cũng ảnh hưởng đến hiệu suất mạng.

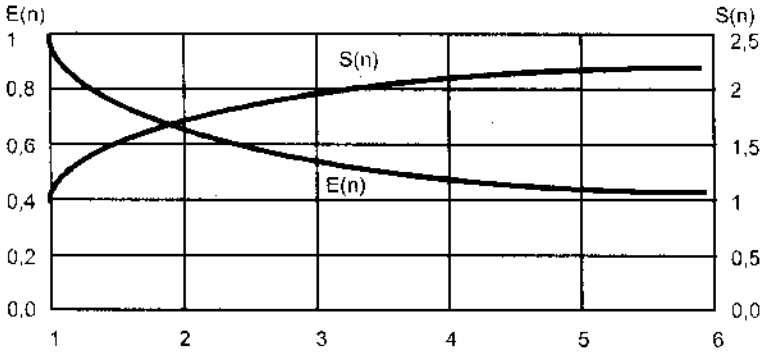
Một trong những thông số xác định mức độ xử lý song song là số đo tăng tốc S(n) (speedup), là tỷ số của thời gian T(1) thực hiện trên máy tính tuần tự và thời gian thực hiện T(n) trên máy tính song song với n bộ xử lý đối với một số dạng lệnh nào đó:

$$S(n) = \frac{T(1)}{T(n)} \qquad (7.1)$$

Cho rằng $nT(n) \geq T(1)$ thì $n \geq S(n)$. Biểu diễn S(n) thông qua các giá trị K_1, K_2 số lượng các đơn vị xử lý (n) ta có:

$$S(n) = \frac{K_1 N}{\frac{K_1 N}{n} + K_2(n-1)} \approx \frac{n}{1 + \frac{Kn}{k}} \qquad (7.2)$$

Trong đó: $K = K_2/K_1$ và $k = N/n$ là một số nguyên. Giá trị n là số lượng các đơn vị xử lý và thể hiện mức độ song song của hệ thống.



Hình 7.20: Hiệu suất và độ tăng tốc thực hiện một bài toán nào đó

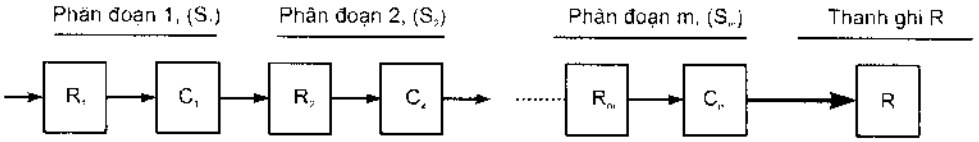
Hiệu suất của xử lý song song E(n) là tỷ số giữa số đo tăng tốc S(n) và n:

$$E(n) = \frac{S(n)}{n} \qquad (7.3)$$

Đôi khi gọi E(n) là độ sử dụng của bộ xử lý (processor utilization). Quan hệ của E(n) và S(n) thể hiện trong hình 7.20.

7.2.4. Các bộ xử lý cấu trúc đường ống

Kỹ thuật đường ống (pipeline) là một trong những đặc điểm thường sử dụng trong xử lý song song. Một phép toán được chia thành k phép toán nhỏ, và một hệ thống k phân đoạn thực hiện tính toán k phép toán nhỏ này lần lượt. Như vậy một đường ống với k phân đoạn xử lý được hình thành. Nếu liên tục gửi dữ liệu vào đường ống thì cũng giống như một dây chuyền sản xuất quá trình xử lý phép toán đều đồng thời xảy ra trên mọi phân đoạn công đoạn của ống. Mỗi phân đoạn đều cho ra một kết quả, và cuối đường ống là kết quả cuối cùng của phép toán.



Hình 7.21: Cấu trúc của bộ xử lý ống

Bộ xử lý cấu trúc ống gồm có một chuỗi liên tiếp các mạch xử lý mà các mạch xử lý đó thường được gọi là các phân đoạn (segment) hay tầng (stage) (hình 7.21). Thông qua chuỗi này dòng toán hạng được xử lý. Mỗi phân đoạn xử lý từng phần các toán hạng và kết quả cuối cùng chỉ nhận được khi toàn bộ chuỗi các toán hạng đã đi qua hết các phân đoạn của ống.

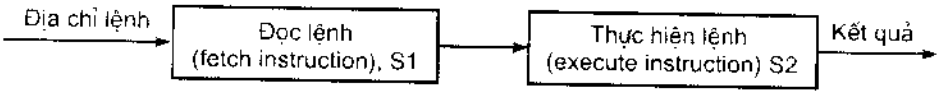
Trong cấu trúc ống ở hình 7.21, các thanh ghi R_i là những bộ đệm nhận dữ liệu D_{i-1} từ phân đoạn S_{i-1}, ngoại trừ thanh ghi R_1 nhận dữ liệu từ bên ngoài ống. Các dữ liệu D_{i-1} là các kết quả tính được trong chu kỳ nhịp đồng hồ trước của các đơn vị tính C_{i-7}. Khi D_{i-1} đã được nạp vào R_i, thì R_i tách logic khỏi đầu vào, và C_i sử dụng D_{i-1} để tính ra kết quả mới D_i. Như vậy, trong mỗi chu kỳ nhịp đồng hồ mỗi một phân đoạn chuyển dữ liệu cũ của nó tới phân đoạn sau và cũng tính kết quả mới nhờ dữ liệu nhận từ phân đoạn trước.

Có hai loại đường ống cơ bản:

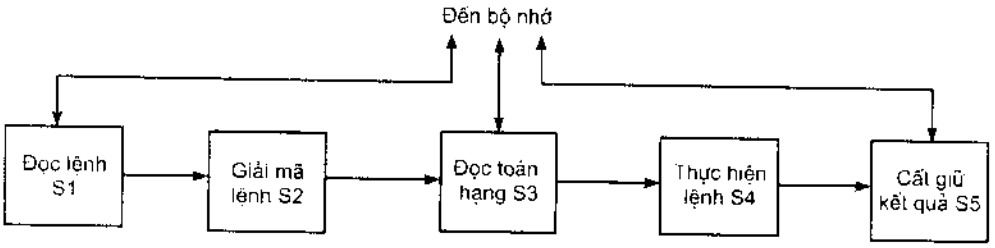
1. Đường ống lệnh (instruction pipeline): trong đó, các phân đoạn khác nhau của đọc lệnh và thực hiện lệnh được thực hiện trong đường ống.

2. Đường ống số học (arithmetic pipeline): trong đó, các phân đoạn khác nhau của phép toán số học được thực hiện trong đường ống.

Một đường ống lệnh đơn giản có thể được tổ chức gồm hai phân đoạn: phân đoạn đọc lệnh (S1), phân đoạn thực hiện lệnh (S2) (hình 7.22).



Hình 7.22: Đường ống lệnh 2 phân đoạn



Hình 7.23: Đường ống lệnh 5 phân đoạn

Một đường ống lệnh cơ bản có thể gồm 5 đoạn và hình thành nhờ phân theo chu kỳ lệnh: đọc lệnh (S1), giải mã lệnh (S2), đọc các toán hạng (S3), thực hiện lệnh (S4), cất giữ kết quả (S5) (hình 7.23).

Để đơn giản, ta xét phân chia đường ống lệnh phổ biến thành 4 phân đoạn:

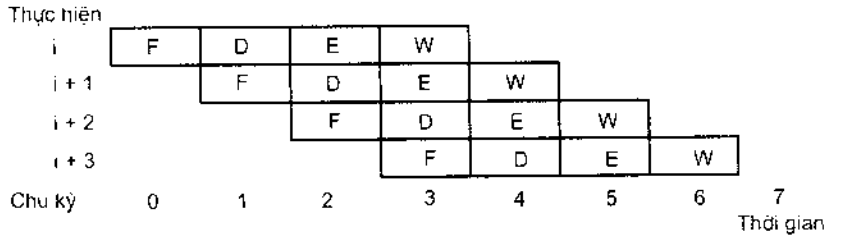
1. F(S1)-fetche (đọc lệnh): Lệnh được đọc từ bộ nhớ vào CPU.

2. D(S2-S3)-decode (giải mã lệnh): Lệnh được giải mã bên trong CPU và tệp thanh ghi được tham chiếu để dùng cho các toán hạng. Phân đoạn này là gộp của S2 và S3 (đọc toán hạng) của đường ống 5 phân đoạn (hình 7.11).

3. E(S4) - execute (thực hiện lệnh): Lệnh được thực hiện

4. W(S5)- - write (ghi kết quả): Kết quả thực hiện lệnh được ghi trở lại bộ nhớ.

Với sự phân chia này một đường ống lệnh được biểu diễn bằng đồ thị thời gian thực hiện theo từng phân đoạn sẽ đơn giản (hình 7.24). Với sự đi vào liên tục của dữ liệu ta có sự liên tục ghi kết quả ở cuối đường ống lệnh. Mỗi một phân đoạn của đường ống lệnh được biểu diễn là i.



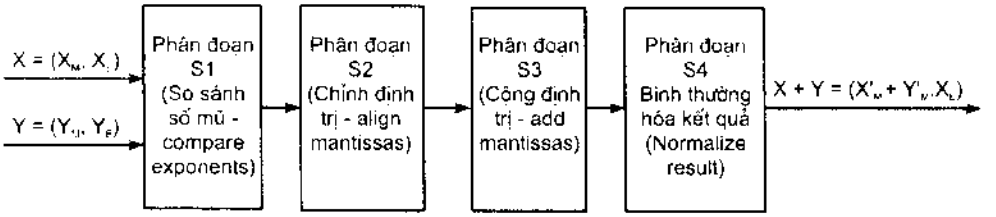
Hình 7.24: Đồ thị đường ống lệnh 4 phân đoạn

Cho rằng mỗi một công đoạn xử lý trong mỗi phân đoạn được hoàn thành trong một chu kỳ (cycle). Nếu một lệnh có hơn một từ mà bus dữ liệu chỉ có độ rộng một từ thì quá trình đọc lệnh kéo dài hơn một từ. Chỉ trong các hệ thống mà tất cả các lệnh có độ dài nhỏ hơn hoặc bằng độ rộng của bus dữ liệu thì các đoạn đọc lệnh chỉ kéo dài trong một chu kỳ. Những lệnh như vậy có thể có trong hệ thống RISC (ví dụ, Intel 860). Công đoạn W có thể tính đến nếu kết quả ghi vào một trong những thanh ghi của CPU. Ví dụ, đối với các lệnh thao tác thanh ghi-đến-thanh ghi. Nếu ghi ra bộ nhớ (ngoài CPU) thì công đoạn này có thể kéo dài hơn một chu kỳ.

Có một vấn đề đáng kể trong kỹ thuật đường ống, đó là, nếu truy nhập tới bộ nhớ để đọc một lệnh (giai đoạn F) và đồng thời lại có truy nhập bộ nhớ để ghi kết quả thực hiện lệnh (giai đoạn W) thì có thể xảy ra xung đột. Nếu một lệnh phụ thuộc vào kết quả của lệnh trước nó, như sử dụng toán hạng đích của lệnh trước, thì cũng lại là một vấn đề phải tính đến. Nếu có lệnh rẽ nhánh hoặc nhảy thì xảy ra ngắt dòng thực hiện lệnh trong đường ống, và chuyển qua dòng lệnh khác. Như

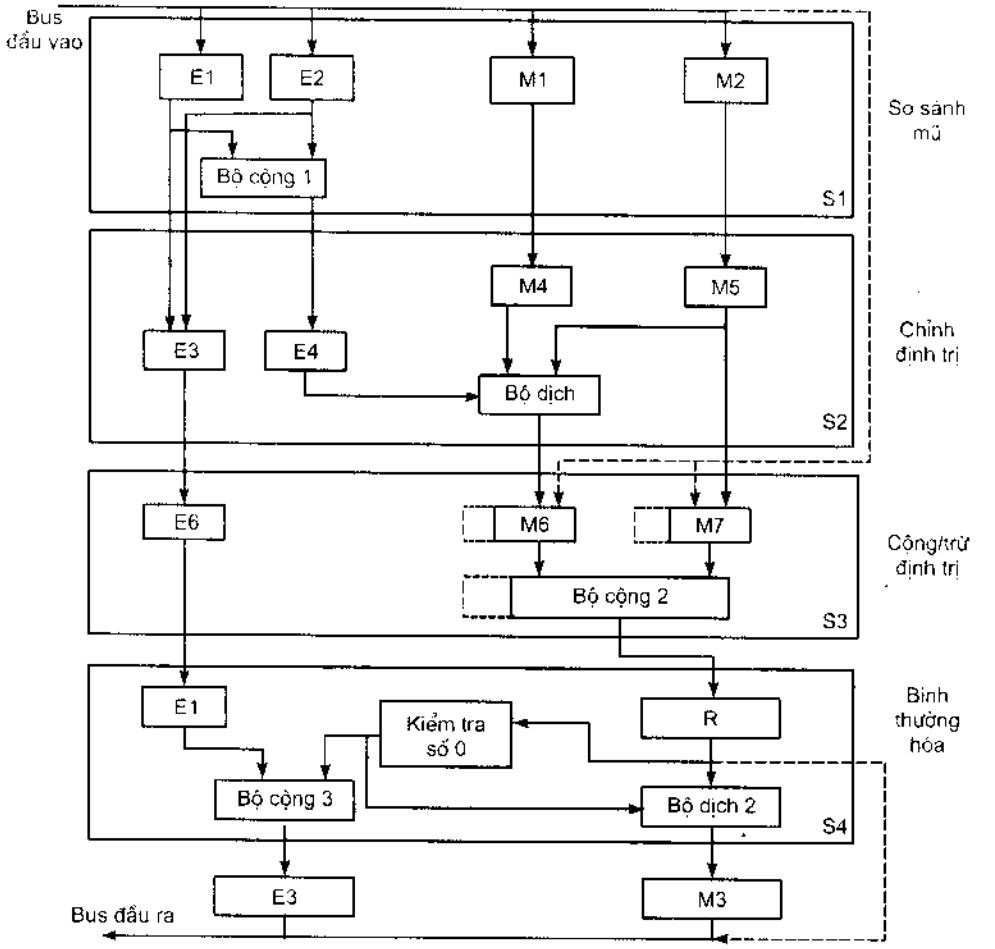
thế không có sự thực hiện hoàn toàn trôi chảy trong đường ống. Vấn đề này gọi là các chướng ngại (pipiline hazards), và chúng cần được tối thiểu hóa. Chúng ta sẽ xét vấn đề này sau.

Trong thực tế, nhiều nhóm lệnh số học phức tạp có thể được kiến trúc thực hiện theo ống, ví dụ: nếu ta cộng hai số dấu phẩy động $x = (x_M, x_E)$ và $y = (y_M, y_E)$, trong đó: x_M, y_M là phần định trị (mantissa), y_E, y_E là số mũ với dạng $B = 2^K$. Khi đó ta có một đường ống để thực hiện phép cộng x và y như cho trong hình 7.25.



Hình 7.25: Đường ống 4 phân đoạn cộng 2 số dấu phẩy động

Hình 7.26 là một thiết kế ống bộ cộng dấu phẩy động với 4 phân đoạn: S1 - so sánh số mũ (exponent comparison); S2 - chỉnh định trị (mantissa alignment); S3 - cộng/ trừ định trị và S4 - bình thường hóa kết quả (normalization result). Phân đoạn S4 phân chia ra làm hai phân đoạn riêng: phân đoạn tính k số lượng số 0 (hoặc 1) đứng trước trong số định trị chưa chuẩn hóa, và phân đoạn thứ hai để thực hiện dịch k lần để chuẩn hóa phần định trị của số dấu phẩy động. Thiết kế ống 4 phân đoạn này cho cộng các số dấu phẩy động nhưng cũng là bộ một phân đoạn cho các số dấu phảy cố định (fixed-point adder). Phép trừ cũng có thể được thực hiện bằng bộ cộng này. Bộ cộng 5 phân đoạn có thể được xây dựng từ bộ cộng 4 phân đoạn nếu đoạn S4 (bình thường hóa) phân thành hai phân đoạn: một phân đoạn để tính số lượng (k) các số 0 đứng trước (hoặc các số 1) trong phần định trị chưa được bình thường hóa, và phân đoạn thứ hai thì thực hiện dịch k lần để thực hiện bình thường hóa phần định trị.



Hình 7.26: Bộ cộng dấu phẩy động 4 đường ống

Cho rằng một hàm F cần phải được thực hiện hoặc bằng bộ xử lý đường ống hoặc không có đường ống. F được phân tích thành thành m bước độc lập F_1, F_2,... F_i,..., F_m. Cho rằng F_i phải được một mạch logic C_i thực hiện với thời gian trễ lan truyền (thời gian thực hiện F_i tại C_i) là T_i. Trong bộ xử lý ống, đối với tất cả các đoạn thời gian T_i là xấp xỉ như nhau, do đó cho rằng $T_i = T_C$ đối với tất cả các i. Một bộ xử lý không có đường ống P_1 thực hiện hàm F với tổng thời gian trễ là mT_C, tương đương với băng rộng là $1/(mT_C)$. Bộ xử lý đường ống với m đoạn xử lý (P_m) (hình 7.14) thực hiện hàm F với thời gian trễ T_R ở từng đoạn

do đệm tại thanh ghi R_i và các mạch logic liên quan ở từng đoạn. Do đó tổng thời gian trễ tại từng đoạn là $T = T_C + T_R$, với băng rộng là $1/(T_C + T_R)$. Có thể kết luận rằng P_M có băng rộng lớn hơn P_1, nghĩa là bộ xử lý đường ống tăng hiệu suất thực hiện nếu điều kiện sau đây thỏa mãn:

$$mT_C > T_C + T_R$$

Bây giờ ta xét tới các phép tính ma trận, chúng là một dạng bài toán được thực hiện nhanh trong các bộ xử lý đường ống.

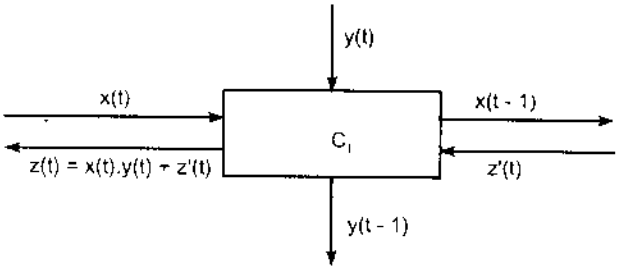
Để tính các ma trận, trong nhiều loại máy tính có trang bị những đường ống chuyên dụng: nhân, chia.

Ví dụ:

X là một ma trận kích thước n x m của các số dấu phảy tĩnh hoặc dấu phẩy động:

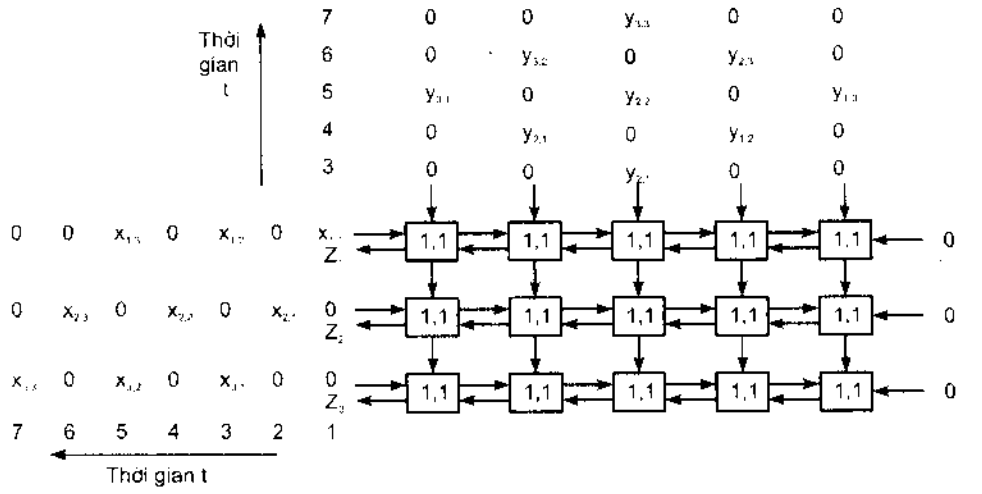
$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \ldots\ldots x_{1,n} \\ x_{2,1} & x_{2,2} \ldots\ldots x_{2,n} \\ \ldots\ldots\ldots\ldots\ldots \\ x_{n,1} & x_{n,2} \ldots\ldots x_{n,n} \end{bmatrix}$$

Khi đó để nhân các ma trận, thiết kế ống được thực hiện đa hướng (scalar) từ những ô (cell), mỗi ô C_{ij} thực hiện phép nhân và cộng: $z(t) = z'(t) + x(t) \times y(t)$. Các giá trị đưa x và y vào C_{ij} từ bên trái và bên trên (hình 7.27).



Hình 7.27: Ô tính nhân và cộng hai số X và Y

Các ô tính liên kết trong một ma trận. Giả sử n = 3 thì ma trận có cấu trúc như ở hình 7.28.



Hình 7.28: Ma trận nhân 3 × 5

Toán hạng $x_{1,1}$ chuyển vào ô $C_{1,1}$ từ bên trái, từ bên hàng đầu chỉ có các giá trị 0 của y và z. Đầu ra của ô $C_{1,1}$ có kết quả:

$$z_{1,1} = x_{1,7}.y_{1,1} + x_{1,2}.y_{2,1} + x_{1,3}y_{3,7}.$$

Tại thời điểm t = 3, ở ô $C_{1,3}$ có giá trị $y_{1,1}$ được đưa vào từ trên. $C_{1,3}$ tính ra kết quả:

$$z_{1,3} = x_{1,7}.y_{1,1} + 0$$

Chuyển $z_{1,3}$ tới $C_{1,2}$, đồng thời $C_{1,3}$ chuyển $y_{1,1}$ xuống tầng thứ hai cho ô $C_{2,3}$ của ma trận, cũng đồng thời $C_{1,3}$ chuyển $x_{1,1}$ sang ô $C_{1,4}$ trong chu kỳ t = 4, $x_{2,1}$ và $y_{2,1}$ chuyển vào ô $C_{1,2}$. $C_{1,2}$ tính:

$$z_{1,2} = x_{1,2}.y_{2,1} + z', \text{ trong đó } z' = x_{1,7}.y_{1,7}.$$

Cuối cùng, ở t = 5 ô $C_{1,1}$ tính được:

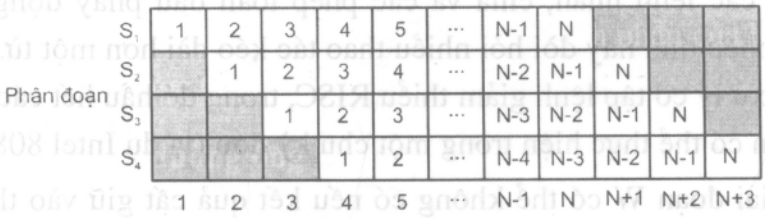
$$z = x_{1,3}.y_{3,1} + z'$$

với $z' = x_{1,7}.y_{1,1} + y_{2,7}.y_{2,1}$ lấy từ $C_{1,2}$.

Giá trị z là $z_{1,7}$. Khi t = 6, ô $c_{1,1}$ đưa ra 0, tại t = 7 ô $C_{1,1}$ đưa ra $z_{1,2}$. Quá trình tiếp diễn cho đến khi nào tất cả các thành phần của hàng đầu tiên của ma trận được tạo ra (hình 7.28).

7.2.5. Điều khiển đường ống và đồ thị không gian thời gian

Mục đích chính của điều khiển đường ống là làm tối đa tốc độ tính toán (speedup), hay tăng hiệu suất sử dụng bộ xử lý. Những thông số này phụ thuộc vào dạng nhiệm vụ mà bộ xử lý phải thực hiện, và thứ tự các nhiệm vụ trong đường ống. Để có thể đánh giá hành vi của đường ống, người ta thường sử dụng đồ thị không gian-thời gian (space-time diagram) đánh giá hiệu quả sử dụng các đoạn trong đường ống như một hàm phụ thuộc thời gian. Đồ thị không gian thời gian là một hình chữ nhật với n × T(N), trong đó: n là số các đoạn của đường ống, T(N) là số chu kỳ nhịp cần phải có để thực hiện một tập hợp với N nhiệm vụ.



Hình 7.29: Đồ thị không gian-thời thực hiện chuỗi N phép cộng trong đường ống 4 đoạn

Hình 7.29 là đồ thị không gian thời gian của chuỗi N phép cộng được thực hiện bằng một đường ống cộng 4 đoạn (S_1, S_2, S_3, S_4). Tất cả các phân đoạn đều được sử dụng đầy đủ ngoại trừ lúc ban đầu và lúc cuối của chuỗi phép cộng. Tỷ số của vùng không bôi đậm (bận) trên toàn bộ vùng (kể cả bôi đậm và không bôi đậm) của đồ thị không gian - thời gian chính là hiệu suất E(n). Trong hình 7.27: tỷ số E(n) = E(4) = N/(N+3). Độ tăng tốc S(n) = nE(n).

7.2.6 Các chướng ngại trong kiến trúc đường ống

Xét đồ thị đường ống 4 phân đoạn (hình 7.24). Sơ đồ này giải thích sự thực hiện theo thời gian 4 lệnh liên tiếp nhau: i, i+1, i+2, và i+3. Quá trình thực hiện các lệnh là liên tục trước lệnh i và sau lệnh i+3, giống như sự liên tục chảy trong đường ống dẫn nước, dẫn dầu, hay dây chuyền lắp ráp máy móc. Nhưng đường ống lệnh trong sơ đồ này được coi là lý tưởng, bởi vì mỗi một giai đoạn chỉ thực hiện trong một chu kỳ. Thực tế, lệnh có thể có độ dài nhiều hơn một từ, và nếu bus dữ liệu lại cũng chỉ có độ dài một từ, thì khi đó, ngay giai đoạn đọc lệnh (F) phải mất hơn một chu kỳ. Chỉ có ở các loại vi xử lý mà tất cả các lệnh của chúng có độ dài không vượt quá độ rộng của bus dữ liệu thì mới định được thời gian đọc lệnh chỉ trong một chu kỳ. Để đạt được một đồ thị thời gian xử lý lệnh trong đường ống lý tưởng như trong hình 7.24 là một khó khăn lớn và không phải lúc nào cũng thực hiện được trong các chip vi xử lý có tập lệnh phức tạp CISC, đặc biệt là đối với các lệnh nhân, chia và các phép toán dấu phẩy động. Bởi vì những phép tính này đòi hỏi nhiều thao tác kéo dài hơn một từ. Chỉ các chip vi xử lý có tập lệnh giảm thiểu RISC, trong đó hầu hết các lệnh độ dài ngắn có thể thực hiện trong một chu kỳ đơn (ví dụ Intel 80860).

Giai đoạn W có thể không có nếu kết quả cất giữ vào thanh ghi bên trong chip vi xử lý. Nếu có W, thì nó kéo dài không chỉ một chu kỳ mà có thể hơn.

Trong các hệ thống RISC, có nhiều thanh ghi, gọi là tệp thanh ghi, do đó chúng đảm bảo các lệnh đều là các thao tác thanh ghi-đến-thanh ghi. Như vậy, không cần có W trong đường ống lệnh.

Có thể có các trường hợp xảy ra làm phá vỡ tiến trình tuần tự thực hiện lệnh trong đường ống lệnh:

1. Nếu xuất hiện cùng một thời điểm yêu cầu truy nhập bộ nhớ để đọc lệnh (F) và truy nhập bộ nhớ để cất giữ kết quả thực hiện lệnh (W) thì sẽ xảy ra đụng độ.

2. Nếu một lệnh đi vào đường ống để thực hiện lại phụ thuộc vào kết quả thực hiện của lệnh trước nó thì chúng ta lại có vấn đề.

3. Nếu lệnh được vào đường ống thực hiện là lệnh rẽ nhánh (branch) hoặc lệnh nhảy, thì phải thực hiện ngắt luồng lệnh trong đường ống để chuyển qua luồng lệnh khác.

Tất cả những trường hợp này thường được gọi là những chướng ngại của đường ống (pipeline hazards).

Phân biệt 3 loại chướng ngại của đường ống: chướng ngại do cấu trúc, chướng ngại do dữ liệu và chướng ngại do điều khiển.

a) Các chướng ngại do cấu trúc (structural hazards)

Chúng xuất hiện do các đụng độ về tài nguyên khi phần cứng không thể hỗ trợ tất cả các tổ hợp các lệnh cùng đồng thời được thực hiện xếp chồng lên nhau trong các giai đoạn khác nhau. Trong đường ống 4 giai đoạn, mỗi giai đoạn cần sử dụng một số tài nguyên như sau:

1. Giai đoạn F: bộ đếm chương trình (PC), thanh ghi địa chỉ bộ nhớ (MAR), bus địa chỉ và dữ liệu (address and bus dữ liệu).

2. Giai đoạn D: bộ giải mã lệnh, bus bên trong (bus trong).

3. Giai đoạn E: ALU, MAR, bus địa chỉ và dữ liệu).

4. Giai đoạn W: bus bên trong.

Xem trong hình 7.24, ta thấy rằng, MAR, bus địa chỉ và dữ liệu cần dùng trong giai đoạn E của lệnh i+1 (có thể để đọc toán hạng từ bộ nhớ trong giai đoạn E), và đồng thời chúng cũng cần dùng trong giai đoạn F của lệnh i+3 (để đọc lệnh i+3). Bus bên trong cần dùng trong giai đoạn D của lệnh i+2 (để chuyển dữ liệu từ tập thanh ghi), và đồng thời nó cũng cần dùng trong giai đoạn W của lệnh i (để cất giữ kết quả vào tệp thanh ghi). Đó là các ví dụ về chướng ngại do cấu trúc.

Để khắc phục chướng ngại do cấu trúc: cần có 2 cache, Icache (cho lệnh) và Dcache (cho dữ liệu), và chúng được truy nhập qua các bus và các thanh ghi địa chỉ bộ nhớ riêng. Khi đó CPU có thể truy nhập Icache ở giai đoạn F để đọc một lệnh, đồng thời truy nhập

Dcache để ghi/đọc dữ liệu trong giai đoạn E của lệnh khác. Những chip vi xử lý hiện nay đều có Icache và Dcache và sử dụng nhiều bus bên trong, cộ tệp thanh ghi nhiều điểm có thể đảm bảo hạn chế được loại chướng ngại này. Có một vấn đề cũng là chướng ngại có thể xảy ra khi cần phải tăng nội dung của bộ đếm lệnh PC sau khi lệnh được đọc ra. Khi đó, nếu định sử dụng ALU thì sẽ có đụng độ với lệnh khác đang trong giai đoạn E (vì giai đoạn E sử dụng ALU). Chướng ngại này có thể được khắc phục bằng cách đưa vào một mạch logic thực hiện tăng trực tiếp nội dung PC trong hầu hết các loại chip vi xử lý hiện nay.

b) Các chướng ngại do dữ liệu (data hazards)

Chúng xuất hiện khi một lệnh thực hiện trong đường ống phụ thuộc vào các kết quả của lệnh thực hiện trước nó theo cách lệnh xuất hiện xếp chồng lên nhau về thời gian trong các giai đoạn khác nhau của đường ống.

Để giải thích loại chướng ngại này, ta xét 2 lệnh số học cộng và trừ thực hiện liên tiếp nhau:

Add r3, r2, r1; (r3) + (r2) → r1, thanh ghi r1 là đích.

Sub r4, r1, r5; (r4) - (r1) → r5, thanh ghi r1 là nguồn.

Nếu xét chúng thực hiện trong đường ống thì sẽ như sau:

Chu kì	**1**	**2**	**3**	**4**	**5**
Cộng	F	D	E	W	
			Giá trị tổng mới tính ở đây	Thanh ghi r1 nhận giá trị mới	
Trừ		F	D	E	W
			Thanh ghi r1 được đọc tại đây		

Chúng ta nhận thấy rằng, trong khi giá trị mới sẽ được cất vào thanh ghi r1 ở chu kỳ 4 (W) của lệnh cộng, thì một nhu cầu đọc nó xảy ra trong chu kỳ 3 (D) của lệnh trừ. Như vậy, lệnh trừ sẽ lấy giá trị cũ của thanh ghi r1 để thực hiện, và kết quả sẽ sai. Đó là sai do dữ liệu, hay chướng ngại do dữ liệu. Để khắc phục loại chướng ngại này, có

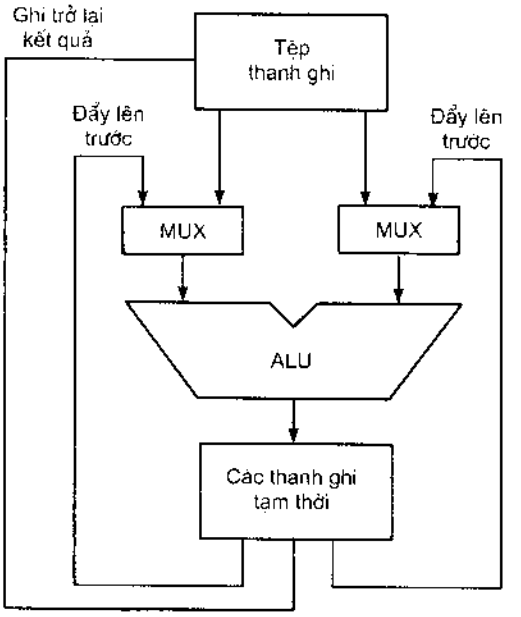
một giải pháp đơn giản là chèn vào 2 chu kỳ ST (stall) tạm dừng xử lý lệnh đủ để giai đoạn D của lệnh trừ (Sub) tiếp sau giai đoạn W của lệnh cộng và nhận được giá trị mới của r7.

Chu ki	1	2	3	4	5	6	7
Cộng	F	D	E	W			
Trừ		F	ST	ST	D	E	W

Có giải pháp khác đòi hỏi phải đầu tư thêm phần cứng trong chip vi xử lý. Giá trị mới của thanh ghi r1 được tính ở chu kỳ 3 (E) của lệnh cộng, rồi sau đó r1 sẽ nhận giá trị này ở chu kỳ 4, giai đoạn W của lệnh cộng và cũng là giai đoạn E của lệnh trừ. Chính trong chu kỳ 3 của lệnh cộng ta cần phải tóm được giá trị mới của r1 và ghi nó vào một thanh ghi đặc biệt của thanh ghi ở lối ra của ALU, rồi chuyển nó trở lại lối vào ALU như là một toán hạng của phép trừ cho giai đoạn W của lệnh trừ. Sau đó r1 nhận giá trị mới ở giai đoạn W của lệnh cộng như bình thường. Lệnh trừ tiếp nhận giá trị mới của r1 ngay lúc bắt đầu giai đoạn E của nó. Tức là kết thúc giai đoạn E của lệnh cộng giá trị mới của r1 mới đảm bảo chắc chắn tạo ra. Kết thúc giai đoạn E của lệnh cộng cũng là kết thúc giai đoạn D của lệnh trừ, vì vậy giá trị mới của r1 tham gia vào tính toán phép trừ ở giai đoạn E của lệnh trừ đòi hỏi phải có hiệu chỉnh kết quả ở giai đoạn này. Phương pháp này gọi là phương pháp đẩy lên trước (forwarding). Hình 7.30 là sơ đồ diễn giải phương pháp đẩy trước thực hiện bằng bổ sung phần cứng.

Không phải bao giờ cũng có thể sử dụng phương pháp đẩy lên trước này để khắc phục loại chướng ngại do dữ liệu. Bởi vì, trường hợp xét ở đây coi quá trình thực hiện lệnh (cộng) đã hoàn toàn xong trong một chu kỳ E và đó là các lệnh chỉ có truy nhập thanh ghi - thanh ghi. Đối với các lệnh phải truy nhập bộ nhớ, thì cần phải có một chu kỳ để tính toán địa chỉ EA, chuyển đổi địa chỉ ảo thành địa chỉ vật lý, sau đó cần một chu kỳ truy nhập Dcache, lấy dữ liệu và cất dữ liệu vào thanh ghi đích. Kết quả của lệnh (ví dụ, lệnh load) không có ngay khi kết

thúc giai đoạn E, và do vậy, phương pháp đẩy trước không thể thực hiện được.



Hình 7.30: Sơ đồ phương pháp đẩy lên trước

Trong đa số các chip vi xử lý với tập lệnh RISC có đưa vào kỹ thuật lập bảng điểm (scoreboarding) để khắc phục chướng ngại dữ liệu. Bản chất của phương pháp này là: có 1 thanh ghi bảng ghi đặc biệt với các bit là đại diện cho các thanh ghi của CPU. Nếu thanh ghi ri là đích trong một lệnh nào đó, thì bit i của thanh ghi bảng điểm đặc biệt được thiết lập. Bit i chỉ được xóa khi lệnh thực hiện xong. Trong khi bit i được thiết lập, không một lệnh nào khác có thể sử dụng được thanh ghi ri.

Có thể phân các chướng ngại do dữ liệu ra các loại:

1. RAW- đọc sau ghi (Read And Write): Lệnh j chủ định đọc một nguồn trước khi lệnh i ghi vào nguồn đó.

2. WAR- ghi sau đọc (Write After Read). Lệnh J chủ định ghi vào một đích trước khi lệnh i đọc đích đó.

3. WAW- ghi sau ghi (Write After Write). Lệnh J chủ định ghi vào một toán hạng trước khi nó được ghi vào nhờ lệnh i. Điều này có thể xảy ra trong các đường ống trong đó ghi được thực hiện trong quãng thời gian hơn một giai đoạn. Nó không thể xảy ra nếu ghi được thực hiện chỉ trong giai một giai đoạn W.

c) Các chướng ngại do điều khiển (control hazard)

Chúng xuất hiện do có các lệnh rẽ nhánh, nhảy, và điều khiển khác làm thay đổi luồng lệnh trong đường ống. Khi trong đường ống bật lên lệnh nhảy hay rẽ nhánh thì luồng lệnh theo sau phải được chỉnh đầy (flush), và lệnh mới (là đích của nhảy hoặc rẽ nhánh) phải được đọc ra đường ống.

Để giảm thiểu các dạng chướng ngại này, tức là tối thiểu số lượng các lệnh bùng phát từ đường ống) cần phải thực hiện các bước như sau:

1. Cố gắng nhận biết lệnh rẽ nhánh trong đường ống thật sớm. Nếu đó là lệnh nhảy vô điều kiện thì sự nhận biết này dễ dàng ở giai đoạn D. Nếu là rẽ nhánh có điều kiện thì sẽ khó nhận biết hơn, vì điều kiện phải được kiểm tra tra sau khi lệnh được giải mã, và như thế cần thêm thời gian.

2. Cố gắng lấy được địa chỉ đích và nạp nó vào PC thật sớm. Nếu địa chỉ đích cần phải tính theo kiểu địa chỉ, thì phải tốn thêm chu kỳ thời gian.

Một trong những cách giảm trễ do rẽ nhánh khi gặp chướng ngại do điều khiển là đưa vào chip vi xử lý cache đích rẽ nhánh BTC (Branch Target cache). Một BTC chứa một tập hợp một số lệnh đầu tiên có khả năng là các đích rẽ nhánh. Nếu đích rẽ nhánh yêu cầu có trong BTC thì sẽ tiết kiệm chỉ cần đọc lệnh từ BTC, mà không cần đọc từ bộ nhớ bên ngoài.

Một phương pháp đơn giản hơn là đưa vào đường ống các chu kỳ chờ cho đến khi lệnh đích rẽ nhánh được đưa tới, trong khi làm đầy các lệnh đọc trước theo sau lệnh nhảy. Nếu rẽ nhánh được phát hiện

trong giai đoạn D, thì PC được nạp địa chỉ lệnh (đích của rẽ nhánh) và lệnh này nhanh chóng có trong BTC.

Phương pháp dự đoán rẽ nhánh (branch prediction): có thể thiết kế một phần cứng dự đoán các rẽ nhánh sẽ không xảy ra. Khi đó, các lệnh theo sau lệnh rẽ nhánh sẽ tiếp tục được thực hiện. Khi và nếu trong đường ống chuyển bật lên lệnh rẽ nhánh thì đường ống dừng lại, các kết quả của các lệnh theo sau lệnh rẽ nhánh được chỉnh đầy, và lệnh đích (phải rẽ nhánh tới) được đọc ra đường ống. Tương tự, có thể thiết kế hệ thống với dự đoán rẽ nhánh sẽ xảy ra.

Hầu hết các chip vi xử lý RISC áp dụng phương pháp rẽ nhánh trễ (delayed branch). Trong phương pháp này, lệnh tiếp theo sau lệnh rẽ nhánh luôn được thực hiện, trong khi sự rẽ nhánh được làm chậm lại một chu kỳ. Như vậy, giảm tối thiểu số chu kỳ phải mất thêm. Đối với các chip vi xử lý có kiến trúc siêu hướng (superscalar), trong đó, có 2 hoặc nhiều lệnh được đọc và xử lý đồng thời thì áp dụng phương pháp rẽ nhánh trễ là rất phức tạp. Chúng ta sẽ xét các hệ thống siêu hướng và mức độ song song của các lệnh ILP (Instruction Level Parallelissm) ở mục tiếp theo.

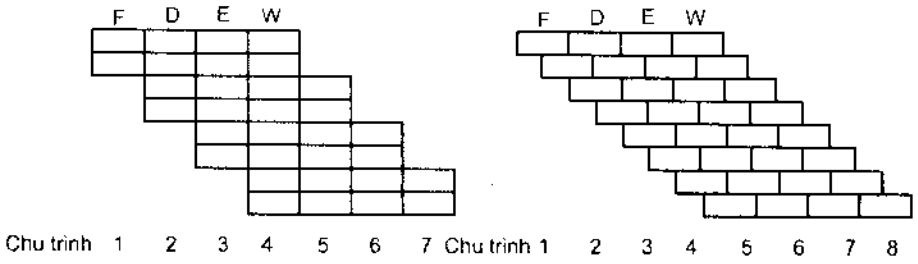
7.2.7 Song song ở mức lệnh (ILP)

ILP (Instruction Level Parallelism) có thể được xác định như là một kỹ thuật đồng thời gọi ra (issue) và xử lý nhiều lệnh bên trong một chip vi xử lý. Trong một hệ thống lý n-issue, có n lệnh được gọi ra trong một chu kỳ của CPU, và n kết quả nhận được trong một chu kỳ. Có một định nghĩa hình thức về ILP:

ILP là một họ của các kỹ thuật thiết kế bộ xử lý và biên dịch (compiler) nhằm tăng tốc sự thực hiện nhờ gây ra các thao tác, như nạp/ghi bộ nhớ, tính toán các số nguyên và dấu phẩy động, thực hiện song song.

Phân biệt những loại ILP chính sau đây:

1. Siêu hướng (superscalar): trong đó, một số lệnh được đồng thời đưa ra trong một chu trình. Hình 7.31a mô tả đường ống 4 phân đoạn thực hiện siêu hướng 2 cửa ra.



Hình 7.31: (a): Thực hiện siêu hướng và (b): Siêu đường ống 2 cửa ra

2. Siêu đường ống (superpipelined): trong đó, một số lệnh đưa ra trong một chu trình nhưng không đồng thời. Trong hệ thống có kỹ thuật siêu hướng với n cửa ra, cứ mỗi 1/n chu trình một lệnh mới được đưa ra. Hình 7.31b mô tả siêu đường ống 2 cửa ra. Nó tương tự như thực hiện đường ống ở tần số gấp đôi. Hệ thống siêu đường ống n cửa ra thực hiện tương tự như một hệ thống đường ống bình thường nhưng nhanh n lần.

3. VLIW- từ lệnh rất dài (very long instruction word): trong đó, một lệnh chứa nhiều mã phép toán với các toán hạng.

Hầu hết các bộ vi xử lý hiện nay đều thực hiện được các thao tác siêu hướng, mà phần lớn là siêu hướng 2 cửa ra. Có một số loại có kỹ thuật siêu hướng 3 cửa ra, ví dụ SuperSPARC, hoặc siêu hướng 4 cửa ra như IBM RS/6000. Tương lai có thể có các chip vi xử lý hỗ trợ kỹ thuật siêu hướng nhiều cửa ra hơn, ví dụ, 8 cửa ra. Hiện nay các hệ thống MIPS R4000 và R4400 là có kỹ thuật siêu đường ống.

Trong hệ thống siêu hướng n cửa ra, n lệnh được đọc ra và được giải mã đồng thời. Sau đó chúng được thực hiện đồng thời để sao cho kết quả của chúng nhận được đồng thời trong một chu kỳ (cycle). Để thực hiện n lệnh đồng thời, cần phải có n tài nguyên để thực hiện. Nếu đọc đồng thời 2 lệnh tính toán số học, thì ta cần phải có 2 đơn vị nguyên (Integer unit) để đồng thời thực hiện 2 lệnh. Trong hệ thống siêu hướng 2 cửa ra dễ dàng nhóm các chương trình thành từng cặp gồm các lệnh độc lập nhờ sự can thiệp của người sử dụng hay chương

trình dịch (compiler). Nhưng sẽ khó khăn hơn đối với các hệ thống siêu hướng 3, 4 cửa ra nếu ta muốn nhóm các chương trình có các lệnh độc lập.

7.2.8 Đa xử lý (multiprocessors)

Một bộ đa xử lý bao gồm một số đơn vị xử lý PE (CPU) giống nhau cùng thực hiện nhiệm vụ chung hay những nhiệm vụ được chia sẻ. Bộ đa xử lý khác với hệ thống nhiều máy tính hoặc mạng máy tính ở chỗ các hệ thống (hay mạng máy tính) có các máy tính (PE) thực hiện độc lập các nhiệm vụ riêng biệt. Một bộ đa xử lý là một máy tính MIMD.

Có hai lý do đưa kiến trúc đa xử lý vào trong một máy tính: đó là nâng cao hiệu suất và độ tin cậy. Hiệu suất của máy tính được nâng cao nhờ các PE chia sẻ tải khi cùng thực hiện một nhiệm vụ quá lớn và đồng thời thực hiện các nhiệm vụ đơn lẻ (xử lý song song). Độ tin cậy được tăng lên vì khi có một PE bị hỏng thì chức năng của nó được các PE còn lại gánh lấy, nhờ vậy, toàn bộ hệ thống máy tính vẫn hoạt động bình thường (hiệu suất có thể giảm đi chút ít).

Các bộ đa xử lý có thể được phân loại dựa trên tổ chức của các hệ thống bộ nhớ chính của chúng. Nếu bộ nhớ chính được tất cả PE truy nhập tới thông qua mạng kết nối bên trong dạng nào đó (hình 7.11a) thì đó là bộ đa xử lý chia sẻ bộ nhớ. Bộ nhớ chung cho tất cả PE gọi là bộ nhớ tổng thể. Trong bộ đa xử lý phân tán bộ nhớ (hình 7.11b) thì mỗi một PE_i có riêng một đơn vị nhớ cục bộ M_i.

Kiến trúc chia sẻ bus (shared bus) là một trong những kiến trúc được áp dụng nhiều cho các hệ thống xử lý song song. Trong kiến trúc này các đơn vị xử lý có các đơn vị nhớ riêng và đơn vị IO riêng. Các đơn vị xử lý kết nối với nhau, chia sẻ bộ nhớ chung (global memory) và IO chung (global IO) thông qua bus chung (global bus). Các đơn vị IO và đơn vị nhớ cục bộ riêng kết nối với PE riêng thông qua bus cục bộ (local bus). Trong họ vi xử lý từ Intel 8086 cho đến Pentium hiện nay đã có thiết kế hỗ trợ cho kiến trúc đa xử lý bus chia sẻ (shared bus multiprocessor architechturres) với đa bus (Multibus) chuẩn IEEE 796

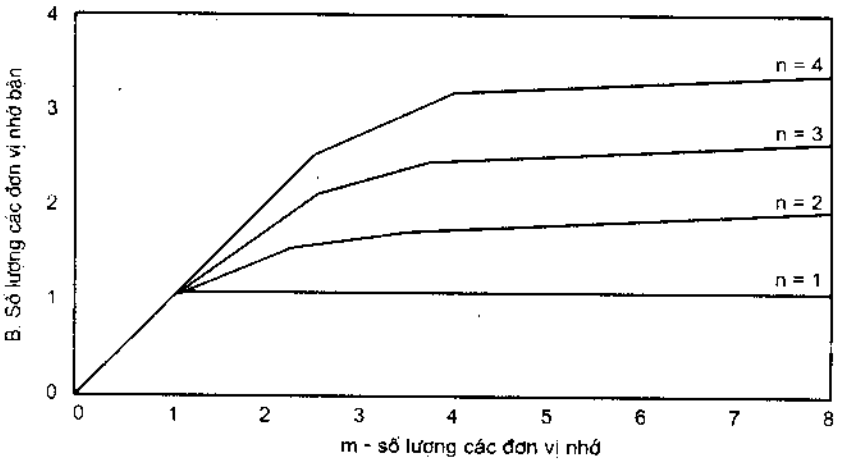
phục vụ như là bus toàn cầu và bus nội bộ. Các máy vi tính thế hệ Intel Pentium hiện nay có bus kết nối một số chip vi xử lý hoạt động song song.

a) Hiệu suất

Do kiến trúc của bộ đa xử lý (có n PE) nên lý tưởng thì có thể cho S(n) = n, với băng thông dữ liệu và lệnh là b thì có thể đạt được hiệu suất là nb. Giả sử, một bộ đa xử lý gồm n đơn vị xử lý: PE_1, PE_2,.., PE_n, và có m đơn vị nhớ: M_1, M_2,..., M_m kết nối theo kiến trúc bus giao nhau (cross). Tất cả các dữ liệu và chương trình cất ở trong mô-đun nhớ M_m. Băng thông của lệnh hoặc dữ liệu của PEj là b tỷ lệ với tốc độ truy nhập của PEj tới đơn vị nhớ, mà tốc độ truy nhập đơn vị nhớ lại phụ thuộc vào số lượng trung bình B các đơn vị nhớ bận. Xác suất truy nhập đơn vị nhớ (đọc hoặc ghi) M_i của bất kỳ đơn vị PEj là 1/m. Xác suất mà đơn vị nhớ M_i bận là $1-(1-1/m)^n$. Khi M_i bận, thì yêu cầu mà nó nhận được không thể phục vụ ngay, phải chờ đến khi M_i. Như vậy, giá trị sẽ là:

$$B = \sum_{i=1}^{m} p_i = m\left[1-\left(1-\frac{1}{m}\right)^n\right] \qquad (7.4)$$

Nếu $n \to \infty$, $B \to m$ và m cố định, tương tự n cố định và $m \to \infty$ thì B = n. Hình 7.32 là đồ thị giá trị B phụ thuộc vào số lượng PE, số lượng M trong một bộ xử lý chia sẻ bộ nhớ.



Hình 7.32: Hiệu suất của bộ xử lý chia sẻ bộ nhớ

b) Hệ điều hành trong các hệ thống đa xử lý

Các bộ đa xử lý cần có hệ điều hành chuyên dụng để điều khiển quá trình hoạt động song song của các thành phần PE và M. Quá trình xử lý song song đòi hỏi hệ điều hành phải có phân phối các tài nguyên tối ưu tránh những đụng độ. Có 3 cách tổ chức hệ điều hành cho bộ đa xử lý:

(1) Hệ điều hành riêng biệt (separate operating system)

Mỗi một PE (CPU) có thể độc lập thực hiện tất cả các chương trình con của hệ điều hành mà nó cần. Trong bộ đa xử lý phân tán bộ nhớ, mỗi một PE có thể có một bản sao toàn bộ hệ điều hành cho riêng mình. Nhưng đa phần chỉ phần lõi của hệ điều hành được sao cho từng PE, những phần còn lại của hệ điều hành để vào bộ nhớ dùng chung (bộ nhớ thứ cấp hay sơ cấp).

(2) Hệ điều hành chủ-tớ (master-slave operating system)

Một đơn vị xử lý là đơn vị xử lý chủ (host) có nhiệm vụ thực hiện tất cả các chương trình con của hệ điều hành. Những đơn vị xử lý còn lại là những đơn vị xử lý tớ không thực hiện các chức năng của hệ điều hành (có thể ngoại trừ chức năng minor), chỉ có thể thực hiện các chức năng của hệ điều hành khi chúng có yêu cầu tới đơn vị xử lý chủ. Dạng hệ điều hành này còn gọi là tập trung (centralized).

(3) Hệ điều hành phân tán (distibuted operating system)

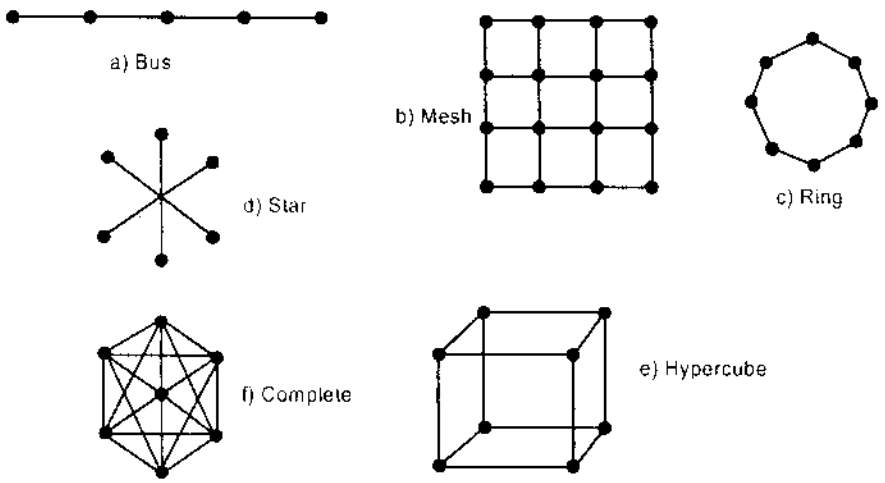
Các chương trình con của hệ điều hành được phân phối cho các đơn vị xử lý sao cho mỗi một chức năng của hệ điều hành được gán cho một đơn vị xử lý ở một thời điểm, nhưng cũng có thể gán cho các đơn vị xử lý khác nhau ở những thời điểm khác nhau.

c) Cấu trúc mạng kết nối tĩnh bên trong đa xử lý

Các bộ đa xử lý có kiến trúc kết nối bên trong rất khác nhau, mỗi cấu hình đấu nối các đơn vị xử lý bên trong với nhau cho hiệu suất xử lý và độ tin cậy khác nhau. Cũng gần như cấu hình đấu nối mạng máy

tính, mạng kết nối bên trong của các bộ đa xử lý có dạng phổ biến sau đây (hình 7.33):

1. Cấu trúc bus (hình 7.33a).
2. Cấu trúc lưới (mesh) (hình 7.33b)
3. Cấu trúc vòng (ring) (hình 7.33c)
4. Cấu trúc sao (star) (hình 7.33d)
5. Cấu trúc lập thể (cub) (hình 7.33e)
6. Cấu trúc toàn bộ (complet) (hình 7.33f).



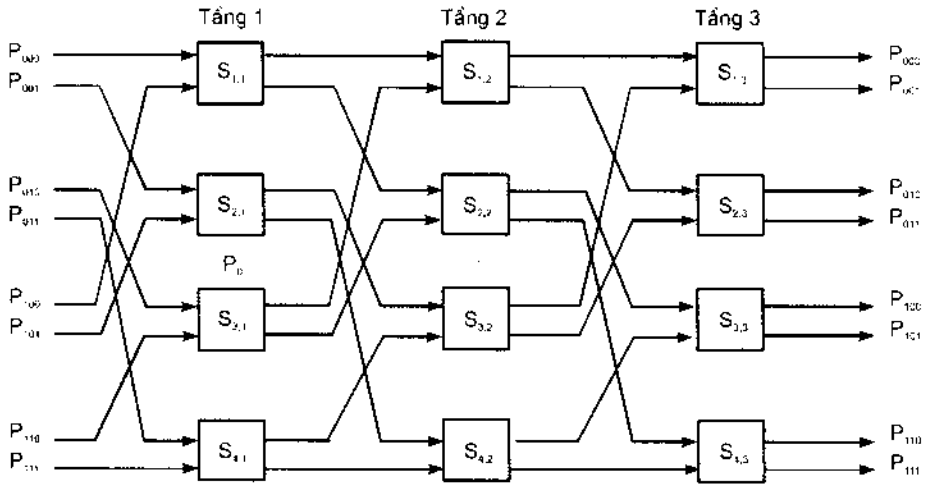
Hình 7.33: Cấu trúc kết nối mạng tĩnh bên trong đa xử lý

Cũng như mạng máy tính, cấu trúc kết nối toàn bộ cho độ tin cậy cao nhất, nhưng điều khiển lại phức tạp, chi phí cao vì công nghệ chế tạo. Các cấu trúc bus, star, ring có độ tin cậy thấp nhưng điều khiển và chi phí thấp.

Toàn bộ các cấu trúc trên đây là kết nối tĩnh không linh hoạt, và do đó, hiệu suất cũng thay đổi phụ thuộc vào chủng loại các nhiệm vụ được xử lý.

Cấu trúc kết nối động bên trong bộ đa xử lý khắc phục nhược điểm của cấu trúc kết nối tĩnh, nhưng lại phức tạp cho điều khiển và

chi phí cao. Một trong những cấu trúc kết nối động thông dụng là mạng kết nối nhiều tầng (hình 7.34).



Hình 7.34: Một trạng thái của mạng chuyển mạch 3 tầng kết nối các đơn vị xử lý bên trong bộ đa xử lý

Mạng kết nối các PE là một mạng nhiều tầng với nhiều tuyến truyền dẫn (liên kết) động, bởi vì nó được định tuyến kết nối nhờ điều khiển từng nút chuyển mạch.

Mạng kết nối đa tầng rất linh hoạt trong kết nối nhờ sự điều khiển các nút chuyển mạch để định tuyến. Bảng định tuyến của mạng được thiết lập và thay đổi trong quá trình xử lý là chìa khóa để nâng cao hiệu suất của bộ đa xử lý. Còn nhiều vấn đề có thể giải quyết trong giải pháp mạng đa tầng để nâng cao độ tin cậy của đa xử lý, ví dụ: tối ưu hóa cấu trúc/chi phí/hiệu suất, và ứng dụng mạng Nơ-ron trong thiết kế mạng liên kết động này.

Hiện nay, xử lý song song là một công nghệ đang được nhiều nhà sản xuất máy tính, kỹ thuật vi xử lý quan tâm áp dụng, đưa vào các sản phẩm thương mại, đặc biệt là các thế hệ vi xử lý công nghệ cao có xử lý ống, siêu hướng có tập lệnh tối thiểu (RISC). Chúng ta đã có dịp xét ở chương 5 kỹ thuật vi xử lý Motorola M68060, một loại kiến trúc đã

được trang bị xử lý ống cho IU và FPU. Để lấy thêm ví dụ về dạng kỹ thuật này, chúng ta xét thêm một số loại vi xử lý cao cấp, thông dụng hiện nay được các nhà sản xuất như: IBM, SUN, DEC,... chế tạo và đã thương mại trên thị trường.

7.3. CÔNG NGHỆ RISC

7.3.1 RISC và CISC

Các bộ vi xử lý Intel x86 và Motorola M68000 đều có tập lệnh lớn và phức tạp, có nhiều chế độ đánh địa chỉ, và có nhiều khuôn dạng lệnh với kích thước khác nhau. Sự điều khiển của các chip vi xử lý này thực hiện bằng vi chương trình, và các lệnh khác nhau có độ dài thời gian thực hiện khác nhau bằng một số chu trình (cycle). Các đơn vị điều khiển của các loại vi xử lý này rất phức tạp bởi vì chúng phải giải mã, phân biệt một số lớn các mã lệnh, các kiểu địa chỉ, và khuôn dạng. Các hệ thống máy tính dùng các loại vi xử lý này làm CPU được xếp vào các loại máy tính có tập lệnh phức tạp CISC (Complex Instruction Set Computer). Mặc dù nhiều loại máy tính CISC có áp dụng kỹ thuật đường ống để nâng cao hiệu suất xử lý nhưng vẫn còn có nhiều phức tạp trong điều khiển xử lý đường ống lệnh do các lệnh có độ dài và thời gian thực hiện rất khác nhau...

Trong các máy tính có tập lệnh rút gọn RISC (Reduced Instruction Set Computer) lý tưởng, tất cả các lệnh các kiểu đánh địa chỉ và khuôn dạng lệnh đều được giảm tối thiểu. Tất cả các lệnh đều có kích thước như nhau (thường là 32-bit) và đều thực hiện chỉ trong một chu trình đơn lẻ của CPU. Trong thực tế, có khoảng 80% các lệnh của hầu hết các hệ thống RISC là được thực hiện trong một chu trình đơn của CPU.

Sự phức tạp của CISC làm cho thời gian thiết kế hệ thống lâu hơn, phức tạp hơn và có thể có xác suất sai lớn hơn trong thiết kế. Tập lệnh lớn của CISC làm cho các chương trình dịch ngôn ngữ bậc cao cũng có nhiều phương án hơn và khó chọn ra một phương án tối ưu.

Kiến trúc đường ống trong các RISC cũng chỉ phát huy hiệu quả nếu các lệnh có kích thước và thời gian thực hiện tương đối giống nhau.

Trong các hệ thống RISC, trong bất kỳ một chu trình đọc lệnh nào, toàn bộ một lệnh được chuyển đến CPU. Nếu độ dài từ cơ bản là 32-bit và bus dữ liệu của hệ thống cũng là 32-bit (trong hầu hết các hệ thống hiện nay) thì một lệnh tiêu chuẩn có độ dài là 32-bit. Một số loại vi xử lý cho ta bus dữ liệu của hệ thống máy tính là 64-bit, điều này có nghĩa là đồng thời có thể đọc ra 2 lệnh. Nhưng đạt được một quãng thời gian chuẩn để thực hiện tất cả các lệnh là điều khó hơn là đạt được mục tiêu đọc tất cả các lệnh trong một chu trình lệnh. Bởi vì một số lệnh không có tham chiếu thêm bộ nhớ mà chỉ thao tác với các thanh ghi bên trong CPU, một số lệnh khác thì phải truy nhập thêm bộ nhớ vì toán hạng, hoặc các lệnh nhân và chia không thể thực hiện được trong một chu trình. Lý tưởng, ta mong muốn có được sự xử lý tất cả các lệnh một cách đồng nhất và có sắp xếp hợp lý, trong đó, các phân đoạn đọc lệnh và thực hiện lệnh đều có thời gian tương tự đối với tất cả các lệnh (trong mô hình đường ống hai phân đoạn, hình 7.10), và tốt nhất là chúng đều được thực hiện trong một chu trình. Đây là một trong những nguyên tắc quan trọng yêu cầu trong thiết kế các hệ thống RISC. Tất cả các lệnh đọc từ bộ nhớ vào CPU để thực hiện phải đi theo một chuỗi có trật tự hợp lý, và tất cả các lệnh đều được thực hiện trong khoảng thời gian giống nhau không có thời gian chờ đợi. Trong trạng thái đó, CPU luôn luôn bận.

7.3.2 Các đặc điểm của RISC

Để đạt được sự thực hiện lệnh theo chuỗi trong CPU của RISC cần có những điều kiện sau đây:

1. Chuẩn hóa kích thước của lệnh, bằng độ dài từ của máy tính và độ rộng của bus dữ liệu. Độ rộng của bus dữ liệu của hệ thống phải bằng tích của kích thước lệnh với một số nguyên.

2. Chuẩn hóa thời gian thực hiện của tất cả các lệnh, tốt hơn hết là chỉ trong một chu trình đơn của CPU. Chỉ một số ít các lệnh, ví dụ như lệnh nhân và chia là phải thực hiện trong một số chu trình của CPU.

Chuẩn hóa kích thước lệnh là cho đơn giản hơn sự giải mã lệnh và các mạch điều khiển cục bộ bên trong vi xử lý, như vậy cũng là cho thời gian trễ tạo ra các tín hiệu điều khiển cũng ít đi. Cũng có thể có làm như: đóng gói 2 lệnh độc lập với nhau có độ dài 16-bit trong một từ 32-bit và cùng được đọc ra trong ống lệnh. Như thế có thể làm tăng tốc độ thực hiện lệnh. Nhưng không phải bao giờ cũng đạt được lệnh có độ dài 16-bit hay 32-bit. Các máy tính hiện nay phần lớn đều có các lệnh 2, 3 toán hạng, nghĩa là cần lệnh có độ dài 32-bit chứ không phải là 16-bit. Chúng có nhiều chế độ đánh địa chỉ (hơn một chục chế độ khác nhau), và các lệnh có tới hơn chục kích thước khác nhau.

Có thể chấp nhận những yêu cầu sau đây cho hệ thống RISC:

- Số lượng các lệnh: nhỏ hơn hoặc bằng 128,

- Số lượng các chế độ đánh địa chỉ: nhỏ hơn hoặc bằng 4,

- Số lượng các khuôn dạng lệnh: nhỏ hơn hoặc bằng 4.

Thực hiện tất cả các yêu cầu này là không thể được, và thực tế trong các hệ thống RISC hiện nay, chỉ có thể đạt được 80% các lệnh thực hiện được trong một chu trình đơn của CPU. Có những nguyên tắc mà trong hầu hết các hệ thống RISC hiện nay chấp nhận là:

- Trong phân đoạn thực hiện lệnh của đường ống lệnh, các truy nhập bộ nhớ chỉ được thực hiện nhờ các lệnh nạp (LOAD) hoặc cất giữ (STORE).

- Tất cả các thao tác thực hiện lệnh, ngoại trừ truy nhập bộ nhớ bằng LOAD và STORE, đều là thao tác thanh ghi đến thanh ghi (rigister-to-register) bên trong CPU.

Để làm cho các lệnh đều là các lệnh thao tác register-to-register, bên trong vi xử lý hiện đại được trang bị nhiều thanh ghi, và do vậy người ta gọi là tệp các thanh ghi. Trong đó, có các thanh ghi dùng làm nơi lưu giữ kết quả, dữ liệu trung gian trong quá trình thực hiện lệnh. Hầu hết các hệ thống vi xử lý kiến trúc RISC có 32 thanh ghi dùng chung.

Các hệ thống RISC cũng như CISC đều được vi chương trình hóa và để thực hiện được sự đồng nhất chuỗi thực hiện lệnh các điều khiển cục bộ bằng vi chương trình bên trong CPU thường được cứng hóa để tạo ra các tín hiệu khiển nhanh hơn.

Một đặc điểm quan trong trong các hệ thống RISC là có phần cứng hỗ trợ cho các chương trình dịch ngôn ngữ bậc cao. Đặc điểm này tạo cho người thiết kế chương trình dịch có được kết quả tối ưu hơn.

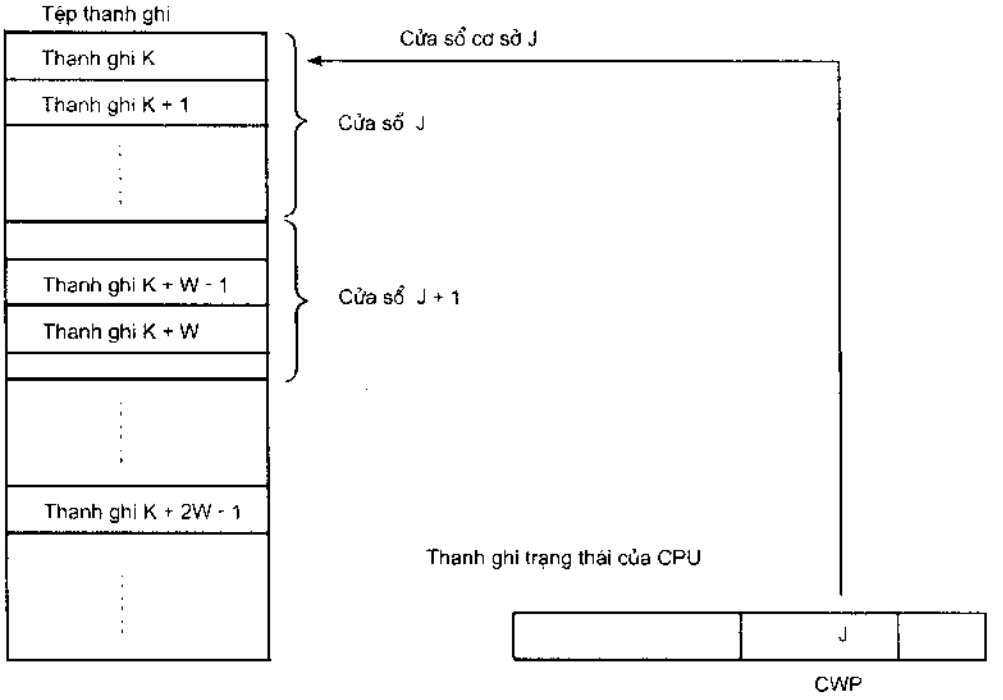
Kiến trúc đường ống là một kỹ thuật sử dụng rộng rãi trong các hệ thống RISC và sẽ phát huy hiệu quả nếu như đạt được sự đồng nhất về kích thước lệnh, thời gian thực hiện lệnh trong một chu kỳ.

Bộ nhớ cache cũng được phân chia làm 2 loại giống như trong các hệ thống CISC: cache cho dữ liệu (Dcache) và cache cho lệnh (Icache). Nhưng hầu hết chỉ phân chia 2 loại cache bên trong (L1), mà không có phân chia 2 loại cho cache bên ngoài (L2), và bộ nhớ chính (L3) lưu trữ đồng thời dữ liệu và lệnh.

Có một đặc tính được các hệ thống RISC đưa vào, đó là cửa sổ thanh ghi (register window).

Một tệp thanh ghi được phân thành các nhóm thanh ghi, mỗi nhóm được gọi là một cửa sổ thanh ghi. Một cửa sổ thanh ghi (một nhóm thanh ghi) nào đó với một số lượng là i thanh ghi, đánh số thứ tự từ r0 đến r(i-1) được chọn làm cửa sổ các thanh ghi toàn cục GRW (Global Register Window). Những thanh ghi toàn cục này có thể được tất cả các thủ tục của hệ thống xâm nhập trong mọi thời gian, mà mỗi thủ tục của hệ thống được ấn định cho từng cửa sổ thanh ghi trong tệp thanh ghi. Cơ sở của cửa sổ (thanh ghi đầu tiên của cửa sổ thanh ghi) được trỏ tới nhờ một trường, gọi là con trỏ cửa sổ hiện hành CWP (Current Window Pointer), chứa trong thanh ghi trạng thái của CPU (SR). Nếu một thủ tục đang thực hiện được gán cho cửa sổ J gồm các thanh ghi K, K+1,..., K+W-1 (trong đó W là số lượng thanh ghi trong cửa sổ) (hình 7. 35), thì CWP có giá trị J để trỏ tới thanh ghi đầu tiên của cửa sổ J, tức là trỏ tới thanh ghi K, cơ sở của cửa sổ J. Nếu thủ tục

tiếp theo thực hiện với của sổ J+1 thì CWP được tăng lên 1, tức là có giá trị J+7.



Hình 7.35: Các cửa sổ thanh ghi

Định cửa sổ thanh ghi nhằm mục đích làm cho truyền các thông số giữa các thủ tục gọi và bị gọi có hiệu quả hơn nhờ sự xếp chồng liền nhau của các thanh ghi. N thanh ghi cuối cùng của cửa sổ J là N thanh ghi đầu tiên của cửa sổ J+7. Nếu một thủ tục lấy cửa sổ J để gọi tới thủ tục được gán cửa sổ J+1, thì nó có thể chuyển N thông số tới thủ tục bị gọi (bằng cách đặt các giá trị các thông số vào trong các thanh ghi từ K+W-N đến K+W-1 của cửa sổ J. Tương tự các thanh ghi như vậy sẽ tự động có cho thủ tục bị gọi mà không cần phải có thêm bất kỳ một lệnh thao tác vận chuyển dữ liệu nào khác. Nói một cách khác, thủ tục gọi chỉ tác động đến CWP làm tăng giá trị lên 7.

Về lý thuyết, định cửa sổ cho tệp thanh ghi có thể thực hiện trong bất kỳ hệ thống máy tính nào. Nhưng vấn đề là trong chế tạo các chip

vi xử lý công nghệ VLSI của các hệ thống CISC, đơn vị điều khiển (CU) chiếm không gian lớn hơn rất nhiều so với không gian chiếm của tệp thanh ghi bên trong chip vi xử lý, mà để định cửa sổ thanh ghi cần có không gian đủ lớn cho tệp các thanh ghi. Trong khi đó, trong các chip vi xử lý RISC, đơn vị điều khiển chiếm không gian nhỏ hơn nhiều và như thế không gian của tệp thanh ghi có thể thực hiện lớn hơn (số lượng các thanh ghi nhiều hơn) tạo điều kiện để lập các cửa sổ thanh ghi.

Một đặc điểm đã được nhắc đến là xử lý song song ở mức lệnh ILP (Istruction Level Parallelism), trong đó: nhiều lệnh (n lệnh) độc lập được đồng thời đưa ra và được thực hiện trong hệ thống ILP nhiều cửa ra (n cửa ra) trong một chu trình đơn của CPU. Kỹ thuật này cũng được phát huy hiệu quả vì có giảm thiểu tập lệnh và tiêu chuẩn hóa kích thước và thời gian thực hiẹen hầu hết các lệnh.

Trong xử lý theo đường ống lệnh, có thể xuất hiện các lệnh rẽ nhánh có điều kiện. Vì luôn có công đoạn tiền đọc lệnh vào đường ống lệnh, khi một lệnh được đưa vào thực hiện là lệnh rẽ nhánh có điều kiện, nếu điều kiện không thỏa mãn thì không có rẽ nhánh nào cả, và lệnh tiếp theo đã được đọc trước đó được đưa vào thực hiện. Như thế sẽ không có mất mát thời gian nào cả. Nhưng nếu điều kiện rẽ nhánh thỏa mãn (hoặc lệnh nhảy vô điều kiện) thì lệnh tiếp theo được đọc trước đó sẽ bị chặn lại và một lệnh mà địa chỉ rẽ nhánh trỏ tới phải được đọc ra để thực hiện. Như vậy, thời gian dành cho đọc lệnh tiếp theo bị chặn lại coi như bị bỏ phí. Sự mất mát thời gian này có thể sửa chữa được bằng cách sử dụng kỹ thuật rẽ nhánh bị trễ. Ví dụ, ta có đoạn lệnh sau đây:

```
CLR r2        ; xóa thanh ghi r2
CMP r1, 10    ; so sánh nội dung r1 với 10
BZ addr1      ; rẽ nhánh đến địa chỉ addr1 nếu r1-10 = 0
ADD r1,10     ; nếu điều kiện không thỏa mãn thì thực hiện lệnh
                tiếp theo này
```

Rõ ràng lệnh tiếp theo ADD r1, 10 được đọc ra và sẽ bị chặn lại nếu (r1) khác 10. Bây giờ ta sắp xếp thứ tự các lệnh như sau:

CMP r1, 10

BZ addr1

CLR r2

ADD r1, 10

Như vậy lệnh tiếp theo của lệnh rẽ nhánh là CLR r2 được đọc ra sau lệnh rẽ nhánh. Sử dụng nguyên tắc rẽ nhánh trễ và cho rằng điều kiện rẽ nhánh thỏa mãn. Lẽ ra lệnh ở địa chỉ addr1 được đọc ngay vào thanh đếm chương trình PC để thực hiện, nhưng ta để trễ lại việc đọc này cho đến khi lệnh tiếp theo CLR r2 được thực hiện. Như thế không có sự lãng phí thời gian đọc lệnh tiếp theo CLR r2, không có sự mất mát thời gian nào cả, không có sự thay đổi nào trong chương trình, bởi vì r2 đằng nào cũng được xóa (trước lệnh rẽ nhánh hay sau lệnh rẽ nhánh), và không phụ thuộc vào rẽ nhánh có thỏa mãn điều kiện hay không. Kỹ thuật làm trễ rẽ nhánh có thể thực hiện được trong bất kỳ vi xử lý hiện đại nào, nhưng thực tế chỉ có một số loại RISC đưa vào kỹ thuật này vì có sự đơn giản trong xử lý đường ống so với trong các loại CISC. Sự thực hiện kỹ thuật rẽ nhánh trong các hệ thống siêu hướng, nhiều đường ống song song rất khó khăn.

Có một vấn đề cũng phải tính đến nếu xét ví dụ đoạn chương trình sau đây:

LOAD memr, r1; nạp vào r1 dữ liệu từ ngăn nhớ có địa chỉ là memr

ADD r1, r2, r3; (r1) + (r2) → r3

Lệnh đầu tiên là nạp vào thanh ghi r1 dữ liệu từ bộ nhớ, và dữ liệu này là cần thiết cho một toán hạng của lệnh tiếp theo (ADD). Cho rằng cả 2 lệnh này đều được đọc trong một chu trình đọc lệnh (F), và lệnh LOAD cần phải có bổ sung thêm chu trình thực hiện (E). Như thế ta có sơ đồ đường ống lệnh như sau:

LOAD F E E

ADD F E

Theo sơ đồ này lệnh ADD được đọc ngay khi bắt đầu thực hiện lệnh LOAD đầu tiên (E), và nó sẵn sàng thực hiện trước khi nội dung của r1 lấy từ bộ nhớ đã sẵn sàng. Như vậy xảy ra lệnh ADD với nội dung cũ của r1, và kết quả phép cộng sẽ sai. Để khắc phục dạng sự kiện này, trong RISC đưa vào một loại thanh ghi điều khiển, gọi là thanh ghi tỷ số (scoreboard register). Thanh ghi này có độ dài 32-bit, mỗi một bit của nó thể hiện một thanh ghi trong tệp thanh ghi có 32 thanh ghi chung. Bit 0 thể hiện thanh ghi r0, bit 1- thanh ghi r1, v.v... Mục đích của thanh ghi tỷ số là để ngăn chặn một thanh ghi nào đó được sử dụng khi bit tương ứng với thanh ghi đó đặt giá trị 1, và thanh ghi chỉ được sử dụng khi bit tương ứng với nó đặt giá trị 0. Trong ví dụ này, giá trị bit của thanh ghi tỷ số được đặt giá trị 1 cho đến khi nào lệnh LOAD thực hiện xong, tức là trong r1 đã có giá trị mới nạp từ bộ nhớ. Một chu trình giữ (giữ giá trị 1 của bit 1 của thanh ghi tỷ số), gọi là H (held) được đưa vào đồng thời với chu trình thực hiện lệnh LOAD thứ 2 nhằm đảm bảo chắc chắn lệnh LOAD thực hiện xong thì mới cho phép lệnh tiếp theo ADD thực hiện:

LOAD F E E

ADD F H E

Như vậy, có thể là phải phí thêm một chu trình, nhưng đảm bảo chắc chắn rằng phép toán ADD thực hiện đúng. Kỹ thuật này gọi là kỹ thuật lập bảng tỷ số (scoreboarding).

Tổng kết lại các đặc điểm quan trọng thực hiện có hiệu quả trong các hệ thống RISC hơn là trong các hệ thống CISC là:

- Hỗ trợ bằng phần cứng cho ngôn ngữ bậc cao;
- Thực hiện định cửa sổ các thanh ghi trong tệp các thanh ghi;
- Kỹ thuật định đường ống (đường ống lệnh);
- Rẽ nhánh trễ;
- Lập bảng tỷ số (scoreboarding);
- Phân chia cache thành cache chứa dữ liệu (Dcache) và chứa lệnh (Icache);
- Xử lý song song ở mức lệnh (ILP).

7.3.3 Các ưu điểm của RISC

Các ưu điểm của RISC được đánh giá trên những đặc điểm sau đây:

Thực hiện công nghệ VLSI, tốc độ xử lý, chi phí thiết kế và độ tin cậy, hỗ trợ ngôn ngữ bậc cao.

a) Thực hiện công nghệ VLSI

Do có tập lệnh tối thiểu, kích thước lệnh nhỏ, đồng nhất thời gian thực hiện lệnh trong một chu trình của CPU, khuôn dạng lệnh ít, ít kiểu đánh địa chỉ là cho thiết kế đường ống và đơn vị điều khiển (CU) bên trong chip vi xử lý RISC đơn giản hơn và vì thế CU cũng chiếm không gian khoảng từ 6 đến 10% không gian của toàn chip RISC (trong khi đó CU chiếm từ 50 đến 70% không gian của chip CISC). Nhờ những đặc tính này mà người thiết kế có thể:

- Có thêm nhiều không gian trống để tăng thêm các tính năng của các khối chức năng như MMU, FPU, điều khiển vào/ra, và dung lượng cache bên trong chip RISC.

- Tăng số lượng thanh ghi trong tệp thanh ghi (có thể hơn 100 thanh ghi), tạo khả năng thực hiện được nhiều loại chương trình với ít thời gian truy nhập đến bộ nhớ ngoài chip vi xử lý.

- Tăng hệ số điều chỉnh: hệ số điều chỉnh được xác định bằng tổng số các khối chức năng trong chip, kể cả ROM chia cho số lượng các mạch chi tiết, như các thanh ghi, ALU, bộ đếm, và các mạch nhỏ khác. Hệ số điều chỉnh lớn, tức là cho phép đưa thêm nhiều khối chức năng.

- Chế tạo các chip bằng công nghệ GaAs VLSI không có được mật độ tích hợp cao so với công nghệ CMOS (Complementary Metal-Oxide Semiconductor). Vì vi xử lý RISC không đòi hỏi mật độ tích hợp cao nên có thể thực hiện bằng công nghệ GaAs VLSI.

b) Tốc độ xử lý

Tốc độ xử lý của RISC tăng lên so với CISC do những yếu tố sau:

- Tập lệnh tối thiểu, ít khuôn dạng lệnh và chế độ đánh địa chỉ, thời gian thực hiện hầu hết các lệnh đều trong một chu trình đơn của CPU.

- Điều khiển đường ống lệnh đơn giản và kỹ thuật ILP nhiều cửa ra thực hiện hiệu quả.

- Đơn vị điều khiển đơn giản do đó thời gian giải mã lệnh, thời gian tạo ra các tín hiệu điều khiển và các vi lệnh ngắn.

- Tập các thanh ghi lớn nên có thể thực hiện lệnh với nhiều thao tác thanh ghi tới thanh ghi bên trong CPU, ít các thao tác truy nhập bộ nhớ ngoài CPU, sử dụng kỹ thuật định cửa sổ thanh ghi để tiết kiệm thời gian vận chuyển dữ liệu.

- Sử dụng kỹ thuật rẽ nhánh trễ để tránh lãng phí thời gian tiền đọc lệnh tiếp theo khi lệnh rẽ nhánh thỏa mãn điều kiện.

c) Chi phí thiết kế và độ tin cậy

Chi phí thiết kế RISC ít và độ tin cậy cao so với CISC vì những lý do sau đây:

- Chi phí thời gian cho thiết kế CU ít, vì vậy, tổng chi phí thời gian cho thiết kế toàn bộ chip RISC sẽ ít đi.

- Hệ số điều chỉnh lớn tạo ra nhiều khả năng lựa chọn thiết kế hơn.

- Khả năng tăng số lượng thanh ghi bên trong chip làm cho thiết kế điều khiển bên trong đơn giản hơn.

- Sự đơn giản của CU làm cho sai sót thiết kế giảm nên độ tin cậy tăng lên.

- Kích thước lệnh được chuẩn hóa, ít khuôn dạng và chế độ đánh địa chỉ lệnh làm cho việc thiết kế MMU đỡ phức tạp hơn.

d) Hỗ trợ ngôn ngữ bậc cao

Sự hỗ trợ cho các chương trình dịch ngôn ngữ bậc cao của RISC đặc biệt hiệu quả do những lý do sau đây:

- Số lượng lệnh ít nên chương trình dịch sẽ nhỏ và kết quả dịch về lệnh máy là tối ưu, không có nhiều chọn lựa.

- Số lượng các thanh ghi bên trong nhiều nên tạo khả năng cho chương trình dịch sử dụng tối đa các thao tác thanh ghi tới thanh ghi hơn là truy nhập bộ nhớ.

- Kỹ thuật định cửa sổ thanh ghi cho phép truyền tham số giữa các thủ tục trực tiếp nhanh hỗ trợ cho chương trình dịch ngôn ngữ bậc cao xử lý các chương trình con và thủ tục.

- Tập lệnh đơn giản sẽ giới hạn các mức dịch, như thế thời gian dịch nhanh hơn.

7.3.4 Các nhược điểm của RISC

Nhược điểm của RISC xuất phát từ nguyên nhân: tập lệnh tối thiểu và đơn giản.

CISC có tệp lệnh phong phú, và một lệnh thực hiện trong CISC bằng 2, 3 hoặc nhiều hơn lệnh phải thực hiện trong RISC. Như thế phải viết nhiều mã lệnh hơn trong RISC để thực hiện một thao tác tương tự trong CISC. Nhiều lệnh hơn đồng nghĩa với việc phải dành bộ nhớ nhiều hơn để chứa lệnh và dữ liệu, điều này kéo theo sự tăng thời gian truy nhập bộ nhớ của CPU. Để thực hiện một chức năng tương tự nhau, một chương trình chạy trên RISC dài hơn 30% so với chương trình chạy trên CISC. Nhược điểm này là một vấn đề quan trọng đang được các nhà sản xuất các hệ thống RISC quan tâm giải quyết.

7.4 CÁC BỘ VI XỬ LÝ TIÊN TIẾN KIẾN TRÚC RISC CỦA MỘT SỐ HÃNG

Chúng ta đã xét các bộ vi xử lý tiên tiến của Intel, từ 80x86 đến Pentiums, của Motorola họ 68xxx. Các loại vi xử lý đã có đưa vào kiến trúc đường ống lệnh, nhưng chúng vẫn có kiến trúc CISC vì các tập lệnh phức tạp của chúng. Intel cũng cho ra đời họ vi xử lý tiên tiến kiến trúc RISC i860, và Motorola có họ vi xử lý kiến trúc RISC M88000 (MC88110 - MC88200). Các hãng cạnh tranh khác như DEC, Sun, IBM, HP, và MIPS cũng các dòng vi xử lý kiến trúc RISC.

7.4.1 Họ vi xử lý Intel i860

Intel i860 RISC được giới thiệu năm 1989. Nó có các phiên bản i860 XR và i860 XP. Intel i860 XP có công nghệ: đóng vỏ 262 chân, 0,8micron CHMOS, 3 lớp metal, 2,56 triệu transistor, công suất tiêu thụ 5 W ở 50 MHz.

