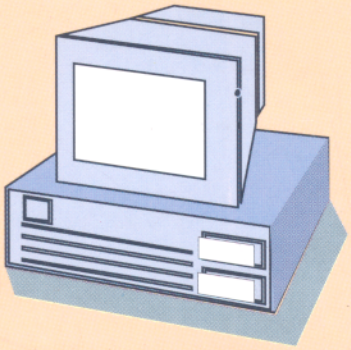
NGUYỄN MẠNH GIANG

# LẬP TRÌNH BẰNG NGÔN NGỮ ASSEMBLY CHO MÁY TÍNH PC-IBM



NHÀ XUẤT BẢN GIÁO DỤC

TS. Nguyễn Mạnh Giang

# LẬP TRÌNH BẰNG NGÔN NGỮ ASSEMBLY CHO MÁY TÍNH PC -IBM

*(Tái bản lần thứ ba)*

NHÀ XUẤT BẢN GIÁO DỤC

# LỜI NÓI ĐẦU

*Máy tính điện tử nói chung, máy vi tính nói riêng, thực hiện chuỗi lệnh của chương trình được ghi sẵn trong khối nhớ. Máy chỉ hiểu và thực hiện lệnh dưới dạng mã máy, tức là tổ hợp các bit nhị phân (dạng 0, 1), nhưng người sử dụng gặp khó khăn khi dùng mã máy vì khó nhớ, phức tạp dễ nhầm lẫn. Do đó đã xuất hiện ngôn ngữ bậc cao hơn, hợp ngữ (Assembly). Hợp ngữ này gồm một tập hợp các ký tự (chữ cái) để đặt tên cho từng lệnh (hành động) của từng máy theo quy tắc thuật nhớ (memonic) - tên hành động là 3 hay 4 chữ cái đầu tiếng Anh của hành động đó. Mỗi máy tính có cách đặt tên riêng nhưng có đặc điểm chung là ngắn gọn, gợi nhớ.*

*Hợp ngữ rất thuận tiện đối với người dùng (dễ nhớ, dễ viết) nhưng máy tính điện tử không thể hiểu được. Do đó, phải có chương trình dịch (gọi là hợp dịch - Assembler) để phiên dịch hợp ngữ ra ngôn ngữ máy cho máy tính hiểu được.*

*Mỗi loại máy tính đều có dạng hợp ngữ riêng và chương trình hợp dịch riêng của mình. Ví dụ, máy vi tính PC AT, PS/2 của hãng IBM có hai chương trình hợp dịch là Macro Assembler (MASM) của hãng Microsoft và Turbo Assembler (TASM) của hãng Borland.*

*Cuốn giáo trình* ***"Lập trình bằng ngôn ngữ assembly cho máy tính PC-IBM"*** *được viết trên cơ sở chương trình của môn học "Vi xử lý và ngôn ngữ Assembly" cho sinh viên năm thứ 4 ngành Điện tử - Viễn thông và Tin học Vật lý của trường Đại học Bách Khoa Hà Nội.*

*Giáo trình gồm 7 chương, chia làm 2 phần:*

*-* ***Phần lập trình cơ sở****: các chương 1, 2, 3, 4 giúp người đọc nắm được những kiến thức cơ sở về thiết bị (phần cứng) và hệ thống chương trình (phần mềm) của một máy vi tính PC-IBM, bộ lệnh dạng hợp ngữ của chúng và các lệnh của chương trình Debug của hệ điều hành MS-DOS, một công cụ đơn giản để soạn thảo, sửa chữa lỗi và cho chạy chương trình dạng hợp ngữ.*

*-* ***Phần lập trình cho các thiết bị ngoài****: các chương 5, 6, 7 giúp người đọc lập trình những chương trình ngắn cho các thiết bị ngoài thông dụng (bàn phím, màn hình, chuột), cho các thiết bị ngoài ghép nối (trao đổi tin song song, nối tiếp) và lập trình thời gian, âm thanh trên cơ sở các lệnh Debug của MS-DOS.*

*Giáo trình được trình bày dựa trên các nguyên tắc sau:*

*- Tóm tắt những điểm cơ bản về phần cứng phục vụ cho phần mềm để dễ dàng lập trình trực tiếp và lập trình dùng ngắt INT nh có sẵn của hệ điều hành MS-DOS để nghiên cứu máy vi tính PC-IBM (hệ máy, các vi mạch vi xử lý và vi mạch phù trợ, các thiết bị ngoài thông dụng và ghép nối, bộ nhớ, đĩa).*

*- Sử dụng các lệnh của chương trình Debug của MS-DOS làm công cụ lập trình (soạn thảo, hợp dịch và chạy chương trình). Người lập trình chỉ cần biết một số lệnh cần thiết của Debug là có thể soạn thảo và cho chạy thử chương trình. Khi đã quen với phương pháp lập*

*trình đơn giản, người lập trình thấy dễ dàng khi lập trình với những bài toán phức tạp, dài với công cụ là chương trình hợp dịch (MASM, TASM).*

*- Thông qua các ngắt của hệ điều hành, giúp bạn đọc làm quen với phần mềm hệ thống và cách lập trình (trực tiếp và dùng ngắt cho chúng) bằng hợp ngữ cho máy vi tính PC-IBM. Những chương trình ngắn này giúp bạn đọc thám hiểm các tính năng của máy vi tính (cả phần mềm lẫn phần cứng) và cho chạy các ví dụ giống như các trò chơi nhỏ trên máy vi tính.*

*Trong quá trình trình bày, tác giả cố gắng dựa trên mục tiêu "cơ bản, đơn giản và hiện đại" để giúp cho các bạn sinh viên và người tự học có thể hiểu được tính năng của máy tính, có thể sử dụng các chương trình con một cách hiệu quả vào việc lập trình cho công việc của mình sau này.*

*Giáo trình có thể giúp ích cho các bạn đọc là thầy giáo, sinh viên ngành Điện tử - Tin học và các ngành khác, cả những người tự học về lập trình bằng hợp ngữ. Các ví dụ trong tài liệu có thể được sử dụng làm các bài thí nghiệm trên máy, các bài tập mẫu trong giờ bài tập và các chương trình con trong các ứng dụng khác. Các ví dụ đã được chạy thử và làm thí nghiệm trên máy. Phần câu hỏi của tài liệu giúp bạn đọc hệ thống hóa và tự kiểm tra kiến thức của mình. Phần bài tập giúp bạn đọc tự viết chương trình và cho chạy thử trên máy dựa trên cơ sở các ví dụ đã cho.*

*Vì kiến thức và thời gian biên soạn, thử nghiệm có hạn, chắc chắn giáo trình còn có thiếu sót, mong các bạn đọc thông cảm và góp ý kiến. Mọi ý kiến đóng góp xin gửi về: Bộ môn Vật lý Tin học, Viện Vật lý kỹ thuật, Trường Đại học Bách Khoa Hà Nội.*

*Tác giả xin vô cùng cảm ơn các đồng chí lãnh đạo trường Đại học Bách Khoa Hà Nội, lãnh đạo Khoa Điện tử - Viễn thông, lãnh đạo Viện Vật lý kỹ thuật và các đồng nghiệp đã khích lệ, đóng góp nhiều ý kiến quý báu cho quyển sách này. Tác giả cũng xin cảm ơn một số anh em sinh viên trường ĐHBK đã cung cấp tài liệu, thử nghiệm giáo trình này.*

***Tác giả***

*CHƯƠNG 1*

# TÓM TẮT VỀ PHẦN CỨNG CỦA HỌ MÁY VI TÍNH PC-IBM

Máy vi tính cá nhân PC (Personal Computer) của hãng IBM PC/XT ra đời tháng 12 năm 1981 với vi xử lý Intel 8086. Tới nay, máy vi tính của hãng trên đã trải qua nhiều thế hệ với các vi xử lý khác nhau 8086, 8088, 80186, 80286, 80386, 80486 và 80586 Pentium và các máy vi tính PC/AT(80286), PS/2(80386) Pentium I, Pentium II đang được sử dụng rộng rãi trên thế giới.

Muốn lập trình bằng hợp ngữ (assembly) cho các máy vi tính (MVT) PC-IBM, ta cần hiểu rõ về cấu trúc của chúng. Tuy là ngôn ngữ bậc cao hơn ngôn ngữ máy (dạng nhị phân 0,1), nhưng hợp ngữ có liên hệ chặt chẽ với phần cứng, các chương trình luôn phục vụ cho các thành phần thiết bị và xử lý các số liệu chứa trong khối nhớ.

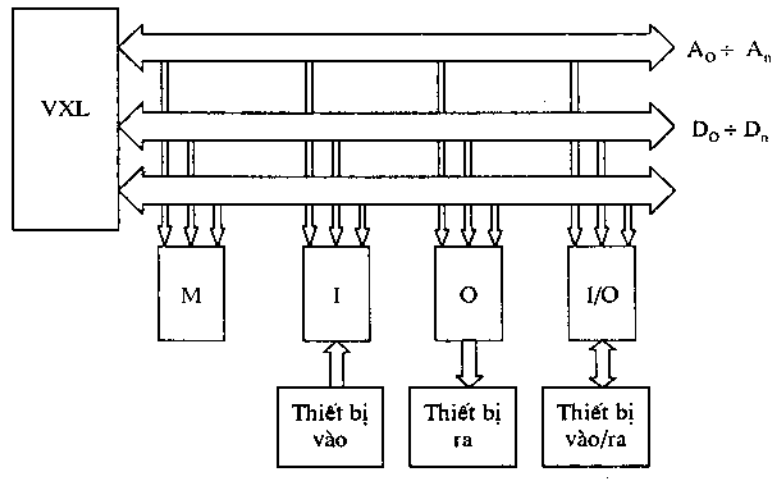
Chương này cung cấp cho người đọc những kiến thức cơ bản về kỹ thuật MVT hay phần cứng của máy, giúp cho người đọc dễ hiểu các hành động của chương trình bằng hợp ngữ.

Chúng ta sẽ điểm qua những nét chủ yếu của các thành phần cấu trúc của MVT như vi xử lý, khối nhớ, cổng vào/ra, các thiết bị ngoài, các linh kiện phụ trợ.

## 1.1. ĐẠI CƯƠNG VỀ CẤU TRÚC MÁY VI TÍNH PC

MVT là loại máy nhỏ nhất (về kích thước và tính năng), trong họ máy tính điện tử (MTĐT) sau máy tính lớn (main frame) và máy tính nhỏ (mini computer).

MVT nói riêng, MTĐT nói chung đều có cấu trúc như hình 1.1, gồm có khối xử lý trung tâm (CPU) hay vi xử lý (VXL), khối nhớ (M), các cổng vào/ra (I/O) để nối với các thiết bị ngoài (TBN các thiết bị đưa tin vào, đưa tin ra và đưa tin vào/ra). Các khối trên cùng các linh kiện phụ trợ liên hệ với nhau qua đường dây (bus) địa chỉ ($A_0 \div A_n$), số liệu ($D_0 \div D_n$) và điều khiển (C).




**Hình 1.1.** Cấu trúc một máy vi tính.

MVT hoạt động theo nguyên tắc của Von Neumann:

- Số liệu và chuỗi lệnh (chương trình) dạng số được tích trong khối nhớ.

- Lệnh được đọc vào VXL, giải mã và thực hiện cùng với số liệu trong các hành động xử lý toán học (cộng, trừ, nhân, chia) hoặc logic (và, hoặc, đảo, loại trừ v.v...).

- Các kết quả xử lý có thể đưa ra hiển thị hoặc lưu trữ trong các thiết bị đưa tin ra qua cổng ra.

- Số liệu hoặc chương trình có thể đưa vào (ghi tin trong khối nhớ hoặc trong VXL) qua các thiết bị đưa tin vào qua cổng vào.

## 1.2. CẤU TRÚC CỦA VI XỬ LÝ INTEL 80X86

### 1.2.1. CẤU TRÚC CHUNG CỦA MỘT BỘ VI XỬ LÝ

Một bộ VXL đơn tinh thể có cấu trúc tổng quát như hình 1.2.

VXL loại này gồm:

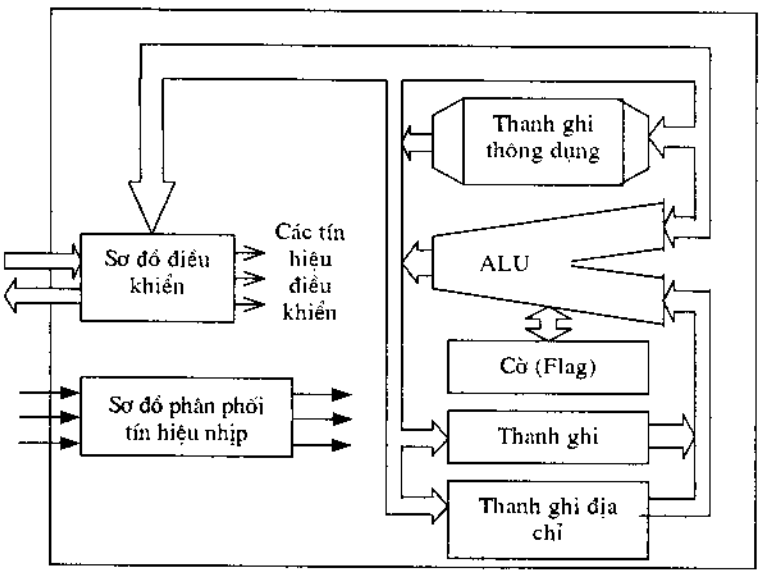
- *Khối điều khiển*: liên hệ với đường dây điều khiển chế độ (ngắt, trao đổi trực tiếp khối nhớ), điều khiển lệnh (đọc, ghi), trạng thái (sẵn sàng, xoá về 0...) để định các chế độ, giải mã và thực hiện các hành động của lệnh chương trình.

- *Khối phân bố các tín hiệu nhịp*: liên hệ với các tín hiệu của xung nhịp của đồng hồ để đưa các tín hiệu nhịp cần thiết cho các bộ phận của VXL.

- *Khối số học - logic* (ALU): xử lý số học và logic.

- *Khối các thanh ghi nội của VXL*: thanh ghi lệnh, thanh ghi cờ (flag), thanh ghi tích lũy (Acc) hay thanh ghi A và các thanh ghi đệm thông dụng (B, C, D, E, F, G).

- *Các đường dây (bus)*: địa chỉ, số liệu và điều khiển (chế độ, lệnh, trạng thái).

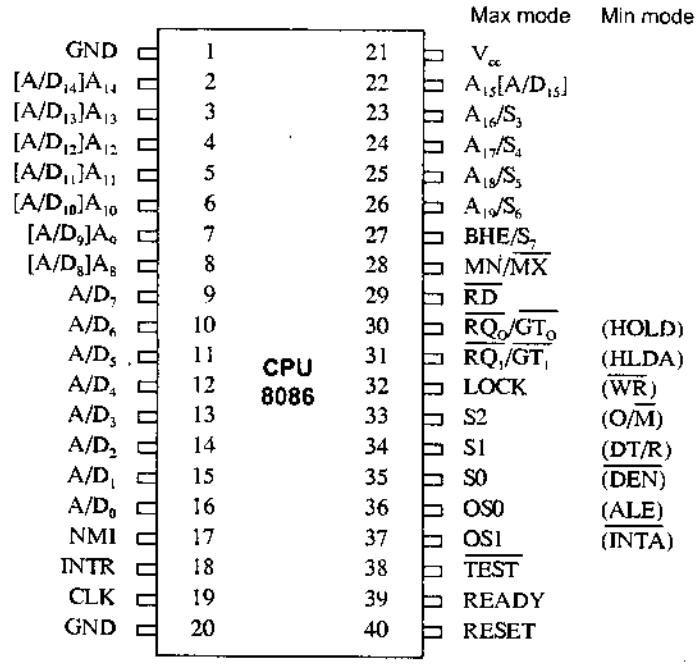



**Hình 1.2.** Cấu trúc của bộ VXL.

### 1.2.2. VI XỬ LÝ 8086/8088

VXL 8086/8088 là thế hệ sau của 4004, 8080, 8085 do hãng Intel chế tạo. Các VXL 8086 và 8088 có cùng một cấu trúc và hệ lệnh, chỉ có các điều khác biệt là:

- 8088 chỉ có 8 đường dây số liệu ($D_0 \div D_7$), còn 8086 có 16 đường dây số liệu ($D_0 \div D_{15}$).

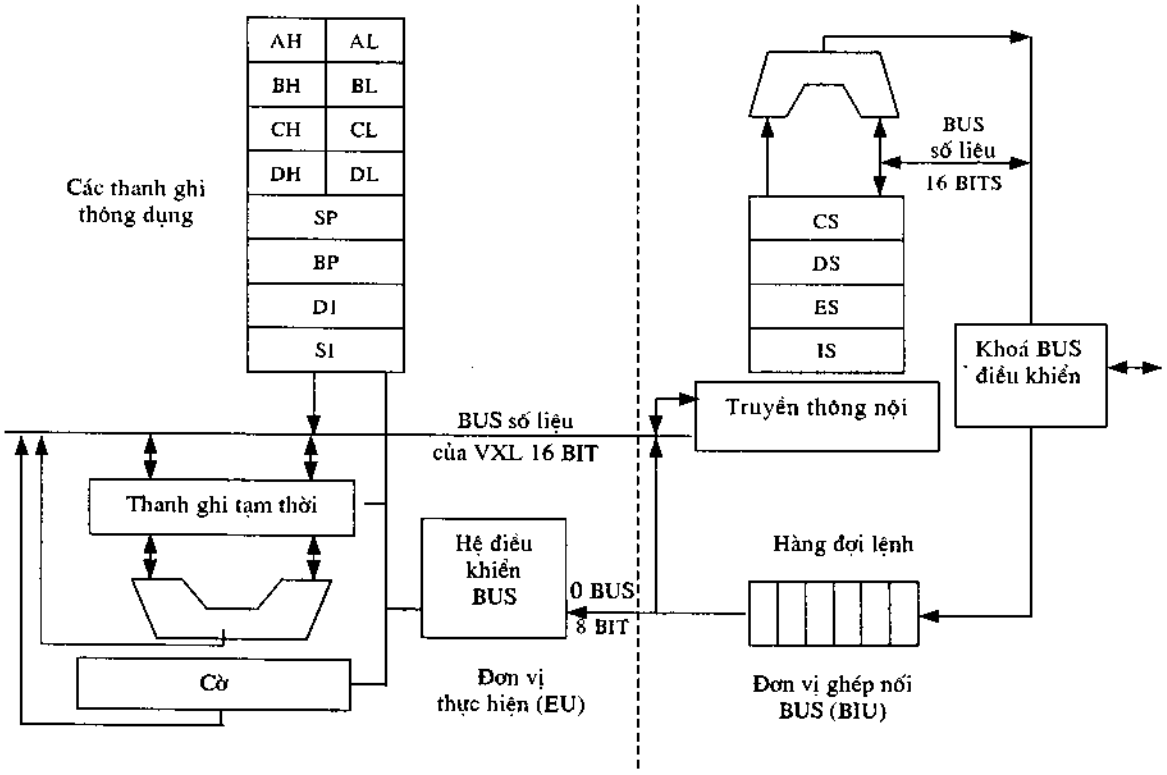



**Hình1.3.** Sơ đồ chân và tín hiệu của vi xử lý 8086/8088.

- 8088 có 4 hàng đợi lệnh trong khi 8086 có 6 hàng.

Hình 1.3 giới thiệu chân và tín hiệu của 8086.

Hình 1.4 giới thiệu sơ đồ khối của 8086. Khác với 8085, 8086/8088 có các tệp đợi lệnh để chứa nội dung các byte lệnh từ bộ nhớ trung tâm đưa vào giải mã và thực hiện trong VXL. 8086/8088 giống 8085 ở đơn vị thực hiện lệnh gồm khối điều khiển, các thanh ghi thông dụng (AX, BX, CX, DX), thanh ghi Stack, thanh ghi cờ và khối số học - logic (ALU). Khác với 8085 là VXL 8086/8088 có thêm các thanh ghi cơ sở (BP) và các thanh ghi chỉ thị nguồn (SI) và đích (DI) của số liệu.




**Hình 1.4.** Sơ đồ khối của 8086/8088.

8086/8088 có thêm đơn vị phối ghép đường dây (BIU - Bus Interface Unit) gồm:

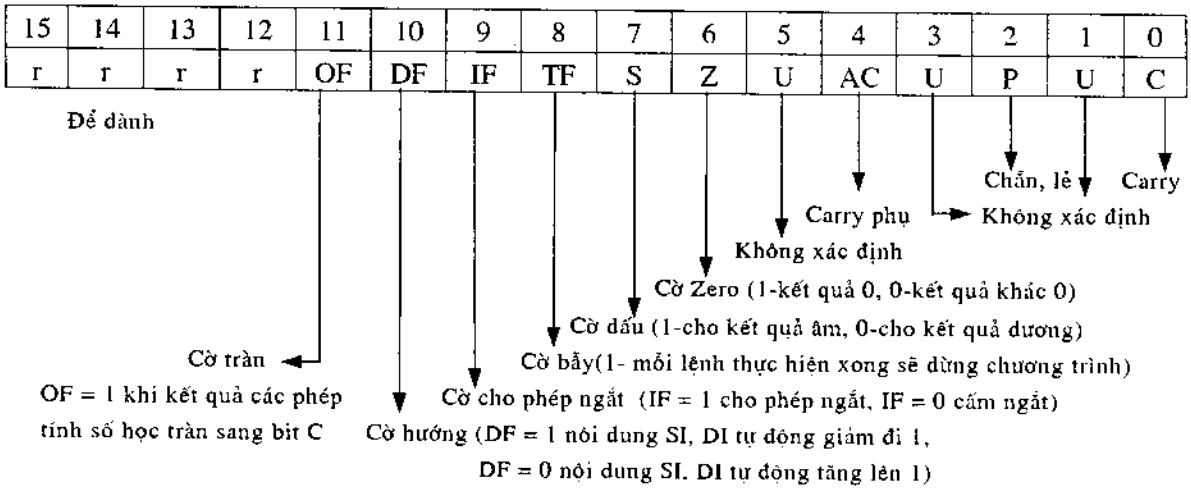
- Các thanh ghi mảng số liệu:
  + CS (mảng mã lệnh - Code Segment).
  + ES (mảng số liệu phụ - Extra Segment).
  + SS (mảng Stack - Stack Segment).
  + DS (mảng số liệu - Data Segment).

Các thanh ghi mảng này chỉ địa chỉ các mảng nhớ dành cho các loại chương trình (CS), số liệu (ES, DS) và lưu trữ số liệu (SS).

Thanh ghi lệnh (IP) (giống 8085) dùng để ghi địa chỉ lệnh trong mảng mã lệnh (CS), (10hxCS + IP).

Bộ tính địa chỉ: lấy tổng nội dung của một trong các thanh ghi mảng được dịch chuyển về phía trái 4 vị trí nhị phân với độ lệch (offset) là giá trị tuyệt đối của địa chỉ trong từng mảng lệnh.

Thanh ghi cờ (flag): của 8086/8088 có các bit như hình 1.5.




**HÌNH 1.5.** Thanh ghi trạng thái của 8086/8088.

## 1.2.3. VI XỬ LÝ 80386

Họ 80386 có các phiên bản:

- **80386DX** (10/1985, 40 MHz; 6 MIPS; 0,275 triệu tranzito, không có nhớ ngầm).

- **80386SX** (6/1988, 33 MHz; 2,5 MIPS; 0,275 triệu tranzito, không có nhớ ngầm).

VXL 80386 có các thanh ghi nội như hình 1.6.

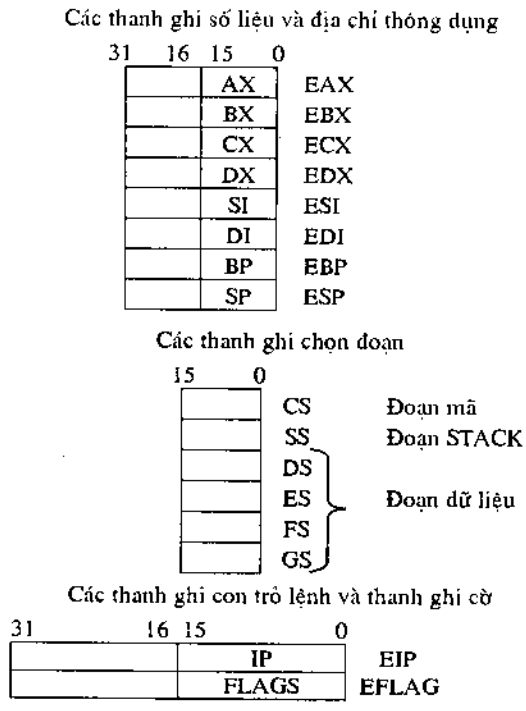
So với VXL Intel 8086, VXL Intel 80386 có thêm:

- Các tín hiệu hội thoại với đồng xử lý toán học (80387) như PEREQ, BUSY, ERROR.

- Số đường dây địa chỉ, số liệu lên tới 32.

- Thêm bộ quản lý trang của khối nhớ. Có chế độ quản lý và bảo vệ bộ nhớ dùng cho các hệ điều hành đa nhiệm (chế độ thực, chế độ bảo vệ, chế độ 8086 ảo).

- Thanh ghi cờ có thêm các bit:




**Hình 1.6.** Các thanh ghi của 80386.

+ *2 bit IOP:* chỉ 4 mức đặc quyền vào/ra (0, 1, 2, 3).

+ *NT* :(Nested task flag) cờ cho phép nhiệm vụ lồng nhau, NT = 1 có một nhiệm vụ đang hoạt động trong một nhiệm vụ khác.

+ *RF* :(Resume flag) cờ thu hồi lệnh, lại khởi động một nhiệm vụ khác.

+ *VM* : (Virtual mode) chế độ 8086 ảo khi VM = 1, tức là VXL 386 thực hiện chương trình của nó như VXL 8086.

- Các thanh ghi nội có thêm:

+ *Các thanh ghi mảng GS và FS* : quản lý mảng số liệu trong bộ nhớ cùng với các thanh ghi mảng DS,ES.

+ *Các thanh ghi mô tả mảng GDTR* (mô tả toàn cục), *LDTR* (mô tả cục bộ), *TR* (ghi nhiệm vụ).

+ *Các thanh ghi điều khiển ($CR_0$, $CR_2$, $CR_3$)*: ghi số liệu để điều khiển VXL.

+ *Các thanh ghi gỡ rối ($DR_0 \div DR_7$)* : ghi địa chỉ và trạng thái của các điểm dừng chương trình để kiểm tra lỗi.

+ *Các thanh ghi kiểm tra (thử)*: $TR_6$ (Test control: kiểm tra điều khiển), $TR_7$ (Test Status).

Thanh ghi $CR_0$ chứa 6 bit điều khiển sau:

.PG: (Paging Enable) PG = 1 cho phép đơn vị trang của bộ nhớ hoạt động.

.ET: (Processor Extension) ET = 1 cho phép mở rộng 32 bit để tương thích với 80387 còn ET = 0 chỉ có 16 bit tương thích với 80287.

.TS: (Task Switched) xác lập lên 1 khi thay đổi nhiệm vụ.

.EM: (Emulate Coprocessor) EM = 1 cấm 80387 hoạt động hay gây ngoại lệ 7 (trừ lệnh WAIT) còn EM = 0 cho phép 80387 hoạt động.

.PE: (Protection Enable) PE = 1 cho phép 80386 hoạt động ở chế độ bảo vệ.

## 1.2.4. VI XỬ LÝ 80486

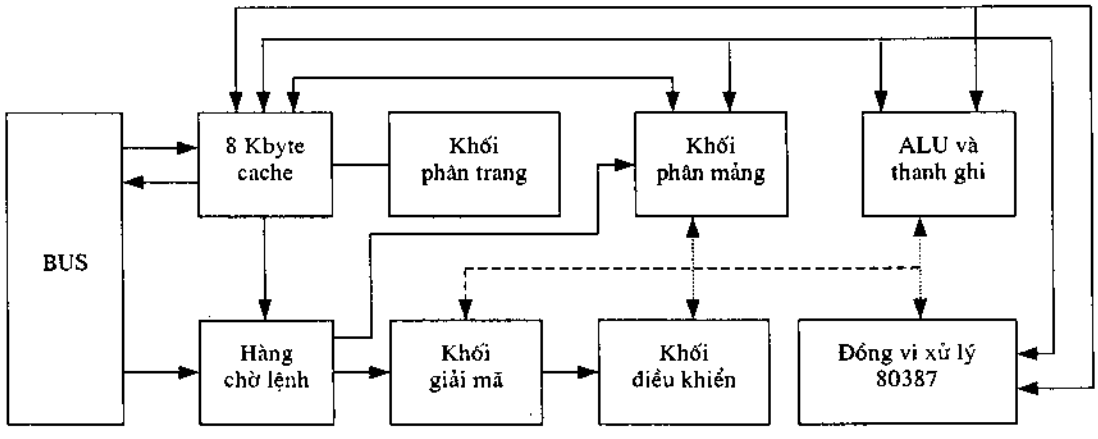
VXL 80486 có ba phiên bản:

- **80486DX**: (4/1989, 50 MHz, 20MIPS; 1,2 triệu tranzito, I/D cache 8KB, 32 bit địa chỉ và số liệu).

- **80486SX**: (4/1991, 25 MHz, 16,5 MIPS; 1,18 triệu tranzito, I/D cache 8KB, 32 bit địa chỉ và số liệu).

- **80486DX2**: (3/1992, 66 MHz, 52 MIPS; 1,2 triệu tranzito, I/D cache 8KB, 32 bit địa chỉ và số liệu).

VXL 80486DX chứa VXL 80386 và VXL 80387, có cấu trúc đường dây như hình 1.8 và cấu trúc bên trong như hình 1.7.




**Hình 1.7.** Sơ đồ khối của 80486.

Riêng VXL 80486 SX không có 80387, giống 80386 nhưng giống 80486 là có thêm hai khối là:

- Khối điều khiển.

- Bộ nhớ ngầm (cache) 8KByte RAM tĩnh để làm tăng tốc độ xử lý lệnh (chỉ cần một chu kỳ xung nhịp để gọi số liệu từ bộ nhớ ngầm hoặc thực hiện hầu hết các lệnh vì ở bên trong VXL, tần số được nhân lên hai hoặc ba lần).

Thanh ghi cờ (Flag) của 80486, ngoài các bit như 80386, còn có thêm các bit sau:

Cờ AC (alignment check-kiểm tra sự cân chỉnh). VXL 80486 bị một sai số cân chỉnh và sinh ra loại trừ 17.

Thanh ghi điều khiển $CR_0$ có thêm 5 bit mới:

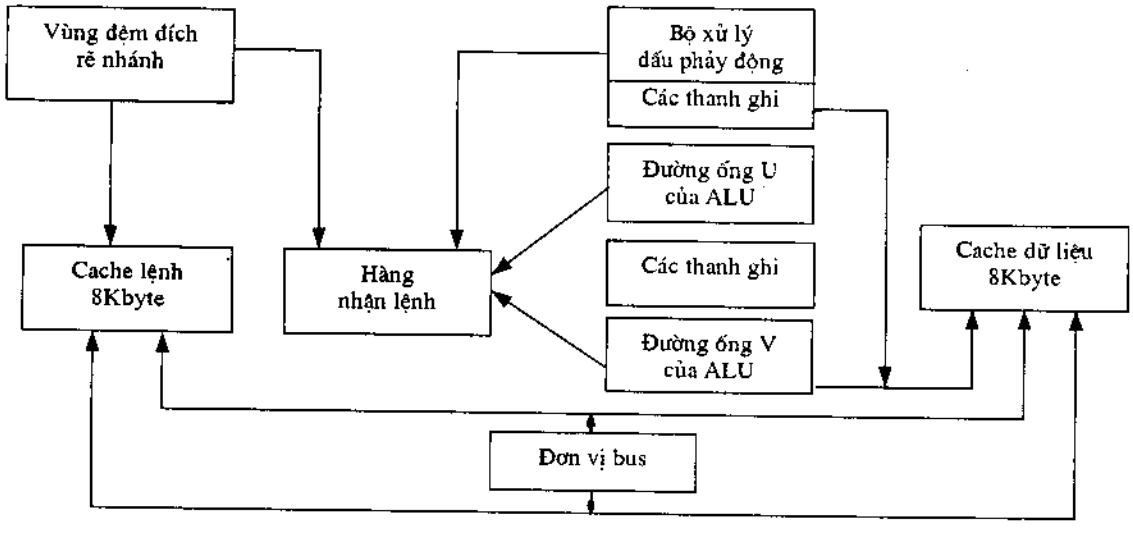
- WP: ($d_{16}$ - write protect) bảo vệ ghi (WP = 1).
- AM: ($d_{18}$ - alignment mask) cấm cân chỉnh (AM = 1).
- NW: ($d_{29}$ - not write through) không thể ghi qua (NW = 1).
- CD: ($d_{30}$ - cache disable) cấm nhớ ngầm (CD = 1).
- PG: ($d_{31}$ - paging enable) cho phép chia trang (PG = 1).

Cũng có sáu lệnh riêng của 80486, ngoài các lệnh giống của 80386 là:

- INVD: cấm khối nhớ ngầm hoạt động.
- WBINVD: cấm khối nhớ ngầm dữ liệu hoạt động.
- INVLPG: cấm khối vào TLB.
- BSWAP: đổi lệnh bytes thành lệnh thanh ghi 32 bit.
- XADD: trao đổi và cộng thanh ghi nguồn và đích.
- CMPXCHG: so sánh và trao đổi các toán hạng nguồn và đích phụ thuộc vào kết quả của sự so sánh.

## 1.2.5. VI XỬ LÝ 80586

VXL 80586 (5/1993,100 MHz,112 MIPS, 3,1 triệu tranzito, 32 bit số liệu và địa chỉ)




**Hình 1.8.** Sơ đồ khối của 80586.

còn gọi là Pentium với kỹ thuật 0,8 micro BI CMOS, tần số nhịp bắt đầu 66MHz và hiện nay tới trên 400MHz, tốc độ hoạt động lớn hơn 110MIPS (million instruction per second). Ngoài hệ lệnh rút gọn RISC (Reduced instruction Set Computer) trong hệ lệnh phức hợp CISC (Complex instruction Set Computer) như 486, Pentium còn có cấu trúc superscale (siêu vô hướng) với hai đường ống (pipeline) U và V hoạt động song song. Với cùng tần số nhịp như 486, Pentium chạy nhanh gấp 2 lần về phép tính số nguyên và từ 3 đến 7 lần về phép tính số học có dấu phảy động.

Pentium có sơ đồ khối như hình 1.8.

Sau đây là đặc điểm chính của Pentium:

- Tương thích với i386 và i486 về cấu trúc các thanh ghi, các chế độ và tập lệnh.

- Có hai bộ nhớ ngầm (dữ liệu, lệnh) 8K để tăng tốc độ hoạt động.

- Có cấu trúc Super scale (siêu vô hướng): có hai đường ống dẫn (U và V) cho số nguyên (với hai khối ALU độc lập) và một đường ống dẫn dấu phảy động. Cấu trúc trên cho phép tốc độ xử lý số nguyên nhanh gấp 1,9 lần 486 vì xử lý song song hai số liệu.

- Là một vi xử lý 32 bit nhưng đường dây số liệu ngoài 64 bit với chế độ gọi số liệu theo đường ống và viết trở về (Write back).

- Có một hàng đoán trước sự rẽ nhánh (branch prediction) BTB (branch target buffer) của lệnh đối với 256 lệnh rẽ nhánh mới đây nhất.

- Sử dụng giao thức MESI (Modifiel Exclusive Shared Invalid) để hoạt động nhịp nhàng giữa các nhớ ngầm (cache coherence) và các vi xử lý khác trong chế độ đa xử lý.

- Đơn vị dấu phảy động (đồng xử lý toán) cũng sử dụng cấu trúc liên hợp, cho phép thực hiện hai lệnh 80x87 trong một chu trình đồng hồ đơn nhanh gấp 5÷10 lần CPU 80486 cho các chương trình CAD và đồ hoạ.

- Khối điều khiển có đơn vị điều khiển truy cập trực tiếp khối nhớ.

- Ngoài các lệnh của 486 còn có thêm các lệnh sau:

+ *CMPXCHG8B* : so sánh và hoán vị 8 bytes.

+ *CPUID* :đồng nhất CPU (trong thanh ghi EAX).

+ *WRMSR*: (Write model specific register): lệnh bảo vệ một thanh ghi riêng cho mẫu (ví dụ thanh ghi thử v.v...) với giá trị trong EDX:EAX.

## 1.2.6. CÁC VI XỬ LÝ INTEL BẬC CAO

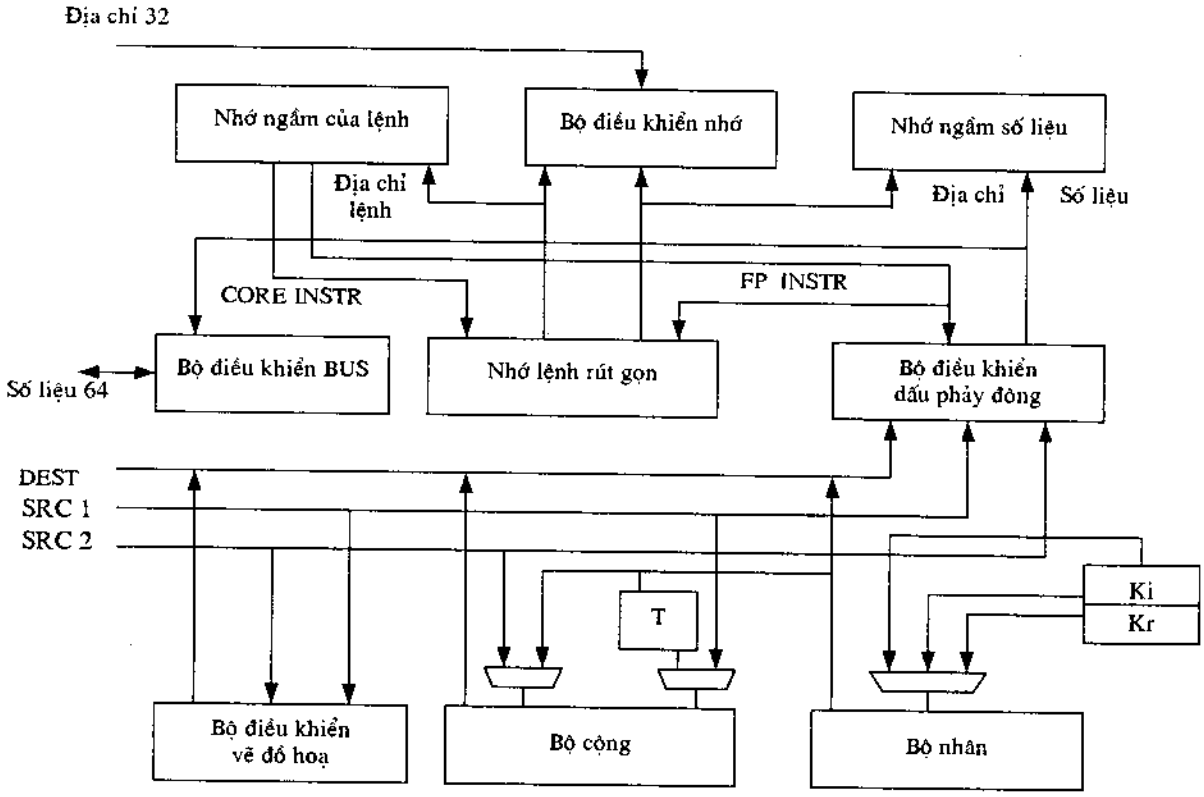
– 80686: (1996) chứa khoảng 5 triệu tranzito có bộ nhớ ngầm và hệ lệnh lớn hơn 586. Sơ đồ khối của 80686 như hình 1.9.

– 80786: (khoảng giữa những năm 1990) chứa 2 MB nhớ ngầm, 6 xử lý số nguyên và dấu phảy động phân biệt nhau (xử lý cộng, trừ, nhân, chia...) và một bộ tương tác video số hoặc giao diện với người dùng. Ngoài ra 786 còn giữ sự tương thích với bộ lệnh 386/486 và hoạt động với tần số nhịp 250MHz.

– 80860: chứa 64 bit với cấu trúc Harvard (đơn chip loại RISC), có 1,2 triệu tranzito có đồng VXL dấu phảy động và một đồng VXL đồ hoạ 3D. Các vi xử lý trong 860 có thể hoạt động độc lập tương đối và song song. Với tần số nhịp400MHz nó có thể thực hiện tốc độ 80 triệu hành động dấu phảy động (MF lips) hay 85.000 drystone (tốc độ drystone là hiệu suất tương đối của một máy tính thực hiện một chương trình "benchmark" chuẩn không dùng sự chia mảng khối nhớ, mà cho phép 386 loại trang nhớ ảo).

## 1.3. BỘ NHỚ TRUNG TÂM

Bộ nhớ trung tâm khác với bộ nhớ ngoài là đặt bên trong máy vi tính. Tùy theo vật liệu và công dụng, ta có các loại nhớ khác nhau, nhưng có cùng một đặc điểm chung về nguyên tắc hoạt động, cách ghi/đọc và thông số.




**HÌNH 1.9.** Sơ đồ khối của 80686.

### 1.3.1. ĐẶC ĐIỂM CHUNG

Phần tử nhớ gồm các linh kiện điện tử (tranzito, điện trở, tụ) có hai trạng thái cân bằng (0-xoá, 1-nhớ) khi có các tín hiệu sau:

- Chọn vi mạch (CS-chip select) hay CE (chip enable).
- Chọn phần tử nhớ bằng tín hiệu giải mã địa chỉ $A_i$.
- Lệnh ghi/đọc (R/W).
- Số liệu ($D_0 \div D_n$).

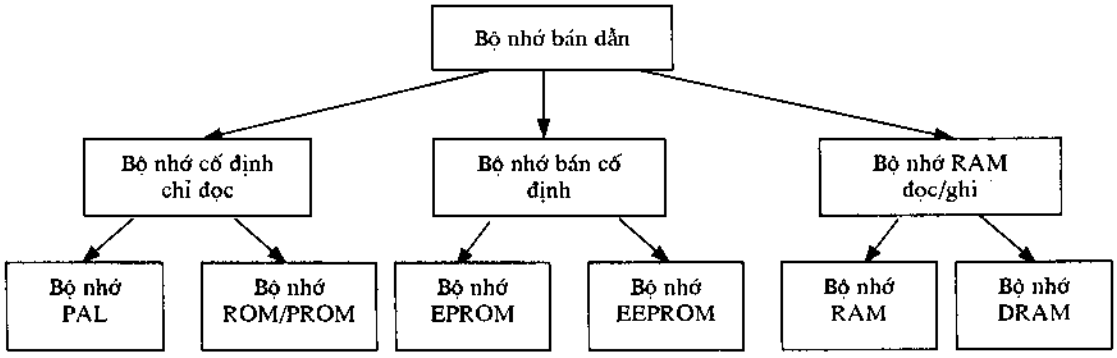
Bộ nhớ gồm nhiều phần tử nhớ, được tổ chức thành thanh ghi, ma trận, vi mạch, modun, băng (dải) và khối nhớ để có độ dài lời và số địa chỉ lời cần thiết.

Các thông số của khối nhớ là:

- Dung lượng nhớ tức số phần tử hay số bit như: 8, 16, 1K(1024), M(1024K), G(1024M), T(1024G).

- Đơn vị số liệu nhớ cho một địa chỉ có thể là nibble (4 bit), octed hay byte (8 bit), word-lời (16 bit), lời kép (double word-32 bit) và quadrupt word-4 lời (64 bit).

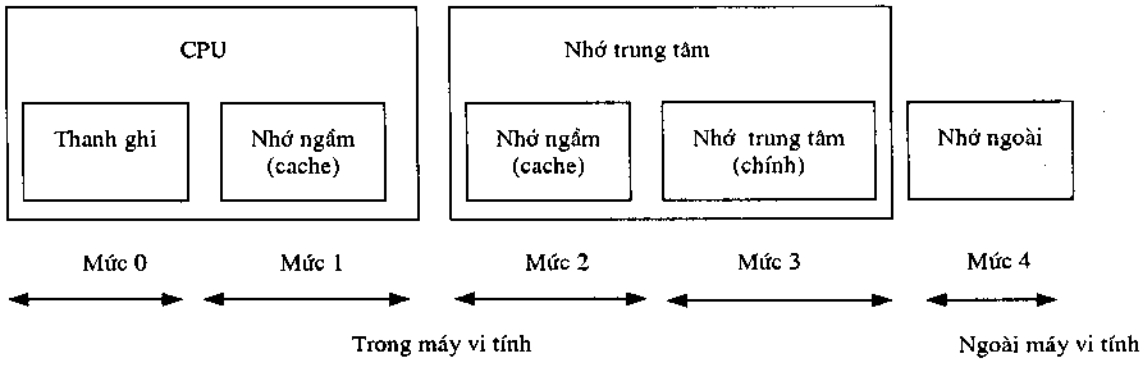
- Thời gian (chu trình) đọc, viết: thời gian cần thiết kể từ lúc vi xử lí đưa ra địa chỉ ô nhớ tới khi số liệu được ghi hay đọc.




**Hình 1.10.** Phân loại bộ nhớ theo cách đọc ghi.

Tùy theo quan điểm, có cách phân loại linh kiện nhớ khác nhau:

- Theo cách đọc ghi (hình 1.10).
- Theo cấp sử dụng trong MVT (hình 1.11).




**Hình 1.11.** Phân loại bộ nhớ theo cấp sử dụng.

Các linh kiện nhớ được ghép nối với vi xử lý qua các đường dây của MVT và các khối giải mã địa chỉ, các khối điều khiển.

## 1.3.2. NHỚ RAM

Nhớ RAM là vi mạch nhớ có thể đọc và viết được (RAM - random access memory - truy cập ngẫu nhiên). Có hai loại RAM là SRAM (Static RAM) và DRAM (Dynamic RAM).

### 1. Nhớ SRAM

***a) Phần tử nhớ***

Phần tử nhớ RAM bán dẫn như hình 1.12.

***b) Linh kiện nhớ***

Các phần tử nhớ được tổ chức thành ma trận (hình 1.14). Các bộ giải mã hàng-cột cho địa chỉ dạng thập phân để chọn phần tử nhớ. Mạch vào/ra được đưa vào để khuếch đại và đệm số liệu.

*c) Vi mạch nhớ SRAM*

Các vi mạch SRAM thường gặp là:

+ TMS 4016 - loại MOS 2048 x 8 bit.

+ TMS 4014 - loại MOS 2048 x 8 bit với thời gian xâm nhập 250 ns.

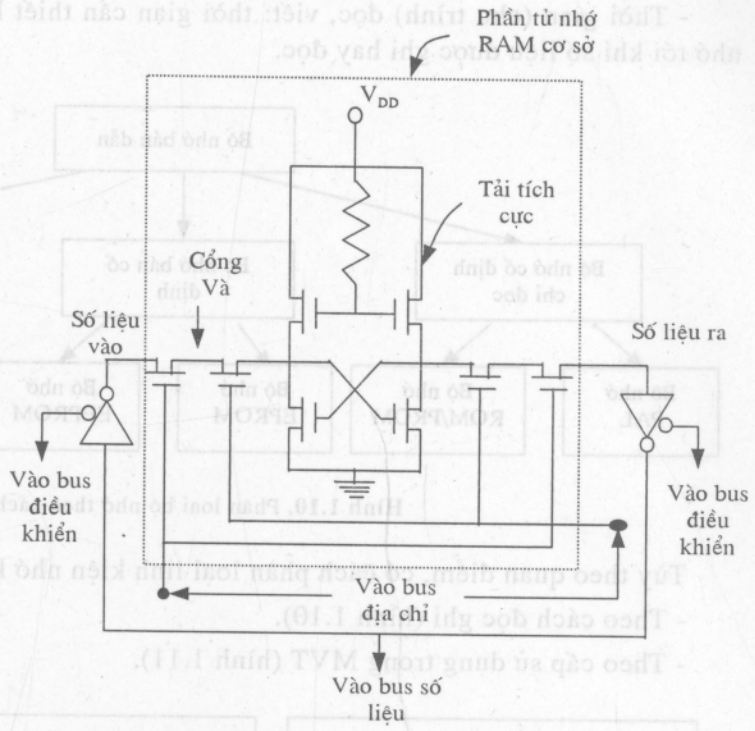
+ TMS 4044 - loại MOS 4F x 1 bit.

Vi mạch nhớ SRAM có các đặc điểm sau:

+ Không cần khối điều khiển như DRAM.

+ Lưu trữ tin ổn định, không cần phải làm tươi như DRAM.

+ Tổ chức cổng kênh (nhiều tranzito cho một phần tử nhớ), mất tin khi mất nguồn nuôi.



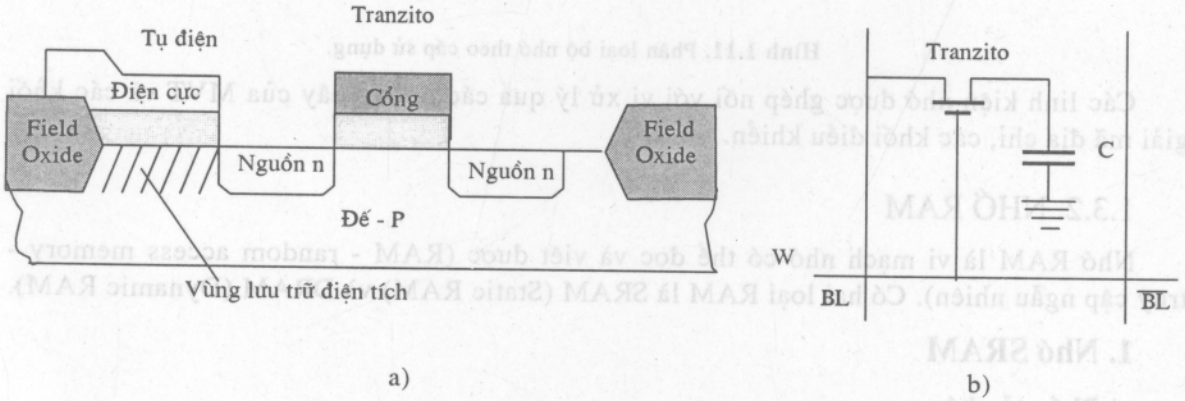

**Hình 1.12.** Cấu trúc phần tử nhớ RAM bán dẫn.

## 2. Nhớ DRAM

Nhớ DRAM ra đời nhằm khắc phục nhược điểm của SRAM, nhằm tăng cường mật độ phần tử nhớ.

*a) Phần tử DRAM*

Phần tử nhớ DRAM như hình 1.13.




**Hình 1.13.** Cấu tạo (a) và sơ đồ (b) của phần tử nhớ DRAM.

Nguyên tắc nhớ của DRAM là tích điện cho tụ-ghi 1 và phóng điện cho tụ-ghi 0

*b) Vi mạch nhớ DRAM*

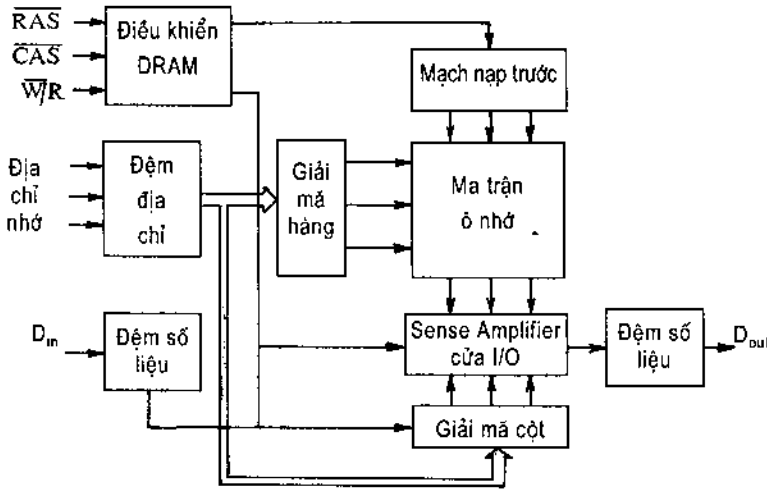
Vi mạch nhớ DRAM cũng tổ chức dưới dạng ma trận như hình 1.14 và hình 1.15.

Đặc điểm của một vi mạch nhớ DRAM là:

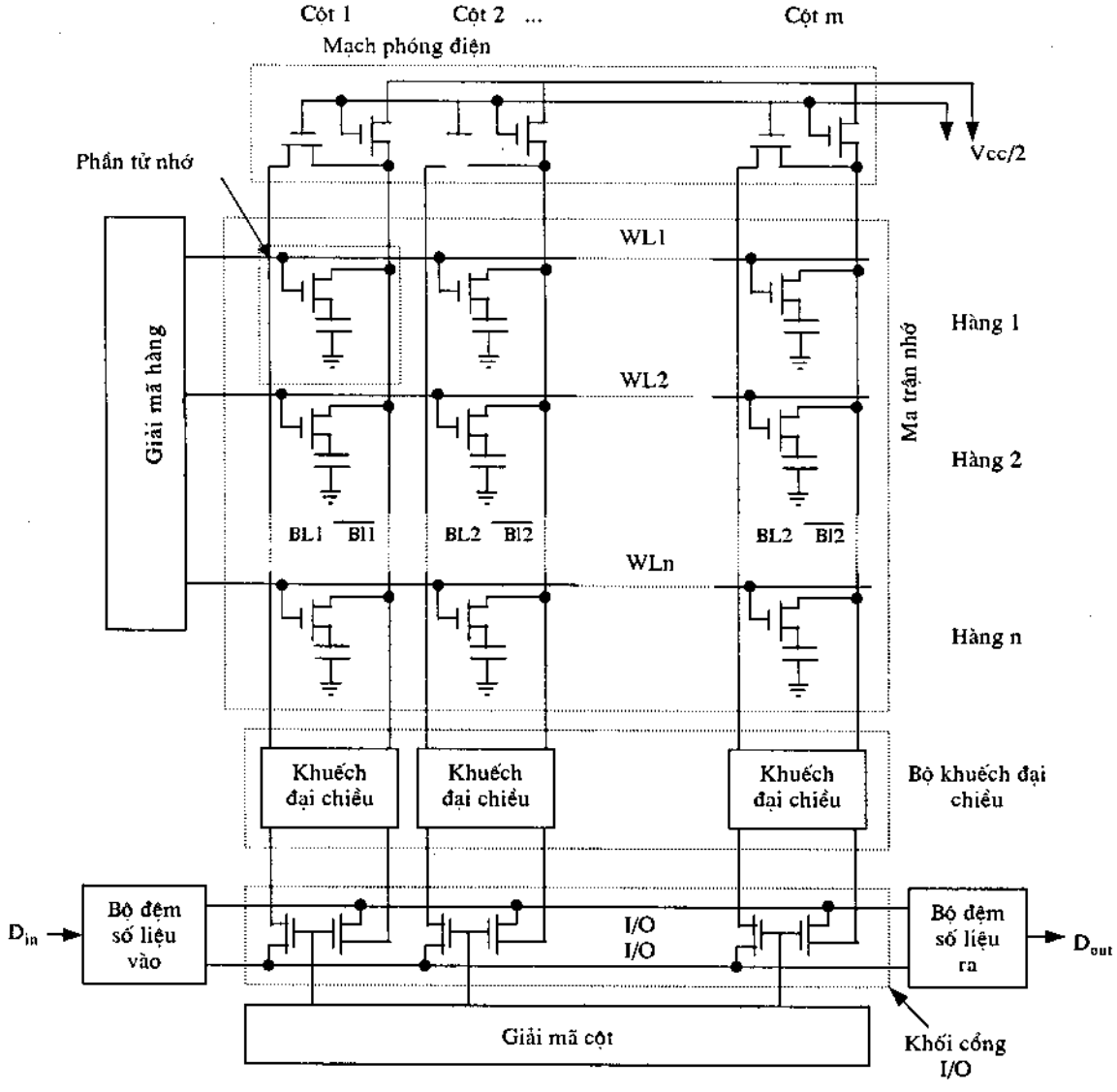
- Mật độ cao.

- Có sự dồn kênh địa chỉ (A0÷Ai) chung cho cả hàng và cột nên có các tín hiệu điều khiển hàng *RAS* và điều khiển cột *CAS*. Do đó, vi xử lý điều khiển DRAM phải qua một khối điều khiển.

- Vì phần tử nhớ là tụ C nên điện tích đã ghi (trạng thái 1) bị rò qua điện trở vào của tranzito điều khiển phóng




**Hình 1.14.** Sơ đồ khối của một vi mạch nhớ DRAM.



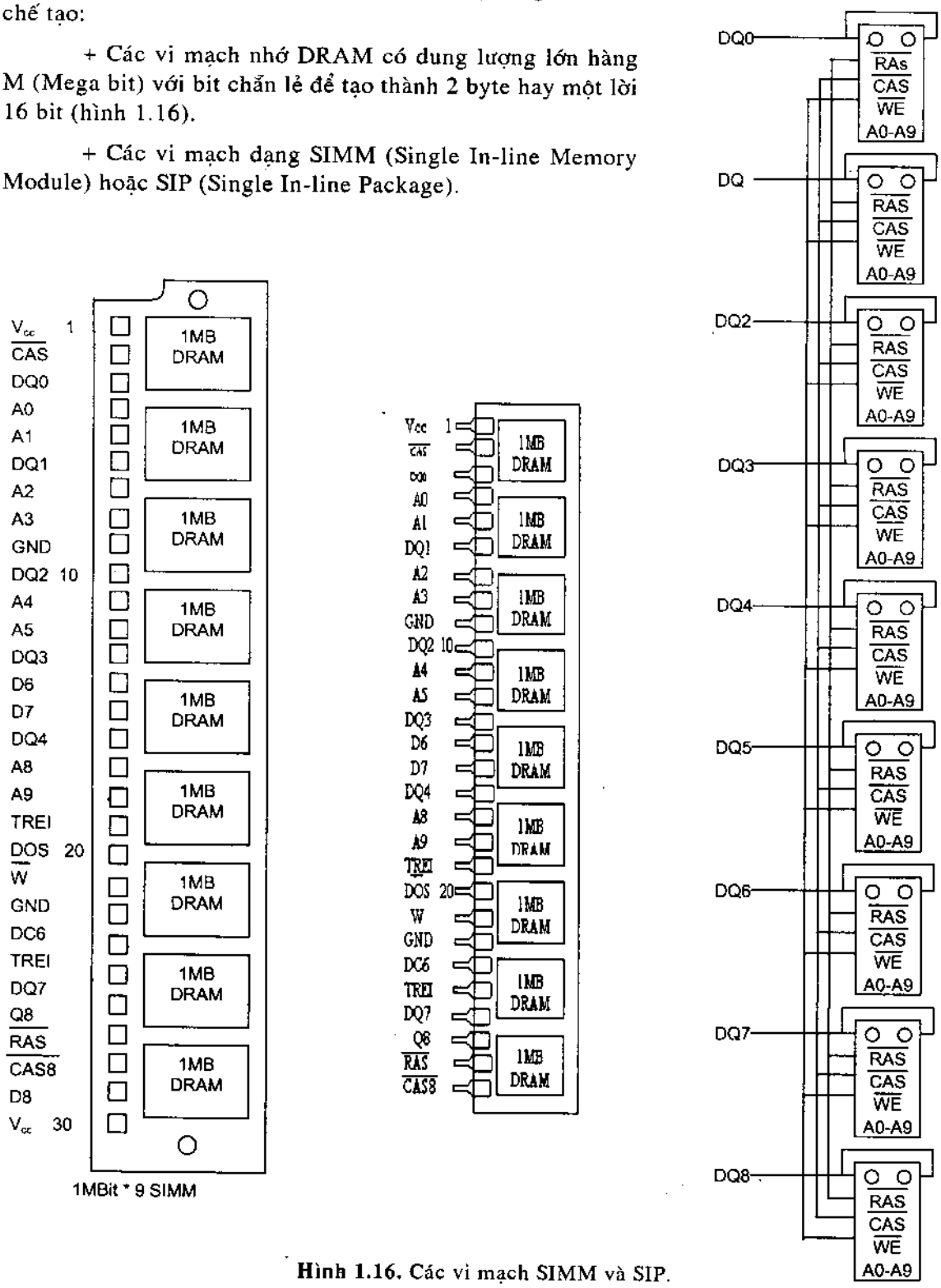

**Hình 1.15.** Ma trận nhớ DRAM.

nạp điện tích cho tụ. Do đó, cần phải làm tươi (refresh), tức nạp điện lại cho tụ tới mức 1 khi đã ghi 1 và không nạp cho tụ khi đã ghi 0 bởi một xung của máy phát xung qua bộ khuếch đại dòng.

Ngày nay, nhờ tiến bộ của kỹ thuật vi mạch người ta đã chế tạo:

+ Các vi mạch nhớ DRAM có dung lượng lớn hàng M (Mega bit) với bit chẵn lẻ để tạo thành 2 byte hay một lời 16 bit (hình 1.16).

+ Các vi mạch dạng SIMM (Single In-line Memory Module) hoặc SIP (Single In-line Package).




**Hình 1.16.** Các vi mạch SIMM và SIP.

*c) Vi mạch nhớ DRAM*

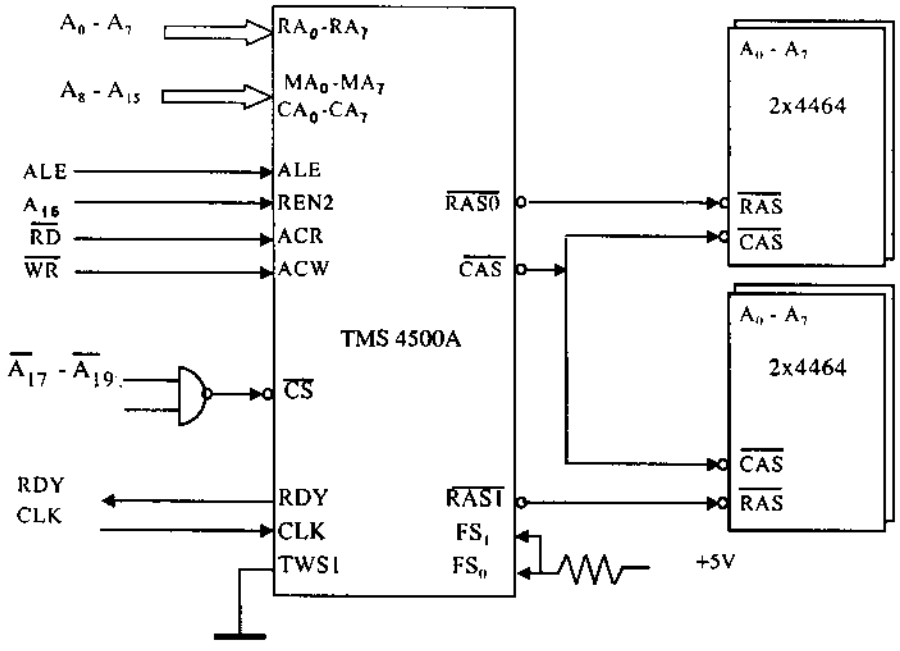
Cũng như SRAM, DRAM cũng tổ chức dạng ma trận nhưng có sự dồn kênh địa chỉ (đường dây địa chỉ chung cho cả hàng và cột) và các tín hiệu chỉ hàng (RAS) và tín hiệu chỉ cột (CAS). Do đó phải có vi mạch điều khiển để tạo ra các tín hiệu trên.

Các vi mạch DRAM thường gặp là:

+ 2104A (4096 x 1bit)
+ TMS 4464 (64K x 4bit)
+ TMS 4116 (16K x 1bit)
+ TMS 44C256 (256K x 4bit)
+ TM 258FL8 (dạng 30 chân một hàng của 44C256)
+ 2164B (64K x 1bit)
+ 21256 (256K x 1bit)
+ 421000 (1M x 1bit)
+ 424256 (256K x 4bit).

*d) Vi mạch điều khiển DRAM*

Vi mạch này được mắc trực tiếp với đường dây của vi xử lý, trước vi mạch DRAM làm hai nhiệm vụ:




**Hình 1.17.** Sơ đồ khối điều khiển TMS 4500A.

- Dồn kênh địa chỉ: cho ra một đường dây địa chỉ chung cho cả hàng và cột và hai tín hiệu điều khiển hàng ( $\overline{RAS}$ ) và cột ( $\overline{CAS}$ ).

- Làm tươi DRAM.

Các vi mạch điều khiển thường gặp như:

+ 3222: cho vi mạch nhớ 2K, chỉ làm tươi, không dồn kênh.

+ 8203: cho vi mạch 2164 (64K).

+ TMS4500A: cho hai vi mạch 4464 (64K) (hình 1.17).

+ 82C08: cho các vi mạch nhớ TMS 44C256 tới 1M x 8 bit.

+ 74ALS6301 cùng với các vi mạch:

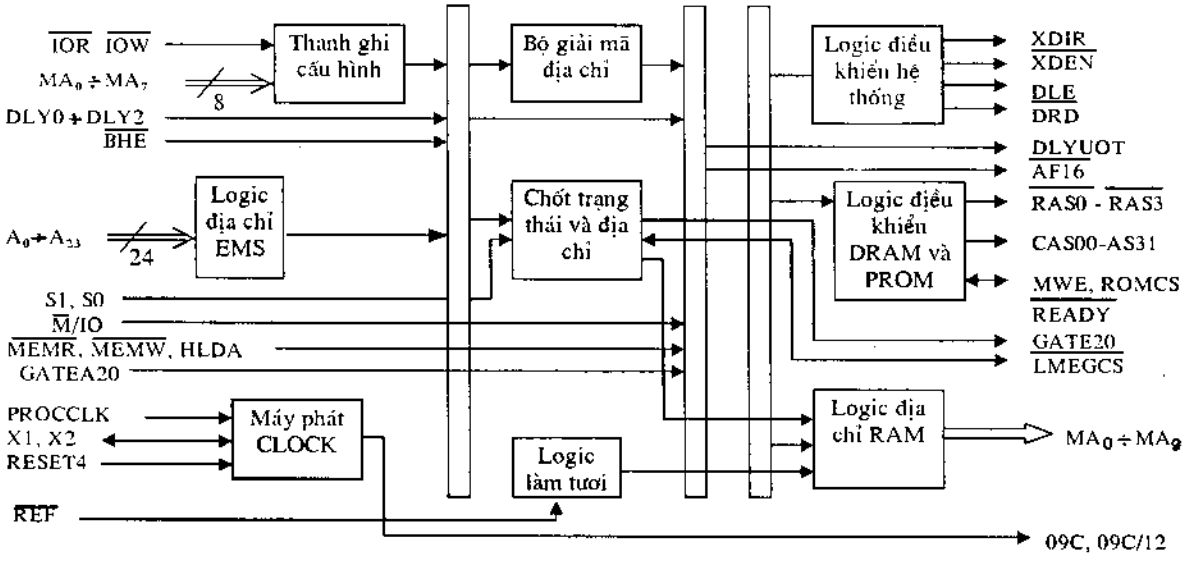
.PAL 16R8: máy định thời khoảng làm tươi.

.82S105B: điều khiển nhịp làm tươi.

.74AS632: bit kiểm tra chẵn lẻ.

.74BCT2828: khuếch đại đường dây địa chỉ.

+ 82C212 chip gồm 84 chân, dùng cho 4 băng (1M) DRAM với độ dài lời 16 bit cộng với 2 bit chẵn lẻ và dung lượng 16 M (hình 1.18).




**Hình 1.18.** Sơ đồ khối điều khiển nhớ 82C212A.

## 1.3.3. NHỚ CHƯƠNG TRÌNH

Các vi mạch nhớ RAM có ưu điểm là có thể đọc và ghi số liệu nhưng có nhược điểm cơ bản là tin đã ghi bị mất khi bị mất nguồn nuôi. Khắc phục nhược điểm trên, người ta chế tạo các bộ nhớ chỉ đọc ROM (Read only Memory). Các bộ nhớ này được nhà sản xuất ghi dữ liệu (chương trình, bảng, hằng số) không thể ghi lại (ROM) hoặc ghi lại được (nhớ bán cố định) như: EPROM, PAL, PLD v.v...

### 1. Nhớ ROM

Linh kiện nhớ ROM đơn giản là ma trận diode. Các hàng của ma trận được nối với anot của các diode, còn cột của ma trận nối với điện thế nuôi qua các điện trở tải và lối ra của tín hiệu. Khi có tín hiệu địa chỉ hàng, lối ra sẽ có tín hiệu:

– Là 0 khi có diode nối giữa hàng và cột.

– Là 1 khi không có diode nối giữa hàng và cột.

Nhớ ROM được ghi cố định bằng cách nối các diode giữa các hàng và cột do các nhà sản xuất chế tạo. Các chip ROM hiện nay có thời gian xâm nhập 120ns÷150ns.

## 2. Nhớ PROM (Programmable ROM)

Là loại ROM nhưng có thể lập trình được do người sử dụng tiến hành ghi tin. Cấu trúc nhớ ROM cũng theo ma trận như ma trận diode, phần tử nối giữa hàng và cột có thể là diode hoặc tranzito nhưng nối tiếp với một cầu chì dễ đứt.

Khi ghi tin, phải dùng một điện thế nuôi cao (thường 12V hoặc hơn) để:

- Làm đứt cầu chì khi ghi tin 1 (hay có tín hiệu).

- Không làm đứt cầu chì khi ghi tin 0 (hay không có tín hiệu).

So với nhớ ROM thì nhớ PROM người sử dụng dễ dàng ghi tin nhưng chỉ ghi được một lần.

## 3. Nhớ EPROM - EEPROM (erasable PROM - Electric Erasable PROM)

Là các vi mạch nhớ có thể xóa được (erable) tin để ghi lại. Mỗi phần tử nhớ là một tranzito. Ngoài vùng cửa điều khiển (control gate) còn có vùng cửa thả nổi (floating gate) không nối với cực nào ở giữa cực cửa điều khiển và đế. Cực cửa thả nổi này có tác dụng:

- Cho dòng điện giữa cực nguồn - máng đi qua khi điện tích tích tụ là dương khi muốn ghi 0 bởi cho điện thế cao (+12V) ở cực cửa điều khiển G.

- Không cho dòng đi qua khi không có điện tích tụ dương (hoặc tích tụ điện tích âm) khi muốn ghi 1.

Khác với PROM là có thể xóa được tin ghi 1, tức xóa điện tích tích tụ trên cực cửa thả nổi này bằng cách:

+ Chiếu tia cực tím vào vùng cực cửa (qua cửa sổ bằng thạch anh trên vi mạch nhớ) trong thời gian cỡ nhiều giờ (thường ≥ 8 giờ).

+ Dùng điện thế ngược tác dụng vào cực cửa - đế, tức xóa bằng điện (EEPROM - Electric Erasable PROM).

Một biến thể của EEPROM là nhớ chớp nhoáng (flash memory) tạo ra từ năm 1983, gồm các ô nhớ Famost, dung lượng lớn (cỡ 8MO), được ghi bởi các xung điện ngắn 10μs, biên độ 12V (1M bit mất cỡ 2s) và xóa từng bit một (khác EPROM xóa cả khối, chừng 1s cho 1M bit. Khối nhớ flash này dùng để thay thế đĩa mềm hoặc đĩa cứng dung lượng nhỏ.

Chip nhớ flash điển hình như Intel - 28F010 có dung lượng 128K x 8 bit, thời gian thâm nhập 120ns (nhanh hơn DRAM kiểu cũ) và độ bền tới 1,6 triệu giờ (đĩa cứng cỡ 500000 giờ). Tới năm 2000, dự kiến chế tạo nhớ flash có dung lượng 512MO.

Vi mạch nhớ EPROM thông dụng như:

| | | |
|---|---|---|
| . 2708 (1Kx 8) ; | . 27512 (64Kx 8) ; | . 2764 (8Kx 8). |
| . 2716 (2Kx 8) ; | . 27128 (16Kx 8) ; | |
| . 2732 (4Kx 8) ; | . 27256 (32Kx 8) ; | |

## 4. Nhớ PLD (Programable Logic Device)

PLD là vi mạch bán dẫn có thể lập trình được. Ngoài việc tạo các mạch tạo hàm logic AND, OR và tổ hợp của chúng, người ta còn sử dụng PLD để tạo nhớ PROM (EPROM, EEPROM).

PLD có các loại PROM, PAL (Programable array logic) và PLA (Programable logic array).

Cấu trúc PLD cũng tương tự ma trận nhớ với:

- Các lối vào theo hàng là địa chỉ của từng bit, lối ra có cổng logic AND.

- Các lối vào theo cột là địa chỉ vào theo hàm số, có bộ khuếch đại với hai đường dây ra không đảo và đảo cực tính tín hiệu. Giữa đường dây cột và hàng có thể có cầu nối (giống ma trận diode của ROM) để tạo thành các hàm AND.

- Lối ra lại có các đường dây cột với vi mạch OR để tạo ma trận OR.

Ngoài các vi mạch đơn giản trên còn các vi mạch có nhớ (lối ra có flip flop) và các đường dây phản hồi từ lối ra trở về lối vào.

## 1.3.4. CÁC LOẠI NHỚ KHÁC

Ngoài các loại nhớ RAM, ROM trên còn có các loại nhớ CCD, nhớ liên hợp.

### 1. Thanh ghi

Là tổ hợp các flip flop bán dẫn tác động nhanh. Chính các thanh ghi của vi xử lý hay bộ nhớ ngầm (nhớ đệm) nằm trong hay ngoài vi xử lý là các thanh ghi trên.

### 2. Nhớ CCD (Charge Coupled Device)

Ngay từ năm 1970 người ta đã phát minh ra loại nhớ CCD này. Nguyên tắc như sau:

- Tin 1 của ánh sáng chiếu lên chất bán dẫn, tạo nên một lỗ trống (giếng tích điện dương) vì mất điện tử, còn tin 0 của ánh sáng (không có chùm tia) không tạo nên lỗ trống.

- Đọc tin 0 bằng cách cho một chuỗi xung kích thích nối tiếp như đọc tin của thanh ghi dịch.

Nhớ CCD thường dùng cho camera hoặc scanner cần nhiều bit nhớ bố trí theo một mặt phẳng (ma trận mặt phẳng).

### 3. Nhớ liên hợp (Associative Memory - CAM)

Nhớ này còn có tên là nhớ mà cách gọi địa chỉ theo nội dung của nó. Nó tìm tin (là 0 hay 1) hay bộ mô tả là có mặt hay không ở trong ô nhớ nếu cung cấp cho nó một tin liên quan làm địa chỉ.

Nó gồm hai phần:

- Bộ so sánh để so sánh địa chỉ tới (nội dung tin) với địa chỉ của các ô nhớ để tìm tin 0 hay tin 1.

- Bộ phát tin 0 hay tin 1 tùy theo việc đã ghi tin (1) hay chưa lên tin (0).

Loại nhớ này thường dùng để phát số liệu (mã) cho một bảng tra cứu chuyển đổi mã (ví dụ: thập phân - 2,8,16 hay ASCII) hay phát chương trình (chuỗi mã lệnh).

## 1.3.5. SỬ DỤNG VI MẠCH NHỚ TRONG MÁY VI TÍNH PC- IBM

Sau vi xử lý, bộ nhớ đóng vai trò quan trọng vì:

- Nhớ chương trình điều khiển máy (hệ điều hành) và bảng số liệu trong nhớ ROM.

- Nhớ chương trình của người sử dụng (nhớ nháp) trong RAM.

- Nhớ đệm (cache) giữa vi xử lý và bộ nhớ trung tâm hay nhớ ngoài (đĩa, băng).

Ta xét chi tiết các sử dụng trên:

### 1. Chế độ của vi xử lý sử dụng bộ nhớ

**- Chế độ địa chỉ thực**: chế độ hệ điều hành gọi địa chỉ bộ nhớ bằng địa chỉ thực hay địa chỉ vật lý:

+ Địa chỉ thực của 8086 là 640KB.

+ Địa chỉ thực của 80286, 80386 là 1MB

- **Chế độ bảo vệ**: vùng nhớ chứa 4 vùng cho 4 nhiệm vụ (hay chương trình của các người sử dụng khác nhau - nhiều nhiệm vụ nhiều người dùng) và 4 lớp có mức ưu tiên khác nhau kể từ trong (lõi) ra ngoài.

- **Chế độ nhớ ảo**: (hay địa chỉ ảo) có từ 80286 trở đi là chế độ gọi địa chỉ của bộ nhớ ngoài (đĩa từ) với địa chỉ lớn hơn 1MB. Có sự chuyển đổi khối, gán một số khối nhớ tới cùng một không gian địa chỉ vật lý (địa chỉ có thực). Bộ nhớ thực được mở rộng LIM tới 8MB, 16MB, 32MB với sự điều khiển bởi lệnh của DOS COFIG. SYS (khởi tạo) và ngắt INT67 với các hàm LIM/EMM khác nhau.

## 2. Bản đồ nhớ ở chế độ thực

Tùy theo loại vi xử lý 8086, 80386, 80586 hay số đường dây địa chỉ, có dung lượng bộ nhớ khác nhau nhưng đều có chung bản đồ nhớ cơ bản (hình 1.19).

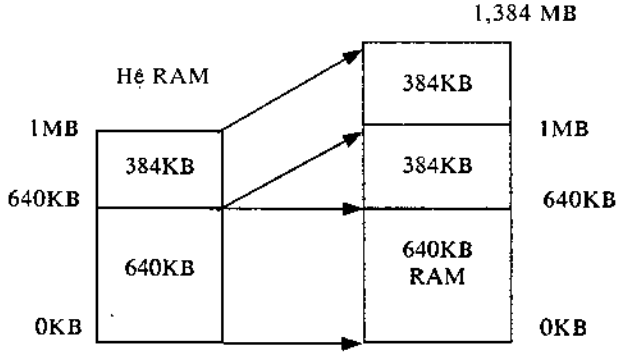
## 3. Nhớ cache (ngầm)

Là bộ nhớ đệm, không có địa chỉ vật lý (nên gọi là nhớ cache). Có thể:

- Nằm trong vi xử lý (từ 80486) và còn chia riêng cho lệnh, số liệu (586) với dung lượng 8KB.

- Nằm ngoài vi xử lý (80386) với khối điều khiển 82385.

Bộ nhớ cache này làm nhiệm vụ đệm giữa CPU nhanh và nhớ chính chậm để phối hợp vận tốc hoạt động.




**Hình 1.19.** Sơ đồ phân bố bộ nhớ thực.

## 4. Quản lý bộ nhớ

Về dung lượng, bộ nhớ được chia theo byte - 8 bit (hay octet - B hay O), lời 16 bit (W), lời kép 32 bit (DW), 4 lời (QW) 64 bit.

Bộ nhớ dung lượng lớn từ 8086 được quản lý bởi khối quản lý ở trong vi xử lý với các thanh ghi điều khiển (CR) và đoạn (CS, DS...) (cho nhớ thực), bảng các bộ mô tả và thanh ghi đoạn (cho nhớ ảo).

DOS quản lý bộ nhớ tới 640KB với hai vùng (hệ điều hành, diện tích của chương trình chuyển đổi - TPA - Transient Program Area). DOS cung cấp cho mỗi chương trình một vùng nhớ, được quản lý nhờ một khối dữ liệu đứng ngay trước các vùng nhớ gọi là MCB (Memory Control Block) có kích thước 16 byte (1 paragraph).

Trong môi trường đa nhiệm (đa sử dụng), vùng nhớ chia làm 4 vùng với 4 lớp có đặc quyền giảm dần từ trong ra ngoài L = 0÷3 (chương trình quản lý các tài nguyên hay BIOS, quản lý hệ điều hành, quản lý tệp, thư viện và chương trình ứng dụng). Việc chuyển nhiệm vụ từ mức cao tới mức thấp có mức đặc quyền cao hơn phải qua các cửa (gate) bảo vệ bộ nhớ thực hiện:

- Cách ly chương trình hệ thống và chương trình ứng dụng.

- Cách ly giữa các nhiệm vụ.

- Kiểm chứng thời điểm thâm nhập vào đối tượng.

## 1.4. CÁC CỔNG VÀO/RA

Các cổng vào/ra của MVT làm nhiệm vụ trao đổi tin giữa vi xử lý và các thiết bị ngoài. Đó là:

- Các cổng song song cho máy in (LPT).
- Các cổng nối tiếp cho chuột, đường dây truyền thông.
- Bìa màn hình.
- Bìa điều khiển đĩa.
- Bìa điều khiển âm thanh.

### 1.4.1. CÁC CỔNG VÀO/RA SONG SONG

Cổng này trao đổi tin số (8 bit, 16 bit) cho các thiết bị ngoài số, tin song song.

#### 1. Cổng ra 8 bit

Cổng ra 8 bit cho máy in LPT có sơ đồ khối như hình 1.20 với các thanh ghi điều khiển, trạng thái, số liệu (8 bit), các khối xử lý ngắt (IRQ logic) và giải mã địa chỉ/lệnh (address decoder control).

#### 2. Cổng vào/ra (8 bit) 8255A/82C55A

Vi mạch 8255A được chế tạo cho bộ vi xử lý Intel 8 bit (8080, 8085) với 3 cổng vào/ra PA, PB, PC lập trình được (chế độ, chiều cổng, đưa xung ra điều khiển) (hình 1.21). Các cửa PA, PB để vào/ra số liệu, PC để vào/ra số liệu (chế độ 0), để làm tín hiệu hội thoại cho các cửa PA, PB (chế độ 1,2) và điều khiển với từ điều khiển xác lập/xóa từng bit cửa C.

*a) Từ điều khiển*

Có hai từ điều khiển: cấu hình và lập xóa bit cửa C

*- Từ điều khiển cấu hình*: với các bit sau:

$d_7 = 1$: chỉ điều khiển cấu hình.

$d_6$, $d_5$ : chọn chế độ cho cửa PA:

00 - chế độ 0 (không hội thoại, một chiều).

01 - chế độ 1 (có hội thoại, một chiều).

1x - chế độ 2 (có hội thoại, hai chiều cho cửa PA).

$d_4$ : xác định chiều truyền số liệu của cửa PA: 0 - ra, 1 - vào.

$d_3$ : xác định chiều truyền số liệu của nửa cửa PC cao (PC7, PC6, PC5, PC4) 0 - ra, 1 - vào.

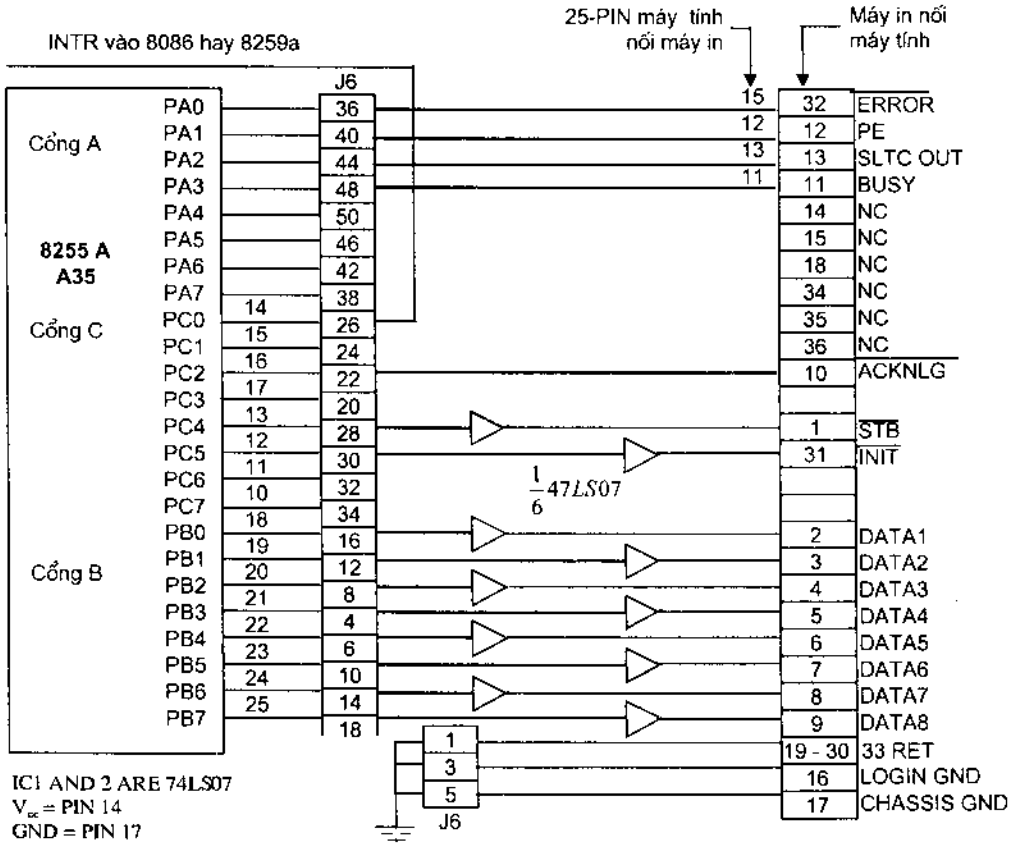
$d_2$ : chọn chế độ cho cửa PB: 0 - chế độ 0, 1 - chế độ 1.

$d_1$ : chọn chiều truyền số liệu cửa PB: 0 - ra, 1- vào.

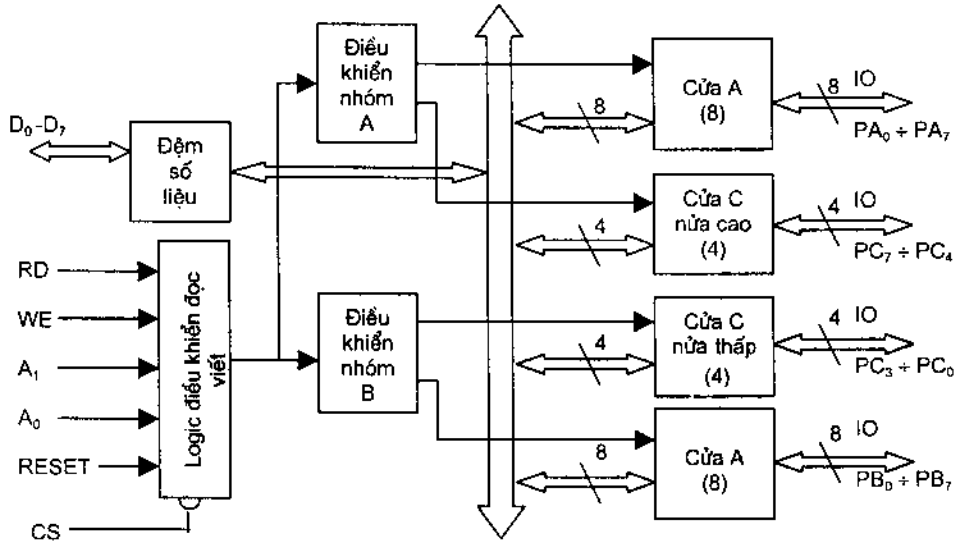
$d_0$ : chọn chiều truyền số liệu cho nửa cửa PC thấp (PC0, PC1, PC2, PC3).

*- Từ điều khiển xác lập xóa các bit của cửa PC:*

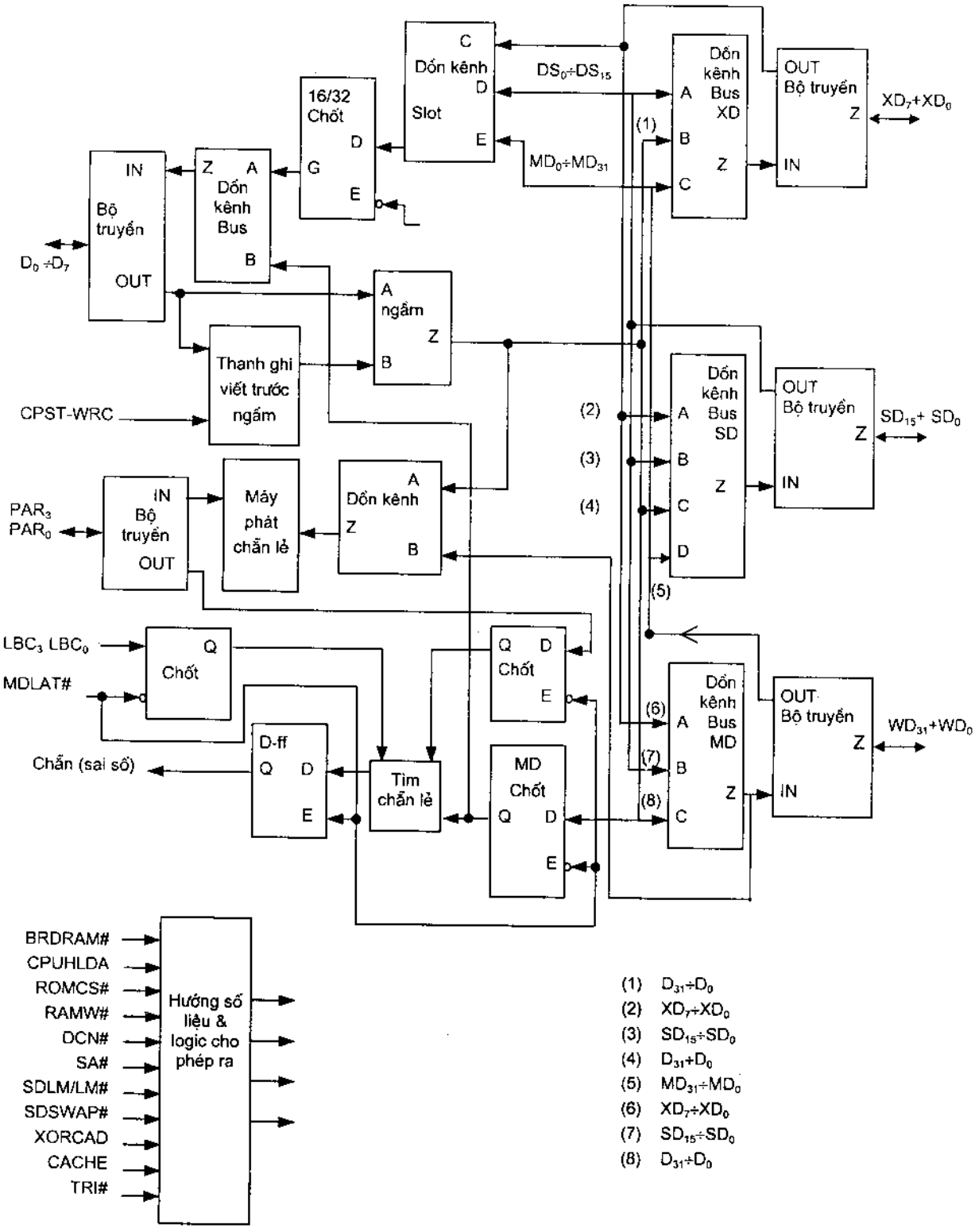
Các bit PCi của cửa PC này có thể xác lập ($D_0$ của khối điều khiển = 1) hay xóa ($D_0$= 0). Ba bit ($D_1 \div D_3$) xác định mã của PCi. Việc lập/xóa này dùng để điều khiển hay hội thoại với thiết bị ngoài.




**Hình 1.20.** Sơ đồ khối cổng ra máy in (8bit).




**Hình 1.21.** Sơ đồ khối của 8255A/82C55A.




**Hình 1.22.** Sơ đồ khối của 82345.

*b) Từ trạng thái*

Từ trạng thái cho biết trạng thái của 8055A ở các chế độ hội thoại M = 1, 2 vi xử lý đọc các từ này để biết trạng thái của 8055A. Có hai từ trạng thái khác nhau phân biệt cho chiều truyền số liệu vào hoặc ra để biết:

- Có yêu cầu ngắt chương trình (INTRA, INTRB).
- Được vi xử lý cho phép ngắt (INTEA, INTEB).
- Các thanh ghi đệm ra đầy (có số liệu IBFA, IBFB) hay đệm vào đầy (OBFA, OBFB).

*c) Từ số liệu*

8 bit số liệu ($D_0 \div D_7$) có thể đưa ra hoặc đưa vào, tùy thuộc chiều của cổng PA, PB, PC.

## 3. Bộ đệm số liệu 82345

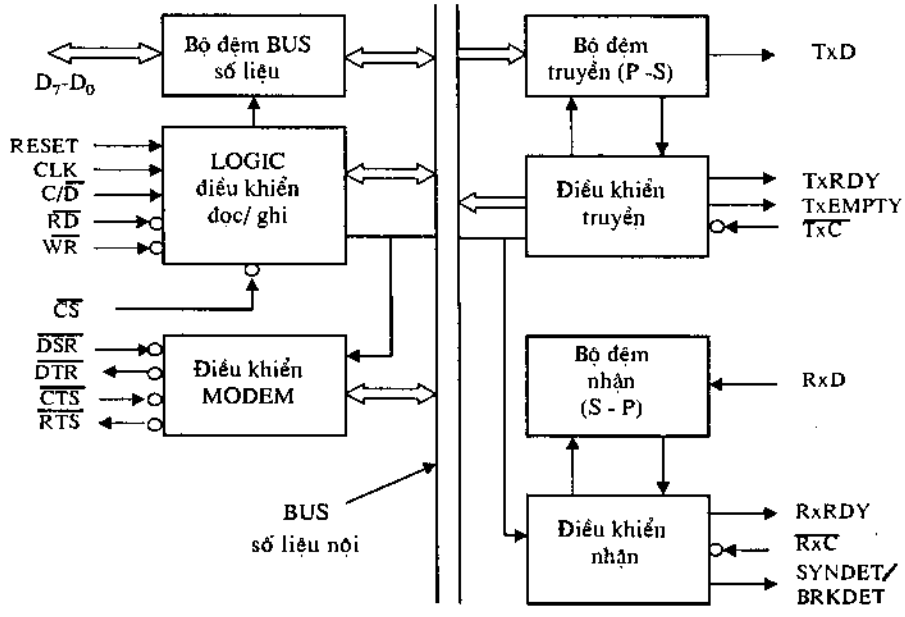
Là vi mạch mật độ rất cao (VLSI), dùng để truyền số liệu theo 3 kênh (8 bit, 16 bit, 32 bit) cho MVT PC/AT. Khối còn có bộ nhớ đệm, kiểm tra chẵn/lẻ và dồn kênh (hình 1.22).

## 1.4.2. CÁC CỔNG VÀO/RA NỐI TIẾP

Các cổng này biến đổi tin song song thành nối tiếp giúp VXL trao đổi tin nối tiếp (đồng bộ, không đồng bộ) với các lời tin độ dài khác nhau (5÷8 bit) và 1 bit chẵn lẻ để kiểm tra sự đúng đắn của truyền tin.

## 1. Cổng 8251A

Cổng 8251A có sơ đồ khối như hình 1.23.




**HÌNH 1.23.** Sơ đồ khối của 8251A.

- Thanh ghi chế độ chỉ các hành động sau:
  + Số bit dừng ($d_7$, $d_6$).
  + Độ dài mã kí tự ($d_3$, $d_2$).

+ Hệ số nhân tốc độ ($d_1$, $d_0$).

+ Kiểm tra chẵn lẻ ($d_5$, $d_4$).

- Thanh ghi lệnh chỉ các hành động sau:

+ Cho phép tìm kiếm ký tự đồng bộ ($d_7 = 1$).

+ Xóa (reset) các thanh ghi nội ($d_6 = 1$).

+ Gửi ký tự gián đoạn ($d_3 = 1$).

+ Cho phép thu ($d_2 = 1$), cho phép phát ($d_0 = 1$).

+ Xóa có lỗi ($d_4 = 1$).

+ Điều khiển thiết bị cuối modem sẵn sàng ($d_1 = 1$).

- Thanh ghi trạng thái chỉ trạng thái của vi mạch:

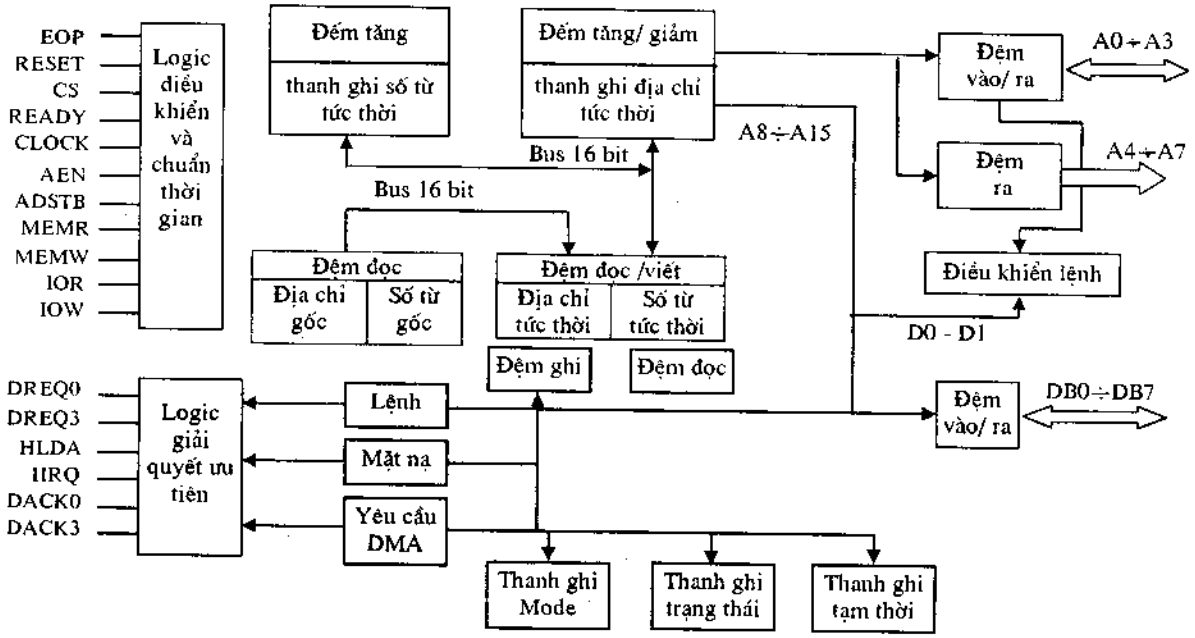
+ Modem sẵn sàng ($d_7 = 1$).

+ Nhận được ký tự đồng bộ ($d_6 = 1$).

+ Các sai số (tràn khung $d_5 = 1$, thu đè $d_4 = 1$, lỗi chẵn lẻ $d_3 = 1$).

+ Đệm thu và phát rỗng ($d_2 = 1$).

+ Sẵn sàng thu ($d_1 = 1$) và sẵn sàng phát ($d_0 = 1$).




**Hình 1.24.** Sơ đồ khối 8237A.

- Thanh ghi này giúp bộ vi xử lý trao đổi tin nối tiếp qua và không qua modem với thiết bị ngoài nối tiếp. Nếu không có modem, phải nối các cặp tín hiệu $\overline{DTR} - \overline{DSR}$, $\overline{RTS} - \overline{CTS}$, còn có modem, đó là các tín hiệu điều khiển modem (DTR, RTS) và nhận trả lời về trạng thái của modem ($\overline{DSR}$, $\overline{CTS}$).

## 2. Cổng 8250/16450

Là thế hệ sau của 8251A, có nhiều tính năng hơn (10 thanh ghi), có riêng một thanh ghi điều khiển modem, có tín hiệu báo chuông, các tín hiệu ra, vào theo chuẩn RS-232C của truyền thông và hai lối ra không theo chuẩn truyền thông. Cổng có các chế độ khác nhau (đồng bộ, không đồng bộ và lai).

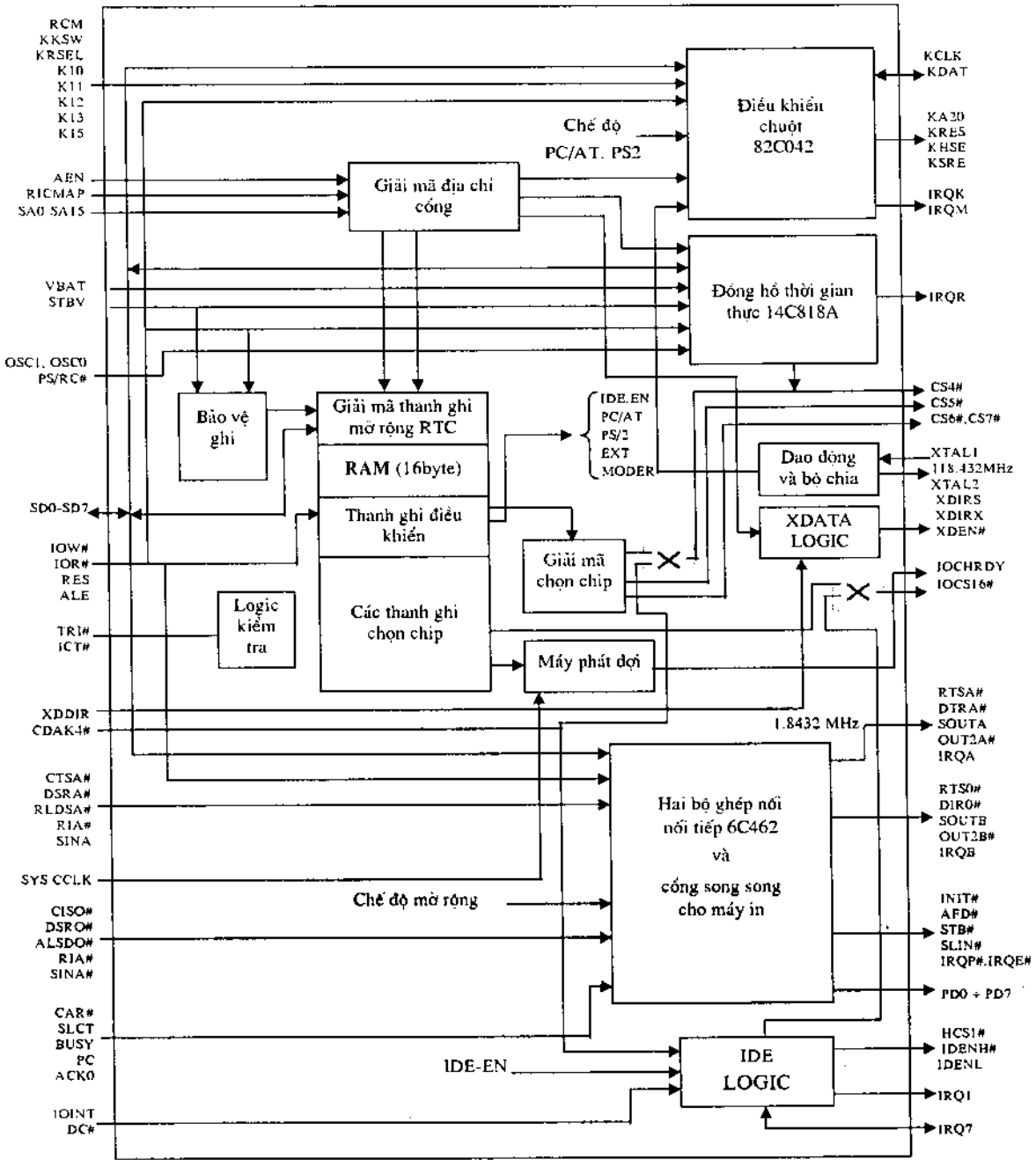
Vi mạch 16450 tương thích với 8250 nhưng làm việc với tốc độ cao hơn trong môi trường đa nhiệm và có 11 thanh ghi.

## 1.4.3. CỔNG VÀO/RA TRỰC TIẾP KHỐI NHỚ - DMA 8237A

Cổng này thay mặt vi xử lý (VXL bị cô lập với đường dây) để trao đổi thông tin vào/ra trực tiếp khối nhớ mà không qua thanh ghi chứa (AX, EAX) của VXL.

Vi mạch 8237A có sơ đồ khối như hình 1.24.

Trước khi 8237A điều khiển đường dây trao đổi thông tin trực tiếp giữa khối nhớ M và




**Hình 1.25.** Sơ đồ khối của 82341.

thiết bị ngoài, 8237A phải trao đổi tin với VXL để chuẩn bị quá trình trao đổi tin trực tiếp qua 4 thanh ghi:

- ***Thanh ghi yêu cầu DMA***: chọn 1 trong 4 kênh.

- ***Thanh ghi lệnh***: (đọc/viết bình thường, đọc/viết nhanh, cấm/cho phép địa chỉ kênh 0, chế độ ưu tiên, vòng/cố định, yêu cầu/trả lời ở các mức tích cực thấp/cao... ).

- ***Thanh ghi ở chế độ:*** ghi nhận

  + Chế độ chuyển tin (từng lời, mảng, ghép tầng).

  + Chế độ khởi động (tự khởi động, khi có lệnh).

  + Chọn kênh (0, 1, 2, 3).

  + Hành động (kiểm tra, viết, đọc, không hợp lệ).

- ***Thanh ghi mặt nạ:*** ghi nhận

  Lập/xóa mặt nạ kênh (3, 2, 1, 0).

### 1.4.4. CỔNG VÀO/RA TÍCH HỢP CAO 82341

Cổng này được trang bị trong MVT PC/AT, nó chứa:

- Hai cổng trao đổi tin nối tiếp không đồng bộ 16C450.
- Một đồng hồ nhịp thời gian.
- Một RAM.
- Các khối điều khiển bàn phím và chuột.
- Một khối ghép nối đĩa cứng.

Vì thế vi mạch này được gọi là chip combo thiết bị ngoài, được đóng trong vỏ PQFP 128 chân.

Sơ đồ khối của 82341 ở hình 1.25.

## 1.5. CÁC LINH KIỆN BỔ TRỢ CỦA MÁY VI TÍNH

Ngoài ba thành phần chính là vi xử lý, khối nhớ, cửa vào/ra máy vi tính còn các thành phần bổ trợ là:

- Đồng hồ xung nhịp.
- Các bộ đệm đường dây.
- Khối điều khiển ngắt.
- Timer.
- Giải mã.

Các thành phần này cũng được tạo dưới dạng vi mạch và tùy theo thế hệ của vi xử lý, chúng cũng có biến đổi theo cho phù hợp.

### 1.5.1. ĐỒNG HỒ XUNG NHỊP

Đồng hồ xung nhịp có nhiệm vụ phát chuỗi xung nhịp cho vi xử lý để:

- Phát các xung tín hiệu lệnh cho các thành phần của khối điều khiển trong vi xử lý để giải mã và thực hiện mã lệnh.

- Phát các chu trình Bus cho vi xử lý như tìm lệnh, đọc/ ghi khối nhớ và các cổng vào/ra.

Đối với họ vi mạch Intel, có các mạch đồng xung nhịp sau.

- **8224** cho vi xử lý (riêng 8085 không cần đồng hồ xung nhịp vì có sẵn trong vi xử lý).

- **8284** cho vi xử lý 8086/8088 của máy vi tính PC/XT.

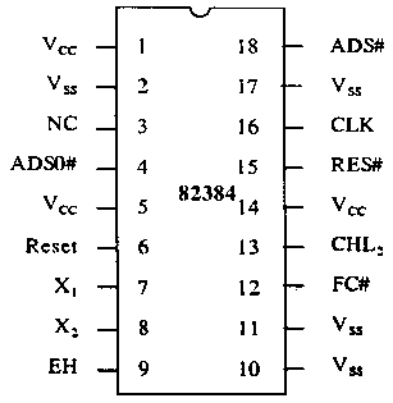
- **82284** cho vi xử lý 80286 của máy vi tính PC/AT và 80384 cho vi xử lý 80386 của máy vi tính OS/2.

- **82489DX** có một đồng hồ xung nhịp 32 bit và các khối khác (xem ở dưới) cho vi xử lý Pentium.

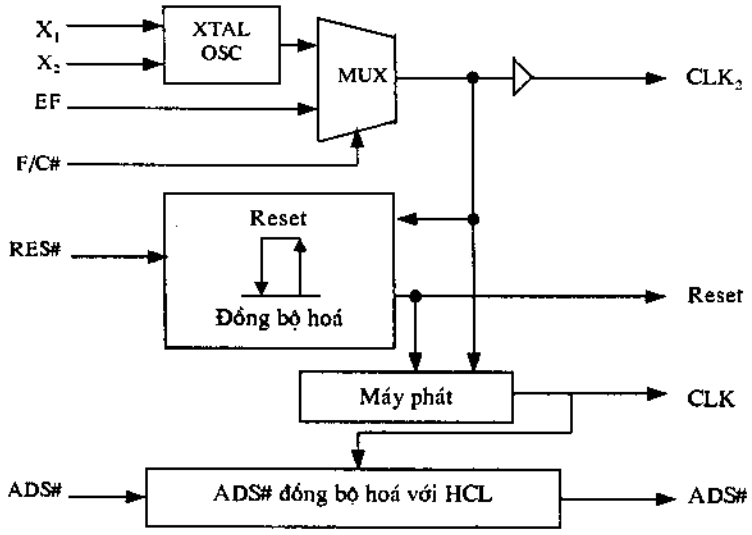
## 1. Đồng hồ xung nhịp 8224

8224 là đồng hồ xung nhịp đầu tiên của hãng Intel chế tạo dành cho vi xử lý 8088, có sơ đồ chân tín hiệu và sơ đồ khối như hình 1.26 và 1.27.

8224 có thể phát chuỗi xung có tần số khác nhau (do tinh thể thạch anh mắc vào các lối vào XTAL1,XTAL2 và cuộn cảm TANIC, ví dụ trong hệ KMOV1 có tần số 18,432 MHz) bắt đầu phát khi có xung kích thích vào RESIN, RDYIN (để đồng bộ). Các xung đồng hồ ra được sử dụng với mức điện áp khác nhau ($\phi$1,$\phi$2,$\phi$3 (TTL)), còn các xung ra khác (OSC, STSTB) để điều khiển vi mạch khác.




Hình 1.26. Sơ đồ chân tín hiệu của 82384.




Hình 1.27. Sơ đồ khối của 82384.

## 2. Đồng bộ xung nhịp 82384

82384 được chế tạo tương tự 8224 nhưng đơn giản hơn cho vi xử lý 8086/8088 của máy vi tính PC/XT. Sơ đồ chân và sơ đồ khối của 82384 như hình 1.26 và 1.27.

## 1.5.2. CÁC BỘ ĐỆM BUS (ĐƯỜNG DÂY)

Các đường dây của hệ vi xử lý cần phải có các bộ đệm để:

- Ghi lại các tín hiệu về địa chỉ, số liệu, trạng thái (hay lệnh) từ vi xử lý để phát cho các vi mạch ngoài.

- Đổi chiều thuận nghịch cho các tín hiệu một chiều.

- Tăng công suất của tín hiệu.

Về nguyên tắc ta có thể dùng các thanh ghi (ví dụ, 74273/373, 74 LS245 cho số liệu, 74LS 244 cho địa chỉ) như có thể dùng các thanh ghi chế tạo riêng cho các hệ vi xử lý Intel như:

- 8212 cho hệ 8080A, đệm cả số liệu và địa chỉ.

- 8282 đệm cho hệ 8086.

- 8216 (không đảo) và 8226 (đảo tín hiệu để biến đổi đường dây một chiều thành hai chiều thuận nghịch).

- 82340 gồm:

  + 82345 đã đệm số liệu cho máy tính PC/AT.

  + 82344 đã đệm đường dây ISA cho máy vi tính PC/SA.

  + 82346 đã đệm đường dây điều khiển cho hệ 386.

Riêng đường dây điều khiển, cần vi mạch đệm trạng thái để giải mã các tín hiệu trạng thái (ví dụ S0, S1, S2 của vi xử lý 8086/ 8088) để đưa ra các tín hiệu lệnh cho vi xử lý độ maximum.

Ta có:

- 8228 cho vi xử lý 8080.

- 8288 cho vi xử lý 8086/8088.

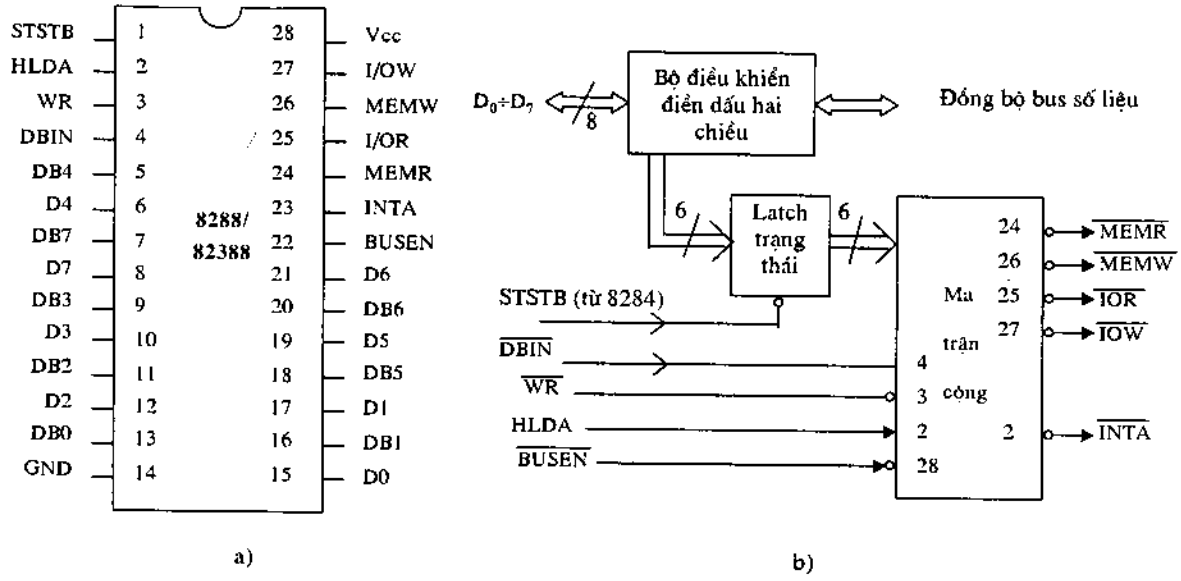
- 82288 cho vi xử lý 80286.

- 82346 cho vi xử lý 80386.

- 82346 cho vi xử lý 80386.

***Bộ đệm Bus 8228/8238***

Vi mạch có sơ đồ chân và sơ đồ khối như hình 1.28.




**Hình 1.28.** Sơ đồ chân tín hiệu (a) và sơ đồ khối (b) của 8288.

Các tín hiệu của vi xử lý có:

- Các tín hiệu trạng thái (S0, S1, S2).

- Các tín hiệu điều khiển (nhịp CLK, lệnh đọc I/O, địa chỉ AEN, lệnh CEN) các tín hiệu phát ra thành hai nhóm.

- Nhóm lệnh đọc/ ghi khối nhớ, đọc/ ghi cổng và các lệnh đọc/ ghi khối nhớ, cổng đọc/ghi bộ nhớ mở rộng.

## 1.5.3. KHỐI XỬ LÝ NGẮT

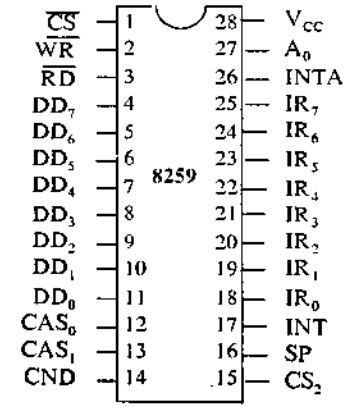
Khối điều khiển ngắt PIC (progammable) được chế tạo để đưa ra tín hiệu yêu cầu ngắt (INTR) vào vi xử lý khi có nhiều yêu cầu phục vụ cho các thiết bị ngoài.

Nguyên tắc chung của sự hoạt động của PCI như sau:

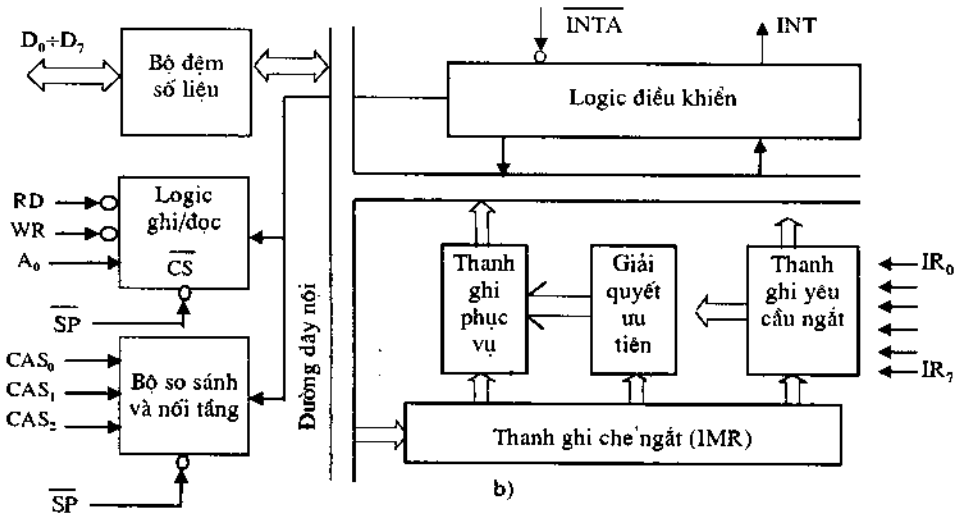
- Các tín hiệu yêu cầu trao đổi tin hay phục vụ từ thiết bị ngoài được đưa ghi vào thanh ghi yêu cầu ngắt.

- Tùy theo có được cho phép yêu cầu ngắt (do vi xử lý ghi vào thanh ghi chắn trước). Và mức ưu tiên (do vi xử lý ghi mức ưu tiên hiện hành) sẽ có yêu cầu ngắt (INTR) của vi xử lý.

- Vi xử lý nhận yêu cầu, làm thủ tục ngắt, đưa ra tín hiệu xác nhận ngắt (INTA) để đọc véctơ ngắt mã chứa




a)




b)

**Hình 1.29.** Sơ đồ chân tín hiệu (a) và sơ đồ khối (b) của xử lý ngắt 8259A.

địa chỉ của chương trình con phục vụ ngắt của thiết bị ngoài tương ứng đã đưa yêu cầu cần ngắt và được vi xử lý cho phép.

Các khối xử lý ngắt của vi xử lý Intel là:

- **8214** cho các vi xử lý 8080/8085.

- **8259A** cho các vi xử lý 8086/8080, 80286.

- **82489 DX** cho các vi xử lý Pentium (còn chức năng đồng hồ đệm bus).

Sơ đồ chân, ký hiệu và sơ đồ khối của 8259A như hình vẽ 1.29 a, b.

Từ sơ đồ khối. ta có:

- Thanh ghi yêu cầu ngắt (IRR), địa chỉ $A_0 = 0$ dùng để ghi 8 yêu cầu ngắt $IR_0 \div IR_7$ của thiết bị ngoài ($IR_0$ ưu tiên cao nhất).

- Thanh ghi chắn (mặt nạ ngắt - IMR), địa chỉ $A_0 = 1$, dùng để ghi thanh chắn ngắt từ vi xử lý.

- Thanh ghi phục vụ hay mức ngắt hiện hành (ISR) cho biết mã hay véctơ ngắt của một mức ngắt đi qua được thanh ghi chắn ngắt được đọc vào vi xử lý.

- Mạch giải quyết ưu tiên PR cho một mức ngắt được ghép tại một thời điểm để:

+ Mã hóa thành véctơ ngắt (tức số mức thập phân thành nhị phân) để ghi vào thanh ghi phục vụ (ISR).

+ Sinh ra tín hiệu yêu cầu ngắt INTR ở khối logic điều khiển để đưa vào lối vào INTR của vi xử lý.

+ Khối logic điều khiển tạo tín hiệu yêu cầu (INTR) và nhận tín hiệu xác nhận ngắt INTA từ vi xử lý để đọc véctơ ngắt (nội dung thanh ghi phục vụ ISR) vào vi xử lý. Ngoài ra khối logic này còn có các thanh ghi điều khiển khởi động ($ICW_1$, $ICW_2$ , $ICW_3$) và các thanh ghi điều khiển hành động ($OCW_1$, $OCW_2$ , $OCW_3$).

Trong một hệ máy vi tính, ta có thể nối tầng một 8259A chủ với nhiều 8259A tớ để tạo nhiều mức ngắt bằng các tín hiệu $CAS_0$, $CAS_1$, $CAS_2$ (để so sánh) và SP (Slave Program) để chọn 8259A tớ (SP = 1 thì 8259A là chủ SP = 0 thì 8259A là tớ).

Trong máy vi tính PC/AT có 16 mức ngắt (nối tầng một chủ, một tớ) và các chương trình phục vụ ngắt tương ứng của hệ điều hành INTn với chỉ số n chỉ véctơ ngắt như sau:

| IRQ | INTn | Dành cho |
|---|---|---|
| IRQ0 | INT 8h | Bộ đếm 8254 cho TC |
| IRQ1 | INT 9h | Bàn phím |
| IRQ2 | INT Ah | Yêu cầu ngắt từ 8259A |
| IRQ3 | INT Bh | COM2 |
| IRQ4 | INT Ch | COM1 |
| IRQ5 | INT Dh | Cổng máy in song song LPT2 |
| IRQ6 | INT Eh | Điều khiển đĩa mềm |
| IRQ7 | INT Fh | Cổng máy in song song LPT1 |
| IRQ8 | INT 70h | CMOS của đồng hồ thời gian thực |
| IRQ9 | INT 71h | Định hướng lại chương trình INT 0Ah |
| IRQ10 | INT 72h | Dành cho người sử dụng |
| IRQ11 | INT 73h | Dành cho người sử dụng |
| IRQ12 | INT 74h | Chuột của PS/2 |
| IRQ13 | INT 75h | Đồng xử lý toán học |
| IRQ14 | INT 76h | Ổ đĩa cứng |
| IRQ15 | INT 77h | Dành cho người sử dụng |

## 1.5.4. BỘ ĐỊNH THỜI KHOẢNG/ BỘ ĐẾM (TIMER/ COUNTER PIT 8253/8254)

Trong các hệ vi xử lý, vi mạch bộ định thời khoảng/ bộ đếm PIT (programmable Interval Timer) sử dụng để phát ra chuỗi xung tạo nhịp, âm thanh ra loa,một xung với độ kéo dài định sẵn, đếm thời gian và sự kiện, ghi tần số v.v...

Hệ vi xử lý 386 có vi mạch 82C54 (tương đương 8254) sơ đồ chân tín hiệu và sơ đồ khối 8253/54 như hình 1.30.

8253/8254 gồm có bộ đếm số liệu, logic điều khiển ghi/ đọc, thanh ghi lời điều khiển và ba bộ đếm. Các bộ đếm dùng để đếm (số đếm đọc ở $D_0 \div D_7$) hoặc chia tần số (vào bảng 1.1 chọn bộ đếm/ thanh điều khiển $CLK_0$,$CLK_1$, $CLK_2$ ra out0, out1, out2) với các tín hiệu điều khiển Gate (Gate 0, Gate 1, Gate 2). Các chân tín hiệu $A_0$, $A_1$ để chọn một trong bốn thanh ghi lời điều khiển và ba bộ đếm.

***Bảng 1.1.*** **Địa chỉ của các bộ đếm**

| $A_1$ | $A_0$ | Chọn |
|---|---|---|
| 0 | 0 | Bộ đếm 0 |
| 0 | 1 | Bộ đếm 1 |
| 1 | 0 | Bộ đếm 2 |
| 1 | 1 | Thanh ghi lời điều khiển |

Trong máy vi tính PC có các cổng cho các thành phần trên:

| Cổng 1 | Cổng 2 | Thanh ghi | Lệnh |
|---|---|---|---|
| 040H | 048H | Bộ đếm 0 | Đọc/ ghi |
| 041H | 049H | Bộ đếm 1 | Đọc/ ghi |
| 042H | 04AH | Bộ đếm 2 | Đọc/ ghi |

Chế độ đọc ngược chỉ có ở 8254.

Trong máy tính PC, mạch 8254 được sử dụng với:

- Xung nhịp, tần số 1,19318 MHz vào các lối vào khiển $CLK_0$, $CLK_1$, $CLK_2$.

- Các cổng Gate 0, Gate 1, nối với 5V (luôn cho phép bộ đếm làm việc còn cổng Gate 2 nối với tín hiệu cho phép loa kêu từ một lối ra của 8255A).

- Các lối ra out cho:

+ Out 0 cho tín hiệu 18,2 Hz đến IRQ0 của 8259A để yêu cầu cho đồng hồ hệ thống.

+ Out 1 cho chuỗi xung số 866278 Hz đến $IRQ_0$ của 8237A để làm tươi bộ nhớ DRAM.

+ Out 2 cho chuỗi xung tần số 896 Hz đến khuếch đại cho loa.

## 1.6. CÁC VI MẠCH GHÉP NỐI VỚI THIẾT BỊ NGOÀI

Máy vi tính nói chung, PC-IBM nói riêng đều nối với thiết bị ngoài để đưa tin vào, đưa tin ra như bàn phím, màn hình đĩa mềm, đĩa cứng và các thiết bị tương tự khác.

Về nguyên tắc, có thể dùng các cổng vào/ra (song song, nối tiếp) để trao đổi thông tin với thiết bị ngoài kiểu số. Nhưng người ta đã chế tạo ra các vi mạch riêng cho từng loại trên vì còn các nhiệm vụ khác như tạo mã ( bàn phím), điều khiển (ổ đĩa quay, hạ đầu từ v.v...). Với các thiết bị ngoài tương tự, ngoài trao đổi tin và điều khiển thiết bị ngoài, vi mạch ghép nối phải biến đổi tin từ tương tự sang số (ADC) và số sang tương tự (DAC).

### 1.6.1. VI MẠCH GHÉP NỐI BÀN PHÍM

Vi mạch có nhiệm vụ:

- Biến đổi mã quét từ bàn phím thành mã ASCII (7 bit, 8 bit) đặc trưng cho chữ (A÷Z), (số 0 ÷10) và các ký hiệu khác.

- Ghi nhận mã dạng song song và truyền nối tiếp vào vi xử lý.

- Điều khiển bàn phím và quá trình trao đổi thông tin với vi xử lý.

Vì nhiệm vụ phức tạp như vậy (nhất là quá trình tạo mã), vi mạch ghép nối là một vi xử lý chuyên dùng đối với họ Intel, ta có:

- Vi xử lý 8042 cho máy tính PC/ XT với vi xử lý 8086/8088, 80286 và PS/2.

- Vi xử lý 8048 cho máy tính AT với vi xử lý 80386, 80486 và pentium.

- Vi xử lý 8079, 8279 sử dụng cho các bàn phím đơn giản và hiển thị số cho các vi xử lý 8079, 8279 sử dụng cho các bàn phím đơn giản và hiển thị số cho các vi xử lý đơn giản (kit).

#### 1. Vi mạch điều khiển bàn phím 8048

8048 thực chất là một hệ vi xử lý gồm:

- Một vi xử lý 8 bit.

- Bộ nhớ RAM, ROM.

- Cổng I/O.

- Bộ đệm định thời gian và cách mắc với ma trận bàn phím, bộ khuếch đại chiều của 8048 như hình 1.30.

Trên sơ đồ khối ta thấy:

- Hai cổng đệm vào/ ra và địa chỉ 60h, điều khiển ghi/ đọc bởi các tín hiệu lệnh WR, RD của vi xử lý trung tâm.

- Hai cổng đệm (ghi bởi WR) và trạng thái (đọc bởi RD).

Với các giá trị khác nhau của các bit, ta có một số lệnh cho bộ điều khiển bàn phím như dưới đây.

Thanh ghi trạng thái xác định trạng thái hiện tại của bộ điều khiển bàn phím. Thanh ghi này chỉ đọc qua lệnh ở cổng 64h.

Pare: sai số chẵn lẻ từ byte cuối cùng của bàn phím hoặc thiết bị phụ (chỉ ở PS/2).

1 = byte cuối cùng có sai số lẻ chẵn, 0 = không có.

TIM: lỗi quá thời gian (time out)

1 = lỗi, 0 = không.

AUXB: đệm ra cho thiết bị phụ (chỉ ở PS/2)

1 = giữ số liệu cho thiết bị, 0 = giữ số liệu cho bàn phím.

KEYL: trạng thái khoá bàn phím

1 = không khoá, 0 = khoá.

C/$\overline{D}$: lệnh/ số liệu

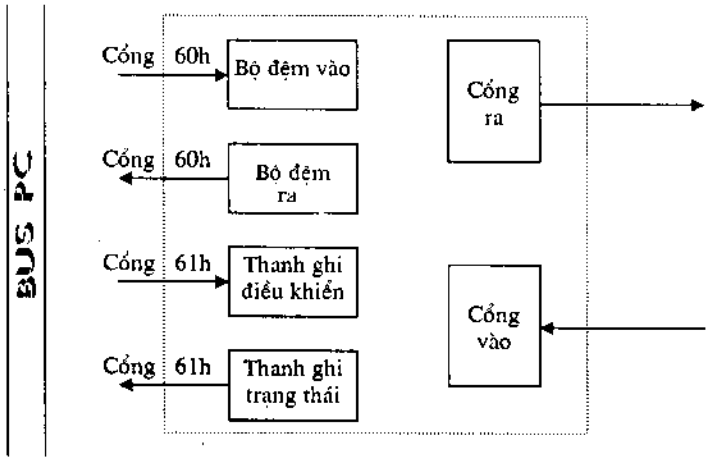
1 = ghi byte lệnh lên qua cổng 64h, 0 = ghi byte số liệu qua cổng 64h.

INPB: trạng thái đệm vào

1 = số liệu vi xử lý trong bộ đệm vào, 0 = đệm vào rỗng.

OUTB: trạng thái đệm ra

1 = số liệu bộ điều khiển bàn phím trong bộ đệm ra, 0 = đệm vào rỗng.




**Hình 1.30.** Sơ đồ khối của 8048.

## 2. Vi mạch điều khiển bàn phím và hiển thị số 8279

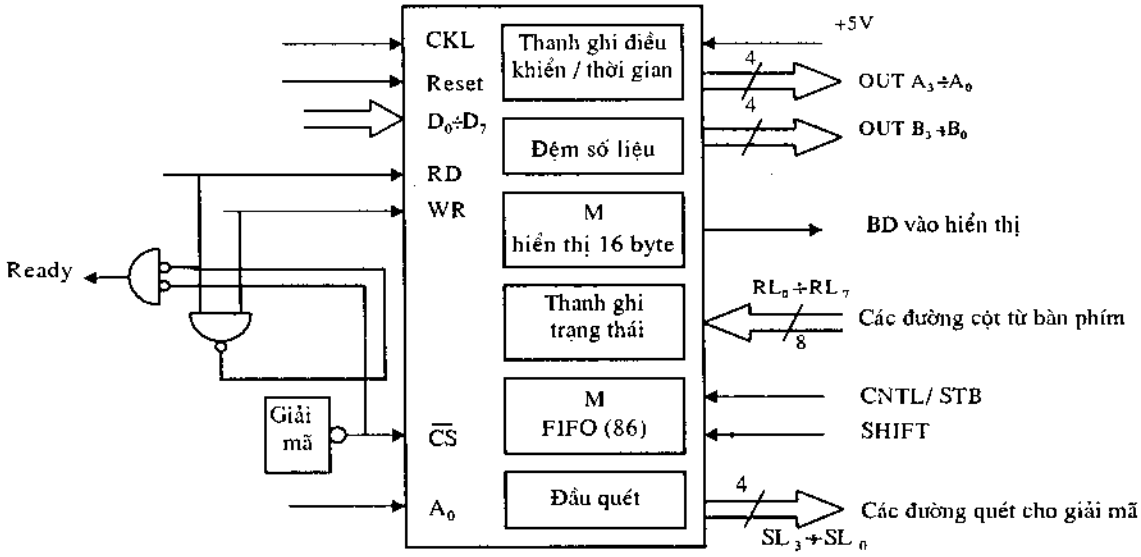
8279 là vi mạch điều khiển và hiển thị số dùng cho các kit của hệ vi xử lý. Sơ đồ khối của 8279 như hình 1.31.