Intel i860 XP có bên trong chip FPU, cache đôi 32 kB (16 kB Icache và 16 kB Dcache), đơn vị đồ họa (GU) và 2 tệp thanh ghi chung:

1. Tệp thanh ghi số nguyên (IRF): gồm 32 thanh ghi 32-bit (R0-R31). Như thường trong các vi xử lý RISC, thanh ghi R0 được gán cứng giá trị 0.

2. Tệp thanh ghi dấu phẩy động (FRF): gồm 32 thanh ghi 32-bit (F0-F31). Các thanh ghi F0 và F1 được gán cứng giá trị 0. Khi truy nhập các giá trị dấu phẩy động 64-bit, một đôi chẵn-lẻ thanh ghi có thể được sử dụng như (F2, F3). Khi truy nhập giá trị 128-bit, một tập hợp gồm 4 thanh ghi thứ tự liên tiếp nhau, như (F0, F1, F2, F3), (F4, F5, F6, F7),..., hoặc (F28, F29, F30, F31) có thể được sử dụng. Khi đó, trong lệnh máy địa chỉ trỏ tới thanh ghi có số hiệu chẵn thấp nhất trong đôi, hay tập hợp các thanh ghi.

Intel i860 (hay 80860) có bus dữ liệu 64-bit và xử lý 64-bit cả trong và bên ngoài chip. Nó có thể thực hiện các lệnh với các toán hạng 8-bit, 16-bit, 32-bit, 64-bit, và 128-bit. Các số dấu phẩy động thỏa mãn chuẩn IEEE 754-1985 cho các dạng 32-bit và 64-bit. Đơn vị đồ họa hỗ trợ các điểm (pixel) 8-bit, 16-bit, 32-bit.

Intel i860 có 32-bit địa chỉ, đánh tới 4GB nhớ (2^{32}), và có các chế độ đánh địa chỉ rất đơn giản, đó là:

1. Thanh ghi (register): toán hạng là thanh ghi trong chip

2. Ngay lập tức (immediate): toán hạng là một phần của lệnh máy

3. Offset: là địa chỉ tuyệt đối trong không gian nhớ 32 kB

4. Gián tiếp thanh ghi với độ dịch - EA (register indirect with offset - EA): (reg) + offset; viết ở hợp ngữ là: #const(reg)

5. Gián tiếp thanh ghi với chỉ số - EA (register indirect with index - EA): (reg1) + (reg2); viết ở hợp ngữ là: reg1(reg2).

Intel i860 có tập lệnh RISC nhưng không có các lệnh nhân và chia. Bởi vì phép nhân được thực hiện nhờ bộ nhân số dấu phẩy động, và phép chia có thể thực hiện hàm lấy giá trị nghịch đảo (reciprocal value instruction) trong tập lệnh.

Intel i860 có một số đường ống lệnh cho phép lập trình điều khiển trực tiếp các lệnh đường ống. Đường ống lệnh nguyên có 4 giai đoạn:

1. F: đọc lệnh từ Icache

2. D: giải mã lệnh và truy nhập tệp thanh ghi

3. E: thực hiện lệnh

4. W: ghi (cất giữ) kết quả vào tệp thanh ghi

Nếu là lệnh rẽ nhánh, thì giai đoạn 3 trở thành giai đoạn I:

I: truy nhập Icache để lấy lệnh đích (của rẽ nhánh) và giải mã lệnh trong quãng tiếp theo lệnh rẽ nhánh. Nếu là lệnh rẽ nhánh có điều kiện, thì kiểm tra mã điều kiện. Nếu là lệnh gọi tới chương trình con, thì giai đoạn 4 (W) ghi địa chỉ trở về vào tệp thanh ghi.

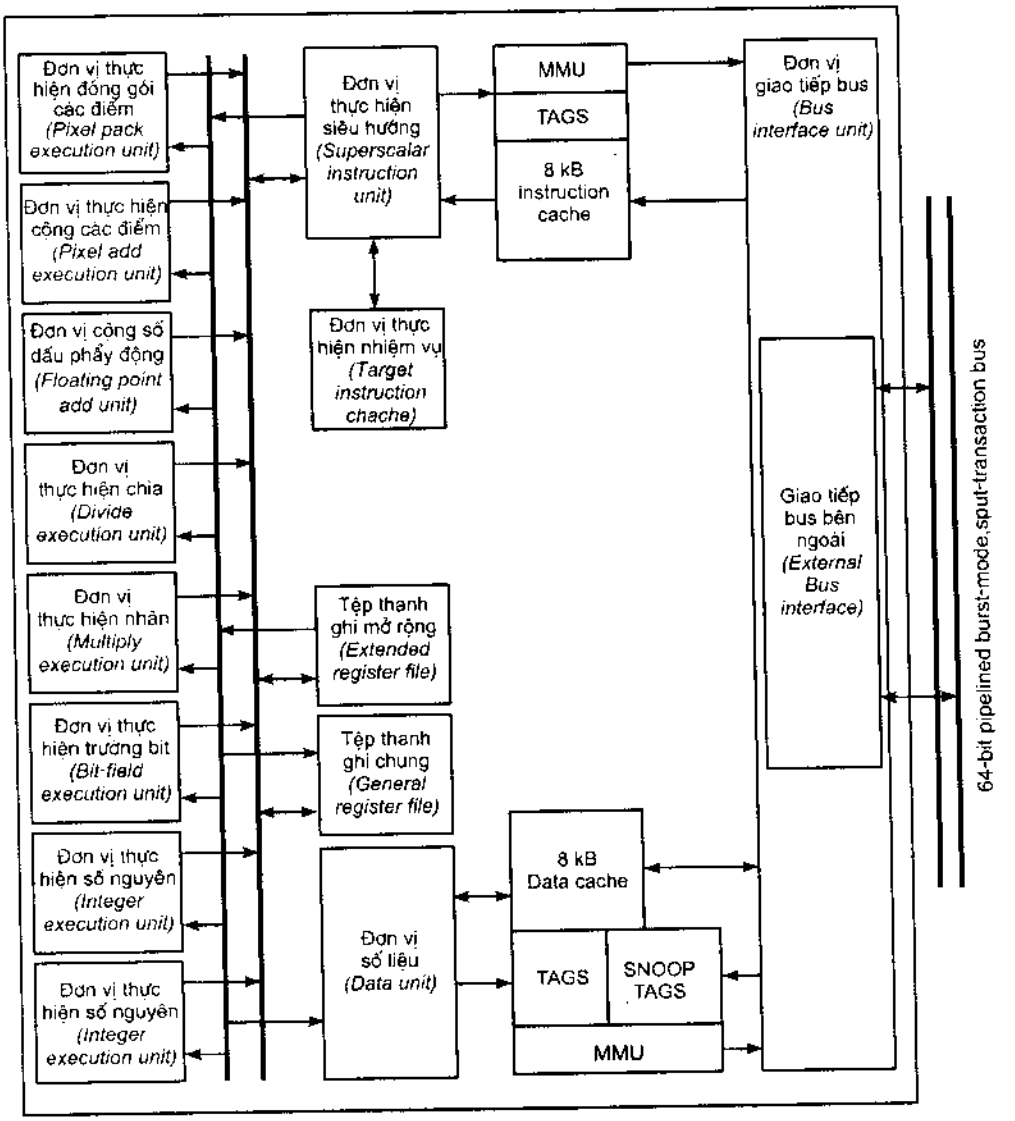
Đường ống lệnh dấu phẩy động có 6 giai đoạn vì thay vào giai đoạn E là các giai đoạn E1, E2, và E3.

Kỹ thuật song song ở mức lệnh (ILP) được đảm bảo nhờ thực hiện đồng thời cả lệnh với số nguyên và lệnh với số dấu phẩy động trong chế độ song song 2-lệnh (dual-instruction mode). Trong trường hợp này i860 vận hành như là siêu hướng 2-đường ra (2-issue superscalar). Trong chế độ song song 2 lệnh, một chuỗi lệnh 64-bit với sắp xếp: lệnh dấu phẩy động (32 bit) nằm ở 32 bit thấp và lệnh số nguyên (32 bit) nằm ở 32 bit cao của chuỗi lệnh.

Intel i860 XP được ứng dụng chủ yếu trong các hệ thống đa xử lý, như Alliant 2800, Paragon. Hiệu suất của nó cạnh tranh với các kiến trúc SuperSPARC của Sun.

7.4.2 Họ vi xử lý Motorola M88000

Tương tự như Intel, Motorola bắt đầu kiến trúc RISC bằng họ vi xử lý M88000 sau vài năm phổ biến họ M68000. Các phiên bản đầu tiên của họ này là MC88100 CPU và MC88200 cache MMU (CMMU). Tiếp sau là RISC MC88110.



Hình 7.36: Sơ đồ khối của MC88110

MC88110 có công nghệ: đong vỏ 299 chân, 0,8 micron CMOS, 1,3 triệu transistor. Nó là hệ thống siêu hướng 2-đường ra vơi 10 đơn vị thực hiện (execution units). Sơ đồ khối của MC88110 mo tả ở hình 7.36.

Vi xử lý MC88100 đóng vỏ 180-chân HCMOS với FPU bên trong. MC88200 có bên trong 16 kB cache, và kết hợp với MC88100 tạo thành một hệ thống vi xử lý RISC tối thiểu. Một cấu hình tối thiểu gồm 2 CMMU nối với MC88100. Một CMMU tác động như một Dcache 16 kB, và CMMU khác tác động như là một Icache 16 kB, làm thành cache đôi 32 kB. Có thể kết nối tới 4 CMMU để làm thành 128 kB cache (64 kB Dcache và 64 kB Icache).

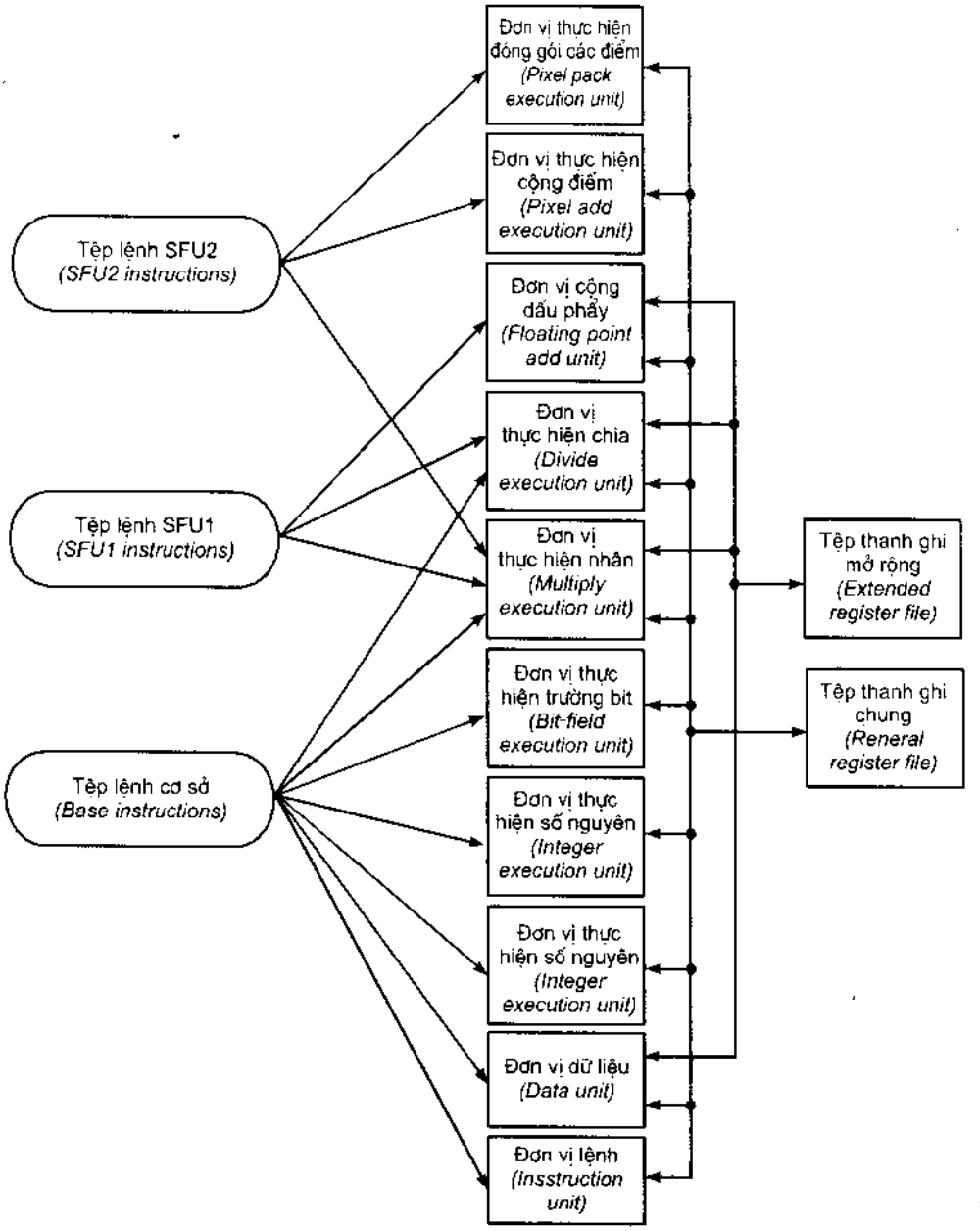
Các đơn vị thực hiện là: số nguyên (integer, 2), rẽ nhánh (branch, 1), trường bit (bit-field, 1), đường ống nhân (multiply pipelined, 1), đường ống cộng dấu phẩy động (floating add, 1), chia (divide, 1), đồ họa (graphics, 2), nạp/cất giữ (load/store, 1).

MC88110 xác định tập hợp các mã lệnh chính như là một đơn vị đặc biệt (SFU), trong đó, tất cả các lệnh dấu phẩy động được mã hóa trong SFU1, và tất cả các lệnh đồ họa mã hóa trong SFU2. Các đơn vị SFU3 đến SFU7 để dự phòng cho mở rộng (hình 7.37).

Các vi xử lý của họ MC88000 có tệp thanh ghi bên trong (mô hình lập trình) gồm:

1. Tệp thanh ghi chung GRF (General Register File): gồm 32 thanh ghi 32-bit (R0-R31). Thanh ghi R0 được gán cứng bằng 0. Thanh ghi R1 chứa con trỏ trở về của chương trình con. Nội dung của R1 có thể ghi vào/đọc ra.

2. Tệp thanh ghi mở rộng XRF (eXtended Register File): gồm 32 thanh ghi 80-bit (X0-X31) mở rộng để cất giữ các toán hạng dấu phẩy động thỏa mãn chuẩn IEEE 754-1985 chính xác đơn 32-bit, chính xác kép 64-bit, hoặc chính xác mở rộng. Thanh ghi X0 được gán cứng bằng 0.



Hình 7.37: Sử dụng phần cứng SFU trong MC88110

3. Thanh ghi điều khiển dấu phẩy động 32-bit FPCR (Floating-Point Control Register).

4. Thanh ghi trạng thái dấu phẩy động 32-bit FaiSR (Floating-point Status Register).

5. Các thanh ghi điều khiển CR0-CR5, 32-bit dùng trong chế độ giám sát.

6. Thanh ghi nguyên nhân ngoại lệ dấu phẩy động FCR0 32-bit (FPECR) dùng trong chế độ giám sát.

Họ MC88000 có các kiểu dữ liệu:

1. Nguyên (Integer): byte 8-bit, nửa từ 16-bit, từ 32-bit, từ kép 64-bit.

2. Trường bit (bit-field): từ 1 đến 32 bit trong thanh ghi 32-bit.

3. Dấu phảy động: chính xác đơn 32-bit, chính xác kép 64-bit, chính xác mở rộng gấp đôi 80-bit.

4. Đồ họa: đóng gói 32-bit theo các nibbles (1 nibble = 4 bit), các byte, nửa từ; đóng gói 64-bit theo các nibbles, các byte, nửa từ và từ.

Họ MC88000 có các chế độ đánh địa chỉ phân theo 3 loại:

1. Tính toán
   a. Thanh ghi (chung, mở rộng, điều khiển)
   b. Ngay lập tức (6-bit, 10-bit, 16-bit)
2. Nạp/cất giữ/trao đổi (Load/store/exchange)
   a. Gián tiếp thanh ghi với độ dịch ngay lập tức
   b. Gián tiếp thanh ghi với chỉ số
   c. Gián tiếp thanh ghi với chỉ số cấp
3. Điều khiểu luồng (flow control)
   a. Gián tiếp thanh ghi
   b. Gián tiếp thanh ghi với chỉ số của bảng véc-tơ 9-bit
   c. Tương đối PC với độ dịch 16-bit
   d. Tương đối PC với độ dịch 26-bit

Họ MC88000 có tập lệnh giảm thiểu gồm 66 lệnh và được chia thành 7 nhóm:

1. Số học nguyên (integer arithmetic)	9 lệnh
2. Trường bit (bit-field)	8 lệnh
3. Logic	4 lệnh
4. Đồ họa (graphics)	9 lệnh
5. Điều khiển luồng (flow control)	13 lệnh
6. Nạp/cất giữ/trao đổi (load/store/exchange)	7 lệnh
7. Dấu phảy động (floating point)	16 lệnh

Họ MC88000 có các lệnh tham chiếu thanh ghi-đến-thanh ghi 3 toán hạng.

MC88110 có cache đôi 16 kB (8 kB Icache, 8 kB Dcache). Mỗi cache được đánh địa chỉ vật lý, 2-đường liên hợp tệp (2-way set-associative) dung lượng 32 Byte trên một đường (32 byte/đường). Như vậy cần 5 bit cho trường đường (line), 7 bit cho trường tập hợp (set), và 20 bit cho tags). Điều này đúng để đánh địa chỉ 4 kB cho một trang = 2^{12} byte/trang. MC88110 cũng có thể kết nối với cache bên ngoài, mà cache bên ngoài được điều khiển bởi mạch điều khiển cache MC88410. cache bên ngoài có thể dùng các chip SRAM 62110. Dung lượng cache ngoài có thể đạt từ 256 kB đến 1 MB và dùng để sắp xếp trực tiếp, ghi lại với 32 hoặc 64 byte/đường. MC88110 có đơn vị quản lý bộ nhớ lệnh IMMU (Instruction Memory Management Unit) và đơn vị quản lý bộ nhớ dữ liệu DMMU (Data Memory Management Unit). IMMU kết hợp với Icache thành một đơn vị nhớ lệnh IMU (Instruction Memory Unit) và DMMU kết hợp với Dcache thành đơn vị bộ nhớ dữ liệu DMU (Data Memory Unit). MC88110 có bộ đệm lịch sử (history buffer) để ghi trạng thái máy có liên quan đến thực hiện lệnh, và CPU sử dụng nội dung của thanh ghi này để trở lại điểm thực hiện lệnh bị dừng.

Họ MC88000 có 512 vector ngoại lệ, cất giữ trong bảng vector, và được trỏ tới nhờ thanh ghi cơ sở véc-tơ VBR (Vector Based

Register), đó là thanh ghi CR7. Các véc-tơ có số hiệu từ 0 đến 127 dành cho phần cứng và nhu cầu sử dụng ở chế độ giám sát. 384 véc-tơ còn lại (từ 128 đến 511) dùng cho các bẫy phần mềm (software traps). Mỗi một véc-tơ ngoại lệ chứa 2 lệnh, thường là: một lệnh rẽ nhánh để thực hiện chương trình con ngoại lệ, và lệnh thứ 2 là lệnh đầu tiên của chương trình xử lý ngoại lệ. Kích thước của mỗi một véc-tơ ngoại lệ là 8 Byte, do đó bảng véc-tơ các ngoại lệ có dung lượng là:

$$512 \times 8 = 4 \text{ kB}$$

Do có kiến trúc siêu hướng 2 đường ra (2-way issue) mà xử lý song song đạt được ở đây với sự thực hiện đồng thời 2 lệnh.

Họ MC88000 được ứng dụng trong một số hệ thống xử lý song song đã được thương mại, ví dụ, như hệ thống BBN Butterfly và Encore.

7.4.3 Họ vi xử lý SUN SPARC

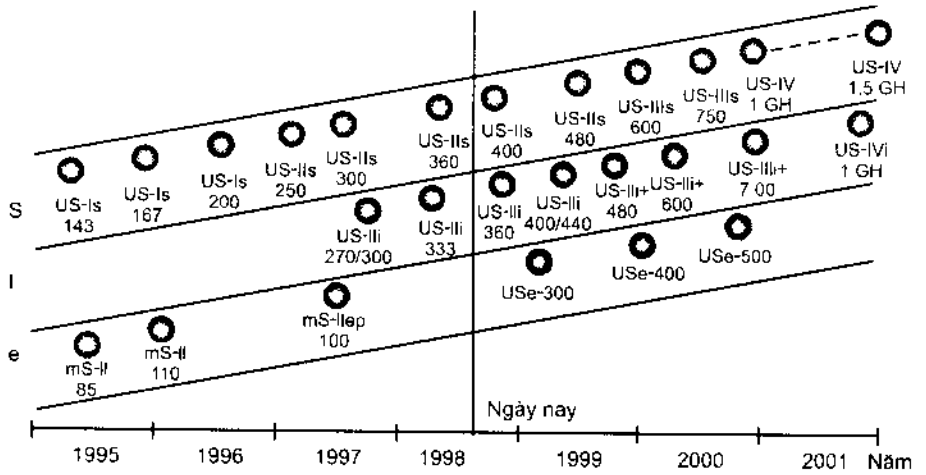
a) Giới thiệu về SUN SPARC

Kiến trúc SPARC được tạo ra bởi hãng máy tính SUN. Trước thế hệ vi xử lý SPARC, SUN đã có sản phẩm nổi tiếng là họ M68000. SUN là một trong những nhà chế tạo các bộ vi xử lý kiểu RISC đầu tiên. Các chip do SUN thiết kế được sản xuất tại Mĩ, châu Âu và Nhật Bản. Thực tế sản phẩm đầu tiên đã được sản xuất tại Japanese Fujitsu Microelectronics inc. Các loạt vi xử lý SuperSPARC gần đây là sản phẩm hợp tác giữa Sun microsystems và Texas Instrumentions (TI) ở Houston, Texas. Các loạt vi xử lý Tên của SPARC có nghĩa là Scalable Processor Architecture (kiến trúc xử lý hướng). Nét nổi bật của SPARC là sử dụng các file thanh ghi của CPU và các cửa sổ thanh ghi giống nhau.

Sự phát triển của công nghệ SUN SPARC được mô tả trong hình 7.38.

Trong những năm gần đây, do đòi hỏi của nhu cầu xử lý đa phương tiện mới (new-media): ảnh 2D, 3D, phát triển cơ sở dữ liệu,

công nghệ audio, video,... kiến trúc của các loại vi xử lý đã thay đổi rất nhanh, trong đó đã ra đời loại vi xử lý UltraSPARC của SUN với các tập lệnh VIS (Visual Instruction Set) cho xử lý đa phương tiện mới. Loại vi xử UltraSPARC gồm:



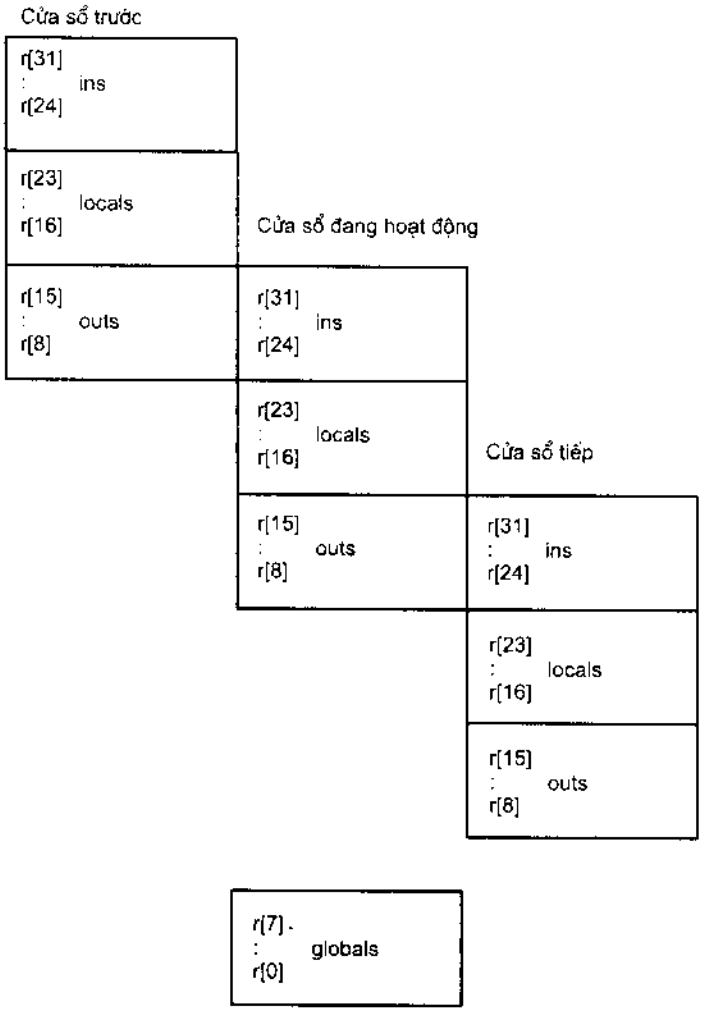
Hình 7.38: Sự phát triển của công nghệ SUN SPARC

- UltraSPARC-I 167 MHz, và 200 MHz
- UltraSPARC-II 250 MHz, 300 MHz, 333 MHz, 360 MHz, 400 MHz, 450 MHz, và 480 MHz
- UltraSPARC-IIi 64-bit RISC với PCI interface
- UltraSPARC-III scalable 600 MHz multi-processing (đa xử lý)
- 16 kB Dcache bên trong, 16 kB Icache bên trong
- Có bộ điều khiển cache thứ cấp (L2) hỗ trợ kết nối đến 2 MB SRAM đồng bộ
- Hỗ trợ bộ nhớ EDO Dram tốc độ 400 MB/s
- Hỗ trợ giao tiếp PCI cho I/O và DMA
- Nguồn cung cấp là 2,5V cho chip và 3,3V I/O
- Công nghệ CMOS 0,35micron
- Đóng vỏ gốm (ceramic) 587-chân LGA kích thước 37,50 (±0,20) × 34,00 (±0,05) × 4,05 (±0,3) mm

b) Kiến trúc của SPARC

(1) Tệp thanh ghi

Đặc điểm của kiểu SPARC là có hơn 100 thanh ghi trong tệp thanh ghi. Tệp thanh ghi được chia nhỏ thành các nhóm thanh ghi, mỗi nhóm được gọi là 1 cửa sổ thanh ghi (Register window). Các cửa sổ có thể được chồng lên nhau (tức là 2 cửa sổ liên tiếp có thể chung một số thanh ghi nhất định).



Hình 7.39: 3 cửa sổ chồng nhau và các thanh ghi tổng thể

Mọi thủ tục chạy trong SPARC chỉ có thể truy nhập tới 32 thanh ghi, được biểu diễn từ R0 đến R31. Tám thanh ghi từ R0 đến R7 là các thanh ghi tổng thể (global) mà tất cả các thủ tục có thể truy nhập vào, còn 24 các thanh ghi khác là các thanh ghi cửa sổ (windows register), được gán cho từng thủ tục. Giữa hai cửa sổ có chung 8 thanh ghi. 24 thanh ghi cửa sổ được chia thành 3 nhóm, mỗi nhóm 8 thanh ghi như chỉ ra ở hình 7.39.

Các thanh ghi được chia nhỏ thành các nhóm là:

- r31 đến r24 ins, chứa các tham số được đưa tới các thủ tục nhờ thủ tục gọi;

- r23 đến r16 locals, chứa các tham số cục bộ của thủ tục;

- r15 đến r8 outs, chứa các tham số được đưa tới các thủ tục được gọi.

Các thanh ghi outs của thủ tục gọi, về vật lý chính là các thanh ghi ints trong thủ tục được gọi. Thủ tục gọi đưa các tham số đến các thủ tục được gọi thông qua các thanh ghi outs. Cửa sổ thanh ghi của thủ tục hiện đang chạy gọi là *cửa sổ* hoạt động (active window), được trỏ tới bởi con trỏ cửa sổ hiện hành CWP (Current Window Pointer). CWP nằm trong thanh ghi trạng thái (status Register) của bộ xử lý.

Số các cửa sổ (NWINDOW) được sử dụng trong các phiên bản khác nhau của SPARC có từ 2 đến 32, các thanh ghi đa năng của đơn vị nguyên IU (general - purpose IU registers) kể cả 8 thanh ghi tổng thể (global) có từ 48 đến 548. Hầu hết các bộ vi xử lý SPARC hiện nay có 8 cửa sổ với 136 thanh ghi, các cửa sổ được đánh thứ tự liên tiếp từ 0 đến NWINDOW-1.

Đơn vị xử lý trung tâm (CPU) chứa thanh ghi điều khiển 32-bit được gọi là mặt nạ của cửa số sai WIM (Window Invalid Mask), mỗi bit của WIM, Wi (i = 1, 2, 3,..., 31) tương ứng với một trong 32 cửa sổ (số cửa sổ có thể ít hơn). Nếu Wi = 1 cửa sổ thứ i được coi là sai (invalid), và sẽ có điều kiện bẫy (trap condition).

Thanh đếm chương trình PC của CPU là một thanh ghi riêng, không nằm trong tệp thanh ghi đa năng. SPARC có thể có vài thanh ghi PC chứa các địa chỉ các lệnh tuần tự. Trong SPARC, một số thanh ghi của IU (Interger Unit) có những nhiệm vụ riêng biệt:

Thanh ghi R0 chứa giá trị 0, giống như trong nhiều hệ thống vi xử lý kiến trúc RISC khác. Lệnh CALL ghi địa chỉ của chính nó vào thanh ghi outs r15.

Con trỏ cửa sổ hiện hành (CWP) sẽ bị giảm trong lệnh SAVE khi gọi đến thủ tục và tăng lên nhờ lệnh RESTORE khi trở về từ thủ tục. Các thủ tục cũng có thể được gọi mà không cần thay đổi cửa sổ. Giả sử trong trường hợp số cửa sổ bằng 8, và cửa sổ 0 là cửa sổ hoạt động hiện hành. Trong trường hợp này, CWP = 0. Nếu cửa sổ 0 là cửa sổ trống cuối cùng, khi một thủ tục sử dụng cửa sổ 0 để gọi một thủ tục khác, thì một sự tràn cửa sổ (window overflow) sẽ xảy ra. Khi một cửa sổ thanh ghi mới ghi đè lên cửa sổ được sử dụng trước đó (cửa sổ 7), toàn bộ nội dung phải được ghi vào bộ nhớ nhờ phần mềm. Sau khi trở về từ thủ tục, và khi tệp thanh ghi đã ở ngoài các cửa sổ (out of windows), chúng ta có một cửa sổ tràn dưới. Trong trường hợp này bằng phần mềm phải phục hồi các cửa sổ thanh ghi được sử dụng trước đó. Bẫy xử lý tràn sử dụng phần locals của cửa sổ 7 để con trỏ tới vùng nhớ nơi cửa sổ tràn được cất giữ. Cửa sổ 7 bị hỏng trong quá trình bẫy xử lý nhờ thiết lập w7 của thanh ghi WIM.

(2) Các kiểu dữ liệu

Kiến trúc SPARC sử dụng các kiểu dữ liệu sau đây (hình 7.40):

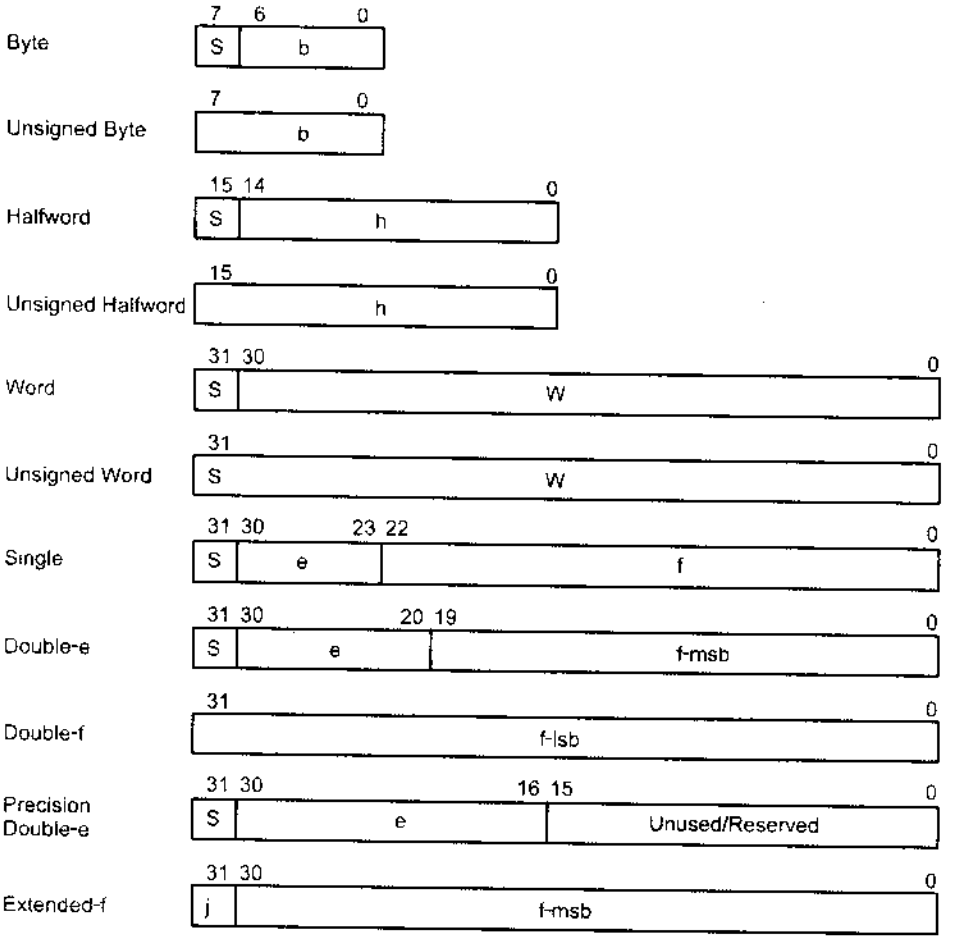
- Integer:

  Byte có dấu, không dấu (8 bit)

  Nửa từ (halfword) có dấu, không dấu (16 bit).

  Từ (word) có dấu, không dấu 32 bit.

  Từ kép (double word), 64 bit



Hình 7.40: Các kiểu dữ liệu sử dụng trong SPARC

- Floating-point (chuẩn IEEE 754 - 1985):

  Số chính xác đơn, 32 bit.

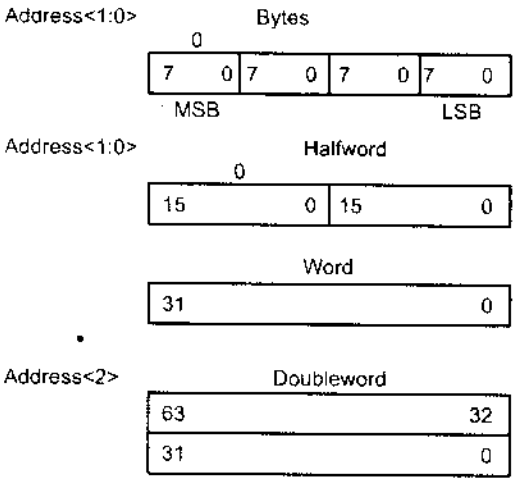
  Số chính xác kép, 64 bit.

  Số mũ chính xác bốn (quad-precision): 15 bit, định trị: 63 bit.

Nửa từ (halfword) được sắp xếp trên biên địa chỉ 2-byte (địa chỉ chẵn, có thể chia được cho 2, tức là nếu địa chỉ của halfword đầu là 100, thì địa chỉ của halfword tiếp theo là 102).

Tương tự, địa chỉ của từ (word) có thể chia được cho 4, của từ kép doubleword có thể chia được cho 8.

Sự lưu giữ các loại dữ liệu trong bộ nhớ được mô tả trong hình 7.41 sau đây:

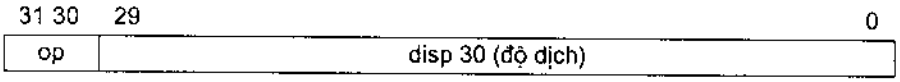


Hình 7.41: Lưu giữ dữ liệu trong bộ nhớ của SPARC

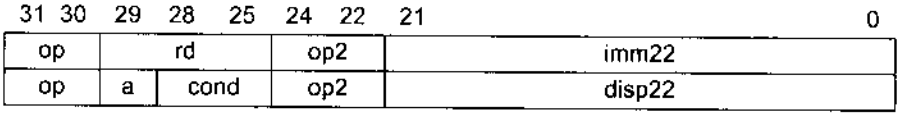
(3) Các dạng lệnh

- SPARC có ba nhóm lệnh cơ bản:
- Lệnh gọi CALL
- Lệnh rẽ nhánh
- Lệnh thao tác (giữa các thanh ghi)

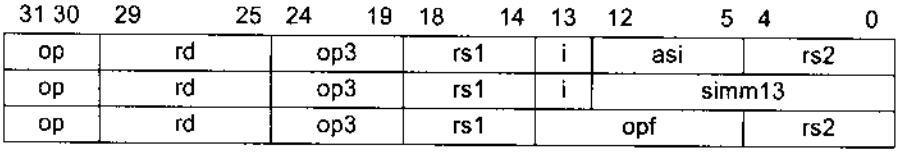
Dạng 1: lệnh CALL



Dạng 2: Lệnh SETHI và các lệnh rẽ



Dạng 3: Các lệnh còn lại:



Hình 7.5: Các khuôn dạng lệnh của SPARC

Mã lệnh (op: bit 31, 30) với tất cả các dạng, được phân tích như sau:

Mã lệnh (op)	Các lệnh
01	CALL
00	Bicc, FBcc, CBccc, SETHI
10 hoặc 11	Tất cả các lệnh khác

Từ bit 24 đến 22 là mã lệnh op2 cho dạng 2, chúng mã hóa để chọn các lệnh như sau:

Mã lệnh 2	Lệnh
000	UNIMP
010	Bicc
100	SETHI
110	FBfcc
111	CBccc

- rd: từ bit 29 đến 25 trong dạng 2, 3. Chọn thanh ghi nguồn cho các lệnh cất giữ (store instructions) và thanh ghi đích cho tất cả các lệnh khác.

- a: bit 29 trong dạng 2, annul bit, chuyển trạng thái của lệnh ngay sau khi chuyển điều khiển (control transfer)

- Cond: từ bit 28 đến 25 trong dạng 2, chọn mã điều kiện cho các lệnh rẽ nhánh có điều kiện.

- imm22: từ bit 21 đến 0 trong dạng 2, một hằng số 22-bit được sử dụng bởi lệnh SETHI

- disp22: từ bit 21 đến 0 trong dạng 2, độ dịch disp mở rộng dấu đến 22 bit trong lệnh rẽ nhánh.

- disp30: từ bit 29 đến 0 trong dạng 1, độ dịch disp mở rộng dấu đến 30 bit cho các lệnh gọi

- op3: từ bit 24 đến 19 trong dạng 3, mở rộng của mã lệnh

- i: bit 13 trong dạng 3, chọn kiểu cho toán hạng ALU thứ hai trong các lệnh không phải dấu phẩy động.

+ i = 0: toán hạng thứ hai ở trong thanh ghi rs2.

+ i = 1: toán hạng thứ hai là phần mở rộng dấu của simm13

- asi: từ bit 12 đến 5 trong dạng 3, địa chỉ 8-bit (an 8 bit address space identifier) được tạo bởi các lệnh nạp/cất giữ

- rs1: bit 18 đến 14 trong dạng 3, chọn thanh ghi toán hạng nguồn thứ 1.

- rs2: Bit 4 đến 0 trong dạng 3, chọn toán hạng nguồn thứ 2.

- Simm13: bit 2 đến 0 trong dạng 3, một giá trị tức thời mở rộng dấu đến 13 bit

- opf: bit 13 đến 5 trong dạng 3, xác nhận một lệnh thao tác dấu phẩy động.

(4) Các chế độ địa chỉ

Ngoài các chế độ địa chỉ trực tiếp thanh ghi chuẩn (standard register direct) và các chế độ đánh địa chỉ ngay lập tức (immediate addressing modes), SPARC còn có 3 chế độ địa chỉ khác:

1. Gián tiếp thanh ghi với độ dịch (register indirect with disp); register + signed 13- bit constant.

2. Gián tiếp thanh ghi chỉ số (register indirect indexed); register1 + register2.

3. Tương đối thanh đếm lệnh (PC- relative), được sử dụng trong lệnh CALL cùng với độ dịch 30-bit.

(5) Tập lệnh

Kiến trúc SPARC có các kiểu lệnh sau:

1. Nạp/cất giữ.

2. Số học/logic/dịch.

3. Chuyển điều khiển (control transfer).

4. Đọc/ghi/các thanh ghi điều khiển

5. Thao tác dấu phẩy động.

6. Thao tác đồng xử lý (nhóm lệnh này không cần thiết cho các bộ vi xử lý hiện nay).

Tập lệnh của SPARC liệt kê trong bảng 7.1.

Bảng 7.1: Tập lệnh của Sun SPARC

Mã lệnh	Tên
Ticc	Trap on integer condition codes
RDY	Read Y register
RDPSR*	Read Processor State Register
RDWIM*	Read Window Invalid Mask register
RDTBR*	Read Trap Base Register
WRY	Write Y register
WRPSR*	Write Processor State Register
WRWIM*	Write Window Invalid Mask register
WRTBR*	Write Trap Base Register
UNIMP	Unimplemented instruction
IFLUSH	Instruction cache Flush
FPop	Floating-point Operate: FiTO(s,d,x), F(s,d,x)TOi FsTOd, FsTOx, FdTOs, FdTOx, FxTOs, FxTOd, FMOVs, FNEGs, FABs, FSQRT(s,d,x), FADD(s,d,x) FSUB(s,d,x), FMUL(s,d,x) FDIV(s,d,x), FCMP(s,d,x) FCMPE(s,d,x)
CPop	Coprocessor operate
LDSB (LDSBA*)	Load Signed Byte (from Alternate space)
LDSH (LDSHA*)	Load Signed Halfword (from Alternate space)
LDUB (LDUBA*)	Load Unsigned Byte (from Alternate space)
LDUH (LDUHA*)	Load Unsigned Halfwod (from Alternate space)
LD (LDA*)	Load Word (from Alternate space)
LDD (LDDA*)	Load Doubleword (from Alternate space)
LDF	Load Floating-point
LDDF	Load Double Floating-point
LDFSR	Load Floating-point State Register
LDC	Load Coprocessor
LDDC	Load Double Coprocessor
LDCSR	Load Coprocessor State Register

Mã lệnh	Tên
STB (STBA*)	Store Bytes (into Alternate space)
STH (STHA*)	Store Halfword (into Alternate space)
ST (STA*)	Store Word (into Alternate space)
STD (STDA*)	Store Doubleword (into Alternate space)
STF	Store Floating-point
STDF	Store Double Floating-point
STFSR	Store Floating-point State Register
STDFQ*	Store Double Floating-Point Queue
STC	Store Coprocessor
STDC	Store Double Coprocessor
STCSR	Stare Coprocessor State Register
STDCQ*	Store Double Coprocessor Queue
LDSTUB (LDSTUBA*)	Atomic Load-Sture Unsigned Byte (in Alternate space)
SWAP (SWAPA*)	Swap r Register with Memory (in Alternate space)
ADD(ADDcc)	Add (and modify icc)
ADDX (ADDXcc)	Add with Carry (and modify icc)
TADDCC (TADDccTV)	Tagged Add and modify icc (and Trap on overflow)
SUB (SUBcc)	Subtract (and modify icc)
SUBX (SUBXcc)	Subtract with Carry (and modify icc)
TSUBcc (TSUBccTV)	Tagged Subtract and modify icc (and Trap on overflow)
MULScc	Multiply Step and modify icc
AND (ANDcc)	And (and modify icc)
ANDN (ANDNcc)	And Not (and modify icc)
OR (ORcc)	Inclusive-Or (and modify icc)
ORN (ORNcc)	Inclusive-or Not (and modify icc)
XOR (XORcc)	Exclusive-Or (and modify iec)
XNOR (XNORcc)	Exclusive-Nor (and modify icc)
SLL	Shift Left Logical
SRL	Shirt Right Logicai
SRA	Shift Right Arithmetic
SETHL	Set High 22 bits or register
SAVE	Save caller's window
RESTORE	Restore caller's window
Bicc	Branch on integer condition codes

Mã lệnh	Tên
FBRcc	Branch on floating-point condition codes
CBCcc	Branch on coprocessor condition codes
CALL	Call
JMPL	Jump and Link
RETT*	Return from Trap

Các lệnh nạp/cất giữ (chỉ dùng một loại để truy nhập bộ nhớ), sinh ra địa chỉ 32-bit cho byte. Cộng thêm địa chỉ này, vi xử lý luôn luôn tạo ra một xác định (identifier) độ dài 9-bit cho không gian địa chỉ, gọi là asi (address space identifier). Giá trị của asi (theo hệ mười) có ý nghĩa như sau:

asi (Hệ 10)	Gán (assignment)
0-7	Thực hiện - có xác định.
8	Không gian lệnh của người sử dụng.
9	Không gian lệnh của người giám sát
10	Không gian dữ liệu của người sử dụng
11	Không gian dữ liệu của người giám sát
12-255	Bổ sung định nghĩa

Ví dụ:

LDSB addr, rd ; nạp byte có dấu từ bộ nhớ tại địa chỉ addr vào thanh ghi Rd

LD addr, rd ; nạp word từ địa chỉ addr vào thanh ghi rd

Hầu hết các lệnh số học và logic đều có từng cặp lệnh (như ADD and ADDcc) để sửa đổi mã điều kiện số nguyên (icc) như một ảnh hưởng phụ (side effect).

Ví dụ:

ADD rs1, rs2,rd; (rs1)+(rs2) → rd, không sửa icc

ADDcc rs1, rs2, rd; (rs1) + (rs2) → rd, sửa icc

Trong một vài trình dịch hợp ngữ của SPARC, các thanh ghi có thể được biểu thị như sau:

%0 đến	%31	tất cả các thanh ghi chung từ r0 đến r31
%g0 đến	%g7	các thanh ghi tổng thể giống như %0 đến %7
%00 đến	% 07	thanh ghi outs, giống như %8 đến %15
%10 đến	%17	thanh ghi local giống như %16 đến %23
%i0 đến	%i7	thanh ghi ins giống như %24 đến %31

Ví dụ:

```
SUB%12, %11,    %15   ;(r12) - (r11)→ r15
AND%5,  %3,     %1    ;r5) AND (r3) → r1
```

Lệnh SETHI (set high 22 bit of rd) ghi 1 hằng 22 bit từ lệnh vào bit cao của thanh ghi đích rd. Nó xoá 10 bit bậc thấp của rd và không thay đổi mã điều kiện.

Nó được sử dụng trong dạng sau (dạng 2):

SETHI const22, rd;const22 là một hằng số 22 bit

Có 5 kiểu lệnh truyền có điều khiển:

1. Rẽ nhánh có điều kiện (Bicc, Fbfcc, CBccc)
2. Nhảy và liên kết
3. Call (CALL)
4. Bẫy (Ticc)
5. Trở về từ bẫy (RETT).

Mỗi lệnh trên còn được phân nhỏ thành nhiều loại, theo chế độ đánh địa chỉ (PC- relative, Register indirect) hoặc theo cách thi hành lệnh (delayed or non-delayed).

Trong lệnh truyền có điều khiển, địa chỉ đích được tính bằng cách cộng giá trị của PC với độ dịch disp. Một lệnh truyền có điều khiển sẽ bị trễ nếu nó nhảy tới địa chỉ đích sau một thời gian trễ. Bit a (annul) trong dạng 2 ảnh hưởng đến sự thi hành của lệnh trễ. Nó có ảnh hưởng như sau:

a	Kiểu rẽ nhánh	Lệnh trễ có thực hiện hay không?
1	Không điều kiện	No
	Có điều kiện, thỏa mãn	Yes
	Có điều kiện, không thỏa mãn	No
0	Không điều kiện	Yes
	Có điều kiện, thỏa mãn	Yes
	Có điều kiện, không thỏa mãn	Yes

Một thủ tục sử dụng cửa sổ thanh ghi được gọi bằng cách thực hiện đồng thời hai lệnh CALL (hoặc JMPL) và SAVE. Một thủ tục không cần cửa sổ thanh ghi, gọi là leaf routine và được gọi nhờ thực hiện lệnh CALL (hoặc JMPL). Leaf routine có thể chỉ sử dụng các thanh ghi outs. Lệnh CALL cất giữ nội dung PC vào thanh ghi r15 (thanh ghi outs), mà nội dung PC này trỏ tới chính lệnh CALL. Lệnh JMPL (lệnh nhảy và liên kết) cất PC vào thanh ghi chỉ định (specifield Register).

Ví dụ:

```
JMPL      addr,rd        ; PC được cất trong rd
```

Lệnh SAVE tương tự như lệnh ADD, ngoại trừ một điều là con trỏ cửa sổ hiện hành CWP của nó bị giảm đi 1, làm cho cửa sổ hoạt động trở thành cửa sổ trước đó,bằng cách ấy ghi lại cửa sổ gọi (caller's window).

Ví dụ:

```
SAVE rs1,rs2,rd; các toán hạng rs1, rs2 từ cửa sổ trước, rd ở
                cửa sổ mới, được đánh địa chỉ bởi giá trị
                mới trong CWP.
```

Kiến trúc SPARC có một số lệnh hỗ trợ như sau:

- SWAP: trao đổi nội dung của thanh ghi IU với một từ của bộ nhớ, trong khi ngăn chặn các truy nhập bộ nhớ khác. SWAP là một lệnh nguyên tử (atomic instruction - lệnh atomic là lệnh không thể ngắt, tức là trong thời gian thi hành lệnh atomic, không một bộ xử lý nào có thể truy nhập vào tài nguyên của nó).

- LSTUB: nạp và cất byte không dấu, đọc một byte từ bộ nhớ vào một thanh ghi IU sau đó ghi lại byte đó ở trong bộ nhớ tất cả giá trị 1, trong khi ấy, không cho quá trình khác can thiệp vào.

Tập lệnh của SPARC liệt kê trong bảng 7.1

c) Kiến trúc của Super SPARC

(1) Đặc điểm và sơ đồ khối của SuperSPARC

SuperSPARC là một chip gồm 3,1 triệu transitor, 0,8 micron, 3 lớp kim loại, công nghệ BiCMOS, đóng vỏ 293-chân PGA (Pin Grid Array), được hợp tác sản xuất giữa Texas Instrumentions (TI) và SUN. Chip gồm có các đơn vị IU, FPU, MMU, và cache đôi bên trong chip (20 kB Icache, 16 kB Dcache). Sơ đồ của Super SPARC được trình bày ở hình 7.42.

Chip vi xử lý Super SPARC có thể thực hiện đồng thời ba lệnh trong một chu kì của đường ống theo nguyên tắc sau:

1. Tối đa là hai kết quả nguyên

2. Tối đa là một tham chiếu dữ liệu bộ nhớ

3. Tối đa là một lệnh số học dấu phẩy động

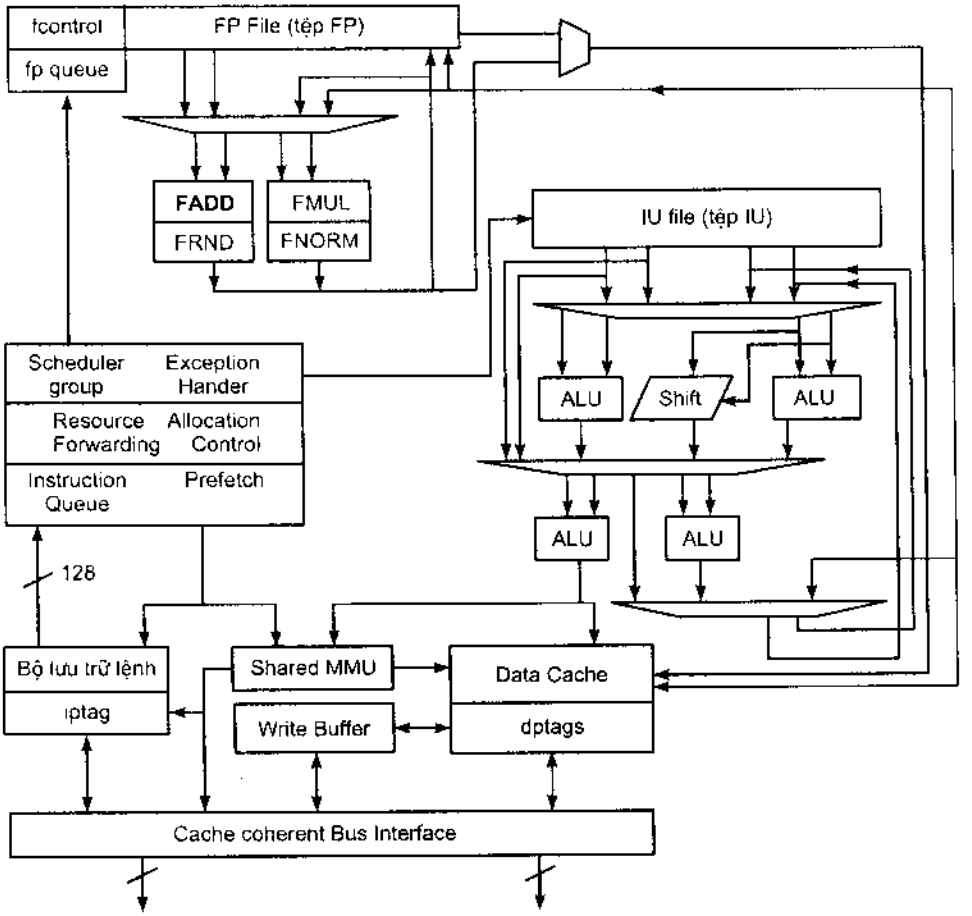
4. Nhóm kết thúc các lệnh sau từng chuyển điều khiển (Terminate group of ins after arithmetic control transfer).

Trong một đường ống, có nhiều lệnh cùng được xử lý. Giả thiết có hai lệnh liên tiếp mà kết quả của lệnh này là toán hạng nguồn của lệnh kia thì dữ liệu đó gọi là dữ liệu phụ thuộc (data dependency). Nếu kết quả ở lệnh đầu sinh ra sau chu kỳ thực hiện lệnh thứ hai, thì lệnh thứ hai sẽ có kết quả sai. Đây được gọi là lỗi đường ống (hazard pipeline). Vấn đề hazard (chướng ngại) của đường chúng ta đã đề cập đến trong mục 7.2.6.

Super SPARC giải quyết các dữ liệu phụ thuộc bằng cách:

1. Phân cấp các lệnh phụ thuộc trong cùng một nhóm.

2. Gởi các lệnh phụ thuộc trong các nhóm liên tiếp.



Hình 7.42: Sơ đồ khối chức năng của SuperSPARC

Trong cấu trúc của đơn vị IU gồm có bốn đơn vị ALU và một đơn vị dịch (Shifter). Đơn vị ALU bên dưới phía trái dùng để tính địa chỉ. Đầu ra của nó được gửi tới MMU và Dcache. Ba đơn vị ALU còn lại được sử dụng cho xử lý dữ liệu. Nếu có hai lệnh nguyên mà có dữ liệu phụ thuộc, thì lệnh đầu tiên có thể được xử lý bằng một đơn vị ALU bên trên, sau đó gửi kết quả và toán hạng khác vào đơn vị ALU bên dưới, hoàn thành hai phép toán trong một chu kì, sau đó cả hai kết quả được cất trong thanh ghi của IU.

(2) Các đường ống của SuperSPAR

Đường ống nguyên IU của SPARC gồm có bốn giai đoạn, mỗi giai đoạn có hai pha, vậy nó gồm có 8 pha:

1. F0: truy nhập Icache và tìm kiếm TLB

2. F1: tìm thấy trong Icache lệnh phù hợp, bốn lệnh được gửi đến hàng lệnh.

3. D0: đưa ra 1, 2 hoặc 3 lệnh. Chọn các thanh ghi cho lệnh cất và nạp

4. D1: đọc các tệp thanh ghi cho các lệnh cất nạp, chỉ định nguồn cho các lệnh ALU, tính giá trị địa chỉ rẽ nhánh đích.

5. D2: đọc tệp thanh ghi cho các toán hạng ALU, tính địa chỉ hiệu dụng EA (Effective Address) cho các lệnh nap và cất

6. E0: giai đoạn đầu tiên của ALU, truy nhập Dcache, tìm kiếm TLB, lệnh dấu phẩy động được gửi đi.

7. E1: giai đoạn thứ hai của ALU, tìm được trong Dcache dữ liệu phù hợp, nạp dữ liệu có sẵn, xử lý lỗi loại trừ (các loại trừ).

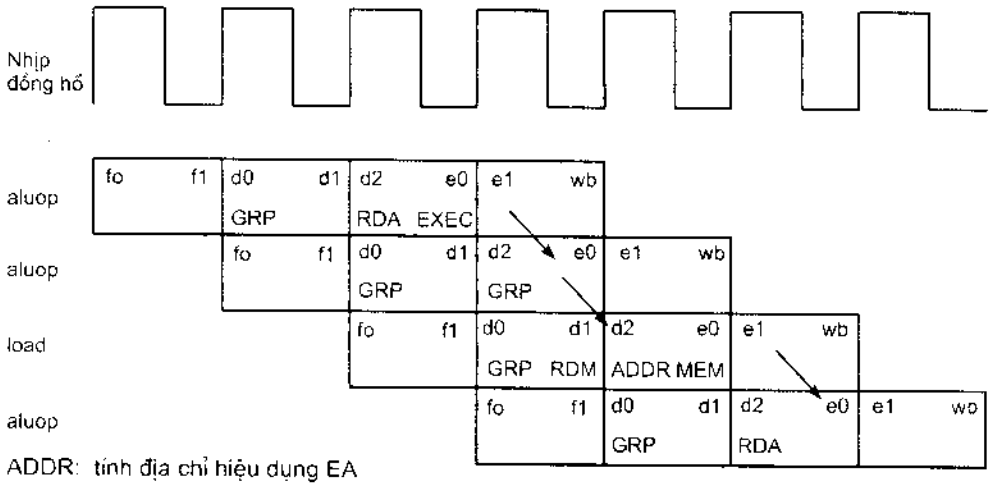
8. WB: ghi kết quả vào tệp thanh ghi, cất vào vùng đệm.

Đường ống FPU được kết hợp chặt chẽ với đường ống IU, một phép thao tác (phép toán) đều có thể bắt đầu từ mọi chu kì, độ trễ của hầu hết các thao tác dấu phẩy động là ba chu kì. Trong pha E0, một lệnh số học dấu phẩy động được chọn để thực hiện và các toán hạng được đọc trong suốt pha E1. Bộ cộng và bộ nhân số chính xác kép FPU cần hai giai đoạn trễ thực hiện. Chu kì đầu tiên của bộ cộng là kiểm tra số mũ, phần định trị và tính toán ra kết quả. Chu kì đầu của bộ nhân là tính toán và cộng các phần kết quả. Giai đoạn thứ hai là làm tròn và chuẩn hóa. Chuỗi kết quả của một thao tác FPU được đưa vào phần nguồn của thao tác tiếp theo.

Hình 7.7 mô tả một ví dụ đường ống thực hiện một tập hợp các thao tác (các lệnh) của ALU, cùng với lệnh nạp ở giữa (chu kì thứ ba). Trong suốt chu kì (F0, F1) có thể có bốn lệnh được chuyển tới, nhưng

chỉ có thể tối đa ba lệnh được đưa ra như một nhóm (GRP) trong pha D0. Kết quả được đưa tới nhóm lệnh tiếp theo như mũi tên chỉ đến.

Đường ống thực hiện một tập các lệnh, gồm các lệnh rẽ nhánh, như ở hình 7.43. Trong đó, trường hợp lệnh rẽ nhánh thỏa mãn điều kiện diễn giải bằng hình 7.8 và không thỏa mãn điều kiện bằng hình 7.9. Các lệnh tuần tự đầu tiên được biểu thị là S1, S2 và lệnh đích như T1, T2, T3, T4. Một lệnh trễ được đặt sau một lệnh rẽ nhánh có điều kiện (trong ví dụ này là BNE) được biểu thị là DI. Bộ xử lý có thể nhóm các lệnh so sánh (CMP) và các lệnh rẽ nhánh có điều kiện (BNE) để tăng tốc độ xử lý.

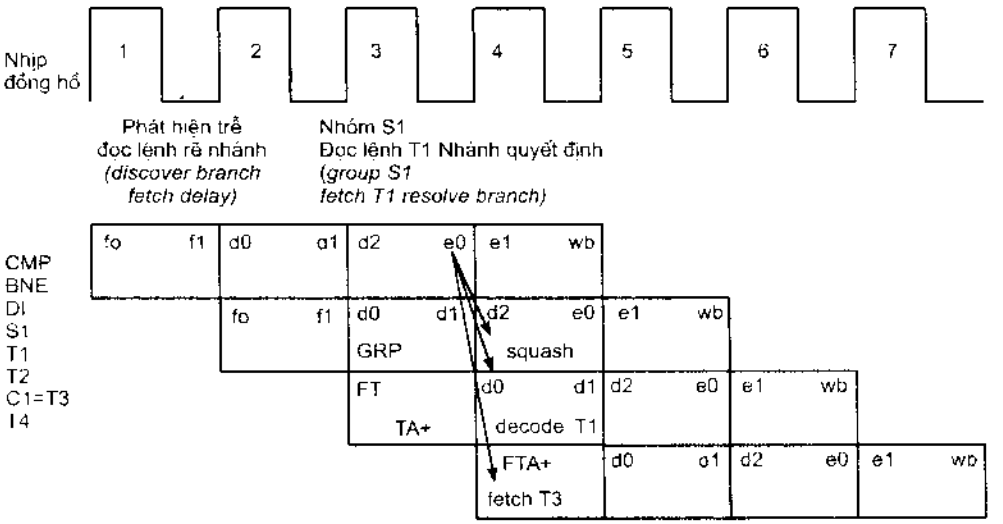


ADDR: tính địa chỉ hiệu dụng EA
EXEC: thi hành
GRP: đưa ra 1 nhóm lệnh
MEM: truy nhập bộ lưu trữ dữ liệu
RDA: đọc tệp thanh ghi cho các toán hạng của ALU
RDM: đọc tệp thanh ghi để truy nhập bộ nhớ

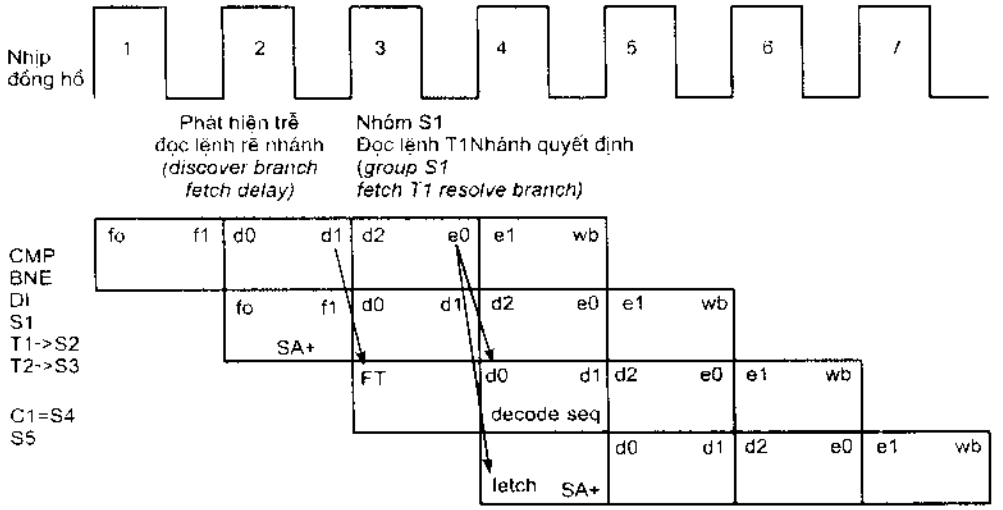
Hình 7.43: Quá trình thực hiện đường ống

Giả thiết rằng các lệnh rẽ nhánh có điều kiện thỏa mãn. Khi một lệnh nhảy đến địa chỉ PC được đưa ra thì DI cũng được đọc ra đồng thời. Địa chỉ đích được tính trong pha D1. Khi lệnh rẽ nhánh vào pha D2, thì lệnh đích cũng được đọc ra (FT). Quá trình đọc lệnh được hoàn thành khi DI đến pha D1 và lệnh so sánh và rẽ nhánh vào pha E0. Lệnh so sánh tính toán mã điều kiện mới trong pha E0 và hướng rẽ nhánh.

Khi một lệnh rẽ nhánh thỏa mãn thì các lệnh tuần tự không có hiệu lực. Khi lệnh rẽ nhánh không thỏa mãn, các lệnh tuần tự vẫn có hiệu lực và những lệnh đích bị bỏ. Giá trị PC trong cả hai tình huống đều được tính toán trước. Trong Super SPARC, việc xử lý các lệnh rẽ nhánh không thỏa mãn điiều kiện có hiệu quả hơn các lệnh rẽ nhánh thỏa mãn diều kiện.



Hình 7.44: Thực hiện đường ống và rẽ nhánh có điều kiện được lấy ra



Hình 7.45: Thực hiện đường ống và rẽ nhánh có điều kiện không được lấy ra

Super SPARC thực hiện mô hình chính xác loại trừ. Bất kể lúc nào cũng có đến 09 lệnh trong đường ống IU và 04 lệnh trong hàng dấu phẩy động. Lỗi loại trừ và các lệnh gây ra các lỗi loại trừ lan truyền suốt đường ống IU. Chúng được xử lý trong giai đoạn thi hành (execute stage) trước khi kết quả của chúng có thể thay đổi trạng thái trong tệp thanh ghi, thanh ghi điều khiển và bộ nhớ.

Đơn vị FPU bao gồm một bộ điều khiển dấu phẩy động FPC (Floating Point Controler), hai đường ống độc lập FADD và FMUL, một hàng dấu phẩy động và một tệp 32 thanh ghi dấu phẩy động, mỗi thanh ghi có 32 bit, mỗi tệp thanh ghi dấu phẩy động FP được tổ chức như 16 thanh ghi 64-bit (trong đó mỗi từ có 32 bit) để tối thiểu hóa thao tác của số chính xác kép. Tuy nhiên mỗi từ 32-bit của FP có thể truy nhập riêng biệt. Tệp FP có ba cổng đọc, hai cổng ghi. Bộ điều khiển dấu phẩy động FPC được nối chặt chẽ với đường ống IU và có thể thực hiện các thao tác dấu phẩy động trong một chu kỳ. FPC cũng xử lý các lỗi loại trừ của các phép toán dấu phẩy động.

Có hai kiểu lệnh dấu phẩy động:

1. FPOP: các thao tác dấu phẩy động như cộng, nhân, chuyển đổi v.v...

2. FPEVENT: các lệnh như nạp/cất đến hoặc từ các thanh ghi dấu phẩy động, nạp/cất đến hoặc từ các thanh ghi trạng thái, cất các hàng dấu phẩy động, nhân/ chia số nguyên. Các lệnh này được thực hiện bởi FPU nhưng không xếp vào hàng FPU (FPU queue).

Đường ống FPU gồm bốn giai đoạn:

1. FRD - giải mã và đọc

2. FM/FA- thực hiện nhân và cộng

3. FN/FR- chuẩn hóa và làm tròn

4. FWD - ghi trở lại tệp thanh ghi dấu phẩy động (FP file).

Tất cả các lệnh dấu phẩy động được đưa ra bởi IU trong pha E0 của đường ống IU. Mỗi lần đưa ra, nó được đưa tới đường ống FPU.

Đường ống FPU trì hoãn đường ống IU trong vài trường hợp: hàng FPU sẽ đầy sau vài lệnh trễ số học. Chuỗi kết quả của một lệnh FPOP được gửi đến toán hạng nguồn của một FPOP tiếp theo. Tương tự, một lệnh PREVENT (ngăn chặn) sẽ gửi dữ liệu đến toán hạng nguồn của lệnh FPOP và kết quả của FPOP lại cất bằng một lệnh PREVENT.

Các lệnh FPU được bắt đầu và hoàn thành có trật tự. Chúng được thực hiện bằng một trong hai đường ống riêng rẽ FADD và FMUL. Trong hàng, các lệnh thực hiện theo nguyên tắc vào trước, ra trước (FIFO). Hàng gồm có bốn điểm vào (entry), mỗi điểm vào có thể chứa một lệnh FPOP 32-bit và một địa chỉ FPOP 32-bit.

Ngoài các lệnh đã có trong tập lệnh của SPARC, SuperSPARC có bổ sung thêm một số lệnh mới, đó là:

1. UMUL (UMULcc): nhân các số nguyên không dấu (và thay đổi icc).

2. SMUL (SMULcc): nhân các số nguyên có dấu (và thay đổi icc).

3. UDIV (UDIVcc): chia các số nguyên không dấu (và thay đổi icc).

4. SDIV (SDIVcc): chia các số nguyên có dấu (và thay đổi icc).

5. RDASR: đọc thanh ghi trạng thái phụ thuộc (ancillary state register).

6. WRASR: ghi vào thanh ghi trạng thái phụ thuộc.

7. STBAR: cất giữ barrier.

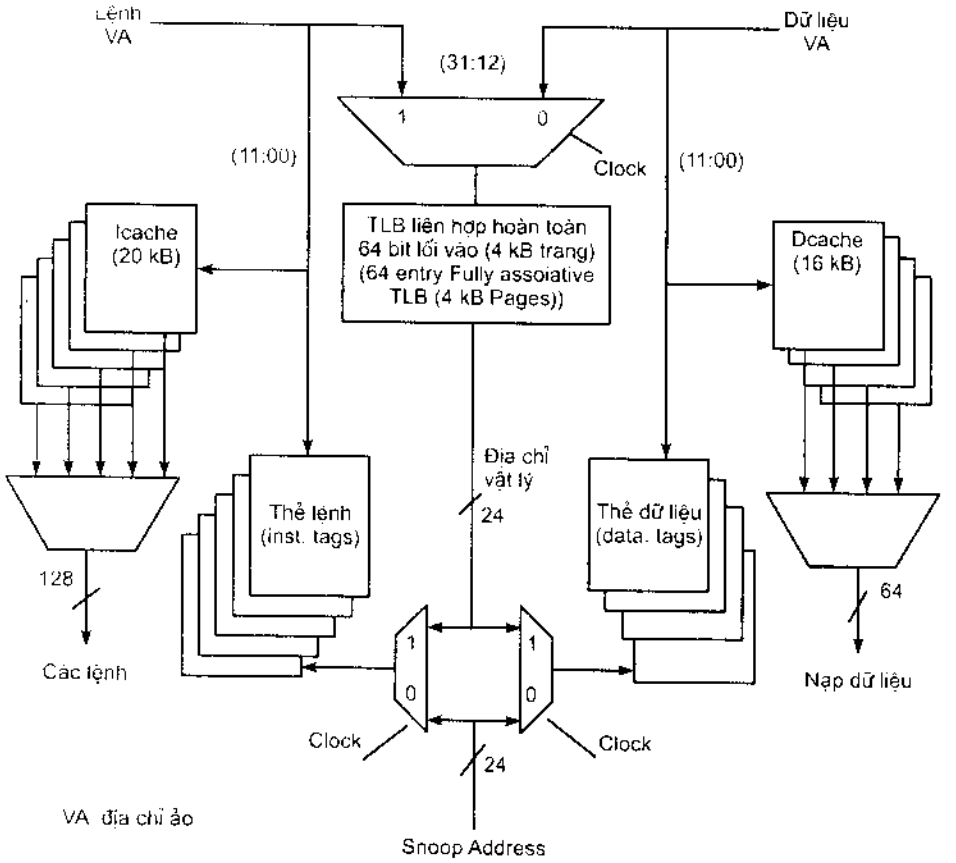
8. FsMULd: nhân các số dấu phẩy động chính xác đơn thành số dấu phẩy động chính xác kép.

9. FdMULq: nhân các số dấu phẩy động chính xác kép thành số dấu phẩy động chính xác gấp 4.

d) Kiến trúc của cache trong SuperSPARC

Trên chip Super SPARC, có cache đôi: 20 kB Icache, 16 kB Dcache. Nó hỗ trợ cho cache thứ cấp ngoài chip. Hình 7.46 mô tả tổ chức của cache và đơn vị quản lý bộ nhớ MMU. Trong cấu trúc

SuperSPARC, mỗi tập hợp (set) của cache là một khối dung lượng 4 kB. Kích thước của trang cũng là 4 kB. Icache có 5 tập hợp như vậy, nghĩa là Icache có dung lượng 20 kB (5 trang). Dcache có 4 tập hợp như vậy, nghĩa là có dung lượng 16 kB. Chính vì thế Icache được coi là 5-đường liên hợp tập (fine-way set-asociative), tức là 5 dòng/1 tập và Dcache là 4-đường liên hợp tập (four-way set-asociative), 4 dòng/1 tập. Kích thước mỗi dòng trên Icache là 64 byte, trên Dcache là 32 byte. Icache được truy nhập nhờ đường dẫn đọc dài 128-bit. Đường dẫn này cho phép đọc đồng thời bốn lệnh. Dcache được truy nhập nhờ đường dẫn dài 64-bit. Đường dẫn này cho phép vận chuyển dữ liệu số chính xác kép trong 1 chu kỳ nhịp.



Hình 7.46: Tổ chức Cache và MMU của SuperSPARC

Tỉ lệ trúng cache (hit) là 98% với Icache, 90% với Dcache (hit là khả năng truy nhập trúng lệnh/dữ liệu).

7.4.4 Họ vi xử lý PowerPC

a) Giới thiệu về PowerPC

Các máy tính có tập lệnh giảm thiểu RISC bắt đầu phát triển vào những năm 1970 tại IBM, là hệ thống 801. Mặc dù đến tận năm 1980 thuật ngữ RISC mới hình thành ở Berkeley. Những nhà thiết kế 801 IBM đã tạo ra một hệ thống mới, hướng tới sử dụng ngôn ngữ bậc cao (HLL). Hệ thống này cho phép mỗi lệnh được thi hành trong khoảng một chu kì xung nhịp. Tất nhiên là có một vài lệnh, như nạp, cất (phải truy nhập bộ nhớ), nhân chia không thể thực hiện trong vòng một chu kì. Muốn có nhiều lệnh thực hiện trong vòng 1 chu kì thì nó phụ thuộc vào công nghệ phần cứng (bộ vi xử lý), nó phải là bộ vi xử lý kiểu RISC.

Vào giữa những năm 1980, các bộ vi xử lý kiểu RISC đã mang tính thương mại. IBM cho ra đời bộ vi xử lý ROMP. So với 801, nó có ít lệnh thực hiện trong một chu kì hơn, có 65% lệnh có chiều dài là 16-bit, số còn lại là 32-bit, trong khi đó tất cả các lệnh của 801 là 32-bit. ROMP được sử dụng trong các máy RT 6150, RT 6151. Đến năm 1990, IBM cho ra đời bộ vi xử lý RS/600. Đến năm 1991, IBM hợp tác với Motorola, Apple cho ra đời bộ vi xử lý họ Power PC. Nó là họ vi xử lý mạnh và dẫn đầu trong công nghệ máy tính để bàn. Phiên bản đầu tiên của họ PowerPC là PowerPC 601 (Motorola còn gọi là MPC 601, IBM là PPC 601). Tiếp theo là các phiên bản MPC 603, 604, 620,... Tất cả những hệ thống mới này dựa trên kiến trúc PowerPC.

b) Kiến trúc của PowerPC

Kiến trúc PowerPC gồm 3 lớp:

1. Kiến trúc tập lệnh của người sử dụng: bao gồm các thanh ghi cấp người sử dụng, mô hình lập trình, kiểu dữ liệu, chế độ địa chỉ và tập lệnh cơ sở cấp người sử dụng (trừ một vài lệnh điều khiển bộ nhớ).

2. Kiến trúc môi trường ảo: mô tả các ngữ nghĩa của mô hình bộ nhớ mà có thể được giả thiết bởi các tiến trình phần mềm: mô hình cache, các lệnh điều khiển cache, gán địa chỉ,...

3. Kiến trúc môi trường điều hành: gồm cấu trúc của mô hình quản lý bộ nhớ, các thanh ghi cấp người giám sát, các lệnh thẩm quyền và mô hình loại trừ (exception model).

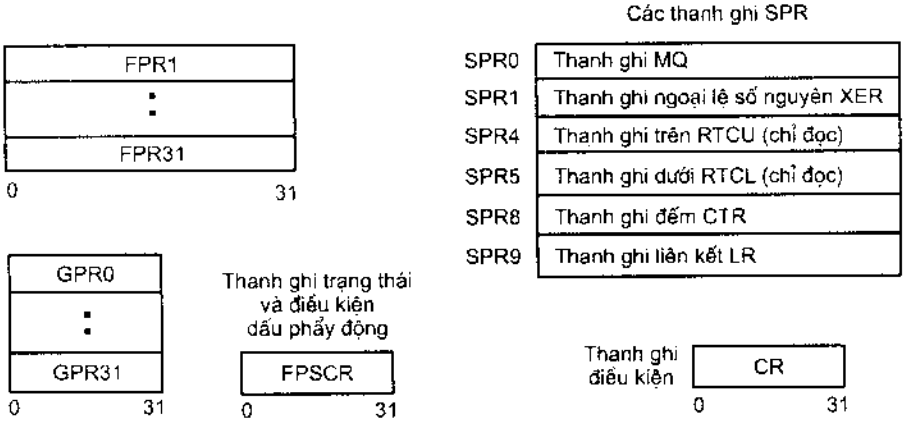
Các hệ thống dựa trên kiến trúc của Power PC vận hành trong hai chế độ cơ bản: chế độ người sử dụng và chế độ giám sát tương tự như họ M68xxx.

(1) Các thanh ghi và mô hình lập trình

Các thanh chia làm hai phần chính với hai chế độ thao tác: mô hình lập trình người sử dụng, mô hình lập trình người giám sát.

Những chương trình thực hiện trong chế độ người giám sát có thể truy nhập tất cả các thanh ghi. Những chương trình trong chế độ người sử dụng chỉ có thể truy nhập vào các thanh ghi trong mô hình chương trình người sử dụng.

Mô hình lập trình người sử dụng (hình 7.47)



Hình 7.47: Mô hình lập trình của người sử dụng

Mô hình lập trình người sử dụng gồm các thanh ghi sau:

1. Các thanh ghi toàn cục (GPR)

Tệp thanh ghi GPR gồm 32 thanh ghi từ GPR0 đến GPR31. Các thanh ghi GPR có độ dài 32 bit trong hệ thống 32-bit, và 64 bit trong hệ thống 64-bit. Kiến trúc Power PC mở rộng đến 64 bit, nhưng có một phần là 32 bit. Chú ý rằng trong kiến trúc Power PC bit có trọng số lớn nhất (MSB) là bit 0, và bit có trọng số thấp nhất (LSB) là bit 31, ngược lại so với các bộ vi xử lý khác mà ta đã xét. Các thanh ghi GPR được coi như nguồn và đích dữ liệu trong các lệnh số nguyên và cung cấp địa chỉ trong các lệnh truy nhập bộ nhớ.

2. Các thanh ghi dấu phẩy động (FPR)

Tệp thanh ghi FPR gồm 32 thanh ghi 64-bit từ FPR0 đến FPR31. Các thanh ghi FPR được coi như nguồn và đích dữ liệu trong các lệnh dấu phẩy động. Chúng có thể chứa số chính xác đơn và kép theo chuẩn IEEE 754.

3. Các thanh ghi điều khiển và trạng thái dấu phẩy động (FPSCR)

Các thanh ghi FPSCR là 32-bit và chứa bit tín hiệu loại trừ dấu phẩy động, bit tổng kết loại trừ, bit có thể loại trừ và các bit điều khiển làm tròn thỏa mãn chuẩn IEEE 754.

4. Thanh ghi điều kiện (CR)

Thanh ghi CR (hình 7.48) có chiều dài 32 bit, chia làm 8 phần, mỗi phần 4 bit, làm thành các thanh ghi điều khiển từ CR0 đến CR7. Mỗi phần chỉ tới kết quả của một thao tác số học nào đó và cung cấp kĩ thuật kiểm tra, rẽ nhánh.

Nội dung của các thanh ghi điều kiện CR được thiết lập trong những trường hợp sau:

- Vùng chỉ định trong thanh ghi CR có thể bằng 1 bởi một lệnh chuyển từ một thanh ghi GPR (**mtcrf** hoặc **mcrfs**).

- Vùng chỉ định của CR có thể chuyển từ một vùng CRi đến một vùng khác bằng lệnh **mcrf.**

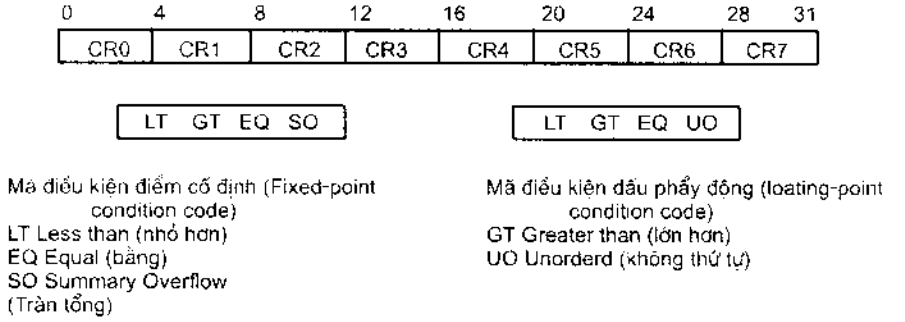
• Vùng chỉ định của CR có thể thiết lập 1 bằng lệnh chuyển (**mcrxr**) từ thanh ghi XER.

• Các lệnh logic của CR được sử dụng để thao tác trên các bit chỉ định trong CR.

• CR0 có thể là kết quả không rõ ràng của một thao tác số nguyên.

• CR1 có thể là kết quả không rõ ràng của một thao tác số dấu phẩy động

• Vùng CR chỉ định có thể là kết quả rõ ràng của các lệnh so sánh số nguyên và dấu phẩy động.



Hình 7.48: Thanh ghi điều kiện CR

Sự thiết lập các bit CR0 và CR1 là kết quả tác động ẩn của lệnh. Có các lệnh đảm bảo kiểm tra từng bit riêng biệt của CR. Thanh ghi CR được xóa bằng phần cứng.

Các bit của CR (CRi) được thiết lập trong các thao tác so sánh như sau:

- Bit 0, nhỏ hơn, LT đối với số nguyên, FL đối với dấu phẩy động.

- Bit 1, lớn hơn, GT đối với số nguyên, FG đối với dấu phẩy động.

- Bit 2, ngang bằng, EQ đối với số nguyên, FE đối với dấu phẩy động.

- Bit 3, tràn tổng (summary overflow), SO đối với số nguyên, FU (hoặc UO) đối với dấu phẩy động.

Trường của CR0 có chỉ định đặc biệt, nó được thiết lập trong các lệnh số học:

- Bit 0, negative (LT), thiết lập 1 khi kết quả âm
- Bit 1, positive (GT), thiết lập 1 khi kết quả dương
- Bit 2, zero (EQ), thiết lập 1 khi kết quả bằng 0
- Bit 3, tràn tổng (SO), thiết lập 1 khi xảy ra tràn

5. SPR (thanh ghi dùng cho mục đích đặc biệt)

Có 6 thanh ghi SPR trong mô hình lập trình của người sử dụng, đó là:

- Thanh ghi MQ
- Thanh ghi loại trừ số nguyên (XER)
- Thanh ghi phần cao của nhịp đồng hồ thời gian thực RTCU (Real Time Clock Upper register)
- Thanh ghi phần thấp của nhịp đồng hồ thời gian thực RTCL (Real Time Clock Lower register)
- Thanh ghi liên kết (LR)
- Thanh đếm (CTR)

6. Các thanh ghi MQ, RTCU, RTCL

Các thanh ghi này không nằm trong kiến trúc PowerPC, chúng là sản phẩm của kiến trúc IBM Power. Kiến trúc PowerPC dùng thanh ghi cơ sở thời gian (time-base) thay cho RTC.

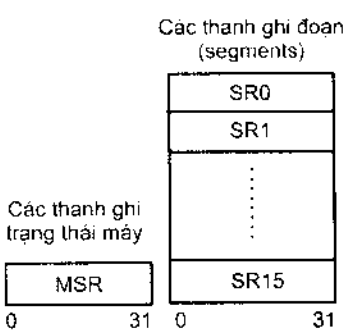
Mô hình lập trình người giám sát (hình 7.49)

Mô hình lập trình của người giám sát gồm các thanh ghi sau đây:

1. Thanh ghi trạng thái máy (MSR) 32-bit: Thanh ghi MSR cho biết trạng thái hiện thời của bộ vi xử lý

2. Các thanh ghi đoạn (SR): có 16 thanh ghi đoạn 32-bit trong PowerPC 32-bit.

3. Các thanh ghi cấp giám sát SPR18-SPR1023: có 27 thanh ghi SPR 32-bit



Các thanh ghi SPR

SPR18	DISISR-DAE/Thanh ghi dịch vụ nguồn lệnh
SPR19	DAR- Thanh ghi địa chỉ dữ liệu
SPR20	RTCU- Thanh ghi RTC bit cao (chỉ dành cho ghi)
SPR21	RTCL- Thanh ghi RTC bit thấp (chỉ dành cho ghi)
SPR22	DEC- Thanh ghi giảm
SPR25	SDR1- Thanh ghi mô tả tìm bảng 1
SPR26	SRR0- Thanh ghi lưu và cất giữ 0
SPR27	SRR1- Thanh ghi lưu và cất giữ 1
SPR272	SPRG0- SPR chung 0
SPR273	SPRG1- SPR chung 1
SPR274	SPRG2- SPR chung 2
SPR275	SPRG3- SPR chung 3
SPR282	EAR- Thanh ghi truy nhập bên ngoài
SPR287	PVR- Thanh ghi phiên bản xử lý
SPR528	IBAT0U- BAT 0 cao 2
SPR529	IBAT0L-BAT 0 thấp 2
SPR530	IBAT1U-BAT 1 cao 2
SPR531	IBAT1L-BAT 1 thấp 2
SPR532	IBAT2U-BAT 2 cao 2
SPR533	IBAT2L-BAT 2 thấp 2
SPR534	IBAT3U-BAT 3 cao 2
SPR535	IBAT3L-BAT 3 thấp 2
SPR1008	HID0
SPR1009	HID1
SPR1010	HID2 (ABR)
SPR1013	HID5 (DABR)
SPR1023	HID15 (PIR)

Hình 7.49: Mô hình lập trình của người giám sát

Các bit của thanh ghi MSR trong hình 6.4 được sử dụng như sau:

0 15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0000000000000000	EE	PR	FP	ME	FE0	SE	0	FE1	0	EP	IT	DT	0	0	0	0

Hình 7.50: Thanh ghi trạng thái máy MSR

- Bit 16 (EE): cho phép ngắt từ ngoài khi EE = 1, không cho phép ngắt từ ngoài khi EE = 0.

- Bit 17 (PR): mức thẩm quyền. Bộ xử lý chỉ có thể thực hiện các lệnh cấp người sử dụng khi PR = 1, và thực hiện tất cả các lệnh khi PR = 0.

- Bit 18 (FP): có các lệnh dấu phẩy động. Bộ xử lý thực hiện các lệnh dấu phẩy động khi FP = 1. Không có lệnh xử lý các số dấu phẩy động khi FP = 0.

- Bit 19 (ME): có thể kiểm tra máy khi ME = 1, không thể kiểm tra khi ME = 0.

- Bit 20 và 23 (FE0 và FE1): chế độ loại trừ của dấu phẩy động, được mã hóa như sau:

FE0	FE1	Chế độ
0	0	Các loại trừ dấu phẩy động không thể được
0	1	Sự không chính xác dấu phẩy động không thể đạt được
1	0	Sự không chính xác dấu phẩy động đạt được
1	1	Chế độ chính xác dấu phẩy động

- Bit 21 (SE): thao tác theo bước đơn (single step operation) khi SE = 1, thao tác bình thường khi SE = 0.

- Bit 25 (EP): tiền tố loại trừ (exeception prefix) - xác định một số gia của véc-tơ loại trừ (exeception vector offset). Khi EP = 0, các loại trừ được chỉ đến địa chỉ vật lý 000n nnnn (hex). Khi EP = 1, các loại trừ được chỉ đến địa chỉ vật lý FFFn nnnn (hex).

- Bit 26 (IT): chuyển đổi địa chỉ của lệnh. Sự chuyển đổi có thể được khi IT = 1, không thể chuyển đổi khi IT = 0.

- Bit 27(DT): chuyển đổi địa chỉ của dữ liệu. Sự chuyển đổi có thể khi DT = 1, không thể khi DT = 0.

- Các bit còn lại của thanh ghi MSR để dự trữ.

(2) Kiểu dữ liệu

Power PC có các kiểu dữ liệu nguyên như sau:

Toán hạng	Kích thước byte	Kích thước bit	Địa chỉ từ bit 28 đến 31
Byte	1	8	xxxx
Halfword	2	16	xxx0
Word	4	32	xx00
Doubleword	8	64	x000

Để đạt được hiệu suất cao nhất thì các toán hạng dữ liệu phải được sắp xếp. Power PC chấp nhận cả hai kiểu đánh địa chỉ byte là: kết thúc lớn (big-endian), trong đó, LSB có địa chỉ thấp, tức là các byte được

đánh địa chỉ 3, 2, 1, 0 từ MSB trở xuống và kết thúc nhỏ (litter-endian), trong đó, LSB có địa chỉ cao, tức là các byte được đánh địa chỉ 0, 1, 2, 3, từ MSB trở xuống. Địa chỉ byte mặc định là big-endian. Kiểu địa chỉ byte có thể đổi sang litter-endian bằng cách thiết lập một bit trong thanh ghi MSR.

(3) Các chế độ địa chỉ

Giống như tất cả hệ thống kiểu RISC khác, các lệnh thực hiện trong các thanh ghi đều sử dụng hai chế độ địa chỉ sau:

1. Trực tiếp thanh ghi (register direct): toán hạng ở trong thanh ghi GPR hoặc FPR.

2. Ngay lập tức (immediate): toán hạng là một phần của lệnh.

Các chế độ địa chỉ cho các lệnh nạp và cất là:

(1) Gián tiếp thanh ghi (register indirect): thanh ghi GPR chứa địa chỉ của toán hạng trong bộ nhớ.

(2) Gián tiếp thanh ghi với chỉ số trực tiếp (register indirect with immediate index): độ dịch trực tiếp d cộng với giá trị trong thanh ghi GPR tạo thành địa chỉ hiệu dụng EA.

(3) Gián tiếp thanh ghi với chỉ số (register indirect with index): cộng nội dung của hai thanh ghi, một được coi là thanh ghi gián tiếp (regs indirect), một được coi là thanh ghi chỉ số (index register), tạo ra địa chỉ hiệu dụng EA.

Bất kì một trong các thanh ghi GPR đều có thể là thanh ghi gián tiếp hoặc thanh ghi chỉ số.

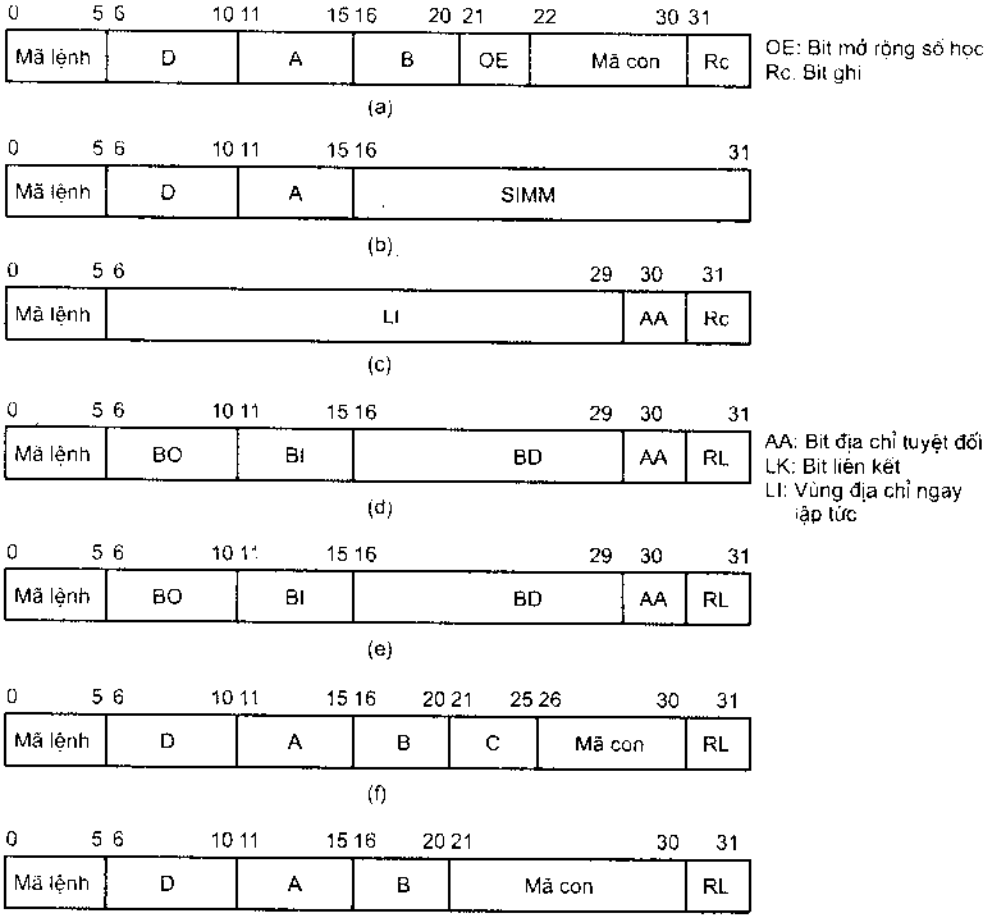
Các chế độ địa chỉ sử dụng cho lệnh rẽ nhánh:

(1) Địa chỉ ngay lập tức (immediate addressing): Địa chỉ rẽ nhánh là một phần của lệnh, mở rộng dấu để tạo thành địa chỉ hiệu dụng đích EA. Chế độ này còn được gọi là chế độ địa chỉ trực tiếp hoặc tuyệt đối.

(2) Gián tiếp thanh ghi liên kết (link register indirect): địa chỉ hiệu dụng đích EA nằm ở trong thanh ghi liên kết.

(3) Gián tiếp thanh đếm (Count register indirect): địa chỉ hiệu dụng đích EA nằm ở trong thanh đếm.

(4) Dạng lệnh



Hình 7.51: Các dạng lệnh chính của kiến trúc PowerPC

Một vài dạng lệnh chính của kiến trúc Power PC được trình bày trong hình 7.51. Các lệnh của Power PC có độ dài 32-bit. Dạng lệnh trong hình 7.51 được sử dụng trong hầu hết các thao tác. 06 bit cao nhất chứa mã lệnh. Mã lệnh có vùng mở rộng là subopcode (từ bit 22 đến 30). Toán hạng nguồn ở trong vùng A (từ bit 11 đến 15) và B (từ bit 16 đến 20). Toán hạng đích trong vùng D (từ bit 6 đến 10). Bit điều

khiển EO (bit 21) có thể phát hiện tràn khi EO = 1. Bit điều khiển Rc (bit 31) được gọi là bit ghi, cập nhật thanh ghi CR nếu Rc = 1. Trong kiến trúc Power PC, trật tự của các toán hạng trong câu lệnh giống như trong dạng lệnh.

Ví dụ:

add rD, rA, rB ; rD ← (rA) + (rB)

Bất kì một thanh ghi GRP (của số nguyên) hay FPR (của dấu phẩy động) đều có thể là các toán hạng rA, rB hoặc rD trong ví dụ này.

Các thao tác với toán hạng tức thời (immediate operand) có dạng ở hình 7.51(b). Vùng SIMM (bit 16 đến 31) chứa toán hạng tức thời 16-bit.

Ví dụ:

addi rA, rD, SIMM ; cộng tức thời, rD ← (rA) + EXTS (SIMM)

EXTS (SIMM) là vùng mở rộng dấu của SIMM.

Các lệnh nạp và cất cũng sử dụng dạng lệnh trong hình 7.51(b) ở chế độ địa chỉ gián tiếp thanh ghi với chỉ số trực tiếp (register indirect with immediate index). Trong trường hợp này vùng A chứa mã hóa gián tiếp của thanh ghi, vùng D là vùng thanh ghi đích với lệnh nạp, thanh ghi nguồn với lệnh cất giữ.

Các lệnh rẽ nhánh sử dụng dạng lệnh trong hình 7.51(c). Vùng LI là vùng địa chỉ tức thời. Nếu bit AA của địa chỉ tuyệt đối bằng 0, thì LI dịch trái hai bit, hai bit thấp nhất gán giá trị 0, cộng với địa chỉ của lệnh tạo thành địa chỉ rẽ nhánh đích. Nếu AA = 1 thì LI dịch trái hai bit cộng với phần mở rộng dấu của LI tạo thành địa chỉ rẽ nhánh đích. Nếu bit liên kết LK (bit 31) bằng 1 thì địa chỉ hiệu dụng EA của lệnh ngay sau lệnh rẽ nhánh được đặt vào thanh ghi LR.

Dạng lệnh rẽ nhánh có điều kiện được minh hoạ ở hình 7.51(d). Vùng BO chỉ định điều kiện rẽ nhánh. Vùng BI chỉ định bit trong thanh ghi CR làm điều kiện rẽ nhánh. Vùng BD là địa chỉ rẽ nhánh

đích ảo. Các bit AA và LK sử dụng tương tự như trong lệnh rẽ nhánh vô điều kiện (hình 7.51).

Đặc điểm của kiến trúc Power PC là tạo ra các lệnh liên quan đến các phép nhân và cộng dấu phẩy động.

Ví dụ:

fmadd frD, frC, frB ; frD ← (frA) *(frC) + (frB); hình 7.51(e) là dạng của lệnh này.

Dạng lệnh hình 7.51(f) tương tự như dạng trong hình 7.51(a), nó được sử dụng cho các lệnh cất và nạp với chế độ địa chỉ gián tiếp thanh ghi với chỉ số (register indirect with index). Vùng A sử dụng cho thanh ghi gián tiếp, vùng B sử dụng cho thanh ghi chỉ số. Vùng D là đích với lệnh nạp và nguồn với lệnh cất.

(5) Tập lệnh

Tập lệnh cơ bản của Power PC có 171 lệnh. Một vài lệnh trong đó như lệnh cộng có 4 mẫu lệnh (bảng 7.1).

Ví dụ:

Cộng số nguyên:

add: cộng với giá trị cập nhật trong CR

addo: cộng có tràn

addo.: cộng có tràn với giá trị cập nhật trong CR

Cộng dấu phẩy động:

fadd: cộng dấu phẩy động

fadd.: cộng dấu phẩy động với giá trị cập nhật trong CR

Các lệnh rẽ nhánh **b** và rẽ nhánh có điều kiện **bc** có 4 tùy chọn khác nhau.

Trong lệnh **b** có 3 tùy chọn bổ sung là:

ba: rẽ nhánh tuyệt đối. AA = 1, LK = 0. Địa chỉ rẽ nhánh đích bằng địa chỉ của lệnh rẽ nhánh cộng với vùng dịch của LT.

bl: rẽ nhánh rồi liên kết AA = 0, LK = 1. Vùng LI là địa chỉ đích của lệnh tiếp theo đặt trong LR.

bla: rẽ nhánh tuyệt đối rồi liên kết. AA = 1, LK = 1, là kết hợp của **ba** và **bl**.

Các lệnh của PowerPC được phân chia thành các nhóm sau đây:

1. Các lệnh xử lý số nguyên (interger instructions): có 4 kiểu lệnh nguyên

a) Các lệnh số học nguyên: cộng (add), cộng ngay lập tức (addi), cộng ngay lập tức và dịch (addis), cộng ngay lập tức có nhớ (addic), cộng có nhớ (addc), cộng mở rộng (adde), cộng với số trừ mở rộng (addme), cộng với 0 mở rộng (addze), trừ (subf), trừ số mở rộng (subfe), trừ nhớ (subfc), trừ nhớ ngay lập tức (subfic), trừ số trừ mở rộng (subfme), trừ số 0 mở rộng (subfze), đảo (neg), nhân phần thấp ngay lập tức (mulli), nhân phần thấp (mullw), nhân từ cao (mullhw), nhân từ cao không dấu (mullhwu), chia số ngắn (divs), chia từ (divw), chia từ không dấu (divwu).

b) Các lệnh so sánh nguyên: so sánh (cmP), so sánh ngay lập tức (cmpi), so sánh logic (cmpl), so sánh logic ngay lập tức (cmpli).

c) Các lệnh logic nguyên: và (and), và có bù (andc), và ngay lập tức (andi), và có dịch ngay lập tức (andis), hoặc (or), hoặc có bù (orc), hoặc ngay lập tức (ori), hoặc có dịch ngay lập tức (oris), xor, xori, xoris, nand, nor, tương đương (eqv), mở rộng byte dấu (extsb), mở rộng nửa từ (extsh).

d) Các lệnh quay vòng và dịch: quay vòng từ sang trái ngay lập tức sau đó đưa vào mạt nạ (rlwimi), quay vòng từ sang trái ngay lập tức sau đó thực hiện logic and với mạt nạ (rlwinm), quay vòng từ sang trái sau đó and với mặt nạ (rlwnm), dịch trái từ (slw), dịch phải từ đại số (sraw), dịch phải từ đại số ngay lập tức (srawi), dịch phải tù (srw).

2. Các lệnh xử lý số dấu phẩy động: có 6 kiểu lệnh dấu phẩy động

a) Các lệnh số học dấu phẩy động: tất cả các lệnh được bắt đầu bằng chữ f trong bảng lệnh.

b) Các lệnh nhân chia dấu phẩy động, bao gồm các lệnh kết hợp nhân-cộng và chia-trừ dấu phẩy động bắt đầu bằng chữ f trong bảng lệnh.

c) Các lệnh làm tròn số và chuyển đổi dấu phẩy động: chuyển số dấu phẩy động thành số nguyên (fctiw), chuyển số dấu phẩy động thành số nguyên với sự làm tròn tới 0 (fctiwz), làm tròn số dấu phẩy động đến chính xác đơn (frsp).

d) Các lệnh so sánh các số dấu phẩy động: so sánh các số đã sắp xếp (fcmpo), so sánh các số chưa sắp xếp (fcmpu).

e) Các lệnh xử lý trạng thái và điều kiện: chuyển từ thanh ghi FPSCR (mffs), chuyển từ thanh ghi CR (mfcr), chuyển từ FPSCR tới CR (mcrfs), chuyển tới các trường FPSCR (mtfsf), chuyển tới các trường FPSCR ngay lập tức (mtfsfi), chuyển tới các trường CR (mtcrf), chuyển tới FPSCR bit 0 (mtfsb0), chuyển tới FPSCR bit 1 (mtfsb1).

f) Lệnh chuyển thanh ghi dấu phẩy động: fmr

3. Các lệnh nạp (load) và cất giữ (store) với có 8 kiểu lệnh

4. Các lệnh điều khiển rx nhánh với xử lý điều kiện (trong CR)

5. Các lệnh điều khiển bộ xử lý (sử dụng các thanh ghi SPR, MSR)

6. Các lệnh điều khiển bộ nhớ (các lệnh điều khiển cache)

7. Các lệnh điều khiển I/O với bên ngoài.

Ví dụ: Trừ: subf rD, rA, rB ; (rB) - (rA) → rD

Nhân: mullw rd, rA, rB ; 32 bit thấp của (rA)*(rB) → rD

mullhw rD, rA, rB ; 32 bit cao của (rA)*(rB) → rD

Chia: divw rD, rA, rB ; 32 bit thấp của (rA)/(rB) → rD

(6) Các ngoại lệ (exceptions)

Kiến trúc của PowerPC hỗ trợ 4 kiểu loại trừ:

1. Đồng bộ (synchronous), chính xác (precise): gây ra bởi một lệnh. Tất cả các lệnh gây ra ngoại lệ được xử lý một cách chính xác. Trạng thái của hệ thống trong thời gian xuất hiện loại trừ được biết và được phục hồi.

2. Đồng bộ, không chính xác: gây ra các loại trừ dấu phẩy động không chính xác. Có thể hoặc không thể khởi động lại.

3. Không đồng bộ, chính xác: gây ra các ngoại lệ của ngắt bên che được từ bên ngoài.

4. Không đồng bộ, không chính xác: gây ra các ngoại lệ của ngắt không chính xác không che được như xóa và kiểm hệ thống. Khi các ngoại lệ xuất hiện, thông tin, như về lệnh phải được thực hiện sau khi điều khiển được trả về cho chương trình chính và nội dung của MSR, phải được cất giữ trong các thanh ghi cất giữ/phục hồi SRR0 và SRR1 (SPR26 và SPR27), điều khiển chương trình chuyển từ cấp người sử dụng sang cấp giám sát, và chương trình bắt đầu thực hiện xử lý loại trừ.

(7) Bộ vi xử lý PowerPC 601 (MPC601)

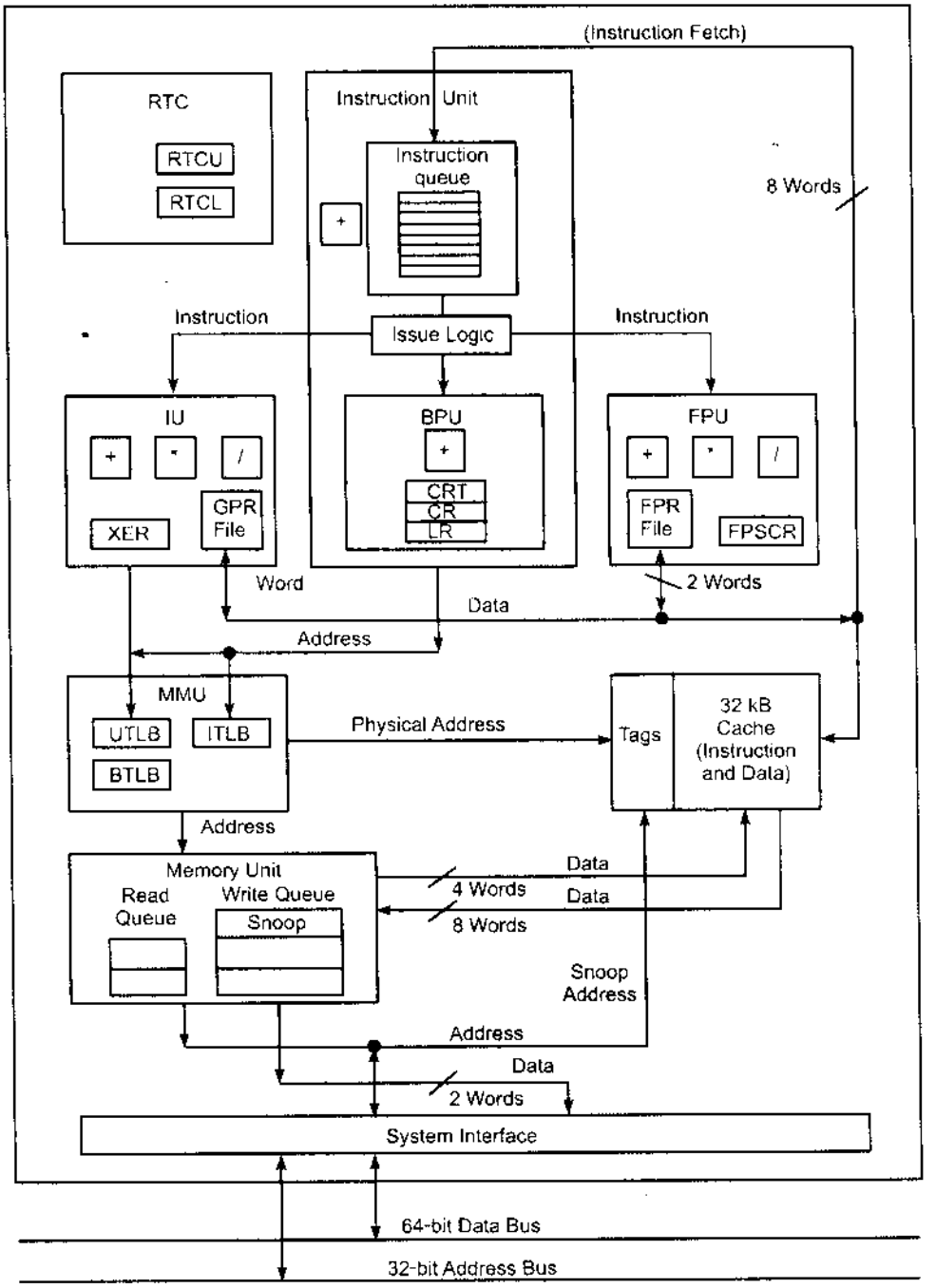
Bộ vi xử lý PowerPC 601, còn gọi là MPC601 là bộ xử lý đầu tiên của kiến trúc PowerPC. MPC601 chỉ là sản phẩm của hợp tác của IBM, Motorola và Apple, bởi vì sau đó loạt các chip vi xử lý ra đời như:

MPC603: có cấu trúc tương tự như MPC601 nhưng có 8 kB Icache và 8 kB Dcache

MPC604: là chip nâng cao so với MPC601, có mức xử lý song song cao. MPC604 có công nghệ CMOS 0,5 micron, 4 lớp kim loại, 3,6 triệu transistor, kiến trúc 4-đường siêu hướng với bộ đệm lệnh 8-điểm vào và 6 đơn vị xử lý (4-issue superscalar with 8-entry instruction buffer and 6 execuction units), 16 kB Icache và 16 kB Dcache (tổng cộng 32 kB cache).

MPC620: cao cấp hơn MPC604, là chip vi xử lý 64-bit thực sự (64-bit data, 64-bit address), có mức độ xử lý song song cao, kiến trúc siêu hướng.

MPC601 có công nghệ CMOS 4 lớp metal, đóng gói gồm 304-chân chữ nhật phẳng (ceramic quad flat pack), mật độ 2,8 triệu transistor, 0,6 micron, làm việc ở tần số làm việc 66 MHz, tiêu tán nhiệt 9 W ở nguồn tiêu thụ 3,6 V, 66 MHz. Sơ đồ khối của MPC601 cho ở hình 7.52.



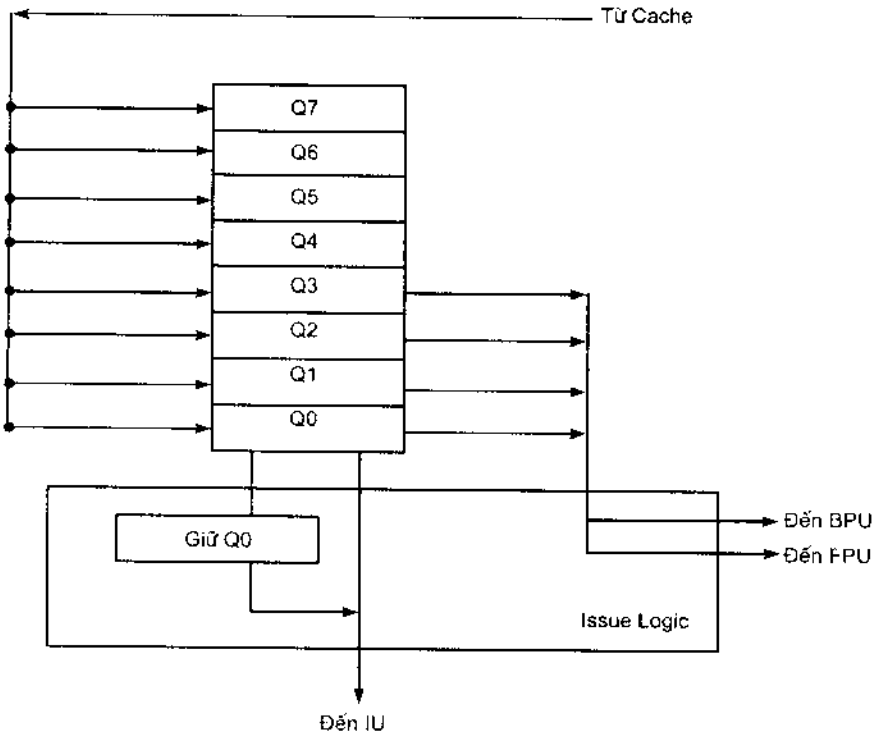
Hình 7.52: Sơ đồ khối của MPC601

MPC601 có các đơn vị chức năng như sau:

• Bus dữ liệu 64-bit, bus địa chỉ 32-bit, Icache và Dcache bên trong dung lượng 32 kB.

• Đơn vị xử lý số nguyên (IU) 32-bit).

IU đọc trước các lệnh từ cache và đặt các lệnh vào hàng đợi (instruction queue). Hàng đợi của lệnh được sắp xếp như trong hình 7.53. Hàng đợi có không gian chứa 8 lệnh trong các khe từ Q0 đến Q7. Cả 8 lệnh có thể được nạp vào hàng đợi trong một chu kỳ (có 32*8 = 256 bit dẫn từ cache đến đơn vị lệnh). Các lệnh được chuyển từ đỉnh của hàng (Q7) đến đáy (Q0). Nửa hàng đợi từ Q4 đến Q7 đảm bảo đệm để giảm nhu cầu truy nhập cache. Sự giải mã lệnh ban đầu được thực hiện ở nửa hàng đợi còn lại từ Q0 đến Q3. Chỉ có 3 lệnh có thể đồng thời đưa ra khỏi hàng lệnh tới các đơn vị thực hiện (IU, BPU và FPU). Một số lệnh có thể được đưa ra khỏi hàng đợi từ bất kỳ 4 vị trí đáy nào của hàng đợi.



Hình 7.53: Hàng đợi của các lệnh trong MPC601

IU của MPC601 thực hiện tất cả các lệnh thao tác với các số nguyên (kể cả các phần nguyên cho các thanh dấu phẩy động) và truy nhập bộ nhớ. Nó có ALU, nhân, chia, thanh ghi loại trừ XER (hay SPR1), và tệp 32 thanh ghi 32-bit GPR.

• Đơn vị xử lý số dấu phẩy động FPU

FPU gồm có mảng thực hiện nhân-cộng các số chính xác đơn, bộ chia, tệp 32 thanh ghi 64-bit FPR, thanh ghi FPSCR (hình 7.53), 2 bộ đệm bổ sung cho lệnh để cho phép các lệnh dấu phẩy động được đưa ra từ bộ đệm chung cho lệnh tới các đơn vị thực hiện khác ngay cả khi FPU đang bận.

• Đơn vị xử lý rẽ nhánh BPU (Branch Procesing Unit)

BPU thực hiện tìm kiếm trong nửa đáy hàng lệnh (từ Q0 đến Q3) lệnh rẽ nhánh và thử xử lý sớm lệnh rẽ nhánh. Nó cũng thực hiện xem xét trước các điều kiện cho rẽ nhánh trong thanh ghi CR. BPU sử dụng 1 bit trong giải mã lệnh để đoán hướng của các lệnh rẽ nhánh có điều kiện không thỏa mãn.

BPU chứa một bộ cộng để tính các địa chỉ đích rẽ nhánh và 3 thanh ghi SPRs: LR (SPR8), CTR (SPR9), và CR (xem hình 7.47). BPU tính địa chỉ nhảy trở về chương trình chính (địa chỉ của lệnh tiếp theo lệnh gọi tới chương trình con) và cất giữ địa chỉ này vào thanh ghi LR. LR có thể chứa địa chỉ đích rẽ nhánh cho lệnh rẽ nhánh có điều kiện tới LR (bclr). Thanh ghi CTR chứa địa chỉ đích rẽ nhánh cho lệnh rẽ nhánh có điều kiện tới CTR (bcctr). Bởi vì BPU sử dụng các thanh ghi trực tiếp hơn là các thanh ghi BPR hoặc FPR. Sự thực hiện các lệnh nhảy phụ thuộc vào sự thực hiện các lệnh nguyên và dấu phẩy động.

Ba lệnh đồng thời đưa ra có thể được thực hiện đồng thời chỉ khi chúng thuộc 3 loại lệnh riêng biệt, độc lập với nhau và phù hợp với 3 đơn vị thực hiện tương ứng (IU, BPU, và FPU).

MPC601 được trang bị 4 loại đường ống xử lý như sau:

(1) Đường ống xử lý rẽ nhánh (branch pipeline) 2 phân đoạn (tất cả trong một chu kỳ lệnh:

Đọc lệnh (fetch)

Chuyển đi (dispatch), giải mã lệnh (decode), thực hiện (execute), predict

(2) Đường ống xử lý các lệnh số nguyên (interger instruction pipeline) 4 phân đoạn:

Đọc lệnh (fetch)

Chuyển đi (dispatch), giải mã lệnh (decode)

Thực hiện (execute)

Ghi kết quả (write back)

(3) Đường ống nạp/cất giữ lệnh (Load/store instruction pipeline) 5 phân đoạn:

Đọc lệnh

Chuyển đi, giải mã lệnh

Tạo địa chỉ (address generation)

Truy nhập bộ nhớ cache

Cất giữ (write back)

(4) Đường ống xử lý các số dấu phẩy động (FP pipeline) 6 phân đoạn:

Đọc lệnh

Chuyển đi

Giải mã lệnh

Thực hiện 1

Thực hiện 2

Cất giữ (write back)

Đường ống nạp/cất giữ cần phải có một chu kỳ phụ để truy nhập cache (các chu kỳ thêm sẽ phải cần đến trong trường hợp trượt cache). Đường ống dấu phẩy động cũng phải cần đến những chu kỳ phụ để cho một số thao tác cho xử lý các số chính xác kép.

Đơn vị MMU của MPC601 (hình 7.54) hỗ trợ tới 4 Petabyte (2^{52} byte) bộ nhớ ảo và 4 GB (2^{32} byte) bộ nhớ vật lý. MMU cũng điều khiển các thêm quyền truy nhập vào các không gian nhớ này theo các khối và trang.

MPC có 3 loại bộ đệm chuyển đổi địa chỉ TLB, đó là:

1. TLB một trường (UTLB), với 256 điểm vào (256-entry), 2-đường liên hợp tập 92-way set-associativ), cho các trang dung lượng 4 kB.

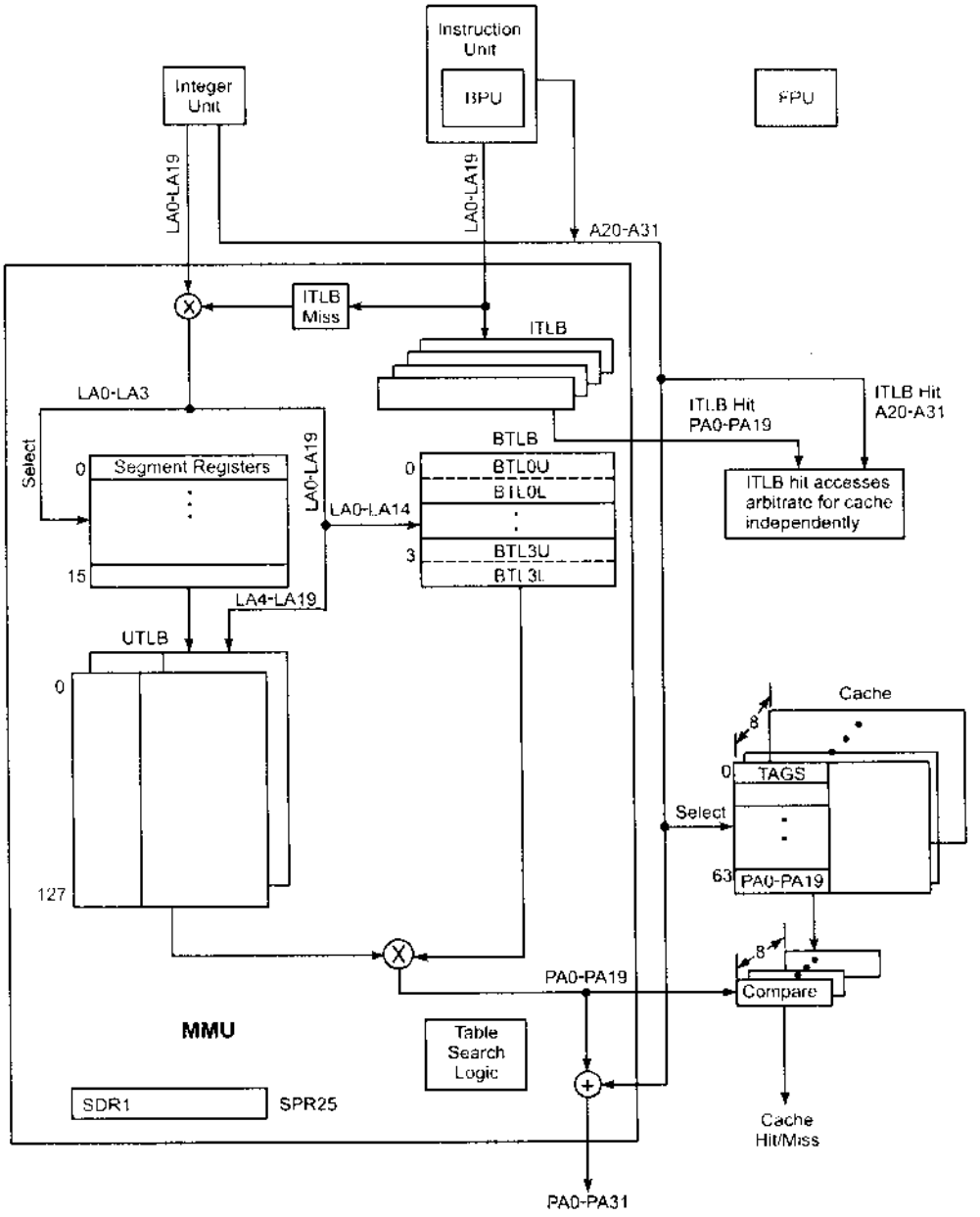
2. TLB khối (BTLB), với 4 điểm vào, liên hợp tập đầy đủ (fully associative), cho các khối có dung lượng từ 128 kB đến 8 MB.

3. TLB lệnh (ITLB), với 4 điểm vào, liên hợp tập đầy đủ, sử dụng hầu hết cho chuyển đổi địa chỉ các lệnh.

Đơn vị thực hiện BPU tạo ra tất cả các địa chỉ của lệnh và dữ liệu. Sau khi một địa chỉ logic (LA) được tạo ra, những bit cao của LA (từ LA0 đến LA19) (hình 7.54) được chuyển đổi thành địa chỉ vật lý (từ PA0 đến PA19) nhờ MMU. Đồng thời, những bit thấp của địa chỉ LA (từ A20 đến A31) được chuyển trực tiếp đến cache bên trong chip để trỏ tới tập hợp và khoản dữ liệu cần truy nhập.

Sau khi chuyển đổi địa chỉ, MMU chuyển 20-bit đến cache để tìm kiếm trùng thẻ. Những bit thấp của LA không tham gia vào chuyển đổi địa chỉ được liên kết với những bit cao đã được chuyển đổi để hình thành địa chỉ vật lý để truy nhập đến bộ nhớ bên ngoài.

16 thanh ghi đoạn 32-bit (SR) điều khiển cả cơ chế chuyển đổi địa chỉ trang và giao tiếp điều khiển vào/ra. Các thanh ghi SRs được chọn nhờ bit bit cao của địa chỉ LA (bit LA0-LA3). Các nội dung của các thanh ghi SR được chọn được liên kết với 28 bit thấp của địa chỉ LA để tạo ra địa chỉ ảo 52-bit. Sự chuyển đổi địa chỉ phù hợp với sự chuyển địa chỉ ảo thành 32-bit địa chỉ vật lý. Trong hầu hết các trường hợp, địa chỉ vật lý của trang có trong UTLB. Nếu không có trong UTLB (trượt UTLB) thì bộ xử lý tự động tìm kiếm các bảng trang trong bộ nhớ.



Hình 7.54: Sơ đồ khối MMU của MPC601

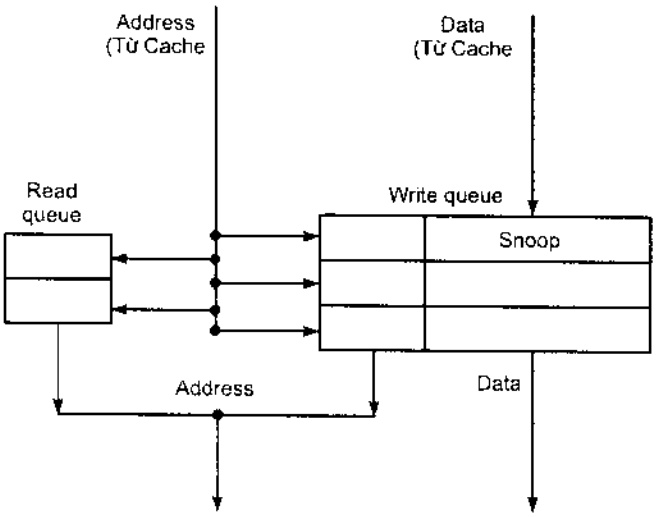
Sự chuyển đổi địa chỉ của khối thực hiện bằng BTLB. Nó cùng thực hiện song song với sự chuyển đổi địa chỉ trang và tương tự như sự chuyển đổi này, nhưng chỉ một số bit cao của LA tham gia chuyển đổi

địa chỉ khối, bởi vì dung lượng các khối lớn hơn nhiều so với với dung lượng 4 kB của trang.

Đối với các truy nhập lệnh, MMU trước hết thực hiện xem xét trong ITLB. Nếu có sự trượt trong ITLB, thì UTLB, và BTLB được xem xét.

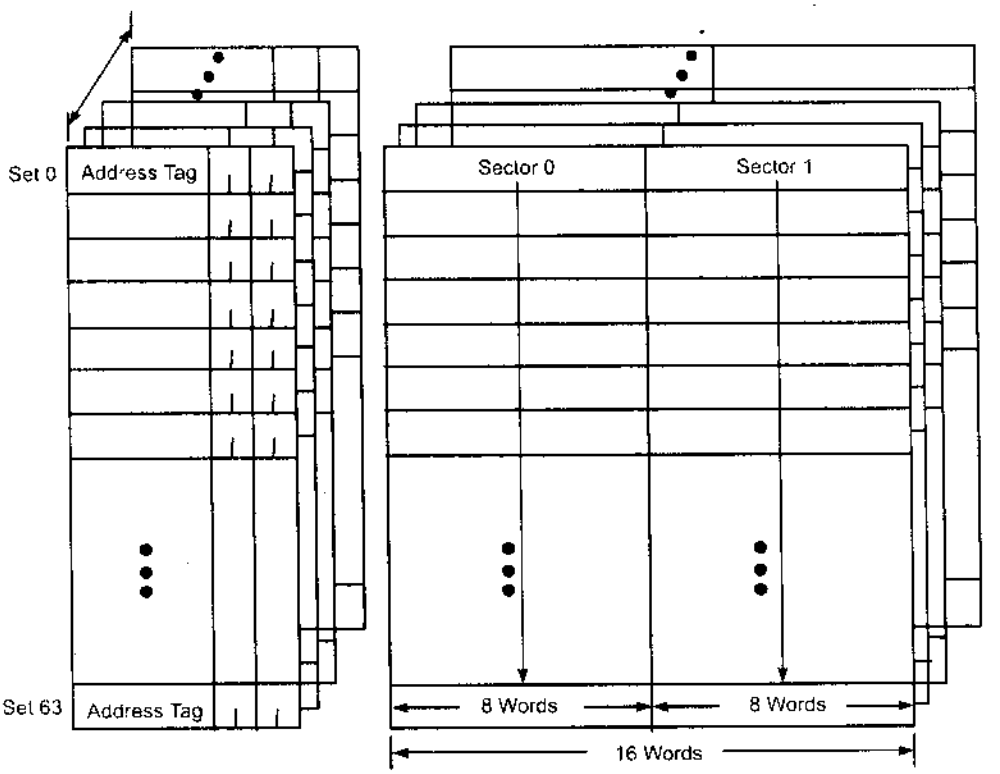
Đơn vị nhớ của MPC601 (hình 7.55) chứa các hàng đợi cho ghi và đọc. Những hàng hàng đợi này là các bộ đệm cho các thao tác giữa giao tiếp bên ngoài và cache.

Các thao tác giữa giao tiếp bên ngoài và cache xảy ra vì cache bị trượt khi thực hiện các lệnh nạp và cất giữ. Và các thao tác này cần để duy trì sự liên kết với cache. Như trong hình 7.55, hàng đợi đọc chứa 2 thành phần và hàng đợi ghi chứa 3 thành phần. Mỗi một thành phần của hàng đợi ghi có thể chứa 8 từ (256 bit hay 32 byte). Mỗi một hàng đợi đọc chứa 32 bit địa chỉ và các bit thông tin điều khiển.



Hình 7.55: Đơn vị nhớ (Memory Unit) của PC601

MPC601 có 32 kB Dcache và 32 kB Icache bên trong chip, 8-đường liên hợp tập, cache 64 byte/đường (64 byte/line cache) như mô tả ở hình 7.56. Mỗi một đường (line) được chia thành 2 sector, mỗi sector 8 từ và có thể được xét duyệt (snooped), được nạp và làm đầy, hoặc bị làm mất hiệu lực độc lập.



Hình 7.56: Tổ chức cache của MPC601

Kích thước của cache là 2^{15} byte, kích thước của đường (line) là 2^6 byte, do đó ta có:

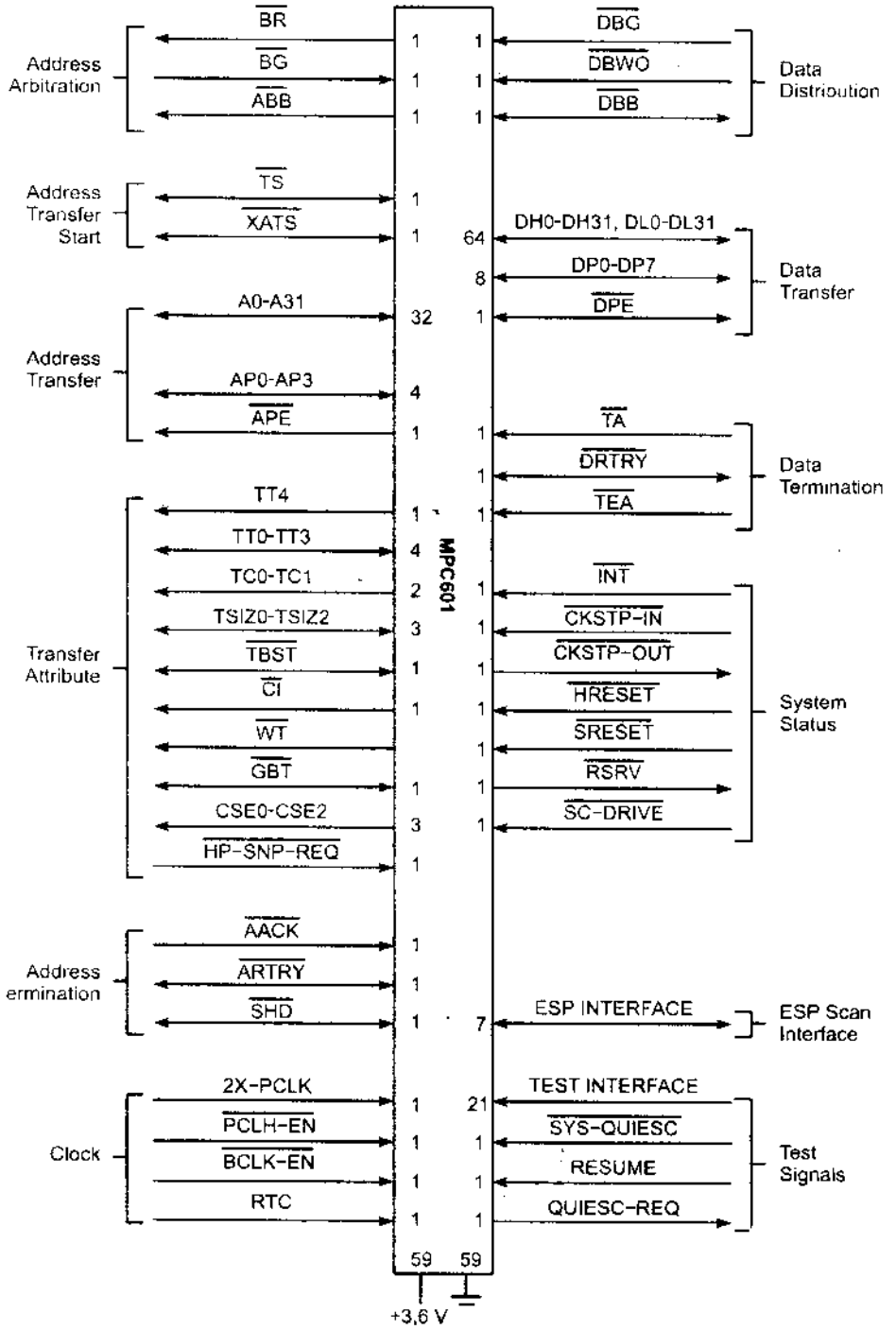
$2^{15}/2^6 = 2^9$ đường trong cache,

$2^9/2^3 = 2^6$ tập hợp trong cache.

Trong khi tạo địa chỉ truy nhập cache, phải cần 6 bit để xác định tổng byte trong một đường (line), và 6 bit để xác định một tập hợp (set), bỏ qua 20 bit cho thẻ (TAG).

PowerPC có thể thực hiện điều khiển các chế độ truy nhập đối với trang hoặc khối nhờ có các đặc tính:

- Có chọn lựa ghi-ghi lại-thông suốt (Write-back/write-through options);
- Chế độ cấm cache (cache-inhibited mode);
- Tính gắn kết bộ nhớ (memory coherence).



Hình 7.57: Các nhóm tín hiệu của MPC601

Sự gắn kết cache được xử lý nhờ giao thức MESI (Modified, Exclusive, Shared, Invalid Protocol - Giao thức thay đổi, thực hiện, chia sẻ và mất hiệu lực), theo 4 trạng thái sau đây:

1. Thay đổi (Modified): đường cache (cache line) được thay đổi và duy trì dữ liệu thực cho địa chỉ này;

2. Loại trừ (Exclusive): đường cache giữ dữ liệu đồng nhất với dữ liệu ở tại địa chỉ này trong bộ nhớ. Không cache nào khác có dữ liệu này;

3. Chia sẻ (Shared): đường cache giữ dữ liệu đồng nhất với dữ liệu ở địa chỉ này trong bộ nhớ và có thể trong cache khác;

4. Mất hiệu lực (invalid): dữ liệu trong đường cache bị mất hiệu lực.

Mỗi một sector có 2 bit để mã hóa 4 trạng thái trên đây. Một thuật toán của LRU được sử dụng để thay thế đường cache.

MPC601 có phân loại các loại trừ (exception) sau đây:

1. Không chính xác không đồng bộ (Asynchronous imprecise) đối với kiểm tra máy và xóa hệ thống (system reset).

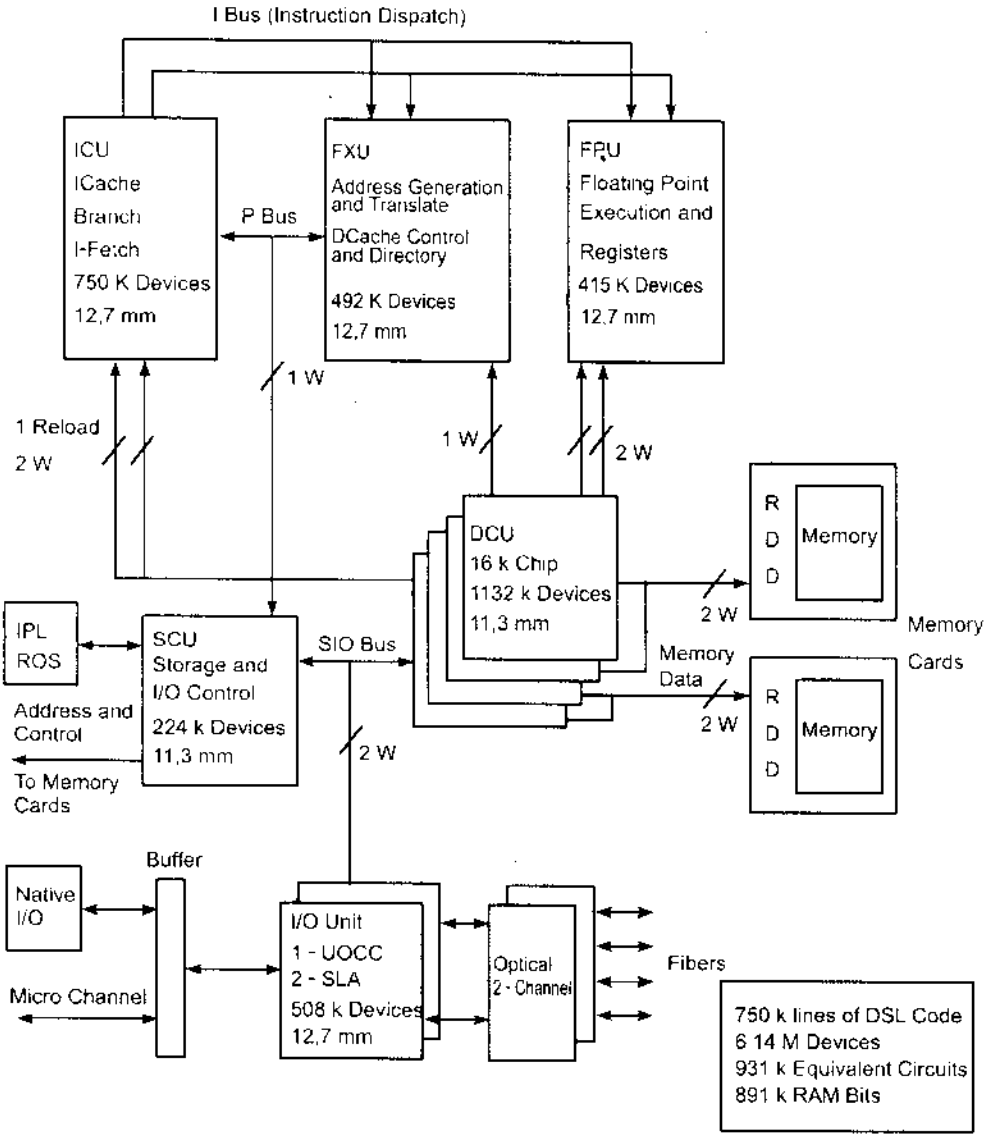
2. Chính xác không đồng bộ (Asynchronous precise) cho các ngắt từ bên ngoài.

3. Chính xác đồng bộ (Synchronous precise) cho các loại trừ gây ra bởi các lệnh.

Các tín hiệu bên ngoài của MPC601 được mô tả trong hình 7.57. PowerPC đã chấp nhận một tập hợp các tín hiệu giao tiếp tương tự như các tín hiệu này ở hệ thống Motorola MC88110 (chúng ta không xét ở đây), nhưng không đồng nhất.

(8) IBM RS/6000

IBM RS/6000 là bộ xử lý có trước kiến trúc PowerPC. Kiến trúc PowerPC dựa vào hầu hết kiến trúc của IBM POWER mà RS/6000 dựa vào. Hầu hết các lệnh của kiến trúc IBM POWER thực tế thực hiện được ở PowerPC. Khoảng 34 lệnh của kiến trúc POWER không thể thực hiện được trên PowerPC. Sơ đồ khối của RS/6000 cho trong hình 7.58.



Hình 7.58: Sơ đồ khối của IBM RS/6000

Phiên bản đầu tiên của RS/6000 không phải là đơn chip mà gồm có các đơn vị nhỏ như sau:

1. Đơn vị cache lệnh ICU (Instruction Cache Unit)
2. Đơn vị dấu phẩy cố định FXU (FiXed-point Unit)
3. Đơn vị dấu phẩy động FPU

4. Đơn vị cache dữ liệu DCU (Data Cache Unit)

5. Đơn vị điều khiển bộ nhớ SCU (Storage Control Unit)

6. Nạp ban đầu chương trình IPL (Initial Program Load)

7. Bộ nhớ chỉ đọc ROS (Read-Only Storage)

8. Điều khiển kênh vào/ra IOCC (I/O Channel Controller)

9. Bộ thích ứng liên kết tuần tự SLA (Serial Link Adapter)

10. Bus vào/ra hệ thống (SIO bus)

11. Mã mô tả ngôn ngữ (DSL code-DeScription Language code)

Kích thước của bus mô tả trong hình 6.12 được tính theo đơn vị từ, mỗi từ 32-bit. Mỗi khối trong hình 6.12 thể hiện một chip riêng biệt (ngoại trừ các khối như các cạc nhớ (memory cards), vào/ra, và IPL ROS. RS/6000 có tới 3 chip FPU. ICU chứa 8 kB Icache và một bộ xử lý rẽ nhánh. Bộ xử lý rẽ nhánh xử lý dòng lệnh đi từ Icache và đẩy một dòng đều đặn các lệnh tới FXU và FPU.

RS/6000 có kiến trúc siêu hướng 4 cửa ra. Một tổ hợp đặc biệt của 4 lệnh có thể được thực hiện đồng thời, ví dụ: 1 lệnh rẽ nhánh, 1 lệnh xử lý số nguyên, 1 lệnh xử lý số dấu phẩy động, và 1 lệnh của CR. Bộ xử lý rẽ nhánh đảm bảo tất cả các rẽ nhánh, ngắt, và các chức năng mã điều kiện bên trong hệ thống. Cũng có phiên bản đơn chip của RS/6000.

Đường ống lệnh của RS/6000 có 5 giai đoạn: đọc lệnh (fetch), phân nhánh (dispatch), giải mã lệnh (decode), thực hiện lệnh (execute), và ghi lại (write back).

Tiếp theo kiến trúc POWER, IBM đã phát triển kiến trúc POWER2. Kiến trúc POWER2 có 8 vi mạch vi xử lý với tổng cộng 20 triệu transistor làm việc ở 72 MHz. Nó tương tự như kiến trúc POWER, chỉ có một số ít khác biệt. POWER2 có Icache 32 kB nằm trên cùng chip với đơn vị rẽ nhánh. Icache của POWER2 là 2-đường liên hợp tập với 128 byte/đường. Dcache của POWER2 có dung lượng 256 byte và được thực hiện trên 4 chip. Nó là 4-đường liên hợp tập với 256 byte/đường.

7.4.5 Họ vi xử lý DEC Alpha

a) Giới thiệu DEC Alpha

Hãng DEC, hay gọi đơn giản là Digital đã nổi tiếng với vai trò tiên phong trong các sản phẩm máy tính loại nhỏ, như họ máy PDP-11, họ máy tính VAX. Cả hai họ máy PDP và VAX đã được sử dụng rộng rãi trên toàn thế giới trong nhiều năm. Họ VAX là một sản phẩm điển hình của kiểu kiến trúc CISC, có sự mở rộng tập lệnh, định dạng (format), và các chế độ đánh địa chỉ. Đồng thời với việc mở rộng chức năng và đa dạng mẫu mã của họ VAX, DEC đang nghiên cứu phát triển của bộ vi xử lý có tập lệnh giảm thiểu RISC (Reduced Instruction Set Computing). Alpha AXP là đỉnh cao trong năm 1991 và là máy điện toán có tập lệnh thu gọn RISC. Trước đó, Digital đã sử dụng bộ vi xử lý kiểu RISC R3000 và MIPS R2000 trong hệ thống các trạm của DEC. Nó là bộ vi xử lý kiến trúc hai siêu hướng (two-way instruction issue superscalar) (tức là trong 1 đường ống pipeline, nó có thể đưa vào 2 lệnh đồng thời trong 1 chu kì nhịp). Nó có một đơn vị xử lý dấu phẩy động FPU (Floating-Pointed Unit), đơn vị quản lý bộ nhớ MMU (Memory Management Unit) và một bộ nhớ cache kép (8 kB Icache và 8 kB Dcache). Tần số đạt được ban đầu là 150 MHz, sau đó đạt tới 300 MHz. Nó vượt qua tất cả các bộ vi xử lý cùng thời về mặt tần số hoạt động (ít nhất là trong các năm 1992-1994). Nó thực sự được gọi là bộ vi xử lý có tốc độ cao nhất của đầu năm 1990. Alpha AXP bắt đầu một họ Digital mới của bộ vi xử lý kiểu RISC. Năm 1997, chip vi xử lý Alpha 21164 ra đời và được coi là chip vi xử lý có tốc độ nhanh nhất hiện nay, nó có những đặc điểm chính như sau:

- Kiến trúc 4-siêu hướng (đồng thời trong đường ống xử lý 4 lệnh)
- 64-bit đầy đủ và kiến trúc RISC
- Công nghệ CMOS 0,35 micron
- Icache 8 kB, Dcache 8 kB cấp 1 (L1) trong chip
- Icache và Dcache 96 kB 3-đường, liên hiệp tập (set-associative), ghi lại (write-back) cấp 2 (L2) trong chip

- Điều khiển cache cấp 3 (L3) bên ngoài chip đồng bộ và không đồng bộ với bộ nhớ SRAM

- Có MMU, FPU bên trong

- Tốc độ 2,4 BIPS (tỷ lệnh/s) với tần số làm việc 600 MHz

- Đóng vỏ 499 chân ceramic IPGA (Interstitial Pin Grid Array)

- Tiêu thụ nguồn Vdd: 3,3 V ± 5%, Vdd: 2,5 V± 0,1 V

b) Kiến trúc của vi xử lý Alpha AXP

Alpha là bộ vi xử lý kiểu RISC, nó có các đặc tính phù hợp với RISC. Đặc biệt nó thực hiện các thao tác thanh ghi với thanh ghi (Register to Register), đó là các thao tác mà kết quả và toán hạng đều nằm trong các thanh ghi của CPU. Tất cả các lệnh của Alpha có chiều dài 32 bit, giống như các bộ vi xử lý kiểu RISC khác. Trong hầu hết các bộ vi xử lý kiểu RISC lệnh của nó gồm 3 toán hạng. Alpha là bộ vi xử lý 64-bit, tất cả các thanh ghi đa năng (general purpose Register) đều là 64-bit. Các bit trong thanh ghi được đánh số: bit LSB là 0, bit MSB là 63. Alpha có 2 bộ 32 thanh ghi 64-bit: một dùng cho đơn vị số nguyên IU (Interger Unit) từ R0, R1,...,31, và một dùng cho đơn vị dấu phẩy động FPU (Floating-Point Unit) từ F0, F2,..., F31, trong đó R31 và F31 luôn chứa giá trị 0 giống như các kiểu RISC khác (nhưng ở RISC khác thường là R0). Thanh đếm chương trình (PC) là một thanh ghi 64-bit riêng biệt. Tất cả hoạt động của Alpha đều được thao tác trong các thanh ghi 64-bit của CPU. Kiểu cấu trúc này cũng hỗ trợ các thao tác số nguyên 32-bit. Alpha đánh địa chỉ kiểu byte, đơn vị đánh địa chỉ cơ bản là byte. Các byte của địa chỉ ảo được đánh theo kiểu *litle-endian* (tức là các bit thấp thì có địa chỉ byte thấp). Địa chỉ ảo nhỏ nhất có 43 bit.

(1) Sơ đồ khối của vi xử lý Alpha 21064

Phiên bản đầu tiên của các vi xử lý kiến trúc Alpha là chip 21064. Nó có công nghệ 0,75micron CMOS đóng vỏ PGA 431-chân, trong đó có 140 pin dùng để kết nối nguồn và đất. Nó chứa 1,68 triệu

transistor trong kích thước chip 16,8 × 13,9 mm, 3 lớp kim loại, và dùng nguồn 3,3 V. Tần số làm việc từ 150 MHz đến 200 MHz. Tiêu tán nhiệt ở tần số 200 MHz là 30 W.

Chip 21064 có kiến trúc siêu hướng 2-cửa ra, có cache đôi: Icache và Dcache đều có dung lượng 8 kB. Nó có bộ đệm ghi với 4 điểm vào và 32 byte/điểm vào. Có đường ống nguyên 64-bit với tệp thanh ghi dung lượng 32 × 64-bit, và đường ống dấu phẩy động với tệp thanh ghi dung lượng 32 × 64-bit. Có hỗ trợ kết nối cache bên ngoài dung lượng từ 128 byte đến 8 MB.

Các cache bên là các cache sắp xếp trực tiếp, tất cả chúng đều có 32 byte/đường. Dcache là cache vật lý có đặc điểm ghi thông suốt, đọc phân phối.

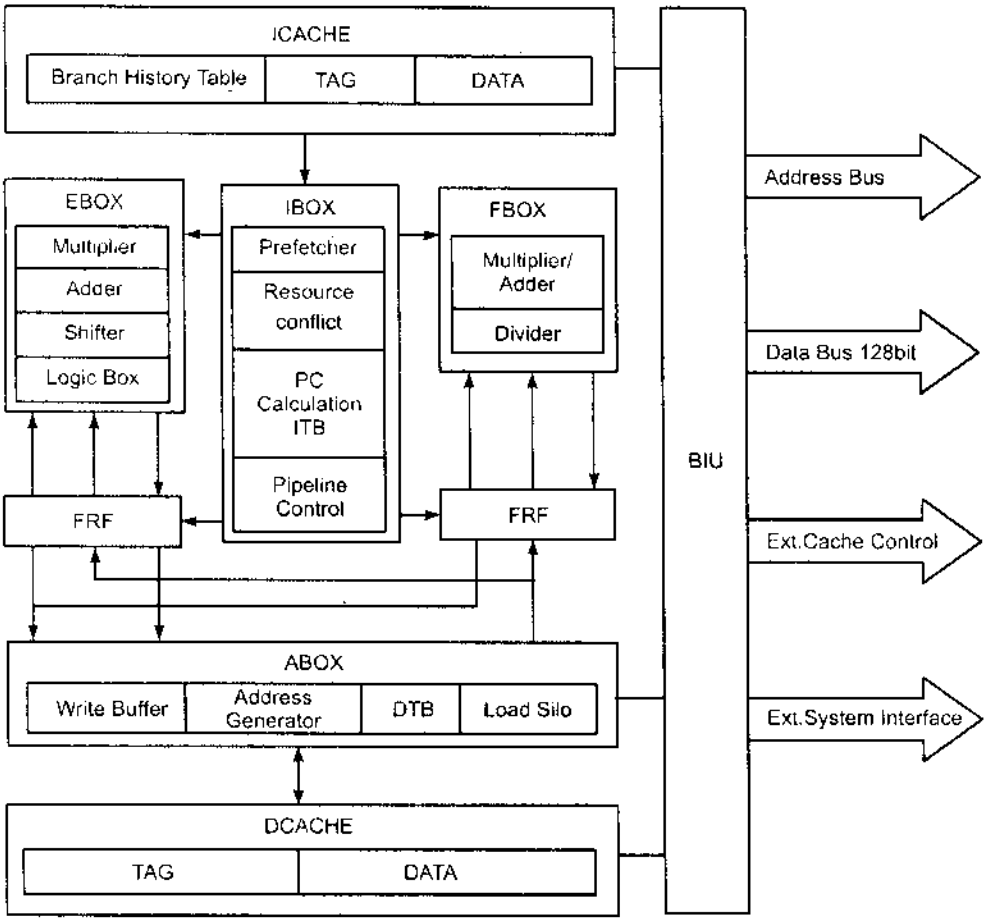
Sơ đồ khối của Alpha 61064 cho trong hình 7.59, trong đó có các khối chức năng như sau:

1. IBOX- đưa ra các lệnh (2 lệnh đồng thời), bảo đảm đường ống nguyên, và thực hiện tính giá trị PC. Nó giải mã đồng thời 2 lệnh và kiểm tra sự tồn tại của các tài nguyên. Không có thứ tự cho đưa ra các lệnh mà phụ thuộc vào có hay không tài nguyên tương ứng cho lệnh. IBOX chứa logic đoán trước rẽ nhánh, các bộ đệm chuyển đổi lệnh ITB (Instruction Translator Buffers), logic ngắt và các bộ đếm hiệu xuất (đưa ra, không đưa ra, tổng số các chu kỳ, rỗng đường ống, trượt cache). Hai loại ITB là:

• ITB trang nhỏ (small-page ITB), 8 điểm vào, liên hợp toàn bộ (fully associative), chứa các điểm vào bảng trang của chuỗi lệnh hiện hành (PTE) cho các trang 8 kB;

• ITB trang lớn (large-page ITB), 4 điểm vào, liên hợp toàn bộ, dùng cho các trang dung lượng 512 × 8 kB = 4 MB.

2. EBOX-đơn vị thực hiện nguyên. Nó chứa bộ cộng 64-bit, hộp logic, thanh ghi dịch ống (barrel shifter), bộ nhân số nguyên, 32 x 64 IRF với 4 cổng đọc và 2 cổng ghi.



Hình 7.59: Sơ đồ khối của 21064

3. ABOX-đơn vị tạo địa chỉ. Nó chứa phần chuyển đổi địa chỉ, ống nạp dữ liệu (load silo), giao tiếp Dcache, các thanh ghi bên trong của vi xử lý (IPR), BIU, và bộ đệm chuyển đổi dữ liệu liên hợp toàn bộ, 32 điểm vào (DTB). Ống nạp là đường ống tham chiếu bộ nhớ có thể tiếp nhận lệnh nạp hoặc lệnh cất giữ mới trong từng chu kỳ cho đến khi Dcache được yêu cầu làm đầy. BIU có bus dữ liệu 128-bit.

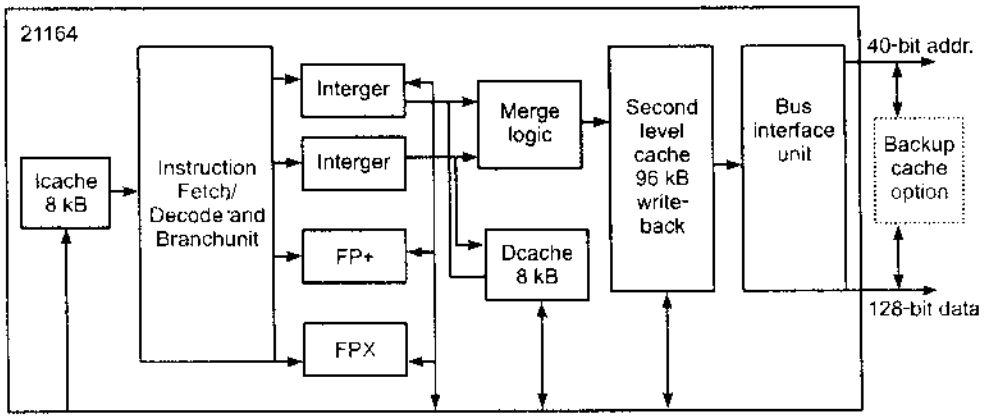
4. FBOX-là FPU. Nó chứa các đơn vị thao tác cho phép cộng và một tệp thanh ghi dung lượng 32 x 64 (FRRF), và thanh ghi điều khiển dấu phẩy động FPCR.

5. Icache- cache lệnh, dung lượng 8 kB, sắp xếp trực tiếp, được đánh địa chỉ vật lý, 32 byte/đường.

6. Dcache- cache dữ liệu, dung lượng 8 kB, sắp xếp trực tiếp, được đánh địa chỉ vật lý, 32 byte/đường, ghi thông suốt, đọc phân phối.

(2) Sơ đồ khối chức năng của vi xử lý Alpha 21164

Alpha 21164 là chip vi xử lý ra sau chip 21064 (hình 7.60). Nó gồm có 5 đơn vị chức năng độc lập: đơn vị đọc lệnh (instruction fetch), giải mã lệnh (decode) và rẽ nhánh (branch unit); đơn vị tính các số nguyên (interger execution unit); MMU; đơn vị điều khiển cache và giao tiếp bus (cache control and bus interface unit); và FPU. Có 3 cache bên trong chip: Icache (8 kB), Dcache (8 kB), và cache mức 2 (L2) 96 kB.



Hình 7.60: Sơ đồ khối của chip Alpha 21164

Alpha 21164 tiêu tán khoảng 28 W ở tần số 366 MHz, do đó chỉ cần áp dụng phương pháp toả nhiệt thông thường: bộ tản nhiệt và quạt cũng đảm bảo tin cậy.

Alpha 21164 có một số loại khác nhau dùng cho các hệ điều hành Windows NT và UNIX/VMS như sau:

21164-P4	300 MHz	NT
21164-P5	366 MHz	NT

21164-EB	366 MHz	UNIX/VMS
21164-P6	433 MHz	NT
21164-JB	433 MHz	UNIX/VMS
21164-P7	500 MHz	NT
21164-P8	533 MHz	NT
21164-MB	600 MHz	UNIX/VMS

(3) Các kiểu dữ liệu của Alpha

Alpha có các kiểu dữ liệu như sau:

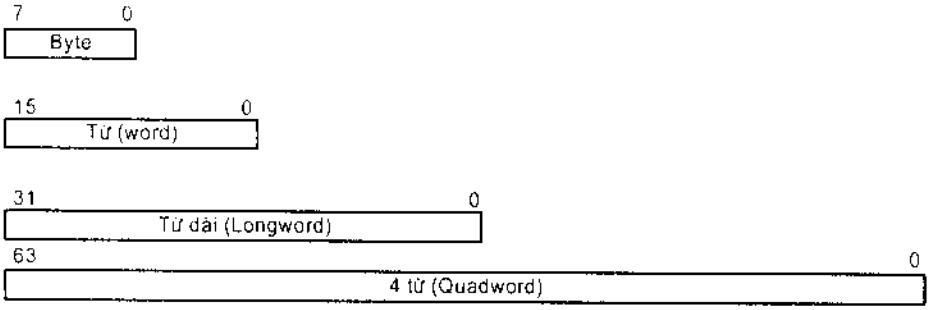
• *Các kiểu dữ liệu nguyên (hình 7.61)*

1. Kiểu byte: 8-bit, đơn vị đánh địa chỉ cơ bản.

2. Kiểu từ (word): 16-bit. Là hai byte liên tục nhau. Địa chỉ của word chính là địa chỉ của byte thấp (byte mà chứa bit 0).

3. Kiểu từ dài (longword): 32-bit. Là bốn byte liên tục nhau. Địa chỉ của longword chính là địa chỉ byte thấp của nó. Bit MSB gọi là bit dấu.

4. Kiểu từ 4 từ (quadword): 64-bit. Là tám byte liên tục nhau. Địa chỉ của quadword chính là địa chỉ byte thấp của nó. Bit MSB gọi là bit dấu.



Hình 7.61: Khuôn dạng dữ liệu nguyên của DEC ALPHA

Mặc dù dữ liệu kiểu từ (word), từ dài (long word), 4 từ (quadword) được cất ở một địa chỉ bất kỳ nhưng sẽ được xử lý tốt hơn nếu chúng được sắp xếp. Đó là, dữ liệu kiểu từ dài được cất ở địa chỉ

có thể chia được cho 4. Ví dụ nếu có 2 từ dài liên tiếp nhau, từ dài đầu có địa chỉ là 1000 thì từ dài tiếp sau có địa chỉ là 1100 (2 bit thấp của địa chỉ là 0), và 4 từ (quadword) được cất trong địa chỉ có thể chia được cho 8 (3 bit thấp của địa chỉ là 0).

• *Kiểu dữ liệu dấu phẩy động*

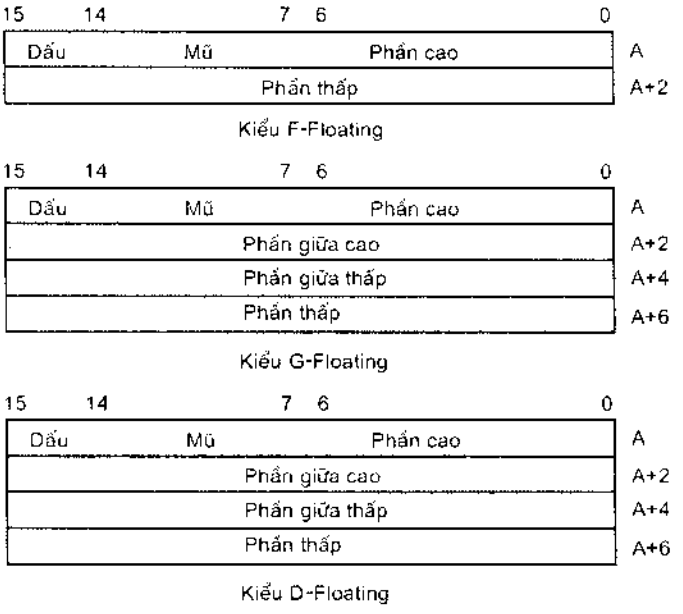
Kiểu dữ liệu dấu phẩy động được chia làm 2 nhóm:

(1) Nhóm dữ liệu dấu phẩy động kiểu VAX, bao gồm:

a. F loating, 32-bit

b. G floating, 64-bit

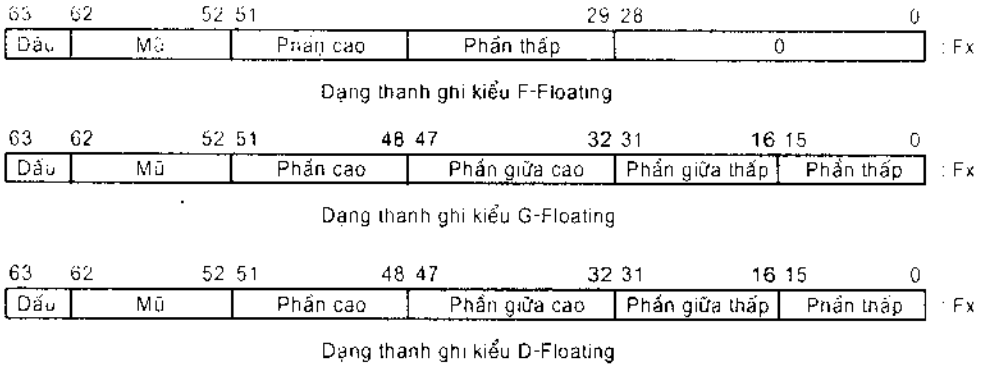
c. D floating, 64-bit



Hình 7.62: Lưu trữ dữ liệu kiểu VAX trong bộ nhớ

Việc lưu trữ trong bộ nhớ của kiểu dữ liệu VAX được minh họa ở hình 7.62 và dạng các thanh ghi F, G, D của kiểu VAX được minh họa ở hình 7.63. Mặc dù các kiểu dữ liệu được cất ở địa chỉ byte bất kỳ trong bộ nhớ, nhưng tốt hơn nếu chúng được sắp xếp (được cất ở địa chỉ có thể chia được cho 4 đối với kiểu F và chia được cho 8 đối với

kiểu D và G). Điểm khác nhau cơ bản giữa G và D là kiểu G có 11-bit luỹ thừa trong khi kiểu D có 8-bit, do đó kiểu G có dải số cao hơn. Kiểu D không hoàn toàn phù hợp với bộ vi xử lý Alpha, nó không được cung cấp các phép toán số học kiểu D mà được cung cấp bởi phần mềm mô phỏng.



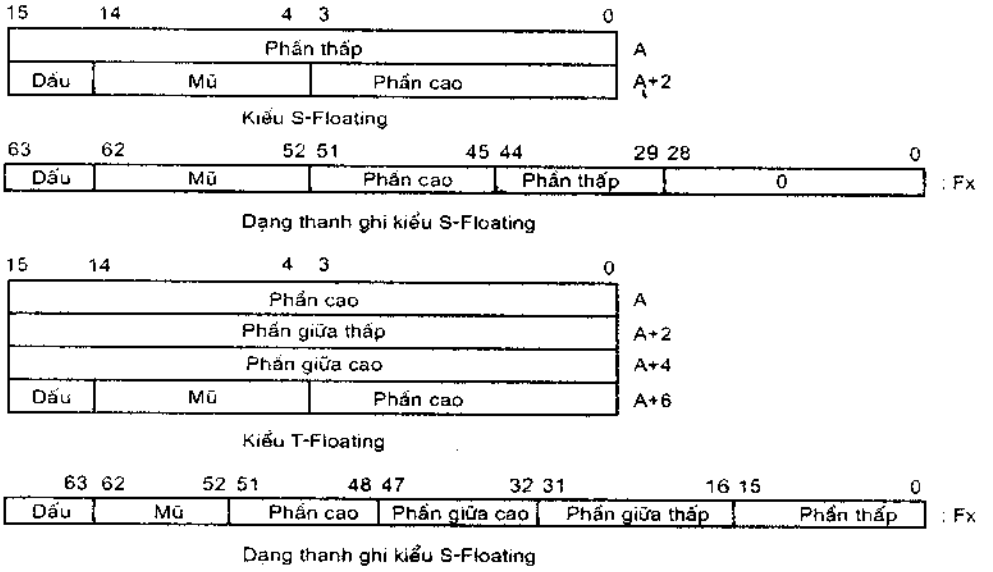
Hình 7.63: Các dạng thanh ghi của DEC ALPHA dùng cho máy tính VAX

Trong kiểu F, có 8-bit lũy thừa được mã hóa từ 0 - 255, và dưới dạng nhị phân nó có giá trị từ -127 đến 127. Một lũy thừa 0 với bit dấu là 0 thì được coi là 0. Một lũy thừa 0 với bit dấu là 1 được coi như *toán hạng dự phòng* (reserved operand). Nếu cố gắng sử dụng 1 toán hạng dự phòng trong lệnh dấu phẩy động sẽ gây ra một *lỗi số học loại trừ* (arithmetic exception). Dải số của F xấp xỉ từ $0,29(10)^{-38}$ đến $1,7(10)^{38}$. Độ chính xác xấp xỉ 2^{-23} với 7 chữ số sau dấu thập phân.

Trong kiểu G có 11 bit lũy thừa mã hóa từ 0 đến 2047, và dưới dạng nhị phân nó có giá trị từ -1032 đến 1023. Một luỹ thừa 0 với bit dấu là 0 được coi như là 0. Một luỹ thừa 0 với bit dấu là 1 được coi như 1 toán hạng dự phòng. Dải số của G xấp xỉ từ $0,56(10)^{-308}$ đến $0,9(10)^{308}$. Độ chính xác của G là 2^{-52} với 15 chữ số sau dấu thập phân. Độ chính xác của D là 2^{-55} với 16 chữ số sau dấu thập phân.

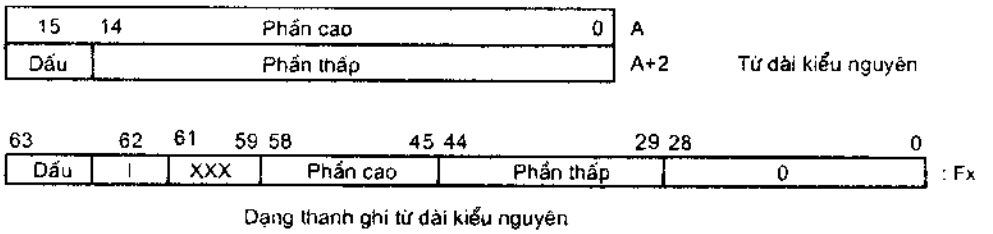
(2) Dạng dấu phẩy động theo chuẩn IEEE

Theo chuẩn IEEE có số chính xác đơn 32-bit (S-Floating) và số chính xác kép 64-bit (T-Floating). Dữ liệu dấu phẩy động theo chuẩn này được mô tả ở hình 7.64.

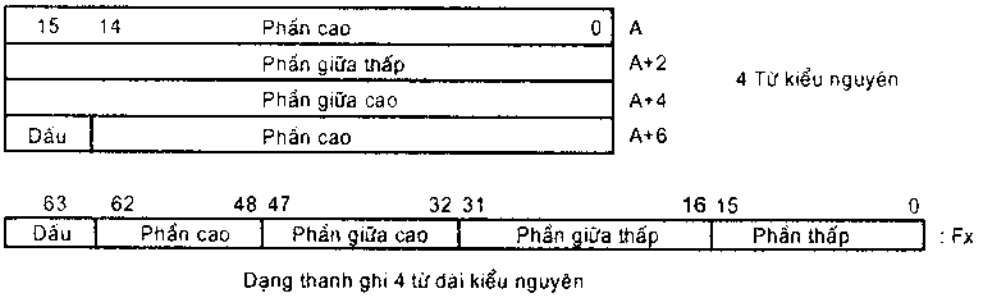


Hình 7.64: Dạng dữ liệu theo chuẩn IEEE

Chúng có thể cất ở bất kỳ đâu trong bộ nhớ. Nhưng tốt hơn là nên biểu diễn có sắp xếp như dạng VAX. Các số nguyên từ dài (longword) và 4 từ quadword được cất trong các thanh ghi FPU và được mô tả ở hình 7.65.



Hình 7.65a: Lưu giữ dữ liệu kiểu nguyên và các dạng thanh ghi của FPU



Hình 7.65b: Lưu giữ dữ liệu kiểu nguyên và các dạng thanh ghi của FPU

(4) Các dạng lệnh

Tất cả các lệnh của Alpha đều 32-bit và trong đó sử dụng các ký hiệu với nghĩa như sau:

Ra, Rb, Rc là các toán hạng trong thanh ghi số nguyên.

Fa, Fb, Fc là các toán hạng trong thanh ghi dấu phẩy động.

Disp là độ dịch, cộng với giá trị trong Rb tạo nên một địa chỉ ảo.

SBZ = 0

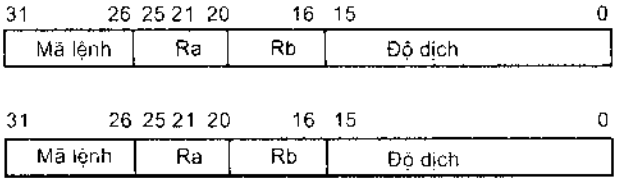
LIT là các giá trị 8-bit từ 0 đến 255.

Các lệnh đều có vùng mã lệnh dài 6-bit (bit <31:26>). Bất kỳ vùng thanh ghi (5 bit) nào không được sử dụng trong lệnh (Ra, Rb, Fa, Fb) phải được đặt giá trị 31 (11111).

Có 5 dạng lệnh trong vi xử lý Alpha:

• *Dạng lệnh nhớ (memory instruction format)*

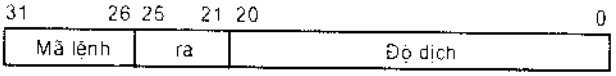
Nhóm lệnh này dùng để trao đổi thông tin giữa thanh ghi và bộ nhớ, để nạp *địa chỉ hiệu dụng* EA (Effective Address) và cho các thủ tục nhảy. Trường độ dịch (memory-disp) có độ rộng 2 byte. Đây là phần *mở rộng dấu* (sign-extended) được cộng với nội dung của thanh ghi Rb tạo thành địa chỉ ảo. Địa chỉ ảo được sử dụng như một địa chỉ nạp/cất giữ (load/store) bộ nhớ hay như một giá trị kết quả, phụ thuộc vào lệnh cụ thể. Đối với một số lệnh, memory-disp được thay thế bằng trường chức năng. Nó phục vụ như là một phần mở rộng của mã lệnh cho các lệnh phụ (miscellaneous instructions).



• *Dạng lệnh rẽ nhánh (Branch instruction format)*

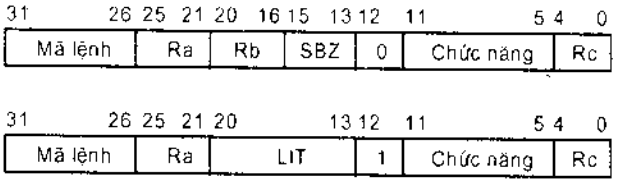
Dạng này được sử dụng trong các lệnh rẽ nhánh có điều kiện (khi đó, Ra chứa mã điều kiện) và cho các thủ tục nhảy. Khi lệnh được

giải mã, giá trị PC tăng lên, nó chỉ đến một lệnh tiếp theo. Giá trị mới của PC được gọi là *updated PC* (cập nhật thanh đếm chương trình). Trường độ dịch (branch_disp) là độ dịch 1 từ dài (longword). Nó dịch trái 2 bit, mở rộng dấu đến 64 bit và cộng với giá trị updated PC tạo ra địa chỉ ảo đích (target virtual address).