Ngoài các thanh ghi thông dụng (điều khiển/ thời gian, trạng thái đệm số liệu) của bất kỳ mạch điện tử nào, 8279 còn có:

- Thanh ghi đếm quét để đưa 4 bit số liệu, $SL_0 \div SL_3$, cho bộ giải mã chọn hàng phím hay số liệu hiển thị.

- Bộ nhớ hiển thị (16 byte) để đưa số liệu cho các bộ hiển thị theo các đường dây ($A_0 \div A_3$), ($B_0 \div B_3$).

- Bộ nhớ sensor (8 byte) loại FIFO (first in first out) để ghi đệm các đường cột từ bàn phím $R_0 \div R_7$ ghi nhận bàn phím đã nhấn.




**Hình 1.31.** Sơ đồ khối của 8279.

## 1.6.2. BẢN MẠCH ĐIỀU KHIỂN MÀN HÌNH

Màn hình mầu là thiết bị điện tử phức tạp, có 3 súng phóng điện tử cho 3 mầu cơ bản (red, blue, green) và các bộ phận điều khiển lệnh quét dòng, quét cột của chùm tia điện tử đập vào màn. Hơn nữa có hai chế độ hiển thị (ký tự, đồ họa) nên để điều khiển màn hình cần:

- Khối điều khiển.

- Khối nhớ RAM có dung lượng bit bằng số pixel (điểm) trên toàn màn (số điểm một hàng x số cột).

- Bộ tạo ký tự.

- Mạch thời gian tạo các tín hiệu đồng bộ ngang, đồng bộ dọc và cường độ chùm tia.

Tùy độ phân giải (số pixel hàng x số pixel cột), số mầu nên dung lượng bộ nhớ khác nhau.

Đối với máy vi tính PC IBM có một số bản mạch (bìa) màn hình sau:

- MDA (1981): Điều khiển màn hình đơn sắc, dung lượng 4 KB nhớ; 2 KB thuộc tính + 2 KB ký tự.

- CGA (4/1987): Độ phân giải 320 x 200 pixel với 4 mầu 4 chế độ, cần 16 K = 4 K x 4 trang. Dùng MC 6845 để điều khiển.

- EGA (1984), (640 x 350) pixel, 16 mầu.

- VGA (4/1987) tương thích với CGA, EGA (640 x 840) pixel, 256 mầu cho máy vi tính PS/2.

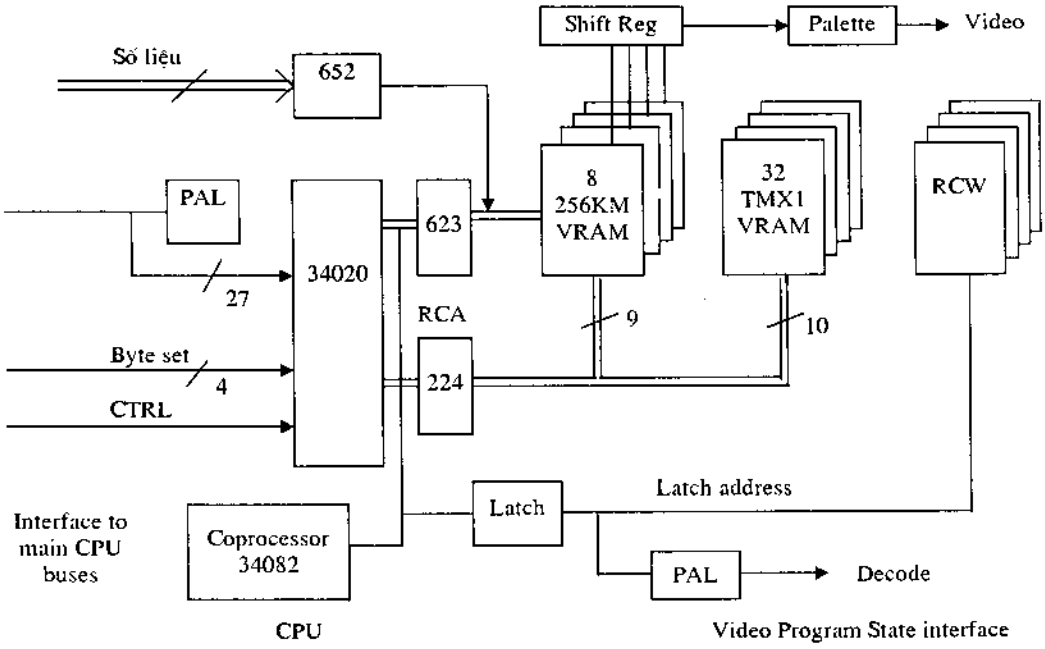
- SVGA, (1024 x768) pixel, 256 mầu.

- XGA (1990), (1024 x768) pixel, 256 mầu có đồng vi xử lý đồ họa dùng cho máy PS/2.

- TMS 340/TIGA dùng vi xử lý 34010 hay 34020 để xử lý đồ họa 3 chiều (3D), độ phân giải 1024 x 768 pixel cho mỗi chiều, tốc độ 60 MHz.

- 8514A (1987) có trong máy vi tính PS/2, độ phân giải 1024 x 768.

Sơ đồ khối bìa điều khiển màn hình với TMS 34020 có sơ đồ như hình 1.32.




**HÌNH 1.32.** Giới thiệu sơ đồ khối TMS 34020
(có thể nối với vi xử lý trung tâm và điều khiển hiển thị 1024 x 1024 x 8 bit/ pixel).

## 1.6.3. VI MẠCH ĐIỀU KHIỂN ĐĨA TỪ

Đĩa từ (mềm và cứng cũng như đĩa quang) là khối nhớ ngoài có dung lượng lớn ( > =1,42 MB tới hàng 1000 GB).

Ổ đĩa gồm các bộ phận chính sau:

- Trục quay giữa đĩa ở tâm và quay với vận tốc 300 vòng/ phút.
- Đầu từ và thanh mang để định vị đầu từ đọc/ghi chính xác trên rãnh và cung (sector) mong muốn.
- Cảm biến chỉ dấu cho biết sự có mặt của lỗ chỉ dấu trong đĩa.
- Cảm biến rãnh 00 cho biết khi đầu từ ở rãnh 0.
- Cảm biến bảo vệ ghi chỉ báo khi không cho phép ghi.
- Ngoài ra còn có thể có hai bộ phận phụ:
  + Cảm biến mở rộng.
  + Cuộn dây ép đầu từ chỉ cho đầu từ tiếp xúc đĩa trong thời gian ghi/ đọc.

Việc điều khiển đĩa cần có động tác như quay đĩa, xác định vị trí (cung, sector, hạ đầu từ, ghi/ đọc).

Việc ghi/ đọc trên đĩa song song, ở chế độ DMA.

## 1. Vi mạch điều khiển đĩa mềm

Sơ đồ của khối điều khiển đĩa mềm (hình 1.33) gồm:

- Vi xử lý điều khiển ổ đĩa.

- Mạch điều khiển: điều khiển việc quay đĩa ghi/ đọc.

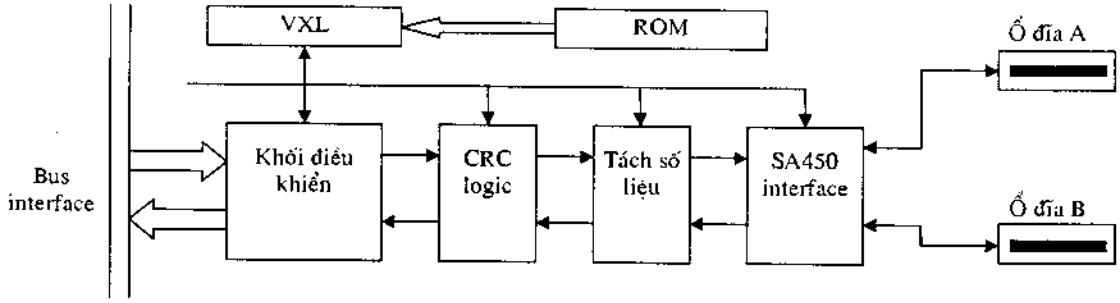
- Mạch logic CRC: kiểm tra sự đúng đắn của việc ghi/ đọc.

- Khối nhớ ROM: ghi vi mã (microcode) của chương trình điều khiển ổ đĩa.

- Bộ phận thao tác số liệu từ chuỗi tín hiệu đã mã hoá điều tần FM, điều tần có sửa đổi (MFM, M2FM) và mã hoá nhóm (GCR).

- Ổ đĩa và khối ghép nối SA 450.

Hãng Intel đã chế tạo các vi mạch mật độ cao để điều khiển đĩa mềm là 8271, 8272. Hình 1.33 giới thiệu sơ đồ khối của 8272.

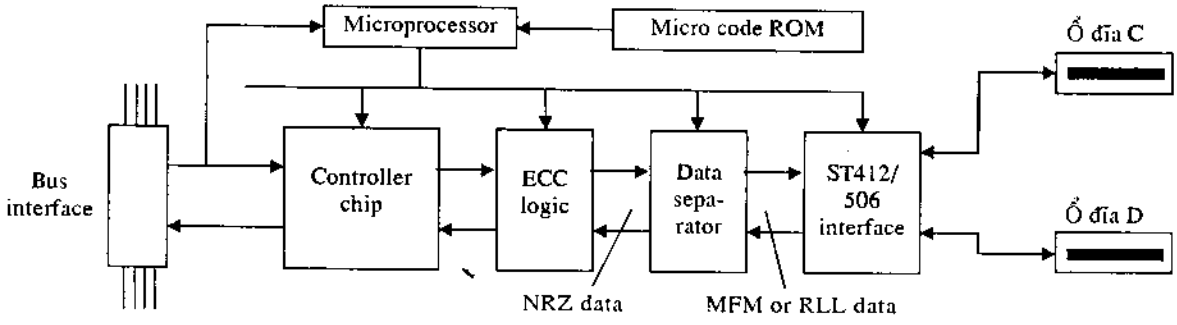



**Hình 1.33**. Giới thiệu sơ đồ khối của 8272.

Vi mạch 82077A cho máy tính PC/AT cho cả hai loại đĩa 5,5 và 3,5 inch. Vi mạch điều khiển đĩa 80277 liên hệ với đường dây của PC/AT qua vi mạch mật độ lớn 82341.

## 2. Vi mạch điều khiển ổ cứng

Vì đĩa cứng có dung lượng lớn nên phải tổ chức thành nhiều đĩa nhỏ xếp chồng nhau với nhiều đầu từ ghi/ đọc, do đó cách điều khiển cũng phức tạp hơn. Hơn nữa tốc độ quay của đĩa phải lớn (3600 vòng/ phút so với 3000 vòng/ phút của đĩa mềm).




**Hình 1.34**. Sơ đồ của khối điều khiển đĩa cứng.

Sơ đồ khối của bộ điều khiển đĩa cứng cũng tương tự ở ổ đĩa mềm (hình 1.34) nhưng có điều khác là:

Mạch logic kiểm tra mã ECC thay cho mạch logic CRC. Mạch ECC này không chỉ kiểm tra sai số (giống CRC) mà còn có thể sửa chúng trong trường hợp lỗi không quá nghiêm trọng bằng các byte ECC.

Khối ghép nối ST412/506 thay cho SA450 để điều khiển ổ đĩa.

Với ổ đĩa có hai loại chuẩn ghép nối.

+ ST- 506 hoặc ESDI điều khiển hai ổ đĩa.

+ SCSI điều khiển tới tám ổ đĩa.

- ST- 504: là khối ghép nối đầu tiên (1980) bởi Seagate (người tạo ra ổ đĩa cứng đầu tiên) cho đĩa 5 MB. Sau đó hãng Hewlett Packard đã chế tạo ST 506/412, ST 412/ HP điều khiển ổ đĩa cứng với dung lượng (20--> 100MB), với tốc độ trao đổi tin 5 MIPS ở mã RLL và có thêm một bộ đệm để tìm tin.

+ ESDI: được sử dụng cho máy tính PS/2 (vi xử lý 80386), là sự tiến triển của ST- 506 có thể điều khiển được đĩa 100MB đến 137 GB (theo lý thuyết) với tốc độ 23 MIPS.

+ IDE hay BVS- AT 1988 với khối điều khiển có thể tích hợp lên đĩa, dung lượng 20MB tới 200MB với tốc độ 32MIPS. Sau đó có loại cải tiến EIDE với đĩa trên 500 MB.

SCSI với các phiên bản SCSI-1, SCSI- 2, SCSI- 3 và tốc độ 4 MIPS đến 40 MIPS. Cấu trúc của hệ điều khiển với SCSI này có thể tổ chức thành một SCSI chủ liên hệ với máy tính và điều khiển bảy SCSI phụ khác (theo lý thuyết có thể tới 31). Hệ SCSI này có thể:

- Trao đổi SCSI phụ thông qua SCSI chủ không cần sự can thiệp của vi xử lý của máy vi tính.

- Có thể ghép nối SCSI với các thiết bị ghi/ đọc từ tính khác (đĩa cứng, đĩa quang, máy đọc, máy in nối tiếp, mạng v.v...).

*CHƯƠNG 2*

# ĐẢM BẢO CHƯƠNG TRÌNH CHO HỆ MÁY VI TÍNH PC - IBM

Đảm bảo chương trình hay phần mềm của một máy vi tính gồm hệ các chương trình của nó. Đó là:

- Các chương trình của hệ điều hành đảm bảo cho hệ thống thiết bị (phần cứng) hoạt động như điều khiển các thiết bị ngoài trao đổi (vào/ra) số liệu, thực hiện một số hành động từ bàn phím...

- Các chương trình biên dịch (compilator), phiên dịch (interpreter) để dịch các ngôn ngữ bậc cao sang mã lệnh dạng ngôn ngữ máy.

- Các chương trình soạn thảo (editor), kết nối (linker) và gỡ rối (debugger) để hỗ trợ cho việc ghi, kết nối và tìm lỗi của chương trình sử dụng trên máy vi tính.

- Các chương trình tiện ích (utilities), các chương trình hữu ích cho người sử dụng máy (NC của norton), soạn thảo văn bản (winword), vẽ đồ hoạ, bảng, biểu đồ (excel)...

Phần mềm có cơ sở là lệnh dạng mã (gọi là mã lệnh), có thể thực hiện trực tiếp trên phần cứng không cần phiên dịch. Sự phát triển của phần cứng kéo theo sự phát triển của phần mềm (các ngôn ngữ phát triển bậc cao và các chương trình tiện ích thuận tiện cho người dùng) và ngược lại, phần mềm đòi hỏi sự phát triển của phần cứng như đòi hỏi chế tạo các vi xử lý nhanh, chuyên dùng (vào/ra, xử lý dấu phẩy động, đồ họa, truyền thông...).

Khi đã có một hệ thống phần cứng và phần mềm cho một máy vi tính, ví dụ PC của hãng IBM, chúng ta phải:

- Hiểu rõ về cấu trúc, hoạt động và ghép nối với phần cứng.

- Hiểu rõ phần mềm hệ thống (hệ điều hành, phần mềm tiện ích - các chương trình tiện dụng) để khai thác, sử dụng.

- Phát triển phần mềm tức là viết các chương trình ứng dụng của người dùng, cho chạy thử, kiểm tra và ghi thường trú trong bộ nhớ, làm phong phú thêm thư viện của máy để khi cần dùng, chỉ cần đưa lệnh gọi.

Đó là mục tiêu của người lập trình nói chung và người lập trình bằng hợp ngữ trên máy vi tính PC của hãng IBM (các máy với các vi xử lý Intel 8086/8088, 80286, 80386, 80486 và Pentium 80586) nói riêng.

Phần mềm hệ thống của một máy vi tính (MVT) gồm:

- Hệ điều hành.
- Các chương trình dịch các ngôn ngữ.
- Các chương trình tiện ích hay thư viện chương trình.
- Các chương trình của người sử dụng MVT.

# 2.1. HỆ ĐIỀU HÀNH

## 2.1.1. ĐẠI CƯƠNG

### 1. Định nghĩa và vai trò

Hệ điều hành (HĐH) là hệ chương trình điều khiển mọi hành động của một máy tính điện tử. HĐH đóng vai trò quan trọng trong cả phần cứng vì nó điều khiển phần cứng hoạt động. Nếu không có HĐH, hệ máy tính không thể hoạt động, chỉ là một chiếc máy chết.

Hệ điều hành thực hiện chương trình do một trong ba nguồn sau:

- Điều khiển các bộ phận của máy tính (vi xử lý, bộ nhớ, cửa vào/ra), do thực hiện chương trình trong các tệp chứa trên đĩa.

- Làm việc theo lệnh điều khiển của người điều khiển máy (operator) truyền lệnh qua bàn phím.

- Làm việc theo yêu cầu của thiết bị ngoài (yêu cầu ngắt chương trình bằng tín hiệu ngắt cứng) hoặc bởi lệnh ngắt chương trình INT nh viết trong chương trình (ngắt mềm).

### 2. Nhiệm vụ

Mỗi một nhiệm vụ của HĐH do một khối chương trình đảm nhiệm, đó là:

***a) Điều khiển xử lý trung tâm để thực hiện tiến trình (processus)***

Một tiến trình là một khoảng thời gian của chương trình đang thực hiện một nhiệm vụ nào đó. Một chương trình của người sử dụng ở thời gian bị phân chia cũng là một tiến trình.

Với quan điểm của hệ điều hành, một tiến trình là một thực thể (entity) nhỏ nhất được điều hành. Nó gồm một mã, những số liệu và đặc tính (attribut). Ở thời điểm này, ta có thể coi một tiến trình là một công việc hay một chương trình ở thời gian được phân chia cho tiến trình.

Trong khi liên hệ với các hành động của tiến trình, hệ điều hành chịu trách nhiệm về những hành động sau:

- Tạo ra và bãi bỏ những tiến trình của người sử dụng và của hệ.

- Hủy và nhận các tiến trình.

- Tạo ra cơ chế đồng bộ hóa các tiến trình.

- Tạo ra các cơ chế liên lạc (communication) giữa các tiến trình.

- Tạo ra các cơ chế để xử lý các che chắn lẫn nhau (interblocage).

***b) Điều khiển bộ nhớ trung tâm***

Muốn thực hiện một chương trình phải biến đổi các địa chỉ của các lệnh và số liệu thành một loại địa chỉ tuyệt đối và tích vào một vùng của bộ nhớ (tệp chương trình có đuôi .COM). Khi thực hiện chương trình, vi xử lý truy nhập tới lệnh và số liệu của bộ nhớ bằng cách phát các địa chỉ tuyệt đối này. Chương trình kết thúc, vùng nhớ của nó đã được khai báo và chương trình tiếp theo có thể được nạp và thực hiện.

Nhằm cải tiến việc sử dụng vi xử lý và vận tốc đáp ứng của máy tính đối với người sử dụng, chúng ta phải ghi giữ nhiều chương trình trong bộ nhớ. Có nhiều sơ đồ điều khiển bộ nhớ khác nhau. Điều đó phản ánh việc tiến gần tới sự điều khiển bộ nhớ và tính hiệu quả của các giải thuật khác nhau, phụ thuộc vào tình hình cụ thể. Sự lựa chọn một sơ đồ điều

khiển bộ nhớ cho một hệ riêng phụ thuộc vào nhiều sự kiện và đặc biệt vào cấu trúc phần cứng của hệ. Mỗi giải thuật đòi hỏi sự hỗ trợ phần cứng riêng của mình.

Liên quan tới điều khiển bộ nhớ, hệ điều hành chịu trách nhiệm những hành động sau:

- Thường xuyên nhận biết phần nào của bộ nhớ đang dùng và bởi ai.

- Quyết định tiến trình nào phải được nạp vào bộ nhớ khi người ta phân bố không gian nhớ.

- Tác động và không tác động tới không gian nhớ khi cần thiết.

Mục đích của việc điều khiển bộ nhớ như sau:

- *Phân bố lại không gian nhớ*: khi một chương trình kết thúc, giải phóng không gian nhớ cần thiết phải tổ chức lại những chương trình trong bộ nhớ nhằm tối ưu hóa việc sử dụng bộ nhớ này.

- *Bảo vệ bộ nhớ*: khi cùng tồn tại nhiều tiến trình ở bộ nhớ trung tâm. Cần bảo vệ mỗi khoảng không gian nhớ khỏi xâm phạm vào các khoảng khác. Hệ quả là tiến trình một chỉ có thể truy nhập tới khoảng nhớ của tiến trình hai khi nó được phép (thông qua mức đặc quyền ghi sẵn - chỉ tới mức đặc quyền thấp hơn).

- *Sự phân chia vùng nhớ*: đôi khi rất cần chia một khoảng nhớ cho nhiều tiến trình. Như vậy, hệ con điều khiển nhớ phải cho phép những sự truy nhập kiểm soát không cần nhân nhượng việc bảo vệ. Ví dụ, một chương trình soạn thảo của một văn bản được chia sẻ bởi nhiều người sử dụng trong một hệ phân chia thời gian.

***c) Điều khiển nhớ phụ***

Khi thực hiện lệnh, những số liệu cũng như chương trình phải nằm trong bộ nhớ chính (trung tâm). Nhưng nhớ chính là quá nhỏ để đặt đồng thời tất cả các chương trình và số liệu. Số liệu sẽ mất khi mất nguồn điện nuôi nên hệ tin học phải cung cấp một bộ nhớ phụ để lưu giữ nhớ chính. Phần lớn hệ tin học hiện đại dùng các đĩa như là thiết bị mang tin theo từng dòng cho cả số liệu và chương trình. Đa số chương trình gồm cả biên dịch (compiler), hợp dịch (assembler), các chương trình con tìm kiếm, những chương trình soạn thảo (editor) và các chương trình tạo dạng (formator) được sử dụng đến khi chúng được tích vào bộ nhớ. Từ nay, sự điều khiển thích hợp bộ nhớ và đĩa là một điều quan trọng trong một hệ tin. Hệ điều hành chịu trách nhiệm những hành động sau, liên quan đến điều khiển đĩa:

– Điều khiển khoảng trống.

– Xếp đặt lại bộ nhớ.

– Điều phối (scheduling) các đĩa.

***d) Điều khiển hệ vào/ra***

Một trong các mục tiêu của HĐH là giấu kín các đặc tính của các thiết bị phần cứng đối với người sử dụng. Ví dụ trong UNIX, những gạch nối riêng rẽ của các thiết bị vào/ra được giấu kín đối với phần lớn chính hệ điều hành bởi hệ vào/ra. Hệ này gồm:

- Một hệ tạo nhớ ngầm (hay ẩn - cache).

- Một giao diện chung của các chương trình điều khiển thiết bị ngoài (driver).

- Những chương trình điều khiển cho các thiết bị ngoài (phần cứng) riêng rẽ.

Chỉ chương trình điều khiển thiết bị ngoài biết đặc tính của thiết bị ngoài riêng rẽ. Sau này, chương trình điều khiển ngắt và các chương trình điều khiển thiết bị ngoài được dùng trong cấu trúc hệ vào/ra có hiệu quả.

*e) Điều khiển tệp*

Điều khiển tệp là một trong những thành phần dễ nhìn thấy nhất của một hệ điều hành. Máy tính có thể tích lũy nhiều loại thiết bi mang tin vật lý khác nhau (băng từ, đĩa từ, đĩa quang). Mỗi thiết bị mang tin có đặc tính riêng và tổ chức vật lý riêng. Mỗi thiết bị được điều khiển bởi một thiết bị như ổ đĩa hay máy quay băng với đặc tính riêng. Các đặc tính riêng này phải kể đến vận tốc, dung lượng, vận tốc chuyển số liệu và phương pháp truy cập (nối tiếp hay trực tiếp).

Nhằm mục đích để có một sự sử dụng thực tế của hệ tin, hệ điều hành cung cấp một quan điểm logic duy nhất. Loại trừ các tính chất vật lý của các thiết bị tích lũy tin để xác định một đơn vị tích lũy logic, đó là tệp. HĐH thiết lập một sự tương ứng giữa các tệp và các thiết bị mang (support) vật lý và truy cập tới các tệp này qua những đơn vị tích lũy.

Một tệp là một tập thể tin liên hệ với nhau, xác định bởi người tạo ra nó. Theo thói quen, các tệp mang các chương trình (dạng nguồn - source và dạng đối tượng - object) và các số liệu. Các tệp số liệu có thể là số, ký tự hay ký tự - số. Chúng có thể chưa được tạo dạng (format), như là các tệp văn bản hay đã tạo dạng triệt để. Một tệp gồm một chuỗi các bit, octet, những đường (track) hay bản ghi (record) mà ý nghĩa được xác định bởi người tạo ra. Khái niệm tệp là một khái niệm rất chung chung. Nhằm sử dụng dễ dàng, các tệp thường tổ chức theo thư mục. Cuối cùng, khi nhiều người dùng, mong muốn kiểm soát sự truy cập và cách truy cập. Liên quan tới tệp, HĐH chịu trách nhiệm các hành động sau:

- Tạo và xóa bỏ tệp.
- Tạo và xóa bỏ thư mục.
- Hỗ trợ ban đầu để thao tác tệp và thư mục.
- Tương ứng giữa các tệp và bộ nhớ phụ.
- Lưu giữ tệp trên những vật mang tin bền vững (không bị xóa).

*f) Hệ bảo vệ*

Nếu một hệ có nhiều người dùng và cho phép các tiến trình (processus) xảy ra đồng thời, các tiến trình khác nhau phải được bảo vệ giữa các hành động của chúng. Có những cơ chế đảm bảo cho các tệp, các mảng nhớ. Vi xử lý và các tài nguyên (resources) khác có thể được sử dụng với những tiến trình có sự cho phép tương xứng của hệ điều hành. Ví dụ: không cho phép người sử dụng nào thực hiện vào/ra riêng của mình, nhằm bảo vệ tính mật độ cao của các thiết bị ngoài khác nhau.

Sự bảo vệ dẫn tới một cơ cấu cho phép kiểm soát sự truy cập của các chương trình, các tiến trình hay những người sử dụng có tài nguyên xác định cho một hệ tin. Cơ chế này phải cung cấp một phương tiện riêng để kiểm soát, cũng như phương tiện áp dụng chúng.

Việc bảo vệ có thể cải tiến độ tin cậy bằng cách phát hiện lỗi ngầm trong các giao diện giữa các thành phần của hệ nhỏ. Một tài nguyên không bảo vệ, không thể tự bảo vệ chống lại việc sử dụng (hay sử dụng kém) của một người dùng không được phép hay kém khả năng. Một hệ có bảo vệ cung cấp một phương tiện phân biệt giữa việc sử dụng cho phép hay không (nhận biết các bit cho phép của các mức đặc quyền hay mật khẩu).

*g) Điều khiển mạng*

Một hệ phân tán là một tập hợp các xử lý có chung bộ nhớ và đồng hồ nhịp. Mỗi xử lý có riêng bộ nhớ địa phương và các xử lý nối chúng lại qua các đường dây khác nhau của

truyền thông (communication) như các bus nhanh và các đường dây điện thoại. Trong một hệ phân tán, các xử lý thay đổi về kích thước và chức năng. Nó có thể bao gồm các vi xử lý, các trạm làm việc (workstation), các máy tính nhỏ mà các hệ tin lớn dùng chung.

Trong hệ, những tiến trình được nối nhờ một mạng truyền thông, trong đó có thể hình dung theo cách khác nhau. Mạng có thể nối từng phần hoặc toàn bộ, hệ điều hành thông thường tổng quát hóa sự truy nhập vào mạng như một dạng truy nhập vào tệp, chi tiết của điều khiển mạng là nội dung trong các chương trình điều khiển (driver) khối ghép nối của mạng.

***h) Hệ phiên dịch lệnh***

Phiên dịch lệnh là một trong những chương trình hệ thống quan trọng nhất cho một hệ điều hành. Nó là một giao diện giữa người dùng và hệ điều hành. Một số HĐH gồm bộ phiên dịch lệnh ở lõi của hệ. Ví dụ MS-DOS, UNIX có bộ phiên dịch lệnh như một chương trình đặc biệt được quay vòng trong khi sự làm việc được bắt đầu hay khi một người dùng được phép xâm nhập vào hệ (trong hệ phân chia thời gian).

Nhiều lệnh được chuyển giao cho HĐH bởi các lệnh điều khiển trung gian. Khi một công việc mới khởi động trong một hệ xử lý theo lô hay khi một người sử dụng xâm nhập vào một hệ trong thời gian được phân chia, một chương trình đọc và phiên dịch các lệnh điều khiển được thực hiện một cách tự động. Có các tên khác nhau cho chương trình này như phiên dịch bìa điều khiển, phiên dịch dòng lệnh Shell (bộ phiên dịch của các lệnh tương tác trong UNIX)... Chức năng của lệnh đơn giản là diễn giải một lệnh thành nhiều lệnh tiếp theo nhau và thực hiện nó. Người dùng sử dụng chuột để chỉ con trỏ vào các hình (icon - biểu tượng) trên màn hình, ở đó giới thiệu những chương trình, những tệp và những chức năng của hệ. Tùy theo chỗ đặt con trỏ, việc ấn nút của chuột có thể gọi một chương trình, chọn một tệp hay thư mục (gọi hồ sơ - dossier) hay lấy ra một thực đơn (menu) chứa các lệnh. Trên các chương trình dịch, những lệnh được gõ lên phím và hiển thị lên màn hình hay máy in với phím Enter (↵) hay CR.

Các lệnh của HĐH bao gồm việc khởi tạo, điều khiển tiến trình, thao tác vào/ra, điều khiển nhớ phụ và nhớ chính, truy cập hệ tệp, cũng như bảo vệ và điều hành mạng.

## 2.1.2. CÁC PHỤC VỤ CỦA HỆ ĐIỀU HÀNH

Một HĐH cung cấp một môi trường để thực hiện chương trình. HĐH đề nghị một số phục vụ cho chương trình và cho người sử dụng chúng. Tất nhiên, các phục vụ riêng phân biệt một HĐH này với HĐH khác nhưng chúng ta có thể nhận dạng một số phần tử chung. Các phục vụ này cải tiến sự thuận tiện cho người viết chương trình, làm dễ dàng cho nhiệm vụ viết chương trình.

### 1. Thực hiện chương trình

Hệ phải nạp một chương trình vào bộ nhớ và thực hiện nó, chương trình phải chỉ rõ lên màn hình sự kết thúc bình thường và không bình thường (chỉ lỗi) của việc thực hiện chương trình.

### 2. Hành động vào/ra

Một chương trình đang thực hiện có thể có yêu cầu vào/ra của tệp hay một thiết bị ngoài vào/ra. Đối với thiết bị đặc biệt, người ta có thể thực hiện các chức năng đặc biệt (như cuộn lại băng hay xóa màn hình).

Với lý do của sự hiệu quả và bảo vệ, người dùng không thể điều khiển trực tiếp các thiết bị vào/ra. Vì vậy HĐH phải cung cấp các phương tiện ảnh hưởng tới vào/ra.

### 3. Thao tác hệ các tệp

Hệ các tệp là một điều có lợi đặc biệt, hiển nhiên là chương trình cần đọc và ghi tệp. Cần thiết phải tạo ra và bãi bỏ những tệp theo tên đặt cho tệp.

### 4. Sự truyền tin (communication)

Trong một số trường hợp, một tiến trình trao đổi tin với tiến trình khác. Một sự trao đổi tin như vậy có thể sinh ra giữa các tiến trình của cùng một máy tính hay trên hệ trong mạng truyền thông. Tin trao đổi có thể được HĐH ghi nhớ và trao đổi theo thời gian được phân chia hay qua trung gian của kỹ thuật trao đổi thông báo giữa các tiến trình.

### 5. Phát hiện lỗi

Hệ điều hành không bao giờ không biết các lỗi bất chợt. Các lỗi này có thể có trong phần cứng của bộ xử lý, của khối nhớ (lỗi của khối nhớ hay hỏng nguồn điện), trong các thiết bị vào/ra (sai số chẵn/lẻ trên một băng), hỏng tiếp xúc trong một mạng, không có giấy trong máy in, trong chương trình của người dùng (tràn số học, muốn truy cập một vùng nhớ không hợp lệ hay dùng một chương trình mất quá nhiều thời gian của bộ xử lý). Đối với mỗi loại lỗi, HĐH phải có hành động thích hợp nhằm đảm bảo tính đúng đắn và gắn bó. Nếu có thể, HĐH còn thông báo mã lỗi ra màn hình cho người sử dụng biết.

### 6. Sắp đặt các tài nguyên (Resource allocation)

Khi có nhiều người dùng hay làm việc cùng một thời điểm, phải sắp đặt những tài nguyên cho mỗi người. Hệ điều hành điều khiển nhiều loại tài nguyên khác nhau. Một số (như các chu trình bộ xử lý, nhớ chính và tích lũy các tệp) có thể có mã sắp đặt riêng, trong khi mà những thiết bị khác (như các thiết bị ngoài vào/ra) có thể có mã yêu cầu và giải phóng. Ví dụ, để xác định tốt hơn việc dùng khối xử lý, HĐH có chương trình con điều phối (scheduling) bộ xử lý, kể đến vận tốc của xử lý, công việc cần làm, số lượng thanh ghi có thể và các sự kiện khác.

### 7. Sự tương thích

Chúng ta muốn biết hiệu suất, người sử dụng nào dùng bao nhiêu và loại tài nguyên tin học nào. Nội dung trên có thể được đề cập vì lý do của sự tương thích (loại nào mà người dùng có thể nhận) hay đơn giản là tích lũy sự thống kê của sử dụng. Điều này có thể tạo nên một công cụ có giá trị lớn đối với người nghiên cứu muốn cấu tạo lại hệ nhằm cải tiến những phục vụ tin học.

### 8. Bảo vệ

Người sở hữu những tin chứa trong một hệ tin nhiều người sử dụng có thể mong muốn kiểm soát việc sử dụng hệ. Khi có nhiều tiến trình cùng thực hiện, thì một tiến trình không thể làm nhiễu các tiến trình khác và cả chính HĐH. Việc bảo vệ đảm bảo là các truy cập tới các tài nguyên của hệ phải được kiểm soát. An toàn (security) của hệ đối với các người dùng bên ngoài cũng quan trọng. Do đó, phải có mật khẩu cho người được truy nhập tài nguyên. HĐH sẽ ngăn cản các thiết bị vào/ra bên ngoài, bao gồm cả modem và bìa mạng khi truy cập không cho phép. HĐH cũng ghi tất cả những sự tiếp xúc để phát hiện những xâm nhập vào hệ không cho phép. Phải tạo ra các dự phòng ở tất cả các mức trong một hệ nếu người ta muốn bảo vệ chắc chắn HĐH, cơ sở dữ liệu, cũng như các chương trình của các người sử dụng khác.

## 2.1.3. CÁC LỜI GỌI HỆ THỐNG

Lời gọi hệ thống cung cấp một giao diện giữa một tiến trình và HĐH. Các lời gọi hệ thống này thường trình bày dưới dạng lệnh của hợp ngữ (assembly) và chúng thường được liệt kê trong sổ tay sử dụng bởi người lập trình bằng hợp ngữ (các lệnh INT nh).

Một số hệ cho phép lời gọi hệ thống tác động bắt đầu từ một chương trình viết ở ngôn ngữ mức cao hơn, trong đó **lời gọi** thường giống như cách **gọi chức năng xác định trước** hoặc **chương trình con.** Hệ có thể phát một lời gọi tới một chương trình thực hiện riêng, thực hiện lời gọi hệ thống hay lời gọi này có thể được phát trực tiếp ở trong dòng (in-line).

Có ba phương pháp chung để *chuyển giao các tham số cho hệ điều hành* là **thanh ghi, bảng trong bộ nhớ và bảng trong ngăn xếp.**

Có năm loại chính của lời gọi là *điều khiển tiến trình, thao tác tệp, điều khiển thiết bị ngoài, gửi tin và truyền thông.*

### 1. Điều khiển tiến trình

HĐH điều khiển các tiến trình sau:

- Dừng bình thường, thực hiện không bình thường.
- Nạp, thực hiện chương trình.
- Tạo và kết thúc tiến trình.
- Có xác định thuộc tính tiến trình.
- Chờ một thời gian.
- Chờ, chỉ báo các sự kiện.
- Chiếm, giải phóng bộ nhớ.

### 2. Thao tác tệp

HĐH thao tác tệp theo các hành động sau:

- Tạo ra, bãi bỏ tệp.
- Mở đóng tệp.
- Đọc ghi, chuyển chỗ.
- Thu, xác định thuộc tính tệp.

### 3. Điều khiển thiết bị ngoài

HĐH điều khiển thiết bị ngoài theo các động tác cơ bản sau:

- Yêu cầu, giải phóng thiết bị ngoài.
- Thu, xác định thuộc tính thiết bị ngoài.
- Móc nối, ngắt một cách logic với thiết bị ngoài.

### 4. Gửi tin

HĐH gửi các tin sau:

- Thu, xác định giờ hay ngày.
- Thu, xác định số liệu của hệ.
- Thu những thuộc tính của tiến trình, tệp và thiết bị ngoài.
- Xác định những thuộc tính của tiến trình, tệp và thiết bị ngoài.

### 5. Truyền thông

HĐH truyền các thông tin với các động tác cơ bản sau:

- Tạo ra, bãi bỏ những tiếp xúc liên lạc.
- Gửi, nhận tin.
- Chuyển tin về trạng thái.
- Móc nối, ngắt những thiết bị ngoài ở xa.

## 2.1.4. CÁC CHƯƠNG TRÌNH HỆ THỐNG

Tập hợp những chương trình hệ thống là một dạng khác của một HĐH hiện tại, ở mức thấp là phần cứng. Sau đó, hệ điều hành, rồi các chương trình hệ thống và cuối cùng là những chương trình ứng dụng.

Những chương trình hệ thống cung cấp một môi trường thực tế hơn cho sự phát triển và thực hiện chương trình. Một số trong chúng, đơn giản là các giao diện của người dùng để cho các lời gọi hệ, trong khi một số khác là rất phức tạp.

Người ta có thể chia chương trình hệ thống theo các hành động sau:

### 1. Thao tác tệp

Những chương trình này tạo, bãi bỏ, chép, đặt lại tên, in, hiện, thống kê, thao tác tệp và thư mục.

### 2. Soạn thảo tệp

Người ta có thể chia nhiều chương trình soạn thảo (editor) để tạo ra và thay đổi nội dung các tệp được lưu giữ trong đĩa hay băng từ.

### 3. Thông tin về trạng thái

Một số chương trình hỏi đơn giản để biết ngày, giờ lượng bộ nhớ cho phép hay không gian của đĩa, số người dùng hay tin tức tương tự. Tin được tạo dạng và in trên một thiết bị đầu cuối (terminal), trên một thiết bị ngoài khác, trên tệp tin đưa ra.

### 4. Hỗ trợ của các ngôn ngữ lập trình

Những chương trình biên dịch (compilator), chương trình hợp dịch (assembler) và phiên dịch (interpreter) cho các ngôn ngữ lập trình hiện hành (Fortran, Cobol, Pascal, Basic, C, $C^{++}$ và LISP) thường được cung cấp cùng với HĐH. Hiện nay phần lớn các chương trình được bán và cung cấp riêng rẽ.

### 5. Nạp và thực hiện chương trình

Mỗi lần chương trình được hợp dịch hay biên dịch, nó phải được nạp trong bộ nhớ để thực hiện. Hệ phải được cung cấp các bộ nạp (Loader) tuyệt đối, các bộ nạp dịch chuyển, các soạn thảo (editor), kết nối (Linker) và các bộ nạp chồng chất (phủ nhau). Cũng phải có những hệ gỡ rối (debugger) cho các ngôn ngữ bậc cao hơn hay cho ngôn ngữ máy.

### 6. Truyền thông

Những chương trình này cung cấp một cơ chế cho phép tạo ra các tiếp xúc (connection) ảo giữa các tiến trình, các người dùng và các hệ tin khác nhau. Chúng cho phép người dùng gửi thông báo rất dài qua báo điện tử hay chuyển các tệp từ máy này qua máy khác và ngay cả dùng một máy ở xa như là các máy đặt ở gần (tiếp xúc ở xa).

### 7. Các chương trình áp dụng

Phần lớn hệ điều hành được trao cho những chương trình tiện ích để giải những bài toán hiện hành. Những chương trình này bao gồm chương trình biên dịch, xử lý văn bản, phần mềm đồ họa, điều khiển cơ sở dữ liệu, những bảng, phần mềm thống kê và những trò chơi.

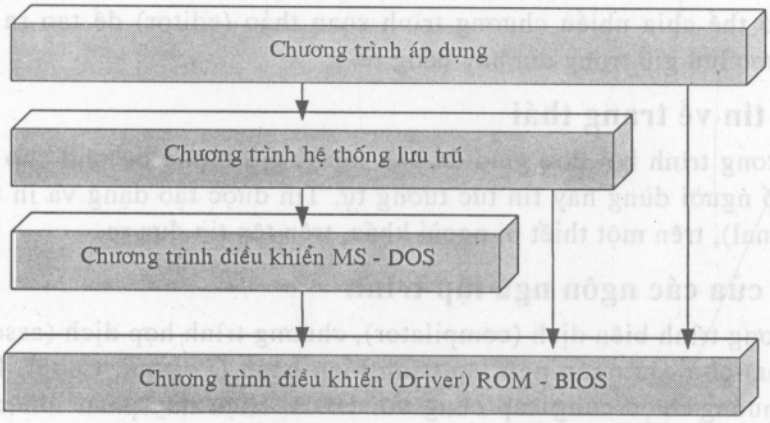
Chương trình hệ thống quan trọng nhất cho một HĐH có thể là chương trình phiên dịch lệnh (command interpreter) mà chức năng chủ yếu là nhận và thực hiện lệnh, được xác định bởi người dùng.

## 2.1.5. CẤU TRÚC CỦA HỆ ĐIỀU HÀNH

Một hệ lớn và phức tạp như hệ điều hành phải được thiết kế một cách cẩn thận để làm việc đúng và dễ dàng thay đổi. Nó phải có cấu trúc gồm nhiều thành phần nhỏ, được chia theo nhiệm vụ hơn là một hệ đơn khối. Mỗi một phần nhỏ trong các khối (môđun) phải gồm một phần xác định rõ của hệ với những lối vào, những lối ra và các chức năng được xác định cẩn thận. Ở mục 2.2 chúng ta sẽ đề cập ngắn gọn tới các thành phần hiện hành của HĐH, nên ở mục này chúng ta nghiên cứu các thành phần này được liên kết và trùng nhau như thế nào.

### 1. Cấu trúc đơn giản

Hệ điều hành MS-DOS đã phát triển từ đơn giản, ngắn tới phức tạp và lớn dần. Cấu trúc hiện nay của nó như hình 2.1, không chia cẩn thận theo khối (môđun) mà theo lớp (layer).




**HÌNH 2.1.**Cấu trúc theo lớp của MS-DOS.

Ngay cả khi MS-DOS có một cấu trúc nào đó, các giao diện của chúng và các mức của chức năng không được phân chia rõ ràng. Ví dụ, các chương trình áp dụng có khả năng truy cập tới chương trình con vào/ra cơ sở nhằm viết trực tiếp lên màn hình và hiện lên đĩa. Sự thoải mái (tự do) như vậy làm cho MS-DOS có bộ mặt dễ bị tổn thương ở các chương trình bị lỗi, điều này gây ra sự che chắn toàn bộ hệ hay xóa đĩa khi những chương trình của người dùng không chạy, MS-DOS chắc chắn bị giới hạn bởi phần cứng mà nó thực hiện. Vi xử lý Intel 8088 đã được viết không cung cấp chế độ kép hay bảo vệ phần cứng, những nhà thiết kế của MS-DOS đã không chọn điều trên và họ đã phải để lại sự truy nhập phần cứng cơ sở.

Hệ điều hành UNIX gốc là một ví dụ khác của cấu trúc bị hạn chế. Đầu tiên, nó cũng bị hạn chế bởi chức năng phần cứng. Nó gồm hai phần được tách biệt: lõi và các chương trình hệ thống. Hơn nữa, lõi được chia thành một loạt các giao diện và chương trình điều khiển thiết bị ngoài (driver) đã được thêm vào và mở rộng dần dần qua nhiều năm để có UNIX phát triển như hiện nay. Hình 2.2 giới thiệu cấu trúc của UNIX theo các lớp.

Các lời gọi hệ thống xác định giao diện của người lập trình của UNIX, tập hợp của các chương trình hệ thống có thể móc nối, xác định giao diện của người sử dụng. Các giao diện của người lập trình và người sử dụng xác định bối cảnh mà lõi phải chịu đựng. Người ta đã phát triển nhiều phiên bản UNIX trong đó lõi bị cắt tùy theo những giới hạn quá chức năng, đó là HĐH AIX, phiên bản UNIX của IBM chia lõi làm hai phần. Match (của đại học Carnagerie Mellon) rút gọn lõi thành một tập hợp nhỏ, các chức năng cơ sở và nó di chuyển tất cả cái gì không phải là phần chủ yếu tới các chương trình hệ thống và cả tới các chương trình của mức người sử dụng. Hệ trên còn lại một HĐH dựa trên cơ sở một lõi cực nhỏ (micro noeud) chỉ chứa một tập hợp nhỏ những nguyên thủy (primitives) cần thiết.

<table>
<tr><td colspan="3">Những chương trình của người sử dụng</td></tr>
<tr><td colspan="3">Shells và các lệnh<br>Biên dịch và thông dịch<br>Các thư viện của hệ</td></tr>
<tr><td colspan="3">Giao diện của các lời gọi hệ thống cho lõi</td></tr>
<tr><td>Các tín hiệu điều khiển thiết bị đầu cuối.<br>Hệ vào/ra của các ký tự chương trình điều khiển thiết bị ngoài.</td><td>Hệ các tệp trao đổi (swapping).<br>Hệ vào/ra của khối chương trình điều khiển đĩa và băng.</td><td>Điều phối xử lý (CPU) thay thế trang, đánh số trang theo yêu cầu nhớ ảo.</td></tr>
<tr><td colspan="3">Giao diện lõi cho phần cứng</td></tr>
<tr><td>Các bộ điều khiển thiết bị đầu cuối.<br>Thiết bị đầu cuối.</td><td>Các khối điều khiển thiết bị ngoài.<br>Đĩa và băng.</td><td>Khối điều khiển bộ nhớ, bộ nhớ vật lý.</td></tr>
</table>

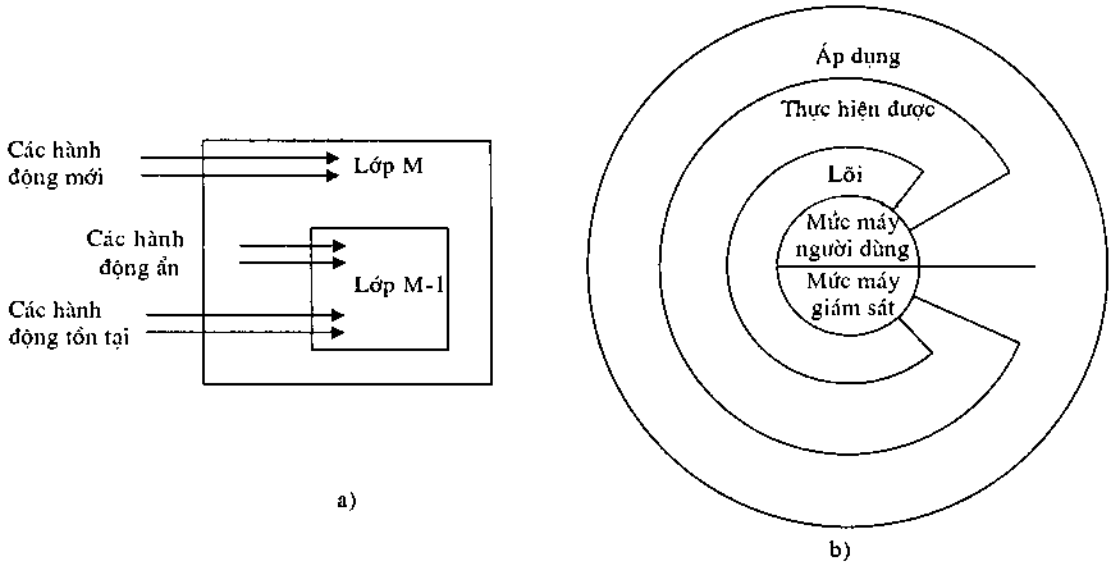
**HÌNH 2.2.** Cấu trúc của hệ UNIX.

## 2. Vùng ven theo lớp (layer)

Các phiên bản mới này của UNIX được thiết kế để dùng với phần cứng phát triển hơn. Với một hỗ trợ phần cứng thích hợp, HĐH có thể được chia thành các phần nhỏ hơn và thích hợp hơn so với MS-DOS gốc và UNIX gốc. Như vậy, HĐH có thể duy trì sự điều khiển máy tính tốt hơn và những áp dụng trên máy tính đó. Các cá nhân có trách nhiệm thực hiện việc can thiệp tự do hơn để tạo nên những thay đổi trong những công việc bên trong của hệ. Người ta dùng các kỹ thuật đã từng biết để tạo nên những hệ điều hành theo khối (môđun). Với gần đúng giảm dần, những chức năng và những đặc tính toàn cục có thể được xác định và chia thành các thành phần. Sự che chắn tin cũng trở nên quan trọng, nó để cho những người lập trình tự do xâm nhập những chương trình ở mức thấp như là sự thuận lợi mà giao diện ngoài của chương trình hầu như không thay đổi và chính chương trình thực hiện nhiệm vụ đã công bố.

Việc khối hóa một hệ có thể thực hiện bằng nhiều cách, thông dụng nhất là vùng ven theo lớp. Theo cách này, hệ điều hành được chia thành một số lớp (mức), mỗi lớp được xây dựng ở trên lớp dưới. Lớp dưới (lớp 0) là phần cứng, lớp cao (lớp N) là giao diện người sử dụng (hình 2.3a, b).

Lần đầu tiên, vùng ven theo lớp được dùng trong hệ điều hành THE như bảng 2.1. Hệ Venus có các lớp thấp (0 đến 4) nó gồm hệ điều phối (scheduling) của khối xử lý và điều khiển bộ nhớ, đã được vi chương trình hóa (microprogrammed). Quyết định này có điều lợi là tốc độ thực hiện nhanh nhất và một giao diện xác định rõ ràng giữa các lớp vi chương trình hóa và các lớp cao.




**HÌNH 2.3.** a - một lớp của hệ điều hành; b - cấu trúc hệ điều hành theo lớp vỏ củ hành.

***Bảng 2.1.*** **Cấu trúc theo lớp của các hệ The và Venus**

| Các lớp | Tên lớp của The | Tên lớp của Venus |
|---|---|---|
| 6 | | Chương trình của người sử dụng |
| 5 | Các chương trình của người sử dụng | Các chương trình điều khiển và điều phối cho thiết bị ngoài |
| 4 | Sự đệm hóa cho thiết bị ngoài vào/ra | Nhớ ảo |
| 3 | Các chương trình điều khiển, bàn điều khiển của người điều hành | Kênh vào/ra |
| 2 | Điều khiển nhớ | Điều phối của khối xử lý |
| 1 | Điều phối của khối xử lý | Phiên dịch lệnh |
| 0 | Phần cứng | Phần cứng |

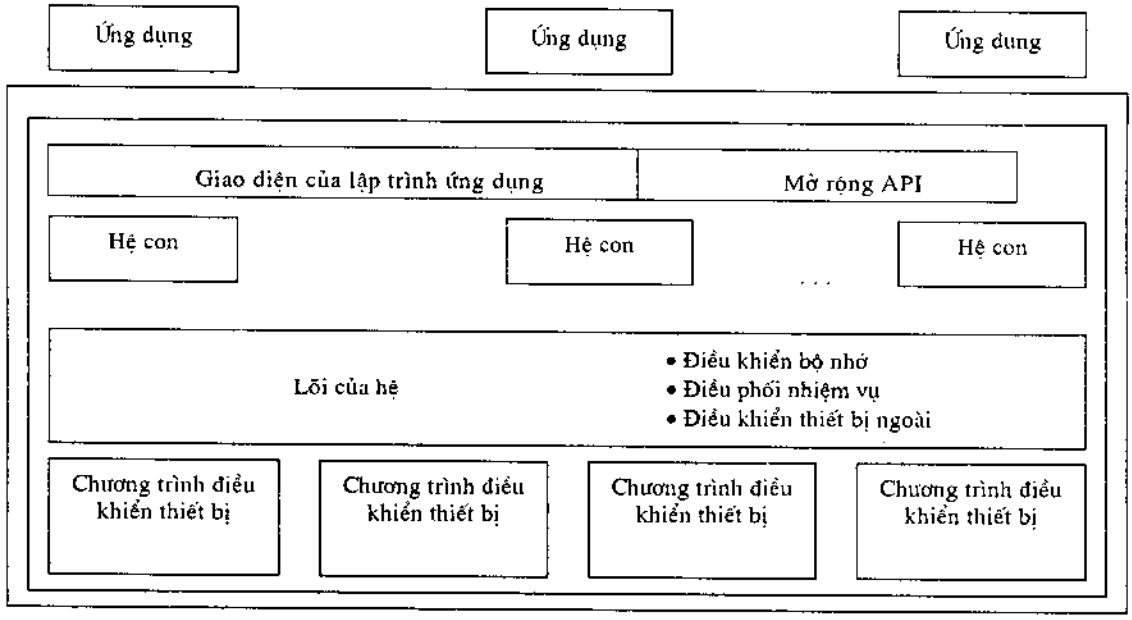
Việc chia lớp của HĐH cũng có khó khăn, như:

- Xác định những lớp khác nhau theo phương thức thích hợp.

- Trong hệ lớn, bộ điều phối của bộ xử lý có quá nhiều tin về mọi hành động của nó và phải đưa vào bộ nhớ, do đó cần phải đưa chương trình con điều khiển bộ nhớ phụ ở dưới bộ điều phối của bộ xử lý.

- Làm giảm hiệu quả của HĐH so với các loại cấu trúc khác.

Những năm gần đây, người ta thiết kế HĐH với các lớp ít hơn các chức năng. Hệ OS/2 là thế hệ sau của MS-DOS. OS/2 thêm khối đa nhiệm và chức năng ở chế độ nhân đôi, cũng như các đặc tính mới (hình 2.4). OS/2 đã được thiết kế, không cho phép người sử dụng truy cập trực tiếp một cách dễ dàng tới các chương trình ở mức thấp. Điều đó làm cho HĐH kiểm soát được phần cứng và một kiến thức nhận dạng các nguồn tài nguyên mà mỗi chương trình của người sử dụng đang được dùng.




**HÌNH 2.4.** Cấu trúc theo lớp của OS/2.

## 2.1.6. CÁC CHẾ ĐỘ CỦA HỆ ĐIỀU HÀNH

Hệ điều hành nói riêng, máy tính điện tử nói chung, có thể hoạt động theo các chế độ khác nhau. Tùy theo quan điểm ta có cách phân biệt các chế độ khác nhau.

### 1. Theo quan điểm chương trình

- *Chế độ đơn chương trình* (monoprogram): trong khoảng thời gian xác định, HĐH điều khiển máy thực hiện chỉ một chương trình duy nhất. Nói khác đi, các chương trình khác nhau thực hiện từng chương trình nối tiếp, hết chương trình nọ tới chương trình kia.

- *Chế độ đa chương trình* (multiprogram): nhiều chương trình có mặt trong bộ nhớ được thực hiện đồng thời, các lệnh của các chương trình được thực hiện lần lượt nối tiếp nhau trong thời gian xác định, phân chia cho mỗi chương trình (phân chia thời gian).

### 2. Theo quan điểm thời gian

- *Thời gian được phân chia* (time-sharing): một số lệnh của một chương trình được thực hiện trong thời gian đã chia cho mình.

- *Thời gian thực* (real time): chương trình thực hiện theo yêu cầu từ bên ngoài (từ các hệ đo - điều khiển, từ hệ yêu cầu từ xa như đặt hàng, bán hàng, thanh toán...).

### 3. Theo quan điểm xử lý tin

- *Xử lý theo lô* (batch processing): các công việc được máy thực hiện liên tiếp thành chuỗi, hết hành động nọ đến hành động kia một cách nối tiếp tới khi kết thúc các hành động đã liệt kê trong danh sách lệnh. Ví dụ: một chuỗi (lô) lệnh của DOS trong tệp Autoexe.bat.

- *Xử lý song song* : bởi nhiều vi xử lý (multiprocessing), mỗi hành động được thực hiện song song trong nhiều vi xử lý trong một máy tính. Ví dụ, vào/ra số liệu do vi xử lý vào ra (Intel 8089), chương trình do vi xử lý trung tâm (Intel 80x86), xử lý toán học do đồng xử lý (Intel 80x87).

- *Xử lý tập trung* : mọi xử lý do một trung tâm xử lý hay một máy vi tính thực hiện.

- *Xử lý phân tán* : mọi xử lý được phân chia cho nhiều bộ phận, nhiều máy tính nằm rải rác được kết nối thành mạng. Mỗi xử lý có thể sử dụng mọi nguồn tài nguyên (thiết bị, chương trình) của mạng.

Tùy chế độ ta có các HĐH khác nhau với cấu trúc, thành phần, cách thức hoạt động khác nhau.

## 2.2. CÁC HỆ ĐIỀU HÀNH CỦA MÁY VI TÍNH

### 2.2.1. CÁC HỆ ĐIỀU HÀNH

Ngay từ giữa những năm 1960 cùng với sự phát triển của phần cứng, hệ điều hành cũng phát triển mạnh mẽ. Chúng ta có thể kể:

- *Hệ chương trình điều khiển (monitor):* cho các hệ vi xử lý đơn giản (Kit), gồm một chương trình giám sát (supervisor) và một số chương trình điều khiển thiết bị ngoài, điều khiển bộ nhớ, dịch lệnh, biên soạn lệnh v.v...

- *Hệ điều hành CP/M (1974):* do Gary Kildall viết cho vi xử lý Intel 8080 trên đĩa mềm của IBM (1970), công cụ hội thoại Teletype ASR33, các tệp chương trình và số liệu chứa trên băng giấy đục lỗ. Từ 1976 trở đi, nó trở thành HĐH cho các máy có VXL 8080, 8085 và 8086, Z80. Đó là hệ đơn chương (đơn nhiệm) chỉ chạy một chương trình, một nhiệm vụ duy nhất.

- *Hệ điều hành CP/M hay CP/M86 (1984):* là sự cải tiến của CP/M nhưng cho các vi xử lý 16 bit (8086/8088, 68000 và Z8000) ở chế độ đa chương (16 chương trình) hay đa nhiệm (multiprogramming). Các chương trình HĐH này chiếm 128KB với hợp ngữ 8086 (đối với máy 68000 và Z8000 dùng C, chứa các chương trình vẽ đồ thị, quản lý tệp, quản lý cơ sở dữ liệu Dbase II, HĐH này được sử dụng trong các máy vi tính như IBM PC, Sirivirus, Altos, Olivetti, Hewlett-Parkard.

- *Hệ điều hành MS-DOS (1978):* do hãng Microsoft viết trên cơ sở CP/M 86 cho các mạng máy tính PC của IBM. Năm 1982, cải tiến làm việc đơn chương dùng cho các bộ vi xử lý 8086, 8088, 80186, 80286 chiếm 128KB với hợp ngữ 8086. Quản lý bộ nhớ theo các phân hoạch. Hệ có khả năng vẽ đồ thị, xử lý văn bản, quản lý tệp, quản lý cơ sở dữ liệu (Dbase II) tương tự CP/M 86. Các máy vi tính IBM PC, Zenith, Altos, Hitachi, DEC, Wang, Northstars, NEC.

Hiện nay, hệ MS-DOS có phiên bản 6.22, DOS 7 có các đặc tính gần giống hệ điều hành UNIX, có cấu trúc nhiều nhiệm vụ theo cửa sổ, một máy phát điều khiển cửa sổ (MS-WIN) và điều khiển mạng (MS-NET và LAN Manager).

- *Hệ điều hành UNIX*: bắt đầu xây dựng từ năm 1969 do P. Fautrier rồi sau đó do hãng Bell xây dựng năm 1974, làm việc nhiều chương trình (tới 25), chiếm nhiều bộ nhớ (100 ÷ 256KB) với ngôn ngữ C. Trên máy PDP/11 đã có tới ba phiên bản khác nhau. HĐH này làm việc trên các bộ vi xử lý PDP11, 68000, Z8000 và trong các máy vi tính Onyx, Plexus, Zilog, Altos, Lisan, Pixel, Dual. UNIX có các biến thể khác nhau do xây dựng trên cơ sở của UNIX cũ như MINIX, BSD (Unix của Berkeley), XENIX (chuẩn UNIX thương mại kết hợp với vi xử lý Intel do Microsoft viết lại), UNIX system V (gần giống Xenix). Ưu điểm chính của UNIX là:

+ Viết bằng ngôn ngữ C nên dễ đọc và thêm chức năng so với hệ điều hành khác viết bằng hợp ngữ.

+ Là hệ điều hành mạng, điều khiển tốt hơn MS-DOS.

- *Hệ điều hành Prologue* : do Pháp xây dựng (1979), làm việc đa chương (tới 32 chương trình) nhưng hơi chậm và đòi hỏi nhiều bộ nhớ (64KB tới 256KB), khả năng gần giống CP/M86, hợp ngữ 8086. Hệ quản lý cơ sở dữ liệu là hệ Dialogue. Hệ dùng cho nhiều máy vi tính của Pháp (Micral 90/50, Groupil 3, Victor) và một số máy khác (IBM PC, Canon).

- *Hệ điều hành OS/2* : hệ do hãng IBM viết (1987) cho các máy vi tính PS/2, trên cơ sở vi xử lý 8086 (XT), 80286 (AT) và cả 80386 (super AT). Đó là hệ một vị trí (monoposte), nhiều nhiệm vụ (đa chương trình), có điều khiển mạng cục bộ (ECF) và một lõi của quản lý dữ liệu SGBD (BD/DC).

OS/2 có các phiên bản sau:

+ *Version 1.0 (1988)* : tương đương DOS 3.3 đa nhiệm, cho phép địa chỉ tới 16MB, hoạt động trên máy 80286 và 80386 ở một chế độ thực và một số chế độ bảo vệ nên có khó khăn khi chuyển chế độ. Giao diện người dùng cho OS/2 1.0 là một dòng lệnh điển hình tương đương với DOS.

+ *Version 1.1 (1989)* : đưa vào một giao diện người dùng đồ họa (GUI) được gọi là Presentation Manager (quản lý trình diễn) hay PM. PM tương tự với giao diện trên cơ sở màn hình như của máy vi tính Apple Macintosh (thực hiện lệnh bằng di chuyển con trỏ tới lệnh mong muốn của thực đơn hay icon - biểu tượng đặc trưng cho lệnh rồi nháy chuột). PM cũng cho phép có nhiều cửa sổ mở trên màn hình và có thể "cắt" hay "dán" ở trong một hay nhiều cửa sổ. PM còn điều khiển tệp, cho phép ta biểu thị thư mục theo cây và chọn bởi nháy chuột ở tên tệp thích hợp.

+ *Version 1.2* : vẫn giữ PM và thêm HPFS (High Performance File System). Thay cho FAT dùng bởi hệ tệp của DOS, HPFS dùng một hệ khác FAT, cho phép truy nhập tệp nhanh hơn và tên tệp lớn hơn 8 ký tự và khởi phát một khối "thuộc tính mở rộng" cho mỗi tệp.

+ *Version 1.3 (1990)* : dùng cho 80486 cài đặt vào cấu trúc thống nhất với ứng dụng SAA (Systems Application Architecture). Cho phép làm việc với 16 MB nhớ, thâm nhập đồng thời nhiều thủ tục liên lạc khác nhau (SGBG tương quan, hệ quản lý cửa sổ, SDLC, X25, không đồng bộ, mạng v.v...).

+ *Version 2.0 (1991)* : tương đương Windows 3.0, DOS Version 5.0. Thiết kế chỉ cho hệ 80386 và 80486, dùng cho chế độ 8086 ảo. Có thể dùng đa nhiệm trộn lẫn với DOS, OS/2 Version 2.0 có 3 giao diện chương trình áp dụng (Application Program Interfaces - API), hoặc các bộ chức năng (ví dụ INT 21h, các chức năng 16 bits tương thích với các chương trình áp dụng OS/2 1.X và các chức năng 32 bits dùng cho các áp dụng OS/2 2.0).

+ *Version 2.1 (1993)* : cho vi xử lý Pentium, tương đương với DOS Version 6.0 và Windows NT.

- *Hệ điều hành Windows* : Windows 3.0 của hãng Microsoft là một cầu nối, giá tương đối rẻ giữa hệ DOS và hệ điều hành mạnh OS/2 2.0 đã xét. Như tên gọi, hệ điều hành Windows dùng một giao diện đồ họa (GUI) rất giống Presentation Manager (PM). Windows 3.0 là một DOS mở rộng rất linh hoạt, có thể có lợi cho các hành động ở chế độ bảo vệ của các vi xử lý 80286, 80386 và 80486. Nó hành động ở bất kỳ một trong ba chế độ khác nhau, phụ thuộc vào vi xử lý và bộ nhớ có trong hệ. Với hệ 8086/8088, Windows 3.0 phải hành động ở chế độ thực. Ở hệ có 80286 với ít nhất 1MB nhớ mở rộng, Windows 3.0 có thể chạy ở chế độ chuẩn, có lợi cho các hành động bảo vệ của 80286. Ở chế độ thực và chuẩn, chỉ một loại nhiệm vụ của DOS có thể thực hiện ở một thời điểm.

Một hệ trên cơ sở 80386 hoặc 80486 với ít nhất 2MB nhớ mở rộng, Windows có thể được chạy ở chế độ cải tiến của 80386. Ở chế độ này, nó dùng chế độ 8086 ảo của 80386 để chạy nhiều nhiệm vụ của 8086 và đặt nhớ ảo theo trang, có thể chạy chương trình đòi hỏi nhiều địa chỉ nhớ hơn địa chỉ có ở trong hệ.

Khi chạy Windows 3.0, cửa sổ quản lý chương trình xuất hiện trên màn hình với các tên và các biểu tượng (icon) cho các chương trình. Gọi chương trình nào thì chỉ cần di chuyển con trỏ tới chỗ đó và nháy chuột. Windows 3.0 gửi một danh sách nhiệm vụ của nhiệm vụ đang chạy hiện hành. Ta có thể nối mạch tới một nhiệm vụ từ nền tới đỉnh bằng cách di chuyển danh sách nhiệm vụ và nháy chuột lên nhiệm vụ mong muốn.

Một biểu tượng của chương trình trong cửa sổ PM là một chương trình quản lý tệp (File Manager). Khi nháy chuột hai lần lên một biểu tượng, một cửa sổ của tệp được mở và chỉ cho ta cây của tệp và thư mục con (subdirectories) trong thư mục (directory) hiện hành. Trong cửa sổ này ta có thể dùng chuột để thực hiện những hành động tệp thông thường như chép, bỏ, đặt lại tên... thay cho việc gõ bàn phím tên lệnh của DOS. Windows 3.0 cũng có hệ hỗ trợ hay trợ giúp (help system) trên màn hình để gọi ra khi cần chỉ dẫn.

**Windows 3.0** chứa các chương trình, chia làm hai loại có các áp dụng cửa sổ và không cửa sổ. Để viết một chương trình áp dụng Windows 3.0 đơn giản, chúng ta có thể dùng Asymetrix Corp Toolbook đi kèm với Windows 3.0.

**Windows 3.1** (1992) dùng cho máy PS/2 Model 76, 77, 56, 57 và 95, máy Thinkpad Model 700.

**Windows NT** (1993) dùng cho vi xử lý Pentium.

**Windows 95, Windows 98** là những phiên bản mới của Windows, hệ điều hành phổ biến nhất. Nó có ba đặc điểm chú ý sau :

+ Màn hình có độ phân giải cao, có nhiều biểu tượng và tên lệnh theo cây để thi hành lệnh khi di chuyển con trỏ tới và nháy chuột.

+ Có danh mục các lệnh trên menu (thực đơn).

+ Cho phép thực hiện nhiều chương trình tuần tự (đa chương). Công dụng này đặc biệt tiện ích khi muốn chuyển dữ liệu từ một chương trình này sang một chương trình khác.

Hiện nay, phiên bản mới nhất của Windows là Windows 98 có nhiều cải tiến so với Windows 97, nhất là hành động điều khiển mạng.

- *Hệ điều hành Macintosh* : HĐH này chính là version 7 của máy Apple là hệ điều hành cho máy Macintosh, xuất hiện cùng thời với Windows/NT của hãng Microsoft. Máy Apple

của Macintosh là máy vi tính có hai vi xử lý Intel 80486 và Motorola MC 68000, có thể làm việc độc lập từng hệ riêng (theo MS-DOS cho Intel và theo Macintosh cho Motorola) hoặc chế độ tương tác (chuyển từ hệ nọ sang hệ kia).

Ngoài các hệ điều hành thông dụng trên còn có các hệ điều hành đặc biệt khác như:

- *Hệ điều hành theo thời gian phân chia CTSS* : (do Strachey đề nghị từ 1959) xuất hiện năm 1962 và SDC Q-32 bởi hãng System Development (Schwatch và các người khác, năm 1964).

- *Hệ điều hành cho nhiều vi xử lý* (1990) : do Tabak đề xuất.

- *Hệ điều hành cho hệ tin phân tán* (1985): do Tanenbaun và Van Renesse đề nghị.

- *Hệ điều hành cho mạng thông tin*: do Stakovic và Ramanrithan (1989), Zhao (1989), do hãng AT&T (hệ UNIX V) (1986).

## 2.2.2. MONITOR CỦA HỆ VI XỬ LÝ ĐƠN GIẢN

Hệ vi xử lý đơn giản gồm vi mạch vi xử lý, khối nhớ, cửa vào-ra, bàn phím đơn giản và đèn chỉ thị lắp trên một bản mạch gọi là Kit. Hệ làm việc được nhờ một chương trình điều khiển gọi là Monitor. Chương trình Monitor này chính là hệ điều hành của Kit. Vì phần cứng và yêu cầu về hoạt động của Kit cũng đơn giản nên Monitor được viết dưới dạng nhiều chương trình con, được lưu giữ trong bộ nhớ ROM để điều khiển hệ.

### 1. Kit SDK 85 và các nhiệm vụ của Monitor

***a) Nhiệm vụ cơ bản của một Monitor***

Kit đơn giản đòi hỏi hệ điều hành là Monitor để điều khiển mọi hoạt động. Những chương trình hay các động tác tối thiểu cho một hệ Kit là:

- *Chương trình khởi phát*: xóa các thanh ghi của vi xử lý, xóa về 0 nội dung của các thanh ghi đệm số liệu vào - ra và bộ hiển thị về 0.

- *Đọc và dịch lệnh từ bàn phím*: đảm bảo hội thoại giữa người điều hành (operator) và Kit, nhập lệnh của người điều hành từ bàn phím, hiển thị "câu trả lời" của Kit lên bộ đèn hiển thị (display) dạng chữ - số.

- *Nhập, dịch và thực hiện một số lệnh*: (lệnh dạng chữ, số) đặc trưng cho một số hành động cần thiết của Kit do người thiết kế đặt ra như nhập số liệu, nhập lệnh của chương trình, nạp băng giấy, từ, tiến hành thực hiện chương trình, dừng chương trình, xử lý thông số...

- *Thực hiện một số động tác cần thiết*: bởi các chương trình hỗ trợ như xóa bộ hiển thị, đưa số liệu ra hiển thị, xem nội dung bộ nhớ, biến đổi dạng mã số liệu (từ mã 2 - mã 16 và ngược lại ).

- *Tạo một số chương trình con phục vụ ngắt*: với các vector ngắt về một số tình huống bất thường của Kit sẽ xảy ra trong quá trình hoạt động như như nhập số liệu không đúng, địa chỉ nhớ quá giới hạn, quá thời gian định sẵn. Mỗi khi có lỗi, Monitor đều thông báo lên màn hình lỗi của lệnh, để người dùng biết để nhập lại lệnh và sửa lỗi.

Với Kit phát triển (có dung lượng bộ nhớ ROM lớn và đòi hỏi phục vụ cao), còn có hai nhiệm vụ nữa là:

- *Chương trình soạn thảo (Editor)*: nhận dòng lệnh của chương trình dạng hợp ngữ (mã ASC II cho các ký tự, số và dấu), thay đổi ký tự, chèn dòng lệnh vào bộ nhớ.

- *Chương trình thông dịch (interpreter) hay hợp dịch (assembler)*: dịch lệnh dạng ký tự chữ - số sang dạng mã lệnh (mã máy).

Hai chương trình trên chiếm nhiều địa chỉ bộ nhớ (nhiều KB) và viết phức tạp, tốn nhiều công sức và thời gian.

***b) Kit SDK 85 gồm:***

- Vi xử lý Intel 8085A.
- Nhớ ROM 2KB, mở rộng tới 4KB loại 8355/8755.
- Nhớ RAM 256 Byte, mở rộng tới 512 Byte, loại 8155.
- Một lối vào/ra nối với các chân SID; SOD của 8085.
- Bàn phím đơn giản gồm:
  + 8 phím chức năng
  + 16 phím số chỉ từ 0 ÷ 9, A, B, C, D, E, F.
- Bộ đèn hiển thị (thanh ghi sáng) có 8 chiếc.

***c) Monitor của SDK 85***

Trên thực tế có hai Monitor khác nhau trong SDK 85.

- *Monitor điều khiển teletype*: (nối với SID, SOD) được tích trong ROM ở địa chỉ từ 03FAh - 07FFh.

- *Monitor điều khiển Kit* qua bàn phím và đèn hiển thị : được tích ở địa chỉ 004Eh tới 03F9h.

Vùng nhớ 0000h đến 0040h chứa chương trình móc nối với sự lựa chọn Monitor thích hợp. Dưới đây chúng ta chỉ phân tích Monitor bàn phím, đèn hiển thị.

Monitor gồm các chương trình thực hiện các nhiệm vụ sau:

+ Khởi phát
+ Các lệnh và bộ phiên dịch
+ Các hàm tiện ích
+ Các ngắt.

## 2. Chương trình khởi phát

Chương trình khởi phát (bootstap) được tích trong ROM, bắt đầu từ địa chỉ 0000h, để tạo một Reset cứng, xóa con trỏ lệnh (pointer instruction) của 8085 về 0. Chương trình này gồm nhiều giai đoạn.

***a) Chương trình Cold Start (khởi động lạnh)***

Chiếm 8 byte tương ứng với RST0 ở 0000h - 0007h. Chương trình này tiếp tục ở địa chỉ 01F1h - 01FFh, thực hiện các chức năng sau :

- Đọc ký tự điều khiển của chế độ bàn phím - màn hình và thực hiện.
- Xóa các đèn hiển thị về 0 (trắng).
- Xóa thanh ghi điều khiển /trạng thái (cửa C của 8155).

*b) Chương trình Warm Start (khởi động ấm)*

Chiếm địa chỉ 0008h - 0023h tương ứng với RST không sử dụng (RST1, RST2, RST3 và RST4 nhỏ hơn), nó tiếp tục ở địa chỉ 003Fh - 004Dh. Chương trình này có chức năng:

- Lưu trữ các nội dung của những thanh ghi trong vùng ngăn xếp dành cho Monitor (13 byte ở địa chỉ 20E9h - 20F5h).

- Đọc, lưu giữ trạng thái của che ngắt và đường dây ngắt của người dùng (RIM) và gửi chúng trong ngăn xếp trên đây.

- Cấm ngắt trong khi Monitor hoạt động.

- Phát hiện (bởi RIM) nếu một teletype được nối và chọn Monitor thích hợp (nếu được GO về phía Monitor TTY, nếu không tiếp tục Monitor của Kit theo tuần tự).

## 3. Những chương trình và bộ dịch ngôn ngữ

Những chương trình của Monitor chia làm hai nhóm:

- Nhóm các lệnh.

- Nhóm hoàn thành các chức năng cần thiết để thực hiện lệnh gọi là nguyên thủy (primitive) của Monitor.

*a) Những lệnh*

Được nhận biết bởi bộ phiên dịch là các lệnh, gồm:

- *EXAM* : (Examine) cho phép hiển thị và thay đổi nội dung các thanh ghi.

- *GOCMD* : (GO to COMMAND) cho phép thực hiện một chương trình của người dùng.

- *SSTEP* : (Single Step) cho phép thực hiện chương trình ở chế độ từng bước.

- *SUBST* : (Substitute) cho phép thay đổi nội dung ô nhớ RAM.

*b) Các chức năng*

Chứa trong ROM gồm:

- *Clear*: Xóa hiển thị, gửi những ký tự trắng vào vùng địa chỉ và nội dung của bộ hiển thị.

- *CLDIS*: (Clear Display) xóa màn hình theo sau việc kết thúc chương trình.

- *DISPC*: (Display PC) hiển thị giá trị của con trỏ lệnh và byte số liệu.

- *ERR:* (Error) chỉ sai số.

- *GTHEX:* (GET Hex) nhận chuỗi số liệu dạng mã 16 từ bàn phím.

- *HXDSP*: (Hex Display) biến đổi một byte vào thành hai byte để hiển thị.

- *ININT*: (INput INTerrupt Processing) xử lý ngắt gây ra do ấn bàn phím, chờ tin vào bộ đệm.

- *INSGD*: (Insert Digit) chèn một số dạng mã 16 vào thanh ghi DE của vi xử lý.

- *NXTRG*: (Next Register): chuyển sang thanh ghi sau.

- *OUTPT*: (Output) gửi ký tự ra bộ hiển thị.

- *RDKBD*: (Read Keyboard) đọc bàn phím.

- *RETF*: (Return False) xóa bit C (Carry) về 0 và chuyển điều khiển cho chương trình gọi.

- *RETT*: (Return True) xác lập carry lên 1 và chuyển điều khiển cho chương trình gọi.

- *RGLOC*: (Register Location) trả địa chỉ lưu trữ thanh ghi chỉ bởi giá trị hiện hành của con trỏ thanh ghi.

- *RGNAM*: (Register Name) hiển thị trong vùng địa chỉ hiển thị tên của thanh ghi tương ứng với giá trị hiện hành của con trỏ thanh ghi.

- *RSTOR*: (Restore) hồi phục giá trị các thanh ghi thông dụng của vi xử lý.

- *SETRG*: (Set Register Pointer) đọc một ký tự ở bàn phím.

- *UPDAD*: (UPDate ADdress Field) cập nhật vùng địa chỉ của bộ hiển thị bằng cách dùng giá trị hiện hành của byte địa chỉ.

- *UPDDT*: (UPDate Data field) cập nhật vùng số liệu của bộ hiển thị bằng cách dùng giá trị hiện hành của byte số liệu.

### 4. Những ngắt của SDK 85

- *Reset*: gây ra "Cold Start" hay RST0.

- *RST4,5*: (TRAP) gây ra do timer của 8155.

- *RST5,5*: dùng cho chương trình đọc một ký tự của bàn phím, để can thiệp vào vòng đợi của Monitor (xem lại ININT).

- *RST6,5*: dùng để ngắt bởi người dùng bắt đầu từ đường dây ngắt, chương trình xử lý nằm ở RAM bắt đầu ở địa chỉ 20C8h.

- *RST 7,5*: là ngắt "véctơ hóa" kích hoạt bắt đầu từ một bàn phím đặc biệt của bàn phím; chương trình xử lý nằm trong RAM bắt đầu từ địa chỉ 20CEh.

## 2.2.3. HỆ ĐIỀU HÀNH MS-DOS

### 1. Đại cương

#### *a) Đặc điểm*

MS-DOS ra đời trên cơ sở của CP/M (cho 8 bit) để phục vụ cho vi xử lý 16 bit, bắt đầu từ Intel 8088. Ngoài MS-DOS của Microsoft, còn nhiều hệ điều hành cho vi xử lý 16 bit như CP/M-86, Prologue, P-System, PC-Dos của IBM...

CP/M (Control Program for Microcomputer) là hệ điều hành đầu tiên cho vi xử lý 8 bit loại 8085, ghi trên đĩa mềm do Gary Kindall (1974) tìm ra sau khi vi xử lý Intel ra đời (1973). Sau đó CP/M cải tiến cho VXL 8085 (1977) và Z80 (1976).

CP/M chỉ có ba khối và một bộ nạp (Loader) đĩa mềm:

- *CCP:* (Console Command Processor) là bộ dịch các lệnh nhận vào từ bàn phím.

- *BDOS*: (Basic Disk Operating System) chứa hệ điều khiển tệp trên đĩa mềm.

- *BIOS*: (Basic Input-Output System) chứa các chương trình điều khiển (driver) những thiết bị ngoài nối với máy vi tính như màn hình, bàn phím, máy in.

BDOS và BIOS tạo thành FDOS (Functional DISK Operating System).

CP/M+ hay CP/M version 3.0 là sự cải tiến của CP/M theo các điểm sau:

- Tăng kích thước của bộ nhớ trung tâm (64KB).

- Tăng vận tốc của vi xử lý.

- Tăng dung lượng bộ nhớ ngoài và vận tốc ghi (hơn 400KB cho đĩa mềm, hơn 50 MB cho đĩa Winchester).

Ngoài ra còn có các cải tiến sau:

- Cải tiến mối liên hệ giữa hệ và người sử dụng cho thuận tiện hơn, dễ hơn và nhanh hơn.

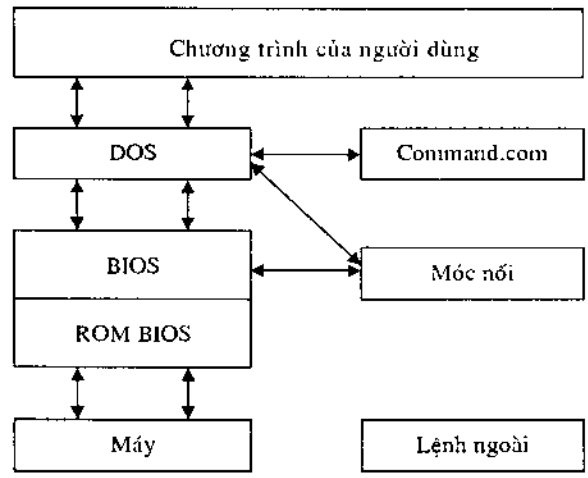
- Cải tiến việc bảo vệ đĩa và tệp bởi mật khẩu (password).

- Tăng cường sự dễ dàng cho người lập trình bằng hợp ngữ.

MS-DOS ra đời năm 1978 do hãng Microsoft viết lại trên cơ sở CP/M 86 (hay CP/M+). Đến 1982, MS-DOS được viết lại có cải tiến cho các vi xử lý từ 8088, 8086 đến 80286. Hệ viết trên hợp ngữ 8086 (tới 128KB). Hệ có khả năng vẽ đồ thị, xử lý văn bản, quản lý tệp, quản lý cơ sở dữ liệu (Dbase II) tương tự CP/M86.

***b) Các thành phần của MS-DOS***

Hệ MS-DOS có sáu thành phần (hình 2.5).




**HÌNH 2.5.** Các thành phần của MS - DOS.

- *ROMBIOS*: là hệ chương trình chứa trong ROM, thực hiện ngay sau khi bật nguồn nuôi của máy vi tính, chứa những chương trình điều khiển các bộ phận của máy (bàn phím, màn hình, bộ nhớ, máy in).

- *Chương trình móc nối* (Boot Sector): nằm trên cung đầu tiên của mỗi đĩa cứng và khởi phát tiến trình nạp vào khối nhớ hai tệp nằm trên đĩa là BIOS và DOS của HĐH MS-DOS.

- *Khối BIOS*: (còn gọi BIOS.COM) với đuôi COM, cung cấp một giao diện ở mức thấp với ROM-BIOS và điều khiển vào/ra của những thiết bị ngoài (bàn phím, màn hình).

- *Khối DOS*: cung cấp một giao diện ở mức cao với các chương trình áp dụng, nó điều khiển thư mục của các tệp cũng như sự ngăn cản ghi lên đĩa. Chính khối này gọi các chức năng DOS bởi sự trung gian của ngắt INT 21h.

- *Khối xử lý COMMAND.COM*: xử lý những lệnh khác nhau mà người sử dụng gõ vào bàn phím để ra lệnh cho HĐH.

- *Những lệnh ngoài (externe)* : đó là những lệnh khác của MS-DOS mà không có trong bộ phiên dịch (interpreter) của lệnh COMMAND.COM hoặc vì các lệnh này ít dùng, hoặc kích thước của lệnh rất lớn chiếm nhiều địa chỉ nhớ.

Để mở rộng, người ta gọi các lệnh ngoài là tất cả các tệp thực hiện được của đĩa (đuôi .EXE hay .COM).

*Ghi chú:*

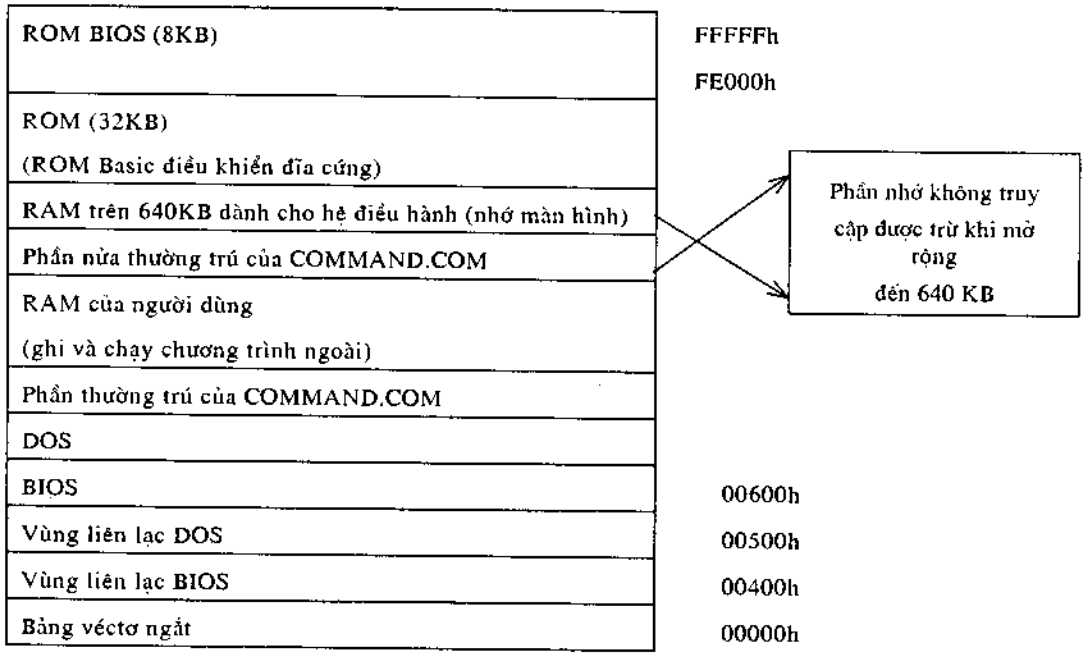
- Khối BIOS là giao diện, mức thấp nên phải viết lại cho mỗi máy vi tính khác nhau.

- Khối DOS, giao diện mức cao của mọi máy và như nhau đối với mọi máy tính. Vậy các giao diện là như nhau (DOS duy nhất), những chương trình ứng dụng như vậy có thể chuyển từ máy này sang máy khác.

*c) Bố trí HĐH trong bộ nhớ trung tâm*

Hệ điều hành được chương trình móc nối (Bootstrap) trên đĩa (cứng, mềm) ở cung đầu tiên, nạp vào bộ nhớ trung tâm (từ đĩa hệ thống mềm hay cứng) theo các vùng của bộ nhớ trung tâm như hình 2.6.

Bảng véctơ ngắt là 256 byte nhớ dùng để chứa các địa chỉ (byte thanh ghi đoạn và byte offset-độ lệch) của tất cả các chương trình con phục vụ ngắt, tức các chương trình phục vụ của BIOS, DOS, COMMAND.COM và ROM.BIOS. Mỗi một chương trình phục vụ ngắt này đảm bảo thực hiện một nhiệm vụ đã xác định rõ ràng của MS-DOS (xem chi tiết ở mục dưới).




**HÌNH 2.6.** Bản đồ nhớ chứa hệ điều hành.

Hai khối DOS và BIOS có hai vùng liên lạc (communication), mỗi vùng có bề rộng 100h = 256D địa chỉ nhớ giống như vétơ ngắt, dùng làm địa chỉ liên lạc hay bắc cầu tới phần cơ bản của DOS và BIOS.

Chương trình xử lý lệnh COMMAND.COM chiếm hai vùng nhớ tách biệt:

- Phần thường trú.

- Phần nửa thường trú.

Như vậy, HĐH chứa ở hai phần:

+ *ROM BIOS* có sẵn trong máy, điều khiển máy khởi động sau khi bật nguồn.

+ *Các vùng còn lại là RAM*, được nạp vào bởi chương trình Bootstrap (móc nối có ở cung đầu tiên của đĩa hệ thống), dùng để nạp phần cơ bản của HĐH (DOS, BIOS và COMMAND.COM).

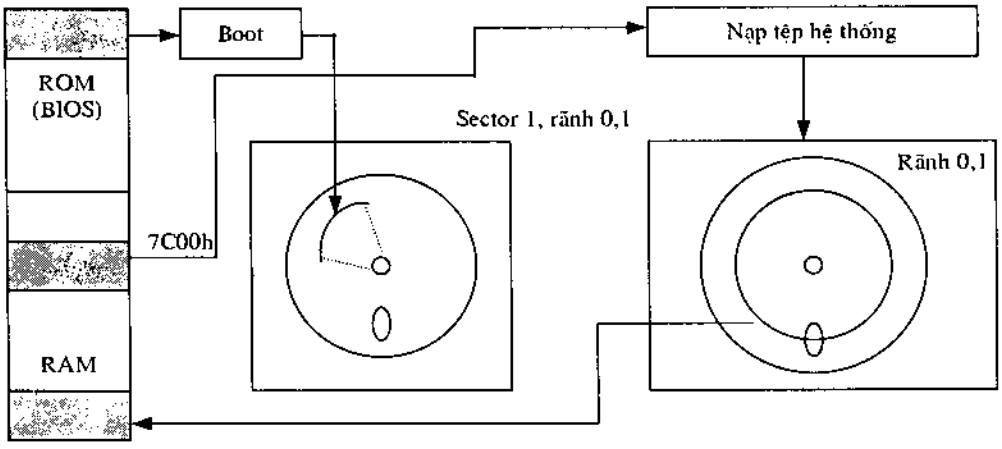
## 2. Chương trình móc nối (Bootstrap Loader)

Nhiệm vụ của chương trình này là nạp các tệp BIOS (hãng IBM gọi là IBM BIO.COM, còn hãng Microsoft gọi là IO.SYS) và DOS (hãng IBM gọi là IBM DOS.COM, hãng Microsoft gọi là DOS.SYS).

Chương trình này nằm ở rãnh 0, cung 1, mặt 0 của đĩa mềm hoặc cung 1, đầu 0, trụ đầu tiên của DOS trong đĩa cứng.

Chương trình này được ROMBIOS nạp từ đĩa vào RAM như hình 2.7 ở địa chỉ 07C00h, sau đó sự điều khiển nạp lại chuyển cho chính chương trình này. Chương trình này nạp vào bộ nhớ RAM gồm các khối:

+ IBMBIO.COM ở mặt 1, rãnh 0, cung 3 tới cung 6.




**HÌNH 2.7.** Các chương trình nạp và thực hiện chương trình móc nối.

+ IBMDOS.COM ở mặt 1, rãnh 0, cung 7 tới 8 (hình 2.7). Với phiên bản (version) khác nhau của DOS, có sự khác nhau về việc ghi chương trình móc nối. Nếu có tiện ích DISKLOOK, ta có thể xem văn bản của chương trình BOOT bằng cách chạy DISKLOOK, ấn núm chức năng F7, gõ vị trí ghi chương trình BOOT (mặt 0, rãnh 0 và cung 1) rồi ấn phím F6 để chương trình BOOT hiện lên màn hình.

## 3. Chương trình vào/ra cơ sở BIOS

BIOS (Basic Input Output System) là các chương trình:

- Điều khiển hệ (các ngắt cứng và ngắt của hệ - xem chi tiết ở dưới).

- Điều khiển vào/ra của các thiết bị ngoài như bàn phím màn hình, máy in.

- Điều khiển ngắt do lệnh của chương trình bởi CTRL-BREAK hay bởi bộ định thời (timer).

- Điều khiển truy cập tới bảng các thông số của màn hình (trong ROM cho khối điều khiển MC 6845), của đĩa hay bảng dữ liệu của các ký tự phụ (ký tự đồ họa).

BIOS được chia thành hai khối:

- *ROM.BIOS*: chứa trong nhớ ROM, có sẵn trong bộ nhớ trung tâm.

- *BIOS.COM*: (đuôi .COM) chứa trên đĩa hệ thống (mềm, cứng) được nạp vào bộ nhớ RAM của bộ nhớ trung tâm.

### *a) ROM.BIOS*

Chứa trong ROM, dung lượng 8KB từ địa chỉ FE000h tới FFFFFh. ROM.BIOS bao gồm các chương trình khác nhau:

- *Chương trình tự kiểm tra PST*: (Power on Self-Test) được thực hiện sau khi bật nguồn điện nuôi hay mỗi lần hồi phục hệ (Ctrl-Alt-Del). Chương trình này thực hiện:

+ Kiểm tra bộ nhớ và các thiết bị ngoài (môi trường) của máy vi tính.

+ Đọc và chép chương trình móc nối từ đĩa vào bộ nhớ RAM (hình 2.7). Nếu không có đĩa hoặc có lỗi trong chương trình được phát hiện, việc điều khiển được chuyển cho chương trình phiên dịch (interpreter) Basic có ở địa chỉ OF6000h đến OFE00h trong ROM-Basic.

- *Các chương trình điều khiển thiết bị ngoài:* (gọi các ngắt cứng bởi vi mạch 8259A với mức ngắt 8 đến F) như đồng hồ (timer), bàn phím, liên lạc không đồng bộ, đĩa cứng, đĩa mềm, máy in.

- *Các chương trình gọi ngắt của hệ* (mức 0 ÷ 7) gồm:

+ Các ngắt logic: chia cho 0, từng bước, dừng, tràn.

+ Các ngắt cứng: NMI (ví dụ sai số chẵn lẻ trong khối nhớ, vi xử lý không tiếp tục làm việc, chép nội dung màn hình ra máy in).

***b) BIOS.COM***

BIOS.COM ra đời nhằm:

- Khắc phục nhược điểm của ROM.BIOS như:

+ Không phù hợp với DOS, cơ cấu chính của HĐH.

+ Phát hiện lỗi của ROM.BIOS và có thể sửa lỗi nhờ BIOS.COM.

+ ROM.BIOS không thể điều khiển được các thiết bị ngoài mới.

- Mở rộng các chức năng của ROM.BIOS bằng các ngắt (10h ÷1Fh) để điều khiển màn hình, xác định kích thước bộ nhớ, hành động đọc và ghi đĩa mềm, cassette, bàn phím, máy in, kiểm tra đồng hồ, truy nhập bảng số liệu của các ký tự đồ họa.

- Chẩn đoán lỗi của các thiết bị ngoài.

Đặc tính chính của BIOS là:

- Xác định trạng thái của thiết bị ngoài.

- Hồi phục các đĩa.

- Khởi động các thiết bị ngoài.

- Nạp các chương trình điều khiển thiết bị ngoài.

- Định nghĩa những véctơ ngắt có số hiệu lớn hơn (10h ÷1Fh).

- Thuộc tính cho địa chỉ nhớ của DOS.COM.

- Gọi DOS.COM.

Về cấu trúc, BIOS có hai phần:

- Phần liên lạc BIOS chiếm địa chỉ nhớ từ 00400h tới 00500h mà các địa chỉ con (từ 00400h) như bảng 2.2.

- Phần lõi của BIOS.

## 4. Khối điều hành đĩa DOS.COM

Khối DOS.COM này là phần cơ bản của HĐH gồm :

- Khối liên lạc của DOS có địa chỉ nhớ đầu là 00500h tới 00600h.

- Khối cơ bản của DOS có địa chỉ đầu là 00B00h.

*Bảng 2.2.* **Địa chỉ nhớ dành cho BIOS liên lạc (bắt đầu từ 0400h)**

| Địa chỉ | Biến số | |
|---|---|---|
| 01 | RS232 base | |
| 08 | PRINTER- base | |
| 10 | EQUIP-Flag | Máy in không đồng bộ |
| 12 | MFG-T&T | (nối tiếp) |
| 13 | Memory-Size | |
| 15 | 10-RAM-SIZE | |
| 17 | KB-Flag | |
| 18 | KB-Flag1 | |
| 19 | ALT-INPUT | Bàn phím |
| 1A | BUFFER-HEAD | |
| 1C | BUFFER-TAIL | |
| 1E | KB-BUFFER | |
| 3E | SEEK-STATUS | |
| 3F | MOTOR-STATUS | |
| 40 | MOTOR- COUNT | Đĩa mềm |
| 41 | DISKETTE-STATUS | |
| 42 | NEC-STATUS | |
| 49 | CRT Mode | |
| 4A | CRT-COLS | Video |
| 4C | CRT-LEN | |
| 4E | CRT-START | |
| 50 | Cursor-POSN | |
| 60 | Cursor-Mode | |
| 62 | ACTIVE-PAGE | Màn hình |
| 63 | ADDR-6845 | |
| 65 | CRT-MODE-SET | |
| 66 | CRT-PALETTE | |
| 67 | EDGE-CNT | |
| 69 | CRC-REG | |
| 6B | LAST-VAL | |
| 6C | TIMER-LOW | |
| 6E | TIMER-HIGH | Đồng hồ |
| 70 | TIMER-OFF | |
| 71 | BIOS-BREAK | Đĩa cứng |
| 72 | RESET-BREAK | |
| 74 đến 76 | | Khác |
| 78 đến 83 | | |

Khối này là giao diện ở mức cao với những chương trình của người dùng, nó chứa:

+ Các chương trình điều khiển tệp (tổ chức thư mục và tệp trên đĩa, khóa và giải khóa việc ghi).

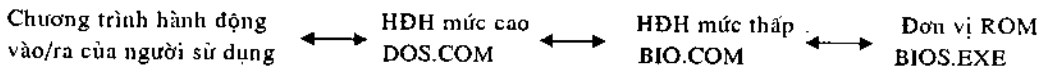
+ Các chương trình gọi các chức năng cho các đơn vị vào/ra theo ký tự (bàn phím, màn hình, máy in...).

+ Các chương trình điều khiển bộ nhớ.

+ Các chương trình gọi thời gian (ngày, giờ, tháng, năm).

+ Chương trình điều khiển việc thực hiện chương trình (dừng, kết thúc).

Khi một chương trình của người dùng thực hiện một chương trình vào/ra, các hành động này phát các chức năng (function) ở mức cao (DOS.COM) bởi trung gian của nội dung các thanh ghi và các khối điều khiển. Những chức năng đó được hoàn thiện bởi các lời gọi tới BIOS.COM như hình 2.8.




**HÌNH 2.8.** Tương tác giữa người điều hành và các thành phần của HĐH.

Các ngắt của DOS (gọi các chức năng) có số hiệu từ 20h đến 3Fh và cả 67h để đưa vào thanh ghi AH trước khi gọi lệnh ngắt INT. Ngắt INT 21h với các chức năng con từ 00h tới 62h để đưa vào thanh ghi ngắt AL trước khi gọi lệnh ngắt chương trình INT nh. Ngắt INT 21h gồm nhiều chức năng con, chính là bộ phận chủ yếu của HĐH MS-DOS .

## 5. Tệp lệnh COMMAND.COM

### *a) Đại cương*

Ngoài BIOS và DOS, COMMAND.COM cũng là thành phần quan trọng của HĐH, nó tương đương CCP (Control Command Processor - bộ xử lý lệnh bàn điều khiển). Khối này có hai chức năng :

- Phiên dịch dòng lệnh đánh vào từ bàn phím.
- Điều khiển những ngắt.

Khi lệnh gõ vào sai về cú pháp; lệnh COMMAND.COM này chỉ ra màn hình là không thấy lệnh (Command not found) và lại quay về dấu nhắc (C >) của màn hình để chờ lệnh mới gõ vào từ bàn phím.

Trong bộ nhớ trung tâm, COMMAND.COM được nạp thành hai đoạn và chứa ba phần:

+ *Phần lưu trú:* nằm ngay trên DOS. Phần này chứa:

. Những khối xử lý những ngắt 22h, 23h và 24h (kết thúc chương trình, dừng khẩn cấp CTRL-BREAK và do sai số).

. Chương trình nạp lại phần bán trú.

. Điều khiển lỗi và thông báo lỗi.

. Những lệnh nội (lưu trú).

+ *Phần khởi phát:* phần này chỉ dùng ở lúc khởi động máy vi tính, vị trí của nó được giải phóng. Sau khi thực hiện (nguyên tắc phủ: overlay). Phần này có:

. Chương trình nạp AUTOEXE.BAT.

. Hiển thị ngày tháng.

. Xác định địa chỉ nạp hay HĐH sẽ nạp những chương trình trước khi thực hiện chúng.

+ *Phần bán trú:* nằm ở bên trên phần bộ nhớ của người dùng (RAM). Nó chứa hầu hết các khối (môđun) điều khiển những tệp của lệnh .BAT (xử lý theo lô). Phần này có thể bị chia nhỏ nếu HĐH cần chỗ, vì có một chương trình của người dùng đang chạy. Nó sẽ được nạp lại khi bộ xử lý lệnh điều khiển máy (phần lưu trú của COMMAND.COM).

***b) Phân loại lệnh theo chức năng***

Với quan điểm chức năng, MS-DOS có 4 nhóm lệnh (hay chương trình) với các đuôi: .COM, .EXE, .SYS và .BAT.

**- Các lệnh loại .COM**

Những chương trình loại .COM (cho COMMAND) mô phỏng những chương trình CP/M có những mảng (segment) của mã lệnh (CS), của số liệu (DS và ES) và của ngăn xếp (SS) là trùng nhau, nghĩa là chúng cùng một giá trị. Mã lệnh và số liệu của một chương trình là xen nhau ở bên trong mảng 64KB và mã lệnh được bắt đầu từ địa chỉ 0100h trong mảng. Những chương trình loại này có thể "quay vòng" ở bên trong một mảng duy nhất; độ lớn của nó, bao gồm vùng hoạt động không thể quá 64KB. Chương trình loại có đuôi .COM này có thể chạy trực tiếp trên máy, không cần một sự biến đổi nào.

**- Các lệnh loại .EXE**

Các lệnh của .EXE (EXEcute) là có thể chuyển đổi được, chúng nằm ở đầu một thông tin chuyển, dành cho chương trình nạp. Các thanh ghi mảng số liệu (DS và ES) được nạp với giá trị (hằng) ở đầu một vùng nhớ có thể sử dụng, trong khi các thanh ghi mã lệnh (CS và IP) và ngăn xếp (SS và SP) nhận giá trị do chương trình kết nối (LINKER) truyền cho. Với cách này, người lập trình có thể khởi phát những thanh ghi ở mọi giá trị mong muốn. Khi nạp chương trình, những tệp loại có đuôi .EXE là phức tạp hơn so với tệp loại .COM và hành động nạp diễn ra lâu hơn. Những tệp .EXE thường sinh ra bởi chương trình dịch assembler của MS-DOS. Chương trình tiện ích EXE2BIN của MS-DOS cho phép biến đổi những tệp .EXE sang tệp .COM để chạy chương trình.

**- Các lệnh loại .SYS**

Đây là các lệnh liên quan tới cấu trúc thuộc nhóm CONFIG.SYS. Đó là các lệnh điều khiển thiết bị ngoài (ANSI.SYS, DISPLAY.SYS, EGA.SYS, Keyboard.SYS) điều khiển khối nhớ (HIMEM.SYS, RAMDRIV.SYS) và các điều khiển khác (CHKSTATE.SYS, DBLSPACE.SYS).

**- Các lệnh loại .BAT**

Những lệnh loại BAT (batch) là những tệp của văn bản chứa các lệnh, tương đương với các tệp .SUB của CP/M. Một số tác giả gọi chúng là các tệp của các chương trình con (thủ tục - procedures). Những lệnh có mặt trong các tệp này có thể là tất cả các lệnh viết đúng của MS-DOS, của loại .COM hay .EXE hoặc cả loại .BAT với các thông số của chúng, các lệnh là độc lập tuyệt đối với nhau.

Những thủ tục là tương đương của chế độ xử lý theo lô (Batch Processing), trong đó, nhiều chương trình (hay các lệnh) được móc nối với nhau một cách tự động, không có sự can thiệp của người điều hành (hay người sử dụng). Những lệnh được dự kiến để cho phép một điều khiển nào đó được diễn ra bởi người dùng MS-DOS như ECHO, REM, PAUSE.

- **ECHO** (tiếng vang): hiển thị lên màn hình những yếu tố của một thủ tục dần dần trong quá trình thực hiện chúng.

- **REM** (ghi chú): cho phép làm hiển thị một ghi chú trong khi tiến hành kiểm tra một số điểm truyền qua.

- **PAUSE** (nghỉ): cho phép treo sự thực hiện một thủ tục để thực hiện một số hành động ngoài (như nạp đĩa vào đơn vị vào/ra; nạp lại giấy lên máy in...)

Bốn loại chương trình hay lệnh trên của MS-DOS chỉ cần gõ vào bàn phím tên của chúng (không cần ghi đuôi mở rộng) và ấn ENTER (CR) sẽ được COMMAND.COM phiên dịch và được vi xử lý thực hiện.

***c) Phân loại lệnh theo vị trí***

Theo vị trí ta chia:

- Lệnh nội trú nằm ở trong bộ nhớ trung tâm.

- Lệnh ngoại trú, chỉ nạp vào bộ nhớ khi gọi đến.

***Những lệnh nội (intern)***

Những lệnh được gọi là nội là những lệnh được nhận biết và xử lý tức thì bởi chương trình phiên dịch (interpreter) COMMAND.COM.

Những lệnh loại này chia thành 7 nhóm (version 2):

+ *Những lệnh khởi phát*

**MODE:** đổi thiết bị đầu cuối chủ.

**PROMPT:** đổi ký tự chờ (cho ổ đĩa A, C).

**DATE:** gọi ngày.

**TIME:** gọi giờ.

+ *Những lệnh điều khiển thư mục*

**DIR:** hiển thị nội dung thư mục.

**CHDIR:** đổi thư mục hiện hành.

**PATH:** chỉ đường dẫn tới tệp hiện hành.

**MKDIR:** tạo thư mục mới.

**RMDIR:** xóa bỏ một thư mục.

+ *Những lệnh điều khiển tệp*

**COPY:** chép nội dung một tệp.

**TYPE:** hiển thị nội dung một tệp.

**RENAME:** đổi tên một tệp.

**ERASE:** xóa một tệp.

+ *Những lệnh điều khiển hoạt động*

**BREAK:** thực hiện CTRL-C.

**CLS:** xóa màn hình.

**SET:** thay nội dung một biến chuỗi.

**EXIT:** ra khỏi DOS.

+ *Những lệnh điều khiển hiển thị và ghi lên đĩa*

**VER:** hiển thị số phiên bản của DOS.

**VERIFY:** kiểm tra lại việc ghi lên đĩa.

+ *Những lệnh của các thủ tục*

**ECHO:** hiển thị lên màn hình những phần tử của thủ tục.

**PAUSE:** dừng, treo một thủ tục.

**REM:** hiển thị những ghi chú.

+ *Những lệnh lập trình các thủ tục*

**FOR...IN...DO:** cho một hành động lặp lại.

**GO TO:** phân nhánh tới.

**IF:** thực hiện có điều kiện.

**SHIFT:** bỏ qua những biến đã xử lý.

**Những lệnh ngoài *(externe):***

Đây là những lệnh không thực hiện bởi bộ phiên dịch lệnh, nhưng được tệp COMMAND.COM điều khiển trong suốt thời gian thực hiện.

MS-DOS còn chia hai loại lệnh ngoài là:

+ Lệnh ngoài do DOS cung cấp.

+ Lệnh ngoài cùng không do DOS cung cấp.

**- Lệnh ngoài của hệ**

+ *Lệnh của phiên bản 1*

*EDLIN.COM*: soạn thảo dòng.

*DEBUG.COM*: để hiệu chỉnh chương trình.

*CHKDSK.COM*: để kiểm tra nội dung của đĩa.

*FORMAT.COM*: tạo dạng đĩa.

*SYS.COM*: chép lại chương trình hệ thống lên đĩa.

*EXE2BIN.COM*: biến đổi một tệp EXE thành COM.

*COMP.COM*: so sánh các tệp trên đĩa.

+ *Lệnh của phiên bản 2*

*CONFIG.SYS*: thay đổi cấu hình của hệ.

*DISCOPY.COM*: chép một đĩa lên đĩa khác.

*MORE.COM*: hiển thị lên màn hình.

*PRINT.COM*: đưa nội dung màn hình lên máy in.

*RECOVER.COM*: lấy một tệp ở đĩa hỏng.

*FIND.EXE*: tìm một văn bản trong một tệp.

*SORT.EXE*: tìm nội dung của một tệp.

*EXEFIX.COM*: biến đổi tệp từ COM sang EXE.

*CAT.COM*: tìm một thư mục.

**- Lệnh ngoài cùng**

Các lệnh này còn chia thành:

+ *Lệnh nhóm 1*: lệnh đặc biệt, tham gia vào hoạt động của hệ.

*LIB.EXE*: điều khiển thư viện của các khối.

*LINK.EXE*: kết nối (soạn thảo kết nối).

*CREF.EXE*: phát các hình chữ thập.

*MASM.EXE*: macro assembler.

+ *Lệnh nhóm 2*: được tạo thành bởi các bộ dịch các ngôn ngữ (biên dịch hay phiên dịch): Basic, Pascal...

+ *Lệnh nhóm 3*: có thể đặt tất cả các phần mềm áp dụng với số lượng không hạn chế.

## 6. Các ngắt của MS-DOS

Các chương trình thực hiện các phục vụ (service) hay các chức năng (function) của hệ MS-DOS được viết và lưu trữ dưới dạng chương trình con được gắn bởi các số liệu ngắt (vector ngắt). Khi cần chỉ cần gọi số hiệu ngắt bởi lệnh gọi ngắt INTnh của vi xử lý thuộc họ Intel IAP86. Trong các số hiệu ngắt hay véctơ ngắt chứa độ dời địa chỉ (IP) và mảng mã lệnh (CS) của chương trình con phục vụ chức năng trên.

Các số hiệu chức năng của MS-DOS như sau:

***a) Các ngắt của BIOS và ROMBIOS***

*- Các ngắt của hệ 0 đến 7 (ROM, BIOS)*

+ Ngắt logic

0 - Chia cho 0

1 - Chế độ từng bước

3 - Điểm dừng thực hiện chương trình

4 - Tràn dung lượng.

+ Ngắt cứng

2 - NMI: ngắt không che được

5 - Chép màn hình (print screen).

*- Ngắt dưới sự điều khiển của 8259A*

Số hiệu ngắt từ 8 đến F (ROM.BIOS) nối vào INTR của VXL, các ngắt có dấu * là chủ yếu.

8* Dao động của đồng hồ (ngắt 18,2 lần/1sec)

9* Chỉ phím ấn của bàn phím dùng để chỉ ngày giờ (INT1Ah) và kiểm tra tốc độ của đĩa

A - Không dùng

B - Liên lạc không đồng bộ (cửa nối tiếp)

C - Bộ phối hợp của liên lạc không đồng bộ

D - Đĩa cố định

E* - Đĩa mềm

F - Máy in.

*- Ngắt điều khiển thiết bị ngoài*

10 - Điều khiển màn hình

11 - Xác định cấu hình của máy vi tính

12 - Xác định kích thước bộ nhớ

13 - Hành động đọc và ghi lên đĩa

14 - Vào/ra cho liên lạc nối tiếp

15 - Hành động vào/ra của cassette

16 - Bàn phím; bổ sung cho INT9h

17 - Máy in

18 - Điểm vào của Basic chứa trong ROM

19 - Điểm vào của khối móc nối (BOOT: địa chỉ 7C00h)

1A - Giờ của ngày.

- *Ngắt phát bởi chương trình*

1B - Ngắt chương trình bởi CTRL-BREAK

1C - Điều khiển bộ đếm đồng hồ.

- *Ngắt truy cập bảng các thông số*

1D - Bảng các thông số màn hình có trong ROM
(dùng để khởi phát khối điều khiển 6845)

1E - Bảng các thông số của ổ đĩa

1F - Bảng các số liệu của các ký tự phụ.

***b) Các ngắt của DOS (gọi các chức năng)***

20 - Kết thúc chương trình, trả điều khiển cho DOS

21 - Gọi các chức năng DOS (chỉ số chức năng trong AH)

22 - Điểm kết thúc chương trình

23 - Điểm dừng khẩn cấp (CTRL-BREAK)

24 - Điểm dừng khi có lỗi

25 - Đọc trực tiếp trên đĩa với địa chỉ trực tiếp

26 - Ghi trực tiếp lên đĩa với địa chỉ trực tiếp

27 - Đặt ở chế độ thường trú của một chương trình

28 ÷ 3F Các ngắt dùng bên trong chương trình.

67 - Ngắt điều khiển khối nhớ.

***c) Các chức năng của ngắt INT 21h của DOS***

Riêng ngắt INT 21h của DOS cho phép truy cập tới 94 chức năng (function) tính tới phiên bản 3.0 trong đó có 13 chức năng được dùng bên trong DOS, còn 81 chức năng có thể được gọi từ một chương trình sử dụng. Trước khi gọi lệnh ngắt INT 21h, số hiệu chức năng phải nạp vào thanh ghi AH của vi xử lý và có thể mỗi chức năng này còn có số hiệu chức năng con để nạp vào AL (xem chương 5).

Danh sách các chức năng của ngắt INT 21h như sau:

| **Số hiệu chức năng** | **Chức năng** |
|---|---|
| 00h | Kết thúc chương trình |
| 01h | Nhập vào từ bàn phím |
| 02h | Đưa ra màn hình |

| | |
|---|---|
| 03h | Vào phụ (bìa không đồng bộ) |
| 04h | Ra phụ (bìa không đồng bộ) |
| 05h | Đưa ra máy in |
| 06h | Vào/ra trực tiếp từ bàn điều khiển |
| 07h | Vào trực tiếp từ bàn điều khiển không đưa ra màn hình |
| 08h | Vào từ bàn điều khiển không đưa ra màn hình |
| 09h | In |
| 0Ah | Vào từ bàn phím ở bộ nhớ đệm |
| 0Bh | Kiểm tra trạng thái của đơn vị vào |
| 0Ch | Xóa bộ đệm bàn phím |
| 0Dh | Hồi phục đĩa |
| 0Eh | Chọn đĩa |
| 0Fh | Mở tệp |
| 10h | Đóng tệp |
| 11h | Tìm số liệu vào đầu tiên trong thư mục |
| 12h | Tìm số liệu vào thứ hai |
| 13h | Bãi bỏ tệp |
| 14h | Đọc nối tiếp |
| 15h | Ghi nối tiếp |
| 16h | Tạo tệp mới |
| 17h | Thay đổi tên tệp |
| 18h | Bên trong DOS |
| 19h | Đọc chỉ số đĩa hiện hành |
| 1Ah | Chỉ địa chỉ của bộ đệm của đĩa (DTA) |
| 1Bh | Các tin tức về bảng các tệp (FAT) |
| 1Ch | Tin tức của bảng các tệp của đĩa |
| 1Dh | Bên trong DOS |
| 1Eh | Bên trong DOS |
| 1Fh | Bên trong DOS |
| 20h | Bên trong DOS |
| 21h | Đọc một tệp |
| 22h | Ghi một tệp |
| 23h | Đọc kích thước một tệp |
| 24h | Khởi phát vùng ghi có chọn lựa trong một FCB |
| 25h | Khởi phát véctơ ngắt |
| 26h | Tạo một tiếp đầu mảng (segment) mới của chương trình (PSP) |
| 27h | Đọc nhiều khối ghi |
| 28h | Ghi nhiều khối, CX = 0: thay đổi kích thước một tệp |
| 29h | Phân tích tên của tệp |
| 2Ah | Đọc ngày trong CX:DX |
| 2Bh | Ghi ngày trong CX:DX |

| | |
|---|---|
| 2Ch | Đọc giờ trong CX:DX |
| 2Dh | Ghi giờ trong CX:DX |
| 2Eh | Chỉ tới chỉ thị đọc kiểm tra sau khi ghi. |

*Chức năng của phiên bản 2.00*

| | |
|---|---|
| 2Fh | Đọc địa chỉ đệm của đĩa (DTA) |
| 30h | Đọc số hiệu phiên bản (0 cho các phiên bản trước 2.0) |
| 31h | Dừng, giữ chương trình thường trú |
| 32h | Bên trong DOS |
| 33h | Kiểm tra mức phát hiện của CTRL-BREAK |
| 34h | Bên trong DOS |
| 35h | Đọc giá trị của một vector ngắt |
| 36h | Đọc các thông số của đĩa (khoảng trống) |
| 37h | Bên trong DOS |
| 38h | Đọc đơn vị tiền tệ, dạng của ngày... |
| 39h | Tạo ra một thư mục con... |
| 3Ah | Hủy bỏ một thư mục |
| 3Bh | Chuyển sang thư mục khác |
| 3Ch | Tạo một tệp |
| 3Dh | Mở một tệp |
| 3Eh | Đóng một tệp |
| 3Fh | Đọc trong bộ đệm DS:DX, n ký tự (n trong CX) của một tệp hay ổ đĩa |
| 40h | Ghi các ký tự trong CX ở bộ đệm địa chỉ DS:DX trong một tệp hay trên đĩa |
| 41h | Bỏ một tệp của một thư mục |
| 42h | Di chuyển con trỏ ở bên trong một tệp |
| 43h | Đọc/chỉ một thuộc tính của tệp |
| 44h | Điều khiển vào/ra, AL xác định chức năng phụ |
| 45h | Có một tên khác bên trong cho cùng một tệp |
| 46h | Thay đổi tệp liên quan tới một chỉ số bên trong |
| 47h | Đọc một thư mục hiện hành |
| 48h | Đặt n đoạn văn bản (n trong BX) |
| 49h | Giải phóng một khối nhớ (mảng nhớ trong ES) |
| 4Ah | Thay đổi kích thước một khối nhớ đã dùng |
| 4Bh | Thực hiện hay nạp một chương trình/tiến trình |
| 4Ch | Kết thúc một tiến trình (EXIT) với sự trở về của mã kết thúc (kết thúc chương trình) |
| 4Dh | Đọc một báo cáo của một tiến trình đã thực hiện |
| 4Eh | Tìm tệp vào đầu tiên |
| 4Fh | Tìm tệp vào tiếp theo |
| 50h | Bên trong DOS |

| | |
|---|---|
| 51h | Bên trong DOS |
| 52h | Bên trong DOS |
| 53h | Bên trong DOS |
| 54h | Trạng thái của việc đọc kiểm tra |
| 55h | Bên trong DOS |
| 56h | Thay đổi tên một tệp |
| 57h | Đọc và chỉ ngày và giờ liên quan tới một tệp. |

*Phiên bản 3.0 và lớn hơn*

| | |
|---|---|
| 59h | Có mã lỗi mở rộng |
| 5Ah | Tạo một tệp tạm thời |
| 5Bh | Tạo một tệp |
| 5Ch | Khóa và mở khóa truy cập tới tệp |
| 62h | Có địa chỉ của PSP (tiếp đầu mảng của chương trình). |

## 2.3. PHẦN MỀM HỖ TRỢ LẬP TRÌNH

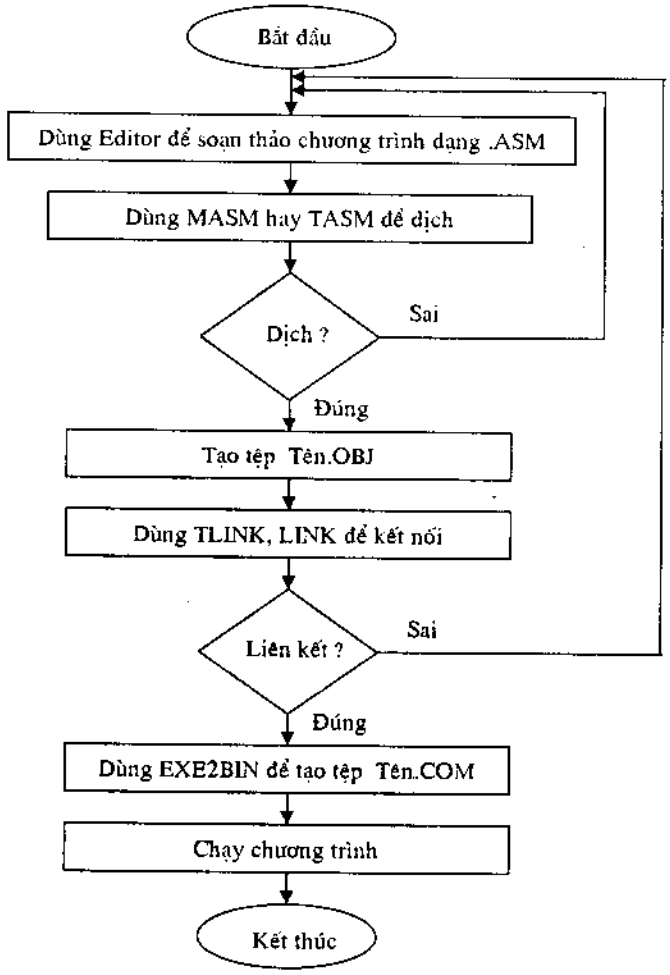
Phần mềm hỗ trợ lập trình là các chương trình hay các lệnh của MS-DOS dùng để giúp cho người lập trình (programmer) soạn thảo, dịch, kết nối và thực hiện (chạy) chương trình trên máy vi tính PC. Đó là:

- Lệnh soạn thảo Editor hay Side kick của TASM giúp ghi văn bản chương trình của người sử dụng vào bộ nhớ để có chương trình dạng ngôn ngữ bậc cao (dạng .ASM, dạng .CCP.)

- Lệnh gỡ rối Debug để ghi văn bản chương trình của người dùng vào bộ nhớ để có chương trình dạng .COM

- Chương trình TASM (Turbo-Assembler của Hãng Borland) hay MASM (Macro - Assembler của hãng Microsoft) để dịch và tạo ra tệp .OBJ.

- Lệnh TLINK (TurboLinker của Borland) hay LINK của Microsoft để kết nối các khối chương trình thành tệp .EXE (Executable Device = Pathname\setver EXE) có thể thực hiện được.




**HÌNH 2.9.** Lưu đồ của quá trình soạn thảo, dịch và chạy chương trình trên hợp ngữ.

- Lệnh EXE2BIN (cho Version 5.0 trở lại hay Device = Path\name) để biến đổi chương trình dạng .EXE sang dạng .COM.

- Lệnh chạy chương trình

+ Ở Debug-G địa chỉ lệnh đầu của chương trình (trong Debug).

+ Ở DOS . A > Tên chương trình <Enter>.

- Lệnh T, P, R của Debug để gỡ rối chương trình (cho chạy từng lệnh, nhóm lệnh, xem nội dung các thanh ghi).

Quá trình biến đổi dạng mã của chương trình của người dùng và việc sử dụng các phần mềm hỗ trợ lập trình được minh họa bởi hình 2.9.

## 2.3.1. LỆNH SOẠN THẢO CHƯƠNG TRÌNH EDLIN

### 1. Đặc điểm chung

Văn bản chương trình của người sử dụng được soạn thảo, ghi vào trong bộ nhớ (dạng mỗi ASCII) bởi chương trình soạn thảo EDLIN của MS-DOS. Đây là một lệnh ngoài, chỉ biên soạn từng dòng một với:

- Số hiệu của dòng lần lượt từ 1 tới 400, sau đó là dấu hai chấm (:)

- Gõ dòng lệnh của chương trình (tối đa tới 253 ký tự)

- Gõ phím Enter (↵) để nhập dòng lệnh từ bộ đệm bàn phím vào bộ nhớ. Nếu trên dòng chưa có ký tự (chưa gõ lệnh) mà ta gõ phím Enter, trên màn hình sẽ để dòng trống chờ nhập vào sau.

- *Cú pháp :* EDLIN [ổ đĩa:] [đường dẫn] tệp [/b] với

- *Thông số: **[ổ đĩa: ] [đường dẫn] tệp*** : xác định nơi và tên tệp văn bản ASCII trên đĩa. Nếu tệp chưa có, EDLIN tạo ra nó trong bộ nhớ và dùng nơi và tên đường dẫn đã chỉ để tạo ra tệp trên đĩa khi ta dùng lệnh E của EDLIN (chép một khối). Chuyển mạch */b* chỉ ra rằng EDLIN bỏ qua ký tự kết thúc tệp là CTRL+Z.

Các lệnh EDLIN là như sau:

- **[dòng]**: hiển thị dòng mong muốn
- **?**: hiển thị danh sách các lệnh EDLIN
- **A**: nạp một phần của tệp vào bộ nhớ khi không gian nhớ không cho phép nạp toàn bộ
- **C**: chép một khối các dòng liên tiếp
- **D**: bỏ một khối dòng liên tiếp
- **E**: ghi tệp đã thay đổi và rời bỏ EDLIN
- **I**: chèn một hay nhiều dòng
- **H**: hiển thị một khối dòng liên tiếp
- **M**: dịch chuyển một khối dòng liên tiếp
- **P**: hiển thị một tệp theo từng trang
- **Q**: rời EDLIN không ghi lại những thay đổi
- **R**: tìm và thay thế một chuỗi ký tự
- **S**: tìm một chuỗi các ký tự
- **T**: hợp nhất nội dung của tệp nằm trong bộ nhớ
- **W**: ghi phần thứ nhất của một tệp vào bộ nhớ trên một đĩa.

## 2. Các nhóm lệnh

*a) Nhóm lệnh thay đổi nội dung chương trình*

Nhóm lệnh này thay đổi nội dung lệnh đã ghi trong khối nhớ. Đó là các lệnh D (Delete - xóa), I (Insert - chèn), M (move - dịch chuyển), R (Replace - thay).

- **D** (Delete - xóa bỏ)

*Cú pháp :* **[dòng1] [dòng 2] d** ↵

**dòng 1**: chỉ dòng đầu tiên phải xoá bỏ

**dòng 2**: chỉ dòng cuối cùng phải xóa bỏ

*Ví dụ:* - 7d ↵: xóa dòng thứ 7

- 22,23 d ↵ xóa dòng 22 tới 23

- , 11 d ↵ xóa từ dòng hiện tại tới hết dòng 11.

- I (Insert - chèn)

*Cú pháp*: **[dòng] I** ↵

*Thông số* : **dòng** chỉ số hiệu dòng trước đó chúng ta muốn EDLIN chèn các dòng.

*Ví dụ:* 8i ↵ màn hình hiện ra

8: -

và chờ gõ tin vào dòng 8 (dấu * chỉ dòng hiện hành), gõ tiếp tin và ↵ cho tới khi muốn ngừng việc chèn bằng cách:

+ Để một dòng trống

+ Ấn CTRL + C.

*Ghi chú:* + Nếu muốn chèn ký tự điều khiển vào băn bản, dùng ^V và theo sau là ký tự điều khiển. Ví dụ muốn chèn một ký tự gây ra một tiếng vang (CTRL+G), ta gõ: ^VG ↵

+ Nếu **[dòng]** là một số hiệu dòng lớn hơn dòng mà ta muốn thay đổi hay nếu [dòng] là ký hiệu #, EDLIN thêm những dòng hiện hành. Nếu chỉ có một phần của tệp có ở trong khối nhớ, dòng được thêm vào cuối của phần có ở trong khối nhớ.

- **R** (Replace - thay)

*Cú pháp:* **[dòng 1] [dòng 2] [?] r [văn bản 1] [ phân cách văn bản 2] R** ↵

*Thông số:* + **dòng 1**: chỉ dòng đầu tiên trong đó ta muốn rằng EDLIN thay chuỗi ký tự chỉ ở văn bản 1.

+ **dòng 2**: chỉ dòng cuối cùng trong đó ta muốn rằng EDLIN thay chuỗi ký tự chỉ trong văn bản 1.

+ *?* (dấu hỏi): chỉ cho EDLIN thông báo một thông tin chúng ta muốn khẳng định trước khi thay thế chuỗi ký tự trong văn bản 1.

+ **Văn bản 1**: chỉ văn bản mà EDLIN phải thay thế.

+ **Phân cách**: phân cách các chuỗi văn bản 1 và văn bản 2. Chỉ có giá trị thừa nhận cho thông số này là ký tự cuối của tệp (CTRL+Z).

+ **Văn bản 2**: chỉ chuỗi ký tự nơi trước khi thay thế các chuỗi ký tự trong văn bản 1.

- **M** (Move - di chuyển)

*Cú pháp:* **[dòng 1] [dòng 2] dòng 3n M** ↵

*Thông số:* + **dòng 1** chỉ dòng đầu tiên mà EDLIN phải di chuyển

+ **dòng 2** chỉ dòng cuối cùng mà EDLIN phải di chuyển

+ **dòng 3** chỉ dòng trước đó chúng ta muốn EDLIN đặt khối các dòng

+ **n** chỉ số dòng chúng ta muốn di chuyển bắt đầu từ dòng hiện hành

***b) Nhóm lệnh quan sát***

Nhóm lệnh này hiển thị và tìm để hiển thị dòng lệnh đã soạn thảo. Đó là các lệnh ? (hiển thị toàn danh sách các lệnh), [dòng] hiển thị dòng lệnh, *l* (hiển thị khối dòng lệnh), p (hiển thị cả trang) và S (tìm và hiển thị).

- **?**: hiển thị danh sách lệnh của EDLIN

*Cú pháp:* **[dòng]** ? ↵

*Thông số*: **dòng** chỉ số dòng mà ta muốn hiển thị. Muốn xem số và văn bản của dòng hiện hành, ấn phím Enter (↵).

- **L** (List - danh sách): hiển thị một danh sách hay một khối các dòng liên tiếp mà ta xác định.

*Cú pháp*: **[dòng 1] [dòng 2] L** ↵

*Thông số:* + **dòng 1** chỉ dòng đầu tiên cần hiển thị

+ **dòng 2** chỉ dòng cuối cùng cần hiển thị.

*Ghi chú:* + Nếu không có thông số [dòng 1], EDLIN hiển thị tới 1 trang (đầy màn hình) bắt đầu từ 11 dòng trước dòng hiện hành và dừng ở dòng có thông số [dòng 2]. Nếu ta quên [dòng 1], phải đưa vào dấu phẩy (,).

+ Nếu quên thông số [dòng 2], EDLIN hiển thị tới một trang bắt đầu từ dòng xác định bởi [dòng 1].

+ Nếu dùng lệnh **L** không có cả hai thông số, EDLIN hiển thị tới 1 trang, bắt đầu ở 11 dòng trước dòng hiện hành.

- **P** (Page - trang): làm hiển thị một phần hay cả trang.

*Cú pháp:* **[dòng 1] [dòng 2] P** ↵

*Thông số:* + **[dòng 1]** chỉ dòng bắt đầu

+ **[dòng 2]** chỉ dòng cuối của trang.

*Ghi chú:* + Nếu không có [dòng 1], hiển thị từ dòng hiện hành

+ Nếu không có [dòng 2] hiển thị một trang

+ Không có cả hai thông số: hiển thị một hay bắt đầu từ dòng hiện hành.

- **S** (Search - tìm): tìm trong một số dòng văn bản mà ta cần chỉ rõ. EDLIN hiển thị dòng đầu tiên chứa chuỗi đã chỉ, rồi dừng việc tìm. Dòng đó trở thành dòng hiện hành.

*Cú pháp:* **[dòng 1] [dòng 2] [?] S [văn bản]**

*Thông số:* + **[dòng 1], [dòng 2]** chỉ các dòng giới hạn văn bản cần tìm

+ **?** (dấu hỏi) chỉ cho EDLIN hiển thị một thông báo chúng ta đề nghị khẳng định khi nó thấy dòng đầu tiên của giá trị đã chỉ trong văn bản.

+ **Văn bản** chỉ văn bản cần tìm. Không đưa vào khoảng trống trước thông số này trên dòng lệnh, trừ khi khoảng trống là một phần của văn bản cần tìm.

*c) Nhóm lệnh ghi văn bản soạn thảo*

Nhóm lệnh này ghi văn bản đã soạn thảo vào khối nhớ và đĩa.

- A (Append): nạp một phần của tệp vào khối nhớ khi không gian nhớ không cho phép nạp toàn bộ.

*Cú pháp:* **[n] A** ↵

*Thông số:* **n** chỉ số dòng của văn bản mà EDLIN phải đọc vào khối nhớ từ đĩa.

*Ghi chú:* + Nếu không có thông số n EDLIN nạp nội dung của tệp tới khi nó chiếm 75% của khối nhớ cho phép. Nếu khối nhớ đã chiếm tới 75%, EDLIN không thêm dòng nào.

+ Để giải phóng không gian nhớ: nếu khối nhớ cho phép đã đầy ta có thể giải phóng không gian nhớ bằng cách ghi một phần của tệp trên đĩa bằng cách đóng một số chương trình ứng dụng hay bằng cách phục hồi MS-DOS sau khi đã từ bỏ EDLIN. Giải pháp cuối cùng này cho phép giải phóng khối nhớ đã được dùng bởi các chương trình thường trú.

+ Thông báo kết thúc tệp, khi lệnh a ghi vào khối nhớ dòng cuối cùng của tệp, EDLIN hiển thị thông báo sau: **END OF FILE**

- C (Copy - chép): chép một khối các dòng liên tiếp trên một hay nhiều vị trí đã cho trong một tệp.

*Cú pháp:* **[dòng1] [dòng 2] [dòng 3] [ số ] C** ↵

*Thông số:* + **dòng 1** chỉ dòng đầu tiên mà EDLIN phải chép

+ **dòng 2** chỉ dòng cuối cùng mà EDLIN phải chép

+ **dòng 3** chỉ dòng trước đó EDLIN phải chèn khối các dòng ở trên

+ **số** chỉ số lần mà EDLIN phải chép khối các dòng.

*Ghi chú:* + Nếu không có các thông số dòng 1 và dòng 2, EDLIN chỉ chép dòng hiện hành. Chúng ta phải đưa dấu phẩy vào dòng lệnh, ngay cả khi quên một trong chúng hay cả hai.

+ Nếu không có số, EDLIN chỉ chép các dòng có một lần.

- E (End - kết thúc): chuyển tệp hiện hành từ khối nhớ lên đĩa và dời EDLIN.

Lệnh E đặt lại tên tệp vào có nguồn gốc trên đĩa bằng cho nó một mở rộng. BAK (trừ trường hợp một tệp mới) chuyển tệp đã thay đổi từ bộ nhớ tới tệp vào có nguồn gốc trên đĩa rồi rời EDLIN.

*Cú pháp:* **E** ↵

*Ghi chú:* + Đĩa và thư mục bất kỳ EDLIN ghi tệp đã thay đổi từ bộ nhớ về đĩa, thư mục và tên của tệp trên đĩa chỉ rõ ở lần gọi EDLIN hiện hành. Nếu ta quên tên đĩa, EDLIN chuyển tệp về đĩa hiện hành. Cũng vậy nếu quên tên thư mục, EDLIN chuyển tệp vào thư mục hiện hành.

+ Trước khi dùng lệnh E, ta phải xác nhận rằng đủ chỗ trên đĩa để cho tất cả tệp đã thay đổi ở trong bộ nhớ. Nếu không EDLIN làm mất một phần hoặc tất cả tệp.

+ Giả sử rằng ta muốn EDLIN ghi một tệp mà phiên bản BAK là một tệp chỉ có đọc, EDLIN hiển thị một thông báo dưới dạng sau: Access refused để chỉ rằng EDLIN không thể thay thế tệp .BAK.

Những phiên bản gốc và việc bảo vệ tệp của ta không thay đổi.

- **T** (Transfer - chuyển): nhập nội dung của tệp trên đĩa với nội dung của tệp ở khối nhớ.

*Cú pháp:* **[dòng ] T [đĩa:] [đường dẫn] tệp** ↵

*Thông số:* + **dòng** chỉ số liệu dòng trước đó ta muốn EDLIN chèn tệp chuyển từ đĩa vào. Nếu không có thông số dòng, tức số hiện dòng hiện hành.

+ **[đĩa:] [đường dẫn] tệp** chỉ nơi và tên tệp mà ta muốn EDLIN chèn trước dòng mà số hiệu chỉ bởi thông số dòng. Nếu không có các thông số, đó là đĩa hiện hành và thư mục hiện hành.

+ **W** (Write - ghi): chuyển phần đầu tiên của tệp đã thay đổi từ bộ nhớ lên đĩa. Khi gọi EDLIN nó đọc những dòng có thể của tệp lên đĩa trong bộ nhớ. Nếu kích thước của tệp vượt quá lượng nhớ có thể ta phải thay đổi tệp từng giai đoạn nghĩa là ta thay đổi phần đầu tiên của tệp, chuyển phần này của tệp lên đĩa bằng lệnh W rồi nạp phần tiếp theo từ đĩa bằng cách dùng lệnh A.

***d) Nhóm lệnh rời EDLIN trở về MS-DOS***

- **E** (End) ghi và rời EDLIN.

- **Q** (Quit - rời bỏ). Rời bỏ EDLIN hiện hành không ghi lên đĩa những sự thay đổi của tệp.

Khi dùng lệnh Q, EDLIN kết thúc và dẫn tới lệnh của MS-DOS.

*Cú pháp:* **Q** ↵

Sau khi gõ Q, EDLIN hiển thị thông báo: *Cancel EDLIN (Y/N) ?* ta phải ấn lên Y ↵.

## 2.3.2. CHƯƠNG TRÌNH DỊCH CÁC NGÔN NGỮ LẬP TRÌNH

### 1. Đặc điểm chung của các ngôn ngữ lập trình

***a) Định nghĩa ngôn ngữ lập trình***

Máy vi tính thực hiện lệnh ghi sẵn trong ngăn nhớ. Các lệnh này tạo thành từ các bit 0 và 1 theo những dạng quy ước sẵn do người chế tạo. Người sử dụng máy tính hay người điều hành (operator) phải sử dụng các lệnh đã quy ước trên để ra lệnh cho máy thực hiện một lệnh nào đó trong bộ lệnh đã định sẵn. Đó chính là ngôn ngữ máy dành cho sự giao dịch giữa người và máy.

Người điều hành sử dụng đúng các lệnh của máy (tức ngôn ngữ máy) để thực hiện một hành động nào đó (như vào/ra số liệu, làm phép toán số học, logic với các dữ liệu... thì máy thực hiện trực tiếp lần lượt các lệnh (tức biến đổi thành dạng mà máy có thể hiểu trực tiếp).

Một chương trình tức là một chuỗi lệnh mà người lập trình (programmer) muốn máy thực hiện với sự đánh dấu lệnh bắt đầu và lệnh kết thúc.

Để thuận tiện cho người lập trình, người ta đã biên soạn ra nhiều ngôn ngữ khác nhau, ngoài ngôn ngữ máy (dạng các bit 0 và 1) trên. Máy tính điện tử không thể hiểu được các ngôn ngữ này mà phải qua sự phiên dịch thành ngôn ngữ máy bởi các chương trình phiên dịch (interpreter) và biên dịch (compilator).

***b) Phân loại các ngôn ngữ lập trình***

Người ta chia ngôn ngữ lập trình thành hai loại lớn:

- Ngôn ngữ hướng máy.

- Ngôn ngữ hướng người sử dụng.

Ngôn ngữ hướng máy (machine oriented) còn gọi là ngôn ngữ cấp thấp, là ngôn ngữ gần với máy, tức máy dễ dàng hiểu hơn, gắn liền với cấu trúc của máy (vi xử lý, bộ nhớ, cổng vào/ra, thiết bị ngoài). Ngôn ngữ hướng máy này gồm:

- Ngôn ngữ máy hay lệnh mã (code instruction) đã giới thiệu ở trên. Máy có thể hiểu ngôn ngữ này một cách trực tiếp không cần chương trình phiên dịch. Nhược điểm của ngôn ngữ này là:

+ Khó nhớ

+ Dễ nhầm lẫn

+ Chương trình viết dài, khó theo dõi và kiểm tra.

- Hợp ngữ cũng là ngôn ngữ gắn liền với phần cứng của máy, có hai tên gọi:

+ Hợp ngữ (dịch chữ assembly) vì mỗi lệnh là sự tập hợp của một trong ba thành phần; mã lệnh, toán hạng nguồn, toán hạng đích (xem chương 3).

+ Thuật nhớ (memonic).

Để khắc phục điểm khó nhớ, dễ lầm lẫn của các lệnh dạng mã (các con số 0,1), người ta gán cho mỗi lệnh một tên gợi nhớ, đó là chữ viết tắt của tên tiếng Anh hành động của lệnh (ví dụ: cộng ADD-ADDition). Như vậy có thể nói hợp ngữ đứng giữa người và máy vì:

+ Gần với người ở dạng thuật nhớ để giúp người dễ nhớ và dễ sử dụng.

+ Gần với máy vì mỗi lệnh nói lên một hành động của máy. Khi dịch, chương trình chỉ cần tra bảng tương ứng giữa tên lệnh và mã lệnh kèm theo các toán hạng.

Hợp ngữ có nhược điểm là:

- Vẫn còn gần với phần cứng nhiều hơn. Người lập trình bằng hợp ngữ assembly cần phải hiểu rõ phần cứng.

- Chương trình bằng ngôn ngữ này còn dài, khó theo dõi.

- Chương trình chỉ sử dụng cho một máy cụ thể nhất định, không dùng cho máy khác (Untransportable).

Khắc phục nhược điểm trên của ngôn ngữ hướng máy (gần máy), người ta đã tạo ra các ngôn ngữ bậc cao như Basic, Turbo, Pascal, C, C $^{++}$, Fortran, ADA...

Đặc điểm của ngôn ngữ bậc cao này là:

- Gần với người sử dụng, gần với sự suy nghĩ của con người, gần với các hành động của các thiết bị ngoài như xóa màn hình, đưa số liệu vào, đưa số liệu ra, in, ghi lên đĩa...

- Có khả năng dùng cho các máy khác, chỉ cần chương trình dịch tương ứng với máy đó. Như vậy chương trình ngôn ngữ bậc cao này trở nên thông dụng cho nhiều người nhiều máy. Không đòi hỏi người sử dụng phải hiểu biết sâu về cấu trúc một máy cụ thể.

- Chương trình dịch phức tạp hơn assembler và có hai loại:

+ *Chương trình phiên dịch (interpreter):* dịch và thực hiện từng lệnh cho hết chương trình. Chương trình dịch này cho các ngôn ngữ truyền đạt (interpretative) dùng trong các hệ điều khiển kỹ thuật (ví dụ APL) và Basic.

+ *Chương trình biên dịch (compilator)*: biên soạn và dịch cả văn bản của chương trình, sau đó mới thực hiện (chạy) cả các lệnh của chương trình. Đây là các chương trình biên soạn cho các ngôn ngữ được biên soạn (compilated) như Pascal, C, C$^{++}$, Fortran, Cobol, Algol...

Ngoài cách phân loại ngôn ngữ trên, người ta còn phân loại:

- *Ngôn ngữ dùng với mục đích khoa học:* Fortran, Basic, Algol, APL.

- *Ngôn ngữ trong điều khiển tự động:* Cobol, LISP, RPG hay GAP, bảng quyết định, ngôn ngữ cho cơ sở dữ liệu.

- *Ngôn ngữ cho nghiên cứu:* ngôn ngữ của bảng danh sách (IPL, V, L6, LISP) ngôn ngữ thao tác các ký tự (COMIT, Snobol, FORMAC), ngôn ngữ để mô phỏng (SIMCRIPT, SIMULA, GPSS, CSMP).

- *Ngôn ngữ có xu hướng tổng hợp*: PL/1, Algol 68, Pascal, ADA.

***c) Đặc điểm chung của các ngôn ngữ lập trình***

Bất cứ ngôn ngữ lập trình nào đều có các đặc điểm chung, chỉ khác cách sử dụng và cách diễn tả các câu lệnh, cách viết chương trình khác nhau. Các đặc điểm chung đó là:

1. Đều sử dụng các bảng chữ cái từ A tới Z, các số 0÷9, các dấu (, . ; : [ ] ; ( ) ..).

2. Định nghĩa và khai báo về các kiểu dữ liệu (số, ký tự) với hằng số, biến số, chuỗi ma trận, mảng, bản ghi (record) và các hàm số, các biểu thức.

3. Các toán tử (số học, logic, gán, địa chỉ, tập hợp, chuỗi).

4. Lệnh và câu lệnh gồm các lệnh đơn (lệnh của máy, lệnh gán biểu thức số học, lệnh gọi chương trình con), lệnh gộp (macro lệnh) hay phức hợp, lệnh có cấu trúc (có điều kiện - IF), chuyển mạch - case, lặp - (repeat, while).

5. Các lệnh giả hay hướng dẫn biên dịch (directive), lệnh dịch có điều kiện chỉ có tác dụng cho biên dịch chương trình.

6. Cú pháp hay cách viết:

+ Các khai báo về số liệu (để dành địa chỉ nhớ)

+ Cách khai báo một câu lệnh, một chương trình con, một chương trình chính, cách gọi chương trình...

## 2. Chương trình biên dịch hợp ngữ (assembler)

***Các chương trình biên dịch của hệ điều hành***

Thông thường một hệ điều hành đều có các chương trình biên dịch và phiên dịch một số ngôn ngữ lập trình cơ bản. Do đó, MS-DOS cũng có:

- Lệnh Debug để dịch chương trình hợp ngữ đơn giản sang dạng .COM.

- Lệnh GWBasic hay QBasic (Quick-Basic) phiên dịch ngôn ngữ basic.

- MASM (của Microsoft) hay TASM (của Borland), biên dịch chương trình ngôn ngữ assembly sang ngôn ngữ máy.

- Turbo Pascal (của Borland) biên dịch chương trình ngôn ngữ Pascal sang ngôn ngữ máy.

- Turbo C (của Borland) hay C Microsoft biên dịch chương trình ngôn ngữ C sang ngôn ngữ máy.

Chương trình biên dịch MASM chuyển một tệp nguồn (.source) viết bằng hợp ngữ thành tệp đối tượng (.OBJ) dạng ngôn ngữ máy. Nó tạo ra ba tệp là:

- **Tệp đối tượng (.OBJ)** chứa bản dịch ra mã máy, của các lệnh trong chương trình nguồn hợp ngữ và các thông tin cần thiết để tạo nên một tệp chương trình.

- **Tệp danh sách (list file)** là một tệp văn bản, đưa ra các mã lệnh assembly và mã máy tương ứng, danh sách các tên (nhãn, lệnh giả, biểu thức, biến...) dùng trong chương trình, các thông báo lỗi và các số liệu thống kê khác. Tệp này rất có ích khi gỡ rối chương trình.

- **Tệp tham khảo chéo (cross-reference file)** liệt kê các tên sử dụng trong chương trình và các dòng mà chúng xuất hiện. Nhờ nó chúng ta dễ dàng theo dõi các chương trình lớn hơn. Chúng ta dùng chương trình tiện ích CREF để đổi nó sang dạng dễ đọc hơn.

Dòng lệnh của MASM version 5.0 như sau: *C > MASM A: First.ASM, A: First.OBJ, A: First.LIST, A: First.CRF* ↵

Trong đó A là ổ đĩa A chứa tên tệp First.

Một cách ngắn gọn có: *C > MASM A: First, A: , A: , A:* để tạo ra ba tệp đầy đủ.

Có thể dùng dấu chấm phẩy (;) để không cần tạo ra tệp không cần thiết.

Ví dụ: *C >MASM A: First, A:;* (chỉ tạo ra tệp First.OBJ)

Có thể trả lời các câu hỏi in ra ở màn hình để tạo các tệp cần.

Ví dụ:

- *Object filename [First.OBJ] : A: <enter>* để cho tệp đối tượng First.
- *Source liting [NuL.LST] : <enter>* không cần tệp LST.

Các tùy chọn có thể sử dụng các tùy chọn (option) [ghi một hay nhiều].

Ví dụ: *C > MASM/D/W2/Z/ZI First;*.

Bảng các tùy chọn như sau:

| Tùy chọn | Ý nghĩa |
|---|---|
| /A | Sắp xếp các đoạn nguồn theo thứ tự a, b, c |
| /C | Tạo lập một tệp tham khảo chéo. |
| /D | Tạo lập danh sách trong lần duyệt thứ nhất |
| /ML | Phân biệt chữ hoa và chữ thường trong tên |
| /R | Chấp nhận các lệnh dấu phẩy động |
| /S | Sắp các đoạn nguồn theo thứ tự ban đầu |
| /W {0\|1\|2} | Thiết lập mức độ hiển thị lỗi (mặc định = 0)<br>0 = các dòng lệnh không hợp lệ<br>1 = các dòng lệnh không rõ nghĩa<br>2 = các dòng lệnh có thể tạo ra các mã lệnh không có hiệu lực |
| /Z | Hiển thị các dòng có lỗi |
| /Z/ | Viết các thông tin về các ký tự dùng trong chương trình nguồn vào file đối tượng (dùng cho CODEVIEW). |

Từ các lỗi ta có thể sửa được cách viết chương trình khi kết hợp với chương trình gỡ rối.

## 2.3.3. CHƯƠNG TRÌNH NẠP VÀ KẾT NỐI (LINKER)

### 1. Yêu cầu nạp và kết nối

Sau khi biên dịch, ta có một văn bản của chương trình thống nhất gồm nhiều khối của chương trình. Các khối này có địa chỉ nhớ ở các mảng (segment) khác nhau. Do đó cần:

- Chuyển chương trình .OBJ thành dạng có thể chạy trên máy vi tính (.EXE), tức phân bố chương trình OBJ vào các vùng nhớ với các địa chỉ tuyệt đối bởi chương trình phân bố (dipatcher).

- Ghép nối các khối của chương trình (các chương trình con, các chương trình gọi từ thư viện, chương trình chính) thành một khối thống nhất bởi chương trình kết nối LINKER (gọi bởi lệnh LINK của Microsoft) hay TLINK (của Borland). Chính quá trình kết nối này nên mới có tên là chương trình hợp dịch (Assembler).

### 2. Sử dụng LINKER

Từ các *tệp đối tượng (.OBJ)* và *tệp của thư viện (.LIB)*, sau khi dùng *lệnh LINK*, ta được các tệp *.EXE* của chương trình và tệp *loadmap* (chứa kích thước và các vị trí tương đối của các đoạn chương trình.

*Cú pháp* của LINK 5.0 như sau:

*C > LINK A: First + Second + Third, A : First, A: First;*

hay chỉ cần *C > LINK A: First + Second + Third, A:, A:;*

Trong đó:
+ *A* là ổ đĩa A chứa các chương trình .OBJ
+ *First, Second, Third* là tên các chương trình cần hợp dịch
+ *A* thứ hai chỉ tệp Loadmap
+ *A* thứ ba chỉ tệp .LIB (thư viện).

Có thể dùng lệnh: *C > LINK First + Second + Third*

rồi trả lời trên màn hình với: *RUN file [First.EXE]: <enter>*

*LIST file [First.MAP] : <enter>*

*Libaries [Nul.LIB] : <enter>*

Có nghĩa là:
+ câu trả lời 1 chấp nhận tên First.EXE cho tệp chương trình đã kết nối
+ câu trả lời 2 chấp nhận tên First.MAP cho tệp loadmap
+ câu trả lời 3 chứng tỏ không có tệp thư viện nào.

## 2.3.4. CÁC CHƯƠNG TRÌNH GỠ RỐI *(DEBUG)*

Các chương trình gỡ rối này dùng để tìm chỗ gây ra lỗi của soạn thảo chương trình nếu chương trình biên dịch không chỉ ra lỗi.

### 1. Nguyên tắc chung của gỡ rối

- Cho chạy từng lệnh hay nhóm lệnh để xem diễn biến của lệnh. Nếu diễn biến sai hoặc lệnh không thực hiện, ta biết lỗi ở chính lệnh đó.

- Dùng các lệnh của chương trình gỡ rối có thể quan sát kết quả lệnh trên màn hình (xem nội dung các thanh ghi bộ nhớ, biểu thức...) hoặc thay đổi lệnh để tìm lệnh đúng.

### 2. Debug

Là chương trình có thể soạn thảo chương trình .COM đơn giản và gỡ rối bằng gõ các lệnh với chữ cái A, B, C (xem chương 4).

### 3. Codeview

Là chương trình gỡ rối mạnh hơn Debug ở chỗ:

- Có hai chế độ tuần tự (gần giống Debug) và cửa sổ (xem phụ chương).

- Có nhiều tùy chọn cho các máy không tương thích với IBM dùng chuột điều khiển Codeview, màn hình không tương thích IBM...

- Chọn lựa lệnh theo biểu tượng ở cửa sổ.

- Chọn lựa lệnh chạy chương trình theo thực đơn (Menu RUN).

- Quan sát cả bộ nhớ, ngăn xếp và cả biểu thức.

- Định địa chỉ gián tiếp thanh ghi.

- Xóa các dòng từ cửa sổ quan sát.

## 2.4. PHẦN MỀM TIỆN ÍCH (UTILITIES)

Phần mềm tiện ích hay phần mềm phục vụ là các chương trình do các hãng viết ra để bán, khiến việc sử dụng máy tính được thuận tiện và có ích (tiện ích). Có rất nhiều loại chương trình tiện ích, phục vụ cho nhiều ngành chuyên môn khác nhau và ngày càng phong phú. Chúng ta có thể kể:

### 1. Các chương trình phục vụ việc điều hành máy tính

- **NU** (Norton Utilities): có khoảng 15 chương trình để lựa chọn và số liệu, khôi phục các tệp bị phá hủy.

- **NC** (Norton Commander): cho phép tìm tệp trong thư mục và thư mục con của đĩa cứng và chọn hầu hết các lệnh của MS-DOS một cách nhanh gọn (thay cho gọi lệnh theo đường dẫn của MS-DOS).

- **PCTOOLS**: ưu điểm hơn NU là nó có thể thường trú trong bộ nhớ. Có thể chọn lệnh hay tệp bằng các phím ↑ và ↓ (khi chọn ấn phím Return - RC).

- **Windows** (tới version Windows 95, Windows 97, Windows 98): chính là hệ điều hành bổ sung cho MS-DOS, chạy trong DOS nhưng điều khiển bằng cửa sổ với biểu tượng (icon), thực đơn và chọn bằng ấn phím hay nháy chuột.

### 2. Các chương trình xử lý văn bản

- *Word (Microsoft, chạy trong Windows)*

- *Word Perfect (SU Software)*

- *Star*

- *Write (Windows).*

### 3. Chương trình phục vụ đồ họa

- *Lotus 1.2.3 (Lotus)*

- *Chart (Microsoft).*

### 4. Các chương trình điều khiển tệp và cơ sở dữ liệu

- *Dbase II, III, Plus và IV của Ashton Tate*

- *RBase 5000 (Microsoft)*

- *Reflex (Borland)*

- *Foxpro*

- *Access* (chạy trong *Windows)*

- *Dialogue II (Bull).*

## 5. Các chương trình xử lý bảng, biểu cho thống kê, kế toán

- *Lotus 1.2.3 (VV Lotus)*
- *Multiplan (Microsoft)*
- *Excel (Microsoft, chạy trong Windows).*

## 6. Các chương trình đa dụng trong phần mềm tích hợp (logiciel integrated)

- *Framwork II* (Ashton Tate) có ngôn ngữ lập trình "FRED" (mạnh nhưng đào tạo lâu). Symphony (Lotus) có bốn chức năng cổ điển và thêm chức năng truyền thông.

+ *SHEET* : bảng với 8192 dòng, 252 cột

+ *DOC* : xử lý văn bản

+ *GRAPH* : đồ họa với 6 loại

+ *FORME* : cơ sở dữ liệu

+ *COMM* : truyền thông với IBM 4341 hay 3083.

- *OPEN Access II* (Software Products International) với các khối

+ Access base : cơ sở dữ liệu

+ Access Calc : bảng đồ họa

+ Access Texte : xử lý văn bản

+ Access bureau : các công cụ văn phòng.

- *Smart* (Innivative Software) với xử lý văn bản, cơ sở dữ liệu (50 tệp, mỗi tệp tới 1 triệu phép ghi, cỡ tối đa 4MB), bảng đồ họa (tới 9999 dòng, 999 cột), mỗi cột từ 0÷99 ký tự truyền thông và nhật ký.

# CÂU HỎI

1. Phần mềm hệ thống là gì ? Liệt kê những thành phần chính của phần mềm hệ thống của một máy vi tính hiện nay.

2. Vai trò, thành phần và tác dụng của từng thành phần của hệ điều hành (HĐH) trong máy vi tính hiện nay.

3. Các phục vụ của một HĐH.

4. Các loại ngắt của một HĐH, nêu tác dụng của từng loại, mục đích sử dụng các chương trình phục vụ ngắt. Muốn phát triển các phục vụ ngắt, ta phải làm gì ? Các chế độ của một HĐH ?

5. Liệt kê một số HĐH điển hình cho máy vi tính trong quá trình phát triển máy vi tính tới ngày nay.

6. Monitor của một hệ vi xử lý là gì ? Kể một số thành phần và lệnh cơ bản của nó cho một hệ vi xử lý nhỏ. So sánh nó với HĐH của một máy vi tính lớn hơn.

7. HĐH MS-DOS : cấu trúc, các thành phần và tác dụng của từng thành phần ?

8. Phân loại, tác dụng của từng loại lệnh và liệt kê một số lệnh chính của HĐH MS-DOS.

9. Các loại ngắt của HĐH MS-DOS. Tác dụng của từng loại. Vai trò của các chương trình con phục vụ ngắt INT nh của HĐH MS-DOS.

10. Vai trò của lệnh và các chương trình con phục vụ ngắt trong quá trình lập trình cho mục đích sử dụng của người dùng máy vi tính.

11. Bạn hãy so sánh các hệ điều hành MS-DOS, Windows và Macintos.

12. Phần mềm hỗ trợ lập trình là gì ? Thành phần và tác dụng của từng thành phần trong quá trình lập trình ?

13. Phần mềm tiện ích là gì ? Liệt kê một số phần mềm tiện ích thông dụng mà bạn thường dùng.

14. Lập trình hệ thống là gì ? Tác dụng của nó đối với người học tập, nghiên cứu và sử dụng máy vi tính.

15. Tầm quan trọng của phần mềm hệ thống đối với người nghiên cứu về phần cứng, về phần mềm hệ thống và người lập trình ứng dụng.

# *CHƯƠNG 3*

# HỆ LỆNH CỦA VI XỬ LÝ INTEL 86

Máy vi tính (MVT) hoạt động được do thực hiện chương trình gồm nhiều lệnh. Lệnh là hành động cơ bản của mỗi vi xử lý (VXL) mà người thiết kế đã dự định để chế tạo. Mỗi lệnh có dạng mã riêng (gồm một tổ hợp các bit 0 và 1) mà chỉ người thiết kế và VXL hiểu được, được gọi là ngôn ngữ máy dạng mã. Mỗi loại máy khác nhau, có dạng mã máy quy ước riêng cho nó. Muốn VXL thực hiện một lệnh nào đó, người sử dụng phải đưa vào nó một mã lệnh có dạng đúng như quy định của người đã thiết kế cho VXL.

Bộ lệnh gồm nhiều lệnh, số lượng lệnh tùy thuộc vào loại VXL (ví dụ 8086 là 111 lệnh). Bộ lệnh này được ghi trong VXL. Nói chung, bất kỳ VXL nào cũng có bộ lệnh gồm các nhóm: chuyển số liệu, xử lý số học (cộng, trừ, nhân, chia), xử lý logic (và, hoặc, đảo, loại trừ, dịch chuyển, quay vòng), xử lý chuỗi ký tự và xử lý chương trình (dừng, không thực hiện gì và rẽ nhánh, điều khiển lập/xoá một số bit của thanh ghi cờ...).

Người sử dụng thường viết chương trình gồm các lệnh dạng ngôn ngữ bậc cao hơn ngôn ngữ máy cho dễ sử dụng, như hợp ngữ (assembly), Turbo Pascal, C, C$^{++}$, Fortran, Algol, Cobol... Muốn VXL nói riêng, MVT nói chung hiểu được dạng ngôn ngữ bậc cao hơn này, phải có chương trình dịch các lệnh này sang mã lệnh tức dạng ngôn ngữ máy.

Ở chương này, chúng ta sẽ xét:

- Dạng lệnh của VXL 8086 dưới dạng mã lệnh.
- Quá trình thực hiện một lệnh.
- Dạng lệnh của VXL 8086 dưới dạng hợp ngữ (assembly).
- Các phương pháp địa chỉ hoá.
- Tập lệnh của 8086 dưới dạng hợp ngữ.
- Các lệnh bổ sung của cácVXL khác (80286, 80386, 80486, 80586).

Chương này có vai trò rất quan trọng đối với việc *lập trình bằng hợp ngữ cho máy vi tính PC-IBM*, là phần cơ bản của việc viết chương trình hợp ngữ cho VXL 8086 và các MVT khác thuộc họ PC của hãng IBM.

## 3.1. LỆNH DẠNG NGÔN NGỮ MÁY

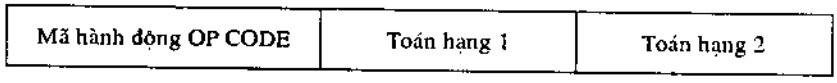
### 3.1.1. DẠNG TỔNG QUÁT CỦA MÃ LỆNH

Máy vi tính chạy chương trình tức là thực hiện một chuỗi các lệnh. Giữa người lập trình và máy trao đổi với nhau qua một trong các ngôn ngữ sau:

- *Ngôn ngữ máy*: gồm tổ hợp các bit 0, 1 được viết dưới dạng mã.
- *Hợp ngữ* (assembly): gồm các ký tự (chữ cái, chữ số và các ký hiệu).
- *Ngôn ngữ bậc cao*: Basic, Pascal, C, C$^{++}$, Fortran v.v...

Ngôn ngữ mã máy không cần sự chuyển đổi mã nhờ chương trình dịch (chương trình thông dịch - interpreter hay chương trình biên dịch - compiler) như hợp ngữ và các ngôn ngữ bậc cao khác.

Một lệnh dạng ngôn ngữ máy gồm tổ hợp các bit 0, 1 được chia theo nhóm hoặc vùng (field) (hình 3.1) gồm mã hành động và các toán hạng.




**Hình 3.1.** Dạng lệnh tổng quát của một vi xử lý.

Lệnh dạng mã hay mã lệnh là hành động mà người điều hành máy ra lệnh cho máy phải thực hiện (được viết dưới dạng mã - tổ hợp các bit 0, 1 theo quy luật nào đó tùy người sử dụng đặt trước). Mã lệnh này được viết trong chương trình hay trực tiếp ghi từ bàn phím, rồi truyền cho vi xử lý (VXL) của máy vi tính. Mã lệnh này có dạng quy ước của người chế tạo đặt ra cho máy. Người sử dụng máy phải tuân theo để có thể trao đổi hay "nói chuyện trực tiếp" với máy, đúng như đã thiết kế và chế tạo. Có như vậy, máy mới hiểu được lệnh của người điều hành và thực hiện đúng hành động mà người điều hành mong muốn.

Dạng mã và số lệnh hoàn toàn phụ thuộc vào người thiết kế và kỹ thuật chế tạo VXL của máy vi tính. Ví dụ: VXL Intel 8080 chỉ có 72 lệnh, còn 8085 có 74 lệnh. Dưới đây chúng ta sẽ xét chi tiết mã lệnh cho VXL 8086/8088.

## 1. Mã hành động (Operation Code - OP Code)

Số bit nhị phân (0, 1) dành cho mã hành động phụ thuộc vào số lệnh mong muốn. VXL họ Intel 86 cần tới 9 bit (6 bit ở byte thứ nhất và 3 bit ở byte thứ hai). Như vậy có thể mã được tới 512 hành động của lệnh. Đa số mã hành động của lệnh cần 6 bit ở byte thứ nhất của lệnh.

## 2. Toán hạng (Operand)

Toán hạng theo nghĩa hẹp là số hạng cho phép toán. Nhưng theo nghĩa rộng, toán hạng chứa các tin tức cần thiết cho hành động của lệnh ghi ở phần mã lệnh. Tùy theo loại lệnh hay tùy hành động của VXL cần thực hiện, có thể xảy ra một trong các trường hợp sau:

- Lệnh không cần toán hạng như các lệnh hiệu chỉnh các phép toán số học (AAA, AAD, AAM, DAS v.v...), các lệnh điều khiển VXL (NOP, HLT, IRET, RET, STC, STD, CLC, CLV v.v...).

- Lệnh cần một toán hạng như các lệnh cần một số liệu liên quan tới một thanh ghi, một ô nhớ, một địa chỉ v.v...

- Lệnh cần hai toán hạng liên quan tới hai số liệu như các phép toán số học, logic.

Toán hạng có thể là:

+ Một con số chỉ một số liệu hoặc một địa chỉ.

+ Địa chỉ của một số liệu nằm trong thanh ghi hay trong ô nhớ.

+ Một con trỏ gián tiếp chỉ tới một số liệu.

+ Tin tức khác liên quan tới số liệu mà lệnh sẽ sử dụng.

Về mặt sử dụng, người ta mong muốn có nhiều toán hạng lớn hơn hai. Số toán hạng càng nhiều, độ dài lệnh càng lớn, số ô nhớ chứa lệnh càng nhiều, thời gian đọc và thi hành lệnh càng lớn, nhưng lại chứa được nhiều số liệu liên quan tới lệnh. Do đó, thường tối đa

một lệnh chỉ có hai toán hạng. Điều này gây khó khăn cho các phép toán cần hai số hạng, kết quả sẽ gửi vào một trong hai toán hạng đó, có nghĩa là một toán hạng cũ bị mất đi, chỉ còn một toán hạng cũ và kết quả của phép toán. Với lệnh có hai toán hạng, một toán hạng chứa trong một thanh ghi, còn toán hạng kia ở trong một thanh ghi hay ô nhớ.

Số liệu của toán hạng có thể là:

- Một con số đứng liền sau mã lệnh, ở *chế độ địa chỉ tức thì.*

- Nội dung của thanh ghi thông dụng của VXL, ở *chế độ địa chỉ thanh ghi.*

- Nội dung của một hay nhiều ô nhớ, ở *chế độ địa chỉ gián tiếp.* Số liệu trực tiếp, nội dung thanh ghi chỉ địa chỉ của ô nhớ. Dưới đây chúng ta sẽ xét kỹ các chế độ địa chỉ này.

## 3.1.2. DẠNG MÃ LỆNH CỦA 8086/8088

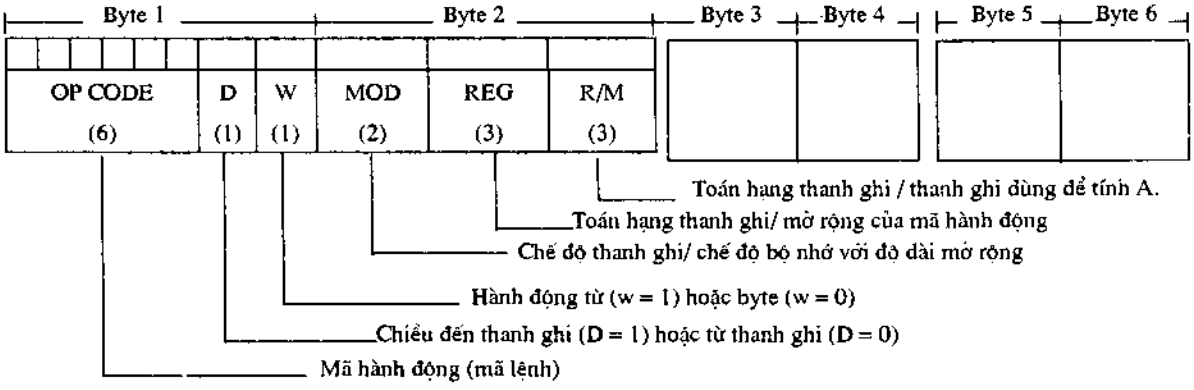
### 1. Độ dài của lệnh

VXL 8086 có các lệnh có độ dài từ 1 đến 6 byte (hình 3.2). Mỗi lệnh đều có mã lệnh (tức là mã hành động (OP code - Operation Code) và có hoặc không có số liệu ở vùng toán hạng dùng cho lệnh.

Số liệu tức thì ghi sau mã hành động.

Số liệu ghi trong các thanh ghi R.

Số liệu ghi trong các khối nhớ.




**HÌNH 3.2.** Dạng tổng quát của một mã lệnh của VXL 8086.

Dịch chuyển d (địa chỉ tương đối) và số liệu trực tiếp (địa chỉ cho ô nhớ) có thể là 8 hoặc 16 bit phụ thuộc vào loại lệnh. Mã lệnh và chế độ địa chỉ nằm ở trong một hay hai byte đầu tiên của một lệnh (hình 3.3). Các byte của lệnh và chế độ địa chỉ được theo sau bởi:

- Không có byte phụ thuộc nào.

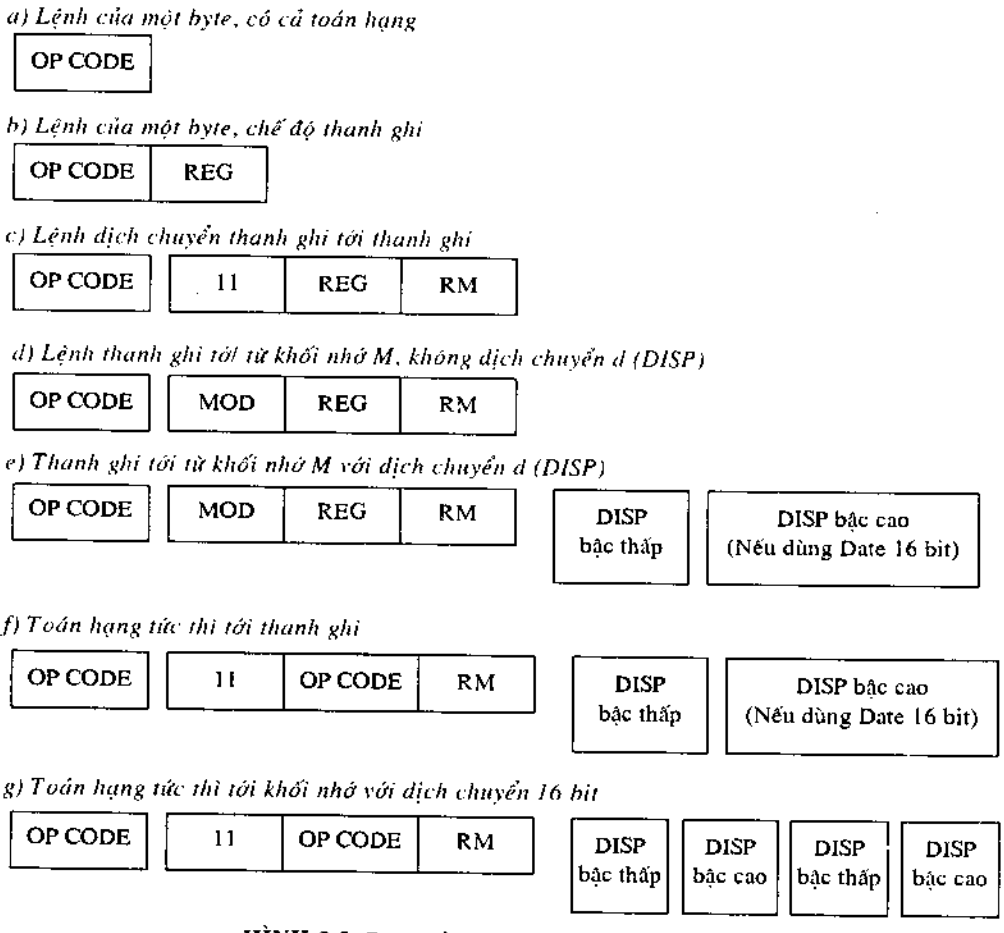
- Hai byte EA (Effective Address - địa chỉ hiệu dụng) chỉ cho địa chỉ trực tiếp.

- Một hoặc hai byte toán hạng tức thì (Immediate operand).

- Một hoặc hai byte dịch chuyển và hai địa chỉ đoạn (cho địa chỉ hoá giữa các đoạn trực tiếp - direct intersegment addressing).

Trên đây là các khả năng được dùng để xác định mã hành động và chế độ địa chỉ hóa. Nếu một địa chỉ hoặc một toán hạng tức thì có độ dài 2 byte thì byte bậc thấp luôn luôn xuất hiện trước tiên.

***a) Lệnh một byte:*** chỉ có mã lệnh, không có toán hạng hoặc một toán hạng ở thanh ghi (hình 3.3a).




**HÌNH 3.3.** Dạng của một lệnh của VXL 8086.

***b) Lệnh hai byte:*** lệnh có hai toán hạng (hình 3.3b) và chế độ địa chỉ (bảng 3.1) như sau:

- **MOD = 11** cho hai bit cao của byte thứ hai, toán hạng thứ nhất ở trong một thanh ghi, toán hạng còn lại ở trong thanh ghi hay ô nhớ.

- **MOD = 00** cho hai bit cao byte thứ hai, toán hạng ở trong thanh ghi gián tiếp với các thanh ghi cơ sở (BX, BP), chỉ thị (DI, SI) và không có dịch chuyển (d = 0) hay địa chỉ tuyệt đối.

***c) Lệnh ba - bốn byte:*** là lệnh có

- **dịch chuyển (d ≠ 0)** hay tương đối với d = 8 bit (3 byte) hay 16 bit (4 byte) (hình 3.3e);
- **toán hạng tức thì** (số liệu tức thì sau byte lệnh với 8 bit (3 byte) hay 16 bit (4 byte)).

***d) Lệnh năm - sáu byte:*** là lệnh có cả dịch chuyển (d ≠ 0) và số liệu tức thì (hình 3.3g).

## 2. Mã lệnh chế độ địa chỉ một byte

Mã hành động thường chiếm byte thứ nhất của một lệnh. Nhưng có một số lệnh trong đó một chỉ định (designation) thanh ghi cũng ở byte thứ nhất cùng với mã lệnh (hình 3.5) và một số lệnh khác, còn có ba bit mã hành động ở byte thứ hai.

Trong đa số mã hành động có một bit chỉ thị đặc biệt, đó là:

- **Bit W:** chỉ độ dài toán hạng hoặc **một byte** (W = 0) hoặc **một lời** (W = 1).
- **Bit D:** chỉ **từ thanh ghi (D = 0)** hay **tới thanh ghi (D =1)**, chỉ dùng cho lệnh có hai toán hạng trong đó có một thanh ghi. Toán hạng còn lại nằm trong khối nhớ M với các phép tính địa chỉ khác nhau hoặc thanh ghi khác.

Với toán hạng nằm trong thanh ghi ở bảng 3.1:

+ Giá trị D = 0 chỉ toán hạng nguồn (Source).
+ Giá trị D = 1 chỉ toán hạng đích (Destination).
+ Bit S (Sign) chỉ mở rộng dấu (Sign Extention). Bit S này xuất hiện tức thì cùng với bit W trong các lệnh cộng, trừ và so sánh thanh ghi, bộ nhớ và ký hiệu như sau:

***Bảng 3.1.*** **Mã địa chỉ các thanh ghi**

| Mã địa chỉ thanh ghi REG | W = 1 | W = 0 | Mã địa chỉ thanh ghi do $S_4S_3$ | Thanh ghi đoạn |
|---|---|---|---|---|
| 000 | AX | AL | 00 | ES |
| 001 | CX | CL | 01 | CS |
| 010 | DX | DL | 10 | SS |
| 011 | BX | BL | 11 | DS |
| 100 | SP | AH | | |
| 101 | BP | CH | | |
| 110 | SI | DH | | |
| 111 | DI | BH | | |

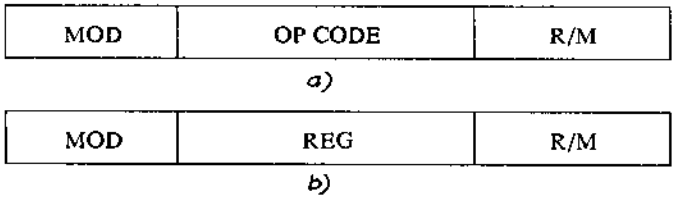
Để chỉ thị thanh ghi được dùng, có các mã như sau:

- Hai bit cho thanh ghi đoạn.
- Ba bit cho các thanh ghi khác.
  + Hành động 8 bit có S : W = 00.
  + Hành động 16 bit với toán hạng tức thì có S : W = 01.
  + Hành động 16 bit với toán hạng có dấu mở rộng 8 bit có S : W = 11.

Với con số nhỏ chỉ dùng đến toán hạng một byte tức thì, bit V được dùng cho lệnh dịch chuyển (Shift) và quay (Rotate) để xác định số lần dịch. Bit Z được dùng cho tiếp đầu ngữ REP.

## 3. Lệnh hai và nhiều byte

Byte thứ hai có một trong hai dạng như hình 3.4.




**Hình 3.4.** Dạng byte thứ hai của mã lệnh.

*Dạng đầu tiên* dùng cho lệnh một toán hạng (hoặc lệnh có hai toán hạng nhưng một trong số chúng được biểu thị bởi mã lệnh OP CODE).

*Bảng 3.2.* **Dạng địa chỉ của các toán hạng**

| MOD = 11 Tính địa chỉ hiệu dụng EA | | | | | | |
|---|---|---|---|---|---|---|
| R/M | W = 0 | W = 1 | R/M | MOD = 00 | MOD = 01 | MOD = 10 |
| 000 | AL | AX | 000 | (BX)+(SI) | (BX)+(SI)+d8 | (BX)+(SI)+d16 |
| 001 | CL | CX | 001 | (BX)+(DI) | (BX)+(DI)+d8 | (BX)+(DI)+d16 |
| 010 | DL | DX | 010 | (BP)+(SI) | (BP)+(SI)+d8 | (BP)+(SI)+d16 |
| 011 | BL | BX | 011 | (BP)+(DI) | (BP)+(DI)+d8 | (BP)+(DI)+d16 |
| 100 | AH | SP | 100 | (SI) | (SI)+d8 | (SI)+d16 |
| 101 | CH | BP | 101 | (DI) | (DI)+d8 | (DI)+d16 |
| 110 | DH | SI | 110 | Trực tiếp | (BP)+d8 | (BP)+d16 |
| 111 | BH | DI | 111 | (BX) | (BX)+d8 | (BX)+d16 |

*Dạng thứ hai* dùng trong lệnh hai toán hạng, trong đó REG chỉ thanh ghi là toán hạng nguồn (D = 0) hay đích (D = 1). Toán hạng chỉ định bởi vùng MOD và R/M được xác định theo bảng 3.2.

Lệnh nhiều hơn hai byte là lệnh có toán hạng là số liệu 16 bit và địa chỉ ô nhớ trực tiếp (số 16 bit) hoặc thanh ghi gián tiếp (cơ sở, chỉ thị, cơ sở chỉ thị) có hay không có dịch chuyển.

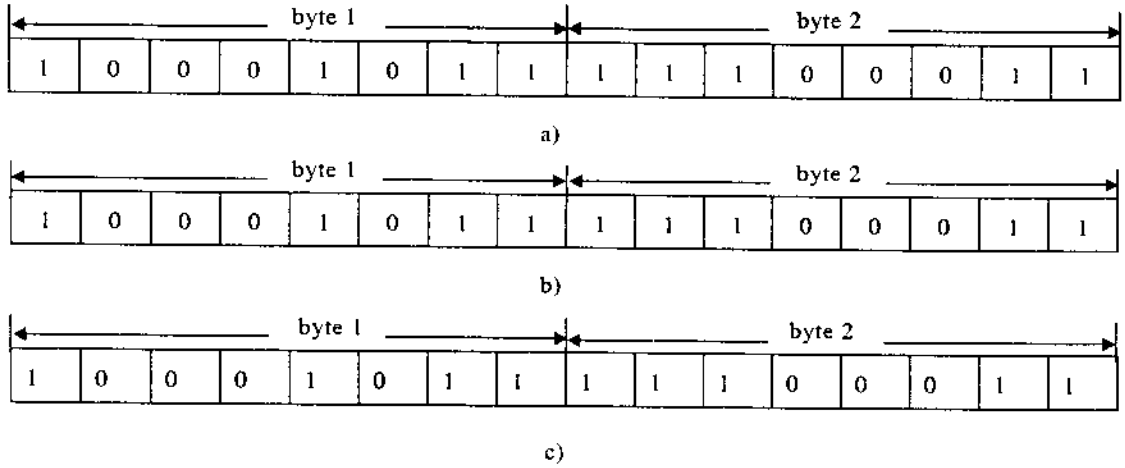
## 4. Ví dụ về lệnh, dạng mã

Để minh họa cho lệnh dạng mã, ta xét mã của một số lệnh như hình 3.5.

### *a) Lệnh MOV SP, BX*

Chuyển nội dung thanh ghi BX vào thanh ghi SP có mã lệnh:

- Mã lệnh MOV = 100010
- D = 1 : gửi tới thanh ghi
- W = 1 : chuyển W- word - lời
- MOD = 11 : thanh ghi tới thanh ghi
- REG = 100 : thanh ghi SP
- R/M = 011 : thanh ghi BX.




**Hình 3.5.** Mã lệnh của các lệnh MOV SP, BX (a), MOV CL, [BX] (b) và MOV 43H [SI],DH (c).

*b) Lệnh MOV CL, [BX]*

Chuyển nội dung ô nhớ có địa chỉ là nội dung của thanh ghi BX vào thanh ghi CL có mã lệnh 1000.1010.0000.1111B = 8A0Fh.

- Mã lệnh MOV = 100010
- D = 1 : tới thanh ghi
- W = 0 : chuyển byte
- MOD = 00 : khối nhớ không dịch chuyển
- REG = 001 : thanh ghi CL
- R/M = 111:[BX].

*c) Lệnh MOV CX, 43H[SI], DH*

Chuyển nội dung thanh ghi DH vào ô nhớ có địa chỉ DS : SI + 43H, có mã lệnh là 1000.1000.0111.0100.0100.0011B = 887443h.

- Mã lệnh MOV = 100010
- D = 0 : từ thanh ghi
- W = 0 : chuyển byte
- MOD = 011 : ô nhớ, dịch chuyển một byte
- REG = 110 : thanh ghi DH
- R/M = 100 : [SI]
- Dịch chuyển d = 0100.0011B.

Bảng 3.3 giới thiệu lệnh chuyển số liệu MOV với các phương pháp địa chỉ khác nhau.

***Bảng 3.3.* Dạng lệnh MOV với các chế độ địa chỉ khác nhau**

| MOD / R/M | 00 (Không dịch chuyển d) | 01 (Dịch chuyển d = 8 bit) | 10 (Dịch chuyển d = 16 bit) | 11 | |
|---|---|---|---|---|---|
| | | | | W = 0 | W = 1 |
| 000 | (BX) ← (SI) | (BX) ← (SI) + D8 | (BX) ← (SI) + D16 | AL | AX |
| Thanh ghi đoạn | DS | DS | DS | | |
| 001 | (BX) ←(DI) | (BX) ← (DI) + D8 | (BX) ← (DI) + D16 | CL | CX |
| Thanh ghi đoạn | DS | DS | DS | | |
| 010 | (BP) ← (SI) | (BP) ← (SI) + D8 | (BP) ←(SI) + D16 | DL | DX |
| Thanh ghi đoạn | SS | SS | SS | | |
| 011 | (BP) ← (DI) | (BP) ←(DI) + D8 | (BP) ← (DI) + D16 | BL | BX |
| Thanh ghi đoạn | ES | ES | ES | | |
| 100 | (SI) | (SI) ← D8 | (SI) ←D16 | AH | SP |
| Thanh ghi đoạn | DS | DS | DS | | |
| 101 | (DI) | (D1) ← D8 | (D1) ← D16 | CH | BP |
| Thanh ghi đoạn | DS | DS | DS | | |
| 110 | D16 | (BP) + D8 | (BP) + D16 | DH | SI |
| Thanh ghi đoạn | SS | SS | SS | | |
| 111 | (BX) | (BX) ←D8 | (BX) ← D16 | BH | DI |
| Thanh ghi đoạn | DS | DS | DS | | |

## 5. Dạng mã lệnh dài và ngắn

Một số lệnh có cả hai dạng dài và ngắn. Hình 3.6 giới thiệu dạng dài và ngắn của lệnh một toán hạng tức thì vào thanh ghi AX. Vì S :W = 01 có hai byte số liệu trong lệnh ở dạng dài, AX được chỉ định bởi vùng R/M và ở dạng ngắn thì AX được chỉ định bởi mã hành động. Tất cả những lệnh có hai toán hạng (MOV, ADD, ADC, SUB, SBB, CMP, AND, OR, XOR, và TEST) trong đó:

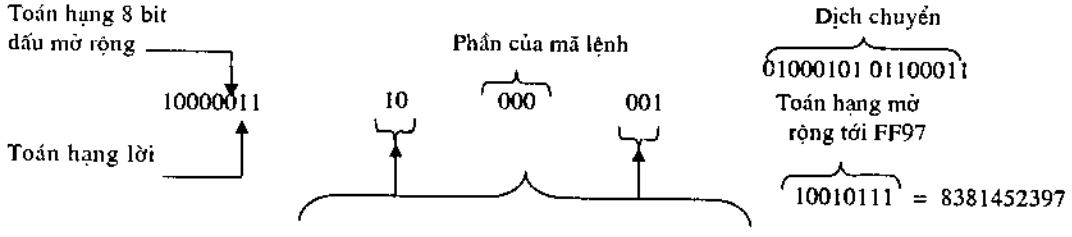
- Toán hạng nguồn là con số tức thì.

- Toán hạng còn lại là thanh ghi AX hoặc AL đều có hai dạng dài và ngắn. Dạng dài dùng AX (cho hành động 16 bit) hoặc dạng ngắn dùng cho AL (cho hành động 8 bit) được chỉ định bởi mã hành động. Hơn nữa lệnh MOV chuyển số liệu có dạng ngắn cho bất kỳ số liệu tức thì nào chuyển vào thanh ghi. Lệnh XCHG (đổi chỗ hai toán hạng) có một dạng ngắn nếu sự đổi chỗ là giữa thanh ghi AX hoặc AL. Một số lệnh chỉ có một toán hạng (ví dụ như INC tăng một đơn vị, DEC giảm bớt một đơn vị, PUSH giữ vào ngăn xếp hay POP lấy ra từ ngăn xếp) có dạng ngắn và nếu toán hạng là thanh ghi.

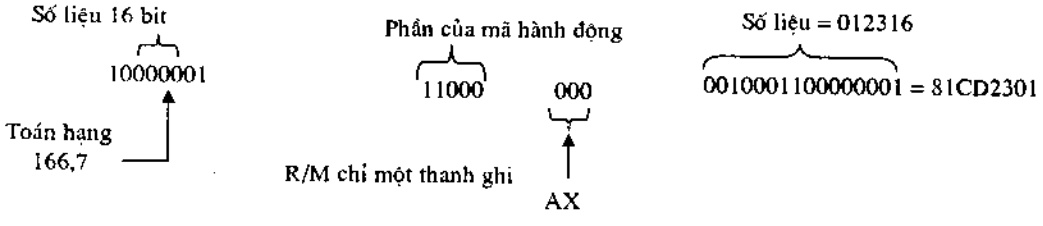
Trong mọi trường hợp, dạng ngắn là ít hơn một byte so với dạng dài tương ứng và đôi khi gần một byte. Như mọi quy luật chung, nếu có thì hợp ngữ (Assembly) luôn viết dưới dạng ngắn. Ví dụ: CMP AX, 5A2H (hình 3.5).

## 6. Mã lệnh ghi trong khối nhớ

Các mã lệnh của chương trình được ghi trong khối nhớ M với những địa chỉ xác định. Vì mã lệnh có độ dài lớn hơn byte nên phải ghi vào nhiều địa chỉ nhớ liên tiếp theo địa chỉ tăng dần từ byte cao tới byte thấp. Hình 3.6 chỉ lệnh ở hình 3.4a, b và hình 3.5b sẽ xuất hiện nếu chúng được đặt liên tiếp từ địa chỉ 00200. Nếu nội dung bắt đầu từ 0018, thanh ghi sẽ được tăng từ 0080 tới 008E.




a)

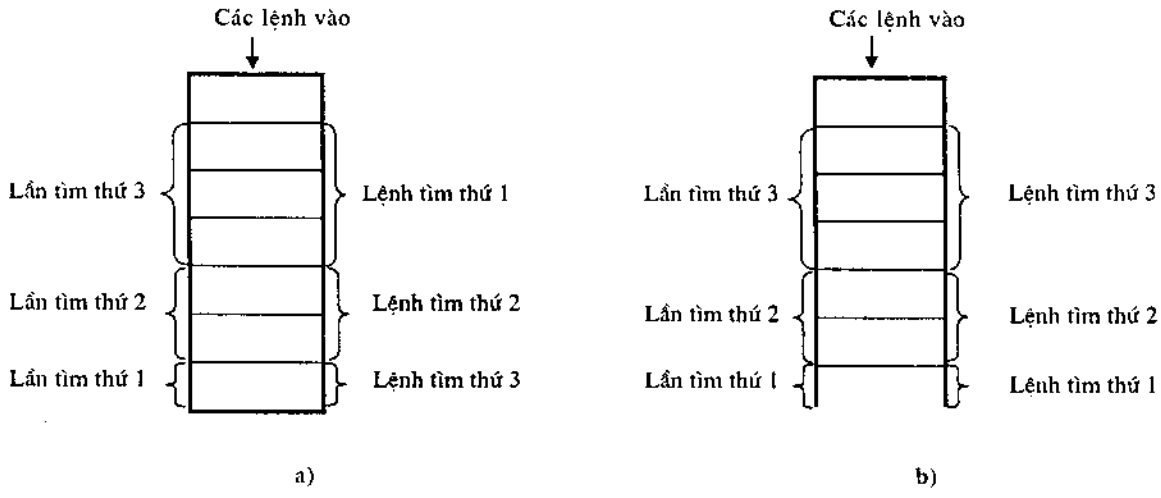



b)

**HÌNH 3.6.** Mã lệnh dạng dài và ngắn của lệnh CMP AX, 5A2H.
a - dạng dài ; b - dạng ngắn.

## 7. Mã lệnh trong hàng đợi lệnh

Trước khi VXL thực hiện mỗi lệnh thì mã lệnh của nó phải được đọc từ khối nhớ và ghi vào hàng đợi lệnh của VXL (8086 có 6 byte cho lệnh dài nhất 6 byte, 8088 có 4 byte). Sau mỗi hành động tìm lệnh, tức VXL đưa địa chỉ ngăn nhớ (là số chẵn - với địa chỉ chẵn và số lẻ với địa chỉ lẻ), các tệp lệnh được đưa vào hàng đợi lệnh. Tùy theo địa chỉ nhớ chẵn hay lẻ mà bố trí các lệnh trong hàng đợi như hình 3.7 cho chuỗi ba lệnh liên tiếp là 1 byte, 2 byte và 3 byte. Mỗi lần tìm hay đọc lệnh, VXL đưa ra địa chỉ của byte nhớ cho lệnh đầu tiên có địa chỉ chẵn, sau đó ba chu kỳ tìm hàng đợi lệnh chứa được 3 lệnh trên (hình 3.7a). Còn trường hợp ngược lại, nếu địa chỉ của lệnh đầu tiên là lẻ, sau ba lần tìm lệnh, chưa nạp hết 6 byte lệnh, lần lượt tìm lệnh thứ tư chưa bắt đầu mà phải chờ tới khi nạp nốt byte cuối cùng của lệnh thứ ba có 3 byte (hình 3.7b).




**HÌNH 3.7.** Lấp đầy hàng đợi lệnh sau một rẽ nhánh:
a - lệnh thứ nhất có địa chỉ chẵn ; b - lệnh thứ nhất có địa chỉ lẻ.

# 3.2. QUÁ TRÌNH THỰC HIỆN MỘT LỆNH

## 3.2.1. TRÌNH TỰ

Một lệnh điển hình được VXL thực hiện theo các bước sau:

- Đọc lệnh từ bộ nhớ vào hàng đợi lệnh.
- Giải mã hành động (OP CODE).
- Tìm các toán hạng nguồn.
- Thực hiện lệnh với các toán hạng nguồn cần thiết.
- Gửi số liệu của kết quả vào toán hạng đích.

Tùy loại lệnh (có toán hạng hay không, một hay hai toán hạng, có cần gửi vào toán hạng đích hay không) mà ta có thể có một số hay tất cả các bước trên và thời gian thực hiện cũng khác nhau.

### 1. Tìm đọc lệnh vào hàng đợi lệnh

Muốn đọc một lệnh ghi ở ô nhớ nào đó (1, 2 hoặc tới 6 byte) ta phải biết được giá trị của hai thanh ghi:

- CS (Code Segment) chỉ địa chỉ bắt đầu của mảng của bộ nhớ chứa mã lệnh.

- IP (Instruction Pointer - con trỏ lệnh) chỉ độ dịch chuyển của địa chỉ ô nhớ chứa lệnh so với địa chỉ gốc trong CS. Nói khác đi cặp (CS, IP) cho biết địa chỉ (vật lý) của ngăn nhớ chứa lệnh (bao gồm cả lệnh và toán hạng).

Bộ cộng của VXL tính địa chỉ ô nhớ trên và đưa giá trị địa chỉ vật lý trên các đường dây địa chỉ $A_0 \div A_{19}$ của VXL. Tùy độ dài lệnh (1 byte tới 6 byte) mà thanh ghi IP tăng tới khi kết thúc một lệnh. Trình tự sắp xếp lệnh trong hàng đợi như hình 3.7

## 2. Giải mã hành động

Mã lệnh (OP CODE) ở byte 1 và byte 2 được đọc vào và giải mã trong bộ phận điều khiển của VXL, bộ phận này tiến hành:

- So sánh mã hành động (OP CODE) với bảng ghi mã lệnh ở một bộ nhớ ngầm của VXL để đọc ra chuỗi hành động tương ứng cho lệnh.
- Đọc bit D để quyết định chiều của số liệu (từ thanh ghi D = 1 và tới thanh ghi D = 0).
- Đọc bit W để quyết định độ dài của số liệu cho toán hạng (W = 1-16 bit, W = 0-8 bit).
- Đọc hai bit MOD để chọn chế độ địa chỉ nhớ (xem hình 3.5 và 3.6).
- Đọc ba bit REG để chọn thanh ghi (hình 3.5) hay mã lệnh mở rộng.
- Đọc ba bit R/M để chọn thanh ghi cho chế độ địa chỉ khác nhau.

## 3. Đọc toán hạng

Toán hạng được đọc từ khối nhớ vào một trong các thanh ghi một hay hai của VXL để thực hiện lệnh do giá trị của địa chỉ có trên đường dây $A_0 \div A_{19}$. Giá trị địa chỉ này được VXL tính toán trên bộ cộng địa chỉ với các số liệu của thanh ghi tùy theo chế độ địa chỉ (hình 3.6).

Nếu lệnh không có toán hạng, động tác này được VXL bỏ qua, lệnh này được thực hiện ngay sau khi giải mã (bước hai ở trên).

Nếu lệnh có hai toán hạng, VXL phải tiến hành đọc toán hạng hai lần.

Nếu lệnh ở chế độ địa chỉ tức thì, tức toán hạng là một con số ghi ngay ở các byte sau của lệnh, VXL sử dụng chúng để thực hiện, không cần đọc toán hạng ở ngăn nhớ.

## 4. Thực hiện lệnh

Sau khi có đầy đủ số liệu cho các toán hạng, bộ điều khiển của VXL đưa ra các tín hiệu xung cho các bộ phận cần thiết (các thanh ghi, khối số học - logic ALU) để thực hiện lệnh.

Nếu lệnh xử lý số học (Cộng, Trừ, Nhân, Chia) hay lệnh logic (Và, Hoặc, Đảo, Loại trừ, Quay, Dịch) thì lệnh chỉ thực hiện trong khối số học-logic (ALU) và trong các thanh ghi nội.

Nếu lệnh nhảy hay rẽ nhánh chương trình có sự đọc và kiểm tra các bit cờ (Flag) của VXL (C, Z, P, S...) và VXL phải tính được địa chỉ mà chương trình rẽ tới. Khi đó, tùy thuộc theo chế độ trong mảng (intrasegment) hay ngoài mảng (intersegment) Mà các thanh ghi CS và IP có các giá trị mới do VXL tính toán địa chỉ rẽ nhánh. Các giá trị của CS và IP này khiến cho VXL đọc lệnh tiếp theo ở vị trí nào đó của ngăn nhớ mà không phải là địa chỉ tiếp theo (do IP tăng dần, CS không đổi) của chương trình.

## 5. Gửi kết quả vào đích

Tùy dạng lệnh trong các phép xử lý số liệu (số học, logic) thường kết quả gửi vào đích của số liệu. Sau các lệnh xử lý trên, số liệu đều có ở thanh ghi chứa AX (hoặc AL) kết quả này có thể được gửi ra thiết bị ngoài, vào vùng nhớ bởi các lệnh bổ sung (vào, ra, ghi).

Nếu lệnh chuyển số liệu, số liệu ở địa chỉ nguồn được gửi tới địa chỉ đích và lưu trữ tại đó (thanh ghi, ô nhớ).

Các quá trình đọc và thực hiện lệnh trên của VXL 8086 được diễn ra trong một chu trình gồm 4 nhịp thời gian (T1 ÷ T4) cho một lệnh không có toán hạng. Nếu lệnh có nhiều byte và có nhiều toán hạng thì cần phải có nhiều chu trình để:

- Đọc các byte lệnh vào hàng đợi.
- Đọc các toán hạng.
- Thực hiện lệnh.

Dưới đây, chúng ta sẽ tính thời gian thực hiện một lệnh cụ thể của VXL 8086.

## 3.2.2. THỜI GIAN THỰC HIỆN CỦA MỘT LỆNH

Thời gian thực hiện của một lệnh có thể được xác định bằng cách nhận số chu trình nhịp (Clock Cycles) cần thiết để thực hiện lệnh với chu trình nhịp. Nó có thể được biểu thị như là tổng số của thời gian thực hiện cơ sở (phụ thuộc vào lệnh và chế độ địa chỉ) cộng thêm thời gian để tính địa chỉ hiệu dụng EA nếu có toán hạng ở bộ nhớ.

Thời gian thực hiện cơ sở là thời gian tìm tính lệnh và hàng đợi lệnh, mặt khác bất kỳ chu trình nhịp bổ sung nào để tìm lệnh phải được cộng thêm. Thời gian thực hiện cơ sở của một số lệnh của 8086/8088 cho ở bảng 3.4 và 3.5.

***Bảng 3.4.* Thời gian thực hiện một số lệnh**

| Lệnh | Số chu trình nhịp | Số dịch chuyển |
|---|---|---|
| *ADD (cộng) hay SUB (trừ)* | | |
| Thanh ghi tới thanh ghi | 3 | 0 |
| Bộ nhớ tới thanh ghi | 0 + EA | 1 |
| Tức thì tới thanh ghi | 16 + EA | 2 |
| Tức thì tới bộ nhớ | 217 + EA | 0 |
| *MOV (chuyển)* | 17 + EA | 2 |
| AX tới bộ nhớ | 10 | 1 |
| Bộ nhớ tới AX | 10 | 1 |
| Thanh ghi tới thanh ghi | 2 | 0 |
| Tức thì tới thanh ghi | 2 | 0 |
| Tức thì tới bộ nhớ | 10 + EA | 1 |
| Thanh ghi tới thanh ghi đoạn | 2 | 0 |
| Bộ nhớ tới thanh ghi mảng | 8 + EA | 1 |
| Thanh ghi mảng tới thanh ghi | 2 | 0 |
| Thanh ghi mảng tới bộ nhớ | 9 + EA | 1 |
| *MUL (nhân không dấu)* | | |
| Số bị nhân thanh ghi 8 bit | 70÷72 | 0 |
| Số bị nhân thanh ghi 16 bit | 118÷133 | 0 |
| Số bị nhân ở bộ nhớ 8 bit | (76÷83) + EA | 1 |
| Số bị nhân ở bộ nhớ 16 bit | (124÷139) + EA | 1 |
| *IMUL (nhân có dấu)* | | |
| Số bị nhân thanh ghi 8 bit | 80÷98 | 0 |
| Số bị nhân thanh ghi 16 bit | 128÷154 | 0 |
| Số bị nhân ở bộ nhớ 8 bit | (86÷104) + EA | 1 |
| Số bị nhân ở bộ nhớ 16 bit | (134÷160) + EA | 1 |

Cột thứ ba của bảng 3.4 chỉ ra số lượng tham chiếu bộ nhớ do lệnh yêu cầu. Để xác định thời gian thực hiện một lệnh, trong đó có sự tham chiếu của bộ nhớ, chúng ta phải chứng minh rằng có đầy đủ toán hạng. Nếu một lời toán hạng được đặt ở địa chỉ lẻ, 8086 phải mất hai chu trình Bus, mỗi chu trình này cần bốn chu trình nhịp để truy cập toán hạng. Do vậy, mỗi lần tham chiếu địa chỉ lẻ, cần bốn chu trình nhịp ngoài. Với 8088 bốn chu trình nhịp sẽ được cộng vào cho mỗi lần chuyển dịch bởi vì 8088 chỉ có thể chuyển được 1 byte cho một chu trình Bus.

Chú ý rằng, một số lệnh có thể có một số thời gian thực hiện cơ sở khác nhau, phụ thuộc vào chế độ địa chỉ. Các ví dụ ở bảng 3.5 chứng tỏ rằng đối với lệnh có hai toán hạng, một hành động thanh ghi tới thanh ghi sẽ nhanh hơn dùng bất kỳ chế độ gọi địa chỉ nào và một hành động khối.

- Điều kiện của lệnh xảy ra sẽ không có rẽ nhánh.

- Điều kiện của lệnh xảy ra có rẽ nhánh. Khi đó phải tìm lệnh tiếp theo để tích cho hàng đợi lệnh.

***Bảng 3.5.* Thời gian cần thiết cho địa chỉ hiệu dụng**

| EA | Số nhịp chu trình |
|---|---|
| Trực tiếp | 6 |
| Thanh ghi gián tiếp | 5 |
| Thanh ghi tương đối | 9 |
| Cơ sở chỉ thị | |
| (BP) + (DI) hay (BX)*(SI) | 7 |
| (BP) + (SI) + DISP | 11 |
| (BX) hoặc (SI) + DISP | |
| (BP) + (DI) + DISP hay 12 | |
| (BX) + (DI) + DISP | |

Ví dụ, nếu nhịp có tần số 5 MHz (chu kỳ 0,2 ms) thì thời gian thực hiện cho các dạng khác nhau của lệnh ADD được tính như sau:

- Cộng thanh ghi với thanh ghi (kết quả đặt trong thanh ghi) đòi hỏi: ba nhịp chu trình cho toán hạng 1 byte hay thời gian là 0,6 ms.

- Cộng khối nhớ tới thanh ghi dùng địa chỉ cơ sở chỉ thị tương đối (kết quả đặt vào thanh ghi) cần: 9 + 12 = 21 chu trình cho toán hạng byte hay lời với lời ở một địa chỉ chẵn. Thời gian sẽ bằng 4,2 ms, 9 + 12 + 4 = 25 chu trình nếu lời ở địa chỉ lẻ, khi đó thời gian là 5,0 ms. Cộng thanh ghi với khối nhớ dùng địa chỉ cơ sở chỉ thị (kết quả đặt vào khối nhớ).

16 + 8 = 24 chu trình cho toán hạng byte hay lời với lời.

Cơ sở Tính địa chỉ

Thời gian = 4,8μs.

6 + 8 + 4 + 4 = 22 chu trình nếu lời ở địa chỉ lẻ.

Chu trình thêm để tìm lời. Chu trình thêm để nạp lời.

Thời gian = 6,4μs.

## 3.3. LỆNH DẠNG HỢP NGỮ

Lệnh dạng mã máy (tổ hợp các bit 0, 1) có ưu điểm là ngắn gọn, VXL hiểu và thực hiện một cách trực tiếp, không cần phải phiên dịch và biên dịch. Nhưng nó cũng có nhược điểm là:

- Khó nhớ và dễ viết sai vì lệnh chỉ toàn các con số (mã) và dài.

- Không đặc trưng cho hành động của lệnh.

Khắc phục nhược điểm trên, người ta đổi tên mỗi mã lệnh bằng một lệnh dạng dễ nhớ (dạng Memonic - thuật nhớ) vì tên lệnh là những chữ cái đầu bằng tiếng Anh của hành động của lệnh, ví dụ lệnh chuyển là MOV (**MOV**ement). Với cách đặt tên lệnh này, người dùng được gợi nhớ đến hành động của lệnh. Còn một ý nghĩa khác của lệnh dạng hợp ngữ là lệnh *tập hợp hành động và các toán hạng trong một câu lệnh*. Mỗi loại VXL khác nhau có một hợp ngữ riêng cho mình. Nhưng một họ VXL của cùng một hãng, ví dụ, họ Intel 86 (8086/8088; 80286; 80386; 80486; 80586) có cùng một hệ lệnh cơ bản như nhau và cùng dạng hợp ngữ. Do đó, ta có thể dùng chương trình viết cho VXL thế hệ trước chạy trên VXL thế hệ sau, nhưng ngược lại, chương trình của VXL thế hệ sau có thể không thực hiện được trên VXL thế hệ trước vì có lệnh bổ sung mới mà thế hệ trước chưa được trang bị.

Nhược điểm cơ bản của lệnh dạng hợp ngữ này là VXL không thể hiểu được một cách trực tiếp mà phải qua quá trình dịch ra mã lệnh hay ngôn ngữ máy bởi chương trình biên dịch hợp ngữ (assembler).

### 3.3.1. DẠNG MỘT CÂU LỆNH HỢP NGỮ

Một câu lệnh ở hợp ngữ có 4 yếu tố sau:

| **Label :** | **Mnemonic** | **Operand, Operand ;** | **Comment** |
|---|---|---|---|
| (Nhãn) | (Thuật nhớ) | (Các toán hạng) | (Ghi chú) |

Một câu lệnh phải viết trên cùng một dòng, ngăn cách giữa nhãn và thuật nhớ bởi dấu ":" (hai chấm) và ngăn cách với phần ghi chú bởi dấu ";"(chấm phẩy). So với dạng mã lệnh, dạng hợp ngữ này có thêm **nhãn (hay Label), ":", ";" và Comment. Câu lệnh dạng hợp ngữ** vẫn có **mã hành động** và **các toán hạng** phân cách nhau bởi dấu "," (phẩy) nhưng thể hiện bằng chữ (cho dễ nhớ) mà không phải bằng con số của mã lệnh.

Mỗi VXL có dạng hợp ngữ riêng của mình, do đó có chương trình thông dịch và biên dịch riêng của mình. Sau đây sẽ trình bày hợp ngữ cho họ MVT PC-IBM và chương trình biên dịch **MASM-86 của hãng Microsoft.**

#### 1. Label (nhãn)

Nhãn là một nhận dạng (Indentify) được gán cho địa chỉ của byte thứ nhất của lệnh. Sự có mặt của nhãn là tùy chọn:

- Có thể không có cho các lệnh không rẽ nhánh.

- Cần có nhãn khi chương trình có các lệnh rẽ nhánh, để VXL biết được chương trình sẽ nhảy (rẽ nhánh) tới nhãn hay địa chỉ của lệnh nào trong chương trình. Nhãn được đặt ở trước của lệnh mà chương trình cần rẽ nhánh tới lệnh đó.

Nhãn hay tên bắt đầu bằng một chữ cái (không bắt đầu bằng số 0) hay chuỗi ký tự có chiều dài từ 1÷31 ký tự (chữ cái, chữ số, ký tự đặc biệt -, ?, @, $...), có thể viết bằng loại chữ bất kỳ vì chương trình biên dịch không phân biệt chữ hoa hay chữ thường.

#### 2. Thuật nhớ

Thuật nhớ là tên của hành động hay toán tử, được viết dưới dạng tượng trưng cho dễ

nhớ. Người ta dùng một số (thường là 3) chữ cái đầu tiên hoặc viết tắt tên lệnh của hành động viết bằng tiếng Anh.

*Ví dụ:* MOV (MOVement): chuyển
ADD (ADDition): cộng
SUB (SUBstration): trừ
JMP (JuMP): nhảy.

## 3. Toán hạng

Toán hạng chứa số liệu cho hành động, một lệnh có thể:

- *Không có toán hạng:* nếu hành động không cần toán hạng.

*Ví dụ:* HLT: là lệnh dừng chương trình.

NOP: là lệnh không hành động gì cả.

- *Có một toán hạng:* khi hành động cần một toán hạng.

*Ví dụ:* INC CX: tăng nội dung của thanh ghi CX lên một đơn vị.

Giữa các lệnh dạng thuật nhớ và toán hạng có một khoảng cách nào đó. Nếu không VXL sẽ hiểu đó chỉ là tên lệnh.

- *Có hai toán hạng:* dành cho lệnh cần hai nguồn số liệu và toán hạng đích luôn luôn được viết trước, còn toán hạng nguồn được viết sau. Giữa hai toán hạng ngăn cách bởi dấu phẩy (",").

***Bảng 3.6.* Các dạng toán hạng cho các chế độ địa chỉ**

| Chế độ địa chỉ | Dạng | Ví dụ |
|---|---|---|
| *Liên quan đến số liệu* | | |
| Tức thì | Biểu thức hằng số | 10110B<br>XY = XX+5 |
| Trực tiếp | Biểu thức | CNT - 5<br>ARRY + 5 |
| Thanh ghi | Thanh ghi | AX<br>DH |
| Thanh ghi gián tiếp | Thanh ghi | [BH] |
| Thanh ghi tương đối | Biến đổi (thanh ghi ± biểu thức hằng số hoặc thanh ghi ± hằng số) | VAR X(BX)<br>(BX-1) |
| Cơ sở chỉ thị | (thanh ghi cơ sở) (thanh ghi chỉ thị) | (BP)(DI) |
| *Liên quan tới địa chỉ rẽ nhánh* | | |
| Intrasegment direct | Label ± biểu thức hằng | OVER ERROR+7 |
| Intrasegment indirect | Như dạng với số liệu trừ địa chỉ tức thời | |
| Intrasegment direct | Label ± biểu thức hằng | Branch EXIT |
| Intrasegment indirect | Như dạng với số liệu trừ địa chỉ tức thì và thanh ghi | |

*Ví dụ:* - MOV AX, BX: chuyển nội dung số liệu thanh ghi từ thanh ghi BX vào thanh ghi AX.

- ADD AX, 1232H: cộng nội dung AX với số 1232H.

Toán hạng đích sẽ thay đổi sau khi thực hiện lệnh vì nó chứa kết quả của lệnh, còn nội dung của toán hạng nguồn sẽ không thay đổi.

Dạng của toán hạng còn phụ thuộc vào chế độ địa chỉ (bảng 3.6). Trong địa chỉ tức thì con số có thể viết dưới dạng:

- *Chữ D hay không có ký hiệu gì,* con số dưới dạng thập phân.

- *Chữ B:* dạng nhị phân (Binary) hay cơ số 2.

- *Chữ O:* dạng cơ số 8 (Octal).

- *Chữ H:* dạng cơ số 16 (Hexadecimal).

Với tất cả các mục đích, chế độ địa chỉ tức thì, biểu thức hằng số có thể có hoặc không.

### 4. Ghi chú

Vùng này dùng để ghi chú, tức giải thích cho hành động của lệnh hoặc dành cả dòng cho ghi chú (không có nhãn, lệnh hoặc toán hạng) mà dấu ";" (chấm phẩy) ở ngay vùng nhãn. Khi dịch, các chữ ghi chú này được bỏ qua. Vùng ghi chú này rất cần thiết cho người viết và người đọc chương trình để dễ theo dõi chương trình nhưng không cần thiết cho máy. Đôi khi người ta dùng dấu ";" mà không có phần chữ ghi chú, để tạo ra các dòng trống trong chương trình cho dễ đọc.

*Ví dụ:*

```
                    ; khởi tạo các thanh ghi
        MOV AX, 00h ; xoá AX
        MOV BX, 00h ; xoá BX
        MOV CX, 00h ; xoá CX.
```

## 3.3.2. DỊCH DÒNG LỆNH HỢP NGỮ

Lệnh hợp ngữ này phải được MVT dịch ra mã lệnh để thực hiện khi chạy chương trình bằng hợp ngữ. Ở đây chỉ xét việc dịch một dòng lệnh dạng hợp ngữ.

**Label** (nhãn) với dấu hai chấm **":"** khi dịch được thay bởi địa chỉ của byte đầu tiên của lệnh, để đánh dấu vị trí cho lệnh rẽ nhánh tới đó.

Lệnh cùng toán hạng được dịch thành mã lệnh dạng ngôn ngữ máy bằng cách tìm ở bảng đối chiếu ra mã lệnh của chương trình dịch đã ghi sẵn trong bộ nhớ.

Ghi chú cùng dấu chấm phẩy ";" không dịch mà chỉ báo cho chương trình dịch biết đã dịch xong một dòng lệnh.

# 3.4. CÁC CHẾ ĐỘ ĐỊA CHỈ CỦA TOÁN HẠNG

## 3.4.1. VẤN ĐỀ GỌI ĐỊA CHỈ CỦA TOÁN HẠNG

Trừ một số ít lệnh không cần toán hạng, đa số lệnh của VXL cần toán hạng tức dữ liệu cho lệnh. Như đã giới thiệu ở phần trước, dữ liệu đó có thể là số liệu cho các phép toán số học hay logic hoặc địa chỉ cho hành động nhảy chương trình. Dữ liệu trên có thể tìm thấy **trong số liệu tức thì** ghi ngay trong lệnh, **trong thanh ghi** và **trong ô nhớ.** VXL sẽ gọi các dữ liệu trên bằng cách phát địa chỉ để đọc các toán hạng vào các thanh ghi của VXL. Mỗi dòng lệnh chỉ có một mã lệnh tức một hành động duy nhất nhưng toán hạng tức số liệu cho lệnh có thể đọc, tìm ở trong ba nguồn số liệu trên, đặc biệt trong ô nhớ có nhiều phương pháp gọi tìm số liệu (trực tiếp, gián tiếp, thanh ghi cơ sở, chỉ số v.v...). Chính sự phong phú của các phương pháp hay chế độ gọi địa chỉ này mà số lệnh mở rộng của một VXL rất lớn tuy rằng số lệnh cơ bản có hạn. Người lập trình có thể thao tác với rất nhiều nguồn số liệu, không cần các lệnh tìm và chuyển số liệu trung gian. Người lập trình hợp ngữ phải nắm vững các **phương pháp và cách thức (phương thức) gọi địa chỉ** hay **chế độ địa chỉ** (addressing mode) để tìm toán hạng cho hành động của lệnh.

## 3.4.2. VIỆC TÍNH VÀ PHÁT ĐỊA CHỈ CỦA VI XỬ LÝ

### 1. Quản lý địa chỉ bộ nhớ của Intel 86

Các VXL Intel 86 quản lý bộ nhớ theo đoạn (segment) hay mảng và trang (page). Các thanh ghi đoạn quản lý các vùng nhớ dành cho các loại dữ liệu khác nhau. Đó là các thanh ghi mã lệnh CS, thanh ghi số liệu DS, thanh ghi số liệu phụ ES, thanh ghi stack SS.

- *Thanh ghi đoạn CS*: quản lý mảng (hay đoạn) nhớ ghi mã lệnh của chương trình. Giá trị của CS chỉ địa chỉ đầu tiên của mảng chứa mã lệnh. Để chỉ địa chỉ khác nhau trong mảng nhớ, VXL dùng thanh ghi **con trỏ lệnh IP** (instruction pointer). Như vậy, cặp thanh ghi CS và IP dùng để biểu diễn một địa chỉ của ô nhớ nằm trong mảng có địa chỉ đầu tiên là **CS** và có địa chỉ cách địa chỉ đầu tiên của mảng là giá trị **IP.** Cặp thanh ghi CS, IP biểu diễn một địa chỉ của ô nhớ trên, được gọi là **địa chỉ logic** và được viết là **CS:IP.** Vì đã **gán cố định CS cho mảng mã lệnh** cho nên, để cho gọn, người ta chỉ ghi địa chỉ mã lệnh là **IP** với **ngầm định** là địa chỉ ban đầu của mảng ở **CS.** Nếu mã lệnh ở vùng khác, ta phải dùng và viết cả cặp thanh ghi mảng và thanh ghi chỉ địa chỉ tương đối trong mảng.

- *Thanh ghi đoạn DS:* quản lý mảng số liệu, cũng với quy ước như CS. Để chỉ địa chỉ tương đối trong mảng số liệu này, người ta có thể dùng một trong ba thanh ghi BX, SI và DI.

+ **BX:** nội dung của nó chỉ **địa chỉ cơ sở** của số liệu so với địa chỉ bắt đầu của mảng số liệu DS, ES, FS, GS trong mảng số liệu ở chế độ địa chỉ cơ sở.

+ **SI:** nội dung của nó chỉ **địa chỉ tương đối** của ô nhớ chứa **số liệu nguồn** (Source) so với địa chỉ gốc **DS** trong **chế độ địa chỉ chỉ thị**, tức **số liệu lấy ra** để thực hiện các phép xử lý (số học hay logic) hoặc di chuyển cả một vùng nhớ.

+ **DI:** nội dung của nó chỉ **địa chỉ tương đối** của ô nhớ chứa **số liệu đích** (Destination) so với gốc **DS, ES, FS, GS** trong **chế độ địa chỉ chỉ thị**, tức **số liệu gửi vào** sau khi đã xử lý hoặc chuyển cả một vùng nhớ. Vì chỉ có tối đa hai toán hạng trong một lệnh nên **DI** còn chứa toán hạng thứ hai để đưa vào xử lý cho phép toán cần hai toán hạng và số liệu của toán hạng thứ hai này bị thay thế bởi kết quả xử lý sau khi thực hiện xong lệnh có hai toán hạng.

- *Các thanh ghi đoạn ES, FS, GS*: quản lý các mảng số liệu phụ ngoài DS. Thanh ghi ES thường được dùng trong hai trường hợp sau:

+ **ES** : quản lý dữ liệu là chuỗi ký tự (string).

+ **ES:DI** : quản lý đích số liệu trong các hành động chuyển số liệu từ nguồn do **DS:SI** quản lý.

Như vậy, tổ hợp của các thanh ghi đoạn số liệu (DS, ES, FS, GS) và các thanh ghi cơ sở BX, chỉ thị SI, DI để tạo thành các địa chỉ logic cho số liệu theo các chế độ địa chỉ:

+ Cơ sở : (DS, ES, FS, GS) : BX.

+ Chỉ thị : (DS, ES, FS, GS) : (SI, DI).

+ Thanh ghi stack SS : quản lý vùng nhớ stack (ngăn xếp) để lưu trữ tạm thời số liệu, đặc biệt là các giá trị của các thanh ghi trong khi chạy chương trình. Giá trị của SS chỉ địa chỉ đầu tiên của vùng nhớ stack. Có hai thanh ghi chỉ địa chỉ tương đối trong vùng nhớ stack là:

+ Thanh ghi SP: nội dung của nó chỉ *địa chỉ của ô nhớ stack trong mảng có địa chỉ đầu tiên SP*, nhưng so với đỉnh của mảng nhớ stack.

+ Thanh ghi BP: chỉ *địa chỉ của ô nhớ stack trong mảng có địa chỉ tương đối BP so với địa chỉ gốc SS của mảng* (tương tự BX, SI, DI).

Như vậy, ta có các địa chỉ logic của vùng nhớ stack như sau:

+ Cặp thanh ghi **SS, SP** chỉ địa chỉ logic **SS:SP** cho ô nhớ của vùng stack **kể từ đỉnh.**

+ Cặp thanh ghi **SS, BP** chỉ địa chỉ logic **SS:BP** cho ô nhớ của vùng nhớ stack **kể từ gốc.**

Tóm lại, giá trị của các thanh ghi đoạn (CS, DS, ES, FS, GS) chỉ địa chỉ gốc hay địa chỉ của ô nhớ đầu tiên của mảng nhớ. Giá trị của các thanh ghi IP, BX, SI, DI, BP, SP chỉ địa chỉ tương đối trong mảng. Công thức tính địa chỉ logic như sau:

**Địa chỉ logic = thanh ghi đoạn : offset.**

Trên đây là các địa chỉ logic cơ bản của các mảng nhớ, ở phần các phương pháp địa chỉ dưới đây, chúng ta sẽ xét thêm các địa chỉ mở rộng khác.

## 2. Việc tính địa chỉ hiệu dụng EA và phát địa chỉ vật lý PA

Địa chỉ hiệu dụng EA (effective addresse) là địa chỉ của ô nhớ so với địa chỉ đầu tiên của mảng nhớ. Đó là các giá trị của các thanh ghi chỉ địa chỉ tương đối trong mảng IP, BX, SI, DI, SP, BP nếu là phương pháp địa chỉ tuyệt đối và còn cộng thêm một dịch chuyển d (displacement) vào EA để có địa chỉ hiệu dụng tương đối. Địa chỉ tương đối trong mảng còn có thể là một con số và chung cả hai trường hợp (thanh ghi, con số) gọi chung là **offset** (độ lệch). Với các phương pháp địa chỉ phức tạp có nhiều thanh ghi chỉ địa chỉ tương đối trong mảng, địa chỉ lệch (offset) là tổng các thanh ghi đó. Địa chỉ hiệu dụng EA tính theo công thức:

**EA = offset + d.**

Địa chỉ hiệu dụng EA chỉ có tác dụng đối với việc viết lệnh dạng hợp ngữ trong chương trình. Khi viết lệnh, ta chỉ viết địa chỉ hiệu dụng EA cho toán hạng, chương trình dịch sẽ hiểu và dịch kèm theo giá trị của thanh ghi mảng ngầm định (CS, DS, ES, SS, FS, GS) cho mỗi mảng nhớ. Thanh ghi mảng không có mặt trong lệnh, được ngầm định là một trong các thanh ghi đoạn trên.

Vi xử lý khi đọc đến **địa chỉ EA,** phải tính ra **địa chỉ vật lý PA** (physic address) hay **địa chỉ thực** để phát ra các tín hiệu cho các đường dây địa chỉ, để đọc toán hạng trong khối nhớ. **Bộ lấy tổng địa chỉ** trong các VXL Intel 86 làm nhiệm vụ tính địa chỉ vật lý này. Vì các thanh ghi địa chỉ lệch chỉ có 16 bit mà số đường địa chỉ tới 20, nên giá trị của thanh ghi đoạn phải dịch về bên trái 4 bit nhị phân (cho tới 16 + 4 = 20) hay một con số 0H của hệ đếm 16.

Ta có:

**PA = (thanh ghi đoạn)x$2^4$+ offset** trong hệ đếm 10 (D)

hay **PA = (thanh ghi đoạn)0h+ offset** trong hệ đếm 16 (H).

## 3.4.3. CÁC PHƯƠNG PHÁP ĐỊA CHỈ CỦA TOÁN HẠNG

Cách thức trong đó toán hạng được xác định gọi là phương pháp hay chế độ địa chỉ. Có hai loại:

- Cho số liệu với những lệnh có liên quan đến số liệu.
- Cho địa chỉ rẽ nhánh đối với những lệnh điều khiển rẽ nhánh chương trình.

### 1. Chế độ gọi địa chỉ cho số liệu (datum)

***a) Quy tắc gọi địa chỉ số liệu***

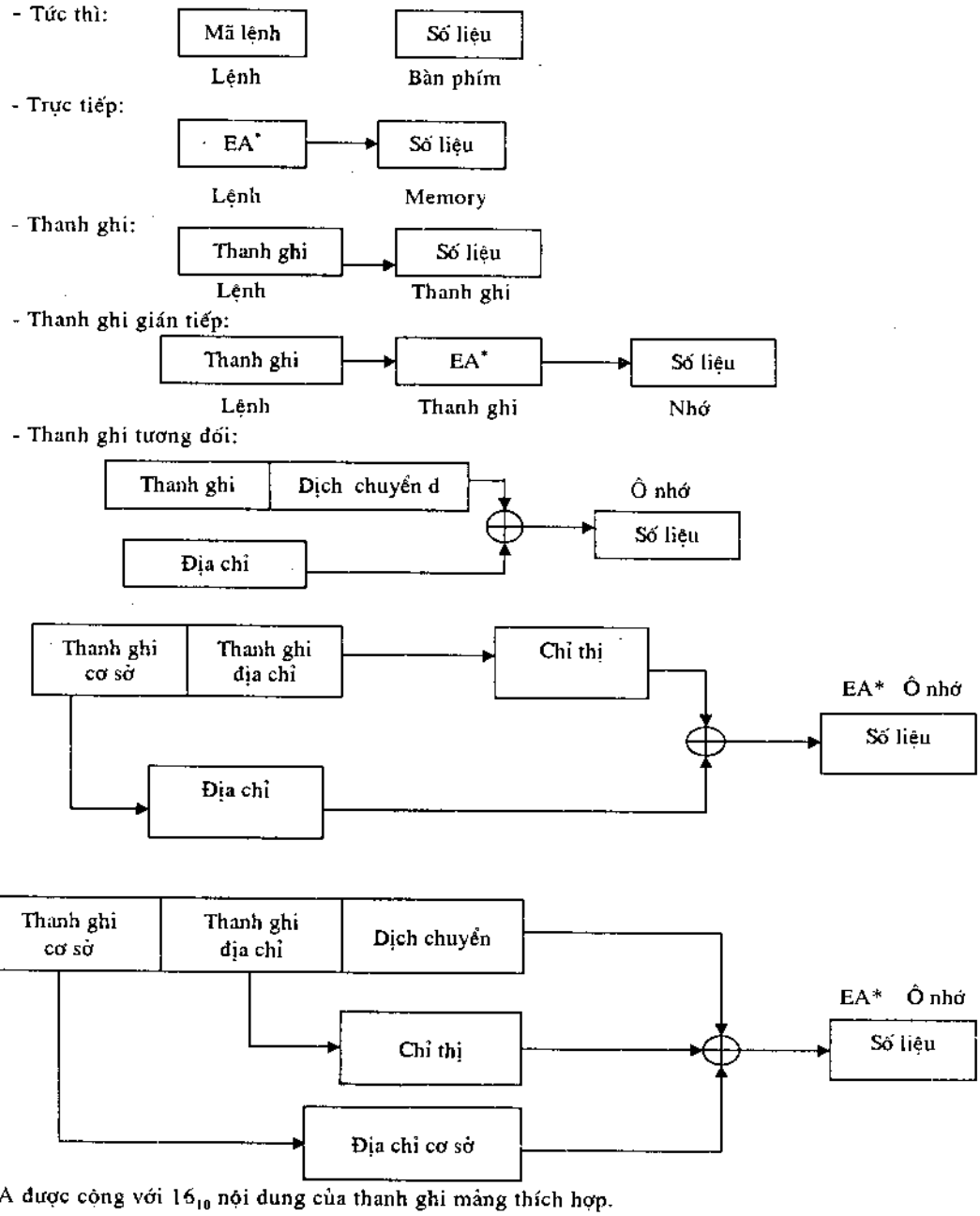
Số liệu của toán hạng có thể là:

- Một con số tức thì ngay sau mã lệnh.
- Nội dung của thanh ghi thông dụng.
- Nội dung của ô nhớ.

Do đó, cũng có ba phương pháp địa chỉ toán hạng.

* *Phương pháp địa chỉ toán hạng tức thì* (Immediate Operation Addressing Mode)

Số liệu (8, 16 hoặc 32 bit) của toán hạng là một số được viết trong một byte - 8 bit (Imm8), hai byte - 16 bit (Imm16) hay bốn byte - 32 bit (Imm32) liền ngay sau mã lệnh của lệnh (hình 3.8).




*EA được cộng với $16_{10}$ nội dung của thanh ghi mảng thích hợp.

**Hình 3.8.** Các chế độ chỉ thị liên quan tới số liệu.

*Ví dụ:* MOV AL, 15h ; chuyển số 15h vào thanh ghi AL.

Ở ví dụ trên, lệnh chuyển MOV đã phải dùng hai **toán hạng nguồn** (15h ở **chế độ địa chỉ tức thì**) và **toán hạng đích** ở chế độ địa chỉ thanh ghi. Toán hạng tức thì này được ghi ở một byte của lệnh ngay sau mã lệnh của lệnh MOV (11000110).

* *Phương pháp địa chỉ toán hạng thanh ghi* (Register Operand Addressing Mode)

Số liệu của toán hạng nằm trong thanh ghi được lệnh xác định bằng mã quy ước cho thanh ghi:

- Nếu toán hạng là một số có 8 bit thì thanh ghi có thể là byte thấp hay cao của các thanh ghi thông dụng, tức là AH, AL, BH, BL, CH, CL, DL, DH.

- Nếu toán hạng là một số có 16 bit, các thanh ghi có thể là: AX, BX, CX, DX, SI, DI, SP, BP.

- Nếu toán hạng là 32 bit thì thanh ghi phải là thanh ghi mở rộng (của 80386) EAX, EBX, ECX, EDX, v.v... hoặc thanh ghi kép DX:AX và CX:BX cho VXL 8086.

Bảng 3.7 chỉ dẫn các độ lớn số liệu của phương pháp địa chỉ toán hạng thanh ghi.

*Ví dụ:* MOV AX, BX ; chuyển số liệu của thanh ghi BX vào thanh ghi AX.

Ở lệnh trên có hai toán hạng thanh ghi AX và BX và chúng ta cần lưu ý là cả hai toán hạng phải cùng độ lớn. Nếu chuyển thanh ghi có độ lớn vào thanh ghi có độ lớn nhỏ hơn, các bit cao sẽ bị mất, còn ngược lại, các bit cao của thanh ghi đích không được chuyển và vẫn giữ nguyên như trước khi chuyển.

Phương pháp địa chỉ hoá toán hạng thanh ghi này có dạng lệnh và thời gian thực hiện nhỏ hơn phương pháp địa chỉ toán hạng ô nhớ vì VXL không phải tính và phát địa chỉ cho khối nhớ.

* *Phương pháp gọi địa chỉ toán hạng khối nhớ* (Memory Operand Addressing Mode)

Phương pháp này tìm số liệu của toán hạng ở ô nhớ của khối nhớ. Muốn vậy, ngoài mã lệnh, lệnh cần phải ghi địa chỉ logic để VXL tính địa chỉ hiệu dụng và địa chỉ vật lý cho ô nhớ cần tìm toán hạng.

***Bảng 3.7.* Các độ lớn số liệu cho địa chỉ thanh ghi**

| Thanh ghi | Kích thước Byte (Reg 8) | Toán hạng Word (Reg 16) | Double word (Reg 32) |
|---|---|---|---|
| Tích lũy | *AL, AH* | *AX* | *EAX* |
| Cơ sở | *BL, BH* | *BX* | *EBX* |
| Đếm | *CL, CH* | *CX* | *ECX* |
| Số liệu | *DL, DH* | *DX* | *EDX* |
| Con trỏ ngăn xếp | - | *SP* | *ESP* |
| Con trỏ cơ sở | - | *BP* | *EBP* |
| Chỉ thị nguồn | - | *SI* | *ESI* |
| Chỉ thị đích | - | *DI* | *EDI* |
| Đoạn mã | - | *CS* | - |
| Đoạn số liệu | - | *DS* | - |
| Đoạn ngăn xếp | - | *SS* | - |
| Đoạn số liệu E | - | *ES* | - |
| Đoạn số liệu F | - | *FS* | - |
| Đoạn số liệu G | - | *GS* | - |

Vì có các thanh ghi chỉ các offset và có hay không có dịch chuyển d nên có nhiều phương pháp địa chỉ toán hạng ở ô nhớ.

***b) Các phương pháp địa chỉ hoá toán hạng số liệu thông dụng***

Hình 3.8 minh họa cách tính toán hạng cho các chế độ gọi chương trình khác nhau liên quan tới các số liệu. Các chế độ đó là:

*- Gọi địa chỉ tức thì (immediate)*

Số liệu là 8 bit hoặc 16 bit và là một phần của lệnh.

*- Gọi địa chỉ trực tiếp*

16 bit của địa chỉ hiệu dụng (EA) hay offset của một số liệu là một phần của lệnh.

*Ví dụ*: EA = 1B57, (DS) = 2100.

Địa chỉ vật lý (PA - Physic Addresss) = EA + (DS)O = 1B57 + 2100O = 22B57.

*- Gọi địa chỉ thanh ghi*

Số liệu trong thanh ghi mà thanh ghi này được lệnh xác định:

+ Cho toán hạng 16 bit, các thanh ghi có thể là: AX, BX, CX, DX, SI, DI, SP hoặc BP.

+ Cho toán hạng 8 bit, các thanh ghi có thể là: AL, AH, BL, BH, CL, CH, DL hoặc DH.

Ở đây, số hiệu chỉnh là nội dung của thanh ghi mà không có EA để tìm số liệu trong bộ nhớ.

*- Gọi địa chỉ gián tiếp của thanh ghi*

Địa chỉ hiệu dụng của số liệu EA ở trong thanh ghi cở sở (Base Register) hoặc một thanh ghi chỉ thị (index register) được lệnh chỉ thị:

$$\left[ EA = \begin{Bmatrix} (BX) \\ (SI) \\ (DI) \end{Bmatrix} \right]$$

*Ví dụ*: Địa chỉ hoá thanh ghi BX = 0158h cho EA = 0158h + 21000h = 21158h với DS = 21000h.

*- Gọi thanh ghi tương đối*

Địa chỉ hiệu dụng EA là tổng của một dịch chuyển (d) 8 bit hoặc 16 bit và nội dung của một thanh ghi cơ sở hoặc chỉ thị.

*Ví dụ*: Thanh ghi tương đối dùng BX:

$$EA = \left\{ \frac{(BX)}{(DX)} + \frac{(SI)}{(DI)} \right\} + d$$

EA = 0158h + 1B57h = 1CAFh

PA = 1CAFh + 21000h = 22CAFh.

*- Gọi địa chỉ cơ sở, chỉ thị*

Địa chỉ hiệu dụng EA là tổng của một thanh ghi cơ sở (BX, DX) và chỉ thị (SI, DI), cả hai thanh ghi được chỉ định bởi lệnh:

EA = BX (DX) + SI (DI)

*Ví dụ:* Dùng các thanh ghi BX và DI.

EA = 0158h + 10A5h = 11FDh

PA = 11FDh + 21000h = 221FDh

*- Gọi địa chỉ cơ sở, chỉ thị, tương đối*

Địa chỉ hiệu dụng EA là tổng của một dịch chuyển 8 bit hoặc 16 bit và một địa chỉ cơ sở, chỉ thị nghĩa là:

$$EA = \left\{ \frac{(BX)}{(DX)} + \frac{(SI)}{(DI)} \right\} + d$$

*Ví dụ:* Dùng BX và DI

EA = 0158h + 10A5h + 1B57h

PA = 2D54h + 21000h = 23D54h

Công thức tính các địa chỉ của ô nhớ là :

EA = [địa chỉ trực tiếp]

LA = Thanh ghi cơ sở : địa chỉ trực tiếp

$$\begin{matrix} \\ DS \\ LA = \\ ES \\ FS \end{matrix} \begin{pmatrix} CS \\ \\ SS \\ \\ GS \end{pmatrix} : \text{[địa chỉ trực tiếp]}$$

*Ví dụ:* MOV BX, [1234h] ; chuyển nội dung ô nhớ có địa chỉ tương đối hay địa chỉ lệch 1234h trong mảng có địa chỉ đoạn DS vào thanh ghi BX.

*Ví dụ:* MOV BH, [1234h] ; chuyển nội dung ô nhớ có địa chỉ tương đối hay địa chỉ lệch 1234h và địa chỉ đoạn DS vào thanh ghi BH.

*Ví dụ:* Cho EA = 1B57h, (DS) = 2100h

thì PA = 1B57h + 21000h = 22B57h.

Theo cách tính trên, toán hạng của lệnh là nội dung của ô nhớ có địa chỉ là 22B57h.

Với cách gọi địa chỉ trên, ta có:

- VXL có số liệu 8 bit, có thể gọi tối đa là 65536 ô nhớ (64kB).
- VXL có số liệu 16 bit, có thể gọi tối đa 4 GB.

## 2. Chế độ gọi địa chỉ cho rẽ nhánh chương trình

Trong các lệnh chuyển điều khiển thực hiện chương trình, tức là thực hiện các lệnh nhảy (jump), rẽ nhánh (branch), VXL phải tính được địa chỉ cần thiết để chuyển tới địa chỉ mới của khối nhớ M chứa lệnh phải thực hiện. Dĩ nhiên là địa chỉ mới này có thể vẫn nằm trong mảng do thanh ghi CS quản lý (intrasegment) hoặc ngoài mảng do CS quản lý (intersegment).

***a) Trực tiếp trong mảng (intrasegment direct)***

Địa chỉ nhánh hiệu dụng EA này là tổng nội dung của thanh ghi lệnh IP hiện hành với một dịch chuyển (8 hoặc 16 bit) :

EA = (IP) + d (8 hoặc 16 bit).

Khi dịch chuyển d là 8 bit, lệnh nhảy gọi là ngắn. Nhiều tài liệu còn gọi là địa chỉ tương đối bởi vì dịch chuyển được tính tương đối so với IP. Chế độ địa chỉ này có thể được dùng cho sự rẽ nhánh có hoặc không có điều kiện, nhưng một lệnh rẽ nhánh có điều kiện chỉ có một dịch chuyển 8 bit.

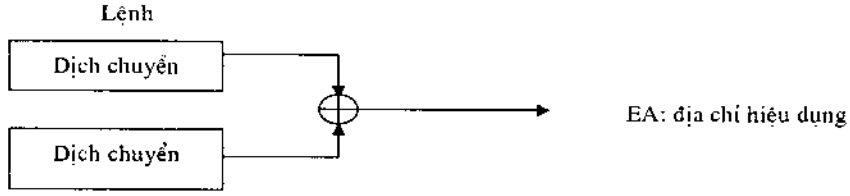
***b) Gián tiếp trong mảng***

Địa chỉ nhánh hiệu dụng EA là nội dung của một thanh ghi lệnh hoặc vùng nhớ đã được truy cập đã dùng ở trên cho chế độ địa chỉ của số liệu, trừ chế độ tức thì (Immididiate Mode). Nội dung của IP được thay đổi bởi địa chỉ nhánh hiệu dụng. Chế độ địa chỉ này có thể được dùng trong các lệnh nhảy không điều kiện.

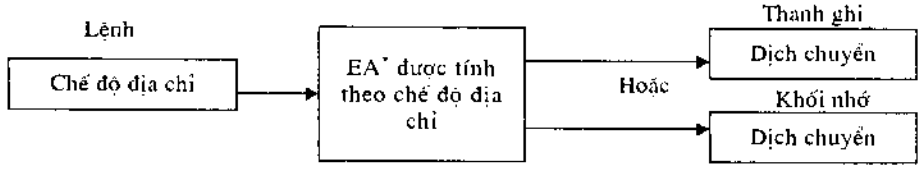
***c) Trực tiếp ngoài mảng***

Thay thế nội dung của IP với phần lệnh và nội dung của CS với phần khác của lệnh.

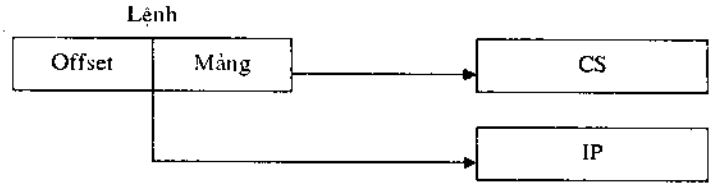
- Trực tiếp trong mảng (intrasegment direct).



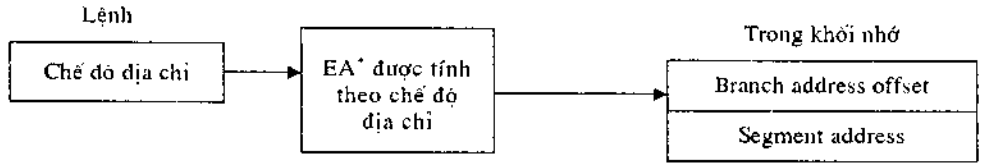

- Gián tiếp trong mảng (intrasegment indirect).




- Trực tiếp ngoài mảng (intersegment direct).




- Gián tiếp ngoài mảng (intersegment indirect).




**Hình 3.9.** Giới thiệu bốn loại tính địa chỉ cho lệnh rẽ nhánh.

Mục đích của chế độ địa chỉ này là cung cấp phương tiện cho sự rẽ nhánh từ một mảng mã lệnh (CS) này tới một mảng mã lệnh khác (hình 3.9).

*d) Gián tiếp ngoài mảng*

Thay nội dung của IP và CS với nội dung của hai từ liên tiếp trong khối nhớ đã được tham chiếu khi dùng bất kỳ chế độ địa chỉ nào liên quan tới số liệu ở trên, trừ chế độ tức thì và các thanh ghi. Chú ý là địa chỉ nhánh hiệu dụng là nội dung của IP cộng với nội dung của CS nhân với $16_{10}$.

Một lệnh nhánh ngoài mảng phải là lệnh không điều kiện. Để minh hoạ công việc rẽ nhánh gián tiếp như thế nào với một số chế độ địa chỉ liên quan tới số liệu, giả sử rằng:

(BX) = 1256; SI = 52BF, dịch chuyển d = 20A1 thì :

Với địa chỉ trực tiếp, địa chỉ nhánh hiệu dụng là nội dung của: 20A1 + (DS) x $16_{10}$.

Với địa chỉ thanh ghi tương đối dùng BX, địa chỉ nhánh hiệu dụng là nội dung của:

1256 + 20A1 + (DS) x $16_{10}$.

Với địa chỉ cơ sở dùng các thanh ghi BX và (SI), địa chỉ nhánh hiệu dụng là nội dung của: $1256 + 258F + (DS) \times 16_{10}$.

Bảng 3.8 tóm tắt các chế độ địa chỉ chung liên quan tới số liệu và địa chỉ.

***Bảng 3.8.* Tóm tắt chế độ địa chỉ liên quan tới số liệu và địa chỉ**

| Chế độ địa chỉ | Dạng | |
|---|---|---|
| *Liên quan đến số liệu* | | |
| Tức thì | Biểu thức hằng số chỉ toán hạng | 10110B<br>XY = XX + 5 |
| Trực tiếp | Biểu thức hằng số trong [ ], chỉ địa chỉ ô nhớ chứa toán hạng | CNT - 5<br>ARRY + 5 |
| Thanh ghi | Nội dung thanh ghi chỉ toán hạng | AX,AH |
| Thanh ghi gián tiếp | Nội dung thanh ghi trong [ ], chỉ ô nhớ | [BX] |
| Thanh ghi tương đối | Biến số (Thanh ghi ± biểu thức hằng số) hoặc (Thanh ghi ± biểu thức hằng số). | VAR X[BX]<br>[BX-1] |
| Cơ sở chỉ thị | Tổng số nội dung của thanh ghi cơ sở và nội dung của thanh ghi chỉ thị là địa chỉ khối nhớ. | [BP][DI] |
| *Liên quan tới địa chỉ rẽ nhánh* | | |
| Intrasegment direct | Label ± biểu thức hằng | |
| Intrasegment indirect | Như dạng với số liệu trừ địa chỉ tức thì | OVER ERROR+7 |
| Intrasegment direct | Label ± biểu thức hằng | |
| Intrasegment indirect | Như dạng với số liệu trừ địa chỉ tức thì và thanh ghi | Branch EXIT |

## 3.5. CÁC NHÓM LỆNH

Bộ lệnh của các vi xử lý thuộc họ 80x86 có các nhóm như sau:

- Chuyển số liệu.
- Xử lý số học (cộng, trừ, so sánh, nhân, chia).
- Xử lý lôgic (và, hoặc, loại trừ, đảo, kiểm tra).
- Xử lý bit (dịch, quay...).
- Xử lý chuỗi (chuyển, so sánh, lưu nạp).
- Rẽ nhánh chương trình (gọi, nhảy không và có điều kiện).
- Điều khiển vi xử lý.
- Hỗ trợ ngôn ngữ bậc cao.

### 3.5.1. NHÓM LỆNH CHUYỂN

Trong quá trình hoạt động, máy vi tính nói chung, vi xử lý nói riêng cần đưa số liệu vào, đưa số liệu ra. Hơn nữa trong khi hoạt động, số liệu hoặc trạng thái các thanh ghi, các bảng mô tả cần di chuyển (nạp vào, lưu giữ, lấy ra) với các thanh ghi, với vùng nhớ ngăn xếp (stack) và với ngăn nhớ. Do đó khi chế tạo, phải có các lệnh chuyển số liệu.

Có thể phân các lệnh chuyển số liệu trên thành các nhóm nhỏ sau:

- Chuyển số liệu
- Nạp địa chỉ
- Trao đổi số liệu với vùng nhớ ngăn xếp
- Trao đổi số liệu cho các bảng mô tả và thanh ghi nhiệm vụ.

## 1. Nhóm chuyển số liệu

Nhóm lệnh này thực hiện các hành động vào-ra số liệu (in-out), hoán chuyển giá trị giữa các thanh ghi (XCHG) và chuyển số liệu (MOV).

***Bảng 3.9.*** **Các lệnh chuyển số liệu**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| MOV | Chuyển | MOV D,S | (S)→(D) | Không |
| MOVS | Chuyển chuỗi | MOVBS, MOVSN, MOVSD | (ES)0+(SI)→(ES)0+(DI) | Không |
| MOVSX* | Chuyển với mở rộng dấu | MOVSX D,S | (S) →(D)<br>MSB của D được kéo dài với bit dấu của S | Không |
| MOVZX* | Chuyển với mở rộng 0 | MOVEX D,S | (S) →(D)<br>MSB của D được kéo dài với 0 | Không |
| XCHG | Hoán chuyển | XCHG D,S | (D)↔(S) | Không |
| IN | Vào | IN D, S | (AL, AX, EAX)←(DX) | |
| OUT | Ra | OUT D, S | (AL, AX, EAX) →(DX), EDX | |
| BSWAP* | Sắp xếp lại byte | BSWAP | Sắp xếp lại 4 byte trong lời 32 bit | |

Bảng 3.9 tóm tắt các danh sách các lệnh của nhóm chuyển số liệu. Lệnh với dấu * cho vi xử lý từ 80386 trở lên.

Các lệnh vào, ra (IN, OUT) thực chất cũng là lệnh chuyển (MOV) số liệu, nhưng trao đổi giữa thanh ghi chứa (AL, AX, EAX) với cảng vào ra của thiết bị ngoài có địa chỉ ở trong thanh ghi DX, EDX (xem chương 5).

***a) Các lệnh chuyển MOV***

Thực chất các loại lệnh MOV đều ra di chuyển số liệu giữa nguồn (S) tới đích (D), số liệu ở nguồn (S) không thay đổi, nhưng có thể có các dạng sau :

- Chuyển một số liệu (MOV).
- Chuyển chuỗi số liệu trong ô nhớ với địa chỉ tăng hay giảm dần (xem nhóm xử lý chuỗi ở dưới).
- Chuyển số liệu với mở rộng bit 0 (MOVZX) và bit dấu (MOVSX).

Với lệnh MOV ta có bảng các nguồn (S) và đích (D) số liệu như bảng 3.10

*Ví dụ:*
```
MOV AX, 1234H     ; chuyển số 1234H vào AX
MOV AX, [BX]      ; chuyển nội dung ô nhớ có địa chỉ ở BX vào AX
MOV AH, "A"       ; chuyển mã ASCII của chữ A vào AH.
```

Qua bảng 3.10b ta thấy:

- Cấm chuyển số liệu cho số liệu tức thì Im
- Cấm chuyển số liệu cho các thanh ghi mảng Sreg.
- Cấm di chuyển số liệu giữa hai ô nhớ.

*Bảng 3.10* **a) Các toán hạng của lệnh MOV**
**b) Các khả năng của các toán hạng của lệnh MOV**
**c) Các khả năng của lệnh MOVSX, MOVZX**

| Đích D | Nguồn S |
|---|---|
| Ô nhớ Mem | Thanh chứa A |
| Thanh chứa A | Ô nhớ Mem |
| Thanh ghi Reg | Thanh ghi Reg |
| Thanh ghi Reg | Ô nhớ Mem |
| Ô nhớ Mem | Thanh ghi Reg |
| Thanh ghi Reg | Tức thì Imm |
| Ô nhớ Mem | Tức thì Imm |
| Thanh ghi mảng Sreg | Reg 16 |
| Thanh ghi mảng Sreg16 | Mem 16 |
| Mem 16 | Sreg |
| Reg | Sreg |
| Thanh ghi đặc biệt Spec Reg | Spec Reg |
| | Reg |

a)

| DS | Reg | Sreg | Mem | Imm |
|---|---|---|---|---|
| Reg | Có | Có | Có | Không |
| Sreg | Có | Không | Có | Không |
| Mem | Có | Có | Không | Không |
| Imm | Có | Không | Có | Không |

b)

| Đích D | Nguồn S |
|---|---|
| Reg 16 | Reg 8 |
| Reg 32 | Reg 8 |
| Reg 32 | Reg 16 |
| Reg 16 | Mem 8 |
| Reg 32 | Mem 8 |
| Reg 32 | Mem 16 |

c)

*b) Lệnh hoán chuyển XCHG*

Lệnh hoán chuyển này chỉ xảy ra giữa:

- Các thanh ghi thông dụng Reg, ví dụ XCHG BX, DX.
- Giữa thanh ghi và thanh ghi chứa, ví dụ XCHG BX, AX.
- Giữa ô nhớ và thanh ghi, ví dụ XCHG [BX], AX.

Không có hoán chuyển với thanh ghi mảng Sreg và giữa các ô nhớ.

## 2. Nhóm nạp, chuyển địa chỉ

Nhóm này nạp (L) địa chỉ (EA) cho thanh ghi Reg và các thanh ghi mảng (DS, SS, ES, FS, GS - trừ CS), cũng như chuyển đổi địa chỉ và đọc bảng số liệu (XLAT).

Như vậy, mỗi lệnh nạp (L) thanh ghi và thanh ghi mảng trên tương đương hai lệnh nạp.

*Ví dụ:* - LDS BX, MEM [SI] tương đương với: MOV BX, MEM [SI]
MOV AX, MEM [SI+1]
MOV DS, AX.

- LDS SI [200h] kết quả của lệnh là:

+ Nạp vào thanh ghi SI nội dung của ô nhớ mà địa chỉ offset (địa chỉ lệch) của nó là: PA = 12000h + 0200h = 12200h (vì DS = 1200, dịch chuyển đi 4 bit).

+ Hai byte sau có địa chỉ là 12202h và 12303h nạp vào thanh ghi DS (ví dụ 00 và 13). Như vậy, địa chỉ của DS = 1300h và địa chỉ tuyệt đối của đoạn số liệu là 13000h.

- LEA BX, DS: [500h]; chuyển 500h vào BX.

- LEA SI, table; chuyển offset của bảng table cộng với nội dung của thanh ghi SI vào thanh ghi BX.

*Ví dụ:* DS = 300h; BX = 100h; AL = 0Dh thì PA = 03000h + 0100h + 0Dh = 0310Dh

(0310Dh) → (AL) tức nội dung ô nhớ có địa chỉ 0310Dh ghi vào AL.

Thường bảng chứa các mã (ví dụ EBCDIC) do đó ta có thể biến đổi mã ASCII thành EBCDIC nếu biết địa chỉ BX tương ứng với số mã ASCII (bảng 3.11).

***Bảng 3.11.*** **Nhóm lệnh nạp, chuyển địa chỉ**

| Lệnh | Ý nghĩa | Dạng | Hành động | Cờ |
|---|---|---|---|---|
| LEA | Nạp địa chỉ hiệu dụng EA | LEA Reg 16, EA | (EA) → (Reg16) | Không |
| | | LEA Reg 32, EA | (EA) → (Reg32) | |
| LDS | Nạp thanh ghi và DS | LDS Reg 16, EA | (EA) → (Reg16) | Không |
| | | | (EA+2) → (DS) | |
| | | LDS Reg 32, EA | (EA) → (Reg32) | |
| | | | (EA+4) → (DS) | |
| LSS | Nạp thanh ghi và SS | LSS Reg 16, EA | (EA) → (Reg16) | Không |
| | | | (EA+2) → (SS) | |
| | | LSS Reg 32, EA | (EA) → (Reg32) | |
| | | | (EA+4) → (SS) | |
| LES | Nạp thanh ghi và ES | LES Reg 16, EA | (EA) → (Reg16) | Không |
| | | | (EA+2) → (ES) | |
| | | LES Reg 32, EA | (EA) → (Reg32) | |
| | | | (EA+4) → (ES) | |
| LFS | Nạp thanh ghi và FS | LFS Reg 16, EA | (EA) → (Reg16) | Không |
| | | | (EA+2) → (FS) | |
| | | LFS Reg 32, EA | (EA) → (Reg32) | |
| | | | (EA+4) → (FS) | |
| LGS | Nạp thanh ghi và GS | LGS Reg 16, EA | (EA) → (Reg16) | Không |
| | | | (EA+2) → (GS) | |
| | | LGS Reg 32, EA | (EA) → (Reg32) | |
| | | | (EA+4) → (GS) | |
| XLAT | Chuyển đổi (Translate) | XLAT bảng nguồn | ((AL)+(BX)+(DS)0) → AL | Không |
| XLATB | Chuyển đổi | XLATB | ((AL)+(BX)+(DS)0) → AL | |

## 3. Nhóm lệnh chuyển, lưu giữ

Nhóm lệnh này (xem bảng 3.12) có các nhóm nhỏ sau:

- Chuyển giữa AH và flag: để kiểm tra flag (LAHF-SAHF).

- Chuyển vào-lấy ra (PUSH-POP) giữa vùng nhớ ngăn xếp với thanh ghi, bộ nhớ để lưu giữ tạm nội dung của chúng.

- Chuyển vào lấy ra thanh ghi trạng thái (MSW) với khối nhớ để lưu giữ.

*a) Cặp lệnh LAHF - SAHF*

Hai lệnh này có tác dụng ngược nhau, không có toán hạng tường minh, mà toán hạng ẩn là F được ghi ngay trong lệnh, thực hiện việc chuyển các bit của thanh ghi cờ F vào AH để kiểm tra rồi lại trả giá trị cũ về F.

*b) Cặp lệnh PUSHS(D) - POPS(D)*

Hai lệnh này cũng có tác dụng ngược nhau, có toán hạng S hoặc D và có thể là:

- Số liệu tức thì (chỉ lệnh PUSH).
- Thanh ghi thông dụng(Reg) hoặc thanh ghi đoạn (Sreg).
- Ô nhớ.

Còn toán hạng còn lại là ô nhớ ngăn xếp với địa chỉ ở (SP). Sau mỗi lần thực hiện lệnh địa chỉ ô nhớ ngăn xếp (con trỏ ngăn xếp SP) lại giảm đi 2 (lệnh PUSH) và tăng lên 2 (lệnh POP).

Lệnh trên có các trường hợp riêng là:

+ *Reg là F*: thanh ghi Flag lưu giữ (PUSHF) và phục hồi (POPF) do trao đổi với ngăn xếp.

+ *Reg là tất cả các thanh ghi* (thông dụng, trạng thái, mảng): lưu giữ bởi PUSHA và hồi phục POPA (bảng 3.12).

***Bảng 3.12.* Các lệnh thuộc nhóm chuyển, lưu giữ**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| LAHF | Lưu Flag vào AH | LAHF | (F) → (AH) | Không |
| SAHF | Trả AH về F | SAHF | (AH) → (F) | Không |
| PUSH S | Gửi S vào ngăn xếp | PUSH S | (S) → [SP] | Không |
| POP D | Lấy D từ ngăn xếp | POP D | [SP] → (D) | Không |
| PUSH F | Gửi F vào ngăn xếp | PUSH F | (F) → [SP] | Không |
| POP F | Lấy F từ ngăn xếp | POP F | [SP] → (F) | Không |
| PUSH A* | Gửi nội dung các thanh ghi vào ngăn xếp | PUSH A | (Reg) → [SP] | Không |
| POP A* | Lấy nội dung các thanh ghi từ ngăn xếp | POP A | [SP] → (Reg) | Không |
| LMSW* | Nạp thanh ghi trạng thái vào khối nhớ | LMSW | (MSW) → (D) | Không |
| SMSW* | Lấy thanh ghi trạng thái từ khối nhớ | SMSW | (D) → (MSW) | Không |

*c) Cặp lệnh LMSW-SMSW*

Trao đổi thanh ghi trạng thái (AX, IP, F) với ô nhớ có địa chỉ đích D.

## 4. Nhóm trao đổi tin của các bảng mô tả (GDT, IDT, LDT) và thanh ghi nhiệm vụ (TR), giới hạn (SL)

Nhóm này có từ 80286 dùng ở chế độ bảo vệ.

Danh sách các lệnh như bảng 3.13.

Các lệnh này cung cấp bảng số liệu cho các thanh ghi, bảng ở chế độ bảo vệ.

*Bảng 3.13.* **Các lệnh đảm bảo ở chế độ bảo vệ**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ tác động |
|---|---|---|---|---|
| LSL | Nạp thanh ghi giới hạn mảng | LSL S (S = thanh ghi/ bộ nhớ) | (S) → (SL) | Không |
| LTR | Nạp thanh ghi nhiệm vụ TR | LTR S (S = Reg/Mem) | (S) → (TR) | Không |
| STR | Phục hồi TR | STR D | (D) → (TR) | Không |
| LGDT | Nạp bảng GDT | LGDTS (S - bảng thanh ghi) | (S) → GDT | Không |
| SGDT | Phục hồi GDT | SGDTD (D - bảng thanh ghi) | (D) → GDT | Không |
| LIDT | Nạp bảng IDT (mô tả ngắt) | LIDTS | (S) → (IDT) | Không |
| SIDT | Phục hồi IDT | SIDTD (S, D - bảng thanh ghi) | (D) → (DT) | Không |
| LLDT | Nạp bảng LDT (mô tả cục bộ) | LLDTS | (S) → LDT | Không |
| SLDT | Phục hồi bảng mô tả cục bộ | SLDTD (S, D - bảng thanh ghi) | (D) → LDT | Không |
| ARPL | Hiệu chỉnh mức ưu tiên | ARPL | (S) → (AR) | Không |
| LAR | Nạp quyền truy nhập | LAR S | (S) → LAR | Không |

## 3.5.2. NHÓM LỆNH SỐ HỌC

### 1. Đại cương

Vi xử lý Intel 8086 nói riêng, họ 80x86 nói chung đều có 4 loại lệnh số học là cộng, trừ, nhân, chia với các dạng số liệu khác nhau: nhị phân (không dấu và có dấu), BCD (không nén và nén hay đóng gói) với các độ dài khác nhau (byte, word, độ chính xác cao, hình 3.10).

***Các loại lệnh***

Với các lệnh cộng trừ ta không có các lệnh cho các số nguyên (integer - số có dấu) mà phải có việc xử lý riêng bằng chương trình (xem chương 6). Còn các lệnh nhân và chia có thêm các lệnh cho số liệu có dấu với tiếp đầu ngữ I (integer).

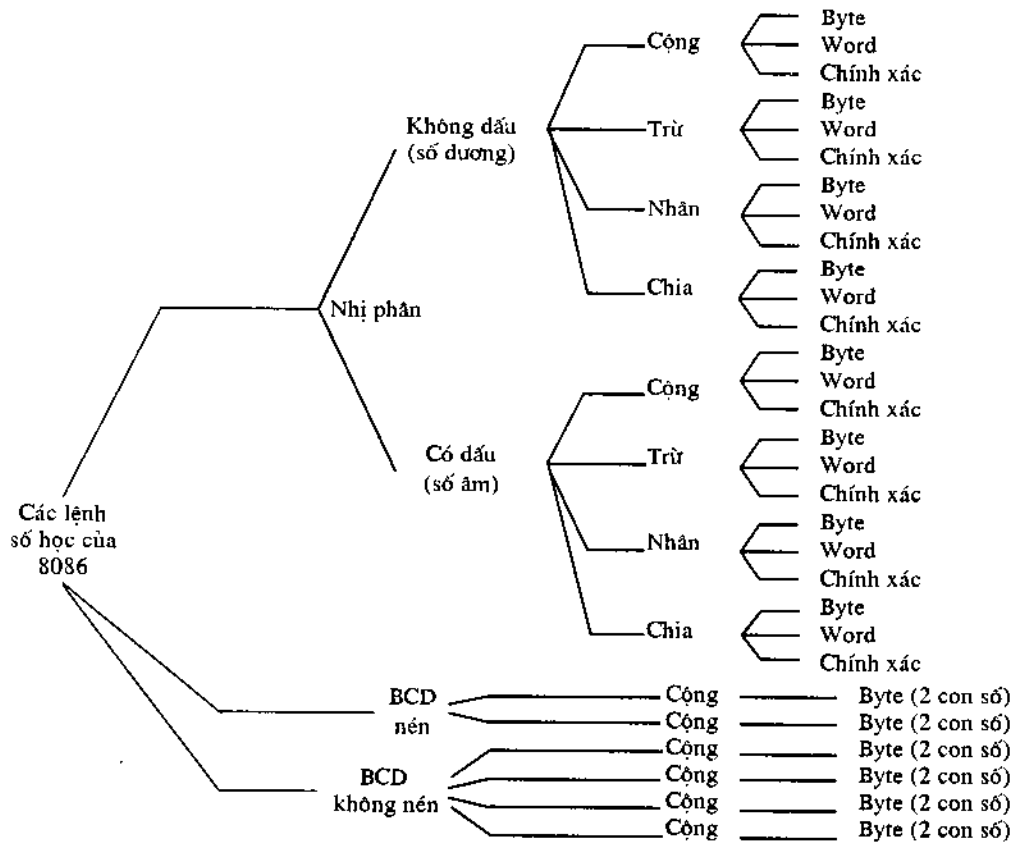
Với lệnh cộng, trừ có thêm lệnh cộng và trừ có nhớ (ADC, SBC), cộng và trừ với đơn vị (1) tức tăng (INC) hay giảm 1 (DEC). Riêng phép trừ còn có lệnh đảo dấu NEG tức trừ 0 với số liệu là bit 0 hoặc 1 (NEGA = 0 - A).

Các lệnh hiệu chỉnh ASCII cho các phép tính cộng (lệnh AAA), trừ (lệnh AAS), và nhân (lệnh âm) phải đứng sau các lệnh cộng, trừ và nhân cho các số liệu dạng BCD không nén (mã nhị phân của một con số thập phân từ 0 ÷ 9 đặt trong một byte 8 bit). Muốn đổi từ mã BCD không nén sang mã ASCII ta phải thực hiện toán tử OR của mã BCD trên với số 3030h. Các lệnh hiệu chỉnh trên chỉ thực hiện sau khi đã dùng lệnh xử lý số liệu cộng trừ, nhân, tức xử lý trước, hiệu chỉnh sau.