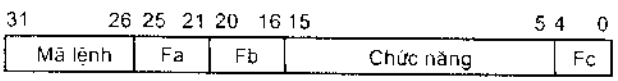
• *Dạng lệnh thao tác (operate instruction format)*

Dạng này dùng cho những lệnh thực hiện thao tác giữa các thanh ghi của số nguyên. Ra, Rb là toán hạng nguồn, vùng Rc là toán hạng đích. Vùng chức năng là mã lệnh mở rộng. Nếu bit 12 có giá trị 0 thì Rb được coi là toán hạng thanh ghi nguồn. Nếu bit 12 có giá trị 1, thì từ 8 bit, bit <20:13 > sẽ là phần hằng số. Hằng số là số nguyên dương từ 0 đến 255 và mở rộng các bit giá trị 0 (zero_extended) đến 64 bit.



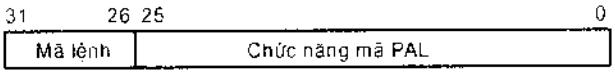
• *Dạng lệnh thao tác dấu phẩy động (floating - point operate instruction format)*

Dạng này dùng cho những lệnh thực hiện thao tác giữa các thanh ghi đối với dấu phẩy động. Fa, Fb là các toán hạng nguồn, Fc là toán hạng đích. Các lệnh chuyển đổi dấu phẩy động sử dụng một vài lệnh này và thực hiện các thao tác chuyển đổi giữa các thanh ghi. Fb là toán hạng nguồn, Fa là F31, Fc là toán hạng đích.



• *Dạng lệnh PAL (PALcode instruction format)*

Dạng này được sử dụng để mở rộng các chức năng của bộ xử lý. Trường chức năng PALcode có 26 bit và xác định thao tác PALcode cụ thể. Các toán hạng nguồn và đích đối với các lệnh PALcode nằm trong các thanh ghi cố định. Nếu mã lệnh là 0 và có chức năng PALcode là 0 thì ta có lệnh HALT.



(6) Các chế độ địa chỉ (address mode)

Kiến trúc Alpha có 4 chế độ địa chỉ đơn giản:

1. Thanh ghi (Register): toán hạng nằm trong thanh ghi của CPU.

2. Ngay lập tức (Immediate): toán hạng là 1 hằng số.

3. Gián tiếp thanh ghi với độ dịch (Register indirect with disp): địa chỉ EA bằng địa chỉ trong thanh ghi CPU cộng với độ dịch (disp).

4. Tương đối với PC (PC-relative): giống như chế độ 3, nhưng địa chỉ là PC.

(7) Tập lệnh

Tập lệnh của các loại vi xử lý Alpha gồm có:

- Các lệnh nạp/cất giữ số nguyên (interger load and store)
- Điều khiển số nguyên (interger control)
- Các lệnh số học (interger arithmitic)
- Các lệnh logic và dịch (logical and shift)
- Các lệnh xử lý byte (byte manipulation)
- Các lệnh nạp/cất giữ với dấu phẩy động (floating-point load and store)
- Điều khiển dấu phẩy động (floating-point control)
- Thao tác với dấu phẩy động (floating-point operate)
- Các lệnh hỗn hợp (miscellaneous)

Các toán hạng của lệnh có dạng sau:

<name><access type><data type>

Trong đó:

<name>: có thể là một trong các thanh ghi Ra, Rb, Rc, Fa, Fb, Fc hoặc:

disp: độ dịch chuyển

fnc: vùng chức năng PAL của lệnh

#b: Toán hạng số nguyên trong vùng Rb của lệnh

<access type>: là một chữ cái biểu thị kiểu truy nhập của toán hạng. Nó có thể là các chữ:

a: được sử dụng trong tính toán địa chỉ

al: trong phạm vi bốn byte (long word)

aq: trong phạm vi tám byte(quan word)

ab: trong phạm vi 1 byte

i: toán hạng là 1 số

r: chỉ đọc

m: cả đọc và ghi

w: chỉ ghi

<data type>: là một chữ cái biểu thị kiểu dữ liệu của toán hạng, có thể là các chữ:

b: byte

f: F-floating

g: G floating

l: longword

q: quadword

s: S - floating

t: T - floating

w: word

x: được chỉ rõ bởi lệnh

• *Các lệnh nạp và cất số nguyên (interger load and store instructions)*

Các lệnh này được tổng kết ở bảng 7.3.

Bảng 7.3: Các lệnh nạp cất giữ các số nguyên

Tên lệnh	Toán hạng	Thao tác
LDA	Ra.wq,disp.ab(Rb.ab)	Nạp địa chỉ
LDAH	Ra.wq,disp.ab(Rb.ab)	Nạp địa chỉ cao
LDL	Ra.wq,disp.ab(Rb.ab)	Nạp phần mở rộng dấu của longword
LDQ	Ra.wq,disp.ab(Rb.ab)	Nạp quadword
LDQ-U	Ra.wq,disp.ab(Rb.ab)	Nạp quadword không sắp xếp
LDL-L	Ra.wq,disp.ab(Rb.ab)	Load SE longword locked
LDQ-L	Ra.wq,disp.ab(Rb.ab)	Load quadword locked
STL	Ra.rq,disp.ab(Rb.ab)	Cất longword
STQ	Ra.rq,disp.ab(Rb.ab)	Cất quadword
STL-C	Ra.mq,disp.ab(Rb.ab)	Cất longword có điều kiện
STQ-C	Ra.mq,disp.ab(Rb.ab)	Cất quadword có điều kiện
STQ-U	Ra.mq,disp.ab(Rb.ab)	Cất quadword không sắp xếp

Trong tất cả các lệnh này, ngoại trừ lệnh LAHD, địa chỉ bộ nhớ ảo được tính bằng cách cộng nội dung thanh ghi Rb với độ dịch 16-bit.

Ra ← Rbv + SE (disp)

Rbv: là giá trị trong Rb

SE: phần mở rộng dấu (sign extended) trong lệnh LDAH độ dịch được dịch trái 16 bit, tức là nhân với $2^{16} = 65536$ và được cộng với dấu mở rộng cùng với Rbv:

Ra ← Rbv + SE (disp * 65536)

• *Các lệnh điều khiển số nguyên (interger control instructions)*

Các lệnh điều khiển số nguyên gồm các lệnh rẽ nhánh có điều kiện, không có điều kiện và các lệnh nhảy. Chúng được tổng kết ở bảng 7.4.

Bảng 7.4: Các lệnh điều khiển số nguyên

Tên lệnh	Toán hạng	Thao tác
BEQ	Ra.rq,disp.al	Rẽ nhánh nếu Rav=0
BGE	Ra.rq,disp.al	Rẽ nhánh nếu Rav >=0
BGT	Ra.rq,disp.al	Rẽ nhánh nếu Rav >0
BLBC	Ra.rq,disp.al	Rẽ nhánh nếu Ra LSB =0
BLBS	Ra.rq,disp.al	Rẽ nhánh nếu Ra LSB =1
BLE	Ra.rq,disp.al	Rẽ nhánh nếu Rav <= 0
BLT	Ra.rq,disp.al	Rẽ nhánh nếu Rav <0
BNE	Ra.rq,disp.al	Rẽ nhánh nếu Rav ! = 0
BR	Ra.wq,disp.al	Rẽ nhánh không điều kiện
BSR	Ra.wq,disp.al	Rẽ nhánh đến 1 thủ tục
JMP	Ra.wq,(Rb.ab),hint	Nhảy
JSR	Ra.wq,(Rb.ab),hint	Nhảy đến 1 thủ tục
RET	Ra.wq,(Rb.ab),hint	Trở về từ thủ tục
JSR-COROUTINE	Ra.wq,(Rb.ab),hint	Nhảy đến 1 thủ tục, trở về

Trườngđộ dịch (disp <15:14>) dùng để mã hóa các thông tin sau:

disp<15:14>	Tên lệnh	Predicted Target <15:0>	Prediction stack action
00	JMP	PC+(4*disp<13:0>)	
01	JSR	PC+(4*disp<13:0>)	Push PC
10	RET	Prediction Stack	pop
11	JSR-COROUTINE	Prediction Stack	pop, Push PC

Bốn mã lệnh khác nhau được mã hóa bằng 2 bit ở disp <15:14> và toán hạng chỉ dẫn (Hint) đặt ở disp <13:0>.

Giá trị updated PC (được tăng ngay sau khi lệnh được đọc từ bộ nhớ) được cất trong Ra, và sau đó địa chỉ ảo đích lấy từ Rb nạp vào PC. Hai bit thấp nhất của Rb được bỏ. Các trường Ra, Rb có thể xác định cùng một thanh ghi. Việc tính toán địa chỉ đích được thực hiện nhờ sử dụng giá trị cũ của Rb trước khi giá trị mới được gán.

• *Các lệnh số học của số nguyên (interger arithmetical instructions)*

Nhóm lệnh này được liệt kê ở bảng 7.5. Các toán hạng có thể có hai dạng:

Ra.rq, Rb.rq, Rc.wq hoặc

Ra. rq, #b.ib, Rc.wq.

Lệnh ADD làm việc như sau:

Rc ← Rav + Rbv

Đối với lệnh xử lý từ dài là ADDL, 32 bit cao của Ra và Rb bỏ đi. Tổng được mở rộng dấu:

Rc ← SE[(Rav + Rbv)<31:0>]

Bảng 7.5: Các lệnh số học số nguyên

Tên lệnh	Thao tác
ADDL	Cộng longword
ADDQ	Cộng quadword
S4ADDL	Cộng longword phạm vi 4 byte
S8ADDL	Cộng longword phạm vi 8 byte
S4ADDQ	Cộng quadword phạm vi 4 byte
S8ADDQ	Cộng quadword phạm vi 8 byte
CMPEQ	So sánh = với quadword có dấu
CMPLT	So sánh < với quadword có dấu
CMPLE	So sánh = hoặc < với quadword có dấu
CMPULT	So sánh < với quadword không dấu
CMPULE	So sánh = hoặc < với quadword không dấu
MULL	Nhân longword
MULQ	Nhân quadword
UMULH	Nhân quadword phần cao không dấu
SUBL	Trừ longword
SUBQ	Trừ quadword
S4SUBL	Trừ longword phạm vi 4 byte
S8SUBL	Trừ longword phạm vi 8 byte
S4SUBQ	Trừ quadword phạm vi 4 byte
S8SUBQ	Trừ quadword phạm vi 8 byte

Trong lệnh cộng SxADDx thanh ghi Ra được dịch trái 2 bit (đối với S4) hoặc 3 bit (đối với S8) rồi cộng với với Rb, hoặc với một hằng số, kết quả ghi ở Rc. Trong các lệnh xử lý từ dài, 32 bit cao của Ra và Rb được bỏ đi.

Trong các lệnh so sánh (CMPxx, CMPUxx) giá trị thanh ghi Ra được so sánh với giá trị trong thanh ghi Rb hoặc với một hằng số #b. Nếu điều kiện so sánh thỏa mãn, thanh ghi Rc chứa giá trị 1. Nếu không, Rc chứa giá trị 0;

CMPLE R1, R2, R5; so sánh 4 từ (quadword) có dấu, nhỏ hơn hoặc bằng.

Với Rav = 50, Rbv = 100, thì Rav < Rbv → Rc ← 1 (quan hệ nhỏ hơn hoặc bằng thỏa mãn, do đó Rc = 1).

Trong lệnh nhân MULQ, Rav nhân với Rbv, kết quả 64-bit được cất trong Rc. Trong phép nhân từ dài 32 bit cao của Ra, Rb bị bỏ và kết quả 32-bit với dấu mở rộng cất trong Rc:

Do đó, MULQ:Rc ← Rav* Rbv

Đối với MULL:Rc ← SE[(Rav * Rbv) < 31: 0 >]

Nếu xảy ra hiện tượng tràn trong MULQ, 64 bit thấp được của kết quả cất ở Rc. 64 bit cao của kết quả 128-bit có thể được tạo ra khi sử dụng lệnh HMULH (nhân 4 từ không dấu-unsigned multiply quadword high), mà trong lệnh này chỉ 64 bit cao của kết quả 128-bit không dấu của Ra và Rb (hoặc một hằng số) được cất vào Rc:

Đối với UMULH: Rc ← unsigned(Rav*Rbv) < 127:64>

Trong các lệnh trừ: giá trị của Rb hoặc hằng số được trừ đi giá trị của Ra và kết quả cất vào Rc. Cũng như phép ADD, 32 bit cao của Ra và Rb được bỏ trong phép trừ các từ dài, và kết quả được mở rộng dấu:

Đối với SUBQ: Rc ← Rav - Rbv

Đối với SUBL: Rc ← SE[(Rav - Rbv) < 31:0 >]

Các lệnh trừ SxSUBx làm việc tương tự như các lệnh cộng SxADDx. Ví dụ, trong lệnh S8SUBQ, Ra được dịch trái 3 bit sau đó trừ Rbv, kết quả được ghi vào Rc.

Alpha không có lệnh chia số nguyên, nhưng có lệnh chia dấu phẩu động.

• *Các lệnh logic và dịch (logical and shift instructions)*

Bảng 7.6: Các lệnh dịch và logic

Tên lệnh	Thao tác
AND	logic AND
BIC	logic AND và bù
BIS	logic OR
EQV	logic OR và bù
ORNOT	tương đương với NOT
XOR	exclusive OR
CMOVxx	di chuyển có điều kiện
SLL	dịch trái logic
SRA	dịch phải số học
SRL	dịch phải logic

Các lệnh logic thực hiện các phép tính logic đối với quadword. Các lệnh chuyển có điều kiện thực hiện chuyển có điều kiện từ một thanh ghi đến thanh ghi mà không có rẽ nhánh. Các lệnh dịch thực hiện dịch trái, dịch phải logic, và dịch phải số học. Các thao tác thực hiện như sau:

Tên lệnh	Thao tác
AND	Rc ← Rav AND Rbv
BIC	Rc ← Rav AND (NOT Rbv)
BIS	Rc ← Rav OR Rbv
ORNOT	Rc ← Rav OR (NOT Rbv)
XOR	Rc ← Rav XOR Rbv
EQV	Rc ← Rav XOR (NOT Rbv)

Các dạng toán hạng được sử dụng là:

Ra. rq, Rb.rq, Rc.wq hoặc

Ra. rq, #b.ib, Rc.wq.

Phép toán NOT chính là ORNOT nếu Ra = R31 (luôn bằng 0):

ORNOT R31, Rb, Rb; 0 OR (NOT Rbv) = NOT Rbv → Rb.

Phép toán OR và BIS là có thể thay thế cho nhau.

Các lệnh CMOVxx kiểm tra điều kiện Rav. Nếu điều kiện thỏa mãn, nội của thanh ghi Rb sẽ được chuyển đến Rc. Có 8 điều kiện của CMOVxx là:

Điều kiện(xx)	CMOVE nếu thanh ghi Ra:
EQ	=0
GE	>=0
GT	>0
LBC	bit thấp =0
LBS	bit thấp =1
LE	<=0
LT	<0
NE	!0

Trong các lệnh dịch, thanh ghi Ra dịch trái hoặc dịch phải từ 0 đến 63 bit. Giá trị của 6 bit thấp Rb (Rbv<5:0>) hoặc hằng số #b chỉ số lượng bit cần dịch. Kết quả của phép dịch được cất trong Rc. Trong lệnh dịch logic, bit 0 được cho vào vị trí bit trống. Trong lệnh dịch số học, bit dấu (Rav<63>) được dịch chuyển vào các vị trí bit trống.

• *Các lệnh xử lý byte (byte manipulation instruction)*

Nhóm lệnh này có 5 dạng sử dụng. Đây là điểm đặc biệt so với các hệ lệnh khác. Các lệnh này sử dụng trong các lệnh nạp/cất để chuyển đổi các chuỗi của các byte. Các toán hạng của nó cũng giống như các lệnh dịch và logic ở trên.

1. So sánh byte (compare byte)

Loại này chứa lệnh CMP-BGE, so sánh byte lớn hơn hoặc bằng (compare byte greater or equal). Lệnh này so sánh 8 byte không dấu giữa Rav và Rbv, 8 kết quả được cất trong 8 bit thấp của Rc, 56 bit cao của Rc đặt giá trị 0. Bit 0 của Rc tương ứng với byte 0 của Ra và Rb.

Một bit kết quả được thiết lập trong Rc nếu byte tương ứng của Rav ≥ Rbv. Lệnh so sánh byte cho phép tìm kiếm chuỗi ký tự và so sánh 8 byte cùng một lúc.

2. Trích byte (extract byte)

Sau đây là 7 tùy chọn của lệnh EXTxx:

Tùy chọn EXTxx	Trích
BL	Byte low
WL	Word low
LL	Longword low
QL	Quadword low
WH	Word high
LH	Longword high
QH	Quadword high

Các lệnh EXTxL dịch thanh ghi Ra sang phải từ 0 đến 7 byte, chèn 0 vào vị trí các bit trống, sau đó trích ra 1, 2, 4 hoặc 8 byte chuyển vào thanh ghi Rc. Các lệnh EXTxH dịch Ra sang trái từ 0 đến 7 byte và chèn 0 vào vị trí trống, trích ra 2, 4 hoặc 8 byte vào Rc. Số lượng các byte cần dịch được xác định bằng ba bit thấp của Rb (Rbv <2:0>) hoặc bằng giá trị hằng số trực tiếp #b. Số các byte được trích được xác định trong mã hàm (function code), các byte còn lại được điền 0. Một lệnh EXTxL có thể nạp một byte hoặc một word (các lệnh nạp chỉ thao tác với longword và quadword), lấy dữ liệu của quadword và cất chúng vào phần thấp của thanh ghi, mà phần cao của thanh ghi được điền các giá trị 0. Một cặp lệnh EXTxL và EXTxH có thể lấy hai phần dữ liệu không được sắp xếp trong hai quadword và cất chúng vào thanh ghi kết quả, mà ở đó dữ liệu được kết hợp bằng phép OR.

Ví dụ: Giả sử cho một chuỗi các ký tự HGFEDCBA không được sắp xếp, cất trong hai vùng của một quadword: X (R11) và X + 7(R11), 3 bit nhỏ nhất của X(R11) là 101 (=5). Giá trị trong X(R11) là CBAx xxxx, trong X + 7(R11) là xxxH GFED (x là giá trị bất kỳ). Sau đây là một chương trình cất số lượng ký tự đã cho vào thanh ghi R1:

LDQ _ U R1, X(R11); R1 ← CBAx xxxx

LDQ _ U R2, X + 7(R11); R2 ← xxxH GFED

LDA R3, X(R11); R3 < 2: 0 > ← 101 (=5)

FXTQL R1, R3, R1; R1 ← 0000 0CBA

EXTQH R2, R3, R2; R2 ← HGFE D000

BIS R2, R1, R1; R1 ← HGFE DCBA

3. Chèn byte (byte insert)

Các lệnh chèn byte cũng có 7 tùy chọn như ở extract byte trên. Các lệnh INSxL và INSxH dịch các byte của Ra và chèn chúng vào trường gồm các số 0, kết quả được cất ở Rc, số lượng dịch được chỉ định bằng giá trị của ba bit thấp của Rb hoặc trực tiếp bằng hằng số #b. Mã chức năng (function code) xác định số lượng byte chèn (1, 2, 4 hoặc 8 byte). Các lệnh này có thể tạo ra dữ liệu kiểu byte, word, longword, hoặc quadword chiếm cả hai thanh ghi.

4. Mặt nạ byte (byte mask)

Các lệnh mặt nạ byte cũng có 7 tùy chọn như ở trên.

Lệnh MSKxL và MSKxH đặt các byte được chọn của Ra là 0, kết quả được cất vào Rc. Giá trị của 3 bit thấp nhất của Rb (Rbv<2:0>) hoặc bằng hằng số #b sẽ chỉ ra vị trí bắt đầu của trường gồm các byte 0, và mã lệnh chỉ ra độ rộng tối đa:1, 2, 4 hoặc 8 byte.

5. Zero byte

Gồm có 2 lệnh:

ZAP Zero byte

ZAPNOTZero byte not

Những lệnh này đặt các byte được chọn của Ra về 0, cất kết quả trong Rc. 8-bit của thanh ghi Rb (Rbv<7:0>) của Rb hoặc #b xác định số các byte được gán giá trị 0. Bit 0 của Rb tương ứng với byte 0 của Ra, bit 1 của Rb tương đương với byte 1 của Ra... Byte kết quả là 0 nếu bit tương ứng của Rb là 1 đối với lệnh ZAP và 0 đối với lệnh ZAPNOT.

6. Các lệnh nạp và cất giữ số dấu phẩy động (floating-point load and store instruction)

Có tất cả 8 lệnh làm nhiệm vụ chuyển dữ liệu giữa bộ nhớ và thanh ghi. Các lệnh được trình bày ở bảng 7.7.

Bảng 7.7: Các lệnh nạp và cất giữ số dấu phẩy động

Tên lệnh	Toán hạng	Thao tác
LDF	Fa.wf,disp.ab(Rb.ab)	Load F-floating
LDG	Fa.wg,disp.ab(Rb.ab)	Load G-floating
LDS	Fa.ws,disp.ab(Rb.ab)	Load S-floating
LDT	Fa.wt,disp.ab(Rb.ab)	Load T-floating
STF	Fa.rf,disp.ab(Rb.ab)	Store F-floating
STG	Fa.rg,disp.ab(Rb.ab)	Store G-floating
STS	Fa.rs,disp.ab(Rb.ab)	Store S-floating
STT	Fa.rt,disp.ab(Rb.ab)	Store T-floating

Trong các lệnh nạp Fa là thanh ghi đích. Trong các lệnh cất và nạp, địa chỉ bộ nhớ được tính bằng cách cộng phần mở rộng dấu của disp với Rbv. Trong lệnh cất, Fa là thanh ghi nguồn.

7. Các lệnh điều khiển dấu phẩy động (floating-point control instruction)

Có 6 lệnh rẽ nhánh dấu phẩy động. Những lệnh này kiểm tra giá trị của Fa và thay đổi có điều kiện giá trị của PC. Dạng toán hạng trong các lệnh là: Fa.rq, disp.al.

Các lệnh được trình bày ở bảng 7.8.

Bảng 7.8: Các lệnh điều khiển dấu phẩy động

Tên lệnh	Thao tác
FBEQ	Rẽ nhánh nếu bằng
FBGE	Rẽ nhánh nếu lớn hơn hoặc bằng
FBGT	Rẽ nhánh nếu lớn hơn
FBLE	Rẽ nhánh nếu nhỏ hơn hoặc bằng
FBLT	Rẽ nhánh nếu nhỏ hơn
FBNE	Rẽ nhánh nếu không bằng

Những lệnh này sử dụng địa chỉ tương đối của PC. Nếu điều kiện so sánh Fav với 0 là đúng, thì địa chỉ rẽ nhánh đích mới (new branch target address) sẽ bằng độ dịch* 2 cộng với PC.

8. Các lệnh thao tác dấu phẩy động (floating - point operate instruction)

Tổng số có tất cả 47 lệnh, chia làm 4 loại:

- Số học 16 lệnh
- Chuyển đổi 13 lệnh
- So sánh 7 lệnh
- Các lệnh phụ 11 lệnh

a. Các lệnh số học cho dấu phẩy động được trình bày ở bảng 7.9.

Bảng 7.9: Các lệnh số học dấu phẩy động

Tên lệnh	Thao tác
ADDF	Cộng F-floating
ADDG	Cộng G-floating
ADDS	Cộng S-floating
ADDT	Cộng T-floating
SUBF	Trừ F-floating
SUBG	Trừ G-floating
SUBS	Trừ S-floating
SUBT	Trừ T-floating
MULF	Nhân F-floating
MULG	Nhân G-floating
MULS	Nhân S-floating
MULT	Nhân T-floating
DIVF	Chia F-floating
DIVG	Chia G-floating
DIVS	Chia S-floating
DIVT	Chia T-floating

Có 4 lệnh thao tác số học như sau:

Cộng(Addition): Fc ← Fav + Fbv

Trừ (Subtraction): Fc ← Fav - Fbv

Nhân (Multiplication): Fc ← Fav * Fbv

Chia (Division): Fc ← Fav / Fbv

Các toán hạng cho tập lệnh số học như sau: Fa.rx, Fb.rx, Fc.wx

b. Các lệnh chuyển đổi được tổng kết ở bảng 7.10

Trong nhóm lệnh này Fb chứa các số liệu cần được chuyển đổi và kết quả được cất vào Fc.

Bảng 7.10: Các lệnh chuyển đổi số học dấu phẩy động

Tên lệnh	Toán hạng	Thao tác
CVTLQ	Fb.rq,Fc.wx	Chuyển đổi longword thành quadword
CVTQL	Fb.rq,Fc.wx	Chuyển đổi quadword thành longword
CVTDG	Fb.rx,Fc.wx	Chuyển đổi D thành G-floating
VTGD	Fb.rx,Fc.wx	Chuyển đổi G thành D-floating
CVTGF	Fb.rx,Fc.wx	Chuyển đổi G thành F-floating
CVTGQ	Fb.rx,Fc.wq	Chuyển đổi G-floating thành quadword
CVTQF	Fb.rq,Fc.wx	Chuyển đổi quadword thành F- floating
CVTQG	Fb.rq,Fc.wx	Chuyển đổi quadword thành G-floating
CVTQS	Fb.rq,Fc.wx	Chuyển đổi quadword thành S-floating
CVTQT	Fb.rq,Fc.wx	Chuyển đổi quadword thành T-floating
CVTTQ	Fb.rx,Fc.wq	Chuyển đổi T-floating thành quadword
CVTTS	Fb.rx,Fc.wq	Chuyển đổi T thành S-floating
CVTST	Fb.rx,Fc.wx	Chuyển đổi S thành floating

c. Lệnh so sánh

Có hai kiểu so sánh:

- CMPGxx (so sánh G floating)

Dạng toán hạng: Fa. rg, Fb.rg, Fc.wq, xx là các tùy chọn sau:

EQ =

LE <=

LT <

- CMPTxx (so sánh T floating)

Dạng toán hạng: Fa.rx, Fb.rx, Fc.wq.

xx: là các tùy chọn như trên: EQ, LE, LT và thêm 1 tùy chọn nữa:

UN: unordered.

Trong tất cả các lệnh so sánh, các toán hạng trong Fa và Fb được so sánh, nếu quan hệ được xác định là đúng → một giá trị khác 0 được ghi vào Fc (0,5 đối với các lệnh CMPG xx, 2.0 với các lệnh CMPTxx), nếu không đúng, giá trị 0 được ghi vào Fc.

d. Các lệnh thao tác hỗn hợp (floating - point miscellneous operate instruction)

Nhóm lệnh này được tổng kết ở bảng 7.11.

Bảng 7.11: Các lệnh thao tác hỗn hợp dấu phẩy động

Tên lệnh	Toán hạng	Thao tác
CPYS	Fa.rq,Fb.rq,Fc.wq	Sao dấu
CPYSE	Fa.rq,Fb.rq,Fc.wq	Sao dấu và số mũ
CPYSN	Fa.rq,Fb.rq,Fc.wq	Sao dấu đảo
FCMOVxx	Fa.rq,Fb.rq,Fc.wq	Vận chuyển có điều kiện số dấu phẩy động
MF-FPCR	Fa.rq,Fa.rq,Fa.wq	Vận chuyển từ FPCR
MT-FPCR	Fa.rq,Fa.rq,Fa.wq	Vận chuyển đến FPCR

Trong đó, FPCR: thanh ghi điều khiển dấu phẩy động.

Trong các lệnh FCMOVxx, xx là một trong các tùy chọn sau:

EQ = 0

GE >= 0

GT > 0

LE <= 0

LT < 0

NE ! 0

Với các lệnh CPYS và CPYSN, bit dấu của Fa (được bù trong trường hợp đối với CPYSN) được liên kết với các bit luỹ thừa và bit thập phân (fraction bit) của Fb và kết quả cất tại Fc. Đối với CPYSE bit dấu và bit lũy thừa của Fa được liên kết với bit dấu thập phân của Fb, kết quả được cất trong Fc. Những lệnh này có thể được sử dụng để tạo ra những thao tác đặc biệt không có sẵn trong các tập lệnh thường như:

Chuyển nội dung thanh ghi (register move) CPYS Fx, Fx, Fy

Đảo số dấu phẩy động (floating- point negation) CPYSN Fx, Fx, Fy

Trong các lệnh khác CMOV xx thanh ghi Fa được kiểm tra và so sánh với 0. Nếu mối quan hệ được xác định là đúng, Fbv được chuyển vào Fc, ngược lại, sẽ không có thao tác nào được thực hiện.

Trong các lệnh MF_FPCR và MT_FPCR nội dung của FPCR được chuyển đến Fa, và Fav được chuyển đến FPCR.

e. Các lệnh hỗn hợp (misellaneous instructions)

Có 7 lệnh được liệt kê ở bảng 7.12.

Lệnh CALL _ PAL gây ra bẫy (trap) tới mã PAL (PALcode).

Trong các lệnh FETCH và FETCH_M có một khối dữ liệu 512 byte được định bởi Rbv.

Bảng 7.12: Các lệnh hỗn hợp

Tên lệnh	Toán hạng	Thao tác
CALL–PAL	fnc.ir	Gọi chương trình PAL
FETCH	0(Rb.ab)	Dữ liệu tiền đọc
FETCH–M	0(Rb.ab)	Dữ liệu tiền đọc, thay đổi mục đích
MB	—	Chắn bộ nhớ
WMB	—	Chắn ghi bộ nhớ
RPCC	Ra.wq	Đếm chu kì xử lý đọc
TRAPB	—	Chắn bẫy

Quá trình thực hiện có thể tùy chọn để chuyển tất cả hoặc một phần của khối dữ liệu đến vùng nhớ có tốc độ truy nhập nhanh trong toàn bộ cấu trúc nhớ.

Lệnh MB được dùng trong hệ thống đa xử lý. Nó tạo thuận lợi để đồng bộ giữa các vòng lặp phụ thuộc.

Trong lệnh RPCC nội dung của thanh đếm chu kì xử lý (process cycle counter - PCC) được ghi vào Ra. PCC sử dụng cho định các quãng thời gian (timing intervals) trong từng bộ xử lý của một hệ thống đa bộ vi xử lý.

Lệnh TRAPB cho phép phần mềm đảm bảo rằng trong thực hiện của đường ống, tất cả các lệnh số học trước đó sẽ được hoàn thành mà không phải chịu các bẫy số học nào trước khi các lệnh sau lệnh TRAPB được đưa ra.

- *Mã thư viện kiến trúc có thẩm quyền PAL (Privileged Achitecture Liblary - PAL code)*

Mã PAL cung cấp một cơ chế thực hiện những chức năng dưới đây mà không phải dùng đến vi lệnh:

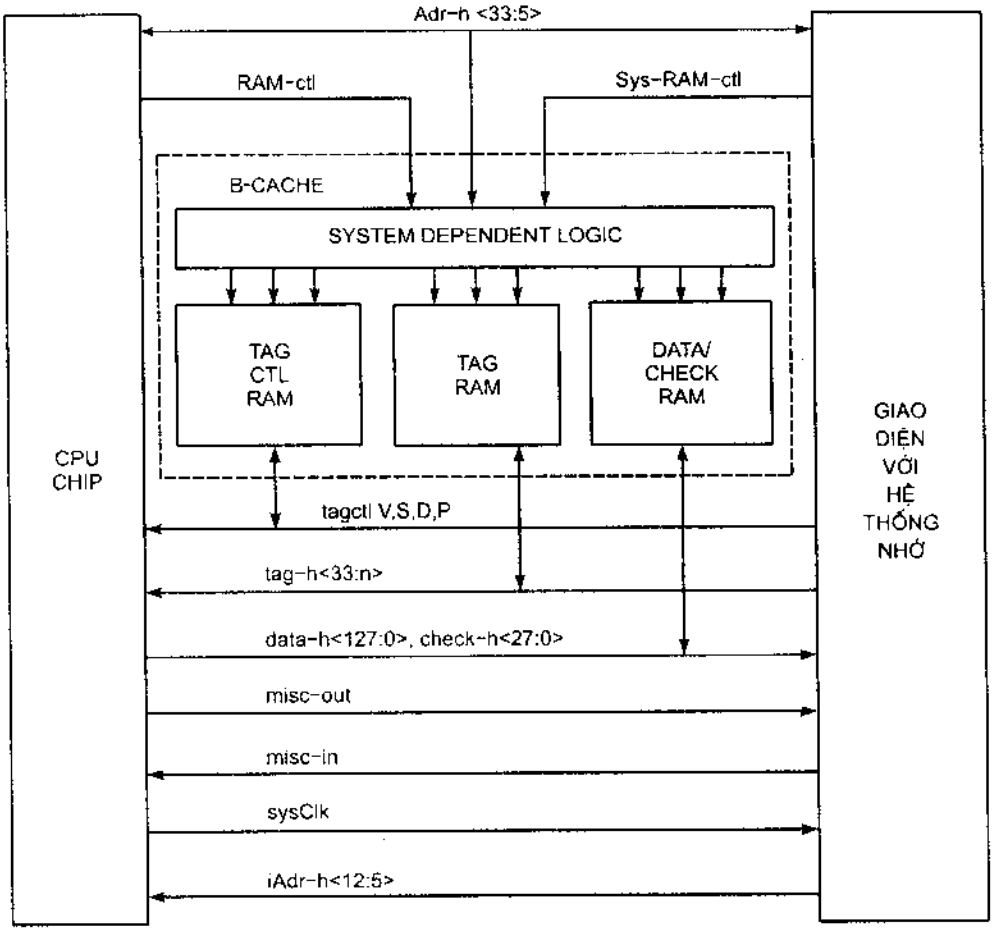
1. Những lệnh yêu cầu sắp xếp cần tuần tự như một thao tác nguyên tử (atomic operation) (lệnh atomic là lệnh không thể ngắt, tức là trong thời gian thi hành lệnh atomic, không một bộ xử lý nào có thể truy nhập vào tài nguyên của nó).
2. Những lệnh yêu cầu truy nhập bộ nhớ kiểu VAX
3. Những lệnh đặc quyền (privilged instructions)
4. Điều khiển quản lý bộ nhớ
5. Chuyển đổi văn bản (context swapping)
6. Xử lý ngắt và loại trừ (interrupt and exception dispatching)
7. Khởi tạo nguồn và khởi động hệ điều hành (power-up initialization and booting)
8. Các chức năng của Console (vào ra chuẩn)
9. Mô phỏng lệnh không có hỗ trợ của phần cứng

Các chức năng của PAL được thi hành trong kiến trúc Alpha bằng mã máy chuẩn nằm trong bộ nhớ chính.

Mã PAL sử dụng các tập lệnh Alpha trong hầu hết các thao tác của nó. Có 5 mã lệnh để thực hiện các chức năng của mã PAL là: PALRES 0, 1, 2, 3, 4. Những lệnh này sẽ sinh ra bẫy lệnh sai nếu những lệnh này thực hiện ngoài môi trường của mã PAL. Mã PAL được đặt trong bộ nhớ vật lý (bộ nhớ chính).

(8) Giao diện bên ngoài của vi xử lý Alpha

Giao diện với hệ thống bên ngoài (External system Interface) dùng để kết nối chip vi xử lý Alpha đầu tiên 21064. Hình 7.66 là ví dụ kết nối với bên ngoài của chip 21064.



Hình 7.66: Giao diện của CPU 21064 với bên ngoài

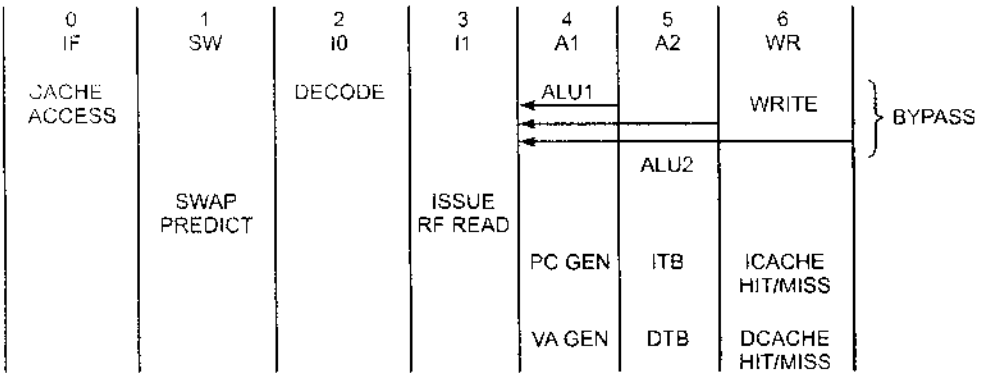
Giao diện bên ngoài của 21164 sau này cũng tương tự như của 21064. Trong sơ đồ khối chức năng của 21164 (hình 7.60) chúng ta nhận thấy có khối tùy chọn cache sao lưu (backup cache option) trong khung nét đứt. Đó là khối giao diện với cache bên ngoài. Nó được thiết kế để hỗ trợ trực tiếp bộ nhớ cache thứ cấp nằm ngoài chip (còn gọi là backup cache, hoặc B_cache). B_cache có khả năng lưu trữ từ 128 kB đến 8 MB và được thiết kế từ những SRAM thường. Giao diện được thiết kế để cho phép tất cả các quyết định của cache được điều khiển bởi khối logic bên ngoài đến CPU. Có 3 bit điều khiển kết hợp với từng dòng của B_cache là: valid (V), shared(S), dirty(D). Bộ vi xử lý đọc B_cache khi bit V thiết lập "1", và ghi vào B_cache khi bit V là 1 và bit S là 0. Trong quá trình ghi, bit D được thiết lập giá trị 1. Trong các trường hợp khác, chip trì hoãn các thao tác bên ngoài để trao đổi thông tin. Trong hình 7.59, B_cache được nối với chip CPU và giao diện bộ nhớ (memory system interface). B_cache được truy nhập nhờ địa chỉ ở trên các đường adr_h và các kết quả ở trong ctl, tag, data. Trong chế độ 128-bit, Sự truy nhập tới B_cache cần 2 chu kì dữ liệu bên ngoài để chuyển dòng cache 32 byte (256 = 2 × 128 bit) qua bus 16 byte. Trong chế độ 64-bit phải cần 4 chu kì. Khả năng đọc tối đa của B_catche là 1,2 GB/s và ghi là 711 Mbit/s.

(9) Kiến trúc đường ống của 21064:

Đường ống nguyên (IU pipeline) của 21064 có 7 giai đoạn (hình 7.67):

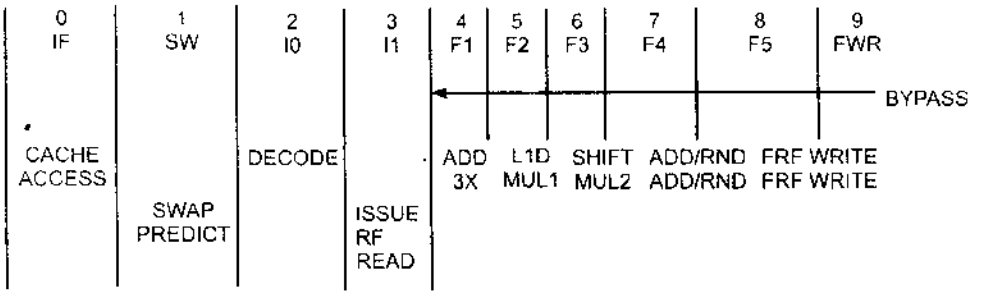
0_IF (đọc lệnh), 1-SW (đổi chỗ), 2-I0, 3-I1, 4-A1, 5-A2, 6-WR (ghi).

Trong đó, bốn giai đoạn đầu liên quan đến đọc lệnh, giải mã lệnh, kiểm tra bảng điểm (scoreboard) của các toán hạng về tính độc lập của dữ liệu. Hầu hết các thao tác của ALU thực hiện trong 4 chu kỳ nhịp (A1). Các truy nhập cache hoàn thành trong chu kỳ 6 (WR), như vậy trễ do truy nhập cache trong 3 chu kỳ. Chuỗi lệnh được tự động đọc trước trong chu kỳ 0 và chu kỳ 1 với kết quả cuối cùng của sự trúng ICACHE không ở trong chu kỳ 5.



Hình 7.67: Đường ống nguyên (IU pipeline) của chip Alpha 21064

Đường ống dấu phẩy động có 10 giai đoạn (hình 7.68). Nó có cấu trúc tương tự như với đường ống nguyên ở 4 giai đoạn đầu.



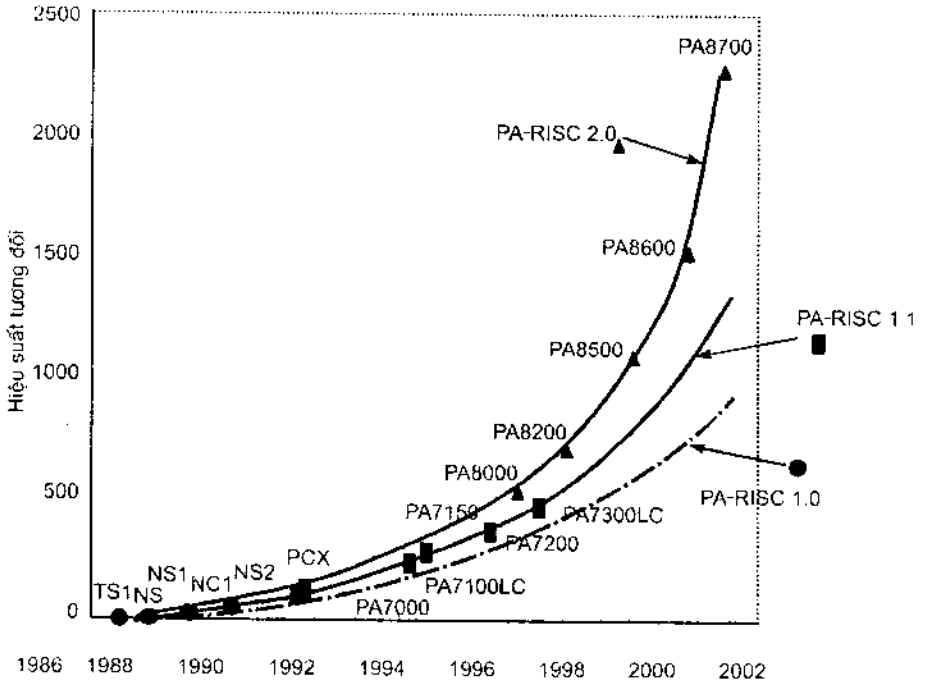
Hình 7.68: Đường ống dấu phẩy động

Tất cả các thao tác 32-bit và 64-bit có định thời giống nhau trừ phép chia (divide). Phép chia được xử lý không theo đường ống mà bằng đơn vị chia dành riêng với mỗi bit một chu kỳ. Trong chu kỳ 4 (F1), dữ liệu của tập thanh ghi được định khuôn dạng dấu phẩy động, số mũ, và dấu. Trong giai đoạn này xảy ra tính cộng số mũ khác nhau và 3 lần (3x) số bị nhân được tạo ra cho phép nhân, đồng thời có sự đoán trước giá trị bit bằng 0 hoặc 1 của số bị nhân để thực hiện phép bình thường hóa. Trong chu kỳ 5 (F2) và chu kỳ 6 (F3), phép cộng/trừ, dịch để hiệu chỉnh hoặc bình thường hóa được thực hiện. Trong các chu kỳ 7 (F4) và 8 (F5) kết quả của phép cộng và làm tròn

được thực hiện đồng thời và kết quả cuối cùng được chọn và điều khiển trở lại tập thanh ghi trong chu kỳ 9 (FWR). Với sự cho phép bỏ qua (BYPASS) sự ghi dữ liệu vào thanh ghi, trễ dấu phẩy động là 6 chu kỳ.

7.4.6 Họ vi xử lý HP Precision

a) Giới thiệu công nghệ HP Precision



Hình 7.69: Sự phát triển của họ kiến trúc PA-RISC

Hewlett-Packard (HP) là một trong những nhà sản xuất hàng đầu các hệ thống kiến trúc RISC. Dòng sản phẩm có tên gọi "Kiến trúc chính xác" PA (Precision Architecture), hay PA-RISC là các hệ thống xử lý có tập lệnh tối thiểu (RISC). PA-RISC và hệ điều hành Unix HP-UX trong các máy tính trạm và các máy tính dịch vụ thuộc các loại N-class, V-class, L-class, và A-class có hiệu suất sử dụng rất cao cho nhiều loại ứng dụng. Đại diện cho dòng sản phẩm này là họ PA71XX, và dòng họ từ PA8000 đến PA8700. PA-RISC được HP giới thiệu vào năm 1985 với nhịp đồng hồ 8 MHz. Ngày nay có tới 16 loại

vi xử lý PA-RISC khác nhau với PA8600/550 MHz và PA8700/800 MHz trong năm 2001. Hiện nay tiếp tục với PA-RISC với 1 GHz. Sự phát triển của công nghệ PA-RISC được mô tả trong hình 7.69. Có 3 loại kiến trúc PA-RISC, và sự phát triển của 3 loại này thể hiện bằng 3 đường cong, đó là: loại 1.0 (PA-RISC 1.0), loại 1.1 (PA-RISC 1.1), loại 2.0 (PA-RISC 2.0). Các bộ xử lý PA-RISC có thể được dùng trong các hệ thống đơn xử lý hay nhiều bộ xử lý.

Loại kiến trúc PA-RISC 1.1 bắt đầu bằng PA7000 (1991) có một số thay đổi quan trọng so với loại kiến trúc 1.0 là:

- Kích thước của trang nhớ tăng từ 2 kB (loại 1.0) lên 4 kB.
- Có thêm 16 thanh ghi dấu phẩy động so với loại 1.0.
- Tăng tốc độ và hiệu quả xử lý bẫy (traps) so với loại 1.0.

Trong họ 7xx0 của loại kiến trúc PA-RISC 1.1 tần số làm việc tiếp tục được gia tăng. Kiến trúc siêu hướng (thực hiện đồng thời hơn 1 lệnh) là một trong những kỹ thuật được sử dụng để tăng hiệu suất của các chip xử lý. Bắt đầu trong năm 1992, PA7100 ra đời với kiến trúc siêu hướng 2 cửa ra cho phép đồng thời thực hiện 2 lệnh. PA7100LC (1993) và PA7200 (1994) đều có thêm một đơn vị IU thứ 2 cho phép thực hiện linh hoạt đồng thời 2 lệnh xử lý các số nguyên. Do đó đã cho phép tăng 20% hiệu suất so với PA7100.

Trong năm 1996, HP đã giới thiệu PA8000, khởi đầu của loại kiến trúc PA-RISC 2.0. Loại kiến trúc này so với loại 1.1 có những sự thay đổi quan trọng sau đây:

- Có mở rộng địa chỉ và dữ liệu lên 64-bit
- Có các mở rộng cho đa phương tiện
- Hỗ trợ vào/ra liên kết chặt chẽ (coherent I/O)
- Có khối nhân-tích luỹ FMAC (Multiply-accumulate)
- Thực hiện thứ tự hóa bộ nhớ yếu (weak memory ordering)

Kiến trúc Pa-RISC 2.0 đảm bảo hỗ trợ xử lý dữ liệu 64-bit đầy đủ bằng các thanh ghi 64-bit và bus dữ liệu 64-bit. Các thanh ghi 64-bit

cũng đảm bảo xử lý dữ liệu 32-bit. Nó cũng đảm bảo địa chỉ ảo 64-bit và địa chỉ vật lý lớn hơn 32-bit cho phép người sử dụng làm việc với các tập hợp lớn dữ liệu và lệnh. PA8000 có 40 bit để địa chỉ vật lý, nghĩa là nó có thể địa chỉ bộ nhớ vật lý dung lượng 1 TByte.

PA8200 được giới thiệu năm 1997 đã có sự cải tiến so với PA8000 nhờ:

- Tăng tần số làm việc tới 240 MHz;
- Gấp 4 lần kích thước của bảng lịch sử rẽ nhánh BHT (Branch History Table) từ 256 điểm vào đến 1024 điểm vào;
- Tăng hiệu suất của thuật toán đoán trước rẽ nhánh;
- Tăng số lượng điểm vào của bộ đệm chuyển đổi địa chỉ TLB (Translation Look-aside Buffer) từ 96 điểm vào lên 120 điểm vào;
- Gấp đôi kích thước của cache bên ngoài từ 1 MB đến 2 MB.

PA8500 được giới thiệu năm 1998 đạt được mức hiệu suất đáng kể nhờ:

- Công nghệ CMOS 0,25 micron;
- Tăng tần số làm việc lên 440 MHz;
- Tích hợp các cache bên trong chip: 1 MB Dcache, 0,5 MB Icache với 4-đường liên hợp tập;
- Gấp đôi số các điểm vào của BHT từ 1024 (PA8200) đến 2048.
- Tăng số điểm vào của TLB từ 120 điểm (PA8200) lên 160 điểm.
- Tăng độ rộng của bus hệ thống.

Đến năm 2000, PA8600 được giới thiệu năm đã có nhịp đồng hồ hệ thống 550 MHz.

Năm 2001 PA8700 có một số đặc điểm chính sau đây:

- Tần số nhịp lớn hơn 800 MHz, gần gấp đôi so với PA8600 (550 MHz);
- Tăng kích thước Dcache từ 1 MB (PA8500) lên 1,5 MB;

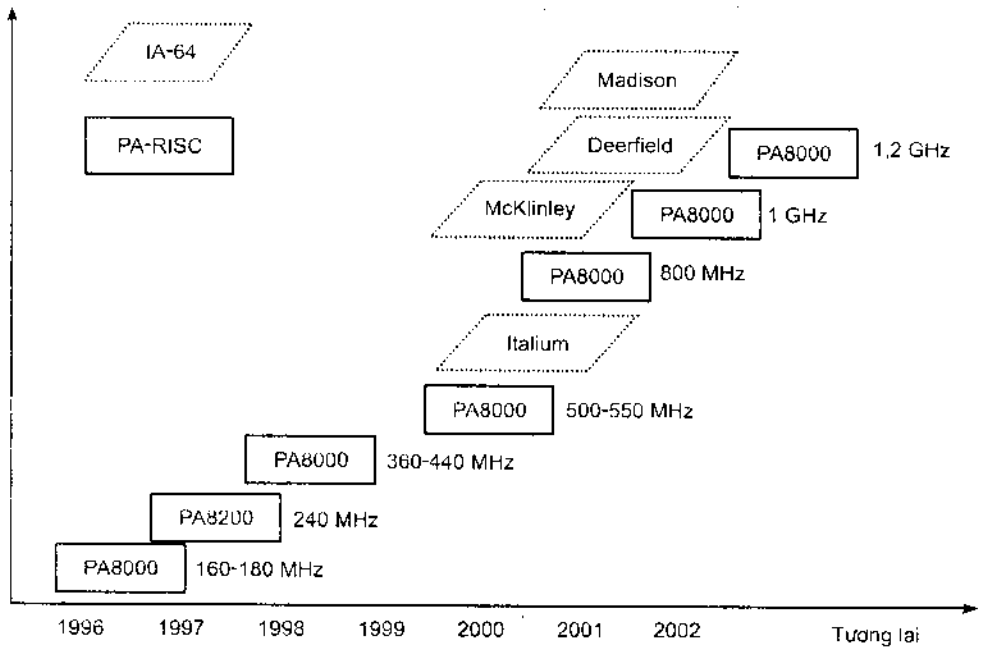
- Tăng kích thước Icache từ 0,5 MB lên 0,75 MB;
- Có khả năng tiền đọc dữ liệu;
- Hỗ trợ 44-bit địa chỉ vật lý, đánh địa chỉ tới 16 Tbyte;
- Công nghệ Silicon-on-insulator (SOI) CMOS 0,18micron, với 7 lớp kim loại so với PA8600 như bảng 7.13.
- Tiêu thụ nguồn chỉ bằng 75% so với PA8600.

Bảng 7.13: So sánh 7 lớp kim loại PA8600 và PA8700

Lớp	PA8600	PA8700
Metal 1	0,64 micron	0,49 micron
Metal 2	0,93 micron	0,63 micron
Metal 3	0,93 micron	0,63 micron
Metal 4	1,60 micron	0,63 micron
Metal 5	2,56 micron	0,63 micron
Metal 6	Không có	1,26 micron
Metal 7	Không có	1,26 micron

PA8700 trong những hệ thống máy tính dịch vụ hiện nay đạt được hiệu suất rất cao so với với những hệ thống PA-RISC trước đó. HP và Intel đã bắt đầu hợp tác từ đầu năm 1994 với kiến trúc IA-64. Kiến trúc IA-64 giới thiệu khái niệm EPIC (Explicity Parallel Instruction Computing) - một kiến trúc xử lý song song các lệnh trong từng chu kỳ nhịp hệ thống. Kiến trúc IA-64 là một chuẩn công nghiệp 64-bit HP coi IA-64 là một xu hướng chuyển đổi cách mạng từ kiến trúc PA-RISC, trong khi vẫn duy trì sự tồn tại và phát triển của PA-RISC trong những năm 2000. Con đường phát triển của công nghệ vi xử lý của HP từ PA-RISC 2.0 hiện tại và tương lai được mô tả trong hình 7.70. Họ vi xử lý IA-64 (sản phẩm chung của HP và Intel) bắt đầu với Itatium vào cuối năm 2000 và tiếp tục với McKinley™, Deerfield™, và Madison™. Nó cùng song song phát triển với công nghệ PA-RISC của riêng HP. Một số lớn các lệnh của kiến trúc PA-RISC có trong vi xử lý kiến trúc IA-64. Điều này cho phép chuyển trực tiếp 1:1 các lệnh

máy của PA-RISC sang IA-64, nghĩa là chương trình nguồn được dịch ở hệ thống PA-RISC có thể chạy được trên hệ thống IA-64. HP sẽ là nhà cung cấp các phần mềm ứng dụng chạy trên các hệ điều hành khác nhau: HP-UX, Linux, Windows NT64, và tích hợp các ứng dụng từ PA-RISC vào IA-64. Còn các ứng dụng chính cho IA-64 sẽ có những ưu điểm của công nghệ EPIC. HP sẽ dẫn đầu trong các nhà cung cấp các giải pháp cho IA-64.



Hình 7.70: Sự phát triển của vi xử lý của HP

b) Đặc điểm kiến trúc của PA-RISC

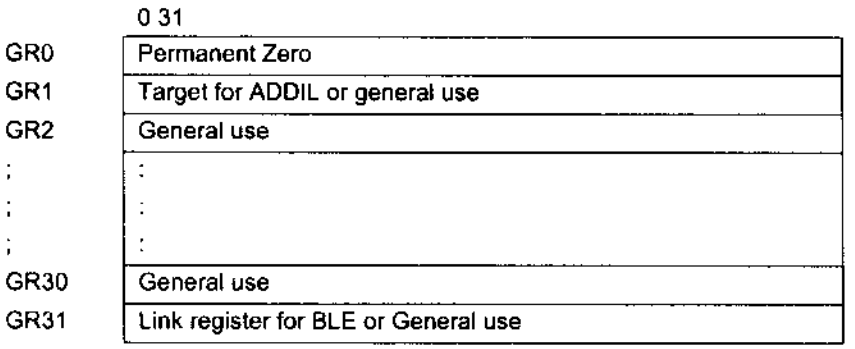
(1) Các thanh ghi bên trong của PA-RISC

Chip vi xử lý HP-RISC có các thanh ghi bên trong như sau:

• *Các thanh ghi chung*

Có tất cả 32 thanh ghi chung độ dài 32-bit: GR0-GR31 (hình 7.71). Cũng giống kiến trúc của PowerPC, bit giá trị cao nhất (MSB) là bit 0 và bit có giá trị thấp nhất (LSB) là bit 31. Thanh ghi GR0 luôn gắn liền với giá trị zero, cũng như hầu hết các hệ thống RISC khác.

Thanh ghi GR1 là đích của phép dịch trái và cộng ngay lập tức (ADDIL). Thanh ghi GR31 là thanh ghi liên kết (link register) độ dịch địa chỉ tương đối cơ sở để gọi tới thủ tục rẽ nhánh và liên kết bên ngoài BLE (Branch And Link External). Các thanh ghi GR1-GR31 có thể sử dụng như những thanh ghi chung.



Hình 7.71: Các thanh ghi chung của PA-RISC

• *Các thanh ghi nền (shadow registers)*

Có 7 thanh ghi 32-bit loại này, và chúng có thể được khởi tạo tùy chọn từ những thanh ghi chung, ví dụ, nếu chọn 7 thanh ghi chung như GR1, GR8, GR9, GR16, GR17, GR24, và GR25 để tạo 7 thanh ghi nền của chúng, thì khi xảy ra ngắt, nội dung của những thanh ghi chung này được cất giữ vào những thanh ghi nền của chúng. Nội dung của những thanh ghi chung này được phục hồi từ những thanh ghi nền của chúng khi thực hiện nhánh trở về từ xử lý ngắt.

• *Các thanh ghi không gian (space registers) SR0-SR7*

Các thanh ghi này chứa giá trị địa chỉ ảo trực tiếp cho các lệnh. Các lệnh rẽ nhánh có thể sử dụng bất kỳ thanh ghi nào trong nhóm này để tính địa chỉ. Thanh ghi SR0 là địa chỉ lệnh dùng cho lệnh gọi thủ tục rẽ nhánh và liên kết BLE. Các toán hạng có thể được định nghĩa trực tiếp bằng các thanh ghi từ SR1 đến SR3 và gián tiếp bằng các thanh ghi SR4 đến SR7, thông qua các thanh ghi chung GRi. Nội dung và ý nghĩa của các thanh ghi này cho trong hình 7.72.

	0 ... 31
SR0	Link code space ID
SR1	General use
SR2	General use
SR3	General use
SR4	Tracks IA space
SR5	Process private data
SR6	Shared data
SR7	Operating system's public code, literals and data

Hình 7.72: Các thanh ghi không gian của PA-RISC

• *Các thanh ghi điều khiển (control register) CR0, CR8-CR31*

Có tất cả 25 thanh ghi điều khiển độ dài 32-bit. Chúng chứa đựng thông tin về trạng thái của hệ thống. Các thanh ghi từ CR1 đến CR7 dành cho dự phòng trong tương lai.

• *Các thanh ghi dấu phẩy động*

Có tới 32 thanh ghi dấu phẩy động có độ dài 64-bit. Chúng có thể được sử dụng với 32-bit cục bộ. Các thanh ghi từ 0 đến 3 là một thanh ghi chứa trạng thái 32-bit và 7 thanh ghi loại trừ 32-bit. Các thanh ghi từ 4 đến 31 là những thanh ghi dữ liệu. Trong lệnh có tiền tố "L" để xác định cho nửa 32-bit bên trái và tiền tố "R" để xác định cho nửa bên phải của thanh ghi 64-bit. Ví dụ: thanh ghi 12 64-bit có nửa bên trái (bit 0-31) được viết là 12 hay 12L, nửa bên phải còn lại (bit 32-63) được viết là 12R.

• *Từ trạng thái của bộ xử lý PSW*

PSW có độ dài 32-bit (hình 7.73) và có nội dung như sau:

- Bit 6 (S): Đoạn thời gian an toàn. Khi S = 1 thì quãng thời gian chỉ có thể đọc được nhờ lệnh thực hiện ở mức có thẩm quyền. Khi S = 0 thì quãng thời gian có thể được đọc ở bất kỳ mức thẩm quyền nào.

- Bit 7 (T): Cho phép đặt bẫy rẽ nhánh. Khi T = 1, bất kỳ rẽ nhánh nào cũng được kết thúc bằng một bẫy rẽ nhánh.

- Bit 8 (H): Cho phép bẫy chuyển mức thẩm quyền cao. Khi H = 1, thì một bẫy chuyển mức thẩm quyền cao xảy ra và khi đó lệnh tiếp theo ở mức thẩm quyền cao.

- Bit 9 (L): Cho phép bẫy chuyển mức thẩm quyền thấp. Khi L = 1 thì một bẫy chuyển mức thẩm quyền thấp sẽ xảy ra và lệnh tiếp theo sẽ ở mức thẩm quyền thấp.

- Bit 10 (N): Làm bằng 0. Khi N = 1 thì lệnh hiện hành sẽ bị đưa về 0 (nullified).

- Bit 11 (X): Cấm ngắt bộ nhớ dữ liệu. Khi X = 1 thì các bẫy ngắt bộ nhớ dữ liệu bị cấm. X có thể được thiết lập khi có lệnh trở về từ ngắt, và được xóa bằng lệnh khác.

- Bit 12 (B): Rẽ nhánh được lấy ra. Giá trị B được thiết lập bằng 1 khi có rẽ nhánh được lấy ra, và B = 1 nhờ các lệnh khác.

- Bit 13 (C): Cho phép chuyển đổi địa chỉ của lệnh. Khi C = 1 thì các địa chỉ của lệnh được chuyển đổi và các quyền truy nhập được kiểm tra.

- Bit 14 (V): Hiệu chỉnh bước chia. Lệnh chia số nguyên ghi trạng thái tức thời vào bit V để đảm bảo có một số chia không phục hồi.

- Bit 15 (M): Mặt nạ kiểm tra máy mức thẩm quyền cao. Khi M = 1 thì các kiểm tra máy mức thẩm quyền cao (HPMCs) bị cấm. Bình thường M = 0. M được thiết lập bằng 1 sau khi có một HPMC và nó được xóa sau tất cả các lệnh.

- Bit 16 đến 23 (C/B): Nhớ/vay. Các bit này được thiết lập trong các lệnh cộng số học. Chúng được xóa khi có vay mượn trong các lệnh trừ.

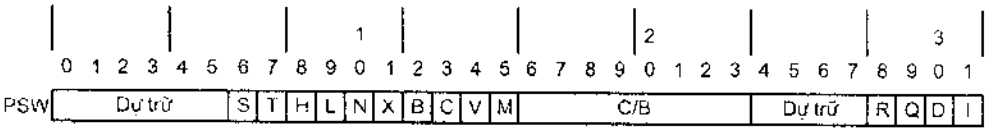
- Bit 27 (R): Cho phép bộ đếm phục hồi. Khi R = 1 thì một bẫy cho phép bộ đếm phục hồi xuất hiện nếu bit 0 của bộ đếm phục hồi bằng 1. Bit này cũng cho phép xác định bộ đếm phục hồi.

- Bit 28 (Q): Cho phép thu thập trạng thái ngắt. Khi Q = 1 thì trạng thái ngắt được thu thập.

- Bit 29 (P): Cho phép kiểm tra các xác định bảo vệ thực. Khi P = 1 và C = 1, thì các tham chiếu của lệnh kiểm tra các xác định bảo vệ PID thực (protection identifiers). Khi P = 1 và D = 1 thì các tham chiếu dữ liệu kiểm tra các PID thực. Khi P = 1, thì các lệnh thử kiểm tra các PID thực.

- Bit 30 (D): Cho phép chuyển đổi địa chỉ dữ liệu. Khi D = 1 thì các địa chỉ dữ liệu được chuyển đổi và các quyền truy nhập được kiểm tra.

- Bit 31 (I): Bỏ mặt nạ cho các ngắt bên ngoài, ngắt do lỗi nguồn. Khi I = 1 những ngắt này được bỏ mặt nạ ngắt và có thể gây ra một ngắt. Khi I = 0, những ngắt này bị chặn lại.



Hình 7.73: Từ trạng thái (PSW) của bộ xử lý PA-RISC

(2) Các kiểu dữ liệu

Kiến trúc PA-RISC có các kiểu dữ liệu (hình 7.74) sau đây:

• Số nguyên không dấu và có dấu:

- Byte: 8-bit

- Nửa từ (Halfword): 16-bit

- Từ (Word): 32-bit

• Số nguyên không dấu:

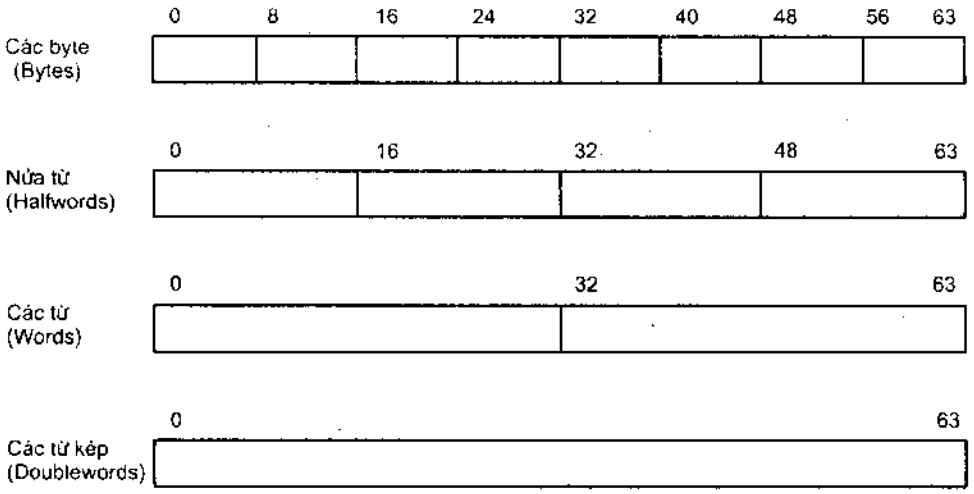
- Từ kép (Doubleword): 64-bit

- Dấu phẩy động

• Số theo chuẩn IEEE 754-1985:

- Chính xác đơn (single-precision): 32-bit

- Chính xác kép (Double-precision): 64-bit



Hình 7.74: Các kiểu dữ liệu của HP PA-RISC

- Chính xác mở rộng gấp 2 (Quadrupe-word): 128-bit (gồm bit dấu (s), 15-bit số mũ (e), 112-bit định trị (m)).

• Số hệ mười đóng gói (Packet decimal):

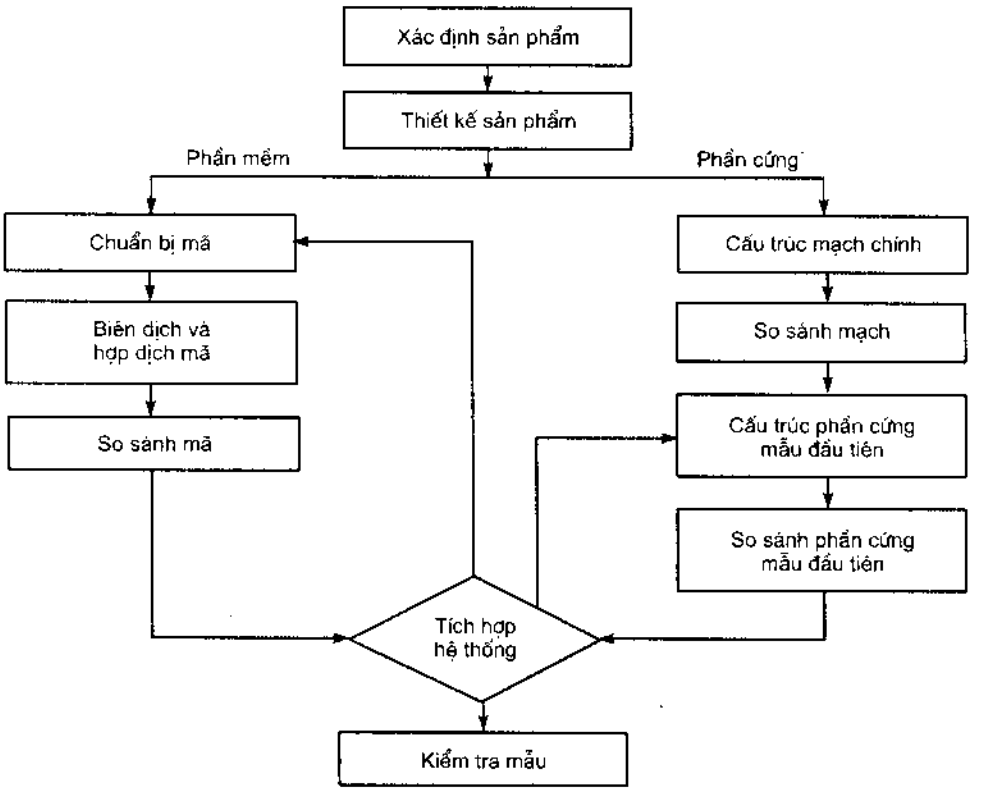
Các số này gồm có 7, 15, 23 hoặc 31 chữ số (digit) BCD, mỗi chữ số 4-bit sắp xếp trong ranh giới một từ và mỗi chữ số có giá trị từ 0 đến 9.

Sau cùng là 4-bit dấu. Dấu tiêu chuẩn đối với số dương là X'C (X' là số hệ 16), nhưng bất kỳ giá trị nào ngoại trừ X'DF sẽ được diễn giải như là một số dương. Dấu âm là X'D:

(3) Các khuôn dạng lệnh và tập lệnh

Các khuôn dạng lệnh của PA-RISC cho trong hình 7.75. Có 3 trường thanh ghi dành cho các lệnh của đơn vị ALU (số học và logic). Một trường thanh ghi có thể trao đổi với trường toán hạng tức thời (ngay lập tức). Trong các khuôn dạng lệnh nạp và cất giữ, một trường thanh ghi dành cho thanh ghi cơ sở, và một trường dành cho thanh ghi chỉ số (dùng cho độ dịch). Đối với các lệnh nhảy, có các trường cho thanh ghi cơ sở và cho độ dịch.

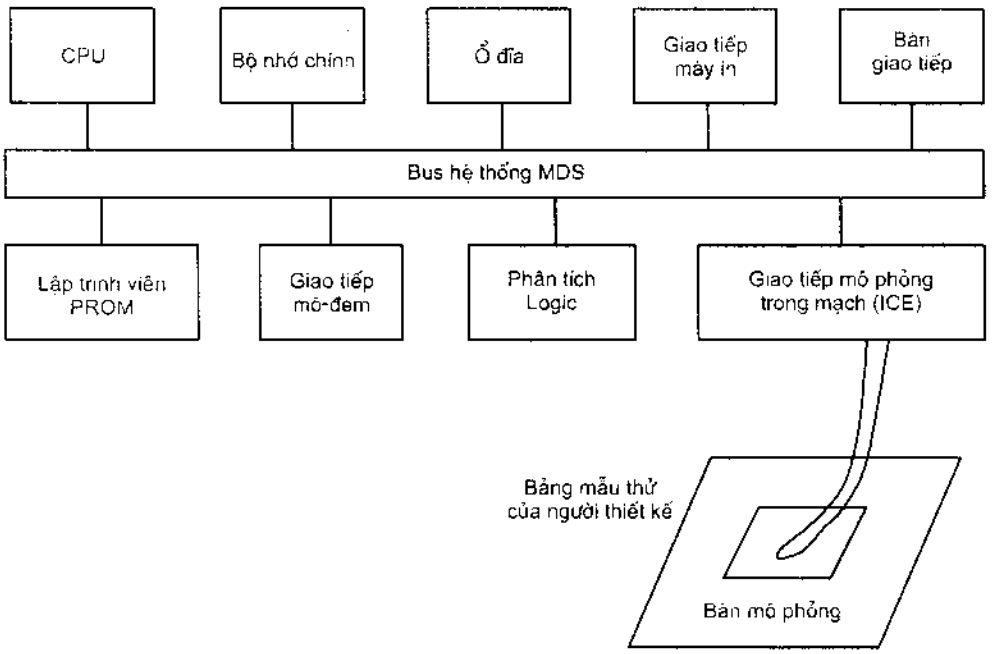
đoạn so sánh. Giai đoạn phát triển hệ thống bắt đầu bằng xây dựng các mạch chính của sản phẩm. Giai đoạn xây dựng mẫu đầu tiên được thực hiện bằng cách từng bước xếp đặt các đế cắm cho các chip vi mạch phù hợp với vị trí liên kết của chúng mà bước thiết kế sản phẩm đã chuẩn bị. Phần cứng và phần mềm của thiết kế mẫu được chuyển đến giai đoạn tích hợp hệ thống. Giai đoạn này thực hiện sự hiệu chỉnh cho cả phần cứng và phần mềm. Khi có phát sinh lỗi thì quay trở lại hiệu chỉnh (hiệu chỉnh phần cứng và phần mềm). Giai đoạn kiểm tra mẫu chỉ được thực hiện khi tích hợp hệ thống hoàn thành. Sự kiểm tra mẫu thường thực hiện cho từng module. Toàn bộ chu trình phát triển cần được thực hiện nhờ công cụ gọi là hệ phát triển máy vi tính MDS (Microcomputer Devolopement System).



Hình 7.83: Chu trình phát triển máy vi tính

7.5.2 Hệ phát triển máy vi tính (MDS)

MDS cung cấp cho người phát triển tất cả các hệ thống nhỏ cần thiết trong từng giai đoạn của chu trình phát triển. MDS là một máy vi tính có phần cứng và phần mềm thân thiện, hiệu quả đủ cho người sử dụng phát triển các hệ thống dựa trên cơ sở vi xử lý. Sơ đồ khối của một hệ thống MDS đặc trưng cho trong hình 7.84.



Hình 7.84: Hệ phát triển máy vi tính (MDS)

Vì MDS là một máy vi tính nên nó bao gồm:

- CPU, bộ nhớ chính (RAM, ROM), bộ nhớ thứ cấp (bộ nhớ ngoài trên đĩa từ), các giao tiếp vào/ra (máy in, mô-đem, màn hình, bàn phím).

- Một số loại MDS có thể có vài CPU để thực hiện kiểm tra và các chương trình giao tiếp đồng thời. Các hệ thống con trong MDS là những khối chuyên dụng để phát triển phần cứng như:

- Thiết bị lập trình PROM, giao tiếp phân tích logic, giao tiếp mô phỏng trong mạch ICEI (In-circuit emulator interface).

Hầu hết các MDS có những đặc điểm sau đây:

- Có bảng hiển thị mặt tiền để xem nội dung của các thanh ghi của vi xử lý và của các ngăn nhớ

- Có khả năng thay đổi nội dung của các thanh ghi và của các ngăn nhớ

- Điều khiển xóa (reset) khởi động CPU vào trạng thái xác định trước

- Có điều khiển chạy từng bước (single-step control)

- Điều khiển chạy chương trình cho phép chương trình bắt đầu thực hiện ở một lệnh trong ngăn nhớ xác định

- Chương trình con nạp khởi động (bootstrap loader) thực hiện đưa các chương trình khởi tạo ban đầu vào trong bộ nhớ

- Các chương trình tiện dụng để nạp các chương trình của người sử dụng vào trong bộ nhớ từ bảng phím

- Một chương trình hợp dịch (assembler)

- Một màn hình hiển thị của hệ thống phục vụ thao tác hệ điều hành của MDS

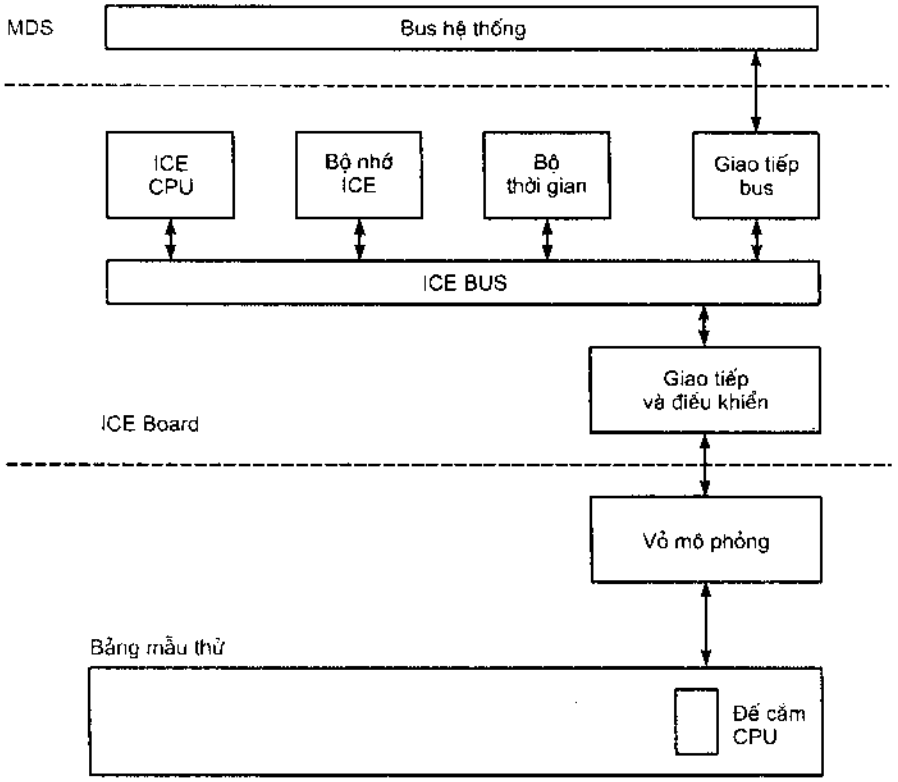
- Có trang bị khả năng thiết lập ngắt hoặc chạy theo tuần tự (Trace)

- Các đế cắm để nối với các thiết bị ngoài

- Một hệ soạn thảo để có thể sửa chữa các chương trình của người sử dụng

Bên cạnh các giao tiếp đã nói ở trên, có giao tiếp với bộ phân tích logic cho phép người thiết kế nhìn thấy được các mức logic xuất hiện trong thời gian thực của phần cứng. Thiết bị lập trình cho EPROM là một thiết bị dùng để nạp các chương trình vào trong các chip PROM. Khi nó được sử dụng với MDS, nó có thể được điều khiển qua chương trình. Bảng mặt của thiết bị lập trình PROM có các đế cắm cho các chip PROM để lập trình và kiểm tra.

ICE là một hệ vi xử lý hoàn chỉnh có CPU riêng, bộ nhớ, mạch định thời (logic tạo nhịp riêng), và giao tiếp vào/ra. ICE giao tiếp trực tiếp với mẫu thiết kế thông qua một cáp nối bên ngoài và đầu thử mô phỏng (hình 7.85).



Hình 7.85: Giao tiếp MDS-ICE

Có các hệ thống ICE khác nhau phụ thuộc vào loại vi xử lý, ví dụ, có ICE-86 để phát triển hệ thống dùng vi xử lý 8086/8088, hoặc ICE-486 cho hệ vi xử lý 80486. Nghĩa là hệ thống ICE-X dùng để phát triển các hệ vi xử lý X. ICE gồm có vi xử lý X (CPU), bộ nhớ (ROM, EPROM, RAM), các bộ định thời gian (timer), khối giao tiếp và logic điều khiển. Các giao tiếp qua bus với hệ thống MDS và với bảng mẫu thử bằng khối mô phỏng, gọi là lớp vỏ mô phỏng (emulator pod). Khối mô phỏng này kết nối với đế cắm CPU (CPU socket) trên bảng mẫu

thử. Như vậy CPU của ICE có thể thay thế chính ngay CPU X của bảng mẫu thử (vì CPU của ICE giống như CPU của mẫu thử) trong quá trình phát triển hệ vi xử lý X. Giao tiếp giữa MDS, ICE và bảng mẫu thử cho phép phần cứng của hệ thống được chia sẻ cho mẫu thử và hệ phát triển để điều khiển sự mô phỏng thông qua MDS. ICE có thể cũng được vận hành mà không cần kết nối với mẫu thử. Như vậy ta có thể thực hiện phát triển phần mềm cho mẫu hệ vi xử lý thử nghiệm ngay cả khi chưa có phần cứng của mẫu hệ thống vi xử lý thử nghiệm.

Phần mềm chạy mô phỏng cho mẫu thử nghiệm có thể được lưu trữ toàn bộ trên bộ nhớ của ICE, hoặc lưu trữ toàn bộ trên bộ nhớ của mẫu thử, hoặc một phần lưu trên ICE một phần lưu trên mẫu thử. Điều này cho phép người phát triển có thể thực hiện kiểm tra từng mô-đun nhớ của mẫu thử riêng biệt. Các khối giao tiếp với các thiết bị ngoài của MDS có thể được sử dụng cho mẫu thử thông qua các thành phần vào/ra của MDS và của mẫu thử. ICE có cũng có thể chạy các chương trình cất giữ trong bộ nhớ của mẫu thử nhờ sử dụng bộ nhớ của mẫu thử và các chip giao tiếp vào/ra.

Có một số lệnh đặc trưng thường dùng trong nhiều thiết bị ICE, đó là:

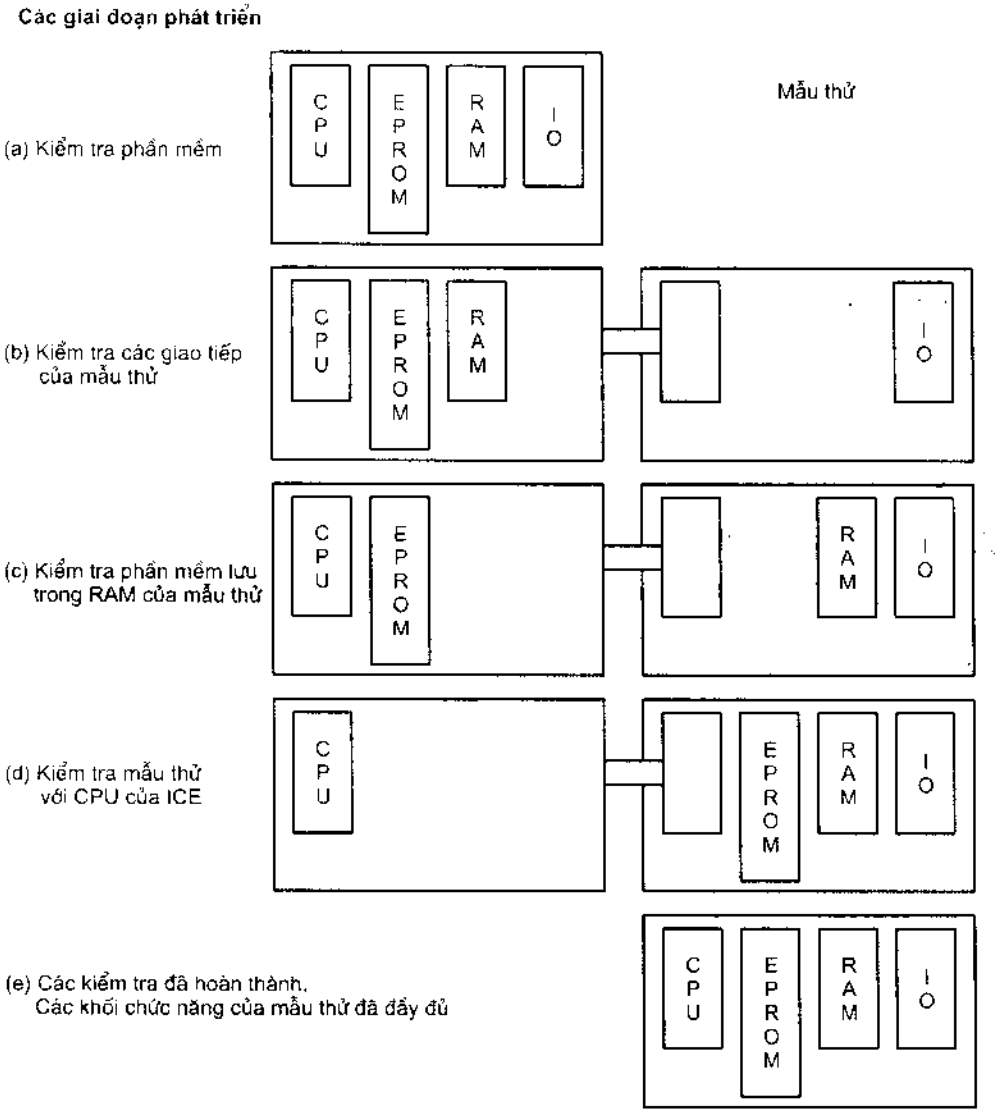
- Go: Bắt đầu mô phỏng cho đến khi một điều kiện dừng thỏa mãn. Người phát triển có thể xác định điểm bắt đầu của quá trình mô phỏng và điểm dừng của quá trình. Điều kiện dừng có thể được xác định bằng các giá trị địa chỉ, dữ liệu, và/hoặc trạng thái bus.
- Step: Cho phép người phát triển chay một chương trình từng bước bắt đầu từ địa chỉ nào đó cho trước.
- Trace: Cho phép người phát triển xác định các điều kiện cho phép và cấm truy xét sự thu thập dữ liệu trong thời gian thực và mô phỏng từng bước. Thông tin truy xét có thể bao gồm thanh ghi hoặc/và các nội dung của bộ nhớ, các cờ, và các tín hiệu ở các-chân tín hiệu nào đó (pin).

- Display: Hiển thị các điều kiện cho truy xét dữ liệu và mô phỏng. Có thể có các dạng biểu diễn hiển thị khác nhau như ASCII, nhị phân, thập phân, thập lục phân).
- Clock: Xác định nhịp đồng hồ lấy từ đâu để sử dụng (bên trong ICE, trong mẫu thử) cho quá trình mô phỏng trên ICE.
- Memory Map: Các hiển thị và khẳng định bộ nhớ nào của hệ phát triển được truy nhập và khoảng nhớ được mượn từ MDS/ICE.

Hình 7.86 là một ví dụ đặc trưng về quá trình phát triển hệ vi xử lý X sử dụng hệ phát triển MDS/ICE-X.

Khởi đầu của quá trình, giai đoạn (a): hệ thống MDS/ICE-X được sử dụng như là một máy vi tính chủ để thực hiện kiểm tra phần mềm của hệ vi xử lý dự định thiết kế (mẫu thử). Tiếp sau đó, giai đoạn (b): Các chip giao tiếp vào/ra được đặt vào bảng mẫu thử và ICE (đã loại bỏ các chip giao tiếp vào/ra) được kết nối với đế cắm CPU của bảng mẫu thử. Như vậy, thực chất hệ thống MDS/ICE-X và bảng mẫu thử vẫn là một máy vi tính như đã có ở giai đoạn (a), chỉ khác là các chip giao tiếp vào/ra ở vị trí trên bảng mẫu thử. Giai đoạn này không chỉ kiểm tra các chip giao tiếp vào/ra mà còn cũng là kiểm tra kết nối của đế cắm chip vi xử lý X (CPU) của mẫu thử, bởi vì đế cắm này nối với bus hệ thống của mẫu thử. Tiếp theo, bước (c) thực hiện kiểm tra bộ nhớ của mẫu thử (RAM) bằng cách lưu chương trình của mẫu thử lên RAM của mẫu thử và thực hiện chạy chương trình. Dĩ nhiên RAM của hệ thống MDS/ICE-X phải được loại ra. Tiếp theo, giai đoạn (d) bằng cách tương tự ta thực hiện kiểm tra hệ thống nhớ gồm EPROM, RAM của mẫu thử với CPU còn lại trên hệ thống MDS/ICE-X. Trong giai đoạn này, cấu hình của mẫu thử là tương tự cấu hình của hệ thống máy vi tính MDS/ICE-X chỉ khác là CPU nằm trên MDS/ICE-X. Sự thành công của giai đoạn này cho phép ta hoàn thành quá trình phát triển hệ vi xử lý X với kết quả nhận được ở giai đoạn (e) là mẫu thử hoàn chỉnh

với việc cắm vào đế cắm CPU chip vi xử lý X. Bây giờ mẫu thử có cấu hình của một máy vi tính tối thiểu.



Hình 7.86: Quá trình phát triển hệ vi xử lý X sử dụng hệ phát triển MDS/ICE-X

Quá trình phát triển một hệ thống vi xử lý X như trên đây là đơn giản. Nó tương tự như quá trình phát triển của bào thai thành một đứa

bé trong bụng mẹ, trong đó người mẹ là hệ thống MDS, ICE là dạ con, và bảng mẫu thử là bào thai. Ban đầu chỉ có người mẹ ở giai đoạn (a). Mẫu thử từng bước lớn dần theo các giai đoạn từ (b) tới (d). Khi trưởng thành ở giai đoạn (e), mẫu thử đã đầy đủ mọi chức năng và khả năng như một máy vi tính tối thiểu, cũng giống như đứa trẻ đã có hình hài đầy đủ và có thể "đứng bằng chính đôi-chân của nó".

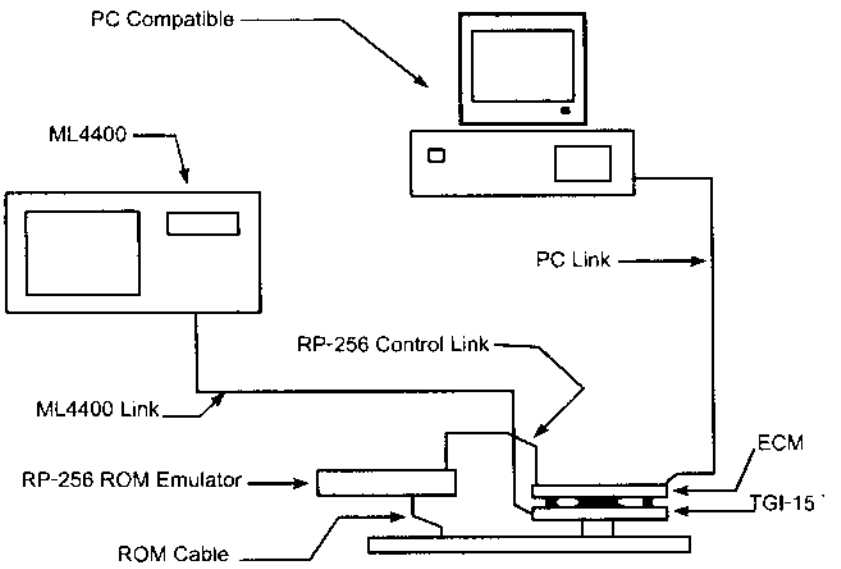
Trên thị trường, có một số nhà sản xuất chuyên về các hệ phát triển. Nhưng có lẽ thông dụng nhất là các hệ phát triển cho các hệ thống vi xử lý Pentium. Do vậy, chúng ta xét qua các loại hệ phát triển này.

7.5.3 Các hệ phát triển các hệ thống vi xử lý Pentium

Có hai loại hệ phát triển cho các hệ thống vi xử lý Pentium của hai nhà chế tạo, đó là công ty American Arium (Tustin, CA) và Microtek International (Hillsboro, OR).

a) Hệ phát triển LA/ICE (American Arium Co.)

Hệ phát LA/ICE dựa trên máy phân tích LA (Analyzer) ML4400 và hệ thống ICE với vi xử lý Pentium. Hệ thống này còn được gọi là ICE-I5 (hình 7.87).



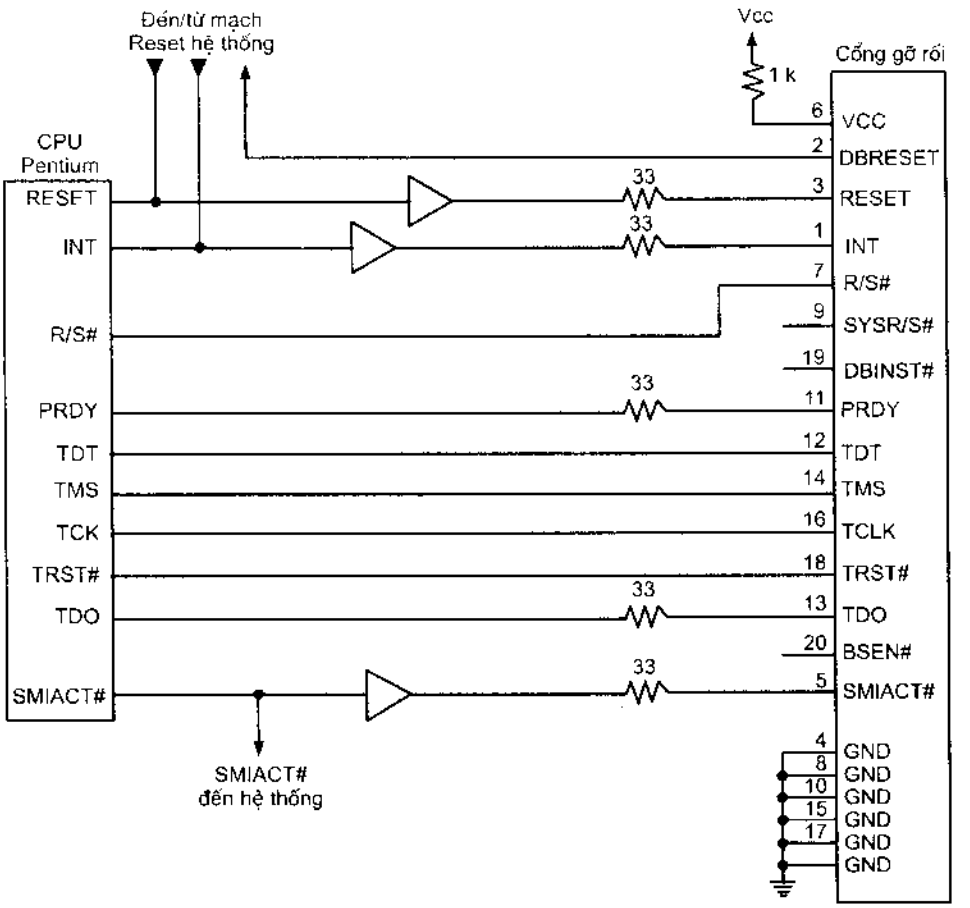
Hình 7.87: Hệ phát triển ML4400

Máy phân tích LA ML4400 là một hệ thống được mô-đun hóa để có người phát triển có thể lựa chọn cấu hình tùy thuộc vào yêu cầu thiết kế. Nó trang bị 400 kênh dữ liệu, bộ nhớ 128 kB với khả năng kiểm tra được theo tiến trình sâu, có tới 14 mức kích hoạt cấu hình khác nhau, đồng thời hỗ trợ tới 4 vi xử lý, hơn 250 bảng hiển thị nội dung cho hỗ trợ và cho tính toán số học, một màn hình CRT cỡ 7 inch, một ổ đĩa mềm 1,44 MB tương thích MS-DOS, mô phỏng ROM, cổng vào/ra nối tiếp để trao đổi dữ liệu với máy tính PC hoặc với máy in, và một khung để cắm các bìa điện tử bổ sung, như giao tiếp điều khiển theo chuẩn IEEE-488. Hệ thống có thể từ 1 đến 4 mô-đun thu để lưu trữ và lọc thông tin lấy từ các hệ thống điện tử khác nhau. Các mô-đun này cắm trên các khe mở rộng phía đằng sau của hệ thống ML4400. Sự kết nối của hệ thống với các mẫu thử được thực hiện thông qua bộ thích ứng giao tiếp TGI (Target Interface Adaptor). Đây là bộ thích ứng giao tiếp tốc độ cao ứng dụng cho cả các loại vi xử lý kiến trúc CISC và RISC. ML4400 có thể được sử dụng cùng với các chip vi điều khiển của Intel như 8031, 8035, 8096, loại vi xử lý 8-bit 8085, Pentium và các loại vi xử lý công nghệ cao X86, 80960A, các loại vi xử lý của Motorola như 6800, 6809, họ M68000, họ M88000, MIPS R3000, và nhiều loại vi xử lý khác nữa.

LA/ICE làm việc ở tần số 66 MHz. Nó có phần cứng phức tạp điều khiển các điểm dừng, gỡ rối (debuging) chương trình nguồn ngôn ngữ C, cổng gỡ rối các chip vi xử lý Intel, mô phỏng ROM dung lượng 1 MB, và chọn lựa thời gian phân tích đến tần số 1GHz. Hệ phát triển này có thể được nâng cấp cho các ứng dụng đa xử lý.

Bộ thích ứng TGI cho phép truy nhập tới các-chân tín hiệu (pin) của vi xử lý mà không cần phải ngắt kết nối địa chỉ, dữ liệu, hoặc các tín hiệu điều khiển bus. Sự kết nối giữa TGI và module điều khiển của bộ mô phỏng ROM ECM (Emulator Control Module) được điều khiển bởi một máy tính PC/X86 chủ và chạy với MS-DOS để chuyển trực tiếp dữ liệu giữa mô-đun thu của ML4400 và bộ mô phỏng ROM. Nhờ phần mềm của máy tính PC mà ML4400 được tự động yêu cầu thiết

lập một cấu trúc kiểm tra theo tiến trình phức tạp, nạp mã nguồn hoặc các thay đổi vào trong bộ mô phỏng ROM, và điều khiển, kiểm soát theo tiến trình các-chân tín hiệu (pin) của CPU. Logic thực hiện các điểm dừng gồm nhiều thành phần, như các bộ xác nhận, các bộ thời gian, các bộ đếm, lọc trước và sau khi kiểm soát tiến trình (trace), và 12 mức của các điểm dừng theo kiểu if-then-else.



Hình 7.88: Sơ đồ thực hiện gỡ rối tối thiểu của hệ thống LA/ICE

Để thực hiện gỡ rối các tín hiệu của chip vi xử lý, hệ phát triển có một cổng gỡ rối, đối với các chip vi xử lý Intel cổng gỡ rối là cổng 20-chân. Hình 7.88 là một cấu hình kết nối tối thiểu cổng gỡ rối (debug port) với chip vi xử lý Pentium, trong đó ký hiệu O chỉ đó là tín

hiệu ra từ vi xử lý, và ký hiệu là tín hiệu vào vi xử lý. Kết nối này không ảnh hưởng đến hoạt động bình thường của chip vi xử lý.

Một số tín hiệu quan trọng trong kết nối này là:

- TDO (O): Test data output (đầu ra dữ liệu kiểm tra), là tín hiệu ra nối tiếp cho logic kiểm tra. Các lệnh và dữ liệu của cổng truy nhập kiểm tra, TAP (Test Access Port) được đẩy từ vi xử lý ra qua-chân TDO này.

- TDI (I): Test data input (đầu vào dữ liệu kiểm tra), là tín hiệu vào nối tiếp cho logic kiểm tra. Các lệnh và dữ liệu của TAP được đẩy vào Pentium thông qua-chân TDI này.

- TMS (I): Test mode select (chọn chế độ kiểm tra), điều khiển trình tự của các thay đổi trạng thái của điều khiển của TAP.

- TCK (I): nhịp đồng hồ khả năng kiểm tra.

- TRST# (I): Xóa kiểm tra, khi chuyển xuống mức tích cực âm (0), nó cho phép điều khiển của TAP được khởi tạo một cách không đồng bộ.

- SMIACT# (O): System management interrupt active (tích cực ngắt quản lý hệ thống), khi được tích cực (0), nó chỉ ra rằng bộ xử lý được vận hành trong chế độ quản lý hệ thống SMM (System Management Mode).

- RESET (O): Khi là tín hiệu ra, nó chỉ cho các mạch bên ngoài biết rằng vi xử lý đã được xóa.

- INIT (O): Khi là tín hiệu ra, nó chỉ ra rằng chip vi xử lý đã được khởi tạo để bắt đầu thực hiện trong một trạng thái đã được biết.

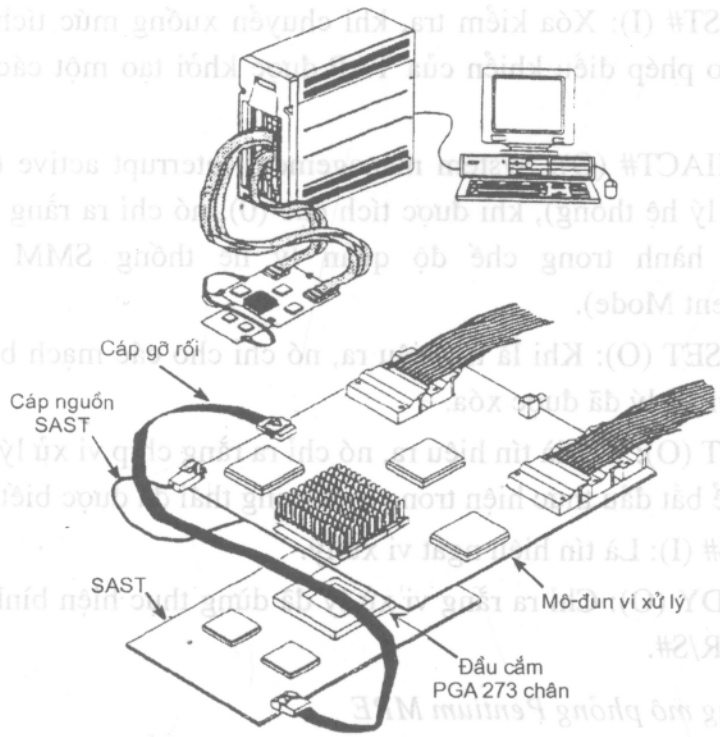
- R/S# (I): Là tín hiệu ngắt vi xử lý.

- PRDY (O): Chỉ ra rằng vi xử lý đã dừng thực hiện bình thường để đáp lại R/S#.

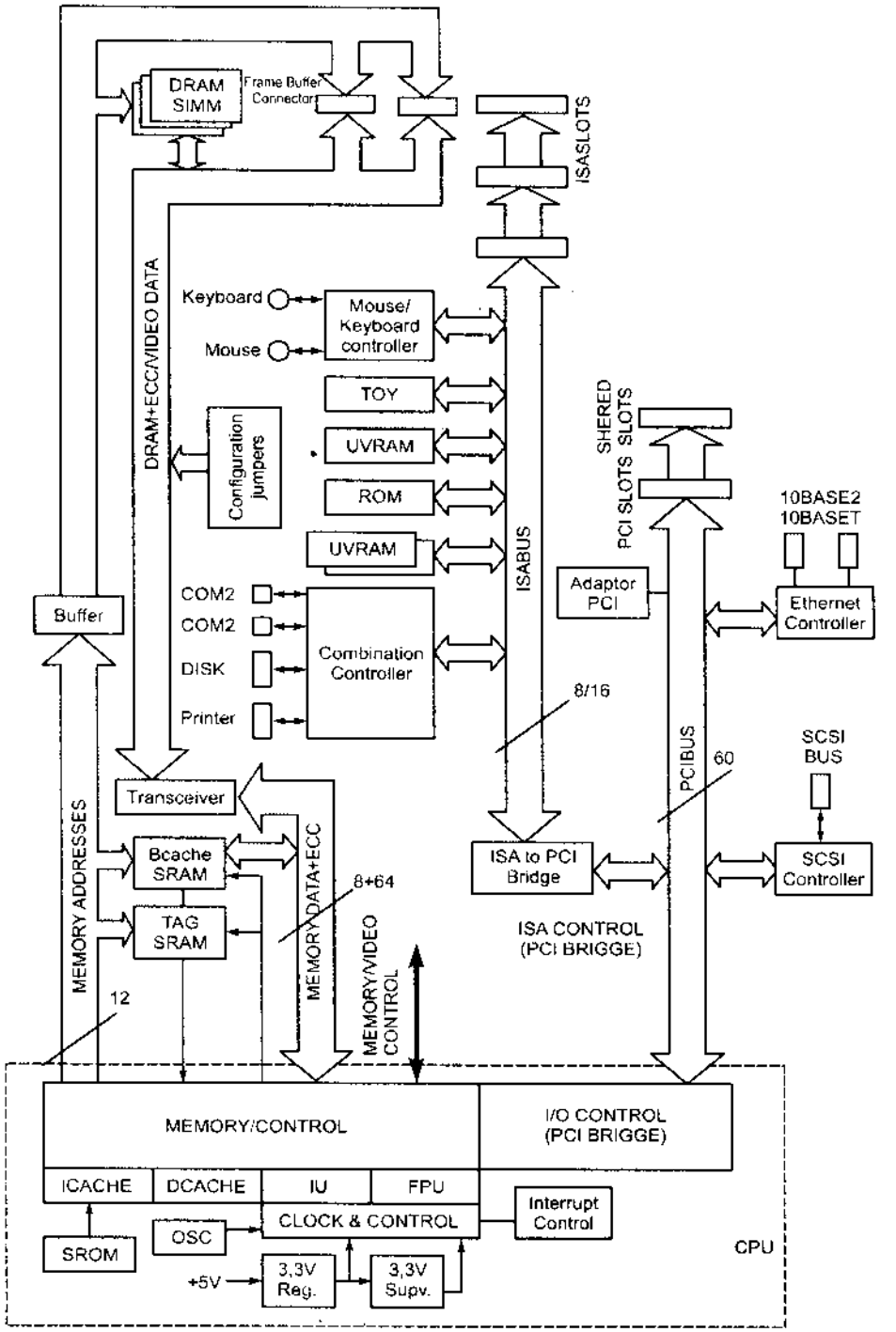
b) Hệ thống mô phỏng Pentium MPE

Hệ thống mô phỏng Pentium của Mikrotek International MPE (Mikrotek Pentium Emulator) làm việc với tần số 66 MHz, có giao tiếp

với Windows 3.1, gỡ rối chương trình nguồn ngôn ngữ C, thực hiện các điểm dừng bằng phần cứng, thực hiện kiểm soát theo tiến trình, truy nhập tới tất cả các thanh ghi bên trong của chip vi xử lý. Hệ thống MPE yêu cầu máy tính PC chủ có cấu hình: vi xử lý từ Intel 80386/33 MHz trở lên, 8 MB RAM, màn hình VGA, HDD tối thiểu 10 MB, COM1, COM2, và MS-DOS phiên bản từ 5.0 trở lên, Windows 3.1. Hình 7.89 là hệ thống MPE và kết nối với PC chủ, mô-đun vi xử lý (processor module), và bảng tự kiểm tra độc lập SAST (Stand-Alone Selt-Test). Hệ thống này cho phép kiểm soát theo tiến trình, gỡ rối chương trình nguồn chương trình nguồn viết ở hợp ngữ hoặc ngôn ngữ C, xác định các điểm dừng, truy nhập các thanh ghi bên trong của vi xử lý, hiển thị nội dung bộ nhớ.



Hình 7.89: Hệ phát triển MPE



Hình 7.90: Sơ đồ khối hệ phát triển EB66

c) Các hệ phát triển của Alpha AXP

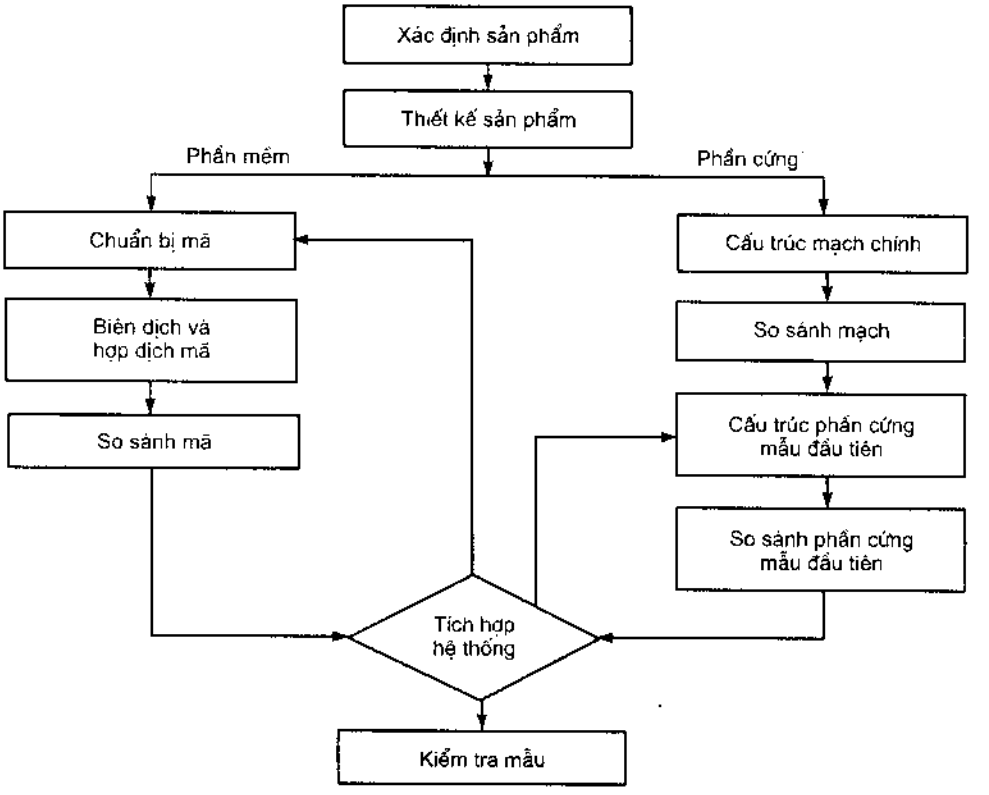
Hệ phát triển của DEC như EB66 (hình 7.90) dùng cho phát triển các hệ thống vi xử lý của DEC như 21066 và 21068. EB66 có phần cứng (ở dạng một bảng mạch độc lập) với kết nối PCI và ISA. CPU của EB66 là chip vi xử lý 21066 làm việc ở 150 MHz hoặc 166 MHz, hoặc là chip 21068/66 MHz. Cả hai chip này đều có các thuộc tính cache bên trong giống như của chip vi xử lý 21064. EB66 có DRAM dung lượng từ 8 đến 256 MB, 256 kB hoặc 1 MB cache dự phòng (backup cache), có hai connectors 100-chân, giao tiếp PCI, bộ điều khiển giao tiếp với hệ thống máy tính nhỏ (SCSI), điều khiển kết nối LAN Ethernet, với ISA-IEEE-P966, bộ điều khiển ngoại vi tốc độ chậm, con chuột và bàn phím điều khiển, ROM (1 MB) giám sát gỡ rối.

BÀI TẬP ÔN LUYỆN

1. Nêu những đặc điểm chung trong cấu trúc của các loại vi xử lý công nghệ cao.
2. Đơn vị ... bên trong làm tăng tốc độ xử lý lệnh:
   a) IU; b) FPU; c) PUIQ; d) CU;
   e) Icache; f) Dcache; g) MMU; h) Tất cả a, b, c, d, e, f, g
3. Thế nào là xử lý song song.
4. Thế nào là đồng xử lý.
5. Thế nào là kiến trúc siêu hướng.
6. Thế nào là kiến trúc đường ống.
7. Phân loại kiến trúc máy tính theo Flynn.
8. Phân loại theo cấu trúc máy tính.
9. Thế nào là hiệu suất của máy tính.
10. Nêu các cấu hình kết nối mạng đa xử lý và so sánh các cấu hình này theo các tiêu chí: tốc độ, độ tin cậy.
11. Thế nào là hiện tượng Hazard trong kỹ thuật đường ống, cách khắc phục.
12. Đặc điểm khác biệt giữa 80286 và 8086/8088.
13. Nêu cấu trúc đường ống lệnh của 80286.
14. Có bao nhiêu thanh ghi hệ thống bên trong 80286, tại sao gọi là thanh ghi hệ thống.
15. Chu kỳ bus của 80286 có bao nhiêu trường hợp.
16. Các tín hiệu INTA#, IOC#, IOWC#, MRDC#, và MWTC# được tạo ra như thế nào?
17. Thế nào là chế độ 8086 của 80286.

18. Thế nào là chế độ bảo vệ của 80286.
19. Hãy nêu các kiểu đánh địa chỉ của 80286.
20. Phân biệt chức năng và tên gọi các giá trị nội dung của các thanh ghi đoạn của 80286 trong chế độ 8086 và chế độ bảo vệ.
21. Chương trình viết trên 8086 có chạy được trên 80286 không, vì sao?
22. Đặc điểm tập lệnh của 80286.
23. Những lệnh nào của 80286 chạy được ở cả 2 chế độ, và những lệnh nào chỉ dùng cho chế độ bảo vệ.
24. Nhiệm vụ của MMU là gì, nhờ MMU không gian đánh địa chỉ sẽ như thế nào?
25. Kỹ thuật bảo vệ theo quyền truy nhập trong hệ thống đa nhiệm là như thế nào, những thanh ghi nào tham gia vào quá trình quản lý bảo vệ này.
26. Thế nào là Call gate, nêu cơ chế chuyển địa chỉ của Call gate.
27. Phân biệt kỹ thuật phân trang và phân đoạn bộ nhớ củaq 80286.
28. Tín hiệu nào phục vụ cho chọn băng nhớ trong hệ thống 80286.
29. Nêu một số vi mạch hỗ trợ cho 80286, chức năng, và kết nối của chúng trong hệ thống 80286.
30. Nêu sự khác về đặc điểm cấu trúc, tính năng kỹ thuật của loại 80386SX, 80386DX.
31. Những hạn chế của 80286.
32. Nêu mô hình lập trình của 80386.
33. Đặc điểm của tập lệnh 80386.
34. Nêu đặc điểm của kỹ thuật đường ống trong 80386.
35. Nêu những đặc điểm của chế độ 8086 và chế độ bảo vệ của 80386, so sánh chúng trong 80286.
36. Nêu đặc điểm của kỹ thuật nhớ xen kẽ.
37. 80386 có những bảng mô tả nào, có giống của 80286 không.
38. Nêu cơ chế chuyển địa chỉ của Call gate của 80386.

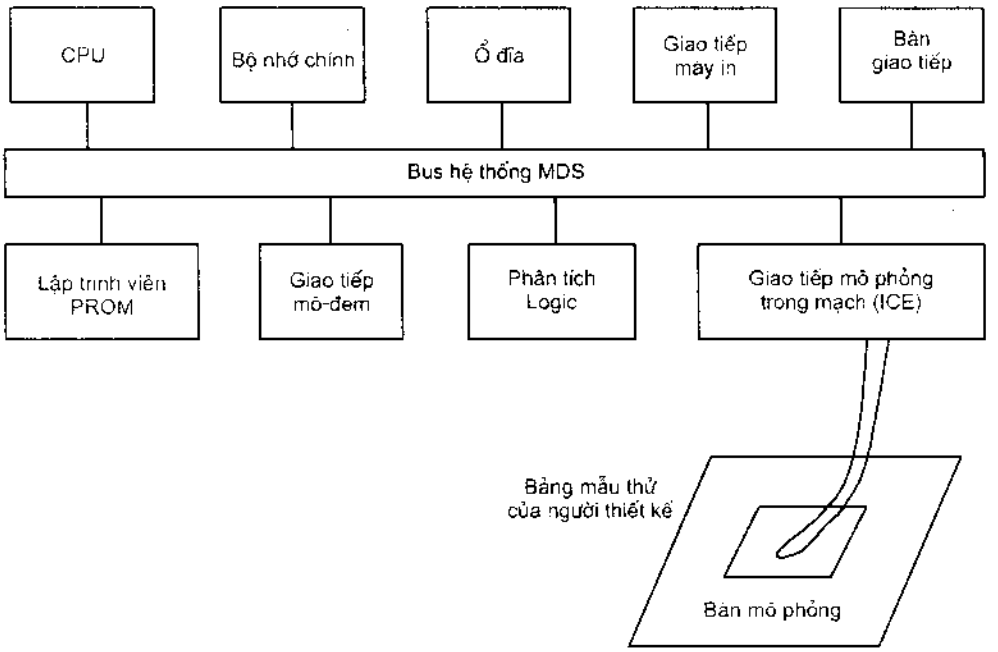
đoạn so sánh. Giai đoạn phát triển hệ thống bắt đầu bằng xây dựng các mạch chính của sản phẩm. Giai đoạn xây dựng mẫu đầu tiên được thực hiện bằng cách từng bước xếp đặt các đế cắm cho các chip vi mạch phù hợp với vị trí liên kết của chúng mà bước thiết kế sản phẩm đã chuẩn bị. Phần cứng và phần mềm của thiết kế mẫu được chuyển đến giai đoạn tích hợp hệ thống. Giai đoạn này thực hiện sự hiệu chỉnh cho cả phần cứng và phần mềm. Khi có phát sinh lỗi thì quay trở lại hiệu chỉnh (hiệu chỉnh phần cứng và phần mềm). Giai đoạn kiểm tra mẫu chỉ được thực hiện khi tích hợp hệ thống hoàn thành. Sự kiểm tra mẫu thường thực hiện cho từng module. Toàn bộ chu trình phát triển cần được thực hiện nhờ công cụ gọi là hệ phát triển máy vi tính MDS (Microcomputer Devolopement System).



Hình 7.83: Chu trình phát triển máy vi tính

7.5.2 Hệ phát triển máy vi tính (MDS)

MDS cung cấp cho người phát triển tất cả các hệ thống nhỏ cần thiết trong từng giai đoạn của chu trình phát triển. MDS là một máy vi tính có phần cứng và phần mềm thân thiện, hiệu quả đủ cho người sử dụng phát triển các hệ thống dựa trên cơ sở vi xử lý. Sơ đồ khối của một hệ thống MDS đặc trưng cho trong hình 7.84.



Hình 7.84: Hệ phát triển máy vi tính (MDS)

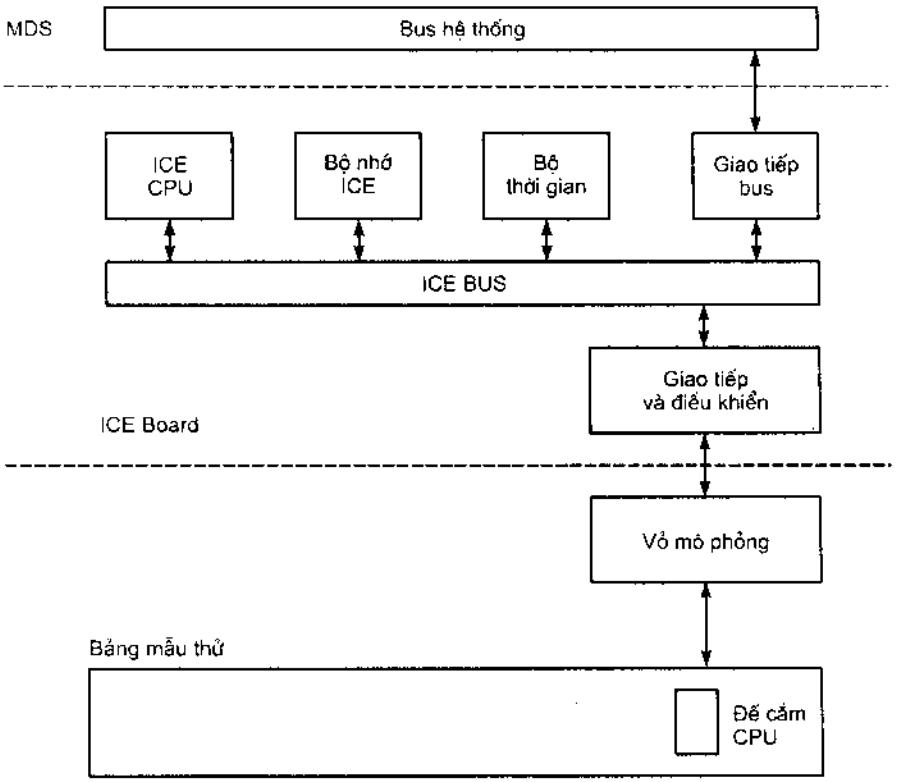
Vì MDS là một máy vi tính nên nó bao gồm:

- CPU, bộ nhớ chính (RAM, ROM), bộ nhớ thứ cấp (bộ nhớ ngoài trên đĩa từ), các giao tiếp vào/ra (máy in, mô-đem, màn hình, bàn phím).

- Một số loại MDS có thể có vài CPU để thực hiện kiểm tra và các chương trình giao tiếp đồng thời. Các hệ thống con trong MDS là những khối chuyên dụng để phát triển phần cứng như:

- Thiết bị lập trình PROM, giao tiếp phân tích logic, giao tiếp mô phỏng trong mạch ICEI (In-circuit emulator interface).

ICE là một hệ vi xử lý hoàn chỉnh có CPU riêng, bộ nhớ, mạch định thời (logic tạo nhịp riêng), và giao tiếp vào/ra. ICE giao tiếp trực tiếp với mẫu thiết kế thông qua một cáp nối bên ngoài và đầu thử mô phỏng (hình 7.85).



Hình 7.85: Giao tiếp MDS-ICE

Có các hệ thống ICE khác nhau phụ thuộc vào loại vi xử lý, ví dụ, có ICE-86 để phát triển hệ thống dùng vi xử lý 8086/8088, hoặc ICE-486 cho hệ vi xử lý 80486. Nghĩa là hệ thống ICE-X dùng để phát triển các hệ vi xử lý X. ICE gồm có vi xử lý X (CPU), bộ nhớ (ROM, EPROM, RAM), các bộ định thời gian (timer), khối giao tiếp và logic điều khiển. Các giao tiếp qua bus với hệ thống MDS và với bảng mẫu thử bằng khối mô phỏng, gọi là lớp vỏ mô phỏng (emulator pod). Khối mô phỏng này kết nối với đế cắm CPU (CPU socket) trên bảng mẫu

thử. Như vậy CPU của ICE có thể thay thế chính ngay CPU X của bảng mẫu thử (vì CPU của ICE giống như CPU của mẫu thử) trong quá trình phát triển hệ vi xử lý X. Giao tiếp giữa MDS, ICE và bảng mẫu thử cho phép phần cứng của hệ thống được chia sẻ cho mẫu thử và hệ phát triển để điều khiển sự mô phỏng thông qua MDS. ICE có thể cũng được vận hành mà không cần kết nối với mẫu thử. Như vậy ta có thể thực hiện phát triển phần mềm cho mẫu hệ vi xử lý thử nghiệm ngay cả khi chưa có phần cứng của mẫu hệ thống vi xử lý thử nghiệm.

Phần mềm chạy mô phỏng cho mẫu thử nghiệm có thể được lưu trữ toàn bộ trên bộ nhớ của ICE, hoặc lưu trữ toàn bộ trên bộ nhớ của mẫu thử, hoặc một phần lưu trên ICE một phần lưu trên mẫu thử. Điều này cho phép người phát triển có thể thực hiện kiểm tra từng mô-đun nhớ của mẫu thử riêng biệt. Các khối giao tiếp với các thiết bị ngoài của MDS có thể được sử dụng cho mẫu thử thông qua các thành phần vào/ra của MDS và của mẫu thử. ICE có cũng có thể chạy các chương trình cất giữ trong bộ nhớ của mẫu thử nhờ sử dụng bộ nhớ của mẫu thử và các chip giao tiếp vào/ra.

Có một số lệnh đặc trưng thường dùng trong nhiều thiết bị ICE, đó là:

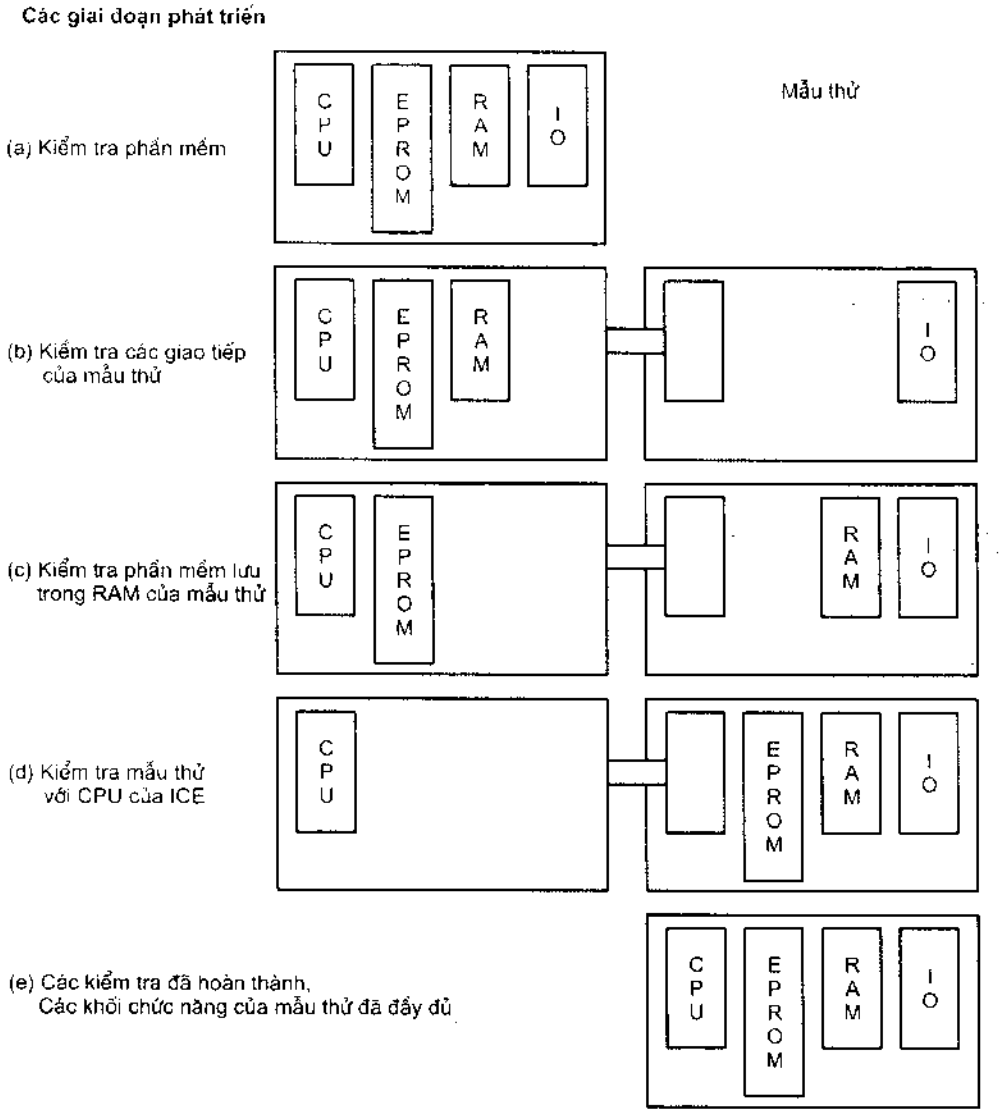
- Go: Bắt đầu mô phỏng cho đến khi một điều kiện dừng thỏa mãn. Người phát triển có thể xác định điểm bắt đầu của quá trình mô phỏng và điểm dừng của quá trình. Điều kiện dừng có thể được xác định bằng các giá trị địa chỉ, dữ liệu, và/hoặc trạng thái bus.
- Step: Cho phép người phát triển chay một chương trình từng bước bắt đầu từ địa chỉ nào đó cho trước.
- Trace: Cho phép người phát triển xác định các điều kiện cho phép và cấm truy xét sự thu thập dữ liệu trong thời gian thực và mô phỏng từng bước. Thông tin truy xét có thể bao gồm thanh ghi hoặc/và các nội dung của bộ nhớ, các cờ, và các tín hiệu ở các-chân tín hiệu nào đó (pin).

- Display: Hiển thị các điều kiện cho truy xét dữ liệu và mô phỏng. Có thể có các dạng biểu diễn hiển thị khác nhau như ASCII, nhị phân, thập phân, thập lục phân).
- Clock: Xác định nhịp đồng hồ lấy từ đâu để sử dụng (bên trong ICE, trong mẫu thử) cho quá trình mô phỏng trên ICE.
- Memory Map: Các hiển thị và khẳng định bộ nhớ nào của hệ phát triển được truy nhập và khoảng nhớ được mượn từ MDS/ICE.

Hình 7.86 là một ví dụ đặc trưng về quá trình phát triển hệ vi xử lý X sử dụng hệ phát triển MDS/ICE-X.

Khởi đầu của quá trình, giai đoạn (a): hệ thống MDS/ICE-X được sử dụng như là một máy vi tính chủ để thực hiện kiểm tra phần mềm của hệ vi xử lý dự định thiết kế (mẫu thử). Tiếp sau đó, giai đoạn (b): Các chip giao tiếp vào/ra được đặt vào bảng mẫu thử và ICE (đã loại bỏ các chip giao tiếp vào/ra) được kết nối với đế cắm CPU của bảng mẫu thử. Như vậy, thực chất hệ thống MDS/ICE-X và bảng mẫu thử vẫn là một máy vi tính như đã có ở giai đoạn (a), chỉ khác là các chip giao tiếp vào/ra ở vị trí trên bảng mẫu thử. Giai đoạn này không chỉ kiểm tra các chip giao tiếp vào/ra mà còn cũng là kiểm tra kết nối của đế cắm chip vi xử lý X (CPU) của mẫu thử, bởi vì đế cắm này nối với bus hệ thống của mẫu thử. Tiếp theo, bước (c) thực hiện kiểm tra bộ nhớ của mẫu thử (RAM) bằng cách lưu chương trình của mẫu thử lên RAM của mẫu thử và thực hiện chạy chương trình. Dĩ nhiên RAM của hệ thống MDS/ICE-X phải được loại ra. Tiếp theo, giai đoạn (d) bằng cách tương tự ta thực hiện kiểm tra hệ thống nhớ gồm EPROM, RAM của mẫu thử với CPU còn lại trên hệ thống MDS/ICE-X. Trong giai đoạn này, cấu hình của mẫu thử là tương tự cấu hình của hệ thống máy vi tính MDS/ICE-X chỉ khác là CPU nằm trên MDS/ICE-X. Sự thành công của giai đoạn này cho phép ta hoàn thành quá trình phát triển hệ vi xử lý X với kết quả nhận được ở giai đoạn (e) là mẫu thử hoàn chỉnh

với việc cắm vào đế cắm CPU chip vi xử lý X. Bây giờ mẫu thử có cấu hình của một máy vi tính tối thiểu.



Hình 7.86: Quá trình phát triển hệ vi xử lý X sử dụng hệ phát triển MDS/ICE-X

Quá trình phát triển một hệ thống vi xử lý X như trên đây là đơn giản. Nó tương tự như quá trình phát triển của bào thai thành một đứa

bé trong bụng mẹ, trong đó người mẹ là hệ thống MDS, ICE là dạ con, và bảng mẫu thử là bào thai. Ban đầu chỉ có người mẹ ở giai đoạn (a). Mẫu thử từng bước lớn dần theo các giai đoạn từ (b) tới (d). Khi trưởng thành ở giai đoạn (e), mẫu thử đã đầy đủ mọi chức năng và khả năng như một máy vi tính tối thiểu, cũng giống như đứa trẻ đã có hình hài đầy đủ và có thể "đứng bằng chính đôi-chân của nó".

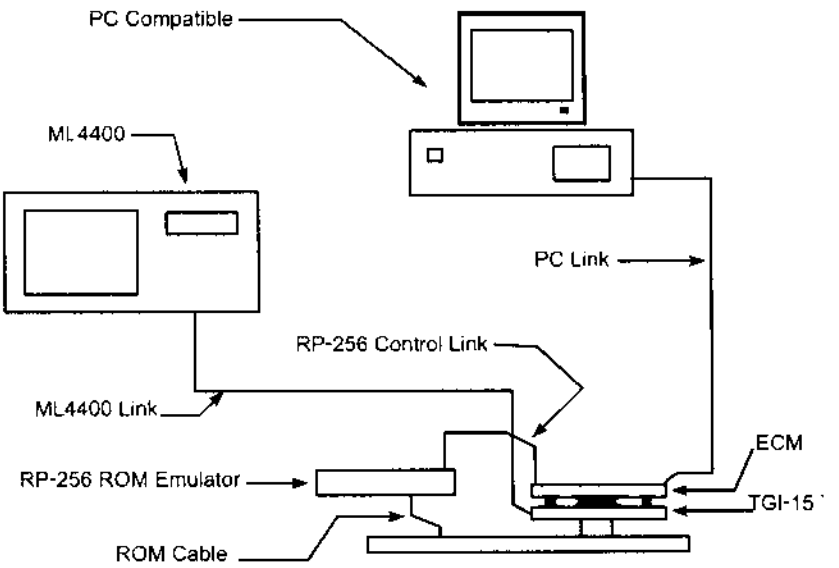
Trên thị trường, có một số nhà sản xuất chuyên về các hệ phát triển. Nhưng có lẽ thông dụng nhất là các hệ phát triển cho các hệ thống vi xử lý Pentium. Do vậy, chúng ta xét qua các loại hệ phát triển này.

7.5.3 Các hệ phát triển các hệ thống vi xử lý Pentium

Có hai loại hệ phát triển cho các hệ thống vi xử lý Pentium của hai nhà chế tạo, đó là công ty American Arium (Tustin, CA) và Microtek International (Hillsboro, OR).

a) Hệ phát triển LA/ICE (American Arium Co.)

Hệ phát LA/ICE dựa trên máy phân tích LA (Analyzer) ML4400 và hệ thống ICE với vi xử lý Pentium. Hệ thống này còn được gọi là ICE-I5 (hình 7.87).



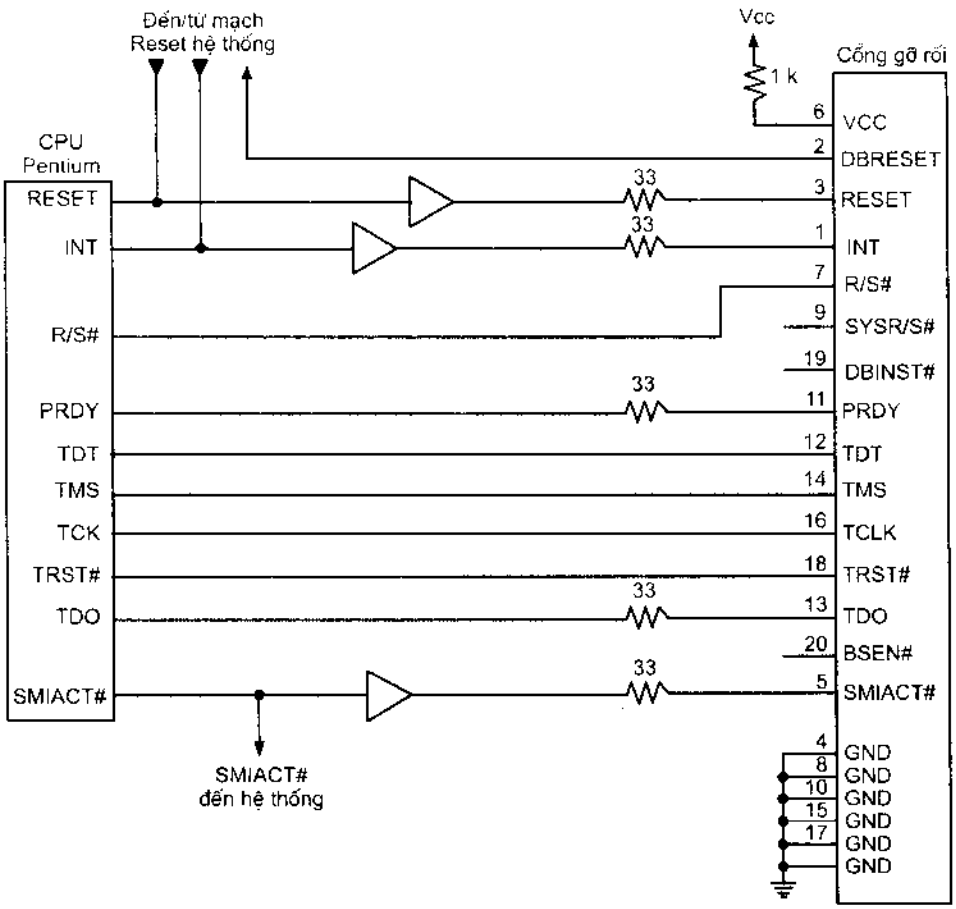
Hình 7.87: Hệ phát triển ML4400

Máy phân tích LA ML4400 là một hệ thống được mô-đun hóa để có người phát triển có thể lựa chọn cấu hình tùy thuộc vào yêu cầu thiết kế. Nó trang bị 400 kênh dữ liệu, bộ nhớ 128 kB với khả năng kiểm tra được theo tiến trình sâu, có tới 14 mức kích hoạt cấu hình khác nhau, đồng thời hỗ trợ tới 4 vi xử lý, hơn 250 bảng hiển thị nội dung cho hỗ trợ và cho tính toán số học, một màn hình CRT cỡ 7 inch, một ổ đĩa mềm 1,44 MB tương thích MS-DOS, mô phỏng ROM, cổng vào/ra nối tiếp để trao đổi dữ liệu với máy tính PC hoặc với máy in, và một khung để cắm các bìa điện tử bổ sung, như giao tiếp điều khiển theo chuẩn IEEE-488. Hệ thống có thể từ 1 đến 4 mô-đun thu để lưu trữ và lọc thông tin lấy từ các hệ thống điện tử khác nhau. Các mô-đun này cắm trên các khe mở rộng phía đằng sau của hệ thống ML4400. Sự kết nối của hệ thống với các mẫu thử được thực hiện thông qua bộ thích ứng giao tiếp TGI (Target Interface Adaptor). Đây là bộ thích ứng giao tiếp tốc độ cao ứng dụng cho cả các loại vi xử lý kiến trúc CISC và RISC. ML4400 có thể được sử dụng cùng với các chip vi điều khiển của Intel như 8031, 8035, 8096, loại vi xử lý 8-bit 8085, Pentium và các loại vi xử lý công nghệ cao X86, 80960A, các loại vi xử lý của Motorola như 6800, 6809, họ M68000, họ M88000, MIPS R3000, và nhiều loại vi xử lý khác nữa.

LA/ICE làm việc ở tần số 66 MHz. Nó có phần cứng phức tạp điều khiển các điểm dừng, gỡ rối (debuging) chương trình nguồn ngôn ngữ C, cổng gỡ rối các chip vi xử lý Intel, mô phỏng ROM dung lượng 1 MB, và chọn lựa thời gian phân tích đến tần số 1GHz. Hệ phát triển này có thể được nâng cấp cho các ứng dụng đa xử lý.

Bộ thích ứng TGI cho phép truy nhập tới các-chân tín hiệu (pin) của vi xử lý mà không cần phải ngắt kết nối địa chỉ, dữ liệu, hoặc các tín hiệu điều khiển bus. Sự kết nối giữa TGI và module điều khiển của bộ mô phỏng ROM ECM (Emulator Control Module) được điều khiển bởi một máy tính PC/X86 chủ và chạy với MS-DOS để chuyển trực tiếp dữ liệu giữa mô-đun thu của ML4400 và bộ mô phỏng ROM. Nhờ phần mềm của máy tính PC mà ML4400 được tự động yêu cầu thiết

lập một cấu trúc kiểm tra theo tiến trình phức tạp, nạp mã nguồn hoặc các thay đổi vào trong bộ mô phỏng ROM, và điều khiển, kiểm soát theo tiến trình các-chân tín hiệu (pin) của CPU. Logic thực hiện các điểm dừng gồm nhiều thành phần, như các bộ xác nhận, các bộ thời gian, các bộ đếm, lọc trước và sau khi kiểm soát tiến trình (trace), và 12 mức của các điểm dừng theo kiểu if-then-else.



Hình 7.88: Sơ đồ thực hiện gỡ rối tối thiểu của hệ thống LA/ICE

Để thực hiện gỡ rối các tín hiệu của chip vi xử lý, hệ phát triển có một cổng gỡ rối, đối với các chip vi xử lý Intel cổng gỡ rối là cổng 20-chân. Hình 7.88 là một cấu hình kết nối tối thiểu cổng gỡ rối (debug port) với chip vi xử lý Pentium, trong đó ký hiệu O chỉ đó là tín

hiệu ra từ vi xử lý, và ký hiệu là tín hiệu vào vi xử lý. Kết nối này không ảnh hưởng đến hoạt động bình thường của chip vi xử lý.

Một số tín hiệu quan trọng trong kết nối này là:

- TDO (O): Test data output (đầu ra dữ liệu kiểm tra), là tín hiệu ra nối tiếp cho logic kiểm tra. Các lệnh và dữ liệu của cổng truy nhập kiểm tra, TAP (Test Access Port) được đẩy từ vi xử lý ra qua-chân TDO này.

- TDI (I): Test data input (đầu vào dữ liệu kiểm tra), là tín hiệu vào nối tiếp cho logic kiểm tra. Các lệnh và dữ liệu của TAP được đẩy vào Pentium thông qua-chân TDI này.

- TMS (I): Test mode select (chọn chế độ kiểm tra), điều khiển trình tự của các thay đổi trạng thái của điều khiển của TAP.

- TCK (I): nhịp đồng hồ khả năng kiểm tra.

- TRST# (I): Xóa kiểm tra, khi chuyển xuống mức tích cực âm (0), nó cho phép điều khiển của TAP được khởi tạo một cách không đồng bộ.

- SMIACT# (O): System management interrupt active (tích cực ngắt quản lý hệ thống), khi được tích cực (0), nó chỉ ra rằng bộ xử lý được vận hành trong chế độ quản lý hệ thống SMM (System Management Mode).

- RESET (O): Khi là tín hiệu ra, nó chỉ cho các mạch bên ngoài biết rằng vi xử lý đã được xóa.

- INIT (O): Khi là tín hiệu ra, nó chỉ ra rằng chip vi xử lý đã được khởi tạo để bắt đầu thực hiện trong một trạng thái đã được biết.

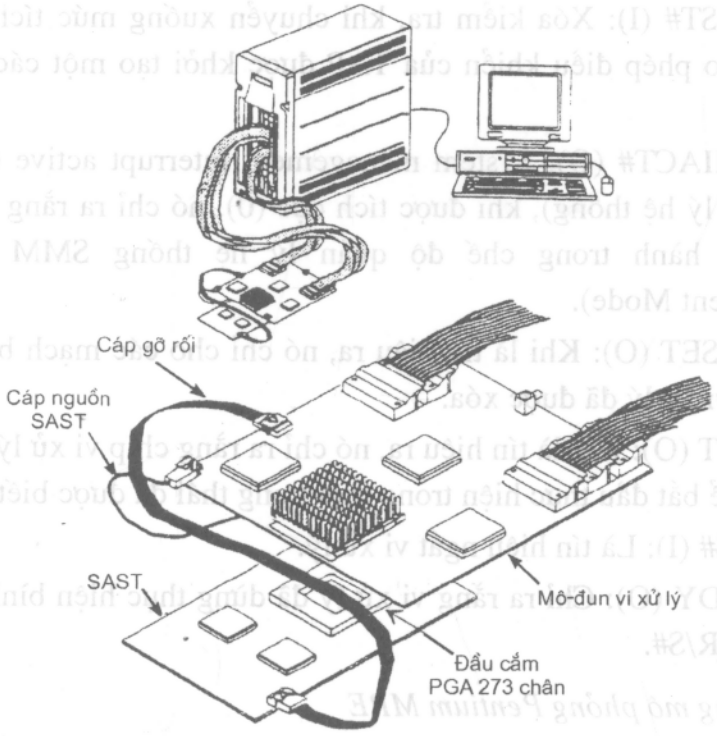
- R/S# (I): Là tín hiệu ngắt vi xử lý.

- PRDY (O): Chỉ ra rằng vi xử lý đã dừng thực hiện bình thường để đáp lại R/S#.

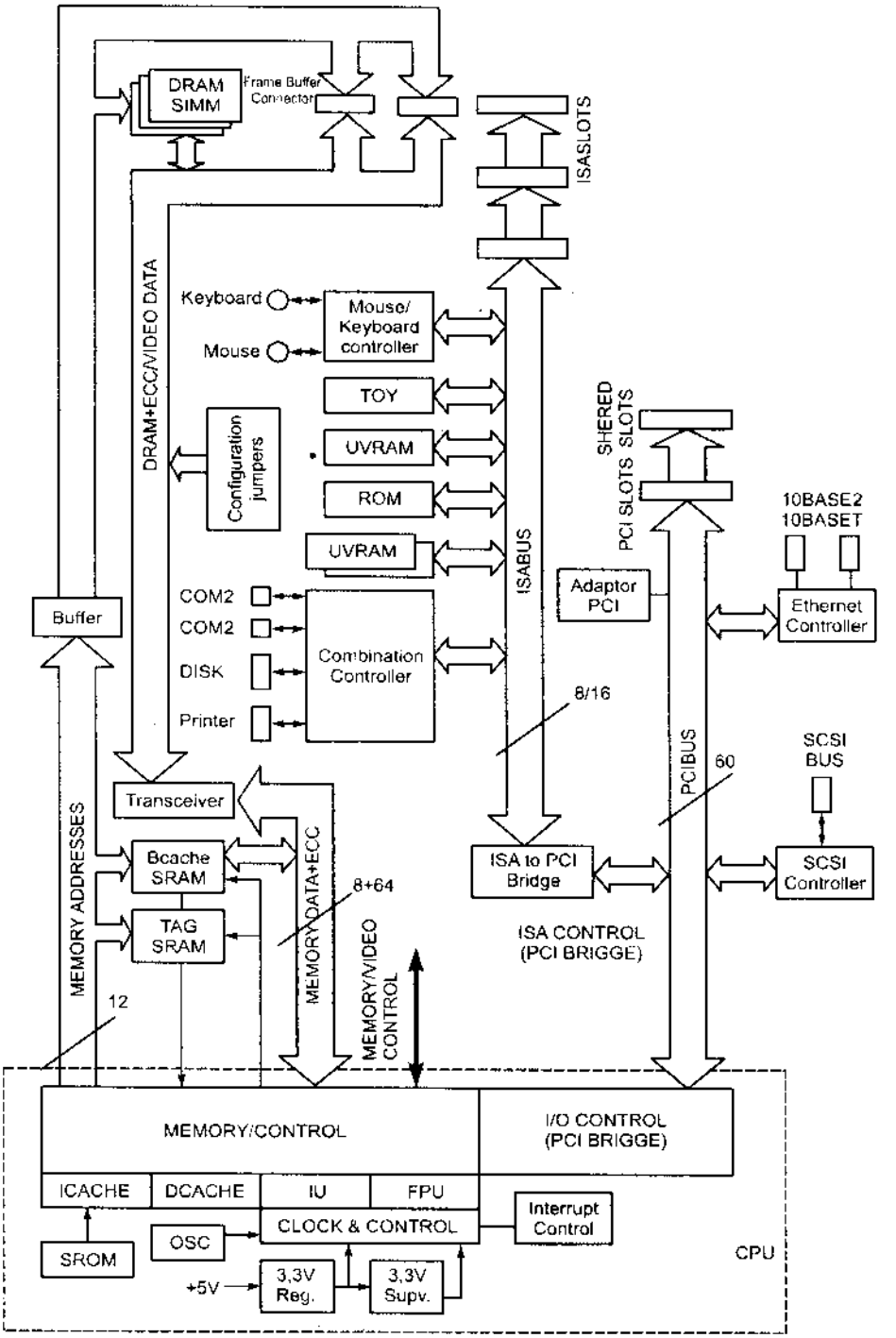
b) Hệ thống mô phỏng Pentium MPE

Hệ thống mô phỏng Pentium của Mikrotek International MPE (Mikrotek Pentium Emulator) làm việc với tần số 66 MHz, có giao tiếp

với Windows 3.1, gỡ rối chương trình nguồn ngôn ngữ C, thực hiện các điểm dừng bằng phần cứng, thực hiện kiểm soát theo tiến trình, truy nhập tới tất cả các thanh ghi bên trong của chip vi xử lý. Hệ thống MPE yêu cầu máy tính PC chủ có cấu hình: vi xử lý từ Intel 80386/33 MHz trở lên, 8 MB RAM, màn hình VGA, HDD tối thiểu 10 MB, COM1, COM2, và MS-DOS phiên bản từ 5.0 trở lên, Windows 3.1. Hình 7.89 là hệ thống MPE và kết nối với PC chủ, mô-đun vi xử lý (processor module), và bảng tự kiểm tra độc lập SAST (Stand-Alone Selt-Test). Hệ thống này cho phép kiểm soát theo tiến trình, gỡ rối chương trình nguồn chương trình nguồn viết ở hợp ngữ hoặc ngôn ngữ C, xác định các điểm dừng, truy nhập các thanh ghi bên trong của vi xử lý, hiển thị nội dung bộ nhớ.



Hình 7.89: Hệ phát triển MPE



Hình 7.90: Sơ đồ khối hệ phát triển EB66

c) Các hệ phát triển của Alpha AXP

Hệ phát triển của DEC như EB66 (hình 7.90) dùng cho phát triển các hệ thống vi xử lý của DEC như 21066 và 21068. EB66 có phần cứng (ở dạng một bảng mạch độc lập) với kết nối PCI và ISA. CPU của EB66 là chip vi xử lý 21066 làm việc ở 150 MHz hoặc 166 MHz, hoặc là chip 21068/66 MHz. Cả hai chip này đều có các thuộc tính cache bên trong giống như của chip vi xử lý 21064. EB66 có DRAM dung lượng từ 8 đến 256 MB, 256 kB hoặc 1 MB cache dự phòng (backup cache), có hai connectors 100-chân, giao tiếp PCI, bộ điều khiển giao tiếp với hệ thống máy tính nhỏ (SCSI), điều khiển kết nối LAN Ethernet, với ISA-IEEE-P966, bộ điều khiển ngoại vi tốc độ chậm, con chuột và bàn phím điều khiển, ROM (1 MB) giám sát gỡ rối.

BÀI TẬP ÔN LUYỆN

1. Nêu những đặc điểm chung trong cấu trúc của các loại vi xử lý công nghệ cao.
2. Đơn vị ... bên trong làm tăng tốc độ xử lý lệnh:
   a) IU; b) FPU; c) PUIQ; d) CU;
   e) Icache; f) Dcache; g) MMU; h) Tất cả a, b, c, d, e, f, g
3. Thế nào là xử lý song song.
4. Thế nào là đồng xử lý.
5. Thế nào là kiến trúc siêu hướng.
6. Thế nào là kiến trúc đường ống.
7. Phân loại kiến trúc máy tính theo Flynn.
8. Phân loại theo cấu trúc máy tính.
9. Thế nào là hiệu suất của máy tính.
10. Nêu các cấu hình kết nối mạng đa xử lý và so sánh các cấu hình này theo các tiêu chí: tốc độ, độ tin cậy.
11. Thế nào là hiện tượng Hazard trong kỹ thuật đường ống, cách khắc phục.
12. Đặc điểm khác biệt giữa 80286 và 8086/8088.
13. Nêu cấu trúc đường ống lệnh của 80286.
14. Có bao nhiêu thanh ghi hệ thống bên trong 80286, tại sao gọi là thanh ghi hệ thống.
15. Chu kỳ bus của 80286 có bao nhiêu trường hợp.
16. Các tín hiệu INTA#, IOC#, IOWC#, MRDC#, và MWTC# được tạo ra như thế nào?
17. Thế nào là chế độ 8086 của 80286.

18. Thế nào là chế độ bảo vệ của 80286.
19. Hãy nêu các kiểu đánh địa chỉ của 80286.
20. Phân biệt chức năng và tên gọi các giá trị nội dung của các thanh ghi đoạn của 80286 trong chế độ 8086 và chế độ bảo vệ.
21. Chương trình viết trên 8086 có chạy được trên 80286 không, vì sao?
22. Đặc điểm tập lệnh của 80286.
23. Những lệnh nào của 80286 chạy được ở cả 2 chế độ, và những lệnh nào chỉ dùng cho chế độ bảo vệ.
24. Nhiệm vụ của MMU là gì, nhờ MMU không gian đánh địa chỉ sẽ như thế nào?
25. Kỹ thuật bảo vệ theo quyền truy nhập trong hệ thống đa nhiệm là như thế nào, những thanh ghi nào tham gia vào quá trình quản lý bảo vệ này.
26. Thế nào là Call gate, nêu cơ chế chuyển địa chỉ của Call gate.
27. Phân biệt kỹ thuật phân trang và phân đoạn bộ nhớ cuaq 80286.
28. Tín hiệu nào phục vụ cho chọn băng nhớ trong hệ thống 80286.
29. Nêu một số vi mạch hỗ trợ cho 80286, chức năng, và kết nối của chúng trong hệ thống 80286.
30. Nêu sự khác về đặc điểm cấu trúc, tính năng kỹ thuật của loại 80386SX, 80386DX.
31. Những hạn chế của 80286.
32. Nêu mô hình lập trình của 80386.
33. Đặc điểm của tập lệnh 80386.
34. Nêu đặc điểm của kỹ thuật đường ống trong 80386.
35. Nêu những đặc điểm của chế độ 8086 và chế độ bảo vệ của 80386, so sánh chúng trong 80286.
36. Nêu đặc điểm của kỹ thuật nhớ xen kẽ.
37. 80386 có những bảng mô tả nào, có giống của 80286 không.
38. Nêu cơ chế chuyển địa chỉ của Call gate của 80386.

39. Thế nào là chuyển đổi địa chỉ tuyến tính, nêu ví dụ chuyển đổi địa chỉ tuyến tính.
40. MMU của 80386 có khác gì so với MMU trong 80286.
41. Kỹ thuật phân trang và phân đoạn trong hệ thống 80386.
42. Giải thích cơ chế quản lý nhiệm vụ trong hệ thống 80386.
43. Dùng kỹ thuật nào để kết nối với các bộ nhớ RAM chậm trong hệ thống 80386.
44. Những đặc điểm cấu trúc của 80486 so với 80386
45. Phân biệt 80486SX và 80486DX.
46. Các kỹ thuật quản lý bộ nhớ, phân trang, phân đoạn của 80486 và so sánh chúng với 80386.
47. So sánh Cache của Pentium với 80X86.
48. Các tín hiệu kết nối chọn băng bộ nhớ của 80486, vẽ các sơ đồ kết nối các băng nhớ.
49. Đặc điểm của Cache trong của 80486 so với 80386.
50. Chiến lược thay thế Cache trong 80486 là như thế nào.
51. Pentium được gọi là vi xử lý bao nhiêu bit, tại sao.
52. So sánh kỹ thuật đường ống lệnh trong Pentium với 80X86.
53. Nêu kỹ thuật MMU, phân trang, phân đoạn, quản lý nhiệm vụ của pentium và so sánh với 80X86.
54. Phân biệt các kỹ thuật đóng của các loại Pentium
55. Sự khác nhau cơ bản của các loại Pentium MMX, Celeron, và Xeon.
56. Thế nào là kỹ thuật MMX.
57. Thế nào là các mở rộng SIMD trong các loại Pentium.
58. Phân biệt các Chipsset cho các loại Pentium.
59. Vẽ sơ đồ kết nối CPU, một số thiết bị ngoại vi, bộ nhớ với một loại Chipset.

60. Nêu đặc điểm của mô hình lập trình của họ vi xử lý Motorola 68XXX, so sánh chúng với mô hình lập trình của họ Intel X86.
61. Hãy so sánh các chế độ bảo vệ nhiều lớp của 80486 với các loại Pentium.
62. Bus 64-bit có trong các hệ thống vi xử lý intel nào.
63. Pentium IV cho phép ghép nối bọ nhớ dung lượng tối đa là bao nhiêu, tại sao.
64. Nêu đặc điểm công nghệ Centrino Mobile.
65. So sánh các đặc điểm kiến trúc của Pentium V, Pentium VI.
66. So sánh các đặc điểm kiến trúc của Pentium II, Pentium III, Pentium IV.
67. So sánh các đặc điểm kiến trúc của Pentium IV, Pentium M
68. Nêu đặc điểm cơ chế hoạt động của chip điều khiển nguồn nuôi cho lõi vi xử lý Pentium M.
69. Nêu các loại chip điều khiển nguồn nuôi cho lõi vi xử lý, so sánh chúng, và trình bày các ứng dụng của chúng.
70. Vì sao công nghệ Centrino Mobile lại cho phép tiết kiệm nguồn nuôi.
71. Chipset 855 gồm bao nhiêu loại.
72. Giao tiếp mạng vô tuyến Intel Pro 2100 thỏa mãn chuẩn IEEE nào với những đặc điểm gì.
73. Tần số vô tuyến mà mạng WLAN sử dụng là những loại nào, thuộc băng tần nào. Có phải đăng ký tần số sử dụng không.
74. Công suất phát vô tuyến trong kết nối WLAN được giới hạn là bao nhiêu?
75. Họ 68XXX có kiến trúc ....
    a) CISC;
    b) RISC;
    c) Tất cả a, b
76. Nêu các chế độ đánh địa chỉ của họ 68XXX, so sánh với Intel X86.

77. Nêu đặc điểm tập lệnh và cấu trúc lệnh của 68XXX, so sánh với tập lệnh của Intel X86.
78. Nêu cơ chế chuyển đổi địa chỉ trong hệ thống 68XXX, và so sánh với Intel X86.
79. Đặc điểm của Cache trong 68XXX, so sánh với Intel X86.
80. Tìm hiểu cấu trúc các loại vi xử lý trong họ 68XXX, ghi nhớ những đặc điểm chính của chúng, tập lệnh.
81. Nếu thanh ghi D5 chứa giá trị 00F3D2CF và ta thực hiện lệnh CLR D5 WORD thì ta sẽ có kết quả là .....
    a) 00000000; b) 00F3D2CF;
    c) 00F30000; d) 00F3D200.
82. Các lệnh có thẩm quyền có thể được thực hiện nếu bộ xử lý 68XXX không ở trong trạng thái supervisor.
83. Nêu đặc điểm kiến trúc đường ống lệnh của họ 68XXX.
84. Tất cả họ 68XXX sử dụng.... input cho thiết bị ngoại vi nói với bộ xử lý rằng nó lấy dữ liệu từ data bus hoặc dữ liệu nó gửi lên bus đã có thực.
    a) DTACK#; b) BR#;
    c) BG#; d) VMA#.
85. Tín hiệu RESET# của 68XXX khác với X86 vì
    a) Nó lấy mức logic 1 để xóa bộ xử lý
    b) Nó lấy mức logic 0 để xóa bộ xử lý
    c) Đường RESET# có thể được nối với các thiết bị ngoại vi
    d) Đường RESET# là tín hiệu 2 chiều và có thể đưa ra hoặc đưa vào tín hiệu reset.
86. Các đường tín hiệu vào ngắt (IPL0, IPL1, và IPL2) được sử dụng để
    a) Chọn xóa, các ngắt che mặt nạ và không che mặt nạ
    b) Cấp cho bộ xử lý một mức ưu tiên ngắt
    c) Cung cấp các mức xử lý lệnh
    d) Tất cả trên

87. Họ Vi xử lý DEC Alpha có kiến trúc....
    a) CISC; b) RISC; c) Tất cả a, b
88. Đặc điểm kiến trúc đường ống ctrong DEC Alpha.
89. Các kiểu dữ liệu trong DEC Alpha AXP, so sánh với họ 68XXX và Intel X86.
90. Cấu trúc lệnh và tập lệnh của DEC Alpha AXP.
91. Các kiểu đánh địa chỉ của DEC Alpha AXP.
92. Nêu đặc điểm kiến trúc của họ vi xử lý PowerPC.
93. Họ PowerPC có kiến trúc ...
    a) CISC; b) RISC; c) Tất cả a, b
94. Nêu mô hình lập trình của PowerPC, so sánh chúng với Intel X86, 68XXX, DEC Alpha (nên lập bảng so sánh).
95. Nêu đặc điểm của khuôn dạng lệnh và tập lệnh của PowerPC.
96. Đặc điểm của MMU trong MPC601, so sánh với Intel X86, 68XXX, DEC Alpha (nên lập bảng so sánh).
97. Tổ chức Cache trong MPC601, so sánh với Intel X86.
98. Nêu đặc điểm kiến trúc của họ vi xử lý SUN SPARC.
99. Nêu đặc điểm chính của kỹ thuật đường ống lệnh trong SUN SPARC.
100. Nêu đặc điểm của Cache trong SUN SPARC, so sánh với Intel X86, 68XXX, DEC Alpha, PowerPC.
101. Nêu đặc điểm kiến trúc của họ vi xử lý HP PA-RISC.
102. Nêu đặc điểm về khuôn dạng lệnh và tập lệnh của họ PA-RISC.
103. Nêu định nghĩa về hệ phát triển.
104. Nêu chu trình phát triển máy vi tính.
105. Đặc điểm chính của hệ phát triển MDS.
106. Cấu trúc của hệ phát triển các hệ thống Pentium.
107. Hãy tìm hiểu thêm một số chủng loại hệ phát triển các hệ thống vi xử lý và đề xuất cấu trúc kết nối và qui trình sử dụng trong chu trình phát triển.

Chương 8
BỘ VI ĐIỀU KHIỂN

8.1 GIỚI THIỆU CHUNG VỀ VI ĐIỀU KHIỂN

8.1.1 Thế nào là vi điều khiển

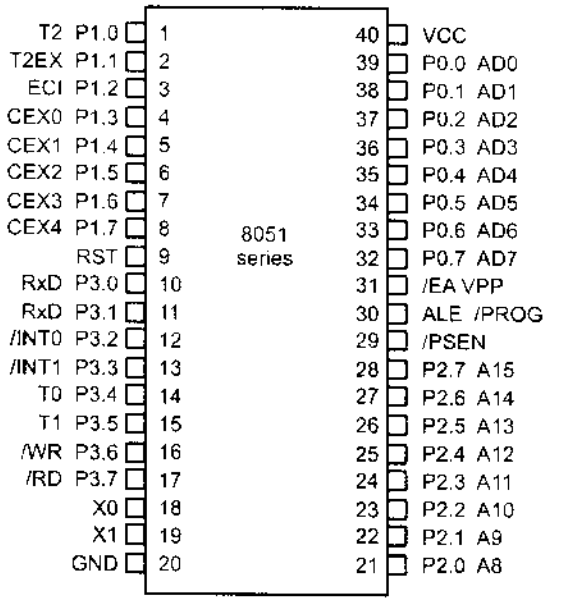
Vi điều khiển (microcontroller) là một hệ thống vi xử lý đầy đủ tích hợp trong một chip vi mạch, thường gọi là vi điều khiển đơn chip và chủ yếu sử dụng cho thực hiện các chức năng điều khiển. Các bộ vi điều khiển được phát triển để thỏa mãn nhu cầu thiết kế các hệ thống dùng vi xử lý với chi phí thấp. Bởi vì, nếu xây dựng một hệ thống có tính ứng dụng hẹp mà phải dùng tới các chip vi xử lý công suất lớn, thì sẽ là một sự lãng phí về thiết kế và đầu tư. Một hệ thống vi xử lý đầy đủ, phải có tối thiểu các đơn vị kết nối với nhau, đó là, đơn vị vi xử lý, RAM, ROM (hoặc EPROM), các cổng vào/ra song song, vào/ra tuần tự, các mạch đếm thời gian (timers), và nhịp đồng hồ. Như vậy để tích hợp hệ vi xử lý đầy đủ bên trong một đơn chip, cần phải có một công nghệ tích hợp cao, và công nghệ CMOS thường được sử dụng. Các vi điều khiển có kiến trúc Havard cho phép xử lý thông tin số nhanh. Để nghiên cứu vi điều khiển chúng ta khảo sát các loại vi điều khiển thông dụng của Intel.

8.1.2 Một số chip vi điều khiển của Intel

Intel có một loạt các vi điều khiển đơn chip thông dụng, đó là 8048, 8049, 8051, 8052, 8054, 8058, 8751, 8752, 8754, 8758, 8031, 8032, trong đó, loạt 8051 có đóng vỏ DIP 40 chân như mô tả ở hình 8.1.

Trong dạng đóng vỏ PLCC 44-chân (PLCC44), loạt vi điều khiển Intel 8051 (8051, 8052, 8054, 8058, 8751, 8752, 8754, 8758, 8031,

8032) có sắp xếp các chân tín hiệu như liệt kê trong bảng 8.1. Các chip 8x54 and 8x58 thêm một chân GND.



Hình 8.1: Đóng vỏ DIP các vi điều khiển Intel loạt 8051: 8051, 8052, 8054, 8058, 8751, 8752, 8754, 8758, 8031, 8032

Bảng 8.1: Sắp xếp chân của vi điều khiển 8051

7	P1.5 CEX2	18	P3.6 /WR	29	P2.5 A13	40	P0.3 AD3
8	P1.6 CEX3	19	P3.7 /RD	30	P2.6 A14	41	P0.2 AD2
9	P1.7 CEX4	20	X0	31	P2.7 A15	42	P0.1 AD1
10	RST	21	X1	32	/PSEN	43	P0.0 AD0
11	P3.0 RxD	22	GND	33	ALE /PROG	44	VCC
12		23		34		1	(GND)
13	P3.1 TxD	24	P2.0 A8	35	/EA VPP	2	P1.0 T2
14	P3.2 /INT0	25	P2.1 A9	36	P0.7 AD7	3	P1.1 T2EX
15	P3.3 /INT1	26	P2.2 A10	37	P0.6 AD6	4	P1.2 ECI
16	P3.4 T0	27	P2.3 A11	38	P0.5 AD5	5	P1.3 CEX0
17	P3.5 T1	28	P2.4 A12	39	P0.4 AD4	6	P1.4 CEX1

a) Các chip 8048, 8049, 8050

Thế hệ đầu tiên của công nghệ vi điều khiển 8-bit là chip vi điều khiển Intel 8048. Sau đó không lâu, một loại vi điều khiển nâng cấp của 8048 là chip 8049 có dung lượng bộ nhớ ROM và RAM gấp đôi 8049. Chip 8048 và 8049 có cấu trúc bên trong tương tự nhau và được dùng làm bộ điều khiển trung tâm trong bàn phím dùng cho các họ máy tính cá nhân hoặc tương đương IBM-PC/XT, PC/AT. 8048 có 1 kB ROM bên trong, hoặc 1 kB EPROM (phiên bản 8748). Còn 8049 có 2 kB ROM, và 8050 có 4 kB ROM, bên trong. 8048 có 64B RAM bên trong (kể cả 32 thanh ghi), 8049 có 128 byte RAM và 8050 có 256 byte RAM bên trong, trong đó cùng có 32 byte thanh ghi nhớ. Các chip 8048, 8049 được sử dụng để làm bộ vi xử lý của bàn phím ở PC/XT và AT.

b) Chip 8051

8051 có kiến trúc của chúng có sự khác nhau không nhiều so với kiến trúc của 8051. 8051 là thế hệ thứ 2 của các chip vi điều khiển. 8051 là hệ vi xử lý đầy đủ tích hợp trong một đơn chip thế hệ thứ của công nghệ vi điều khiển 8-bit, tiếp sau các vi điều khiển 8048, 8049, 8050. Chúng có kiến trúc bên trong tương tự nhau ngoại trừ dung lượng của bộ nhớ. Kiến trúc bên trong của chip 8051 có những đặc điểm chính: 8-bit ALU, 4k × 8 ROM (hoặc EPROM), 128k × 8 RAM, bộ thời gian 16 bit, 4 cổng vào/ra 8-bit, đánh địa chỉ không gian nhớ chứa chương trình 64 kB, đánh địa chỉ không gian nhớ chứa dữ liệu 64 kB, tập lệnh gồm 111 lệnh, cổng vào/ra tuần tự đa năng, nhịp đồng hồ đạt tới 12 MHz, và 2 ngắt bên ngoài. Chip Intel 8051 đóng vỏ DIP 40 chân. Các kết nối với thiết bị ngoài rất đơn giản. Đó là 32 chân tín hiệu của 4 cổng vào/ra 8-bit 2 chiều (Port 0 - Port 3 drivers). Tám chân tín hiệu còn lại dành cho định thời và điều khiển: PSEN, ALE, EA, RST, kết nối thạch anh tạo nhịp đồng hồ: XTAL1, XTAL2, và kết nối nguồn: Vcc = 5 V, Vss = GND.

c) Chip 8052

Sau 8051 là 8052 với kiến trúc bên trong tự nhưng dung lượng RAM và ROM được mở rộng hơn. 8052 là phiên bản mở rộng của 8051. Bởi vì nó chỉ có 2 sự khác biệt với 8051. Trước hết, 8052 có bảng mạch 8 kB ROM và bảng mạch 256 kB RAM (onboard). Như vậy, nó cho phép người lập trình viết các chương trình lớn hơn và sử dụng nhiều dữ liệu thay đổi hơn. Như thế giá thành của 8052 cao hơn 8051. Thứ hai, 8052 có bộ đếm-thời gian phụ 16-bit TM2 cho phép người lập trình linh hoạt sử dụng các bộ đếm-thời gian, đặc biệt trong điều khiển tốc độ truyền thông (baud) cho cổng tuần tự. Bởi vì cả 8051 và 8052 đều có sẵn bảng mạch ROM, nên chúng cần được nhà sản xuất lập trình trước bằng các thiết bị riêng. Như vậy, 8051 và 8052 chỉ thường được sử dụng với số lượng lớn để giảm chi phí cho chế tạo. Trong những ứng dụng số lượng nhỏ, có hai phiên bản đặc biệt được dùng là 8751 và 8752. Chúng thay thế bảng mạch ROM bằng bảng mạch EPROM. Sử dụng EPROM, người sử dụng có thể xoá chương trình bằng nguồn cực tím và lập trình lại bằng các tín hiệu điện bên ngoài. Phiên bản thứ 3 của 8052 là 8052 AH-BASIC. Đây là chip 8052 có ngôn ngữ lập trình BASIC bên trong ROM. Sử dụng các lệnh của BASIC, người lập trình có thể viết các lệnh cho 8052 thay vì phải viết bằng hợp ngữ (assembly).

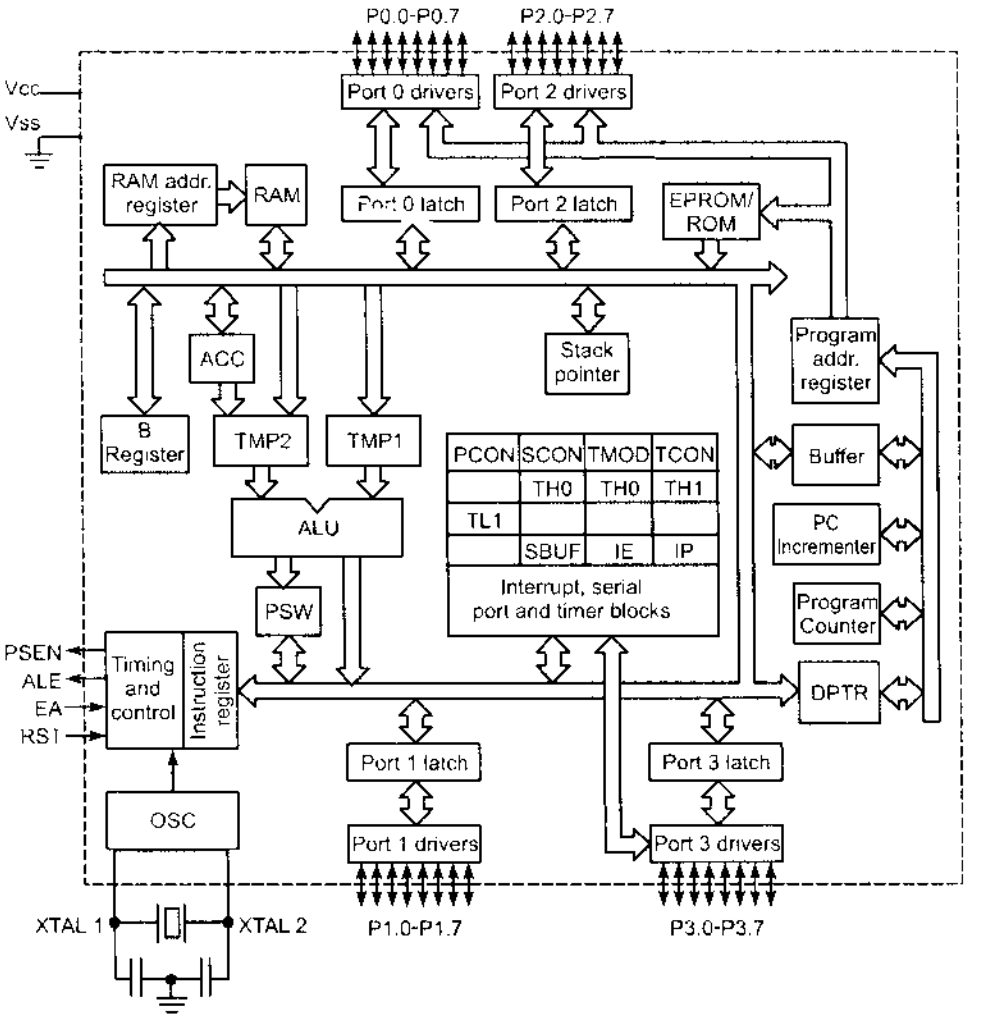
d) Các chip 8031, 8032

Phiên bản khác của 8051 và 8052 là các chip vi xử lý 8031 và 8032. Những chip này không có bất kỳ bảng mạch ROM nào, vì bảng mạch ROM có thể bị hỏng, và để tạo bộ nhớ chương trình ta cần phải sử dụng ROM (EPROM) bên ngoài. Một chip 8031 cộng với EPROM 2764 8 k có chức năng tương tự như một hệ thống 8751. Nhưng một cấu trúc kết hợp 8031 và một EPROM 2764 8k rẻ hơn nhiều so với 8751, mà lại có nhiều gấp đôi số cổng vào/ra và không gian cho bảng mạch in.