Các lệnh thực hiện hiệu chỉnh ASCII cho các phép tính chia (lệnh AAD) phải được dùng trước khi tiến hành phép chia (hiệu chỉnh trước, thực hiện xử lý sau) tức biến đổi các số dạng không nhị phân thành dạng nhị phân để làm phép chia.

Với phép cộng, trừ, còn có sự hiệu chỉnh thập phân (DAA, DAS) cho các phép tính với số liệu dạng BCD nén (mã nhị phân của hai con số thập phân trong một byte 8 bit).

Cách viết các lệnh được giới thiệu trên bảng 3.14.




**Hình 3.10**. Các lệnh số học với các dạng số liệu của VXL Intel 80x86.

## 2. Các lệnh cộng

Các lệnh cộng với dạng, hành động và cờ bị tác động như bảng 3.15.

- Không thể cộng trực tiếp thanh ghi đoạn.
- Không thể cộng trực tiếp hai ô nhớ.

Muốn thực hiện các phép cộng trên phải dùng một trung gian là thanh ghi chứa (AX, AH, AL).

***a) Lệnh ADD (Addition) D, S***

Lệnh cộng hai số liệu (trực tiếp), thanh ghi hay ở trong bộ nhớ với các phương pháp địa chỉ khác nhau.

*- Cộng hai thanh ghi*

*Ví dụ:*
```
ADD AL, AH      ; cộng nội dung hai thanh ghi 8 bit, AH+AL→AL
ADD AX, CX      ; cộng nội dung hai thanh ghi 16 bit, AX+CX→AX
ADD [BX], AL    ; cộng nội dung ô nhớ có địa chỉ ở
                ; thanh ghi BX với AL, [BX]+AL → AL.
```

*- Cộng thanh ghi với bộ nhớ*

*Ví dụ:*
```
ADD ES: [SI], DL    ; cộng nội dung ô nhớ có địa chỉ ES:SI với DL,
                    ; [ES : SI] + DL→[ES:SI]
```

ADD DS: [150h], AL ; [DS : 0150h] + AL→[DS : 0150h]
ADD AL, mem [BX] ; AL + [BX]→AL

ADD biếnW, AX: cộng biến nhớ với AX, biếnW + AX→biếnW.

- *Cộng số trực tiếp*

*Ví dụ:* ADD AL, 0B5h ; AL + 0B5h→AL
ADD mem [BX], 15h ; [BX] +15h→[BX]
ADD byte ptr DS: [150h], 10h ; cộng byte nhớ có địa chỉ DS:0150h với số 150h
ADD word ptr [BX], 5AB7h ; cộng từ nhớ có địa chỉ BX với số 5AB7h.

***Bảng 3.14.* Các lệnh số học của 80x86**

| Cộng | |
|---|---|
| ADD | Cộng byte hay word |
| ADC | Cộng byte hay word có nhớ |
| INC | Tăng byte hay word lên 1 |
| AAA | Hiệu chỉnh ASCII cho phép cộng |
| DAA | Hiệu chỉnh thập phân cho phép cộng |
| **Trừ** | |
| SUB | Trừ byte hay word |
| SBB | Trừ byte hay word có nhớ |
| DEC | Giảm byte hay word đi 1 |
| NEG | Lấy đảo byte hay word |
| AAS | Hiệu chỉnh ASCII cho phép trừ |
| DAS | Hiệu chỉnh thập phân cho phép trừ |
| **Nhân/Chia** | |
| MUL | Nhân byte hay word số không dấu |
| IMUL | Nhân byte hay word số có dấu (số nguyên) |
| AAM | Hiệu chỉnh ASCII cho phép nhân |
| DIV | Chia byte hay word số không dấu |
| IDIV | Chia byte hay word số có dấu (số nguyên) |
| AAD | Hiệu chỉnh ASCII cho phép chia |
| CBW | Biến đổi byte thành word |
| CWD | Biến đổi word thành 2 word trong DX và AX |
| CWDE | Biến đổi word thành 2 word trong EAX |
| CWDQ | Biến đổi 2 word thành 4 word |

***b) Lệnh ADC D, S***

Lệnh này tương tự ADD, chỉ khác là kết quả còn cộng thêm với CF. Nếu CF = 0 lệnh ADC như lệnh ADD, còn CF = 1, kết quả phải cộng thêm 1. Thường lệnh này được sử dụng trong các phép cộng hai byte cao của hai số nhiều byte trong một hay nhiều thanh ghi và ta không biết được kết quả của các phép cộng các byte thấp có nhớ hay không. Ví dụ: Cộng hai số nằm trong cặp hai thanh ghi DX:AX và CX:BX, ta có chuỗi lệnh sau:

ADD AX, BX ; cộng hai thanh ghi số liệu thấp.
ADC DX, CX ; cộng hai thanh ghi số liệu cao và CF. Nếu kết quả phép cộng trên có nhớ (CF =1) tổng số được cộng thêm 1, nếu không có nhớ (CF = 0), tổng số được cộng thêm 0.

***c) Lệnh INC S***

Làm tăng nội dung toán hạng S lên 1 tức cộng toán hạng S với 1.

*Ví dụ:* INC AL ; nếu AL = 10h, thì INC AL cho AL = 10h + 1h = 11h
INC byte ptr [BX+SI] ; tăng nội dung của byte nhớ có địa chỉ ở tổng BX+SI lên 1.
INC mem w[BX] ; tăng nội dung biến nhớ có địa chỉ BX lên 1.

*Bảng 3.15* **a) Các lệnh cộng số học**
**b) Các toán hạng cho các phép cộng (ADD, ADC)**
**c) Toán hạng cho lệnh INC.**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| ADD | Cộng | ADD D, S | (S)+(D) → (D) | OF, SF, ZF, AF, PF, CF |
| ADC | Cộng có nhớ | ADC D, S | Carry → (CF) | OF, SF, ZF, AF, PF, CF |
| INC | Tăng lên 1 | INC D | (S)+(D)+(CF) → (D) | OF, SF, ZF, AF, PF |
| DAA | Hiệu chỉnh thập phân cho phép cộng | DAA | Carry → (CF) | SF, ZF, AF, PF, CF, OF không xác định |
| AAA | Hiệu chỉnh ASCII cho phép cộng | AAA | (D)+1→ (D) | AF, CF |

a)

| Đích số liệu (D) | Nguồn số liệu (S) |
|---|---|
| Thanh ghi (Reg) | Thanh ghi (Reg) |
| Thanh ghi (Reg) | Bộ nhớ (Mem) |
| Bộ nhớ (Mem) | Thanh ghi (Reg) |
| Thanh ghi (Reg) | Số liệu tức thì Im |
| Bộ nhớ (Mem) | Số liệu tức thì Im |
| Thanh ghi chứa (AX) | Số liệu tức thì Im |

b)

| Đích (D) |
|---|
| Thanh ghi ($Reg_8$) |
| Thanh ghi ($Reg_{16}$) |
| Ô nhớ (Mem) |

c)

***d) Lệnh AAA (Adjust After Addition) - hiệu chỉnh sau phép cộng (bảng 3.15)***

Có tài liệu quan niệm lệnh AAA này là Adjust ASCII for Addition - hiệu chỉnh mã ASCII cho phép cộng, nhưng không thực sự như vậy và quan niệm hiệu chỉnh sau phép cộng là chính xác hơn như trình bầy ở dưới đây.

Lệnh này không có toán hạng tường minh mà toán hạng ẩn trong thanh ghi AX. Lệnh này hiệu chỉnh kết quả của phép cộng hai số dạng mã BCD không nén để cho kết quả là số mã BCD không nén của một chữ số (mã ASCII khác mã BCD của một chữ số thập phân từ 0 ÷ 9 một lượng là 30h).

Lệnh này dùng sau lệnh cộng hai chữ số dạng mã BCD không nén hay hai số mã ASCII.

*Ví dụ:* AL chứa số 5 thập phân, mã ASCII là 35h.
BL chứa số 3 thập phân, mã ASCII là 33h.

Cộng hai thanh ghi trên ta có:

ADD AL, BL: kết quả trong AL = 35h + 33h = 68h.

AAA: kết quả trong AL = 38h, tức đã trừ đi 30h để cho kết quả mã BCD không nén là 08h.

Nếu kết quả tổng hai số thập phân lớn hơn 10, mã ASCII của tổng lớn hơn hay bằng 70h thì kết quả mã ASCII phải là hai byte.

***e) Lệnh DAA (Decimal Adjust Addition) - hiệu chỉnh thập phân cho phép cộng***

Lệnh dùng để hiệu chỉnh kết quả của phép cộng của hai số dạng mã BCD nén (Binary Coded Decimal) với 2 chữ số thập phân trong 1 byte.

Phép hiệu chỉnh xảy ra khi phép cộng cho kết quả có cờ nhớ phụ của phép cộng hai bit thứ tư kể từ bên phải sang (AC = 1), tức 4 bit (nible) thấp của 1 byte lớn hơn 9h.

Sự hiệu chỉnh tiến hành như sau:

- Cộng 6h cho số thập phân BCD hàng đơn vị.
- Cộng 60h cho số thập phân BCD hàng chục.
- Cộng 600h cho số thập phân BCD hàng trăm.
- Cộng 6000h cho số thập phân BCD hàng nghìn.

*Ví dụ:* AL chứa số 12D = 00010010 (BCD)
AH chứa số 49D = 01001001 (BCD)
ADD AH, AL ; = 01011011 B = 5Bh = 50D + 11D = 61D
DAA ; 6D = 00000110 B vì có AC = 1
= 01100001 B = 61 (BCD).

## 2. Các lệnh trừ

Các lệnh trừ với dạng hành động và cờ bị tác động như bảng 3.16.

Các toán hạng của phép trừ cũng tương tự phép cộng (bảng 3.15).

***Bảng 3.16.*** **Các lệnh của phép trừ**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị ảnh hưởng |
|---|---|---|---|---|
| SUB | Trừ | SUB D, S | (D) - (S) → (D) | OF, SF, ZF, AF, PF, CF |
| SBB | Trừ có nhớ | SBB D,S | (D) - (S) - (CF) → (D) | Như trên |
| DEC | Giảm đi 1 | DEC D | (D) - 1 → (D) | OF, SF, ZF, AF, PF, CF |
| NEG | Lấy đảo (âm) | NEG D | 0 - (D) → (D) | OF, SF, ZF, AF, PF, CF |
| DAS | Hiệu chỉnh thập phân của dấu trừ | DAS | | SF, ZF, AF, PF, CF |
| AAS | Hiệu chỉnh ASII của phép trừ | AAS | | AF, CF |
| CMP | So sánh | CMP D, S | (D) - (S) → (D) | Các cờ |

***a) Lệnh SUB (Subtraction) - lệnh trừ hai số***

Lệnh này tương tự lệnh cộng hai số hay hai toán hạng đích (D) và nguồn (S) rồi gửi vào đích (D).

*Ví dụ:* SUB D, S ; (D) - (S) → (D)

Các ví dụ của phép cộng có thể áp dụng cho phép trừ.

***b) Lệnh SBB (Subtract with borrow) - trừ có mang sang (Borow) hay có nhớ***

Tương tự lệnh cộng có nhớ ADC, lệnh trừ có mang sang này (đã vay 1 ở hàng cao hơn để trừ ở hàng thấp hơn, bây giờ phải trừ đi 1 ở hàng cao hơn) phải trừ kết quả với CF = 1 (đã vay).

Lệnh này cũng được dùng để trừ hai byte cao của hai số nhiều byte chứa trong một hay nhiều thanh ghi.

*c) Lệnh DEC (Decrement) - giảm*

Tương tự lệnh INC tăng thêm 1, lệnh này giảm toán hạng đi 1.

*Ví dụ:* - DEC CX
- DEC byte ptr [BX+SI]
- DEC word ptr DS: [500h].

*d) Lệnh NEG (Negate)*

Lấy số âm tức bù 2 hay lấy đảo các giá trị của các bit rồi lấy kết quả cộng thêm 1.

NEG D có tác dụng 0 - (D) → (-D) hay $\overline{D}$.

Vì D là dạng nhị phân, nên cách lấy đảo theo hai bước như sau:
- Lấy đảo các bit 1 → 0 hay 0 → 1
- Cộng kết quả lấy đảo với bit 1.

*Ví dụ:* NEG (10110101) là:

$\overline{10110101}$ = 01001010
+ 1 = + 1
- D = 01001011

Lệnh lấy số bù 2 này dùng để làm phép cộng với số âm, khi đó số âm được biến thành số bù 2 rồi cộng kết quả với số hạng còn lại bằng lệnh cộng (ADD).

*e) Lệnh DAS (Decimal Adjust for Subtraction) - hiệu chỉnh thập phân cho phép trừ*

Lệnh này được sử dụng sau phép trừ hai số BCD nén, để hiệu chỉnh kết quả của phép trừ chứa trong thanh ghi AL, AH thành số biểu diễn dạng BCD nén (hai con số thập phân trong một byte).

Nếu: - 4 bit thấp của AL nhỏ hơn 9 (hoặc AF = 0) thì không cần hiệu chỉnh.
- 4 bit thấp của AL lớn hơn 9 (hoặc AF = 1) thì:
  . AL - 6 → AL (AL giảm đi 9).
  . 4 nible cao của AL cộng thêm 1.
- 4 bit cao của AL lớn hơn 9 (hoặc AF = 1) thì:
  . AL - 60h → AL (AL giảm đi 90 hay 4 bit cao giảm đi 6).
  . Cờ CF = 1 (có nhớ).

*Ví dụ:* - Không hiệu chỉnh:

```
                    ; AL = 00110101 BCD = 35h = 35D
                    ; BL = 00100100 BCD = 24h = 24D
SUB AL, BL          ; AL = 00010001 BCD = 11h = 11D
DAS                 ; AL = 00010001B = 11h = 11D
```

- Hiệu chỉnh ở nible thấp:

```
                    ; AL = 0101010 BCD = 5h = 56D
                    ; BL = 00111001 BCD = 39h = 39D
SUB AL, BL          ; AL = 00011101 BCD và AF = 1
DAS                 ; AL = 00010111 BCD.
```

- Hiệu chỉnh ở nible cao:

```
                    ; AL  = 0011 1000BCD = 38D
```

```
                    ; BL  = 01010100BCD = 54D
SUBAL, BL           ; AL  = 11100100BCD
DAS                 ; -60h = 01100000
                    ; AL  = 10000100BCD = 84D với nhớ CF = 1.
```

*f) Lệnh AAS (ASCII Adjust after Subtraction) - hiệu chỉnh ASCII cho phép trừ*

Lệnh được sử dụng sau phép trừ hai số dạng mã ASCII để cho kết quả dạng BCD không nén.

Cũng tương tự lệnh DAS nhưng ở đây chỉ có sự hiệu chỉnh khi phép trừ có kết quả âm, tức nible cao có giá trị CF và cờ AF=1 (có nhớ ở nible thấp-mã BCD của hai toán hạng của phép trừ).

Việc hiệu chỉnh xảy ra khi có AF=1 và CF.

- AL chứa mã BCD của kết quả trừ (số dương)
- Xóa nible cao của AL
- CF =1 (chuyển AF = 1 sang CF) để trừ tiếp trong byte cao, tức hàng đơn vị cao hơn.

Ví dụ: - Không hiệu chỉnh

```
                    ; AL  = 0011.0111 ASCII = 37h = 7D
                    ; BL  = 0011.0011 ASCII = 33h = 3D
        SUB AL, BL  ; AL  = 0000.0100h = 04h = 4D
        AAS         ; vì AF = CF = 0 nên AAS không thực hiện sự hiệu chỉnh.
```

- Không hiệu chỉnh, số âm

```
                    ; AL = 0011.0111ASCII = 37h = 7D
                    ; BL = 00111.001 ASCII = 39h = 9D
        SUB AL, BL  ; AL = 1111.011 BF6h với AF = CF = 1
        AAS         ; AL = 0000.1000BCD = 08D
                    ; AF = CF = 1 (có nhớ).
```

*g) Lệnh CMP (Compare) - so sánh*

Lệnh này giống lệnh trừ [(D) - (S)] nhưng hiệu số không gửi vào D mà kết quả so sánh chỉ tác động tới các cờ, ví dụ D lớn hơn S, tức hiệu là số dương (S = 0), D bé hơn S, hiệu là số âm (S = 1), D, S bằng nhau, hiệu số bằng 0 hay Zero (Z = 1).

## 3. Các lệnh nhân và chia

Các lệnh nhân và chia với toán hạng như ở bảng 3.17.

Khác với lệnh cộng và trừ, ở đây còn có thêm lệnh nhân và chia số nguyên (IMUL, IDIV) dùng để thực hiện phép tính nhân và chia các toán hạng có dấu (- và +).

Lệnh hiệu chỉnh phép tính nhân (AAM), đặt sau lệnh nhân để biến đổi tích số dạng nhị phân ra dạng BCD không nén.

Trái lại lệnh hiệu chỉnh phép chia lại đặt trước lệnh chia để biến đổi số bị chia dạng BCD không nén thành dạng nhị phân, rồi tiến hành phép chia bởi lệnh chia (DIV hay IDIV).

Các lệnh biến đổi byte thành word (CBW) word thành hai word (CWD) và CWDE trong EAX) hai word thành 4 word (CDQ) để mở rộng độ lớn và dấu của các toán hạng có số ít byte thành nhiều byte.

*a) Lệnh nhân hai số MUL*

Lệnh này chỉ có một toán hạng nguồn S (số liệu tức thì, thanh ghi hay ô nhớ) chỉ số nhân, còn toán hạng kia là số bị nhân chứa sẵn trong AL (8bit) hoặc AX (16 bit).

Nếu: - các toán hạng là số 8 bit, tích AL x S chứa trong AX;

- một trong các toán hạng là số 16 bit, kết quả chứa trong cặp thanh ghi DX:AX (DX chứa các bit cao).

*Ví dụ:* Nhân hai số 02 và 04

ta có các lệnh sau:

```
MOV AL, 02
MOV BL, 04
MUL BL      ; kết quả lưu trong AX là 08 có trong AL.
```

***b) IMUL S (Integer Multiplication) - nhân hai số có dấu (nguyên)***

Nhân AL, AX với S tùy S là 8 bit hay 16 bit. Các số có thể có dấu hay không dấu.

Có thể: - kết quả chứa cả trong AH khi CF = 1;

- kết quả chứa cả trong DX khi OF =1;
- kết quả là số âm khi SF = 1.

Nếu nhân hai số âm có thể dùng lệnh MUL, về dấu của kết quả là dương (+).

Lệnh IMUL có toán hạng có nhiều dạng hơn (bảng 3.17); có thể nhân với cả số tức thì (Imm).

*Ví dụ:* Viết các lệnh để dịch lệnh gán C = 4A - 9B của ngôn ngữ bậc cao ra các lệnh của assembly. Coi A và B là các biến kiểu word (16 bit) và giả sử không có lệnh tràn xảy ra và sử dụng lệnh MUL cho các phép nhân vì chưa biết là số có dấu hay không dấu.

```
MOV AX, 4   ; AX = 4
IMUL A      ; AX = 4*A
MOV A, AX   ; A  = 4*A
MOV AX, 9   ; AX = 9
IMUL B      ; AX = 9*A
SUB A, AX   ; AX = 4*A - 9*B
MOV C, AX   ; C  = 4*A - 9*B.
```

***c) Các lệnh chia hai số DIV, IDIV (bảng 3.17)***

Các lệnh chia hai số không dấu (DIV) và có dấu (IDIV) đều có:

- Toán hạng đích (destination) là số bị chia chứa trong thanh ghi chứa AX hay cặp DX, AX ghi là (DX:AX).
- Số chia có thể có trong thanh ghi (Reg), ô nhớ (mem) nhưng không thể là hằng số.
- Kết quả của phép chia có thể là 8 bit thì:
  + AL chứa số thương Q.
  + AH chứa số dư.
- Kết quả của phép chia có thể là 16 bit thì:
  + AX chứa số thương.
  + DX chứa số dư (bảng 3.17a).
- Nếu kết quả của phép chia có thể là 32 bit (VXL ≥ 386) thì:
  + EAX chứa số thương.
  + EDX chứa số dư.

Phép chia có thể bị tràn (OF = 1) vì số chia quá nhỏ so với số bị chia.

Nếu chia hai số không dấu hoặc cùng dấu thì hai lệnh DIV và IDIV cho kết quả giống nhau.

*d) Các lệnh hiệu chỉnh phép nhân và chia AAM, AAD (bảng 3.17)*

- **Lệnh AAM** (Adjust AL after Multiplication): để hiệu chỉnh kết quả của phép nhân dạng nhị phân trong AX sang số BCD không nén trong AX.

- **Lệnh AAD** (ASCII Adjust before Division): để đổi hai số BCD không nén ở AH và AL sang hệ số 2 tương đương để trong AH và AL. Lệnh này phải đặt trước lệnh chia một số BCD không gói (gồm 2 chữ số) để trong AX cho một số BCD không nén khác. Trình tự biến đổi:

+ Số trong AH được nhân với 10 (vì hàng chục đổi thành đơn vị).

+ Cộng kết quả với số hàng đơn vị trong AL.

+ Biến đổi thành số nhị phân.

*Bảng 3.17* **a) Các lệnh nhân và chia số học**
**b) Toán hạng cho phép cho MUL, DIV và IDIV**
**c) Toán hạng cho phép cả lệnh IMUL**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị ảnh hưởng |
|---|---|---|---|---|
| MUL | Nhân (không dấu) | MUL S | (AL).(S8) → (AX)<br>(AL).(S16) → (DX), (AX)<br>(EAX).(S32) → (EDX), (EAX) | OF, CF<br>SF, ZF, AF, PF<br>Không xác định |
| DIV | Chia (không dấu) | DIV D, S | (1) Q((AX)/(S8)) (AL)<br>R((AX)/(S8) → (AX)<br>(2) Q((DX, AX)/(S16) → (AX)<br>R((DX, AX)/(S16) → (AX)<br>(3)Q((EDX, EAX)/(S32)) → (EAX)<br>R((EDX, EAX)/(S32)) → (EDX) | Tất cả các cờ không xác định |
| IMUL | Nhân số có dấu | IMUL S<br><br><br>IMUL R, I<br><br>IMUL S, I | (AL).(S8) → (AX)<br>(AL).(S16) → (DX), (AX)<br>(EAX).(S32) → (EDX), (EAX)<br>(R16).(Imm8) → (R16)<br>(R16).(Imm16) → (R16)<br>(S16).(Imm8) → (S16)<br>(S16).(Imm16) → (S16) | |
| IDIV | Chia số có dấu | IDIV S | DIV | |
| AAM | Adjust AL after Multiplication | AAM | Q((AL)/10) → AH<br>R((AL)/10) → AH | |
| AAD | Adjust AX before Division | AAD | (AH).10+AL → AL | |
| CBW | Convert byte to Word | CBW | (MSB AL) → (Các bit của AH) | Không |
| CWD | Convert Word to Double word | CWD | (MSB AX) → (Các bit của DX) | Không |
| CWDE | Như trên | CWDE | (MSB AX) → 16MSB của EAX | Không |
| CDQ | Convert Double word to quard word | CDQ | (MSB EAX) → Các bit của EAX | |

a)

*(tiếp bảng 3.17)*

| Destination | Source |
|---|---|
| AL | Reg 8 |
| AX | Reg 16 |
| EAX | Reg 32 |
| AL | Mem 8 |
| AX | Mem 16 |
| EAX | Mem 32 |

b)

| Destination (D) | Source (S) |
|---|---|
| | Reg 8 |
| | Reg 16 |
| | Reg 32 |
| Reg 16 | Imm 8 |
| Reg 32 | Imm 8 |
| Reg 16 | Imm 16 |
| Reg 32 | Imm 32 |
| Reg 16 | Reg 16, Imm 8 |
| Reg 32 | Reg 16, Imm 8 |
| Reg 32 | Mem 32, Imm 8 |
| Reg 16 | Reg 16, Imm 16 |
| Reg 16 | Mem 16, Imm 16 |
| Reg 32 | Reg 16, Imm 32 |
| Reg 32 | Mem 16, Imm 32 |
| Reg 16 | Reg 16 |
| Reg 16 | Mem 16 |
| Reg 32 | Reg 32 |
| Reg 32 | Mem 32 |

c)

*Ví dụ 1:* Dùng lệnh AAM: nhân 9x7 và chỉnh kết quả

```
MOV AL, 9    ; AL = 00001001B
MOV BL, 7    ; BL = 000000111B
MUL BL       ; AL = 00111111B
AAM          ; AH = 6D = 00000110B
             ; AL = 3D = 00000011B.
```

*Ví dụ 2:* Dùng lệnh AAD để chia hai số 63 dạng BCD, ta có: AH = 00000110B = 6D và AL = 0000 0011B = 3D chia cho BL = 00000111B = 7D

```
MOV AH, 6D
MOV AL, 3D
MOV BL, 7D
AAD          ; Biến đổi AL = 6x10+3 = 63 thành dạng nhị phân
             ; 00111111B = 3Fh.
DIV BL       ; Kết quả cho số thương chứa trong AL = 4 = 00000100B
             ; và số dư AH = 00001011B = 11D.
```

***e) Các lệnh mở rộng dấu: CBW, CWD, CWDE, CDQ***

Các lệnh mở rộng dấu này chuyển bit dấu (bit cao nhất - MSB) của 1 byte vào byte cao của thanh ghi cao của cặp thanh ghi.

*Ví dụ:* - Bit $D_7$ = 0 của AL sau lệnh CBW ta có AH = 00000000B = 00h.

- Bit $D_7$ = 1 của AL, sau lệnh CBW ta có AH = 11111111B = FFh.

Tương tự cho lệnh CWD có DX = 0000h hay DX = FFFFh.

Người ta dùng lệnh mở rộng dấu này trước một lệnh chia số bị chia có dấu (số âm).

*Ví dụ:* MOV AL, mem8
CBW BL, -25
IDIV BL.

### 3.5.3. NHÓM LỆNH LOGIC

Vi xử lý Intel 80x86 có các lệnh thực hiện các hành động logic AND (và), OR (hoặc), Exclusive OR (XOR- hoặc - loại trừ) và NOT (phủ định).

#### 1. Lệnh AND

Lệnh AND tiến hành trên các bit tương ứng của hai thanh ghi D và S. Kết quả như sau:

- Các bit tương ứng (cùng thứ tự trong byte hay từ) đều bằng 1, kết quả là 1.
- Một trong hai bit tương ứng bằng 0, kết quả là 0. Cả hai bit bằng 0, kết quả là 0.

*Ví dụ:*
```
        S  = 01110100B
        D  = 10010101B
  ------------------------
AND D, S  = 00010100B.
```

Lệnh AND thường dùng để:

- Che từng bit của một số liệu ở toán hạng nào đó, nếu lấy trị số của bit tương ứng của toán hạng đích là 0.

- Cho phép truyền qua từng bit của số liệu của toán hạng nào đó, nếu lấy trị số của bit tương ứng của toán hạng đích là 1.

Vậy dùng lệnh AND để lọc (0-chắn, 1-cho qua) một số liệu, tức là cần kiểm tra số liệu đó. Người ta hay dùng lệnh này trong việc kiểm tra 1 bit hay nhiều bit trạng thái của một thiết bị ngoại nối với vi xử lý. Khi đó chỉ cần đọc vào vi xử lý và lọc bởi lệnh AND những bit trạng thái nào cần xét đến.

*Ví dụ:* Cần kiểm tra các bit $D_7$, $D_5$, $D_3$, $D_1$ của một thanh ghi trạng thái, ta có giá trị của toán hạng đích là 10101010B = 0CAh và tiến hành lệnh AND 0CAh, trạng thái.

#### 2. Lệnh OR

*Ví dụ:*
```
      S  = 10110010B
      D  = 01101100B
  ----------------------
OR D, S  = 11111110B.
```

Qua ví dụ trên ta thấy:

- Nếu bit của toán hạng nguồn (S) là 1, lệnh vẫn giữ nguyên giá trị 1 của bit đó.
- Các bit 1 của toán hạng đích (D) là 1 làm kết quả lên 1, bất luận bit đó là 0 hay 1 của toán hạng nguồn.

Người ta dùng lệnh OR này để xác lập một hay nhiều bit của một số nào đó lên 1.

*Ví dụ: B*iến đổi sang mã ASCII một số, biết sang mã BCD của nó, ví dụ 5D.

Số 5D có mã nhị phân là 0000101B = 05h.

Lệnh OR 00110000B, 00000101B hay OR 30h, 05h; AL = 35h.

Vậy AL chứa mã ASCII của số 5D là 35h.

### 3. Lệnh XOR

*Ví dụ:* D = 10110101B

S = 01010110B

XOR D, S = 11100011B

Kết quả cho: + Là bit 0 nếu hai bit của D, S giống nhau (là 0 hay 1) tức loại trừ cùng loại.

+ Là bit 1 nếu hai bit của D, S là khác nhau.

Người ta dùng lệnh này để:

- Kiểm tra các bit khác nhau (hay giống nhau) của hai số liệu.

- Xóa thanh ghi nếu dùng lệnh XOR R, R tức loại trừ chính nó, để có các bit đều bằng 0 vì đều giống nhau.

### 4. Lệnh NOT

Lệnh này chỉ có một toán hạng đích (D), thường thực hiện ở thanh ghi chứa AX. Lệnh này đổi giá trị của bit $0 \rightarrow 1$ và $1 \rightarrow 0$ nên có tên là bù 1 (bù 2 = bù 1 + 1).

*Ví dụ:* NOT AL = - AL.

*Ví dụ* NOT (11001010B) = $\overline{11001010B}$ = 00110101B.

Nói khác đi, lệnh NOT thực hiện phép lấy đảo hay bù 1 của 1 số, tức lấy 1 trừ đi số đó:

NOT A = 1 - A = $\overline{A}$ = bù 1 của A.

Nếu $\overline{A}$ + 1 = bù 2 của A = NEGA.

Vậy: NEG A = NOT A + 1

*Ví dụ:* Tính bù 2 của số

| | | |
|---|---|---|
| A | = | 11001010B |
| NOTA | = | 00110101B |
| + 1 | = | + 1 |
| bù 2 của A = NEG A | = | 10110110B |
| A | = | 11001010B |
| NEGE + A | = | 100000000B |

CF

Số A cộng với bù 2 của nó bằng 0 và CF = 1 nên coi bù 2 là số đối của một số để thực hiện phép cộng với số âm trong phép tính số có dấu (integer).

### 5. Lệnh Test

Test D, S tác dụng lấy Đích^Gốc (hay D^S).

Đích, Gốc có thể là thanh ghi, ô nhớ cùng độ lớn (số bit) nhưng:

- Không thể cả hai là thanh ghi đoạn.

- Không thể cả hai là ô nhớ.

Lệnh này tương tự lệnh AND nhưng chỉ khác là Đích không chứa kết quả. Chỉ có các cờ bị ảnh hưởng (xóa CF, OF), ảnh hưởng tới PF, SF, ZF còn AF không xác định.

## 3.5.4. CÁC NHÓM LỆNH XỬ LÝ BIT

### 1. Các lệnh dịch vị trí của các bit

Có sáu lệnh dịch vị trí (Shift-SH) của một bit trong byte hay word (bảng 3.18 a, b, c).

Có thể dịch thanh ghi (Reg) dịch ô nhớ (Mem) với một số bit 8, 16, 32.

Số lần dịch có thể là số trực tiếp (Imm, tối đa 8 bit) hay nội dung của thanh ghi CL, CX.

Về chiều dịch có hai loại:

- Dịch trái (Left)
- Dịch phải (Right).

Về loại dịch có hai loại:

- Số học cho bit đầu tiên kể từ chiều dịch được giữ nguyên giá trị cũ
- Logic cho bit đầu tiên nạp số 0.

Người ta thường dùng lệnh dịch để thực hiện phép nhân (dịch trái) và chia (dịch phải) nhị phân (xem chương 6) hoặc để kiểm tra một bit của thanh ghi (chuyển qua CF để xét). Tùy giá trị của bit này có thể có các lệnh nhảy có điều kiện tương ứng (xem mục về các lệnh nhảy).

*Bảng 3.18* **a) Các lệnh dịch**

**b) Đích DI và đếm số lần các lệnh dịch một đích**

**c) Đích và đếm các lệnh hai đích (D1, D2)**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| SAL | Dịch trái số học | SAL D, Count | $0 \leftarrow D_7 \leftarrow D_0 \leftarrow 0$ | |
| SHL | Dịch trái logic | SHR D, Count | $0 \leftarrow D_7 \leftarrow D_0 \leftarrow 0$ | SF, ZF, PF, CF |
| SAR | Dịch phải số học | SAR D, Count | $D_7 \rightarrow D_0 \rightarrow C$ | AF không xác định |
| SHR | Dịch phải logic | SHR D, Count | $0 \rightarrow D_7 \rightarrow D_0 \rightarrow C$ | OF không xác định |
| SHLD | Dịch trái kép, chính xác | SHLD D1, D2, Count | $C \leftarrow D_1 \leftarrow D_2 \leftarrow 0$ | nếu count ≠ 0 |
| SHRD | Dịch phải kép, chính xác | SHRD D1, D2, Count | $0 \rightarrow D_2 \rightarrow D_1 \rightarrow C$ | |

a)

| Đích | Đếm (Count) |
|---|---|
| Reg | 1 |
| Reg | CL, |
| Reg | Im8 |
| Mem | 1 |
| Mem | CL, |
| Mem | Im8 |

b)

| $D_1$ | $D_2$ | Count |
|---|---|---|
| Reg16 | Reg16 | Im8 |
| Reg16 | Reg16 | CL, CX |
| Reg32 | Reg32 | Im8 |
| Reg32 | Reg32 | CL, CX |
| Reg16 | Reg16 | Im8 |
| Reg16 | Reg16 | CL, CX |
| Reg32 | Reg32 | Im8 |
| Reg32 | Reg32 | CL, CX |

c)

## 2. Các lệnh quay vòng các bit

Một thanh ghi, một ô nhớ có thể thực hiện lệnh quay (Rotation) các bit tức dịch chuyển các bit theo một vòng kín và cả với bit cờ CF (bảng 3.19).

Về loại có hai lệnh:

- Quay không qua CF, đó là RO như ROL (quay trái) và ROR (quay phải).
- Quay qua CF, đó là RC như RCL (quay trái qua CF, tức CF nằm tận cùng phía trái của thanh ghi), và RCR (quay phải qua CF, tức CF nằm tận cùng phía phải của thanh ghi).

Về chiều quay có hai lệnh:

- Quay trái: ROL, RCL.
- Quay phải: ROR, RCR.

Chỉ có lệnh quay với một đích D là thanh ghi hoặc ô nhớ.

Số lần quay có thể là 1, số trực tiếp (Imm-8 bit) hay trong thanh ghi CL.

Người ta sử dụng các lệnh quay này để thực hiện phép nhân, chia, kiểm tra bit nhưng vẫn lưu giữ được giá trị của thanh ghi hay ô nhớ và nén hay giãn các số BCD.

*Ví dụ:* Kiểm tra bit thứ tự $D_3$ của một byte số liệu nhưng cần duy trì lưu giữ byte số liệu đó.

```
MOV AL, data
MOV CL, 4          ; số lần quay
ROR AL, CL         ; quay phải 4 lần
JNC                ; nhảy nếu C = D3 = 0
ROR AL, CL         ; quay tiếp để hồi phục lại giá trị cũ
END
```

*Ghi chú:* Ví dụ trên có thể sử dụng:

- Lệnh quay trái 5 lần.

- Lệnh quay phải qua Carry Flag (CF) RCR 4 lần hay quay trái qua C RCL 5 lần, còn hồi phục lại là 5 lần quay phải hoặc trái 4 lần.

*Bảng 3.19* **a) Các lệnh quay**

**b) Đích và đếm lần quay**

| Đích (D) | Đếm (Count) |
|---|---|
| Reg | 1 |
| Reg | CL |
| Reg | Im8 |
| Mem | 1 |
| Mem | CL |
| Mem | Im8 |

a)

| Thuật nhớ | Ý nghĩa | Dạng | Cờ bị tác động |
|---|---|---|---|
| ROL | Quay trái | ROL D, count | CF |
| ROR | Quay phải | ROR D, count | OF không xác định nếu count ≠ 0 |
| RCL | Quay trái qua CF | RCL D, count | CF |
| RCR | Quay phải qua CF | RCR D, count | OF không xác định nếu count ≠ 0 |

b)

## 3. Các lệnh kiểm tra (test) và quét (scan) bit

Các lệnh này có ở 80386, 80486. Bảng các lệnh này như ở bảng 3.20 gồm:

- *Các lệnh kiểm tra bit (BT, BTR, BTS, BTC):* đây là các hành động kiểm tra (thử) mức logic (1 hay 0) của một bit ở trong thanh ghi (Reg16, Reg 32) hay vùng nhớ (Mem 16, Mem 32). Khi lệnh được thực hiện, giá trị của bit được kiểm tra gửi vào carry flag (CF). Sau khi kiểm tra lệnh còn có thể xóa (reset-R) xác lập (set-S) và lấy bù (Completment-C) hay đảo ngược giá trị của bit CF.

- *Các lệnh quét bit (BSF, BSR):* để kiểm tra lần lượt (theo chiều tiến - Forward, theo chiều lùi hay ngược - Reverse).

Mỗi lần bit kiểm tra được gửi vào CF, quá trình tiếp theo có thể được thực hiện bởi phần mềm (chương trình). Khi ấy một lệnh nhảy có điều kiện có thể được dùng để kiểm tra giá trị có trong CF. Nếu *CF = 1*, giá trị của chỉ thị sẽ được tăng một sự nhảy trở về lệnh BT để so sánh tiếp bit khác của toán hạng được kiểm tra. Chú ý rằng chương trình con này quét các bit của toán hạng khi thấy bit đầu tiên là logic 1.

*Ví dụ 1:* BTR EAX, EDI.

Thanh ghi 32 bit EAX được chọn bởi chỉ số (Index) trong EDI đã được chọn. Giá trị của bit kiểm tra đầu tiên được lưu giữ vào CF và bit này trong thanh ghi EAX được xoá.

*Bảng 3.20.* **a) Các lệnh logic**
**b) Toán hạng cho phép của AND, OR, XOR**
**c) Toán hạng cho phép của NOT**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| AND | Logical AND | AND D, S | (S).(D) → (D) | OF, SF, ZF, PF, CF, AF không xác định |
| OR | Logical | OR D, S | (s) + (D) → (D) | Như trên |
| XOR | Logical exclusive OR | XOR D, S | (S).(D) → (D) | Như trên |
| NOT | Logical NOT | NOT D | (D) → (D) | Không |
| TEST | Test | TEST D, S | (S).(D) | PF, SF, ZF, CF, OF |

a)

| Đích (D) | Nguồn (S) |
|---|---|
| Thanh ghi (R) | Ô nhớ M |
| Thanh ghi (R) | Thanh ghi (R) |
| Ô nhớ M | Thanh ghi (R) |
| Thanh ghi (R) | Tức thì (Imm) |
| Ô nhớ M | Tức thì (Imm) |
| Thanh ghi chứa AX | Tức thì (Imm) |

b)

| Đích |
|---|
| Thanh ghi (R) |
| Ô nhớ M |

c)

*Bảng 3.21.* **Các lệnh xử lý bit**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| BT | Bit Test | BT D, S | Lưu giá trị của bit trong D (xác định bởi giá trị trong S) vào CF | CF (các cờ cấu khác không xác định) |
| BTR | Bit Test and Reset | BTR D, S | Như trên và xoá bit vừa kiểm tra | |
| BTS | Bit Test and Set | BTS D, S | Như trên và xác lập bit vừa kiểm tra | |
| BTC | Bit Test and Completment | BTC D, S | Như trên và lấy bù bit vừa kiểm tra | Như trên |
| BSF | Bit scan forward | BSF D, S | Quét toán hạng nguồn từ bit 0 ZF=0 nếu các bit là 0, khác đi thì ZF=1 và đích được tính với chỉ số của bit đầu tiên của bộ bit | Như trên |
| BSR | Bit scan reverse | BSR D, S | Quét toán hạng nguồn bắt đầu từ MSB. ZF=0 nếu các bit là 0, khác đi thì ZF=1 và đích được nạp với chỉ số của bit đầu tiên của bộ bit | Như trên |

a)

| Đích (D) | Nguồn (S) |
|---|---|
| Reg 16 | Reg 16 |
| Reg 16 | Imm8 |
| Reg 32 | Reg 32 |
| Reg 32 | Imm8 |
| Mem 16 | Reg 16 |
| Mem 16 | Imm8 |
| Mem 16 | Reg 32 |
| Mem 32 | Imm8 |

b)

| Đích (D) | Nguồn (S) |
|---|---|
| Reg 16 | Reg 16 |
| Reg 16 | Reg 16 |
| Reg 32 | Reg 32 |
| Reg 32 | Reg 32 |

c)

*Ví dụ 2:* Mô tả hành động thực hiện bởi lệnh: BTC BX, 7

Giả sử rằng thanh ghi BX chứa giá trị 03FOh.

Giải: (BX) = 0000 0011 1111 0000B.

Thực hiện kiểm tra bit và lấy bù gây ra giá trị của bit thứ 8 được kiểm tra đầu tiên và lấy bù. Vì bit này là logic 1, CF xác lập lên 1. Chiều này cho:

CF = 1B

(BX) = 0000.0011.0111.0000B = 037h.

*Ví dụ 3:* Lệnh BSF ESI, EDX.

Các bit của thanh ghi EDX 32 bit được kiểm tra liên tiếp bắt đầu từ bit 0. Nếu tất cả các bit được tìm là 0, cờ ZF bị xóa. Mặt khác nếu nội dung của EDX không bằng 0, ZF được xác lập lên 1 và giá trị chỉ thị (index value) của bit thứ nhất được kiểm tra là 1, được chép vào thanh ghi đích ESI. Các chỉ thị này bằng giá trị của vị trí bit cộng 1.

## 3.5.5. CÁC LỆNH XỬ LÝ CHUỖI

Các lệnh này tác động lên một chuỗi số liệu nhiều byte, nhiều word hay word kép nằm ở các địa chỉ liên tiếp bộ nhớ. Bảng 3.22 giới thiệu các lệnh xử lý chuỗi của 80x86, chia thành các nhóm:

- Chuyển chuỗi (MOV) theo byte (B), word (W) và word kép (D).
- So sánh chuỗi (CMP) theo byte (B), word (W) và word kép (D).
- Quét chuỗi (SCAN) theo byte (B), word (W) và word kép (D).
- Lưu chuỗi (Store) theo byte (B), word (W) và word kép (D).

Các lệnh chuyển số liệu (MOV), so sánh (CMP) đều có hai toán hạng, các lệnh còn lại (quét, nạp, lưu) có một toán hạng, còn toán hạng kia là thanh ghi chứa (AL, AX, EAX).

Toán hạng đều là nội dung của các ô nhớ thuộc mảng DS và ES với địa chỉ ở thanh ghi chỉ thị nguồn (SI) và chỉ thị đích (DI). Tùy theo giá trị của cờ hướng (DF) ta có:

- DF = 0 địa chỉ tăng dần.
- DF = 1 địa chỉ giảm dần.

Tùy lệnh, ta có sự thay đổi địa chỉ một lượng n với:

n = 1 lệnh theo byte (B).

n = 2 lệnh theo word (W).

n = 4 lệnh theo Double word (D).

### 1. Lệnh chuyển chuỗi MOVS

Chuyển từng chuỗi nội dung của mảng nhớ có địa chỉ nguồn DS : SI sang mảng nhớ có địa chỉ lại tăng (DF = 0) hay giảm (DF = 1) đi 1 (đối với lệnh B), đi 2 (lệnh W) và 4 (lệnh D).

Ta có: ((ES)0 + (DI ± n)) ← ((DS)0 + (SI ± n))

*Ví dụ:*

```
MOV AX, ADDR
MOV DS, AX
MOV ES, AX
MOV SI, ADDR1
MOV DI, ADDR2
```

```
        MOV CX, N
        CLD
NX:     MOV B
        LOOP NX
```

*Bảng 3.22.* **Các lệnh xử lý chuỗi**

| Thuật nhớ | Ý nghĩa | Dạng | Hành đông | Cờ bị tác động |
|---|---|---|---|---|
| MOVS B/<br>MOVS W/<br>MOVS D/ | Move String<br>Move Word<br>Move Double Word | MOVS B<br>MOVS W<br>MOVS D | $M_{SL} \rightarrow M_{DD}$, SI ± n → SI<br>DI ± n → DI.<br>n = 1: Byte; n = 2:Word<br>n = 4: Double word | Không |
| CMPS B/<br>CMPS W/<br>CMPS D/ | Compare String<br>Compare Word<br>Compare Double Word | CMPS B<br>CMPS W<br>CMPS D | $M_{SL} \rightarrow M_{DD}$, SI ± n → SI<br>DI ± n → DI. | CF, PF, AF, ZF, SF, OF |
| SCAS B/<br>SCAS W/<br>SCAS D/ | Scan String<br>Scan Word<br>Scan Double Word | SCAS B<br>SCAS W<br>SCAS D | AL → $M_{DI}$<br>DI ± n → DI .<br>AX → $M_{DI}$ | CF, PF, AF, ZF, SF, OF |
| LODS B/<br>LODS W/<br>LODS D/ | Load String<br>Load Word<br>Load Double Word | LODS B<br>LODS W<br>LODS D | $M_{SI}$ → AL; SI ± n → SI | Không |
| STOS B/<br>STOS W/<br>STOS D/ | Store String<br>Store Word<br>Store Double Word | STOS B<br>STOS W<br>STOS D | AL → $M_{DI}$ ; DI ± n → DI | Không |
| INS<br>OUTS | Vào chuỗi<br>Ra chuỗi | INS<br>OUTS | [DX] → (AX)<br>(AX) → [DX] | Không |

## 2. Lệnh so sánh chuỗi CMPS

So sánh từng cặp chuỗi thuộc hai mảng nhớ có địa chỉ tương ứng ở DS : SI và ES : DI. Sau mỗi lần so sánh các bit cờ bị ảnh hưởng tương tự lệnh so sánh CMP hai số. Có thể biểu diễn lệnh so sánh CMPS là **((ES)0 + (DI ± n)) ← ((DS)0 + (SI ± n)).**

## 3. Lệnh quét chuỗi SCAS

Thực hiện so sánh nội dung nhớ trong ES:DI với nội dung một chuỗi nằm trong AL, AX hoặc EAX.

Ta có: (AL, AX hay EAX) - ((ES)0 + (DI ± n))

Các cờ bị ảnh hưởng là CF, PF, AF, ZF, SF và OF.

*Ví dụ:*

```
MOV AX, 00h ;
MOV DS, AX
MOV ES, AX
MOV AL, 00h
MOV DI, A0000h
MOV CX, 0Fh
```

```
        CLD
AGAIN:  SCASB
        LOOPNE AGAIN
```

## 4. Lệnh nạp chuỗi LODS

Lệnh nạp từng byte, word, word kép và AL, AX hay EAX. Sau đó lại tiếp tục nạp với địa chỉ tăng (DF = 0) hay giảm (DF = 1) một lượng 1, 2, 4 byte.

## 5. Lệnh lưu (Store) chuỗi bit vào bộ nhớ STOS

Lệnh này thực hiện việc chuyển byte số liệu trong AL, lời trong AX hay lời kép trong EAX vào các ô nhớ có địa chỉ (ES : DI ± n).

Ta có: **((ES0 +(DI ±n)) ← (AL, AX hay EAX).**

*Ví dụ:*

```
        MOV AH, 00h
        MOV DS, AX
        MOV ES, AX
        MOV AL, 05h
        MOV DI, A0000h
        MOV CX, 0Fh
        CLD
AGAIN:  STOSB
        LOOPNE AGAIN.
```

## 6. Các lệnh nhận vào INS và đưa ra chuỗi OUTS

Lệnh này nhận số liệu vào hoặc đưa số liệu ra qua AX và cảng có địa chỉ DX.

***Bảng 3.23.*** **Các tiếp đầu ngữ dùng cho các lệnh chuỗi cơ sở**

| Tiếp đầu ngữ | Dùng với | Ý nghĩa |
|---|---|---|
| REP | MOVS | Lặp lại khi CX ≠ 0 |
| REPE/ | STOS | Lặp lại khi CX ≠ 0 và chuỗi bằng nhau (ZF = 1) |
| REPZ | CMPS | Lặp lại khi CX ≠ 0 và chuỗi không bằng nhau (ZF = 0) |
| REPNE/ | SCAS | |
| REPN | CMPS | |
| | SCAS | |

Các lệnh xử lý chuỗi trên có thể được lặp lại nhiều lần cho hết cả mảng nhớ đích và nguồn. Các lệnh trên lặp lại khi có các tiếp đầu ngữ:

- **REP:** lặp lại không điều kiện cho các lệnh MOVS, LODS, STOS. Mỗi lần thực hiện xong lệnh, số đếm trong CL giảm đi 1, khi CL = 0, sự lặp lại kết thúc.
- **REPE:** lặp lại nếu bằng (equal) tức cờ ZF = 1 của sự so sánh CL với 0.
- **REPNE:** lặp lại nếu không bằng, tức CL không bằng 0.
- **REPZ:** lặp lại nếu kết quả có cờ ZF = 1, giống REPE.
- **REPNZ:** lặp lại nếu kết quả có cờ ZF = 0 tức CL khác 0.

Các tiếp đầu ngữ lặp lại này dùng cho các lệnh so sánh (CMPS) và quét (SCAS) để tiến hành so sánh tiếp hay dừng lại (bảng 3.23).

## 3.5.6. NHÓM LỆNH RẼ NHÁNH CHƯƠNG TRÌNH

Các lệnh rẽ nhánh chương trình làm thay đổi thứ tự thực hiện các lệnh của chương trình, tức thay đổi đột ngột địa chỉ ô nhớ chứa chương trình cần thực hiện (CS : IP). Trong các lệnh này, toán hạng là địa chỉ mà lệnh phải tính toán để đưa ra địa chỉ hiệu dụng EA (hay địa chỉ vật lý PA) cho lệnh của chương trình cần chuyển tới.

Có các loại rẽ nhánh sau:

- *Rẽ nhánh không điều kiện:* lệnh gọi chương trình con, ngắt chương trình, **nhảy** chương trình, trở về v.v... không cần một điều kiện nào.

- *Rẽ nhánh có điều kiện:* điều kiện đây là kết quả của sự so sánh, giá trị của một bit cờ v.v...

- *Vòng lặp có điều kiện:* lặp lại một số lệnh, một số lần với điều kiện nào đó.

### 1. Các lệnh rẽ nhánh không điều kiện

Các lệnh thuộc loại này được giới thiệu ở bảng 3.24.

Các lệnh rẽ nhánh không điều kiện có thể chia làm hai nhóm:

- *Nhóm rẽ nhánh không trở về:* lệnh JMP, chương trình **nhảy** đến đoạn chương trình khác có địa chỉ ở toán hạng (xem bảng 3.24b).

- *Nhóm rẽ nhánh có trở về:* để tiếp tục thực hiện tiếp chương trình chính bị gián đoạn. Nhóm này lại chia làm hai nhóm nhỏ:

+ **Call - Ret:** ngắt chương trình chính gọi chương trình con để thực hiện, khi gặp lệnh RET ở chương trình con sẽ trở về thực hiện tiếp chương trình chính ở chỗ bị ngắt.

+ **INT nh - IRET:** ngắt chương trình chính, chuyển tới chương trình con phục vụ ngắt và trở về chỗ bị ngắt của chương trình chính khi gặp lệnh IRET ở chương trình con phục vụ ngắt. Riêng INTO là trường hợp đặc biệt (n = 0) của INTn, gọi ngắt khi có sự tràn số liệu (chia cho 0 hay số rất nhỏ, thanh ghi không chứa đủ số liệu của số thương), máy bị treo. Các toán hạng của hai lệnh Call và JMP hầu như giống nhau, trừ lệnh JMP có thêm lệnh nhảy ngắn. Lệnh Call gọi tên chương trình con còn lệnh nhảy JMP tới nhãn (chỉ thị bằng địa chỉ).

*Lệnh gọi Call*

Gọi chương trình con với tên (thay nhãn). Có hai loại gọi chương trình con.

- **Gọi gần:** địa chỉ của lệnh đầu tiên của chương trình con nằm trong cùng mảng mã với chương trình chính (cùng CS) chỉ có giá trị IP là khác nhau.

- **Gọi xa:** địa chỉ chương trình con nằm ở mảng nhớ khác tức có giá trị CS:IP khác nhau.

Thủ tục gọi sau khi gặp lệnh Call và trở về (sau khi gặp lệnh RET) của vi xử lý như sau:

**Lệnh gọi gần**

• *Thủ tục*:

- Chuyển tới chương trình con:

+ IP → (SP) với lệnh PUSH: độ rời của lệnh tiếp theo của chương trình chính gửi vào Stack. SP - 2 → SP: địa chỉ Stack giảm đi 2.

+ Offset của chương trình con → IP: địa chỉ lệnh (Offset) của lệnh đầu tiên của chương trình con gửi vào IP.

- Trở về chương trình chính để tiếp tục tại chỗ bị gián đoạn. Khi gặp lệnh RET:

+ (SP) → IP hồi phục IR cũ bởi lệnh POP.

SP + 2 → SP: thanh ghi địa chỉ Stack (con trỏ ngăn xếp) tăng lên 2 đơn vị.

Với gọi gần này, ta có hai cách viết lệnh gọi tùy chế độ địa chỉ.

• *Cú pháp:* **Call NPROC**

Nhãn NPROC chỉ gọi chương trình con ở gần (cùng đoạn mã CS) với chương trình chính. NPROC xác định một độ dời d 16 bit, được mã như là một toán hạng trực tiếp theo sau mã lệnh (địa chỉ tức thì). Như vậy, đó là phép địa chỉ độ dời (offset address) tương đối với dịch chuyển bị giới hạn bởi ± 32K bytes.

• *Call BX*

Đây là toán hạng memptr 16 và Regptr16 tức địa chỉ gián tiếp (ô nhớ 16, thanh ghi 16 là con trỏ địa chỉ). Khi lệnh được thực hiện nội dung của BX được nạp vào IP và tiếp tục thực hiện lệnh với IP có giá trị mới là BX, chương trình con bắt đầu ở địa chỉ vật lý (PA) tính từ CS không đổi và giá trị mới trong IP.

• *Call word PTR [BX]:* (giống trường hợp trên cho dễ dịch) gọi chương trình con có địa chỉ gián tiếp ở vùng nhớ mà địa chỉ là nội dung của DS và BX (bảng 3.24).

***Bảng 3.24*** **a) Các lệnh rẽ nhánh không điều kiện**

**b) Các toán hạng của các lệnh Call và JMP**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| Call | Gọi chương trình con | Call tên. | Việc thực hiện được tiếp tục ở địa chỉ chương trình con xác định bởi tên (toán hạng) | Không |
| INTn | Gọi chương trình phục vụ ngắt thứ n | INT nh | Ngắt chương trình chuyển sang phục vụ chương trình con phục vụ ngắt | Không |
| INT0 | Gọi chương trình thực hiện ngắt 0 (bị tràn số liệu) | INT0 | Phép chia cho 0, hoặc số quá bé, kết quả quá lớn, dừng máy | Không |
| RET | Trở về từ chương trình con | RET hay RET operand | Trở về chương trình chính từ chương trình con bằng cách phục hồi IP và CS (Far Process) | Không |
| IRET | Trở về từ ngắt | IRET | | Không |
| JMP | Nhảy chương trình | JMP nhãn | Nhảy tới lệnh ở nơi khác trong chương trình | Không |

a)

| Operand của Call | Operand của JMP |
|---|---|
| | Short Label |
| Near Proc | Near Label |
| Far Proc | Far Label |
| Memptr16 | Memptr16 |
| Regptr16 | Regptr16 |
| Memptr32 | Memptr32 |
| Regptr32 | Regptr32 |

b)

**Lệnh gọi xa**

• *Thủ tục*

- Chuyển tới chương trình con:

+ CS → (SP) bởi lệnh PUSH: mảng lệnh của chương trình chính gửi vào Stack. SP - 2 → SP.

+ Offset độ lệch của chương trình con → IP.

+ Địa chỉ cơ sở BX của chương trình con → CS.

- Trở về chương trình chính để tiếp tục tại chỗ bị gián đoạn khi gặp lệnh RET:

+ (SP) → CS: hồi phục CS cũ bởi lệnh POP.

SP + 2 → SP: tăng địa chỉ Stack hai đơn vị.

+ (SP) → IP hồi phục IP cũ bởi lệnh POP.

Giữa gọi gần và gọi xa chỉ khác nhau về thủ tục gửi và hồi phục địa chỉ của mảng nhớ CS.

• *Cú pháp:* - **Call FPROC**

Chương trình con FPROC có toán hạng tức thì 32 bit được tích ở 4 byte theo ngay sau mã lệnh Call. Hai byte này được nạp trực tiếp vào CS và IP.

**- Call DWord PTR [DI]**

Đây là địa chỉ gián tiếp, địa chỉ vật lý của con trỏ 4 byte trong bộ nhớ được lấy từ nội dung của DS và DI.

**- Lệnh trở về RET**

Khi gặp lệnh RET ở chương trình con, vi xử lý hồi phục các giá trị đã cất giữ của các thanh ghi CS, IP (đổi các lệnh POP) tiếp tục thực hiện chương trình chính.

**- Lệnh nhảy JMP**

Cũng giống lệnh Call, cũng có nhảy gần, nhảy xa tới địa chỉ trực tiếp và gián tiếp. Khác với lệnh Call, lệnh JMP không có sự lưu trữ tin của VXL trước khi thực hiện lệnh nhảy và trở về chương trình chính mà chỉ có sự chuyển sang đoạn chương trình mới bởi việc thay giá trị mới của địa chỉ lệnh vào CS, IP.

+ *Với lệnh nhảy ngắn* (Short operand): chỉ thay IP + dịch chuyển → IP chỉ có thêm dịch chuyển (8 bit) chứa trong lệnh (sau mã lệnh).

*Cú pháp:* **JMP Short Nhãn**

+ *Với lệnh nhảy gần* (near) cũng thay như trên nhưng với dịch chuyển 16 bit (phạm vi ± 32 Kbyte)

**JMP BX:** thay nội dung của BX (16 bit) cho IP, địa chỉ trực tiếp.

**JMP [BX]:** địa chỉ gián tiếp.

+ *Với lệnh nhảy xa:* operand là Farlabel; Regptr 32 và memptr 32.

- **Farlabel:** có địa chỉ trực tiếp, với 32 bit số liệu tức thì với 16 bit nạp vào IP và 16 bit số liệu còn lại nạp vào CS (đoạn mã 64Kbyte).

+ **Regptr 32:** có địa chỉ trực tiếp với số liệu 32 có trong thanh ghi chỉ địa chỉ.

*Cú pháp:* **JMP DS:SI**

+ Memptr 32 địa chỉ gián tiếp, giá trị của CS và IP nằm trong ô nhớ có địa chỉ ở DS:DI.

+ JMP DWord PTr[DI]

**- Lệnh gọi ngắt INTn/IRET**

Với n = 1: FFH. Như vậy chương trình có thể gọi 256 mức ngắt, tương ứng 256 véctơ ngắt. Khi có lệnh INT thì:

+ FR→ (SP) bởi lệnh PUSH, gửi thanh ghi cờ vào Stack
SP - 2 → SP giảm SP đi 2 .

+ O → IF (cấm ngắt khác tác động), 0 → TF không có bẫy.

+ CS → (SP) bởi lệnh Push, gửi thanh ghi mảng vào Stack
SP - 2 → SP.

+ (Nx4) → IP chuyển sang đọc véctơ ngắt.

+ (Nx4+2) để chuyển sang chương trình con phục vụ ngắt.

Lệnh IRET cũng tương tự lệnh RET. Khi chương trình con phục vụ ngắt gặp lệnh IRET, các thanh ghi IP, CS, FR của vi xử lý được hồi phục lại giá trị cũ (có trước khi chuyển sang chương trình con phục vụ ngắt) bởi các lệnh POP. Sau đó chương trình chính lại tiếp tục.

## 2. Các lệnh nhảy có điều kiện Jcondition

Lệnh nhảy có điều kiện có dạng tổng quát như bảng 3.25a và danh sách ở bảng 3.25b. Lệnh nhảy có điều kiện này chỉ thực hiện nhảy (thay đổi trình tự thực hiện chương trình) khi điều kiện xảy ra. Đó là sự có mặt hay không có mặt (xảy ra hay không xảy ra) của một số điều kiện của trạng thái sau một phép kiểm tra (lệnh kiểm tra Test hay so sánh - CMP). Các điều kiện ở đây là:

- Các bit của thanh ghi cờ (Flag) như CF (Carry), ZF (Zero), OF (Over), PF (Parity), SF (Sign).

- Sự tương quan đơn và kép hai điều kiện giữa hai đại lượng so sánh (bằng, lớn hơn, nhỏ hơn).

- Nội dung bằng 0 của thanh ghi CX, ECX.

*Một số chú ý khi sử dụng các lệnh trên:*

- Các lệnh JE tương đương JZ.

- Các lệnh Above (trên và below (thấp) dùng cho sự so sánh các số không dấu (số dương).

- Các lệnh less (nhỏ hơn) và greater (lớn hơn) cho phép so sánh các số có dấu.

- Các cặp lệnh sau tương đương: JNA - JBE; JNAE - JB; JNB - JAE; JNBE - JA; JNC - JAE; JNG - JLE; JNGE - JL; JNL - JGE; JNLE - JG.

## 3. Các lệnh xác lập số liệu có điều kiện SETcondition (SET cc)

Cú pháp SETccD đích D được xác lập hay (1) → d. Khi có điều kiện thỏa mãn (như của lệnh nhảy Jcc. Nếu điều kiện không xảy ra, đích 0 được nạp 0 (0→D). D là Reg 8 hay M6M8.

## 4. Các lệnh vòng lặp LOOP

Có ba lệnh vòng lặp thực hiện lặp lại các lệnh của một đoạn chương trình. Việc lặp lại chỉ xảy ra hay ra khỏi vòng lặp với điều kiện:

- Kết thúc một số lần lặp (CX = 0).

- Lặp lại khi bằng/lặp lại khi 0.

- Lặp lại khi không bằng/lặp lại khi không bằng 0.

Danh sách các lệnh lặp lại như bảng của bảng 3.26.

*Ví dụ:*

```
        MOV AX, Add0
        MOV DS, AX
        MOV SI, Add1
        MOV DI, Add2
        MOV CX, n
NXT:    MOV AH, [ST]
        MOV [DI], AH
        INC SI
        INC DI
        LOOP NXT
        HLT
```

*Bảng 3.25.* **Các lệnh nhảy có điều kiện**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động | Cờ bị tác động |
|---|---|---|---|---|
| Jec | Nhảy có điều kiện. | Jec Operand. | Nếu điều kiện là thực (True) chương trình nhảy tới lệnh có địa chỉ ở Operand. Nếu không thì tiếp tục chương trình chính. | Không |

a)

| Thuật nhớ | Ý nghĩa | Điều kiện |
|---|---|---|
| JA (>) | Nhảy nếu lớn hơn (above) | CF = 0 và ZF = 0 |
| JAE(.) | Nhảy nếu lớn hơn hay bằng (above, equal) | CF = 0 |
| JB(<) | Nhảy nếu nhỏ hơn (below) | CF = 1 |
| JBE(≤) | Nhảy nếu nhỏ hơn hay bằng (below, equal) | CF = 1 hoặc ZF = 1 |
| JC | Nhảy nếu có nhớ | CF = 1 |
| JCXZ | Nhảy nếu có thanh ghi CX = 0 | CX = 0000h |
| JECXZ | Nhảy nếu thanh ghi ECX = 0 | ECX = 0000h |
| JE | Nhảy nếu bằng | ZF = 1 |
| JG | Nhảy nếu lớn hơn | ZF = 0 và SF = OF |
| JGE | Nhảy nếu lớn hơn hoặc bằng | SF = OF |
| JL | Nhảy nếu bé hơn. | (SF X or OF) = 1 |
| JLE | Nhảy nếu bé hơn hoặc bằng | ((SF X or OF) orZF) = 1 |
| JNA | Nhảy nếu không lớn hơn | CF = 1 hoặc ZF = 1 |
| JNAE | Nhảy nếu không lớn hơn, không bằng | CF = 1 |
| JNB | Nhảy nếu không nhỏ hơn | CF = 0 |
| JNBE | Nhảy nếu không nhỏ hơn, không bằng | CF = 0 và ZF = 0 |
| JNC | Nhảy nếu không nhớ | CF = 0 |
| JNE | Nhảy nếu không bằng | ZF = 0 |
| JNG | Nhảy nếu không lớn hơn | ((SF X or OF) orZF) = 1 |
| JNGE | Nhảy nếu không lớn hơn hay không bằng | |
| JNL | Nhảy nếu không nhỏ hơn | SF = OF |
| JNLE | Nhảy nếu không nhỏ, không bằng | ZF = 0 và SF = OF |
| JNO | Nhảy nếu không tràn | OF = 0 |
| JNP | Nhảy nếu không chẵn | PF = 0 |
| JNS | Nhảy nếu không có dấu (dương) | SF = 0 |
| JNZ | Nhảy nếu không bằng 0 | ZF = 0 |
| JO | Nhảy nếu tràn | OF = 1 |
| JP | Nhảy nếu chẵn | PF = 1 |
| JPE | Nhảy nếu tính chẵn lẻ là chẵn | PF = 1 |
| JPO | Nhảy nếu tính chẵn lẻ là lẻ | PF = 0 |
| JS | Nhảy nếu có dấu (số âm) | SF = 1 |
| JZ | Nhảy nếu là không | ZF = 1 |
| JCXZ | Nhảy nếu CX = 0 | ZF = 1 |

b)

## 3.5.7. CÁC LỆNH ĐIỀU KHIỂN VI XỬ LÝ

Các lệnh điều khiển vi xử lý có thể chia làm ba nhóm nhỏ là đồng bộ ngoại, xác lập/xóa các bit cờ và điều khiển vi xử lý.

## 1. Nhóm đồng bộ ngoài

- ESC.
- HLT: dừng thực hiện lệnh của chương trình.
- LOCK: khóa đường dây.
- NOP: không hành động gì.
- WAIT: chờ.

## 2. Nhóm xác lập/xóa các bit cờ

- CLC : xóa cờ nhớ (CF = 0)
- CLD : xóa cờ hướng (DF = 0)
- CLI : xóa cờ ngắt (IF = 0)
- CMC : lấy bù cờ nhớ (CF = 0)
- STC : xác lập CF (CF = 1)
- STD : xác lập DF (DF = 1)
- STI : xác lập IF (IF = 1)

***Bảng 3.26.*** **Các lệnh lặp**

| Thuật nhớ | Ý nghĩa | Dạng | Hành động |
|---|---|---|---|
| LOOP | | LOOP Short Label | (CX), (CX)-1. Nhảy tới vùng xác định bởi Short Label nếu (CX)≠ 0. Nếu (CX) = 0 thì tiếp tục thực hiện lệnh tiếp theo lệnh lặp. |
| LOOPE/LOOPZ | Lặp vòng khi bằng/ lặp lại khi bằng 0. | LOOPE/LOOPZ Short Label | Nhảy tới điều kiện (ZF = 1) và (CX) ≠ 0.(CX), (CX) -1. |
| LOOPNE/LOOPNZ | Lặp vòng khi không bằng/lặp lại khi khác 0. | LOOP | Nhảy tới điều kiện (CX) ≠ 0 và ZF = 0. |

## 3. Nhóm lệnh điều khiển

- ARPL mức đặc quyền đòi hỏi hiệu chỉnh.
- CLTS xóa cờ chuyển mạch nhiệm vụ.
- INVD số liệu ẩn không giá trị (invalid data cache).
- INVLPG vào TLB không còn giá trị.
- VERW cho phép đọc các mảng (verify write).
- WBINVD ghi trở lại và số liệu ẩn không giá trị.

Các lệnh điều khiển trên đều không có toán hạng.

## 3.5.8. CÁC LỆNH HỖ TRỢ NGÔN NGỮ BẬC CAO

Với vi xử lý thế hệ sau 80 286 có thêm các lệnh hỗ trợ các ngôn ngữ bậc cao sau:

- BOWND: Check carry bounds tìm biên giới mạng.
- CMPXCHG xếp lại số liệu 16 bit và so sánh (compare and exchange).
- ENTER vào một khung Stack (stack frame).
- Leave ra khỏi khung Stack (stack frame).
- XADD đổi và cộng (exchange and add).

# CÂU HỎI

1. Các dạng ngôn ngữ một lệnh của máy vi tính, ưu nhược điểm của từng loại ?

2. Dạng tổng quát của lệnh dạng mã máy, phân loại theo độ dài, theo số toán hạng và theo hành động.

3. Các dạng địa chỉ của một toán hạng, kể các loại toán hạng theo số liệu trực tiếp, thanh ghi và bộ nhớ.

4. Những thanh ghi nào trong vi xử lý tham gia vào việc quản lý địa chỉ của một máy vi tính.

5. Các dạng địa chỉ của một lệnh có hai toán hạng, lấy ví dụ lệnh MOV B, A.

6. Quá trình và thời gian thực hiện một lệnh của vi xử lý.

7. Các loại của nhóm lệnh chuyển dữ liệu. Vẽ giản đồ chỉ rõ tác dụng và chiều chuyển dữ liệu.

8. Có một số liệu là một con số tức thì, hãy dùng các lệnh để chuyển số đó vào các thanh ghi của vi xử lý.

9. Kể các loại của nhóm lệnh nạp địa chỉ. Nếu cho địa chỉ là một con số trực tiếp, hãy nêu các lệnh để thực hiện việc tìm toán hạng có địa chỉ trên trong bộ nhớ.

10. Kể tên các loại lệnh chuyển lưu giữ (Push, pop) giá trị của một thanh ghi trong một vi xử lý và cách thức thay đổi địa chỉ của bộ nhớ ngăn xếp.

11. Kể các loại lệnh số học, ứng dụng của từng lệnh trong xử lý số liệu về toán học.

12. Kể các loại lệnh logic, nêu ứng dụng của từng lệnh.

13. Kể các lệnh xử lý bit, so sánh các cặp lệnh liên quan, ứng dụng của từng loại cặp lệnh.

14. Các lệnh xử lý chuỗi ký tự, so sánh với các lệnh tương ứng trong việc xử lý số liệu.

15. Kể tên và so sánh các lệnh rẽ nhánh không điều kiện và có điều kiện. Những điều kiện nào được dùng để rẽ nhánh có điều kiện.

16. Nêu các trường hợp sử dụng các loại lệnh lặp lại (REPT), quay vòng (LOOP), nhảy (JMP) và gọi chương trình con (CALL).

17. Nêu các trường hợp sử dụng của các lệnh điều khiển vi xử lý đồng bộ hoạt động với bên ngoài.

18. Các lệnh lập/xoá các bit của thanh ghi cờ, nêu các trường hợp sử dụng. Muốn lưu trữ, đọc và kiểm tra giá trị một bit của cờ ta phải làm gì ?

19. Kể các lệnh thuộc nhóm điều khiển, nêu các trường hợp sử dụng.

20. Thống kê các lệnh riêng của các vi xử lý bậc cao hơn 8086/8088.

# BÀI TẬP

*A. Nhóm lệnh chuyển dữ liệu*

1. Viết chương trình nạp số liệu 124Fh vào các thanh ghi nội của vi xử lý 8086. Vào trong môi trường Debug (xem chương 4), dùng lệnh A để biên soạn và lệnh T (hay P) để theo dõi sự dịch chuyển số liệu trên vào các thanh ghi.

2. Viết chương trình đọc các ô nhớ có địa chỉ được quản lý bởi thanh ghi đoạn có giá trị 1234h và offset là 4321h. Cũng dùng lệnh A, T của Debug để biên soạn và chạy chương trình.

3. Viết chương trình chuyển dữ liệu giữa các thanh ghi nội của vi xử lý 8086, dùng lệnh A, T của Debug để biên soạn và chạy chương trình kiểm tra việc thực hiện các lệnh trên.

4. Viết chương trình chuyển lưu giữ dữ liệu của các thanh ghi nội của vi xử lý 8086, dùng lệnh A, T của Debug để biên soạn và chạy chương trình kiểm tra việc thực hiện các lệnh trên.

*B. Nhóm lệnh số học*

5. Viết và cho chạy chương trình (với lệnh A và T của Debug) các phép toán cộng trừ nhân, chia các cặp số liệu sau:

a) 1234h và 05FCh.

b) +15D và -39D.

c) 123456h và - 4321Fh.

d) 123D và - 543D.

Có thể dùng các lệnh MOV AH, 02 ; MOV AL, kết quả và INT 21 để đưa kết quả ra màn hình (chương 5).

6. Với các số liệu tự chọn tùy ý, hãy viết chương trình làm ảnh hưởng tới các bit của thanh ghi cờ. Kiểm tra bằng biên soạn và cho chạy chương trình với lệnh A và T của Debug.

7. Không dùng lệnh trừ SUB A, B, hãy viết và cho chạy chương trình trừ hai số A và B với 4 con số dạng BCD. Ví dụ cho hai số A = 1352D và B = 437D. Có thể dùng các lệnh MOV AH, 02 ; MOV DL, kết quả và INT 21 để đưa kết quả ra màn hình.

8. Viết chương trình tính tổng của:

- Các số nguyên liên tiếp từ n = 1 tới n = 1000D.
- Các số lẻ liên tiếp từ 1 tới 999D.
- Các số chẵn liên tiếp từ 2 tới 1000D.

9. Không dùng lệnh nhân MUL A, B, hãy viết và cho chạy chương trình nhân hai số A và B với hai con số dạng BCD bằng phương pháp cộng liên tiếp. Ví dụ cho hai số A = 13D và B = 7D. Có thể dùng các lệnh MOV AH, 02 ; MOV DL, kết quả và INT 21 để đưa kết quả ra màn hình .

10. Viết và cho chạy chương trình tính N! (giai thừa) với N = 100.

11. Viết chương trình tính tích các số lẻ từ 1 tới 999.

12. Viết chương trình tính tích các số chẵn từ 2 tới 1000.

13. Không dùng lệnh chia DIV A, B, hãy viết và cho chạy chương trình chia hai số A và B với 4 con số dạng BCD bằng phương pháp trừ liên tiếp. Ví dụ cho hai số A = 1232D và B =17D. Có thể dùng các lệnh MOV AH, 02 ; MOV AL, kết quả và INT 21 để đưa kết quả ra màn hình.

14. Viết và cho chạy chương trình tính số nghịch đảo của N!.

15. Viết và cho chạy chương trình đổi một số nguyên (chẵn và thập phân) ra số cơ số 2, cơ số 8 và 16.

16. Viết và cho chạy chương trình tìm ước số chung lớn nhất của hai số nguyên M và N theo giải thuật sau:

- Chia số M cho N. Nếu chia hết, tức số dư R = 0 thì số nhỏ là ước số chung lớn nhất.
- Lấy số N thay số M (số lớn) và số dư R thay số N và lại tiến hành phép chia như trên. Nếu chia hết, thì số dư R là ước số chung lớn nhất.
- Cứ tiếp tục bước trên cho tới khi R = 0 và số thương cuối cùng này là ước số chung lớn nhất.

Như vậy, nếu số thương cuối cùng bằng 1, số dư bằng 0, thì ước số chung lớn nhất của hai số là 1. Chúng ta nói rằng hai số trên nguyên tố cùng nhau.

17. Viết và cho chạy chương trình phân tích một số nguyên thành các số nguyên tố.

18. Viết và cho chạy chương trình tìm ước số chung nhỏ nhất bằng cách phân tích hai số nguyên thành các số nguyên tố và tìm thừa số chung.

19. Có thể viết và cho chạy chương trình tìm bội số chung nhỏ nhất của hai số bằng:

- Giải thuật tương tự giải thuật tìm ước số chung nhỏ nhất trên.

- Phân tích ra thừa số nguyên tố và tìm thừa số chung và riêng.

20. Không dùng các lệnh chỉnh mã ASCII, viết và cho chạy chương trình cộng, trừ hai số dạng mã ASCII. Có thể dùng các lệnh MOV AH, 02 ; MOV AL, kết quả và INT 21 để đưa kết quả ra màn hình.

21. Dùng bảng mã ASCII (xem phụ lục), viết và cho chạy chương trình biến đổi các chữ cái A, B, C v.v... từ chữ hoa thành chữ thường và ngược lại.

*C. Nhóm lệnh logic*

22. Viết và cho chạy chương trình để :

- Xóa một bit di bất kỳ của một byte. Cho ví dụ bằng số.

- Xác lập một bit di bất kỳ của một byte. Cho ví dụ bằng số.

- Đảo (biến 0 → 1 và 1 → 0) một bit di bất kỳ của một byte. Cho ví dụ bằng số.

- Kiểm tra (xem là 0 hay 1) một bit di bất kỳ của một byte. Cho ví dụ bằng số.

23. Viết và cho chạy chương trình để:

- Xóa một số bit bất kỳ của một byte hay từ. Cho ví dụ bằng số.

- Xác lập một số bit bất kỳ của một byte hay từ. Cho ví dụ bằng số.

- Đảo một số bit bất kỳ của một byte hay từ. Cho ví dụ bằng số.

- Kiểm tra một số bit bất kỳ của một byte hay từ. Cho ví dụ bằng số.

24. Viết và cho chạy chương trình để :

- Xóa một thanh ghi. Cho ví dụ bằng số.

- Xác lập cả thanh ghi. Cho ví dụ bằng số.

- Đổi dấu của một số một số thực. Cho ví dụ bằng số.

- Trừ một số với một số có dấu theo nguyên tắc cộng với số đối của nó. Cho ví dụ bằng số.

*D. Nhóm lệnh xử lý bit*

25. Có những phương pháp nào để kiểm tra giá trị của một bit di bất kỳ trong một byte. Viết và cho chạy chương trình cho một ví dụ cụ thể bằng số. Xét trường hợp kiểm tra nhưng vẫn giữ nguyên giá trị cũ của byte.

26. Viết và cho chạy chương trình để :

- Tính xem tổng các bit 1 của một byte là lẻ (kết quả là 1) hay chẵn (kết quả là 0).

- Tính xem trong một byte có bao nhiêu bit 1, bao nhiêu bit 0.

27. Không dùng lệnh nhân MUL A, B, hãy viết và cho chạy chương trình nhân hai số A và B với hai con số dạng BCD bằng phương pháp dịch bit và cộng. Ví dụ cho hai số A = 13D và B = 7D. Có thể dùng các lệnh MOV AH, 02 ; MOV AL, kết quả và INT 21 để đưa kết quả ra màn hình.

28. Không dùng lệnh chia DIV A, B, hãy viết và cho chạy chương trình chia hai số A và B với bốn con số dạng BCD bằng phương pháp dịch bit và trừ. Ví dụ cho hai số A = 1232D

và B =17D. Có thể dùng các lệnh MOV AH, 02 ; MOV AL, kết quả và INT 21 để đưa kết quả ra màn hình.

29. Viết và cho chạy chương trình để :

- Biến đổi một số nhị phân sang BCD dạng không nén và nén và ngược lại.
- Biến đổi một số thập phân sang BCD dạng không nén và nén và ngược lại.
- Biến đổi một số hexadecimal sang BCD dạng không nén và nén và ngược lại.

30. Viết và cho chạy chương trình dùng lệnh TEST (hoặc CMP) để:

- Xác lập cờ ZF nếu nội dung một thanh ghi bằng 0 hay số dương.
- Xóa cờ PF nếu nội dung một thanh ghi là một số lẻ.
- Xác lập cờ PF nếu nội dung một thanh ghi là một số chẵn.
- Xác lập cờ SF nếu nội dung một thanh ghi là một số âm.
- Xóa cờ SF nếu nội dung một thanh ghi là một số dương.

31. Viết và cho chạy chương trình kiểm tra một số đã cho xem :

- Có số bit nhỏ hơn hay bằng 16 ?
- Số âm hay số dương ?
- Số chẵn hay số lẻ ?
- Có bằng 0 hay không ?

32. Dùng lệnh MOV AH, 01h và INT 21 của ngắt INT 21 để nhập ký tự từ bàn phím (chương 5). Viết và cho chạy chương trình nhập các số dạng BCD không nén và nén cho các số có 1, 2, 3, 4, 5, 6 chữ số. Dùng các lệnh MOV AH, 02 và INT 21 để đưa kết quả nhập số ra màn hình và thông báo sự nhập số không phù hợp.

*E. Nhóm lệnh xử lý chuỗi*

33. Viết và cho chạy chương trình kiểm tra việc nhập một chữ số hay ký tự từ bàn phím.

34. Viết và cho chạy chương trình:

- Nhập một chuỗi ký tự " Trường Đại học Bách khoa Hà nội/ 1999" vào các ngăn nhớ có địa chỉ DS:DX = 1200 : 200.
- Dùng các lệnh MOV AH, 02 và INT 21 để đưa kết quả nhập chuỗi ký tự trên ra màn hình.
- Chuyển chuỗi ký tự trên sang vùng nhớ khác có địa chỉ DS : DX = 500 : 0000.
- Đổi các chữ Đại học Bách khoa thành các chữ in.

35. Viết và cho chạy chương trình:

- Nối hai chuỗi ký tự " Trường Đại học" ở vùng nhớ có địa chỉ 1200 : 200 và "Bách khoa Hà nội " ở vùng nhớ 1750 : 1234.

36. Viết và cho chạy chương trình kiểm tra và đếm số ký tự "a" trong dòng chữ ở bài tập 33 trên.

*G. Nhóm xử lý chuỗi và rẽ nhánh chương trình.*

37. Viết và cho chạy chương trình tìm tên "Nguyễn Văn A" trong một danh sách có 120 dòng cho họ và tên và nằm ở địa chỉ bắt đầu là 1200: 0000. Hiển thị kết quả tìm được lên màn hình:

- Không tìm thấy tên đó.
- Ghi địa chỉ bắt đầu của tên tìm thấy.

38. Viết và cho chạy chương trình;

- Sắp xếp một danh sách theo alphabet có địa chỉ đầu là 1200:0200 và địa chỉ cuối là 1200: 0400.

- Kê danh sách các tên có cùng họ Nguyễn, cùng tên Nam.

39. Viết và cho chạy chương trình chèn một tên mới nhập vào danh sách đã sắp xếp theo alphabet.

40. Viết và cho chạy chương trình con INSERT chèn một chuỗi String1 vào chuỗi STRING 2 tại vị trí xác định (tác dụng như ấn phím INSERT bàn phím).

41. Viết và cho chạy chương trình con DELETE để xoá một ký tự hiện trên màn hình và dịch chuyển các ký tự sau lên một vị trí (tác dụng của phím DELETE của bàn phím).

42. Viết và cho chạy chương trình khi gặp tiếp đầu ngữ REP và REPN của một lệnh sẽ thực hiện lệnh nhiều lần.

43. Viết và cho chạy chương trình đưa N số liệu ra thiết bị ngoài mà cổng có địa chỉ 1200:0200.

44. Viết và cho chạy chương trình phát hiện, thống kê và in danh sách ra màn hình tên của những người có:

- Có tuổi từ 55 trở lên.
- Có tuổi dưới 30.
- Có mức lương từ 500.000đ trở lên.

45. Viết và cho chạy chương trình thay thế một *nhãn (LABEL)* của một lệnh bằng địa chỉ của lệnh đó khi gặp một *nhãn* ở trong một lệnh sử dụng nhãn như *CALL nhãn*.

46.Viết và cho chạy chương trình giải một phương trình bậc nhất một ẩn số, in các kết quả và biện luận nghiệm số ra màn hình.

47. Viết và cho chạy chương trình giải một hệ phương trình bậc nhất hai ẩn số, in các kết quả và biện luận nghiệm số ra màn hình.

48. Viết và cho chạy chương trình giải một phương trình bậc hai một ẩn số, in các kết quả và biện luận nghiệm số ra màn hình.

49. Viết và cho chạy chương trình giải một phương trình trùng phương một ẩn số, in các kết quả và biện luận nghiệm số ra màn hình.

50. Viết và cho chạy chương trình giải hệ phương trình sau :

$$y = mx \text{ và } 3x-5y = 0$$

biện luận kết quả theo m và in các kết quả và biện luận nghiệm số ra màn hình.

51. Viết và cho chạy chương trình giải hệ phương trình sau:

$$y = 2m-4 \text{ và } 3y- 5x^2 + 1 = 0$$

biện luận điều kiện và dấu của nghiệm theo giá trị của m, in các kết quả ra màn hình.

*CHƯƠNG 4*

# LẬP TRÌNH VỚI DEBUG

Ở chương 3, chúng ta đã xét các lệnh của vi xử lý 8086. Trong chương này, chúng ta dùng Debug, một chương trình có sẵn trong DOS của máy vi tính PC để viết, nhập, sửa và chạy những chương trình ngắn trên cơ sở các lệnh của Debug.

## 4.1. TỔNG QUAN VỀ DEBUG

### 4.1.1. ĐẶC ĐIỂM CỦA DEBUG

Debug là một chương trình hay một lệnh lớn (xử lý theo lô), lệnh ngoài của DOS (hệ điều hành đĩa), nó dùng để giúp người làm chương trình có thể:

. Nhập và dịch các lệnh dạng hợp ngữ (gợi nhớ) sang mã lệnh của máy và ngược lại.

. Nạp, xem và thay đổi nội dung các tệp tin ở trong khối nhớ.

. Chạy chương trình theo từng lệnh, nhóm lệnh hay cả chương trình.

. Sửa từng byte, nhiều byte lệnh hay cả chương trình.

. Xem và thay đổi nội dung các thanh ghi của vi xử lý như các thanh ghi mảng (CS, DS, ES, SS) thanh ghi thông dụng (AX, BX, CX, DX), DI, SI, IP và thanh ghi cờ.

Debug có tất cả 24 lệnh (DOS version 5.0), đều bắt đầu bằng một (đa số) hay hai chữ cái (alphabet) trong đó có một dấu hỏi (?).

### 4.1.2. KHỞI ĐỘNG VÀ RA KHỎI DEBUG

#### 1. Khởi động

Muốn gọi các lệnh nhỏ của Debug của DOS, ta phải gọi chương trình Debug của hệ điều hành DOS bằng lệnh: **debug** *[[ ổ đĩa:][ đường dẫn] tên tệp[ thông số]]*

Các thông số:

- *[Ổ đĩa:][đường dẫn]tên tệp* xác định vị trí và tên tệp thực hiện được (có đuôi .EXE) để kiểm tra.

- *Thông số* xác định tất cả các thông tin có trên dòng lệnh đòi hỏi bởi chương trình chạy được mà ta muốn kiểm tra.

Giả sử máy vi tính đang chịu sự điều hành của DOS, tức là chương trình Debug. com của MS DOS hiện có ở ổ C, ta khởi động Debug, có nghĩa là vào chương trình Debug. com bằng một trong ba cách sau :

- *Màn hình in C>:* chỉ cần gõ tiếp chữ **Debug** sau dấu nhắc > rồi ấn Enter (↵). DOS điều khiển vào chương trình Debug và hiện lên màn hình dấu nhắc của Debug là dấu ngang nhỏ " - " (hyfen).

- *Nếu hệ điều hành chưa có ở trong ổ cứng*: phải nạp vào từ đĩa ở ổ A. Ta phải:

+ Chuyển sang ổ đĩa A, để màn hình có dấu A>.

+ Gõ Debug và nhấn Enter, Debug từ đĩa A sẽ nhập vào đĩa cứng và máy vi tính đã ở chương trình Debug và hiện lên trên màn hình dấu gạch ngang nhỏ (-).

- *Nếu chương trình đã có ổ đĩa A* với tên là *file name*, ta có thể nhập chương trình và chuyển sang Debug bằng cách gõ:

**A> Debug[file name [arglist]]** thì:

+ DOS đã nạp phần mềm Debug vào bộ nhớ.

+ DOS chuyển điều khiển cho phần mềm Debug với dấu (-) ở trên màn hình.

+ Debug đã nạp tệp tin có tên là *file name* với chuỗi các ký tự dùng để xác định các tham số cần dùng cho tệp tin là *arglist* vào bộ nhớ bắt đầu ở địa chỉ *CS:0100h* và cập thanh ghi BX, CX đã lưu kích thước của tệp tin.

## 2. Ra khỏi Debug trở về DOS

Chỉ cần gõ **Q<Enter>** sau dấu (-) ở trên màn hình. Lệnh Q này của Debug trả lại sự điều khiển cho DOS, trên màn hình, ta có dấu nhắc C> của DOS.

## 4.1.3. TẬP HỢP LỆNH CỦA DEBUG

Debug có 24 lệnh như ở bảng 4.1 (theo thứ tự A, B, C).

***Bảng 4.1.* Các lệnh của Debug**

| Lệnh | Tên tiếng Anh | Chức năng |
|---|---|---|
| ? | Help | Hiển thị bảng trợ giúp về cú pháp các lệnh Debug |
| A | Assemble | Dịch lệnh dạng hợp ngữ sang mã dạng H |
| C | Compare | So sánh hai vùng của bộ nhớ |
| D | Dump | Hiển thị nội dung một vùng nhớ của bộ nhớ |
| E | Enter | Nhập mã máy vào ô nhớ |
| F | Fill | Lấp đầy một vùng nhớ với các giá trị đã cho |
| G | Go | Thực hiện chương trình đã có |
| H | Hexadecimale | Thực hiện các phép toán số học cộng, trừ với các giá trị H |
| I | Input | Đưa vào và hiển thị một byte từ một cổng đã chỉ định |
| L | Load | Nạp một tệp tin hay các cung từ đĩa vào bộ nhớ |
| M | Move | Chuyển một lượng lớn tin sang vùng khác của bộ nhớ |
| N | Name | Đặt tên hay gọi một tệp tin (cho lệnh L và W) và ghi các thông số lên tệp điều chỉnh |
| O | Output | Đưa ra cổng một giá trị một byte |
| P | Proceed | Thực hiện một vòng lặp, một lệnh lặp lại, một ngắt hay một chương trình con (thủ tục) |
| Q | Quit | Rời khỏi Debug |
| R | Register | Hiển thị hay thay đổi giá trị của một hay nhiều thanh ghi |
| S | Search | Tìm trong bộ nhớ một chuỗi byte chỉ định |
| T | Trace | Thực hiện một lệnh và hiển thị nội dung của các thanh ghi, cờ và mã của lệnh tiếp theo |
| U | Unassemble | Dịch ngược lại các byte và hiển thị các lệnh nguồn tương ứng |
| W | Write | Ghi tệp đang hiệu chỉnh lên đĩa |
| XA | | Đặt số trang cho bộ nhớ mở rộng |
| XD | | Làm mất tác dụng của bộ mô tả của bộ nhớ chia trang |
| XM | | Đổi từ trắng logic thành trang vật lý |
| XS | | Hiển thị các tin về tình trạng của bộ nhớ chia trang |

Có thể chia tệp lệnh này thành các nhóm sau:

- Nhóm biên soạn chương trình: **A, E, F, U.**
- Nhóm quan sát tin: **?, R, C, D.**
- Nhóm dịch chuyển tin: **I, O, M, L, W, N.**
- Nhóm thực hiện chương trình: **T, P, G.**
- Nhóm lệnh cộng và trừ hai số hexadecimal: **H.**
- Nhóm liên quan đến bộ nhớ chia trang: **XA, XD, XM, XS.**
- Lệnh thoát khỏi Debug: **Q.**

### 4.1.4. DẠNG LỆNH CỦA DEBUG

Lệnh Debug có một số đặc điểm sau :

**1. Lệnh:** Luôn bắt đầu bằng một hoặc hai chữ cái như bảng 4.1, được gõ từ bàn phím, sau dấu nhắc "-" trên màn hình. Sau khi ấn phím Enter (↵), lệnh được nhập và sau đó in ngay sau dấu nhắc "-" của Debug (chứng tỏ đang trong môi trường Debug).

**2. Tham số**: Có thể không có và có thể có (tùy chọn) là :

***a) Drive:*** là một giá trị hexadecimal 1 chữ số, chỉ ổ đĩa cần truy xuất (0-ổ A, 1-ổ B và 2-ổ C).

***b) Value:*** một giá trị hexadecimal về số liệu tối đa gồm 4 chữ số.

String: một chuỗi ký tự dạng mã ASCII ở trong hai nháy ( " " ).

***c) Address:*** một giá trị về địa chỉ, gồm một hay hai phần, ngăn cách bởi dấu hai chấm " : ".

Phần đầu là tên thanh ghi đoạn, quản lý đoạn nhớ (CS, DS, ES, SS), phần sau chỉ offset trong một đoạn nhớ.

***d) Range:*** giải địa chỉ của vùng nhớ. Có hai dạng :

- **Address 1 address 2** : biểu thị vùng nhớ từ địa chỉ 1 đến vị trí 2.

*Ví dụ:* D CS : 110 120, địa chỉ vùng nhớ từ địa chỉ 110 đến địa chỉ 120, trong đoạn CS. Hai địa chỉ phải cách nhau bằng các khoảng trắng.

- **Address L value** : biểu thị địa chỉ vùng nhớ có địa chỉ bắt đầu là address và có chiều dài do **value** quy định.

*Ví dụ:* D CS : 200 L 20

Vùng nhớ có địa chỉ ban đầu là 200 và có 20 ô nhớ tiếp theo.

***e) Byte:*** là một giá trị hexadecimal gồm hai chữ số dạng hexadecimal (0-F).

***f) List:*** là một danh sách các byte hay các string (chuỗi ký tự).

***g) Filename***: tên một tệp tin, có thể gồm cả ổ đĩa, tên đường dẫn, các tham số được phân cách nhau bằng một hay nhiều khoảng trắng (space) gọi là một Tab.

***h) First sector:*** là một giá trị hexadecimal xác định sector (cung) đầu tiên trên đĩa cần truy xuất.

## 4.2. CÁC LỆNH CỦA DEBUG

### 4.2.1. NHÓM LỆNH BIÊN SOẠN CHƯƠNG TRÌNH

Với 4 lệnh **A, E, F, U** ta có thể ghi và dịch chương trình dạng hợp ngữ sang mã máy (lệnh A), sửa hay thay lệnh số liệu (lệnh E), nạp nhiều lệnh (lệnh F) viết bằng mã lệnh và dịch ngược trở lại (từ mã máy sang dạng hợp ngữ) (lệnh U).

**1. Lệnh A** (assemble - tập hợp): Tập hợp những thuật nhớ 8086/8088/8087 trực tiếp vào bộ nhớ. Lệnh này tạo ra mã máy thực hiện được bắt đầu từ lệnh dạng hợp ngữ. Tất cả các giá trị số ở dưới dạng hexadecimal và ta phải gõ dưới dạng một chuỗi từ 1 tới 4 ký tự. Các tiếp đầu ngữ thuật nhớ trước mã lệnh chỉ để tham khảo.

*Cú pháp:* - **A [address]** ↵

*Thông số :* **Address** ghi vị trí bắt đầu của các lệnh dạng hợp ngữ của chương trình. Dùng một số hexadecimal gồm 4 chữ số (0-F) nhưng không có chữ "h" ở cuối. Do đó, khi viết chương trình, ta để chữ "h" để chỉ dạng số, nhưng khi nhập chương trình vào máy, ta phải bỏ chữ "h" (nếu không, máy sẽ báo lỗi). Địa chỉ **address** có thể là **CS : OFFSET** hay chỉ có **OFFSET** vì ngầm định là cùng **CS** hiện hành. Nếu không có **address** thì lấy địa chỉ của **CS : OFFSET** hiện hành mà lệnh trước dừng tại đó.

*Công dụng :* Lệnh cho phép ta nhập các lệnh của chương trình dạng hợp ngữ vào mỗi địa chỉ ô nhớ. Có thể dùng trong hai trường hợp :

+ *Dùng để ghi lệnh mới của chương trình:* kể từ địa chỉ **address** đầu tiên nào đó. Thường mọi chương trình với địa chỉ đầu tiên là **CS : 0100h** vì phải dành **0100h** byte nhớ cho PSP (Prefix Segment Program- đoạn đứng trước chương trình). Việc biên soạn này có thể tiếp tục chương trình bắt đầu từ địa chỉ **address** của bộ nhớ còn trống mà ta cần tiếp nối chương trình.

+ *Dùng để sửa chương trình:* muốn bỏ một hoặc nhiều lệnh của chương trình cũ, ta chỉ cần ghi đè lệnh mới lên các địa chỉ cũ như ghi lệnh mới.

*Ví dụ :* A 200 ↵ hay A 0200 ↵

Màn hình hiện ra địa chỉ ô nhớ xxxx : 0200 để ta gõ tên lệnh vào địa chỉ trên. Ta gõ lệnh, ví dụ, MOV AX, D2 ↵. Sau khi ấn Enter (↵), dòng lệnh được:

+ Dịch ra mã máy.

+ Ghi vào ô nhớ có địa chỉ trên.

+ Màn hình hiện tiếp địa chỉ nối tiếp sau (máy tính sẽ tính địa chỉ tương ứng) ở dòng tiếp theo của màn hình, để chờ nhập lệnh tiếp theo.

Cứ tiếp tục như vậy, ta có thể nhập được cả chương trình.

Muốn ra khỏi lệnh A để trở về Debug, tức kết thúc chương trình hoặc tạm dừng việc nhập chương trình, chỉ cần ấn phím Enter (↵) sau số liệu của địa chỉ (chưa ghi lệnh), màn hình lại hiện ra dấu " - " của Debug.

*Ghi chú :* - **Dùng các thuật nhớ** : những thuật nhớ để xác định đoạn là **CS:, DS:, ES:, SS:** . Thuật nhớ của sự trở về xa là **reft**. Những thuật nhớ thao tác chuỗi phải ấn định kích thước của chuỗi. Ví dụ, dùng MOVW, để dịch chuyển các từ (16 bit) và **MOVB để dịch chuyển các chuỗi byte** (8 bit).

- **Sự tập hợp rẽ nhánh và gọi :** chương trình hợp dịch tập hợp một cách tự động một rẽ nhánh và gọi loại ngắn, gần hay xa, tùy sự dịch chuyển của các byte về địa chỉ đích. Ta có thể dùng một tiếp đầu ngữ **NEAR** hay **FAR** để đặt cho loại rẽ nhánh hay gọi như minh hoạ dưới đây : - A 0100 : 0500 ↵

```
0100 : 0500   JMP 502          ;  rẽ nhánh ngắn 2 byte
0100 : 0502   JMP NEAR 505     ; rẽ nhánh gần 3 byte
0100 : 0505   JMP FAR 50A      ; rẽ nhánh xa 5 byte.
```

Tiếp đầu ngữ **NEAR** có thể viết tắt thành **NE.**

- **Phân biệt giữa các địa chỉ của lời và byte** : khi một toán hạng quy chiếu tới *một địa chỉ của từ* hay *một địa chỉ của byte*, ta phải xác định chính xác loại các dữ liệu bởi một tiếp đầu ngữ có thể là **WORD PTR** hay **BYTE PTR**. Các viết tắt có thể chấp nhận liên tiếp là **WO** và **BY**. *Ví dụ* : **DEC WO [SI]** và **NEG BYTE PTR [128]**.

- **Chỉ định các toán hạng :** Debug dùng quy ước thường xuyên chỉ một toán hạng giữa các dấu móc làm tham chiếu cho địa chỉ nhớ. Lý do là Debug không phân biệt được giữa toán hạng tức thì và một toán hạng làm tham chiếu cho một địa chỉ nhớ. Ví dụ sau làm minh hoạ hai dạng này :

```
MOV AX, 21      ; nạp giá trị 21 vào AX
MOV AX, [21]    ; nạp nội dung của ô nhớ địa chỉ 21h (DS hiện hành) vào AX.
```

- **Dùng lệnh giả :** có hai lệnh giả **DB** và **DW** đặt cho lệnh **A**, nó tác động một cách liên tiếp lên tập hợp của **BYTE** và **Từ** trực tiếp vào khối nhớ. Ví dụ các lệnh giả sau :

```
DB 1, 2, 3, 4, ' Đây là một ví dụ '
DB " Đây là một móc kép : "
DB ' Đây là một dấu phẩy : '
DW 1000, 2000, 3000, ' batch '.
```

*Ví dụ :* Lệnh **A** nhận biết tất cả các dạng của lệnh thanh ghi gián tiếp. Ví dụ :

```
ADD BX, 34[BP+2]  [SI-1]
POP [ BP+DI]
PUSH [SI].
```

Tất cả các mã hành động đồng nghĩa có thể được chấp nhận, như ví dụ sau :

LOOPZ 100 và LOOPE 100 hay JA 200 và JNBE 200.

Đối với các mã hành động của 8087, các tiếp đầu ngữ WAIT và FWAIT phải được chỉ định như trong ví dụ sau: FWAIT FADD ST, ST (3) ; dòng này tập hợp một tiếp đầu ngữ FWAIT.

**2. Lệnh E** (Enter- đưa vào): Nhập hay sửa mã của ô nhớ.

*Cú pháp:* **E address [list]**

*Các thông số :* - **Address** chỉ địa chỉ ô nhớ cần xem và sửa nội dung. Địa chỉ đây có thể đầy đủ (CS:OFFSET) hay chỉ có OFFSET. Nếu có **list** (danh sách) các số liệu cần ghi vào nhiều ô nhớ liên tiếp thì địa chỉ là địa chỉ bắt đầu của danh sách nhiều byte nhớ.

- **List** chỉ các số liệu mà ta muốn đặt vào các byte nhớ liên tiếp trong bộ nhớ.

*Công dụng:* - Dùng để xem và sửa nội dung một ô nhớ đã ghi.

- Dùng để xem nội dung nhiều ô nhớ liên tiếp đã ghi. Trường hợp này, ta chỉ cần ghi địa chỉ đầu tiên rồi gõ Enter để hiện ra nội dung của ô nhớ đó. Muốn hiện ra tiếp các nội dung của các ô nhớ liên tiếp theo sau, chỉ cần ấn thanh khoảng trắng (Space), nội dung của ô nhớ sau sẽ hiện. Khi cần dừng việc xem nội dung ô nhớ hay thoát khỏi lệnh E, ta chỉ cần ấn Enter.

- Dùng để sửa hay nhập nội dung mới của ô nhớ. Sau mỗi nội dung cũ của ô nhớ hiện ra, ta phải gõ nội dung mới bên cạnh và ấn Enter để nhập vào. Nội dung cũ được thay thế bởi nội dung mới. Nếu ta không muốn thay đổi nội dung của một ô nhớ nào đó, ta chỉ cần ấn Enter, nội dung cũ vẫn được giữ nguyên.

Với lệnh E này, ta có thể sửa một phần của lệnh (đặc biệt là thông số hay toán hạng của lệnh) mà không phải viết đè lên lệnh cũ bằng nội dung lệnh mới với lệnh A đã trình bày ở

trên (tiết kiệm thời gian và công sức). Bằng cách này, ta có thể nhập toàn bô một đoạn lệnh mà không dùng lệnh A, nhưng phải tự dịch lệnh dạng hợp ngữ ra mã máy. Lệnh E này còn dùng để ghi những chuỗi ký tự vào trong các ô nhớ mà không cần phải khai báo phức tạp như sử dụng với chương trình hợp dịch (assembler) sẽ xét ở chương 8.

*Ví dụ: 1. E* ↵ thì nội dung ô nhớ có địa chỉ xxxx : 100 hiện ra, chờ gõ vào giá trị cần thay đổi (nếu đang ở địa chỉ xxxx : 0100h trong đó xxxx là giá trị của thanh ghi đoạn CS).

*2. E 25* ↵ thì nội dung ô nhớ có địa chỉ xxxx: 0025 hiện ra.

*3. E 0F25 : 0100* ↵ màn hình hiện ra nội dung ô nhớ 0F25 : 0100 như sau:

0F25 : 0100 04

trong đó: 04 là nội dung đã ghi ở địa chỉ 0F25 : 0100.

Nếu ta muốn sửa 04 thành 5F, ta gõ 5F liền sau 04 và ấn Enter.

0F25 : 0100 04 5F ↵

*4. E 100 0D 3E "abcde"* ↵ thay đổi 7 byte, bắt đầu từ địa chỉ DS : 100 thành các 0D, 3E, 41, 42, 43, 44, 45 vì các mã ASCII của các chữ A là 41h, chữ B là 42h, chữ C là 43h, chữ D là 44h và chữ E là 45h. Các chữ trong hai dấu " " chỉ là mã ASCII của chúng.

*5. E ES : 100 80 81 82 83* ↵ dùng để ghi nội dung 80, 81, 82, 83 vào các ô nhớ địa chỉ ES : 100, ES : 101, ES : 102, ES : 103.

*Ghi chú:* - Muốn xem hay đổi tiếp ô nhớ sau, bỏ qua ô nhớ trước (để nguyên giá trị cũ đã ghi) ta ấn phím Space (khoảng trống).

- Muốn xem ô nhớ địa chỉ trước (lùi) ta ấn phím (-).
- Muốn kết thúc việc sửa đổi (khối lệnh E, trở về Debug) ta ấn ↵ liền sau địa chỉ.

Lệnh E này cũng tương tự lệnh F (nhập vào lượng lớn tin vào các ô nhớ) về cách nhập mã của tin, còn tương tự lệnh D (DUMP) về măt hiển thị lượng lớn tin trong các ô nhớ.

**3. Lệnh F** (Fill): Lấp đầy các ô nhớ bằng một danh sách mã của dữ liệu (lệnh và số liệu).

*Cú pháp :* **F range list**

Ghi đầy vùng nhớ có *địa chỉ range* bằng các giá trị ghi trong *danh sách list.*

*Thông số:* - **Range,** xác định giải của vùng nhớ, có thể là:

địa chỉ bắt đầu và địa chỉ kết thúc;

địa chỉ bắt đầu và chiều dài của vùng nhớ.

- **List,** chỉ danh sách các dữ liệu (mã lệnh, số liệu) muốn lấp đầy *vùng nhớ* ***range.***

*Công dụng :* Lệnh này có thể thay cho lệnh E để nhập nhiều giá trị (danh sách) cùng một lúc, mà không nhập từng ô nhớ như lệnh E (lâu).

*Ví dụ :* 1. F 100 L 100 42 45 52 54 41 sẽ ghi đầy vùng nhớ có địa chỉ ban đầu DS:100 đến địa chỉ DS : 1FF (100 địa chỉ) bằng các số có giá trị 42, 45, 52, 54, 41. Năm giá trị trên sẽ lặp lại tới khi toàn bộ vùng nhớ được ghi đầy.

2. F 04BA : 100 100 42 45 52 54 41 sẽ ghi đầy 100h byte nhớ có địa chỉ đầu là 04BA : 0100 lần lượt bởi các mã 42, 45, 52, 54, 41.

*Chú ý:* - Nếu một **range** chứa nhiều địa chỉ hơn **list,** lệnh **F** lấp đầy tuần hoàn các giá trị liên tiếp của danh sách cho tới khi các byte của giải **range** được lấp đầy.

- Nếu một trong các địa chỉ nhớ của **range** không có giá trị hoặc không tồn tại, Debug sẽ hiển thị một thông báo lỗi và làm dừng lệnh **F.**

- Nếu một **list** chứa số dữ liệu nhiều hơn số ô nhớ của **range,** các giá trị thừa của **list** không được sử dụng.

**4. Lệnh U** (Unassemble): Dịch ngược từ mã máy sang dạng hợp ngữ và hiển thị các địa chỉ, các giá trị của các ô nhớ và các câu lệnh dạng hợp ngữ. Đó chính là sự xuất hiện của một danh sách của tệp hợp ngữ gồm địa chỉ, mã lệnh và số liệu, lệnh và các toán hạng.

*Cú pháp:* **- U [range]** ↵ để dịch ngược 20h (32D) byte bắt đầu từ địa chỉ đầu tiên của **range**

**- U** ↵ để dịch ngược 20h (32D) byte bắt đầu từ địa chỉ đầu tiên của lần hiển thị trước.

*Thông số :* **Range** chỉ: + địa chỉ bắt đầu và kết thúc hay

+ địa chỉ bắt đầu và chiều dài của mã cần dịch ngược.

*Công dụng:* Lệnh này trái ngược với lệnh A, dịch ngược từ mã máy sang dạng hợp ngữ để thu hồi lại chương trình ban đầu (dạng hợp ngữ) sau khi đã nhập vào máy tính.

*Ví dụ:* 1. U 100 ↵, dịch ngược 32 byte từ địa chỉ ban đầu có CS mặc định và IP = 100.

2. U CS : 100 110 ↵, dịch ngược dữ liệu từ địa chỉ CS: 100 tới CS: 110.

3. U200 L20 ↵, dịch ngược 20 dòng lệnh bắt đầu từ địa chỉ CS: 200.

*Ví dụ*: lệnh U 04BA : 100 110 ↵ sau khi dịch ngược, ta có danh sách các lệnh như sau:

```
04BA : 0100     206472    AND [SI+72], AH
04BA : 0103     69        DB  69
04BA : 0104     7665      JBE 016B
04BA : 0106     207370    AND [BP+DI+70], DH.
```

Lệnh U này giống lệnh A về sự tập hợp các tên lệnh dạng thuật nhớ, còn giống lệnh D về sự hiển thị nội dung của một vùng nhớ.

## 4.2.2. NHÓM LỆNH DỊCH CHUYỂN TIN

Nhóm lệnh này dùng để :

- Đưa tin vào (I) đưa tin ra (O) thiết bị ngoài.
- Nạp tên tin từ đĩa (L) có tên N và ghi tin lên đĩa (W) với lệnh đặt tên (N) trước đó.
- Sao chép hay làm dịch chuyển một vùng nhớ (M) và so sánh (C) hai vùng của bộ nhớ.

**1. Lệnh I** (Input): Đọc và hiển thị giá trị của một byte tin đưa vào từ một cổng được chỉ định.

*Cú pháp:* I cổng ↵

*Thông số:* **Cổng** chỉ *địa chỉ của cổng vào*, có thể mã hoá thành 16 bit.

Công dụng: Đọc giá trị của cổng (1 byte) có địa chỉ là Cổng.

*Ví dụ:* I 2F8 ↵, đọc giá trị 8 bit của cổng có địa chỉ 2F8 và in số liệu trên lên màn hình. Lệnh này trái ngược với lệnh **O** (output) là đưa số liệu từ cổng vào và hiển thị trên màn hình.

**2. Lệnh O** (Output): Đưa tin ra cổng được chỉ định.

*Cú pháp:* **O cổng giá trị** ↵.

*Thông số* : - **cổng** chỉ định địa chỉ của cổng ra.

- **giá trị** chỉ **giá trị** của byte số liệu để đưa ra cổng

*Công dụng:* Đưa **giá trị** (tính ra byte) ra cổng có địa chỉ là **cổng**.

*Ví dụ:* O 61 3A ↵ đưa số liệu 3A ra cổng có địa chỉ **61**.

Lệnh này giống lệnh I ở chỗ trao đổi số liệu với cổng nhưng theo chiều ngược lại là đưa số liệu ra.

**3. Lệnh L** (Load): Nạp một tệp hay nội dung của các cung của đĩa đã chỉ định vào khối nhớ.

*Cú pháp:* - **L [address]** ↵ ghi nội dung của một số byte đã chỉ định trong các thanh ghi BX : CX của một tệp lên đĩa.

- **L Address drive firstsector number** ↵ quay vòng hệ các tệp của MS-DOS và nạp trực tiếp các cung đã được chỉ định.

*Thông số:* - **Address** là **địa chỉ** mà ở đó ta muốn nạp tệp hay nội dung của cung. Nếu ta không chỉ định địa chỉ, Debug dùng địa chỉ chứa trong thanh ghi CS.

- **Drive** chỉ định ổ chứa đĩa trên đó các cung được chỉ định phải được đọc. Có các giá trị 0 = ổ đĩa A, 1 = ổ đĩa B, 2 = ổ đĩa C và cứ như vậy.

Chỉ dùng các thông số **drive, firstsector** và **number** nếu ta muốn nạp nội dung của các cung được chỉ định hơn là tệp được chỉ định trên dòng lệnh của Debug, hay trong lệnh N (**name**) thường gặp nhất.

- **Firstsector** chỉ định (ra số hexadecimal) chỉ số của cung để nạp.

- **Number** chỉ định ((ra số hexadecimal) số các cung liên tiếp để nạp.

*Công dụng:* - ***Dạng 1,*** nạp nội dung tệp tin có tên N xác định vào bộ nhớ của máy vi tính có địa chỉ **address**. Nếu không ghi **address**, tệp tin được nạp vào địa chỉ bắt đầu CS : 100. Cặp thanh ghi BX : CX sẽ chứa kích thước tệp tin.

- ***Dạng 2,*** nạp nội dung các cung (sector) trên đĩa (drive) vào bộ nhớ bắt đầu bởi địa chỉ **address**. Sector đầu tiên trên đĩa cần được xác định trong **firstsector**. **Number** là số **sector** cần đọc.

*Ghi chú:* - Nếu trong địa chỉ address không xác định địa chỉ mảng (chỉ có offset) thì Debuge chỉ lấy địa chỉ mảng được ghi bởi CS.

- Đối với tệp tin thực hiện được (đuôi .COM hay .EXE) ta chỉ có thể nạp chúng vào địa chỉ ban đầu CS : 0100h.

*Ví dụ:* 1. Nạp tệp tin **a.txt** vào địa chỉ ban đầu CS : 0000h

**+ N a.text** ↵

**+ L O** ↵

2. Nạp 5 sector đầu tiên của đĩa A vào bộ nhớ tại địa chỉ DS:0000.

**+ L DS : 0000 0 1 5**

Đối với tất cả các thông tin để chỉ định cho một tệp đối với lệnh **L**, chúng ta sẽ gặp lại ở lệnh **N**.

**4. Lệnh W** (Write): Ghi một tệp hay các sector đã được chỉ định lên đĩa.

Cần chỉ định tên của tệp trên đĩa khi chạy Debug hoặc của lệnh N vừa gặp. Mỗi một trong các phương pháp này cho phép lưu trữ một cách đúng đắn tên của tệp cho một khối kiểm tra tệp (FCB) bắt đầu từ địa chỉ CS : 005C.

*Cú pháp:* - **W [address]** ↵ để ghi vào trong một tệp của đĩa nội dung của số byte đã được chỉ định trong các thanh ghi BX : CX.

- **W address drive firstsector number** để quay vòng hệ các tệp của MS-DOS và ghi trực tiếp các sector đã được chỉ định.

Các thông số của lệnh này cũng giống các thông số của lệnh L.

*Công dụng:* - **Dạng 1,** ghi một vùng nhớ có *địa chỉ bắt đầu* ***address***, có *chiều dài* được cho trong cặp thanh ghi BX : CX lên một tệp tin có **tên** *xác định bởi lệnh N*. Nếu không có thanh ghi địa chỉ, thì tệp tin được ghi bắt đầu từ địa chỉ CS : 0100.

- **Dạng 2,** ghi nội dung các **sector** trong bộ nhớ bắt đầu tại địa chỉ quy định bởi address, lên **đĩa drive**. **Firstsector** xác định ghi lên **sector đầu tiên**. **Number** xác định **số các sector** cần ghi.

*Ghi chú:* - Không ghi tệp tin có đuôi mở rộng .EXE mà phải đổi tên sang .COM hay .BAT.

- Nếu trong địa chỉ không xác định địa chỉ mảng thì Debuge chọn CS.

*Ví dụ:* 1. Có một chương trình định đặt tên INIT. com có 9 địa chỉ nhớ, muốn đặt tên và ghi lên đĩa, ta thực hiện các lệnh sau:

+ N **INIT. com** ↵, đặt tên cho chương trình là INIT. com.
+ R CX ↵
**: 09** ; chỉ nội dung BX.
+ W ↵ ; ghi lên đĩa.

2. Ghi 10 sector trong bộ nhớ bắt đầu tại địa chỉ CS : 0000 vào sector thứ 200 của đĩa A

**W 0000 1000 200 10**

BX CX sector thứ 200 số sector

**5. Lệnh N** (Name): đặt tên cho tệp sắp ghi, ghi trước lệnh **W** hay **L** như ở ví dụ trên.

*Cú pháp :* - **N[đĩa:][đường dẫn]tệp**

- **N param** để chỉ định các thông số của tệp thực hiện được mà ta sửa.
- **N** để xoá những chỉ định hiện hành

*Thông số:* - *[ Đĩa: ][ đường dẫn]tệp* chỉ định vị trí và tên tệp thực hiện được mà ta cần sửa.

- *Param* chỉ định các thông số và cầu nối cho mạch đối với tệp thực hiện được để sửa.

*Ghi chú :* - *Hai chức năng của lệnh N.* Ta có thể dùng lệnh N theo hai cách:

+ Chỉ định một tệp mà có thể dùng trước kia trong một lệnh **L** hoặc **W.** Nếu ta gọi lệnh Debug không chỉ rõ tệp để sửa, ta phải gõ lệnh N tệp trước khi có thể dùng lệnh **L** để nạp tệp. Tên của tệp được tạo dạng một cách đúng đắn cho một khối kiểm soát tệp (FCB) ở địa chỉ CS : 005C.

+ Dùng lệnh N để chỉ định các thông số và các cầu nối (commutator) trên dòng lệnh cho tệp đã được sửa.

- *Các vùng nhớ :* Có bốn vùng nhớ có thể ảnh hưởng tới việc sử dụng lệnh N.

| Địa chỉ | Nội dung | Địa chỉ | Nội dung |
|---|---|---|---|
| CS : 005C | FCB cho tệp 1 | CS : 0080 | Chiều dài của dòng lệnh N |
| CS : 006C | FCB cho tệp 2 | CS : 0081 | Bắt đầu của dòng lệnh N |

*Ví dụ:* N \ SK \ SK. com ↵

*Công dụng:* Gọi hoặc đặt tên tệp cho tệp trước khi nạp (L) hoặc ghi (W).

**6. Lệnh M** (Move): Dịch chuyển một vùng nhớ tới một vùng nhớ khác.

*Cú pháp:* **M range address** ↵.

*Thông số:* - **Range** chỉ định các địa chỉ bắt đầu và kết thúc hay địa chỉ bắt đầu và chiều dài của một vùng nhớ mà ta muốn chép nội dung.

- **Address** chỉ định địa chỉ bắt đầu của chỗ mà ta muốn chép nội dung của **range** (vùng nhớ).

*Công dụng:* Chép (chuyển) nội dung vùng nhớ có *địa chỉ* **range** (địa chỉ đầu, địa chỉ cuối) sang vùng nhớ có *địa chỉ đầu* là **address**.

*Ví dụ:* 1. **M 100 2FE 3000 : 0000** ↵ ; chuyển vùng nhớ có địa chỉ đầu DS: 100 đến DS : 02FE sang vùng nhớ có địa chỉ đầu là 3000 : 0000.

2. **M 100 L100 3000 : 0000** ↵ ; chuyển vùng nhớ có địa chỉ đầu DS : 0100, kích thước 100 sang vùng nhớ có địa chỉ đầu là 3000: 0000.

### 4.2.3. NHÓM LỆNH QUAN SÁT TIN

Nhóm lệnh này dùng để xem danh sách và chỉ dẫn tập lệnh của Debug (lệnh ?), đọc tin (lệnh R), tìm tin (lệnh S) và quan sát cả một vùng nhớ (D).

**1. Lệnh ?** (Help): Dùng để quan sát cả tệp lệnh Debug và chỉ dẫn.

*Cú pháp:* **-?** ↵.

**2. Lệnh R** (Register): Hiển thị hay thay nội dung của một hay nhiều thanh ghi của vi xử lý.

*Cú pháp:* - **R** ↵ để hiển thị nội dung của mọi thanh ghi, kể cả cờ.

- **R [ register ]** ↵ để hiển thị nội dung của thanh ghi có tên là **register**.

*Công dụng :*

- *Dạng 1:* đọc nội dung của các thanh ghi nội của vi xử lý. Trên màn hình sẽ in ra nội dung các thanh ghi (AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP) và các bit cờ FL (bởi các bit P, T,O, S, Z, C) và lệnh cuối cùng của chương trình sắp thực hiện.

Tên các bit của thanh ghi cờ có ý nghĩa như bảng sau.

- *Dạng 2:* xem và sửa nội dung chỉ một thanh ghi. Sau lệnh này màn hình in ra nội dung thanh ghi và dấu hai chấm (:) để ta gõ vào nội dung cần sửa, rồi gõ ↵. Nếu không sửa, gõ ↵ sau khi hiển thị, nội dung thanh ghi để nguyên.

Riêng thanh ghi cờ, ta có các bit với giá trị khác nhau sau khi thực hiện các lệnh lập và xóa cờ như bảng 4.2.

***Bảng 4.2.* Các cờ khi có lệnh xác lập và xóa**

| Tên cờ | Xác lập | Xóa |
|---|---|---|
| Cờ tràn OF | OV | NV |
| Cờ hướng DF | DN (xuống) | UP (lên) |
| Cờ ngắt IF | EI (cho phép ngắt) | DI (cấm ngắt) |
| Cờ dấu SF | NG (âm) | PL (dương) |
| Cờ không ZF | ZR (zero = 0) | NZ (not zero) |
| Cờ carry phụ AF | AC (carry phụ) | NA (không có carry phụ) |
| Cờ chẵn lẻ PF | PE (chẵn) | PO (lẻ) |
| Cờ nhớ CF | CY (có nhớ) | NC (không nhớ) |

*Ví dụ:* 1. R ↵ : Nội dung các thanh ghi in ra màn hình như sau:

AX = 0000 BX = 0000 CX = 0000 DX = 0000 SP = FFEE BP = 0000

SI = 0000 DI = 0000 DS = 28D2 ES = 28D2 SS = 28D2 CS = 28D2

IP = 100 NV UP EI PL NZ PO NC

28D2: 0100 B80 0300 MOV AX, 0003.

2. R CX ↵ : Xem và sửa thanh ghi CX.

Màn hình in ra : CX 0000

:

: ABCD ↵

gõ mới vào để sửa.

Màn hình hiện ra : CX ABCD

:

: ↵

gõ mới.

Lại trở về Debug với dấu (-) trên màn hình.

**3. Lệnh D** (Dump): Hiển thị nội dung của một vùng nhớ lên màn hình.

*Cú pháp:* - **D [address] ↵ :** in nội dung bộ nhớ gồm 128 byte bắt đầu tại địa chỉ (address).

- **D [range] ↵ :** in vùng nhớ trong dải [range] ra màn hình.

*Thông số :* - **Address** chỉ địa chỉ ô nhớ đầu tiên của vùng nhớ chứa 128 byte.

- **Range** chỉ dải nhớ kể từ địa chỉ đầu tiên tới địa chỉ kết thúc.

Nếu không có thông số, màn hình sẽ hiển thị 128 byte nhớ kể từ địa chỉ kết thúc được chỉ định bởi địa chỉ kết thúc của lệnh D trước đó.

*Ghi chú:* Khi dùng lệnh **D**, Debug hiển thị nội dung của dải nhớ thành hai phần, một phần dưới dạng hexadecimal và một phần dưới dạng văn bản ASCII. Mỗi ký tự không in được, được hiển thị dưới dạng một dấu chấm (.). Mỗi dòng cho phép hiển thị nội dung 16 byte với một dấu gạch ngang nhỏ giữa byte thứ 8 và thứ 9. Mỗi dòng được hiển thị bởi một byte của địa chỉ nhân với 16.

*Công dụng:* Để quan sát nội dung của cả một mảng nhớ.

*Ví dụ:* 1. D100 ↵ : màn hình in nội dung bộ nhớ gồm 128 byte bắt đầu tại địa chỉ DS:100.

2. D CS : 200 220 ↵ : xem nội dung bộ nhớ từ địa chỉ CS:120 đến địa chỉ CS:220.

3. D 0A00 : 100 L10 ↵ : xem nội dung16 byte nhớ ($10_{16}$) bắt đầu tại địa chỉ 0A00 : 0100.

4. D ↵ : xem tiếp 128 byte nhớ tiếp theo địa chỉ nhớ vừa xem.

**4. Lệnh C** (Compare): So sánh vùng nhớ.

*Cú pháp:* **C range address.**

*Thông số :* - **Range** chỉ định địa chỉ bắt đầu và kết thúc hay địa chỉ ban đầu và độ dài của vùng nhớ thứ nhất để so sánh. Đối với tất cả tin về các độ lớn của **range,** hãy xem lại lệnh Debug.

- **Address** chỉ định địa chỉ bắt đầu của vùng nhớ thứ hai của khối nhớ để so sánh. Đối với tất cả tin về các độ lớn của **range**, hãy xem lại lệnh Debug.

*Chú ý:* Nếu hai phần của bộ nhớ đã được xác định bởi **range** và **address** là đồng nhất, Debug không hiển thị gì cả và ta gửi trực tiếp yêu cầu của Debug. Nếu chúng khác nhau Debug hiển thị chúng theo dạng sau :

address 1 octet 1 octe↑2 address 2.

Xem ví dụ dưới đây cho việc minh họa của dạng này.

*Công dụng :* So sánh vùng nhớ, vùng nhớ một có dải địa chỉ range với vùng nhớ hai có địa chỉ đầu là address.

*Kết quả so sánh :*

- Không in gì lên màn hình bởi Debug, thì hai vùng nhớ giống nhau.
- In ra màn hình địa chỉ của các ô nhớ khác nhau.

Lệnh này luôn theo sau lệnh M để kiểm tra việc dịch chuyển một dải nhớ.

*Ví dụ :* 1. C 100 1EF 4000 : 0000 ↵

2. C 100 L100 4000 : 0000 ↵

So sánh vùng nhớ một có địa chỉ DS : 0100 tới DS : 01FF với vùng nhớ có hai địa chỉ đầu là 4000 : 0000.

3. Hai lệnh sau có cùng một tác dụng:

E 100 10F 300
E 100 110 300

Mỗi một lệnh so sánh khối nhớ có địa chỉ 100h đến 10Fh với khối nhớ có từ 300h đến 30Fh.

Debug đáp ứng một trong hai lệnh trên với một danh sách tương tự với danh sách sau (giả sử rằng DS = 197F) :

| | | |
|---|---|---|
| 197F : 0100 | 23 45 | 197F : 0100 |
| 197F : 0101 | 43 21 | 197F : 0101 |
| 197F : 0102 | 45 67 | 197F : 0102 |
| 197F : 0103 | 3D 5F | 197F : 0103 |
| 197F : 0104 | 12 65 | 197F : 0104 |
| 197F : 0105 | 45 6C | 197F : 0105 |
| 197F : 0106 | 76 34 | 197F : 0106 |
| 197F : 0107 | 4D C7 | 197F : 0107 |
| 197F : 0108 | 35 6C | 197F : 0108 |
| 197F : 0109 | 34 12 | 197F : 0109 |
| 197F : 010A | 56 78 | 197F : 010A |
| 197F : 010B | 56 12 | 197F : 010B |
| 197F : 010C | 34 56 | 197F : 010C |
| 197F : 010D | 34 67 | 197F : 010D |
| 197F : 010E | 54 67 | 197F : 010E |
| 197F : 010F | 78 9C | 197F : 010F |

**5. Lệnh S** (Search): Tìm một danh sách các byte trong vùng nhớ chỉ định.

*Cú pháp:* **S range list** ↵

*Thông số:* - **Range** chỉ các địa chỉ bắt đầu và kết thúc của vùng nhớ cần tìm.

- **Liste** chỉ một hay nhiều byte hoặc chuỗi để tìm. Tách biệt những byte bởi các khoảng trống hay các dấu phẩy. Đặt giá trị của chuỗi trong các ngoặc kép.

*Ghi chú:* - Nếu thông số **list** chứa hơn một giá trị của một byte, Debug chỉ hiển thị địa chỉ đầu tiên nơi giá trị của byte có mặt. Nếu list chỉ có một byte, Debug hiển thị tất cả các địa chỉ của *range* đã được chỉ định ở nơi có

- Nếu trong **range** không quy định địa chỉ mảng thì Debug sẽ lấy mảng được chỉ bởi DS (hay CS).

*Công dụng:* Tìm **danh sách** (list) trong vùng nhớ **range**.

Nếu : - Tìm thấy có mã của **list**, màn hình sẽ in ra địa chỉ đầu tiên của tất cả các ô nhớ, *trùng với danh sách list.*

- Không tìm thấy có mã trùng nhau, màn hình không in ra gì cả.

*Ví dụ:* S CS: 100 300 42 ↵. Tìm giá trị 42 trong vùng nhớ từ địa chỉ CS:100 đến CS: 300. Nếu tìm thấy, Debug in ra màn hình địa chỉ tất cả các ô nhớ chứa giá trị 41.

## 4.2.4. LỆNH CỘNG, TRỪ SỐ HEXADECIMAL

*Cú pháp:* **H value1 value2**

*Thông số:* - **value1** là một số hexadecimal thứ nhất, bao gồm giữa 0 và FFFFh.

- **value2** là một số hexadecimal thứ hai, bao gồm giữa 0 và FFFFh.

*Công dụng*: Tính và in kết quả của hai phép tính cộng và trừ lên màn hình. Các số hạng là value1 (số cộng, số trừ). Kết quả trước là tổng số, kết quả sau là hiệu số của **value1** và **value2**.

*Ví dụ:* **H 19F 10A** ↵ . Trên màn hình in 02A9 (tổng) 0095 (hiệu).

## 4.2.5. NHÓM LỆNH THỰC HIỆN CHƯƠNG TRÌNH

Nhóm lệnh này điều khiển thực hiện chương trình từng lệnh hay từng bước (lệnh **T**), thực hiện một lệnh vòng lặp, gọi chương trình con hay gọi ngắt (lệnh **P**) và cả chương trình (lệnh **G**).

**1. Lệnh T** : (trace - theo dõi dấu vết), thực hiện một lệnh của chương trình và hiển thị nội dung của tất cả các thanh ghi, trạng thái của tất cả các cờ và dạng mã của lệnh sắp thực hiện.

*Cú pháp :* **T [ = address] [value]** ↵

*Thông số:* - **= address** chỉ định *địa chỉ mà Debug phải bắt đầu những lệnh cần theo dõi.*

Nếu ta quên thông số **address**, việc theo dõi bắt đầu từ địa chỉ đã được chỉ định bởi các thanh ghi CS : IP.

- **Value** chỉ *số lệnh cần theo dõi.* Số này phải là một số hexadecimal. Giá trị ngầm định là 1.

*Công dụng:* Thực hiện từng lệnh một, bắt đầu từ địa chỉ [**address**]. Sau mỗi lệnh nội dung các thanh ghi hiện lên màn hình, **value** xác định số liệu cần thực hiện nên gọi là lệnh duyệt. Nếu không có địa chỉ và value, thì chỉ thực hiện một lệnh ở địa chỉ xác định bởi nội dung thanh ghi CS : IP.

*Ghi chú: - Các lệnh theo dõi nhớ ROM* : lệnh T dùng chế độ theo dõi phần cứng của vi xử lý 8086/8088. Như vậy, nó cũng theo dõi những lệnh trong nhớ ROM.

- *Dùng các thông số địa chỉ* : chúng ta phải đặt trước **address** dấu = để phân biệt nó với thông số **value.**

*Ví dụ:* 1. T ↵ : thực hiện lệnh tại địa chỉ có giá trị ở cặp thanh ghi CS : IP.

Nếu vị trí lệnh trong chương trình là 04BA : 011A và lệnh ở ô nhớ tiếp theo là INT 21, sau khi thực hiện lệnh T↵, màn hình hiển thị :

AX = 0E00 BX = 00FF CX = 0007 DX = 01FF SP = 039D BP = 0000 SI = 005C DI = 0000
DS = 04BA ES = 04BA SS = 04BA CS = 04BA IP = 011A NV UP DI NG NZ AC PE NC
04BA : 011A CD21 INT 21.

2. T = 100 ↵ : thực hiện lệnh tại địa chỉ có giá trị tại cặp CS : 0100.

3. T = 100 5 ↵ : thực hiện duyệt lệnh bắt đầu địa chỉ CS : 0100. Cả năm lệnh được thực hiện và in nội dung các thanh ghi lên màn hình.

*Các lệnh liên quan:* - Đối với tất cả các tin tức về việc thực hiện một vòng lặp, một lệnh chuỗi lặp lại, một ngắt logic hay một chương trình con, hãy xem lệnh **P** (Proceed).

- Đối với tất cả tin tức về thực hiện chương trình có mặt trong khối nhớ, hãy xem lệnh **G** (Go).

**2. Lệnh P**: (Proceed - xử lý, tiến hành), thực hiện một vòng lặp, một lệnh chuỗi được lặp lại, một ngắt logic hay một chương trình con hoặc theo dõi một lệnh.

Lệnh thực hiện từng lệnh như lệnh T nhưng có khác là khi gặp các lệnh gọi chương trình con, lệnh lặp hay lệnh ngắt, nó thực hiện cả nhóm lệnh trên.

*Cú pháp:* **P [= address] [value]** ↵

*Các thông số:* - = **address** chỉ định địa chỉ của lệnh đầu tiên cần thi hành. Nếu không ghi địa chỉ này, giá trị ngầm định là địa chỉ hiện hành đã được chỉ định trong những thanh ghi CS:IP.

- **Value** chỉ định số lệnh cần thực hiện trước khi trả sự điều khiển cho Debug theo ngầm định là 1.

*Ghi chú: - Chuyển sự điều khiển cho chương trình đang vận hành.* Khi lệnh **P** chuyển sự điều khiển của Debug cho chương trình đang vận hành, chương trình này thực hiện không bị ngắt tới khi mà vòng lặp, lệnh lặp lại chuỗi, ngắt logic hay một chương trình con ở địa chỉ đã chỉ định phải được kết thúc hoặc tới khi số lệnh đã được chỉ định của mã lệnh đã được thực hiện. Sự điều khiển trả về cho Debug.

*- Các giới hạn của thông số địa chỉ.* Nếu thông số **address** không chỉ định thanh ghi đoạn, Debug dùng thanh ghi CS của chương trình hiện đang vận hành. Nếu ta quên thông số, việc thực hiện chương trình bắt đầu từ địa chỉ đã được chỉ định bởi các thanh ghi CS : IP. Khi đó, ta phải đặt trước thông số **address** một dấu = để phân biệt với thông số **value.** Nếu lệnh có mặt ở địa chỉ đã được chỉ định không phải là một vòng lặp, một lệnh chuỗi lặp lại, một ngắt logic (mềm) hoặc một chương trình con, lệnh **P** hoạt động giống như lệnh **T.**

*- Các thông báo được hiển thị bởi lệnh* **P.** Một lần, một lệnh đã được thực hiện bởi **P,** Debug hiển thị nội dung các thanh ghi của chương trình, trạng thái của các cờ của chúng và dạng mã hóa của lệnh sẽ được thực hiện tiếp theo.

*Chú ý :* Ta không thể dùng lệnh **P** để theo dõi sự thực hiện lệnh chứa trong nhớ ROM.

*Ví dụ:* 1. Giả sử rằng chương trình đang chạy chứa một lệnh **CALL** ở địa chỉ CS : 143F. Để thực hiện chương trình con là đích của lệnh **CALL** và sau đó lại quay trở về sự điều khiển của Debug, ta gõ lệnh :

**P 143F** ↵. Debug hiển thị một danh sách như sau :

AX = 0000 BX = 0000 CX = 0000 DX = 0000 SP = FE00 BP = 0000 SI = 0000 DI = 0000
DS = 2246 ES = 2246 SS = 2246 CS = 2246 IP = 1443 NV UP EI PL NZ AC PO NC
2246 : 1442 7505 JNZ 144A.

2. P = 100 4 ↵, thực hiện 4 lệnh chương trình bắt đầu tại địa chỉ CS : 0100.

**3. Lệnh G** (Go) : thực hiện hay chạy cả chương trình nạp trong bộ nhớ.

*Cú pháp:* **G [= address ] [ điểm dừng]**

*Thông số:* - = **address** chỉ định địa chỉ mà ta muốn bắt đầu thực hiện lệnh **G.** Nếu ta không chỉ định địa chỉ **address,** Debug bắt đầu thực hiện chương trình ở địa chỉ hiện tại trong cặp thanh ghi CS : IP. Có thể dùng lệnh **G** để chạy từ địa chỉ bắt đầu chương trình (CS : 0100) nếu trước đó, ta dùng lệnh R IP ↵ để sửa IP = 0100.

- **Điểm dừng** chỉ định từ 1 tới 10 điểm dừng tạm thời cho lệnh **G.**

*Ghi chú:* - **Dùng địa chỉ tùy chọn.** Ta phải đặt trước thông số **address** một dấu bằng (=) để phân biệt địa chỉ khởi đầu (address) với điểm dừng.

- **Chỉ thị các điểm dừng.** Chương trình làm dừng việc thực hiện ở điểm dừng đầu tiên gặp được một cách độc lập vào vị trí của nó trong danh sách các điểm dừng. Debug thay lệnh gốc của mỗi điểm dừng bởi một mã ngắt. Khi chương trình đạt tới một điểm dừng, Debug khôi phục tất cả các địa chỉ của điểm dừng về lệnh gốc, rồi hiển thị nội dung của tất cả các thanh ghi, trạng thái của các cờ và dạng mã của lệnh vừa thực hiện. Debug cũng hiển thị các thông tin mà nó hiển thị nếu ta gõ lệnh R (Register) và ta chỉ định điểm dừng. Nếu ta không dừng chương trình ở điểm dừng, các mãcủa ngắt không được thay thế bởi lệnh gốc.

- **Các giới hạn cho việc xác định các điểm dừng.** Ta có thể đặt những điểm dưng giống như các địa chỉ chứa trong byte đầu tiên của một mã lệnh của 8086. Nếu ta định nghĩa hơn 10 điểm dừng, Debug hiển thị sai số như sau: ERROR BP.

- **Các điều kiện cho con trỏ ngăn xếp của người sử dụng.** Con trỏ ngăn xếp của người sử dụng phải có nghĩa (valid) và chứa ít nhất 6 byte cho lệnh **G.** Lệnh này dùng một lệnh **IRET** để gây ra một rẽ nhánh trong chương trình hiện hành đang chạy. Debug chỉ con trỏ ngăn xếp của người sử dụng và hàng các cờ, thanh ghi đoạn của mã và con trỏ lệnh (IP) trong ngăn xếp. (Một địa chỉ của ngăn xếp không hợp lệ hay quá nhỏ có thể gây ra một điểm dừng của hệ điều hành). Debug đặt mã ngắt (0CCh) vào (các) địa chỉ của điểm dừng đã chỉ ra.

- **Lại chạy chương trình.** Không nên thử chạy lại một chương trình khi MS-DOS có thông báo sau : **Program terminated normally.**

*Ví dụ:* 1. **G = 100** ↵ hay **G** ↵ thực hiện chương trình bắt đầu tại địa chỉ CS:100.

2. **G = 100 200** ↵, thực hiện chương trình từ địa chỉ CS : 100 đến địa chỉ CS : 200.

## 4.2.6. NHÓM LỆNH LIÊN QUAN TỚI BỘ NHỚ CHIA TRANG

Những lệnh này bắt đầu xuất hiện ở DOS với những phiên bản gần đây từ version 4. 0 cho vi xử lý từ 386, khi bắt đầu có quản lý bộ nhớ chia trang và các lệnh này giúp ta thâm nhập sự quản lý đó. Đó là các lệnh đặt bộ nhớ mở rộng (lệnh XA), làm vô hiệu việc đặt một bảng mô tả bộ nhớ chia trang (lệnh XD), biến đổi một trang nhớ logic thành một trang nhớ vật lý (lệnh XM) và hiển thị trạng thái bộ nhớ mở rộng (lệnh XS).

**1. Lệnh XA** : (Allocate Expanded Memory), đặt số trang cho bộ nhớ mở rộng.

*Cú pháp :* **XA [number]** ↵

*Thông số :* **Number** là số trang 16 KB đặt cho bộ nhớ chia trang.

Nếu số trang đặt thích hợp, Debug hiển thị một thông báo để chỉ số hexadecimal của bảng mô tả tạo ra. Nếu không, Debug hiển thị một thông báo lỗi.

*Ví dụ :* Để đặt 8 trang của bộ nhớ chia trang, ta gõ lệnh XA 8 ↵.

Nếu lệnh được thực hiện thành công, Debug hiển thị một thông báo tương tự như sau: Descriptor gây ra 0008.

**2. Lệnh XD** (Deallocate Expanded Memory): Hủy bỏ nhớ mở rộng.

*Cú pháp:* **XD [Descriptor]** ↵

*Thông số :* **Descriptor** là bảng mô tả mà chúng ta muốn hủy bỏ.

*Ví dụ :* XD003 ↵ nếu lệnh thực hiện thành công, Debug hiển thị một thông báo tương tự sau :

Descriptor 003 bị hủy bỏ.

**3. Lệnh XM** (Map Expanded Memory Pages): Các trang của bộ nhớ mở rộng của bản đồ.

Tác động lên một trang logic của bộ nhớ chia trang thuộc một bộ mô tả đã chỉ định thành một trang vật lý của bộ nhớ.

*Cú pháp :* **XM [Lpage] [Ppage] [Descriptor]** ↵

*Các thông số:* + **Lpage,** là số trang logic của bộ nhớ chia trang mà ta muốn đặt thành trang vật lý **ppage**.

+ **Ppage,** là số trang vật lý mà **Lpage** phải tác động.

+ **Descriptor,** bảng mô tả.

*Ví dụ :* Muốn đổi trang logic 5 của bảng mô tả 0003 mà bộ nhớ chia trang thành trang vật lý 2, gõ lệnh sau : XM 5 2 0003 ↵

Nếu hành động thành công, Debug hiển thị như sau:

*Page logic 05*, tác động lên trang vật lý 02.

**4. Lệnh XS** (Display Expanded Memory Status): Hiển thị của những tin về trạng thái bộ nhớ chia trang.

*Cú pháp :* **XS** ↵

Lệnh này không cần một thông số nào.

Debug hiển thị những thông tin trong dạng sau:

*+ 0000 trang bị đặt lên bộ mô tả 0000.*

*+ 0002 trang bị đặt lên bộ mô tả 0001.*

*+ Trang vật lý 00 = mảng (segment) của khung C000.*

*+ Trang vật lý 01 = mảng (segment) của khung C400.*

*+ Trang vật lý 02 = mảng (segment) của khung C800.*

*+ Trang vật lý 03 = mảng (segment) của khung CC00.*

*+ 2 trang EMS bị đặt lên một tổng của 80.*

*+ 2 bộ mô tả EMS bị đặt lên một tổng của FF.*

## 4.3. LẬP TRÌNH HỢP NGỮ CHẠY VỚI DEBUG

Chương trình dạng hợp ngữ chạy với Debug là chương trình ngắn gọn, ít lệnh, không có các khai báo và ký hiệu phức tạp, thường chỉ có nhãn (label) để đặc trưng cho địa chỉ của lệnh và các bộ đệm số liệu.

Các chương trình này gồm ít lệnh (≤ 64 KB), được Debug dịch sang mã máy và ghi vào bộ nhớ.

Ở mục này, chúng ta sẽ xét thủ tục viết một chương trình, cấu trúc một chương trình thực hiện trên Debug và giới thiệu một số dạng chương trình cơ bản về cấu trúc.

## 4.3.1. THỦ TỤC VIẾT MỘT CHƯƠNG TRÌNH

Viết một chương trình tức viết tuần tự các lệnh để thực hiện một nhiệm vụ nào đó như:

- Phục vụ cho các thiết bị ngoài thông dụng (bàn phím, màn hình, chuột, máy in, đĩa từ, cassette) để máy vi tính điều khiển chúng hoạt động trong các công việc đưa số liệu vào, lưu trữ, xử lý và đưa số liệu ra để hiển thị và lưu trữ ở các bộ nhớ ngoài.

- Xử lý số liệu, như biến đổi dạng mã (ASCII, BCD, nhị phân, octal, hexadecimal v.v...), xử lý số học (cộng, trừ, nhân, chia) và xử lý logic (và, hoặc, đảo, loại trừ...).

- Phục vụ các hệ On-line (trực tuyến) để đo các thông số và điều khiển các cơ cấu chấp hành trong các hệ đo, điều khiển kỹ thuật.

- Các phục vụ khác như xử lý ảnh, xử lý âm thanh v.v...

Muốn vậy, ta phải theo các thủ tục hay trình tự các bước sau:

+ Xác định nhiệm vụ, trong đó cần chi tiết hoá các nhiệm vụ con, trình tự (thứ tự theo thời gian) thực hiện và mối quan hệ giữa chúng trong điều kiện ràng buộc nào đó.

+ Vẽ giải thuật hay luận lý (Algorithm) của chương trình lớn.

+ Viết các lệnh cho việc thực hiện các bước giải thuật, trong đó có thể có các chương trình con.

+ Lắp ghép thành chương trình lớn.

### 1. Xác định các nhiệm vụ con và vẽ giải thuật

***a) Xác định các nhiệm vụ con***

Nhiệm vụ con là các nhiệm vụ nhỏ của nhiệm vụ lớn phải thực thi bằng chương trình.

*Ví dụ:* Muốn cộng hai số A và B và đưa biểu diễn tổng lên màn hình ta có các nhiệm vụ sau:

+ Đưa số liệu A vào thanh ghi chứa AX.

+ Đưa số liệu B vào thanh ghi đệm, ví dụ BX.

+ Thực hiện phép cộng.

+ Đưa kết quả lên màn hình.

Nếu các nhiệm vụ con này phải lặp lại nhiều lần ta có thể dùng kỹ thuật:

+ Quay vòng (loop) để thực hiện lặp lại nhiều lần.

+ Gọi chương trình con nhiều lần tức gọi nhiệm vụ lặp lại trên.

***b) Vẽ giải thuật***

Vẽ giải thuật, tức là:

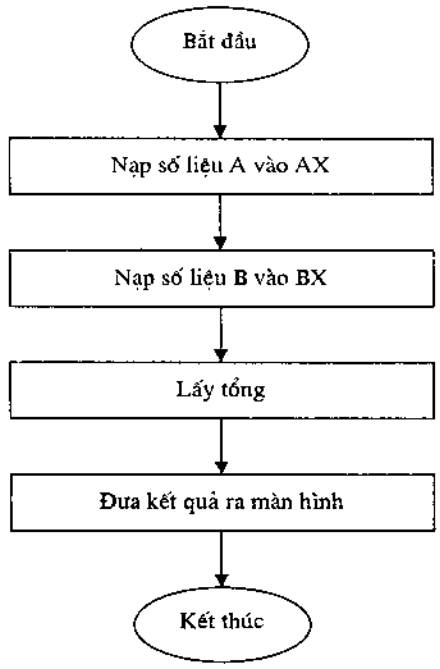
- Vẽ các khối của các nhiệm vụ con.

- Xác định trình tự thực hiện các nhiệm vụ con.

- Xác định trật tự thực hiện các nhiệm vụ con.

- Xác định các điều kiện thực hiện các nhiệm vụ con, tức là điều kiện và mối liên hệ giữa các nhiệm vụ con. Nói khác đi, nhiệm vụ con này thực hiện lúc nào, với điều kiện gì và điều kiện này do kết quả thực hiện của nhiệm vụ nào đã thực hiện trước.

Sau đây là giải thuật thực hiện nhiệm vụ lấy tổng của hai số A và B (hình 4.1).




**Hình 4.1.** Giải thuật của phép cộng hai số A và B.

Giải thuật trên có các đặc điểm sau:

- Có bắt đầu và kết thúc.

- Các nhiệm vụ con chỉ thực hiện theo trình tự thời gian chứ không có điều kiện ràng buộc nào về kết quả số liệu hoặc điều kiện khác ngoài điều kiện nạp xong hai số liệu mới tính tổng.

- Có thể thay đổi trình tự thực hiện hai lệnh nhập số liệu A và B mà kết quả không thay đổi.

Nếu là phép trừ, số A là số bị trừ phải được nhập trước vào A và khi dùng lệnh trừ phải thực hiện A - B.

Nói chung, vẽ giải thuật có lợi cho những bài toán có điều kiện, tức nhiệm vụ con này chỉ thực hiện khi kết quả thực hiện nhiệm vụ con khác thoả mãn một điều kiện nào đó đặt ra trước. Qua đây, phải có sự so sánh kết quả và có các luận lý nhất định.

*Ví dụ:* Ở ví dụ trên, chỉ nhận số liệu 7 bit cho số dương. Do đó, phải có nhiệm vụ con là kiểm tra số bit và chỉ nhận kết quả có 7 bit để đưa vào cộng lấy tổng. Ta có giải thuật như hình 4.2a. Từ hình 4.2a ta thấy:

- Sau khi nhập một số (A hoặc B), chương trình phải thực hiện việc kiểm tra số bit của một số (A hoặc B).

- Nếu thoả mãn điều kiện (các số nhỏ hơn hay bằng 7 bit), chương trình thực hiện nhiệm vụ nhập số A và B rồi tiến hành lấy tổng của hai số.

- Nếu không thoả mãn điều kiện (các số bằng và lớn hơn 8 bit) thực hiện nhiệm vụ quay trở về chờ nhập số mới.

Trong giải thuật, động tác cho sự luận lý (hay quyết định) là:

- So sánh hoặc kiểm tra số nhập vào với 8 (dùng lệnh so sánh CMP, Test).

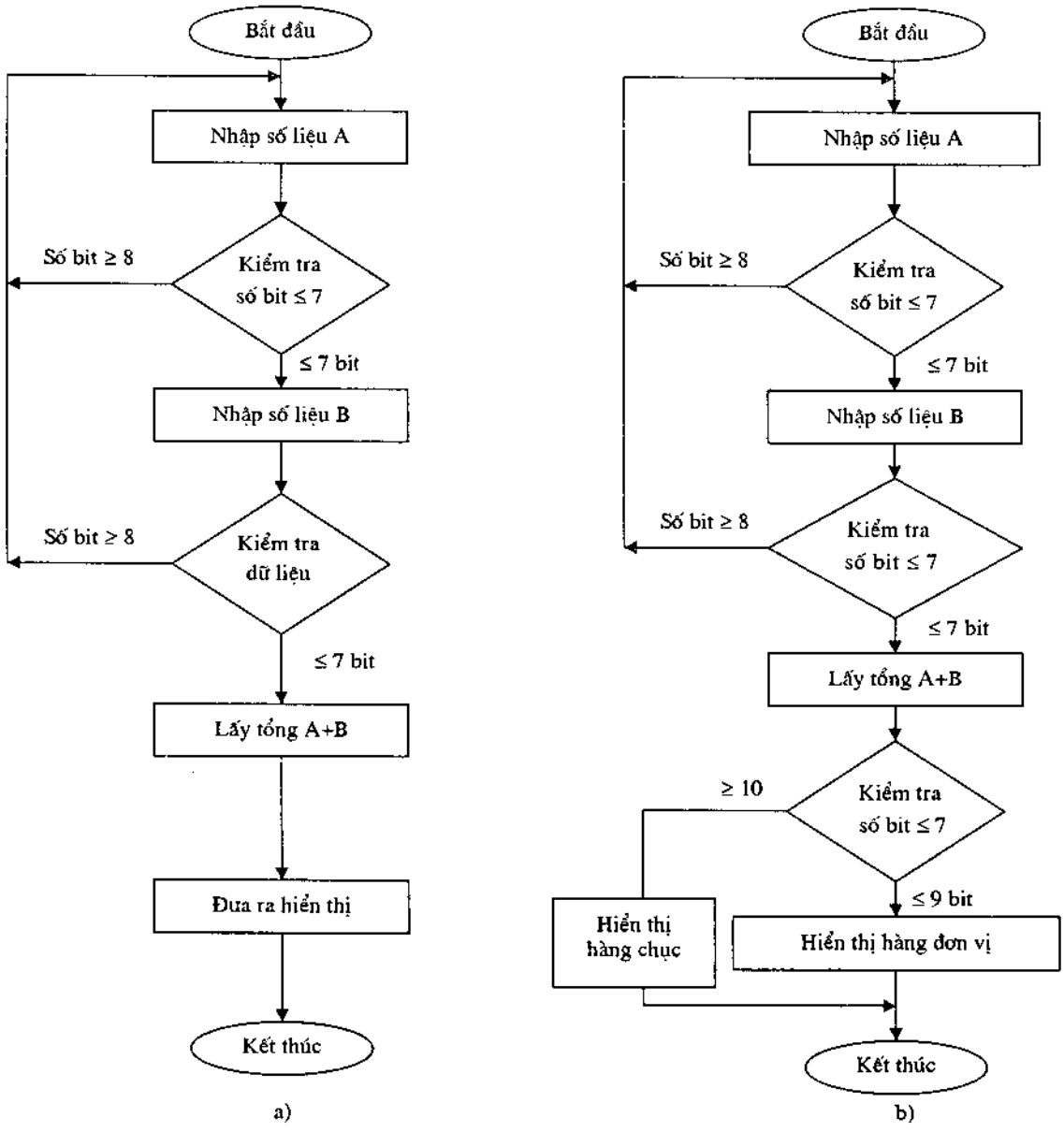
- Rẽ nhánh theo điều kiện nếu số đó bằng hoặc lớn hơn 8 bằng cách dùng lệnh nhảy hay rẽ nhánh có điều kiện (JMP điều kiện).

Ở hình 4. 2b ta thấy còn có thêm điều kiện số bit của kết quả để đưa ra biểu diễn trên màn hình. Mỗi lần đưa hiển thị, chỉ hiển thị một con số thập phân nhỏ hơn 9. Nếu tổng số lớn hơn 9, phải có thêm nhiệm vụ kiểm tra và xử lý tổng số với hai điều kiện:

- Nếu ≤ 9 đưa hiển thị một lần con số của tổng số.

- Nếu > 9 tức kết quả ≥ 10 và con số hiển thị lần đầu là phần đơn vị và ta phải hiển thị cả con số hàng chục.

Người lập trình phải và nên vẽ giải thuật trước để thấy được lập luận (hay logic) của nhiệm vụ được giao, để biết được các mối quan hệ giữa các nhiệm vụ con và điều kiện thực hiện chúng.




**Hình 4.2.** Giải thuật phép cộng hai số có kiểm tra số bit của số nhập vào (a) và có kết quả kiểm tra số bit để đưa ra hiển thị (b).

Nhìn vào giải thuật ta có thể hình dung được:

+ Trình tự điều kiện thực hiện các nhiệm vụ con.
+ Sơ đồ khối của chương trình.

## 2. Viết chương trình

Một chương trình bao giờ cũng có ba phần:

- Mở đầu của chương trình.
- Thân của chương trình.
- Kết thúc của chương trình.

Ở đây, chỉ xét việc viết chương trình để chạy với Debug của hệ MS. DOS. Còn chương trình chạy với chương trình dịch hợp ngữ (Assembler) sẽ xét ở chương 10.

*a) Mở đầu của chương trình*

Với Debug, không cần có lệnh hay ký hiệu gì để chứng tỏ là bắt đầu của chương trình vì khi chạy chương trình (xét ở dưới) chỉ cần gõ địa chỉ của lệnh bắt đầu. Nhưng khi viết chương trình, phải chừa ra 0100h byte nhớ của đoạn mã lệnh CS cho PSP (Prefix Segment Program - đoạn tiền tố của chương trình) để ghi các thông số về chương trình như lệnh kết thúc chương trình INT 20h, đỉnh của vùng nhớ, số byte trong đoạn, FCB (khối kiểm tra tệp, các thông số của chương trình v.v...). Khi soạn thảo lệnh đầu tiên của chương trình, ta dùng lệnh A 0100 ↵ để gọi địa chỉ của lệnh đầu tiên là 0100h. Trên màn hình sẽ hiện lên địa chỉ đầu tiên của chương trình như sau:

**xxxx: 0100,** để chờ ta gõ lệnh vào.

*b) Thân của chương trình*

Chương trình gồm cả lệnh đầu tiên ở **địa chỉ 0100h**. Sau lệnh đầu tiên trên, chỉ cần gõ **Enter** (↵), lệnh sẽ được dịch ra mã máy và ghi vào bộ nhớ. Sau đó DEBUG tính địa chỉ và hiện mới tiếp theo sau dòng lệnh đầu tiên và chờ gõ vào lệnh tiếp theo. Cứ như vậy, ta có thể soạn thảo và dịch được toàn bộ chương trình. Cần lưu ý là:

- Chương trình có thể có các lệnh LOOP, JMP và CALL nhưng không có nhãn đứng trước lệnh để đánh dấu vị trí cho các lệnh rẽ nhánh hay gọi chương trình con mà phải dùng địa chỉ thay nhãn.

- Trong các lệnh rẽ nhánh, tên nhãn được thay bằng địa chỉ của lệnh cần rẽ nhánh tới. Nếu rẽ nhánh gần, chỉ cần ghi giá trị của OFFSET, còn nếu rẽ nhánh xa, phải ghi cả giá trị của thanh ghi đoạn lệnh CS và OFFSET dưới dạng CS:OFFSET.

- Trong chương trình có thể có dữ liệu dạng một hay nhiều byte nhớ (được nạp trước bằng lệnh E hay F) và khi sử dụng chúng bằng cách gọi ngăn nhớ với địa chỉ CS: OFFSET chứa dữ liệu.

- Thường chương trình con nằm ngoài và ở dưới chương trình chính, bắt đầu bằng địa chỉ xuất phát (để chương trình chính gọi) và kết thúc bằng lệnh RET (trở về).

- Có thể gọi nhiều chương trình con lồng nhau.

*c) Kết thúc của chương trình*

Với Debug, kết thúc chương trình là lệnh ngắt của hệ điều hành DOS. Có thể dùng một trong các lệnh ngắt sau:

- INT 20h (mã lệnh CD20) của DOS: Debug điều khiển việc thực hiện chương trình, khi gặp lệnh trên, chương trình dừng và máy tính trở về môi trường gọi tức vẫn chịu sự điều khiển của Debug mà không trở về môi trường DOS với dấu nhắc C:>.

- INT 21h (mã lệnh CD21) với hai hàm kết thúc chương trình (nạp vào cho thanh ghi AH) trước khi có lệnh INT 21h là:

+ AH = 00h tương tự INT 20h, ít dùng.

```
Ví dụ:  MOV AH,00h    ; nạp 00h vào AH
        INT 21h       ; gọi ngắt INT21h của DOS.
```

+ AH = 4Ch, kết thúc chương trình và trở về môi trường DOS.

```
Ví dụ:  MOV AH 4Ch    ; chức năng 4C nạp vào AH
        INT 21h       ; gọi ngắt INT21 để thực hiện.
```

+ AH = 31h kết thúc và lưu chương trình ở lại thường trú trong bộ nhớ.

- INT 27h : kết thúc chương trình và lưu chương trình ở lại thường trú trong bộ nhớ.

- INT 3h : kết thúc chương trình và dừng chương trình tại điểm sau lệnh INT 3h. Người ta hay dùng INT 20h để kết thúc và chạy chương trình trong Debug. Ngoài ra, có thể kết thúc chương trình bằng các lệnh HLT (dừng).

***d) Viết chương trình***

Viết chương trình, tức là viết các chuỗi lệnh thực hiện các nhiệm vụ con của bài toán, tức là nhiệm vụ lớn theo hình vẽ của giải thuật. Khi gặp sự quyết định, ta phải dùng lệnh:

- So sánh (CMP), kiểm tra (test).

- Nhảy có điều kiện (Jcondition address), ở lệnh này, ta phải có địa chỉ (cho chạy trong Debug) hay nhãn (cho chạy trong chương trình hợp dịch) của lệnh mà chương trình sẽ nhảy tới (xem ví dụ cụ thể ở dưới).

*Ví dụ:* Theo ví dụ trên ta có các nhiệm vụ con của bài toán cộng hai số là nạp các số A, B và lấy tổng, mỗi nhiệm vụ tương ứng với một lệnh như sau:

```
MOV  AX, A  ; nạp số A vào AX
MOV  BX, B  ; nạp số B vào BX
ADD AX, BX  ; cộng hai số A, B, kết quả đặt trong AX.
```

Ở phần thân của chương trình này, người ta dùng các lệnh ngắt INT nh của DOS (xem chương về hệ điều hành). Muốn vậy, trước khi gọi ngắt INT nh, ta phải nạp các hàm (function) hay chức năng vào các thanh ghi AH, AL, BH, BL... như quy định của DOS. Dưới đây và các chương sau, chúng ta sẽ tận dụng các ngắt này của DOS, tức là các chương trình con viết sẵn để phục vụ cho hoạt động của máy vi tính.

Muốn lập trình tốt bằng hợp ngữ, chúng ta phải biết và tận dụng khả năng này của DOS, ở các mục sau, dần dần chúng ta sẽ giới thiệu các sử dụng trên.

## 4.3.2. MỘT SỐ DẠNG CẤU TRÚC CƠ BẢN CỦA CHƯƠNG TRÌNH

Chương trình ngắn có thể có một số dạng cấu trúc cơ bản là :

+ Cấu trúc tuần tự
+ Cấu trúc có vòng lặp
+ Cấu trúc rẽ nhánh
+ Cấu trúc có gọi chương trình con.

### 1. Cấu trúc tuần tự

Trong cấu trúc tuần tự, việc thực hiện chương trình theo thứ tự từng lệnh từ trên xuống dưới, từ đầu tới cuối chương trình, không có sự rẽ nhánh, không có vòng lặp. Nói khác đi, trong cấu trúc tuần tự, việc thực hiện lệnh theo địa chỉ tăng dần như danh sách ghi các lệnh, không có lệnh nào bị bỏ qua, cũng như không có lệnh nào được thực hiện hai lần.

*Ví dụ 1:* Chương trình cộng hai số A và B không có điều kiện ở trên.

```
MOV  AX, 17h      ; 17h → AX
MOV  AX, 25h      ; 25h → BX
ADD   AX, BX      ; cộng hai số trên
MOV  CX, AX       ; AX → CX. Kết quả chuyển vào CX
```

```
PUSH CX          ; gửi CX vào Stack
INT20h           ; kết thúc chương trình.
```

## 2. Cấu trúc có vòng lặp

Dùng lệnh lệnh LOOP hoặc lệnh nhảy có điều kiện Jcondition mà địa chỉ nằm trước lệnh đang thực hiện trong chương trình, ta đã tạo ra vòng lặp như ví dụ 2. Trường hợp đặc biệt của vòng lặp là chỉ lặp lại một lệnh nhiều lần khi có tiếp đầu ngữ REPT. Dĩ nhiên là cũng có sự lặp lại một lệnh có điều kiện như lệnh LOOP.

Chúng ta có thể tạo vòng lặp với lệnh LOOP, REPT hoặc Jcondition. Trước khi sử dụng lệnh LOOP, REPT cần nạp số lần lặp (quay vòng) vào thanh ghi CX. Trước khi nhảy trở về địa chỉ cần lặp lại lần thứ hai, thanh ghi CX được giảm đi 1 rồi mới lặp lại và khi CX = 0, lệnh LOOP hoặc mất tác dụng và chương trình thực hiện lệnh tiếp theo sau lệnh LOOP. Sau lệnh LOOP, có tham số địa chỉ của lệnh đầu tiên của vòng lặp. Dĩ nhiên là địa chỉ này phải có giá trị nhỏ hơn giá trị của địa chỉ của lệnh LOOP (nhảy trở về).

Tất nhiên là không có vòng lặp mà địa chỉ trở về lại lớn hơn hay đứng sau địa chỉ của lệnh LOOP.

*Ví dụ 2:* In ra màn hình 10 con số từ 0 tới 9 (có lệnh LOOP).

```
A100 ↵
xxxx: 0100   MOV  AH, 02h     ; chức năng hiển thị màn hình
xxxx: 0102   MOV   CX, 09h    ; nạp 9 → CX là số lần lặp
xxxx: 0105   MOV   DL, 30h    ; nạp mã ASCII của số 0
xxxx: 0107   INC DL           ; tăng DL lên 1
xxxx: 0109   INT 21h          ; in ra màn hình
xxxx: 010B   LOOP  107h       ; lặp lại 9 lần
xxxx: 010D   INT 20h          ; kết thúc chương trình.
```

Người đọc tự vẽ giải thuật và tìm hiểu.

## 3. Cấu trúc có rẽ nhánh

Rẽ nhánh là sự thay đổi tuần tự thực hiện lệnh ghi theo thứ tự từ trên xuống dưới như cách viết của chương trình. Rẽ nhánh là sự trở về thực hiện lại các lệnh đứng trước lệnh vừa thực hiện hoặc bỏ qua một số lệnh đứng sau lệnh vừa thực hiện. Đây là chương trình có lệnh nhảy không điều kiện JMP và có điều kiện (Jcondition).

Khi dùng lệnh nhảy không điều kiện, không cần phải xét điều kiện (xem có xảy ra hay không), mà hành động rẽ nhánh xảy ra tức thì, không có các lệnh so sánh hay thử điều kiện. Các lệnh đứng sau lệnh rẽ nhánh này không được thực hiện mà chỉ được thực hiện khi có điều kiện nào đó sẽ xét ở chỗ khác.

Trước khi dùng lệnh nhảy có điều kiện, ta phải có sự quyết định (decision) bằng cách dùng lệnh so sánh (CMP), lệnh thử (test) để xét điều kiện xảy ra xem có rẽ nhánh hay không.

Sau các lệnh thử điều kiện trên, các bit cờ (flag) bị ảnh hưởng và sẽ dùng các lệnh nhảy có điều kiện theo cờ (JC, JZ, JNE, JO v.v...).

Các lệnh rẽ nhánh (JMP) có hoặc không có điều kiện bao giờ cũng có tham số là địa chỉ ô nhớ mà lệnh cần chuyển tới. Về dạng, địa chỉ này có thể:

- *Chỉ có OFFSET:* dạng JMP OFFSET, đây là sự rẽ nhánh gần, lệnh cần rẽ nhánh tới nằm trong cùng một đoạn với chương trình chính, chúng có cùng một CS có giá trị như nhau.

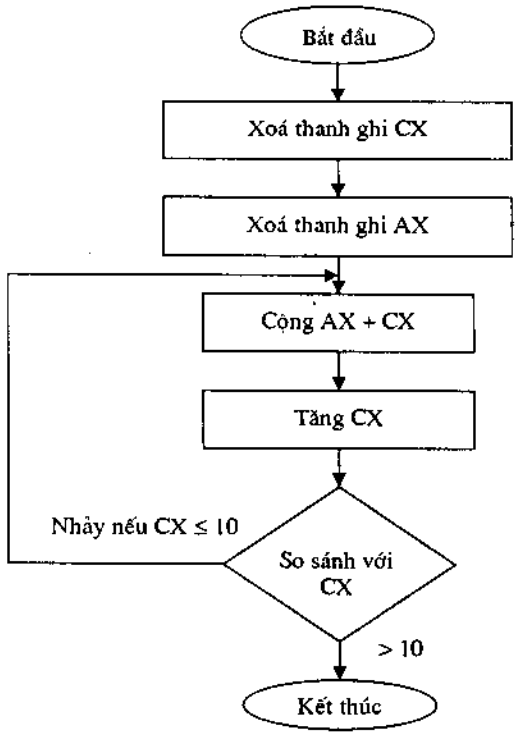
- *Có cả đoạn CS và OFFSET:* dạng CALL CS: OFFSET đây là sự rẽ nhánh xa, lệnh cần rẽ nhánh nằm trong các đoạn có giá trị CS khác nhau.

Về thứ tự của lệnh trong chương trình, lệnh cần rẽ nhánh tới có thể :

- *Đứng trước lệnh rẽ nhánh:* như vậy sẽ có một số hành động lặp lại để cho điều kiện xảy ra hay chờ điều kiện (kiểm tra trạng thái sẵn sàng, số lần giảm tới không, kết quả bằng 0 v.v...).

- *Đứng sau lệnh rẽ nhánh*: một số địa chỉ nhớ hay một số lệnh bị bỏ qua không thực hiện.

*Ví dụ 3*: Tính tổng 10 số nguyên dương đầu tiên (1 đến 10) dùng điều kiện là so sánh số 10 (hay dạng hexadecimal là OA). Giải thuật và chương trình như hình 4. 3.




a)

```
A 100 ↵
xxxx:0100 MOV CX,0000h    (1)
xxxx:0103 MOV AX,0000h    (2)
xxxx:0106 ADD AX,CX       (3)
xxxx:0108 INC CX          (4)
xxxx:0109 CMP CX,0A       (5)
xxxx:010C JBE 106         (6)
xxxx:010E NOP             (7)
```

b)

**HÌNH 4. 3.** Giải thuật (a) và chương trình (b) của phép cộng 10 số nguyên đầu tiên.

Qua chương trình trên, ta thấy:

- Lệnh (0) yêu cầu Debug dịch và ghi mã lệnh đầu tiên vào địa chỉ nhớ CS:IP = xxxx:0100 (xxxx: có thể tùy chọn).

- Các địa chỉ tiếp theo là do Debug tự điều khiển in và chờ lệnh. Qua thông số và địa chỉ có thể biết được độ dài mọi lệnh (1, 2, 3, 4 byte).

- Lệnh (5) so sánh CX với 10 (dạng thập phân) hay OA (dạng lục thập phân). Kết quả so sánh này ảnh hưởng đến cờ CF và ZF.

- Lệnh (6) nhảy nếu bé hơn hoặc bằng JBE tới địa chỉ 0106 tức địa chỉ của lệnh cộng (B).

Quá trình này cứ tiếp tục lặp lại cho tới khi CX = OAh. Khi CX = OAh, điều khiển không

thoả mãn cho lệnh nhảy (6) JBE 106 nên chương trình tiếp tục thực hiện lệnh của địa chỉ tiếp theo xxxx: 010E là lệnh NOP (không hành động gì) ở đây ta dùng NOP để kết thúc chương trình.

Chúng ta còn gặp lệnh nhảy này trở về địa chỉ của lệnh đứng trước lệnh so sánh như hình 4. 3 hoặc chờ trạng thái sẵn sàng của thiết bị ngoài (xem chương 5).

*Ví dụ 4:* Chương trình in ra hai số hexa của thanh ghi BL dùng lệnh nhảy có điều kiện bỏ qua một số lệnh (tham số địa chỉ của lệnh nhảy đứng sau địa chỉ của lệnh).

```
A 0100 ↵
xxxx: 0100  MOV AH, 02h      ; chuyển 02→AH
xxxx: 0102  MOV DH, BH       ; BH→DH
xxxx: 0104  AND DH, 0F       ; chỉ giữ lại 4 bit cuối của DH
xxxx: 0107  ADD DH, 30h      ; cộng với 30h
xxxx: 0109  CMP DH, 3A       ; so sánh với 00111010B
xxxx: 010D  JL 0112          ; nếu nhỏ hơn nhảy tới lệnh DH → DL
xxxx: 010F  ADD DH, 07h      ; cộng với 00000111B
xxxx: 0112  MOV DL, DH       ; chuyển DH → DL
xxxx: 0114  INT 21h          ; in ra màn hình số thứ nhất
xxxx: 0116  MOV DL, BL       ; BL → DL
xxxx: 0118  AND DL, 0Fh      ; lọc số, chỉ giữ lại 4 bit cuối 00001111B
xxxx: 011B  ADD DL, 30h      ; cộng với 00110000B
xxxx: 010E  CMP DL, 3A       ; so sánh với 00110110B
xxxx: 0121  JL 0126          ; nhảy tới địa chỉ 0126
xxxx: 0123  ADD DL, 07h      ; cộng với 07h
xxxx: 0126  INT 21h          ; in ra màn hình số thứ hai
xxxx: 0128  INT 20h          ; kết thúc chương trình.
```

Chương trình trên có sử dụng ngắt INT 21h với chức năng AH = 02h là chức năng in ra màn hình (xét kỹ sau ở chương 5).

Người đọc có thể tự vẽ giải thuật của bài toán và tìm hiểu.

## 4. Cấu trúc gọi chương trình con

Trong chương trình, đôi khi có các nhóm lệnh lặp đi lặp lại nhiều lần để thực hiện một nhiệm vụ cơ bản nào đó. Để tránh viết lệnh lặp lại nhiều lần, ta viết chúng thành một đoạn chương trình con (Subroutine) hay thủ tục (Procedure).

Chỉ cần thêm lệnh RET (trở về) ở dưới đoạn hành động lặp lại đó. Mỗi khi cần lặp lại chuỗi hành động cơ bản trên, ta chỉ cần nạp thông số cho nó (nếu cần) và dùng lệnh Call (gọi) với tham số là địa chỉ của lệnh đầu tiên của chuỗi hành động lặp lại.

*Ví dụ:* Hành động in ra màn hình được lặp lại nhiều lần, nên ta chọn làm chương trình con. Chương trình con, thường đặt ở cuối chương trình chính, sau lệnh kết thúc chương trình.

*Ví dụ 5:* In 5 chữ cái A B C D E lên màn hình (dùng vòng lặp và chương trình con ghi lên màn hình).

```
A 100 ↵
xxxx: 0100  MOV DL, 41h      ; mã ASCII chữ A→DL
xxxx: 0102  MOV CX, 05h      ; 5 lần → CX
```

```
xxxx: 0105    Call 10Eh          ; gọi chương trình con ở địa chỉ 10E
xxxx: 0106    LOOP 105h          ; trở lại lệnh Call
xxx: 010A     MOV AH, 4Ch        ; kết thúc chương trình về DOS
xxx: 010C     INT21h
xxx: 010E     MOV AH, 02h        ; 02h → AH chức năng in ra màn hình
xxx: 0110     INT21h             ; gọi ngắt đưa ký tự ra màn hình
xxx: 0112     INC DL             ; tăng giá trị mã của ký tự tiếp theo
xxx: 0114     RET                ; trở về chương trình chính tức lặp lại chu kỳ
                                   tiếp theo, tới 5 chữ (CX = 0).
```

## 4.4. NHẬP VÀ SỬA CHỮA CHƯƠNG TRÌNH BẰNG DEBUG

Sau khi viết chương trình trên giấy, ta phải nhập vào bộ nhớ, sửa và cho chạy trên máy vi tính.

Quá trình nhập, kiểm tra và sửa này là quá trình sử dụng các lệnh Debug thích hợp để ghi mã và thay đổi mã trong bộ nhớ.

### 4.4.1. NHẬP CHƯƠNG TRÌNH VÀO BỘ NHỚ

Việc nhập có hai dạng : dạng thuật nhớ hay hợp ngữ và mã lệnh.

#### 1. Nhập từ dạng hợp ngữ

Sau khi đã ở Debug, ta khởi động bằng gõ ký tự **A 100** ↵.

Trên màn hình in : **A100** ↵

**xxxx : 0100**

và chỗ gõ tên lệnh dạng hợp ngữ.

Nếu muốn chương trình bắt đầu ở đoạn bất kỳ nào đó, không phải giá trị của CS hiện tại (dùng lệnh R CS ↵ để xem), ta cần ghi vào thanh ghi CS một địa chỉ nào đó. Ta dùng lệnh R CS ↵ để ghi giá trị CS mới như sau: ghi lệnh **R CS** ↵

màn hình in **CS xxxx**

:

Nếu muốn sửa nội dung của CS, ta gõ giá trị mới của CS (4 số hexa) rồi gõ ↵. Nếu không muốn sửa, ta chỉ cần gõ ↵ thì lại trở về Debug và màn hình in dấu gạch ngang nhỏ (-).

Nếu muốn sửa nội dung IP ta cũng làm tương tự hoặc ngay từ đầu gõ A xxxx mong muốn.

Gõ lệnh ghi vào bộ nhớ: ta đánh máy từng lệnh dạng hợp ngữ, rồi ấn Enter (↵). Lệnh được Debug dịch và ghi vào ô nhớ có địa chỉ CS : 0100h.

Sau mỗi lệnh, địa chỉ nhớ được tự động tăng và in lên màn hình. Cứ như vậy, chương trình được nhập, dịch và ghi vào bộ nhớ tới lệnh cuối cùng của chương trình.

Muốn trở về Debug, chỉ cần gõ Enter (↵) sau dấu *địa chỉ :* của màn hình.

#### 2. Nhập từ dạng mã lệnh

Ở trường hợp này, ta phải có chương trình dạng mã lệnh do chúng ta tự dịch và bây giờ chỉ cần nhập các con số mã vào bộ nhớ. Có hai trường hợp:

*a) Nhập từng dòng lệnh*

*Khởi động:* gõ **E 100**↵.

Màn hình in ra: E 100 xxxx: 0100 $M_1$ $M_2$ ; $M_1$ $M_2$ là mã máy cũ.

Nhập từng mã lệnh: gõ mã mới $M_3$ $M_4$ ↵.

E 100↵.

xxxx: 0100 $M_3$ $M_4$

xxxx: 0102 $M_3$ $M_4$.

Nếu muốn nhập mã $M_3$ $M_4$ sau $M_1$ $M_2$, ta gõ thanh khoảng cách (Space bar) để di chuyển con trỏ về phía phải sau $M_2$ và gõ tiếp $M_3$ $M_4$.

Nếu trên màn hình đã có các mã $M_1$ $M_2$ $M_3$ $M_4$ , muốn sửa mã $M_2$ thành mã $C_2$, ta phải di chuyển con trỏ về phía trái tới vị trí $M_2$ rồi gõ $C_2$ ↵. Như vậy $C_2$ sẽ thay $M_2$ và in ra màn hình mã $M_1$ $C_2$ $M_3$ $M_4$ và con trỏ ở vị trí $M_3$.

*b) Nhập cả khối mã lệnh*

Nhập từng dòng bằng lệnh E rất lâu, ta có thể nhập cả khối mã lệnh bằng lệnh F.

*Ví dụ:* F100 L100 42 43 44 45. . . ↵ thì Debug sẽ nhập 100 dòng lệnh tại địa chỉ xxxx: 0100 bởi các mã 42, 43, 44, 45. . . là các mã lệnh của chương trình.

## 4.4.2. KIỂM TRA VÀ SỬA CHƯƠNG TRÌNH

### 1. Xem lại dạng văn bản của hợp ngữ

Dùng lệnh **U** để dịch ngược lại từ mã máy sang hợp ngữ để kiểm tra. Ta gõ:

**U xxxx** ↵ ; trong đó xxxx là địa chỉ đầu tiên của chuỗi lệnh của chương trình muốn hiện lên màn hình. Mỗi lần, màn hình chỉ hiện lên 16 dòng, nên muốn xem một chương trình dài, ta phải dùng nhiều lần lệnh U với các giá trị xxxx tiếp theo thích hợp.

### 2. Kiểm tra dạng mã

Kiểm tra từng dòng lệnh: dùng lệnh **E** ↵.

Kiểm tra cả vùng nhớ: dùng lệnh **D** ↵.

### 3. Sửa chương trình

Sửa lại cách viết lệnh và số liệu ở dạng hợp ngữ của một hoặc nhiều lệnh, dùng lệnh **A** ↵.

Sửa lại dạng mã cả một lệnh hoặc thay đổi thông số của lệnh, dùng lệnh **E** ↵ để xem nội dung lệnh và gõ byte lệnh hay thông số cần sửa.

Chèn thêm một số lệnh vào chương trình: dùng lệnh **M** ↵ để dịch chuyển cả vùng nhớ đi một số dòng. Viết chương trình bổ sung vào chỗ khối nhớ vừa dịch chuyển.

## 4.5. CHO CHẠY THỬ CHƯƠNG TRÌNH

### 1. Chạy thử từng dòng lệnh

Dùng lệnh **T** ↵ để thực hiện và dùng lệnh **R** ↵ để quan sát nội dung các thanh ghi trước, xóa thanh ghi và quan sát sau khi thực hiện lệnh. Nên dùng lệnh **P** ↵ để cho chạy từng lệnh hoặc cả nhóm lệnh trong một tiến trình (như chương trình con).

## 2. Chạy thử một khối (nhiều lệnh) của chương trình

Dùng lệnh **P** ↵ xem các lệnh lặp (LOOP) lệnh gọi chương trình con (Call) hay lệnh ngắt (INT nh). Trường hợp muốn thực hiện một số lệnh, ghi: **P = 0100 4** tức là thực hiện 4 lệnh kể từ địa chỉ đầu 0100h.

Dùng lệnh **G** với cú pháp: G address1 address2 ↵

thực hiện lệnh từ địa chỉ address1 đến địa chỉ address2.

## 3. Chạy cả chương trình

Dùng lệnh **G** với cú pháp: **G0100** ↵ ; chạy cả chương trình từ địa chỉ đầu 0100h tới lệnh kết thúc chương trình (INT 20h để ở lại Debug hay AH = 4Ch và INT 21h để trở về môi trường DOS) hay **G** ↵.

Ta cần lưu ý là muốn chạy cả chương trình, từ địa chỉ 0100h, trước khi thực hiện lệnh **G** ↵, ta phải sửa lại IP sao cho IP = 0100h bởi lệnh **R IP** ↵. Nếu tại địa chỉ khác 0100h, ta dùng **G address1** ↵, trong đó **address1** là địa chỉ ban đầu bất kỳ hoặc sửa IP có giá trị mong muốn bằng lệnh **R IP** ↵.

Có thể cho chạy chương trình có nhiều điểm dừng, ta ghi lệnh như sau:

G address1 address2 address3 ...

Bằng cách này, có thể đặt được tối đa tới 10 điểm dừng trong một chương trình. Sau mỗi điểm dừng đã được thực hiện, điểm dừng đó bị xoá đi và chương trình hiện lại lệnh G với các điểm dừng còn lại. Cứ lần lượt như vậy, ta có thể thực hiện chương trình cho hết điểm dừng. Thứ tự các điểm dừng có thể tùy ý, không phải theo trật tự từ thấp đến cao.

Nếu chương trình chạy tốt, trên màn hình in ra nội dung các thanh ghi của vi xử lý, cả các bit của thanh ghi cờ và dòng chữ:

**"Program terminated normally"** (chương trình đã kết thúc bình thường).

Nếu không có hàng chữ trên, ta phải xem lại chương trình (dùng lệnh A, U, E) để sửa cho chạy tốt. Trường hợp xấu nhất, máy bị treo hoặc chạy trong vòng lặp không thể thoát được ta lại phải xác lập lại máy (RESET) và chương trình ghi ở bộ nhớ đệm bị mất, ta lại phải nhập lại từ đầu. Để tránh sự kiện này, trước khi cho chạy cả chương trình, ta cần cho chương trình chạy từng bước (từng lệnh hoặc nhóm lệnh) để "gỡ rối" (debug), tức tìm ra những chỗ sai (từng lệnh hoặc từng nhóm lệnh) để sửa bằng cách sửa thông số của lệnh cho phù hợp.

# 4.6. GHI/NẠP LẠI CHƯƠNG TRÌNH LÊN ĐĨA TỪ

Tiến hành theo các bước sau:

*1. Đặt tên cho chương trình lên đĩa* bởi lệnh **N filename [arglist]** ↵.

*2. Ghi tên N lên đĩa*

Bởi lệnh W [[address [drive first sector number]] ↵

hay W [address] ↵, ghi chương trình lên đĩa.

*3. Nạp lại chương trình vào bộ nhớ MS. DOS*

Khởi động Debug bởi lệnh **Debug** ↵ trên màn hình có dấu nhắc (-).

Nạp đĩa bởi lệnh: **N\SK\SK. com** ↵.

**L** ↵ nạp từ địa chỉ **CS:0100.**

Hoặc L[address], nạp vào địa chỉ [address]

L address drive firstsector number ↵.

## CÂU HỎI

1. Debug là gì ? Vai trò và tác dụng của Debug trong lập trình bằng hợp ngữ ?

2. Các nhóm lệnh của Debug, những nhóm nào là hay dùng nhất cho việc lập trình.?

3. Kể tên các lệnh soạn thảo chương trình của Debug để so sánh với chương trình soạn thảo EDIT của MS-DOS, xem còn thiếu lệnh nào và cách khắc phục. Nhận xét ưu nhược điểm của cả hai loại.

4. Thống kê và so sánh các lệnh trong nhóm biểu diễn tin. Các lệnh của MS-DOS thực hiện được những lệnh nào. Muốn biểu diễn tin mà không dùng lệnh của Debug và của DOS, ta phải lập trình như thế nào? Kể nguyên tắc.

5. Kể các lệnh của nhóm di chuyển tin. So sánh chúng và so sánh với các lệnh tương đương của MS-DOS. Kể các lệnh không có của MS-DOS so với Debug?

6. Kể và so sánh các lệnh thực hiện chương trình (T, P, G) của Debug.

7. Nếu sử dụng MS-DOS, ta có thể thực hiện lệnh tương đương với T, P, G không ? Cách sử dụng chúng ?

8. Kể tác dụng của lệnh H, cách sử dụng ?

9. Kể mối liên quan giữa các lệnh của Debug ? Khi thực hiện một nhiệm vụ như biểu diễn nội dung của thanh ghi, ta có thể dùng những lệnh nào? Có thể kể thêm cho các nhiệm vụ khác ?

10. Kể tên các cặp lệnh mà hành động của chúng trái ngược nhau trong số các lệnh của Debug ?

## BÀI TẬP

1. Muốn gọi một địa chỉ bất kỳ của một ngăn nhớ ta thực hiện những lệnh gì?

2. Lập trình để gọi một địa chỉ bất kỳ của lệnh, số liệu, ngăn xếp trong bộ nhớ của máy vi tính PC-IBM.

3. Lập trình để thực hiện lệnh A, E, F của Debug.

4. Lập trình thực hiện kết quả như lệnh H của Debug, tiến hành thay đổi số liệu A và trong ví dụ của lệnh H của Debug để cộng và trừ hai số bất kỳ mà không phải viết lại lệnh của chương trình.

5. Dùng lệnh E hay F để nạp dữ liệu dạng ký tự (mã ASCII) vào các ngăn nhớ, rồi lập trình đưa dòng chữ trên lên màn hình (dùng chức năng AH = 02 của ngắt INT 21).

6. Lập trình cho các hành động tương đương với lệnh I và O của Debug.

7. Lập trình cho các hành động tương đương của L và W của Debug.

8. Lập trình cho hành động tương đương của lệnh D của Debug.

9. Lập trình cho hành động tương đương của lệnh C của Debug.

10. Lập trình cho hành động tương đương của lệnh S của Debug.

*CHƯƠNG 5*

# LẬP TRÌNH CHO THIẾT BỊ NGOÀI THÔNG DỤNG

Máy vi tính hoạt động được nhờ trao đổi tin (đưa tin vào, đưa tin ra) với các thiết bị ngoài. Chúng ta sẽ nghiên cứu sâu về thiết bị ngoài (chương 6, 7, 8) của tài liệu Kỹ thuật ghép nối máy vi tính của cùng tác giả, còn ở đây chỉ xét vấn đề đảm bảo chương trình bằng hợp ngữ, tức là viết chương trình để điều khiển chúng hoạt động.

Thiết bị ngoài (TBN) của máy vi tính (MVT) có thể chia làm hai loại lớn:

- *Thiết bị ngoài thông dụng*: như bàn phím, màn hình, chuột, máy in song song, máy in nối tiếp, đĩa từ, cassette. Chúng được bán kèm MVT.

- *Thiết bị ngoài ứng dụng*: như các thiết bị đo lường, thiết bị điều khiển trong các hệ đo-điều khiển dùng MVT, do người sử dụng tự trang bị, ghép nối và viết chương trình phục vụ trao đổi tin giữa MVT và TBN.

Ở chương này, chúng ta sẽ xét cách lập trình điều khiển các TBN hoạt động, điều khiển MVT đưa thông tin vào, đưa thông tin ra các TBN. Để đơn giản, ta chỉ giới thiệu các lệnh chính mà bỏ qua phần khai báo của dạng hợp ngữ. Chương trình này có thể biên soạn, thông dịch và chạy với Debug (xem chương 4) của MS-DOS.

## 5.1. MỞ ĐẦU

### 5.1.1. VIẾT CHƯƠNG TRÌNH PHỤC VỤ THIẾT BỊ NGOÀI

Các chương trình điều khiển TBN thông dụng đã viết sẵn ở trong hệ điều hành MS-DOS (xem chương 2) dưới dạng các ngắt INT nh tương ứng để gọi chúng. Ở đây, n là số hiệu ngắt, dạng hexadecimal (16) dành cho từng TBN, do hệ điều hành MS-DOS quy định. Mỗi một TBN có một số hiệu riêng, đặc trưng cho TBN đó. Khi gọi các chương trình phục vụ ngắt này, chỉ cần viết lệnh **INT nh** trong chương trình dạng hợp ngữ. Cuối mỗi chương trình phục vụ ngắt có sẵn trong hệ điều hành MS-DOS, đều có lệnh trở về chỗ bị ngắt của chương trình chính.

Có hai phương pháp lập trình cho các TBN:

- Lập trình trực tiếp cho phần cứng tức điều khiển các khối điều khiển các TBN như bàn phím, màn hình, chuột, đĩa từ v.v...

- Sử dụng các ngắt INT nh của ROM-BIOS và DOS của hệ điều hành hay phần mềm hệ thống của một máy vi tính.

Theo phương pháp lập trình trực tiếp, người lập trình cần nắm vững cấu trúc và hoạt động của các khối điều khiển TBN, địa chỉ các thanh ghi, cấu trúc các thanh ghi điều khiển và trạng thái, địa chỉ của các ngăn nhớ đệm dành cho thiết bị ngoài đó.

Theo phương pháp lập trình dùng ngắt (phần mềm), người lập trình cần nắm vững phần mềm hệ thống hay hệ điều hành của MVT, trong đó có các ngắt. Người sử dụng cần biết cách dùng

các ngắt INT nh với các thông số vào (nạp vào các thanh ghi) và các thông số ra (đọc ở các thanh ghi) để biết kết quả thực hiện chương trình phục vụ ngắt INT nh. Chính chương trình của lập trình trực tiếp là tương tự với chương trình phục vụ ngắt, hệ điều hành của máy vi tính đã viết sẵn chương trình phục vụ các thiết bị ngoài để chúng ta sử dụng khi gọi ngắt.

Đối với cách lập trình trực tiếp can thiệp vào phần cứng hay cho các thiết bị ngoài ứng dụng, chúng ta phải tự viết lấy bằng cách:

+ Gán địa chỉ cho các cổng vào/ra quy định sẵn cho MVT - PC mà chúng ta nối ghép thiết bị ngoài ứng dụng vào cổng.

+ Đọc và kiểm tra trạng thái sẵn sàng hay yêu cầu trao đổi tin của thiết bị ngoài.

+ Gán số liệu cần đưa ra vào thanh ghi AX.

+ Đưa số liệu ra bằng lệnh đưa tin ra (OUT) hoặc đọc số liệu vào bằng lệnh đưa tin vào (IN).

Khi sử dụng các ngắt INT nh của hệ điều hành, ta cần:

- Nạp các chức năng (function) hay hàm phục vụ (service) thích hợp cho từng nhiệm vụ của TBN vào thanh ghi AH và chức năng con vào thanh ghi AL.

- Nạp các thông số cần thiết khác vào các thanh ghi thông dụng khác như số hiệu của trang màn hình (BH), tọa độ của ảnh điểm (CX, DX), v.v...

- Gọi ngắt INT nh tương ứng cho thiết bị đó.

## 5.1.2. CÁC NGẮT CỦA MS-DOS DÀNH CHO THIẾT BỊ NGOÀI

Các ngắt chung của hệ điều hành (BIOS và DOS) đã trình bày ở chương 2. Ở đây chúng ta chỉ nhắc lại danh sách các ngắt riêng cho các thiết bị ngoài thông dụng.

### 1. Ngắt của BIOS

- *INT 9h* : ngắt phục vụ cho bàn phím mỗi khi phím được ấn hay nhả. Phục vụ ngắt này sẽ đọc mã quét (scan) và chứa nó trong bộ nhớ đệm của bàn phím.

- *INT 10h* : ngắt phục vụ màn hình.

- *INT 13h* : ngắt phục vụ đĩa.

- *INT 14h* : ngắt phục vụ cổng nối tiếp cho việc liên lạc của MVT.

- *INT 15h* : ngắt phục vụ cassette và các bộ phận khác.

- *INT 16h* : ngắt phục vụ bàn phím.

- *INT 17h* : phục vụ máy in.

### 2. Ngắt của DOS

- *INT 20h* : ngắt dùng để kết thúc chương trình trả điều khiển về cho Debug hay DOS.

- *INT 21h* : dùng cho nhiều chức năng khác nhau để điều khiển nhiều thiết bị và hành động khác nhau. Riêng điều khiển TBN, ta có các hàm sau:

+ *00h* : kết thúc chương trình, không truyền mã lỗi.

+ *01h* : đưa ký tự vào từ bàn phím và in ra màn hình.

+ *03h* : chờ đưa vào một ký tự trên cổng nối tiếp.

+ *04h* : đưa ra một ký tự trên cổng nối tiếp.

+ *05h* : đưa ra máy in trên cổng song song.

+ *06h* : đưa vào/ra trực tiếp từ bàn điều khiển.

+ *07h :* đưa vào/ra trực tiếp từ bàn điều khiển (Console) không in ra màn hình.

+ *08h :* vào từ bàn điều khiển, không in ra màn hình.

+ *09h :* ra chuỗi ký tự chứa trong vùng nhớ có địa chỉ DS:DX (con trỏ đến chuỗi ký tự có kết thúc bởi dấu $).

+ *0Ah :* đưa từ bàn phím vào bộ nhớ đệm (địa chỉ bắt đầu DS:DX+2).

+ *0Bh:* đọc kiểm tra trạng thái bàn phím (AL = FF nếu một ký tự đang chờ).

+ *0Ch:* xóa bộ nhớ đệm của bàn phím và đọc bàn phím.

+ *4Ch:* kết thúc chương trình, trả điều khiển cho chương trình gọi nó.

- *INT 25h/INT 26h :* đọc/ghi trực tiếp lên đĩa từ.

- *INT 27h :* chấm dứt chương trình thường trú trong bộ nhớ.

- *INT 33h :* phục vụ chuột.

- *INT 67h :* phục vụ nhớ mở rộng EMS.

Các ngắt trên sẽ dùng với các chức năng khác nhau để viết chương trình điều khiển TBN.

## 5.1.3. CÁC THÔNG SỐ PHẦN CỨNG CỦA THIẾT BỊ NGOÀI

### 1. Cổng vào/ra

Khi lập trình trực tiếp bằng hợp ngữ, người viết chương trình phải biết địa chỉ của cổng vào và cổng ra nối với TBN đó để viết lệnh vào (IN) và ra (OUT) tương ứng. Không những các TBN thông dụng, mà các thiết bị ngoài ứng dụng cũng đều có địa chỉ riêng do MS-DOS quy định. Sau đây là một số cổng của một số TBN thông dụng cho hệ máy IBM PC.

| Địa chỉ | Thiết bị |
|---|---|
| 000-03F | Bộ điều khiển DMA 8237A thứ 1 |
| 020-03F | Bộ điều khiển ngắt 8259A thứ 1 |
| 060-06F | Bộ điều khiển bàn phím 8042 |
| 0A0-0BF | Bộ điều khiển ngắt 8259A thứ 2 |
| 0C0-0DF | Bộ điều khiển DMA 8237A thứ 2 |
| 170-177 | Đĩa cứng 1 |
| 1F0-1F7 | Đĩa cứng 2 |
| 200-207 | Gậy điều khiển trò chơi |
| 2F8-2FF | Vào/ra nối tiếp COM2 |
| 370-377 | Đĩa mềm 2 |
| 378-37F | Máy in song song LTP1 và LTP2 |
| 380-38F | Cổng cho mạng theo SDLC |
| 3B0-3DF | Màn hình VGA |
| 3F8-3FF | Vào ra nối tiếp COM1 |
| 3D0-3DF | Card màn hình mầu CGA và EGA |
| 3F0-3F7 | Điều khiển đĩa mềm |

Trong các chương trình con phục vụ các ngắt INT nh, hệ điều hành đã sử dụng địa chỉ n (hay véctơ ngắt) để viết chương trình con. Với quan điểm của người lập trình cho các TBN ứng dụng, tức viết chương trình cho các thiết bị vào/ra song song và nối tiếp với các cổng còn trống, họ chỉ quan tâm tới địa chỉ cổng còn trống để ghép nối. Chúng ta sẽ dùng chúng cho bài toán cụ thể ở cuối chương này.

## 2. Các vùng nhớ tham số

Các biến hay các tham số của các ngắt BIOS, DOS chứa các dữ liệu cho các ngắt. Có các địa chỉ nhớ dành riêng cho từng ngắt hay từng thiết bị ngoài để trỏ tới các bảng tham số cho các thiết bị ngoài đó. Đặc biệt là khi lập trình trực tiếp, ta phải biết địa chỉ của các bảng tham số này để cung cấp tham số cho chương trình. Còn khi lập trình dùng ngắt của hệ điều hành, chương trình con phục vụ ngắt cũng sẽ đọc các tham số này, người lập trình không cần phải đọc tham số nên có thể không cần biết chúng.

***a) Địa chỉ nhớ cho các biến của BIOS cho các thiết bị ngoài***

Bảng 5.1 thống kê các địa chỉ và kích thước vùng nhớ chứa các tham số cho các thiết bị ngoài thông dụng và ghép nối.

***Bảng 5.1.*** **Các địa chỉ nhớ cho các thiết bị ngoài**

| Địa chỉ | Kích thước | Nội dung |
|---|---|---|
| 0000:0400 | 2 byte | Địa chỉ cơ sở của khối ghép nối RS 232 thứ nhất (COM 1) |
| 0000:0402 | 2 byte | Địa chỉ cơ sở của khối ghép nối RS 232 thứ hai (COM 2) |
| 0000:0404 | 2 byte | Địa chỉ cơ sở của khối ghép nối RS 232 thứ ba (COM 3) |
| 0000:0406 | 2 byte | Địa chỉ cơ sở của khối ghép nối RS 232 thứ tư (COM 4) |
| 0000:0408 | 2 byte | Địa chỉ cơ sở của khối ghép nối máy in song song thứ nhất (LPT1) |
| 0000:040A | 2 byte | Địa chỉ cơ sở của khối ghép nối máy in song song thứ hai (LPT2) |
| 0000:040C | 2 byte | Địa chỉ cơ sở của khối ghép nối máy in song song thứ ba (LPT3) |
| 0000:040E | 2 byte | Địa chỉ cơ sở của khối ghép nối máy in song song thứ tư (LPT4) |
| 0000:0410 | 2 byte | Danh sách trang thiết bị (phần cứng) |
| 0000:0412 | 1 byte | Các lỗi trong phần kết nối với bàn phím PC jr |
| 0000:0413 | 2 byte | Tổng bộ nhớ theo Kbyte |
| 0000:0415 | 2 byte | Đệm dùng cho kiểm tra lỗi chế tạo |
| 0000:0417 | 2 byte | Các cờ trạng thái bàn phím |
| 0000:0419 | 1 byte | Giá trị của các phím Alt + số (khung phím số) |
| 0000:041A | 2 byte | Địa chỉ đầu vùng đệm bàn phím |
| 0000:041C | 2 byte | Địa chỉ cuối vùng đệm bàn phím |
| 0000:041E | 32 byte | Vùng đệm bàn phím (0000:041Eh tới 0000:043Dh) |
| 0000:0449 | 1 byte | Chế độ màn hình hoạt động |
| 0000:044A | 2 byte | Chiều rộng của màn hình tính theo cột văn bản |
| 0000:044C | 2 byte | Chiều dài (byte) của vùng nhớ màn hình |
| 0000:044E | 2 byte | Offset tính từ phân đoạn màn hình của trang vùng nhớ màn hình |
| 0000:0450 | 16 byte | Vị trí con trỏ (8 cặp byte; byte thấp = cột, byte cao = hàng) |
| 0000:0460 | 2 byte | Kích thước con trỏ (byte thấp = dòng quét cuối cùng, byte cao = dòng đầu) |
| 0000:0462 | 1 byte | Số hiệu trang màn hình hoạt động hiện hành |
| 0000:0463 | 2 byte | Địa chỉ cổng của chip MC 6845 |
| 0000:0465 | 1 byte | Giá trị hiện tại chế độ màn hình của MC 6845 (thanh ghi cổng 3x8h) |
| 0000:0466 | 1 byte | Giá trị hiện tại của mầu màn hình của MC 6845 (thanh ghi cổng 3x9h) |
| 0000:0478 | 4 byte | Giá trị vượt quá thời gian của máy in (478h = LPT1, 47Ah = LPT2) |
| 0000:047C | 2 byte | Giá trị vượt quá thời gian của RS 232 (47Ch = COM1, 47Dh = COM2) |
| 0000:0480 | 2 byte | Địa chỉ offset bắt đầu vùng đệm bàn phím cho AT, PS/2 |
| 0000:0482 | 2 byte | Địa chỉ offset bắt đầu kết thúc vùng đệm bàn phím cho AT, PS/2 |
| 0000:0484 | 1 byte | Hàng ký tự của EGA thứ nhất (cao nhất) |
| 0000:0485 | 2 byte | Số byte cho mỗi ký tự cho EGA (số dòng quét/ký tự cho chế độ đang chạy) |
| 0000:0487 | 1 byte | Thông tin phụ về EGA dùng màn hình đơn sắc |
| 0000:0488 | 1 byte | Thông tin phụ về EGA dùng màn hình đơn sắc |
| 0000:0496 | 1 byte | Bit 4 của trạng thái cờ của bàn phím AT = 1 (10h) |

*(tiếp bảng 5.1)*

| Địa chỉ | Kích thước | Nội dung |
|---|---|---|
| 0000:0497 | 1 byte | Trạng thái cờ của bàn phím AT để hiện các phím LOCK |
| 0000:0498 | 4 byte | Con trỏ trỏ đến trạng thái đợi 8 bit của người dùng (xem chức năng 86h của INT 15h, máy AT) |
| 0000:049C | 4 byte | Một micro giây trước khi người dùng máy AT chờ xong |
| 0000:04A0 | 1 byte | Trạng thái cờ của hành động chờ của người dùng (1= bận, 80h = đã qua, 0 = nhận biết) |
| 0000:04A8 | 4 byte | Địa chỉ bảng con trỏ SAVE-PTR của EGA |
| 0000:04F0 | 16 byte | Vùng trao đổi tin của các chương trình ứng dụng |
| 0000:0500 | 1 byte | Trạng thái in màn hình (00h = cho phép in, 01h= đang in, 0FFh= lỗi trong khi in màn hình) |

***b) Bảng các tham số***

Ngoài địa chỉ nhớ dành cho thông tin ngắn gọn về thiết bị ngoài, còn có các bảng tham số mô tả chi tiết các đặc trưng của thiết bị ngoài như bảng font chữ, bảng tham số màn hình, vùng dữ liệu của màn hình EGA, của màn hình VGA, bảng trạng thái/chức năng của VGA v.v... Bằng cách gọi các chức năng của ngắt, ta biết được địa chỉ của bảng và từ đó có thể đọc nội dung của bảng. Ví dụ, bảng tham số màn hình, có thể dùng ngắt INT 1Dh (địa chỉ 0000:0074). Bảng này được sử dụng khi đổi chế độ màn hình bằng chức năng 00h của ngắt INT 10h. Các giá trị trong bảng này mô tả các giá trị của thanh ghi phải được cất vào bộ điều khiển màn hình MC 6845, kèm theo các dữ liệu khác (ví dụ, số cột).

Vùng dữ liệu cho tham số màn hình có địa chỉ 0000:0449 (xem bảng trên), nhưng offset của bảng như bảng 5.2 sau.

***Bảng 5.2.* Các offset của bảng thông số màn hình**

| Offset | Kích thước | Nội dung |
|---|---|---|
| 00h | 10h | Giá trị các thanh ghi của MC 6845 cho các chế độ 40x25 |
| 10h | 10h | Giá trị các thanh ghi của MC 6845 cho các chế độ 80x25 |
| 20h | 10h | Giá trị các thanh ghi của MC 6845 cho các chế độ đồ họa |
| 30h | 10h | Giá trị các thanh ghi của MC 6845 cho màn hình đơn sắc |
| 40h | 02h | Kích thước vùng nhớ RAM của một trang màn hình (các chế độ 40x25) |
| 42h | 02h | Kích thước vùng nhớ RAM của một trang màn hình (các chế độ 80x25) |
| 44h | 02h | Kích thước gần đây của các chế độ màn hình có độ phân giải thấp |
| 46h | 02h | Kích thước gần đây của các chế độ màn hình có độ phân giải cao |
| 48h | 08h | Số đếm cột cho 8 chế độ |
| 50h | 02h | Các giá trị cho thanh ghi định chế độ (cổng 03D8h cho mỗi chế độ) |

## 5.1.4. DẠNG TỔNG QUÁT CỦA CHƯƠNG TRÌNH PHỤC VỤ THIẾT BỊ NGOÀI SỬ DỤNG NGẮT

Mỗi một chương trình phục vụ trao đổi tin đều có các phần sau :

- Ghi chức năng (hàm phục vụ) vào các thanh ghi AH (chức năng lớn) và AL (chức năng con) của VXL bởi lệnh MOV.

- Ghi các thông số vào cần thiết cho các chức năng vào các thanh ghi BX, CX, DX bằng lệnh MOV.

- Gọi ngắt hệ thống bằng lệnh INT nh dành cho TBN đó.

- Kết thúc chương trình : có thể dùng một trong các ngắt của MS-DOS như sau:

+ *Ngắt INT 20h*: sau khi thực hiện lệnh kết thúc này, MVT vẫn ở trong môi trường Debug nhưng có thông báo lên màn hình dòng chữ *" Program terminated normally"*.

+ *Ngắt INT 21*: trước khi gọi ngắt INT 21h, ta phải nạp hàm chức năng vào thanh ghi AH (lệnh MOV AH, giá trị của hàm chức năng). Có các chức năng sau:

AH = 00h : kết thúc này tương tự ngắt INT 20h.

AH = 4Ch : kết thúc này trả điều khiển MVT về cho MS-DOS (màn hình in dấu nhắc của DOS (C>)).

AH = 31h : kết thúc này tương tự AH = 4Ch, cần ghi mã trở về vào AL (lệnh MOV) nhưng còn ghi vào DX kích thước của bộ nhớ trong đoạn mã lệnh CS và lưu trữ chương trình trong bộ nhớ.

+ *Ngắt INT 27h*: kết thúc này ở lại thường trú trong bộ nhớ. DX ghi độ lệch (offset) của mã lệnh. Nếu không cần lưu chương trình thường trú trong bộ nhớ, chúng ta tuyệt đối không dùng lệnh ngắt này vì nó sẽ xóa chương trình đã ghi trong vùng nhớ mà chúng ta đã sử dụng các địa chỉ CS:IP để ghi chương trình của chúng, ta viết vào vùng nhớ đệm. Thông thường, chúng ta hay dùng các lệnh ngắt INT 20h, INT 21h.

Chương trình chưa ghi thường trú trong bộ nhớ, sẽ bị xóa đi khi thoát khỏi môi trường Debug.

## 5.2. LẬP TRÌNH PHỤC VỤ BÀN PHÍM

### 5.2.1. SƠ LƯỢC VỀ BÀN PHÍM

Bàn phím là công cụ để người điều hành đưa tin (lệnh, số liệu) vào MVT. Có hai loại bàn phím:

- Bàn phím điều khiển trực tiếp (console): có một số ít phím, thường đặt ở bàn điều khiển hệ tự động hoá dùng máy vi tính.

- Bàn phím thông dụng chứa vi xử lý và bộ nhớ đệm: có nhiều phím ấn (83, 84, 101, 102).

Hiện nay máy vi tính PC - IBM dùng bàn phím mở rộng 101-102 phím với 12 phím điều khiển F1÷F12.

Các phím có thể chia làm ba nhóm:

- *Các phím mã ASCII:* gồm các ký tự (chữ A÷Z), số (0÷9), dấu (. , ...).

- *Các phím chức năng và điều khiển chương trình* (cho bàn phím mở rộng): các phím này dùng để điều khiển chương trình như F1÷F12, các phím điều khiển ESC (ESCAPE - thoát).

- *Các phím dùng để soạn thảo văn bản:* HOME (về đầu dòng), END (về cuối dòng), Page Up (về trang trên), Page Down (về trang dưới), Ins (INSERT- chèn vào hay ghi đè lên), và DEL (Delete - hủy bỏ hay xóa ký tự sau đó), ENTER (CR - CARRIAGE RETURN - trở về đầu dòng), BACKSPACE - trở về vị trí cũ để xóa ký tự đã ghi) và TAB - bảng tức con trỏ nhảy 4 khoang trống dùng để vẽ bảng, dịch sang trái (←), dịch sang phải (→), dịch lên dòng trên (↑), dịch xuống dòng dưới (↓).

- *Các phím chuyển đổi trạng thái:* đó là các phím SHIFT(dịch), CTRL (điều khiển) và ALT (dừng). Tổ hợp của một trong các phím trên với các phím khác tạo nên một mã quét (xem phụ lục) để tạo các trạng thái khác nhau của một phím nào đó. Ví dụ:

+ Ấn phím SHIFT và một phím chức năng tạo mã ASCII của chữ hoa tương ứng với ký tự đó.

+ Ấn phím $F_i$ đồng thời với SHIFT hay CTRL tạo được thêm 2 trạng thái khác của $F_i$.

+ Ấn đồng thời hai phím CTRL và C tạo một mã để ngắt việc thực hiện chương trình.

- *Các phím khoá trạng thái:* đó là các phím:

+ Caps Lock : chuyển đảo chữ thường thành chữ hoa và ngược lại (giống phím SHIFT nhưng không có tác động tới chữ số).

+ Num Lock : chuyển đảo giữa các phím số và phím chức năng ($F_1 \div F_9$) trên phần phím đánh số.

+ Scroll Lock : chuyển đảo giữa việc di chuyển con trỏ và cuộn cửa sổ văn bản.

- *Các phím nóng* (hot key) : các phím này còn được gọi là các phím tác dụng tạm thời, được tạo bởi ngắt INT 09h của BIOS, có một số phục vụ tức thời trong IBM là :

+ *Chức năng khởi động lại* : các phím CTRL-ALT-DEL, khởi động lại khi máy bị treo.

+ *Chức năng in nội dung của màn hình ra máy in* (print Screen) : phím Print Scrn/SysRq, thủ tục thích hợp được đặt ở ngắt INT 5h của BIOS.

+ *Chức năng tạm ngừng* (pause) : phím Pause/Break, thủ tục được kích hoạt bởi INT 09h.

+ *Chức năng hủy bỏ* (break) : dùng kết hợp với phím CTRL thành CTRL/BREAK và ngắt INT 1Bh được thực hiện (tương tự phím Esc với mã 1Bh).

+ *Chức năng đáp ứng hệ thống* (System Request) : phím Print Scrn/SysRq, hoạt động tức thời cho việc ngắt nhiệm vụ đang thi hành hiện thời trong phần mềm ở chế độ bảo vệ (80286, 80386). Có thể dùng thay thế bằng CTRL/Esc hay Alt/Esc đối với hệ điều hành OS/2. Dùng phím này với chức năng AH = 85h của ngắt INT 15h, thanh ghi AL = 00h nếu phím được nhấn hoặc AL = 01h nếu phím đã được nhả.

Trong bàn phím có một vi xử lý (8048 cho PC XT và 8042 cho PC AT) để tạo mã cho một phím có mã quét và mã ASCII khi ấn và khi nhả. Bàn phím còn có bộ nhớ đệm 16 từ để ghi các mã quét trước khi truyền vào MVT. Bộ phận điều khiển bàn phím có thanh ghi điều khiển-trạng thái để máy vi tính ghi lời lệnh điều khiển bàn phím hoặc đọc thông báo về trạng thái (cờ) của phím (ấn, nhả, có mã quét ở bộ nhớ đệm, bộ đệm đầy v.v...) của bàn phím. Ngoài ra, còn có thanh ghi đệm số liệu để ghi mã tức số liệu đưa vào máy vi tính hoặc số liệu do máy vi tính đưa vào bàn phím.

Mỗi phím nhấn có tin được chứa trong một từ với :

- Byte cao chứa mã quét.

- Byte thấp chứa mã ASCII nếu là phím ASCII hay chứa các bit 0 nếu là phím chức năng.

Có thể tạo mã cho sự nhấn nhiều phím đồng thời.

Nếu có yêu cầu nhập mã của phím mà bộ đệm rỗng hệ thống đợi đến khi một phím được nhấn. Khi bộ đệm đã đầy, nếu người sử dụng nhấn phím, VXL sẽ phát ra tiếng kêu bíp và tự động chuyển mã vào bàn phím, mã từ bộ đệm có thể chuyển vào MVT khi ta gõ ENTER (↵) hay Carriage Return để yêu cầu VXL nhận tin vào và bộ đệm bị xóa về 0. Thứ tự nhận tin theo đúng thứ tự của phím đã bị gõ.

Khi có nhấn phím, thủ tục đọc phím theo trình tự sau :

- Bàn phím gửi một yêu cầu (INTR 9) đến vi mạch xử lý ngắt của VXL.

- Chương trình con phục vụ INT 9h nhận mã quét từ cổng vào/ra cho bàn phím và chứa nó vào một từ trong bộ đệm.

- Có thể dùng ngắt INT 21h chức năng 01h để đọc mã ASCII và hiển thị chúng lên màn hình.

## 5.2.2. CÁC NGẮT DÙNG CHO BÀN PHÍM

Có 5 ngắt liên quan tới bàn phím, có nguồn gốc khác nhau và cách sử dụng khác nhau (bảng 5.3). Ngắt INT 9h là ngắt do bàn phím phát ra (do thiết bị hay phần cứng), các ngắt còn lại do hệ điều hành DOS (BIOS, DOS) do chúng ta đưa lệnh ngắt INT nh vào chương trình để gọi chương trình con tương ứng phục vụ bàn phím hoạt động theo mong muốn của người sử dụng.

*Bảng 5.3.* **Các ngắt của bàn phím**

| Ngắt | Nguồn gốc | Công dụng |
|---|---|---|
| INT 09h | *Do thiết bị bàn phím* | Báo VXL là bàn phím yêu cầu trao đổi tin |
| | *Do chương trình :* | |
| INT 16h | - ROM-BIOS | Yêu cầu phục vụ bàn phím |
| INT 01Bh | - ROM-BIOS | Tạo ngắt khi có tổ hợp phím CTRL-C |
| INT 21h (nhiều chức năng con) | - DOS | Yêu cầu phục vụ bàn phím |
| INT 23h | - DOS | Yêu cầu xử lý ngắt CTRL-C |

## 1. Ngắt INT 9h

Ngắt này sinh ra do ấn phím của bàn phím, được đưa vào VXL qua vi mạch xử lý ngắt 8259A của PC (ngắt cứng). Ngắt này yêu cầu VXL họ 86 trao đổi tin (đọc trạng thái, xử lý trạng thái) và đọc mã vào VXL (mã quét, mã ASCII, mã chức năng). Ngắt này được các nhà viết chương trình hệ thống sử dụng để viết chương trình, người sử dụng không cần dùng.

## 2. Ngắt INT 16h

Ngắt này thuộc ROM-BIOS, tức là thuộc chương trình vào/ra cơ sở đã ghi nhớ trong bộ nhớ ROM. Ngắt này thường được sử dụng để viết chương trình phục vụ bàn phím cũng như INT 21h của DOS. Chỉ khác là sau khi sử dụng INT 16h, việc điều khiển MVT trả về cho chương trình điều khiển trước đó (ví dụ Debug) mà không trả lại DOS như INT 21h của DOS.

Các hàm phục vụ hay các chức năng của ngắt INT 16h như bảng 5.4.

*Bảng 5.4.* **Các phục vụ của INT 16h**

| Chức năng | Mô tả |
|---|---|
| 00h | Đọc ký tự bàn phím tiếp theo |
| 01h | Đọc và kiểm tra sự sẵn sàng của một phím |
| 02h | Đọc trạng thái phím Shift |
| 03h | Đặt tốc độ gõ phím tự động và trễ |
| 04h | Dành riêng |
| 05h | Ghi mã của một phím vào bộ nhớ đệm |
| 06h - 0Fh | Dành riêng |
| 10h | Đọc bàn phím mở rộng (101 phím) |
| 11h | Đọc trạng thái của bàn phím mở rộng |
| 12h | Đọc trạng thái mở rộng của phím Shift |

***a) Chức năng 00h***: đọc ký tự phím tiếp theo

- Thông số vào: AH = 00h.
- Thông số ra: AH = mã quét của phím
  AL = mã ASCII của phím.

***b) Chức năng 01h***: đọc và kiểm tra sự sẵn sàng của phím (đã nhấn chưa)

ZF = 1 chưa sẵn sàng

ZF = 0 đã sẵn sàng tức đã có một phím đã ấn.

- Thông số vào : AH = 01h.
- Thông số ra : AH = mã quét của phím (ZF = 0)
  AL = mã ASCII của phím (ZF = 0).

***c) Chức năng 02h*** : đọc trạng thái của phím Shift hay kiểm tra trạng thái một số phím điều khiển và một số chế độ vào phím.

- Thông số vào : AH = 02h.
- Thông số ra : AH = trạng thái của phím Shift với các bit trạng thái sau:

| | |
|---|---|
| bit 7 = 1 : Phím Insert đã ấn | bit 3 = 1 : Phím Alt đã ấn |
| bit 6 = 1 : Phím Caps Lock đã ấn | bit 2 = 1 : Phím Ctrl đã ấn |
| bit 5 = 1 : Phím Num Lock đã ấn | bit 1 = 1 : Dịch trái đã ấn |
| bit 4 = 1 : Phím Scroll đã ấn | bit 0 = 1 : Dịch phải đã ấn. |

*d) Chức năng AH= 03h*: đặt tốc độ gõ phím tự động và thời gian trễ.

- Thông số vào: AH = 03h
  AL = 05h
  BL = vận tốc gõ tự động với số :
  00h - 30 lần/sec
  01h - 26, 7 lần/sec (ký tự/sec)
  ...
  0Ah - 12, 0 lần/sec
  ...
  1Fh - 2, 0 lần/sec.
  BH = thời gian trễ với :
  01h - 500 ms
  02h - 750 ms
  03h - 1000 ms.
- Thông số ra : không có.

*e. Chức năng AH= 05h*: ghi vào bộ nhớ đệm

- Thông số vào : AH = 05h
  CH = mã quét của phím
  CL = mã ASCII.
- Thông số ra : AL = 00h (đã ghi)
  AL = 01h (bộ đệm đầy).

*f) Chức năng AH= 10h*: đọc phím mở rộng

- Thông số vào: AH = 10h.
- Thông số ra: AH = mã quét của phím
  AL = mã ASCII.

*g) Chức năng AH= 11h*: trạng thái của phím mở rộng

- Thông số vào: AH = 11h.
- Thông số ra: ZF = 1 không có hành động
  ZF = 0 có hành động
  AH = mã quét của phím
  AL = mã ASCII.

*h) Chức năng AH= 12h*: trạng thái phím Shift của bàn phím mở rộng

- Thông số vào : AH = 12h.
- Thông số ra : AH = trạng thái của Shift với các bit trên đây
  AL = trạng thái của Shift mở rộng:

bit 7 =1: SysReq được ấn
bit 6 =1: Caps Lock được ấn
bit 5 =1: Num Lock được ấn
bit 4 =1: Scroll Lock được ấn
bit 3 =1: Alt bên phải được ấn
bit 2 =1: Ctrl bên phải được ấn
bit 1 =1: Alt bên trái được ấn
bit 0 =1: Ctrl bên trái được ấn.

## 3. Ngắt INT 21h với các hàm cho bàn phím

Có nhiều chức năng (phục vụ) cho bàn phím như bảng 5.5.

Tuy cùng đưa ký tự vào từ bàn phím, so với chức năng INT 16h của BIOS, các chức năng của INT 21h này có các đặc điểm sau:

- Thực hiện chậm hơn.

- Có thể đưa vào và ra trực tiếp màn hình (chức năng 06h), không qua bộ nhớ đệm của bàn phím.

- Có thể hiện ký tự đưa vào ra màn hình (các chức năng 01h, 06h, 0Ah, 0Ch).

***Bảng 5.5.* Các chức năng phục vụ bàn phím của INT 21 của DOS**

| Chức năng AH | Mô tả |
|---|---|
| 01h | Vào ký tự, có in ra màn hình |
| 06h | Vào/ra ký tự trực tiếp từ thanh ghi DL ra màn hình, trừ khi DL = FFh |
| 07h | Vào trực tiếp từ bàn điều khiển (console), không in ra màn hình |
| 08h | Vào các ký tự từ bàn phím vào bộ đệm, không in ra màn hình |
| 0Ah | Vào chuỗi các ký tự từ bàn phím, ghi ở bộ đệm, có in ra màn hình. |
| 0Bh | Đọc và kiểm tra trạng thái của bàn phím |
| 0Ch | Xóa bộ đệm bàn phím và đọc bàn phím |

***a) Đọc trạng thái - xóa bộ đệm***

Đọc và kiểm tra trạng thái của bàn phím : chức năng **0Bh**

- Thông số vào : AH = Bh.
- Thông số ra : AL = mã ASCII của phím ấn, có phím nhấn
  AL = 00, không có phím nhấn
  AL = FF, ít nhất đã có một phím đã nhấn.

Xóa bộ đệm của bàn phím, đọc phím nhấn: **0Ch**

- Thông số vào: AH = 0Ch
  AL = 01h, 06h, 07h, 08h hay 0Ah.
- Thông số ra: AL = mã ASCII của phím nếu đã nhấn phím với các chức năng ghi ở AL với tư cách là các thông số vào.

Chức năng này chủ yếu là xóa bộ nhớ đệm của bàn phím. Sau đó nếu muốn nhập ký tự vào từ bàn phím, thay cho việc sử dụng chức năng vào ký tự riêng rẽ, ta đưa các chức năng này vào thanh ghi AL rồi mới gọi INT 21h. Chức năng 06h tiếp theo (trong AL) không nhập vào một ký tự mà chỉ sử dụng để đưa ký tự ra màn hình.

***b) Đưa vào từ phím nhấn, có in ra màn hình***

Đưa vào từ một phím nhấn : chức năng **01h**

- Thông số vào: AH = 01h

AL = không có.

- Thông số ra : AL = mã ASCII của phím.

Chức năng này thực hiện cả ba hành động sau :

. Chờ để đưa một ký tự từ một phím nhấn của bàn phím.

. In ký tự của phím đã nhấn ra màn hình.

. Chờ một hành động chuẩn của ngắt (CTRL-C).

Đưa vào từ một chuỗi phím nhấn: chức năng **0Ah**

- Thông số vào : AH = 0Ah.

- Thông số ra : mã của chuỗi ký tự ghi vào bộ nhớ có địa chỉ đầu là DS : DX.

Chức năng này dùng để ghi chuỗi 254 ký tự từ bàn phím và đồng thời in ra màn hình. Nếu muốn kết thúc chuỗi, chỉ cần nhấn phím Enter (mã 0D), nếu không, đủ số ký tự dành cho bộ đệm bàn phím, sẽ có tiếng kêu bíp và bộ đệm không nhận ký tự nữa và chờ xóa bộ đệm (chức năng 0Ch).

***c) Đưa vào từ phím nhấn, không in ra màn hình*** : chức năng **08h**

- Thông số vào: AH = 08h.

- Thông số ra : AL = mã ASCII của phím nhấn.

Chức năng này tương tự 01h, cũng chờ đưa vào mã của phím nhấn ở bộ đệm và AL nhưng chỉ khác là không in ra màn hình ký tự của phím đã nhấn.

***d) Vào/ra trực tiếp từ bàn điều khiển (console)***

Vào/ra trực tiếp, có in ra màn hình : chức năng **06h**

- Thông số vào: AH = 06h

DL = mã ASCII của ký tự muốn đưa ra màn hình.

DL = FFh nếu muốn đưa ký tự vào từ bàn phím.

- Thông số ra: AL = mã ASCII của phím nhấn.

Đây là một chức năng phức tạp gồm tổ hợp hai chức năng :

Đưa vào từ bàn phím (nếu đã ghi DL = FFh), thanh ghi đệm AL để ghi mã của phím nhấn:

- Nếu một phím đã nhấn, nó gửi mã ASCII tương ứng và xóa cờ ZF.

- Nếu một phím chưa nhấn, nó giữ cờ ZF.

Đưa in ra màn hình (DL ≠ FFh), mã trong thanh ghi đệm DL đưa ra in trên màn hình.

Vào/ra trực tiếp, không in ra màn hình : chức năng **07h**

- Thông số vào : AH = 07h.

- Thông số ra : AL = mã ASCII của phím nhấn.

Chức năng này chờ nhấn phím, không in ra màn hình, nhưng ghi vào thanh ghi AL. Nó hoạt động giống 01h nhưng khác là không in ký tự ra màn hình.

## 5.2.3. LẬP TRÌNH PHỤC VỤ BÀN PHÍM DÙNG NGẮT CỦA HỆ ĐIỀU HÀNH

### 1. Quy tắc chung

Như đã trình bày ở trên, một chương trình phục vụ thiết bị ngoài thông dụng nói chung, bàn phím nói riêng, ta cần thực hiện chuỗi hành động theo trình tự sau:

- Nạp số liệu của hàm chức năng vào thanh ghi AH, hàm chức năng con vào AL và các thông số khác vào các thanh ghi cần thiết của thông số vào.

- Gọi ngắt tương ứng, ở đây là INT 16h cho bàn phím.

- Nếu cần biết thông tin về bàn phím (ký tự, trạng thái Shift) ta cần đọc và kiểm tra các thông số ra ở các thanh ghi (bằng lệnh R AX ↵, R BX ↵, R CX ↵, R DX ↵ của DEBUG). Để đơn giản và dễ quan sát, chúng ta có thể dùng đưa kết quả ra màn hình bằng chức năng AH = 02h với số liệu trong thanh ghi DL và ngắt INT 21h.

Như đã giới thiệu ở chương 4 về DEBUG, ở đây chúng ta sẽ sử dụng :

- Không cần ký hiệu nào chỉ bắt đầu của chương trình mà chỉ cần ghi lệnh bắt đầu của chương trình vào địa chỉ xxxx: 0100, trong đó xxxx là nội dung của mã lệnh tức thanh ghi đoạn CS, còn 0100 là nội dung của thanh ghi đếm chương trình IP. Địa chỉ này có được khi vào môi trường DEBUG (xem chương 4).

- Dùng lệnh A xxxx: yyyy để soạn thảo (tăng tự động sau mỗi dòng lệnh) và thay thế từng dòng lệnh cần sửa.

- Dùng lệnh INT 20h hoặc MOV AH, 4Ch và INT 21h để kết thúc chương trình.

- Dùng lệnh E xxxx : yyyy để sửa nội dung của từng lệnh (cả mã lệnh lẫn thông số của lệnh).

- Dùng lệnh R ↵ để xem và sửa nội dung của các thanh ghi hoặc R tên thanh ghi ↵ để xem và sửa nội dung của từng thanh ghi.

- Cho chạy chương trình bằng các lệnh sau :

+ T ↵ : thực hiện từng lệnh.

+ P ↵ : thực hiện một nhóm lệnh theo một tiến trình.

+ G ↵: thực hiện cả chương trình. Sau khi thực hiện cả chương trình, màn hình in ra dòng chữ **"Program terminated normally"**.

+ *G địa chỉ đầu, địa chỉ cuối* ↵ : thực hiện đoạn chương trình giữa địa chỉ đầu và địa chỉ cuối.

- Dùng các lệnh *LOOP giá trị của IP* để lặp lại đoạn chương trình từ lệnh LOOP trở về địa chỉ chỉ bởi giá trị của IP.

- Dùng lệnh *JMP giá trị của IP* hoặc $J_{điều\ kiện}$ *giá trị của IP* để nhảy chương trình không điều kiện và có điều kiện tới địa chỉ chỉ bởi giá trị của IP.

- Dùng lệnh *CALL giá trị của IP* để gọi chương trình con có địa chỉ là giá trị của IP. Chương trình con này có địa chỉ bắt đầu là giá trị của IP ở trên và kết thúc bằng lệnh RET (trở về).

- Các số liệu được nhập vào biểu diễn trong hệ hexadecimal và không cần phải có chữ h ở cuối.

- Bỏ các nhãn đứng trước tên lệnh, còn nhãn đứng sau tên lệnh phải thay bằng địa chỉ của lệnh có nhãn đứng trước.

## 2. Một số chương trình ví dụ

### *a) Ví dụ 1*

Đọc và kiểm tra trạng thái sẵn sàng của phím. Nếu có ký tự, nó sẽ nhảy tới chương trình đưa ra màn hình, nếu chưa nó sẽ chờ nhấn phím. Chúng ta soạn thảo chương trình sau bằng lệnh A100 ↵ của DEBUG.

```
xxxx: 0100 MOV AH, 01h    ; đọc và kiểm tra phím đã ấn chưa (sẵn sàng) ?
                          (ZF = 0: đã sẵn sàng, đã ấn phím, ZF=1: chưa sẵn sàng)
```

```
xxxx: 0102 INT 16 h          ; gọi ngắt để kiểm tra.
xxxx: 0104  JNZ 10A          ; nhảy tới địa chỉ xxxx : 10A nếu có ký tự.
xxxx: 0106  MOV AH, 00 h     ; hàm đọc mã của phím nhấn của bàn phím.
xxxx: 0108 INT 16 h          ; gọi ngắt để đọc ký tự vào, AH = mã quét của phím,
                               AL = mã ASCII của phím.
xxxx: 010A  MOV BH, AH       ; lưu giữ mã quét trong BH
xxxx: 010C  MOV BL, AL       ; lưu giữ mã ASCII trong BL
xxxx: 010E  MOV AH, 02       ; in ra màn hình
xxxx: 0110  MOV DL, BH       ; chuyển mã quét vào DL
xxxx: 0112  INT 21h          ; đưa mã quét ra màn hình
xxxx: 0114  MOV DL, BL       ; chuyển mã ASCII vào DL
xxxx: 0116  INT 21h          ; in mã ASCII ra màn hình
xxxx: 0116  INT 20h          ; kết thúc chương trình.
```

Chương trình trên có thể cho thực hiện từng lệnh (lệnh T của DEBUG), từng tiến trình (lệnh P) hay cả chương trình (lệnh G). Khi tới lệnh gọi ngắt để đọc mã của phím nhấn (sau địa chỉ xxxx: 0108), chương trình chờ nhấn phím. Nếu ta nhấn một phím nào đó, ta sẽ thấy mã (quét và mã ASCII) in ra trên màn hình.

Chúng ta có thể :

- Đổi lệnh JNZ 10A bằng JZ 10A để đọc ra màn hình mã của phím đã ghi trong bộ đệm bàn phím.

- Thay các lệnh từ địa chỉ xxxx : 0104 tới xxxx : 0108 bằng các lệnh NOP (không hành động gì) để đọc các bit trạng thái của bàn phím ra màn hình thay vì việc đọc thanh ghi AX sau lệnh INT ở địa chỉ xxxx : 0102.

*b) Ví dụ 2*

Dùng ngắt INT 21h chức năng AH = 01, để nhập từ bàn phím một mã nào đó của một phím, rồi đưa ra màn hình (chức năng AH = 02) ký tự của phím đó và các ký tự khác có số mã tăng dần kể từ ký tự đầu tiên bất kỳ đó.

```
xxxx: 0100   MOV AH, 01h     ; chức năng nhập mã của phím nhấn
xxxx: 0102   INT 21h         ; gọi ngắt chờ nhập mã của phím nhấn vào AL
xxxx: 0104   MOV BH, AH      ; chuyển mã quét vào thanh ghi đệm BH
xxxx: 0106   MOV BL, AL      ; chuyển mã ASCII vào thanh ghi đệm BL
xxxx: 0108   MOV AH, 02h     ; chức năng đưa ra màn hình mã của phím
xxxx: 010A   MOV DL, BH      ; chuyển mã quét vào DH để đưa ra màn hình
xxxx: 010C   INT 21h         ; gọi ngắt đưa mã quét ra màn hình
xxxx: 010E   MOV DL, BL      ; chuyển mã ASCII vào DL
xxxx: 0110   INT 21h         ; gọi ngắt đưa mã ASCII ra màn hình
xxxx: 0112   INC BH          ; tăng mã quét
xxxx: 0114   INC BL          ; tăng mã ASCII
xxxx: 0116   CMP FFh         ; so sánh mã ASCII với FFh
xxxx: 0118   JNZ 108h        ; lặp lại nếu mã ASCII chưa có
xxxx: 011A   INT 20h         ; kết thúc chương trình nếu mã ASCII là FFh.
```

*c) Ví dụ 3:* Chương trình kiểm tra mã của phím nhấn để chuyển tới chương trình con tương ứng cho từng mã phím.

Trong điều khiển thực hiện chương trình, người ta dùng các phím nhấn để chuyển tới các chương trình con tương ứng được gán cho từng phím. Đặc biệt trong Directive của Monitor của hệ điều hành (chương 3) hay trong thực đơn của các chương trình ứng dụng.

Ví dụ, chương trình sau thể hiện thuật toán :

- Nhấn phím $F_1$ có mã quét 3Bh sẽ hiện lên màn hình dấu ";" có mã ASCII là 3Bh.
- Nhấn phím $F_2$ có mã quét 3Ch sẽ hiện lên màn hình dấu "<" có mã ASCII là 3Ch.
- Nhấn phím Esc có mã quét 1Bh sẽ hiện lên màn hình dấu "←" có mã ASCII là 1Bh.
- Hiện lên màn hình các ký tự (chữ cái a÷z, chữ số 0÷9 và các dấu) của phím nhấn tương ứng.

Người đọc có thể tự vẽ lấy giải thuật.

```
xxxx: 0100 XOR AX, AX     ; xóa AX về 0
xxxx: 0102 XOR BX, BX     ; xóa BX về 0
xxxx: 0104 XOR DX, DX     ; xóa DX về 0
xxxx: 0106 MOV AH, 00h    ; hàm nhập ký tự từ bàn phím
xxxx: 0108 INT 16h        ; gọi ngắt phục vụ bàn phím
xxxx: 010A MOV BH, AH     ; gửi mã quét vào BH
xxxx: 010C CMP AL, 00h    ; so sánh có phải mã quét của phím chức năng không
xxxx: 010E JNZ 11A        ; nếu không, nhảy tới địa chỉ IP = 11A
xxxx: 0110 CMP AH, 3Bh    ; xem mã quét có phải của F1 không ?
xxxx: 0113 JZ 12A         ; nếu đúng, nhảy tới IP = 12A để hiện ra màn hình
xxxx: 0115 CMP AH, 3Ch    ; xem mã quét có phải của F2 không ?
xxxx: 0118 JZ 131         ; nếu đúng, nhảy tới IP = 131 để hiện ra màn hình
xxxx: 011A CMP AL, 1Bh    ; xem có phải mã ASCII của phím Esc không ?
xxxx: 011C JZ 138         ; nếu đúng, nhảy tới IP = 138 để hiện ra màn hình
xxxx: 011E MOV DL, AL     ; xem có phải mã ASCII của phím ký tự không ?
xxxx: 0120 CALL 125       ; gọi chương trình con hiện ra màn hình
xxxx: 0123 INT 20h        ; kết thúc chương trình
xxxx: 0125 MOV AH, 02h    ; hàm ghi ra màn hình
xxxx: 0127 INT 21h        ; gọi ngắt phục vụ bàn phím
xxxx: 0129 RET            ; trở về nơi gọi chương trình con
xxxx: 012A MOV DL, AH     ; chuyển cho DL để đưa ra màn hình
xxxx: 012C CALL 125       ; gọi chương trình con đưa ra màn hình
xxxx: 012F INT 20h        ; kết thúc chương trình
xxxx: 0131 MOV DL, BH     ; chuyển cho DL để đưa ra màn hình
xxxx: 0133 CALL 125       ; gọi chương trình con đưa ra màn hình
xxxx: 0136 INT 20h        ; kết thúc chương trình
xxxx: 0138 MOV DL, AL     ; chuyển cho DL để đưa ra màn hình
xxxx: 013A CALL 125       ; gọi chương trình con đưa ra màn hình
xxxx: 013D INT 20h        ; kết thúc chương trình.
```

## 5.2.4. LẬP TRÌNH TRỰC TIẾP CHO BÀN PHÍM

Ngoài sử dụng chức năng AH = 02h (đọc trạng thái của phím SHIFT) và AH = 03h (đặt tốc độ gõ phím tự động và thời gian trễ) người ta còn lập trình trực tiếp cho bàn phím mà không sử

dụng ngắt INT 16h hay INT 21h. Muốn vậy, ta phải hiểu rõ phần thiết bị của bàn phím (bàn phím, khối điều khiển) và kỹ thuật lập trình cho thiết bị ngoài của máy vi tính PC - IBM.

## 1. Đặc điểm của điều khiển bàn phím

Máy vi tính loại PC XT và PC Portable được trang bị bộ điều khiển là vi xử lý 8048 với các địa chỉ cổng như sau :

- Cổng 60h, cổng số liệu, ghi mã quét, nối với cửa PA của vi mạch cổng song song lập trình được 8255.

- Cổng 61h, cổng trạng thái, ghi trạng thái của bàn phím (bit $d_6$ - cho phép/cấm bàn phím hoạt động, bit $d_7$ - bàn phím xác nhận), nối với cửa PC của 8255.

Máy vi tính PC AT được trang bị bộ điều khiển là vi xử lý 8042 có các cổng như sau :

- Cổng 60h, cổng số liệu để ghi mã từ bàn phím hoặc từ máy vi tính. Vi xử lý của máy vi tính phải đưa lệnh đọc cổng này (IN AL, 60h) để biết phím nào đã được nhấn. Ngược lại máy vi tính có thể đưa số liệu ra cho bàn phím.

- Cổng 64h, cổng điều khiển và trạng thái với các lệnh từ máy vi tính như sau :

+ *Ghi lời điều khiển bàn phím* : lệnh OUT 64h, AL (hoặc OUT DX, AL nếu trước đó đã nạp 64h vào DX). Lời điều khiển này dùng để điều khiển bàn phím hoạt động.

+ *Đọc trạng thái bàn phím:* lệnh IN AL, 64h (hoặc IN AL, DX nếu trước đó đã nạp 64h vào DX). Lời trạng thái này cho biết tình trạng hoạt động của bàn phím.

Dưới đây là dạng của lời điều khiển và lời trạng thái.

Lời điều khiển có dạng như sau :

| $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|
| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |

Trong đó:

- C0 dùng để cho phép (C0 = 1) và cấm (C0 = 0) bàn phím đưa yêu cầu ngắt vào INTR.
- C1 dùng để cho phép (C1 = 1) và cấm (C1 = 0) ngắt của thiết bị phụ.
- C2 dùng để đọc cờ của hệ cờ của hệ (bit 2 của thanh ghi trạng thái).
- C3 dùng để cho phép ghi đè (C3 =1) và cấm ghi đè (C3=0) như chức năng của phím Insert.
- C4 dùng để cấm (C4=1) bàn phím hoạt động.
- C5 dùng để điều khiển chế độ PC (C5 =1- cho phép bàn phím của PC AT hoạt động hay cấm thiết bị phụ của PS/2).
- C6 dùng để cho phép tương thích với PC (C6 =1- biến đổi mã quét thành giá trị của PC).
- C7 = 0.

Với tổ hợp của các bit (C6÷C0) của thanh ghi điều khiển ở trên, ta có tập hợp lệnh cho bàn phím của PC AT, trong đó một số lệnh hay dùng nhất như sau :

| Mã | Lệnh |
|---|---|
| 20h | đặt byte lệnh vào thanh ghi đệm ra |
| 60h | viết byte lệnh vào cổng 60h |
| AAh | tự kiểm tra. Kết quả là 55h ở cổng 60h nếu không có lỗi |
| ADh | cấm bàn phím hoạt động (bit 4 của thanh ghi lệnh =1). |
| C0h | đọc cổng vào tới cổng 60h |
| D0h | đọc nội dung cổng xuất tới cổng 60h nên được thực hiện khi bộ đệm ra trống. |

Lời trạng thái có dạng như sau :

| $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|
| PARE | TIM | TED | KEYL | C / $\overline{D}$ | SYSF | INPB | OUTB |

trong đó: - OUTB: bộ đệm ra bị rỗng (0) hay có số liệu (1).

- INPB: bộ đệm vào rỗng (0) hay có số liệu(1).
- SYSF: trạng thái của hệ (1- tự kiểm tra đã xong, 0 - bật nguồn hay xóa).
- C/$\overline{D}$ : chỉ dữ liệu là lệnh (C =1) ghi vào cửa 64h hay số liệu ($\overline{D}$ = 0) ghi vào cửa 60h.
- KEYL: chỉ bàn phím bị khóa (0) hay không (1).
- TED : chỉ lỗi trong khi truyền tin (1- có lỗi).
- TIM : chỉ lỗi nhận tin quá thời gian (time out, =1 - có lỗi).
- PARE: chỉ sai số chẵn (1), lẻ (0) của byte cuối cùng hoặc thiết bị phụ (chỉ ở PS/2).

## 2. Đặc điểm của lập trình trực tiếp cho bàn phím

Lập trình trực tiếp bàn phím có các nhiệm vụ sau :

- Điều khiển bàn phím hoạt động (cấm/cho phép, đặt tốc độ trễ, tốc độ truyền...).
- Trao đổi dữ liệu với khối điều khiển bàn phím (8042) qua cổng 64h.
- Trao đổi dữ liệu với bàn phím (8042) qua cổng 60h.

Các lời trao đổi có thể là lệnh (ghi ra), trạng thái (đọc vào) hay số liệu của mã (quét, ASCII).

Trong lập trình, chủ yếu có các hành động và các lệnh sau:

- Gán địa chỉ cổng cho thanh ghi DX bằng lệnh MOV DX, AL, trong đó AL phải chứa địa chỉ cổng bằng lệnh MOV AL, địa chỉ cổng.
- Đọc trạng thái của bàn phím bằng lệnh IN AL, 64h.
- Kiểm tra và chờ trạng thái sẵn sàng trao đổi số liệu của bàn phím bằng các lệnh :

+ TEST AL, trạng thái cần ; ở đây trạng thái đã đọc vào AL, còn trạng thái cần là một con số (nhị phân, thập phân) tương ứng với thanh ghi trạng thái đã trình bầy ở trên.

+ JNZ địa chỉ của lệnh nhảy tới ; ở đây địa chỉ lệnh nhảy tới thường là lệnh đọc trạng thái để chờ trạng thái sẵn sàng. Có thể dùng lệnh CMP AL, trạng thái cần nhưng nội dung mới của AL sau lệnh so sánh là hiệu số của AL với trạng thái cần.

- Trao đổi dữ liệu bằng các lệnh sau :

+ OUT địa chỉ cổng, AL để đưa dữ liệu từ máy vi tính ra bàn phím hay khối điều khiển bàn phím.

+ IN AL, địa chỉ cổng để đọc dữ liệu từ bộ đệm vào máy vi tính.

## 3. Các chương trình ví dụ của lập trình trực tiếp cho bàn phím

### *a) Chương trình ghi lệnh*

Chương trình này thực hiện cấm ngắt, đọc và chờ trạng thái sẵn sàng của bàn phím (CX từ FFFFh tới 0000h) và ghi lệnh điều khiển ADh (cấm bàn phím hoạt động) ra cổng 64h.

```
        CLI                     ; cấm ngắt
        MOV CX, 0FFFFh          ; nạp số lần giảm CX
WAIT:   IN AL, 64h              ; đọc trạng thái tại cổng 64h
        TEST AL, 00000010h      ; bộ đệm vào bị đầy ?
```

```
        LOOPNZ WAIT          ; chờ bộ đệm đầy
        MOV AL, 0ADh         ; gửi lệnh mã 0ADh cấm bàn phím
        OUT 64h, AL          ; hoạt động
        STI                  ; cho phép ngắt trở lại
        INT 20 h             ; kết thúc chương trình
```

Nếu chạy chương trình trong môi trường DEBUG, ta chỉ cần :

- Bỏ chữ WAIT trước lệnh IN AL, 64h.
- Thay chữ WAIT sau lệnh LOOPNZ bằng địa chỉ của lệnh IN AL, 64h.

***b) Chương trình ghi lệnh hay byte số liệu vào cổng 60h***

Chương trình này cũng kiểm tra và chờ trạng thái xóa các bit :

- bit 4 = thông báo đã nhận
- bit 5 = cờ nhận gửi lại của byte cờ của bàn phím ở địa chỉ nhớ 0040: 0097h và kết thúc chương trình nếu có lỗi (bit 7 = 1).

```
        MOV CX, 40h          ; offset của nhớ số liệu ES
        MOV ES, CX
        MOV BH, AL           ; lưu trữ AL trong BH
        MOV BL, 03           ; số lần lặp lại
REPEAT: CLI                  ; cấm ngắt
        MOV CX, 0FFFFh       ; số lần giảm CX để chờ trạng thái sẵn sàng
        MOV AL, 01001111B    ; xóa bit 4 và bit 5
        AND ES: [ 0097h], AL ; chắn bit 4 và bit 5, bit 7 của ngăn nhớ ES:[0097h]
WAIT1: IN AL, 64h            ; đọc trạng thái
        TEST AL, 00000010B   ; kiểm tra trạng thái bộ đệm đầy (bit 1 = 1)
        LOOPNZ WAIT1         ; trở về WAIT1 nếu chưa sẵn sàng và CX ≠ 0
        MOV AL, DATA         ; DATA là dữ liệu cần ghi vào bàn phím
        OUT 60h, AL          ; ghi ra cổng 60h
        STI                  ; cho phép ngắt
        MOV CX, 0FFFFh       ; số lần giảm CX để chờ trạng thái sẵn sàng
WAIT2:  TEST BYTE PTR ES: [[0097h]], 00010000B ; kiểm tra bit 4 =1
        JNZ EXIT             ; thoát nếu bit 4 = 0
        LOOP WAIT2           ; trở về WAIT2 nếu CX ≠ 0 và bit 4 ≠ 1
        DEC BL               ; lặp lại BL = 3 lần
        JNZ REPEAT           ; trở về REPEAT
        OR BYTE PTR ES : [[0097h]] ,1000000h  ; xác lập sai số (bit 7) = 1
EXIT:   INT 20h
```

## 5.3. LẬP TRÌNH CHO MÀN HÌNH

Cùng với bàn phím, màn hình là thiết bị đưa tin ra để người sử dụng MVT biết được hoạt động của máy và là thiết bị thông dụng tối thiểu nhất của một hệ MVT.

## 5.3.1. CẤU TRÚC, ĐẶC TÍNH CỦA MÀN HÌNH VÀ BÌA GHÉP NỐI

Chúng ta chỉ xét những đặc điểm cơ bản nhất liên quan tới lập trình cho màn hình và bìa ghép nối (có thể xem chi tiết ở chương 7 tài liệu Kỹ thuật ghép nối máy vi tính của cùng tác giả).

### 1. Màn hình

Màn hình của máy vi tính có cấu trúc và hoạt động như màn hình của một máy thu hình (tivi). Súng điện tử bắn các tia điện tử đập vào màn hình phủ phốtpho tạo nên các điểm sáng. Các cuộn lái tia điều khiển chùm tia điện tử lệch ngang (H) và lệch thẳng đứng (V) làm di chuyển vị trí của tia điện tử. Lưới G điều khiển cường độ tia điện tử cho độ đậm nhạt khác nhau của chấm sáng trên màn hình. Ở chế độ ký tự, tia điện tử được đánh dấu trên màn hình bởi hình mũi tên (con trỏ), còn ở chế độ đồ họa, chỉ là một điểm sáng (pixel). Có hai loại màn hình:

- *Màn hình đơn sắc:* (đen trắng) chỉ có một súng điện tử, cho ảnh sáng (trắng) và tối (đen).

- *Màn hình mầu:* có ba súng điện tử ứng với ba mầu R (đỏ), B (xanh da trời) và G (xanh lá cây). Sự trộn của cường độ khác nhau của ba mầu này cho một mầu tổng hợp trên màn hình phốtpho có các điểm nhậy của ba màu cơ bản trên đặt xen kẽ và mắt nhìn có cảm giác như một điểm có màu tổng hợp.

### 2. Bìa ghép nối màn hình

Việc hiển thị trên màn hình được điều khiển bởi bìa ghép nối màn hình có chứa vi xử lý (ví dụ, VXL 6845 của hãng Motorola cho bìa CGA). Ngoài ra còn có bộ nhớ đệm để nhớ đệm các tin đưa ra từ bộ nhớ trung tâm của MVT theo chế độ DMA (trực tiếp khối nhớ) và các mạch điện tử điều khiển sự quét ngang, quét dọc và cường độ của chùm điện tử.

Tùy chế độ (đen trắng, mầu) và độ phân giải (số pixel ngang x số pixel dọc) ta có các loại bìa ghép nối khác nhau như:

- **MDA**(1981) : (Monochrom Display Adapter - bộ phối ghép màn hình đơn sắc đen - trắng), có độ phân giải cao (25 dòng 80 cột, ma trận ký tự 9x14 điểm), được lắp trong các máy tính PC- IBM ngoại trừ PCjr. Điều khiển bởi vi xử lý MC 6845.

- **CGA**(1981) : (Color Graphic Adapter - bộ phối ghép màn hình đồ họa mầu), có thể hiển thị ở cả hai chế độ văn bản (80x25, 40x25, ma trận ký tự 8x8 điểm) và đồ họa (320x200 với 4 mầu hay 640x200 với 2 mầu). Điều khiển bởi vi xử lý MC 6845.

Hai bìa phối ghép trên được hãng IBM cung cấp cho các máy PC đầu tiên (loại XT).

- **MCGA:** (Multi Color Graphic Array - ma trận đồ họa nhiều mầu) do IBM trang bị cho máy vi tính sau PS/2.

- **HGC:** (1983, Hercules) hoàn toàn tương thích với MDA, CGA nhưng chất lượng cao hơn (2 trang màn hình đồ họa 720x348).

- **EGA:** (1985, Enhanced Graphic Adapter - bộ phối ghép đồ họa cải tiến) do IBM đưa ra, có độ phân giải cao, chế độ văn bản (80x25, 40x25, ma trận ký tự 8x14) và đồ họa có thêm 4 chế độ mở rộng mà các bìa trước không có là 13 tới 16).

- **VGA:** (1987, Video Graphic Array - ma trận đồ họa video) vi mạch điều khiển có mật độ tích hợp cao hơn EGA, gửi tín hiệu mầu tương tự (qua khối biến đổi tương tự - số DAC) nên tạo được nhiều mầu (260.000 ở các chế độ 2, 4, 16). Độ phân giải cao nhất cho 640x480 với 2, 4, 16 mầu tùy chế độ. Chế độ mở rộng (320x200) cho tới 256 mầu đồng thời nên cần 256K tới 512K RAM của bộ nhớ màn hình. Bìa được sử dụng cho thế hệ máy PS/2.

- **Super VGA:** giống VGA nhưng hiển thị điểm nhanh hơn, hiển thị 256 mầu cho 3 chế độ khác nhau (640x200, 640x350, 640x480). Nếu có đủ RAM, có thể dùng cho đồ họa với 800x600 và 1024x768.

- **TMS340/TIGA**: (1986), là bìa do hãng Texas Instruments dựa trên cơ sở vi xử lý họ TMS340 như TMS34010 (1986) và TMS 34020 (1990), có khả năng xử lý đồ họa 32 bit, có thư viện đồ họa của phần mềm phục vụ TIGA (Texas Instruments Graphic Architecture). TIGA vẽ lại màn hình Windows nhanh hơn 5 lần so với VGA thường, nên thay thế cho VGA và Super VGA.

- **XGA**: (1990, Extended Graphic Array - ma trận đồ họa mở rộng), tương thích với các máy từ 80386. XGA-2 (1991) là phiên bản thứ hai có các tính năng sau:

+ Độ phân giải tối đa 1024x768 trong 256 mầu.
+ Điều khiển bởi bộ ghép nối 8514/A.
+ Cần 1024KB RAM.
+ Tương thích với chuẩn VGA.
+ Có thể biểu diễn theo không gian ba chiều, theo ánh xạ bit.
+ Quản lý ảnh trong môi trường đa nhiệm.

## 3. Các chế độ của màn hình

Trước khi cho màn hình hoạt động, ta phải xác định chế độ cho màn hình (văn bản, đồ họa, đen trắng, mầu, độ phân giải) bằng viết chương trình (trình bày ở dưới).

Ta có bảng 5.6 mô tả các chế độ màn hình văn bản ứng với các bộ ghép nối khác nhau.

***Bảng 5.6.* Các chế độ văn bản của màn hình**

| Chế độ | Đặc tính | Các bộ ghép nối | Địa chỉ bộ đệm | Kích thước | Cỡ chữ |
|---|---|---|---|---|---|
| 0 | 40x25 văn bản 16 mầu (xám) | CGA, EGA, MCGA, VGA | B8000h | 2KB | 8x8 |
| 1 | 40x25 văn bản 16 mầu | CGA, EGA, MCGA, VGA | B8000h | 2KB | 8x8 |
| 2 | 80x25 văn bản 16 mầu (xám) | CGA, EGA, MCGA, VGA | B8000h | 2KB | 8x8 |
| 3 | 80x25 văn bản 16 mầu | CGA, EGA, MCGA, VGA | B8000h | 4KB | 8x8 |
| 7 | 80x25 văn bản đơn sắc | MDA, EGA, VGA | B0000h | 4KB | 9x14 |

*Ghi chú*: Các chế độ 0 và 2 tắt tín hiệu bật mầu với các màn hình tổng hợp, các màn hình RGB vẫn hiển thị 16 mầu.

Ký hiệu 80x25 chỉ có 80 ký tự/1 dòng và cả màn hình có 25 dòng.

Ở chế độ đồ họa (graphic) tức biểu diễn từng điểm (pixel), ta có bảng chế độ như bảng 5.7 cho các bìa phối ghép khác nhau.

Tích số trên phản ánh độ phân giải của màn hình với số điểm của hàng (ngang) và cột (dọc). Chỉ số chế độ chính là số (dạng hex) sẽ phải ghi vào thanh ghi đệm, xác định chế độ (xét ở dưới). Khi viết chương trình, tùy loại bìa phối ghép và tùy yêu cầu về độ phân giải để chọn chỉ số chế độ thích hợp.

Như vậy, với độ phân giải 80x25 cần 2000 vị trí của màn hình, chứa 2000 ký tự và cần 4000 byte của bộ nhớ RAM để nhớ đệm. Địa chỉ của bộ nhớ RAM màn hình nằm ở địa chỉ B000:0000 tới B800:0000. Tùy loại bìa, dung lượng của bộ nhớ đệm trên có thể nhớ nội dung cho các trang khác nhau (xem bảng 5.7).

*Bảng 5.7.* **Các chế độ đồ họa của màn hình**

| Chỉ số chế độ | Đặc tính | Loại bìa | Địa chỉ | Kích thước | Cỡ chữ |
|---|---|---|---|---|---|
| 4 | 320x200 pixel, 4 mầu | CGA, EGA, VGA, MCGA | B8000 | 8KB | 8X8 |
| 5 | 320x200 pixel, 4 mầu (xám) | CGA, EGA, VGA, MCGA | B8000 | 8KB | 8X8 |
| 6 | 320x200 pixel, 4 mầu | CGA, EGA, VGA, MCGA | B8000 | 16KB | 8X8 |
| 8 | 160x200 pixel, 4 mầu | PCjr | B8000 | 16KB | 8X8 |
| 9 | 320x200 pixel, 4 mầu | PCjr | B8000 | 32KB | 8X8 |
| A | 320x200 pixel, 4 mầu | PCjr | B8000 | 32KB | 8X8 |
| D | 320x200 pixel, 16 mầu | EGA, VGA | A8000 | 8KB | 8X8 |
| E | 640x200 pixel, 16 mầu | EGA, VGA | A8000 | 16KB | 8X8 |
| F | 640x350 pixel, đơn sắc | EGA, VGA | A8000 | 28KB | 8X14 |
| 10 | 640x350 pixel, 16 mầu | EGA, VGA | A8000 | 28KB | 8X14 |
| 11 | 640x480 pixel, 2 mầu | VGA, MCGA | A8000 | 38KB | 8X16 |
| 12 | 640x480 pixel, 16 mầu | VGA | A8000 | 38KB | 8X16 |
| 13 | 320x200 pixel, 256 mầu | VGA, MCGA | A8000 | 38KB | 8X8 |

Bộ ký tự được tiêu chuẩn hóa, bộ có 256 ký tự chuẩn của IBM là giống nhau cho tất cả các loại bìa màn hình, còn byte thuộc tính chỉ đơn sắc hay mầu khác nhau ứng với mầu khác nhau cần biểu diễn.

## 4. Lập trình cho màn hình

Nhiệm vụ của lập trình cho màn hình là viết chương trình (bằng hợp ngữ) cho hoạt động của màn hình. Đó là :

- Xác định hay đọc loại bìa hoặc chế độ (văn bản, đồ họa) với độ phân giải, cỡ chữ (số dòng, số cột) của màn hình theo các bảng 5.6 và 5.7.

- Xác định số hiệu của trang màn hình.

- Xác định vị trí và kích thước của con trỏ (chế độ văn bản) và của ảnh điểm (chế độ đồ họa).

- Xác định kích thước của cửa sổ màn hình.

- Xác định mầu viền của màn hình (chế độ văn bản), mầu nền của màn hình (đồ họa) hay mầu nền, mầu ký tự hay ảnh điểm.

- Đưa ra màn hình hay đọc vào từ màn hình các ký tự hay các ảnh điểm với kích thước và mầu khác nhau của nền (phông), ký tự hay ảnh điểm.

Cũng như bàn phím hay thiết bị ngoài khác, có hai phương pháp lập trình:

- *Phương pháp phần mềm :* sử dụng các ngắt của MS-DOS (INT 10h và INT 21h với chức năng 02h). Phương pháp này tận dụng được các chương trình con viết sẵn của hệ điều hành MS-DOS nên người lập trình chỉ cần:

+ Biết rõ công dụng của các chức năng của ngắt INT 10h.

+ Ghi các dữ liệu tức các thông số vào của chức năng cho các thanh ghi mà chức năng sử dụng.

+ Gọi ngắt tương ứng với nhiệm vụ của lập trình.

+ Đọc thông số ra ở các thanh ghi quy định cho chức năng tương ứng để biết thông số của màn hình sau khi thực hiện chức năng đã gọi.

- *Phương pháp phần cứng:* ghi các lệnh điều khiển trực tiếp bộ điều khiển màn hình (ví dụ, MC-6845 cho CGA) và bộ nhớ đệm của màn hình. Phương pháp này phức tạp hơn, đòi hỏi người lập trình hiểu rõ phần cứng và kỹ thuật lập trình bằng hợp ngữ.

Dưới đây, chúng ta sẽ lần lượt xét các phương pháp lập trình trên.

## 5.3.2. NGẮT VÀ CÁC CHỨC NĂNG CỦA MÀN HÌNH

Cũng như bàn phím, màn hình cũng có các ngắt của BIOS và DOS phục vụ màn hình.

### 1. Ngắt của BIOS

BIOS phục vụ màn hình bởi ngắt INT 10h với 23 hàm chức năng (từ 00h tới 1Ch) như bảng 5.8.

***Bảng 5.8.* Các chức năng của INT 10h**

| Chức năng (phục vụ) | Hành động |
|---|---|
| 00h | Xác định chế độ màn hình |
| 01h | Xác định kích thước con trỏ |
| 02h | Xác định vị trí con trỏ |
| 03h | Đọc vị trí con trỏ |
| 04h | Đọc vị trí dấu chấm sáng |
| 05h | Xác định trạng thái hoạt động |
| 06h | Kéo dài cửa sổ lên cao |
| 07h | Kéo dài (cuốn) cửa sổ xuống thấp |
| 08h | Đọc ký tự và thuộc tính |
| 09h | Ghi ký tự và thuộc tính |
| 0Ah | Ghi ký tự |
| 0Bh | Xác định bảng 4 mầu |
| 0Ch | Ghi điểm ảnh (pixel) |
| 0Dh | Đọc điểm ảnh (pixel) |
| 0Eh | Ghi ký tự ở chế độ teletype |
| 0Fh | Đọc chế độ màn hình hiện hành |
| 10h | Giao diện bảng mầu (thanh ghi mầu) |
| 11h | Giao diện máy phát ký tự |
| 12h | Lựa chọn tuần tự |
| 13h | Ghi chuỗi ký tự |
| 1Ah | Ghi/đọc những mã của tổ hiển thị |
| 1Bh | Lấy lại tin về trạng thái hoạt động |
| 1Ch | Giữ và tìm lại trạng thái màn hình |

Có thể chia các ngắt trên thành các nhóm: chế độ-trạng thái, trang, mầu, kích thước cửa sổ và ghi - đọc ký tự hay điểm ảnh. Dưới đây chúng ta sẽ trình bầy chi tiết và sử dụng các chức năng trên cho lập trình ở chế độ văn bản và đồ họa.

### 2. Ngắt của DOS cho màn hình

Ngắt INT 21h của DOS chỉ có một chức năng in ra màn hình, đó là chức năng 02h mà ta đã sử dụng trong lập trình với bàn phím để kiểm tra kết quả lập trình bàn phím.

- Thông số vào: AH = 02h

DL = mã ASCII của ký tự (khác với ngắt INT 10h cho màn hình của BIOS là mã ASCII ở trong thanh ghi AL).

- Thông số ra : không có.

## 5.3.3. LẬP TRÌNH CHUNG CHO MÀN HÌNH

Lập trình cho màn hình là viết chương trình cho màn hình để xác định:

- Chế độ màn hình : theo loại văn bản hay đồ họa, loại bìa, độ phân giải.
- Trang của màn hình theo bảng 5.9.
- Vị trí và kích thước của con trỏ.
- Kích thước của cửa sổ.
- Mầu sắc của nền, của ký tự hay của ảnh điểm.
- Ghi hay đọc ký tự hay ảnh điểm ra màn hình.

***Bảng 5.9.*** **Số trang cho các bìa**

| Chế độ | Số trang lớn nhất | | |
|---|---|---|---|
| | CGA | EGA | VGA |
| 0 ÷ 1 | 8 | 8 | 8 |
| 2 ÷ 3 | 4 | 8 | 8 |
| 7 | Không dùng | 8 | 8 |

Các nhiệm vụ xác định chế độ, xác định trang, vị trí và kích thước con trỏ là giống nhau cho cả hai chế độ, còn các nhiệm vụ khác sẽ khác nhau cho từng chế độ.

### 1. Xác định chế độ - trạng thái màn hình

Nhiệm vụ đầu tiên của lập trình màn hình là xác định chế độ hay trạng thái màn hình. Có thể:

- Xác định chế độ mới khi biết loại bìa màn hình và chế độ (văn bản, đồ họa) cần sử dụng.
- Đọc chế độ màn hình đang sử dụng.

Với chế độ văn bản ta có bảng chế độ như bảng 5.6.

***a) Ghi chế độ hiển thị:*** chức năng **AH = 00h**

- Thông số vào : AH = 00h
  AL = số hiệu chế độ (bảng 5.6 và 5.7).
- Thông số ra : không có.

Theo bảng này ta có thể chọn :

- Theo mầu của màn hình với đen trắng (chế độ 7) hay mầu (chế độ 0, 1, 2, 3).
- Theo độ phân giải hay kích thước ký tự với 40x25 (chế độ 0, 1) hay 80x25 (chế độ 2, 3).