8.2 CẤU TRÚC VÀ HOẠT ĐỘNG CỦA BỘ VI ĐIỀU KHIỂN

Chúng ta xét cấu trúc và hoạt động của bộ vi xử lý qua các bộ vi điều khiển thông dụng của Intel 8051, 8052.

8.2.1 Cấu trúc của Intel 8051



Hình 8.2: Sơ đồ khối cấu trúc bên trong của Intel vi điều khiển 8051

Nếu chúng ta so sánh với Z80 thì 8051 hơn hẳn Z80 về mức độ tích hợp với đủ các chức năng của một hệ vi xử lý. Đặc điểm này làm

cho kết nối với các mạch và thiết bị bên ngoài đơn giản hơn và đòi hỏi ít linh kiện hơn. Suy rộng ra, nó làm cho sản phẩm hoàn chỉnh gọn nhỏ hơn về kích thước mà vẫn đảm bảo tính vạn năng của ứng dụng. Sơ đồ khối của 8051 cho ở hình 8.2.

Cấu trúc bên trong của 8051 gồm có:

- Đơn vị xử lý trung tâm của 8051 gồm ALU, thanh tổng (ACC), con trỏ ngăn xếp (stack pointer), khối các thanh ghi, và thanh ghi chung - thanh ghi B. Tất cả các đơn vị này kết nối bus dữ liệu 8-bit bên trong để trao đổi dữ liệu.

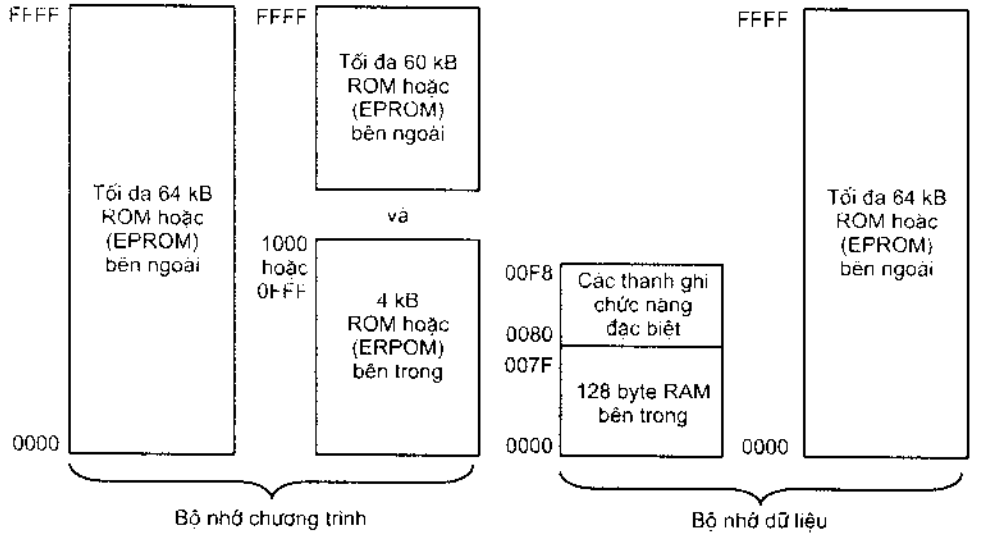
- Mỗi một cổng vào/ra kết nối với bus dữ liệu bên trong thông qua các thanh ghi chốt (latch) và điều khiển (driver). Những thanh ghi này lưu giữ dữ liệu trong quá vận chuyển và điều khiển các cổng vào/ra.

- 8051 có 4 kB ROM và 128 byte RAM, trong đó kể các các thanh ghi chức năng đặc biệt. Để mở rộng dung lượng ROM và RAM cần phải kết nối thêm bên ngoài chip 8051. Điều đáng chú ý là 8051 đánh địa chỉ riêng biệt 2 không gian nhớ. Vì sao như vậy? 8051 sử dụng một không gian nhớ để chứa các chương trình và không gian nhớ khác chứa dữ liệu thay đổi.

- Không gian nhớ để chứa chương trình là vùng chỉ được đọc, và bộ vi xử lý chỉ có thể đọc chương trình ra từ vùng nhớ này, mà không thể đọc hoặc ghi dữ liệu. Bộ nhớ ROM trong 8051 thuộc về không gian nhớ này.

- Không gian nhớ để chứa dữ liệu là vùng nhớ mà bộ vi xử lý có thể đọc-ghi dữ liệu được. Một chương trình có thể được ghi vào vùng nhớ này để thực hiện. Bộ nhớ RAM bên trong 8051 thuộc về vùng nhớ này. 128 byte RAM bên trong 8051 chiếm khoảng địa chỉ từ 0000h đến 007Fh đảm bảo cho các truy nhập ghi-đọc bộ nhớ chung.

Một phần của 128 byte RAM này là các thanh ghi dùng chung (R0-R7) không thể hiện trong sơ đồ khối (hình 8.2). Sự sắp xếp hệ thống nhớ của 8051 cho trong hình 8.3.



Hình 8.3: Sắp xếp bộ nhớ của hệ thống 8051

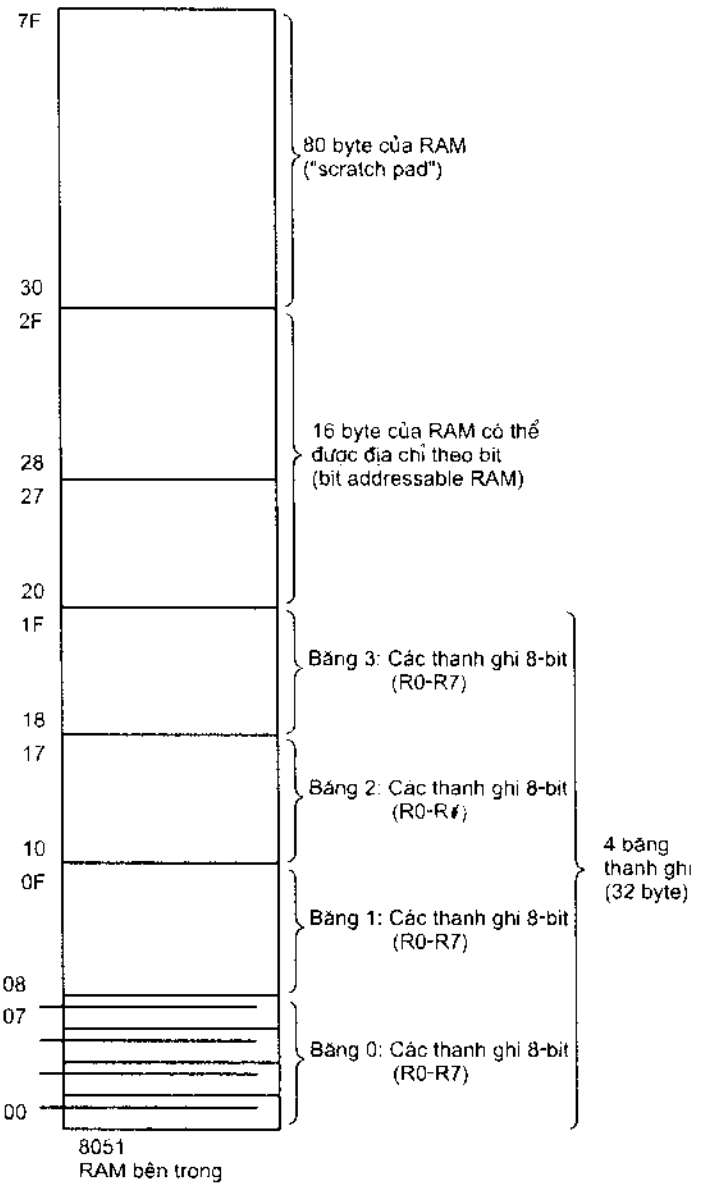
8051 có 22 thanh ghi chức năng đặc biệt mà không phải là một phần của 128 byte của RAM bên trong. Các thanh ghi chức năng đặc biệt này chiếm một không gian nhớ từ địa chỉ 0080H đến 00F8H. Nếu cần bổ sung thêm RAM để chứa dữ liệu, ta có thể sử dụng đầy đủ. 64 kB RAM bên ngoài và chúng được đánh địa chỉ riêng biệt từ 0000h đến FFFFh.

Nếu cần phải có thêm dung lượng nhớ ROM (hay EPROM) để chứa chương trình thì 4 kB nhớ bên trong chứa chương trình có thể mở rộng thêm 60 kB (từ địa chỉ 1000h đến FFFFh), đảm bảo tận dụng đánh địa chỉ tối đa 64 kB nhớ của 8081. Nếu chân tín hiệu EA# nối đất (GND) thì 4 kB ROM (hoặc EPROM) bên trong 8051 không được dùng. Như vậy bộ nhớ ngoài sẽ được đánh địa chỉ từ 0000h đến FFFFh (đầy đủ 64 kB nhớ ngoài).

Với kiểu tổ chức bộ nhớ này, 8051 có thể nhận một khối dữ liệu 64 kB thông qua cổng truyền thông tuần tự vào bộ nhớ dữ liệu, và sau đó thực hiện khối dữ liệu như là một chương trình. Đây gọi là chương trình nạp (download program). Kỹ thuật này thường được sử dụng để thay đổi một chương trình vận hành ở một hệ thống điều khiển xa.

Đối với một hệ thống điều khiển, chương trình thường nhỏ, gọn và phải làm sao vừa đủ chứa bên trong ROM.

8.2.2 Sắp xếp RAM bên trong 8051



Hình 8.4: Sắp xếp RAM bên trong 8051

8051 có 128 byte RAM, trong đó kể các các thanh ghi chức năng đặc biệt. 128 byte RAM chiếm khoảng địa chỉ từ 00h đến 7Fh đảm bảo cho các truy nhập ghi-đọc bộ nhớ chung. Sự sắp xếp RAM trong 8051 cho trong hình 8.4.

32 byte đầu tiên của RAM được sắp xếp theo 4 băng thanh ghi. Mỗi băng có 8 thanh ghi 8-bit, được đánh địa chỉ như R0-R7. Các thanh ghi này được truy nhập để ghi/đọc dữ liệu bằng các lệnh xử lý theo byte. Một số lệnh xử lý theo bit cũng có thể thao tác với các bit của các thanh ghi này. Các lệnh xử lý theo bit đặc biệt mạnh khi chúng được sử dụng để xử lý các từ dữ liệu đọc từ các thiết bị ngoài trong quá trình điều khiển. Các lệnh vào/ra của 8051 có thể thực hiện vận chuyển dữ liệu theo byte và theo từng bit.

Vì sao lại tổ chức theo các băng nhớ? Bởi vì, thứ nhất, tất cả các thao tác vận chuyển ghi/đọc dữ liệu với các thanh ghi R0-R7 được thực hiện theo từng băng và nếu muốn sử dụng thêm các thanh ghi của băng khác thì cần phải chuyển băng. Vả lại, 8051 có con trỏ ngăn xếp SP được khởi tạo ban đầu giá trị 08h trỏ tới R0 của băng 1 để cất giữ byte dữ liệu đầu tiên. Sau đó, SP được tăng lên, nghĩa là, byte dữ liệu bổ sung sẽ cất giữ vào R1,... Như vậy, ngăn xếp bắt đầu từ băng 1. Thứ 2, mỗi một băng chính là một vùng nhớ 32-bit, mà một số kiểu đánh địa chỉ yêu cầu sử dụng những vùng nhứ 32-bit này. Tiếp sau 4 băng nhớ gồm các thanh ghi là vùng nhớ 16 byte từ địa chỉ 20h đến 2Fh. 16 byte nhớ này được gọi là vùng RAM có thể được đánh địa chỉ theo bit. Tức là, có thể đánh địa chỉ bất kỳ ngăn nhớ nào theo byte hoặc theo bit. Ví dụ, các lệnh của 8051 cho phép người lập trình thực hiện phép bù bất kỳ bit nào đó, AND một bit với bit khác trong thanh tổng ACC hoặc thanh ghi trạng thái, hoặc thực hiện các thao tác đơn giản khác. 80 byte còn lại từ địa chỉ 30h đến 7Fh là vùng nhớ được có thể tuỳ ý ghi/đọc/xóa (scratch pad). Người lập trình có thể ghi nhớ các biến nếu cần thiết. Nhưng cần chú ý rằng, khi khởi tạo SP để sử dụng một phần của vùng nhớ này, không được ghi đè lên dữ liệu đã lưu trữ, và ngược lại, khi ghi dữ liệu vào vùng này, không được ghi lên phần dành cho ngăn xếp.

8.2.3 Các thanh ghi chức năng đặc biệt bên trong 8051

Hình 8.5 là sự sắp xếp các thanh ghi chức năng đặc biệt bên trong 8051. Mỗi một thanh ghi chức năng được ấn định một nhiệm vụ riêng. Một số nhỏ chúng là một phần của đơn vị vi xử lý. Còn lại đảm nhiệm các chức năng vào/ra.

a) Các thanh ghi thuộc đơn vị vi xử lý

Các thanh ghi chức năng đặc biệt thuộc đơn vị vi xử lý là: thanh tổng ACC địa chỉ E0h, thanh ghi B địa chỉ F0h, từ trạng thái của chương trình PSW địa chỉ D0h, con trỏ ngăn xếp SP địa chỉ 81h, con trỏ dữ liệu DPTR 16-bit với 2 byte: DPL địa chỉ 82h và DPH địa chỉ 83h, và thanh ghi điều khiển công suất PCON địa chỉ 87 (power control register). Một số trong các thanh ghi này có thể được đánh địa chỉ cả theo bit và theo byte, một số khác chỉ có thể đánh địa chỉ theo byte.

(1) Thanh tổng ACC

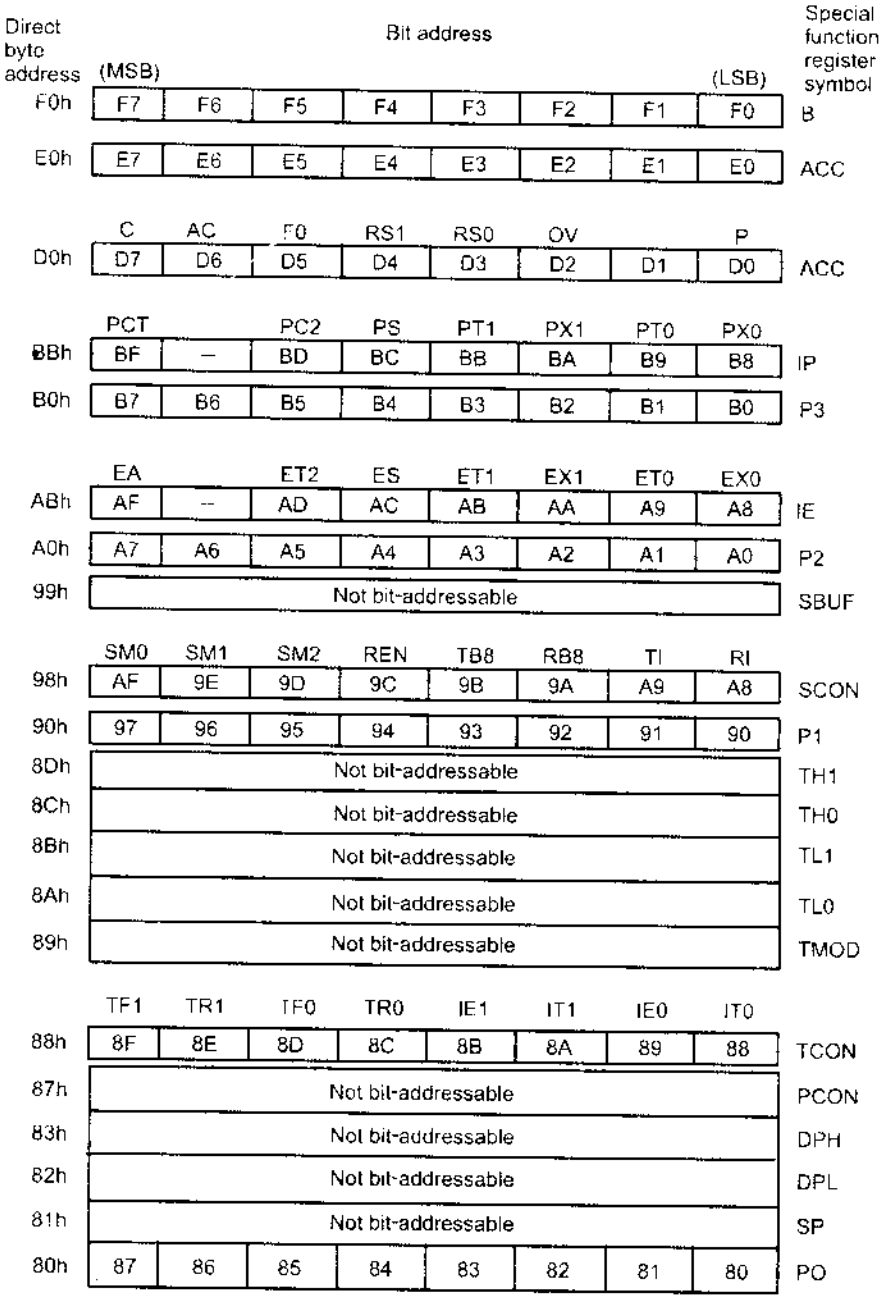
ACC có thể được đánh địa chỉ theo bit và theo byte. Nó chứa toán hạng (operand) hoặc một trong các toán hạng của lệnh số học và logic và chứa kết quả của lệnh thực hiện. Hầu hết các lệnh có tham chiếu đến ACC bằng chữ A.

(2) Thanh ghi B

Thanh ghi B được sử dụng trong các lệnh nhân và chia như là nơi chứa toán hạng nguồn và để chứa 8 bit cao. Nếu không sử dụng B cho phép nhân và chia, thì nó có thể được dùng như một thanh ghi ghi/đọc tuỳ ý cho bất cứ một lệnh nào đọc hoặc ghi dữ liệu vào ngăn nhớ địa chỉ F0h.

(3) Con trỏ dữ liệu (DPTR)

DPTR là thanh ghi 16-bit, gồm 2 byte: byte cao DPH địa chỉ 83h và byte thấp DPL địa chỉ 82h. DPTR chứa một địa chỉ 16-bit để đánh địa chỉ vùng nhớ bên ngoài chip 8051. Các lệnh của 8051 cho phép sử dụng DPTR 16-bit hoặc sử dụng trong byte của DPTR. Để nạp giá trị 16-bit cho DPTR cần phải có lệnh dài 3 byte.



Hình 8.5: Các thanh ghi chức năng đặc biệt trong 8051

(4) Từ trạng thái chương trình (PSW)

PSW có 8 bit (hình 8.6), nhưng không dùng hết. Các bit CY, AC, OV, P có ý nghĩa tương tự như trong thanh ghi cờ của Z80. Các bit khác là đặc thù của 8051, đó là bit F0 (Flag 0) dành cho người sử dụng, các bit RS1 và RS0 để chỉ thị chọn băng nhớ (4 băng nhớ RAM). RS1 và RS0 có thể được thiết lập để chọn băng nhớ hoặc được đọc để bết được băng nhớ nào đang được chọn.

CY	AC	F0	RS1	RS0	OV	-	P

CY: PSW.7: Carry (nhớ)
AC: PSW.6: Auxilarry carry (nhớ phụ)
F0: PSW.5: Flag 0 có cho người dùng như một chỉ thị (indicator) cho mục đích chung
RS1 PSW.4: Bit 1 cho chọn băng nhớ
RS0: PSW.3: Bit 0 cho chọn băng nhớ
OV: PSW.2: Tràn
-: PSW.1: Dự phòng
P: PSW.0: Chẵn lẻ (được thiết lập nhờ phần cứng ở từng chu kỳ lệnh để chỉ thị một số Lẻ/chẵn các bit có giá trị "1" trong ACC

Hình 8.6: Từ trạng thái chương trình của 8051

(5) Thanh ghi điều khiển nguồn PCON

PCON (Power CONtrol register) là thanh ghi 8-bit (hình 8.7). Hai bit PD (Power Down), và IDL (idle) chỉ được sử dụng cho phiên bản công nghệ CMOS của 8051, đó là 80C51. Những bit này đặt chip 80C51 vào các chế độ đặc biệt duy trì nguồn cho đến khi đưa vi xử lý vào hoạt động. Các bit GF1 và GF0 có thể được người lập trình sử dụng để cất giữ bit thông tin. Bit SMOD đảm bảo chức năng đặc biệt điều khiển giao tiếp tuần tự. Chúng ta sẽ xét bit này trong phần nói về giao tiếp tuần tự.

	-		-	GF1	GF0	PD	IDL

SMOD: Bit tốc độ baud đôi (double baud rate bit)
GF1: Bit dùng cho mục đích chung
GF0: Bit dùng cho mục đích chung
PD: Bit mất nguồn (power down bit)
IDL: Bit chế độ cấm (Idle mode bit) Thiết lập bit này kích hoạt kiểm soát mất nguồn trong Chip 80C51

Hình 8.7: Thanh ghi PCON

(6) Con trỏ ngăn xếp SP

Con trỏ ngăn xếp là thanh ghi 8-bit. Nội dung của nó được tăng lên trước khi dữ liệu được đẩy vào ngăn xếp nhờ các lệnh gọi tới chương trình con (ACALL, LCALL), hay khi có ngắt. Cần phải phân biệt sự khác nhau giữa SP của 8051 và SP trong các hệ vi xử lý 8-bit nói chung (8080, Z80). Trước hết, SP của 8051 là 8-bit, vì nó chỉ địa chỉ vùng nhớ bên trong RAM dung lượng 128 byte, nên ngăn xếp của 8051 nhỏ. Trong khi đó SP của Z80 là 16-bit. Thứ 2 là, nội dung của SP của 8051 đa phần được tăng hơn là giảm. SP của 8051 được khởi tạo giá trị ban đầu bằng 07h khi có xóa (reset). Điều này có nghĩa là, dữ liệu đầu tiên được đẩy vào ngăn xếp ở địa chỉ 08h. Nếu người lập trình không muốn vùng ngăn xếp bắt đầu từ địa chỉ 08h, thì cần phải khởi tạo lại giá trị cho SP trước khi sử dụng ngăn xếp.

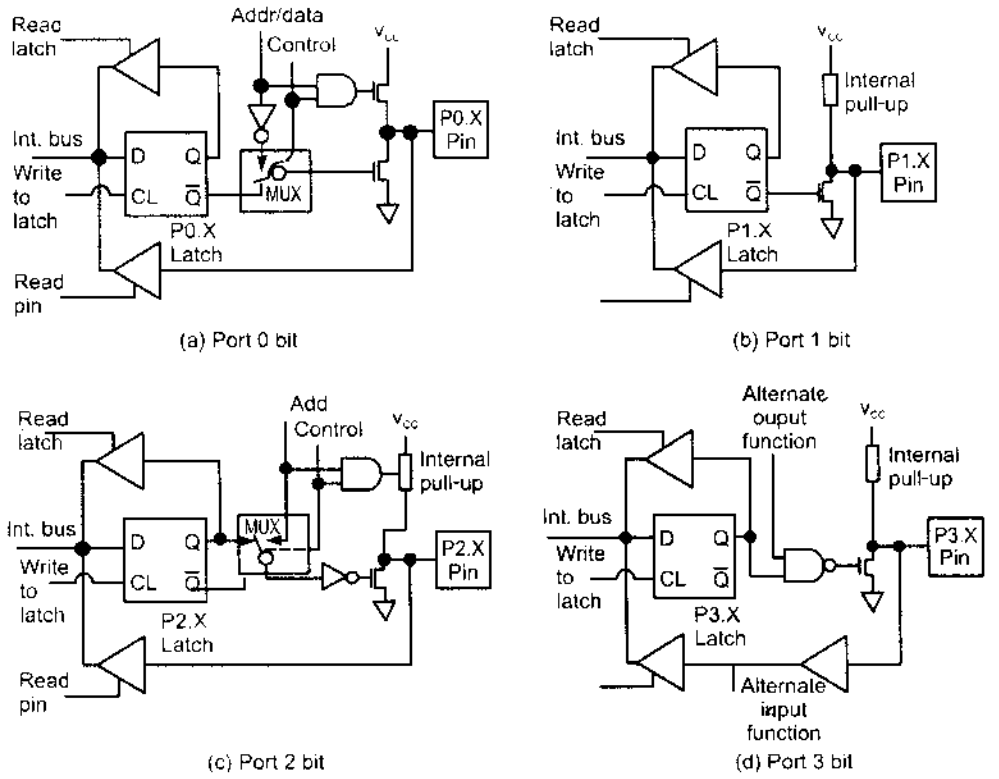
b) Các thanh ghi đặc biệt khác

Đó là các thanh ghi được sử dụng để điều khiển các cổng vào/ra, thu/phát dữ liệu tuần tự, các bộ đếm thời gian và hệ thống xử lý ngắt.

8.2.4 Các loại cổng vào/ra của 8051

Trong sơ đồ khối kiến trúc bên trong của 8051 có thể hiện 4 cổng vào/ra sóng song 8-bit song hướng. Nhưng những cổng này phức tạp về chức năng hơn so với vào/ra song song thông thường. Mỗi một cổng vào/ra có những đặc tính riêng biệt, trong đó, có 3 cổng có thể được sử dụng trong một số cách khác nhau. Hình 8.8 cho ta sơ đồ logic của từng bit trong một cổng vào/ra. Qua đó có thể thấy được thiết kế mạch điện và logic của 4 cổng vào/ra song song (P0-P3). Chốt cổng (port latch) cho phép cất giữ dữ liệu đi ra hoặc nhập về cổng. Chốt có thể được thiết lập nhờ dữ liệu ở trên bus dữ liệu của đơn vị vi xử lý hoặc ở các chân (pin) của cổng. Cũng có thể, chốt có thể đặt dữ liệu lên bus dữ liệu của đơn vị vi xử lý hoặc gửi dữ liệu đến chân của cổng.

Các bit trên cổng P0 và cổng P2 có các mạch điều khiển kéo-đẩy (pull_up) (hình 8.8a, c). Những mạch này được tích cực hay ngắt nối phụ thuộc vào chế độ của cổng.



Hình 8.8: Sơ đồ logic của từng bit trong một cổng vào/ra của 8051

Nếu bộ nhớ ngoài (hoặc chứa chương trình hoặc chứa dữ liệu) được nối với 8051, thì cổng P0 và cổng P2 địa chỉ bộ nhớ này. Cổng P0 cũng dùng để trao đổi dữ liệu với bộ nhớ ngoài, do đó nó có 2 chức năng. Trước hết nó phải đưa ra byte thấp của địa chỉ ngăn nhớ 16-bit, sau đó, nó phải hoặc nhận dữ liệu từ ngăn nhớ hoặc đưa ra dữ liệu cho ngăn nhớ được địa chỉ. Vậy làm thế nào đảm bảo thực hiện 2 chức năng cho một cổng. Ta biết rằng, vi xử lý thực hiện địa chỉ một vùng nhớ trong phần đầu tiên của chu kỳ lệnh và trao đổi dữ liệu với vùng nhớ trong phần thứ 2 của chu kỳ lệnh. Bởi vì sự đánh địa chỉ và trao đổi dữ liệu xảy ra ở thời gian khác nhau, nên có thể dùng chung bus địa chỉ - dữ liệu. Trước hết 16-bit địa chỉ đưa lên bus địa chỉ, và 8 bit thấp của địa chỉ được chốt ở bộ nhớ bên ngoài. Trong nửa sau của chu kỳ lệnh, 8 bit thấp lại là 8 bit dữ liệu trên bus dữ liệu.

Chip 8051 đưa ra tín hiệu ALE (chốt địa chỉ) chỉ cho mạch chốt địa chỉ bên ngoài biết rằng đã có giá trị dữ liệu ở 8 bit thấp của bus địa chỉ. Một khi 8 bit thấp của địa chỉ đã được chốt ở bộ nhớ để giải chọn ngăn nhớ, thì 8-bit thấp của bus địa chỉ chuyển thành 8-bit bus dữ liệu. Vì dữ liệu ở cổng P2 không thay đổi nên không cần thiết phải chốt 8 bit cao của địa chỉ. Kỹ thuật chung bus dự liệu - địa chỉ có ưu việt là làm giảm số chân tín hiệu phải dành riêng địa chỉ và dữ liệu.

Không giống như cổng P0 và P2, cổng P3 vừa có thể sử dụng như cổng vào/ra 8-bit song song hướng với chức năng bình thường, vừa có thể được dùng cho chức năng khác như vào/ra tuần tự với các ấn định chức năng cho từng bit như cho trong bảng 8.2.

Bảng 8.2: Chức năng các bit của cổng P3

Pin N	Chức năng bình thường	Chức năng khác
P3.0	BIT 0	RXD (dữ liệu nhận tuần tự)
P3.1	BIT 1	TXD (dữ liệu phát tuần tự)
P3.2	BIT 2	INT0# (ngắt 0 từ ngoài)
P3.3	BIT 3	INT1# (ngắt 1 từ ngoài)
P3.4	BIT 4	T0 (đầu vào 0 cho bộ đếm-thời gian 16-bit)
P3.5	BIT 5	T1 (đầu vào 1 cho bộ đếm-thời gian 16-bit)
P3.6	BIT 6	WR# (xung ghi dữ liệu ra bộ nhớ ngoài)
P3.7	BIT 7	RD# (xung đọc dữ liệu từ bộ nhớ ngoài)

Hai chân tín hiệu P3.6 và P3.7 của cổng P3 cần phải được sử dụng để làm việc cùng với các cổng P0 và P2 khi 8051 kết nối với bộ nhớ bên ngoài. Tại chân P3.6 tín hiệu xung ghi WR# thực hiện ghi dữ liệu ra bộ nhớ ngoài và tại chân P3.7 tín hiệu xung RD# thực hiện đọc dữ liệu từ bộ nhớ ngoài. Các tín hiệu không có ý nghĩa đối với bộ nhớ bên trong chip 8051. Tín hiệu điều khiển PSEN# dùng để đọc bộ nhớ chương trình bên ngoài. Dữ liệu tuần tự được đọc qua chân P3.0 (RXD) và được phát đi qua chân P3.1 (TXD). Các chân vào INT0# và INT1# là các tín hiệu ngắt từ mạch ngoài. Các chân P3.4 và P3.5 là các tín hiệu vào T0 và T1 cho các bộ đếm-thời gian 16-bit, cổng P1 ở mỗi

bit có một mạch điều khiển kéo-đẩy cố định bên trong (internal pull-up) (hình 8.8b). Chúng đảm bảo vận chuyển dữ liệu song song 8-bit.

Bởi vì cổng P0 và cổng P2 có các mạch điều khiển đẩy kéo, nên cần phải ghi "1" vào từng bit trong từng cổng trước khi bit đó được sử dụng như một đầu vào. Sự ghi giá trị logic "1" vào cổng ra làm cho cổng ra đó bị đẩy lên giá trị logic "1". Một thiết bị ngoài có thể sau đó đẩy cổng tới logic "0" hoặc để ở "1". Khi đọc từ cổng đó, dữ liệu tạo bởi tín hiệu sẽ được đọc.

Có 2 kiểu lệnh khác nhau để đọc dữ liệu từ cổng vào/ra. Một số lệnh đọc dữ liệu từ các chốt, một số khác đọc dữ liệu trực tiếp từ chân tín hiệu (pin). Vậy cái gì là sự khác nhau giữa chúng?

Các lệnh đọc dữ liệu từ chốt có thể đọc một giá trị, thay đổi giá trị đọc nếu cần thiết, và ghi dữ liệu trở lại chốt đó. Đó là các lệnh đọc-thay đổi-ghi (read-modify-write instructions) (bảng 8.3).

Bảng 8.3: Lệnh đọc thay đổi ghi

Lệnh	Thực hiện
ANL	Logical AND. Ví dụ: ANL P1,A
ORL	Logical OR. Ví dụ: ORL P2,A
XRL	Logical EX-OR. Ví dụ: XRL P3,A
JBC	Nhảy (Jump) nếu bit = 1 và xóa bit. Ví dụ: JBC P1.1, LABEL
CPL	Bù bit. Ví dụ: CPL P3.0
INC	Tăng (increment). Ví dụ: INC P2
DEC	Giảm (decrement). Ví dụ: DEC P2
DJNZ	Giảm và nhảy nếu không phải zero. Ví dụ: DJNZ P3, LABEL
MOV PX.Y,C	Chuyển bit carry (CY) đến bit Y của cổng X
CLR PX.Y	Xóa bit Y của cổng X
SET PX.Y	Thiết lập bit Y của cổng X

Vì sao lại cần cả 2 loại lệnh như thế này?

Các lệnh đọc-thay đổi-ghi dữ liệu ở chốt tránh được lỗi đọc do mức điện thế ở các chân. Ví dụ, nếu ta kết nối bit 1 của cổng với bazơ của transistor, thì transistor sẽ chuyển vào trạng thái bật khi ta ghi giá trị logic "1" vào bit 1. Mặc dù vậy, nếu đọc ngay tại chân (pin), thì sẽ đọc một điện thế thấp do liên kết bazơ-emitơ có điện thế 0,7 V.

Một trong những đặc tính ưu việt của 8051 là có giao tiếp tuần tự. Trong khi đó, hầu hết các loại vi xử lý 8-bit cần phải có thêm chip UART riêng, làm tăng chi phí và kích thước của thiết kế.

8.2.5 Giao tiếp truyền thông tuần tự của 8051

Giao tiếp tuần tự của 8051 là cổng truyền thông tuần tự song công đầy đủ (full_duplex transmission). Nghĩa là, cùng một thời gian có thể thu và phát dữ liệu. Giao tiếp tuần tự có bộ đệm nhận dữ liệu, do đó, có thể nhận một từ dữ liệu thứ 2 trước khi từ dữ liệu thứ nhất được chuyển vào trong vi xử lý. Giống như bất kỳ một giao tiếp truyền thông tuần tự, byte dữ liệu đầu tiên phải được đọc vào vi xử lý trước khi byte dữ liệu thứ hai nhận xong. Nếu không dữ liệu sẽ bị mất.

Giao tiếp tuần tự sử dụng thanh ghi thu (receive register) và thanh ghi phát (transmit register) dữ liệu. Cả hai thanh ghi này được truy nhập thông qua đánh địa chỉ thanh ghi đặc biệt SBUF. Nếu ghi dữ liệu vào SBUF thì đó là nạp dữ liệu cho thanh ghi phát. Nếu đọc dữ liệu từ SBUF, thì đó là đọc dữ liệu từ thanh ghi nhận.

Điều khiển giao tiếp tuần tự được thực hiện thông qua thanh ghi điều khiển cổng tuần tự SCON (hình 8.9). Các bit SM0 và SM1 để thiết lập 4 chế độ của cổng tuần tự (mode 0, mode 1, mode 2, và mode 3).

Trong chế độ 0, cổng tuần tự làm việc như với thanh ghi dịch 8-bit. Nhịp cho cổng được cố định bằng 1/12 tần số của nhịp đồng hồ của 8081.

Trong chế độ 1, cổng tuần tự tiếp nhận và phát các ký tự 10-bit, trong đó có 1 bit Start, 8 bit dữ liệu và 1 bit Stop. Tốc độ truyền thông (baud) có thể được lập trình nhờ 1 trong 2 bộ đếm-thời gian của 8051.

Trong chế độ 2, cổng tuần tự thu và phát các ký tự 11-bit, gồm 8 bit dữ liệu, bit thứ 9 lập trình, 1 bit Start, và 1 bit Stop. Bit thứ 9 được lấy từ bit TB8 của thanh ghi SCON. Trong chế độ này, chỉ có thể chọn tốc độ baud hoặc bằng 1/32 hoặc bằng 1/64 của tần số nhịp đồng hồ của 8051.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0: SCON.7: Định chế độ cho cổng tuần tự

SM1: SCON.6: Định chế độ cho cổng tuần tự

SM2: SCON.5: Cho phép kết nối vi xử lý ở chế độ 2 và 3. Trong chế độ 2 hoặc 3, nếu SM2 = 1 thì RI sẽ không tích cực nếu bit RB8 = 0. trong chế độ 1, nếu SM2 = 1 thì RI sẽ không tích cực nếu stop bit không nhận được. Trong chế độ 1, SM2 = 0.

REN: SCON.4: Được thiết lập/xóa nhờ lệnh để cho phép/cấm nhận

TB8: SCON.3: Bit này được thiết lập/xóa nhờ lện và được phát đi ở chế độ 2 và 3.

RB8: SCON2: Trong chế độ 2 và 3 nếu bit dữ liệu thứ 8 được nhận. Trong chế độ 1, nếu SM2 = 0, thì RB8 là bit Stop nhận được. Trong chế độ 0, TB8 không được sử dụng.

TI: SCON.1: Cờ ngắt khi phát. TI được thiết lập nhờ phần cứng ở thời gian kết thúc của bit thứ 8 trong chế độ 0, hoặc khi bắt đầu cua bit Stop trong các chế độ khác. TI được xoá nhờ lệnh.

RI: SCON0: Cờ ngắt khi nhận. RI được thiết lập nhờ phần cứng ở thời gian kết thúc của bit thứ 8 trong chế độ 0, hoặc ở một nửa thời gian của bit Stop trong các chế độ khác (trừ SM2). RI được xóa nhờ lệnh.

Các chế độ (Mode):

SM0	SM1	Mode	Mô tả	Tốc độ (baud)
0	0	0	Thanh ghi dịch	$F_{osc}/12$
0	1	1	8-bit UART	Giá trị biến
1	0	2	9-bit UART	$F_{osc}/64$ hoặc $F_{osc}/32$
1	1	3	9-bit UART	Giá trị biến

Hình 8.9: Thanh ghi SCON

Chế độ 3 cũng đảm bảo thu phát ký tự 11-bit, nhưng bit thứ 9 lập trình lấy từ bit RB8 của thanh ghi SCON. Khác với chế độ 2, chế độ 3 cho phép tốc độ baud thay đổi bằng lập trình.

Một cách để sử dụng giao tiếp tuần tự là kết nối truyền thông giữa các bộ vi xử lý (multiprocessor communication). Nó cho phép 2 hoặc hơn các chip 8051 kết nối với nhau ở tốc độ cao. Khi đó, bit dữ liệu thứ 9 được sử dụng để tạo một ngắt một chip 8051 nhận dữ liệu.

Trong 8051, tốc độ baud trong các chế độ 1 và 3 được thiết lập tốc độ tràn (overflow rate) nhờ bộ thời gian 1 (timer 1). Ta có thể nhân gấp đôi tốc độ baud bằng cách thiết lập bit SMOD trong thanh ghi PCON. Bảng 8.4 liệt kê các tốc độ baud khác nhau có thể thiết lập cho cổng truyền thông tuần tự của 8051. Qua đó, nhận thấy rằng, tốc độ baud phụ thuộc các giá trị thiết lập cho bộ thời gian 1, chế độ của bộ thời gian, bit SMOD được thiết lập và tần số nhịp đồng hồ của 8051. Thường các chip vi xử lý sử dụng tần số lẻ, ví dụ: 11,059 MHz, để tạo ra tốc độ baud chuẩn.

Bảng 8.4 Các tốc độ baud thiết lập cho cổng truyền thông tuần tự của 8051

Tốc độ baud	F_{osc}	SMOD	Timer 1		
			C/T#	Mode	Giá trị nạp lại
Mode 0 MAX: 1 MHz	12 MHz	X	X	X	X
Mode 2 MAX: 375 k	12 MHz	1	X	X	X
Mode 1,3: 62,5 k	11,059 MHz	1	0	2	FFh
19,2 k	11,059 MHz	1	0	2	FDh
9,6 k	11,059 MHz	0	0	2	FDh
4,8 k	11,059 MHz	0	0	2	FAh
2,4 k	11,059 MHz	0	0	2	F4h
1,2 k	11,059 MHz	0	0	2	F8h
137,5	11,059 MHz	0	0	2	1Dh
110	6 MHz	0	0	2	72h
110	12 MHz	0	0	1	FEBh

8.2.6 Các bộ đếm-thời gian trong 8051

8051 có 2 thanh ghi đếm-thời gian 16-bit bên trong. Chúng được gọi là bộ thời gian 0 (timer 0) và 1 (timer 1). Cả 2 bộ thời gian này có thể làm việc như các bộ thời gian hay như bộ đếm sự kiện. Khi sử dụng một bộ thời gian trong chế độ thời gian, thì thanh ghi 16-bit được tăng lên 1 cứ mỗi một chu kỳ máy. Một chu kỳ máy của 8051 có 12

chu kỳ nhịp đồng hồ của 8051, do đó, tốc độ đếm của bộ thời gian là 1/12 của tần số nhịp đồng hồ của 8051.

Khi sử dụng ở chế độ đếm, các thanh ghi 16-bit được tăng lên 1 ứng với mỗi lần chuyển từ logic "1" đến logic "0" ở các chân tín hiệu vào T0 hoặc T1. Bởi vì các bộ thời gian của 8051 cần phải lấy mẫu các cổng vào thay vì giám sát các cổng vào, 2 chu kỳ máy cần phải có trước khi bộ đếm có thể xác nhận sự chuyển từ logic "1" xuống logic "0". Điều này có nghĩa là tần số tối đa có thể tính được là 1/24 của tần số nhịp đồng hồ của 8051 (F_{OSC}).

Bit C/T# trong thanh ghi điều khiển chế độ đếm-thời gian TCON (hình 8.9) để chọn chức năng đếm hay thời gian cho các bộ thời gian. Các bit M1 và M0 để chọn 4 chế độ khai thác. Chế độ 0 làm cho bộ thời gian giống như bộ thời của 8048, một chip điều khiển thường dùng điều khiển bàn phím trong IBM-PC/XT. Đó là một bộ đếm 8-bit với giá trị chia trước là 32. Trong chế độ này, thanh ghi của bộ thời gian tác động giống như một thanh ghi 13-bit. Khi số đếm của thanh ghi chuyển từ tất cả 1 về tất cả 0 (đếm hết), thì cờ ngắt thời gian được thiết lập. Các tín hiệu đi tới các thanh ghi 13-bit được chặn cổng ở các đầu vào.

Chế độ 1 gần giống như chế độ 0, chỉ có một sự khác nhau là thanh ghi của bộ thời gian làm việc đủ 16-bit chứ không phải là 13-bit như trong chế độ 0. Tín hiệu được chặn cổng nhờ logic "1" ở chân INT. Điều này có nghĩa là chế độ 0 hay chế độ 1 có thể được sử dụng, ví dụ, để định thời gian độ rộng của xung logic "1" ở đầu vào ngắt.

Chế độ 2 định thanh ghi của bộ thời gian như là một bộ đếm 8-bit với sự nạp lại tự động. Có nghĩa là mỗi khi thanh ghi chuyển từ tất cả "1" về tất cả "0", thì một số 8-bit từ byte cao được nạp vào byte thấp của thanh ghi của bộ thời gian. Nội dung của byte cao của thanh ghi của bộ thời gian được thiết lập nhờ lệnh. Chế độ này cho phép người lập trình tạo ra một ngắt định trước theo quãng thời gian. Ví dụ, nếu tần số nhịp của 8081 là 12 MHz, thì bộ đếm tăng lên với tốc độ 1/12 của tần số nhịp, hoặc 1 triệu lần trên 1 giây. Nếu giá trị nạp lại được thiết lập là 100, thì bộ thời gian ngắt 8051 cứ mỗi lần 0,1 ms (100 μs).

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Trong đó:

TF1: TCON.7: Cờ tràn bộ thời gian 1 (timer 1). Bit này được thiết lập nhờ phần cứng khi bộ đếm-thời gian 1 tràn. Bit được xoá nhờ phần cứng khi véc-tơ ngắt trỏ tới một chương trình con xử lý ngắt

TR1: TCON.6: Bit điều khiển chạy bộ thời gian 1. Thiết lập/xóa nhờ lệnh để bật/tắt bộ đệm-thời gian 1.

TF0: TCON.5: Cờ tràn bộ thời gian 0 (timer 0). Bit này được thiết lập nhờ phần cứng khi bộ đếm-thời gian 0 tràn. Bit được xoá nhờ phần cứng khi véc-tơ ngắt trỏ tới một chương trình con xử lý ngắt

TR0: TCON.4: Bit điều khiển chạy bộ thời gian 0. Thiết lập/xóa nhờ lệnh để bật/tắt bộ đệm-thời gian 0.

IE1: TCON.3: Bit ngắt bên ngoài loại 1. Được thiết lập nhờ phần cứng khi xung ngắt bên ngoài được xác nhận. Bit được xóa nhờ phần cứng khi ngắt được xử lý.

IT1: TCON.2: Cờ điều khiển ngắt bên ngoài loại 1, được thiết lập/xóa nhờ lệnh để xác định sườn xuống/mức thấp của trigger ngắt bên ngoài.

IE0: TCON.1: Bit ngắt bên ngoài loại 0. Được thiết lập nhờ phần cứng khi xung ngắt bên ngoài được xác nhận. Bit được xóa nhờ phần cứng khi ngắt được xử lý.

IT0: TCON.0: Cờ điều khiển ngắt bên ngoài loại 0, được thiết lập/xóa nhờ lệnh để xác định sườn xuống/mức thấp của trigger ngắt bên ngoài.

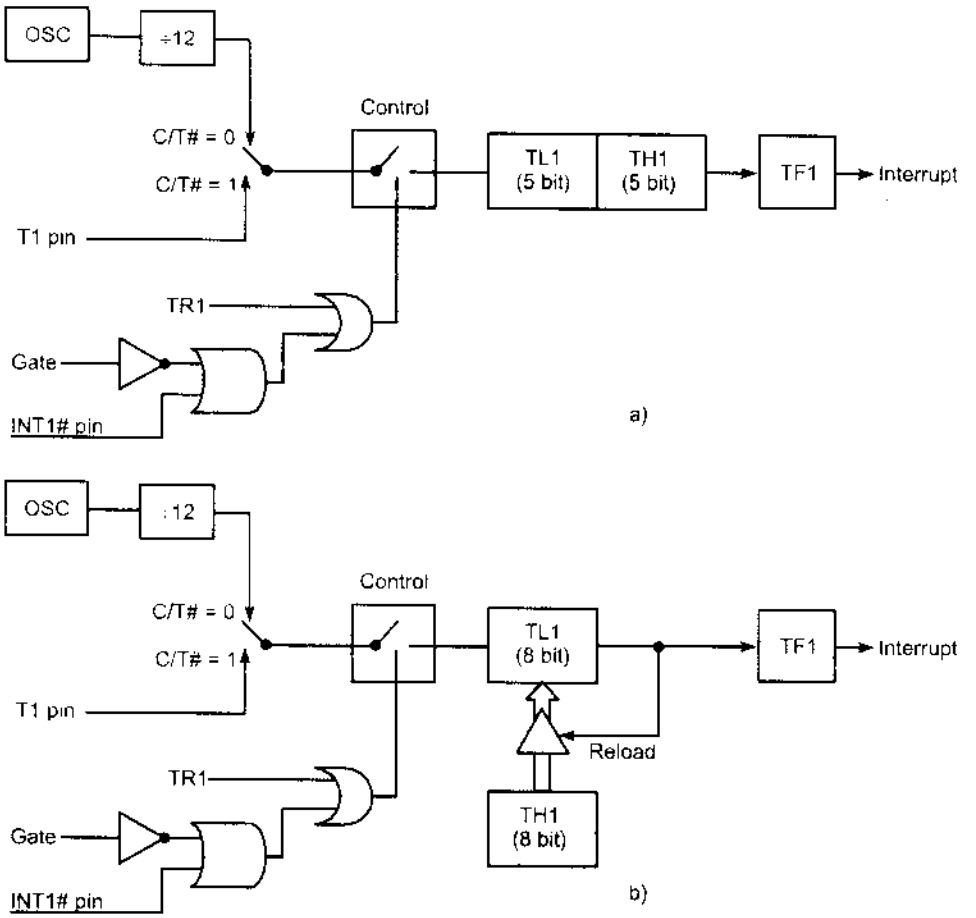
Hình 8.10: Thanh ghi điều khiển bộ đếm-thời gian TCON. Các bit cờ của TCON chỉ ra trạng thái của các bộ đếm-thời gian

GATE	C/T#	M1	M0	GATE	C/T#	M1	M0
Timer 1				Timer 2			

GATE	Khi TRx (trong TCON) được thiết lập và GATE = 1, thì TOMER/COUNTERx sẽ chạy chỉ trong khi chân INT ở trạng thái cao (điều khiển bằng phần cứng). Khi GATE = 0, thì TIMER/COUNTERx sẽ chạy chỉ trong khi TRx = 1 (điều khiển bằng lệnh).
C/T#	Chọn chế độ thời gian hay đếm. C/T# = 0: đặt chế độ thời gian, C/T# = 1: đặt chế độ đếm.
M1	Bit chọn chế độ
M0	Bit chọn chế độ

M1	M0	Mode	Khai thác
0	0	0	Bộ thời gian 13-bit
0	1	1	Bộ đếm thời gian 16-bit
1	0	2	Bộ đếm thời gian tự nạp lại 8-bit
1	1	3	(timer 0) TL0 là bộ đếm thời gian 8-bit được điều khiển bởi các bit của bộ thời gian 0 Timer 0) chuẩn. TH0 là bộ thời gian 8-bit và được điều khiển bởi các bit của bộ thời gian 1.
1	1	3	(Timer 1) dừng bộ đếm thời gian 1

Hình 8.11: Thanh ghi điều khiển chế độ bộ đếm-thời gian TMOD



Hình 8.12: Hai sơ đồ logic tương đương các bộ đếm-thời gian được cấu hình ở chế độ 1 (a), và chế độ 2 (b), đây chỉ là 2 trong 4 chế độ khác nhau có thể thiết lập

Trong chế độ 3, bộ thời gian 1 (timer 1) chứa số đếm. Mặc dù vậy, bộ thời gian 0 thiết lập TL0 và TH0 như là 2 bộ đếm 8-bit riêng biệt. Mỗi bộ đếm này có thể tạo ra một ngắt. Như thế chế độ 3 cho ta 3 bộ đếm-thời gian, trong đó, 2 bộ đếm trong bộ thời gian 0 có thể được sử dụng để tạo ngắt, và bộ thời gian 1 dùng để tính cho 65536 sự kiện.

Các bộ thời gian của 8051 được điều khiển bởi 4 thanh ghi. Mỗi bộ thời gian 1 thanh ghi byte cao và 1 thanh ghi byte thấp. Những thanh 8-bit này có thể được sử dụng để thiết lập số đếm giá trị từ 00h đến FFh.

Ngoài ra, có thanh ghi điều khiển bộ đếm-thời gian TCON và thanh ghi điều khiển chế độ bộ đếm-thời gian (hình 8.10).

Hình 8.12 là sơ đồ logic tương đương các bộ đếm-thời gian được sử dụng trong 2 chế độ (mode 1 và 2). Nó cho thấy rằng có nhiều tổ hợp khác nhau cho các bộ đếm-thời gian.

8.2.7 Hệ thống ngắt của 8051

a) Ngắt từ bên ngoài nhờ INT0# và INT1#

Có 5 cách khác nhau để ngắt 8051. Hai trong số đó được tạo ra từ các tín hiệu bên ngoài. Các cách còn lại được tạo ra do các thao tác của phần cứng vào/ra bên trong 8051.

Hai đầu vào ngắt từ bên ngoài là INT0# và INT1#. Một mức logic "0" ở INT0# thiết lập bit IE0 trong thanh ghi điều khiển của bộ đếm-thời gian TCON. Một mức logic "0" ở INT1# thiết lập bit IE1 trong TCON. Sự thiết lập các bit IE0 và IE1 của TCON gây ra một ngắt. Có 2 loại tín hiệu điện khác nhau có thể gây một ngắt ở các đầu vào INT0# và INT1#. Trước hết, ta có thể đặt 8051 xác nhận một sự thay đổi mức điện thế từ logic "1" đến "0". Đây gọi là ngắt chuyển đổi-tích cực (transition-activated interrupt). Nghĩa là, một sự thay đổi mức điện thế của tín hiệu INT0# hay INT1# sẽ gây ra ngắt. Thứ 2, một mức logic "0" ở đầu vào ngắt sẽ gây ngắt. Đó gọi là đầu vào mức- tích cực (level-activated input).

Khi 8051 trong chế độ chuyển đổi-tích cực, các bit của thanh ghi TCON là IE0 và IE1 được thiết lập khi có xuất hiện ở các đầu vào ngắt INT0# và INT1# sự chuyển đổi mức điện thế. Ngắt xảy ra và các véc-tơ ngắt trỏ tới chương trình xử lý ngắt. Khi đó các bit IE0 và IE1 sẽ được xóa.

Trong chế độ mức-tích cực, các bit IE0 và IE1 được xóa khi các đầu vào INT0# và INT1# từ mức logic "0" về mức "1". Như vậy, chương trình con xử lý ngắt khi kết thúc có thể phải tác động đến, ví

dụ, các trigger ngắt bên ngoài để xóa chúng về trạng thái đảm bảo INT0# và INT1# chuyển lên mức "1". Nếu chương trình xử lý ngắt không thể thực hiện tác động này thì các trigger ngắt bên ngoài sẽ không bị xóa, và quá trình điều khiển ngắt sẽ là không ổn.

b) Hai ngắt của 2 bộ đếm-thời gian trong 8051

Hai bộ thời gian 16-bit trong 8051 có thể tạo ra các ngắt. Các bit cờ TF0 và TF1 trong thanh ghi điều khiển bộ đếm-thời gian TCON được thiết lập nhờ sự tràn thời gian. Khi TF0 hoặc TF1 được thiết lập thì ngắt được tạo ra. Sau khi ngắt được tạo ra, và véc-tơ ngắt trỏ tới chương trình con phục ngắt để thực hiện thì TF0 hoặc TF1 được xóa.

c) Ngắt của cổng vào/ra tuần tự trong 8051

Loại ngắt thứ 5 được tạo ra nhờ mạch logic OR của cờ ngắt nhận dữ liệu RI và cờ ngắt phát dữ liệu TI trong thanh ghi điều khiển cổng tuần tự SCON. Khi nhận một từ dữ liệu từ cổng tuần tự hoặc phát xong từ dữ liệu ra cổng tuần tự có thể tạo ra ngắt. Chương trình phục vụ ngắt sẽ phải xóa các cờ ngắt sau khi kiểm tra cả hai cờ để xem cờ nào đã gây ra ngắt.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA:	IE.7:	Cấm các ngắt. Nếu EA=0, không một ngắt nào được chấp nhận. Nếu EA=1, từng ngắt sẽ được phép hoặc không nhờ vào sự thiết lập/hoặc xóa bit IE tương ứng của ngắt (IE0 hoặc IE1).
ES:	IE.4:	Cho phép hoặc cấm ngắt của cổng tuần tự
ET1:	IE.3	Cho phép hoặc cấm bộ thời gian 1 (timer 1) ngắt tràn
EX1:	IE.2:	Cho phép hoặc cấm ngắt 1 từ bên ngoài (INT1#)
ET0:	IE.1:	Cho phép hoặc cấm bộ thời gian 2 (timer 2) ngắt tràn
EX0:	IE.0:	Cho phép hoặc cấm ngắt 0 từ bên ngoài (INT0#)

Hình 8.13: Thanh ghi cho phép ngắt IE

Thanh ghi cho phép ngắt IE (Interrupt Enable register) và thanh ghi ưu tiên ngắt IP (Interrupt Priority register) điều khiển hệ thống

ngắt của 8051. Các thanh ghi này có địa chỉ là A8h và B8h. Trong hình 8.13 là nội dung của thanh ghi IE. Thanh ghi IE được sử dụng để cho phép hay cấm tất cả các ngắt của 8051 và để cho phép hoặc cấm có chọn lựa một trong 5 loại ngắt đã xét. IE là thanh ghi 8-bit và có thể được địa chỉ theo bit hoặc theo byte. 5 bit thấp của để bật/tắt 5 loại ngắt của 8051. Bit cao nhất IE7 được sử dụng để cho phép hoặc cấm tất cả các ngắt.

Ngắt nào được phục vụ trước tiên nếu ta cho phép hơn một ngắt. Mỗi một ngắt của 8051 có thể được thiết lập ở mức ưu tiên cao hoặc ở mức ưu tiên thấp nhờ thanh ghi IP (hình 8.14). Người lập trình có thể thay đổi mức ưu tiên ngắt thông qua IP.

-	-	-	PS	PT1	PX1	PT0	PX0

PS: IP.4: Xác định mức ưu tiên ngắt của cổng tuần tự
PT1: IP.3: Xác định mức ưu tiên ngắt của bộ thời gian 1 (timer 1)
PX1: IP.2: Xác định mức ưu tiên ngắt của ngắt 1 bên ngoài (INT1#)
PT0: IP.1: Xác định mức ưu tiên ngắt của bộ thời gian 0 (timer 0)
PX0: IP.0: Xác định mức ưu tiên ngắt của ngắt 0 bên ngoài (INT0#)

Hình 8.14: Thanh ghi ưu tiên ngắt IP

Ví dụ, cho rằng ta thiết lập trường hợp: PS = 1, PT0 = 1 và PT1 = 0, PX1 = 0, PX0 = 0, và trường hợp tất cả các bit của thanh ghi IP được thiết lập bằng 1 hoặc bằng 0 thì các mức ưu tiên ngắt của 8051 sẽ theo thứ tự như sau:

Mức ưu tiên ngắt	**Với PS = 1, PT0 = 1, PT1 = 0, PX1 = 0, PX0 = 0**	**Mức ưu tiên ngắt**	**Với tất cả các bit của thanh ghi IP = 1 hoặc =0**
1	PS	1	PS
2	PT0	2	PT1
3	PT1	3	PX1
4	PX1	4	PT0
5	PX0	5	PX0

8051 có các véc-tơ ngắt tương ứng với 5 loại ngắt:

Nguồn ngắt		Địa chỉ của Véc-tơ ngắt
Tín hiệu ngắt	Các bit ngắt	
INT0# = 0	IE0	0003h
Timer 0 overflow	TF0	0008h
INT1# = 0	IE1	0013h
Timer 1 overflow	TF1	001Bh
Cổng tuần tự	RI hoặc TI	0023h

Véc-tơ ngắt trỏ đến địa chỉ ngăn nhớ, mà ngăn nhớ đó chứa địa chỉ đầu tiên của chương trình con phục vụ ngắt tương ứng. Ví dụ, tín hiệu INT1# = 0 gây ra thiết lập bit IE1 = 1. Bit IE1 tác động bộ xử lý để lấy địa chỉ lệnh tiếp theo cho bộ đếm chương trình trong ngăn nhớ địa chỉ 0013h. Các chương trình con phục vụ ngắt đều chứa trong ROM của 8051.

8.3 TẬP LỆNH VÀ LẬP TRÌNH TRÊN BỘ VI ĐIỀU KHIỂN

8.3.1 Tập lệnh của 8051

Tập lệnh của 8051 có 111 lệnh, trong đó có: 49 lệnh 1 byte, 45 lệnh 2 byte và 17 lệnh 3 byte. Tất cả phân thành 5 loại lệnh khác nhau và sử dụng 8 chế độ đánh địa chỉ (xem phụ lục), đó là các nhóm lệnh: vận chuyển dữ liệu (data transfer), số học (arithmetic), logic, chuyển đổi điều khiển (control transfer), và định hướng bit (bit oriented).

a) Các lệnh vận chuyển dữ liệu (bảng F.1)

Chúng được sử dụng để sao chép dữ liệu giữa các vùng nhớ khác nhau và các thanh ghi. Có 3 loại: mục đích chung (general-purpose), đặc trưng thanh tổng và hướng đối tượng. Các lệnh vận chuyển dữ liệu không làm thay đổi các bit cờ trong thanh ghi trạng thái PSW, ngoại trừ các lệnh liên quan trực tiếp đến PSW.

(1) Nhóm lệnh mục đích chung: Có 3 lệnh chính, đó là: lệnh MOV, sao chép 1 bit hoặc 1 byte từ toán hạng nguồn đến toán hạng đích; lệnh PUSH tăng con trỏ ngăn xếp SP rồi sau đó chuyển một byte dữ liệu từ toán hạng nguồn đến vùng ngăn xếp; Lệnh POP sao chép 1 byte từ ngăn xếp mà địa chỉ được trỏ bởi SP. Sau đó SP được giảm đi để trỏ tới vùng nhớ tiếp theo.

(2) Các lệnh đặc trưng thanh tổng: Thực hiện chuyển dữ liệu vào và ra khỏi thanh tổng (ACC) của 8051. Lệnh XCH (exchange) trao đổi một byte dữ liệu trong toán hạng nguồn với thanh tổng. Lệnh XCHD trao đổi 4 bit thấp (low-order nible) trong toán hạng nguồn với 4 bit thấp của thanh tổng.

Các lệnh MOVX chuyển dữ liệu giữa bộ nhớ ngoài và thanh tổng. Có 2 loại thuộc nhóm này. Nhóm thứ nhất sử dụng địa chỉ bộ nhớ ngoài ghi trong thanh ghi DPTR. Nhóm thứ 2 sử dụng thanh ghi 8-bit R0 hoặc R1 để địa chỉ 256 byte của bộ nhớ chương trình bên ngoài.

Các lệnh MOVC chuyển một byte từ bộ nhớ chương trình tới thanh tổng. Chúng sử dụng đánh địa chỉ tương đối, và bắt đầu bằng nạp giá trị offset (độ dịch) vào thanh tổng. Có 2 loại thuộc nhóm này. Địa chỉ tương đối nhận được trong thanh tổng bằng phép cộng nội dung của thanh tổng với nội dung của DPTR.

(3) Lệnh hướng đối tượng: Là lệnh 3 byte và được sử dụng để nạp 16 bit của địa chỉ vào thanh ghi DPTR.

b) Các lệnh số học (bảng F.2)

Có 4 loại lệnh số học như trình bày trong bảng F.2. 8051 thực hiện các phép toán với các số dấu không có dấu. Nhưng nhờ sử dụng cờ tràn, ta có thể thực hiện các lệnh cộng và trừ cho các số nhị phân không dấu hay có dấu. Cờ nhớ phụ AC (Auxiliary Carry flag) trong PSW cho phép thực hiện các phép tính với các ở mã BCD. Các lệnh hiệu chỉnh số hệ 10 dùng để hiệu chỉnh kết quả phép cộng hai số BCD với 2 digit. Nếu kết quả cộng hai số BCD hai digit vượt quá 99_{10} thì cờ CY sẽ thiết lập.

Các lệnh số học có thể lập cờ CY, AC, OV và P của PSW. Cũng có thể kết hợp cờ CY để thực hiện các phép tính số học với các số chính xác.

Lệnh nhân (MUL) thực hiện nhân nội dung của thanh tổng và thanh ghi B. Kết quả có độ dài 2 byte, trong đó byte thấp nằm trong thanh tổng còn byte cao nằm trong thanh ghi B. Nếu kết quả trong thanh ghi B bằng 0 thì cờ OV của PSW được xóa.

Lệnh chia thực hiện phép chia nội dung của thanh tổng với nội dung của thanh ghi B. Thương số ở lại thanh tổng, và mọi số dư còn lại nằm trong thanh ghi B.

c) Các lệnh logic (bảng F.3)

Các lệnh logic có thể thao tác theo tong bit hay theo byte. Có các lệnh logic 1 và 2 toán hạng. Các lệnh 1 toán hạng là các lệnh: xóa, thiết lập, lấy bù, và 5 lệnh dịch quay vòng. Nhưng lệnh dịch quay vòng làm việc với thanh tổng A. Đó là các lệnh: dịch trái (RL), dịch trái quay vòng với CY (RCL), dịch phải quay vòng (RR), và dịch phải quay vòng với CY (RRC). Lệnh quay vòng thứ 5 là lệnh SWAP, thực hiện quay vòng nội dung thanh tổng A về bên trái 4 bit, tức là chuyển vị trí các bit 0-3 với 4-7.

Các lệnh logic 2 toán hạng là các lệnh AND, OR, và XOR. Chúng thực hiện phép logic với 2 toán hạng nguồn và kết quả năm loại toán hạng thứ nhất. Trong cách viết lệnh, toán hạng thứ nhất đứng trước dấu phẩy (hay cạnh mã lệnh).

d) Các lệnh chuyển điều khiển (bảng F.4)

Có 3 loại lệnh chuyển điều khiển: các lệnh gọi tới chương trình con không điều kiện, trở về từ chương trình con, các lệnh nhảy không điều kiện, nhảy có điều kiện, và các lệnh ngắt.

Các lệnh gọi, trở về, và nhảy không điều kiện khi thực hiện sẽ nạp giá trị mới vào con trỏ chương trình. Giá trị mới có thể lấy trực tiếp từ 1 byte hoặc các byte dữ liệu trong lệnh, hoặc có thể là giá trị kết quả của một địa chỉ gián tiếp.

Lệnh ACALL (gọi tuyệt đối) và LCALL (gọi dài) bắt đầu các chương trình con. ACALL là lệnh dài 2 byte. Nó sử dụng kiểu đánh địa tuyệt đối 11-bit. 8 bit trong byte thứ 2 cộng với 3 bit từ byte thứ nhất được cộng (không dấu) vào nội dung của bộ đếm chương trình. Nghĩa là, địa chỉ của chương trình con bị gọi có giá trị từ 0 đến + 2 kB. LCALL là lệnh 3 byte. Nó địa chỉ đủ cho vùng nhớ chương trình dung lượng 64 kB.

Các lệnh AJMP và LJMP vận hành giống như ACALL và LCALL, ngoại trừ đó là nhảy đến một đoạn nào đó trong chương trình chính chứ không phải là gọi tới chương trình con. Lệnh SJMP (nhảy ngắn) sử dụng địa chỉ tương đối (tương đối với với giá trị hiện hành của bộ đếm chương trình). Do vậy, lệnh SJMP có địa chỉ trong khoảng -128 đến +127.

Các lệnh nhảy có điều kiện chỉ sử dụng kiểu đánh địa chỉ tương đối, và dải địa chỉ trong khoảng -128 đến +127. Chú ý là, lệnh JZ (nhảy zero), và JNZ (nhảy không zero) không theo giá trị bit cờ zero bởi vì bit này không có trong PSW, nhảy theo nội dung của thanh tổng. Lệnh JB là lệnh nhảy nếu bit địa chỉ trực tiếp thiết lập = 1.

Các lệnh CJNE là những lệnh 3 byte. Chúng so sánh các giá trị trong 2 toán hạng và thực hiện nhảy nếu các giá trị không bằng nhau.

Lệnh DJNZ là lệnh 2 byte, nó giảm toán hạng nguồn và trả kết quả về toán hạng nguồn. Nhảy sẽ được thực hiện nếu kết quả khác zero.

8051 không có các lệnh vào/ra riêng biệt. Để nhận hoặc phát dữ liệu, ta cần phải sử dụng các lệnh MOV làm việc với các thanh ghi cổng vào/ra. Nếu cần phải có nhiều cổng vào/ra trong 8051 thì phải dùng tới các ngăn nhớ trong vùng nhớ dữ liệu.

e) Các lệnh xử lý bit (bảng F.5)

Các lệnh xử lý bit của 8051 gồm các lệnh: xóa, thiết lập, bù, AND, OR, vận chuyển bit nào đó và bit cờ nhớ (carry). Nhóm lệnh này được sử dụng cho các mục đích điều khiển và xử lý từng bit của byte dữ liệu.

Có một hạn chế lớn của 8051, đó là ngăn xếp bị giới hạn về dung lượng. Nếu muốn tăng dung lượng ngăn xếp, cần phải mở rộng thêm ở bộ nhớ ngoài, do vậy, cần phải sử dụng nhiều lệnh để đẩy hoặc đọc dữ liệu với ngăn xếp bên ngoài. Ngoài ra, so với các bộ vi xử lý 8-bit (ví dụ Z80), tập lệnh của 8051 có những hạn chế về khả năng ứng dụng vạn năng, nhưng bù lại, nó có tính chuyên dụng, phù hợp để tổ chức hệ thống điều khiển nhỏ, rẻ tiền.

8.3.2 Chu kỳ máy của 8051

Đối với 8051, cần tối thiểu 12 chu kỳ nhịp đồng hồ để thực hiện một lệnh. Một chu kỳ máy T_{mc} được tính bằng:

T_{mc} = số nhịp đồng hồ trong chu kỳ máy/tần số nhịp đồng hồ

Nếu 8051 sử dụng nhịp đồng hồ ở 11,0592 MHz thì:

T_{mc} = 12/11,0592 = 1,085 micro giây

Thời gian để thực hiện một lệnh T_{instr} được tính bằng:

T_{instr} = số chu kỳ máy * T_{mc}

Ví dụ, lệnh NOP có số chu kỳ máy = 1 thì $T_{instr} = T_{mc}$ = 1,085 μs.

8.3.3 Lập trình 8051

a) Các kiểu đánh địa chỉ của 8051

Có 5 kiểu đánh địa chỉ của 8051.

(1) Thanh ghi (register)

- Sử dụng các thanh ghi R0 - R7 để chứa dữ liệu;
- Đặt toán hạng vào thanh ghi và các lệnh thao tác với các toán hạng bằng cách tham chiếu tới thanh ghi, ví dụ:

```
mov A, R0
mov R2, A
add A, R1
```

- Các thanh ghi nguồn và đích phải cùng kích thước;

• Không có các lệnh vận chuyển dữ liệu giữa các thanh ghi, ví dụ:

mov R4, R7 ; là lỗi

(2) Trực tiếp (direct)

• Để địa chỉ trực tiếp dữ liệu cất trong bộ nhớ RAM và trong các thanh ghi, các cổng (ports), các thanh ghi chức năng đặc biệt (SFR)

• Địa chỉ của toán hạng ghi trực tiếp trong lệnh, ví dụ:

movA, 40h ; chuyển nội dung ngăn nhớ có địa chỉ 40h vào A

• Đánh địa chỉ thanh ghi bằng giá trị địa chỉ trực tiếp 8-bit, ví dụ:

movA, 4h ; chuyển nội dung của thanh ghi R4 vào A.
; Giá trị 4h là địa chỉ thanh ghi R4

movA, R4 ; tương tự như trên.

• Tất cả các thanh ghi và các thanh ghi chức năng đặc biệt đều có các địa chỉ

• Ngăn xếp trong 8051 chỉ sử dụng các chế độ đánh địa chỉ trực tiếp

(3) Ngay lập tức (immediate)

• Toán hạng (dữ liệu) được xác định trực tiếp ngay trong mã lệnh (opcode);

• Toán hạng là một hằng số trong hợp ngữ, có thể 8-bit, 16-bit;

• Dữ liệu trực tiếp được đặt phía dấu #.

Ví dụ:

```
mov A, #25h          ; hằng số 8-bit chuyển vào A
mov DPTR, #1FFFh ; hằng số 16-bit chuyển vào thanh ghi DPTR
temp EQU40           ; gán biến temp = 40
movR1, #temp         ; chuyển giá trị 40 trong temp vào R1
```

(4) Gián tiếp thanh ghi (register indirect), @Rn

• Thanh ghi chứa giá trị là địa chỉ của dữ liệu trong bộ nhớ

• Chỉ các thanh ghi R0, R1 và DPTR có thể được sử dụng trong chế độ này.

• R0 và R1 có thể được sử dụng cho bộ nhớ bên trong 256 byte (kể cả SFR) hoặc từ 00h đến FFh của bộ nhớ ngoài, ví dụ:

```
mov A, @R0  ; chuyển nội dung ngăn nhớ bên trong 8051,
            ; mà địa chỉ của nó chứa trong R0, vào A
mov @R1, A  ; nội dung của A chuyển vào ngăn nhớ bên,
            ; trong mà địa chỉ của ngăn nhớ chứa trong R1
movx A, @R0; đọc nội dung của ngăn nhớ bên ngoài 8051,
            ; mà địa chỉ của nó chứa trong R1, vào A
```

• Thanh ghi DPTR có thể được sử dụng cho bộ nhớ dữ liệu bên ngoài, ví dụ:

```
movx A, @DPTR  ; chuyển nội dung của ngăn nhớ dữ liệu bên
               ; ngoài mà địa chỉ của nó chứa trong DPTR,
               ; vào A
movx @DPTR, A  ; chiều chuyển dữ liệu ngược với lệnh trên.
```

(5) Chỉ số (indexed)

• Sử dụng một thanh ghi để chứa giá trị con trỏ và các thanh ghi khác chứa giá trị độ dịch (offset);

• Địa chỉ hiệu dụng EA là tổng của giá trị cơ sở (base) và offset, ví dụ:

```
movc A, @A+DPTR; chuyển byte mã hóa tương đối với DPTR
               ; tới A, địa chỉ EA là kết của DPTR + A
movc A, @A+PC  ; chuyển byte mã hóa tương đối với PC tới
               ; A, địa chỉ EA là kết quả PC + A
```

• Chế độ này được dùng rộng rãi để thực hiện tham chiếu nhanh các bảng, các mảng dữ liệu, các bộ mã ký tự trong bộ nhớ chương trình (ROM).

b) Một số ví dụ lập trình

(1) Các lệnh chuyển điều khiển

• Rẽ nhánh không điều kiện:

```
ajmp addr11   ; nhảy tuyệt đối đến địa chỉ tượng trưng addr11
ljmp addr16   ; nhảy xa đến địa chỉ tượng trưng addr16.
              ; Địa chỉ đích dài 2 byte, lệnh dài 3 byte
sjmp rel      ; nhảy ngắn đến địa chỉ tương đối rel từ
              ; PC + 127 (00h – 7h) đến PC - 128 (80h - FFh)
              ; nếu PC = 15h, rel = 0FEh thì địa chỉ đích là
              ; 15h + FEh = 13h
jmp @A+DPTR   ; nhảy đến địa chỉ gián tiếp
```

• Rẽ nhánh có điều kiện:

```
jz rel        ; nhảy ngắn đến địa chỉ tương đối nếu Z = 1
jnz rel       ; nếu Z = 0
djnz rel      ; giảm và nhảy tương đối nếu Z = 0
cjne rel      ; so sánh và nhảy tương đối nếu không bằng
```

• Gọi tới chương trình con:

```
acall addr11  ; gọi tuyệt đối đến chương trình con trong không
              ; nhớ của chương trình, 2 kB. Lệnh dài 2 byte,
lcall addr16  ; gọi xa đến chương trình con trong không
                gian nhớ
              ; dung lượng 64 kB.
              ; Lệnh dài 3 byte. Khi gọi PC cất vào ngăn xếp
ret           ; trở về từ chương trình con, phục hồi PC từ
                ngăn xếp
reti          ; trở về từ chương trình phục vụ ngắt
```

(2) Vòng lặp sử dụng lệnh djnz

• Cộng 10 lần giá trị 3 với A:

```
        mov A, #0        ; xóa A
        mov R2, #10      ; nạp R2 = 10
again:  add A, #03       ; A <- A + 3
        djnz R2, again   ; lặp lại phép cộng cho đến khi R2 = 0
        mov R5, A        ; cất nội dung A vào R5
```

• Vòng lặp bên trong vòng lặp sử dụng lệnh djnz:

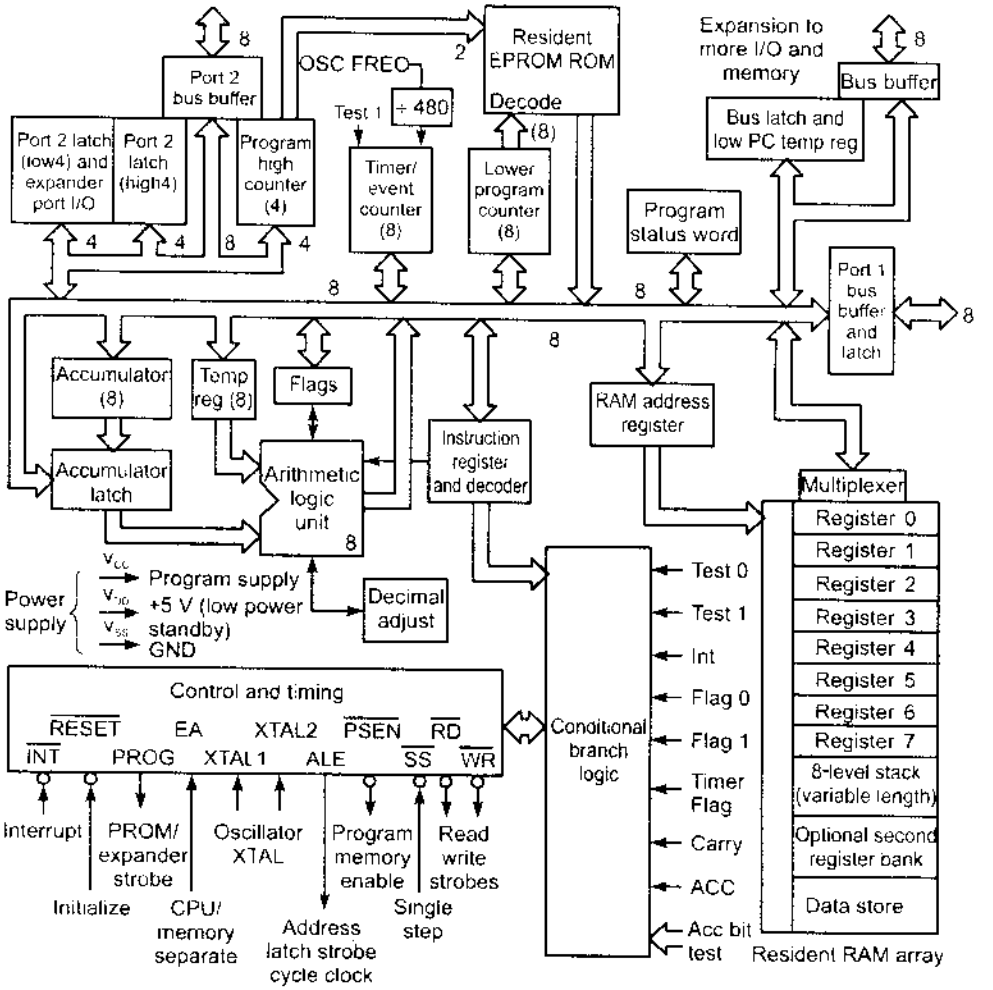
```
        mov R3, #100
loop1:  movR2, #100      ; thử 1000 lần lặp
loop2:  nop              ; no operation
        djnz R2, loop2   ; lặp lại vòng lặp 2 cho đến khi R2 = 0
        djnz R3, loop1   ; lặp lại vòng lặp 1 cho đến khi R3 = 0
```

(3) Đọc một giá trị x từ cổng P1 và gửi x^2 ra cổng P2

```
        ORG 0
        mov DPTR, #LUT   ; 300h là địa chỉ của LUT
        mov A, #0FFh     ;
        mov P1, A        ; lập trình cổng P1 để đọc dữ liệu vào
back:   movA, P1         ; đọc giá trị X từ cổng P1
        movc A, @A+DPTR  ; đọc X2 từ LUT
        mov P2, A        ; đưa X2 ra cổng P2
        sjmp back        ; lặp
        ORG 300h
LUT: DB0, 1, 4, 9, 16, 25, 36, 49, 64, 81
```

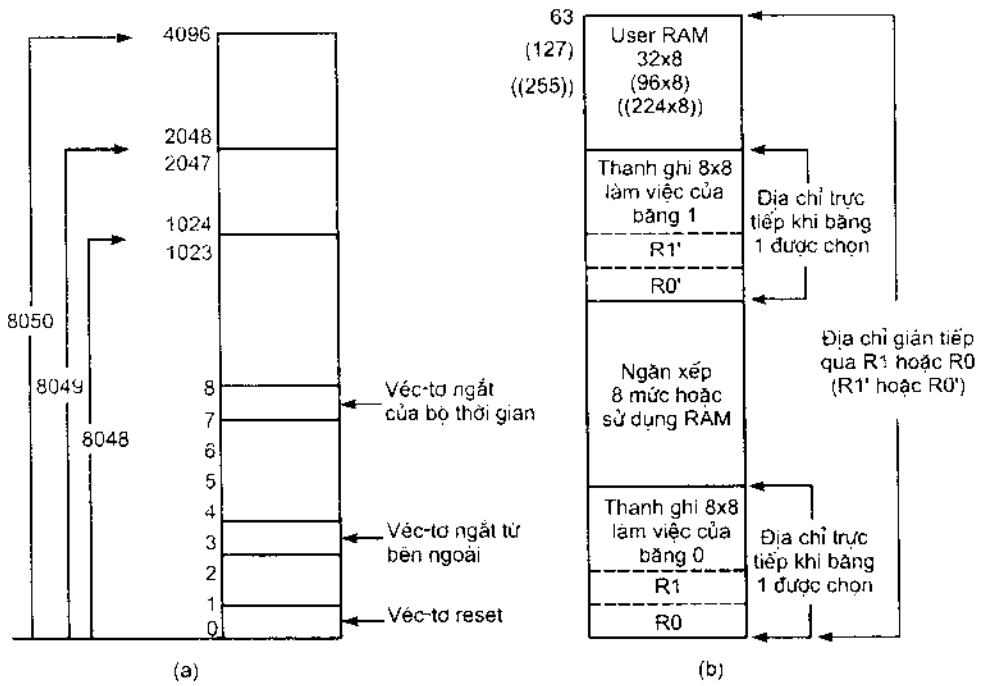
8.3.4 Cấu trúc của các bộ vi điều khiển 8048, 8049, 8050

8048, 8049, 8050 có kiến trúc bên trong tương tự nhau ngoại trừ dung lượng của bộ nhớ, và sơ đồ khối kiến trúc trình bày trong hình 8.15.



Hình 8.15: Sơ đồ khối kiến trúc bên trong của 8048, 8049 và 8050

Sự sắp xếp bộ nhớ chương trình và bộ nhớ dữ liệu của các chip 8048, 8049, và 8050 được cho ở hình 8.16a, b.



Hình 8.16: (a): Tổ chức bộ nhớ chương trình
(b): và bộ nhớ dữ liệu của 8048,8049, 8050

BÀI TẬP ÔN LUYỆN

1. Nếu cần mở rộng bộ nhớ chương trình của 8051 thì ta có thể ..... và .....(Chọn 2 trong 4 phương án):
   a) Bổ sung 64 kB ROM bên ngoài vào không gian nhớ dữ liệu.
   b) Bổ sung 60 kB ROM bên ngoài cho các địa chỉ lệnh chương trình lệnh phía trên 4 kB ROM bên trong.
   c) Sử dụng EA# và tất cả bộ nhớ chương trình bên ngoài.
   d) Bổ sung 64 kB tất cả RAM bên ngoài.
2. ROM bên trong 8051:
   a) Nằm trong không gian nhớ dữ liệu.
   b) Được dùng để cất dữ liệu thay đổi của chương trình.
   c) Là 4 kB của ROM trong không gian nhớ của chương trình.
3. Nếu cần mở rộng bộ nhớ RAM của 8051 cho dữ liệu ta có thể:
   a) Bổ sung 64 kB ROM bên ngoài vào không gian nhớ dữ liệu.
   b) Bổ sung 60 kB ROM bên ngoài cho các địa chỉ lệnh chương trình lệnh phía trên 4 kB ROM bên trong.
   c) Sử dụng EA# và tất cả bộ nhớ chương trình bên ngoài.
   d) Bổ sung 64 kB tất cả RAM bên ngoài.
4. Đánh địa chỉ theo bit là đặc điểm quan trọng của 8051 khi:
   a) Nó được sử dụng như là một vi điều khiển.
   b) Được sử dụng với 64 kB ROM bên ngoài.
   c) Được sử dụng như một máy tính miro đa năng.
   d) Được sử dụng cùng với RAM mở rộng.
5. 8051 có cờ đặc biệt để:
   a) Chỉ ra rằng băng thanh ghi được chọn.

b) Sử dụng cùng với đánh địa chỉ thanh ghi

c) Bổ sung vào địa chỉ thanh ghi sao cho 8051 có thể chọn đúng vùng nhớ.

d) Tất cả a, b, c.

6. Các thanh ghi R0 đến R7 có thể tìm thấy trong:

a) Băng 0, 1, 2, hoặc 3.

b) Các vùng nhớ địa chỉ từ 80h đến F8h

c) Chỉ băng 0

d) Tất cả a, b, c.

7. Khi ngăn xếp của 8051 được khởi tạo nó sử dụng ngăn nhớ 08h. Ngăn nhớ này cũng được sử dụng để:

a) Thay đổi chọn băng.

b) Cất giữ dữ liệu của chương trình.

c) Địa chỉ RAM bên ngoài.

d) Cho thanh ghi R0 trong băng thứ 2 (băng 1).

8. Các lệnh của 8051 cho phép đánh địa chỉ theo bit cho:

a) Tất cả các vùng nhớ trong 128 byte RAM bên trong.

b) Tất cả 128 byte vùng nhớ ban đầu bên trong và bên ngoài.

c) 16 byte nhớ từ 20h đến 30h.

d) 80 byte nhớ cuối cùng của bộ nhớ bên trong.

9. Con trỏ ngăn xếp của 8051 khác với con trỏ ngăn xếp của các vi xử lý đa năng vì nó là ..... (Chọn 2 trong 4 phương án).

a) Thanh ghi 8-bit và do đó ngăn xếp chỉ gồm 128 byte của RAM bên trong.

b) Thanh ghi 8-bit và do đó ngăn xếp chỉ gồm 128 byte của RAM bên ngoài.

c) Thanh ghi tăng dần.

d) Thanh ghi giảm dần.