*Ví dụ 1:* Xác lập chế độ văn bản (25 dòng, 80 cột), 16 mầu xám, dùng bìa CGA.

```
MOV AH, 00h      ; có thể thay bằng lệnh XOR AH, AH
MOV AL, 02h      ; chế độ 02h cho bìa CGA, 25 dòng, 80 cột, 16 mầu xám
INT 10h          ; gọi ngắt phục vụ màn hình
INT 20h          ; kết thúc chương trình.
```

Chúng ta có thể cho chương trình này chạy trong môi trường DEBUG để thấy tác dụng của việc đổi chế độ. Muốn thay chỉ số chế độ, ta chỉ cần thay giá trị của AL tương ứng bằng dùng lệnh E0103 ↵ để giá trị cũ của AL hiện ra, rồi gõ giá trị mới vào và ấn ↵ để nhập nó vào chương trình. Cứ như vậy, ta có thể thử tất cả các chế độ cho chế độ văn bản.

***b) Đọc chế độ màn hình hiện hành*** : chức năng **AH = Fh**

- Thông số vào : AH = Fh.
- Thông số ra : AL = số ký tự trên một dòng
  BH = trang hoạt động (bảng 5.9).

***c) Lựa chọn chế độ thích hợp cho loại bìa màn hình***: chức năng **AH = 1Ah**

Chức năng này trả về một mã hai byte, chỉ tổ hợp của màn hình và hiển thị trong MVT.

Chức năng này cho phép lựa chọn một trong hai phục vụ nhỏ bằng cách dùng giá trị trong thanh ghi AL = 00h hay 01h.

- *Phục vụ nhỏ 00h (0)*: ghi trở về mã của hai byte trong thanh ghi BX. Nếu máy vi tính có hai chế độ màn hình nhỏ, giá trị trong BL chỉ rằng nó là tích cực (activ); điều đó muốn nói rằng BIOS đang dùng nó. Nếu chỉ có một màn hình nhỏ, giá trị trong BH = 0.

- *Phục vụ nhỏ 01h (1)*: tác động nhiệm vụ ngược lại. Nó cho phép thay đổi mã của tổ hợp hiện hành. Chỉ dùng phục vụ nhỏ này khi ta biết chính xác cần phải làm gì.

Các mã của tổ hợp hiển thị cho BIOS 1Ah như ở bảng 5.10.

***Bảng 5.10.*** **Các mã cho 1Ah**

| Mã | Bìa màn hình |
|---|---|
| 00h | (Không dùng một bìa nào) |
| 01h | MDA |
| 02h | CGA |
| 03h | Để dành |
| 04h | EGA với hiển thị mầu |
| 05h | EGA với hiển thị đơn sắc |
| 06h | Khối điều khiển đồ họa chuyên dùng |
| 07h | VGA đơn sắc |
| 08h | VGA hiển thị mầu |
| 09h, 0Ah | Để dành |
| 0Bh | MCGA với hiển thị đơn sắc |
| 0Ch | MCGA với hiển thị mầu |
| 0FFh | Chưa sử dụng |

***d) Giữ lại tin về trạng thái màn hình***: chức năng **1Bh**

Chức năng này sử dụng cho các bìa EGA/VGA hay máy vi tính PS/2.

- Thông số vào: AH = 1Bh
  ES = địa chỉ bắt đầu của mảng nhớ của bộ đệm 64KB
  DI = địa chỉ offset của bộ đệm 64KB.
- Thông số ra : AL = 1Bh.

BIOS lấp đầy bộ đệm 64 octet (địa chỉ ở cặp thanh ghi ES:DI) những tin về chế độ màn hình hiện tại (chỉ số chế độ, dòng và cột của ký tự, số mầu) cũng như về cấu hình vật lý (nhớ màn hình tổng cộng, mã tổ hợp của hiển thị v.v...).

*e) Bảo vệ và tìm trạng thái màn hình* : chức năng **1Ch**

Chỉ dùng cho bìa VGA của PS/2. Phục vụ này cho phép giữ mọi tin của màn hình về trạng thái của DAC, vùng số liệu trong RAM hay những giá trị hiện hành của mọi thanh ghi điều khiển màn hình. Có ba chế độ phục vụ nhỏ được ghi vào AL:

- *Chế độ 00h (0)*: xem loại tin nào muốn đọc, bằng cách đặt một hoặc hơn ba bit thấp của giá trị vào CX. BIOS quay trở lại kích thước của bộ đệm mà ta cần để tích lũy tin trong thanh ghi BX.

- *Chế độ 01h (1)*: giữ tin về trạng thái màn hình hiện tại trong bộ đệm mà địa chỉ ghi ở BX. Rồi ta có thể thay đổi chế độ màn hình, đổi mầu, hay đổi thông số khác về phần cứng của màn hình.

- *Chế độ 02h (2)*: cho phép thu lại trạng thái cũ.

## 2. Xác định trang màn hình

Vì bìa màn hình khác nhau, độ phân giải khác nhau, dung lượng bộ nhớ đệm tổng cộng như nhau (xem bảng 5.6 và 5.7) nên số trang màn hình khác nhau (xem bảng 5.9).

Do đó, khi sử dụng, ta phải lập trình để xác định trang nào đưa vào sử dụng. Ta dùng chức năng **AH = 05h** với :

- Thông số vào: AH = 05h
  AL = số trang (xem bảng 5.9, riêng CGA chỉ có 0÷3 trang).
- Thông số ra: không.

*Ví dụ 2:* Xác định chế độ văn bản 40x25, 16 mầu, trang 1 cho bìa VGA.

```
MOV AH, 00h      ; xác định chế độ màn hình
MOV AL, 01h      ; chế độ 01h
INT 10h          ; xác lập chế độ
MOV AH, 05h      ; xác lập trang màn hình
MOV AL, 01h      ; trang số 01h
INT 10h          ; xác lập trang
INT 20h          ; kết thúc chương trình
```

Ngoài việc xác định trang màn hình một cách riêng rẽ, ta còn xác định trang của màn hình ở thanh ghi BH trong các chức năng riêng rẽ, ta còn xác định trang của màn hình ở thanh ghi BH trong các chức năng khác như xác định vị trí con trỏ, ghi ký tự ra màn hình.v.v...

## 3. Xác định vị trí, kích thước con trỏ

*a) Ghi vị trí con trỏ (dịch chuyển con trỏ)*: chức năng **AH = 02h**

- Thông số vào : AH = 02h
  BH = số hiệu trang
  DH = dòng mới (0÷24) nhưng ghi vào dưới dạng hexa
  DL = cột mới (0÷79) trong chế độ văn bản 80x25, (0÷39) trong chế độ văn bản 40x25, nhưng ghi vào dưới dạng hexa.
- Thông số ra : không có.

*b) Đọc vị trí và kích thước con trỏ hiện tại:* chức năng **AH = 03h**

- Thông số vào : AH = 03h
  BH = số hiệu trang.
- Thông số ra : DH = dòng
  DL = cột
  CH = dòng quét đầu
  CL = dòng quét cuối.

Các giá trị của các thanh ghi trên cũng được ghi dưới dạng hexa và cần chú ý tới giá trị tới hạn (40 hay 80 cột và 25 dòng), nhưng ghi vào dưới dạng hexa.

*c) Thay đổi kích thước con trỏ:* chức năng **AH = 01h**

- Thông số vào: AH = 01h
  CH = dòng quét đầu
  CL = dòng quét cuối.
- Thông số ra: không có.

## 4. Xác định kích thước của cửa sổ màn hình

Người ta thay đổi kích thước cửa sổ của màn hình so với vị trí của mốc bằng cách :

- Thay đổi kích thước phía trên, tức cuốn màn hình lên, dùng chức năng AH = **06h,** khi đó lấy góc dưới (giá trị dòng, cột tại thanh ghi DX) làm mốc, còn AL chứa số dòng cuốn, tức số dòng sáng của cửa sổ.

- Thay đổi kích thước phía dưới, tức cuốn màn hình xuống, dùng chức năng **AH = 07h,** khi đó lấy góc trên (giá trị dòng, cột tại thanh ghi CX) làm mốc, còn AL chứa số dòng cuốn, tức số dòng sáng của cửa sổ.

*a) Cuốn màn hình lên* : chức năng **06h**

- Thông số vào: AH = 06h
  AL = số dòng cuốn (AL = 0 có nghĩa cuốn cả màn hình hay cửa sổ)
  BH = thuộc tính của các dòng trống
  CH, CL = dòng, cột góc trái trên của cửa sổ
  DH, DL = dòng, cột góc phải dưới của cửa sổ.
- Thông số ra: không có.

Vì là góc phải dưới (DX) của cửa sổ nên giá trị dòng và cột của nó phải lớn hơn các giá trị của góc trái trên (CX) của cửa sổ.

*b) Cuốn màn hình xuống*: chức năng **07h**

- Thông số vào: AH = 07h
  AL = số dòng cuốn
  BH = thuộc tính của các dòng trống
  CH, CL = dòng, cột của góc trái phía trên của cửa sổ
  DH, DL = dòng, cột của góc phải phía dưới của cửa sổ.
- Thông số ra: không có.

Cuốn màn hình hay cửa sổ xuống một dòng có nghĩa là mỗi dòng được chuyển xuống một hàng và một dòng trống được điền vào dòng trên cùng, dòng cuối cùng biến mất.

*Ví dụ 3*: Xác định cửa sổ có tọa độ góc trái trên 0527h, tọa độ góc phải dưới 1527h cho màn hình có chế độ 25x80.

```
MOV AH, 00h       ; hàm xác định chế độ của màn hình
MOV AL, 02h       ; chế độ 02h (văn bản, 25x80)
INT 10h           ; gọi ngắt phục vụ màn hình
MOV AH, 06h       ; hàm cuốn màn hình lên
MOV AL, 12h       ; số dòng cuốn = 1x16 + 2 = 18
MOV CX, 0527h     ; góc trên có dòng = 05, cột = 2x16 + 7 = 39
MOV DX, 1547h     ; góc dưới có dòng = 1x16 + 5 = 21,  cột 4x16 + 7 = 71
MOV BH, 42h       ; nền đỏ (4), ký tự xanh lá cây
INT 10h           ; gọi hàm cuốn cửa sổ màn hình.
INT 20h
```

Chúng ta có thể cho chương trình trên chạy trong môi trường DEBUG và thay đổi các thông số của các tọa độ, của số dòng cuốn, đặc biệt, có thể cho một mầu nền mong muốn hay xóa đen cả màn hình với tọa độ cả màn hình (CX=0000h, DX=184Fh) và BH=07h.

## 5.3.4. LẬP TRÌNH Ở CHẾ ĐỘ VĂN BẢN

Ở chế độ này, trên màn hình ghi các ký tự theo từng ma trận (8x8, 9x14 hay 8x16) ảnh điểm, cả màn hình có 80 hoặc 40 dòng và 25 cột ký tự. Mầu sắc của nền, mầu viền của màn hình cũng như mầu của ký tự tùy thông số vào ghi ở các thanh ghi thông dụng trước khi gọi ngắt INT 10h của BIOS hay INT 21h của DOS.

Đối với màn hình ở chế độ văn bản, nhiệm vụ lập trình cũng phải theo các nhiệm vụ như đã trình bầy ở trên, trước tiên phải xác định chế độ và các nhiệm vụ khác. Ở đây, còn thêm nhiệm vụ ghi và đọc ký tự ra màn hình, thay đổi cỡ ký tự.

### 1. Ghi chế độ hiển thị : chức năng AH = 00h

- Thông số vào : AH = 00h
  AL = số hiệu chế độ (bảng 5.6).
- Thông số ra: không có.

Theo bảng này ta có thể chọn :

- Theo mầu của màn hình với đen trắng (chế độ 7) hay mầu (chế độ 0, 1, 2, 3).
- Theo độ phân giải hay kích thước ký tự với 40x25 (chế độ 0, 1) hay 80x25 (chế độ 2, 3).

*Ví dụ 4:* Viết chương trình đổi chế độ màn hình như sau:

- Ấn phím CTRL-$F_1$ (mã 5Eh) chuyển sang chế độ 40x25 văn bản, 16 mầu xám (chế độ 0).
- Ấn phím CTRL-$F_2$ (mã 5Fh) chuyển sang chế độ 40x25 văn bản, 16 mầu (chế độ 1).
- Ấn phím CTRL-$F_3$ (mã 60h) chuyển sang chế độ 80x25 văn bản, 16 mầu xám (chế độ 2).
- Ấn phím CTRL-$F_4$ (mã 61h) chuyển sang chế độ 80x25 văn bản, 16 mầu (chế độ 3).
- Ấn phím CTRL-$F_5$ (mã 62h) chuyển sang chế độ 80x25 văn bản, đen trắng (chế độ 7).
- Ấn phím Esc (mã 001Bh) thoát khỏi chương trình.

```
xxxx: 0100  XOR AX, AX      ; xóa AX về 0
xxxx: 0102  XOR BX, BX      ; xóa BX về 0
xxxx: 0104  XOR DX, DX      ; xóa DX về 0
```

```
xxxx: 0106 MOV AH, 00h     ; hàm nhập ký tự từ bàn phím
xxxx: 0108  INT 16h        ; gọi ngắt phục vụ bàn phím
xxxx: 010A CMP AH, 5Eh     ; xem mã quét có phải của CTRL_F1 không ?
xxxx: 010D  JZ 128         ; nếu đúng, nhảy tới địa chỉ IP = 128 để đổi chế độ
xxxx: 010F  CMP AH, 5Fh    ; xem mã quét có phải của CTRL_F2 không ?
xxxx: 0112  JZ 12E         ; nếu đúng, nhảy tới địa chỉ IP = 12E để đổi chế độ
xxxx: 0114 CMP AH, 60h     ; xem mã quét có phải của CTRL_F3 không ?
xxxx: 0117  JZ 135         ; nếu đúng, nhảy tới địa chỉ IP = 135 để đổi chế độ
xxxx: 0119  CMP AH, 61h    ; xem mã quét có phải của CTRL_F4 không ?
xxxx: 011C  JZ 13C         ; nếu đúng, nhảy tới địa chỉ IP = 13C để đổi chế độ
xxxx: 011E  CMP AH, 62h    ; xem mã quét có phải của CTRL_F5 không ?
xxxx: 0121  JZ 143         ; nếu đúng, nhảy tới địa chỉ IP = 143 để đổi chế độ
xxxx: 0123 CMP AL, 1Bh     ; xem có phải mã ASCII của phím Esc không ?
xxxx: 0126  JZ 148         ; nếu đúng, nhảy tới địa chỉ IP = 148 để thoát
xxxx: 0128  MOV AL, 07h    ; nạp chế độ 7
xxxx: 012A CALL 14A        ; gọi chương trình con đổi chế độ màn hình
xxxx: 012C  JMP 148        ; kết thúc
xxxx: 012E  MOV AL, 00h    ; nạp chế độ 0
xxxx: 0130  CALL 14A       ; gọi chương trình con đổi chế độ màn hình
xxxx: 0133  JMP 148        ; kết thúc
xxxx: 0135  MOV AL, 01h    ; nạp chế độ 1
xxxx: 0137  CALL 14A       ; gọi chương trình con đổi chế độ màn hình
xxxx: 013A  JMP 148        ; kết thúc
xxxx: 013C  MOV AL, 02h    ; nạp chế độ 2
xxxx: 013E  CALL 14A       ; gọi chương trình con đổi chế độ màn hình
xxxx: 0141  JMP 148        ; kết thúc
xxxx: 0143  MOV AL, 03h    ; nạp chế độ 3
xxxx: 0145  CALL 14A       ; gọi chương trình con đổi chế độ màn hình
xxxx: 0148  INT 20h        ; kết thúc chương trình
xxxx: 014A MOV AH, 00h     ; thay đổi chế độ
xxxx: 014C  INT 10h        ; gọi ngắt thay đổi chế độ
xxxx: 014E  RET            ; trở về từ chương trình con.
```

## 2. Xác định mầu cho bìa CGA

### *a) Byte thuộc tính của ký tự*

Như đã giới thiệu ở phần trên, trong màn hình mầu, mầu sắc của một ảnh điểm do tổ hợp của cường độ (1- có, 0- tắt) của 3 súng bắn điện tử đỏ (R), xanh da trời (B) và xanh lá cây (G) quyết định. Để điều khiển mầu qua 3 súng điện tử này ta phải lập trình để ghi vào khối điều khiển màn hình. Một mầu bất kỳ là tổ hợp của 3 mầu trên (bảng 5.11).

Mỗi một ký tự trong màn hình mầu được đặc trưng bởi 2 byte :

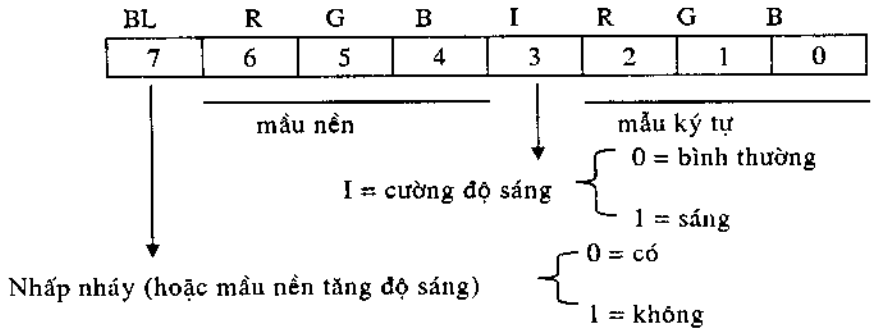
- Byte mã ASCII của ký tự.
- Byte thuộc tính chỉ mầu.

Với màn hình đơn sắc MDA hay mầu CGA, cấu trúc byte thuộc tính có các bit như sau :

- Các bit 6, 2 ứng với mầu đỏ (R-Red)

- Các bit 5, 1 ứng với mầu xanh lá cây (G-Green)
- Các bit 4, 0 ứng với mầu xanh lá da trời (B-Blue)

và ta có tổ hợp RGB như sau :




+ *Với ký tự đen trắng:* byte thuộc tính như sau:

| *Các bit* | 7 | 6 | 5 | 4 | 3 (I) | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Không có ký tự (đen/nền đen) |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Chữ trắng, gạch dưới trên nền đen |
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Chữ trắng trên nền đen |
| | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Chữ đen trên nền trắng |
| | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | Không ký tự (trắng, nền trắng) |

+ *Với ký tự mầu:* các bit 6, 2 ứng với mầu đỏ (Red)
các bit 5, 1 ứng với mầu xanh lá cây (Green)
các bit 4, 0 ứng với mầu xanh da trời (Blue)

Tổ hợp 4 bit IRGB ta có bảng trộn mầu (bảng 5.11).

Với mầu nền, ta cũng có các bit RGB tương tự như mầu của ký tự nhưng chỉ có mầu thường (I=0) mà không có mầu sáng (I=1) nhưng bit 7 = 1 có sự nhấp nháy của mầu và ký tự.

Ở chế độ văn bản, ngoại trừ chế độ 7 cho màn hình đen trắng (bìa MDA), các chế độ khác (0, 1, 2, 3), đối với bìa CGA, ta xác định mầu viền, mầu nền của màn hình và mầu của ký tự theo bảng 5.11. Nói chung, mầu được xác định do byte thuộc tính BL như bảng 5.11, nhưng đối với cửa sổ byte thuộc tính lại là BH và cũng như BL trong các trường hợp khác, 4 bit cao là mầu nền, còn 4 bit thấp là mầu ký tự. Mầu của cửa sổ, ta sẽ xét ở phần cuốn cửa sổ.

***Bảng 5.11.* Bảng mầu theo các bit IRGB của ký tự của bìa CGA**

| Số hex | Các bit thuộc tính | | | | Mầu |
|---|---|---|---|---|---|
| | I | R | G | B | |
| 0 | 0 | 0 | 0 | 0 | Đen |
| 1 | 0 | 0 | 0 | 1 | Xanh da trời |
| 2 | 0 | 0 | 1 | 0 | Xanh lá cây |
| 3 | 0 | 0 | 1 | 1 | Xanh cô ban |
| 4 | 0 | 1 | 0 | 0 | Đỏ |
| 5 | 0 | 1 | 0 | 1 | Hồng |
| 6 | 0 | 1 | 1 | 0 | Vàng tối |
| 7 | 0 | 1 | 1 | 1 | Trắng xám |
| 8 | 1 | 0 | 0 | 0 | Xám |
| 9 | 1 | 0 | 0 | 1 | Xanh da trời sáng |
| Ah | 1 | 0 | 1 | 0 | Xanh lá cây sáng |
| Bh | 1 | 0 | 1 | 1 | Xanh ngọc sáng |
| Ch | 1 | 1 | 0 | 0 | Đỏ sáng |
| Dh | 1 | 1 | 0 | 1 | Hồng sáng |
| Eh | 1 | 1 | 1 | 0 | Vàng sáng |
| Fh | 1 | 1 | 1 | 1 | Trắng sáng |

*b) Xác định mầu của nền và ký tự* : chức năng **09h**

Số hiệu mầu của ký tự theo bảng 5.5 được nạp vào thanh ghi BL trong chức năng ghi ký tự ra màn hình AH = 09h với thuộc tính mong muốn, trong đó 4 bit cao là mầu nền còn 4 bit thấp là mầu của ký tự. Ta sẽ lấy ví dụ ở phần dưới cho việc ghi ký tự. Ta sẽ xét ví dụ về mầu nền và mầu của ký tự cùng với ghi ký tự ra màn hình.

*c) Xác định mầu viền của màn hình:* chức năng **0Bh**

– Thông số vào : AH = 0Bh
BH = 00h (hàm con chỉ mầu viền như bảng mầu 5.5)
BL = chỉ số mầu (0÷15 của bảng mầu 5.5).

– Thông số ra: không có.

*Ví dụ 5:* Viết chương trình chọn bảng 1 và mầu viền xanh nhạt (có số hiệu mầu là 9h).

```
MOV AH, 0Bh      ; chức năng chọn mầu viền
MOV BL, 04h      ; mầu đỏ
INT 10h          ; gọi ngắt 10h
INT 20h          ; kết thúc chương trình.
```

## 3. Xác định mầu của màn hình EGA/VGA

Hàm 10h của ngắt 10h đã đưa vào EGA/VGA và được cải tiến trong BIOS của PS/2. Nó gồm tập hợp nhiều phục vụ con (bảng 5.12) cho phép ta điều khiển bảng mầu, nhấp nháy và DAC trên các màn hình MCGA và VGA. Những phục vụ con khác nhau được dùng cho bìa phối ghép khác nhau, mà trước khi dùng chúng trong chương trình, chúng ta phải biết bìa loại nào (dùng hàm 1Ah).

Chức năng AH = 10h trên có nhiều hàm con với nhiều thông số cho các thanh ghi thông dụng CX, BX, DX. Vì khuôn khổ của tài liệu, chúng ta không xét tất cả các chức năng con, mà chỉ xét ví dụ hai hàm con AL=0 và AL = 10h.

Khác với bìa CGA, trong hai bìa EGA và VGA, thanh ghi thuộc tính (mầu) cho ký tự có 6 bit GBRgbr. Với hàm con AL=0, ký tự có thể biểu thị 16 mầu (sáng với GBR=1) và tối với GBR=0).

***Bảng 5.12.* Các phục vụ con cho AH=10h**

| Phục vụ con AL | Mô tả | Loại bìa |
|---|---|---|
| 00h | Đặt thanh ghi bảng mầu | EGA/VGA |
| 01h | Đặt mầu viền | EGA/VGA |
| 02h | Đặt 16 mầu của bảng thêm mầu viền | EGA/VGA |
| 03h | Đặt cường độ mầu hay thuộc tính nhấp nháy | EGA/VGA |
| 07h | Đọc một thanh ghi bảng mầu của bộ điều khiển thuộc tính | EGA |
| 08h | Đọc một thanh ghi mầu viền | VGA |
| 09h | Đọc 16 thanh ghi mầu và mầu viền | VGA |
| 10h | Đặt một thanh ghi mầu DAC | VGA |
| 12h | Đặt nhiều thanh ghi mầu DAC | VGA |
| 13h | Đặt phương pháp chọn mầu hoặc chọn nhóm thanh ghi DAC | VGA |
| 15h | Đọc một thanh ghi DAC | VGA |
| 17h | Đọc một nhóm thanh ghi của một DAC màn hình | VGA |
| 18H | Đặt thanh ghi che | VGA |
| 19h | Đọc thanh ghi che | VGA |
| 1Ah | Đọc trạng thái trang mầu DAC | VGA |
| 1Bh | Đảo ngược nhóm thanh ghi mầu DAC | VGA |

Với hàm con AL=10h, bìa VGA với sự hỗ trợ của DAC, có khả năng biểu diễn 256 mầu với:

- Thông số vào : AH = 10h
  AL = 10h
  BX = giá trị cho mầu xanh lá cây (G)
  CL = giá trị cho mầu xanh da trời (B)
  DL = giá trị cho mầu đỏ (R).
- Thông số ra : không có.

Đây cũng là sự trộn của 3 mầu cơ bản R, G, B theo chế độ DAC (trộn số thành tuyến tính bằng khối biến đổi số - tương tự), mỗi thanh ghi mầu cơ bản có 16 mầu và mầu tổng hợp có 256 mầu.

*Ví dụ 6:* Đưa trị số mầu 30 cho mầu đỏ (R), 20 cho xanh lá cây (G) và 10 cho xanh da trời (B).

```
MOV AH, 10h        ; chức năng đặt thanh ghi mầu
MOV AL, 10h        ; dịch vụ con trộn mầu DAC
MOV BX, 0005h      ; thanh ghi mầu số 5 theo các giá trị của các thanh ghi cơ bản
MOV DH, 30h        ; giá trị mầu đỏ (R)
MOV CH, 20h        ; trị số cho mầu xanh lá cây (G)
MOV CL, 10h        ; trị số cho mầu xanh da trời (B)
INT 10h            ; gọi ngắt trộn mầu.
```

Các hàm con khác được sử dụng để lập trình trực tiếp cho màn hình EGA/VGA và MVGA.

## 4. Đọc/ghi ký tự với ngắt INT 10h

***a) Đọc ký tự tại vị trí con trỏ*** : chức năng **AH = 08h**

- Thông số vào : AH = 08h
  BH = số hiệu trang.
- Thông số ra: AH = thuộc tính ký tự
  AL = mã ASCII của ký tự.

***b) Ghi (hiển thị) ký tự mầu (thuộc tính) tại vị trí con trỏ*** : chức năng **09h**

Ghi ký tự ra màn hình tại vị trí con trỏ với mầu (thuộc tính) mong muốn.

- Thông số vào : AH = 9h
  BH = số hiệu trang
  AL = mã ASCII của ký tự
  CX = số lần viết ký tự kể từ vị trí con trỏ
  BL = thuộc tính của ký tự (bảng 5.5).
- Thông số ra : không có.

Khác với chức năng 02h của INT 21, con trỏ không được chuyển đến vị trí tiếp theo sau khi hiển thị ký tự và trước khi ghi ký tự, phải xác định vị trí con trỏ bằng chức năng AH=02h.

*Ví dụ 7:* Đổi thuộc tính của ký tự tại vị trí của con trỏ thành thuộc tính đảo cho màn hình đơn sắc.

```
MOV AH, 08h        ; đọc ký tự
XOR BH, BH         ; BH = 0 trang 0
```

```
INT 10 h          ; gọi ngắt
MOV AH, 09h       ; hiển thị ký tự với thuộc tính
MOV CX, 0001h     ; hiển thị 1 ký tự
MOV BL, 70h       ; thuộc tính đảo màn hình
INT 10 h          ; hiển thị ký tự
INT 20h           ; kết thúc chương trình.
```

*c) Ghi ký tự với thuộc tính có sẵn* : chức năng **Ah**

Chức năng này giống chức năng 09 h, ngoại trừ byte thuộc tính không thay đổi, nghĩa là ký tự được hiển thị với thuộc tính có sẵn từ trước.

- Thông số vào : AH = 0Ah
  BH = số hiệu trang
  AL = mã ASCII của ký tự
  CX = số lần viết ký tự.
- Thông số ra : không có.

*d) Ghi ký tự và dịch con trỏ* : chức năng **0Eh** (chế độ teletype)

Chức năng này giống chức năng 09h, nhưng có điều khác là sau mỗi lần ghi ký tự ra màn hình, con trỏ tự động dịch đi một vị trí của ký tự.

- Thông số vào : AH = 0Eh
  AL = mã ASCII của ký tự
  BH = số hiệu trang
  BL = thuộc tính mầu.
- Thông số ra : không có.

*Ví dụ 8:* Ghi mỗi lần 5 ký tự từ chữ "a" tới chữ "n" ra khoảng giữa màn hình với mầu nền tăng dần từ mầu xanh lá cây, còn mầu ký tự tăng dần từ mầu đỏ.

```
        MOV AH, 00h       ; xác định chế độ màn hình
        MOV AL, 02h       ; chế độ 2, văn bản, 16 mầu, 25x80
        INT 10h           ; gọi ngắt
        MOV AH, 02h       ; di chuyển vị trí con trỏ
        MOV  BH, 00h      ; trang 0
        MOV DH, 0Ch       ; dòng thứ Ch =12
        MOV DL, 28h       ; cột thứ  2x16 + 8 = 40
        INT 10h           ; gọi ngắt cho di chuyển vị trí
        MOV AH, 09h       ; ghi ký tự ra màn hình với thuộc tính mong muốn
        MOV AL, 61h       ; mã ASCII của chữ a
        MOV BL, 01h       ; nền đen (0), ký tự xanh lá cây (1)
        MOV CX, 0005h     ; mỗi lần ghi 5 ký tự giống nhau
LẶP:    INT 10h           ; gọi ngắt
        INC AL            ; tăng mã ký tự
        ADD BL, 11h       ; tăng mầu nền và mầu ký tự
        CMP AL, 6Fh       ; hạn chế tới chữ n
```

```
JNZ LẶP          ; trở về LẶP
INT 20           ; kết thúc chương trình.
```

Nếu :

- Chạy chương trình trên với DEBUG, ta chỉ cần thay địa chỉ của lệnh gọi ngắt INT 10h để ghi ký tự ra màn hình vào chữ LẶP trong lệnh JNZ LẶP và bỏ chữ LẶP trước lệnh INT 10h đó.

- Chạy chương trình từng tiến trình (Proceed) bằng lệnh P của DEBUG, ta sẽ quan sát được sự hiển thị lần lượt của các ký tự trên màn hình.

- Chạy chương trình bằng lệnh G của DEBUG, ta có các ký tự đè lên nhau và chỉ quan sát được dòng 5 chữ "n" cuối cùng ở khoảng giữa màn hình với nầu nền và mầu ký tự cuối cùng.

- Thay chức năng 09h bằng 0Ah, ta có kết quả giống như trên nhưng ký tự mầu trắng trên nền đen ban đầu (thuộc tính có sẵn, không thay đổi được).

- Thay chức năng 09h bằng 0Eh, ta có một hàng có 14 chữ từ "a" tới "n" ở khoảng giữa màn hình.

***e) Ghi chuỗi ký tự*** : chức năng **13h**

Chức năng này chỉ thực hiện trên BIOS của PC/AT, EGA và PS/2.

Có 4 chức năng con ghi vào AL :

- *Chức năng con 00h:* ghi chuỗi ký tự lên màn hình, thuộc tính trong thanh ghi BL, không thay đổi vị trí con trỏ.

- *Chức năng con 01h:* ghi chuỗi ký tự lên màn hình, thuộc tính trong thanh ghi BL, chuyển vị trí con trỏ tới cuối chuỗi.

- *Chức năng con 02h:* ghi một chuỗi ký tự, thuộc tính trong vùng đệm có độ dời trong DX, chuyển con trỏ tới cuối chuỗi.

## 5. Phát ký tự với kích thước thay đổi : Chức năng 11h

Cũng như phục vụ 10h, phục vụ 11h xuất hiện với BIOS của VGA, MCGA. Số phục vụ nhỏ tăng lên trong BIOS của PS/2. Có 4 nhóm nhỏ (bảng 5.12).

- *Chức năng con (00h÷04h)* : thay đổi cỡ chữ dùng ở chế độ văn bản trên EGA và VGA hay MCGA.

- *Chức năng con (10h÷14h):* thay đổi chiều cao của ký tự.

- *Chức năng con (20h÷24h):* thay đổi cỡ ký tự trong chế độ đồ họa.

- *Chức năng con 30h:* lấy lại thông tin liên quan tới cỡ chữ hiện hành hiển thị và những cỡ chữ có thể của BIOS (bảng 5.13). Chức năng con 30h gửi trả lại nhiều bảng các bit mô phỏng ký tự cho cỡ ký tự bởi BIOS. Những giá trị trong BH khi ta gọi phục vụ nhỏ này xác định địa chỉ nào BIOS trả lại trong ES:BP (bảng 5.14).

## 6. Ghi ký tự dùng ngắt INT 21h của DOS

***a) Ghi ra một ký tự:*** chức năng **02h**

- Thông số vào: AH = 02h

  DL = mã ASCII của ký tự.

- Thông số ra : không có.

*Bảng 5.13.* **Các chức năng con của ngắt INT 10h**

| Chức năng con AL | Mô tả |
|---|---|
| | **a) Tích một cỡ chữ ở chế độ văn bản** |
| 00h | Tích một cỡ chữ do người dùng chọn |
| 01h | Tích cỡ chữ 8x14 của BIOS |
| 02h | Tích cỡ chữ 8x8 của BIOS |
| 03h | Chọn cỡ chữ đã hiển thị |
| 04h | Tích cỡ chữ 8x16 của BIOS |
| | **b) Tích một cỡ chữ ở chế độ văn bản và chỉnh kích thước** |
| 10h | Tích cỡ chữ do người dùng chọn |
| | **c) Tích một cỡ chữ ở chế độ đồ họa** |
| 20h | Tích một cỡ tương thích với CGA được người dùng chọn |
| 21h | Tích một cỡ chữ chọn bởi người dùng |
| 22h | Tích cỡ 8x14 của BIOS |
| 23h | Tích cỡ 8x14 của BIOS |
| | **d) Lấy tin của máy phát ký tự** |
| 30h | Đọc tin của máy phát ký tự |

*Bảng 5.14.* **Các địa chỉ của bảng mô phỏng của các bit của các ký tự**

| Thông số BH | Gửi lại |
|---|---|
| 00h | Các ký tự 8x8 ở chế độ đồ họa tương thích CGA (nội dung của véctơ ngắt 1Fh) |
| 01h | Ký tự của chế độ đồ họa hiện hành (nội dung của véctơ ngắt 43h) |
| 02h | Ký tự 8x14 của BIOS |
| 03h | Ký tự 8x8 của BIOS |
| 04h | Nửa thứ hai của bảng các ký tự 8x8 của BIOS |
| 05h | Các ký tự thứ cấp 9x14 của BIOS |
| 06h | Các ký tự 8x16 của BIOS (chỉ MCGA và VGA) |
| 07h | Các ký tự thứ cấp 9x14 của BIOS (chỉ cho MCGA và VGA) |

*b) Ghi ra chuỗi ký tự:* chức năng **09h**

- Thông số vào: AH = 09h
  DS : DX chứa chuỗi ký tự, kết thúc bởi mã của $.
- Thông số ra : không có.

## 5.3.5. LẬP TRÌNH Ở CHẾ ĐỘ ĐỒ HỌA

Chế độ đồ họa của màn hình là chế độ mà màn hình chỉ vẽ một điểm ảnh (pixel) tại một thời điểm, khác với chế độ văn bản là ghi liên tiếp nhiều điểm ảnh liên tiếp từng dòng theo các cột của ma trận. Tất nhiên là trong chế độ ảnh điểm hay đồ họa này, màn hình cũng ghi được các ký tự nhưng theo các điểm ảnh của từng dòng liên tiếp cho hết cả trang của bộ nhớ màn hình theo bản đồ bit (bitmap) chứ không theo từng ma trận. Do đó, biểu diễn văn bản trong chế độ đồ họa này chậm hơn chế độ văn bản rất nhiều. Chế độ đồ họa có các ưu điểm sau mà chế độ văn bản không có được là:

- Biểu diễn được hình vẽ tinh tế với độ phân giải cao (tới 1280x1024) - mầu sắc đẹp (tới 256 mầu hoặc hơn nữa).

- Có thư viện các hàm phục vụ màn hình.

Bìa màn hình TIGA (Texas Intruments Graphic Adapter) dùng bộ vi xử lý riêng cho việc in ra đồ họa, làm giảm khối lượng công việc của bộ xử lý chính. Trong khi đó, bìa màn hình trước đó là Super VGA chỉ có độ phân giải 800x600.

Có ba bìa màn hình đồ họa mầu phổ biến cho các máy PC là:

- CGA : Color Graphic Adapter - bộ phối ghép đồ họa mầu.
- EGA : Enhanced Graphic Adapter - bộ phối ghép đồ họa nâng cao.
- VGA : Video Graphic Adapter - bộ phối ghép đồ họa kiểu ma trận điểm.

Đặc tính của các bìa này đã giới thiệu sơ bộ ở phần trên.

Nhiệm vụ của lập trình đồ họa này cũng theo nhiệm vụ chung của màn hình, chỉ có điều khác biệt là:

- Mỗi lần đưa ra màn hình là một ảnh điểm.
- Chỉ có 4 mầu cho các bìa màn hình thông dụng (độ phân giải trung bình).
- Chỉ có 2 mầu cho các bìa màn hình có độ phân giải cao.
- Có thể đưa ra màn hình ký tự theo nguyên tắc vẽ từng điểm ảnh trong bản đồ bit (bitmap).

## 1. Xác định chế độ hiển thị : chức năng 00h

- Thông số vào : AH = 00h
  AL = số hiệu chế độ (bảng 5.4).
- Thông số ra : không có.

Tích số 640x480 nói lên độ phân giải của màn hình trong đó 640 là số điểm ảnh của một cột còn 480 là số điểm ảnh của một hàng (dòng).

Trừ các chế độ 8h, 9h, Ah cho máy loại xách tay, còn lại là các chế độ cho các màn hình khác nhau.

Nhóm chế độ 4h, 5h, 6h chủ yếu dùng cho bìa CGA.

Nhóm chế độ Dh, Eh, Fh 10h chủ yếu dùng cho bìa EGA.

Nhóm chế độ 11h, 12h, 13h chủ yếu dùng cho bìa VGA.

Xác định chế độ cho chế độ đồ họa cũng giống như chế độ văn bản, đều dùng chức năng AH = 00h nhưng chỉ số chế độ theo bảng 5.7 ở trên.

*Ví dụ 1:* Viết chương trình đổi chế độ màn hình như sau:

- Ấn phím số 4 (mã ASCI 34h) chuyển sang chế độ 4 (320x200, 4 mầu).
- Ấn phím số 5 (mã ASCII 35h) chuyển sang chế độ 5 (640x200, 4 mầu xám).
- Ấn phím số 6 (mã ASCII 36h) chuyển sang chế độ 6 (640x200, 2 mầu).
- Ấn phím chữ D (mã ASCII 44h) chuyển sang chế độ D (320x200, 16 mầu).
- Ấn phím chữ E (mã ASCII 45h) chuyển sang chế độ E (640x200, 16 mầu).
- Ấn phím chữ F (mã ASCII 46h) chuyển sang chế độ F (640x350, đơn sắc).
- Ấn phím $F_{10}$ (mã quét 44h) chuyển sang chế độ $F_{10}$(640x350, 16 mầu).
- Ấn phím $F_{11}$(mã quét 85h) chuyển sang chế độ $F_{11}$ (640x480, 2 mầu).
- Ấn phím $F_{12}$ (mã quét 86h) chuyển sang chế độ $F_{12}$ (640x200, 16 mầu).
- Ấn phím Esc (mã 001Bh) thoát khỏi chương trình.

```
xxxx: 0100  XOR AX, AX        ; xóa AX về 0
xxxx: 0102  XOR BX, BX        ; xóa BX về 0
```

```
xxxx: 0104  XOR DX, DX       ; xóa DX về 0
xxxx: 0106  MOV AH, 00h      ; hàm nhập ký tự từ bàn phím
xxxx: 0108  INT 16h          ; gọi ngắt phục vụ bàn phím
xxxx: 010A CMP AH, 44h       ; xem mã quét có phải của F10 không ?
xxxx: 010D  JZ 128           ; nếu đúng, nhảy tới địa chỉ IP = 128 để đổi chế độ
xxxx: 010F  CMP AH, 85 h     ; xem mã quét có phải của F11 không ?
xxxx: 0112  JZ 12E           ; nếu đúng, nhảy tới địa chỉ IP = 12E để đổi chế độ
xxxx: 0114  CMP AH, 86h      ; xem mã quét có phải của F12 không ?
xxxx: 0117  JZ 135           ; nếu đúng, nhảy tới địa chỉ IP = 135 để đổi chế độ
xxxx: 0119  CMP AL, 34h      ; xem mã ASCII có phải của số 4 không ?
xxxx: 011C  JZ 13C           ; nếu đúng, nhảy tới địa chỉ IP = 13C để đổi chế độ
xxxx: 011E  CMP AL, 35h      ; xem mã ASCII có phải của số 5 không ?
xxxx: 0121  JZ 143           ; nếu đúng, nhảy tới địa chỉ IP = 143 để đổi chế độ
xxxx: 011E  CMP AL, 36h      ; xem mã ASCII có phải của số 6 không ?
xxxx: 0121  JZ 14A           ; nếu đúng, nhảy tới địa chỉ IP = 14A để đổi chế độ
xxxx: 011E  CMP AL, 44h      ; xem mã ASCII có phải của chữ D không ?
xxxx: 0121  JZ 151           ; nếu đúng, nhảy tới địa chỉ IP = 151 để đổi chế độ
xxxx: 011E  CMP AL, 45h      ; xem mã ASCII có phải của chữ E không ?
xxxx: 0121  JZ 158           ; nếu đúng, nhảy tới địa chỉ IP = 158 để đổi chế độ
xxxx: 011E CMP AL, 46h       ; xem mã ASCII có phải của chữ F không ?
xxxx: 0121  JZ 15F           ; nếu đúng, nhảy tới địa chỉ IP = 15F để đổi chế độ
xxxx: 0123 CMP AL, 1Bh       ; xem có phải mã ASCII của phím Esc không ?
xxxx: 0126  JZ 164           ; nếu đúng, nhảy tới địa chỉ IP = 164 để thoát
xxxx: 0128  MOV AL, 10h      ; nạp chế độ 10
xxxx: 012A CALL 166          ; gọi chương trình con đổi chế độ màn hình
xxxx: 012C  JMP 164          ; kết thúc
xxxx: 012E  MOV AL, 11h      ; nạp chế độ 11
xxxx: 0130  CALL 166         ; gọi chương trình con đổi chế độ màn hình
xxxx: 0133  JMP 164          ; kết thúc
xxxx: 0135  MOV AL, 12h      ; nạp chế độ 12
xxxx: 0137  CALL 166         ; gọi chương trình con đổi chế độ màn hình
xxxx: 013A  JMP 164          ; kết thúc
xxxx: 013C MOV AL, 04h       ; nạp chế độ 4
xxxx: 013E  CALL 166         ; gọi chương trình con đổi chế độ màn hình
xxxx: 0141  JMP 164          ; kết thúc
xxxx: 0143  MOV AL, 05h      ; nạp chế độ 5
xxxx: 0145  CALL 166         ; gọi chương trình con đổi chế độ màn hình
xxxx: 0148  JMP 164          ; kết thúc chương trình
xxxx: 014A MOV AL, 06h       ; nạp chế độ 6
xxxx: 014C  CALL 166         ; gọi chương trình con đổi chế độ màn hình
xxxx: 014F  JMP 164          ; kết thúc chương trình
xxxx: 0151  MOV AL, 0Dh      ; nạp chế độ D
```

```
xxxx: 0153  CALL 166          ; gọi chương trình con đổi chế độ màn hình
xxxx: 0156  JMP 164           ; kết thúc chương trình
xxxx: 0158  MOV AL, 0Eh       ; nạp chế độ E
xxxx: 015A  CALL 166          ; gọi chương trình con đổi chế độ màn hình
xxxx: 015D JMP 164            ; kết thúc chương trình
xxxx: 015F MOV AL, 0Fh        ; nạp chế độ F
xxxx: 0161  CALL 166          ; gọi chương trình con đổi chế độ màn hình
xxxx: 0164  INT 20            ; kết thúc chương trình
xxxx: 0166  MOV AH, 00h       ; thay đổi chế độ
xxxx: 0168  INT 10h           ; gọi ngắt thay đổi chế độ
xxxx: 016A  RET               ; trở về từ chương trình con.
```

## 2. Xác định mầu sắc của nền, điểm ảnh và ký tự

Bìa CGA có thể hiển thị 16 mầu, tùy thuộc các bit R, G, B của byte thuộc tính như chế độ văn bản (bảng 5.11). Tuy nhiên, ở chế độ đồ họa, số mầu mà bìa hiển thị bị hạn chế tùy thuộc vào chế độ phân giải :

- Độ phân giải trung bình chỉ hiển thị nhóm 4 mầu khác nhau theo 2 bảng mầu. Mỗi bảng mầu là tập hợp các mầu có thể hiện cùng một lúc. Mỗi bảng mầu chứa 3 mầu cố định (bảng 5.15) và một mầu nền có thể chọn từ 16 mầu chuẩn. Mầu nền là mầu mặc định của tất cả các điểm ảnh, do đó màn hình sẽ thể hiện một mầu nền nếu không có dữ liệu ghi lên nó.

- Độ phân giải cao : chỉ thể hiện 2 mầu, mỗi điểm ảnh có giá trị 0 (mầu đen hoặc trùng với mầu nền) hoặc 1(mầu trắng).

***Bảng 5.15.*** **Các bảng 4 mầu của CGA**

| | | Mầu AL là | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| Bộ mầu | 0 | Xanh lá cây | Đỏ | Vàng |
| Bộ mầu | 1 | Xanh da trời | Hồng | Trắng |

Người ta chọn mầu nền và tập hợp mầu của điểm ảnh (bảng mầu) bằng chức năng 0Bh như sau :

***a) Xác định mầu nền*** : chức năng **0Bh** và **BH = 00h**

- Thông số vào: AH = 0Bh
  BH = 00h
  BL = số hiệu của mầu (0 ÷ 15 như bảng 5.11).
- Thông số ra : không.

Cũng chức năng này nhưng ở chế độ văn bản, xác định mầu viền của màn hình.

*Ví dụ 2:* Chương trình thay đổi chế độ và thay đổi mầu nền của màn hình ở chế độ đồ họa.

```
xxxx: 0100    MOV AL, 04h      ;  chế độ 4
xxxx: 0102    MOV BH, 00h      ; trang 0
xxxx: 0104    MOV BL, 00h      ; mầu nền 0
xxxx: 0106    MOV AH, 00h      ; xác định chế độ
xxxx: 0108    INT 10h          ; gọi ngắt
```

```
xxxx: 010A   MOV AH, 0Bh      ; xác định mầu nền
xxxx: 010C   INT 10h          ; gọi ngắt
xxxx: 010E   INC  BL          ; tăng số hiệu của mầu nền
xxxx: 0110   INC  AL          ; tăng số chế độ
xxxx: 0112   CMP AL, 08h      ; xem tới chế độ 8 chưa ?
xxxx: 0114   JNZ 106          ; trở về xác định chế độ và mầu nền khác
xxxx: 0116   ADD AL, 05h      ; chế độ Dh
xxxx: 0118   MOV AH, 00h      ; xác định chế độ
xxxx: 011A   INT 10h          ; gọi ngắt
xxxx: 011C   MOV AH, 0Bh      ; xác định mầu nền
xxxx: 011E   INT 10h          ; gọi ngắt
xxxx: 0120   INC BL           ; tăng mầu nền
xxxx: 0122   INC  AL          ; tăng chế độ
xxxx: 0124   CMP AL, 14h      ; xem hết chế độ chưa ?
xxxx: 0126   JNZ 118          ; trở về chế độ và mầu nền mới
xxxx: 0128   INT 20h          ; kết thúc chương trình.
```

***b) Xác định mầu của ảnh điểm*** : chức năng **0Bh** và **BH = 01h**

- Thông số vào: AH = 0Bh
  BH = 01h
  BL = số hiệu của bảng mầu (0 hay 1)

(AL = số hiệu của mầu (bảng 5.13) trong chức năng ghi ảnh điểm AH = 0Ch).

- Thông số ra : không.

***c) Xác định mầu của ký tự ở chế độ đồ họa***: ở chế độ đồ họa, cũng cần ghi ký tự với mầu ra màn hình với chức năng và mầu như của ảnh điểm.

## 3. Ghi/ đọc ảnh điểm

***a) Ghi ảnh điểm***: chức năng **0Ch**

- Thông số vào: AH = 0Ch
  AL = trị số mầu của ảnh điểm (bảng 5.9)
  BH = số trang (với CGA - giá trị này bỏ qua)
  CX = tọa độ cột (tính theo hệ hexa)
  DX = tọa độ dòng (tính theo hệ hexa).
- Thông số ra : không có.

Khác với chế độ văn bản (cần xác định vị trí con trỏ trước khi ghi ký tự), ở đây ghi ảnh điểm có kèm theo vị trí (dòng, cột của ảnh điểm tính theo hệ hexa).

***b) Đọc ảnh điểm*** : chức năng **0Dh**

- Thông số vào : AH = 0Dh
  BH = số trang (với CGA - bỏ qua)
  CX = tọa độ cột (tính theo hệ hexa)
  DX = tọa độ dòng (tính theo hệ hexa).
- Thông số ra : AL = trị số ảnh điểm.

Trị số trong AL chỉ mầu đã biết từ bảng chọn mầu 5.9 ở trên với:

1 - mầu trắng (cho CGA phân giải cao)

0 ÷ 3 - mầu với CGA trung bình với bảng 0 hay 1 ở trên.

*Ví dụ 3:* Viết chương trình vẽ viền màn hình với nền xanh da trời, đường vẽ mầu đỏ.

```
xxxx: 0100   MOV AH, 00h       ; xác định chế độ
xxxx: 0102   MOV AL, 04h       ; chế độ 4
xxxx: 0104   INT 10h           ; gọi ngắt
xxxx: 0106   MOV CX, 0001h     ; tọa độ cột
xxxx: 0109   MOV DX, 0001h     ; tọa độ dòng
xxxx: 010C   CALL 138          ; gọi chương trình con vẽ hình
xxxx: 010F   INC CX            ; tăng số cột của dòng mép trên
xxxx: 0110   CMP CX, 13Fh      ; so với mép màn hình
xxxx: 0114   JNZ 10C           ; gọi chương trình con vẽ hình
xxxx: 0116   INC DX            ; tăng số dòng của cột bên phải
xxxx: 0117   CMP DX, 0C8h      ; so với mép màn hình
xxxx: 011B   JZ 122            ; chuyển sang vẽ dòng cuối
xxxx: 011D   CALL 138          ; gọi chương trình con vẽ hình
xxxx: 0120   JMP 116           ; tiếp tục vẽ cột
xxxx: 0122   DEC CX            ; vẽ dòng cuối
xxxx: 0123   CMP CX, 00h       ; so với mép màn hình
xxxx: 0126   JZ 12D            ; chuyển sang vẽ cột trái
xxxx: 0128   CALL 138          ; gọi chương trình con vẽ hình
xxxx: 012B   JMP 122           ; tiếp tục vẽ dòng cuối
xxxx: 012D   DEC DX            ; vẽ cột trái
xxxx: 012E   CMP DX, 00h       ; so với mép màn hình
xxxx: 0131   JZ 14D            ; kết thúc vẽ
xxxx: 0133   CALL 138          ; gọi chương trình con vẽ hình
xxxx: 0136   JMP 12D           ; tiếp tục vẽ cột trái
xxxx: 0138   MOV AH, 0B        ; xác định mầu
xxxx: 013A   MOV BH, 00h       ; mầu nền
xxxx: 013C   MOV BL, 01h       ; xanh da trời
xxxx: 013E   INT 10h           ; gọi ngắt
xxxx: 0140   MOV BH, 01h       ; xác định mầu điểm ảnh
xxxx: 0142   MOV BL, 00h       ; bảng mầu 0
xxxx: 0144   INT 10h           ; gọi ngắt
xxxx: 0146   MOV 0Ch           ; ghi điểm ảnh
xxxx: 0148   MOV AL, 02h       ; mầu đỏ của bảng mầu
xxxx: 014A   INT 10h           ; gọi ngắt
xxxx: 014C   RET               ; trở về từ chương trình con vẽ điểm ảnh
xxxx: 014D   INT 20h           ; kết thúc chương trình.
```

## 4. Ghi ký tự trong chế độ đồ họa

Ở chế độ đồ họa cũng cần ghi ký tự với hàm ghi ký tự **09h** như chế độ văn bản, nhưng trước khi ghi ký tự với thuộc tính ở 4 bit thấp của thanh ghi BL ta phải xác định:

- Xác định vị trí con trỏ bằng chức năng AH = 02h như chế độ văn bản.
- Mầu nền (BH = 0) với chức năng 0Bh và số hiệu mầu trong 4 bit thấp trong BL (như chế độ văn bản).
- Mầu của ký tự như mầu của ảnh điểm (theo bảng 5.11).

*Ví dụ 4:* Chương trình viết lệnh hiển thị chữ 'C' mầu đỏ ở góc phải trên màn hình. Sử dụng chế độ 4 với nền mầu xanh da trời.

```
MOV AH, 00h      ; thiết lập chế độ
MOV AL, 04h      ; chế độ 4
INT 10h          ; xác lập chế độ màn hình
MOV AH, 0Bh      ; chọn mầu nền
MOV BH, 00h      ; mầu nền
MOV BL, 03h      ; mầu xanh da trời
INT 10h          ; xác lập mầu nền
MOV AH, 02h      ; xác định vị trí con trỏ
MOV BH, 00h      ; trang 0
MOV DH, 00h      ; dòng 0
MOV DL, 39h      ; cột 39
INT 10h          ; xác định vị trí (0, 39)
MOV AH, 09h      ; hàm ghi ký tự
MOV AL, 43h      ; mã ASCII của chữ C
MOV BL, 02h      ; mầu đỏ của bảng với BL mặc định bằng 0
MOV CX, 01h      ; ghi một ký tự
INT 10h          ; ghi ký tự
INT 20h          ; kết thúc chương trình.
```

## 5.3.6. LẬP TRÌNH TRỰC TIẾP CHO MÀN HÌNH

Lập trình trực tiếp cho màn hình là viết chương trình cho màn hình không sử dụng một ngắt nào của hệ điều hành. Lập trình trực tiếp có ưu điểm là điều khiển nhanh, can thiệp trực tiếp vào thiết bị (phần cứng), không qua ROM BIOS của hệ điều hành.

Muốn vậy, chúng ta phải hiểu rõ màn hình, bìa màn hình trong đó có khối điều khiển bằng vi xử lý và khối nhớ đệm RAM cho màn hình.

**1. Bìa màn hình**: Một bìa màn hình điển hình có các khối sau:

- Khối điều khiển màn hình (CRTC) có tác dụng điều khiển toàn bìa hoạt động. Khối này thường chứa vi xử lý (MC 6845 cho các bìa CGA, 8514/A cho các bìa EGA/VGA và TI 34010, TI 34020 và Intel 82786 cho bìa TIGA).
- Khối nhớ ROM chứa mẫu ký tự để đưa phát các ký tự.
- Khối phát các ký tự.
- Khối điều khiển thuộc tính để điều khiển phát mầu (viền, nền và mầu ký tự hoặc ảnh điểm).
- Khối nhớ đệm RAM cho màn hình để ghi nhớ và để phát mọi ảnh điểm của màn hình.

- Khối điều khiển tín hiệu để phát các tín hiệu quét ngang, quét dọc, tín hiệu video v.v... cho ống hình.

Hai khối quan trọng nhất và liên quan tới lập trình trực tiếp là khối điều khiển CRT và nhớ đệm RAM.

*a) Khối điều khiển CRT*

Khối điều khiển dùng vi xử lý sớm nhất là MC 6845 của hãng Motorola. Nó có thể điều khiển 16K của bộ nhớ RAM (qua 14 đường dây), nhận ký tự từ nhớ ROM (qua 5 đường dây).

Liên hệ với xử lý trung tâm (qua 4 đường dây điều khiển và 8 đường dây số liệu), điều khiển ống hình qua các đường dây điều khiển tín hiệu (quét ngang, quét dọc, bật màn hình) và nhận các tín hiệu xóa, nhịp, bút sáng, con trỏ.

MC 6845 có 18 thanh ghi nội. Tùy theo loại bìa sử dụng MC 6845 mà địa chỉ này khác nhau, ta cần biết loại bìa để có địa chỉ tương ứng. Các thanh ghi như bảng 5.16.

***Bảng 5.16.* Các thanh ghi của MC 6845**

| Địa chỉ nền | Tên thanh ghi | Số | Chức năng |
|---|---|---|---|
| 3X4h | Thanh ghi địa chỉ | | Chọn thanh ghi hoạt động |
| 3x5h + độ dời 00h tới 05h | Dạng ngang | 0÷3 | Xác định các tham số màn hình ngang |
| + độ dời 06h | Dạng dọc | 4÷8 | Xác định các tham số màn hình dọc (tổng dòng) |
| độ dời 07h tới 09h | Chế độ khối ghép nối | 9 | Chọn xung đồng bộ (tràn, đặt lại quét hàng, hàng tối đa) |
| + độ dời 0Ah | Bắt đầu con trỏ | 10 | Xác định dòng quét cho vị trí đầu con trỏ và thao tác nhấp nháy/không nhấp nháy |
| + độ dời 0Bh | Kết thúc con trỏ | 11 | Xác định dòng quét cho vị trí kết thúc con trỏ |
| + độ dời 0Ch và 0D | Địa chỉ bắt đầu (cao và thấp) | 12÷13 | Xác định địa chỉ bắt đầu bộ nhớ màn hình |
| + độ dời 0E và 0Fh | Địa chỉ con trỏ (cao và thấp) | 14÷15 | Xác định vị trí con trỏ trên màn hình |
| + độ dời 10h và 11h | Địa chỉ bút sáng (cao và thấp) | 15÷17 | Xác định vị trí bút sáng trên màn hình |

Đối với mỗi bìa khác nhau (MDA, Hercules, CGA, EGA, VGA, Super VGA) có các địa chỉ và cấu trúc các thanh ghi khác nhau. Ở đây, ta chỉ xét cho bìa CGA thông dụng. Bìa CGA có :

- *Thanh ghi chọn chế độ:* địa chỉ 3D8h, có các bit như sau:

$d_0$ - chỉ chế độ văn bản với 40 ký tự (giá trị 0), 80 ký tự (giá trị 1).

$d_1$ - chỉ chế độ đồ họa với các chế độ khác (giá trị 0), 320 điểm ảnh (giá trị 1).

$d_2$ - chỉ loại màn hình đơn sắc (giá trị 1), mầu (giá trị 0).

$d_3$ - chỉ tắt (giá trị 0) hay bật (giá trị 1) màn hình.

$d_4$ - chỉ chế độ đồ họa 640 cột (giá trị 1) hay chế độ khác (giá trị 0).

$d_5$ - chỉ không (giá trị 0) hay cho phép nhấp nháy (giá trị 1).

$d_6$ và $d_7$ không dùng (giá trị 0).

Giá trị của thanh ghi chế độ tương ứng với các chế độ như sau: 02Ch (chế độ 0), 28h (chế độ 1), 02Dh (chế độ 2), 029h (chế độ 3), 02Eh (chế độ 4), 02Ah (chế độ 5), 01Ch (chế độ 6).

- *Thanh ghi chọn mầu viền và bảng mầu:* địa chỉ 3D9h, có các bit như sau:

  $d_0$÷ $d_3$ là các bit BGRI cho mầu viền (theo bảng mầu 5.11).

  $d_4$ bit tăng cường nền (chế độ văn bản) với tắt (giá trị 0) và bật (giá trị 1).

  $d_5$ bit chọn bảng mầu (chế độ đồ họa) với bảng 0 (giá trị 0) hay 1 (giá trị 1).

- *Thanh ghi trạng thái:* địa chỉ 3DAh, có các bit sau:

  $d_0$ = màn hình sáng (1) hay tắt (0).

  $d_1$ = bút sáng tắt (= 1) hay bật (= 0).

  $d_2$ = xung của bút sáng không tác dụng (0) hay sử dụng (1).

  $d_3$ = 1 tín hiệu quét thẳng đứng trở về.

  $d_4$ ÷ $d_7$ không sử dụng.

- *Thanh ghi chỉ số :* địa chỉ 3D4h, chỉ số hiệu (0 đến 17) của thanh ghi nội, các bit $d_0$ đến $d_4$.

- *Thanh ghi đệm số liệu :* địa chỉ 3D5h, 8 bit $d_0$÷$d_7$.

***b) Bộ nhớ đệm RAM cho màn hình***

Màn hình cần bộ nhớ đệm để ghi các tin (ký tự, điểm ảnh ra màn hình) theo nguyên tắc ánh xạ (một điểm ảnh của màn hình tương ứng với một ô nhớ). Tùy chế độ với độ phân giải khác nhau, ta cần dung lượng nhớ khác nhau, nhưng có điểm chung là:

- Các bìa màn hình có địa chỉ nhớ đầu tiên là B000 : 0000 (đoạn B) hay A000 : 0000 (đoạn A).

- Mỗi bìa thường được máy vi tính dành cho 32KB nhớ (bìa MDA, CGA, Hercules) hay 256KB (bìa EGA, VGA). Trong đó:

  + Bìa MDA dùng 4KB đầu, dùng đoạn B.

  + Bìa CGA dùng 16KB đầu, dùng đoạn B.

  + Bìa Hercules dùng 32KB đầu, đoạn A và nửa thứ hai của đoạn B.

- Mỗi ký tự cần 2 byte (mã và thuộc tính) nên màn hình đơn sắc (MDA) với độ phân giải 80x25 cần 2000x2 = 4000 byte nhớ với địa chỉ từ B0000h - B0FFFh cho 25 dòng, 80 cột (địa chỉ B0FA00h - B0FFFh không sử dụng).

- Bộ nhớ đệm chia theo trang 4KB nhớ (bìa MDA tối đa có 8 trang).

Bộ nhớ đệm của CGA chia làm hai khối (vùng): vùng cho mã của ký tự (byte chẵn) và vùng cho thuộc tính (byte lẻ). Vùng 1 bắt đầu từ địa chỉ B800h và vùng 2 từ địa chỉ BA00h.

Khi bìa ở chế độ đồ họa, vùng 1 chứa các tin về điểm ảnh chẵn, còn vùng 2 chứa các tin về điểm ảnh lẻ (bắt đầu từ địa chỉ B000:A000h). Bắt đầu từ đỉnh của vùng nhớ đệm của bìa CGA, hàng 0 (80 byte đầu tiên) được theo sau bởi hàng 2 và sau đó là hàng 4 v.v...

Để biết vị trí trong khối nhớ đệm của một điểm ảnh, ta phải xác định được bit thấp nhất của chỉ số hàng của khối 1 hay khối 2. Ta phải dùng công thức:

**Đoạn (mảng) = B800h + (Y và X) x 200h** với X và Y là các tọa độ của ảnh điểm.

Địa chỉ của dịch chuyển (offset) sau đó được tính nhờ những bit còn lại. Phải thêm số bit trong dòng có ở "bên trái" của ảnh điểm. Như vậy, ta có được một con trỏ vào địa chỉ mong muốn.

**Dịch chuyển** (độ phân giải 320x200) = **(Y và 254)x 40 + (X chia 4).**

**Dịch chuyển** (độ phân giải 640x200) = **(Y và 254)x 40 + (Y chia 4).**

## 2. Lập trình trực tiếp màn hình

*a) Nguyên tắc*

Lập trình trực tiếp màn hình là ghi các tin về điều khiển, về thông số vào các thanh ghi nội của vi xử lý điều khiển màn hình (ví dụ MC 6845), vào bộ nhớ đệm RAM của màn hình, vào các thanh ghi thông dụng của vi xử lý trung tâm. Hơn nữa, ta cũng cần đọc và xử lý tin về trạng thái, về dữ liệu của các thanh ghi trên của khối điều khiển, của bộ nhớ đệm RAM của màn hình và của các thanh ghi thông dụng. Trước khi ghi tin, ta cần:

- Xác định địa chỉ của thanh ghi của khối điều khiển. Vì số địa chỉ có hạn nên số liệu của thanh ghi chỉ số chính là chỉ số thanh ghi mà ta cần trao đổi số liệu.

- Xác định dữ liệu cần ghi theo dạng các thanh ghi điều khiển đã cho của khối điều khiển.

Trước khi đọc tin (về dữ liệu, trạng thái) ta cũng cần xác định địa chỉ của thanh ghi cần đọc. Sau khi đọc, ta cần xử lý tin về trạng thái như đã quy định của thanh ghi trạng thái.

Về nhiệm vụ lập trình, cũng theo các nhiệm vụ của lập trình sử dụng ngắt đã xét như xác định chế độ, xác định vị trí và kích thước cửa sổ, con trỏ, điểm ảnh, xác định mầu (viền, nền, ký tự, ảnh điểm) và ghi, đọc ký tự hoặc ảnh điểm. Dưới đây, chúng ta chỉ giới thiệu một số ví dụ cho các nhiệm vụ của lập trình trực tiếp trên.

*b) Một số ví dụ về lập trình trực tiếp màn hình*

Sau đây là một số ví dụ dưới dạng chương trình con, có thể chạy trong môi trường DEBUG khi thay lệnh trở về RET bằng lệnh kết thúc chương trình INT 20h.

+ *Ví dụ 1:* **Ghi vào thanh ghi của khối điều khiển** (*setvk*)

```
MOV DX, địa chỉ 6845      ; ghi địa chỉ của thanh ghi chỉ số
MOV AL, số hiệu           ; số hiệu thanh ghi nội
OUT DX, AL                ; ghi số hiệu ra
JMP Short $+2             ; tạm dừng một khoảng thời gian ngắn
INC DX                    ; tăng số hiệu cho thanh ghi tiếp theo
MOV AL, AH                ; AH chứa nội dung mới của thanh ghi
OUT DX, AL                ; ghi vào nội dung mới
RET
```

+ *Ví dụ 2* : **Đọc thanh ghi của khối điều khiển** (*getvk*)

```
MOV DX, địa chỉ 6845      ; ghi địa chỉ của thanh ghi chỉ số
MOV AL, số hiệu           ; số hiệu thanh ghi nội
OUT DX, AL                ; ghi số hiệu ra
INC DX                    ; tăng số hiệu cho thanh ghi tiếp theo
JMP Short $+2             ; tạm dừng một khoảng thời gian ngắn
IN AL, DX                 ; đọc vào nội dung mới
RET
```

+ *Ví dụ 3:* **Tạo cấu hình cho thanh ghi điều khiển màn hình** (*setmode*)

```
MOV DX, CONTR REG         ; ghi địa chỉ của thanh ghi điều khiển
MOV AL, 00101101b         ; cho phép hiển thị màn hình (d3 = 1)
OUT DX, AL                ; ghi tin ra
RET
```

+ *Ví dụ 4:* **Lập trình bộ điều khiển video** (*vcprog*)

```
        MOV CX, OCh             ; 12 thanh ghi được đặt
        XOR BH, BH              ; BH = 0,  bắt đầu từ thanh ghi 0
vcp1:   LODSB                   ; lấy giá trị thanh ghi từ bảng
        MOV AH, AL              ; giá trị thanh ghi cho vào AH
        MOV AL, BH              ; số hiệu thanh ghi cho vào AL
        CALL setvk              ; chuyển giá trị đến bộ điều khiển
        INC BH                  ; địa chỉ thanh ghi tiếp theo
        LOOP vcp1               ; đặt các thanh ghi phụ
        OR BL, 08h              ; bit d3 = 1, màn hình bật
        CALL setmode            ; đặt chế độ mới
        RET
```

+ *Ví dụ 5:* **Xác lập chế độ văn bản** (*text*)

```
        MOV SI, OFFSET TEXT     ; địa chỉ offset của bảng thanh ghi
        MOV BL, 00100001B       ; chế độ văn bản 80x25 nhấp nháy
        CALL vcprog
        RET
```

+ *Ví dụ 6:* **Xác lập chế độ đồ họa** (*graph*)

```
        MOV BL, 00010010B       ; chế độ đồ họa 320x200
        MOV SI, OFFSET GRAPHIC  ; địa chỉ offset của bảng thanh ghi
```

+ *Ví dụ7:* **Xác lập trang màn hình** (*setpag*)

```
        MOV BX, chỉ số trang    ; nạp chỉ số trang
        MOV CL, 5               ; nhân với 2, 048
        ROR BX, CL
        MOV AL, thanh ghi 12    ; byte cao của địa chỉ trang
        MOV AH, AL
        CALL setvk              ; gọi điều khiển video
        MOV AL, thanh ghi 13    ; byte thấp của địa chỉ trang
        MOV AH, BL
        CALL setvk
        RET
```

+ *Ví dụ 8:* **Xác lập vị trí con trỏ** (*setblink*)

```
        MOV BX, DI              ; chuyển offset của con trỏ vào DI
        MOV AL, byte cao        ; byte cao của vị trí con trỏ
        MOV AH, BH
        CALL setvk
        MOV AL, byte thấp       ; byte thấp của vị trí con trỏ
        MOV AH, BL
        RET
```

+ *Ví dụ 9:* **Xác lập kích thước con trỏ** (*cdef*)

```
        MOV AL, thanh ghi 10    ; thanh ghi dòng đầu con trỏ
```

```
MOV AH, CL
CALL setvk                  ; xác lập điều khiển
MOV AL, thanh ghi 11        ; thanh ghi dòng cuối
MOV AH, CHCALL setvk
RET
```

+ *Ví dụ 10:* **Xác định mầu** (*set col*)

```
MOV AL, giá trị mầu                 ; gửi giá trị mầu
MOV DX, địa chỉ thanh ghi mầu       ; gửi địa chỉ thanh ghi mầu
OUT DX, AL                          ; gửi giá trị mầu ra
RET
```

Ghi chú: Trong chế độ văn bản, 4 bit thấp là mầu viền.

Trong chế độ đồ họa, 4 bit thấp là mầu nền.

bit $d_5$ = chọn bảng mầu (0 bảng 0, 1 bảng 1).

+ *Ví dụ 11* : **Ghi điểm ảnh** (*pixl320*)

```
MOV CL, 07
MOV AH, BL              ; chuyển cột đến AH
AND AH, 11B             ; cột chế độ 4
SHL AH, 1               ; cột x 2
SUB CL, AH              ; 7- cộtx2
MOV AH, 11B             ; giá trị bit
SHL AX, CL              ; chuyển đến vị trí điểm
NOT AH                  ; đảo AH
SHR BX, 1               ; chia BX cho 2
SHR BX, 1               ; chia BX cho 2
XOR DI, DI              ; địa chỉ offset trong RAM video
MOV CX, VIO SEG         ; địa chỉ đoạn của trang màn hình
MOV ES, CX              ; địa chỉ đoạn vào thanh ghi đoạn
MOV AX, DX              ; chuyển dòng đến AX
SHR AX, 1               ; chia dòng cho 2
MOV CL, 80              ; hệ số là 90x90
MUL CL                  ; nhân dòng với 80
AND DX, 01              ; dòng chế độ 2
MOV CL, 03              ; 3 lần dịch
ROR DX, CL              ; quay phải (x 2000h)
MOV DI, AX              ; 80x int (dòng/2)
ADD DI, DX              ; + 2000hx (dòng chế độ 4)
ADD DI, BX              ; cộng với offset cột
MOV BL, ES:[DI]         ; lấy điểm ảnh
AND BL, AH              ; xóa các bit
OR BL, AL               ; thêm điểm ảnh
MOV ES:[DI]             ; ghi điểm ảnh
RET                     ; trở về nơi gọi
```

*+ Ví dụ 12:* **Ghi chuỗi ký tự** (*print*)

```
        CLD             ; chuỗi theo lệnh đếm xuôi
        MOV DX, VIO SEG ; địa chỉ đoạn của RAM video vào DX
        MOV CL, WAIT    ; lấy cờ chờ
        MOV ES, DX      ; địa chỉ đoạn vào ES
PRINT1: OR CL, CL       ; đã chú ý hiệu ứng rung?
        JE PRINT2       ; chưa đến PRINT2
        PUSH AX         ; lưu AX vào stack
        MOV DX, 3Dh     ; địa chỉ thanh ghi trạng thái
  HR1:  IN AL, DX       ; đọc trạng thái
        TEST AL, 1      ; kiểm tra có đường hồi ngang?
        JNE HR1         ; chờ nếu chưa có
        CLI             ; cấm ngắt
  HR2:  IN AL, DX       ; đọc trạng thái
        TEST AL, 1      ; kiểm tra đường hồi ngang?
        JE HR2          ; có, chờ
        POP AX          ; khôi phục AX chứa thuộc tính mầu
        STI             ; cho phép ngắt
PRINT2: STOSW           ; ghi thuộc tính/mầu vào RAM video
PRINT3: LODSB           ; lấy ký tự tiếp theo của chuỗi
        OR AL, AL       ; là 0?
        JNE PTINT1      ; nếu khác 0, in ra
        RET
```

## 5.4. LẬP TRÌNH CHO CHUỘT

### 5.4.1. ĐẠI CƯƠNG VỀ CHUỘT

Sau bàn phím và màn hình, chuột cũng là thiết bị thông dụng nhất, được dùng để đưa tin về vị trí của con trỏ chuột trên màn hình và tin về điều khiển máy vào máy vi tính.

#### 1. Cấu tạo chuột

Chuột có quả bóng xoay (viên bi) bằng kim loại, bọc cao su. Khi di chuột trên bàn hay trên tấm thảm chuột bằng cao su, quả bóng truyền chuyển động vào hai thanh nhỏ X, Y đặt vuông góc với nhau. Khi bóng xoay, hai trục X, Y xoay theo, làm các đĩa gắn liền với trục xoay theo. Trên đĩa có một khe hở cho ánh sáng của một photôđiốt truyền qua tạo thành một xung điện ở bộ cảm biến. Số xung điện này có số lượng khác nhau (tùy thuộc số vòng quay của đĩa) và tốc độ khác nhau (tùy thuộc tốc độ quay của đĩa). Máy vi tính đọc các xung trên và chỉ thị tọa độ X, Y của quả bóng thành vị trí của mũi tên trên màn hình, được gọi là con trỏ chuột. Chuột còn có 2 (hay 3) phím ấn để đưa tin về điều khiển (gọi chương trình phục vụ theo thực đơn theo ký tự hay biểu tượng của chương trình trên màn hình). Các thiết bị mô tả trên được lắp đặt trên một tấm có nắp hình khum giống hình con chuột nên được gọi là chuột. Chuột đã thay thế hoạt động của các phím dịch chuyển (trái, phải, lên, xuống) vì dùng chuột tiện lợi hơn. Một biến thể của chuột là quả

cầu tọa độ (track ball) vì chuột cũng là một quả cầu có bụng để trống và tiếp xúc với mặt tiếp xúc để di chuyển tọa độ.

## 2. Mạch ghép nối chuột

Chuột được ghép nối trao đổi tin với máy vi tính qua một cổng nối tiếp (thường là COM1) để trao đổi tin nối tiếp với máy. Tin được truyền theo gói với các thông số sau: vận tốc truyền: 1200 baud (xung/giây), số bit số liệu/một lời tin : 7, số bit khởi phát : 1, số bit dừng : 1 và không có kiểm tra chẵn lẻ. Trong khoảng thời gian xác định, chuột gửi những tin tức về vị trí (tọa độ X, Y) của con trỏ và trạng thái của phím (ấn, nhả). Những số liệu được gửi dưới dạng ba giá trị 7 bit, tập hợp trong một gói số liệu với các bit bắt đầu và dừng. Giá trị đầu tiên tương ứng với trạng thái của phím ấn của chuột và chứa 2 bit cao của những tọa độ của chuột như sau :

| | |
|---|---|
| bit 6 | 1 |
| bit 5 | núm ấn bên phải (0 = nhả, 1 = ấn) |
| bit 4 | núm ấn bên trái (0 = nhả, 1 = ấn) |
| bit 3÷2 | hai bit bậc cao của tọa độ Y |
| bit 1÷0 | hai bit bậc cao của tọa độ X. |

Các byte tiếp theo là 7 bit thấp của tọa độ X và Y:

- 7 bit bậc thấp của tọa độ X.

- 7 bit bậc thấp của tọa độ Y.

## 3. Điều khiển chuột

Có hai loại điều khiển chuột :

- Điều khiển được cài đặt qua một lệnh DEVICE = MOUSE của tệp CONFIG.SYS của hệ điều hành MS-DOS.

- Điều khiển bằng những chương trình thường trú (TSR) trong bộ nhớ, được khởi phát sau khi bật nguồn nuôi máy vi tính.

Trong cả hai trường hợp, sự điều khiển chuột được điều khiển bởi ngắt INT 33h, với nhiều chức năng khác nhau, để gọi các chương trình con phục vụ chuột khác nhau. Thanh ghi AH cho phép áp dụng các lựa chọn những chức năng con của ngắt này. Các thanh ghi khác ghi các thông số cho chuột nếu chức năng có yêu cầu.

Ngoài việc sử dụng các chương trình con có sẵn của MS-DOS trên, chúng ta có thể lập trình điều khiển chuột trực tiếp bằng cách viết chương trình điều khiển chuột qua các cổng dành cho chuột.

## 4. Màn hình ảo cho chuột

Khi lập trình cho chuột, cần xác định vị trí của con trỏ chuột trên màn hình. Để thuận tiện hơn, dùng một màn hình ảo cho sự xác định vị trí của con trỏ chuột. Kích thước của màn hình ảo tương ứng với các chế độ của màn hình như bảng 5.17.

***Bảng 5.17.*** **Kích thước của màn hình ảo**

| Chế độ | Kích thước của màn hình ảo |
|---|---|
| 00 ÷ 07h | 640 x 200 |
| 0D ÷ 0Eh | 640 x 200 |
| 0F ÷ 10h | 640 x 350 |
| 11h ÷ 12h | 640 x 480 |
| 13h | 640 x 200 |

Khi chuyển tọa độ từ chế độ đồ họa sang văn bản, ta phải chia cho 8 (mỗi dòng và cột tương ứng với 8 điểm) và ngược lại phải nhân với 8 để có tọa độ ở chế độ đồ họa.

## 5.4.2. NGẮT CỦA CHUỘT - INT 33H

Những chương trình sử dụng một chuột được bắt đầu bởi việc kiểm tra có bộ điều khiển chuột đã cài đặt chưa. Đó là chức năng 00h. Để kích hoạt, đặt giá trị 00h vào thanh ghi AH và thực hiện ngắt INT 33h.

Nếu một chuột đã nối, thanh ghi AX chứa giá trị FFFFh trở về của chức năng. Nếu thanh ghi chứa giá trị khác, có nghĩa là chưa có chuột nào được cài đặt.

BIOS có ngắt INT 33h đảm bảo cho hoạt động của chuột với các chức năng như bảng 5.18.

***Bảng 5.18.*** **Các chức năng của INT 33h**

| Chức năng AH | Mô tả |
|---|---|
| 00h | Khởi động và đọc về trạng thái của chuột |
| 01h | Hiển thị con trỏ của chuột |
| 02h | Dấu con trỏ của chuột |
| 03h | Đọc vị trí và trạng thái các phím của chuột |
| 04h | Thay đổi vị trí con trỏ chuột |
| 05h | Trạng thái ấn của núm điều khiển chuột |
| 06h | Trạng thái nhả của núm điều khiển chuột |
| 07h | Xác định cột cực đại và cực tiểu của chuột (tọa độ X) |
| 08h | Xác định hàng cực đại và cực tiểu của chuột (tọa độ Y) |
| 09h | Xác định dạng của con trỏ trong chế độ đồ họa |
| 0Ah | Xác định dạng của con trỏ trong chế độ văn bản |
| 0Bh | Đọc khoảng dịch chuyển của chuột |
| 0Ch | Cài đặt chương trình xử lý chuột |
| 0Dh | Kích hoạt sự mô phỏng của bút chỉ sáng |
| 0Eh | Hủy bỏ sự mô phỏng của bút chỉ sáng |
| 0Fh | Xác định tỷ số Mickey/pixel cho chuyển dời vị trí |
| 10h | Đặt vùng cấm cho con trỏ chuột |
| 13h | Đặt ngưỡng cho vận tốc gấp 2 |
| 14h | Thay chương trình xử lý chuột |
| 15h | Xác định kích thước vùng đệm cho trạng thái chuột |
| 16h | Ghi lại trạng thái của bộ điều khiển chuột |
| 17h | Khôi phục lại trạng thái của chuột |
| 18h | Cài đặt chương trình xử lý tùy chọn (mở rộng) |
| 19h | Đọc địa chỉ của chương trình xử lý tùy chọn (mở rộng) |
| 1Ah | Đặt độ nhậy của chuột |
| 1Bh | Đọc độ nhậy của chuột |
| 1Ch | Đặt tần số ngắt của chuột |
| 1Dh | Đặt số trang của màn hình |
| 1Eh | Đọc số trang của màn hình |
| 1Fh | Làm ngừng chương trình điều khiển chuột |
| 20h | Khởi động lại chương trình điều khiển chuột do 1Fh gây ra |
| 21h | Khởi tạo lại chương trình điều khiển chuột, dấu con trỏ |
| 22h | Đặt ngôn ngữ sử dụng |
| 23h | Đọc số ngôn ngữ sử dụng |
| 24h | Xác định loại chuột, số IRQ và phiên bản của chuột |

Ngoài các chức năng phụ cho sự mô phỏng bút sáng (0Dh- kích hoạt và Eh- hủy bỏ) và ngôn ngữ (22h - đặt ngôn ngữ sử dụng và 23h - đọc ngôn ngữ sử dụng), các ngắt INT 33h dành cho chuột và có thể chia thành các nhóm nhỏ liên quan tới sự điều khiển chuột, điều khiển con trỏ chuột, phím ấn của chuột và tốc độ di chuyển chuột.

## 1. Các chức năng về kích hoạt và xác định loại chuột

*a) Chức năng khởi tạo:* chức năng **AH = 00h**

Khởi tạo chương trình điều khiển và thực hiện kiểm tra xem có chương trình diều khiển đã cài đặt (thông số ra AX=FFFFh) hay chưa (AX=0000h). Nếu đã cài đặt, thanh ghi BX chứa số phím của chuột. Chúng ta phải gọi chức năng này trước ghi gọi các chức năng khác. Chức năng này đặt nhiều giá trị của chuột về giá trị ngầm định. Con trỏ chuột về trang 0 của màn hình, có dạng là một hình chữ nhật âm bản nằm ở giữa màn hình. Con trỏ sẽ tự động biến mất ngay sau đó.

*b) Ngừng và kích hoạt lại*

- *Chức năng AH = 1Fh:* làm mất tác dụng của chương trình điều khiển chuột và cho biết địa chỉ (thông số ra ES:DX) của chương trình điều khiển chuột trước đó. Ngoài ra, ta còn biết được lỗi (AX=FFFFh) hay không có lỗi (AX=001Fh).

- *Chức năng AH = 20h:* kích hoạt lại chương trình điều khiển chuột mà chức năng 1Fh làm mất tác dụng.

- *Chức năng AH = 21h:* khởi tạo lại chương trình điều khiển chuột, dấu con trỏ của chuột, thôi kích hoạt các chương trình xử lý đã cài đặt.

*c) Xác định loại chuột:* chức năng **AH = 24h**

- Thông số vào: AX = 0024h.
- Thông số ra: BH = số lớn của thế hệ
  BL = số nhỏ của thế hệ
  CH = kiểu chuột (01- chuột song song, 02 - chuột nối tiếp, 04h - chuột InPort, 08h - chuột PS/2, 10h - chuột HP).
  CL = số hiệu ngắt cứng IRQ (01h- PS/2, còn giá trị khác là PC).

## 2. Các chức năng về trạng thái chuột

*a) Đọc trạng thái :* chức năng **AH = 03h**

Xác định trạng thái trong thông số ra :

- Thanh ghi BX chỉ các phím ấn:
  + Phím trái ($d_0$ =1)
  + Phím phải ($d_1$ =1)
  + Phím giữa ($d_2$ =1).
- Thanh ghi CX = tọa độ ngang (X) của màn hình ảo của chuột.
- Thanh ghi DX = tọa độ dọc (Y) của màn hình ảo của chuột.

*b) Cất trạng thái chuột :* chức năng **AH = 16h**

Chức năng này cất trạng thái chuột vào vùng đệm của chương trình gọi, có địa chỉ là giá trị của ES:DX.

*c) Khôi phục lại trạng thái cũ của chuột*: chức năng **AH = 17h**

Chức năng này ngược với chức năng 16h, tức đọc lại trạng thái đã cất ở địa chỉ ES:DX.

*d) Đọc kích thước vùng đệm ghi trạng thái chuột* : chức năng **AH = 15h**

Kích thước vùng đệm đọc được ở thông số ra, trên thanh ghi BX, ra số byte nhớ.

*e) Đặt tần số ngắt cứng của chuột* : chức năng **AH = 1Ch**

Chức năng này đặt tần số để phần cứng của chuột đọc vị trí hiện thời và trạng thái các phím ấn của chuột để truyền các tham số này cho chương trình điều khiển chuột. Cần đặt tần số ngắt vào thanh ghi BX với các bit có giá trị sau:

$d_0$ =1 - không ngắt; $d_1$ =1 cho 30 ngắt/giây; $d_2$ =1 cho 50 ngắt/giây; $d_3$ =1 cho 100 ngắt/giây và $d_4$=1 cho 200 ngắt/giây.

## 3. Các chức năng về con trỏ chuột

*a) Trang màn hình chứa chuột*

- **Đặt trang màn hình:** *chức năng AH = 1Dh*

Số hiệu của trang trong thanh ghi BX.

- **Đọc trang màn hình:** *chức năng AH = 1Eh*

Số hiệu của trang đọc ra ở thanh ghi BX.

*b) Xác định hình dáng con trỏ*

Hình dáng con trỏ có thể thay đổi tùy chương trình điều khiển, thậm chí có thể thay đổi hình dáng ngay trong chương trình ứng dụng. Tùy chế độ màn hình (văn bản, đồ họa) và loại con trỏ (cứng, mềm) ta có hình dáng và kích thước con trỏ khác nhau.

Các chương trình xử lý văn bản thường cho hiện con trỏ dưới dạng một hình khối, giống như là con trỏ văn bản. Đối với con trỏ cứng của chế độ văn bản, chương trình ứng dụng chỉ có thể thay đổi dòng bắt đầu và dòng kết thúc của con trỏ. Kích thước của con trỏ phụ thuộc ma trận ký tự hiện thời và chế độ màn hình.

Kỹ thuật tạo con trỏ mềm (bằng chương trình) khá phức tạp, phải sử dụng:

- Hai byte ký tự và thuộc tính trong RAM màn hình.
- Thanh ghi che màn hình (Screen Mask) có 16 bit.
- Thanh ghi che con trỏ (Cursor Mask) có 16 bit.

Chương trình điều khiển chuột phải xác định lại hình dạng con trỏ mỗi khi con trỏ thay đổi vị trí trên màn hình. Quá trình định dạng con trỏ kết hợp các yếu tố trên theo các bước sau:

- Mã ký tự và byte thuộc tính kết hợp với Screen Mask qua phép tính AND logic.
- Kết quả kết hợp trên kết hợp với Cursor Mask cho hiện lên màn hình.

Có 4 khả năng chính của dạng con trỏ là:

- Con trỏ là một ký tự đặc biệt với một mẫu đặc biệt.
- Con trỏ là một ký tự đặc biệt, nhưng mẫu thay đổi khi con trỏ đè lên ký tự (ví dụ, đảo ngược).
- Con trỏ là một ký tự đặc biệt, nhưng mẫu thay đổi khi con trỏ đè lên ký tự.
- Con trỏ là một ký tự đặc biệt, nhưng mẫu ký tự chuyển thành một biến thể khi con trỏ đè lên ký tự.

+ **Con trỏ trong chế độ đồ họa:** *chức năng AH = 09h.*

Có một bảng trong nhớ RAM gồm 64 byte (32 byte đầu được AND và 32 byte sau được OR với mẫu của con trỏ hiện thời.

Cần phải đưa vào:

- Khoảng cách từ điểm làm chuẩn tới mép bên trái của bảng vào thanh ghi BX.
- Khoảng cách từ điểm làm chuẩn tới mép bên phải của bảng vào thanh ghi CX.

+ **Con trỏ trong chế độ văn bản**: *chức năng AH = 0Ah.*

Còn phải đưa vào:

- Kiểu con trỏ vào thanh ghi BX (kiểu mềm - giá trị 0, kiểu cứng - giá trị 1).
- Mặt nạ AND (con trỏ mềm) hoặc dòng bắt đầu (con trỏ cứng) vào thanh ghi CX.
- Mặt nạ XOR (con trỏ mềm) hoặc dòng kết thúc (con trỏ cứng) vào thanh ghi DX.

***c) Hiện /dấu con trỏ***

+ **Hiện con trỏ**: *chức năng AH = 01h.*

Con trỏ hiện lên màn hình khi bộ đếm trong có giá trị 0. Chương trình điều khiển chuột theo dõi chuyển động của chuột ngay cả khi chuột không hiện lên màn hình, nên chức năng này chỉ thực hiện việc hiện lên màn hình tại vị trí của thời điểm hiện tại.

+ **Dấu con trỏ**: *chức năng AH = 02h.*

Con trỏ không hiện lên màn hình khi bộ đếm trong có giá trị -1. Khi gọi chức năng này, con trỏ không hiện lên màn hình, nhưng chương trình điều khiển vẫn theo dõi vị trí chuột. Số lần gọi chức năng hiện (01h) và dấu (02h) chuột phải đảm bảo một tỷ lệ như nhau. Thường gọi 00h và 01h ở đầu chương trình, còn 02h ở cuối chương trình. Nếu chương trình ghi trực tiếp RAM màn hình, các hàm trên phải gọi nhiều hơn vì :

- Con trỏ chuột sẽ biến mất nếu bị ghi đè bởi một ký tự khác.
- Ký tự sẽ khôi phục khi con trỏ rời sang vị trí mới.

***d) Vị trí con trỏ***

Đơn vị đo trong xác định vị trí chuột là Mickey (tên chú chuột trong phim) với giá trị: 1 Mickey = 1/ 200 inch. Các tọa độ ở đây được tính với màn hình ảo.

Chức năng 00h đặt phạm vi di chuyển con trỏ trong phạm vi cả màn hình, chức năng 04h xác định vị trí mới, còn 07h (giới hạn dưới) và 08h (giới hạn trên) đặt giới hạn di chuyển con trỏ. Vị trí con trỏ có thể đọc được cùng với trạng thái phím bằng chức năng 03h.

- Xác định vị trí mới: *chức năng AH = 04h.*

Cần đưa vào tọa độ X (trong màn hình ảo) vào thanh ghi CX và tọa độ Y vào thanh ghi DX. Chức năng này cho phép chuyển vị trí con trỏ không cần di chuyển chuột.

- Xác định giới hạn di chuyển

+ **Theo tọa độ X**: *chức năng 07h.* Cần ghi tọa độ cực tiểu vào thanh ghi CX và tọa độ cực đại vào thanh ghi DX.

+ **Theo tọa độ Y**: *chức năng 08h.* Cần ghi tọa độ cực tiểu vào thanh ghi CX và tọa độ cực đại vào thanh ghi DX.

- Đặt vùng cấm của chuột: *chức năng AH = 10h.*

Đặt vùng cấm chuột di chuyển tới trong cửa sổ có kích thước xác định bởi:

+ Tọa độ góc cao bên trái (CX = tọa độ X, DX = tọa độ Y).

+ Tọa độ góc thấp bên phải (SI = tọa độ X, DI = tọa độ Y).

Khi con trỏ lọt ra ngoài cửa sổ trên sẽ bị biến mất. Có thể bỏ vùng cấm bằng gọi chức năng 01h (hiện con trỏ) hay 00h (khởi tạo con trỏ).

- Xác định khoảng cách dịch chuyển: *chức năng 0Bh*.

Đọc được khoảng cách giữa vị trí hiện thời của chuột và vị trí gọi chức năng này lần trước. Thanh ghi CX cho khoảng cách ngang và DX cho khoảng cách dọc.

## 4. Phím ấn

*a) Xác định số lần ấn một phím:* chức năng **05h**

Cần xác định đọc phím tại thanh ghi BX với phím trái (bit $d_0 = 1$), phím phải (bit $d_1 = 1$) hay phím giữa (bit $d_2 = 2$). Thông số ra cho biết:

- Trạng thái các phím của chuột tại thanh ghi AX giống như các bit của thanh ghi BX.
- Số lần phím được ấn từ khi bắt đầu (xóa về 0) gọi chức năng này lần trước đó tại BX.
- Tọa độ X của phím tại thanh ghi CX trong màn hình ảo.
- Tọa độ Y của phím tại thanh ghi DX trong màn hình ảo.

*b) Xác định số lần nhả một phím:* chức năng **06h**

Chức năng này cho biết một phím đã được nhả bao nhiêu lần kể từ khi gọi chức năng này lần cuối, nó cũng cho biết vị trí con trỏ tại thời điểm phím được nhả lần cuối.

- Thông số vào: AX = 0006h

  BX = số hiệu phím (0 - phím trái, 1- phím phải, 2- phím giữa).

- Thông số ra: AX = trạng thái tất cả các phím ấn của chuột:

  = 01 - phím trái bị ấn

  = 02 - phím phải bị ấn

  = 04 - phím giữa bị ấn

  BX = số lần phím được nhả kể từ lần nhả trước.

  CX = tọa độ ngang X của con trỏ tại thời điểm phím được nhả lần cuối.

  DX = tọa độ dọc Y của con trỏ tại thời điểm phím được nhả lần cuối.

## 5. Tốc độ di chuyển của con trỏ chuột

*a) Đặt sự tương quan giữa mickey và điểm ảnh:* chức năng **0Fh**.

Cần đưa vào các thanh ghi:

- CX = số lượng mickey tương ứng với 8 điểm (chiều ngang).
- DX = số lượng mickey tương ứng với 8 điểm (chiều dọc). Trước khi đặt (sau khi gọi chức năng 00h) giá trị ngầm định là 8 mickey ngang và 16 mickey dọc.

*b) Đặt ngưỡng tăng gấp đôi tốc độ con trỏ chuột*: chức năng **13h**.

Khi tốc độ con trỏ chuột vượt quá một ngưỡng (đặt vào DX) thì tốc độ di chuyển con trỏ chuột sẽ tăng gấp đôi. Muốn cấm sự tăng gấp đôi này, chỉ cần đặt ngưỡng thật cao.

*c) Đặt độ nhậy của con trỏ chuột:* chức năng **1Ah**.

Chức năng này là sự kết hợp của các chức năng 0Fh và 13h, nó cho phép đặt sự tương ứng giữa sự di chuyển của chuột và sự di chuyển của con trỏ chuột và đặt ngưỡng tăng gấp đôi. Cũng phải ghi các số liệu vào các thanh ghi BX, CX, DX như các chức năng 0Fh và 13h.

*d) Đọc độ nhậy của chuột:* chức năng **1Bh**.

Chức năng này ngược với chức năng 1Ah, tức đọc ra các giá trị đã đặt vào cho độ nhậy của chuột. Các thanh ghi BX, CX, DX cũng ghi các giá trị như các chức năng trên.

## 6. Cài đặt chương trình xử lý sự kiện

Chuột được di chuyển tới vị trí mong muốn, nơi có ký tự, dòng ký tự của thực đơn hay biểu tượng trên màn hình và được ấn phím. Chương trình điều khiển chuột sẽ đọc vị trí và trạng thái của phím này (bằng chức năng 03h) để biết vị trí và phím nào được ấn. Từ đó, chương trình điều khiển chuột sẽ chuyển tới lệnh đầu tiên của chương trình xử lý sự kiện tương ứng trong bộ nhớ để thực hiện. Như vậy, phím ấn của chuột gây ra một ngắt chương trình cho vi xử lý, còn ngắt nào còn phụ thuộc vào vị trí của chuột trên màn hình (tùy hành động mà ta muốn chuột gọi chương trình xử lý ngắt tương ứng với vị trí của chuột). Muốn vậy, ta phải biết cách cài đặt chương trình xử lý sự kiện hay xử lý ngắt tại vị trí địa chỉ nhớ mong muốn. Sau đây là các chức năng cài đặt đó.

***a) Chức năng AH = 0Ch***: cài đặt chương trình xử lý chuột

Cần ghi vào các thanh ghi ES:DX = địa chỉ chương trình xử lý của các bit chỉ sự thay đổi vị trí chuột ($d_0$), chỉ phím trái bị ấn ($d_1$) hay được nhả ($d_2$), phím phải bị ấn ($d_3$) hay được nhả ($d_4$), chỉ phím giữa được ấn ($d_5$) hay được nhả ($d_6$).

***b) Chức năng AH = 14h***: thay chương trình xử lý chuột

Chức năng này cài đặt một chương trình xử lý mới (giống như chức năng 0Ch) nhưng còn giữ lại tin về chương trình xử lý cũ (CX = sự kiện cũ và ES:DX = địa chỉ chương trình xử lý cũ).

***c) Chức năng AH = 18h***: cài đặt chương trình xử lý sự kiện thay thế

Khác với chức năng 14h và 1Ch, chức năng này cho phép cài đặt tới 3 chương trình xử lý khác nhau, các chương trình này có thể được gọi khi xẩy ra một sự kiện liên quan đến chuột hay bàn phím (các phím SHIFT, CTRL và ALT).

Thanh ghi CX, cũng có các bit ghi các sự kiện để gọi các chương trình xử lý (các bit $d_0$-$d_4$ giống chức năng 14h, phím SHIFT bị tác động-bit $d_5$, phím CTRL-bit $d_6$ và phím ALT-bit $d_7$. Thông số ra AX cho biết chương trình dã được cài đặt (= 0018h) hay không (= FFFFh).

***d) Chức năng AH = 19h***: xác định địa chỉ của chương trình xử lý sự kiện thay thế

Thanh ghi CX cũng ghi các sự kiện mà chương trình xử lý (như cài đặt AH = 18h). Thông số ra ES:DX cho biết địa chỉ chương trình xử lý, còn CX cho biết lỗi (= 0000h) hay không (!= 0).

## 5.4.3. LẬP TRÌNH CHO CHUỘT

Lập trình cho chuột tức viết chương trình thực hiện các nhóm chức năng đã phân loại theo hành động của chuột như ở trên: kích hoạt và xác định loại chuột, đọc và cất trạng thái, hành động liên quan tới con trỏ, tới phím ấn, tới tốc độ và đặc biệt quan trọng là cài đặt chương trình xử lý phím ấn của chuột. Chúng ta sẽ lập trình thông qua các ví dụ.

### 1. Kích hoạt và xác định loại chuột

*Ví dụ 1:*

```
XOR AX, AX      ; xóa thanh ghi AX
XOR BX, BX      ; xóa thanh ghi BX
XOR CX, CX      ; xóa thanh ghi CX
XOR DX, DX      ; xóa thanh ghi DX
MOV AH, 00h     ; kích hoạt chuột
INT 33h         ; gọi ngắt
MOV AH, 01h     ; hiện chuột
INT 33h
MOV AH, 02h     ; dấu chuột
```

```
INT 33h
MOV AH, 01h        ; hiện chuột
INT 33h
MOV AH, 1Fh        ; hủy kích hoạt
INT 33h
MOV AH, 01h        ; hiện chuột
INT 33h
MOV AH, 20h        ; kích hoạt lại
INT 33h
MOV AH, 24h        ; xác định loại chuột
INT 33h
MOV AH, 21h        ; khởi tạo lại chuột, dấu con trỏ và thôi cài đặt các
INT 33h            ; chương trình xử lý
INT 20h            ; kết thúc chương trình.
```

*Chú ý:* Cần cho chương trình chạy từng bước trong DEBUG, với lệnh **P** để biết tác dụng của từng chức năng.

## 2. Trạng thái chuột

*Ví dụ 2:*

```
XOR AX, AX         ; xóa thanh ghi AX
XOR BX, BX         ; xóa thanh ghi BX
XOR CX, CX         ; xóa thanh ghi CX
XOR DX, DX         ; xóa thanh ghi DX
MOV AX, 00h        ; khởi tạo chuột
INT 33h            ; gọi ngắt
MOV AX, 01h        ; hiện con trỏ chuột
INT 33h            ; gọi ngắt
MOV AX, 03h        ; đọc trạng thái của con trỏ chuột
INT 33h            ; gọi ngắt
MOV AX, 15h        ; xác định kích thước vùng đệm của con trỏ chuột
INT 33             ; gọi ngắt
MOV AX, 16h        ; cất trạng thái chuột
MOV CX, 132Eh      ; gán CX = 132Eh
MOV ES, CX         ; chuyển cho ES
MOV DX, BX         ; chuyển BX cho DX
INT 33h            ; gọi ngắt
MOV AX, 17h        ; lấy lại trạng thái của chuột
MOV CX, 1500h      ; chuyển CX = 1500H
MOV ES, CX         ; chuyển cho ES
MOV DX, 0120h      ; chuyển 0120h cho DX
INT 33h            ; gọi ngắt
MOV AX, 03h        ; đọc trạng thái chuột
INT 33h            ; gọi ngắt
MOV AX, 02h        ; dấu con trỏ chuột
INT 33h            ; gọi ngắt
INT 20h            ; kết thúc chương trình.
```

## 3. Vị trí con trỏ chuột

*Ví dụ 3:*

```
XOR AX, AX          ; xóa thanh ghi AX
XOR BX, BX          ; xóa thanh ghi BX
XOR CX, CX          ; xóa thanh ghi CX
XOR DX, DX          ; xóa thanh ghi DX
MOV AH, 00h         ; kích hoạt chuột
INT 33h             ; gọi ngắt
MOV AH, 01h         ; hiện con trỏ chuột
INT 33h             ; gọi ngắt
MOV AH, 03h         ; đọc trạng thái con trỏ chuột để bit vị trí con trỏ chuột
INT 33h             ; gọi ngắt
MOV AH, 04h         ; xác định vị trí mới của con trỏ chuột
MOV CX, 0278h       ; tọa độ cột cuối phải
MOV DX, 0000h       ; tọa độ dòng đầu
INT 33h             ; gọi ngắt để đọc vị trí mới
MOV AH, 03h         ; đọc trạng thái con trỏ chuột
INT 33h             ; gọi ngắt
MOV AH, 07h         ; hạn chế di chuyển con trỏ theo dòng
MOV CX, 0050h       ; hạn chế dưới
MOV DX, 0100h       ; hạn chế trên
INT 33h             ; gọi ngắt
MOV AH, 08h         ; hạn chế của di chuyển con trỏ theo cột
MOV CX, 0050h       ; hạn chế dưới
MOV DX, 00A0h       ; hạn chế trên
INT 33h             ; gọi ngắt
MOV AH, 03h         ; đọc trạng thái
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ tới điểm A
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ tới điểm B
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ tới điểm C
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ tới điểm D
MOV AH, 0Bh         ; đọc kích thước di chuyển con trỏ chuột
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ từ A tới B
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ từ B tới C
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ từ A tới C
INT 33h             ; gọi ngắt sau khi di chuyển con trỏ từ A tới D
MOV AH, 03h         ; dấu con trỏ
INT 33h             ; gọi ngắt
INT 20h             ; kết thúc chương trình.
```

*Chú ý:* A, B, C, D là 4 điểm giới hạn ở 4 góc của hình chữ nhật mà con trỏ chuột được phép di chuyển.

## 4. Hình dáng con trỏ

*Ví dụ 4:*

```
XOR AX, AX          ; xóa thanh ghi AX
XOR BX, BX          ; xóa thanh ghi BX
```

```
XOR CX, CX          ; xóa thanh ghi CX
XOR DX, DX          ; xóa thanh ghi DX
MOV AH, 01h         ; hiện con trỏ chuột
INT 33h             ; gọi ngắt
MOV AH, 03h         ; đọc trạng thái con trỏ chuột để bit vị trí con trỏ chuột
INT 33h             ; gọi ngắt
MOV AH, 0Ah         ; thay dạng con trỏ
MOV BX, 0000h       ; kiểu con trỏ mềm
MOV CX, 00AAh       ; mặt nạ AND đối với con trỏ mềm
MOV DX, 0055h       ; mặt nạ XOR đối với con trỏ mềm
INT 33h             ; gọi ngắt
MOV AH, 0Ah         ; thay dạng con trỏ
MOV BX, 0001h       ; kiểu con trỏ cứng
MOV CX, 00AAh       ; dòng bắt đầu của con trỏ cứng
MOV DX, 0055h       ; dòng kết thúc của con trỏ cứng
INT 33h             ;  gọi ngắt
MOV AH, 00h         ; đổi chế độ màn hình
MOV AL, 04h         ; chế độ 4 đồ họa
INT 10h             ; gọi ngắt để xem dạng con trỏ trong chế độ đồ thị
MOV AH, 00h         ; đổi chế độ màn hình
MOV AL, 00h         ; chế độ 0 cho văn bản
INT 10h             ; gọi ngắt cho  màn hình
MOV AH, 09h         ; con trỏ trong chế độ đồ thị
MOV BX, 0002h       ; khoảng cách từ điểm chuẩn tới mép trái của bảng
MOV CX, 0030h       ; khoảng cách từ điểm chuẩn tới mép phải của bảng
INT 33h             ; gọi ngắt.
```

## 5. Cài đặt chương trình xử lý khi dùng chuột

*Ví dụ 5:*

```
MOV AH, 00h         ; kích hoạt chuột
INT 33h             ; gọi ngắt
MOV AH, 01h         ; hiện con trỏ chuột
INT 33h             ; gọi ngắt
MOV AX, 000Ch       ; chức năng cài đặt chương trình xử lý chuột
MOV CX, 0012h       ; đặt ấn phím trái,  00010010B
MOV DX, 011Eh       ; offset của chương trình cần thực hiện
MOV AX, 132Eh       ; ES=132E của đoạn của chương trình xử lý
MOV ES, AX          ; gửi cho ES
INT 33h             ; gọi ngắt
NOP                 ; không hành động gì
132E:011E MOV AH, 00h   ; chức năng xác định chế độ màn hình
MOV AL, 04h         ; chế độ đồ họa
INT 10h             ; gọi ngắt
INT 20h             ; kết thúc chương trình.
```

*Chú ý:*

- Cho chương trình chạy trong DEBUG với lệnh P↵ tới lệnh INT 33h trước lệnh NOP, ấn phím trái của chuột, sẽ thực hiện chương trình kích hoạt có địa chỉ 132E:011E, đó là chương trình đổi chế độ màn hình.

- Cho chương trình chạy trong DEBUG với lệnh G↵, chương trình sẽ chờ ấn phím trái để kích hoạt chương trình xử lý chuột đổi chế độ màn hình.

- Có thể thay giá trị CX bằng các giá trị khác như đã kê ở trên để ấn hoặc nhả một trong các phím khác(phải, trái) hoặc di chuyển chuột để kích hoạt chương trình xử lý.

- Nếu đặt chương trình xử lý chuột thường trú (TSR) trong bộ nhớ (bằng các ngắt INT 27h hay chức năng 31h của ngắt INT 21h), sau khi chạy chương trình đặt kích hoạt chuột, có thể kích hoạt chương trình xử lý tại các thời điểm bất kỳ.

## CÂU HỎI

1. Cấu trúc, nguyên tắc hoạt động và các loại bàn phím.
2. Nguyên tắc tạo mã và các loại mã của bàn phím.
3. Cấu trúc của hệ thống màn hình.
4. Cấu trúc và phân loại bìa màn hình cho máy vi tính.
5. Các loại chế độ màn hình.
6. Nguyên tắc của việc hiện ký tự hay điểm ảnh ra màn hình.
7. Nguyên tắc của việc ghi và lập trình trực tiếp ký tự hay điểm ảnh cho màn hình.
8. Cấu tạo và nguyên tắc hoạt động của chuột.
9. Nguyên tắc của việc xác định hình dáng, vị trí con trỏ trên màn hình.
10. Nguyên tắc của sự kích hoạt chương trình xử lý chuột.

## BÀI TẬP

1. Bằng phương pháp lập trình trực tiếp, viết và cho chạy chương trình để:
   - Kiểm tra trạng thái của bàn phím.
   - Đọc một ký tự đã nhập từ bàn phím.
   - Ghi một ký tự vào bộ đệm của bàn phím rồi đọc trở lại.
   - Điều khiển bật tắt một số đèn chỉ thị liên quan tới bàn phím như NumLock, Cap Lock, Scroll Lock và kiểm tra chúng bằng mắt và bằng chương trình.

2. Dùng ngắt INT 16h của BIOS, viết và cho chạy chương trình kiểm tra ký tự sẵn sàng.

3. Dùng ngắt INT 16h của BIOS, viết và cho chạy chương trình kiểm tra trạng thái của bàn phím và cho hiển thị lên màn hình kết quả của kiểm tra.

4. Dùng ngắt INT 16h của BIOS, viết và cho chạy chương trình kiểm tra sự nhập một ký tự từ bàn phím và đưa hiện mã của ký tự lên màn hình (dùng ngắt INT 21h của DOS).

5. Dùng ngắt INT 16h của BIOS, viết và cho chạy chương trình kiểm tra sự nhập một chuỗi ký tự từ bàn phím (ấn cả phím ký tự và điều khiển đơn và kép) và đưa hiện mã của ký tự lên màn hình (dùng ngắt INT 21h của DOS).

6. Viết và cho chạy chương trình nhận biết một số phím điều khiển đơn như F1, F2, F3, v.v... tới F12. Mỗi lần nhập phím, cho hiện lên màn hình tên của phím đó và cho thoát chương trình bằng cách ấn phím ESC.

7. Viết và cho chạy chương trình nhận biết một số phím điều khiển kép như CTRL-Fi (i=1-12), ALT- Fi (i=1-12) và SHIFT-Fi (i=1-12), ALT- ký tự. Mỗi lần nhập phím, cho hiện lên màn hình tên của phím đó và cho thoát chương trình bằng ấn phím ESC.

8. Viết và cho chạy chương trình nhận biết một số phím điều khiển như UP, DOWN, LEFT, RIGHT, NUL, DEL, END, HOME v.v...

9. Viết và thực hiện chương trình chờ nhập một chuỗi ký tự theo một trong hai trường hợp:

- Có đưa hiện lên màn hình.
- Không hiện lên màn hình.

10. Viết và cho chạy chương trình phân biệt phím ký tự, phím điều khiển và phím điều khiển kép.

11. Viết và thực hiện chương trình đổi chế độ màn hình bằng việc điều khiển bằng ấn phím điều khiển Fi với i = 1- 12 là các chỉ số chế độ tương ứng của màn hình. Kết thúc chương trình bằng phím ESC.

12. Viết và cho chạy chương trình biến đổi các chữ "a, b, c, d" v.v... thường thành "A, B, C, D" v.v... hoa.

13. Viết và cho chạy chương trình biến đổi các chữ "a, b, c, d, e, f, g, h" thành các chữ "h, g, f, d, e, c, b, a".

14. Viết và cho chạy chương trình kiểm tra loại bìa của màn hình, nếu là CGA, viết chương trình kiểm tra chế độ và dung lượng của bộ nhớ RAM của màn hình đang dùng.

15. Viết và cho chạy chương trình trực tiếp cho màn hình với bìa EGA với các thông số sau:
- 63h - offset của cổng điều khiển của màn hình.
- Địa chỉ bộ đệm như sau: + 3B4h - màn hình trắng đen
  + 3D4h - màn hình mầu.
- Địa chỉ bộ đệm cho dữ liệu mầu nền: + 3C0h - cho dữ liệu ghi
  + 3C1h - cho dữ liệu đọc.
- Bảng mầu có các bit rgRGB tương tự RGB của CGA.

16. Viết chương trình trực tiếp (không dùng ngắt) cho một dòng ký tự "abcd" lên giữa màn hình.

17. Viết và cho chạy chương trình khởi phát màn hình CGA có mầu nền mầu xanh, mầu viền đỏ và con trỏ đặt ở vị trí đầu tiên (góc trái trên) của màn hình.

18. Viết và cho chạy chương trình xóa màn hình và đưa con trỏ về vị trí đầu tiên của màn hình khi ấn phím CTRL-F10, còn ấn phím bất kỳ, đưa ký tự ra màn hình ở vị trí hiện tại.

19. Viết và cho chạy chương trình xác định cửa sổ của màn hình ở góc trái trên, góc trái dưới, góc phải trên, góc phải dưới và cửa sổ có kích thước 12D dòng 30D cột.

20. Viết và cho chạy chương trình in dòng ký tự "Chuẩn bị dùng màn hình" cỡ chữ 14x7n tại khoảng giữa màn hình.

21. Viết và cho chạy chương trình xử lý các phím điều khiển:

- HOME: chuyển con trỏ tới góc trái trên.

- END: chuyển con trỏ tới góc trái dưới.
- PGdn: chuyển con trỏ tới góc phải dưới.
- ESC: kết thúc chương trình.
- Các phím khác không làm gì cả.

22. Viết và cho chạy chương trình các phím chức năng sau:

- Phím mũi tên trái: con trỏ và ký tự dịch về phía trái một vị trí (nếu con trỏ không ở lề trái).

- Phím mũi tên phải: con trỏ và ký tự dịch về phía phải một vị trí (nếu con trỏ không ở lề phải).

- Phím mũi tên lên: con trỏ và ký tự dịch về lên một dòng (nếu con trỏ không ở mép trên cùng của màn hình).

- Phím mũi tên xuống: con trỏ và ký tự dịch về phía dưới một dòng (nếu con trỏ không ở mép dưới cùng của màn hình).

23. Viết và cho chạy chương trìnhphục vụ các phím Insert, DEL và ESC như sau:

- *Phím Insert :* để ghi đè ký tự mới lên ký tự cũ bằng cách :
  + Chuyển con trỏ và ký tự bên phải (trong dòng chứa con trỏ) sang phải một vị trí.
  + Xuất hiện một khoảng trống tại vị trí con trỏ trước đó.
  + Ký tự cuối cùng trên dòng cuối cùng của màn hình bị đẩy khỏi màn hình.
- *Phím DEL:* để xóa một ký tự bằng cách:
  + Chuyển các ký tự bên phải con trỏ sang trái một vị trí.
  + Một khoảng trống được điền vào cuối dòng.
- *Phím ESC:* kết thúc chương trình.

24. Viết và cho chạy chương trình in các chữ dấu cộng (+) thành dấu chữ thập ở giữa màn hình.

25.Viết và cho chạy chương trình in các chữ A viền khung và các đường chéo của màn hình.

26. Viết và cho chạy chương trình vẽ các điểm ảnh mầu đỏ viền màn hình mầu xanh và ghi dòng ký tự "Màn hình sẵn sàng" ở khoảng giữa màn hình.

27. Vẽ một tam giác cân bằng các điểm ảnh mầu nâu trên nền trắng có kích thước 200 pixel ở khoảng giữa màn hình.

28. Viết và thực hiện chương trình ghi một thực đơn ra góc trái trên của màn hình:

- Kích thước cửa sổ hình chữ nhật 100x 50 pixel.
- Có 5 chương trình phục vụ từ F1÷F5 sắp xếp từ trên xuống dưới.
- Khi trỏ vào từng chữ Fi và ấn phím trái của chuột, sẽ chuyển sang chương trình phục vụ i tương ứng với địa chỉ tại DS=1300H và offset = 0100hxi (i=1÷5).

29. Thay chương trình ở bài tập 27 bằng việc ấn phím phải của chuột, chương trình của các Fi được kích hoạt.

30. Viết chương trình đọc tọa độ, khoảng cách (ra pixell) và hiện kết quả lên màn hình của hai điểm ảnh bất kỳ trên màn hình.

*CHƯƠNG 6*

# LẬP TRÌNH CHO THIẾT BỊ NGOÀI GHÉP NỐI

Ngoài các thiết bị ngoài thông dụng (bàn phím, màn hình, chuột) đã xét ở chương 5, máy vi tính còn trao đổi tin với các thiết bị ngoài khác như máy in, gậy điều khiển trò chơi, thiết bị thông tin và các thiết bị ngoài ghép nối khác trong các hệ đo, hệ điều khiển kỹ thuật. Ở chương 5, chúng ta đã xét hai phương pháp lập trình cho các thiết bị ngoài thông dụng là phương pháp lập trình trực tiếp cho phần cứng và sử dụng các ngắt mềm INT nh của hệ điều hành (của BIOS và của DOS). Trước khi xét những lập trình trực tiếp và sử dụng ngắt cho hai loại trao đổi tin song song và nối tiếp, ta sẽ nêu những quy tắc chung cho các phương pháp lập trình. Về thứ tự trình bày, khác với chương trước, chúng ta sẽ xét lập trình trực tiếp trước vì nó gắn liền với phần cứng (thiết bị), giúp ta hiểu rõ phần cứng và các chương trình phục vụ thiết bị ngoài đó chính là các chương trình con phục vụ ngắt của hệ điều hành, lưu trú trong khối nhớ để mỗi lần gọi ngắt INT nh, các chương trình phục vụ ngắt này được gọi. Chúng ta có thể viết các chương trình con mới phục vụ các thiết bị ngoài ghép nối bằng cách lập trình trực tiếp, đặt lưu trú trong bộ nhớ, gán cho một mức ngắt mới (sẽ xét ở các chương sau) và chỉ cần gọi ngắt này mỗi khi cần điều khiển trao đổi tin với thiết bị ngoài.

## 6.1. QUY TẮC CHUNG VỀ LẬP TRÌNH CHO THIẾT BỊ NGOÀI

Trước khi xét quy tắc lập trình trực tiếp và sử dụng ngắt, chúng ta phải biết về phương pháp trao đổi tin của máy vi tính với thiết bị ngoài và cấu trúc chung của khối ghép nối điều khiển thiết bị ngoài.

### 6.1.1. TRAO ĐỔI TIN CỦA MÁY VI TÍNH VỚI THIẾT BỊ NGOÀI

Ngoài trao đổi tin với các thiết bị ngoài thuộc hệ máy vi tính (MVT) như khối nhớ, đĩa, loa, vi xử lý (VXL) còn trao đổi tin với các thiết bị ngoài khác thông qua các thanh ghi của các khối ghép nối điều khiển thiết bị ngoài.

#### 1. Trình tự trao đổi tin

Khi cần trao đổi tin với TBN, VXL thực hiện các thao tác thông qua các lệnh theo trình tự sau:

- VXL đưa ra địa chỉ thanh ghi đệm của khối ghép nối điều khiển TBN (hay địa chỉ cổng) trên đường dây địa chỉ $A_0 \div A_n$ để chọn vi mạch cổng tương ứng với loại tin (điều khiển, trạng thái, số liệu) cho TBN.

- Nếu máy vi tính đưa tin ra TBN, VXL ghi số liệu vào thanh ghi tích lũy (chứa) AX của VXL.

- VXL đưa lệnh ghi số liệu $\overline{WR}$ vào thanh ghi. Sau tín hiệu ghi này, dữ liệu trên đường dây $D_0 \div D_m$ sẽ được ghi vào thanh ghi đệm ghi số liệu của khối ghép nối điều khiển TBN.

Nếu VXL cần đọc dữ liệu của TBN, VXL phải đưa tín hiệu đọc $\overline{RD}$ sau khi đưa địa chỉ. Sau tín hiệu đọc này, dữ liệu trên thanh ghi đệm đọc (về trạng thái, về số liệu) sẽ đưa lên đường dây dữ liệu $D_0 \div D_n$ và ghi vào thanh ghi chứa AX.

## 2. Chế độ trao đổi tin

Có ba chế độ trao đổi tin là trao đổi trực tiếp (đồng bộ), hỏi vòng trạng thái và ngắt chương trình. Hai chế độ trên do máy vi tính (MVT) khởi xướng còn phương pháp thứ ba do TBN khởi xướng.

***a) Trao đổi tin trực tiếp***

Chế độ trao đổi này do MVT khởi xướng bằng đưa địa chỉ, dữ liệu và lệnh cho cổng, không cần biết TBN có yêu cầu hay đã sẵn sàng nhận tin chưa. Phương pháp trao đổi này nhanh vì không tốn thời gian kiểm tra sự sẵn sàng hay yêu cầu của TBN, hai quá trình đưa và nhận tin là đồng thời (đồng bộ), nhưng không tin cậy (có thể mất tin truyền vì TBN hay máy vi tính chưa sẵn sàng trao đổi tin).

***b) Trao đổi tin có kiểm tra trạng thái***

Trước khi tiến hành trao đổi tin (ghi tin ra, đọc tin vào) như phương pháp trực tiếp, VXL cần kiểm tra trạng thái sẵn sàng trao đổi tin của TBN. Muốn vậy, MVT phải:

- Đọc dữ liệu của thanh ghi trạng thái.

- Kiểm tra bit sẵn sàng trao đổi tin của TBN.

- Nếu bit trạng thái xác lập lên 1 tức TBN đã sẵn sàng hay có yêu cầu trao đổi tin, VXL thực hiện trao đổi tin như chế độ trao đổi tin trực tiếp, tức thực hiện lệnh trao đổi tin (đưa tin vào, đưa tin ra) nằm sau lệnh kiểm tra trạng thái trong chương trình.

- Nếu bit trạng thái vẫn bằng 0, tức TBN chưa sẵn sàng hoặc không có yêu cầu trao đổi tin, thì:

+ VXL chuyển sang kiểm tra trạng thái TBN khác nếu hệ MVT có nhiều TBN theo lần lượt một vòng kín (hỏi vòng trạng thái).

+ Quay trở lại đọc và kiểm tra trạng thái của TBN cần trao đổi tin này, tới khi bit trạng thái được xác lập để thực hiện lệnh trao đổi tin tiếp theo. Nói khác đi, máy vi tính chờ trạng thái sẵn sàng trao đổi tin của TBN.

Lập trình trực tiếp cho TBN dưới đây sẽ dùng phương pháp trao đổi tin hỏi trạng thái này.

Chế độ trao đổi tin này tin cậy vì biết chắc chắn TBN có yêu cầu, nhưng tốn thời gian thực hiện các lệnh đọc, kiểm tra và chờ bit trạng thái xác lập hoặc hỏi vòng các TBN khác.

***c) Chế độ trao đổi tin theo ngắt chương trình***

Chế độ trao đổi tin này khắc phục được việc tốn thời gian chờ hoặc hỏi vòng trạng thái. Sự trao đổi tin ở chế độ này do TBN khởi xướng bằng cách đưa yêu cầu trao đổi tin vào lối vào, yêu cầu ngắt chương trình INTR của VXL ngay trong khi VXL đang thực hiện một chương trình nào đó. Khi nhận được tín hiệu INTR, VXL sẽ:

– Làm thủ tục ngắt chương trình (nếu đang thực hiện một chương trình nào đó) bằng các lệnh PUSH các thanh ghi thông dụng, gửi nội dung các thanh ghi thông dụng của VXL vào các ô nhớ ngăn xếp (stack).

– Phát tín hiệu xác nhận ngắt INTA để tạo và đọc véctơ ngắt (địa chỉ chứa địa chỉ của lệnh bắt đầu chương trình con phục vụ trao đổi tin) từ khối ghép nối điều khiển TBN vào thanh ghi chứa AX của VXL.

– Chuyển sang thực hiện chương trình con phục vụ ngắt (có địa chỉ đầu tiên là véctơ ngắt) cho tới khi thực hiện lệnh RET (Return - trở về) hay IRET (trở về từ ngắt).

– Thực hiện lệnh RET, tức trở về chỗ bị ngắt của chương trình chính bằng cách hồi phục lại các giá trị các thanh ghi đã gửi vào các ô nhớ ngăn xếp (bằng lệnh POP các thanh ghi).

– Thực hiện tiếp chương trình chính bị ngắt.

Chế độ trao đổi tin này vừa tin cậy (có kiểm tra sự sẵn sàng của TBN) vừa không lãng phí thời gian chờ hoặc hỏi vòng trạng thái nhưng:

– Tốn thiết bị xử lý ưu tiên ngắt (ghi nhận các yêu cầu ngắt của TBN, sắp xếp ưu tiên, phát yêu cầu ngắt INTR và tạo véctơ ngắt) bằng vi mạch xử lý ngắt 8214 hoặc 8259A.

– Tốn thời gian thực hiện các lệnh chuẩn bị ngắt (PUSH các thanh ghi) và trở về chương trình chính bị ngắt (POP các thanh ghi).

Tuy nhiên, trong MVT PC - IBM cũng như các MVT khác đều dùng chế độ trao đổi tin này, chính các ngắt INT nh của BIOS và DOS của hệ điều hành MS-DOS là các chương trình con phục vụ ngắt. Khi có tín hiệu yêu cầu ngắt cứng do các TBN đưa vào lối vào yêu cầu ngắt INTR của VXL hay gặp lệnh ngắt mềm INT nh, VXL làm thủ tục ngắt chương trình rồi chuyển tới địa chỉ tương ứng với mức ngắt nh để chạy chương trình con phục vụ ngắt. Ngoài ra, còn có các ngoại trừ trong VXL (chia cho 0, tràn số liệu v.v...), VXL cũng thực hiện lệnh ngắt mềm tương ứng giống như lệnh ngắt mềm dành cho các chức năng của hệ điều hành.

### 3. Dạng tin trao đổi

Tin trao đổi giữa VXL và TBN có hai dạng song song và nối tiếp tùy thuộc dạng tin của TBN và khoảng cách giữa MVT và TBN.

***a) Dạng trao đổi tin song song***

Tin trao đổi được truyền song song, mỗi bit tin trên một đường dây riêng rẽ nên số đường dây bằng số bit của lời tin (8bit - theo byte hay 16 bit theo lời). Trao đổi tin song song này nhanh (cùng một thời điểm trao đổi nhiều bit tin) nhưng tốn đường dây và bị nhiễu nhiều khi TBN ở xa MVT.

***b) Dạng trao đổi tin nối tiếp***

Tin trao đổi được truyền nối tiếp từng bit một trên một hay hai đường dây truyền (có thể có đường phát Tx và đường thu Rx hoặc trên cùng một đường dây hai chiều). Ưu điểm của dạng trao đổi tin này là tiết kiệm được đường dây, tránh nhiễu, nhưng cần các khối ghép nối để biến đổi tin song song (từ MVT hay TBN song song) sang nối tiếp cho đường dây và biến đổi nối tiếp sang song song (cho MVT hoặc TBN song song). Các vi mạch 8251A, 8250, 8273, MC 6854 làm nhiệm vụ biến đổi song song-nối tiếp và nối tiếp song song này. Trao đổi tin nối tiếp còn có hai chế độ là không đồng bộ (thu và phát không cùng một nhịp) và đồng bộ (thu và phát cùng một nhịp).

## 6.1.2. KHỐI GHÉP NỐI ĐIỀU KHIỂN TRAO ĐỔI TIN

Giữa MVT và TBN phải lắp khối ghép nối (interface-giao diện, adaptor-mạch điều hợp) để điều khiển trao đổi tin song song hay nối tiếp. Trường hợp đơn giản nhất của khối ghép nối là một thanh ghi hay một cổng, nhưng nói chung khối ghép nối TBN gồm nhiều thanh ghi.

### 1. Cấu trúc khối ghép nối

***a) Khối ghép nối song song***

Khối ghép nối song song gồm các khối sau:

- Khối đệm đường dây cho các đường dây địa chỉ, dữ liệu và điều khiển để tăng công suất, cách ly VXL với TBN.

- Khối giải mã địa chỉ và lệnh cho các thanh ghi đệm.

- Các thanh ghi đệm điều khiển, trạng thái và dữ liệu. Trường hợp đơn giản nhất, các thanh ghi điều khiển và trạng thái (CSR) chung nhau, có cùng một địa chỉ, nhưng phân biệt ở lệnh ghi dữ liệu điều khiển (C- Control) và đọc trạng thái (S-Status). Với hai thanh ghi số liệu cũng vậy, có thể chung một thanh ghi với một địa chỉ, nhưng ghi số liệu ghi ra TBN bằng lệnh ghi, còn đọc số liệu từ TBN vào VXL bằng lệnh đọc.

Thanh ghi điều khiển có các bit xác định chế độ hoạt động (của vi mạch, của dạng trao đổi tin v.v...), các bit của lệnh điều khiển khối ghép nối hay đưa tín hiệu điều khiển TBN.

Thanh ghi trạng thái, ngoài các bit ghi trạng thái sẵn sàng trao đổi tin của TBN, còn có các bit phản ánh trạng thái của khối ghép nối điều khiển TBN hay trạng thái khác của TBN.

- Khối xử lý ngắt (ghi yêu cầu, che yêu cầu, xét ưu tiên, phát tín hiệu yêu cầu ngắt INTR, tạo véctơ ngắt khi nhận xác nhận ngắt INTA từ VXL).

Ngoài các thành phần thông dụng trên, tùy thuộc TBN, còn có các thành phần điều khiển TBN. Các mạch ghép nối này, tùy nhiệm vụ và mức độ phức tạp, được chế tạo dưới dạng vi mạch (8255A, 8251A, 8250 v.v...) hay dạng bìa (carte) gồm nhiều vi mạch khác nhau như bìa màn hình, bìa âm thanh, bìa mạng v.v...

***b) Khối ghép nối nối tiếp***

Ngoài các thành phần như khối ghép nối song song, khối ghép nối nối tiếp còn thêm:

- Các thanh ghi dịch để biến đổi tin song song - nối tiếp và nối tiếp-song song.

- Khối điều khiển modem (điều chế và giải điều chế) cho tin truyền xa trên đường dây điện thoại.

- Khối phát nhịp thời gian TxC, RxC cho tín hiệu phát và thu.

- Khối phát hiện các bit Start (bắt đầu) và Stop (kết thúc) của byte tin hay byte đồng bộ (Sync) hoặc cờ (flag) của khung tin.

- Khối kiểm tra lỗi cho truyền tin.

## 2. Hoạt động của khối ghép nối

Khối ghép nối hoạt động được theo trình tự sau:

***a) Khởi phát***

Bằng chương trình, khối ghép nối được ghi các thông tin về chế độ, lệnh điều khiển. Các thông tin này được ghi ở bộ nhớ đệm, có địa chỉ dành cho từng TBN.

***b) Đọc, kiểm tra và chờ trạng thái sẵn sàng trao đổi tin của TBN***

***c) Trao đổi tin với TBN***

Tin ở đây có thể là lệnh điều khiển TBN hay số liệu $D_0 \div D_n$.

Khi lập trình trực tiếp cho trao đổi tin với TBN, chúng ta phải tuân theo trình tự trên.

## 6.1.3. LẬP TRÌNH TRAO ĐỔI TIN VỚI THIẾT BỊ NGOÀI

Có hai cách lập trình, đó là lập trình trực tiếp và lập trình sử dụng ngắt của BIOS và DOS của hệ điều hành MS-DOS.

## 1. Lập trình trực tiếp

Lập trình trực tiếp là viết các lệnh dạng hợp ngữ để điều khiển khối ghép nối trao đổi tin giữa MVT và TBN. Lập trình loại này đòi hỏi người lập trình phải:

- Nắm vững cấu trúc và hoạt động của MVT, khối ghép nối, TBN ngoài.

- Phải biết rõ các địa chỉ các thanh ghi của khối ghép nối trong MVT. Trong MVT PC/AT-IBM có các địa chỉ vùng địa chỉ dành cho các TBN như bảng 6.1.

Trình tự các lệnh của chương trình trực tiếp theo trật tự hoạt động của khối ghép nối ở trên là: khởi phát khối ghép nối (ghi lời chế độ và lệnh vào thanh ghi điều khiển), đọc, kiểm tra và chờ trạng thái sẵn sàng, trao đổi số liệu khi có trạng thái sẵn sàng. Nếu đưa số liệu từ MVT ra TBN, ta phải đưa số liệu vào thanh ghi chứa AX trước rồi đưa ra khối ghép nối TBN, còn đọc số liệu từ TBN thì ngược lại, đưa lệnh đọc từ khối ghép nối vào AX trước rồi mới chuyển vào bộ nhớ.

***Bảng 6.1.*** **Các vùng địa chỉ dành cho các thiết bị ngoài trong PC**

| Loại thiết bị ngoài | Vùng địa chỉ |
|---|---|
| Bộ điều khiển thời gian | 040-05F |
| Bộ ghép nối song song (8255A) | 069-063 |
| Bàn phím (8042) | 060-06F |
| Đồng hồ thời gian thực (MC 146812) | 070-07F |
| Bộ điều khiển đĩa cứng | 1F0-1F8 |
| Cổng gậy điều khiển trò chơi | 200-207 |
| Khối ghép nối máy in song song thứ 2 | 278-27F |
| Bìa mạng | 360-36F |
| Khối ghép nối máy in song song thứ nhất | 378-37F |
| Khối ghép nối nối tiếp đồng bộ SDLC | 380-38F |
| Khối ghép nối nối tiếp thứ hai | 3A8-3AF |
| Bìa màn hình MDA | 3B0-3BF |
| Bìa màn hình EGA | 3C0-3CF |
| Bìa màn hình CGA | 3D0-3DF |
| Bộ điều khiển đĩa mềm | 3F0-3F7 |
| Khối ghép nối nối tiếp thứ nhất | 3F8-3FF |

Các lệnh hợp ngữ dùng để lập trình trực tiếp là:

- *Lệnh chuyển MOV A, B* để:

+ Gán địa chỉ cho cổng hay thanh ghi với A là DX, B là độ lệch (offset) địa chỉ của thanh ghi.

+ Gán số liệu đưa ra cho AX với A là AX, B là giá trị số liệu.

- *Lệnh IN C, D* để:

+ Đọc trạng thái từ thanh ghi trạng thái vào AX với C là AX, D là DX đã gán địa chỉ của thanh ghi trạng thái (có thể D là giá trị địa chỉ trực tiếp để bớt lệnh gán địa chỉ).

+ Đọc số liệu từ thanh ghi đệm số liệu vào AX với C là AX, D là DX đã gán địa chỉ của thanh ghi đệm số liệu (có thể D là giá trị địa chỉ trực tiếp để bớt lệnh gán địa chỉ).

- *Lệnh OUT E, F* để:

+ Đưa dữ liệu về điều khiển (chế độ, lệnh) từ AX ra thanh ghi điều khiển với E là DX đã gán địa chỉ hay giá trị địa chỉ trực tiếp của thanh ghi điều khiển và F là AX đã đưa số liệu về điều khiển.

+ Đưa số liệu từ AX ra thanh ghi đệm số liệu ra hay số liệu ghi với E là DX đã gán địa chỉ hay địa chỉ trực tiếp của thanh ghi đệm số liệu, F là AX đã đưa số liệu ra.

- *Lệnh chọn logic AND AX, dữ liệu*, để chắn hay lọc các bit về trạng thái, tức chỉ chừa lại các bit cần thiết của trạng thái cần xét.

- *Lệnh kiểm tra trạng thái* bằng một trong hai lệnh:

+ *TEST AX, dữ liệu*, để kiểm tra. Nếu các bit của trạng thái trùng với các bit của dữ liệu, sẽ có bit cờ ZF =1 (hiệu số AX - dữ liệu = 0, nhưng không gửi hiệu số vào AX).

+ *CMP AX, dữ liệu*, để kiểm tra. Nếu các bit của trạng thái trùng với các bit của dữ liệu, sẽ có bit cờ ZF =1 (hiệu số AX - dữ liệu = 0, gửi hiệu số vào AX).

- *Lệnh nhảy về lệnh đọc lại trạng thái* nếu trạng thái chưa sẵn sàng JNZ A (A là nhãn của lệnh đọc trạng thái trong chương trình dùng chương trình dịch hợp ngữ hay địa chỉ của lệnh đọc trạng thái nếu chạy chương trình bằng DEBUG của MS-DOS.

Ngoài các lệnh thông dụng trên, chương trình còn dùng các lệnh chuyển MOV để gán địa chỉ và gọi lại các thông số trong các bộ nhớ đệm dành cho các bộ ghép nối TBN, các lệnh xử lý số liệu đọc, số liệu ghi v.v...

## 2. Lập trình sử dụng ngắt INT nh

Lập trình sử dụng ngắt đơn giản và ngắn gọn hơn lập trình trực tiếp, không đòi hỏi người lập trình phải hiểu rõ phần thiết bị vì:

- Các chương trình con phục vụ ngắt chính là các chương trình trực tiếp đã viết sẵn cho khối ghép nối.

- Các số liệu phục vụ cho lập trình trực tiếp đã ghi sẵn trong các bộ nhớ đệm dành cho từng khối ghép nối TBN.

Lập trình sử dụng ngắt cần:

- Hiểu rõ các ngắt và các chức năng phụ dành cho khối ghép nối nào và làm gì.

- Ghi đầy đủ các thông số vào cho các thanh ghi thông dụng của VXL mà ngắt yêu cầu, trong đó AH = chức năng, AL = chức năng phụ (nếu có) hoặc mã ASCII của ký tự, CX là số lần hay tọa độ màn hình, chuột, DX = số hiệu TBN hay tọa độ, BX = số hiệu trang màn hình, bảng mầu v.v...

- Đọc các thông số ra để biết được các kết quả thực hiện ngắt hay chương trình phục vụ ngắt, trạng thái của khối ghép nối v.v...

Các lệnh thường sử dụng trong lập trình sử dụng ngắt là:

- *Lệnh MOV A, B* trong đó A thường là các thanh ghi, B là con số chỉ số hiệu ngắt hay chức năng chính và phụ, số hiệu của trang màn hình, tọa độ, số lần v.v...

- *INC A, DEC A* (tăng, giảm giá trị của thanh ghi A)

- *TEST AX, số liệu* hay *CMP AX, số liệu*, để so sánh giá trị của thanh ghi AX với số liệu.

- *JNZ B, JZ B, LOOP B* trong đó B là nhãn (dùng chương trình hợp dịch) hay địa chỉ (dùng DEBUG) của lệnh cần lặp lại.

- *INT nh* gọi số hiệu ngắt *nh* dành cho thiết bị ngoài tương ứng.

- *Lệnh kết thúc chương trình INT 20h* (trở về môi trường DEBUG) hay MOV AH, 4Ch và INT 21h (trở về môi trường DOS).

Dưới đây, chúng ta sẽ áp dụng hai cách lập trình trên cho các phương pháp trao đổi tin song song và nối tiếp với các TBN như máy in, gậy điều khiển trò chơi, thiết bị trao đổi tin nối tiếp và mạng.

## 6.2. LẬP TRÌNH CHO THIẾT BỊ TRAO ĐỔI TIN SONG SONG

### 6.2.1. VI MẠCH CỔNG SONG SONG CỦA MÁY PC

Họ máy vi tính cá nhân PC-IBM có các vi mạch cổng song song như sau:

- 8255A cho máy loại PC XT (VXL 8086/8088).

- 82345 (có thêm hai máy phát xung và bộ kiểm tra chẵn lẻ) cho các máy PC AT (VXL 80286) và PS/2 (VXL 82386).

Ta sẽ xét lập trình cho 8255A trong ví dụ với TBN ghép nối dưới đây, ở đây ta chỉ xét cổng vào-ra 82345.

Các cổng song song trong PC nói chung có tối đa bốn cổng, được ký hiệu LTP1, LPT2, LPT3, LPT4 với các địa chỉ cơ sở như bảng 6.1.

Cũng như mọi cổng song song, 82345 có ba thanh ghi:

- Thanh ghi điều khiển có địa chỉ là địa chỉ cơ sở + 2, ghi tin điều khiển cho khối ghép nối và TBN, có thể đọc lại vào MVT.

- Thanh ghi trạng thái có địa chỉ là địa chỉ cơ sở + 1, chỉ đọc vào tin về trạng thái của khối ghép nối và TBN.

- Thanh ghi số liệu 8 bit, có địa chỉ là địa chỉ cơ sở. Thanh ghi này được ghi cho số liệu ra TBN và được đọc cho số liệu vào từ TBN.

Các bit của thanh ghi điều khiển như sau:

- $d_0 = 1$ : tín hiệu Strobe để điều khiển ghi số liệu ra cho TBN.
- $d_1 = 1$ : điều khiển tự động xuống dòng sau mỗi dòng.
- $d_2 = 0$ : điều khiển khởi phát TBN.
- $d_3 = 1$ : lựa chọn TBN.
- $d_4 = 1$ : cho phép TBN đưa tín hiệu yêu cầu trao đổi tin.
- $d_5$ : bit điều khiển chiều số liệu: = 0 - đưa số liệu ra; = 1 - đưa số liệu vào.
- $d_6 \div d_7$ : không dùng.

Các bit của thanh ghi trạng thái như sau:

- $d_0 \div d_1$ : không dùng.
- $d_2 = 0$ : TBN đã nhận số liệu.
- $d_3 = 0$ : TBN bị lỗi.
- $d_4 = 1$ : TBN đã được lựa chọn.
- $d_5 = 1$ : TBN bị lỗi.
- $d_6 = 0$ : TBN sẵn sàng hoạt động.
- $d_7 = 0$ : TBN bị bận.

Cổng song song nối với TBN qua ổ cắm DB-25 (25 chân) theo các tín hiệu sau:

+ Số 1: tín hiệu STROBE để ghi số liệu ra.
+ Số 2 ÷ 9 : các số liệu $d_0 \div d_7$.
+ Số 10 : xác nhận của TBN.
+ Số 11 : bận của TBN.
+ Số 12 : lỗi (ví dụ hết giấy của máy in).
+ Số 13 : TBN được lựa chọn.
+ Số 14 : điều khiển (tự động xuống dòng cho máy in).
+ Số 15 : lỗi của TBN.
+ Số 16 : khởi phát TBN.
+ Số 17 : điều khiển chọn lối vào.
+ Số 18÷25 : nối đất.

## 6.2.2. LẬP TRÌNH TRỰC TIẾP CHO MÁY IN SONG SONG

Máy in song song nối với cổng vào/ra song song LPT1, địa chỉ 378h hoặc 278h (LPT2) qua ổ cắm DB-25.

Lập trình trực tiếp cho máy in, tức là thực hiện các hành động sau:

- Khởi phát máy in.
- Đưa một ký tự ra máy in sau khi đã kiểm tra trạng thái sẵn sàng của máy in.

### 1. Khởi phát máy in

Khởi phát máy in là ghi các bit thích hợp cho cổng và máy in để chúng có thể sẵn sàng hoạt động và nhận tin vào.

Các bit cho cổng: cho phép ngắt ($d_4 = 1$) và đưa số liệu ra ($d_5 = 0$).

Các bit cho máy in: xuống dòng sau mỗi dòng ($d_1 = 1$), khởi phát ($d_2 = 0$), lựa chọn lối vào ($d_3 = 1$).

Như vậy lời điều khiển sẽ là 00011010B = 8Ah. Lời điều khiển này phải được:

- Chuyển vào thanh ghi chứa AX bởi lệnh MOV AX, 8Ah
- Ghi vào thanh ghi điều khiển (địa chỉ = địa chỉ cơ sở + 2) bởi lệnh OUT DX, AX.

Để đảm bảo độ tin cậy, sau lệnh khởi phát trên, phải có độ trễ khoảng 1/20 giây bằng thực hiện chương trình trễ tương ứng.

Chương trình cho máy in LPT1, chạy với DEBUG như sau:

```
        MOV DX, 0378        ; không có h sau số hexadecimal của địa chỉ
        ADD DX, 2           ; DX chứa địa chỉ của thanh ghi điều khiển
        MOV AL, 8A          ; lời điều khiển 8Ah gửi vào AL để đưa ra máy in
        OUT DX, AL          ; ghi lời điều khiển ra thanh ghi điều khiển
        CALL DELAY          ; gọi trễ
        MOV AL, 00001100B ; lời điều khiển 0Ch (các bit về các giá trị ngầm định)
        OUT DX, AL          ; ghi lời điều khiển ra thanh ghi điều khiển
        CALL DELAY          ; gọi trễ
        INT 20              ; kết thúc chương trình
DELAY:  MOV AH, 00          ; chức năng đọc bộ đếm đồng hồ
```

```
        INT 1A          ; ngắt phục vụ đếm đồng hồ
        ADD DX, 2       ; từ thấp của bộ đếm + 2
        MOV BX, DX      ; gửi từ thấp đếm +2 vào BX
A:      INT 1A          ; đọc bộ đếm mới
        CMP DX, BX      ; so sánh với bộ đếm cuối
        JB A            ; quay trở về A để đọc số đếm và so sánh
        RET             ; trở về từ chương trình con.
```

Khi chạy chương trình trên bằng DEBUG ta phải:

- Bỏ các nhãn Delay và A trước các lệnh.
- Thay địa chỉ thích hợp của các nhãn trên vào các lệnh có nhãn.

## 2. Chương trình đưa ký tự ra máy in

Sau khi khởi phát máy in, máy vi tính chạy chương trình đưa ký tự ra máy in sau khi đã kiểm tra trạng thái sẵn sàng nhận ký tự. Chương trình đưa ký tự ra máy in gồm ba giai đoạn:

- Chuyển và ghi số liệu ra thanh ghi đệm số liệu của cổng (địa chỉ cơ sở).
- Kiểm tra trạng thái và đưa ký tự ra. Nếu trạng thái máy in chưa sẵn sàng, chương trình phải đợi bằng cách quay trở về các lệnh đọc và kiểm tra cho tới khi có trạng thái sẵn sàng.
- Đưa bit $d_0 = 1$ vào thanh ghi điều khiển để đưa tín hiệu STROBE cho máy in, để ghi số liệu ra máy in.

```
        MOV AL, Mã          ; chuyển mã vào thanh ghi AL
        MOV DX, 378         ; chuyển địa chỉ vào thanh ghi DX
        OUT DX, AL          ; đưa mã của ký tự vào thanh ghi số liệu
        CALL DELAY          ; gọi trễ
        INC  DX             ; tăng DX để có địa chỉ thanh ghi trạng thái
B:      IN AL, DX           ; đọc trạng thái vào AL
        TEST AL, 1000000B   ; kiểm tra sự sẵn sàng nhận tin của máy in
        JZ B                ; trở về B nếu trạng thái chưa sẵn sàng
        INC  DX             ; tăng DX để chỉ thanh ghi điều khiển
        MOV AL, 00001101B   ; đưa lệnh cho tín hiệu STROBE cho máy in
        OUT DX, AL          ; đưa tín hiệu STROBE
        MOV AL, 00001100B   ; chuyển lệnh xoá tín hiệu STROBE
        OUT DX, AL          ; đưa lệnh xoá tín hiệu STROBE
        CALL DELAY          ; gọi trễ
DELAY:  MOV AH, 00          ; chức năng đọc bộ đếm đồng hồ
        INT 1A              ; ngắt phục vụ đếm đồng hồ
        ADD DX, 2           ; từ thấp của bộ đếm + 2
        MOV BX, DX          ; gửi từ thấp đếm +2 vào BX
A:      INT 1A              ; đọc bộ đếm mới
        CMP DX, BX          ; so sánh với bộ đếm cuối
        JB A                ; quay trở về A để đọc số đếm và so sánh
        RET                 ; trở về từ chương trình con.
```

Cũng như chương trình khởi phát, nếu chạy chương trình trên với DEBUG, ta phải bỏ các nhãn A, B và thay các nhãn trên trong câu lệnh bằng địa chỉ của lệnh có nhãn A, B đứng trước.

## 6.2.3. LẬP TRÌNH SỬ DỤNG NGẮT CHO MÁY IN

Hệ điều hành có 3 ngắt phục vụ máy in: INT 5h và INT 17h của BIOS và chức năng 05h của ngắt INT 21h của DOS.

### 1. Chép màn hình ra máy in

Ngắt INT 5h là ngắt riêng của VXL, được sử dụng trong máy để:

- Chép nội dung của màn hình ra máy in khi ấn phím Print Screen. Khi nhận được tín hiệu ấn phím trên, BIOS khởi tạo véctơ ngắt để trỏ tới thư mục chép màn hình của BIOS trong ROM BIOS.

- Gửi một số kết quả trung gian ra máy in trong quá trình thực hiện các chương trình hợp ngữ và ngôn ngữ bậc cao khi được gọi trong chương trình.

Ngắt INT 5h trên không có các thông số vào và ra.

### 2. Lập trình dùng ngắt INT 17h

Ngắt INT 17h của BIOS với ba chức năng 00h (in ra một ký tự), 01h (khởi động máy in) và 02h (kiểm tra trạng thái).

***a) In ra một ký tự:*** chức năng **AH=00h**

- Thông số vào: AH = 0h
  AL = mã ASCII của ký tự
  DX = số của máy in (0-2).
- Thông số ra: AH = trạng thái máy in (xem thanh ghi trạng thái).

***b) Khởi động cổng máy in***

Cần gọi chức năng này trước khi in ra ký tự.

- Thông số vào: AH = 02h
  DX = số hiệu máy in.
- Thông số ra: AH = trạng thái máy in.

***c) Đọc trạng thái máy in:*** chức năng **AH=02h**

- Thông số vào: AH = 02h
  DX = số hiệu máy in.
- Thông số ra: AH = trạng thái máy in với các bit như sau:

$d_7$ = 1- bận; $d_6$ = 1- từ chối ; $d_5$ = 1- hết giấy ; $d_4$ = 1 - sẵn sàng nhận ; $d_3$ = 1 - lỗi truyền ; $d_2 \div d_1$ không dùng ; $d_0$ = 1 - lỗi quá thời gian (time out).

Nếu in nhiều ký tự liên tiếp, ta phải lặp lại chức năng trên nhiều lần với nội dung nạp vào thanh ghi AL thay đổi theo mã của ký tự. Nếu cần xuống dòng hay cách dòng, ta phải đưa các mã ASCII tương ứng vào AL trước khi đưa ra máy in.

*Ví dụ:* Chương trình in chữ 'A' ra máy in LPT1

```
        MOV  DX, 00      ; máy in LPT1
        MOV  AH, 01h     ; khởi phát máy in
        INT  17h         ; gọi ngắt
        MOV  CX, 10h     ; in 16 chữ
A:      MOV  AL, 41      ; chữ 'A'
        CALL IN          ; ghi số liệu ra LPT1
```

```
        DEC   CX          ; giảm số lần
        CMP   CX, 00      ; so sánh xem đã in xong chưa
        JNZ   A           ; nhảy về A nếu CX chưa bằng 0
        MOV   AL, 0Dh     ; đưa lời  cho thanh ghi điều khiển
        CALL  IN
        INT   20h
IN:     MOV   AH,00
        INT   17h
        RET
```

## 6.2.4. LẬP TRÌNH CHO GẬY ĐIỀU KHIỂN TRÒ CHƠI

### 1. Thiết bị điều khiển trò chơi

Ngoài sử dụng các phím nhấn (phải, trái, lên, xuống), trò chơi trên MVT còn được điều khiển bằng hai gậy điều khiển A, B (cho hai người chơi).

Thiết bị điều khiển gồm hộp điều khiển (ở bên ngoài MVT) và bìa ghép nối. Chúng được nối với nhau bằng các đường dây song song qua ổ cắm DB-15 có 15 chân cắm.

Mỗi hộp điều khiển có:

- Gậy điều khiển nối với hai biến trở X, Y đặt vuông góc với nhau. Gậy có các vị trí, tương ứng với các giá trị của điện trở từ 0 kΩ tới 100 kΩ. Giá trị của điện trở này tạo nên một điện thế tương tự, được bìa đọc vào và biến đổi thành các giá trị số của tọa độ X, Y của gậy. Mỗi phím 1 hay 2 của hộp A hay B có hai vị trí ấn (0), nhả(1) được đọc vào thanh ghi trạng thái ở các bit BB2, BB1, BA2, BA1, còn các tọa độ của gậy (ở vị trí trái (0) hay phải (1)) được đọc vào các bit BY, BX (hộp B), còn của hộp A ở AY, AX như sau: BB2 ($d_7$), BB1 ($d_6$), BA2 ($d_5$), BA1 ($d_4$), BY ($d_3$), BX ($d_2$), AY ($d_1$), AX ($d_0$).

- Ổ cắm DB-15 có các chân cắm tương ứng với các tín hiệu như sau:
  + Số 2 - phím 1 của hộp A (BA1)
  + Số 3 - biến trở X của hộp A
  + Số 6 - biến trở Y của hộp A
  + Số 7 - phím 2 của hộp A (BA2)
  + Số 7 - phím 1 của hộp B (BB1)
  + Số 11 - biến trở X của hộp B
  + Số 13 - biến trở Y của hộp B
  + Số 14 - phím 2 của hộp B (BB2)
  + Số 1, 8, 9, 15 - $V_{cc}$ (+5V)
  + Số 4, 5, 12 - GND (0V).
- Bìa cho gậy điều khiển gồm:
  + Bộ đệm cho thanh ghi trạng thái phím và gậy 1.
  + Máy phát xung vuông.
  + Các bộ biến đổi tương tự-số cho các tọa độ X,Y của các gậy.
  + Các bộ đệm đường dây và giải mã địa chỉ-lệnh.

### 2. Lập trình trực tiếp

Mỗi trò chơi có một chương trình phức tạp. Mỗi lần ấn phím và vị trí gậy tương ứng với một chương trình xác định. Ở đây ta chỉ viết chương trình đọc trạng thái của các phím 1, 2

và gậy của mỗi hộp A, B, sau đó kiểm tra xem phím nào được ấn, gậy A hay B ở vị trí trái hay phải bằng cách kiểm tra bit.

Chương trình kiểm tra bit BA2 được nhả, thanh ghi trạng thái có giá trị 40h như sau:

```
MOV DX, 201h      ; gán địa chỉ của cổng trò chơi
IN AL, DX         ; đọc trạng thái
TEST AL, 20h      ; kiểm tra bit BA2 được nhả.
```

## 3. Lập trình sử dụng ngắt

Ngắt INT 15h với chức năng 84h (ghi vào AH) phục vụ gậy điều khiển, chỉ có trong BIOS của AT và PS/2. Ngắt INT 15h có hai chức năng phụ (00h, 01h) ghi vào thanh ghi DX.

- Chức năng phụ 00h chỉ trạng thái của núm gạt.
- Chức năng phụ 01h chỉ vị trí của gậy.

***a) Chương trình kiểm tra trạng thái gậy và phím***

- Thông số vào: AH = 84h
  DX = 00h.
- Thông số ra: cờ carry CF = 1 cho biết không có hộp điều khiển trò chơi
  cờ carry CF = 1 cho biết có hộp điều khiển trò chơi
  AL = các bit của thanh ghi trạng thái như sau:
  $d_7$ = 1 - phím 1 của hộp A bị ấn
  $d_6$ = 1 - phím 2 của hộp A bị ấn
  $d_5$ = 1 - phím 1 của hộp B bị ấn
  $d_4$ = 1 - phím 2 của hộp B bị ấn
  $d_3$ = 1 - tọa độ Y của gậy B
  $d_2$ = 1 - tọa độ X của gậy B
  $d_1$ = 1 - tọa độ Y của gậy A
  $d_3$ = 1 - tọa độ X của gậy A.

*Ví dụ:* Chương trình đọc và kiểm tra phím số 2 của hộp B có ấn không?

```
      MOV AH, 84        ; chức năng 84
      MOV DX, 00        ; chức năng phụ
A:    INT 15            ; ngắt 15h
      JC B              ; nhảy tới kết thúc vì không có hộp điều khiển
      AND AL, 10        ; lọc trạng thái phím thứ 2 của hộp B bị ấn
      TEST AL, 10       ; kiểm tra bit d4 tức phím 2 của hộp B bị ấn
      JNZ A             ; nhảy trở lại A để chờ
      JMP C             ; nhảy tới C để thực hiện chương trình phục vụ phím
B :   INT 20            ; kết thúc chương trình vì không có hộp điều khiển
C :   .....             ; bắt đầu chương trình phục vụ phím.
```

***b) Chương trình xác định vị trí gậy***

- Thông số vào: AH = 84h chức năng
  DX = 01h chức năng phụ.
- Thông số ra : cờ carry CF = 1 - không có bộ điều khiển trò chơi

cờ carry CF = 0 - có hộp điều khiển
AX = tọa độ X của gậy A
BX = tọa độ Y của gậy A
CX = tọa độ X của gậy B
DX = tọa độ Y của gậy B

*Ví dụ:* Chương trình đọc tọa độ X, Y của gậy A. Chương trình này phải kiểm tra trạng thái trước, rồi đọc tọa độ vào các thanh ghi nên chương trình như sau:

```
        MOV AH, 84          ; chức năng 84
        MOV DX, 00          ; chức năng phụ
A :     INT 15              ; ngắt 15h
        JC B                ; nhảy tới kết thúc vì không có hộp điều khiển
        AND AL, 03          ; lọc trạng thái tọa độ  của hộp A
        TEST AL, 03         ; kiểm tra bit d1 và d0 tức tọa độ X, Y của gậy A
        JNZ A               ; nhảy trở lại A để chờ
        JMP C               ; nhảy tới C  để thực hiện chương trình đọc tọa độ
C:      MOV AH, 84          ; chức năng cho gậy điều khiển trò chơi
        MOV DX, 01          ; chức năng đọc tọa độ X, Y
B:      INT 20              ; kết thúc chương trình vì không có hộp điều khiển.
```

## 6.2.5. LẬP TRÌNH CHO THIẾT BỊ NGOÀI GHÉP NỐI SONG SONG

Ở trên chúng ta đã xét lập trình cho cổng song song thông dụng cho máy PC-IBM để đưa tin ra máy in, ở đây ta xét lập trình trao đổi tin song song bằng vi mạch bất kỳ, trong đó sẽ xét chi tiết với vi mạch Intel 8255A, một vi mạch thông dụng, có trong máy PC/XT với VXL 8086. Vi mạch này cũng thường được sử dụng trong các hệ VXL tự thiết kế cho các hệ đo-điều khiển kỹ thuật.

### 1. Cổng song song

Các thiết bị ngoài ghép nối song song bất kỳ yêu cầu trao đổi tin theo 8 bit, 16 bit. Ngoài số liệu trao đổi, thiết bị còn cần trao đổi:

- Các tin về điều khiển thiết bị ngoài như đóng mạch nguồn nuôi, điều khiển quay băng, quay đĩa, cuốn băng (từ, giấy) v.v... Các tin này có chiều từ vi xử lý ra thiết bị ngoài qua cửa vào/ra.

- Các tin về trạng thái thiết bị ngoài như sẵn sàng trao đổi, bận, hết băng, hết giấy, lỗi v.v... Các tin này có chiều từ thiết bị ngoài vào vi xử lý qua cửa vào/ra.

Vậy, muốn trao đổi tin với một thiết bị ngoài, cần phải có ba cửa:

- Cửa vào hoặc ra: đưa số liệu vào hoặc ra hay vào/ra thuận nghịch (đĩa từ).

- Cửa ra: đưa tin về điều khiển.

- Cửa vào: đưa tin về trạng thái.

Về mặt chương trình, chúng ta coi các cửa trên như các cửa ra hoặc vào hoặc vào/ra thuận nghịch như của số liệu. Nếu cửa đọc trạng thái vào, ta phải kiểm tra trạng thái đó xem có sẵn sàng trao đổi tin không để trao đổi số liệu. Nếu chưa sẵn sàng, phải đọc lại trạng thái và kiểm tra lại cho tới khi sẵn sàng.

Với tư cách cửa vào, chúng ta có thể dùng :

- Các vi mạch trên thanh ghi đệm (7475, 7474, 74273, 74274 v.v...) để làm cửa vào ra.

- Vi mạch Intel 8212A cho cửa 8 bit số liệu (hoặc vào hoặc ra) có 1 flip flop cờ chỉ số liệu đã ghi (để yêu cầu đọc) hoặc số liệu đã đọc (để yêu cầu ghi).

- Vi mạch Intel 8255A chương trình hoá (xem chương 1 tóm tắt về phần cứng). Thông thường người ta dùng cửa vào/ra 8255A này vì :

+ Có ba cửa A, B, C, mỗi cửa trao đổi tin 8 bit, trong đó cửa C dành cho:

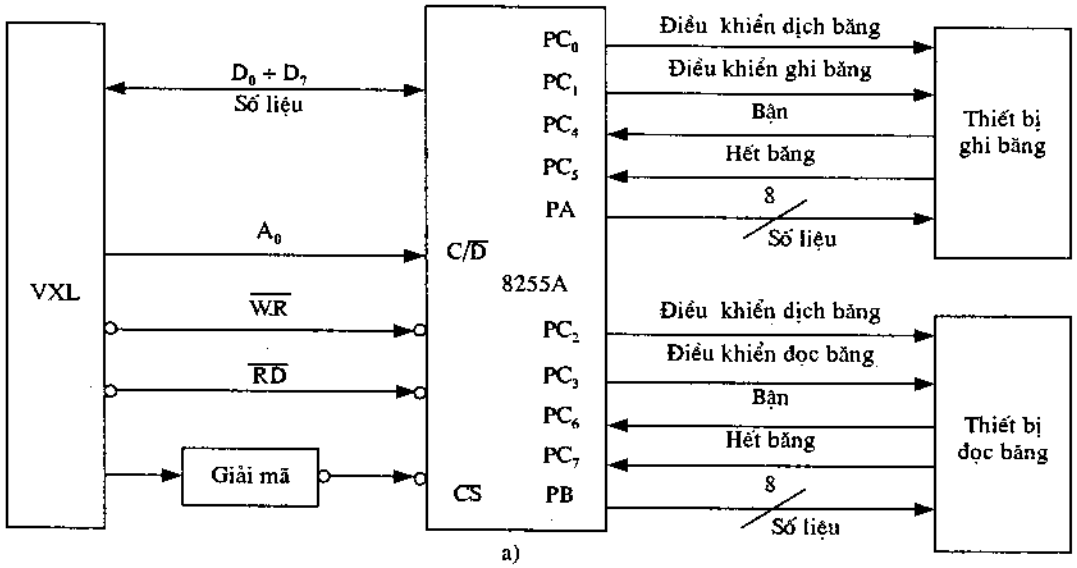
.Trao đổi số liệu khi chế độ M = 0 (ba cửa A, B, C độc lập).

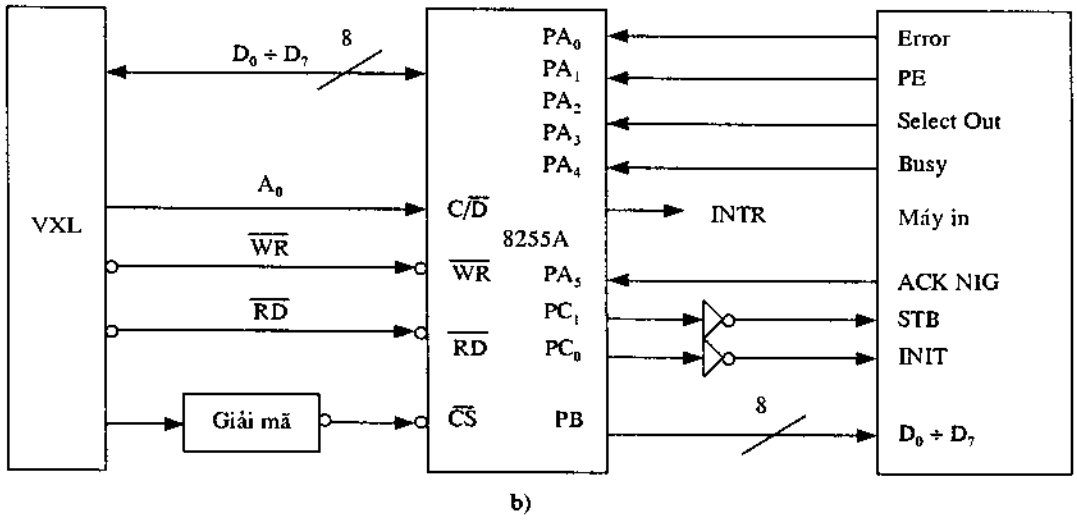
.Trao đổi tin hội thoại (4 bit thấp của nửa C cho cửa B, còn 4 bit cao cho cửa A) khi chế độ M =1, 2).

.Trao đổi tín hiệu lệnh khi bit $d_7 = 0$ của lời lệnh bằng ghi lệnh lập/xóa các bit của cửa C.

+ Có nhiều chế độ như vào hoặc ra không hội thoại (M = 0), vào hoặc ra có hội thoại (M = 1) và cả vào/ra thuận nghịch (M = 2) cho cửa A.

Nếu số lượng tín hiệu hội thoại lớn (hơn 4), ta có thể dùng cửa ra để đưa tín hiệu điều khiển và cửa vào để đọc tin về trạng thái.




**Hình 6.1**. a - sơ đồ mắc thiết bị ghi và đọc băng;
b - sơ đồ mắc máy in Centronic.

Hình 6.1a giới thiệu cách mắc VXL với thiết bị ghi và đọc băng dùng 8255A.

Hình 6.1b giới thiệu cách mắc VXL với máy in Centronix.

Trên hình vẽ ta thấy có ba loại tín hiệu :

- Tín hiệu điều khiển TBN có chiều từ 8255A ra TBN.
- Tín hiệu trạng thái có chiều từ TBN vào 8255A.
- Tín hiệu số liệu có cả hai chiều (ghi ra TBN và đọc từ TBN) vào 8255A.

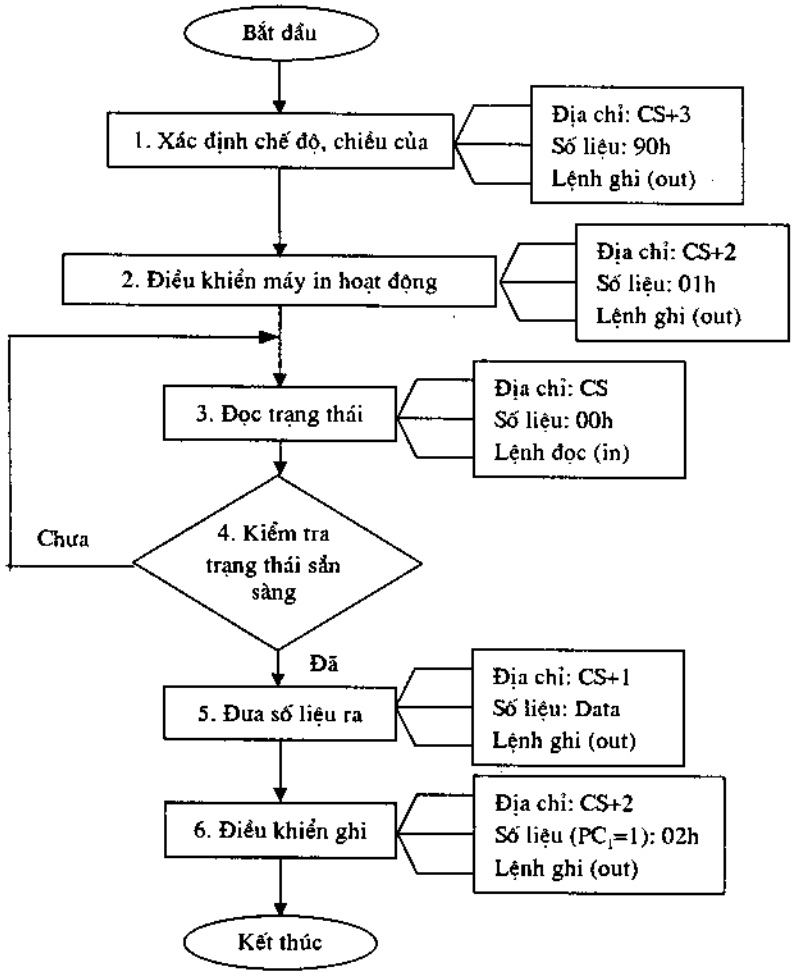
## 2. Lập trình phục vụ trao đổi tin song song

Sau khi thiết kế (vẽ) mạch như các hình trên, lập trình theo các bước sau:

- Vẽ lưu đồ của chương trình (hình 6.2) theo các bước hành động của chương trình. Để tiện theo dõi, ta đánh số cho mỗi hành động của chương trình. Mỗi bước trong khung lại có ba hành động nhỏ là xác định địa chỉ, dữ liệu trao đổi và lệnh.
- Xác định địa chỉ cổng, số liệu và lệnh cho mỗi hành động.
- Viết chương trình.

*a) Lưu đồ*

Lưu đồ của chương trình như hình 6.2 gồm các bước:




**Hình 6.2**. Lưu đồ của chương trình phục vụ máy in.

1. Xác định chế độ và chiều của cửa: đây chính là hành động khởi phát vi mạch, xác định chế độ M (0, 1 hay 2) và chiều của ba cửa A, B, C (nếu chế độ M = 0) hay chiều của hai cửa A, B (nếu chế độ M = 1) và cửa A vào/ra thuận nghịch (nếu chế độ M = 2).

2. Ghi lệnh điều khiển TBN như điều khiển dịch băng, đóng mạch nguồn nuôi hay khởi phát TBN v.v...

3. Đọc trạng thái sẵn sàng trao đổi tin của TBN bằng đọc (lệnh IN).

4. Kiểm tra trạng thái sẵn sàng bằng các lệnh lọc bit trạng thái cần xét (lệnh AND), so sánh (lệnh so sánh CMP) hay kiểm tra (lệnh TEST) và nhảy trở lại đọc trạng thái nếu chưa sẵn sàng.

5. Đưa số liệu ra máy in (lệnh OUT).

6. Điều khiển khởi động máy in bằng đưa lệnh ghi bit điều khiển tương ứng vào thanh ghi điều khiển.

***b) Xác định các địa chỉ dữ liệu và lời dữ liệu cho mỗi hành động***

1. Địa chỉ thanh ghi điều khiển xóa = địa chỉ cơ sở $CS_0$+3.

Lời điều khiển chế độ = 90h =100100000B vì lời chế độ ($d_7$ = 1), cửa A vào ($d_4$ = 1), nửa C cao ($PC_7$, $PC_6$, $PC_5$, $PC_4$) không dùng ($d_2$ = 0), cửa B ra ($d_1$ = 1) và nửa C thấp ($PC_3$, $PC_2$, $PC_1$, $PC_0$) ra ($d_0$ = 0). Lệnh ghi ra OUT DX, AL.

2. Địa chỉ thanh ghi PC thấp điều khiển máy in hoạt động = địa chỉ cơ sở $CS_0$+2.

Lời điều khiển máy in, chính là lệnh xác lập $PC_0$ = 1, có giá trị là 01h.

Lệnh ghi ra OUT DX, AL.

3. Địa chỉ thanh ghi - trạng thái = địa chỉ cơ sở $CS_0$+ của cửa PA.

Lời đọc trạng thái = 00h (vì $PA_0 = PA_1 = PA_2 = PA_3 = PA_4 = 0$) cho biết máy in không bận và còn giấy.

Lệnh đọc vào IN AL, DX.

4. Các bit trạng thái cần kiểm tra là $d_5$, $d_4$, $d_3$, $d_2$, $d_1$ và $d_0$.

Các lệnh : lọc bit trạng thái AND, kiểm tra TEST hay so sánh CMP và lệnh nhảy nếu so sánh không bằng 0 JNZ.

5. Địa chỉ cửa PB là địa chỉ cơ sở $CS_0$ + 1.

Dữ liệu là số liệu vào tùy ý, đưa vào Ah.

Lệnh đưa ra OUT AL, DX.

6. Địa chỉ là địa chỉ thanh ghi điều khiển PC thấp, bằng $CS_0$ + 2.

Dữ liệu là lời điều khiển, xác lập $PC_1$ = 1 để ghi băng.

Lệnh ghi ra OUT DX, AL.

***c) Chương trình***

```
1.      MOV DX, CS0+3     ; gán địa chỉ thanh ghi điều khiển - trạng thái
        MOV AL, 90h       ; lệnh xác lập chế độ
        OUT DX, AL        ; ghi lệnh
2.      MOV DX, CS0+2     ; gán địa chỉ thanh ghi điều khiển - máy in PC thấp
        MOV AL, 01h       ; ghi lời điều khiển PC0  cho máy in hoạt động
        OUT DX, AL        ; đưa lời điều khiển ra
3.      MOV DX, CS0       ; gán địa chỉ thanh ghi trạng thái PA
   A :  IN AL, DX         ; đọc trạng thái máy in vào thanh ghi PA
```

```
4.      AND AL, 00h          ; lọc bit trạng thái
        CMP AL, 00h          ; so sánh trạng thái sẵn sàng
        JNZ A                ; nhảy trở về A nếu trạng thái chưa sẵn sàng
5.      MOV DX, CS0+1        ; gán địa chỉ thanh ghi số liệu PB
        MOV AL, DATA         ; gán số liệu đưa ra
        OUT DX, AL           ; đưa số liệu ra
6.      MOV DX, CS0+2        ; gán địa chỉ thanh ghi điều khiển - trạng thái
        MOV AL, 02h          ; gán lệnh in PCi = 1
        OUT DX, AL           ; đưa lệnh in PCi = 1 ra thanh ghi điều khiển PC thấp
        INT 20               ; kết thúc chương trình.
```

*Ghi chú* :

- Chương trình trên cho việc điều khiển ghi số liệu bởi mức logic 1 hay mặt tăng của xung dương. Nếu dùng mức âm hay mặt giảm của xung, ta phải đưa lệnh xóa xung $PC_i$ bằng cách thêm hai lệnh, MOV AL, 02h ; $PC_i = 0$

OUT DX, AL ; đưa lệnh xóa $PC_i$.

- Chương trình đọc tin vào cho thiết bị đọc băng cũng tương tự, nhưng thay dữ liệu như sau:
  + Lệnh xác lập $PC_2$ là 04h.
  + Lệnh xác lập $PC_3$ là 11h.
  + Lời trạng thái là 0C0h vì $PC_6 = PC_7 = 1$.
  + Thay lệnh OUT DX, AL ở 6 bằng IN AL, DX.

- Ta có thể thay lệnh xác lập từng bit của cửa C bằng cách đưa số liệu = 1 ra bit tương ứng của cửa C đó với lệnh OUT DX, AL. Với cách này, có thể đồng thời đưa nhiều bit ra, tức đồng thời nhiều lệnh.

## 6.3. LẬP TRÌNH TRAO ĐỔI TIN NỐI TIẾP

### 6.3.1. CỔNG TRAO ĐỔI TIN NỐI TIẾP

Ngoài trao đổi tin song song, máy vi tính còn trao đổi tin nối tiếp với TBN nối tiếp như máy in nối tiếp, thiết bị truyền thông nối tiếp (không nối và nối modem, khối ghép nối chuyển mức tín hiệu đường dây điện thoại RS-232C và đường dây điện thoại)

#### 1. Đại cương về trao đổi tin nối tiếp

Trao đổi tin song song có ưu điểm là nhanh (đồng thời các bit) nhưng tốn đường dây và bị nhiễu khi truyền đi xa. Khắc phục nhược điểm trên, người ta trao đổi tin nối tiếp từng bit / một thời điểm nhưng đòi hỏi thiết bị (biến đổi tin song song-nối tiếp và ngược lại) và thủ tục trao đổi tin phức tạp.

Trao đổi tin nối tiếp có thể theo các chế độ :

- *Không đồng bộ* : mỗi lời tin từ 5÷8 bit tin cùng với 1 bit Start, 1÷2 bit Stop và 1 bit kiểm tra tính chẵn/lẻ, xung nhịp phát TxC và thu RxC có tốc độ khác nhau.

- *Đồng bộ:* truyền theo khối tin lớn nhiều bit, trong một khung tin có 1 byte cờ (8 bit), byte địa chỉ, byte lệnh, nhiều byte tin, lời (16 bit) kiểm tra và tốc độ thu/phát RxC, TxC như nhau.

Về thiết bị, trao đổi tin nối tiếp có thể :

- *Không có modem :* (modulation/demodulation - điều biến/giải điều biến) được gọi là zeromodem tức không dùng modem để tải tin số (0,1) hay âm tần bằng một sóng mang cao tần để chống nhiễu và truyền được xa.

- *Có modem:* sóng mang cao tần có biên độ, tần số và pha biến đổi, biến đổi theo tin truyền (tin số 0,1 hay âm tần) để chống nhiễu và truyền được xa.

- *Truyền theo đường dây điện thoại* : (để không cần mắc thêm dây truyền) với khối ghép nối chuyển mức tín hiệu RS-232C, RS 429 v.v...

Về chiều trao đổi tin, có thể theo :

- *Đơn công* : số liệu chỉ truyền theo một hướng từ bộ phận phát tới bộ phận thu (một đường dây một chiều).

- *Bán song công* (half duplex) : số liệu được truyền theo hai hướng nhưng tại một thời điểm chỉ truyền theo một hướng (một đường dây hai chiều).

- *Song công* (Full duplex) : số liệu được gửi đi đồng thời theo hai hướng theo hai đường dây riêng rẽ.

Hiện nay, trao đổi tin nối tiếp được dùng rộng rãi, đặc biệt trong các mạng trao đổi tin (mạng cục bộ-LAN, mạng vùng-WAN và mạng toàn cầu-Internet).

## 2. Vi mạch cổng trao đổi tin nối tiếp 8250/16450

Hãng Intel đã chế tạo các vi mạch cổng trao đổi tin nối tiếp như:

- 8251A cho hệ vi xử lý 8080/8085.

- 8250 cho hệ VXL 8086 trong máy PC/XT.

- 16450 hay 82450 cho hệ 80286, trong máy PC/AT.

- 16550AFN được sử dụng trong máy PS/2 tương tự 16540 nhưng có bộ nhớ đệm FIFO, thu/phát 16 ký tự.

- Vi mạch mật độ cao 82341 là một tổ hợp nhiều khối ghép nối với TBN, trong đó có cổng nối tiếp không đồng bộ tương thích với 16C450.

Các vi mạch cổng trên đều nhận tin song song từ VXL (ở thanh ghi đệm) rồi biến đổi thành tin nối tiếp (ở thanh ghi dịch đệm truyền) và nhận tin nối tiếp rồi biến đổi thành song song để đưa vào VXL.

Trong 8251A chỉ có 3 thanh ghi là điều khiển (có thể ghi được 2 lời chế độ và lời lệnh liên tiếp), còn 8250 và các vi mạch cổng thế hệ sau có tới 12 thanh ghi vì còn điều khiển cả modem. Vi mạch 8250/16450 hay 16550 có các khối chính sau:

- *Bộ phận phát tin nối tiếp:* thanh ghi song song để nhận tin song song từ VXL rồi biến đổi thành tin nối tiếp ở thanh ghi dịch để phát ở lối ra OUTS, thanh ghi điều khiển truyền và thanh ghi trạng thái nối tiếp (chung cho cả bộ phận phát).

- *Bộ phận thu tin nối tiếp:* tin nối tiếp từ lối vào SIN được ghi vào thanh ghi dịch, biến đổi thành song song để ghi thành tin song song đưa vào VXL, thanh ghi dạng dữ liệu và thanh ghi điều khiển nhận (nhận lối vào RCLK).

- *Bộ phận điều khiển modem:* gồm thanh ghi điều khiển modem và trạng thái modem.

- *Logic điều khiển ngắt:* gồm thanh ghi kích hoạt (cho phép) ngắt và thanh ghi ngắt ID.

- *Máy phát xung nhịp:* (máy phát baud) gồm máy phát xung, thanh ghi chốt của bộ chia chứa byte thấp LSB và byte cao MSB.

Vi mạch 8250 có 10 thanh ghi điều khiển và trạng thái, còn 16450 có thêm một thanh ghi đặc biệt nữa. Địa chỉ cơ sở của các thanh ghi trong máy PC -I BM cho các cổng COM1-COM4, các ngắt tương ứng và các bộ nhớ đệm cho BIOS như bảng 6.2.

***Bảng 6.2.*** **Địa chỉ của các ngắt cứng cho các cổng nối tiếp**

| Tên cổng | Địa chỉ cơ sở | Địa chỉ BIOS | Ngắt cứng IRQ |
|---|---|---|---|
| COM1 | 3F8h | 0040:0000h | IRQ4 |
| COM2 | 2F8h | 0040:0002h | IRQ3 |
| COM3 | 3E8h | 0040:0004h | IRQ4 (hoặc hỏi vòng) |
| COM4 | 2E8h | 0040:0006h | IRQ3 (hoặc hỏi vòng) |

Địa chỉ offset của các thanh ghi cho mỗi cổng được chọn bởi các địa chỉ $A_0 \div A_2$ như bảng 6.3. Chúng ta sẽ trình bày dạng các bit của một số thanh ghi đồng thời với lập trình trực tiếp.

***Bảng 6.3.*** **Các thanh ghi của 8250**

| Thanh ghi | Offset | DLAB* | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|
| Đệm số liệu thu | 00h | 0 | 0 | 0 | 0 |
| Giữ số liệu phát | 00h | 0 | 0 | 0 | 1 |
| Cho phép ngắt(che ngắt) | 01h | 0 | 0 | 0 | 1 |
| Nhận diện ngắt | 02h | 0 | 0 | 1 | 0 |
| Dạng số liệu (điều khiển kênh truyền) | 03h | 0 | 0 | 1 | 1 |
| Điều khiển modem | 04h | 0 | 1 | 0 | 0 |
| Trạng thái nối tiếp | 05h | 0 | 1 | 0 | 1 |
| Trạng thái modem | 06h | 0 | 1 | 1 | 0 |
| Lưu trữ đọc/viết | 07h | 0 | 1 | 1 | 1 |
| Chốt bộ chia LSB | 00h | 1 | 0 | 0 | 0 |
| Chốt bộ chia MSB | 01h | 1 | 0 | 0 | 1 |

## 3. Chân nối của cổng nối tiếp

Tùy loại ổ cắm theo các thủ tục khác nhau, ta có số hiệu của chân cắm khác nhau.

### *Ổ cắm loại DB9*

Loại ổ cắm này có 9 chân cắm, phân bố như bảng 6.4.

DLAB (divisor latch access bit - bit xâm nhập chốt bộ chia).

***Bảng 6.4.*** **Phân bố chân cắm cho ổ cắm DB9**

| Chân DB9 | Chân DB25 | Tên | Ý nghĩa |
|---|---|---|---|
| 9 | 22 | RI | Ring Indicator - báo hiệu chuông |
| 8 | 5 | CTS | Clear To Send - xoá sự gửi |
| 7 | 4 | RTS | Request To Send - yêu cầu gửi |
| 6 | 6 | DSR | Data Set Ready - bộ số liệu sẵn sàng |
| 5 | 7 | GND | Ground - đất |
| 4 | 20 | DTR | Data Terminal Ready- thiết bị đầu cuối sẵn sàng |
| 3 | 2 | TxD | Transmitter Data- dữ liệu truyền |
| 2 | 1 | RxD | Receiver Data - dữ liệu nhận |
| 1 | 8 | DCD | Data Carrier Detect - phát hiện sóng mang |
| Không có | 23 ÷ 25 | Không dùng | Không dùng |
| Không có | 21 | Không dùng | Không dùng |
| Không có | 9 ÷ 9 | Không dùng | Không dùng |
| Không có | 1 | Không dùng | Không dùng |

Theo chuẩn V.24, số chân cắm cho ổ cắm DB25 như bảng 6.5.

*Bảng 6.5.* **Phân bố chân cắm cho ổ cắm DB25**

| Chân | Tên | Ý nghĩa |
|---|---|---|
| 1 | GND CHASIS | Đất của bệ máy |
| 2 | TxD | Transmitter Data- dữ liệu truyền |
| 3 | RxD | Receiver Data - dữ liệu nhận |
| 4 | RTS | Request To Send - yêu cầu gửi |
| 5 | CTS | Clear To Send - xoá sự gửi |
| 6 | DSR | Data Set Ready - bộ số liệu sẵn sàng |
| 7 | GND | Đất |
| 8 | DCD | Data Carrier Detect - phát hiện sóng mang dữ liệu |
| 9 | TV+ | Tension volte + - điện thế dương |
| 10 | TV- | Tension volte - - điện thế âm |
| 11 | STF | Selection Transmitter frequency - chọn tần số truyền |
| 12 | 2.DCD | 2. Data Carrier Detect - phát hiện sóng mang dữ liệu thứ 2 |
| 13 | 2.CTS | 2. Clear To Send - xoá sự gửi thứ 2 |
| 14 | 2.TxD | 2. Transmitter Data - dữ liệu truyền thứ 2 |
| 15 | TxC | Transmitter Clock - nhịp truyền |
| 16 | 2.RxD | 2.Receiver Data - dữ liệu nhận thứ 2 |
| 17 | RxC | Receiver Clock - nhịp nhận |
| 18 | | Không dùng |
| 19 | 2.RTS | 2.Request To Send - Yêu cầu gửi thứ 2 |
| 20 | DTR | Data Terminal Ready- thiết bị đầu cuối sẵn sàng |
| 21 | SQD | Select quality data - chất lượng dữ liệu nhận |
| 22 | RI | Ring Indicator - báo hiệu chuông |
| 23 | DRS | Data rate select - chọn tốc độ dữ liệu |
| 24 | TxC | Transmitter Clock - nhịp truyền |
| 25 | BUSY | Busy- bận |

## 6.3.2. LẬP TRÌNH TRỰC TIẾP CỔNG NỐI TIẾP

### 1. Lập trình điều khiển và đọc trạng thái kênh truyền

Lập trình điều khiển kênh truyền tức ghi các thông số cho các thanh ghi của khối điều khiển như thanh ghi điều khiển hay thanh ghi dạng số liệu (điều khiển khởi phát), cho phép ngắt, nhận dạng ngắt, thanh ghi chia tốc độ truyền dữ liệu. Trạng thái kênh truyền có thể biết bằng việc đọc thanh ghi trạng thái.

***a) Thanh ghi điều khiển***

Thanh ghi điều khiển có các bit như sau :

- $d_1$-$d_0$ : chỉ độ dài ký tự với các giá trị 00 - 5bit ; 01 - 6bit ; 10 - 7 bit ; 11 - 8 bit.
- $d_2$ : chỉ số bit dừng với giá trị 0-1 bit dừng; 1-1,5 bit dừng với 5 bit, còn khác đi là 2 bit dừng.
- $d_3$ : chỉ khả năng chẵn(1) hoặc lẻ (0).
- $d_4$ : chỉ lựa chọn (nếu $d_3 = 1$) chẵn (1) hay lẻ (0).
- $d_5 = 1$: bit chẵn/lẻ là logic 0 nếu $d_4 = 1$,
  $= 0$: bit chẵn/lẻ là logic 1 nếu $d_4 = 0$.

- $d_6$ : chỉ có thể phát hiện các tín hiệu ngắt dòng tin (break),

= 1 : lối ra nối tiếp bị cưỡng bức lên dấu hiệu trống,

= 0 : không thể tạo ngắt dòng tin.

- $d_7$ : bit truy cập chốt của bộ chia tần số,

= 1 : truy cập tới các thanh ghi của bộ chia tốc độ baud,

= 0 : truy cập tới thanh ghi giữ số liệu, thanh ghi dữ liệu truyền nhận và thanh ghi cho phép ngắt.

***b) Thanh ghi trạng thái kênh truyền***

Thanh ghi có các bit sau :

- $d_0$ = 1 : RxDY, báo đã có số liệu trong thanh ghi đệm nhận.
- $d_1$ = 1 : OE, báo có lỗi vì dữ liệu không truyền kịp thời từ thanh ghi đệm nhận.
- $d_2$ = 1 : PE, báo có lỗi khi kiểm tra về tính chẵn/lẻ.
- $d_3$ = 1 : FE, báo có lỗi tràn khung (không có bit stop ở ký tự nhận).
- $d_4$ = 1 : BRK, báo có chỉ thị dòng tin bị gián đoạn do tín hiệu ngắt (break).
- $d_5$ = 1 : TxEM, báo thanh ghi giữ dữ liệu truyền rỗng (yêu cầu VXL đưa số liệu ra để truyền).
- $d_6$ = 1 : TEM, báo cả hai thanh ghi chứa và dịch chuyển số liệu bị trống (empty-rỗng); $d_6$ = 0, báo thanh ghi dịch số liệu có số liệu.
- $d_7$ = 0 : không sử dụng.

***c) Thanh ghi cho phép ngắt***

Thanh ghi này ghi lời điều khiển cho phép tín hiệu ngắt vào lối vào INTR để yêu cầu VXL ngắt chương trình do một trong 4 nguồn gây ngắt sau:

- $d_0$ = 1: có thể tạo ngắt do dữ liệu nhận.
- $d_1$ = 1: có thể tạo ngắt do thanh ghi giữ số liệu truyền bị rỗng (đã truyền số liệu, yêu cầu đưa số liệu ra tiếp).
- $d_2$ = 1: có thể tạo ngắt do trạng thái đường dây nhận
- $d_3$ = 1: có thể tạo ngắt do trạng thái modem.

***d) Thanh ghi nhận dạng ngắt***

Thanh ghi này được dùng để biết có ngắt chưa xử lý, tìm nguồn gây ngắt chương trình(lỗi, yêu cầu nhận, yêu cầu phát tin). Các bit của thanh ghi như sau :

- $d_0$ = 0 : báo có một ngắt chưa xử lý.
- $d_1$ và $d_2$ : cho biết nguyên nhân gây ngắt như bảng 6.6.

***Bảng 6.6.*** **Nguyên nhân gây ngắt trong thanh ghi nhận dạng ngắt**

| Các bit | | Mức ưu tiên ngắt | Loại | Nguyên nhân | Hoạt động |
|---|---|---|---|---|---|
| $d_2$ | $d_1$ | | | | |
| 1 | 1 | 1 | Trạng thái kênh nhận | Các loại lỗi | Đọc trạng thái |
| 1 | 0 | 2 | Sẵn sàng dùng số liệu nhận | Ngắt dừng | Đọc dữ liệu nhận |
| 0 | 1 | 3 | Thanh ghi đệm truyền rỗng | Không có dữ liệu | Đọc nhận ngắt hay ghi dữ liệu truyền |
| 0 | 0 | 4 | Trạng thái modem | Xóa để gửi, dữ liệu có sẵn, chuông, dò tín hiệu kênh nhận | Đọc trạng thái modem |

*e) Thanh ghi chia tốc độ truyền dữ liệu*

Hai thanh ghi này dùng để ghi hệ số chia D (MSB, LSB) hay ước số mong muốn cho tốc độ truyền Br mong muốn theo công thức sau:

D = Ck/ 16*Br trong đó D = ước số mong muốn.

Ck = tốc độ xung nhịp của máy phát xung tính bằng Hz.

Br = tốc độ truyền mong muốn.

## 2. Lập trình cho modem

Các vi mạch 8251A, 8250/16450 hay 16550 đều có khối điều khiển modem, nhưng trừ vi mạch 8251A, các vi mạch sau đều có riêng hai thanh ghi điều khiển và trạng thái modem để đưa và nhận các tín hiệu điều khiển (DTR, RTS), trạng thái (DSR, CTS, DCD, RI) và đưa tin ra OUT1 (space signal - tín hiệu rỗi) và OUT2 (cho phép ngắt khi modem đã kích hoạt).

*a) Thanh ghi điều khiển modem*

Thanh ghi điều khiển modem có các bit sau :

$d_0$ : dành cho tín hiệu dữ liệu của thiết bị đầu cuối sẵn sàng DTR với giá trị =1 (modem được kích hoạt).

$d_1$ : dành cho tín hiệu yêu cầu gửi RTS với giá trị = 1 (modem được báo chuẩn bị gửi, kích hoạt phát sóng mang).

$d_2$ : dành cho tín hiệu OUT1 (space signal - tín hiệu rỗi) khi = 1.

$d_3$ : dành cho tín hiệu OUT2 (interrupt enable signal- tín hiệu cho phép ngắt), với giá trị =1.

$d_4$ : để cho phép tin ra ở bộ truyền quay vòng trở về thanh ghi nhận với giá trị =1. Bit điều khiển này được dùng để kiểm tra thiết bị xem việc thu, phát có đúng không.

*b) Thanh ghi trạng thái modem*

Thanh ghi này ghi trạng thái modem, để cung cấp cho VXL thông tin liên quan tới trạng thái của các kênh điều khiển modem hay thiết bị giống modem. Thanh ghi modem nhãn Delta có 8 bit, 4 bit đầu $d_0 \div d_3 = 1$ dành cho sự thay đổi trạng thái, còn 4 bit sau dành cho sự thay đổi trạng thái ngược lại của các tín hiệu của 4 bit đầu. Cụ thể là:

- bit $d_0 = 1$ : chỉ tín hiệu CTS đã thay đổi trạng thái;
- bit $d_1 = 1$ : chỉ tín hiệu DSR đã thay đổi trạng thái;
- bit $d_2 = 1$ : chỉ tín hiệu RI đã thay đổi trạng thái;
- bit $d_3 = 1$ : chỉ tín hiệu DCD đã thay đổi trạng thái.

## 3. Lập trình phát, thu tin

*a) Trình tự lập trình phát tin*

Muốn phát một tin từ thanh ghi đệm dữ liệu truyền (địa chỉ 3F8h cho cổng COM1, 2F8h cho cổng COM2), phải theo trình tự sau :

- Gọi chương trình khởi phát, tức ghi các lệnh điều khiển cho thanh ghi điều khiển, thanh ghi cho phép ngắt và thanh ghi ước số chia tốc độ.
- Đọc thanh ghi nhận diện ngắt để xử lý ngắt tương ứng.
- Đọc thanh ghi trạng thái kênh truyền để xác nhận không có lỗi và sẵn sàng trao đổi tin.
- Ghi tín hiệu DTR vào thanh ghi điều khiển modem và đọc trạng thái modem để kiểm tra bit DSR.

- Ghi dữ liệu ra thanh ghi dữ liệu truyền.

- Ghi lệnh RTS vào thanh ghi điều khiển modem.

- Đọc thanh ghi trạng thái modem để xem đã có tín hiệu kết thúc truyền CTS chưa.

- Đọc thanh ghi trạng thái truyền để xem đã truyền đúng (không lỗi) chưa. Nếu có lỗi, phải truyền lại.

***b) Trình tự nhận tin***

Chương trình nhận tin phát trên đường dây theo trình tự sau:

- Gọi chương trình khởi phát, tức ghi các lệnh điều khiển cho thanh ghi điều khiển, thanh ghi cho phép ngắt và thanh ghi ước số chia tốc độ.

- Đọc thanh ghi nhận diện ngắt để xử lý ngắt tương ứng.

- Đọc thanh ghi trạng thái kênh truyền để xác nhận không có lỗi và sẵn sàng trao đổi tin.

- Ghi tín hiệu DTR vào thanh ghi điều khiển modem và đọc trạng thái modem để kiểm tra và chờ bit DSR và bit RI (tín hiệu báo chuông RI) và DCD (phát hiện sóng mang).

- Ghi tín hiệu RTS vào thanh ghi điều khiển modem để điều khiển modem thu tin.

- Đọc thanh ghi trạng thái modem để phát hiện bit CTS chứng tỏ đã thu xong.

- Đọc trạng thái kênh truyền ở thanh ghi trạng thái kênh truyền để phát hiện lỗi khi thu.

***c) Các chương trình ví dụ***

Các ví dụ sau đây là những chương trình con ngắn, và cuối cùng là chương trình tổng hợp, có thể chạy trực tiếp trong DEBUG, chỉ cần :

- Bỏ nhãn trước câu lệnh và thay nhãn trong câu lệnh bằng địa chỉ của lệnh tương ứng mà nhãn đã gán.

- Bỏ chữ h (tượng trưng cho số dạng hexadecimal) vì DEBUG chỉ dùng số dạng hexadecimal.

Nếu chạy các chương trình trên dưới dạng chương trình chính .EXE, .COM hay chương trình con hay dạng Macro lệnh, ta chỉ cần lắp chúng vào khung với các khai báo mở đầu và kết thúc.

*Ví dụ 1:* Chương trình khởi phát

```
MOV DX, 3FBh      ; gán địa chỉ thanh ghi điều khiển
MOV AL,80h        ; lời điều khiển có bit d7=1 (DLAB)
OUT DX,AL         ; đưa ra thanh ghi điều khiển
PUSH DX           ; lưu giữ DX
MOV DX, 3F8h      ; gán địa chỉ thanh ghi dữ liệu
MOV AL, 12h       ; tốc độ truyền 9600 (d4=1, d1=1)
OUT DX, AL        ; đưa ra thanh ghi dữ liệu cho chốt độ chia (DLAB)
POP DX            ; hồi phục địa chỉ thanh ghi điều khiển
MOV AL, 03h       ; truyền một lời 8 bit (d0=d1=1), 1 bit Stop (d2=0)
OUT DX, AL        ; đưa ra thanh ghi điều khiển
INT 20h           ; kết thúc chương trình trong DEBUG.
```

*Ví dụ 2* : Chương trình ghi cho phép ngắt để sẵn sàng nhận dữ liệu.

```
MOV DX, 378h      ; nạp địa chỉ cổng thanh ghi dữ liệu
ADD DX, 3h        ; nạp địa chỉ cổng thanh ghi điều khiển
```

```
IN AL, DX          ; đọc nội dung thanh ghi trạng thái
CALL RELAY         ; gọi trễ
AND AL,7Fh         ; xóa DLAB
OUT DX, AL         ; đưa ra thanh ghi điều khiển
CALL RELAY         ; gọi trễ
MOV DX, 0F9h       ; địa chỉ thanh ghi cho phép ngắt
MOV AL, 01h        ; giá trị d0=1cho phép ngắt nhận dữ liệu
OUT DX, AL         ; ghi vào thanh ghi cho phép ngắt
CALL RELAY         ; gọi trễ
INT 20h            ; kết thúc chương trình.
```

*Chú ý:* Chương trình con gọi trễ RELAY có ở phần lập trình cho máy in song song, ta phải thay các mã lệnh của nó vào lệnh CALL RELAY.

*Ví dụ 3*: Chương trình kiểm tra trạng thái kênh truyền

```
CHECK: MOV DX, 0FDh        ; gán địa chỉ thanh ghi trạng thái kênh truyền
       IN AL, DX           ; gửi vào DX
       CALL RELAY          ; gọi trễ, có ở phần trao đổi tin song song
       TEST AL,00011110B   ; kiểm tra các bit trạng thái về lỗi
       JZ ERROR            ; nhảy tới chương trình con xử lý lỗi
       TEST AL,00000001B   ; kiểm tra bit d0=1, sẵn sàng nhận tin
       JZ RECEIVE          ; nếu có, chuyển tới chương trình con RECEIVE
       TEST AL,00100000B   ; kiểm tra bit d5=1, sẵn sàng truyền tin
       JZ SEND             ; nếu có, chuyển tới chương trình con SEND
       JMP CHEC            ; trở lại CHECK để kiểm tra tiếp.
```

*Ví dụ 4* : Chương trình truyền dữ liệu

```
REPS : MOV DX, 03FDh       ; gán địa chỉ thanh ghi trạng thái kênh truyền
       IN AL, DX           ; đọc trạng thái
       CMP AL, 04h         ; so sánh bit d6=1 xem đã sẵn sàng truyền chưa?
       JNE REPS            ; trở về kiểm tra lại trạng thái
       MOV DX, 3F8h        ; gán địa chỉ thanh ghi số liệu truyền
       MOV AL, DATA        ; gán DATA (số mã ASCII) để đưa ra
       OUT DX, AL          ; đưa DATA ra cổng truyền
       INT 20h             ; kết thúc chương trình.
```

*Chú ý:* Nếu cho thực hiện trong DEBUG, cần thay lệnh RET bằng INT 20 để kết thúc chương trình và gán thay DATA bằng mã ASCII của ký tự nào đó.

*Ví dụ 5* : Chương trình nhận dữ liệu

```
REPS : MOV DX, 03FDh       ; gán địa chỉ thanh ghi trạng thái kênh truyền
       IN AL, DX           ; đọc trạng thái
       CMP AL, 20h         ; so sánh bit d2= 1 xem sẵn sàng truyền?
       JNZ REPS            ; trở về kiểm tra lại trạng thái
       MOV DX, 3F8h        ; gán địa chỉ thanh ghi số liệu truyền
       IN AL, DX           ; đọc dữ liệu vào
       INT 20h             ; kết thúc chương trình.
```

*Chú ý:* Đối với việc truyền, nhận nhiều dữ liệu, ta có thể thay các lệnh IN, OUT bằng các lệnh cho chuỗi INS (INSB, INSW, INSD) và OUTS (OUTSB, OUTSW, OUTSD) với dữ liệu ở địa chỉ DX đưa vào vùng nhớ có địa chỉ ES:DI và DI = DI+1 (cho lệnh theo B), DI+2 (cho lệnh theo W) và DI+4 (cho lệnh theo D).

Chương trình điều khiển và đọc trạng thái modem, ta cũng viết tương tự chương trình điều khiển và đọc trạng thái thanh ghi truyền nhận, chỉ cần lưu ý là thay địa chỉ và lời lệnh hoặc trạng thái tương ứng.

*Ví dụ 6* : Chương trình nhận nối tiếp ký tự từ bàn phím

```
        MOV DX, 3FBh      ; gán địa chỉ thanh ghi điều khiển của cổng COM1
        MOV AL,80h        ; đặt DLAB = 1
        OUT DX, AL        ; đưa ra thanh ghi
        MOV DX, 3F8h      ; gán địa chỉ thanh ghi dữ liệu
        MOV AL, 12h       ; gán tốc độ truyền 9600 baud
        OUT DX, AL        ; đưa ra thanh ghi
        MOV AL, 03h       ; truyền một từ 8 bit, 1bit stop
        OUT DX, AL        ; đưa ra thanh ghi
        MOV AH, 00h       ; chức năng vào ký tự từ bàn phím
        INT 16h           ; gọi ngắt của bàn phím
        MOV DX, 3FDh      ; gán địa chỉ thanh ghi trạng thái
A:      IN AL, DX         ; đọc trạng thái
        AND AL, 20h       ; đọc trạng thái bit d5 = 1?
        JNZ A             ; chưa có, trở về A để đọc lại
        SUB DX, 05h       ; gán địa chỉ thanh ghi 3F8h là cổng dữ liệu
        OUT DX, AL        ; đưa dữ liệu ra
        INT 20h           ; kết thúc chương trình.
```

## 6.3.3. LẬP TRÌNH SỬ DỤNG NGẮT

BIOS có ngắt INT 14h để điều khiển sự vào ra của cổng với các chức năng sau :

- AH = 00h khởi tạo cổng
- AH = 01h truyền ký tự
- AH = 02h nhận ký tự
- AH = 03h kiểm tra trạng thái.

**1. Khởi tạo cổng**: chức năng **AH = 00h**

- Thông số vào: AH = 00h

DX = số hiệu cổng (0, 1, 2, 3, 4)

AL = các tham số của cấu hình, có các bit $d_4 \div d_0$ giống các bit của thanh ghi điều khiển truyền nhận, còn các bit $d_7 \div d_5$ chỉ tốc độ truyền như sau :

| $d_7\ d_6\ d_5$ | | $d_7\ d_6\ d_5$ | |
|---|---|---|---|
| 0 0 0 = 110 | baud | 1 0 0 = 1200 | baud |
| 0 0 1 = 150 | baud | 1 0 1 = 2400 | baud |
| 0 1 0 = 300 | baud | 1 1 0 = 4800 | baud |
| 0 1 1 = 600 | baud | 1 1 1 = 9600 | baud |

- Thông số ra : AH = trạng thái với các bit như sau :

$d_7$ = 1: time out (quá thời gian) — $d_3$ = 1 : lỗi định khung
$d_6$ = 1: thanh ghi shift rỗng — $d_2$ = 1 : lỗi chẵn lẻ
$d_5$ = 1: thanh ghi chứa chuyển đổi rỗng — $d_1$ = 1 : lỗi
$d_4$ = 1: ngừng, ngắt quãng được chỉ ra — $d_0$ = 1 : dữ liệu sẵn sàng.

AL = trạng thái modem, có các bit như sau :

$d_7$ = 1: đường dây sẵn sàng — $d_3$ = 1: đường nhận delta được xác lập
$d_6$ = 1: chuông điện thoại reo — $d_2$ = 1: (delta) chuông điện thoại
$d_5$ = 1: số liệu sẵn sàng (Modem xác lập) — $d_1$ = 1: dữ liệu delta sẵn sàng
$d_4$ = 1: xóa để gửi (Modem sẵn sàng truyền) — $d_0$ = 1: xóa để gửi kiểu delta.

### 2. Truyền ký tự : chức năng AH = 01h

- Thông số vào: AH = 1, gửi một ký tự ra cổng: RS-232
  DX = số hiệu cổng (0, 1, 2, 3, 4)
  AL = ký tự gửi (mã ASCII).
- Thông số ra: AH = bit $d_7$ = 0 ký tự được truyền
  = 1 lỗi khi truyền, thì các bit $d_6 \div d_0$ là trạng thái kênh truyền (xem ở trên).

### 3. Nhận ký tự: chức năng AH = 02h

- Thông số vào: AH = 02h
  DX = số hiệu cổng (0, 1, 2, 3).
- Thông số ra: AH có bit $d_7$= 0 - ký tự được nhận, khi đó
  AL = mã ASCII của ký tự
  AH có bit $d_7$=1 việc nhận có lỗi và các bit $d_6 \div d_0$ chỉ trạng thái lỗi (xem ở trên).

### 4. Kiểm tra trạng thái truyền: chức năng AH = 03h

- Thông số vào: AH = 3h
  DX = số hiệu cổng (0, 1, 2, 3).
- Thông số ra: AH = trạng thái truyền với các bit như sau:

$d_7$=1: time out (quá thời gian) — $d_3$= 1: lỗi định khung
$d_6$=1: thanh ghi Shift rỗng — $d_2$ = 1: lỗi chẵn lẻ
$d_5$=1: thanh ghi chứa chuyển đổi rỗng — $d_1$ = 1: lỗi
$d_4$=1: ngừng ngắt quãng được chỉ ra — $d_0$ = 1: số hiệu sẵn sàng

AL = trạng thái modem có các bit sau:

$d_7$=1: xác lập liên lạc với modem thu — $d_3$ = 1: (delta) đường nhận được xác lập
$d_6$ = 1: chuông điện thoại reo — $d_2$ = 1: (delta) chuông điện thoại reo
$d_5$ = 1: modem thu xác lập, sẵn sàng đặt số liệu. — $d_1$ = 1: (delta) modem sẵn sàng đặt dữ liệu
$d_4$ = 1: xóa để gửi (modem sẵn sàng truyền) — $d_0$ = 1: (delta) modem sẵn sàng nhận.

### 5. Khởi phát cổng nối tiếp mở rộng: chức năng AH = 04h

Chức năng này tương tự AH = 00h nhưng cho máy PS/2. Nó tăng khả năng của 00h để điều khiển cổng nối tiếp cho PS/2, bốn thông số trong AL đối với chức năng 00h được mở rộng trong bốn thanh ghi:

- CL cho vận tốc với các giá trị từ 110 baud tới 19200 baud (CL = 08h).

- CH cho độ dài tin: 5 bit (00h), 6 bit (01h), 7 bit (02h) và 8 bit (03h).

- BH chọ tính chẵn lẻ: chẵn (02h), lẻ (01h), chỉ thị tính chẵn (04h), chỉ thị tính lẻ (03h).

- BL cho số bit dừng: một bit (00h), hai bit (01h). Riêng độ dài 5 bit, chỉ có 1,5 bit dừng (01h).

Chức năng này cũng trả về (thông số ra) trạng thái kênh truyền và modem vào AX như chức năng 03h.

## 6. Điều khiển truyền thông cổng mở rộng: chức năng AH = 05h

Chức năng này chỉ của BIOS của PS/2, cho phép ta đọc và ghi vào thanh ghi điều khiển modem của một cổng nối tiếp, tương tự chức năng AH=00h, nhưng số hiệu của cổng ghi vào DX khi AL= 0, còn BL ghi giá trị của thanh ghi điều khiển modem. Khi gọi với AL= 01h, BIOS chép giá trị ta đã ghi vào BL sang thanh ghi modem. Trong cả hai trường hợp, chức năng AH = 05h gửi các trạng thái của đường dây và của modem vào thanh ghi AH và AL như chức năng AH = 03h.

## 7. Các chương trình ví dụ

***a) Chương trình truyền dữ liệu qua cổng COM1***

```
        MOV AH, 00h       ; khởi phát cổng
        MOV DX, 00h       ; cổng COM1
        MOV AL, F3h       ; 9600 baud, 2 stop, kiểm tra chẵn, 8 bit tin
        INT 14h           ; gọi ngắt
A:      MOV AH,03h        ; kiểm tra trạng thái
        MOV DX, 00h       ; COM1
        INT 14H           ; gọi ngắt
        TEST AH, 0C2h     ; kiểm tra lỗi
        JNZ A             ; kiểm tra lại nếu có lỗi
        TEST AL, 0F0h     ; kiểm tra modem
        JNZ A             ; kiểm tra lại nếu có lỗi
        MOV AL, DATA      ; gán ký tự truyền
        MOV AH, 01h       ; gọi chức năng truyền
        INT 14h           ; gọi ngắt
        RLC AH, 01h       ; dịch trái qua carry CF
        JC A              ; truyền lại nếu gặp lỗi truyền
        INT 20h           ; kết thúc chương trình.
```

***b) Chương trình nhận dữ liệu qua cổng COM1***

```
        MOV AH, 00h       ; khởi phát cổng
        MOV DX, 00h       ; cổng COM1
        MOV AL, F3h       ; 9600 baud, 2 stop, kiểm tra chẵn, 8 bit tin
        INT 14h           ; gọi ngắt
A:      MOV AH,03h        ; kiểm tra trạng thái
        MOV DX, 00h       ; COM1
        INT 14H           ; gọi ngắt
```

```
TEST AH, 0C2h      ; kiểm tra lỗi
JNZ A              ; kiểm tra lại nếu có lỗi
TEST AL, 0F0h      ; kiểm tra modem
JNZ A              ; kiểm tra lại nếu có lỗi
MOV AH,02h         ; gọi chức năng nhận
INT 14h            ; gọi ngắt
RLC AH, 01h        ; dịch trái qua carry CF
JC A               ; nhận lại nếu gặp lỗi truyền
INT 20h            ; kết thúc chương trình.
```

## 6.4. LẬP TRÌNH TRUYỀN THÔNG TRÊN MẠNG

Mạng máy tính ngày càng được sử dụng rộng rãi nên MS-DOS bắt đầu từ phiên bản 4.0 đã có phần mềm phục vụ mạng và gần đây có Window NT. Lúc đầu, có chương trình tiện ích SHARE.EXE. Sau đây là các chức năng điều khiển mạng của ngắt INT 21h (bảng 6.7).

***Bảng 6.7.*** **Các chức năng cho mạng của ngắt INT 21h**

| Chức năng | Ý nghĩa | Chức năng | Ý nghĩa |
|---|---|---|---|
| 5C00h | Truy cập tới vùng của tệp | 5E03h | Đọc chuỗi ký tự để in |
| 5C01h | Bỏ truy cập tới vùng của tệp | 5F02h | Đọc một lối vào trong danh sách các xác nhận |
| 5E00h | Biết tên khách hàng | 5F03h | Thêm vào một lối vào trong danh sách |
| 5E01h | Thay đổi tên khách hàng | 5F04h | Bãi bỏ một lối vào trong danh sách |
| 5E02h | Thay đổi chuỗi ký tự để in | | |

### 1. Truy cập tới vùng của tệp: chức năng AH = 5Ch

- Thông số vào: AX = 5C00h
  BX = bộ mô tả của tệp cần bảo vệ
  CX : DX con trỏ để DOS cấm thâm nhập
  SI : DI = chiều dài số byte của vùng phải được bảo vệ.

*Chú ý:* Khi muốn bảo vệ cả vùng của tệp, đặt 0000h vào các thanh ghi CX, DX và giá trị FFFFh trong thanh ghi SI và DI.

- Thông số ra: AX = mã lỗi (00h - không lỗi, 01h - không có tệp, 06h - không có bộ mô tả, 21h- tệp đã bảo vệ, 24h - danh sách các lối vào được bảo vệ đã bão hòa.

### 2. Giải phóng một vùng của một tệp: chức năng AH = 5C01h

- Thông số vào: AX = 5C01h
  BX = bộ mô tả của tệp mà nó sẽ giải phóng
  CX : DX = độ dời (offset) của vùng giải phóng
  SI : DI kích thước của vùng giải phóng, tính ra byte.

*Chú ý:* Muốn giải phóng cả vùng của tệp, đặt 0000h vào CX, DX và FFFFh vào SI và DI.

- Thông số ra : AX = mã lỗi như chức năng 5C00h.

**3. Đọc tên của khách hàng (hay trạm):** chức năng **AH = 5E00h**

- Thông số vào: AX = 5E00h
  DS : DX = địa chỉ vùng nhớ đệm ghi tên khách hàng (ít nhất 16 byte).
- Thông số ra : AX = 01h là mã lỗi, mạng không có tên.

**4. Đọc danh sách đã xác nhận:** chức năng **AH = 5F02h**

- Thông số vào: AX = 5F02h
  BX = chỉ số muốn đọc trong danh sách
  DS : SI = bộ đệm của tên địa phương (chiếm 16 byte và kết thúc bằng 00h)
  ES : DI = bộ đệm của tên đơn vị.
- Thông số ra: AX = mã lỗi (00h - không lỗi; 01h - không có trong mạng; 21h - chỉ số không tồn tại).
  BL = mã của đơn vị tương ứng (03h - máy in, 04h - thiết bị đọc).

## CÂU HỎI

1. Nêu vai trò và tác dụng của các cổng trong máy vi tính.

2. Ý nghĩa của địa chỉ cổng, so sánh với địa chỉ của khối nhớ, có gì giống và khác nhau không?

3. Tác dụng vùng nhớ đệm của cổng?

4. Phân biệt cổng song song và nối tiếp? Kể những cổng song song và nối tiếp có trong một máy vi tính PC/XT, PC/AT và PS/2, cách sử dụng chúng?

5. Tác dụng của khối ghép nối điều khiển thiết bị ngoài? Cấu trúc và hoạt động của chúng?

6. Cấu trúc tối thiểu và tối đa cho một cổng? Tác dụng và vai trò các thanh ghi đệm của một cổng?

7. So sánh thành phần, hoạt động của một cổng song song và cổng nối tiếp? Có điểm nào chung? Điểm nào riêng?

8. So sánh hai phương án lập trình cho cổng: trực tiếp và sử dụng ngắt của MS-DOS. Chúng có điểm gì giống và khác nhau.

9. So sánh sự lập trình cho cổng máy in và cổng cho thiết bị ngoài ghép nối? Có điểm nào chung và riêng?

10. So sánh lập trình cho cổng nối tiếp không có modem và có modem? Có những điểm nào chung và riêng?

## BÀI TẬP

1. Viết và cho chạy chương trình kiểm tra sự tồn tại của các cổng song song và nối tiếp của một máy vi tính. Đưa hiện lên màn hình thông báo kết quả kiểm tra.

2. Viết và cho chạy chương trình khởi tạo cổng đưa tin ra máy in và lệnh điều khiển in cho máy in.

3. Viết và cho chạy chương trình kiểm tra trạng thái máy in và chỉ thị các lỗi ra màn hình.

4. Viết và cho chạy chương trình in một chuỗi tin "Trường Đại học Bách khoa"
"Viện Vật lý và Kỹ thuật"
ra máy in bằng cả hai phương pháp trực tiếp và sử dụng ngắt.

5. Viết và cho chạy chương trình kéo một trang giấy (mã điều khiển 12h) của máy in bằng cả hai phương pháp trực tiếp và dùng ngắt.

6. Viết chương trình khởi tạo cổng song song (vi mạch 8255) cho cổng A ở ba chế độ M(0), M(1), M(2).

7. Viết chương trình kiểm tra trạng thái của cổng song song A và B của vi mạch 8255 ở các chế độ M(1), M(2).

8. Viết và cho chạy chương trình kiểm tra các cổng nối tiếp hiện có trong một máy tính, thông báo kết quả kiểm tra lên màn hình.

9. Viết và cho chạy chương trình đọc và ghi các thanh ghi của cổng nối tiếp 8250.

10. Viết chương trình khởi phát cổng nối tiếp để truyền và nhận tin với tốc độ 9600 baud, 7 bit tin, 2 bit dừng, có kiểm tra chẵn.

11. Viết chương trình để phát và thu tin nối tiếp (trực tiếp và dùng ngắt) cho mạch không có modem.

12. Viết chương trình điều khiển và kiểm tra trạng thái modem.

13. Lặp lại câu 11 cho mạch có modem.

14. Viết chương trình trao đổi một chuỗi tin nối tiếp (thông số như câu 10) như sau: 1Bh, 40h, " khoa Tin học".

15. Viết và cho chạy chương trình đưa ra chuỗi tin nối tiếp (thông số như câu 10) chứa trong 200 byte nhớ có địa chỉ ban đầu là DS:BX = 1200h:0200h.

16. Viết chương trình ghi nhận 100D byte tin nối tiếp (thông số như câu 10) qua cổng COM2 vào vùng nhớ có địa chỉ nhỏ nhất DS:BX = 132Eh:0500h.

17. Viết chương trình đọc và kiểm tra trực tiếp trạng thái của gậy điều khiển trò chơi và đưa kết quả lên màn hình.

18. Viết chương trình đọc trực tiếp tọa độ của gậy điều khiển trò chơi A và B.

19. Dùng chức năng AH = 84h, chức năng con DX = 00h của ngắt INT 15h, viết và cho chạy chương trình đọc và kiểm tra tiếp trạng thái của gậy điều khiển trò chơi và đưa kết quả lên màn hình.

20. Dùng chức năng AH = 84h, chức năng con DX = 01h của ngắt INT 15h, viết và cho chạy chương trình đọc các vị trí của các gậy điều khiển trò chơi A và B.

*CHƯƠNG 7*

# LẬP TRÌNH THỜI GIAN VÀ TẠO ÂM THANH

Trong các máy tính PC XT, AT và PS/2 đều có đồng hồ (hệ thống và thời gian thực) và loa (đặt trong máy) để theo dõi thời gian (năm, tháng, ngày, giờ, phút, giây) và báo hiệu bằng âm thanh khi có sự cố trong máy. Mỗi khi truy cập tệp, máy đều tự động ghi lại thời điểm sử dụng tệp. Cả BIOS và DOS của hệ điều hành đều có các chức năng liên quan tới việc đọc và thay đổi thời gian trên. Người lập trình bằng hợp ngữ cần biết cách lập trình (trực tiếp và dùng ngắt) của hệ điều hành để đọc, đặt lại thời gian, đặt giờ báo thức bằng âm thanh ra loa nhỏ đặt trong máy.

## 7.1. THIẾT BỊ THỜI GIAN VÀ TẠO ÂM THANH

Trước khi lập trình về thời gian và âm thanh, ta cần biết các thiết bị tạo nên thời gian và phát ra âm thanh trong các máy tính cá nhân (PC) của hãng IBM. Đó là máy phát xung nhịp, bộ đếm định thời khoảng 8253/8354 và hệ phát âm ra loa đặt trong máy.

### 7.1.1. ĐỒNG HỒ HỆ THỐNG

Máy phát xung nhịp, bộ đếm/định khoảng thời gian (timer) tạo nên đồng hồ hệ thống tạo ngắt cho VXL theo từng khoảng thời gian để làm tươi (refresh) bộ nhớ động (DRAM).

#### 1. Máy phát xung nhịp

Bất kỳ máy vi tính (MVT) nào cũng đều có máy phát xung (MPX) để điểm nhịp (CLK-clock) thời gian cho các hành động của vi xử lý (VXL). Để ổn định tần số phát, người ta dùng tụ điện bằng thạch anh mắc bên ngoài, có điện dung không đổi theo môi trường để ổn định tần số của xung phát. Tùy tốc độ hoạt động của VXL, tần số của MPX sẽ tăng lên. Hãng Intel đã chế tạo các vi mạch khác nhau cho các thế hệ VXL khác nhau như:

- 8224 cho VXL 8080. Riêng VXL 8085 không có MPX riêng rẽ mà tích hợp luôn trong VXL, chỉ có chân nối với tụ điện ổn định điện dung bằng thạch anh ở bên ngoài.
- 8284 cho VXL 8086/8088 với tần số 4,7MHz.
- 82284 cho VXL 80286 với tần số tới 16MHz.
- 82384 cho VXL 82386 với tần số 32 MHz.
- VXL Pentium có đồng hồ đặt bên trong, tần số từ 66MHz trở lên (tới 400MHz).

Cấu trúc chung của một MPX là một mạch đa hài, có lối ra nối với tụ điện thạch anh. Ngoài ra còn mạch điều khiển lối ra, bộ chia tần số cho các xung nhịp khác nhau và đồng bộ với xung vào kích thích. MPX sẽ tự phát khi bật nguồn nuôi MVT, nên MPX này chỉ dùng làm đồng hồ của hệ thống của MVT để theo dõi nhịp thời gian khi sử dụng MVT mà

không theo dõi được thời gian thực tế vẫn diễn ra. Do đó, kể từ MVT PC/AT còn có thêm đồng hồ thời gian thực bằng vi mạch CMOS MC 146818 của hãng Motorola.

## 2. Bộ đếm/ định khoảng thời gian PIT 8253/8354

Vi mạch 8253/8254 có tên là PIT (Programmable Interval Timer - bộ định khoảng thời gian chương trình hoá được) gồm: - ba bộ đếm 16 bit độc lập nhau, có thể chương trình hoá. Mỗi bộ đếm có lối vào nhịp (clock), lối vào cổng (gate) để điều khiển bắt đầu đếm và lối ra. Lối vào nhịp lấy từ MPX của hệ thống với nhịp khác nhau cho mỗi loại máy:

- Tín hiệu 14,317180MHz của PC/XT được chia 3 thành 4,77MHz cấp cho VXL và chia 4 thành 1,193180MHz (chu kỳ 0,838μ) cho PIT 8253/8254.

- Với các PC hiện nay có nhịp giữa 4,77MHz và 200MHz cũng được chia tần số và đưa vào các lối vào nhịp $CLK_0 \div CLK_2$ này.

Các lối vào/ra khác của PIT 8253/8254 như sau:

- 8 lối vào/ra $d_0 \div d_7$ để trao đổi dữ liệu với VXL.

- Các lối vào lệnh đọc/ghi (RD/WR), chọn vi mạch CS và địa chỉ $A_0$ , $A_1$ để chọn một trong 3 bộ đếm và thanh ghi điều khiển/trạng thái như sau:

Bộ đếm 0 ($A_1 = 0$, $A_0 = 1$), bộ đếm 1 (01), bộ đếm 2 (10) và thanh ghi điều khiển/trạng thái (00).

Các địa chỉ cổng cho các thanh ghi của hai PIT 8253/8254 trong các máy tính PC như bảng 7.1.

***Bảng 7.1.*** **Các địa chỉ cổng của 8253/8254**

| Cổng PIT 1 | Cổng PIT 2 | Thanh ghi | Lệnh |
|---|---|---|---|
| 040h | 048h | Bộ đếm 0 | Đọc/ghi |
| 041h | 049h | Bộ đếm 1 | Đọc/ghi |
| 042h | 04Ah | Bộ đếm 2 | Đọc/ghi |
| 043h | 04Bh | Điều khiển/trạng thái | Đọc/ghi |

Từ điều khiển 8 bit có dạng như sau:

| $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|
| SC1 | SC0 | RW1 | RW0 | M2 | M1 | M0 | BCD |

Trong đó:

- *SC1, SC0:* dùng để chọn bộ đếm 0 (00), bộ đếm 1 (01), bộ đếm 2 (10) và thanh ghi lệnh điều khiển/trạng thái (11).

- *RW1, RW2:* dùng để xác định đọc/ghi cho chốt bộ đệm (00), byte thấp (01), byte cao (10) và byte thấp trước byte cao sau (11).

- *M2, M1, M0:* chỉ chế độ cho bộ đếm như sau:

+ M2M1M0 = 000, chế độ 0 (Interrupt on Terminal - ngắt trên thiết bị đầu cuối), khi bộ đếm xung đếm về 0, lối ra tăng từ mức âm (logic 0) lên mức dương (logic 1). Chế độ này tạo một mức điều khiển cao (dương), có một độ trễ = $Nt_0$ (N = số đếm, $t_0$ = độ kéo dài của xung nhịp).

+ M2M1M0 = 001, chế độ 1 (Programmable Monoflop-đơn hài chương trình hoá) phát một xung có độ rộng = $Nt_0$ , do số đếm quyết định.

+ M2M1M0 = 010, chế độ 2 (Rate generator - máy phát tỷ lệ) hay đa hài phát nhiều xung âm có độ kéo dài $t_0$, chu kỳ $Nt_0$, thời gian nghỉ (mức dương) $(N-1)t_0$.

+ M2M1M0 = 011, chế độ 3 (Square - Wave Generator - máy phát xung vuông), tương tự chế độ 2 nhưng là xung vuông có độ rộng và thời gian nghỉ bằng nhau, bằng $Nt_0/2$ nếu N = số chẵn hay $(N+1)t_0/2$ nếu N = số lẻ). Chế độ này dùng để phát âm thanh ra loa, có tần số = N/2.

+ M2M1M0 = 100, chế độ 4 (Software-Triggered Strobe-xung được kích thích mềm), giống chế độ 1 nhưng xung cửa (Gate) không khởi phát đếm mà do sườn lên của xung nhịp ở lối vào. Lối ra luôn cao và phát một xung âm khi số đếm về 0 (thời gian nghỉ = $Nt_0$). Nói khác đi, khác với chế độ 1, chế độ phát xung này cho xung âm, độ trễ = $Nt_0$, độ kéo dài $t_0$.

+ M2M1M0 = 101, chế độ 5 (Hardware-Triggered Strobe-xung đột cửa được kích thích cứng), giống chế độ 4, nhưng quá trình đếm khởi phát bởi sườn lên của xung lối vào. Đây cũng là một đơn hài, phát một xung âm có độ rộng = $t_0$, độ trễ $Nt_0$ điều khiển bằng sườn xuống của xung nhịp đếm.

Trong họ máy PC-IBM, vi mạch PIT 8253/8254 dùng để:

- Tạo đồng hồ hệ thống, cho xung định thời gian bởi bộ đếm 0 với lối ra OUT0, còn lối vào từ máy phát xung nhịp.

- Tạo một xung điện ngắt để điều khiển việc làm tươi bộ nhớ DRAM, bởi bộ đếm 1 mà lối vào nhịp từ MPX. Bộ đếm 1 sẽ định kỳ kích hoạt kênh 0 của vi mạch DMA 8237. Bộ đếm 1 hoạt động ở chế độ 3 (phát xung vuông) với giá trị ghi vào bộ đếm là 12h hay 18 thập phân, nên tần số phát ra = 1,19318MHz/18 = 66288Hz. Như vậy, cứ sau 15mili giây, sườn lên của xung sẽ phát ra một xung vuông, tạo một chu kỳ đọc giả (tích điện cho tụ đã ghi logic 1 của DRAM) để làm tươi bộ nhớ (tích lại điện cho tụ vì bị rò).

- Tạo xung âm thanh ra loa hay báo thức điện tử bởi bộ đếm 2 với lối ra OUT2.

### 3. Đồng hồ hệ thống

Trong các máy họ PC-IBM, đồng hồ hệ thống gồm:

- Máy phát xung nhịp: dùng chung với VXL, cho xung nhịp CLK vào lối vào CLK0 của PIT 8253/8254.

- Bộ đếm 0 của PIT 8253/8254 có cửa GATE0 luôn nối với 5V.

- Vi mạch xử lý ngắt PIC 8259A với lối vào IR0 nối với lối ra OUT0 của một bộ đếm 0.

- VXL 80x86 có lối vào INTR nối với lối ra tương ứng của PIC 8259A.

- Các ngăn nhớ của bộ nhớ chính để ghi các số liệu về thời gian (ngày, giờ, phút, giây).

Mạch đồng hồ hệ thống này có bộ đếm ở chế độ 2. Lối vào CLK0 được cấp một xung nhịp tần số 1,19318MHz. Giá trị ghi vào bộ đếm luôn là 0 hoặc 65536 để cho xung ở lối ra có tần số 1,19318MHz/65536 = 18,206 Hz. Sườn lên của mỗi xung này sẽ tạo một ngắt cứng trong 8259. Lối ra INTR của vi mạch này gây ra ngắt mềm INT 08h của hệ điều hành, để cập nhật đồng hồ hệ thống 18,206 lần trong một giây.

## 7.1.2. ĐỒNG HỒ THỜI GIAN THỰC

Đồng hồ hệ thống có nhược điểm là chỉ chạy khi bật nguồn nuôi, tức trong khi sử dụng MVT, với các hành động liên quan tới thời gian. Do đó, phải có thêm một đồng hồ thời gian thực, luôn luôn chạy, ngay cả khi tắt máy. Vì vậy, trong PC-IBM còn có thêm một đồng hồ thời gian thực, luôn được nuôi bằng ắc quy để khi tắt điện đồng hồ vẫn chạy.

Các máy PC AT và PS/2 đã trang bị đồng hồ thời gian thực, dựa trên cơ sở vi mạch CMOS của hãng Motorola MC 146818. Vi mạch này tạo giờ tự trị và tự động ghi thời gian (năm, tháng, ngày, giờ, phút, giây) vào ngăn nhớ riêng, cũng được nuôi bằng ắc quy của đồng hồ thời gian thực.

## 1. Vi mạch MC 146818

Vi mạch này có 24 chân với 2 chân cho hai xung dao động OSC1, OSC2 , 8 chân cho các tín hiệu địa chỉ và dữ liệu AD0÷AD7, 2 chân cho tín hiệu điều khiển ghi địa chỉ AS và chọn vi mạch, 1 chân cho tín hiệu điều khiển ghi dữ liệu DS, 1 chân cho tín hiệu chọn chương trình PS, lệnh đọc/ghi R/$\overline{W}$, chân cho tín hiệu xóa $\overline{RESET}$, ngắt $\overline{INT}$ và 2 chân cho tín hiệu nguồn nuôi.

Địa chỉ của tất cả 64 thanh ghi nội của đồng hồ là:

- Thanh ghi chỉ thị có địa chỉ 70h.
- Thanh ghi dữ liệu có địa chỉ 71h.

Muốn đọc/ghi một trong 64 thanh ghi, ta đều cần dùng hai thanh ghi trên, thanh ghi chỉ thị cho biết số hiệu thanh ghi nào và thanh ghi dưới cho biết dữ liệu.

## 2. Các địa chỉ ô nhớ riêng của MC 146818

*a) Các thanh ghi của MC 146818:* Các thanh ghi nội của MC 146818 được đánh số như bảng 7.2.

***Bảng 7.2.*** **Các địa chỉ nhớ của đồng hồ thời gian thực**

| Loại thanh ghi | Khoảng địa chỉ | Mục đích dùng |
|---|---|---|
| Giây | 00h ÷ 01h | Ghi giây cho đồng hồ, cho báo thức ra dạng số BCD |
| Phút | 02h ÷ 03h | Ghi phút cho đồng hồ, cho báo thức ra dạng số BCD |
| Giờ | 04h ÷ 05h | Ghi giờ cho đồng hồ, cho báo thức ra dạng số BCD |
| Ngày | 06h ÷ 07h | Ghi ngày của tuần (thứ) và của ngày ra dạng số BCD |
| Tháng | 08h | Ghi tháng ra dạng số BCD |
| Năm | 09h | Ghi hai số cuối của năm ra dạng số BCD |
| Thế kỷ | 32h | Ghi hai số đầu của năm (thế kỷ) ra dạng số BCD |
| Trạng thái | 0Ah÷0Eh | Các trạng thái A,B,C,D và chẩn đoán |
| Đóng thiết bị | 0Fh | Giá trị đóng |
| Loại đĩa mềm | 10h | Đĩa A: và B: |
| Loại đĩa cứng | 12h | Đĩa C: và D: |
| Mã của thiết bị | 14h | Ghi mã của thiết bị |
| Bộ nhớ | 15h ÷ 18h | Byte thấp và cao của bộ nhớ tổng và mở rộng |
| Điều khiển | 2Eh ÷ 2Fh | Một số điều khiển RAM CMOS |
| Chỉ thị của trạng thái | 33h | Chỉ thị trạng thái |
| Để dành | 11h,30h÷31h, 34h÷3Fh | |

*b) Các thanh ghi trạng thái:*

*Thanh ghi trạng thái A*: dành cho lập trình đồng hồ, có các bit như sau:

- $d_7$ : cho biết các byte nhớ đang cập nhật (= 1), không nên đọc thời gian.
- $d_6 \div d_4$ : ghi tần số của đồng hồ, ví dụ 010B cho tần số 32,767 kHz.
- $d_3 \div d_0$ : ghi tần số ngắt tức chọn tỷ lệ, ví dụ 0110B tương ứng tần số ngắt 1024 ngắt/giây (976,562 micro giây có một ngắt).

*Thanh ghi trạng thái B:* cho biết một số lớn điều chỉnh và hỏi. Các bit của thanh ghi như sau:

- $d_7$: xóa (= 1) và không xóa (= 0) hay cho phép đọc ngày và giờ.
- $d_6$: cho phép ngắt theo chu kỳ tự động (= 1) và không tự động(= 0).
- $d_5$: cho phép ngắt cho báo thức (= 1) hay không.
- $d_4$: cho phép ngắt cập nhật kết thúc (= 1).

- $d_3$: cho phép phát xung vuông (= 1) hay không(= 0)

- $d_2$: cho phép ghi dạng dữ liệu BCD (= 1) hay nhị phân (= 0).

- $d_1$: dạng của giờ trong ngày theo 24 giờ (= 0) hay 12 giờ (= 1).

- $d_0$: giờ mùa hè (= 1) hay mùa đông (= 0).

Ngoài việc đọc tin về trạng thái, các bit của thanh ghi này còn là các bit điều khiển (ghi tin vào), để ghi những tin điều khiển hệ về thời gian (bit $d_7$, $d_2$, $d_1$, $d_0$), cho các loại ngắt liên quan tới thời gian (bit $d_6$÷$d_4$) và cho phép phát xung vuông ($d_3$).

*Thanh ghi trạng thái C:* chủ yếu xác định các điều kiện thay đổi một ngắt, các bit có dạng sau:

- $d_7$: chỉ thị có (= 1) hay không(= 0) có ngắt của đồng hồ.

- $d_6$: chỉ thị có (= 1) hay không (= 0) có ngắt tuần hoàn.

- $d_5$: chỉ thị có (= 1) hay không (= 0) ngắt báo thức.

- $d_4$: chỉ thị có (= 1) ngắt hay không (= 0) sau khi cập nhật đồng hồ.

- $d_3 \div d_0$: luôn bằng 0.

*Thanh ghi trạng thái D:* chỉ có bit $d_7$, các bit khác bằng 0. Bit $d_7$ chỉ ắc quy còn (= 1) hay hết (= 0) điện.

## 3. Các thông tin về cấu hình máy vi tính

### *a) Thông tin về cấu hình máy*

Các thông tin về cấu hình của máy được ghi vào bộ nhớ RAM CMOS của MC 146818, theo các địa chỉ như bảng 7.3.

***Bảng 7.3.*** **Các địa chỉ nhớ riêng thông tin về cấu hình máy vi tính**

| Địa chỉ | Nội dung |
|---|---|
| 0Eh | Byte dự báo |
| 0Fh | Trạng thái khi tắt máy |
| 10h | Mô tả các đĩa mềm |
| 11h | Để dành |
| 12h | Mô tả các đĩa cứng C:, D: |
| 13h | Để dành |
| 14h | Cấu hình |
| 15h | Kích thước (KB) RAM trên vỉ mẹ (byte thấp) |
| 16h | Kích thước (KB) RAM trên vỉ mẹ (byte cao) |
| 17h | Kích thước (KB) RAM trên bìa mở rộng (byte thấp) |
| 18h | Kích thước (KB) RAM trên bìa mở rộng (byte cao) |
| 19h ÷ 2Dh | Để dành |
| 2Eh | Tổng kiểm tra của các byte từ 16 đến 32 byte (byte cao) |
| 2Fh | Tổng kiểm tra của các byte từ 16 đến 32 byte (byte thấp) |
| 30h ÷31h | Kích thước vùng nhớ mở rộng (giống ở 16h, 17h) |
| 32h | Thế kỷ, dạng BCD (riêng PS/2 chứa byte thế kỷ ở 37 còn byte này chứa byte cao của kết quả kiểm tra tổng CRC của các byte 10h÷31h) |
| 33h | Các thông tin để khởi động hệ thống (riêng PS/2 chứa byte thấp của kết quả kiểm tra tổng CRC của các byte 10h÷31h) |
| 34h ÷ 3Fh | Để dành |

*b) Byte dự báo* (địa chỉ 0Eh)

Byte dự báo hay còn gọi tình trạng phân tích quá trình POST (Power On Selt Test - tự kiểm tra lúc bật nguồn) có các bit sau:

- $d_1 \div d_0$ : không sử dụng.
- $d_2$ : chỉ thời gian hợp lệ (sau POST, = 1 có nghĩa là không phải ngày không đúng như ngày 30/2).
- $d_3$ = 1: đĩa cứng hoặc bộ điều khiển đĩa cứng bị hỏng.
- $d_4$ = 1: kích thước bộ nhớ RAM trong các byte 15h - 0Eh bị sai.
- $d_5$ = 1: cấu hình bị sai.
- $d_6$ = 1: kiểm tra tổng (checksum) trong các byte 3Eh-3Fh bị sai.
- $d_7$ = 1: hết ắc quy nuôi CMOS. Trước khi đọc thời gian nên kiểm tra bit này xem đồng hồ có chạy không.

*c) Byte trạng thái khi tắt máy (Shutdown)* (địa chỉ 0Fh)

Byte này được vi xử lý đọc ngay sau khi được khởi động lại (reset), để xác định xem có phải reset ở đây chỉ là một cách để thoát ra khỏi chế độ bảo vệ hay không. Các bit của byte này như sau:

- $d_0$ = 1: reset mềm (Ctrl-Alt-Del), việc tắt máy không lường trước. Bỏ qua POST.
- $d_1$ = 1: việc tắt máy sau khi kích thước vùng nhớ được xác định.
- $d_2$ = 1: việc tắt máy sau khi đã kiểm tra vùng nhớ.
- $d_3$ = 1: việc tắt máy sau khi có lỗi ở vùng nhớ (kiểm tra chẵn lẻ sai).
- $d_4$ = 1: việc tắt máy với yêu cầu của chương trình tải bootstrap.
- $d_5$ = 1: việc tắt máy với lệnh FAR JUMP, (khởi động lại bộ điều khiển ngắt và nhảy đến địa chỉ 0000:[0467h].
- $d_6$, $d_7$, $d_8$ =1: việc tắt máy sau khi qua được một điểm kiểm tra chế độ bảo vệ.
- $d_9$ = 1: việc tắt máy sau khi di chuyển khối (xem INT 15h, chức năng con 87h).
- $d_{10}$ = 1: việc tắt máy với lệnh FAR JUMP nhảy tức thời đến địa chỉ 0000:[0467h].

*d) Các byte mô tả các đĩa*

*Đĩa mềm* (địa chỉ 10h). Byte có các bit như sau:

- $d_3 \div d_0$: chỉ ổ đĩa thứ nhất.
- $d_7 \div d_4$: chỉ ổ đĩa thứ hai. Cả hai loại ổ đĩa có các mã chỉ loại đĩa như sau: 0000B- không có ổ đĩa; 0001B - ổ 320/360 KB; 0010B - ổ 1,2 MB.
- *Đĩa cứng:* Địa chỉ 11h (dành riêng cho PC/AT, PS/2) chỉ loại ổ đĩa (C hay D).

Nếu: . địa chỉ 19h ghi số 0Fh chỉ đĩa 0 hay ổ C.
. địa chỉ 1Ah ghi số F0h chỉ đĩa 1 hay ổ D.
Địa chỉ 12h: chỉ loại ổ đĩa cứng, có các bit:

- $d_0 \div d_3$ : dành cho ổ C.
- $d_4 \div d_7$ : dành cho ổ D.

Các bit $d_0 \div d_3$ hay $d_4 \div d_7$ có các giá trị như sau: 0000 (chỉ không có đĩa cứng), 1111 (dùng địa chỉ 19h và 1Ah) để ghi loại đĩa là 0Fh (loại đĩa C) hay F0h (loại đĩa D).

*f) Byte mô tả thiết bị*

Byte này mô tả các thiết bị có trong máy, với các bit sau:

- $d_0$ =1: chỉ các ổ đĩa mềm được cài đặt.

- $d_1$ = 1 : chỉ đồng xử lý toán học 80x87 được cài đặt.

- $d_2 \div d_3$ : không sử dụng.

- $d_4 \div d_5$ : chỉ màn hình nguyên thủy, với các giá trị 00 (không có hoặc EGA), 01 (CGA 40 cột, EGA hay VEGA), 10 (CGA 80 cột, EGA hay VGA), 11 (đơn sắc TTL).

- $d_6 \div d_7$ : chỉ số ổ đĩa mềm là 1($d_7d_6$ = 00), 2($d_7d_6$ = 01), 3($d_7d_6$ = 10) và 4($d_7d_6$ = 11).

***g) Các byte mô tả bộ nhớ***

Các byte này mô tả vùng nhớ cơ sở, mở rộng như sau:

- Vùng nhớ cơ sở: với byte thấp (địa chỉ 15h) và byte cao (địa chỉ 16h) với các giá trị 0100h (256K), 0200h (512K), 0280h (640K).

- Vùng nhớ mở rộng trên 1MB: với byte thấp (địa chỉ 17h, 30h) và byte cao (địa chỉ 18h, 31h) theo KB từ 0000÷3C00h (xem INT 15h, chức năng con 88h).

***h) Kết quả kiểm tra tổng***

- Địa chỉ 2Eh: ghi kết quả của các CMOS từ 10h đến 20h cho byte thấp.

- Địa chỉ 2Fh: ghi kết quả của các CMOS từ 10h đến 20h cho byte cao.

***i) Thông tin thêm:*** địa chỉ 33h, có các bit:

- $d_6$: được chương trình cài đặt sử dụng.

- $d_7$: tùy chọn 128K vùng nhớ kiểu IBM có cài đặt.

***k) Các byte nhớ để dành***

Các địa chỉ 1Bh÷20Dh và 34h÷3Fh. Riêng máy PS/2 có:

- Byte thế kỷ, ngày tại địa chỉ 37h.

- Các byte mật khẩu có địa chỉ 38h-3Fh (kiểu máy 50). Để đọc hay bổ sung mật khẩu, đầu tiên ghi vào các địa chỉ 78h-7Fh, rồi máy sẽ ghi vào địa chỉ 38h-3Fh của CMOS.

## 7.1.3. THIẾT BỊ TẠO ÂM THANH

### 1. Mạch tạo âm thanh trong PC AT và PS/2

Mạch tạo âm thanh trong các máy PC gồm:

- Máy phát xung nhịp của đồng hồ hệ thống.

- Bộ chia tần số, chính là bộ đếm thứ 2 của vi mạch PIT 8253/8254.

- Một bit của cửa ra của vi mạch cổng song song 8255 để đưa tín hiệu ra loa khi có xung điều khiển đưa ra (đóng vai trò một cổng logic đóng/mở).

- Loa.

Khi muốn phát một âm có tần số xác định, trình tự như sau:

+ Vi xử lý ghi lời điều khiển B6h ra cổng 43h của 8253/8254.

+ Vi xử lý ghi hai byte thấp và cao của số đếm N vào cổng 42h của 8253/8254 để chia tần số.

+ Đưa tín hiệu điều khiển cổng 61h để đóng mạch 8255, đưa tín hiệu ra loa khi bộ đếm đếm tới 0000h.

### 2. Đặc tính của âm phát ra loa

Nếu số đếm N được ghi vào bộ đếm, thì tần số của âm là 1193180/N. Vì N = 1 tới 65535 nên tần số âm trong khoảng 18,2 Hz (số đếm tối đa 65535) tới 11931MHz (số đếm tối

thiểu 1). Với tai người có thể nghe được âm thanh từ tần số 20Hz tới 20000Hz nên dùng tần số 119Hz (số đếm N=10000) tới tần số 11931Hz (số đếm N =1). Chính vì thay đổi số đếm N này, người ta có thể phát ra một bản nhạc với âm thanh có cao độ (phụ thuộc tần số) và trường độ (phụ thuộc độ kéo dài) khác nhau. Vì không có sự điều chỉnh âm lượng nên âm phát ra có cường độ phụ thuộc tần số và âm sắc kém.

### 3. Phát âm thanh trực tiếp

Bằng lập trình, chúng ta có thể phát âm thanh trực tiếp ra loa (không qua bộ đếm chia tần). Tần số phát ra phụ thuộc thời khoảng lặp lại của âm hay thời gian trễ giữa hai lần phát âm. Phát âm kiểu này có các nhược điểm sau:

- Tốn thời gian dùng VXL.
- Tần số của âm phụ thuộc tốc độ hoạt động của máy tính.
- Ngắt nhịp đồng hồ làm nhiễu chất lượng của âm thanh. Nếu cấm ngắt, lại ảnh hưởng tới sự cảm nhận thời gian của máy tính.

### 4. Phát âm thanh của máy tính PCjr

Máy PCjr cũng có bộ đếm chia tần và loa như PC/AT, PS/2 nhưng còn có thêm các nguồn âm và các đầu ra khác cho tín hiệu âm thanh.

Ngoài đầu vào băng cassette và đường dây âm tần, PCjr còn có vi mạch tạo âm TI SN76496A. TI có 4 bộ tạo âm riêng biệt, gọi là 4 giọng. Ba trong 4 giọng tạo nên các âm tinh khiết (giống 8253/8254 tạo ra), còn giọng thứ tư là "tạp âm", tạo ra tiếng ồn không đều nhau và theo nhiều cách. TI chỉ có bộ chia 10 bit (16 bit cho 8253/8254) và tần số nhịp đồng hồ là 3,579 MHz.

## 7.2. LẬP TRÌNH TRỰC TIẾP CHO THỜI GIAN

Lập trình thời gian là viết chương trình để đọc, thay đổi và định giờ báo thức cho bộ đếm thời gian và đồng hồ thời gian thực. Cũng như các thiết bị khác, ta cũng có hai cách lập trình:

- Lập trình trực tiếp tới các thanh ghi hay khối nhớ của đồng hồ thời gian thực CMOS.
- Lập trình sử dụng ngắt cho đồng hồ.

Ở mục này ta chỉ xét lập trình trực tiếp cho đồng hồ hệ thống, đồng hồ thời gian thực CMOS và chạy trong môi trường DEBUG.

### 7.2.1. LẬP TRÌNH GHI VÀ ĐỌC SỐ ĐẾM CỦA 8253/8254

#### 1. Đặc điểm của 8253/8254

***a) Chế độ của bộ đếm***

Như đã trình bầy ở trên, bằng việc ghi số đếm N vào một trong 4 bộ đếm của vi mạch đếm, ta có thể:

- *Tạo một mức thế đột biến dương:* có độ trễ = $Nt_0$ thay đổi (N = số đếm ghi vào bộ đếm, $t_0$ = độ kéo dài xung nhịp của máy phát xung nhịp cho hệ thống VXL = 18,2 nhịp trong 1 giây).

- *Tạo một xung đơn của một đơn hài*

+ Chế độ 1(M2M1M0 = 001), độ kéo dài = $Nt_0$ và điều khiển bởi cổng GATE (mức dương) và xung nhịp. Chế độ này được dùng để phát một xung Strobe có độ trễ $(N+1)t_0$ để ghi số liệu ra cho đường dây dữ liệu sau khi ghi (bằng lệnh ghi) số đếm vào bộ đếm...

+ Chế độ 4(M2M1M0 = 100), giống chế độ 1 nhưng cổng không điều khiển mà sườn dương của xung nhịp phát một xung âm độ kéo dài $t_0$ sau thời gian trễ $Nt_0$.

+ Chế độ 5(M2M1M0 = 101), giống chế độ 1 nhưng cổng không điều khiển mà sườn âm của xung nhịp phát một xung âm, độ kéo dài $t_0$, thời gian trễ $Nt_0$. Chế độ này được sử dụng để phát một tín hiệu khi mất nguồn nuôi, lối ra ở mức cao khi bật nguồn nuôi.

- *Tạo chuỗi xung của một đa hài:* độ kéo dài, thời gian nghỉ và điều khiển khác nhau:

+ Chế độ 2(M2M1M0 = 010), xung âm độ kéo dài $t_0$, thời gian nghỉ $(N-1)t_0$ tức chu kỳ $Nt_0$, điều khiển bởi cổng và xung nhịp.

+ Chế độ 3(M2M1M0 = 011), xung âm, độ kéo dài = $Nt_0$ /2 nếu N là số chẵn hay $(N+1)t_0/2$ nếu N là số lẻ, điều khiển bằng xung cổng GATE và xung nhịp.

***b) Các lệnh***

Ghi hay đọc các thanh ghi, ngoài lệnh IN AL, địa chỉ hay OUT địa chỉ, AL còn phải ghi thêm vào thanh ghi điều khiển đặc điểm của ghi/đọc ở các bit RW1($d_5$), RW0($d_4$) và dạng số liệu ở bit BCD ($d_0$ = 0 - nhị phân, $d_0$ = 1 - mã BCD) như sau:

00 - lệnh chốt bộ đếm để đọc vào VXL.

01 - chỉ đọc/ ghi byte thấp.

10 - chỉ đọc/ ghi byte cao.

11 - đọc/ghi byte thấp trước, byte cao sau.

***c) Các thanh ghi***

Mỗi vi mạch 8253/8254 có 3 bộ đếm và 1 thanh ghi điều khiển. Việc chọn chúng có thể dùng:

- Địa chỉ cổng như bộ PIT I có 040h (bộ đếm 0), 041h (bộ đếm 0), 042h (bộ đếm 2) và 043h (thanh ghi điều khiển).

- Các bit SC1($d_7$),SC0 ($d_6$) với 00 (bộ đếm 0), 01 (bộ đếm 1), 10 (bộ đếm 2), 11 (lệnh đọc thanh ghi điều khiển).

## 2. Ghi vào thanh ghi điều khiển

Thanh ghi điều khiển của PIT thứ nhất trong các máy PC có địa chỉ 43h với các bit về địa chỉ con (SC1,SC0), đặc điểm đọc/ghi (RW1,RW0), về chế độ phát xung (M2,M1,M0) và dạng dữ liệu (BCD). Trước khi viết lệnh ghi, ta phải xác định lời điều khiển cho đúng chế độ, địa chỉ con và cách thức đọc/ghi. Ví dụ, muốn phát âm thanh ra loa với xung hình vuông (chế độ 3), tần số 10kHz, ta phải ghi vào thanh ghi bộ đếm 2 ở byte thấp. Các bit của thanh ghi điều khiển là 10010110B = 96h. Các lệnh ghi vào thanh ghi điều khiển như sau:

```
MOV AL, 96h      ; gửi 10010110B vào AL
OUT 43h, AL      ; đưa AL ra cổng 43h tức thanh ghi điều khiển.
```

## 3. Ghi vào bộ đếm

Bộ đếm dùng để ghi số đếm N và đếm lùi về 0000h trong phát xung (chuỗi hay một) hay tạo trễ. Muốn vậy, ta phải xác định:

- Chế độ phát, chế độ ghi (byte nào hay cả hai) và bộ đếm nào bằng cách xác định lời điều khiển đã xét ở trên.

- Số đếm N = tần số của xung nhịp/ tần số (hay độ trễ) của xung cần phát. Nếu tần số xung phát 10kHz, ta có N = 1,19318MHz/0,010 MHz = 119D = 77h.

Các lệnh như sau:
```
MOV AL, 77h      ; gửi số N vào AL
OUT 42h, AL      ; đưa N vào bộ đếm 2 có địa chỉ 42h
```

## 4. Đọc bộ đếm của 8253

Đọc bộ đếm có thể thực hiện theo một trong hai cách:

- Đọc trực tiếp bởi một (một byte) hay hai (hai byte) lệnh IN AL, địa chỉ cổng.

- Đọc sau khi chốt bộ đếm, ghi thêm lệnh chốt bộ đếm vào thanh ghi điều khiển với lời lệnh chốt (RW1,RW0 = 00) bộ đếm nào (SC1,SC0). Kiểu đọc này tin cậy vì đọc giá trị lúc chốt số liệu. Ví dụ trên, muốn đọc bộ đếm 2 ta có lời lệnh chốt bộ đếm 10000000B = 80h (các bit $d_3$÷$d_0$ không dùng). Lệnh đọc như sau:

```
MOV AL, 80h      ; gửi lệnh chốt vào AL
OUT 43h, AL      ; đưa ra thanh ghi lệnh
IN AL, 42h       ; đọc byte thấp bộ đếm 2 vào AL
IN AL, 42h       ; đọc byte thấp bộ đếm 2 vào AL.
```

## 5. Đọc trạng thái và số liệu của 8254

Vi mạch 8254 có thể đọc trở lại (read-back) trạng thái với các bit như sau:

| $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|
| pin | null | RW1 | RW0 | M2 | M1 | M0 | BCD |

Trong đó:

- Pin là trạng thái của đầu ra OUTx với giá trị 1 (mức cao) hay giá trị 0 (mức thấp).

- Null cho biết bộ đếm đã được nạp giá trị ban đầu chưa với giá trị 1 (chưa nạp), hay 0 (đã nạp).

- Các giá trị khác như thanh ghi điều khiển. Các giá trị này dùng để đọc lại các giá trị của thanh ghi điều khiển mà ta đã ghi để kiểm tra.

Trước khi đọc trạng thái hay số đếm, ta phải ghi lệnh đọc thanh ghi tương tự lệnh chốt ở trên, với các bit như sau:

| $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | count | status | count2 | count1 | count0 | 0 |

- Bit COUNT: dùng để xác định đọc bộ đếm (= 0).

- Bit STATUS: dùng để xác định đọc trạng thái (= 0).

- Các bit COUNT2-COUNT0 chỉ bộ đếm được chọn để đọc số đếm.

Lệnh đọc số liệu của bộ đếm 0 như sau:

```
MOV AL, E2h      ; 11100010B, đếm (d5 = 0), bộ đếm 0 (d1 = 1).
OUT 43H, AL      ; đưa ra thanh ghi điều khiển
IN AL, 40h       ; đọc bộ đếm 0.
```

Lệnh đọc trạng thái như sau:

```
MOV AL, D0h      ; 11100000B, trạng thái (d4 = 0)
OUT 43h, AL      ; đưa ra thanh ghi điều khiển
IN AL, 43h       ; đọc trạng thái vào AL.
```

Căn cứ vào dạng thanh ghi trạng thái ở trên, ta biết được chế độ đã ghi và pin, null.

## 6. Ví dụ về lập trình

*a) Ví dụ 1.* Chương trình ghi số liệu và đọc bộ đếm 2, phát âm ở chế độ M(3), tần số như ví dụ trên.

```
MOV AL, B7h        ; 10110111B = B7h, bộ đếm 2, byte thấp, byte cao
OUT 43h, AL        ; ghi ra thanh ghi điều khiển
MOV AL, 19Dh       ; gán số 19D vào AL
OUT 42h, AL        ; ghi vào byte thấp bộ đếm 2
MOV AL, 01D        ; gán số 01D và AL
OUT 42h, AL        ; ghi vào byte cao của bộ đếm 2
MOV AL, 80h        ; chốt bộ đếm
OUT 43h, AL        ; ghi ra thanh ghi điều khiển
IN AL, 42h         ; đọc byte thấp
IN AL, 42h         ; đọc byte cao
INT 20h            ; kết thúc chương trình.
```

*Chú ý:* - Khi chạy trong DEBUG cần bỏ chữ "h" sau các số liệu.

- Cần dùng lệnh P↵ của DEBUG để cho chạy từng tiến trình và theo dõi kết quả từng lệnh.

***b) Ví dụ 2.*** Chương trình kiểm tra các chế độ phát xung.

Muốn kiểm tra chế độ phát xung, ta dùng tai để phân biệt tần số của âm phát ra loa cao hay thấp. Ta phải lập trình cho loa kêu và tắt loa (xem giải thích ở phần phát âm trực tiếp ra loa) và cho loa nghỉ (tắt) trong một thời gian trễ (xem ở dưới). Chương trình như sau:

```
        MOV AL, B7h        ; 10110111B = B7h, bộ đếm 2, byte thấp, byte cao
        OUT 43h, AL        ; ghi ra thanh ghi điều khiển
        MOV AL, 19Dh       ; gán số 19D vào AL
        OUT 42h, AL        ; ghi vào byte thấp bộ đếm 2
        MOV AL, 01D        ; gán số 01D và AL
        OUT 42h, AL        ; ghi vào byte cao của bộ đếm 2
A:      IN AL, 61h         ; đọc cổng điều khiển loa
        OR AL, 03h         ; xác lập các bit d1 = d0 = 1
        OUT 61h, AL        ; đưa ra cổng điều khiển loa
        MOVCX, 0000h       ; thanh ghi byte cao của trễ thời gian
        MOV DX, 00FFFh     ; thanh ghi byte cao của trễ thời gian
        CALL RELAY         ; gọi chương trình con trễ
        IN AL, 61h         ; đọc cổng điều khiển loa
        AND AL,FCh         ; xác lập các bit d1 = d0 = 1
        OUT 61h, AL        ; đưa ra cổng điều khiển loa
        MOVCX, 0000h       ; thanh ghi byte cao của trễ thời gian
        MOV DX, 00FFFh     ; thanh ghi byte cao của trễ thời gian
        CALL RELAY         ; gọi chương trình con trễ
        INC BX             ; tăng số lần phát âm
        CMP BX, 000Fh      ; đã phát 15 lần chưa
        JNE A              ; trở về A để phát lại nếu chưa đủ 15 lần
        INT 20h            ; kết thúc
RELAY:  MOV AH, 86h        ; chức năng trễ thời gian
        INT 15h            ;  ngắt trễ thời gian
        RET
```

*Chú ý:* - Khi chạy trong DEBUG cần bỏ chữ "h" và "D" sau các số liệu.

- Có thể dùng lệnh P↵ của DEBUG để cho chạy từng tiến trình để theo dõi kết quả từng lệnh và lệnh G↵ để chạy cả chương trình.

- Thay các giá trị của M(i) trong lời lệnh điều khiển để theo dõi tần số của âm với các thông số như bảng 7.4.

***Bảng 7.4.* Các lời lệnh điều khiển chế độ phát xung**

| Chế độ phát | M2 | M1 | M0 | BCD | B |
|---|---|---|---|---|---|
| Tạo mức trễ $Nt_0$ | 0 | 0 | 0 | 1 B1h | 0 B0h |
| Tạo xung đơn âm, độ kéo dài Nt0 | 0 | 0 | 1 | 1 B3h | 0 B2h |
| Tạo chuỗi xung âm, kéo dài $t_0$, trễ $(N-1)t_0$ | 0 | 1 | 0 | 1 B5h | 0 B4h |
| Tạo chuỗi xung âm, kéo dài N/2, nghỉ N/2 | 0 | 1 | 1 | 1 B7h | 0 B6h |
| Tạo một xung âm, kéo dài $t_0$, trễ $Nt_0$, điều khiển bằng mặt tăng của xung nhịp | 1 | 0 | 0 | 1 B9h | 0 B8h |
| Tạo một xung dương, kéo dài $Nt_0$, điều khiển bằng mặt tăng của xung nhịp. | 1 | 0 | 1 | 1 BBh | 0 BAh |

## 7.2.2. LẬP TRÌNH ĐỌC VÀ GHI TRỰC TIẾP THỜI GIAN

### 1. Đọc và ghi vào bộ nhớ của CMOS

Bộ nhớ của đồng hồ CMOS có 64 ngăn hay thanh ghi. Chúng được ghi và đọc các thông tin liên quan tới thời gian và hệ thống (xem các bảng thống kê). Chúng ta cần đọc nó để biết thông tin, đặc biệt về thời gian (tần số nhịp, giây, phút, giờ, ngày, tháng, năm thế kỷ) và trạng thái hoạt động của đồng hồ. Ngược lại, ta cũng cần ghi lại các tin về sự thay đổi ngắt nhịp, thay đổi thời gian.

***a) Cách gọi địa chỉ ngăn nhớ của đồng hồ***

Bộ nhớ của đồng hồ thời gian thực CMOS có những 64 ngăn nhớ nhưng chỉ có 2 địa chỉ là:

- Địa chỉ 70h cho thanh ghi chỉ thị, tức chỉ số hay số thứ tự trong bộ nhớ của đồng hồ. Do đó, muốn gọi địa chỉ N nào đó của ngăn nhớ, ta chỉ cần ghi chỉ số N của ngăn nhớ vào thanh ghi chỉ số với địa chỉ 70h. Thanh ghi chỉ số này chính là bộ đệm địa chỉ cho các ngăn nhớ của đồng hồ.

- Địa chỉ 71h cho thanh ghi dữ liệu. Thanh ghi này là chung cho mọi ngăn nhớ, chính là bộ đệm dữ liệu của các ngăn nhớ và dữ liệu của các ngăn nhớ sẽ tự động trao đổi với thanh ghi dữ liệu này.

***b) Lệnh đọc dữ liệu của ngăn nhớ***

Muốn đọc dữ liệu của ngăn nhớ có chỉ số N, ta thực hiện như sau:

- Gọi địa chỉ ngăn nhớ bằng ghi chỉ số N vào thanh ghi chỉ số với địa chỉ 70h bằng các lệnh: MOV AL, N và OUT 70, AL.

- Đọc dữ liệu từ ngăn nhớ vào thanh ghi AL bằng lệnh IN AL, 71h.

***c) Lệnh ghi dữ liệu vào ngăn nhớ***

Việc ghi dữ liệu này cũng có:

- Gọi địa chỉ ngăn nhớ có chỉ số N vào thanh ghi chỉ số có địa chỉ 70h bằng các lệnh:

MOV AL, N và OUT 70h, AL.

- Ghi dữ liệu DATA vào thanh ghi AL bằng lệnh MOV AL, DATA.

- Ghi dữ liệu từ thanh ghi AL ra ngăn nhớ bằng lệnh OUT 71h, AL.

## 2. Đọc thời gian hay trạng thái

Đọc thời gian hay trạng thái chính là đọc thanh ghi tương ứng với chỉ số N. Muốn đọc thanh ghi nào, ta chỉ cần thay số N tương ứng.

Ví dụ đọc thời gian hiện hành, ta thay số N = 0 cho giờ, 2 cho phút, 4 cho giờ, 6 cho thứ, 7 cho ngày, 8 cho tháng, 9 cho năm và 32h cho năm (thế kỷ).

Ví dụ đọc thanh ghi trạng thái A, cho số hiệu N (hay địa chỉ nhớ riêng trong CMOS) của thanh ghi trạng thái A, đọc thanh ghi trạng thái B cho cho số hiệu N (hay địa chỉ nhớ riêng trong CMOS) của thanh ghi B, đọc thanh ghi trạng thái C cho số hiệu N (hay địa chỉ nhớ riêng trong CMOS) của thanh ghi C và đọc thanh ghi trạng thái D cho số hiệu N (hay địa chỉ nhớ riêng trong CMOS) của thanh ghi D.

Với chỉ số N khác nhau, ta có kết quả đọc thời gian như bảng 7.5.

Chương trình đọc thanh ghi chỉ số N như sau:

```
MOV AL, Nh      ; chuyển chỉ số Nh cho AL
OUT 70h, AL     ; đưa chỉ số N cho thanh ghi chỉ số
IN AL,71h       ; đọc nội dung thanh ghi dữ liệu vào AL
INT 20h         ; kết thúc chương trình.
```

Chú ý: - Kết quả đọc ra thanh ghi là con số dạng BCD (cho dễ nhận biết) do đã ghi 0 vào bit $d_2$ của thanh ghi trạng thái B.

- Chương trình trên chạy với DEBUG phải soạn thảo không có chữ "h" sau con số.