10. Các cổng vào/ra 0 và 2 của 8051 được sử dụng cho 2 chức năng, đó là ..... và ..... (chọn 2 trong 4 phương án):
    a) Cổng vào/ra 16-bit dùng chung.
    b) Hai cổng vào/ra 8-bit song hướng dùng chung.
    c) 16-bit memory address bus cho bộ nhớ bên ngoài.
    d) Cổng cho memory data bus bên ngoài.
11. Cổng vào/ra 0 của 8051 có thể thực hiện 3 chức năng khác nhau, đó là, nó làm việc như ..., như ..., và như .... (chọn 3 trong số 4 phương án):
    a) Cổng vào/ra 16-bit dùng chung.
    b) Cổng vào/ra 8-bit song hướng dùng chung.
    c) 16-bit memory address bus cho bộ nhớ bên ngoài.
    d) Cổng cho memory data bus bên ngoài.
12. Khi 8051 sử dụng bộ nhớ chương trình hay dữ liệu bên ngoài thì nó sử dụng bus trộn address/data. Điều này được thực hiện để:
    a) Yêu cầu bộ nhớ ngoài sử dụng một chốt địa chỉ.
    b) 8051 sử dụng chân tín hiệu ALE
    c) Tăng tốc độ thao tác với bộ nhớ.
    d) Một số chân tín hiệu được sử dụng cùng với bộ nhớ ngoài.
13. Cổng nối tiếp của 8051 là cổng song công đầy đủ (full-duplex). Điều đó có nghĩa là:
    a) 8051 sử dụng một bộ đệm để điều khiển cổng nối tiếp.
    b) Cổng nối tiếp có thể nhận dữ liệu phát từ đường truyền lần thứ 2 trước khi lần phát trước được đọc.
    c) Cổng nối tiếp có thể phát dữ liệu cùng đồng thời với nhận dữ liệu.
    d) Địa chỉ SBUF của 8051 là hai thanh ghi được chọn nhờ các lệnh đọc và ghi.
14. Cổng nối tiếp của 8051 nhận xung nhịp đồng hồ cho tốc độ baud từ:

a) Bộ tạo nhịp đồng hồ bên trong 8051.

b) Một trong các bộ đếm thời gian của 8051.

c) Bộ đếm thời gian T1 trong tất các các chế độ.

d) Nó không cần nhịp đồng hồ.

15. 8051 có 5 nguồn ngắt khác nhau, đó là:

a) Hai tín hiệu khác nhau bên ngoài.

b) Hai loại khác nhau của các tín hiệu điện.

c) Các bộ đếm thời gian bên trong và cổng nối tiếp.

d) Tất cả a, b, c.

16. Sự thiết lập cờ ngắt nhận RI hoặc cờ ngắt phát TI cho cổng nối tiếp gây ra một ngắt. Chương trình phân biệt được nguyên nhân ngắt (phát hoặc nhận) nhờ:

a) Véc-tơ ngắt được sử dụng

b) Kiểm tra 2 cờ ngắt trong đoạn đầu tiên của chương trình con xử lý ngắt.

c) Biết được rằng một ký từ không đồng bộ đã nhận được từ đường truyền.

d) Không phải những lý do trên.

17. Một chương trình con phục vụ ngắt sẽ bắt đầu thực hiện ở:

a) Vùng nhớ chương trình địa chỉ 0023h.

b) Vùng nhớ dữ liệu địa chỉ 0023h.

c) Vùng thanh ghi-bộ nhớ bên trong địa chỉ 0023h.

d) Vùng nhớ mà ta chọn khi viết chương trình xử lý ngắt.

18. Tất cả các lệnh vận chuyển dữ liệu của 8051 giữa bộ nhớ dữ liệu bên ngoài và thanh tổng (Accumulator) đều sử dụng:

a) Đánh địa chỉ trực tiếp.

b) Một số kiểu đánh địa chỉ gián tiếp.

c) Một byte của dữ liệu chương trình.

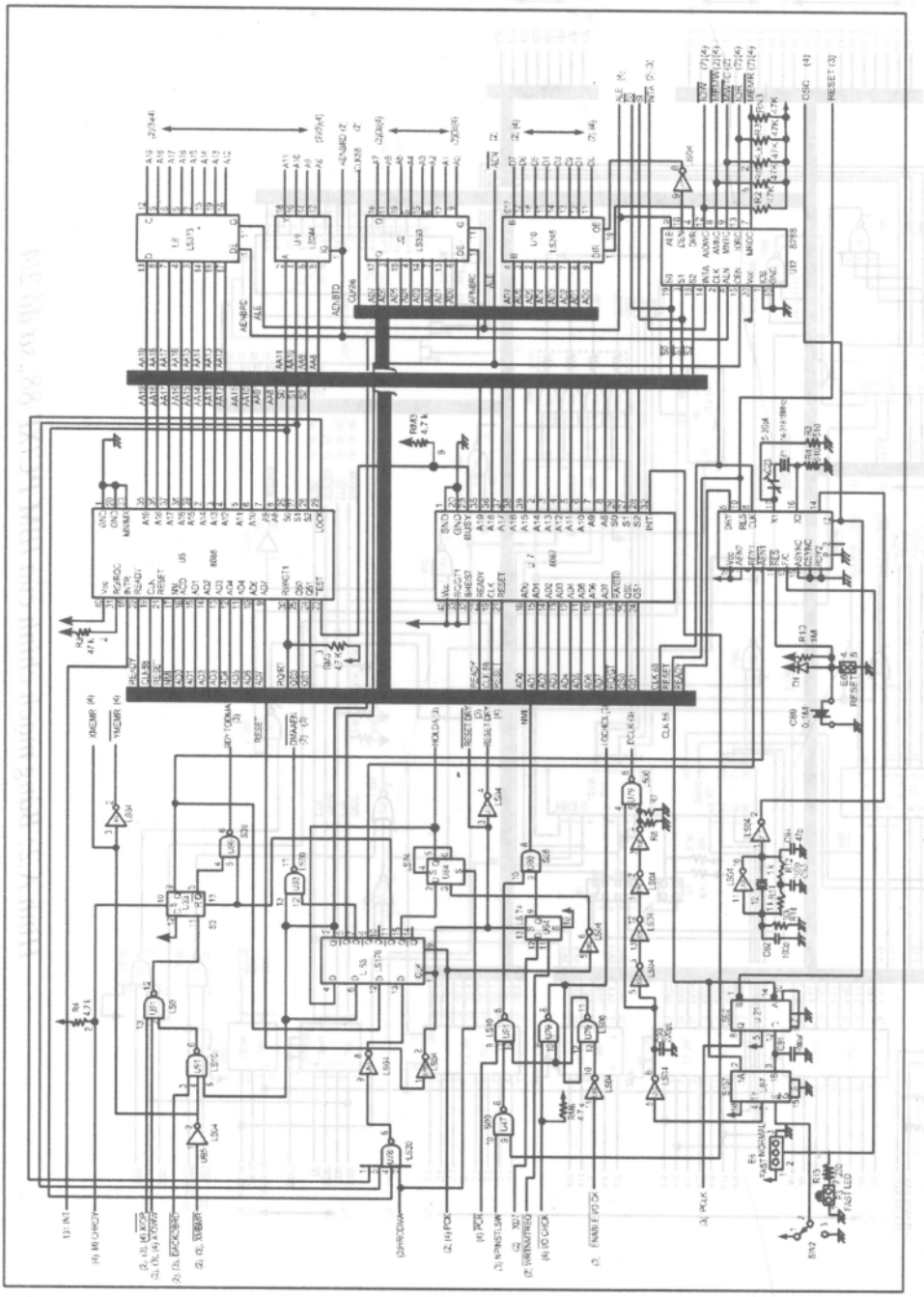
d) Tất cả a, b, c

19. Các lệnh số học của 8051 có lệnh ....., mà lệnh này không có trong các lệnh số học của Z80:
    a) Add with carry; b) Increment;
    c) Multiply; d) Decrement with carry.
20. Sự khác nhau giữ lệnh vào/ra và các lệnh khác của 8051 là:
    a) Lệnh vào/ra coi thanh tổng là đích.
    b) 8051 không có xử lý DMA.
    c) Các cổng vào/ra của 8051 có thể được sử dụng để đánh địa chỉ bộ nhớ.
    d) Không có sự khác nhau.
21. Sự khác nhau cơ bản giữa 8051 và 8052 là:
    a) Dung lượng RAM và ROM
    b) ROM bên trong
    c) Không có cổng nối tiếp.
    d) Không có sự khác nhau cơ bản ngoại trừ tốc độ và chi phí (giá).
22. 8048 và 8049 có thể được sử dụng như thế nào nếu chúng không có cổng nối tiếp?
23. Hãy thiết kế thử một bộ tính thời gian cuộc gọi điện thoại sử dụng một trong các chip 8048, 8048, 8051, 8052, ... Giải thích sơ đồ thiết kế.
24. Hãy thiết kế thử một đồng hồ đo vạn năng (VonAmper kế số) dùng 8051 hoặc 8052. Giải thích thiết kế.
25. Hãy thiết kế thử một bộ ghi vi chương trình vào FLASH ROM, EPROM dùng 8051 hoặc 8052. Giải thích thiết kế.
26. Hãy thiết kế thử một máy đo tốc độ động cơ điện dùng 8051 hoặc 8052. Giải thích thiết kế.
27. Hãy thiết kế thử một máy đo nhiệt độ trong phòng, nhiệt độ không khí, độ ẩm không khí và đưa ra hiển thị LED dùng 8051 hoặc 8052. Giải thích thiết kế.

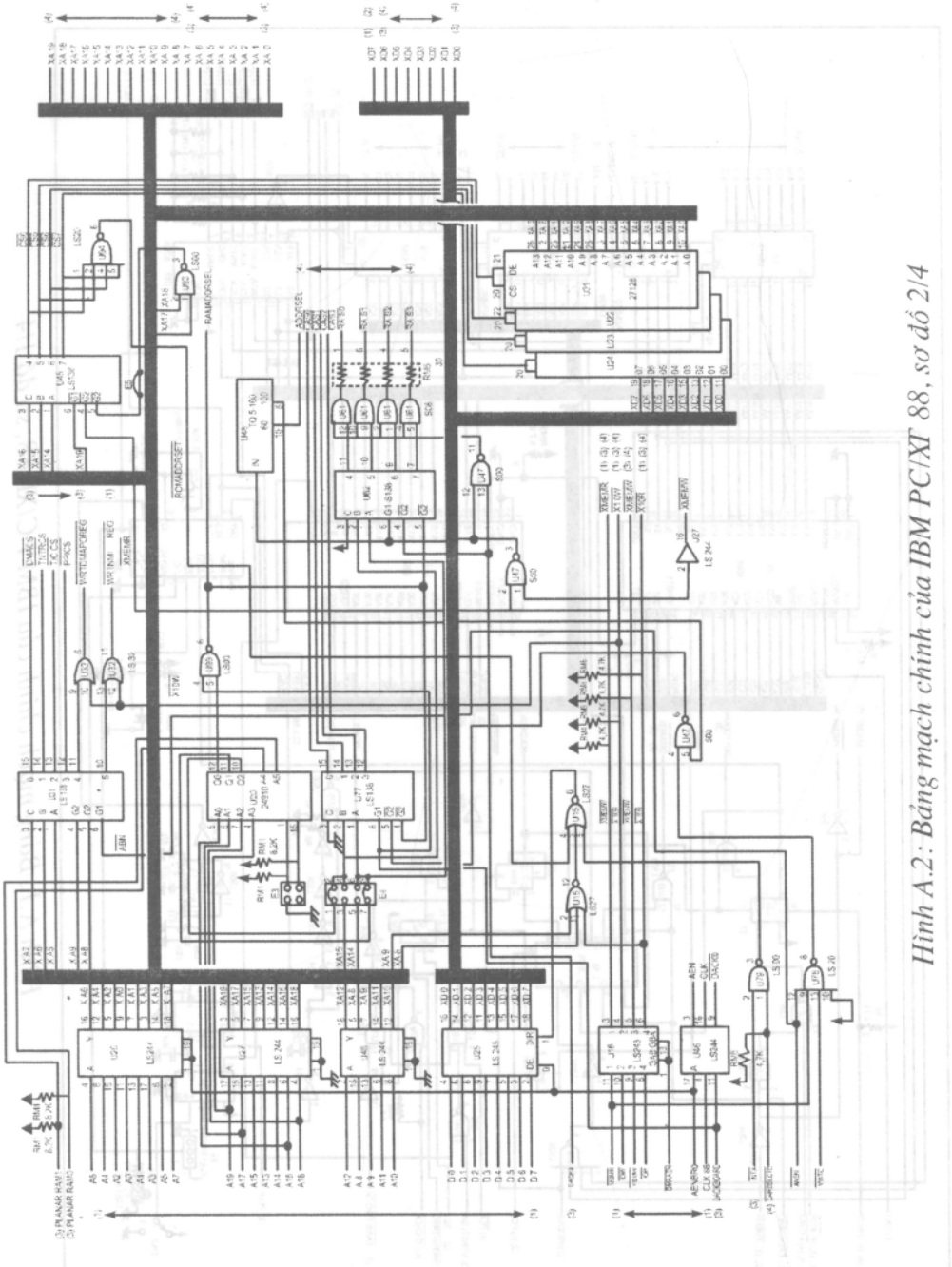
28. Hãy thiết kế thử một đo nhiệt độ trong phòng, nhiệt độ không khí, độ ẩm không khí và đưa ra hiển thị LED dùng 8051 hoặc 8052. Giải thích thiết kế.
29. Có thể dùng 8031, 8032 trong những ứng dụng nào? So sánh chúng với 8751.
30. Vẽ sơ đồ khối của 8051 với các khối: CPU, Bus control, 4 I/O ports, Serial port, Timer 1, Timer 0, RAM 128, ROM 4K, Interrupt control, Clock generator, system bus. Giải thích từng khối chức năng.
31. Vẽ sơ đồ ghép nối của 8051 với EPROM (hoặc FLASH MEMORY) và RAM bên ngoài. Chú ý phải có sử dụng mạch chốt địa chỉ nhờ ALE.
32. Vẽ sơ đồ ghép nối giữa một hệ thống 8051 với PC.

PHẦN PHỤ LỤC

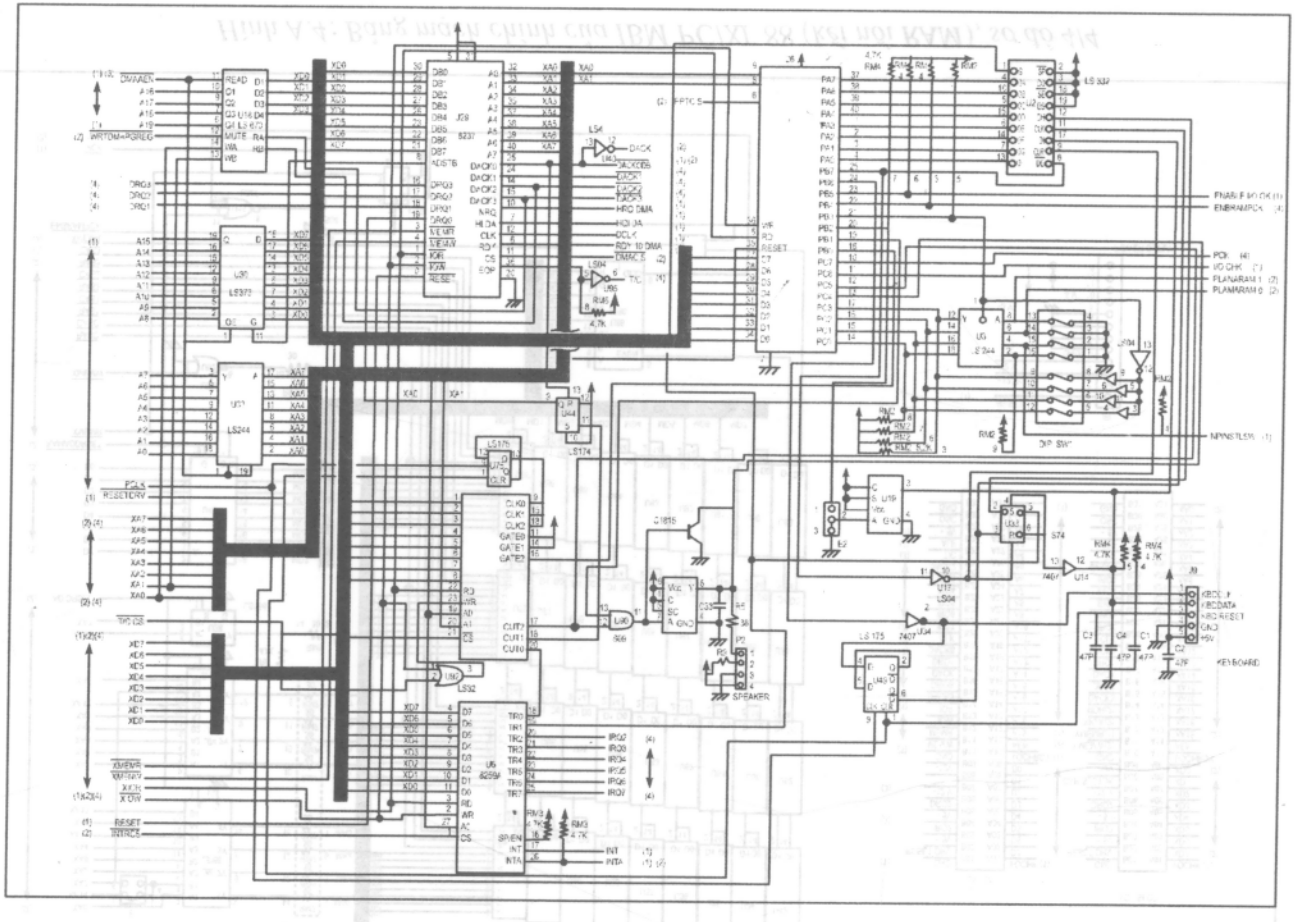
Phụ lục A: BẢNG MẠCH CHÍNH CỦA IBM PC/XT 88 VÀ BẢNG MẠCH VÀO/RA ĐA NĂNG



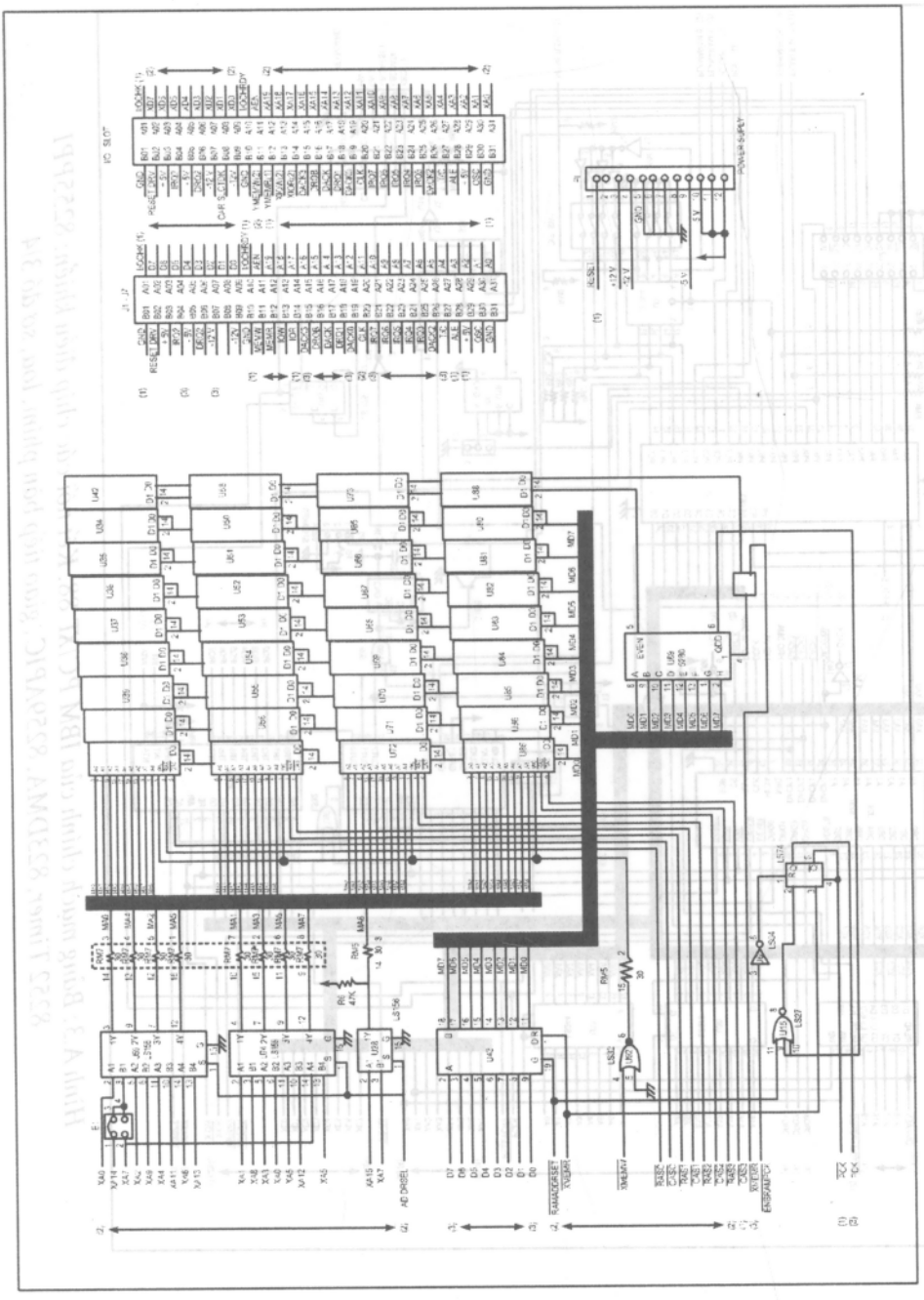
Hình A.1: Bảng mạch chính của IBM PC/XT 88, sơ đồ 1/4



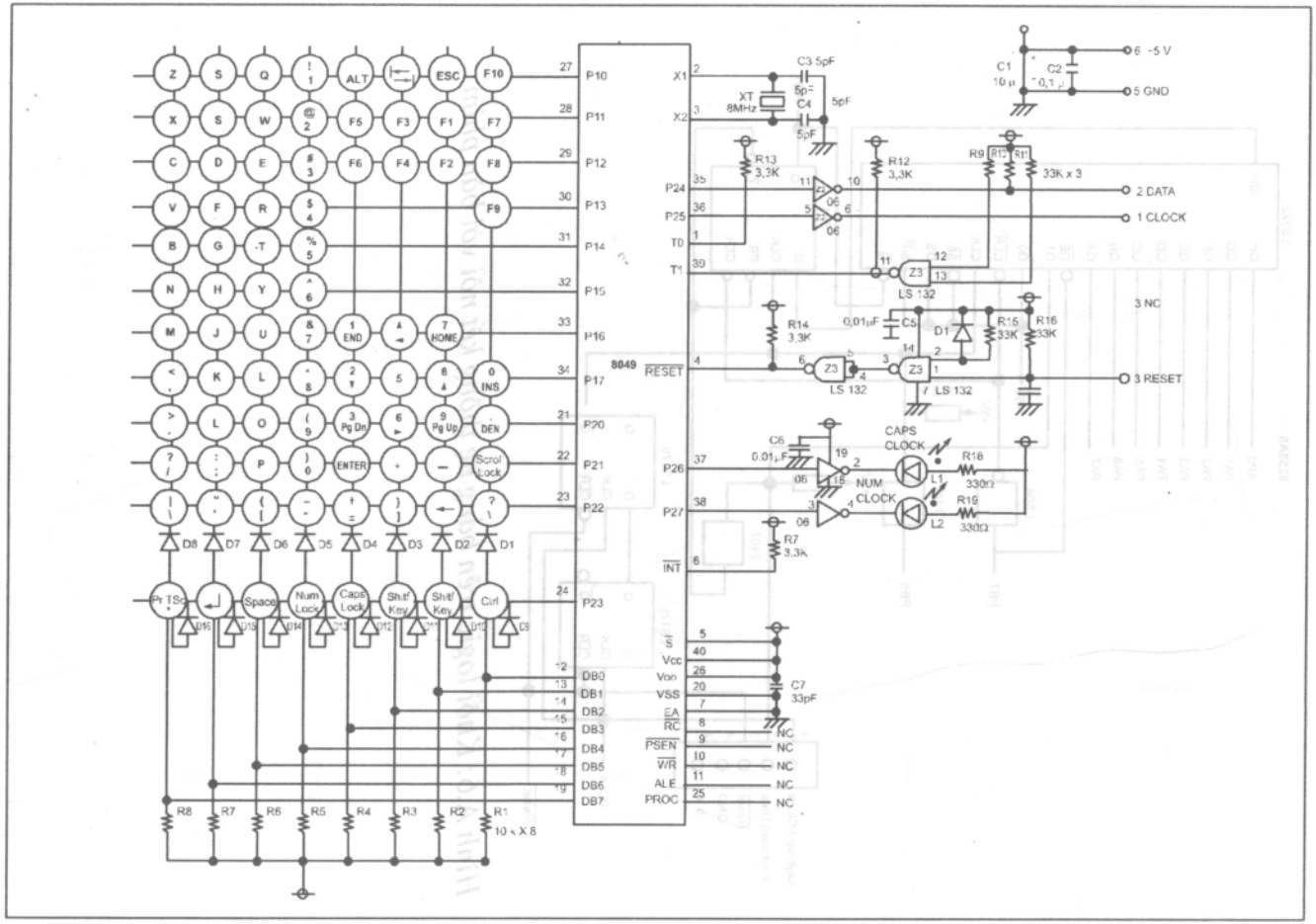
Hình A.2: Bảng mạch chính của IBM PC/XT 88, sơ đồ 2/4



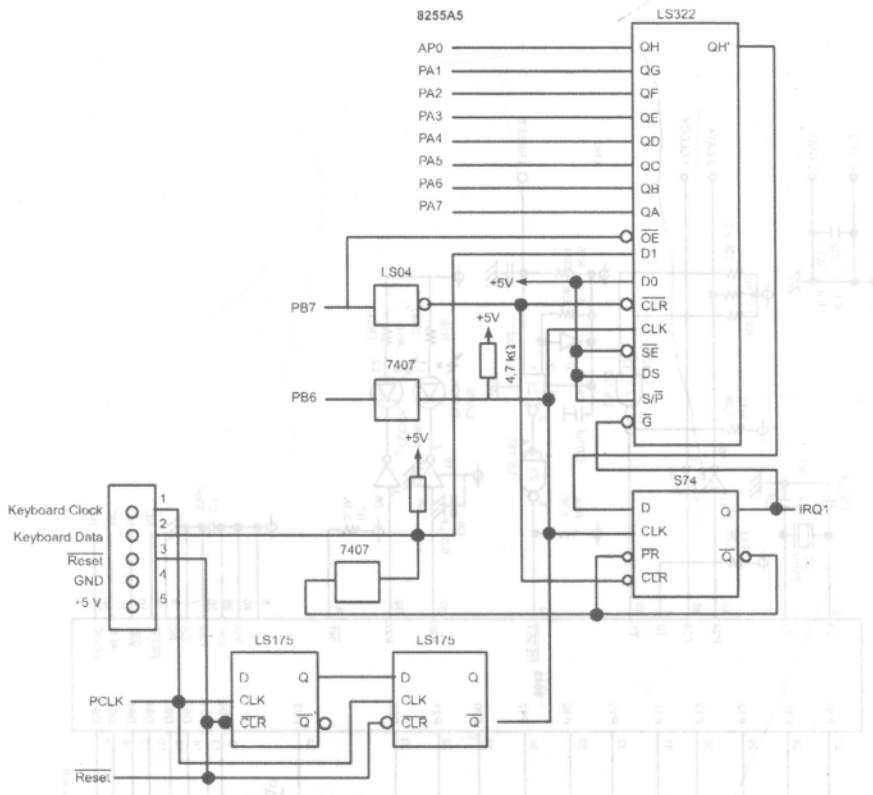
Hình A.3: Bảng mạch chính của IBM PC/XT 88. Kết nối các chíp điều khiển: 8255PPI 8252 Timer, 823DMA, 8259APIC, giao tiếp bàn phím, loa, sơ đồ 3/4



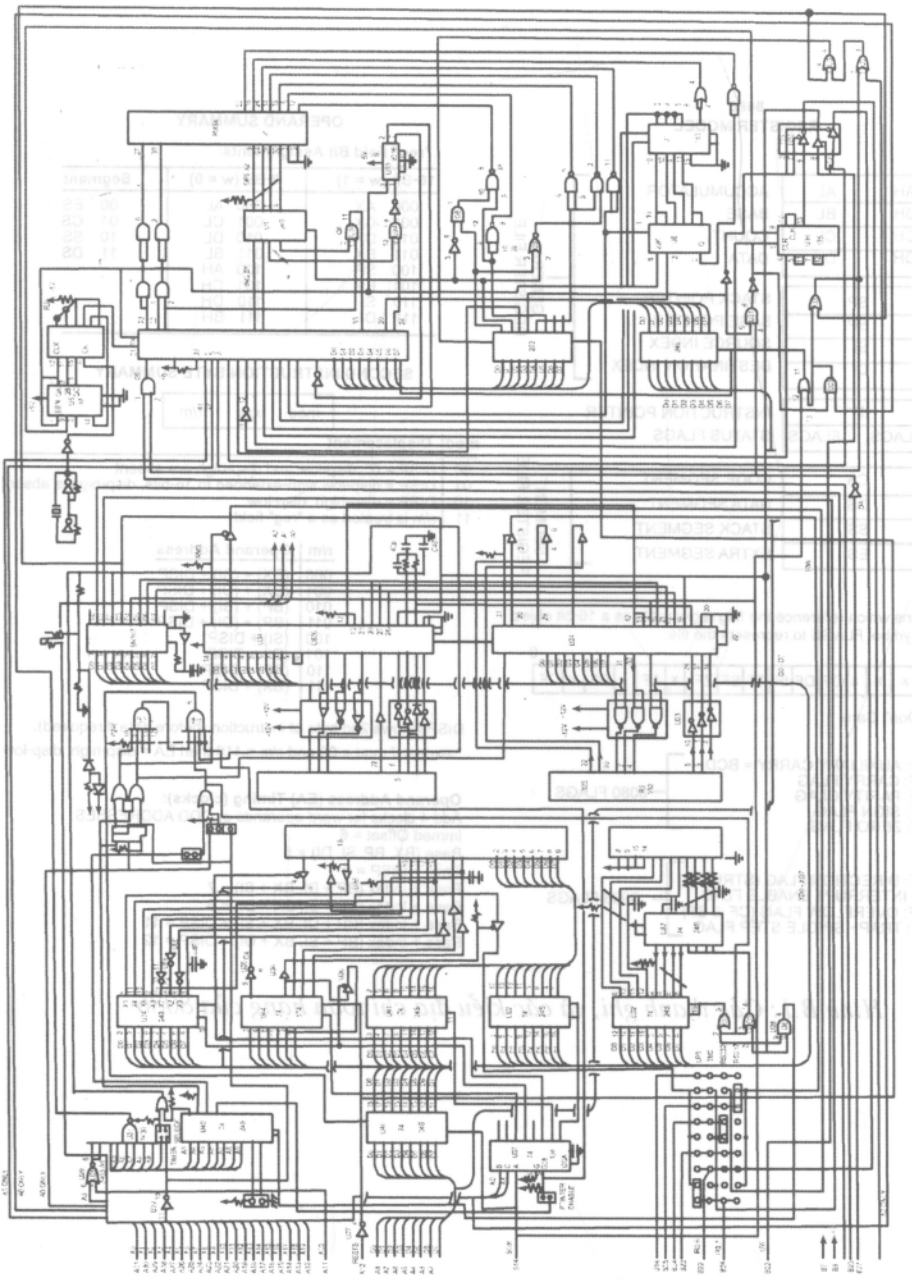
Hình A.4: Bảng mạch chính của IBM PC/XT 88 (kết nối RAM), sơ đồ 4/4



Hình A.5: Kết nối vi xử lý 8048 bên trong bàn phím của IBM PC/XT

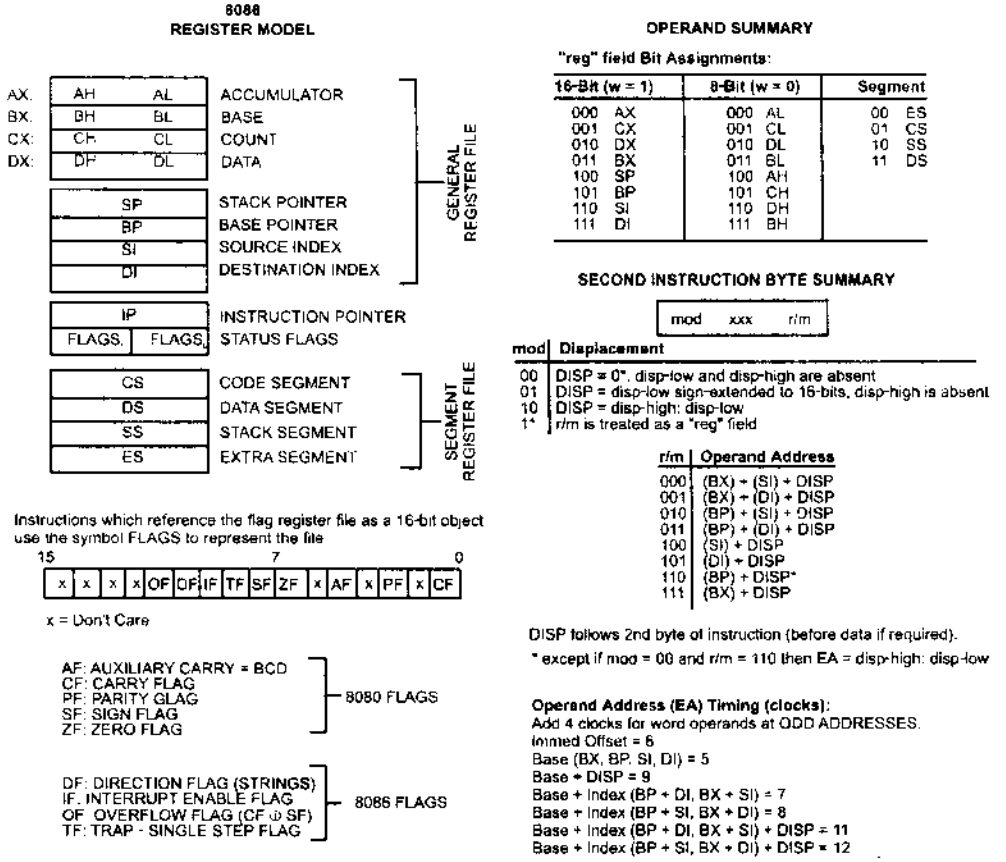


Hình A.6: Khối logic trên bảng hệ thống kết nối với bàn phím



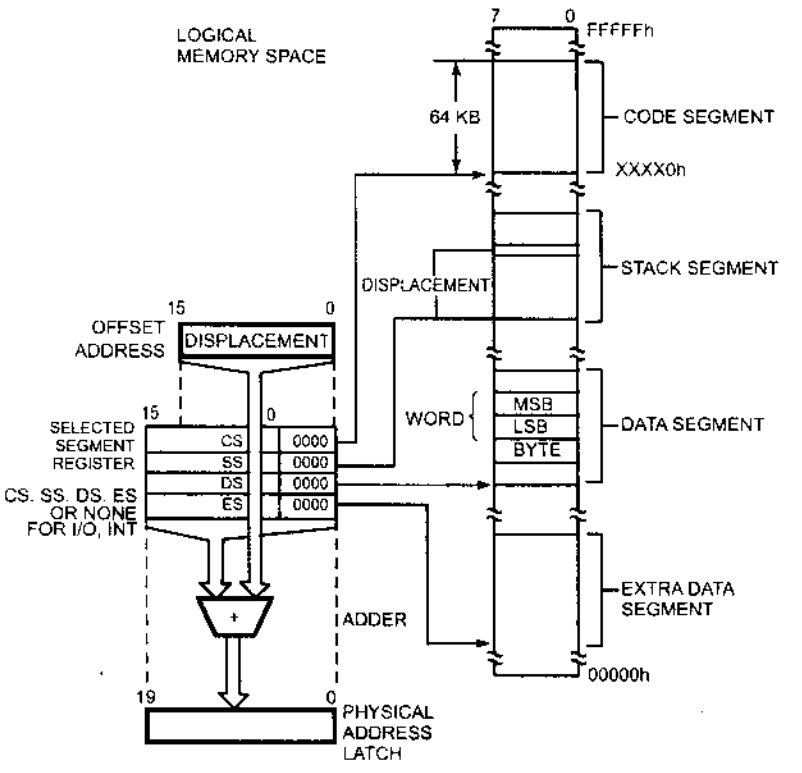
Hình A.7: Bảng điều khiển đĩa và vào/ra đa năng (FDC764, 8520 UART, và kết nối máy in) của IBM PC/XT

Phụ lục B: TẬP LỆNH INTEL 8086



Hình B.1: Các thanh ghi, và các kiểu địa chỉ toán hạng của 8086

MEMORY SEGMENTATION MODEL



SEMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

Timing: 2 clocks

USE OF SEGMENT OVERRIDE

Operand Register	Default	With Override Prefix
IP (code address)	CS	Never
SP (stack address)	SS	Never
BP (stack address or stack marker)	SS	BP + DS or ES, or CS
SI or DI (not inct, strings)	DS	ES, SS, or CS
SI (implicit source addr for strings)	DS	ES, SS, or CS
DI (implicit dest addr for strings)	ES	Never

Hình B.2a): Phân đoạn bộ nhớ của 8086

DATA TRANSFER

MOV = Move

Register/memory to/from register

1 0 0 0 1 0 d w	mod reg r/m

Timing (clocks). register to register 2
memory to register 8+EA
register to memory 9+EA

Immediate to register/memory

1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1

Timing: 10+EA clocks

Immediate to register

1 0 1 1 w reg	data	data if w = 1

Timing: 10+EA clocks

Memory to accumulator

1 0 1 0 0 0 0 w	addr-low	addr-high

Timing: 10 clocks

Accumulator to memory

1 0 1 0 0 0 1 w	addr-low	addr-high

Timing: 10 clocks

Register/memory to segment register

1 0 0 0 1 1 1 0	mod 0 reg r/m

Timing: (clocks) register to register 2
memory to register 8+EA

Register/memory to segment register

1 0 0 0 1 1 1 0	mod 0 reg r/m

Timing: 10 clocks register to register 2
register to memory 9+EA

PUSH = Push

Register/memory

1 1 1 1 1 1 1 1	mod 1 1 0 r/m

Timing: (clocks) register 10
memory 16+EA

Register

0 1 0 1 0 reg

Timing: 10 clocks

Hình B.2b): Cấu trúc các lệnh vận chuyển dữ liệu của 8086

Segment register

0 0 0 reg 1 1 0

Timing: 10 clocks

POP = Pop

Register memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m

Timing (clocks): register 8
memory 17+EA

Register

0 1 0 1 1 reg

Timing: 8 clocks

Segment register

0 0 0 reg 1 1 1

Timing: 8 clocks

XCHG = Exchange

Register/memory with register

1 0 0 0 0 1 0 w	mod reg r/m

Timing (clocks) register with register 4
memory with register 17+EA

Register with accumulator

1 0 0 1 0 reg

Timing: 3 clocks

IN = Input to AL/AX from

Fixed port

1 1 1 0 1 1 0 w	port

Timing: 10 clocks

Variable port (DX)

1 1 1 0 1 1 0 w

Timing: 8 clocks

OUT = Output from AL/AX to

Fixed port

1 1 1 0 0 1 1 w	port

Timing: 10 clocks

Variable port (DX)

1 1 1 0 1 1 1 w

Timing: 8 clocks

XLAT = Translate byte to AL

1 1 0 1 0 1 1 1

Timing: 11 clocks

LEA = Load EA to register

1 0 0 0 1 1 0 1	mod reg r/m

Timing: 16+EA clocks

LDS = Load pointer to DS

1 1 0 0 0 1 0 1	mod reg r/m

Timing: 16+EA clocks

LES = Load pointer to ES

1 1 0 0 0 1 0 0	mod reg r/m

Timing: 16+EA clocks

LAHF = Load AH with flags

1 0 0 1 1 1 1 1

Timing: 4 clocks

SAHF = Store AH into flags

1 0 0 1 1 1 1 0

Timing: 4 clocks

PUSHF = Push flags

1 0 0 1 1 1 0 0

Timing: 10 clocks

POPF = Pop flags

1 0 0 1 1 1 0 1

Timing: 8 clocks

ARITHMETIC

ADD = Add

Reg./memory with register to either

0 0 0 0 0 0 d w	mod reg r/m

Timing (clocks): register to register 3
memory to register 9+EA
register to memory 16+EA

Immediate to register/memory

1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01

Timing (clocks): immediate to register 4
immediate to memory 17+EA

Hình B.3: Cấu trúc các lệnh vận chuyển dữ liệu, số học của 8086

Immediate to accumulator

0 0 0 0 0 1 0 w	cata	data if w = 1

Timing: 4 clocks

ADC = Add with carry
Reg./memory with register to either

0 0 0 1 0 0 d w	mod reg r/m

Timing (clocks): register to register 3
memory to register 9+EA
register to memory 16+EA

Immediate to register/memory

1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=0'

Timing (clocks): immediate to register 4
immediate to memory 17+EA

Immediate to accumulator

0 0 0 1 0 1 0 w	data	data if w=1

Timing: 4 clocks

INC = Increment
Register/memory

1 1 1 1 1 1 1 w	mod 0 0 0 r/m

Timing (clocks): register 2
memory 15+EA

Register

0 1 0 0 0 reg

Timing: 2 clocks

AAA = ASCII adjust for add

0 0 1 1 0 1 1 1

Timing: 4 clocks

DAA = Decimal adjust for add

0 0 1 0 0 1 1 1

Timing. 4 clocks

SUB = Subtract
Reg./memory and register to either

0 0 1 0 1 0 d w	mod reg r/m

Timing (clocks) register from register 3
memory from register 9+EA
register from memory 16+EA

Immediate from register/memory

1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=0'

Timing (clocks). immediate from register 4
immediate from memory 17+EA

Immediate from accumulator

0 0 1 0 1 1 0 w	data	data if w=1

Timing: 4 clocks

SBB = Subtract with borrow
Reg./memory and register to either

0 0 0 1 1 0 d w	mod reg r/m

Timing (clocks): register from register 3
memory from register 9+EA
register from memory 16+EA

Immediate from register/memory

1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s w=01

Timing (clocks): immediate from register 4
immediate from memory 17+EA

Immediate from accumulator

0 0 0 1 1 1 0 w	data	data if w=1

Timing: 4 clocks

DEC = Decrement
Register/memory

1 1 1 1 1 1 1 w	mod 0 0 1 r/m

Timing (clocks): register 2
memory 15+EA

Register

0 1 0 0 1 reg

Timing: 2clocks

NEG = Change sign

1 1 1 1 0 1 1 w	mod 0 1 1 r/m

Timing (clocks): register 3
memory 16+EA

CMP = Compare
Register/memory and register

0 0 1 1 1 0 d w	mod reg r/m

Timing (clocks): register with register 3
memory with register 9+EA
register with memory 9+EA

Immediate with register/memory

1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01

Timing (clocks): immediate with register 4
immediate with memory 17+EA

Immediate with accumulator

0 0 1 1 1 1 0 w	data	data if w=1

Timing. 4 clocks

AAS = ASCII adjust for subtract

0 0 1 1 1 1 1 1

Timing: 4 clocks

DAS = Decimal adjust for subtract

0 0 1 0 1 1 1 1

Timing: 4 clocks

IMUL = Multiply (unsigned)

1 1 1 1 0 1 1 w	mod 1 0 0 r/m

Timing (clocks): 8-bit 71+EA
16-bit 124+EA

IMUL = Integer mulliply (signed)

1 1 1 1 0 1 1 w	mod 1 0 1 r/m

Timing (clocks). 8-bit 90+EA
16-bit 144+EA

AAM = ASCII adjust for multiply

1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0

Timing: 83 clocks

DIV = Divide (unsigned)

1 1 1 1 0 1 1 w	mod 1 1 0 r/m

Timing (clocks): 8-bit 90+EA
16-bit 155+EA

IDIV = Integer divide (signed)

1 1 1 1 0 1 1 w	mod 1 1 1 r/m

Timing (clocks) 8-bit 112+EA
16-bit 177+EA

AAD = ASCII adjust for divide

1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0

Timing: 60 clocks

Hình B.4: Cấu trúc các lệnh số học của 8086

CBW = Conver byte to word

1 0 0 1 1 0 0 0

Timing: 2 clocks

CWD = Conver word to double word

1 0 0 1 1 0 0 1

Timing: 5 clocks

LOGIC

NOT = Invert

1 1 1 1 0 1 1 w	mod 0 1 0 r/m

Timing (clocks):		
	register	3
	memory	16+EA

SHL/SAL = Shift logical/arithmetic left

1 1 0 1 0 0 v w	mod 1 0 0 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

SHR = Shift logical right

1 1 0 1 0 0 v w	mod 1 0 1 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

SAR = Shift arithmetic right

1 1 0 1 0 0 v w	mod 1 1 1 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

ROL = Rotate left

1 1 0 1 0 0 v w	mod 0 0 0 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

ROR = Rotate right

1 1 0 1 0 0 v w	mod 0 0 1 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

RCL = Rotate through carry left

1 1 0 1 0 0 v w	mod 0 1 0 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

RCR = Rotate through carry right

1 1 0 1 0 0 v w	mod 0 1 1 r/m

Timing (clocks):		
	single-bit register	2
	single-bit memory	15+EA
	variable-bit register	8+4/bit
	variable-bit memory	20+EA+4/bit

AND = And
Reg./memory and register to either

0 0 1 0 0 0 d w	mod reg r/m

Timing (clocks):		
	register to register	2
	memory to register	9+EA
	register to memory	16+EA

Immediate to register/memory

1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1

Timing (clocks):		
	immediate to register	4
	immediate to memory	17+EA

Immediate to accumulator

0 0 1 0 0 1 0 w	data	data if w=1

Timing: 4 clocks

TEST = And function to flags, no result
Register/memory and register

1 0 0 0 0 1 0 w	mod reg r/m

Timing (clocks):		
	register to register	3
	register with memory	9+EA

Immediate data and register/memory

1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1

Timing (clocks):		
	immediate with register	4
	immediate with memory	10+EA

Immediate data and accumulator

1 0 1 0 1 0 0 w	data	data if w=1

Timing: 4 clocks

OR = Or
Reg./memory and register to either

0 0 0 0 1 0 d w	mod reg r/m

Timing (clocks):		
	register to register	3
	memory to register	9+EA
	register to memory	16+EA

Immediate to register/memory

1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1

Timing (clocks):		
	immediate to register	4
	immediate to memory	17+EA

Immediate to accumulator

0 0 0 0 1 1 0 w	data	data if w=1

Timing: 4 clocks

XOR - Exclusive or
Reg./memory and register to either

0 0 1 1 0 0 d w	mod reg r/m

Timing (clocks):		
	register to register	3
	memory to register	9+EA
	register to memory	16+EA

Immediate to register/memory

1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1

Timing (clocks):		
	immediate to register	4
	immediate to memory	17+EA

Immediate to accumulator

0 0 1 1 0 1 0 w	data	data if w=1

Timing: 4 clocks

STRING MANIPULATION

REP = Repeat

1 1 1 1 0 0 1 z

Timing: 6 clocks/loop

Hình B.5: Cấu trúc các lệnh logic và dịch của 8086

MOVS = Move String

1 0 1 0 0 1 0 w

Timing: 17 clocks

CMPS = Compare String

1 0 1 0 0 1 1 w

Timing: 22 clocks

SCAS = Scan String

1 0 1 0 1 1 1 w

Timing: 15 clocks

LODS = Load String

1 0 1 0 1 1 0 w

Timing: 12 clocks

STOS = Store String

1 0 1 0 1 0 1 w

Timing: 10 clocks

CONTROL TRANSFER

NOTE: Queue reinitialization is not included in the timing information for transfer operations. To account for instruction loading, add 8 clcks to timing numbers.

CALL = Call

Direct within segment

1 1 1 0 1 0 0 0	disp-low	disp-high

Timing: 11 clocks

Indirect within segment

1 1 1 1 1 1 1 1	mod 0 1 0 r/m

Timing: 13+EA clocks

Direct intersegment

1 0 0 1 1 0 1 0	offset-low	offset-high
	seg-low	seg-high

Timing: 20 clocks

Indirect intersegment

1 1 1 1 1 1 1 1	mod 0 1 1 r/m

Timing: 29+EA clocks

JMP = Unconditional Jump

Direct within segment

1 1 1 0 1 0 0 1	disp-low	disp-high

Timing: 7 clocks

Direct within segment-short

1 1 1 0 1 0 1 1	disp

Timing: 7 clocks

Indirect within segment

1 1 1 1 1 1 1 1	mod 1 0 0 r/m

Timing: 7+EA clocks

Direct intersegment

1 1 1 0 1 0 1 0	offset-low	offset-high
	seg-low	seg-high

Timing: 7 clocks

Indirect intersegment

1 1 1 1 1 1 1 1	mod 1 0 1 r/m

Timing: 16+EA clocks

RET = Return from CALL

Within segment

1 1 0 0 0 0 1 1

Timing: 8 clocks

Within seg. adding immed to SP

1 1 0 0 0 0 1 0	data-low	data-high

Timing: 12 clocks

Intersegment

1 1 0 0 1 0 1 1

Timing: 18 clocks

Intersegment, adding immediate to SP

1 1 0 0 1 0 1 0	data-low	data-high

Timing: 17 clocks

JE/JZ = Jump on equal/zero

0 1 1 1 0 1 0 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JL/JNGE = Jump on less/not greater or equal

0 1 1 1 1 1 0 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JLE/JNG = Jump on less or equal/not greater

0 1 1 1 1 1 1 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JB/JNAE = Jump on below/not above or equal

0 1 1 1 0 0 1 0	disp

Timing (clocks) Jump is taken 8; Jump is not taken 4

JBA/JNA = Jump on below or equal/not above

0 1 1 1 0 1 1 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JP/JPE = Jump on parity/parity even

0 1 1 1 0 1 0 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JO = Jump on overflow

0 1 1 1 0 0 0 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JS = Jump on sign

0 1 1 1 1 0 0 0	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JNE/JNZ = Jump on not equal/not zero

0 1 1 1 0 1 0 1	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

JNL/JGE = Jump on not less/greater or equal

0 1 1 1 0 1 0 1	disp

Timing (clocks): Jump is taken 8; Jump is not taken 4

Hình B.6: Cấu trúc các lệnh chuyển điều khiển của 8086

JNLE/JG = Jump on not les or equal/greater

0 1 1 1 1 1 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

JNB/JAE = Jump on not below/above or equal

0 1 1 1 0 0 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

JNBE/JA = Jump on not below or equal/above

0 1 1 1 0 1 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

JNP/JPO = Jump on not parity/parity odd

0 1 1 1 1 0 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

JNO = Jump on not overflow

0 1 1 1 0 0 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

JNS = Jump on not sign

0 1 1 1 1 0 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

LOOP = Loop CX times

1 1 1 0 0 0 1 0	disp

Timing (clocks): Jump is taken 9
Jump is not taken 5

LOOPZ/LOOPE = Loop while zero/equal

1 1 1 0 0 0 0 1	disp

Timing (clocks): Jump is taken 11
Jump is not taken 5

LOOPNZ/LOOPNE = Loop while not zero/not equal

1 1 1 0 0 0 0 0	disp

Timing (clocks): Jump is taken 11
Jump is not taken 5

JCXZ = Jump on CX zero

1 1 1 0 0 0 1 1	disp

Timing (clocks) Jump is taken 9
Jump is not taken 5

8086 CONDITIONAL TRANSFER OPERATIONS

Instruction	Condition	Interpretation
JE or JZ	ZF = 1	"equal" or "zero"
JL or JNGE	(SF xor OF) = 1	"less" or "not greater or equal"
JLE or JNG	((SF xor OF) or ZF) = 1	"less or "equal" or "not greater"
JB or JNAE	CF = 1	"below" or "not above or equal"
JBE or JNA	(CF or ZF) = 1	"below or equal" or "not above"
JP or JPE	PF = 1	"parity" or "parity even"
JO	OF = 1	"overflow"
JS	SF = 1	"sign"
JNE or JNZ	ZF = 0	"not equal" or "not zero"
JNL or JGE	(SF xor OF) = 0	"not less" or "greater or equal"
JNLE or JG	((SF xor OF) or ZF) = 0	"not less or equal" or "greater"
JNB or JAE	CF = 0	"not below" or "above or equal"
JNBE or JA	(CF or ZF) = 0	"not below or equal" or "above"
JNP or JPO	PF = 0	"not parity" or "parity odd"
JNO	OF = 0	"not overflow"
JNS	SF = 0	"not sign"

"Above" and "below" refer to the relation between two unsigned values while "greater" and "less" refer to the relation between two signed values

INT = Interrupt
Type specified

1 1 0 0 1 1 0 1	type

Timing: 50 clocks
Type 3

1 1 0 0 1 1 0 0

Timing: 51 clocks

INTO = Interrupt on overflow

1 1 0 0 1 1 1 0

Timing 52 clocks if pass 4 clocks if fail

IRET = Interrupt return

1 1 0 0 1 1 1 1

Timing: 24 clocks

PROCESSOR CONTROL

CLC = Clear carry

1 1 1 1 1 0 0 0

Timing: 2 clocks

STC = Set carry

1 1 1 1 1 0 0 1

Timing: 2 clocks

CMC = Complement carry

1 1 1 1 0 1 0 1

Timing: 2 clocks

NOP = No operation

1 0 0 1 0 0 0 0

Timing 3 clocks

CLD = Clear direction

1 1 1 1 1 1 0 0

Timing: 2 clocks

STD = Set direction

1 1 1 1 1 1 0 1

Timing: 2 clocks

CLI = Clear interrupt

1 1 1 1 1 0 1 0

Timing: 2 clocks

STI = Set interrupt

1 1 1 1 1 0 1 1

Timing: 2 clocks

HLT = Halt

1 1 1 1 0 1 0 0

Timing: 2 clocks

WAIT = Wait

1 0 0 1 1 0 1 1

Timing. 3 clocks

LOCK = Bus lock prefix

1 1 1 1 0 0 0 0

Timing: 2 clocks

ESC = Escape (to external device)

mod xxx r/m

Timing: 7+EA clocks

Footnotes
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16 bits of immediate data form the operand
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1, if v = 1 then "count" in (CL)
x = don't care
z is used for some string primitives to compare with ZF FLAG
AL = 8-bit accumulator
AX 16-bit accumulator
CX = Count register
DS = Data segment
DX = Variable port register
ES = Extra segment
above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values

Hình B.7: Các lệnh điều khiển vận chuyển và ngắt của 8086

Hi \ Lo	0	1	2	3	4	5	6	7
0	ADD b.f.r/m	ADD w.f.r/m	ADD b.f.r/m	ADD w.f.r/m	ADD b.ia	ADD w.ia	PUSH ES	POP ES
1	ADC b.f.r/m	ADC w.f.r/m	ADC b.t.r/m	ADC w.t.r/m	ADC b.i	ADC w.i	PUSH SS	POP SS
2	AND b.f.r/m	AND w.f.r/m	AND b.t.r/m	AND w.t.r/m	AND b.i	AND w.i	SEG =ES	DAA
3	XOR b.f.r/m	XOR w.f.r/m	XOR b.t.r/m	XOR w.t.r/m	XOR b.i	XOR w.i	SEG =SS	AAA
4	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6								
7	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JAZ	JBE/ JNA	JNBE/ JA
8	Immed b.r/m	Immed w.r/m	Immed b.r/m	Immed is.r/m	TEST b.r/m	TEST w.r/m	XCHG b.r/m	XCHG w.r/m
9	NOP	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI
A	MOV m → AL	MOV m → AX	MOV AL → m	MOV AX → m	MOVS b	MOVS w	CMPS b	CMPS w
B	MOV i → AL	MOV i → CL	MOV i → DL	MOV i → BL	MOVS i → AH	MOVS i → CH	MOVS i → DH	MOVS i → BH
C			RET (i + SP)	RET	LES	LDS	MOV b.i.r/m	MOV w.i.r/m
D	shift b	shift w	shift b.v	shift w.v	AAM	AAD		XLAT
E	LOOPNZ/ LOOPNE	LOOPZ/ LOOPE	LOOP	JCXZ	IN b	IN w	OUT b	OUT w
F	LOCK		REP	REP z	HLT	CMC	Grp1 b.r/m	Grp1 w.r/m

Hi \ Lo	8	9	A	B	C	D	E	F
0	OR b.f.r/m	OR w.f.r/m	OR b.t.r/m	OR w.t.r/m	OR b.i	OR w.i	PUSH CS	
1	SBB b.f.r/m	SBB w.f.r/m	SBB b.t.r/m	SBB w.t.r/m	SBB b.i	SBB w.i	PUSH DS	POP DS
2	SUB b.f.r/m	SUB w.f.r/m	SUB b.t.r/m	SUB w.t.r/m	SUB b.i	SUB w.i	SEG CS	DAS
3	CMP b.f.r/m	CMP w.f.r/m	CMP b.t.r/m	CMP w.t.r/m	CMP b.i	CMP w.i	SEG DS	AAS
4	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5	POP AX	POP CX	POP DX	PUSH BX	POP SP	POP BP	POP SI	POP DI
6								
7	JS	JNS	JB/ JPE	JNB/ JP0	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG
8	MOV b.f.r/m	MOV w.f.r/m	MOV b.t.r/m	MOV w.t.r/m	MOV sr.t.r/m	LEA	MOV sr.f.r/m	POP r/m
9	CBW	CWD	CALL Id	WAIT	PUSHF	POPF	SAHF	LAHF
A	TEST b.i	TEST w.i	STOS b	STOS w	LODS b	LODS w	SCAS b	SCAS w
B	MOV i → AX	MOV i → CX	MOV i → DX	MOV i → BX	MOV i → SP	MOV i → BP	MOV i → SI	MOV i → DI
C			RET I(i + SP)	RET 1	INT type 3	INT (Any)	INTO	IRET
D	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
E	CALL d	JMP d	JMP i.d	JMP si.d	IN v.b	IN v.w	OUT v.b	OUT v.w
F	CLC	STC	CLI	STI	CLD	STD	Grp2 b.r/m	Grp2 w.r/m

b = byte operation
d = direct
f = from CPU reg
i = immediate
ia = immed to accum
id = indirect
is = immed. byte. sign ext.
l = long ie. intersegment
m = memory
r/m = EA is second byte
si = short intrasegment
sr = segment register
t = to CPU reg
v = variable
w = word operation
z = zero

where

mod.r/m	000	001	010	011	100	101	110	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR		SAR
Grp 1	TEST		NOT	NEG	MUL	IMUL	DIV	IDIV
Grp 2	INC	DEC	CALL id	CALL lid	CALL id	CALL lid	PUSH	—

Hình B.8: Bảng tổng hợp tập lệnh của 8086

Phụ lục C: TẬP LỆNH CỦA CÁC VI XỬ LÝ CÔNG NGHỆ CAO TIÊN TIẾN INTEL X86

Tập lệnh kiến trúc X86

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
AAA - ASCII Adjust after Addition	0011 0111	3	
AAD - ASCII Adjust AX before Division	1101 0101 : 0000 1010	10	
AAM - ASCII Adjust AX after Multiply	1101 0100 : 0000 1010	18	
AAS - ASCII Adjust AL after Subtraction	0011 1111	3	
ADC - ADD with Carry			
reg1 to reg2	0001 000w : 11 reg1 reg2	1	
reg2 to reg 1	0001 001w : 11 reg2 reg2	1	
memory to register	0001 001w : mod reg r/m	2	
register to memory	0001 000w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 010 reg : immediate data	1	
immediate to accumulator	0001 010w : immediate data	1	
immediate to memory	1000 00sw : mod 010 r/m : immediate data	3	U/L
ADD - Add			
reg1 to reg2	0000 000w : 11 reg1 reg2	1	
reg2 to reg 1	0000 001w : 11 reg1 reg2	1	
memory to register	0000 001w : mod reg r/m	2	
register to memory	0000 000w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 000 reg : immediate data	1	
immediate to accumulator	0000 010w : immediate data	1	
immediate to memory	1000 00sw : mod 000 r/m : immediate data	3	U/L
AND - Logical AND			
reg1 to reg2	0010 000w : 11 reg1 reg2	1	
reg2 to reg 1	0010 001w : 11 reg1 reg2	1	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
memory to register	0010 001w : mod reg r/m	2	
register to memory	0010 000w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 100 reg : immediate data	1	
immediate to accumulator	0010 010w : immediate data	1	
immediate to memory	1000 00sw : mod 100 r/m : immediate data	3	U/L
ARPL - Adjust RPL Field of Selector			
from register	0110 0011 : 11 reg1 reg2	7	
from memory	0110 0011 : mod reg/m	7	
BOUND - Check Array Against	0110 0010 : mod reg r/m		
Bounds			
if within bounds		8	
if out of bounds		INT + 32	21
BSF - Bit Scan Forward			
reg1, reg2	0000 1111 : 1011 1100 : 11 reg2 reg1		
word		6-34	MN/MX, 12
doubleword		6-42	MN/MX, 12
memory, reg	0000 1111 : 1011 1100 : mod reg r/m		
word		6-35	MN/MX, 13
doubleword		6-43	MN/MX, 13
BSR - Bit Scan Reverse			
reg1, reg2	0000 1111 : 1011 1101 : 11 reg2 reg1		
word		7-39	MN/MX, 14
doubleword		7-71	MN/MX, 14
memory, reg	0000 1111 : 1011 1101 : mod reg r/m		
word		7-40	MN/MX, 15
doubleword		7-72	MN/MX, 15
BSWAP - Byte Swap	0000 1111 : 1100 1 reg	1	
BT - Bit Test			
register, immediate	0000 1111 : 1011 1010 : 11 100 reg : imm8	4	
data			
memory, immediate	000 1111 : 1011 1010 : mod 100 r/m	4	
imm8 data			

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
reg1, reg 2	0000 1111 : 1010 0011 : 11 reg2 reg1	4	
memory, reg	0000 1111 : 1010 0011: mod reg r/m	9	
BTC - Bit Test and Complement			
register, immediate data	0000 1111 : 1011 1010 : 11 111 reg: imm8	7	
memory, immediate imm8 data	0000 1111 : 1011 1010 : mod 111 r/m:	8	U/L
reg1, reg2	0000 1111 : 1011 1011 : 11 reg2 reg1	7	
memory, reg	0000 1111 : 1011 1011: mod reg r/m	13	U/L
BTR - Bit Test and Reset			
register, immediate data	0000 1111 : 1011 1010 : 11 110 reg: imm8	7	
memory, immediate imm8 data	0000 1111 : 1011 1010 : mod 110 r/m:	8	U/L
reg1, reg2	0000 1111 : 1011 0011 : 11 reg2 reg1	7	
memory, reg	0000 1111 : 1011 0011: mod reg r/m	13	U/L
BTS - Bit Test and Set			
register, immediate data	0000 1111 : 1011 1010 : 11 101 reg: imm8	7	U/L
memory, immediate imm8 data	0000 1111 : 1011 1010 : mod 101 r/m	8	U/L
reg1, reg2	0000 1111 : 1010 1011 : 11 reg2 reg1	7	
memory, reg	0000 1111 : 1011 1011: mod reg r/m	13	U/L
CALL - Call Procedure (in same segment)			
direct	1110 1000 : full displacement	1	23
register indirect	1111 1111 : 11 010 reg	2	23
memory indirect	1111 1111 : mod 010 r/m	2	23
CALL - Call Procedure (in other segment)			
direct	1001 1010 : unsigned full offset, selector	4	R, 23
to same level		4 - 13	P, 9, 23, 24
thru gate to same level		22	P, 9, 25
to inner level, no parameters		44	P, 9, 25

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
to inner level, x parameters (d)words		45+2x	P, 9, 25
to TSS		21+TS	P,10,9,25
thru task gate		22 + TS	P,10,9,25
indirect	1111 1111 : mod 011 r/m	5	R, 23
to same level		5 - 14	P,9,23,24
thru gate to same level		22	P, 9, 25
to inner level, no parameters		44	P, 9, 25
to inner level, x parameters (d)words		45+2x	P, 9, 25
to TSS		21+TS	P,10,9,25
thru task gate		22 + TS	P,10,9,25
CBW - Convert Byte to Word	1001 1000	3	
CWDE - Convert Word Doubleword			
CLC - Clear Carry Flag	1111 1000	2	
CLD - Clear Direction Flag	1111 1100	2	
CLI - Clear Interrupt Flag	1111 1010	7	
CLTS - Clear Task-Switched	0000 1111 : 0000 0110	10	
Flag in CR0			
CMC - Complement Carry Flag	1111 0101	2	
CMP - Compare Two Operands			
reg1 with reg 2	0011 100w : 11 reg1 reg2	1	
reg2 with reg1	0011 101w : 11 reg1 reg2	1	
memory with register	0011 100w : mod reg r/m	2	
register with memory	0011 101w : mod reg r/m	2	
immediate with register	1000 00sw : 11 111 reg : immediate data	1	
immediate with accumulator	0011 110w : immediate data	1	
immediate with memory	1000 00sw : mod 111 r/m	2	
CMPS/CMPSB/CMPSW/CMPSD	1010 011w	5	16
Compare String Operands			
CMPXCHG - Compare and Exchange			
reg1, reg2	0000 1111 : 1011 000w : 11 reg2 reg1	5	
memory, reg	0000 1111 : 1011 000w : mod reg r/m	6	U/L
CMPXCHG8B - Compare and			
Exchange 8 Bytes			
memory, reg	0000 11111 : 1100 0111 : mod reg r/m	10	U/L

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
CWD - Conver Word to Dword	1001 1001	2	
CDQ - Convert Dword to Qword			
DAA - Decima Adjust AL after	0010 0111	3	
Addition			
DAS - Decimal Adjust AL after	0010 1111	3	
Subtraction			
DEC - Decrement by 1			
reg	1111 111w : 11 001 reg	1	
or	0100 1 reg	1	
memory	1111 111w : mod 001 r/m	3	U/L
DIV - Unsigned Divide			
accumulator by register	1111 011w : mod 001 r/m		
divisor -	byte	17	
	word	25	
	doubleword	41	
accumulator by memory	1111 011w : mod 110 r/m		
divisor -	byte	17	
	word	25	
	doubleword	41	
ENTER - Make Stack Frame	1100 1000 : 16-bit displacement : 8-bit		
level (L)			
for Procedure Paramenters			
L = 0		11	
L = 1		15	
L > 1		15 + 2L	8
HLT - Halt	1111 0100		
IDIV - Signed Divide			
accumulator by register	1111 011w : 11 111 reg		
divisor -	byte	22	
	word	30	
	doubleword	46	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
accumulator by register	1111 011w : mod 111 r/m		
divisor -	byte	22	
	word	30	
	doubleword	46	
IMUL - Signed Multiply			
accumulator with register	1111 011w : 11 101 reg		
multiplier -	byte	11	
	word	11	
	doubleword	10	
accumulator with memory	1111 011w : mod 101 reg		
multiplier -	byte	11	
	word	11	
	doubleword	10	
reg1 with reg2	0000 1111 : 1010 1111 : 11 : reg1 reg 2		
multiplier -	byte	10	
	word	10	
	doubleword	10	
register with memory	0000 1111 : 1010 1111 : mod reg r/m		
multiplier -	byte	10	
	word	10	
	doubleword	10	
reg1 with imm, to reg2	0110 10s1 : 11 reg1 reg2 : immediate data		
multiplier -	byte	10	
	word	10	
	doubleword	10	
mem, with imm, to reg	0110 10s1 : mod reg r/m : immediate data		
multiplier -	byte	10	
	word	10	
	doubleword	10	
INC - Increment by 1			
reg	1111 111w : 11 000 reg	1	
or	0100 0 reg	1	
memory	1111 111w : mod 000 r/m	3	U/L

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
INT n - Interrupt Type n	1100 1101 : type	INT + 6	21, 25
INT - Single-Step Interrupt 3	1100 1100	INT + 5	21, 25
INTO - Interrupt 4 on Overflow	1100 1110		
taken		INT + 5	21, 25
not taken		4	21, 25
INVD - Invalidate Cache	0000 1111 : 0000 1000	15	
INVLPG - Invalidate TLB Entry	0000 1111 : 0000 0001 : mod 111 r/m	29	
IRET/IRETD - Interrupt Return	11100 1111		
real mode or virtual 8086 mode		7	R, 23
protected mode			
to same level		10-19	P,9,23,24
to outer level		27	P,9,25
to nester task		10 + TS	P,9,10,25
Jcc - Jump if Condition Is Met			
8-bit displacement	011 tttn : 8-bit displacement	1	23
full displacement	0000 1111 : 1000 tttn : full displacement	1	23
JCXZ/JECXZ - Jump on	1110 0011 : 8-bit displacement	6/5	T/NT, 23
CX/ECX Zero			
address size prefix differentiates JCXZ from JECXZ			
JMP - Uniconditional Jump (to same segment)			
short	1110 1011 : 8-bit displacement	1	23
direct	1110 1001 : full displacement	1	23
register indirect	1111 1111 : 11 100 reg	2	23
memory indirect	1111 1111 : mod 100 r/m	2	23
JMP - Uniconditional Jump (to other segment)			
direct intersegment	1110 1010 : unsigned full offset, selector	3	R, 23
to same level		3-12	P,9,23,24
thru call gate to same level		18	P,9,25
thru TSS		19 + TS	P,10,9,25
thru task gate		20+ TS	P,10,9,25
indirect intersegment	1111 1111 : mod 100 r/m	4	R,23
to same level		4→13	P,9,23,24

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
thru call gate ro same level		18	P,9,25
thru TSS		19 + TS	P,10,9,25
thru task gate		20 + TS	P,10,9,25
LAHF - Load flags into AH Register	1001 1111	2	
LAR - Load Access Rights Byte			
from register	0000 1111 : 0000 0010 : 11 reg1 reg2	8	
from memory	0000 1111 : 0000 0010 : mod reg r/m	8	
LES - Load Pointer to DS	1100 0101 : mod reg r/m	4 →13	9,24
LEA - Load Effective Address	1000 1101 : mod reg r/m	1	
LEAVE - High Level Procedure Exit	1100 1001	3	
LES - Load Pointer to ES	1100 0100 : mod reg r/m	4→13	9,24
LFS - Load Pointer to FS	0000 1111 : 1011 0100 : mod reg r/m	4→13	9,24
LGDT - Load Global Descriptor Table Register	0000 1111 : 0000 0001 : mod 010 r/m	6	
LGS - Load Pointer to GS	0000 1111 : 1011 0101 : mod reg r/m	4→13	9,24
LIDT - Load Interrupt Descriptor Table Register	0000 0001 : mod 011 r/m	6	
LLDT - Load Loca Descriptor Table Register			
LDTR from register	0000 1111 : 0000 0000 : 11 010 reg	9	
LDTR from memory	0000 1111 : 0000 0000 : mod 010 r/m	9	
LMSW - Load Machine Status Word			
from register	0000 1111 : 0000 0001 : 11 110 reg	8	
from memory	0000 1111 : 0000 0001 : mod 110 r/m	8	
LOCK - Assert LOCK# Signal Prefix	1111 0000	1	
LODS/LODSB/LODSW/LODSD - Load String Operand	1010 110w	2	
LOOP - Loop Count	1110 0010 : 8-bit displacement		
LOOPZ/LOOPE - Loop Count while Zero/Equal	1110 0001 : 8-bit displacement	7/8	L/NL,23

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
LSL - Load Segment Limit			
from register	0000 1111 : 0000 0011 : 11 reg1 reg2	8	
from memory	0000 1111 : 0000 0011 : mod reg r/m	8	
LSS - Load Pointer to SS	0000 1111 : 1011 0010 : mod reg r/m	4→13 8→17	RV/P,9,24
LTR - Load Task Register			
from register	0000 1111 : 0000 0000 : 11 011 reg	10	
from memory	0000 1111 : 0000 0000 : mod 011 r/m	10	
MOV - Move Data			
reg1 to reg2	1000 100w : 11 reg1 reg 2	1	
reg2 to reg1	1000 101w : 11 reg1 reg 2	1	
memory to reg	1000 101w : mod reg r/m	1	
reg to memory	1000 100w : mod reg r/m	1	
immediate to reg	1100 011w : 11 000 reg: immediate data	1	
or	1011 w reg: immediate data	1	
immediate to memory	1100 011w : mod 000 r/m: immediate data	1	
memory to accumulator	1010 000w : full displacement	1	
accumulator to memory	1010 001w : full displacement	1	
MOV - Move to/from Control Registers			
CR0 from register	0000 1111 : 0010 0010 : 11 000 reg	22	
CR2 from register	0000 1111 : 0010 0010 : 11 010 reg	12	
CR3 from register	0000 1111 : 0010 0010 : 11 011 reg	21	
CR4 from register	0000 1111 : 0010 0010 : 11 100 reg	14	
register from CR0-4	0000 1111 : 0010 0000 : 11 eee reg	4	
MOV - Move to/from Debug Registers			
CR0-3 from register	0000 1111 : 0010 0011 : 11 eee reg	11	
CR4-5 from register	0000 1111 : 0010 0011 : 11 eee reg	12	
CR6-7 from register	0000 1111 : 0010 0011 : 11 eee reg	11	
register from CR6-7	0000 1111 : 0010 0001 : 11 eee reg	11	
register from CR4-5	0000 1111 : 0010 0001 : 11 eee reg	12	
register from CR0-3	0000 1111 : 0010 0001 : 11 eee reg	2	
MOV - Move to/from Segment Registers			
reg to segment reg	1000 1110: 11 sreg3 reg	2→11	9,24
reg to SS	1000 1110: 11 sreg3 reg	2→11 8→17	RV/P,9,24

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
memory to segment reg	1000 1110: mod sreg3 r/m	3	9,24
memory to SS	1000 1110: mod sreg3 r/m	3→12 8→17	RV/P,9,24
segment reg to reg	1000 1110: 11 sreg3 reg	1	
segment reg to memory	1000 1110: mod sreg3 r/m	1	
MOVS/MOVSB/MOVSW	1010 010w	4	16
MOVSD - Move Data from String to String			
MOVSX - Move with Sign-Extend			
reg2 to reg1	0000 1111 : 1011 111w : 11 reg1 reg 2	3	
memory to reg	1000 1111 : 1011 111w : mod reg r/m	3	
MOVZX - Move with Zero-Extend			
reg2 to reg1	0000 1111 : 1011 011w : 11 reg1 reg 2	3	
memory to reg	0000 1111 : 1011 011w : mod reg r/m	3	
MUL - Unsigned Multiplication of AL or AX			
accumulator with register	1111 011w : 11 100 reg		
multiplier -	byte	11	
	word	11	
	doubleword	10	
accumulator with memory	1111 011w : mod 11 100 reg		
multiplier -	byte	11	
	word	11	
	doubleword	10	
NEG - Two's Complement Negation			
reg	1111 011w : 11 011 reg	1	
memory	1111 011w : mod 011 r/m	3	
NOP - No Operation	1001 0000	1	
NOT - One's Complement Negation			
reg	1111 011w : 11 010 reg	1	
memory	1111 011w : mod 010 r/m	3	U/L
OR - Logical Inclusive OR			
reg1 to reg2	0000 100w : 11 reg1 reg 2	1	
reg2 to reg1	0000 101w : 11 reg1 reg 2	1	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
memory to register	0000 101w : mod reg r/m	2	
register to memory	0000 100w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 001 reg : immediate data	1	
immediate to accumulator	0000 110w : immediate data	1	
immediate to memory	1000 00sw : mod 001 r/m : immediate data	3	U/L
POP - Pop a Word from the Stack			
reg	1000 1111 : 11 000 reg	1	
or	0101 1 reg	1	
memory	1000 1111 : mod 000 r/m	3	1
POP - Pop a Segment Register from the Stack			
segment reg CS, DS, ES	000 sreg2 111	3→12	9,24
segment reg SS	000 sreg2 111	3→12 8→17	R/V/P,9,24
segment reg FS, GS	000 1111 : 10 sreg3 001	3→12	9,24
POPA/POPAD - Pop All General Registers	0110 0001	5	
POPF/POPFD - Pop Stack into FLAGS of EFLAGS Register	1001 1101	4/14	RV/P
PUSH - Push Operand onto the Stack			
reg	1111 1111 : 11 110 reg	1	
or	0101 0 reg	1	
memory	1111 1111 : mod 110 r/m	2	1
immediate	0110 10s0 : immediate data	1	
PUSH - Push Segment Register onto the Stack			
segment reg CS, DS, ES, SS	000 sreg2 110	1	
segment reg FS, GS	0000 1111 : 10 sreg3 000	1	
PUSHA/PHSHAD - Push All General Registers	0110 0000	5	
PUSHA/PUSHAFD - Push Flags Registers onto the Stack	10001 1100	3/9	RV/P
RCL -Rotate thru Carry Left			
reg by 1	1101 000w : 11 010 reg	1	
memory by 1	1101 000w : mod 010 r/m	3	
reg by CL	1101 001w : 11 010 reg	7→24	MN/MX,4

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
memory by CL	1101 001w : mod 010 r/m	9→26	MN/MX,5
reg by immediate count	1100 000w : 11 010 reg : imm8 data	8→25	MN/MX,4
memory by immediate count	1100 000w : mod 010 r/m: imm8 data	10→27	MN/MX,5
RCR - Rotate thru Carry Right			
reg by 1	1101 000w : 11 011 reg	1	
memory by 1	1101 000w : mod 011 r/m	3	
reg by CL	1101 001w : 11 011 reg	7→24	MN/MX,4
memory by CL	1101 001w : mod 011 r/m	9→26	MN/MX,5
reg by immediate count	1100 000w : 11 011 reg : imm8 data	8→25	MN/MX,4
memory by immediate count	1100 000w : mod 011 r/m: imm8 data	10→27	MN/MX,5
RDMSR - REad from Model - Specific Register	0000 1111 : 0011 0010	20→24	MN/MX
REP LODS - Load String	1111 0011 : 1010 110w		
C = 0		7	
C > 0		7 + 3c	16
REP MOVS - Move String	1111 0011 : 1010 010w		
C = 0		6	
C = 1		13	16
C > 1		13 + c	16
REP STDS - Store String	1111 0011 : 1010 101w		
C = 0		6	
C > 0		9 + c	
REP MCPS - Compare String (Find Non-Match)	1111 0011 : 1010 011w		
C = 0		7	
C > 0		8 + 4c	16
REPE SCAS - Scan String (FInd Non-AL/AX/EAX)	1111 0011 : 1010 111w		
C = 0		7	
C > 0		8 + 4c	16

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
REPNE CMPS - Compare	1111 0010 : 1010 011w		
String (Find Match)			
C = 0		7	
C > 0		9 + 4c	16
REPNE SCAS - Scan String	1111 0010 : 1010 111w		
(Find AL/AX/EAX)			
C = 0		7	
C > 0		8 + 4c	16
RET - Return from Procedure (to same segment)			
	1100 0011	2	
adding immediate to SP	1100 0010 : 16-bit displacement	3	
RET - Return from Procedure (to other segment)			
inter segment	1100 1011	4	R,23
to same level		4→13	P,9,23,24
to outer level		23	P,9,25
adding immediate to SP	1100 1010 : 16-bit dispiacement	4	R,23
to same level		4→13	P,9,23,24
to outer level		23	P,9,25
ROL - Rotate (not thru Carry) Left			
reg by 1	1101 000w : 11 000 reg	1	
memory by 1	1101 000w : mod 000 r/m	3	
reg by CL	1101 001w : 11 000 reg	4	
memory by CL	1101 001w : mod 000 r/m	4	
reg by immediate count	1100 000w : 11 000 reg : imm8 data	1	
memory by immediate count	1100 000w : mod 000 r/m: imm8 data	3	
ROR - Rotate (not thru Carry) Right			
reg by 1	1101 000w : 11 001 reg	1	
memory by 1	1101 000w : mod 001 r/m	3	
reg by CL	1101 001w : 11 001 reg	4	
memory by CL	1101 001w : mod 001 r/m	4	
reg by immediate count	1100 000w : 11 001 reg : imm8 data	1	
memory by immediate count	1100 000w : mod 001 r/m: imm8 data	3	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
RSM - Resume from System Management Mode	0000 1111 : 1010 1010		
SAHF - Store AH Into Flags	1001 1110	2	
SAL - Shift Arithmetic Left	same instruction as SHL		
SAR - Shift Arithmetic Right			
reg by 1	1101 000w : 11 111 reg	1	
memory by 1	1101 000w : mod 111 r/m	3	
reg by CL	1101 001w : 11 111 reg	4	
memory by CL	1101 001w : mod 111 r/m	4	
reg by immediate count	1100 000w : 11 111 reg : imm8 data	1	
memory by immediate count	1100 000w : mod 111 r/m: imm8 data	3	
SBB - Integer Subtraction with Borrow			
reg1 to reg2	0001 100w : 11 reg1 reg 2	1	
reg2 to reg1	0001 101w : 11 reg1 reg 2	1	
memory to register	0001 101w : mod reg r/m	2	
register to memory	0001 100w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 011 reg : immediate data	1	
immediate to accumulator	0001 110w : immediate data	1	
immediate to memory	1000 00sw : mod 011 r/m : immediate data	3	U/L
SCAS/SCASB/SCASW/SCASD - Scan String	1101 111w	4	
SETcc - Byte Set on Condition			
reg	0000 1111 : 10 01 tttn : 11 000 reg	1	
memory	0000 1111 : 1001 tttn : mod 000 r/m	2	
SGDT - Store Global Descriptor Table Register	0000 1111 : 0000 0001 : mod 000 r/m	4	
SHL - Shift Left			
reg by 1	1101 000w : 11 100 reg	1	
memory by 1	1101 000w : mod 100 r/m	3	
reg by CL	1101 001w : 11 100 reg	4	
memory by CL	1101 001w : mod 100 r/m	4	
reg by immediate count	1100 000w : 11 100 reg : imm8 data	1	
memory by immediate count	1100 000w : mod 100 r/m: imm8 data	3	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
SHLD - Double Precision Shift Left			
register by immediate count imm8	0000 1111 : 1010 0100 : 11 reg2 reg1:	4	
memory by immediate count imm8	0000 1111 : 1010 0100 : mod reg r/m:	4	
register by CL	0000 1111 : 1010 01001 : 11 reg2 reg1	4	
memory by CL	0000 1111 : 1010 0100 : mod reg r/m	5	
SHR - Shift Right			
reg by 1	1101 000w : 11 101 reg	1	
memory by 1	1101 000w : mod 101 r/m	3	
reg by CL	1101 001w : 11 101 reg	4	
memory by CL	1101 001w : mod 101 r/m	4	
reg by immediate count	1100 000w : 11 101 reg : imm8 data	1	
memory by immediate count	1100 000w : mod 101 r/m: imm8 data	3	
SHRD - Double Precision Shift Right			
register by immediate count imm8	0000 1111 : 1010 1100 : 11 reg2 reg1:	4	
memory by immediate count imm8	0000 1111 : 1010 1100 : mod reg r/m:	4	
register by CL	0000 1111 : 1010 1101 : 11 reg2 reg1	4	
memory by CL	0000 1111 : 1010 1101 : mod reg r/m	5	
SIDT - Store Interrupt Descriptor Table Register	0000 1111 : 0000 0001 : mod 001 r/m	4	
SLDT - Store Local Descriptor Table Register			
to register	0000 1111 : 0000 0000 : 11 000 reg	4	
to memory	0000 1111 : 0000 0000 : mod 000 r/m	4	
SMSW - Store Machine Status Word			
to register	0000 1111 : 0000 0001 : 11 110 reg	2	
to memory	0000 1111 : 0000 0001 : mod 100 r/m	2	
STC - Set Carry Flag	1111 1001	2	
STD - Set Direction Flag	1111 1101	2	
STI - Set Interrupt Flag	1111 1011	7	

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
STOS/STOSB/STOSW/STOSD	1010 101w	3	
- Store String Data			
STR - Store Task Register			
to register	0000 1111 : 0000 0000 : 11 001 reg	2	
to memory	0000 1111 : 0000 0000 : mod 001 r/m	2	
SUB - Integer Subtraction			
reg1 to reg2	0010 100w : 11 reg1 reg 2	1	
reg2 to reg1	0010 101w : 11 reg1 reg 2	1	
memory to register	0010 101w : mod reg r/m	2	
register to memory	0010 100w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 101 reg : immediate data	1	
memory to accumulator	0010 110w : immediate data	1	
immediate to memory	1000 00sw : mod 101 r/m : immediate data	3	U/L
TEST - Logical Compare			
reg1 to reg2	1000 010w : 11 reg1 reg 2	2	
memory to register	1000 010w : mod reg r/m	1	
immediate to register	1111 011w : 11 000 reg : immediate data	1	
immediate and accumulator	1010 100w : immediate data	1	
immediate to memory	1111 011w : mod 000 r/m : immediate data	2	
VERR - Verify a Segment for Reading			
register	0000 1111 : 0000 0000 : 11 100 reg	7	
memory	0000 1111 : 0000 0000 : mod 100 r/m	7	
VERW - Verify a Segment for Writing			
register	0000 1111 : 0000 0000 : 11 101 reg	7	
memory	0000 1111 : 0000 0000 : mod 101 r/m	7	
WAIT - Wait	1001 1011	1/1	
WBINVD - Write-Back and	0000 1111 : 0000 1001	2000+	
Invalidate Data Cache			

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
WRMSR - Write to Model - Specific Register	0000 1111 : 0011 0000	30→45	MN/MX
XADD - Exchange and Add			
reg1, reg2	0000 1111 : 1100 000w : 11 100 reg2 reg1	3	
memory, reg	0000 1111 : 1100 000w : mod reg r/m	4	U/L
XCHG - Exchange Register/Memory with Register			
reg1 with reg2	1000 011w : 01 reg1 reg2	3	2
accumulator with reg	1001 0 reg	2	2
memory with reg	1000 011w : mod reg r/m	3	2
XLAT/XLAB - Table Look-up Translation	1101 0111	4	
XOR - Logical Exclusive OR			
reg1 to reg2	0011 000w : 11 reg1 reg2	1	
reg2 to reg1	0011 001w : 11 reg1 reg2	1	
memory to register	0011 001w : mod reg r/m	2	
register to memory	0011 001w : mod reg r/m	3	U/L
immediate to register	1000 00sw : 11 110 reg : immediate data	1	
immediate to accumulator	0011 010w : immediate data	1	
immediate to memory	1000 00sw : mod 110 r/m : immediate data	3	U/L
Prefix Byte			
address size	0110 0111	1	
LOCK	1111 0000	1	
operand size	0110 0110	1	
CS segment override	0010 1110	1	
DS segment override	0011 1110	1	
ES segment override	0010 0110	1	
FS segment override	0110 0100	1	
GS segment override	0110 0101	1	
SS segment override	0011 0110	1	
External Interrupt		INT + 14	21
NMI - Non-Maskable Interrupt		INT + 6	21
Page Fault		INT + 40	21

Lệnh	Khuôn dạng	Số nhịp đồng hồ	Ghi chú
Virtual 8086 Mode Exceptions			
CLI		INI + 9	21
STI		INI + 9	21
INT n		INI + 9	
PUSHF		INI + 9	21
POPF		INI + 9	21
IRET		INI + 9	
IN			
fixed port		INI + 34	21
variable port		INI + 34	21
OUT			
fixed port		INI + 34	21
variable port		INI + 34	21
INS		INI + 34	21
OUTS		INI + 34	21
REP INS		INI + 34	21
REP OUTS		INI + 34	21

Lệnh I/O

Câu lệnh	Định dạng	Chế độ thực	Chế độ bảo vệ CPL ≤ IOPL	Chế độ bảo vệ CPL1 OPL	Chế độ 8086 ảo	Ghi chú
IN - input from:						
fixed port	1110 010w : port	7	4	21	19	
number						
variable port	1110 110w	7	4	21	19	
OUT - Output to:						
fixed port	1110 011w : port	12	9	26	24	
number						
variable port	1110 111w	12	9	26	24	
INS - Input from DX Port	0110 110w	9	6	24	22	

Câu lệnh	Định dạng	Chế độ thực	Chế độ bảo vệ CPL ≤ IOPL	Chế độ bảo vệ CPL1 OPL	Chế độ 8086 ảo	Ghi chú
OUTS - Output to DX Port	0110 111w	13	10	27	25	1
REP INS - Input String	1111 0011 : 0110	11 + 3c	8 + 3c	25 + 3c	23 + 3c	2
110w						
REP OUTS - Output String	1111 0011 : 0110	13 + 4c	10 + 4c	27 + 4c	25 + 4c	3
111w						

Notes:

1. Two clock cache miss penalty in all cases.
2. c = count in CX or ECX
3. Cache miss penalty in all modes: Add 2 clock every 16 bytes. Entire penalty on second operation.

Lệnh con trỏ động

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
F2XM1 - Compute $2^{ST(0)}-1$	11011 001 : 1111 000	13-57	
FABS - Absolute Value	11011 001 : 1110 0001	1/1	
FADD - Add			
ST(0) ← ST(0) + 32-bit memory	11011 000 : mod 000 r/m	3/1	
ST(0) ← ST(0) + 64-bit memory	11011 100 : mod 000 r/m	3/1	
ST(d) ← ST(0) + ST(i)	11011 d00 : 11 000 ST(i)	3/1	
FADDP - Add and Pop			
ST(0) ← ST(0) + ST(i)	11011 110 : 11 000 ST(i)	3/1	
FBLD - Load Binary Coded Decimal	11011 111 : mod 100 r/m	48-58	
FBSTP - Store Binary Coded Decimal and Pop	1100 111 : mod 110 r/m	148-154	
FCHS - Change Sign	11011 001 : 1110 0000	1/1	
FCLEX - Clear Exceptions	11011 011 : 1110 0010	9/9	2
FCOM - Compare Real			
32-bit memory	11011 000 : mod 010 r/m	4/1	
64-bit memory	11011 100 : mod 010 r/m	4/1	

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
ST(i)	11011 000 : 11 010 ST(i)	4/1	
FCOMPP - Compare Real and Pop Twice	11011 110 : 11 011 001	4/1	
FCOS - Cosine of ST(0)	11011 001 : 1111 1111	18-124	
DSECSTP - Decrement Stack-Top Pointer	11011 001 : 1111 0110	1/1	
FDIV - Divide			
ST(0) ← ST(0) - 32-bit memory	11011 000 : mod 111 r/m	39	1
ST(0) ← ST(0) - 64-bit memory	11011 100 : mod 111 r/m	39	1
ST(d) ← ST(0) - ST(i)	11011 d00 : 1111 R ST(i)	39	1
FDIVP - Divide and Pop			
ST(d) ← ST(0) + ST(i)	11011 110 : 1111 1 ST(i)	39	1
FDIVR - Reverse Divide			
ST(0) ← ST(0) + 32-bit memory	11011 000 : mod 111 r/m	39	1
ST(0) ← ST(0) + 64-bit memory	11011 100 : mod 111 r/m	39	1
ST(d) ← ST(0) + ST(i)	11011 d00 : 1111 R ST(i)	39	1
FDIVRP - Reverse Divide and Pop			
ST(d) ← ST(0) + ST(i)	11011 110 : 1111 0 ST(i)	39	1
FFREE - Free ST(I) Register	11011 101 : 1100 0 (ST(i)	1/1	
FIADD - Add Integer			
ST(0) ← ST(0) + 32-bit memory	11011 110 : mod 000 r/m	7/4	1
ST(0) ← ST(0) + 64-bit memory	11011 010 : mod 000 r/m	7/4	1
FICOM - Compare Integer	11011 d00 : 1111 R ST(i)	39	1
16-bit memory	11011 110 : mod 010 r/m	8/4	
32-bit memory	11011 010 : mod 010 r/m	8/4	
FICOMP - Compare Integer and Pop			
16-bit memory	11011 110 : mod 011 r/m	8/4	
32-bit memory	11011 010 : mod 011 r/m	8/4	
FIDIV			
ST(0) ← ST(0) + 16-bit memory	11011 110 : mod 110 r/m	42	1
ST(0) ← ST(0) + 64-bit memory	11011 010 : mod 110 r/m	42	1
FIDIVR			
ST(0) ← ST(0) + 16-bit memory	11011 110 : mod 111 r/m	42	1

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
ST(0) ← ST(0) + 32-bit memory	11011 010 : mod 111 r/m	42	1
FILD - Load Integer			
16-bit memory	11011 111 : mod 000 r/m	3/1	
32-bit memory	11011 011 : mod 000 r/m	3/1	
64-bit memory	11011 111 : mod 101 r/m	3/1	
FIMUL			
ST(0) ← ST(0) + 16-bit memory	11011 110 : mod 001 r/m	7/4	
ST(0) ← ST(0) + 32-bit memory	11011 010 : mod 001 r/m	7/4	
FINCSTP - Increement Stack Pointer	11011 001 : 1111 01111	1/1	
FINIT - Initialize Floating-Point Unit	11011 011 : 1110 0011	16/12	2
FIST - Store Integer			
16-bit memory	11011 111 : mod 010 r/m	6/6	
32-bit memory	11011 011 : mod 010 r/m	6/6	
FISTD - Store Integer and Pop			
16-bit memory	11011 111 : mod 011 r/m	6/6	
32-bit memory	11011 011 : mod 011 r/m	6/6	
64-bit memory	11011 111 : mod 111 r/m	6/6	
FISUB			
ST(0) ← ST(0) + 16-bit memory	11011 110 : mod 100 r/m	7/4	
ST(0) ← ST(0) + 32-bit memory	11011 010 : mod 100 r/m	7/4	
FISUBR			
ST(0) ← ST(0) + 16-bit memory	11011 110 : mod 101 r/m	7/4	
ST(0) ← ST(0) + 32-bit memory	11011 010 : mod 101 r/m	7/4	
FLD - Load Real			
32-bit memory	11011 001 : mod 000 r/m	1/1	
64-bit memory	11011 001 : mod 000 r/m	1/1	
80-bit memory	11011 011 : mod 101 r/m	3/3	
ST(i)	11011 001 : 11 000 ST(i)	1/1	
FLD1 - Load + 1.0 into ST(0)	11011 001 : 1110 1000	2/2	
FLDCW - Load Control Word	11011 001 : mod 101 r/m	7/7	
FLDENV - Load FPU Environment	11011 001 : mod 100 r/m		

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
FLDENV - Load FPU Environment	11011 001 : mod 100 r/m		
Real and v86 modes: 16-bit address		37	
Real and v86 modes: 32-bit address		37	
protectd modes: 16-bit address		32	
protectd modes: 32-bit address		33	
FLDL2E - Load $\log_2(\varepsilon)$ into ST(0)	11011 001 : 1110 1010	5/3	
FLDL2T - Load $\log_2(10)$ into ST(0)	11011 001 : 1110 1100	5/3	
FLDLG2 - Load $\log_{10}(2)$ into ST(0)	11011 001 : 1110 1100	5/3	
FLDLN2 - Load $\log_e(2)$ into ST(0)	11011 001 : 1110 1100	5/3	
FLDPI - Load π into ST(0)	11011 001 : 1110 1011	5/3	
FLDZ - Load + 0.0 into ST(0)	11011 001 : 1110 1110	2/2	
FMUL - Multiply			
ST(0) ← ST(0) × 32-bit memory	11011 000 : mod 001 r/m	3/1	4
ST(0) ← ST(0) × 64-bit memory	11011 010 : mod 001 r/m	3/1	4
ST(0) ← ST(0) × ST(i)	11011 d00 : 1100 1 ST(i)	3/1	4
FMULP - Multiply			
ST(0) ← ST(0) × ST(i)	11011 110 : 1100 1 ST(i)	3/1	
FNOP - No Operation	11011 001 : 1101 0000	1/1	
FPATAN - Partial Arctangent	11011 001 : 1111 0011	19-134	
FPREM - Partial Remainder	11011 001 : 1111 1000	16-64	
FPREM1 - Partial Remainder (IEEE)	11011 001 : 1111 0101	20-70	
FPTAN - Partial Tangent	11011 001 : 1111 0010	17-173	
FRNDINT - Round to Integer	11011 001 : 1111 1100	9-20	
FRSTOR - Restore FPU State	11011 101 : mod 100 r/m		
real and v86 modes:16-bit address		75/75	
real and v86 modes:32-bit address		95/95	
protectd modes: 16-bit address		70/70	
protectd modes: 32-bit address		70/70	
FSAVE -Store FPU State	1101 101 : mod 110 r/m		
real and v86 modes:16-bit address		127/127	2
real and v86 modes:32-bit address		151/151	2

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
protectd modes: 16-bit address		124/124	2
protectd modes: 32-bit address		124/124	2
FSCALE - Scale		20-31	
FSIN - Sine	11011 001 : 1111 1101	16-126	
FSINCOS - Sine and Cosine	11011 001 : 1111 1011	17-137	
FSQRT - Square Root	11011 001 : 1111 1010	70/70	
FST - Store Real			
32-bit memory	11011 001 : mod 010 r/m	2/2	
64-bit memory	11011 101 : mod 010 r/m	2/2	
ST(i)	11011 101 : 11 010 ST(i)	1/1	
FSTCW -Store Control Word	11011 001 : mod 111 r/m	2/2	2
FSTENV -Store FPU Environment	11011 001 : mod 110 r/m		
real and v86 modes:16-bit address		50/50	
real and v86 modes:32-bit address		48/50	2
protectd modes: 16-bit address		49/50	2
protectd modes: 32-bit address		50/50	2
FSTP - Store Real and Pop			
32-bit memory	11011 001 : mod 011 r/m	2/2	
64-bit memory	11011 101 : mod 011 r/m	2/2	
80-bit memory	11011 011 : mod 111 r/m	3/3	
ST(i)	11011 101 : 11 011 ST(i)	1/1	
FSTSW - Store Status Word	11011 111 : 1110 0000	6/2	2
Into AX			
FSTSW - Store Status Word Into Memory	11011 101 : mod 111 r/m	5/2	2
FSUB - Subtract			
ST(0) ← ST(0) - 32-bit memory	11011 000 : mod 100 r/m	3/1	
ST(0) ← ST(0) - 64-bit memory	11011 100 : mod 100 r/m	3/1	
ST(0) ← ST(0) - ST(i)	11011 d00 : 1100 R ST(i)	3/1	
FSUBP - Subtract and Pop			
ST(0) ← ST(0) - ST(i)	11011 110 : 1110 1 ST(i)	3/1	
FSUBR - Reverse Subtract			
ST(0) ← ST(0) - 32-bit memory	11011 000 : mod 101 r/m	3/1	

Câu lệnh	Định dạng	Nhịp đồng hồ	Ghi chú
ST(0) ← ST(0) - 64-bit memory	11011 100 : mod 101 r/m	3/1	
ST(0) ← ST(0) - ST(i)	11011 d00 : 1110 R ST(i)	3/1	
FSUBRP - Raverse Subtract and Pop			
ST(0) ← ST(i) - ST(0)	11011 110 : 1110 0 ST(i)	3/1	
FTST - Test	11011 001 : 1110 0 0100	4/1	
FUCOM - Unordered Compare Real	11011 101 : 1110 0 ST(i)	4/1	
FUCOMP - Unordered Compare and Pop	11011 101 : 1110 1 ST(i)	4/1	
FUCOMPP - Unordered Compare and Pop Twice	11011 010 : 1110 1001	4/1	
FXAM - Examine	11011 001 : 1110 0101	21/21	
FXCH - Exchange ST(0) and ST(i)	11011 001 : 1100 1 ST(i)	1	3
FXTRACT - Extract Exponent and Signiticand	11011 001 : 1111 0100	13/13	
FYL2X - ST(1) × $\log_2$(ST(0))	11011 001 : 1111 0001	22-111	
FYL2XP1 - ST(1) × $\log_2$(ST(0)) + 1,0)	11011 001 : 1111 1001	22-103	
FWAIT - Wait until FPU Ready	1001 1011	1/1	

L/NL	Clocks apply to loop and no loop cases respectively
MN/MX	Clocks shown define a range from minimum to maximum
P	Clocks apply to protected mode
R	Clocks apply to real-address mode
RV/P	First clock applies to real and V86 mode; second applies to protected mode
T/NT	Clocks apply to take and not taken cases respectively
U/L	Clocks apply to unlocked and locked cases respectively
1.	Assuming that the operand address and stack address fall in different cache interleaves
2.	Always locked. Always forced to miss cache.
4.	Clocks = {quotient(count/operand length) * 7 + 9} = 8 if count ≤ operand length (8/16/32).

5. Clocks = {quotient(count/operand length)} * 7 + 9
   = 9 if count ≤ operand length (8/16/32)
8. Penalty for cache miss: add 2 clocks for every stack value copied to the new stack frame.
9. Add 8 clocks for each load of an unaccessed descriptor
10. Refer to Task Switch Clock Counts Table for value of TS.

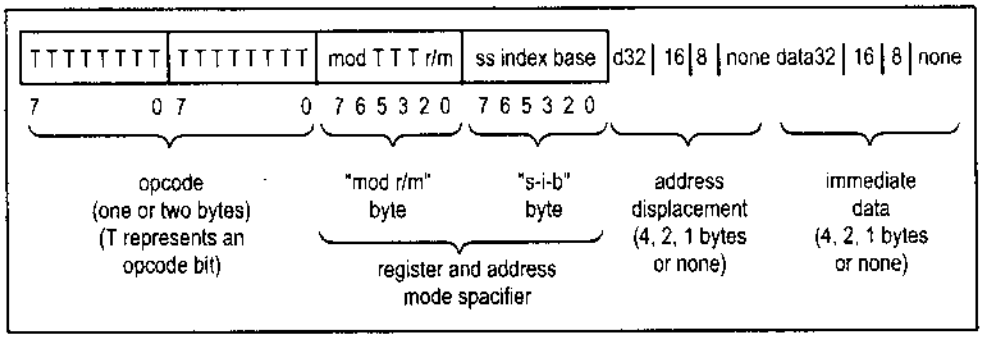
For notes 12 → 23: b = 0 - 3, nonzero byte number;
i = 0 - 1, nonzero nibble number;
n = 0 - 3, nonzero bit number in nibble.

12. Clocks = 8 + 4(b +1) + 3(i + 1) + 3(n + 1)
    = 6 if second operand = 0
13. Clocks = 9 + 4(b + 1) + 3(i + 1) + 3(n + 1)
    = 7 if second operand = 0

For notes 14 - 15: n = bit position (0 - 31)

14. Clocks = 7 + 2(32 - n)
    = 6 if second position (0 - 31).
15. Clocks = 8 + 2(32 - n)
    = 7 if second operand = 0
16. Assuming that the two string addresses fall in different cache interleaves.
21. Refer to the Interrupt Clock Counts Table for value of INT.
23. Add r + 3b for instruction cache miss/ Add 3 for branch misprediction.
24. Clocks shown define a range from minimum to maximum.
25. Add r + 3b for instruction cache miss.

Bảng C.1: Khuôn dạng lệnh X86



Bảng C.2: Các trường bên trong lệnh của X86

Tên trường	Mô tả	Số bit
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bists	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier or CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Bảng C.3: Thanh ghi và mã hóa bit w

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

Encoding of reg Field When w Field is not present in instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Fuction Specified by reg Field	
	(When w = 0)	(When w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

reg	Function of w field	
	(When w = 0)	(When w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

000	ES
001	CS
010	SS
011	DS
100	FS
101	FS
101	GS
110	do not use
111	do not use

Phụ lục D : TẬP LỆNH CỦA Z80

Bảng D.1: Các lệnh nạp 8-bit của Z80

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận r, r'	Reg
LD r, r'	r ← r'	•	•	×	•	×	•	•	•	r, r'	Reg
LD r, n	r ← n	•	•	×	•	×	•	•	•	000	B
										001	C
LD, r (HL)	r ← (HL)	•	•	×	•	×	•	•	•	010	D
LD r, (IX + d)	r ← (IY + d)	•	•	×	•	×	•	•	•	011	E
										100	H
										101	L
LD r, (IY + d)	r ← (IY + d)	•	•	×	•	×	•	•	•	111	A
LD (HL), r	(HL) ← r	•	•	×	•	×	•	•	•		
LD (IX + d), r	(IX + d), r	•	•	×	•	×	•	•	•		
LD (IY + d), r	(IY + d) ← r	•	•	×	•	×	•	•	•		
LD (HL), n	(HL) ← n	•	•	×	•	×	•	•	•		
LD (IX + d), n	(IX + d) ← n	•	•	×	•	×	•	•	•		
LD (IY + d), n	(IY + d) ← n	•	•	×	•	×	•	•	•		
LD A, (BC)	A ← (BC)	•	•	×	•	×	•	•	•		
LD A, (DE)	A ← (DE)	•	•	×	•	×	•	•	•		
LD A, (nn)	A ← (nn)	•	•	×	•	×	•	•	•		
LD (BC), A	(BC) ← A	•	•	×	•	×	•	•	•		
LD (DE), A	(DE) ← A	•	•	×	•	×	•	•	•		
LD (nn), A	(nn) ← A	•	•	×	•	×	•	•	•		
LD A, I	A ← I	I	I	×	0	×	IFF	0	•		
LD A, R	A ← R	I	I	×	0	×	IFF	0	•		
LD I, A	I ← A	•	•	×	•	×	•	•	•		
LD R, A	R ← A	•	•	×	•	×	•	•	•		

n 8-bit value in range < 0, 255>

Bảng D.2: Các lệnh nạp 16-bit

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận	
LD dd, nn	dd ← nn	•	•	×	•	×	•	•	•	dd	Pair
										00	BC
										01	DE
LD IX, nn	IX ← nn	•	•	×	•	×	•	•	•	10	HL
										11	SP
LD IY, nn	IY ← nn	•	•	×	•	×	•	•	•		
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	×	•	×	•	•	•		
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	×	•	×	•	•	•		
LD IX, (nn)	IX_H ← (nn + 1) dd_L ← (nn)	•	•	×	•	×	•	•	•		
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	×	•	×	•	•	•		
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	×	•	×	•	•	•		
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	×	•	×	•	•	•		
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	×	•	×	•	•	•		
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	×	•	×	•	•	•		
LD SP, HL	SP ← HL	•	•	×	•	×	•	•	•		
LD SP, IX	SP ← IX	•	•	×	•	×	•	•	•		
LD SP, IY	SP ← IY	•	•	×	•	×	•	•	•		
										qq	Pair
PUSH qq	(SP – 2) ← qq_L	•	•	×	•	×	•	•	•	00	BC
	(SP – 1) ← qq_H									01	DE
PUSH IX	(SP – 2) – IX_L	•	•	×	•	×	•	•	•	10	HL

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận	
	$(SP-1) \leftarrow IX_H$									11	AF
	$SP - SP - 2$										
PUSH IY	$(SP-2) \leftarrow IYL$	•	•	×	•	×	•	•	•		
	$(SP \leftarrow 1) \leftarrow IY_H$										
	$SP \leftarrow SP - 2$										
POP qq	$qq_H \leftarrow (SP + 1)$	•	•	×	•	×	•	•	•		
	$qq_L \leftarrow (SP)$										
	$SP \rightarrow SP + 2$										
POP IX	$IX_H \leftarrow (SP + 1)$	•	•	×	•	×	•	•	•		
	$IX_L \leftarrow (SP)$										
	$SP \leftarrow SP + 2$										
POP IY	$IY_H \leftarrow (SP + 1)$	•	•	×	•	×	•	•	•		
	$IY_L \leftarrow (SP)$										
	$SP \rightarrow SP + 2$										

Bảng D.3: Các lệnh trao đổi, vận chuyển và tìm kiếm khối

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ								Bình luận
		S	Z		H		P/V	N	C	
EX DE, HL	DE ← HL	•	•	×	•	×	•	•	•	
EX AF, AF	AF ← AF'	•	•	×	•	×	•	•	•	
EXX	BC ← BC' DE ← DE' HL ← HL'	•	•	×	•	×	•	•	•	
EX (SP), HL	H ← (SP + 1) L ← (SP)	•	•	×	•	×	•	•	•	
EX (SP), IX	IX_H ← (SP + 1) IX_L ← (SP)	•	•	×	•	×	•	•	•	
EX (SP), IY	IY_H ← (SP + 1) IY_L ← (SP)	•	•	×	•	×	•	•	•	
LDI	(DE) ← HL) DE ← DE + 1 HL ← HL + 1 BC ← BC – 1	•	•	×	0	×	① ↕	0	•	
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC ← 1 Repeat until BC = 0	•	•	×	0	×	① 0	0	•	
LDD	(DE) ← HL) DE ← DE – 1 HL ← HL – 1 BC ← BC – 1	•	•	×	0	×	① ↕	0	•	
LDDR	(DE) ← (HL) DE ← DE – 1 HL ← HL – 1 BC ← BC – 1 Repeat until BC = 0	•	•	×	0	×	② 0	0	•	If BC ≠ 0 If BC = 0

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận
			③				①			
CPI	A – (HL) HL ← HL + 1 BC ← BC – 1	↕	↕	×	↕	×	↕	1	•	
			③				①			
CPIR	A ← (HL) HL ← HL + 1 BC ← BC – 1 Repeat until A = (HL) or BC = 0	↕	↕	×	↕	×	↕	1	•	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL ← HL – 1 BC ← BC – 1	↕	↕	×	↕	×	↕	1	•	
CPDR	A – (HL) HL ← HL – 1 BC ← BC – 1 Repeat until A = (HL) or BC = 0	↕	↕	×	↕	×	↕	1	•	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

① P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1
② P/V flag is 0 at completion of instruction only
③ Z flag is 1 if A = (HL), otherwise Z = 0

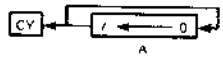
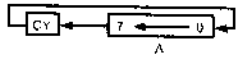
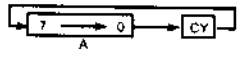
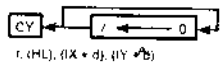
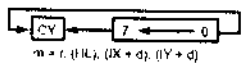
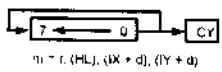
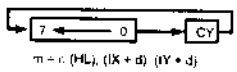
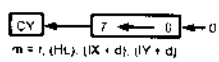
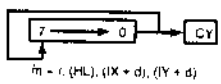
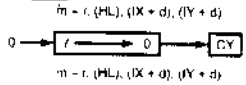
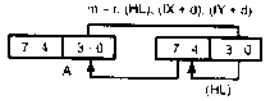
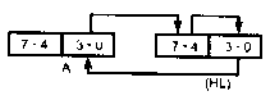
Bảng D.4: Các lệnh số học và logic 8-bit của Z80

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	000 B
										001 C
										010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	100 H
										101 L
										111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	s is any of r, n
SUB s	A ← A – s	↕	↕	X	↕	X	V	0	↕	(HL), (IX + d)
SBC A, s	A ← A – s – CY	↕	↕	X	↕	X	V	0	↕	(IY+d) as shown
AND s	A ← A – s	↕	↕	X	↕	X	V	0	0	for ADD
OR s	A ← A – s	↕	↕	X	↕	X	V	0	0	instruction
XOR s	A ← A ⊕ s	↕	↕	X	↕	X	V	0	0	
CP s	A ← s	↕	↕	X	↕	X	V	0	↕	
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	
INC (IX + d)	(IX + d ← (IX + d) + 1	↕	↕	X	↕	X	V	0	•	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X	↕	X	V	0	•	
DEC m	m ← m 1	↕	↕	X	↕	X	V	0	•	m is any of r, (HL), (IX + d), (IY + d) as shown for INC DEC same format and states as INC.
n	8-bit value in range <0,255>									
d	8-bit value in range <0,255>									

Bảng D.5: Các lệnh số học và logic 16-bit của Z80

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ								Bình luận	
		S	Z		H		P/V	N	C	SS	Reg.
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00	BC
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	01	DE
										10	HL
										11	AP
SBC HL, ss	HL ← HL – ss – CY	↕	↕	X	X	X	V	1	↕		
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	pp	Reg.
										00	BC
										01	DE
										10	IX
										11	SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	r	pp	Reg.
										00	BC
										01	DE
										10	IY
										11	SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•		
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•		
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•		
DEC ss	ss ← ss – 1	•	•	X	•	X	•	•	•		
DEC IX	IX ← IX – 1	•	•	X	•	X	•	•	•		
DEC IY	IY ← IY – 1	•	•	X	•	X	•	•	•		

Hình D.6: Các lệnh dịch và quay vòng của Z80

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Comments
RLCA		•	•	X	0	X	•	0	t	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	t	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	t	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	t	Rotate right accumulator
RLC r		t	t	X	0	X	P	0	t	Rotate left circular register r.
RLC (HL)		t	t	X	0	X	P	0	t	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
RLC (IX + d)		t	t	X	0	X	P	0	t	
RLC (IY + d)		t	t	X	0	X	P	0	t	
RL m		t	t	X	0	X	P	0	t	Instruction format and states are as shown for RLC's
RR m		t	t	X	0	X	P	0	t	
RR m		t	t	X	0	X	P	0	t	
RLA m		t	t	X	0	X	P	0	t	
RRA m		t	t	X	0	X	P	0	t	
RRL m		t	t	X	0	X	P	0	t	
RLD		t	t	X	0	X	P	0	•	Rotate digit left and right between the accumulator and location (HL)
RRD		t	t	X	0	X	P	0	•	The content of the upper half of the accumulator is unaffected

d 8-bit value in range <0,255>

Bảng D.7: Các lệnh số học và điều khiển chung

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận
DAA	Converts acc, content into packed BCD following add or subtract with packed BCD operands	↕	↕	×	↕	×	P	•	↕	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	Complement accumulator (one's complement.
NEG	$A \leftarrow 0 - A$	↕	↕	X	↕	X	V	1	↕	Negate acc. (two's complement
CCF	$CD \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	
HALT	CPU halted	•	•	X	•	X	•	•	•	
DI	IFF ← 0	•	•	X	•	X	•	•	•	
EI	IFF ← 1	•	•	X	•	X	•	•	•	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	

Bảng D.8: Các lệnh thiết lập, xóa và kiểm tra bit

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z	H		P/V	N	C	Bình luận	
									r	Reg
BIT b, r	$z \leftarrow r_b$	X	↕	X	1	X	X	0	•	000	B
										001	C
BIT b, (HL)	$z - (\overline{HL})_b$	X	↕	X	1	X	X	0	•	010	D
										↕	
										011	E
BIT b, $(\overline{IX+d})_b$	$z \leftarrow (\overline{IX+d})_b$	X	↕	X	1	X	X	0	•	100	H
										↕	
										101	L
										111	A
										b	Bit Tested
										000	0
BIT b, $(\overline{IY+d})_b$	$z \leftarrow (\overline{IY+d})_b$	X	↕	X	1	X	X	0	•	001	1
										010	2
										011	3
										100	4
										101	5
										110	6
										111	7
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•		
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•		
SET b, (IX + d)	$(IX + d)_b \leftarrow 1$	•	•	X	•	X	•	•	•		
SET b, (IY + d)	$(IY + d)_b \leftarrow 1$	•	•	X	•	X	•	•	•		
RES b, m	$m_b \leftarrow 0$	•	•	X	•	X	•	•	•		
	m ≡ r, (HL), (IX + d), (IY + d)										
d	8-bit value in range < 0, 255 >.										

Bảng D.9: Các lệnh nhảy của Z80

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z	H		P/V	N	C	Bình luận
			①						
JP nn	PC ← nn	•	•	X	•	X	•	•	•	
									cc Condition
JP cc, nn	If condition cc is	•	•	X	•	X	•	•	•	000 NZ non←zero
	true PC ← nn,									001 Z zero
	otherwise									010 NC non-carry
	continue									011 C carry
										100 PO parity odd
										101 PE parity even
										110 P sign positive
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	111 M sign negative
JR C, e	If C = 0	•	•	X	•	X	•	•	•	If condition not met.
	continue									
	If C = 1									If condition not met.
	PC ← PC + e									
JR NC, e	If C = 1	•	•	X	•	X	•	•	•	If condition not met
	continue									
	If C = 0									If condition not met
	PC ← PC + e									
JR Z, e	If Z = 0	•	•	X	•	X	•	•	•	If condition not met
	continue									
	If Z = 1									If condition not met
	PC ← PC + e									
JR NZ, e	If Z = 1	•	•	X	•	X	•	•	•	If condition not met
	continue									
	If Z = 0									If condition not met
	PC ← PC + e									
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	
JP (IY)	PC ← IX	•	•	X	•	X	•	•	•	

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z	H		P/V	N	C	Bình luận
DJNZ, e	B ← B ← 1	•	•	X	•	X	•	•	•	if B = 0
	if B = 0								
	continue								
	if B ≠ 0								if B ≠ 0
	PC ← PC + e								

e represent the extension in the relative addressing mode

e is a signed two's complement number in the range < - 126, 129 >.

e - 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Bảng D.10: Các lệnh gọi

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận
CALL nn	(SP - 1) ← PC_H	•	•	X	•	X	•	•	•	
	(SP - 2) ← PC_L									
	PC - nn									
CALL cc, nn	If condition cc is false,	•	•	X	•	X	•	•	•	If cc is false
	continue; otherwise same as CALL nn									If cc is false
RET	PC_L ← (SP)	•	•	•	•	X	•	•	•	
	PC_L ← (SP + 1)	•	•	X	•	X	•	•	•	
RET cc	If condition cc is false, continue otherwise same as RET									If cc is false If cc is true cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd
RETI	Return from interrupt	•	•	X	•	X	•	•	•	101 PE parity odd 110 P sign positive 111 M sign negative
RETN	Return from non-maskable	•	•	X	•	X	•	•	•	
RST p	(SP - 1) ← PC_H (SP - 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	t P 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H
nn	16-bit value in range < 0, 65535 >									

Bảng D.11: Các lệnh vào/ra của Z80

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z		H		P/V	N	C	Bình luận
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	n to $A_0 \sim A_7$ Acc, to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) If r = 110, only the flags will be affected	↕	↕	X	↕	X	P	0	•	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B – 1 HL ← HL + 1	X	↕ ① ②	X	X	X	X	1	X	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B – 1 HL ← HL + 1 Repeat until B = 0	X	↕ ①	X	X	X	X	1	X	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B – 1 HL ← HL – 1	X	↕ ②	X	X	X	X	1	X	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B – 1 HL ← HL – 1 Repeat until B = 0	X	↕	X	X	X	X	1	X	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	n to $A_0 \sim A_7$ Acc, to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	• ①	X	•	X	•	•	•	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B – 1 HL ← HL + 1	X	↕ ②	X	X	X	X	1	X	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Câu lệnh mã ngữ	Ký hiệu thực hiện	Các cờ S	Z	H			P/V	N	C	Bình luận
OTIR	(C) ← (HL)	X	↕	X	X	X	X	1	X	C to $A_0 \sim A_7$
	B ← B - 1									B to $A_8 \sim A_{15}$
	HL ← HL + 1									
	Repeat until									
	B = 0		①							
OUTD	(C) ← (HL)	X	↕	X	X	X	X	1	X	C to $A_0 \sim A_7$
	B ← B - 1									B to $A_8 \sim A_{15}$
	HL ← HL - 1		②							
OTDR	(C) ← (HL)	X	↕	X	X	X	X	1	X	C to $A_0 \sim A_7$
	B ← B - 1									B to $A_8 \sim A_{15}$
	HL ← HL - 1									
	Repeat until B = 0									

① If the result of B - 1 is zero the Z flag is set, otherwise it is reset

② Z flag is set upon instruction completions only.

n 8-bit value in range < 0,255 >

Phụ lục E: TẬP LỆNH CỦA CÁC VI XỬ LÝ MOTOROLA M680X0

Bảng E.1: Tập lệnh của MC68060 có sự phân loại cho siêu hướng của đường ống pOEP và sOEP

Mã ngữ	Lệnh	Phân loại siêu hướng
ABCD	Add decimal with extend	pOEP-only
ADD	Add	pOEP | sOEP
ADDA	Add address	pOEP | sOEP
ADDI,Dx	Add immediate	pOEP | sOEP
ADDI, - (Ax)+	Add immediate	pOEP | sOEP
Remaining ADDI	Add immediate	pOEP-until-last
ADDQ	Add quick	pOEP | sOEP
ADDX	Add extended	pOEP-only
AND	AND logical	pOEP | sOEP
ANDI, Dx	AND immediate	pOEP | sOEP
ANDI, -(Ax)+	AND immediate	pOEP | sOEP
Remaining ANDI	AND immediate	pOEP-until-last
ANDI to CCR	AND immediate to condition codes	pOEP-only
ASL	Arithmetic shift left	pOEP | sOEP
ASR	Arithmetic shift right	pOEP | sOEP
Bcc	Branch conditionally	pOEP-only[1]
BCHG Dy,	Test a bit and change	pOEP-only
BCHG #<imm>,	Test a bit and change	pOEP-until-last
BCLR Dy,	Test a bit and clear	pOEP-only
BCLR #<imm>,	Test a bit and clear	pOEP-until-last
BFCHG	Test bit field and change	pOEP-only
BFCLR	Test bit field and clear	pOEP-only
BFEXTS	Extract bit field signed	pOEP-only
BFEXTU	Extract bit field unsigned	pOEP-only
BFFFO	Find first one in bit field	pOEP-only
BFINS	Insert bit field	pOEP-only
BFSET	Set bit field	pOEP-only
BFTST	Test bit field	pOEP-only

Mã ngữ	Lệnh	Phân loại siêu hướng
BRA	Branch always	pOEP-only
BSET Dy,	Test a bit and set	pOEP-only
BSET #<imm>	Test a bit and set	pOEP-until-last
BSR	Branch to subroutine	pOEP-only
BTST Dy,	Test a bit	pOEP-only
BTST #<imm>	Test a bit	pOEP-until-last
CAS	Compare and swap with operand	pOEP-only
CHK	Check register against bounds	pOEP-only
CLR	Clear an operand	pOEP | sOEP
CMP	Compare	pOEP | sOEP
CMPA	Compare address	pOEP | sOEP
CMPI,Dx	Compare immediate	pOEP | sOEP
CMPI,-(Ax)+	Compare immediate	pOEP | sOEP
Remaining CMPI	Compare immediate	pOEP-until-last
CMPM	Compare memory	pOEP-until-last
DBcc	Test condition, decrement and branch	pOEP-only
DIVS.L	Signed divide long	pOEP-only
DIVS.W	Signed divide word	pOEP-only
DIVU.L	Unsigned long divide	pOEP-only
DIVU.W	Unsigned divide word	pOEP-only
EOR	Exclusive OR logical	pOEP | sOEP
EORI,Dx	Exclusive OR immediate	pOEP | sOEP
EORI,-(Ax)+	Exclusive OR immediate	pOEP | sOEP
Remaining EORI	Exclusive OR immediate	pOEP-until-last
EORI to CCR	Exclusive OR immediate to condition codes	pOEP-only
EXG	Exchange registers	pOEP-only
EXT	Sign extend	pOEP | sOEP
EXTB.L	Sign extend byte to long	pOEP | sOEP
ILLEGAL	Take illegal instruction trap	pOEP | sOEP
JMP	Jump	pOEP-only
JSR	Jump to subroutine	pOEP-only

Mã ngữ	Lệnh	Phân loại siêu hướng
LEA	Load effective address	pOEP | sOEP
LINK	Link and allocate	pOEP-until-last
LSL	Logical shift left	pOEP | sOEP
LSR	Logical shift right	pOEP | sOEP
MOVE, Rx	Move data from source to destination	pOEP | sOEP
MOVE Ry,	Move data from source to destination	pOEP | sOEP
MOVE <mem>y ,<mem>x	Move data from source to destination	pOEP-until-last
MOVE #<imm>, <mem>x	Move data from source to destination	pOEP-until-last
MOVEA	Move address	pOEP | sOEP
MOVE from CCR	Move from condition codes	pOEP-only
MOVE to CCR	Move to condition codes	pOEP | sOEP
MOVE16	Move 16-byte block	pOEP-only
MOVEM	Move multiple registers	pOEP-only
MOVEQ	Move quick	pOEP | sOEP
MULS.L	Signed multiply long	pOEP-only
MULS.W	Signed multiply word	pOEP-only
MULUL	Unsigned multiply long	pOEP-only
MULU.W	Unsigned multiply word	pOEP-only
NBCD	Negate decimal with extend	pOEP-only
NEG	Negate	pOEP | sOEP
NEGX	Negate with extend	pOEP-only
NOP	No operation	pOEP-only
NOT	Logical complement	pOEP | sOEP
OR	Inclusive OR logical	pOEP | sOEP
ORI,Dx	Inclusive OR immediate	pOEP | sOEP
ORI,-(Ax)+	Inclusive OR immediate	pOEP | sOEP
Remaining ORI	Inclusive OR immediate	pOEP-until-last
ORI to CCR	Inclusive OR immediate to condition codes	pOEP-only

Mã ngữ	Lệnh	Phân loại siêu hướng
PACK	Pack BCD digit	pOEP-only
PEA	Push effective address	pOEP-only
ROL	Rotate without extend left	pOEP | sOEP
KOK	Rotate without extend right	pOEP | sOEP
ROXL	Rotate with extend left	pOEP-only
ROXR	Rotate with extend right	pOEP-only
RTD	Return and deallocate parameters	pOEP-only
RTR	Return and restore condition codes	pOEP-only
RTS	Return from subroutine	pOEP-only
SBCD	Subtract decimal with extend	pOEP-only
Scc	Set according to condition	pOEP-but allows-sOEF
SUB	Subtract	pOEP | sOEP
SUBA	Subtract address	pOEP | sOEP
SUBI,Dx	Subtract immediate	pOEP | sOEP
SUBI,-(Ax)+	Subtract immediate	pOEP | sOEP
Remaining SUBI	Subtract immediate	pOEP-until-last
SUBQ	Subtract quick	pOEP | sOEP
SUBX	Subtract with extend	pOEP-only
SWAP	Swap register halves	pOEP-only
TAS	Test and set an operand	pOEP-only
TRAP	Trap	pOEP | sOEP
TRAPF	Trap on false	pOEP | sOEP
Remaining TRAPcc	Trap on condition	pOEP-only
TRAPV	Trap on overflow	pOEP-only
TST	Test an operand	pOEP | sOEP
UNLK	Unlink	pOEP-only
UNPK	Unpack BCD digit	pOEP-only

[1] A Bcc instmction is pOEP-but-allows-sOEP if it is not predicted from the branch cache an the direction of the branch is forward or if the Bcc is predicted as a "not-taken" branch.

Source: Courtesy of Motorola. Inc.

Bảng E.2: Phân loại siêu hướng của các lệnh với các số nguyên của M680X0

Mã nguồn	Lệnh	Phân loại siêu hướng
FABS, FDABS, FSABS	Absolute value	pOEP-but-allows-ORP[1]
FADD, FDADD, FSADD	Add	pOEP-only
FBcc	Branch conditionally	pOEP-but-allows-ORP[1]
FCMP	Compare	pOEP-but-allows-ORP[1]
FDIV, FDDIV, FSDIV, FSGLDIV	Divide	pOEP-but-allows-ORP[1]
FINT, FINTRZ	Integer par, round-to-zero	pOEP-but-allows-ORP[1]
FMOVE, FDMOVE, FSMOVE	Move floating-point data register	pOEP-but-allows-ORP[1]
FMOVE	Move system control register	pOEP-only
FMOVEM	Move multiple data registers	pOEP-only
FMUL, FDMUL, FSMUL, FSGLMUL	Multiply	pOEP-but-allows-ORP[1]
FNEG, FDNEG, FSNEG	Negate	pOEP-but-allows-ORP[1]
FNOP	No operation	pOEP-only
FSQRT	Square root	pOEP-but-allows-ORP[1]
FSUB, FDSUB, FSSUB	Subtrac	pOEP-but-allows-ORP[1]
FTST	Test operand	pOEP-but-allows-ORP[1]

[1] These floating-point instructions are pOEP-but-allows-sOEP exept for the following:

F<op> Dm, FPn
F<op> &imm, FPn
F<op>.x <mem>, FPn

which are classified as pOEP-only.

SOURCE: Courtest of Motorola, Inc.

Bảng E.3: Phân loại siêu hướng của các lệnh có thẩm quyền của M680

Mã nguồn	Lệnh	Phân loại siêu hướng
ANDI to SR	AND immediate to status register	pOEP-only
CINV	Invalidate cache lines	pOEP-only
CPUSH	Push and invalidate cache lines	pOEP-only
EORI to SR	Exclusive OR immediate to status register	pOEP-only
MOVE from SR	Move from status register	pOEP-only
MOVE to SR	Move to status register	pOEP-only
MOVE USP	Move user stack pointer	pOEP-only
MOVEC	Move control register	pOEP-only
MOVES	Move address space	pOEP-only
ORI to SR	Inclusive OR immediate to status register	pOEP-only
PFLUSH	Flush ATC entries	pOEP-only
PLPA	Load physical address	pOEP-only
RESET	Reset external devices	pOEP-only
RTE	Return from exception	pOEP-only
STOP	Load status register and stop	pOEP-only

Có sự khác nhau giữa MC68060 và MC68040 trong phần cứng hỗ trợ các lệnh. Những khác nhau này được liệt kê trong bảng E.4. Sự khác nhau chính giữa chúng là sự mở rộng của lệnh MOVEC, lệnh mới ở MC68060: LPSTOP và PLPA, và sự thực hiện lệnh PFLUSH.

Bảng E.4

Mã nguồn	Lệnh	Phân loại siêu hướng
CAS	Compare and swap with operand	Emulation support on 68060 for misaligned operand
CAS2	Compare and swap dual operands	Emulation support on 68060, not implemented in hardware
CHK2	Check register against upper and lower bound	Emulation support on 68060, not implemented in hardware
CMP2	Compare register against upper lower bound	Emulation support on 68060, not and implemented in hardware
DIVS.L	Signed divide	Emulation support on 68060 for 64/32
DIVU.L	Unsigned divide	Emulation support on 68060 for 64/32
FDBcc	FP decrement and branch	Emulation support on 68060, not implemented in hardware
FINT	FP integer part	Implemented in hardware on 68060, not on 68040
FINTRZ	FP integer part, round-to-zero	Implemented on hardware on 68060, not on 68040
FMOVEM	FP move multiple data registers	Emulation support on 68060 for dynamic register list
FScc	FP set according to condition	Emulation support on 68060, not implemented in hardware
FTRAPcc	FP trap on condition	Emulation support on 68060, not implemented in hardware
LPSTOP	Low-power stop	Implemented on 68060, not on 68040
MOVEC	Move control registers	Revised functionality
MOVEP	Move peripheral	Emulation support on 68060, not implemented in hardware
MULS.L	Signed long multiply	Emulation support on 68060 for 32 ×32 → 64
MULU.L	Unsigned long multiply	Emulation support on 68060 for 32 × S2 → 64
PTEST	Test a logical address	Not implemented on 68060, PLPA added on 68060 (equivalent functionality)
PLPA	Load physical address	Implemented on 68060, not on 68040

Phụ lục F: TẬP LỆNH CỦA VI ĐIỀU KHIỂN INTEL 8051

Bảng F.1: Các lệnh vận chuyển dữ liệu

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
MOV A, Rn	1	1	1	0	1	n_2	n_1	n_0	E8 ~ EF	(A) ← (Rn)
MOV A, direct	1 a_7	1 a_6	1 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	E5 Byte 2	(A) ← (direct)
MOV A, @ Ri	1	1	1	0	0	1	1	i	E6 ~ E7	(A) ← ((Ri))
MOV A, # data	0 d_7	1 d_6	1 d_5	1 d_4	0 d_3	1 d_2	0 d_1	0 d_0	74 Byte 2	(A) ← # data
MOV Rn, A	1	1	1	1	1	n_2	n_1	n_0	F8 ~ FF	(Rn) ← (A)
MOV Rn, direct	1 a_7	0 a_6	1 a_5	0 a_4	1 a_3	n_2 a_2	n_1 a_1	n_0 a_0	A8 ~ AF Byte 2	(Rn) ← (direct)
MOV Rn, # data	0 d_7	1 d_6	1 d_5	1 d_4	1 d_3	n_2 d_2	n_1 d_1	n_0 d_0	78 ~ 7F Byte 2	(Rn) ← # data
MOV direct, A	0 a_7	1 a_6	1 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	F5 Byte 2	(direct) ← (A)
MOV direct, Rn	0 a_7	1 a_6	1 a_5	1 a_4	0 a_3	n_2 a_2	n_1 a_1	n_0 a_0	88 ~ 8F Byte 2	(direct) ← (Rn)
MOV direct, direct	1 a_7 a_7	0 a_6 a_6	0 a_5 a_5	0 a_4 a_4	0 a_3 a_3	1 a_2 a_2	0 a_1 a_1	1 a_0 a_0	85 Byte 2 Byte 3	(direct) ← (direct) (source) (destination)
MOV direct, @ Ri	1 a_7	0 a_6	0 a_5	0 a_4	0 a_3	1 a_2	1 a_1	i a_0	86 ~ 87	(direct) ← ((Ri))
MOV direct, # data	0 d_7	1 d_6	1 d_5	1 d_4	0 d_3	1 d_2	0 d_1	1 d_0	75 Byte 2 Byte 3	(direct) ← # data
MOV @ Ri, A	1	1	1	1	0	1	1	i	F6 ~ F7	((Ri)) ← A
MOV @ Ri, direct	1 a_7	0 a_6	1 a_5	0 a_4	0 a_3	1 a_2	1 a_1	i a_0	A6 ~ A7 Byte 2	((Ri)) ← (direct)
MOV @ Ri, # data	0 d_7	1 d_6	1 d_5	1 d_4	0 d_3	1 d_2	0 d_1	1 d_0	76 ~ 77 Byte 2	((Ri)) ~ # data
MOV DPTR, # data 16	1 d_7	0 d_6	0 d_5	1 d_4	0 d_3	01 d_2	0 d_1	1 d_0	90 Byte 2 Byte 2	(DPTR) ← # $data_{15 \leftarrow 0}$ (DPTR) ← # $data_{15-8}$ (DPTR) ← # $data_{7-0}$

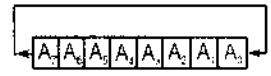
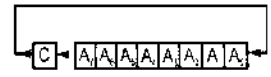
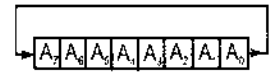
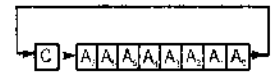
Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
MOVC A, @ A + DPTR	1	0	0	1	0	0	1	1	93	(A) ← ((A) + (DPTR))
MOVC A, @ A + PC	1	0	0	0	0	0	1	1	83	(A) ← ((A) + (PC))
MOVX A, @ Ri	1	1	1	0	0	0	1	i	E2 ~ E3	(A) ← ((Ri) External RAM
MOVX A, @ DPTR	1	1	1	0	0	0	0	0	E0	(A) ← ((DPTR)) External RAM
MOVX @ Ri, A	1	1	1	1	0	0	1	i	F2 ~ F3	((Ri)) ~ (A) External RAM
MOVX @ DPTR, A	1	1	1	1	0	0	0	0	F0	((DPTR)) ← (A) External RAM
PUSH direct	1 a_7	1 a_6	0 a_5	0 a_4	0 a_3	0 a_2	0 a_1	0 a_0	C0 Byte 2	(SP) ← (SP) + 1 ((SP)) ← (direct)
POP direct	1 a_7	1 a_6	0 a_5	1 a_4	0 a_3	0 a_2	0 a_1	0 a_0	D0 Byte 2	(direct) ← ((SP)) (SP) ← (SP) ← 1
XCH A, Rn	1	1	0	0	1	n_2	n_1	n_0	C8 ~ DF	(A) ⇄ (Rn)
XCH A, direct	1 a_7	1 a_6	0 a_5	1 a_4	0 a_3	1 a_2	0 a_1	0 a_0	C5 Byte 2	(A) ← (direct)
XCH A, @ Rn	1	1	0	0	0	1	1	i	C6 ~ D7	(A) ⇄ ((Ri))
XCHD A, @ Ri	1	1	0	1	0	1	1	i	C6 ~ D7	$(A_{3\leftarrow 0}) \rightleftarrows ((Ri_{3\leftarrow 0}))$

Bảng F.2: Các lệnh số học

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
ADD A, Rn	0	0	1	0	1	n_2	n_1	n_0	28 ~ 2F	(A) ← (A) + (Rn)
ADD A, direct	0	0	1	0	0	1	0	1	25	(A) ← (A) + (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ADD A, @ Ri	0	0	1	0	0	1	1	i	26 ~ 27	(A) ← (A) + (Ri)
ADD A, # data	0	0	1	0	0	1	0	0	24	(A) ← (A) + # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
ADDC A, Rn	0	0	1	1	1	n_2	n_1	n_0	38 ~ 3F	(A) ← (A) + (C) + (Rn)
ADDC A, direct	0	0	1	0	0	1	0	1	35	(A) ← (A) + (C) + (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ADDC A, @ Ri	0	0	1	1	0	1	0	i	36 ~ 37	(A) ← (A) + (C) + ((Ri))
ADDC A, # data	0	0	1	0	0	1	0	0	34	(A) ← (A) + (C) + # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
SUBB A, Rn	1	0	0	1	1	n_2	n_1	n_0	98 ~ 9F	(A) ← (A) - (C) - (Rn)
SUBB A, direct	0	0	1	0	0	1	0	1	95	(A) ← (A) - (C) - (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
SUBB A, @ Ri	1	0	0	1	0	1	1	i	96 ~ 97	(A) ← (A) - (C) - ((Ri))
SUBB A, # data	0	0	1	0	0	1	0	0	94	(A) ← (A) - (C) - # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
INC A	0	0	0	0	0	1	0	0	04	(A) ← (A) + 1
INC Rn	0	0	0	0	1	n_2	n_1	n_0	08 ~ 0F	(Rn) ← (Rn) + 1
INC direct	0	0	0	0	0	1	0	1	05	(direct) ← (direct) + 1
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
INC @ Ri	0	0	0	0	0	1	1	i	06 ~ 07	((Ri)) ← ((Ri)) + 1
INC DPTR	1	0	1	0	0	0	1	1	A3	(DPTR) ← (DPTR) + 1
DEC A	0	0	0	1	0	1	0	0	14	(A) ← (A) - 1
DEC Rn	0	0	0	1	1	n_2	n_1	n_0	18 ~ 1F	(Rn) ← (Rn) - 1
DEC direct	0	0	0	1	0	1	0	1	15	(direct) ← (direct) - 1
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0		
DEC @ Ri	0	0	0	1	0	1	1	i	16 ~ 17	((Ri)) ← ((Ri)) - 1
MUL AB	1	0	1	0	0	1	0	0	A4	(B_{15-8}), (A_{7-0}) ← (A) × (B)
DIV AB	1	0	0	0	0	1	0	0	B4	(A_{15-8}), (B_{7-0}) ← (A)/(B)
DAA	1	1	0	1	0	1	0	0	D4	Contents of accumulator are BCD. If [[(A_{3-0}) > 9] OR [(AC) = 1]] THEN (A_{3-0}) ← (A_{3-0}) + 6 AND If [[(A_{7-4}) > 9] OR [(AC) = 1]] THEN (A_{7-4}) ← (A_{7-4}) + 6

Bảng F.3: Các lệnh logic

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
ANL A, Rn	0	1	0	1	1	n_2	n_1	n_0	58 ~ 5F	(A) ← (A) AND (Rn)
ANL A, direct	0	1	0	1	0	1	0	1	55	(A) ← (A) AND (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ANL A, @ Ri	0	1	0	1	0	1	1	i	56 ~ 57	(A) ~ (A) AND ((Ri))
ANL A, # data	0	1	0	1	0	1	0	0	54	(A) ← (A) AND # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
ANL direct, A	0	1	0	1	0	1	0	0	52	(direct) ← (direct) AND (A)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ANL direct, # data	0	1	0	1	0	0	1	1	53	(direct) ← (direct) AND # data
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 3	
ORL A, Rn	0	1	0	0	1	n_2	n_1	n_0	48 ~ 4F	(A) ← (A) OR (Rn)
ORL A, direct	0	1	0	0	0	1	0	1	45	(A) ← (A) OR (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ORL A, @ Ri	0	1	0	0	0	1	1	i	46 ~ 47	(A) ← (A) OR ((Ri))
ORL A, # data	0	1	0	0	0	1	0	0	44	(A) ← (A) OR # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
ORL direct, A	0	1	0	0	0	0	1	0	42	(direct) ← (direct) OR (A)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
ORL direct, # data	0	1	0	0	0	0	1	1	43	(direct) ← (direct) OR # data
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 3	
XRL A, Rn	0	1	1	0	1	n_2	n_1	n_0	68 ~ 6F	(A) ← (A) XOR (Rn)
XRL A, direct	0	1	1	0	0	1	0	1	65	(A) ← (A) XOR (direct)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
XRL A, @ Ri	0	1	1	0	0	1	1	i	66 ~ 67	(A) ← (A) XOR ((Ri))
XRL A, # data	0	1	1	0	0	1	0	0	64	(A) ← (A) XOR # data
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	
XRL direct, A	0	1	1	1	0	0	1	0	62	(direct) ← (direct) XOR (A)
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	
XRL direct, # data	0	1	1	1	0	0	1	1	63	(direct) ← (direct) XOR # data
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 3	
CLR A	1	1	1	0	0	1	0	0	E4	(A) ← 0
CPL A	1	1	1	1	0	1	0	0	F4	(A) ← $\overline{A}$
RL A	0	0	1	0	0	0	1	1	23	 The contents of the accumulator are rotated left by one bit
RLC A	0	0	1	1	0	0	1	1	33	 The contents of the accumulator and carry are rotated left by one bit
RR A	0	0	0	0	0	0	1	1	03	 The contents of the accumulator are rotated right by one bit
RRC A	0	0	0	1	0	0	1	1	13	 The contents of the accumulator and carry are rotated right by one bit
SWAP A	1	1	0	0	0	1	0	0	C4	$(A_{3-0}) \rightleftarrows (A_{7-4})$

Bảng F.4: Các lệnh chuyển điều khiển

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
ACALL addr 11	a_{10}	a_9	a_8	1	0	0	0	1	Byte 1	(PC) ← (PC) + 2
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	(SP) ← (SP) + 1
										((SP)) ← (PC_{7-0})
										(SP) ← (SP) + 1
										((SP)) ← (PC_{15-8})
										(PC) ← page address
ACALL addr 16	0	0	0	0	0	0	1	0	12	(PC) ← (PC) + 3
	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	Byte 2	(SP) ← (SP) + 1
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 3	((SP)) ← (PC_{7-0})
										(SP) ← (SP) + 1
										((SP)) ← (PC_{15-8})
										(PC) ← $addr_{15-0}$
RET	0	0	1	0	0	0	1	0	22	(PC_{15-8}) ← ((PC))
										(SP) ← (SP) - 1
										(PC_{7-0}) ← ((SP))
										(SP) ← (SP) - 1
RETI	0	0	1	1	0	0	1	0	32	(PC_{15-8}) ← ((SP))
										(SP) ← (SP) - 1
										(PC_{7-0}) ← ((SP))
										(SP) ← (SP) - 1
AJMP addr 11	a_{10}	a_9	a_8	1	0	0	0	1	Byte 1	(PC) ← (PC) + 2
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	(PC_{10-0}) ← page addr
AJMP addr 16	0	0	0	0	0	0	1	0	02	(PC) ← $addr_{15-0}$
	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	Byte 2	
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 3	
SJMP rel	1	0	0	0	0	0	0	0	80	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2	(PC) ← (PC) + rel
JMP@ A + DPTR	0	1	1	1	0	0	1	1	73	(PC) ← (A) + (DPTR)
JZ rel	0	1	1	0	0	0	0	0	60	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte	IF (C) = 0 THEN (PC) ← (PC) + rel
JNZ, rel	0	1	1	1	0	0	0	0	70	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte	IF (C) ≠ 0 THEN (PC) ← (PC) + rel
JC, rel	0	1	0	1	0	0	0	0	50	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte	IF (C) = 0 THEN (PC) ← (PC) + rel
JNC, rel	0	1	0	1	0	0	0	0	50	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2	IF (bit) = 1 THEN (PC) ← (PC) + rel

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
JNB bit, rel	0	0	1	1	0	0	0	0	10	(PC) ← (PC) + 3
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	IF (bit) = 1 THEN (PC) ← (PC) + rel
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	
JNB bit, rel	0	0	1	1	0	0	0	0	10	(PC) ← (PC) + 3
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	IF (bit) = 1 THEN (PC) ← (PC) + rel
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	
JBC bit, rel	0	0	1	1	0	1	0	1	30	(PC) ← (PC) + 3
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	IF (bit) = 1 THEN (bit) ← 0
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	(PC) ← (PC) + rel
CJNE A, direct, rel	1	0	1	1	0	1	0	1	5	(PC) ← (PC) + 3
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	IF (direct) < (A) THEN (PC) ← (PC) + rel and (c) ← 0
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	OR
										IF (direct) > (A) THEN (PC) ← (PC) + rel and (C) ← 1
CJNE A, #data, rel	1	0	1	1	0	1	0	0	B4	(PC) ← (PC) + 3
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	Byte 2	IF #data < (A)
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	THEN (PC) ← (PC) + rel and (C) ← 0
										OR
										IF #data > (A) THEN (PC) ← (PC) + rel and (C) ← 1
CJNE Rn, #data, rel	1	0	1	1	0	n_2	n_1	n_0	B8 ~ BF	(PC) ← (PC) + 3
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	IF #data < (Rn) THEN (PC) ← (PC) + rel and (c) ← 0
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	OR
										IF #data > (Rn) THEN (PC) ← (PC) + rel and (C) ← 1
CJNE @Ri, #data, rel	1	0	1	1	0	1	1	i	B6 ~ B7	(PC) ← (PC) + 3
	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	Byte 2	IF #data < ((Ri)) THEN (PC) ← (PC) + rel and (c) ← 0
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 3	OR
										IF #data > ((Ri)) THEN (PC) ← (PC) + rel and (C) ← 1
DJNZ Rn, rel	1	1	0	1	1	n_2	n_1	n_0	D8 ~ DF	(PC) ← (PC) + 2
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2	(Rn) ← (Rn) ← 2
										IF (Rn) ≠ 0 THEN (PC) ← (PC) + rel
DJNZ direct, rel	1	1	0	1	0	1	0	1	D5	(PC) ← (PC) + 3
	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Byte 2	(direct) ← (direct) ← 1
									Byte 3	IF (direct) ≠ 0 THEN (PC) ← (PC) + rel
NOP	0	0	0	0	0	0	0	0	00	(PC) ← (PC) + 1

Bảng F.5: Các lệnh xử lý bit

Câu lệnh mã ngữ	Mã lệnh								Cơ số 16	Diễn giải
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
CLR C	1	1	0	0	0	0	1	1	C3	(C) ← 0
CLR bit	1	1	0	0	0	0	1	0	C2	(bit) ← 0
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
SETB C	1	1	0	1	0	0	1	1	D3	(C) ← 1
SETB bit	1	1	0	1	0	0	1	0	D2	(bit) ← 1
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
CPL C	1	0	1	1	0	0	1	1	B3	(C) ← ($\overline{C}$)
CPL bit	1	0	1	1	0	0	1	0	B2	(bit) ← ($\overline{bit}$)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
ANL C, bit	1	0	1	1	0	0	0	0	82	(C) ← (C) AND bit
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
ANL C, /bit	1	0	1	1	0	0	1	0	B0	(C) ← (C) AND ($\overline{bit}$)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
ORL C, bit	0	1	1	1	0	0	1	0	72	(C) ← (C) OR (bit)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
ORL C, /bit	1	0	1	0	0	0	0	0	A0	(C) ← (C) OR ($\overline{bit}$)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
MOV C, bit	1	0	1	0	0	0	1	0	A2	(C) ← (bit)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	
MOV C, bit	1	0	0	1	0	0	1	0	92	(bit) ← (C)
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Byte 2	

Phụ lục G: TẬP LỆNH CỦA INTEL 8085A

Các nhóm lệnh của INTEL 8085A

1. Vận chuyển dữ liệu (Data Transfer):

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memory
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from Memory
SHLD	Store H & L Registers Directly in Memory
LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register Pair
STAX	Store Accumulator in Address in Register Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

2. Số học (Arithmetic)

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add Immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow Flag (Carry)
INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

3. Logic (Logical)

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate Data
ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data

- ***So sánh***

CMP	Compare
CPI	Compare Using Immediate Data

- ***Dịch và quay vòng***

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry

- ***Bù 2 và cờ carry***

CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

4. Rẽ nhánh (Branch)

JMP	Jump
CALL	Call
RET	Return

- ***Với các điều kiện***

NZ	Not Zero (Z = 0)
Z	Zero (Z = 1)
NC	No Carry (C = 0)
C	Carry (C = 1)
PO	Parity Odd (P = 0)
PE	Parity Even (P = 1)

P	Plus (S = 0)
M	Minus (S = 1)

- ***Các lệnh có điều kiện***

Jumps	*Calls*	*Returns*	
C	CC	RC	(Carry)
INC	CNC	RNC	(No Carry)
JZ	CZ	RZ	(Zero)
JNZ	CNZ	RNZ	(Not Zero)
JP	CP	RP	(Plus)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)

- ***Với PC***

PCHL	Move H & L to Program Counter
RST	Special Restart Instruction Used with Interrupts

Các lệnh với ngăn xếp:

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

5. Các lệnh vào ra

IN	Initiate Input Operation
OUT	Initiate Output Operation

6. Các lệnh điều khiển hệ thống (Machine Control):

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Câu lệnh mã ngữ	Op	SZAPC	~s	Mô tả	Ghi chú
ACI n	CE	*****	7	Add with Carry Immediate	A=A+n+CY
ADC r	8F	*****	4	Add with Carry	A=A+r+CY(21X)
ADC M	8E	*****	7	Add with Carry to Memory	A=A+[HL]+CY
ADD r	87	*****	4	Add	A=A+r (20X)
ADD M	86	*****	7	Add to Memory	A=A+[HL]
ADI n	C6	*****	7	Add Immediate	A=A+n
ANA r	A7	****0	4	AND Accumulator	A=A&r (24X)
ANA M	A6	****0	7	AND Accumulator and Memory	A=A&[HL]
ANI n	E6	**0*0	7	AND Immediate	A=A&n
CALL a	CD	----	18	Call unconditional	-[SP]=PC,PC=a
CC a	DC	----	9	Call on Carry	If CY=1(18~s)
CM a	FC	----	9	Call on Minus	If S=1 (18~s)
CMA	2F	----	4	Complement Accumulator	A=~A
CMC	3F	---*	4	Complement Carry	CY=~CY
CMP r	BF	*****	4	Compare	A-r (27X)
CMP M	BF	*****	7	Compare with Memory	A-[HL]
CNC a	D4	----	9	Call on No Carry	If CY =0 (18~s)
CNZ a	C4	----	9	Call on No Zero	If Z=0 (18~s)
CP a	F4	----	9	Call on Plus	If S=0 (18~s)
CPE a	EC	----	9	Call on Parity Even	If P=1 (18~s)
CPI n	FE	*****	7	Compare Immediate	A-n
CPO a	E4	----	9	Call on Parity Odd	If P=0 (18~s)
CZ a	CC	----	9	Call on Zero	If Z=1 (18~s)
DAA	27	*****	4	Decimal Adjust Accumulator	A=BCD format
DAD B	09	---*	10	Double Add BC to HL	HL=HL+BC
DAD D	19	---*	10	Double Add DE to HL	HL=HL+DE
DAD H	29	---*	10	Double Add HL to HL	HL=HL+HL
DAD SP	39	---*	10	Double Add SP to HL	HL=HL+SP
DCR r	3D	****-	4	Decrement	r=r-1 (0X5)
DCR M	35	****-	10	Decrement Memory	[HL]=[HL]-1

Câu lệnh mã ngữ	Op	SZAPC	~s	Mô tả	Ghi chú
DCX B	0B	----	6	Decrement BC	BC=BC-1
DCX D	1B	----	6	Decrement DE	DE=DE-1
DCX H	2B	----	6	Decrement HL	HL=HL-1
DCX SP	3B	----	6	Decrement Stack Pointer	SP=SP-1
DI	F3	----	4	Disable Interrupts	
EI	FB	----	4	Enable Interrupts	
HLT	76	----	5	Halt	
IN p	DB	----	10	Input	A=[p]
INR r	3C	****-	4	Increment	r=r+1 (0X4)
INR M	3C	****-	10	Increment Memory	[HL]=[HL]+1
INX B	03	----	6	Increment BC	BC=BC+1
INX D	13	----	6	Increment DE	DE=DE+1
INX H	23	----	6	Increment HL	HL=HL+1
INX SP	33	----	6	Increment Stack Pointer	SP=SP+1
JMP a	C3	----	7	Jump unconditional	PC=a
JC a	DA	----	7	Jump on Carry	If CY=1(10~s)
JM a	FA	----	7	Jump on Minus	If S=1 (10~s)
JNC a	D2	----	7	Jump on No Carry	If CY=0(10~s)
JNZ a	C2	----	7	Jump on No Zero	If Z=0 (10~s)
JP a	F2	----	7	Jump on Plus	If S=0 (10~s)
JPE a	EA	----	7	Jump on Parity Even	If P=1 (10~s)
JPO a	E2	----	7	Jump on Parity Odd	If P=0 (10~s)
JZ a	CA	----	7	Jump on Zero	If Z=1 (10~s)
LDA a	3A	----	13	Load Accumulator direct	A=[a]
LDAX B	0A	----	7	Load Accumulator indirect	A=[BC]
LDAX D	1A	----	7	Load Accumulator indirect	A=[DE]
LHLD a	2A	----	16	Load HL Direct	HL=[a]
LXI B,nn	01	----	10	Load Immediate BC	BC=nn
LXI D,nn	11	----	10	Load Immediate DE	DE=nn
LXI H,nn	21	----	10	Load Immediate HL	HL=nn
LXI SP,nn	31	----	10	Load Immediate Stack Ptr	SP=nn

Câu lệnh mã ngữ	Op	SZAPC	~s	Mô tả	Ghi chú
MOV r1,r2	7F	——	4	Move register to register	r1=r2 (1XX)
MOV M,r	77	——	7	Move register to Memory	[HL]=r (16X)
MOV r,M	7E	——	7	Move Memory to register	r=[HL] (1X6)
MVI r,n	3E	——	7	Move Immediate	r=n (0X6)
MVI M,n	36	——	10	Move Immediate to Memory	[HL]=n
NOP	00	——	4	No Operation	
ORA r	B7	**0*0	4	Inclusive OR Accumulator	A=Avr (26X)
ORA M	B6	**0*0	7	Inclusive OR Accumulator	A=Av[HL]
ORI n	F6	**0*0	7	Inclusive OR Immediate	A=Avn
OUT p	D3	——	10	Output	[p]=A
PCHL	E9	——	6	Jump HL indirect	PC=[HL]
POP B	C1	——	10	Pop BC	BC=[SP]+
POP D	D1	——	10	Pop DE	DE=[SP]+
POP H	E1	——	10	Pop HL	HL=[SP]+
POP PSW	F1	——	10	Pop Processor Status Word	{PSW,A}=[SP]+
PUSH B	C5	——	12	Push BC	-[SP]=BC
PUSH D	D5	——	12	Push DE	-[SP]=DE
PUSH H	E5	——	12	Push HL	-[SP]=HL
PUSH PSW	F5	——	12	Push Processor Status Word	-[SP]={PSW,A}
RAL	17	—*	4	Rotate Accumulator Left	A={CY,A}<-
RAR	1F	—*	4	Rotate Accumulator Righ	A=->{CY,A}
RET	C9	——	10	Return	PC=[SP]+
RC	D8	——	6	Return on Carry	If CY=1(12~s)
RIM	20	——	4	Read Interrupt Mask	A=mask
RM	F8	——	6	Return on Minus	If S=1 (12~s)
RNC	D0	——	6	Return on No Carry	If CY=0(12~s)
RNZ	C0	——	6	Return on No Zero	If Z=0 (12~s)
RP	F0	——	6	Return on Plus	If S=0 (12~s)
RPE	E8	——	6	Return on Parity Even	If P=1 (12~s)

Câu lệnh mã ngữ	Op	SZAPC	~s	Mô tả	Ghi chú
RPO	E0	----	6	Return on Parity Odd	If P=0 (12~s)
RZ	C8	----	6	Return on Zero	If Z=1 (12~s)
RLC	07	---*	4	Rotate Left Circular	A=A<-
RRC	0F	---*	4	Rotate Right Circular	A=->A
RST z	C7	----	12	Restart (3X7)	-[SP]=PC,PC=z
SBB r	9F	*****	4	Subtract with Borrow	A=A-r-CY
SBB M	9E	*****	7	Subtract with Borrow	A=A-[HL]-CY
SBI n	DE	*****	7	Subtract with Borrow Immed	A=A-n-CY
SHLD a	22	----	16	Store HL Direct	[a]=HL
SIM	30	----	4	Set Interrupt Mask	mask=A
SPHL	F9	----	6	Move HL to SP	SP=HL
STA a	32	----	13	Store Accumulator	[a]=A
STAX B	02	----	7	Store Accumulator indirect	[BC]=A
STAX D	12	----	7	Store Accumulator indirect	[DE]=A
STC	37	---1	4	Set Carry	CY=1
SUB r	97	*****	4	Subtract	A=A-r (22X)
SUB M	96	*****	7	Subtract Memory	A=A-[HL]
SUI n	D6	*****	7	Subtract Immediate	A=A-n
XCHG	EB	----	4	Exchange HL with DE	HL<->DE
XRA r	AF	**0*0	4	Exclusive OR Accumulator	A=Axr (25X)
XRA M	AE	**0*0	7	Exclusive OR Accumulator	A=Ax[HL]
XRI n	EE	**0*0	7	Exclusive OR Immediate	A=Axn
XTHL	E3	----	16	Exchange stack Top with HL	[SP]<->HL
PSW		-*01		Flag unaffected/affected/reset/set	
S		S		Sign (Bit 7)	
Z		Z		Zero (Bit 6)	
AC		A		Auxilary Carry (Bit 4)	
P		P		Parity (Bit 2)	
CY		C		Carry (Bit 0)	
a p				Direct addressing	

Câu lệnh mã ngữ	Op	SZAPC	~s	Mô tả	Ghi chú
M z				Register indirect addressing	
n nn				Immediate addressing	
r				Register addressing	
DB n(,n)				Define Byte(s)	
DB 'string'				Define Byte ASCII character string	
DS nn				Define Storage Block	
DW nn(,nn)				Define Word(s)	
A B C D E H L				Registers (8-bit)	
BC DE HL				Register pairs (16-bit)	
PC				Program Counter register (16-bit)	
PSW				Processor Status Word (8-bit)	
SP				Stack Pointer register (16-bit)	
a nn				16-bit address/data (0 to 65535)	
n p				8-bit data/port (0 to 255)	
r				Register (X=B,C,D,E,H,L,M,A)	
z				Vector (X=0H, 8H, 10H, 18H, 20H, 28H, 30H, 38H)	
+ -				Arithmetic addition/subtraction	
& ~				Logical AND/NOT	
v x				Logical inclusive/exclusive OR	
<- ->				Rotate left/right	
<->				Exchange	
[ ]				Indirect addressing	
[ ]+ -[ ]				Indirect address auto-inc/decrement	
{ }				Combination operands	
(X)				Octal op code where X is a 3-bit code	
If (~s)				Number of cycles if condition true	

TÀI LIỆU THAM KHẢO

1. Bill Rieken, Lyle Weiman: *Advantures in Unix Network Applications Programming*, John Wwiley & Sons, Inc, 1992.
2. Charles M.Gilmore: *Micrprocessors Principles and Applications*, McGRAW-HILL International Editions, 1995.
3. Daniel Tabak: *Advanced Microprocessors*, McGRAW-HILL International Editions, 1995.
4. John P.Hayes: *Computer Architecture and Organization*, McGRAW-HILL International Editions, 1998.
5. Richard F. Ferraro: *Programmer's Guide to the EGA and VGA Cards*, Addison-Wesley Publishing Company, 1988.
6. R. P. Paul: *SPARC Architecture, Assembly Language Programming, and C*, Prentice-Hall, Englewood Cliffs, NJ, 1994.
7. Compaq Computer Corp.: *AlphaServer 1200, Compaq Tru64 UNIX*, 1999.
8. H. S. Stone: *High-Performance Computer Architecture*, Addition-Wesley, Reading, MA, 1993.
9. E. Strauss: *Inside the 80286*, Brady/Prentice-Hall, New York, 1986.
10. Roy W. Goody: *Intel Microprocessors*, McGRAW-HILL International Editions, 1993.
11. Hewlett Packard Corp.: *HP9000 Computer*, HP-UX Reference Release 10.0, 1995.
12. Microsoft Corp.: *MS-DOS and Windows file systems*, 2002.
13. Analog Devices Cor.: *DSP and ADSP-21XX*, 2002.
14. IBM Corp.: *RISC system/6000 Technology*, 1990.
15. Cisco System, Inc.: *First-Year Companion Guide*, 2001.
16. Clements: *68000 Family Assembly Language*, PWS Publ. Co., Bosston, MA, 1994.
17. www.Intel.com: *Intel Microprocessors technology*, 2004.

MỤC LỤC

Lời giới thiệu 5

Lời nói đầu 7

Chương 5: **Ghép nối CPU với bộ nhớ và các thiết bị ngoại vi** 9

5.1. Các tín hiệu của bộ vi xử lý và các mạch phụ trợ 9

5.1.1. Các tín hiệu của vi xử lý 8086/8088 và các mạch phụ trợ 9

5.1.2. Kết nối các mạch phụ trợ với CPU 8086/8088 17

5.1.3. Kết nối các mạch phụ trợ trong các hệ thống 80x86 và pentiums 22

5.1.4. Các loại bus của máy tính 41

5.2. Phối ghép CPU với bộ nhớ 54

5.2.1. Kết nối RAM/ROM trong hệ thống 8085 54

5.2.2. Kết nối RAM/ROM trong hệ thống 8086/8088/ 80286/80386SX 59

5.2.3. Kết nối bộ nhớ trong hệ thống 80386DX/80486 75

5.2.4. Kết nối bộ nhớ trong các hệ thống Pentiums 77

5.2.5. Độ rộng băng của bus bộ nhớ 80

5.3. Phối ghép CPU với thiết bị ngoại vi 82

5.3.1. Các hình thức phối ghép vào/ra 82

5.3.2. Vào ra song song 87

5.3.3. Vào ra nối tiếp 99

5.3.4. UART - Mạch thu phát không đồng bộ đa năng 101

5.3.5. USART - Mạch thu phát đồng bộ/không đồng bộ đa năng 116

5.3.6. Phối ghép với bàn phím 126

5.3.7. Phối ghép với màn hình 132
5.3.8. Phối ghép với thế giới tương tự 161
5.3.9. Một số phối ghép khác 176
Bài tập ôn luyện 197
Chương 6: **Các phương pháp điều khiển vào/ra dữ liệu** 202
6.1. Giới thiệu các phương pháp điều khiển vào/ra dữ liệu 202
6.1.1. Đặc điểm điều khiển vào/ra bằng phương pháp thăm dò 203
6.1.2. Đặc điểm điều khiển vào/ra bằng phương pháp ngắt 204
6.1.3. Đặc điểm điều khiển vào/ra bằng phương pháp DMA 205
6.2. Vào/ra dữ liệu bằng phương pháp thăm dò 206
6.2.1. Vào/ra dữ liệu bằng phương pháp thăm dò không theo ưu tiên 206
6.2.2. Vào/ra dữ liệu bằng phương pháp tham dò theo ưu tiên 207
6.3. Vào/ra dữ liệu bằng phương pháp ngắt 209
6.3.1. Hệ thống ngắt theo quét 210
6.3.2. Hệ thống ngắt theo véc-tơ 211
6.3.3. Hệ thống ngắt theo chuỗi liên tiếp 212
6.3.4. Các bước xử lý ngắt và cấu trúc ngắt của 8085 213
6.3.5. Vi mạch điều khiển ngắt 8259A 222
6.3.6. Ngắt không che được 237
6.4. Vào/ra dữ liệu bằng DMA 238
6.4.1. Quá trình điều khiển DMA 238
6.4.2. Vi mạch DMAC 8237 240
Bài tập ôn luyện 265

Chương 7: **Giới thiệu các bộ vi xử lý công nghệ tiên tiến** 267
7.1. Sơ đồ khối cấu trúc chung của các bộ vi xử lý công nghệ tiên tiến 267
7.1.1. Đơn vị giao tiếp bus (BIU) 268
7.1.2. Khối tiền đọc lệnh và hàng lệnh (PUIQ) 269
7.1.3. Khối chức năng đặc biệt (SFU) 270
7.1.4. Bộ nhớ cache 270
7.1.5. Đơn vị điều khiển (CU) 272
7.1.6. Đơn vị nguyên (IU) 273
7.1.7. Đơn vị dấu phẩy động (FPU) 274
7.1.8. Đơn vị quản lý bộ nhớ (MMU) 275
7.2. Các đặc điểm kiến trúc của vi xử lý công nghệ tiên tiến ... 276
7.2.1. Khái niệm về xử lý song song 276
7.2.2. Các kiểu của bộ xử lý song song 278
7.2.3. Hiệu suất của xử lý song song 288
7.2.4. Các bộ xử lý cấu trúc đường ống 290
7.2.5. Điều khiển đường ống và đồ thị không gian thời gian 297
7.2.6. Các chướng ngại trong kiến trúc đường ống 298
7.2.7. Song song ở mức lệnh (ILP) 304
7.2.8. Đa xử lý (multiprocessors) 306
7.3. Công nghệ RISC 311
7.3.1. RISC và CISC 311
7.3.2. Các đặc điểm của RISC 312
7.3.3. Các ưu điểm của RISC 319
7.3.4. Các nhược điểm của RISC 321
7.4. Các bộ vi xử lý tiên tiến công nghệ RISC của một số hãng 321
7.4.1. Họ vi xử lý Intel i860 322
7.4.2. Họ vi xử lý Motorola M88000 324
7.4.3. Họ vi xử lý Sun SPARC 329

7.4.4. Họ vi xử lý PowerPC.....351
7.4.5. Họ vi xử lý DEC alpha.....377
7.4.6. Họ vi xử lý HP Precision.....407
7.4.7. So sánh các họ vi xử lý.....426
7.5. Hệ phát triển.....431
7.5.1. Thế nào là phát triển hệ thống.....431
7.5.2. Hệ phát triển máy vi tính (MDS).....434
7.5.3. Các hệ phát triển các hệ thống vi xử lý Pentium.....440
Bài tập ôn luyện.....447
Chương 8. **Bộ vi điều khiển**.....453
8.1. Giới thiệu chung về vi điều khiển.....453
8.1.1. Thế nào là vi điều khiển.....453
8.1.2. Một số chip vi điều khiển của Intel.....453
8.2. Cấu trúc và hoạt động của bộ vi điều khiển.....457
8.2.1. Cấu trúc của Intel 8051.....457
8.2.2. Sắp xếp RAM bên trong 8051.....460
8.2.3. Các thanh ghi chức năng đặc biệt bên trong 8051.....462
8.2.4. Các loại cổng vào/ra của 8051.....465
8.2.5. Giao tiếp truyền thông tuần tự của 8051.....469
8.2.6. Các bộ đếm-thời gian trong 8051.....471
8.2.7. Hệ thống ngắt của 8051.....475
8.3. Tập lệnh và lập trình trên bộ vi điều khiển.....478
8.3.1. Tập lệnh của 8051.....478
8.3.2. Chu kỳ máy của 8051.....482
8.3.3. Lập trình 8051.....482
8.3.4. Cấu trúc của các bộ vi điều khiển 8048, 8049, 8050.....486
Bài tập ôn luyện.....489

Phần phụ lục 495

Phụ lục A: **Bảng mạch chính của IBM PC/XT 88 và bảng mạch vào/ra đa năng** 497

Phụ lục B: **Tập lệnh Intel 8086** 504

Phụ lục C: **Tập lệnh của các vi xử lý công nghệ tiên tiến Intel X86** 513

Phụ lục D: **Tập lệnh của Z80** 540

Phụ lục E: **Tập lệnh của các vi xử lý Motorola M680X0** 555

Phụ lục F: **Tập lệnh của vi điều khiển Intel 8051** 562

Phụ lục G: **Tập lệnh của Intel 8085A** 570

Tài liệu tham khảo

GIÁO TRÌNH
Kỹ thuật vi xử lý
(TẬP 2)

Chịu trách nhiệm xuất bản
LƯU ĐỨC VĂN

Biên tập: TRẦN VŨ THƯỞNG
NGÔ MỸ HẠNH

Chế bản: NGUYỄN MẠNH HOÀNG

Sửa bản in: NGÔ MỸ HẠNH

Trình bày bìa: TRỊNH THẾ VINH

(Giáo trình này được ban hành kèm theo Quyết định số 192/QĐ-QLNCKH ngày 15/4/2005 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông)

NHÀ XUẤT BẢN BƯU ĐIỆN

Trụ sở: 18 - Nguyễn Du, Hà Nội
Điện thoại: 04-9432438, 9431284 Fax: 04-9431285
E-mail: bientap@hn.vnn.vn Website: www.nxbbuudien.com.vn

Chi nhánh TP. HCM: 27 - Nguyễn Bỉnh Khiêm, Quận 1, TP.Hồ Chí Minh
Điện thoại: 08-9100925 Fax: 08-9100924
E-mail: chinhanh-nxbbd@hcm.vnn.vn

Chi nhánh TP. Đà Nẵng: 42 - Trần Quốc Toản, Quận Hải Châu, TP. Đà Nẵng
Điện thoại: 0511-897467 Fax: 0511-897467
E-mail: pnbich@mpt.gov.vn

In 300 bản, khổ 16 x 24 cm, tại Công ty Cổ phần In Bưu điện
Số đăng ký kế hoạch xuất bản 62-2007/CXB/48_3 - 03/BuĐ
Số quyết định xuất bản: 116/QĐ-NXB BĐ ngày 14/6/2007
In xong và nộp lưu chiểu tháng 6 năm 2007.

NHÀ XUẤT BẢN BƯU ĐIỆN

Trụ sở chính: 18 Nguyễn Du - Hà Nội

ĐT Biên tập: 04.9431284 - ĐT Phát hành: 04.9432438

Fax: 04.9431285 E-mail: bientap@hn.vnn.vn

Website: www.nxbbuudien.com.vn

Chi nhánh TP.HCM: 27 Nguyễn Bỉnh Khiêm - Quận I - TP. Hồ Chí Minh

Điện thoại: 08.9100925 Fax: 08.9100924

E-mail: chinhanh-nxbbd@hcm.vnn.vn

Chi nhánh TP. Đà Nẵng: 42 - Trần Quốc Toản, TP. Đà Nẵng

Điện thoại: 0511.897467 Fax: 0511.897467

E-mail: pnbich@mpt.gov.vn

MỜI CÁC BẠN TÌM ĐỌC

1. TỪ ĐIỂN THUẬT NGỮ VIỄN THÔNG ANH - VIỆT
2. CÁC DỊCH VỤ VIỄN THÔNG CỦA TỔNG CÔNG TY BƯU CHÍNH VIỄN THÔNG VIỆT NAM
3. TRUYỀN HÌNH SỐ MẶT ĐẤT
4. CÔNG NGHỆ VoIP VÀ CÁC GIẢI PHÁP NÂNG CAO CHẤT LƯỢNG DỊCH VỤ
5. TIẾNG ANH CHUYÊN NGÀNH ĐIỆN TỬ - VIỄN THÔNG
6. CÁC HỆ THỐNG NGUỒN ĐIỆN, CHỐNG SÉT VÀ TIẾP ĐẤT TRẠM VIỄN THÔNG: NGUYÊN LÝ HOẠT ĐỘNG, QUẢN LÝ, VẬN HÀNH VÀ BẢO DƯỠNG
7. ĂNG TEN THÔNG MINH - GIẢI PHÁP CHO THÔNG TIN DI ĐỘNG TRONG TƯƠNG LAI
8. KỸ THUẬT ĐO KIỂM MẠNG VIỄN THÔNG
9. MẠNG NGOẠI VI - THIẾT KẾ, THI CÔNG VÀ QUẢN LÝ

SÁCH CỦA NHÀ XUẤT BẢN BƯU ĐIỆN CÓ BÁN TẠI:

1. Trung tâm Dịch vụ văn hóa Tiền phong VDC
   292 Tây Sơn, Hà Nội
2. Nhà sách Bách Khoa
   Số 1, Đường Giải Phóng, Hà Nội
   86 - 107 Tô Hiến Thành, Quận 10, TP.HCM
3. Nhà sách Thăng Long
   2 Bis Nguyễn Thị Minh Khai, Quận 1, TP.HCM

Giá: 78.000 đ