- Chỉ cần thay giá trị N theo bảng trên, ta có thể đọc được tất cả các thanh ghi để biết được các thông số liên quan tới thời gian. Chính vì vậy mà trong Windows có lịch và đồng hồ chỉ thời gian (thư mục My Computer/Control panel/Date hay My Computer/ Control panel/time).

***Bảng 7.5.*** **Số hiệu (địa chỉ nhớ riêng) của các thanh ghi thời gian**

| N | Thanh ghi | Chỉ thị |
|---|---|---|
| 0h | Thanh ghi giây | Giây cho đồng hồ ra dạng số BCD |
| 2h | Thanh ghi phút | Phút cho đồng hồ ra dạng số BCD |
| 4h | Thanh ghi giờ | Giờ cho đồng hồ ra dạng số BCD |
| 6h | Thanh ghi thứ | Ngày của tuần (thứ) ra dạng số BCD |
| 7h | Thanh ghi ngày | Ngày ra dạng số BCD |
| 8h | Thanh ghi tháng | Tháng ra dạng số BCD |
| 9h | Thanh ghi năm | Hai số cuối của năm ra dạng số BCD |
| 32h | Thanh ghi thế kỷ | Hai số đầu của năm (thế kỷ) ra dạng số BCD |

## 3. Đọc và kiểm tra trạng thái

Đọc trạng thái chính là đọc thanh ghi trạng thái A, B, C, D, cũng giống đọc dữ liệu ở trên. Kiểm tra trạng thái là kiểm tra các bit bằng 0 hay1 với lệnh AND để chặn các bit không cần kiểm tra và một trong hai lệnh TEST hay CMP xem bit đó là 1 hay không? Sau các lệnh kiểm tra này phải có các lệnh chuyển chương trình (nhảy JUMP và gọi CALL) có điều kiện, phụ thuộc vào kết quả kiểm tra (lớn hơn, bằng, nhỏ hơn, bằng 0, dương, âm v.v...).

## 4. Ghi thời gian

Bằng cách ghi vào thanh ghi, ta có thể ghi lại thời gian (chữa thời gian cũ không đúng) và đặt thời gian cho báo thức. Các số liệu về thời gian này cũng được ghi dưới dạng BCD (đã ghi giá trị 0 vào bit $d_2$ của thanh ghi trạng thái B). Giá trị chỉ số N của thanh ghi trong bộ nhớ CMOS tương ứng với các thanh ghi như bảng 7.6.

***Bảng 7.6.*** **Các thanh ghi thời gian**

| Chỉ số N | Thanh ghi | Giá trị ghi |
|---|---|---|
| 1 | Giây | Giây ra dạng số BCD |
| 3 | Phút | Phút ra dạng số BCD |
| 5 | Giờ | Giờ ra dạng số BCD |

Chương trình ghi lại thời gian hay đặt thời gian báo thức vào thanh ghi chỉ số N như sau:

```
MOV AL, Nh          ; chuyển chỉ số Nh cho AL
OUT 70h, AL         ; đưa chỉ số N cho thanh ghi chỉ số
MOV AL, GIÁ TRỊ MỚI  ; GIÁ TRỊ MỚI = giá trị mới của thời gian
OUT 71h, AL         ; đưa giá trị mới vào thanh ghi N
INT 20h             ; kết thúc chương trình
```

*Chú ý:* - Nếu thay N ở bảng trên và thay GIÁ TRỊ MỚI bằng giá trị muốn ghi, ta có thể ghi được tất cả các thời gian được đặt lại và báo thức.

- Chương trình có thể chạy trên DEBUG (nhập vào phải bỏ chữ "h" sau chữ số).

## 5. Các ví dụ

***a) Đọc các thanh ghi trạng thái và chẩn đoán***

```
        MOV AL,0Ah      ; đọc thanh ghi trạng thái A
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 0Bh     ; đọc thanh ghi trạng thái B
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 0Ch     ; đọc thanh ghi trạng thái C
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 0Dh     ; đọc thanh ghi trạng thái D
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 0EH     ; đọc thanh ghi chẩn đoán E
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 0FH     ; đọc thanh ghi trạng thái khi tắt máy
        CALL A          ; gọi chương trình con A để đọc
        MOV AL,10h      ; đọc mô tả các đĩa mềm
        CALL A          ; gọi chương trình con A để đọc
        MOV AL, 12h     ; đọc mô tả các đĩa cứng
        CALL A          ; gọi chương trình con A để đọc
        INT 20h         ; kết thúc chương trình
A:      OUT 70, AL      ; chương trình con đọc
        IN AL, 71h      ; đọc thanh ghi vào AL
        RET
```

*Chú ý:* - Khi nhập vào với DEBUG, ta bỏ chữ "h" sau các số.

- Thay chữ A trong các lệnh CALL A bằng con số chỉ địa chỉ (IP) của lệnh OUT 70h, AL của chương trình con.

- Cho chạy chương trình với lệnh P của DEBUG, theo dõi giá trị của AL để biết được dạng các thanh ghi trạng thái và các ngăn nhớ khác.

- Thay đổi chỉ số thanh ghi tương ứng N, có thể đọc được tất cả 64 ngăn nhớ hay thanh ghi của đồng hồ thời gian thực.

***b) Chương trình đọc thời gian thực***

Cũng tương tự chương trình đọc các thanh ghi trạng thái, ta chỉ cần thay chỉ số N tương ứng chỉ số của thanh ghi tương ứng và gọi chương trình con để đọc dữ liệu về thời gian vào thanh ghi AL.

```
        MOV AL, 32h     ; thanh ghi chỉ thế kỷ (19)
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 09h     ; thanh ghi chỉ năm (99)
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 08h     ; thanh ghi chỉ tháng
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 07h     ; thanh ghi chỉ ngày
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 06h     ; thanh ghi chỉ thứ
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 04h     ; thanh ghi chỉ giờ
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 02h     ; thanh ghi chỉ phút
        CALL A          ; gọi chương trình con để đọc
        MOV AL, 00h     ; thanh ghi chỉ giây
        CALL A          ; gọi chương trình con để đọc
        INT 20h         ; kết thúc chương trình
A:      OUT 70h, AL     ; đưa lời điều khiển
        IN AL, 71h      ; đọc vào AL
        RET
```

Chúng ta cho chạy chương trình trên giống chương trình đọc các thanh ghi trạng thái. Chúng ta có thể kiểm tra kết quả đọc trên bằng việc dùng lịch và đồng hồ của máy tính (thư mục My computer/control panel/date-time). Chúng ta cần lưu ý rằng giá trị của giây sẽ khác đi, vì thời gian đã trôi đi và ta phải gọi đồng hồ.

***c) Chương trình ghi thời gian thực***

Muốn chỉnh đồng hồ, tức ghi giá trị thời gian đúng cho đồng hồ, ta có các cách sau:

- Thay đổi ngay trên đồng hồ và lịch, sử dụng các chương trình phục vụ có sẵn của Windows. Phương pháp này nhanh, dễ thao tác, không cần phải lập trình, nhưng ở đây, chúng ta muốn học về lập trình nên dùng hai cách sau:

- Lập trình trực tiếp, tức ghi thời gian vào các thanh ghi của đồng hồ.

- Lập trình sử dụng ngắt của BIOS (INT 1Ah) hay DOS (các chức năng 2A÷2D). Chúng ta sẽ xét cách lập trình này ở mục sau.

Chương trình ghi thời gian 11giờ, 35 phút, 50 giây, ngày thứ hai, ngày 5 tháng 12 năm 1995 cho đồng hồ như sau:

```
MOV AL, 33h     ; ghi thế kỷ (19)
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 19      ; thế kỷ 19
OUT 70, AL      ; ghi thế kỷ 19
MOV AL, 09      ; ghi năm
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 95      ; năm 95
OUT 70, AL      ; ghi năm 95
MOV AL, 08      ; ghi tháng
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 12      ; tháng 12
OUT 70, AL      ; ghi tháng 12
MOV AL, 07      ; ghi ngày
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 05      ; ngày 05
OUT 70, AL      ; lệnh ghi ngày 05
MOV AL, 06      ; ghi thứ
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 01      ; thứ hai
OUT 70, AL      ; lệnh ghi thứ
MOV AL, 04      ; ghi giờ
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 11      ; 11 giờ
OUT 70, AL      ; lệnh ghi giờ
MOV AL, 02      ; ghi phút
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 35      ; 35 phút
OUT 70, AL      ; lệnh ghi phút
MOV AL, 00      ; ghi giây
OUT 71, AL      ; ghi lệnh vào thanh ghi điều khiển
MOV AL, 50      ; 50 giây
OUT 70, AL      ; lệnh ghi giây
INT 20          ; kết thúc chương trình.
```

## 7.3. LẬP TRÌNH THỜI GIAN DÙNG NGẮT

### 7.3.1. LẬP TRÌNH THỜI GIAN DÙNG NGẮT CỦA BIOS

#### 1. Các ngắt của BIOS cho thời gian

BIOS có các ngắt liên quan tới thời gian là:

- *Ngắt INT 08h:* Ngắt này được khởi động bởi tần số ngắt cứng (1193180 xung/655356 = 18,2) do xung ra của bộ đếm thời gian 8253/8254. Vì tần số này không phụ thuộc vào tần số đồng hồ của máy tính, nên ngắt này thường được dùng để tính thời gian. BIOS hướng ngắt này tới một

thủ tục thực hiện tính thời gian và dừng động cơ của ổ đĩa mềm nếu sự truy cập đĩa không thành công sau một thời gian nào đó. Ngắt này có vùng nhớ đệm địa chỉ 0040:006Ch để chứa tin tức của ngắt.

- *Ngắt INT 1Ch:* Ngắt này được gọi sau một chu kỳ thời gian 18,2 lần trong một giây của đồng hồ. Ngắt có vùng đệm ở địa chỉ nhớ 0000:0070h.

- *Ngắt INT 1Ah*: Dành cho thời gian, để đọc, đặt lại thời gian và ghi thời gian báo thức. Bảng 7.7 thống kê các ngắt của BIOS cho thời gian.

***Bảng 7.7.* Các chức năng ngắt INT 1AH của BIOS cho thời gian**

| Chức năng AH | Tên | Các thông số |
|---|---|---|
| 00h | Đọc bộ đếm hiện tại của đồng hồ | - Vào: AH = 00h<br>- Ra: CX = số nhịp đếm cao từ khi Reset<br>DX = số nhịp đếm thấp từ khi Reset<br>AL = 00 không bị tràn từ khi Reset |
| 01h | Đặt bộ đếm hiện tại của đồng hồ | - Vào: AH = 01h<br>CX = số nhịp đếm cao<br>DX = số nhịp đếm thấp |
| 02h | Đọc đồng hồ thời gian thực<br>(chỉ PC /AT và PS/2) | - Vào AH = 02h<br>- Ra: CH = giờ theo BCD<br>CL = phút theo BCD<br>DH = giây theo BCD<br>DL = 01h nếu có cất giữ ngày<br>CF = CY = 1 nếu đồng hồ không chạy |
| 03h | Đặt thời gian cho CMOS<br>(chỉ PC /AT và PS/2) | - Vào: AH = 03h<br>CH, CL = giờ, phút theo BCD<br>DH = giây theo BCD<br>DL = 1 cho phép cất giữ ngày<br>- Ra: CF = CY = 1 nếu đồng hồ không chạy |
| 04h | Đọc ngày từ CMOS<br>(chỉ PC /AT và PS/2) | - Vào: AH = 04h<br>- Ra: CH = thế kỷ (19 hay20) ra BCD<br>CL = năm (2 số cuối) ra BCD<br>DH = tháng ra BCD<br>DL = ngày ra BCD<br>CF = CY = 1 nếu đồng hồ không chạy |
| 05h | Đặt ngày cho CMOS<br>(chỉ PC /AT và PS/2) | - Vào: AH = 05h<br>CH = thế kỷ (19 hay 20) ra BCD<br>CL = năm theo BCD<br>DH = tháng ra BCD<br>DL = ngày ra BCD |
| 06h | Đặt giờ báo thức cho thời gian thực CMOS<br>Đặt địa chỉ chương trình con của báo thức vào véctơ ngắt 4Ah trước khi dùng ngắt này. | - Vào: AH = 06h<br>CH = giờ theo BCD<br>CL = phút theo BCD<br>DH = giây theo BCD<br>CF = CY = 1 nếu đồng hồ không chạy<br>hoặc đã đặt giờ báo thức. |
| 07h | Đặt lại giờ báo thức mới cho CMOS | - Vào: AH = 07h<br>- Ra: không |
| 08h | Đọc giờ và trạng thái của báo thức | - Vào: AH = 08h<br>- Ra: CH = giờ theo BCD<br>CL = phút theo BCD<br>DH = giây theo BCD<br>DL = 00h - báo thức chưa thực hiện<br>= 01h - báo thức đã thực hiện |

Sau đây ta sẽ xét chi tiết các chức năng thiết lập đồng hồ, đọc và đặt lại thời gian. Riêng các chức năng ta sẽ xét ở mục lập trình báo thức ở dưới, sau khi đã biết lập trình phát âm thanh ra loa.

## 2. Lập trình thiết lập đồng hồ

Các chức năng AH = 00h và 01h cho phép đọc nội dung bộ đếm và đặt lại bộ đếm đồng hồ. Chức năng 01h đọc nội dung bộ đếm để so sánh với nội dung tính toán thời gian kể từ khi bật máy hoặc từ sau nửa đêm, xem đồng hồ chạy có đúng không với dữ liệu là đồng hồ tăng 18,2 lần trong 1 giây. Nếu đồng hồ chạy sai, phải thiết đặt lại bằng chức năng AH = 01h cho đúng. Cũng cần kiểm tra bit $d_7 = 0$ (cho phép khởi phát lại ngày và giờ) của thanh ghi trạng thái B.

## 3. Lập trình đọc thời gian

### *a) Lập trình đọc giờ*

Ngắt AH = 02h cho phép đọc giờ, phút, giây của đồng hồ thời gian thực sau khi đã kiểm tra:

- Bit $d_7 = 1$ của thanh ghi trạng thái A xem các byte nhớ đang cập nhật không.
- Kiểm tra bit $d_7 = 0$ của thanh ghi trạng thái B để xem có cho phép đặt lại thời gian.
- Bit $d_7 = 1$ của thanh ghi trạng thái D xem đồng hồ còn chạy hay không.
- Bit $d_4 = 1$ (kết thúc việc cập nhật thời gian) của thanh ghi trạng thái B.

Sau khi gọi ngắt, phải kiểm tra:

- Xem cờ CF = 1 hay không. Nếu CF = 1, đồng hồ hết pin, số ghi trên các thanh ghi tương ứng là số cũ, trước khi pin hết.
- Các số thuộc dạng nhị phân (bit $d_2 = 1$) hay BCD (bit $d_2 = 0$). Kết quả về giờ, phút, giây và phần trăm của giây đặt trên các thanh ghi CH, CL, DH, DL.

### *b) Lập trình đọc ngày*

Ngắt AH = 04h cho phép đọc ngày, tháng năm, thế kỷ của đồng hồ thời gian thực. Cũng giống như đọc giờ, phút, giây, phải kiểm tra:

- Bit $d_7 = 1$ của thanh ghi trạng thái A xem các byte nhớ đang cập nhật không.
- Bit $d_7 = 0$ (cho phép đặt lại thời gian) và bit $d_4 = 1$ (kết thúc việc cập nhật thời gian) của thanh ghi trạng thái B.
- Bit $d_7 = 1$ của thanh ghi trạng thái D xem đồng hồ còn chạy hay không.

Sau khi gọi ngắt, phải kiểm tra:

- Xem cờ CF = 1 hay không. Nếu CF = 1, đồng hồ hết pin, số ghi trên các thanh ghi tương ứng là số cũ, trước khi pin hết.
- Các số thuộc dạng nhị phân (bit $d_2 = 1$) hay BCD (bit $d_2 = 0$).

Các số về thế kỷ (hai số cuối của năm), năm, tháng, ngày được ghi trên các thanh ghi CH, CL, DH, DL.

## 4. Lập trình đặt giờ và ngày

Các ngắt AH = 03h và AH = 05h cho phép đặt lại giờ, phút, giây, ngày, tháng, năm của đồng hồ thời gian thực.

### *a) Lập trình đặt giờ, phút, giây*

Chức năng AH = 03h cho phép đặt lại giờ, phút, giây cho đồng hồ nếu đồng hồ chỉ sai. Trước khi đặt, phải:

- Kiểm tra bit d7 = 1 của thanh ghi trạng thái A xem các byte nhớ đang cập nhật không.

- Kiểm tra bit d7 = 0 (cho phép đặt lại thời gian) và d2 = 0 (số dạng BCD) của thanh ghi trạng thái B.

- Kiểm tra bit d7 = 1 của thanh ghi trạng thái D xem đồng hồ còn chạy hay không.

Cần ghi các số liệu mới về giờ, phút, giây, giờ mùa hè (DL = 1) hay mùa đông (DL = 0) dạng BCD vào các thanh ghi CH, CL, DH, DL.

***b) Lập trình đặt lại ngày, tháng, năm***

Chức năng AH = 05h cho phép đặt lại giờ, phút giây cho đồng hồ CMOS nếu đồng hồ chỉ sai. Trước khi đặt, phải:

- Kiểm tra bit d7 = 1 của thanh ghi trạng thái A xem các byte nhớ đang cập nhật không.

- Kiểm tra bit d7 = 0 (cho phép đặt lại thời gian) và d2 = 0 (số dạng BCD) của thanh ghi trạng thái B.

- Kiểm tra bit d7 = 1 của thanh ghi trạng thái D xem đồng hồ còn chạy không.

Cần ghi các số liệu mới về thế kỷ (hai số cuối của năm), năm, tháng, ngày dạng BCD vào các thanh ghi CH, CL, DH, DL.

## 7.3.2. LẬP TRÌNH THỜI GIAN SỬ DỤNG NGẮT INT 21H CỦA DOS

### 1. Các ngắt liên quan tới thời gian của DOS

DOS có các ngắt liên quan tới thời gian như bảng 7.8.

***Bảng 7.8.* Các ngắt của DOS về thời gian**

| Chức năng | Giống BIOS | Hành động | Thông số |
|---|---|---|---|
| 2Ah | 04h | Đọc ngày, tháng | - Vào: AH = 2Ah<br>- Ra: AL = thứ trong tuần (00 = chủ nhật, 01 = thứ hai )<br>CX = năm, DH = tháng, DL = ngày |
| 2Bh | 05h | Đặt lại ngày, tháng | - Vào: AH = 2Bh, CX = năm, DH = tháng, DL = ngày<br>- Ra: AL = 00 đặt đúng, FFh - vào sai ngày tháng |
| 2Ch | 02h | Đọc giờ | - Vào: AH = 2Ch<br>- Ra: CH = giờ, CL = phút, DH = giây, DL = phần trăm giây, AL = 00 đọc đúng |
| 2Dh | 03h | Đặt giờ | - Vào:AH = 2Dh, CH = giờ, CL = phút, DH = giây, DL = 1/100 giây<br>- Ra: AL = 00 đặt đúng = FFh vào sai giờ, phút giây |

Chúng ta cần chú ý rằng các giá trị về thời gian trong các ngắt này đều ở hệ hexadecimal.

### 2. Đọc thời gian

***a) Đọc ngày, tháng, năm***

Chức năng AH = 2Ah. Chức năng này tương tự chức năng AH = 04h của BIOS nhưng thực hiện chậm hơn.

Ví dụ: Chương trình con đọc ngày, tháng, năm.

```
PUSH BP          ; gửi BP vào ngăn xếp
MOV BP, SP       ; gửi SP vào BP
MOV AH, 2Ah      ; đọc ngày hiện tại
INT 21h
```

```
PUSH CX          ; giữ thế kỷ, năm trong ngăn xếp
PUSH DX          ; giữ tháng, ngày trong ngăn xếp
MOV CX, [BP+8]   ; đặt năm
MOV DL, [BP+6]   ; đặt ngày
MOV DH, [BP+4]   ; đặt tháng
MOV AH, 2Bh      ; chức năng đặt ngày mong muốn
INT 21h          ; gọi ngắt
MOV AH, 2Ah      ; trả lại ngày cho DOS
INT 21h          ; (AL = thứ)
POP DX           ; lấy lại ngày, tháng hiện tại
POP CX           ; lấy lại năm hiện tại
PUSH AX          ; gửi thứ vào ngăn xếp
MOV AH, 2Bh      ; đặt ngày hiện tại
INT 21h
POP AX           ; AL = thứ
MOV AH, 00h      ; AH = thứ
POP BP           ; lấy lại BP
RET              ; trở về từ chương trình con.
```

***b) Đọc giờ, phút, giây***

Chức năng AH = 2Ch. Chức năng này tương tự chức năng AH = 02h của BIOS nhưng thực hiện chậm hơn.

## 3. Đặt lại thời gian

***a) Đặt lại ngày, tháng, năm***

Chức năng AH = 2Bh. Chức năng này tương tự chức năng AH = 02h của BIOS nhưng thực hiện chậm hơn.

*Ví dụ:* Đặt lại ngày 20 tháng 3 năm 1999.

```
MOV AH, 2Bh      ; chức năng đặt lại ngày, tháng, năm
MOV CH, 13h      ; đặt thế kỷ19
MOV CL, 63h      ; đặt năm 99
MOV DH, 03h      ; đặt tháng 03
MOV DL, 14h      ; đặt ngày 20
INT 21h          ; gọi ngắt
INT 20h
```

***b) Đặt lại giờ, phút, giây***

Chức năng AH = 2Dh. Chức năng này tương tự chức năng AH = 02h của BIOS nhưng thực hiện chậm hơn.

*Ví dụ:* Đặt lại 15 giờ, 27 phút, 55 giây.

```
MOV AH, 2Dh      ; chức năng đặt lại giờ, phút, giây
MOV CH, 0Fh      ; đặt 15 giờ
MOV CL, 1Bh      ; đặt 27 phút
MOV DH, 37h      ; đặt 55 giây
```

```
MOV DL,00        ; đặt 00 phần trăm giây
INT 21h          ; gọi ngắt
INT 20h          ; kết thúc chương trình.
```

*Chú ý:* - Chương trình chạy trong DEBUG phải bỏ chữ "h" khi nhập vào.

- Các số về giờ, phút, giây dạng hexadecimal.

## 7.4. LẬP TRÌNH CHO TRỄ THỜI GIAN

Trễ thời gian là vấn đề cần thiết cho máy vi tính, người ta cần lập trình trễ thời gian để:

- Chờ thực hiện một lệnh tiếp theo của chương trình, trong các hành động cần chờ phản ứng của thiết bị như trao đổi tin.

- Đo thời gian trễ như đo độ trễ của một tín hiệu trên đường dây trễ LC, thời gian quá độ của mạch điện v.v...

- Báo thức bằng đồng hồ của máy vi tính.

Về nguyên tắc, người ta dùng đồng hồ (hệ thống hay thời gian thực CMOS) để lập trình tạo trễ. Chúng ta cũng có hai cách tạo trễ: lập trình trực tiếp và lập trình sử dụng ngắt.

### 7.4.1. LẬP TRÌNH TẠO TRỄ TRỰC TIẾP

Tạo trễ trực tiếp có các phương pháp sau:

- Cho chương trình thực hiện một số lệnh mà đã biết thời gian thực hiện lệnh đó. Nếu thời gian trễ của lệnh là $t_0$, thì thời gian trễ cho N lệnh là $Nxt_0$.

- Cho đồng hồ hệ thống đếm trễ N xung để phát một mức (chế độ M(0)), một xung trễ (chế độ M(2), M(4), M(5)).

#### 1. Tạo trễ bằng tính thời gian trễ

Phương pháp này cần:

- Ghi số lệnh N cần thực hiện trễ vào thanh ghi đếm CX.

- Thực hiện lệnh bất kỳ mà thời gian trễ cần thiết đã biết.

- Giảm số đếm N đi 1 đơn vị sau mỗi lần thực hiện lệnh trễ.

- Quay trở lại thực hiện trễ bằng các lệnh quay vòng LOOP, so sánh và nhảy hoặc gọi có điều kiện.

Phương pháp này thường dùng trong lập trình không đòi hỏi thời gian trễ chính xác lắm.

*Ví dụ:* Chương trình con tạo trễ RELAY có độ trễ $Nxt_0$ ($t_0$ = thời gian thực hiện các lệnh MOV AX, FFh, DEC CX, CMP CX, 00h, JNZ A và RET) như sau:

```
RELAY:  MOV CX, N       ; ghi số đếm N vào thanh ghi đếm CX
A:      MOV AX, FFh     ; lệnh cần thời gian trễ t0 (có thể thay lệnh với t0 khác)
        DEC CX          ; giảm N đi 1
        CMP CX, 00h     ; so sánh với 00h
        JNZ A           ; nhảy trở về nhãn A nếu CX chưa bằng 00h
        RET             ; trở về từ chương trình con.
```

*Chú ý:* Nếu chạy chương trình trong DEBUG phải:

- Bỏ chữ "h" sau các chữ số.

- Thay nhãn A trong lệnh JNZ A bằng địa chỉ của lệnh có nhãn A (lệnh MOV AX, FFh).

- Có thể kiểm tra độ trễ trên bằng cách thực hiện lệnh đưa một ký tự nào đó ra màn hình (như ví dụ sau).

## 2. Tạo trễ bằng đếm trễ trong bộ đếm

Phương pháp trên có ưu điểm l đơn giản, nhưng phải biết được thời gian $t_0$ của lệnh làm trễ (cần bảng lệnh để tra cứu). Phương pháp đếm trễ bằng bộ đếm có ưu điểm là biết trước được $t_0$ = 1/18,2 giây = 54 miligiây để tính $Nxt_0$.

Phương pháp này cần:

- Ghi số đếm N vào các thanh ghi giây, phút, giờ của bộ đếm.

- Đọc số đếm Ni sau khi đếm.

- So sánh số đếm Ni với 0000h, nếu khác không, tiếp tục lệnh đọc số đếm.

- Kết thúc đếm trễ khi Ni = 0000h.

*Ví dụ:* Chương trình con tạo trễ N (biểu diễn dạng BCD) giây bằng ghi và đọc số đếm một cách trực tiếp, khi kết thúc chương trình, in chữ "A" ra màn hình.

```
        MOV AL, B2h       ; chế độ M(1), phát mức xung trễ Nxt0
        OUT 43h, AL       ; đưa ra thanh ghi điều khiển
        MOV AL, Ncao      ; chuyển byte cao của Ncao vào AL
        OUT 42h, AL       ; ghi vào bộ đếm
        MOV AL, Nthap     ; chuyển byte thấp Nthap của N vào AL
        OUT 42h, AL       ; đưa ra bộ đếm
        IN AL, 61h        ; đọc cổng điều khiển bộ đếm và loa
        AND AL, FCh       ; xóa các bit d1 = -d0 = 0
        OUT 61h, AL       ; đưa ra cổng 61h
A:      IN AL,61h         ; điều khiển đếm bằng đọc cổng 61h (8255)
        OR AL, 03h        ; xác lập các bit d1 = d0 = 1
        OUT 61h, AL       ; đưa ra cổng 61h
        MOV AL, 82h       ; chốt bộ đếm 2, chế độ M(1)
        OUT 43h, AL       ; đưa ra thanh ghi điều khiển
        IN AL, 42h        ; đọc số đếm lùi ở byte thấp
        IN AL, 42h        ; đọc số đếm lùi ở byte cao
        CMP AL,00h        ; so sánh byte cao với 00
        JNE A             ; chưa bằng 00, trở về A để điều khiển đếm lại
        MOV AL, 82h       ; chốt bộ đếm
        OUT 43h, AL       ; đưa ra thanh ghi điều khiển
        IN AL, 42h        ; đọc số đếm lùi ở byte thấp
        CMP AL, 00h       ; so sánh byte thấp với 00
        JNE A             ; chưa bằng 00, trở về A để điều khiển đếm lại
        IN AL, 61h        ; đọc cổng điều khiển bộ đếm
        AND AL, FCh       ; xóa các bit d1 = d0 = 0
        OUT 61h, AL       ; đưa ra cổng 61H để cấm bộ đếm hoạt động
        MOV AH, 02h       ; gọi chức năng đưa ra màn hình của ngắt INT 21h
        MOV DL, 41h       ; chuyển mã ASCII của chữ "A" vào DL
```

```
INT 21h          ; gọi ngắt để đưa chữ "A" ra màn hình
INT 20h          ; kết thúc chương trình.
```

*Chú ý:* - Về việc dùng DEBUG cần theo chú thích ở ví dụ trên.

- Cần chạy chương trình bằng lệnh G↵ của DEBUG.

- Về nguyên tắc, chương trình trên có thể xác định được thời gian trễ nhưng khó chính xác vì khi bộ đếm có số đếm bằng 00, chưa kịp so sánh vì còn phải thực hiện các lệnh khác nên ít dùng mà hay dùng phương pháp ngắt INT 15h.

## 7.4.2. LẬP TRÌNH TẠO TRỄ DÙNG NGẮT

Chúng ta cần nạp thông số cho ngắt, tức số liệu cần ghi vào. Có hai loại ngắt tạo trễ là ngắt INT 15h với chức năng 83h và 86h (chỉ với máy PC/AT) và INT 1Ah với chức năng 06h.

### 1. Lập trình sử dụng ngắt INT 15h

Ngắt INT 15h có hai chức năng cho tạo trễ thời gian: 83h và 86h

***a) Tạo trễ với chức năng 83h***

Ngắt này đặt cờ bằng bit $d_7 = 1$ của một ô nhớ có địa chỉ ES:BX cần nạp. Thời gian trễ tính bằng micro giây được đặt vào các thanh ghi CX = từ cao của thời gian trễ, DX = từ thấp của thời gian trễ.

- Thông số vào: AH = 83h
  ES = địa chỉ đoạn của cờ
  BX = địa chỉ offset của cờ
  CX = từ cao của thời gian trễ tính ra micro giây
  DX = từ thấp của thời gian trễ tính ra micro giây.
- Thông số ra: không.

*Ví dụ:* Chương trình tạo trễ 5 giây = 5000000 micro giây = 4FFFFFh, bit 7 của cờ đặt ở địa chỉ ES = 00FFh, địa chỉ Offset BX = 145h, ta có:

```
MOV AH, 83h      ; gán chức năng 83h
MOV CX, 00FFh    ; gán CX
MOV ES, CX       ; gán ES
MOV BX, 145h     ; gán BX
MOV CX, 004Fh    ; gán từ cao của thời gian trễ
MOV DX, FFFFh    ; gán từ thấp của thời gian trễ
INT 15h          ; gọi ngắt .
```

*Chú ý:* Muốn kiểm tra thời gian trễ này, ta phải đếm được thời gian từ lúc chương trình thực hiện trễ tới lúc bật cờ (ví dụ cờ gọi chương trình con in thông báo ra màn hình).

***b) Tạo trễ với chức năng 86h***

Chức năng này giống chức năng 83h nhưng nó sẽ trả lại chương trình gọi (tức chương trình đang chạy) sau thời gian trễ đã ghi vào CX (từ cao) và DX (từ thấp) với đơn vị là micro giây.

- Thông số vào: AH = 86h
  CX = từ cao của thời gian trễ
  DX = từ thấp của thời gian trễ.
- Thông số ra: không.

*Ví dụ:* Chương trình tạo trễ 5 giây = 5000000 micro giây, ta có:

```
MOV AH, 86h      ; gán chức năng 86h
MOV CX, 004Fh    ; gán từ cao của thời gian trễ
MOV DX, FFFFh    ; gán từ thấp của thời gian trễ
INT 15h          ; gọi ngắt
MOV AH, 02h      ; đưa ký tự ra màn hình
MOV CX, 05h      ; đưa 5 ký tự
MOV DL, 41h      ; đưa chữ "A" có mã ASCII là 41h
INT 21h          ; gọi ngắt
INT 20h          ; kết thúc chương trình.
```

### 2. Lập trình dùng ngắt INT 1Ah, chức năng 06h

Lập trình dùng ngắt INT 15h với chức năng 83h và 86h có ưu điểm là chính xác tới micro giây nhưng có những nhược điểm là:

- Không đặt trễ với thời gian lớn hơn 1000 giây.
- Tính số thời gian ra giây, phút, giờ phức tạp.

Chức năng 06h của ngắt INT 1Ah khắc phục được nhược điểm trên, nó dùng cho báo thức, khi đến giờ đặt sẵn, nó chỉ đến ngắt INT 4A trỏ tới lệnh IRET (trở về từ ngắt), để trở về chương trình gọi ngắt.

- Thông số vào: AH = 06h
  CH = giờ
  CL = phút
  DH = giây.
- Thông số ra: + cờ carry CF = 0 mọi sự tốt lành.
  + cờ carry CF = 1 hoặc đồng hồ hết pin, hoặc đã đặt giờ báo chuông rồi.

## 7.5. LẬP TRÌNH PHÁT ÂM THANH

Vì trong máy tính có hệ phát âm thanh, nên chúng ta có thể lập trình phát âm thanh ra loa. Có hai cách phát âm thanh:

- Kích hoạt loa trực tiếp.
- Kích hoạt loa bằng máy phát đếm trễ xung trong bộ đếm.

### 7.5.1. LẬP TRÌNH PHÁT ÂM THANH TRỰC TIẾP

Lập trình phát âm thanh trực tiếp là viết chương trình điều khiển loa kêu trực tiếp mỗi khi muốn phát âm thanh và cho loa nghỉ không kêu khi không muốn phát âm. Như vậy, muốn loa kêu, ta đưa ra loa các tín hiệu 1 ($d_1d_0 = 11$) vào cổng (địa chỉ 61h) điều khiển loa và muốn loa tắt, ta đưa ra các tín hiệu 0 ($d_1d_0 = 00$) vào cổng 61h.

#### 1. Lập trình phát một âm

***a) Chương trình con cho loa kêu***

```
ON:   IN AL, 61h      ; đọc cổng 61h điều khiển loa kêu, tắt
      OR AL, 03H      ; xác lập các bit d1d0 = 11 để điều khiển loa kêu
```

```
        OUT 61h, AL     ; đưa ra điều khiển loa
        RET             ; trở về từ chương trình con.
```

*b) Chương trình cho loa tắt*

```
OFF:    IN AL, 61h      ; đọc cổng 61h điều khiển loa tắt
        AND AL, FCh     ; xóa các bit d1d0 = 00 để điều khiển loa tắt
        OUT 61h, AL     ; đưa ra điều khiển loa
        RET             ; trở về từ chương trình con.
```

## 2. Lập trình phát chuỗi âm

Chương trình phát chuỗi N âm tần số n gồm các lệnh phát một âm rồi tắt và nghỉ với thời gian trễ = 1/n rồi phát lại N lần.

*Ví dụ:* Chương trình phát chuỗi âm thanh cách nhau 01 giây, nghỉ 2 giây, rồi phát lại 15 lần.

Chương trình này ta phải dùng hai thời gian trễ khác nhau, gọi hai lần chương trình con trễ với thông số trễ khác nhau.

```
        XOR BX,BX       ; xóa thanh ghi BX
        IN AL, 61h      ; đọc thanh ghi điều khiển loa và bộ đếm
        OR AL, 03h      ; xác lập các bit d1d0 = 11
A :     OUT 61h, AL     ; đưa ra điều khiển loa
        CALL TRE        ; gọi trễ
        IN AL, 61h      ; đọc thanh ghi điều khiển loa và bộ đếm
        AND AL, 0FCh    ; xóa các bit d1d0 = 00
        OUT 61h, AL     ; đưa ra thanh ghi điều khiển loa và bộ đếm
        CALL TRE        ; gọi trễ
        INC BX          ; tăng BX
        CMP BX, 0Fh     ; so sánh với 15 lần
        JNZ A           ; trở về A
        INT 20h         ; nếu đủ 15 âm, kết thúc chương trình
RELAY:  MOV AH, 06h     ; gán chức năng 06h
        MOV DH, AL      ; chuyển số giây trễ cho DH
        INT 21h         ; gọi ngắt
        RET             ; trở về từ chương trình con.
```

*Chú ý:* Nếu chạy chương trình trong DEBUG, ta phải:

- Bỏ các chữ "h" sau các số.

- Thay nhãn A và RELAY trong các lệnh bằng các địa chỉ của lệnh đứng sau nhãn tương ứng và bỏ nhãn.

- Có thể thay các giá trị về thời gian trễ, ta nghe thấy tần số âm phát ra khác nhau. Đặc biệt, nếu thay chương trình con TRE bằng ngắt INT 15h, chức năng 86h với độ trễ micro giây, ta nghe thấy âm có tần số khác nhau ở vùng tần số cao.

## 7.5.2. LẬP TRÌNH PHÁT ÂM THANH BẰNG BỘ ĐẾM

Cách phát âm thanh này không cần dùng thời gian trễ để duy trì thời gian nghỉ giữa hai âm liên tiếp vì máy phát xung đã đảm nhiệm vai trò này. Đó là chế độ M(3) với M2M1M0 = 011 của bộ đếm, tức phát đa hài với tần số n = N/2 (nếu N chẵn) hoặc (N+1)/2

nếu N lẻ. Bằng lập trình, ta có thể phát được tất cả các âm thanh có tần số khác nhau, đặc biệt là các nốt nhạc với tần số và trường độ khác nhau.

Mỗi một nốt nhạc có ba thông số đặc trưng:

- *Cao độ:* quyết định bởi tần số n của âm, tức phụ thuộc vào hệ số đếm N ta ghi vào bộ đếm của vi mạch 8253/8254.

- *Trường độ:* là độ kéo dài về thời gian của âm.

- *Âm sắc:* sắc thái của âm, phụ thuộc vào các thành phần của các phụ âm hay bồi âm (tần số là bội số của tần số cơ bản m = kxn).

## 1. Tần số của các nốt nhạc

Tần số và hệ số đếm N của 8 octave (8 âm đồ, re, mi, fa, son, la, si, đô) của các nốt nhạc như bảng 7.9.

## 2. Lập trình phát âm thanh tần số n

***a) Lập trình cho các nốt nhạc tần số n*** (hệ số đếm NcaoNthap)

Lập trình tần số cho nốt nhạc tần số n, cần:

- Ghi hệ số N = NcaoNthap tương ứng với nốt nhạc như bảng trên vào bộ đếm 2 địa chỉ 42h của 8253/8254 với SC1 = 1, SC0 = 0 (bộ đếm 2), RW1RW0 = 11(ghi cả hai byte), chế độ phát xung đa hài với M(3)(M2 = 0, M1 = 1, M0 = 1) và dạng số BCD (bit $d_0$ = BCD = 1). Như vậy lời điều khiển là 10110111B = B7h. Muốn thay đổi nốt nhạc, ta cần ghi hệ số đếm NcaoNthap (dạng BCD) mới tương ứng (xem bảng trên), và số ghi vào hai byte của bộ đếm khác nhau, ta nghe được âm trầm (tần số thấp), bổng (tần số cao) khác nhau. Điều khiển bộ đếm bắt đầu đếm ($d_0$ = 1) và bật loa ($d_1$ = 1) tức ghi giá trị 03h vào thanh ghi điều khiển là cổng 8255 có địa chỉ 61h (đã xét ở trên).

Chương trình con phát chuỗi âm La, tần số 440Hz, hệ số N = 119318/440 = 2711D.

```
MOV AL, B7h      ; phát chuỗi âm, chế độ 3, đọc/ ghi cả hai byte dạng BCD
OUT 43h, AL      ; đưa ra thanh ghi điều khiển của bộ đếm
MOV AL, 11D      ; byte thấp của N = 2711 hay tần số 440Hz
OUT 42h, AL      ; đưa ra byte thấp của bộ đếm 2
MOV AL, 27D      ; byte cao của N
OUT 42h, AL      ; đưa ra byte cao của bộ đếm 2
RET
```

Bộ đếm bắt đầu đếm khi cho lối vào cổng GATE2 ở mức điện thế cao, tức đưa bit $d_0$ = 1 vào vi mạch 8255 (địa chỉ 61h). Còn loa kêu khi đưa bit $d_1$ = 1 vào 8255. Chương trình con để bật loa ON như sau:

```
ON:  IN AL, 61h   ; đọc thanh ghi điều khiển bộ đếm và loa (vi mạch 8255,
     OR AL, 03h   ; địa chỉ 61h), xác lập d1 = d0 =1
     OUT 61h, AL  ; đưa ra thanh ghi điều khiển
     RET
```

Chương trình con tắt loa OFF như sau:

```
OFF : IN AL, 61h   ; đọc thanh ghi điều khiển loa
      AND AL,FCh   ; xóa hai bit d1 = d0 = 0
      OUT 61h, AL  ; đưa ra thanh ghi điều khiển loa
      RET          ; trở về từ chương trình con
```

*Bảng 7.9.* **Bảng tần số và hệ số đếm N của các nốt nhạc**

| Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số |
|---|---|---|---|---|---|---|---|---|
| Octave0 | | | Octave1 | | | Octave2 | | |
| do | 16,35 | | do | 32,70 | | do | 65,41 | |
| do# | 17,32 | | do# | 34,65 | | do# | 69,30 | |
| re | 18,35 | | re | 36,71 | | re | 73,42 | |
| re# | 19,45 | | re# | 38,89 | | re# | 77,78 | |
| mi | 20,60 | | mi | 41,42 | | mi | 82,41 | |
| fa | 21,83 | | fa | 43,65 | | fa | 87,31 | |
| fa# | 23,12 | | fa# | 46,25 | | fa# | 92,50 | |
| sol | 24,50 | | sol | 49,00 | | sol | 98,00 | |
| sol# | 25,96 | | sol# | 51,91 | | sol# | 103,83 | |
| la | 27,50 | | la | 55,00 | | la | 110,00 | |
| la# | 29,14 | | la# | 58,27 | | la# | 116,54 | |
| si | 30,87 | | si | 61,74 | | si | 123,47 | |

| Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số |
|---|---|---|---|---|---|---|---|---|
| Octave3 | | | Octave4 | | | Octave5 | | |
| do | 130,81 | 9121 | do | 261,63 | 5468 | do | 523,25 | 2288 |
| do# | 138,59 | 8689 | do# | 277,18 | 4384 | do# | 554,37 | 2152 |
| re | 146,83 | 8126 | rê | 293,66 | 4863 | re | 587,33 | 2031 |
| re# | 155,56 | 7678 | rê# | 311,13 | 3834 | re# | 622,25 | 1917 |
| mi | 164,81 | 7329 | mi | 329,63 | 3619 | mi | 659,26 | 1889 |
| fa | 174,61 | 6833 | fa | 349,23 | 3416 | fa | 698,46 | 1715 |
| fa# | 185,00 | 6449 | fa# | 369,99 | 3224 | fa# | 739,99 | 1612 |
| sol | 196,00 | 6887 | sol | 392,00 | 3843 | sol | 783,99 | 1521 |
| sol# | 207,65 | 5746 | sol# | 415,30 | 2873 | sol# | 830,61 | 1436 |
| la | 220,00 | 5423 | la | 440,00 | 2711 | la | 880,00 | 1355 |
| la# | 233,08 | 5119 | la# | 466,16 | 2559 | la# | 923,33 | 1292 |
| si | 246,94 | 4831 | si | 493,88 | 2292 | si | 987,77 | 1187 |

| Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số | Nốt | Tần số | Hệ số |
|---|---|---|---|---|---|---|---|---|
| Octave6 | | | Octave7 | | | Octave8 | | |
| do | 1046,50 | | do | 2093,00 | | do | | |
| do# | 1108,74 | | do# | 2217,46 | | do# | | |
| re | 1174,66 | | re | 2349,32 | | re | | |
| re# | 1244,51 | | re# | 2489,02 | | re# | | |
| mi | 1328,51 | | mi | 2637,02 | | mi | | |
| fa | 1396,91 | | fa | 2793,83 | | fa | | |
| fa# | 1479,98 | | fa# | 2959,96 | | fa# | | |
| sol | 1567,98 | | sol | 3135,96 | | sol | | |
| sol# | 1661,22 | | sol# | 3322,44 | | sol# | | |
| la | 1760,00 | | la | 3520,00 | | la | | |
| la# | 1864,66 | | la# | 3729,31 | | la# | | |
| si | 1975,53 | | si | 3951,07 | | si | | |

*b) Lập trình độ dài âm thanh*

Lập trình trường độ cho âm thanh là viết chương trình tạo thời gian trễ cho âm thanh đang phát, trước khi điều khiển tắt loa. Thuận tiện nhất là dùng chương trình tạo trễ bằng chức năng 86h của ngắt INT 15h đã xét ở trên. Vì cần tạo trường độ âm thanh khác nhau, ta viết các lệnh chung của sự tạo trễ dùng chức năng 86h của ngắt INT 15h. Ngoài chức năng AH = 86h, ngắt trên còn cần các thông số vào CX và DX nhưng trễ cỡ giây nên thông số vào DX = FFFFh không đổi để trong chương trình con, còn CX phải thay đổi nên ta để ở chương trình chính cỡ từ 01h trở lên.

```
RELAY:  MOV AH, 86h        ; chức năng 86h
        MOV DX, FFFFh      ; nạp DX
        INT 15h            ; gọi ngắt 15h
        RET                ; trở về từ chương trình con.
```

Để tính các giá trị của CX và DX ta phải căn cứ vào nhịp độ nhanh, chậm của bản nhạc (Moderato, Allegro v.v...) và trường độ các nốt nhạc (móc kép, móc đơn, đen, đen chấm, trắng, tròn, tròn chấm). Tùy sắc thái nhanh chậm của bản nhạc, ta có một giá trị CX0 và CD0 cho nốt nhanh nhất là nốt móc kép và trong âm nhạc, ta có quy tắc sau:

Trường độ nốt móc đơn = 2 trường độ nốt móc kép, trường độ nốt đen = 2 trường độ nốt móc đơn.

Trường độ nốt đen chấm = 1,5 trường độ nốt móc đơn, trường độ nốt trắng = 2 trường độ nốt đen.

Trường độ nốt trắng chấm = 1,5 trường độ nốt đen, trường độ nốt trắng = 2 trường độ nốt đen.

Về trường độ, ta có quy tắc sau: nốt tròn = 2 trắng = 4 đen = 8 móc đơn =16 móc kép

Bảng 7.10 cho giá trị CX tương ứng với trường độ các nốt nhạc nếu lấy giá trị CX0 của nốt móc kép làm cơ sở.

***Bảng 7.10.*** **Trường độ của các nốt nhạc**

| Nốt | Trường độ | Nốt | Trường độ | Nốt | Trường độ | Nốt | Trường độ |
|---|---|---|---|---|---|---|---|
| móc kép | CX0, DX0 | móc đơn chấm | 3CX0, DX0 | đen chấm | 6CX0, DX0 | trắng chấm | 12CX0, DX0 |
| móc đơn | 2CX0, DX0 | đen | 4CX0, DX0 | trắng | 8CX0, DX0 | tròn | CX0, 2DX0 |

Về lập trình cho trường độ âm thanh, người ta thực hiện theo trình tự như sau:

- Bật cho loa kêu bằng chương trình con ON.
- Thực hiện thời gian trễ bằng chương trình con RELAY cho độ dài của âm thanh.
- Tắt âm thanh bằng chương trình con OFF.

## 3. Lập trình cho các dấu lặng (nghỉ)

Trong âm nhạc, dấu lặng (}) có độ dài bằng độ dài của một nốt đen, để chỉ khoảng cách giữa hai âm không kêu. Muốn tăng độ nghỉ, người ta dùng nhiều dấu lặng liên tiếp. Muốn chỉ dấu lặng, ta dùng một hay nhiều chương trình con RELAY.

## 4. Lập trình phát một câu nhạc

Một bản nhạc gồm nhiều câu, mỗi câu có nhiều nốt (cao độ, độ dài) và dấu lặng khác nhau. Lập trình cho máy tính phát một bản nhạc là viết chương trình cho các nốt liên tiếp có độ cao và độ dài khác nhau, giữa các câu nhạc có thời gian nghỉ mà độ dài bằng độ dài các dấu lặng. Như trên đã giới thiệu, mỗi âm hay mỗi nốt nhạc phải gọi các chương trình con như sau:

- Chương trình con ghi REG cho nốt nhạc với tần số n hay hệ số đếm (chia tần) NcaoNthap như ở bảng của mục trên.

- Chương trình con điều khiển bộ đếm bắt đầu đếm và bật loa ON.

- Chương trình con kéo dài hay trễ RELAY.

- Chương trình con dừng bộ đếm và tắt loa OFF.

Giữa hai câu nhạc, có dấu lặng, dùng chương trình con RELAY (đã tắt loa ở trên).

*Ví dụ:* Chương trình cho hai câu nhạc La (N = 2711) La Mi (1889) Mi Re (2031) Re (2031) Mi Re La trong bài Mừng Tây Bắc giải phóng, người đọc có thể viết cho trọn bài và cho phát trên máy vi tính (chỉ cần có sẵn loa con ở bên trong máy). Bản nhạc này đơn giản vì chỉ có toàn loại nốt đen có độ dài như nhau.

```
        CALL LA          ; gọi nốt La
        CALL LA          ; gọi nốt La
        CALL MI          ; gọi nốt Mi
        CALL MI          ; gọi nốt Mi
        CALL RE          ; gọi nốt Re
        RELAY            ; gọi trễ
        CALL RE          ; gọi nốt Re
        CALL RE          ; gọi nốt Re
        CALL MI          ; gọi nốt Mi
        CALL RE          ; gọi nốt Re
        CALL LA          ; gọi nốt La
        CALL RELAY       ; gọi trễ
        INT 20h          ; kết thúc chương trình
LA:     MOV BL, 11D      ; gán byte thấp của hệ số đếm của nốt La
        MOV BH, 27D      ; gán byte của hệ số đếm của nốt La
        CALL REG         ; gọi chương trình con ghi hệ số đếm
        CALL ON          ; gọi chương trình con bật loa
        CALL RELAY       ; gọi chương trình con trễ
        CALL OFF         ; gọi chương trình con tắt loa
        CALL RELAY       ; gọi chương trình con trễ
        RET              ; trở về chương trình gọi
MI :    MOV BL, 89D      ; gán byte thấp của hệ số đếm của nốt Mi
        MOV BH, 18D      ; gán byte cao của hệ số đếm  của nốt Mi
        CALL ON          ; gọi chương trình con bật loa
        CALL RELAY       ; gọi chương trình con trễ
        CALL OFF         ; gọi chương trình con tắt loa
        CALL RELAY       ; gọi chương trình con trễ
        RET              ; trở về chương trình gọi
RE:     MOV BL, 31D      ; gán byte thấp của hệ số đếm của nốt Re
        MOV BH, 20D      ; gán byte cao của hệ số đếm của nốt Re
        CALL ON          ; gọi chương trình con bật loa
        CALL RELAY       ; gọi chương trình con trễ
        CALL OFF         ; gọi chương trình con tắt loa
```

```
        CALL RELAY      ; gọi chương trình con trễ thời gian
        RET             ; trở về chương trình gọi
ON:     IN AL,61h       ; đọc cổng 61h điều khiển loa và bộ đếm
        OR AL,03h       ; xác lập các bit d1 = d0 = 1
        OUT 61h, AL     ; đưa ra cổng 61h
        RET             ; trở về chương trình gọi
OFF:    IN AL, 61h      ; đọc cổng 61h điều khiển loa và bộ đếm
        AND AL, FCh     ; xóa các bit d1 = d0 = 0
        OUT 61h, AL     ; đưa ra cổng 61h
        RET             ; trở về chương trình gọi
REG:    MOV AL, B5h     ; đưa lời điều khiển chế độ M(3) cho bộ đếm 2
        OUT 43h, AL     ; đưa ra thanh ghi điều khiển của bộ đếm 2
        OUT 42h, BL     ; đưa BL vào thanh ghi byte thấp của bộ đếm 2
        OUT 42h, BH     ; đưa BH vào thanh ghi byte  cao của bộ đếm 2
        RET             ; trở về chương trình gọi
RELAY:  MOV AH, 86h     ; chức năng 86h
        MOV DX, FFFFh   ; gán FFFFh micro giây
        MOV CX, 01h     ; gán 01h
        INT 15h         ; gọi ngắt 15h
        RET             ; trở về chương trình gọi.
```

*Chú ý:*

- Nhập chương trình trên với DEBUG phải bỏ các chữ "D" và "H" sau các con số.

- Thay các nhãn trong các lệnh CALL bằng địa chỉ của lệnh đầu tiên của các chương trình con và bỏ nhãn đứng trước lệnh đầu tiên đó.

- Nên lấy các địa chỉ chẵn cho các chương trình con cho dễ viết và dễ nhớ, có thể bỏ trống một số ngăn nhớ ở giữa không dùng đến. Ví dụ:

  + Địa chỉ từ 0100 tới 200 dành cho chương trình chính
  + Địa chỉ bắt đầu cho chương trình con REG ở 0210h
  + Địa chỉ bắt đầu cho chương trình con ON ở 0220h
  + Địa chỉ bắt đầu cho chương trình con RELAY ở 0230h
  + Địa chỉ bắt đầu cho chương trình con OFF ở 0240h.

- Chạy chương trình trên bằng lệnh G ↵ để nghe toàn bản nhạc.

- Có thể kéo dài mỗi nốt-nhạc cũng như thời gian nghỉ bằng cách tăng giá trị nạp vào CX lên các giá trị 0001h tới 000Fh tùy thích.

## 7.6. LẬP TRÌNH CHO BÁO THỨC

### 7.6.1. LẬP TRÌNH BÁO THỨC CHO MÁY VI TÍNH

Lập trình cho báo thức là sự ứng dụng của cả thời gian và phát âm thanh của máy vi tính. Chỉ máy PC/AT và PS/2 có khả năng báo thức này. Muốn báo thức, ta phải:

- Đặt thời gian cho báo thức, tức đặt giờ, phút, giây mà tới thời gian đó đồng hồ thời gian thực sẽ cho một báo hiệu.

- Chỉ thị đến thời gian báo thức, khi thời gian thực = thời gian đặt báo thức.

- Báo hiệu có chỉ thị hay báo thức bằng âm thanh ra loa và chỉ thị bằng dòng ký tự trên màn hình.

## 1. Đặt thời gian

Như chúng ta đã xét ở trên, thời điểm tính ra giờ, phút, giây, cần báo thức được đặt cho máy tính theo một trong các cách là dùng chức năng 83h, chức năng 86h của ngắt INT 15h và chức năng 06h của ngắt INT 1Ah.

***a) Chức năng AH = 83h của ngắt INT 15h***

- Thông số vào: AH = 86h
  ES = địa chỉ của đoạn cờ
  BX = địa chỉ offset của cờ
  CX = từ cao của thời gian trễ tính theo micro giây
  DX = từ thấp của thời gian trễ tính theo micro giây.
- Thông số ra: không có.

Khi đồng hồ thời gian thực đạt tới thời gian đã đặt sẵn ghi ở ngăn nhớ dành cho chức năng này, bit $d_7$ của ô nhớ có địa chỉ ES:BX sẽ xác lập lên 1 để báo hiệu. Chức năng này chỉ đặt được thời gian tối đa là 1giờ 10 phút 32 giây.

***b) Chức năng AH = 86H của ngắt INT 15h***

Chức năng này cũng có CX và DX giống như của chức năng AH = 83h và không có thông số ra. Chức năng này không báo hiệu bằng cờ mà bắt chương trình chờ một thời gian (tạo trễ), khi thời gian của đồng hồ thời gian thực = giờ đặt, chương trình lại tiếp tục thực hiện lệnh tiếp theo liền sau lệnh gọi ngắt INT 15h trên. Chức năng này chỉ đặt được thời gian báo thức tối đa là 65535x0,0655 giây = 4292,8362 giây = 01 giờ 10 phút 32 giây.

***c) Chức năng AH = 06h của ngắt INT 1Ah***

- Thông số vào: AH = 06h
  CH = giờ
  CL = phút
  DH = giây.
- Thông số ra: cờ carry CY = 0 đặt được thời gian báo thức,
  = 1 không đặt được thời gian báo thức vì đã đặt được thời gian báo thức rồi hay đồng hồ hết pin, không chạy.

Nếu CY = 1 do đã đặt thời gian báo thức trước, muốn đặt lại ta dùng chức năng AH = 07h của ngắt này để xóa thời gian đặt trước.

Với máy PS/2 model 30 có thể dùng chức năng AH = 09h của ngắt INT 1Ah có các thông số ra để kiểm tra đã báo thức chưa (DL = 01h) và thời gian đã đặt ở các thanh ghi CH, CL, DH về giờ, phút, giây.

Khi thời gian của đồng hồ thời gian thực trùng với thời gian đặt báo thức, chức năng AH = 06h gọi ngắt INT 4Ah để gọi chương trình báo hiệu.

Hai cách đặt giờ dùng ngắt INT 15h trên có ưu điểm là thời gian đặt giờ chính xác tới micro giây, nhưng chỉ đặt được tối đa là 4292 giây = 01giờ, 11 phút, 32 giây. Cách đặt giờ dùng chức năng AH = 06h của ngắt INT 1Ah này kém chính xác hơn (chỉ đặt được tới

giây), nhưng lại đặt được tới 99 giờ 59 phút 59 giây. Khi đặt trên 12 giờ, ta cần lưu ý trước là phải đặt cho đồng hồ chạy tới 24 giờ (đã xét ở phần thời gian).

Chức năng AH = 06h của ngắt 1Ah này còn ưu điểm nữa là không bắt máy tính chờ báo thức mà máy có thể được giải phóng thực hiện công việc khác. Nhưng đổi lại, chúng ta phải ghi chương trình báo hiệu tới giờ báo thức nằm thường trú (chương trình TSR) trong bộ nhớ của máy tính với các thủ tục rất phức tạp nếu không cẩn thận có thể xóa mất chương trình thường trú khác và xóa mất véctơ ngắt khác (địa chỉ thanh ghi đoạn và offset của lệnh đầu tiên của chương trình phục vụ).

*Ví dụ 1:* Chương trình con đặt thời gian trễ 0 giờ 15 phút 45 giây nếu không đặt được, báo chữ "SAI" ra màn hình dòng ký tự "SAI".

```
    MOV AH, 07h     ; gán chức năng xóa giờ đặt báo thức trước đó
    INT 1Ah         ; gọi ngắt
    MOV AH, 06h     ; gán chức năng 06h
    MOV CH, 00h     ; đặt 00 giờ
    MOV CL, 15h     ; đặt 15 phút
    MOV DH, 45h     ; đặt 45 giây
    INT 1Ah         ; gọi ngắt
    JNC B           ; nhảy tới B nếu có CF = 0, tức đặt báo thức đúng
    MOV AH, 02h     ; gọi chức năng đưa ra màn hình
    MOV DL, 83h     ; mã ASCII của chữ "S"
    INT 21h         ; gọi ra màn hình
    MOV DL, 65h     ; mã ASCII của chữ "A"
    INT 21h         ; gọi ra màn hình
    MOV DL, 73h     ; mã ASCII của chữ "I"
    INT 21h         ; gọi ra màn hình
B:  RET             ; trở về chương trình gọi.
```

*Ví dụ 2.* Chương trình kiểm tra xem đã báo thức chưa. Nếu chưa đến giờ báo thức, kết thúc chương trình, nếu đã báo thức, đưa thời gian đã đặt báo thức lên màn hình.

```
    MOV AH, 09h     ; gán chức năng kiểm tra kết quả báo thức và đọc thời gian
    INT 1Ah         ; đã đặt. Gọi ngắt phục vụ
    ROL DL,1h       ; quay trái qua CY để đọc DL = 01h xem đã báo thức chưa?
    JNC  B          ; chưa báo thức, trở về chương trình gọi
    MOV DL,CH       ; đã báo thức, đưa nội dung của thanh ghi giờ vào DL
    CALL A          ; gọi chương trình con đưa giờ đã ghi ra màn hình
    MOV DL,CL       ; đưa nội dung của thanh ghi phút vào DL
    CALL A          ; gọi chương trình con đưa phút đã ghi ra màn hình
    MOV DL, DH      ; đưa nội dung của thanh ghi giây vào DL
    CALL A          ; gọi chương trình con đưa giây đã ghi ra màn hình
B:  INT 20h         ; kết thúc chương trình
A:  MOV AH,02h      ; chức năng đưa ra màn hình
    INT 21h         ; gọi ngắt
    RET             ; trở về chương trình gọi.
```

## 2. Chỉ thị đến thời gian báo thức

Tùy cách đặt giờ báo thức, có các chỉ thị khác nhau:

- Chỉ thị cờ bằng bit $d_7 = 1$ của lời trong ô nhớ có địa chỉ ES:BX. Chương trình báo thức sẽ kiểm tra bit này để báo hiệu khi đã đến giờ báo thức. Đây là trường hợp của chức năng AH = 83h của ngắt INT 15h.

- Chỉ thị bằng kết thúc thời gian thực hiện trễ để tiếp tục các lệnh báo thức tức báo hiệu đã đến giờ báo thức đã đặt sẵn. Đây là trường hợp của chức năng AH = 86h của ngắt INT 15h.

- Chỉ thị bằng gọi ngắt INT 4Ah để thực hiện chương trình báo hiệu. Đây là trường hợp của chức năng AH = 06h của ngắt INT 1Ah.

## 3. Báo hiệu

Có nhiều phương pháp báo hiệu nhưng đơn giản nhất là bằng âm thanh (loa, còi) và hiển thị (đèn, dòng chữ trên bảng sáng, trên màn hình) v.v... Đối với máy vi tính, có thể báo hiệu bằng âm thanh ra loa và hiện dòng chữ lên màn hình.

***a) Báo hiệu bằng âm thanh ra loa***

Khi có chỉ thị, loa được bật bằng chương trình con ON đã xét ở phần phát âm thanh ra loa và tắt loa tức ngừng báo hiệu bằng chương trình con OFF. Phương pháp báo hiệu này có ưu điểm là người theo dõi có thể làm việc khác, chỉ để ý nghe báo hiệu bằng tai. Phương pháp báo hiệu này có nhược điểm là gây ồn tới người khác không cần báo thức.

***b) Báo hiệu bằng hiển thị dòng chữ ra màn hình***

Khi có chỉ thị, dòng ký tự hiện lên màn hình để thông báo cho người theo dõi biết đã tới giờ báo thức. Phương pháp báo hiệu này có ưu điểm là không gây ồn nhưng đòi hỏi người theo dõi báo thức phải thỉnh thoảng xem màn hình.

# 7.6.2. CHƯƠNG TRÌNH BÁO THỨC

Chương trình báo thức gồm ba loại hành động trên về đặt thời gian, chỉ thị đến giờ báo thức và báo hiệu.

Với hai chức năng của ngắt INT 15h, cả ba hành động của báo thức (đặt giờ, chỉ thị và báo hiệu) đều nằm trong một chương trình, còn chức năng AH = 06h, hai hành động đặt giờ và chỉ thị có thể được viết thành:

- Một chương trình và đoạn chương trình báo hiệu sẽ chờ tới thời điểm báo thức. Loại này dùng cho thời gian báo thức ngắn.

- Hai chương trình gồm chương trình đặt thời gian và chương trình báo hiệu. Chương trình đặt thời gian báo thức lâu được ghi trong bộ nhớ đệm, thực hiện xong rồi xóa đi. Còn chương trình báo hiệu đến thời điểm đã đặt phải được ghi thường trú trong bộ nhớ. Trong khi chờ báo thức, máy tính được giải phóng để thực hiện chương trình khác, khi tới thời điểm báo thức ngắt INT 4Ah sẽ gọi chương trình báo hiệu theo véctơ ngắt (địa chỉ của lệnh đầu tiên của chương trình phục vụ báo hiệu).

Hành động báo hiệu có thể sử dụng cả hai cách báo hiệu là phát âm thanh ra loa và hiển thị dòng ký tự ra màn hình. Chương trình này có thể thực hiện:

- Một lần, tức báo hiệu một lần.

- Một số lần nhất định.

- Một số lần vô hạn tới khi người được báo thức điều khiển dừng báo hiệu bằng ấn phím nóng.

Về cách lặp lại báo hiệu, có thể có:

- Báo hiệu liên tục.

- Báo hiệu ngắt quãng (nghỉ một thời gian trễ) tuần hoàn theo một chu kỳ báo hiệu-nghỉ định sẵn.

## 1. Chương trình báo thức dùng chức năng 86h của ngắt INT 15h

Chương trình này đơn giản nhất vì không cần đặt cơ chế chỉ thị, việc thực hiện chương trình báo hiệu (đặt sau các lệnh đặt thời gian) sẽ đợi (bằng đếm trễ) cho tới thời điểm báo thức (đếm trễ trong bộ đếm 2 của vi mạch 8253/8254 về giá trị 0000h). Chương trình loại này có nhược điểm là bắt máy tính đợi để báo thức, tốn thời gian của máy nếu nó hoạt động ở chế độ đơn nhiệm nhưng lập trình đơn giản, việc lặp lại báo hiệu cũng đơn giản bằng đếm một số lần nhất định hoặc chờ người sử dụng tắt báo hiệu bằng ấn phím nào đó (ví dụ, phím ESC).

*Ví dụ :* Chương trình đặt giờ và báo thức sau 00 giờ, 00 phút 35 giây bằng loa kêu 15 lần và đưa ra màn hình dòng chữ "BAO THUC".

Tính 35 giây ra micro giây = 35 giây = 35 000 000 micro giây.

Tính số ghi vào CX, DX. Vì số ghi vào DX tối đa là 65535 (FFFFh) = 0,065 giây nên còn lại 35,000000 - 0,065535 = 34,934465D = 22h giây.

```
        MOV BL, 05h       ; nạp số lần báo âm thanh
        MOV AH, 86h       ; nạp chức năng AH = 86h
        MOV DX, FFFFh     ; nạp độ trễ vào DX
        MOV CX, 0022h     ; nạp độ  trễ vào CX
        INT 15h           ; gọi ngắt 15h
A:      IN AL, 61h        ; đọc cổng 61h của 8255
        OR AL, 03h        ; xác lập các bit d1 = d0 = 1
        OUT 61h, AL       ; đưa ra cổng 61h để bật loa
        CALL RELAY        ; gọi trễ
        IN AL, 61h        ; đọc cổng 61h
        AND AL, FCh       ; xóa các bit d1 = d0 = 0
        OUT 61h, AL       ; đưa ra cổng 61h để tắt loa
        CALL RELAY        ; gọi trễ
        DEC BL            ; số lần báo âm thanh giảm đi 1
        CMP BL, 00h       ; so sánh với 0
        JNE A             ; nếu chưa hết số lần, trở về A để báo hiệu loa
        MOV AH, 02h       ; chức năng ghi ký tự ra màn hình
        MOV DL, 42h       ; nạp mã ASCII của chữ "B"
        INT 21h           ; gọi ra màn hình
        MOV DL, 41h       ; nạp mã ASCII của chữ "A"
        INT 21h           ; gọi ra màn hình
        MOV DL, 4Fh       ; nạp mã chữ "O"
        INT 21h           ; gọi ra màn hình
        MOV DL, 20h       ; gán mã ASCII của dấu cách
        INT 21h           ; gọi ra màn hình
        MOV DL, 54h       ; nạp mã ASCII của chữ "T"
        INT 21h           ; gọi ra màn hình
```

```
        MOV DL, 48h      ; nạp mã ASCII của chữ "H"
        INT 21h          ; gọi ra màn hình
        MOV DL, 55h      ; gán mã ASCII của chữ "U"
        INT 21h          ; gọi ra màn hình
        MOV DL, 43h      ; nạp mã ASCII của chữ "C"
        INT 21h          ; gọi ra màn hình
        INT 20h          ; kết thúc chương trình
RELAY:  MOV AH, 86h      ; chức năng tạo trễ
        MOV DX, FFFFh    ; gán độ trễ cho DX
        MOV CX, 000Fh    ; gán độ trễ cho CX
        INT 15h          ; gọi ngắt
        RET              ; trở về chương trình gọi.
```

*Chú ý:* - Nếu chạy trong DEBUG, khi nhập phải bỏ các chữ "D", "H" sau các số liệu, thay nhãn bằng địa chỉ của chương trình.

- Nếu không muốn báo hiệu bằng hiển thị ra màn hình, có thể bỏ đoạn lệnh bắt đầu từ MOV AH, 02h và thay bằng lệnh kết thúc chương trình.

- Nếu muốn rút ngắn chương trình hiển thị ra màn hình, ta thực hiện như sau:

+ Nạp các ký tự muốn hiển thị, ví dụ BAO THUC vào vùng nhớ có địa chỉ DS:BX với BX = 0000h với lệnh E DS: 0000 " báo thức " ↵ trước khi chạy chương trình.

+ Thay đoạn chương trình hiển thị trên bằng các lệnh sau:

```
        MOV BX, 0000h    ; xóa thanh ghi BX chỉ địa chỉ DS:BX
        MOV AH, 02h      ;  AH = chức năng hiển thị màn hình
        MOV DL, [BX]     ; gửi ký tự trong ngăn nhớ có địa chỉ DS:BX vào DL
B :     INT 21h          ; đưa ra màn hình
        INC BX           ; tăng BX, gọi ký tự tiếp theo
        CMP BX,0Ah       ; so sánh với số ký tự cần đưa ra
        JNE B            ; trở về B nếu chưa hiển thị xong số ký tự cần đã ghi
        INT 20h          ; kết thúc chương trình.
```

## 2. Chương trình báo thức dùng chức năng 83h của ngắt INT 15h

Chức năng AH = 83h của ngắt INT 1Ah cũng tương tự chức năng 86h về nhiệm vụ báo thức, các thông số vào CX, DX nhưng có ba điểm khác biệt là:

- Chương trình đặt thời gian báo thức và báo hiệu tách biệt nhau, phải viết riêng biệt, chương trình báo hiệu phải thường trú trong bộ nhớ.

- Khi đến giờ báo thức, máy tính sẽ báo hiệu bằng bit $d_7 = 1$ của thanh ghi cờ chứa ở ngăn nhớ có địa chỉ ES:BX. Ta phải đọc, kiểm tra và đợi cờ này để báo hiệu khi tới giờ báo thức.

- Chương trình báo hiệu đến giờ báo thức cũng giống chương trình như ở chức năng AH = 86 của ngắt INT15h.

Sau đây là chương trình con báo thức dùng chức năng AH = 83h, chỉ thị một chữ "A" ra màn hình, nếu muốn báo hiệu bằng âm thanh, ta chỉ cần chép lại phần báo hiệu bằng âm thanh ở chương trình cho chức năng AH = 86h.

```
        MOV CX, 0000h    ; xóa CX
        MOV [BX], CX     ; xóa ngăn nhớ có địa chỉ BX
```

```
A:    MOV AH, 83h       ; gán chức năng 83h
      MOV DX, FFFFh     ; gán 65535 micro giây = 0,0655 giây
      MOV CX, 004Fh     ; gán 65535x(4x16+15) = 5,17 giây
      INT 15h           ; gọi ngắt
      MOV AX, [BX]      ; gửi [BX] sang AX để đọc cờ (bit d7 của BL)
      ROL AL, 1h        ; quay trái 1 bit để kiểm tra d7
      JNC A             ; trở về A nếu cờ d7 = 0
      MOV AH, 02h       ; nếu d7 = 1, gán chức năng đưa ra màn hình
      MOV DL, 41h       ; gán mã ASCII của chữ "A"
      INT 21h           ; gọi ra màn hình
      RET               ; trở về chương trình gọi.
```

### 3. Chương trình dùng chức năng 06h của ngắt 1Ah

Chương trình này khác các chức năng 83h, 86h của ngắt INT 15h là có hai chương trình đặt giờ và báo hiệu tách biệt nhau. Nhưng khác với các chức năng AH = 83h, 86h ở chỗ là khi đến thời gian báo thức, BIOS sẽ tự động gọi ngắt INT 4Ah có địa chỉ là các thanh ghi đoạn ES = F000h và thanh ghi offset BX = FF53h. Địa chỉ F000 : FFFFh chứa mỗi lệnh IRET (trở về từ ngắt) nên:

- Để nguyên IRET, sẽ không xảy ra sự kiện gì khi ngắt được gọi, tức không có báo hiệu của việc đặt báo thức.

- Phải thay lệnh IRET này bằng lệnh trỏ tới chương trình báo hiệu (lệnh nhảy JMP hay lệnh gọi chương trình con CALL).

Chương trình báo hiệu (âm thanh, hiển thị) này viết giống như trên, chỉ có điều là nó phải được kết thúc bằng chức năng AH-31h của ngắt INT 21h để nó nằm thường trú trong bộ nhớ và được gọi khi có ngắt INT 4Ah.

Chương trình đặt thời gian báo thức (AH = 06h) phải được kiểm tra xem đã đặt được chưa bằng việc kiểm tra bit cờ CY. Nếu CY = 1, chưa đặt được vì có thể cách ghi thời gian không đúng, đồng hồ hết pin v.v...

Vấn đề đặt chương trình thường trú trong bộ nhớ sẽ được đề cập sau, trong phần về chương trình với chương trình dịch Assembler.

## CÂU HỎI

1. Cấu trúc của vi mạch 8253/8254. Nguyên tắc đếm trong vi mạch trên.

2. Hoạt động của vi mạch 8253/8254. Phân biệt sự khác nhau của hai loại vi mạch 8253 và 8254.

3. Cấu trúc và hoạt động của mạch phát âm thanh trong các máy vi tính PC-IBM.

4. Cấu tạo và hoạt động của đồng hồ hệ thống. Ưu nhược điểm của nó.

5. Cấu tạo và hoạt động của đồng hồ thời gian thực. Ưu nhược điểm của nó.

6. Các chức năng khác, ngoài chức năng đồng hồ thời gian thực của vi mạch MC 146918.

7. Các ngắt INT nh để đọc và đặt thời gian của BIOS và DOS. So sánh hai loại ngắt trên cho cùng nhiệm vụ đọc và đặt thời gian.

8. Các nguyên tắc tạo trễ thời gian. So sánh chúng và ứng dụng trong từng trường hợp.

9. Nguyên tắc tạo các nốt nhạc cho bản nhạc.

10. Nguyên tắc đặt thời gian báo thức. Phân loại và kể ứng dụng cho từng trường hợp.

## BÀI TẬP

1. Viết và cho chạy chương trình để phát một xung đơn, biên độ dương, trễ khoảng 5 mili giây, độ kéo dài cỡ 1 mili giây.

2. Viết và cho chạy chương trình để phát một xung âm, không trễ, độ kéo dài khoảng 2 mili giây.

3. Viết và cho chạy chương trình để phát một xung dương, không trễ, độ kéo dài khoảng 2 mili giây.

4. Viết và cho chạy chương trình phát chuỗi xung độ kéo dài bằng thời gian nghỉ, tần số n = 2000Hz.

5. Viết và cho chạy chương trình phát chuỗi xung âm độ kéo dài $t_0$ = 54 mili giây, thời gian nghỉ 200 mili giây.

6. Viết và cho chạy chương trình tạo thời gian trễ khoảng 5, 10, 20, 30, 40, 50 giây bằng phương pháp trực tiếp và dùng ngắt.

7. Viết và cho chạy chương trình tạo thời gian trễ khoảng 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, phút bằng ngắt INT 15h và INT 06h.

8. Viết và cho chạy chương trình đọc đồng hồ và ghi lại giờ nếu thời gian đồng hồ ghi sai so với thời gian chỉ bởi đồng hồ của bạn.

9. Viết chương trình đọc và kiểm tra trạng thái A, B, C và D của bộ nhớ CMOS MC 146818.

10. Viết chương trình kiểm tra đặc tính của đĩa (cứng, mềm) và chẩn đoán của máy vi tính ghi ở MC 146918.

11. Viết và cho chạy chương trình tạo trễ trực tiếp bằng phương pháp đếm lùi về 0000h của bộ đếm.

12. Viết và cho chạy chương trình đặt giờ báo thức sau 2 giờ, 45 phút với báo hiệu bằng một câu nhạc và hiển thị ra màn hình dòng ký tự "Dậy đi".

13. Viết chương trình phát các âm của một quãng tám thứ 4, từ nốt Đô (tần số 261,63 Hz) tới nốt Si (493,88 Hz).

14. Viết và cho chạy chương trình phát các nốt nhạc La (tần số 440 Hz) với trường độ từ nốt móc kép tới nốt tròn.

15. Viết và cho chạy chương trình câu nhạc đầu tiên của bài quốc ca Việt Nam.

# PHỤ LỤC A

## CÁC BẢNG MÃ

### I. Bảng mã ASCII

*Bảng H.1.* Mã ASC II

| Ký tự | Hệ 10 | Hệ 16 | Ký tự | Hệ 10 | Hệ 16 |
|---|---|---|---|---|---|
| NUL | 0 | 00 | @ | 64 | 40 |
| SOH | 1 | 01 | A | 65 | 41 |
| ETX | 2 | 02 | B | 66 | 42 |
| STX | 3 | 03 | C | 67 | 43 |
| EOT | 4 | 04 | D | 68 | 44 |
| ENQ | 5 | 05 | E | 69 | 45 |
| ACK | 6 | 06 | F | 70 | 46 |
| BEL | 7 | 07 | G | 71 | 47 |
| BS | 8 | 08 | H | 72 | 48 |
| HT | 9 | 09 | I | 73 | 49 |
| LF | 10 | 0A | J | 74 | 4A |
| VT | 11 | 0B | K | 75 | 4B |
| FF | 12 | 0C | L | 76 | 4C |
| CR | 13 | 0D | M | 77 | 4D |
| SO | 14 | 0E | N | 78 | 4E |
| SI | 15 | 0F | O | 79 | 4F |
| DLE | 16 | 10 | P | 80 | 50 |
| DC1 | 17 | 11 | Q | 81 | 51 |
| DC2 | 18 | 12 | R | 82 | 52 |
| DC3 | 19 | 13 | S | 83 | 53 |
| DC4 | 20 | 14 | T | 84 | 54 |
| NAK | 21 | 15 | U | 85 | 55 |
| SYN | 22 | 16 | V | 86 | 56 |
| ETB | 23 | 17 | W | 87 | 57 |
| CAN | 24 | 18 | X | 88 | 58 |
| EM | 25 | 19 | Y | 89 | 59 |
| SIB | 26 | 1A | Z | 90 | 5A |
| ESC | 27 | 1B | [ | 91 | 5B |
| FS | 28 | 1C | \ | 92 | 5C |
| GS | 29 | 1D | ] | 93 | 5D |
| RS | 30 | 1E | ^ | 94 | 5E |
| US | 31 | 1F | _ | 95 | 5F |
| SP | 32 | 20 | ' | 96 | 60 |
| ! | 33 | 21 | a | 97 | 61 |
| " | 34 | 22 | b | 98 | 62 |
| # | 35 | 23 | c | 99 | 63 |

*(tiếp bảng H.1)*

| | | | | | |
|---|---|---|---|---|---|
| $ | 36 | 24 | d | 100 | 64 |
| % | 37 | 25 | e | 101 | 65 |
| & | 38 | 26 | f | 102 | 66 |
| ' | 39 | 27 | g | 103 | 67 |
| ( | 40 | 28 | h | 104 | 68 |
| ) | 41 | 29 | i | 105 | 69 |
| * | 42 | 2A | j | 106 | 6A |
| + | 43 | 2B | k | 107 | 6B |
| , | 44 | 2C | l | 108 | 6C |
| - | 45 | 2D | m | 109 | 6D |
| . | 46 | 2E | n | 110 | 6E |
| / | 47 | 2F | o | 111 | 6F |
| 0 | 48 | 30 | p | 112 | 70 |
| 1 | 49 | 31 | q | 113 | 71 |
| 2 | 50 | 32 | r | 114 | 72 |
| 3 | 51 | 33 | s | 115 | 73 |
| 4 | 52 | 34 | t | 116 | 74 |
| 5 | 53 | 35 | u | 117 | 75 |
| 6 | 54 | 36 | v | 118 | 76 |
| 7 | 55 | 37 | w | 119 | 77 |
| 8 | 56 | 38 | x | 120 | 78 |
| 9 | 57 | 39 | y | 121 | 79 |
| : | 58 | 3A | z | 122 | 7A |
| ; | 59 | 3B | { | 123 | 7B |
| < | 60 | 3C | \| | 124 | 7C |
| = | 61 | 3D | } | 125 | 7D |
| > | 62 | 3E | ~ | 126 | 7E |
| ? | 63 | 3F | DEL | 127 | 7F |

## II. Các mã quét bàn phím (keyboard scan codes)

Bàn phím liên lạc với bộ vi xử lý thông qua các mã quét được gán cho các phím. Khi một phím được nhấn, bàn phím gửi một mã nhấn đến máy tính và khi một phím được nhả ra, một mã nhả được gửi đi. Bảng H.2 chỉ ra các mã nhấn của bàn phím IBM nguyên thủy 83 phím. Từ mã nhấn ta có thể nhận được mã nhả qua phép OR nó với 80h.

Phục vụ ngắt 9h có trách nhiệm nhận mã quét từ cổng vào/ra. Sau đó đưa một mã ASCII và một mã quét đến bộ đệm bàn phím. Chúng ta có thể phân loại các phím thành các phím ASCII, các phím chức năng và các phím dịch. Các phím ASCII bao gồm các phím ký tự, phím trắng và các phím điều khiển: Esc, Enter, Backspase, Tab. Các phím này có các mã ASCII tương ứng. Các phím chức năng bao gồm các phím F1 ÷ F10, các phím mũi tên, Page Up, Page Down, Home, Del, Insert và End. Các phím này không có mã ASCII và trong bộ đệm bàn phím, một số 0 được dùng để xác định một phím chức năng. Các phím dịch gồm có các phím Shift trái, Shift phải, Caps lock, Ctrl , Alt, Num Lock, Scroll Lock. Các mã quét của phím dịch không được đưa vào bộ đệm bàn phím. Thay vào đó phục vụ ngắt 9h dùng các byte cờ bàn phím (địa chỉ 0000:0417) để theo dõi các phím này. Ta có thể nhận được cờ bàn phím thông qua hàm 2 của ngắt 16h.

Khi một phím dịch nào đó được nhấn cùng với các phím khác, phục vụ ngắt 9 đưa một mã quét khác vào bộ đệm bàn phím để xác định các tổ hợp phím.

Bàn phím nâng cao 101 phím sử dụng một hệ thống mã nhả và mã nhấn khác. Tuy vậy, để giữ tính tương thích, ngoại trừ các phím F11 và F12. Phục vụ ngắt 9 vẫn phát sinh 83 mã quét đến bộ đệm bàn phím. Bảng H.4 chỉ ra các mã quét mới phát sinh bởi phục vụ ngắt 9 cho bàn phím nâng cao. Ta cũng thấy một số mã quét tổ hợp mới. Các mã quét mới này chỉ có thể nhận được thông qua ngắt 16h, hàm 10h và 11h.

Khi phím Ctrl được nhấn, ngắt 9 sẽ phát sinh mã ASCII khác cho các phím chữ cái. Bảng H.5 chỉ ra mã quét và mã ASCII của các tổ hợp phím.

***Bảng H.2.*** **Các mã quét bàn phím 83 phím**

| Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím |
|---|---|---|---|---|---|
| 01 | Esc | 1D | Ctrl | 39 | Space bar |
| 02 | 1 | 1E | A | 3A | Caps Lock |
| 03 | 2 | 1F | S | 3B | F1 |
| 04 | 3 | 20 | D | 3C | F2 |
| 05 | 4 | 21 | F | 3D | F3 |
| 06 | 5 | 22 | G | 3E | F4 |
| 07 | 6 | 23 | H | 3F | F5 |
| 08 | 7 | 24 | J | 40 | F6 |
| 09 | 8 | 25 | K | 41 | F7 |
| 0A | 9 | 26 | L | 42 | F8 |
| 0B | 0 | 27 | ; : | 43 | F9 |
| 0C | - _ | 28 | ' " | 44 | F10 |
| 0D | = + | 29 | ` ~ | 45 | Num Lock |
| 0E | Back Space | 2A | Shift trái | 46 | Scroll Lock |
| 0F | Tab | 2B | \ \| | 47 | 7 Home |
| 10 | Q | 2C | Z | 48 | 8 mũi tên lên |
| 11 | W | 2D | X | 49 | 9 PgUp |
| 12 | E | 2E | C | 4A | - (num) |
| 13 | R | 2F | V | 4B | 4 mũi tên trái |
| 14 | T | 30 | B | 4C | 5 (num) |
| 15 | Y | 31 | N | 4D | 6 mũi tên phải |
| 16 | U | 32 | M | 4E | + (num) |
| 17 | I | 33 | , < | 4F | 1 End |
| 18 | O | 34 | . > | 50 | 2 Mũi tên xuống |
| 19 | P | 35 | / ? | 51 | 3 PgDn |
| 1A | { [ | 36 | Shift phải | 52 | 0 Ins |
| 1B | } ] | 37 | Prt Sc | 53 | Del |
| 1C | Enter | 38 | Alt | | |

*Bảng H.3.* **Mã quét các phím tổ hợp**

| Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím |
|---|---|---|---|---|---|
| 54 | Shift + F1 | 65 | Ctrl + F8 | 75 | Ctrl + End |
| 55 | Shift + F2 | 66 | Ctrl + F9 | 76 | Ctrl + PgDn |
| 56 | Shift + F3 | 67 | Ctrl + F10 | 77 | Ctrl + Home |
| 57 | Shift + F4 | 68 | Alt + F1 | 78 | Alt + 1 |
| 58 | Shift + F5 | 69 | Alt + F2 | 79 | Alt + 2 |
| 59 | Shift + F6 | 6A | Alt + F3 | 7A | Alt + 3 |
| 5A | Shift + F7 | 6B | Alt + F4 | 7B | Alt + 4 |
| 5B | Shift + F8 | 6C | Alt + F5 | 7C | Alt + 5 |
| 5C | Shift + F9 | 6D | Alt + F6 | 7D | Alt + 6 |
| 5D | Shift + F10 | 6E | Alt + F7 | 7E | Alt + 7 |
| 5E | Ctrl + F1 | 6F | Alt + F8 | 7F | Alt + 8 |
| 5F | Ctrl + F2 | 70 | Alt + F9 | 80 | Alt + 9 |
| 60 | Ctrl + F3 | 71 | Alt + F10 | 81 | Alt + 0 |
| 61 | Ctrl + F4 | 72 | Ctrl + PrtSc | 82 | Alt + - |
| 62 | Ctrl + F5 | 73 | Ctrl + ← | 83 | Alt + = |
| 63 | Ctrl + F6 | 74 | Ctrl + → | 84 | Ctrl + PgUp |
| 64 | Ctrl + F7 | | | | |

*Bảng H.4.* **Mã quét cho bàn phím nâng cao**

| Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím | Mã quét dạng HEX | Phím |
|---|---|---|---|---|---|
| 85 | F11 | 90 | Ctrl + (num) | 76 | Alt + ← |
| 86 | F12 | 91 | Ctrl + ↓ | 77 | Alt + → |
| 87 | Shift + F11 | 92 | Ctrl + Ins | 78 | Alt + End |
| 88 | Shift + F12 | 93 | Ctrl + Del | 79 | Alt |
| 89 | Ctrl + F11 | 94 | Ctrl + Tab | 7A | Alt + PgDn |
| 8A | Ctrl + F12 | 95 | Ctrl + (num) | 7B | Alt + Ins |
| 8B | Alt + F11 | 96 | Ctrl + (num) | 7C | Alt + Del |
| 8C | Alt + F12 | 97 | Alt + Home | 7D | Alt + (num) |
| 8D | Ctrl + ↑ | 98 | Alt + ↑ | 7E | Alt + Tab |
| 8E | Ctrl + (num) | 99 | Alt + PgUp | 7F | Alt + Enter (num) |
| 8F | Ctrl 5 (num) | | | | |

*Bảng H.5.* **Các mã ASCII của các phím tổ hợp**

| Phím | Mã ASCII dạng HEX | Mã quét dạng HEX | Phím | Mã ASCII dạng HEX | Mã quét dạng HEX |
|---|---|---|---|---|---|
| Ctrl + A | 1 | 1E | Ctrl + N | E | 41 |
| Ctrl + B | 2 | 30 | Ctrl + O | F | 18 |
| Ctrl + C | 3 | 2E | Ctrl + P | 10 | 19 |
| Ctrl + D | 4 | 20 | Ctrl + Q | 11 | 10 |
| Ctrl + E | 5 | 12 | Ctrl + R | 12 | 13 |
| Ctrl + F | 6 | 21 | Ctrl + S | 13 | 1F |
| Ctrl + G | 7 | 22 | Ctrl + T | 14 | 14 |
| Ctrl + H | 8 | 23 | Ctrl + U | 15 | 16 |
| Ctrl + I | 9 | 17 | Ctrl + V | 16 | 2F |
| Ctrl + J | A | 24 | Ctrl + W | 17 | 11 |
| Ctrl + K | B | 25 | Ctrl + X | 18 | 2D |
| Ctrl + L | C | 26 | Ctrl + Y | 19 | 15 |
| Ctrl + M | D | 32 | Ctrl + Z | 1A | 2C |

# PHỤ LỤC B

## CÁC LỆNH CỦA VI XỬ LÝ HỌ INTEL 80X86

### A. CÁC LỆNH CỦA 8086/8088

Các lệnh xếp theo thứ tự a, b, c. Các cờ có thể bị tác động sau khi thực hiện lệnh (= 1) và không bị tác động (= 0).

1. **AAA** (Adjust After for Addition - chỉnh sau khi cộng hai số dạng mã ASCII).

Lệnh này đổi nội dung của thanh ghi AH, AL chứa con số của tổng số thành con số dạng mã ASCII.

Lệnh dùng sau lệnh cộng (ADD) hai số mã ASCII hay BCD không nén.

*Cú pháp:* AAA (không có toán hạng hay toán hạng ẩn là AH, AL).

*Cờ*: AF, CF bị tác động; OF, PF, SF, ZF không bị tác động.

2. **AAD** ( ASCII Adjust before Division - chỉnh mã ASCII trước khi chia).

Lệnh này đổi hai số BCD không nén trong AH, AL sang số hệ nhị phân tương đương tại AL để thực hiện phép chia nhị phân tiếp theo.

Lệnh dùng trước lệnh chia (DIV, IDIV).

*Cú pháp:* ADD (không có toán hạng hay toán hạng ẩn là AH, AL).

*Cờ*: Không bị tác động.

3. **AAM** (ASCII Adjust after Multiplication - chỉnh mã ASCII sau khi nhân).

Lệnh này đổi một con số hệ nhị phân (hay Hexadecimal) sang số BCD không nén tại AX.

Lệnh này dùng để biến đổi tích số của hai số dạng nhị phân sau khi thực hiện lệnh nhân (MUL, IMUL) sang số dạng mã BCD không nén.

*Cú pháp:* AMM (không có toán hạng hay toán hạng ẩn là AH, AL).

*Cờ:* PF, SF, ZF bị tác động; AF, CF, OF không bị tác động.

4. **AAS** (ASCII Adjust after Subtraction - chỉnh mã ASCII sau khi trừ).

Lệnh này đổi số hệ nhị phân là hiệu số của hai số BCD không nén ở trong AL sang số BCD.

*Cú pháp:* AAM (không có toán hạng hay toán hạng ẩn là AH, AL).

*Cờ*: AF, CF bị tác động; OF, PF, SF, ZF không bị tác động.

5. **ADC** (Addition with carry - cộng có nhớ ).

Lệnh cộng hai số trong hai toán hạng Đích, Gốc và CF. Nếu CF = 0, kết quả giống như lệnh cộng ADD, nhưng thường CF = 1. Kết quả của phép cộng được gửi vào đích là một trong hai thanh ghi AL hoặc AH. Toán hạng Gốc có thể là số liệu trực tiếp, số liệu trong thanh ghi khác hoặc trong ô nhớ. Các toán hạng trên chứa dữ liệu có cùng độ dài, không được phép đồng thời là hai ô nhớ và cũng không được là thanh ghi đoạn.

*Cú pháp:* ADC Đích, Gốc.

*Mô tả:* Đích ← Đích + Gốc + CF.

*Cờ*: AF, CF, OF, TF, SF, ZT bị tác động

6. **ADD** (Addition - cộng hai toán hạng).

Phép cộng này giống phép cộng có nhớ, nhưng CF = 0.

*Cú pháp:* ADC Đích, Gốc.

*Mô tả:* Đích ← Đích + Gốc.

*Cờ*: AF, CF, OF, TF, SF, ZT bị tác động

7. **AND** (And Corresponding Bits of Two Operands - và tương ứng với từng bit của hai toán hạng).

*Cú pháp*: AND Đích, Gốc

*Mô tả* : Đích ← Đích ∧ Gốc (cho từng bit của Đích và Gốc).

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa dữ liệu cùng độ dài (số bit bằng nhau) và không được phép đồng thời là 2 ô nhớ hoặc 2 thanh ghi đoạn. Phép toán logic AND thường dùng để che hoặc giữ lại một vài bit nào đó của một toán hạng, trong đó số liệu tức thì là các bit 0 hoặc 1 ở chỗ cần che đi hoặc giữ lại các bit tương ứng của một thanh ghi. Toán hạng tức thì này còn gọi là mặt nạ.

*Cờ:* CF, OF bị xóa; PF, SF, ZF và PF bị thay đổi khi toán hạng là 8 bit; AF không xác định.

8. **CALL** - Call a Procedure (gọi chương trình con).

*Cú pháp:* Call tên chương trình con.

*Mô tả:* Lệnh này dùng để gọi một chương trình con tức chuyển hoạt động của bộ vi xử lý từ chương trình chính sang chương trình con có tên kèm theo lệnh gọi trên. Có hai cách gọi chương trình con:

- *Gọi gần (Near call)* : Nếu chương trình con ở cùng một đoạn mã với chương trình chính hay địa chỉ của chương trình con ở trong thanh ghi IP còn thanh ghi CS không thay đổi.

- *Gọi xa (Far call)* : Chương trình con nằm ở đoạn mã khác so với đoạn mã của chương trình chính, tức địa chỉ của chương trình con có giá trị CS: IF, IP.

Cuối mỗi chương trình con phải có lệnh trở về RET (Return). Do đó trước khi gọi chương trình con vi xử lý phải tự động cất giữ vào ngăn xếp có địa chỉ SP, địa chỉ trở về lệnh của chương trình chính cần thực hiện sau khi gọi chương trình con. Địa chỉ này được tự động lấy ra tại ngăn xếp để chương trình chính tiếp tục được thực hiện sau khi gọi chương trình con.

Như vậy, nếu là gọi gần:

+ SP ← SP - 2 và địa chỉ lệnh trở về được cất vào ngăn xếp {SP} ← IP

+ IP ← Địa chỉ lệnh của chương trình con

+ Khi gặp lệnh RET tại cuối ctc thì sẽ có thao tác ngược lại:

IP ← {SP} và SP ← SP + 2.

còn nếu gọi xa:

+ SP ← SP - 2 là địa chỉ lệnh của lệnh trở về được cất vào ngăn xếp {SP} ← CS

+ SP ← SP - 2 và địa chỉ cơ sở của lệnh trở về được cất vào ngăn xếp {SP} ← IP

+ IP ← Địa chỉ lệnh của chương trình con.

CS ← Địa chỉ cơ sở của chương trình con.

+ Khi gặp lệnh RET tại cuối ctc thì sẽ có thao tác ngược lại:

IP ← {SP} và SP ← SP + 2

CS ← {SP} và SP ← SP + 2.

*Viết lệnh:* Lệnh gọi chương trình con trên có thể viết theo một trong các dạng sau:

+ **CALL Multiple**: Gọi chương trình con có tên là Multipe, trong đó tên Multipe đã được xác định có địa chỉ nằm trong cùng đoạn mã với chương trình chính nhưng chương trình con này nằm trong giới hạn dịch chuyển là - 32Kbyte (dịch về phía địa chỉ thấp) hoặc 32K-1byte (dịch về phía địa chỉ cao) so với lệnh tiếp theo ngay sau lệnh Call. Sau khi cất IP cũ (địa chỉ trở về ) vào ngăn xếp, IP mới được tính: IP ← IP + dịch chuyển.

+ **CALL Divi**: Gọi chương trình con có tên Divi có địa chỉ ở đoạn mã khác, đã được khai báo là một chương trình con ở xa như sau: Divi Proc Far.

Địa chỉ của chương trình con nằm trong cặp thanh ghi CS: IP.

+ **CALL BX**: Gọi trực tiếp một chương trình con trong cùng đoạn mà lệnh đầu tiên của chương trình con được dịch chuyển một lượng bằng nội dung của thanh ghi BX, do đó IP←BX. Các thanh ghi SI, DI có thể dùng thay chỗ của BX.

+ **CALL WORD PTR [BX]**: Gọi chương trình con nằm trong cùng đoạn mã, chương trình con này có địa chỉ dịch chuyển (tính từ lệnh tiếp ngay sau lệnh gọi tới lệnh đầu tiên của chương trình con chứa trong 2 ô nhớ có địa chỉ là BX và BX+1 trong đoạn DS. Địa chỉ lệch này sẽ được đưa vào IP. SI, DI có thể dùng thay chỗ cho BX.

+ **CALL DWORD PTR [BX]**: Gọi chương trình con không nằm trong cùng một đoạn mã, chương trình con có địa chỉ CS:IP mà giá trị gán cho IP và CS chứa trong 4 ô nhớ do BX và BX+1 (IP, BX+2 và BX+3) (cho CS) chỉ ra trong đoạn DS. SI, DI có thể dùng thay chỗ của BX.

9. **CBW** (Convert a Byte to a Word - chuyển byte thành từ).

*Cú pháp:* CBW (không có toán hạng hay toán hạng ẩn là AH, AL).

*Mô tả :* Lệnh này mở rộng bit dấu của thanh ghi AL sang 8 bit của AH. AH lúc này được gọi là phần mở rộng dấu của AL. Ta dùng CBW để mở rộng dấu cho số có dấu nằm trong AL trước khi chia nó cho một số có dấu 8 bit khác bằng lệnh IDIV (lệnh chia các số có dấu) hoặc muốn nhân nó với một số có dấu 16 bit khác bằng lệnh IMUl (lệnh nhân các số có dấu).

*Cờ:* Không tác động đến các cờ.

10. **CLC** (Clear the Carry Flag- xóa cờ nhớ).

*Cú pháp:* CLC (không có toán hạng hay toán hạng ẩn là CF).

*Mô tả*: Xóa cờ nhớ CF về 0, CF ← 0.

*Cờ:* Không tác động đến các cờ khác ngoài cờ CF trên.

11. **CLD** (Clear the Direction Flag - xóa cờ hướng).

*Cú pháp:* CLD (không có toán hạng hay toán hạng ẩn là DF).

*Mô tả:* Xóa cờ hướng DF về 0, DF ← 0.

Lệnh này định hướng chiều tiến của địa chỉ chứa số liệu nằm trong bộ nhớ. Thường dùng lệnh này trong các lệnh liên quan tới chuỗi ký tự hoặc bảng số liệu. Lệnh này trái ngược với lệnh xác lập D (STD) để định hướng chiều giảm của địa chỉ chứa số liệu nằm trong bộ nhớ.

*Cờ:* Không tác động đến các cờ khác ngoài cờ DF trên.

12. **CLI** (Clear the Interrupt Flag - xóa cờ ngắt IF).

*Cú pháp:* CLI (không có toán hạng hay toán hạng ẩn là IF).

*Mô tả:* Xóa cờ ngắt IF về 0, IF ←0.

*Cờ:* Không tác động đến các cờ khác ngoài cờ IF trên.

13. **CMC** (Complement the carry flag - lấy bù cờ nhớ).

*Cú pháp:* CMC (không có toán hạng hay toán hạng ẩn là CF).

*Mô tả:* Lấy bù hay đảo giá trị bit của cờ nhớ CF (tức 0← 1 hay 1← 0). CF ← CF.

*Cờ:* Không tác động đến các cờ khác ngoài cờ CF trên.

14. **CMP** (Compare - so sánh).

*Cú pháp:* CMP Đích, Gốc

*Mô tả:* Đích - Gốc

Lệnh này lấy toán hạng Đích trừ đi toán hạng Gốc (giống lệnh trừ SUB Đích, Gốc) nhưng:

- Không có hiệu số gửi vào Đích (khác với lệnh trừ) hay các toán hạng không bị thay đổi.
- Các cờ chỉ kết quả so sánh hay kết quả của phép trừ (giống lệnh trừ).

Lệnh này thường được dùng để tạo cờ cho các lệnh nhảy có điều kiện tức nhảy chương trình theo các cờ (thường là các cờ CF, ZF).

Các toán hạng được tìm theo các chế độ địa chỉ khác nhau, chứa dữ liệu có cùng độ dài (số bit bằng nhau), nhưng không được phép đồng thời là hai ô nhớ.

Kết quả so sánh của hai số Đích, Gốc không dấu căn cứ vào hai cờ chính CF, ZF như sau:

| | CF | ZF |
|---|---|---|
| Đích > Gốc | 0 | 0 |
| Đích = Gốc | 0 | 1 (kết quả phép trừ = 0 nên ZF = 1) |
| Đích < Gốc | 1 | 0 (kết quả phép trừ < 0 nên có nhớ và CF = 1) |

*Các cờ:* Tất cả các cờ AF, CF, OF, PF, SF, ZF bị tác động.

15. **CMPS/CMPSB/CMPSW** (Compare String Bytes or String Words - so sánh 2 chuỗi byte hay 2 chuỗi từ).

*Cú pháp:* CMPS chuỗi Đích, chuỗi Gốc(chuỗi có độ dài chưa biết).

CMPSB chuỗi Đích, chuỗi Gốc (chuỗi là một byte).

CMPSW chuỗi Đích, chuỗi Gốc ( chuỗi là một từ).

*Mô tả:* Lệnh so sánh từng phần tử (byte hay từ) của hai xâu có các phần tử cùng loại byte hay từ. Cũng giống lệnh so sánh CMP, kết quả so sánh chỉ tác động lên các cờ và các toán hạng không bị thay đổi. Các toàn hạng chuỗi Đích và Gốc là nội dung của các ô nhớ có địa chỉ ở trong các thanh ghi đoạn như sau:

- DS:SI là địa chỉ của phần tử so sánh của chuỗi Gốc.
- ES:DI là địa chỉ của phần tử so sánh của chuỗi Đích.

Sau mỗi lần so sánh, giá trị của các thanh ghi SI, DI tăng lên (DF = 0) hay giảm đi (DF = 1) một đơn vị (so sánh hai byte) hay hai đơn vị (so sánh hai từ).

Lệnh so sánh trên có thể dùng kèm theo tiếp đầu ngữ REPE (lặp lại sự so sánh nếu kết quả so sánh bằng nhau để tìm chuỗi tương ứng khác nhau) và REPNE (lặp lại sự so sánh nếu kết quả không bằng nhau để tìm chuỗi tương ứng bằng nhau).

*Cờ:* Tác động lên mọi cờ AF, CF, OF, PF, SF, ZF.

16. **CWD** (Convert a Word to a Double Word - biến đổi một từ thành hai từ).

*Cú pháp:* CWD (không có toán hạng hoặc toán hạng ẩn nằm trong các thanh ghi AX (Gốc) và DX:AX (Đích).

*Mô tả:* Lệnh này tương tự lệnh mở rộng dấu CBW, nhưng chỉ khác là mở rộng bit dấu trong AX sang 16 bit của DX, để thực hiện phép chia số có dấu (lệnh IDIV) có 16 bit.

*Cờ:* Không tác động tới các cờ.

17. **DAA** (Decimal Adjust after Addition - hiệu chỉnh thập phân sau phép cộng).

*Cú pháp:* DAA (không có toán hạng hoặc toán hạng ẩn nằm trong thanh ghi AX).

*Mô tả:* Lệnh này dùng để hiệu chỉnh kết quả nằm ở AL sau phép cộng hai số mã BCD có nén (ADD Đích, Gốc) vì khối số học-logic ALU của vi xử lý chỉ thực hiện được phép cộng hai số nhị phân.

Kết quả hiệu chỉnh như sau:

- Nếu 4 bit thấp của AL lớn hơn 9 nên cờ nhớ phụ AF = 1 thì AL ← AL+6 để đổi mã dạng 16 - hexadecimal thành dạng thập phân của hàng đơn vị, tức chuyển một số lớn hơn 10 (A, B, C, D, E, F) của hàng đơn vị thành số hàng chục và đơn vị.

- Nếu 4 bit cao của AL lớn hơn 9 nên cờ nhớ CF = 1 thì AL ← AL + 60H để đổi mã dạng 16 thành dạng thập phân của hàng chục, tức chuyển một số hàng chục lớn hơn 10 thành số hàng trăm và hàng chục.

*Cờ:* AF, CF, PF, SF, ZF bị tác động, OF không bị tác động.

18. **DAS** (Decimal Adjust after Subtraction - hiệu chỉnh thập phân sau phép trừ).

*Cú pháp:* DAS (không có toán hạng hay toán hạng ẩn trong AX).

*Mô tả:* Tương tự lệnh DAA cho phép cộng, lệnh này hiệu chỉnh kết quả cho phép trừ hai số BCD nén, dùng sau lệnh trừ hai số (SUB Đích, Gốc). Phép hiệu chỉnh như sau:

- Nếu 4 bit thấp của AL lớn hơn 9 nên cờ nhớ phụ AF = 1 thì AL ← AL - 6 .

- Nếu 4 bit cao của AL lớn hơn 9 nên cờ nhớ CF = 1 thì AL ← AL - 60H .

*Cờ:* AF, CF, PF, SF, ZF bị tác động, OF không bị tác động.

19. **DEC** (Decrement- giảm).

*Cú pháp:* DEC Đích (chỉ có một toán hạng).

*Mô tả:* Giảm toán hạng đích đi 1 đơn vị . Đích ← Đích - 1.

Đích có thể tìm được theo các chế độ địa chỉ khác nhau. Lệnh này tương đương lệnh SUB Đích 1 nhưng chạy nhanh hơn.

*Cờ:* AF, OF, PF, SF, ZF bị tác động, CF không bị tác động.

20. **DIV** (Divide - chia).

*Cú pháp:* DIV Gốc ( Đích ẩn là thanh ghi AX cho gốc là số 8 bit, còn cặp thanh ghi DX:AX cho Gốc là thanh ghi 16 bit).

Tùy độ dài của Gốc, ta có kết quả như sau:

- *Nếu Gốc là 8 bit:* số thương trong AL, số dư trong AH.

- *Nếu Gốc là 16 bit:* số thương trong AX, số dư trong DX.

- *Nếu Gốc = 0:* số thương lớn vô cùng, vi xử lý bị treo với lệnh ngắt INT 0.

Nếu số thương không phải là số nguyên, nó được làm tròn theo số nguyên sát dưới hay lệnh chia DIV chỉ chia lấy thương là số nguyên.

*Cờ:* Các cờ không bị tác động.

21. **ESC** (Escape - giải thoát).

*Cú pháp:* ESC (không có toán hạng).

*Mô tả :* Lệnh này dùng để truyền các lệnh cho bộ đồng xử lý toán học (80x87). Còn vi xử lý 80x86 coi như lệnh NOP ( không hoạt động) hoặc truy cập dữ liệu từ bộ nhớ rồi đưa đến cho bộ đồng xử lý toán.

*Cờ:* các cờ không bị tác động.

22. **HLT** (Halt - dừng).

*Cú pháp:* HLT (không có toán hạng).

*Mô tả:* Vi xử lý dừng ở trạng thái dừng. Thường dùng lệnh này để kết thúc một đoạn chương trình. Muốn thoát khỏi trạng thái dừng này, cần tác động một xung kích thích vào một trong các chân INTR, NMI hoặc RESET của bộ vi xử lý.

*Cờ:* Các cờ không bị tác động.

23. **IDIV** (Integer Division - chia số nguyên).

*Cú pháp:* IDIV Gốc ( Đích ẩn là AX hay DX:AX).

*Mô tả:* Chia số bị chia là số nguyên hay số có dấu (+ hay -) trong AX hay DX:AX cho số chia là toán hạng Gốc. Toán hạng Gốc tìm được theo chế độ địa chỉ khác nhau.

Thanh ghi lưu trữ kết quả cũng giống như ở lệnh chia DIV (thương số trong AL hoặc AX, còn số dư trong AH hoặc DX).

Nếu Gốc = 0, vi xử lý cũng xảy ra bị tràn với ngắt INT 0 như lệnh DIV.

*Cờ:* Các cờ không bị tác động.

*24.* **IMUL** (Integer Multiplication - nhân số nguyên).

*Cú pháp:* IMUL Gốc (Đích ẩn là AX).

*Mô tả:* Nhân số bị nhân có dấu (trong AL hoặc AX) với số nhân chứa trong Gốc theo các phương pháp địa chỉ khác nhau. Kết quả của phép nhân đặt trong:

- Thanh ghi AX nếu Gốc có 8 bit.
- Cặp thanh ghi DX:AX nếu Gốc có 16 bit.

Nếu tích số là nhỏ, không lấp đầy hết các chỗ dành cho nó, các bit không dùng đến được thay bằng bit dấu.

Nếu byte cao của 16 bit (hoặc 32 bit) kết quả chỉ chứa các giá trị của bit dấu thì CF = OF = 0.

Nếu byte cao của 16 bit (hoặc 32 bit) kết quả chứa một phần kết quả thì CF = OF = 1 (CF và OF báo độ dài của kết quả không kể dấu).

*Cờ:* CF, OF bị tác động; các bit khác (AF, PF, SF, ZF) không bị tác động.

*25.* **IN** (Input - đưa vào).

*Cú pháp:* IN thanh chứa, cổng. Thanh chứa ← {Cổng}.

*Mô tả:* Lệnh đưa số liệu (nhập) từ cổng vào thanh ghi chứa (AL hoặc AX) của vi xử lý. Cổng đây có thể là:

- 8 bit hay 16 bit của một số tức thì chỉ địa chỉ của cổng (hay ô nhớ).
- Thanh ghi DX chỉ địa chỉ của cổng (hay ô nhớ) được gán trước đó.

Nếu: + Thanh chứa là AL thì dữ liệu 8 bit được đưa vào từ cổng .

+ Thanh chứa là AX thì dữ liệu 16 bit được đưa vào từ địa chỉ cổng và cổng + 1.

*Cờ:* Không cờ nào bị tác động.

26. **INC** (Increment - tăng lên).

*Cú - pháp*: INC Đích.

*Mô tả:* Đích được tăng lên 1 đơn vị. Đích ← Đích + 1. Lệnh này trái ngược với lệnh DEC Đích, tương đương lệnh ADD Đích, 1 nhưng chạy nhanh hơn.

*Cờ:* Các bit bị tác động trừ bit CF.

27. **INT** (Interupt - ngắt hay gián đoạn)

*Cú pháp:* INT n trong đó n = một số từ 0 tới FFh.

*Mô tả:* Làm ngắt hay làm gián đoạn việc thực hiện chương trình theo mức ngắt n được quy định sẵn ở hệ điều hành MS-DOS.

*Thao tác:* Khi thực hiện lệnh INT n thì:

- Thanh ghi cờ (FLAG) được lưu trữ vào ngăn xếp (địa chỉ ở thanh ghi quản lý ngăn xếp SP), bit TF = 0 (xóa về 0 để cấm các ngắt khác tác động) và bit TF = 0 (xóa về 0 để chương trình chạy suốt không bị bẫy). Thanh ghi SP tự động giảm đi 2.

- Thanh ghi CS được lưu trữ vào ngăn xếp (địa chỉ ở SP-2). Thanh ghi SP tự động giảm đi 2.

- Thanh ghi IP được lưu trữ vào ngăn xếp (địa chỉ ở SP-4). Thanh ghi SP tự động giảm đi 2.

- Véctơ ngắt của chương trình phục vụ ngắt thay vào cặp thanh ghi CS:IP để vi xử lý nhảy tới chương trình phục vụ ngắt. Cụ thể là:

+ IP ← {nx4} trong đó n là số hiệu ngắt của lệnh INT n (n có giá trị từ 1 tới 255 ).

+ CS ← {nx4 + 2}.

28. **INT 0** (Interrupt on Overflow - ngắt nếu bị tràn)

*Mô tả*: Tạo ra ngắt 4 nếu bị tràn số (chia cho 0 hay nạp vào số quá lớn mà thanh ghi không chứa hết).

*Cờ:* OF = 1, các bit cờ khác không bị tác động.

29. **IRET** (Interrupt RETurn - trở về chỗ ngắt ngắt).

*Cú pháp:*IRET (không có toán hạng).

*Mô tả:* Trở về chương trình chính từ chương trình con phục vụ ngắt. Lệnh này là lệnh cuối cùng của chương trình con phục vụ ngắt, để vi xử lý thực hiện các thao tác hồi phục trạng thái của vi xử lý trước khi nhảy tới chương trình con và tiếp tục thực hiện chương trình chính đã bị ngắt do chạy chương trình con phục vụ ngắt.

*Thao tác:* - Hồi phục cặp thanh ghi đoạn lệnh:

+ IP ← { SP-4}, SP - 4 tự động tăng lên 2 thành SP-2.

+ CS ← { SP-2}, SP-2 tự động tăng lên 2 thành SP.

- Hồi phục thanh ghi FLAG: FLAG ← SP.

*Cờ:* Các bit cờ bị tác động.

30. **J** (điều kiện) (Jump condition - nhảy chương trình có điều kiện).

*Cú pháp:* J (điều kiện) nhãn.

*Mô tả*: Nếu điều kiện xảy ra, việc thực hiện chương trình *nhảy* tới *nhãn*.Nhãn ở đây là một ký hiệu đã được khai báo hay định nghĩa bởi một con số chỉ địa chỉ lệnh của chương trình trong khối nhớ. Nhãn phải nằm trong phạm vi từ 128 byte đến 127 byte tính từ lệnh tiếp theo của chương trình (nhảy gần hay ngắn).

*Thao tác:* Có hai hành động xảy ra khi vi xử lý gặp lệnh J (điều kiện):

- Kiểm tra điều kiện để nhảy chương trình:

+ Nếu điều kiện không đúng hay không thoả mãn sẽ không có nhảy chương trình mà thực hiện lệnh tiếp theo của chương trình nằm sau lệnh J (điều kiện).

+ Nếu điều kiện đúng hay thoả mãn điều kiện, sẽ có nhảy chương trình bằng cách cộng nhãn với nội dung hiện tại của thanh ghi IP, rồi vi xử lý sẽ thực hiện các lệnh bắt đầu từ địa chỉ có giá trị là IP + nhãn. Vì nhãn có thể có giá trị âm hay dương nên:

- Chương trình có thể quay trở về đoạn đã thực hiện để thực hiện lại lệnh đã chạy (nhãn có giá trị âm).

- Chương trình có thể bỏ qua một số lệnh ở sát chỗ nhảy mà tiếp tục thực hiện phần sau của chương trình.

*Cờ:* Các bit không bị tác động.

Tuỳ theo điều kiện, ta có tên các lệnh nhảy có điều kiện như bảng sau:

***Bảng H.6.* Tên các lệnh nhảy có điều kiện**

| Lệnh | Điều kiện | Điều kiện của cờ |
|---|---|---|
| JA | Above (lớn hơn) | CF = 0 và ZF = 0 |
| JAE | Above or equal (lớn hơn hoặc bằng) | CF = 0 |
| JB | Below ( nhỏ hơn) | CF = 1 |
| JBE | Below or equal (nhỏ hơn hoặc bằng) | CF = 1 hay ZF = 1 |
| JC | Carry (nhớ) | CF = 1 |
| JCXZ | CX = 0 | (CF or ZF) = 0 |
| JE | Equal (bằng) | ZF = 1 |
| JG | Greater (lớn hơn) | ZF = 0 và SF = 0 |
| JGE | Greater or equal (lớn hơn hoặc bằng) | ZF = OF |
| JL | Less (nhỏ hơn) | (SF xor OF) = 1 |
| JLE | Less or equal (nhỏ hơn hoặc bằng) | (SF xor OF) = 0 , ZF = 1 |
| JNA | Not Above (không lớn hơn) | CF = 1 hay ZF = 1 |
| JNAE | Not Above or equal (không lớn hơn) | CF = 1 |
| JNB | Not Below(không lớn hơn) | CF = 0 |
| JNBE | Not Below or Equal(không lớn hơn hoặc bằng) | CF = 0 hay ZF = 0 |
| JNC | Not Carry(không nhớ) | CF = 0 |
| JNE | Not Equal(không bằng) | ZF = 0 |
| JNG | Not Greater(không lớn hơn) | (SF xor OF) = 0, ZF = 1 |
| JNGE | Not Greater or Equal(không lớn hơn hoặc bằng) | (SF xor OF) = 1 |
| JNL | Not Less (không nhỏ hơn) | SF = OF |
| JNLE | Not Less or Equal (không nhỏ hơn hoặc bằng) | ZF = 0 và SF = 0 |
| JNO | Not Overflow (không tràn) | OF = 0 |
| JNP | Not Parity (không chẵn) | PF = 0 |
| JNS | Not Signe (không dấu) | SF = 0 |
| JNZ | Not Zero (không bằng 0) | ZF = 0 |
| JO | Overflow (Tràn) | OF = 0 |
| JP | Parity (tính chẵn lẻ) | PF = 1 |
| JPE | Parity Even (tính chẵn lẻ là chẵn) | PF = 1 |
| JPO | Parity Odd (tính chẵn lẻ là lẻ) | PF = 0 |
| JS | Sign (số có dấu) | SF = 1 |
| JZ | Zero (bằng 0) | ZF = 1 |

*Chú ý:* 1. Các bit của thanh ghi cờ đều được sử dụng để xét điều kiện nhảy chương của chương trình.

2. Mỗi bit của thanh ghi cờ có hai lệnh nhảy với điều kiện thoả mãn và không thoả mãn (Not). Do đó số lượng lệnh nhảy tăng gấp đôi số điều kiện.

3. Có điều kiện thoả mãn hai điều kiện nhỏ hay hai bit cờ (ví dụ không hoặc bằng, không nhỏ, không lớn hơn).

4. Các lệnh JA/JB dùng để nhảy khi điều kiện so sánh các số có dấu(-,+) còn JG/JL cho các số không dấu hay số dương (+).

5. Có các cặp lệnh tương đương vì cùng một điều kiện của bit cờ với tên gọi khác nhau. Ví dụ lệnh JB và JNAE có điều kiện CF = 1.

6. Ngoài lệnh nhảy chương trình có điều kiện J (điều kiện), có vi xử lý có các *lệnh gọi chương trình con có điều kiện C (điều kiện)* tương ứng với các lệnh nhảy chương trình có điều kiện như bảng trên.

31. **JMP** (Jump - nhảy).

*Cú pháp:* JMP nhãn.

*Mô tả:* Nhảy chương trình không điều kiện tới lệnh có địa chỉ là nhãn. Nhãn ở đây có thể ở:

- Trong đoạn mã lệnh (chỉ có IP thay đổi giá trị, còn CS giữ nguyên giá trị cũ), được gọi là nhảy gần.

- Ngoài đoạn mã lệnh (cả IP lẫn CS thay đổi giá trị), được gọi là nhảy xa.

Người ta còn phân biệt lệnh theo chế độ địa chỉ:

+ *Địa chỉ trực tiếp:* Nhãn là thanh ghi BX, tức giá trị của thanh ghi BX là địa chỉ của lệnh cần nhảy tới của chương trình. Cách viết lệnh:

JMP BX hoặc JMP NEAR PTR BX.

+ *Địa chỉ gián tiếp*: Nhãn là [BX], tức nội dung ô nhớ có địa chỉ là giá trị của BX sẽ là địa chỉ của lệnh cần nhảy tới của chương trình. Cách viết lệnh:

- Nhảy gần: JMP [BX] hoặc JMP NEAR PTR [BX] chỉ hai nội dung của các ô nhớ có địa chỉ là giá trị của BX và BX+1 được nạp vào IP để thực hiện lệnh nhảy.

- Nhảy xa: JMP DWORD PTR [BX] chỉ hai từ của 4 ô nhớ có các địa chỉ BX, BX+1 được nạp vào IP và nội dung của các địa chỉ BX+3, BX+4 được nạp vào CS để thực hiện lệnh nhảy.

32. **LAHF** (Load AH from Flag - nạp vào AH từ Flag).

*Cú pháp:* LAHF (không có toán hạng, toán hạng đích ẩn là AH và toán hạng nguồn ẩn là FLAG).

*Mô tả:* Chuyển 8 bit thấp của thanh ghi cờ (CF, ZF, AF, SF, PF) vào thanh ghi AH để kiểm tra hoặc cất vào ngăn xếp). AH ← FLAG.

*Cờ:* Không bị tác động.

33. **LDS** (Load Data Segment - nạp thanh ghi đoạn dữ liệu).

*Cú pháp:* LDS Đích, Gốc.

Đích là một trong các thanh ghi thông dụng (AX, BX, CX, DX, SP, BP, SI, DI).

Gốc là một ô nhớ có độ dài 2 từ.

*Thao tác:* - Nội dung ô nhớ có địa chỉ là Gốc gửi vào Đích. Đích ← [Gốc]

- Nội dung ô nhớ có địa chỉ là Gốc+2 gửi vào DS. DS ← [Gốc+2]

34. **LEA**(Load Effective Address - nạp địa chỉ hiệu dụng).

*Cú pháp:* LEA Đích, Gốc

Đích thường là một trong các thanh ghi thông dụng (BX, CX, DX, BP, SI, DI).

Gốc là một ô nhớ có độ dài 2 từ.

*Thao tác:* Địa chỉ lệch (offset) của toán hạng Gốc (nguồn) được đặt vào toán hạng Đích là một thanh ghi thông dụng. Đích ←Địa chỉ lệch của Gốc ( hoặc địa chỉ hiệu dụng của Gốc).

Lệnh này dùng để *nạp địa chỉ của ô nhớ Gốc vào một trong các thanh ghi thông dụng.*

*Cờ:* Không bị tác động.

35. **LES** (Load ES - nạp thanh ghi ES).

*Cú pháp:* LES Đích, [Gốc].

Đích là một trong các thanh ghi thông dụng (AX, BX, CX, DX, SP, BP, SI, DI).

Gốc là nội dung ô nhớ có độ dài 2 từ.

*Thao tác:* Lệnh này nạp địa chỉ cho cặp thanh ghi đoạn số liệu phụ và một thanh ghi thông dụng (ES: thanh ghi thông dụng), cụ thể là:

- Nạp từ thấp của ô nhớ có địa chỉ là Gốc vào thanh ghi Đích. Đích ←[Gốc].

- Nạp từ cao của ô nhớ có địa chỉ là Gốc + 2 vào thanh ghi đoạn ES (Đích ẩn của cặp thanh ghi ES: thanh ghi thông dụng). ES ← [Gốc + 2].

Đây là lệnh nạp vào thanh ghi đã chọn và ES từ nội dung của 4 ô nhớ liên tiếp có địa chỉ ban đầu là địa chỉ của Gốc. Thường dùng lệnh này để nạp địa chỉ đầu tiên của vùng nhớ chứa chuỗi Đích số liệu trong các lệnh thao tác chuỗi số liệu.

*Cờ:* Không bị tác động.

36. **Lock** (Lock - khoá).

*Cú pháp:* Lock (không có toán hạng).

*Thao tác:* Vi xử lý đưa ra tín hiệu khoá BUS trong hệ có nhiều bộ xử lý (ví dụ đồng xử lý 8087). Lệnh này còn đứng trước một lệnh khác (đóng vai trò tiền tố) để tránh sự va chạm BUS trong thời gian thực hiện lệnh đó. (Ví dụ LOCK XCHG AL, MEM để tránh vi xử lý khác xâm nhập BUS trong thời gian thực hiện lệnh XCHG giữa AL và ô nhớ MEM).

*Cờ:* Không cờ nào bị tác động.

37. **LODS/LODSB/LODSW** (LOAd String Byte/Word - nạp chuỗi byte/từ).

*Cú pháp:* LODS chuỗi Gốc (toán hạng Đích ẩn là AL.AX).

LODSB chuỗi Gốc

LODSW chuỗi Gốc

*Thao tác :* Nội dung ô nhớ Gốc có địa chỉ là nội dung của cặp thanh ghi DS:SI được chuyển vào thanh ghi AL (lệnh LODSB) hay AX (lệnh LODW). Tùy theo giá trị của DF mà giá trị của SI được tự động tăng (DF = 0) hay giảm (DF = 1) đi 1 đơn vị (LODSB) hay hai đơn vị (LODSW).

- *LODSB cho:* AL ← [ DS:SI] và SI ←SI + 1 nếu DF = 0.
  SI ←SI - 1 nếu DF = 1.
- *LODW cho:* AX ← [ DS:SI ] và SI ←SI + 2 nếu DF = 0.
  SI ←SI - 2 nếu DF = 1.

*Cờ:* Không bị tác động.

38. **LOOP** (lặp).

*Cú pháp:* LOOP Nhãn

Nhãn là một ký hiệu hoặc một con số chỉ địa chỉ của lệnh trong chương trình. Nếu là ký hiệu thì ký hiệu phải được khai báo trong chương trình để ký hiệu đó được gán một địa chỉ nào đó.

*Thao tác:* Thanh ghi được giảm đi 1 (CX-1) và nếu kết quả khác 0 thì chương trình lại quay trở về lệnh có địa chỉ Nhãn. Cứ như vậy sẽ có CX lần lặp lại, chương trình thực hiện CX lần vòng lặp (gồm các lệnh từ Nhãn tới LOOP Nhãn) và khi CX = 0 chương trình không lặp lại nữa mà thực hiện lệnh tiếp theo sau lệnh LOOP Nhãn trên.

*Cờ:* Không bị tác động.

Ngoài lệnh lặp không điều kiện trên (thực chất là có điều kiện CX khác 0), còn có các lệnh lặp có điều kiện tương tự lệnh J (điều kiện). Điều kiện đây là giá trị 0 của bộ đếm CX hay bit cờ ZF. Đó là các lệnh:

- *LOOPE/ LOOPZ* (LOOP if Equal/ LOOP if Zero - lặp lại nếu bằng/ lặp lại nếu là 0).

Hai lệnh này là tương đương nhau hay nói khác đi, khi sử dụng, chỉ cần viết một trong hai dạng trên. Đây là lệnh lặp lại có điều kiện và điều kiện là điều kiện kép (phải thoả mãn đồng thời cả hai điều kiện) với:

+ CX ≠ 0

+ Cờ ZF = 0 (hiệu hai số ≠ 0 hay hai số không bằng nhau.).

*Cờ:* Không bị tác động.

- LOOPNE/LOOPNZ (Loop if Not Equal/Loop if Not Zero - lặp nếu không bằng/lặp nếu không bằng 0).

Hai lệnh này cũng tương đương nhau, cũng giống cặp lệnh trên về thao tác nhưng điều kiện ngược lại với cặp lệnh trên.

39. **MOV** (MOVE - chuyển).

*Cú pháp:* MOV Đích, Gốc.

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, có cùng một độ dài (byte, từ) và không được phép đồng thời là hai ô nhớ, hai thanh ghi đoạn và hai số tức thì.

*Thao tác:* Chuyển nội dung của Gốc sang cho Đích, còn nội dung của Gốc được giữ nguyên.

*Cờ:* Không bị tác động.

- MOVS/MOVSB/MOVSW (MOVe String - chuyển chuỗi/chuyển chuỗi byte/chuyển chuỗi từ).

Các lệnh này có thể coi là biến thể của lệnh MOV nhưng áp dụng cho chuỗi. Chuỗi ở đây nằm trong các toán hạng Gốc là ô nhớ có địa chỉ DS:SI, còn toán hạng Đích có địa chỉ là giá trị của DS:DI và:

- Nếu DF = 0, các địa chỉ Đích, Gốc tự tăng (tăng 1 hoặc 2 tùy thuộc lệnh theo byte hay từ).
- Nếu DF = 1, các địa chỉ Đích, Gốc tự giảm (giảm 1 hoặc 2 tùy thuộc lệnh theo byte hay từ).

*Cờ:* Không bị tác động.

40. **MUL** (MULtiply - nhân).

*Cú pháp:* MUL Gốc (Đích ẩn là AX).

Toán hạng Gốc theo các chế độ địa chỉ khác nhau.

*Thao tác:* Thực hiện phép nhân không dấu của số bị nhân trong AX với số nhân ở trong Gốc, kết quả gửi vào Đích là AX (nhân với byte) hay DX: AX (nhân với từ).

AX ← AL x byte hay DX:AX ← AX x Từ.

*Cờ:* CF, OF bị tác động và các bit khác không bị tác động.

41. **NEG** (NEGate - đảo dấu hay lấy bù 2).

*Cú pháp:* NEG Đích (Gốc ẩn là AL hay AX).

Đích tìm được theo chế độ địa chỉ khác nhau.

*Thao tác :* Đảo dấu của một số hay Đích 0-(Đích).

Lệnh trên hoàn toàn tương đương với hành động lấy bù hai tức tính Đích+1. Việc tính này chia thành hai động tác:

- Tính $\overline{\text{Đích}}$ tức đảo bit của Đích ($\overline{\text{bit 0}}$ → bit 1 và $\overline{\text{bit 1}}$ → bit 0).

- Cộng kết quả của phép đảo bit với 1.

*Chú ý:* Nếu lấy bù 2 của 128 hoặc 32768 thì ta được một kết quả không đổi nhưng cờ OF = 1 để báo là kết quả bị tràn (số lớn nhất biểu diễn được là 128 và 32767).

*Cờ:* Các bit cờ đều bị tác động.

42. **NOP** (NOPeration - không hành động).

*Cú pháp:* NOP (không toán hạng).

*Thao tác:* Vi xử lý không hành động gì chỉ tăng nội dung của thanh ghi lệnh IP (để đọc lệnh tiếp theo và tiêu tốn 3 chu kỳ đồng hồ. Nó thường được dùng cho một trong hai mục đích sau:

- Tăng thời gian trễ cho một vòng lặp đợi thời gian trễ.

- Dành chỗ cho lệnh mới cần bổ sung vào chương trình mà không muốn làm ảnh hưởng tới độ dài của chương trình.

*Cờ:* Không bị tác động.

43. **NOT** (NOT - không).

*Cú pháp:* NOT Đích ( không có toán hạng Gốc).

Đích được tìm theo các chế độ địa chỉ khác nhau.

*Thao tác:* Tạo dạng bù 1 tức đảo bit ($\overline{\text{bit 0}}$ → bit 1 và $\overline{\text{bit 1}}$ → bit 0) của Đích rồi gửi lại vào Đích. Đích ← $\overline{\text{Đích}}$.

Cờ : Không bị tác động.

44. **OR** (OR - hoặc).

*Cú pháp:* OR Đích, Gốc.

Toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, chứa các số liệu cùng độ dài (byte, từ) và không được phép là thanh ghi đoạn hay hai ô nhớ đồng thời.

*Thao tác:* Thực hiện phép cộng logic từng bit một của hai số ở Đích và Gốc rồi gửi vào Đích (0+0 = 0, 1+0 = 1, 0+1 = 1, 1+1 = 1).

Đích ← Đích + Gốc hay Đích ← Đích + Gốc.

Lệnh OR này thường dùng để xác lập một số bit nào đó của toán hạng bằng cách cộng logic toán hạng đó với toán hạng tức thời có các bit 1 tại các vị trí tương ứng cần thiết lập.

*Cờ :* Các bit CF, OF, PF, ZF bị tác động; AF không xác định.

45. **OUT** (OUT - đưa ra).

*Cú pháp:* OUT PORT, AL (hoặc AX)
OUT DX, AL (hoặc AX).

PORT là nhãn của một cổng, có thể là một con số tức thì chỉ địa chỉ cổng (8 bit) hoặc DX (16 bit) chứa giá trị của địa chỉ của một cổng (tương tự cho trường hợp lệnh IN).

*Thao tác:* Đưa số liệu từ thanh ghi tích lũy AX (hoặc AL) ra cổng có địa chỉ là PORT (số hiệu cổng cố định) hay ở DX (số hiệu cổng thay đổi cần nạp vào trước khi có lệnh OUT. Hành động này ngược với hành động của lệnh đưa số liệu vào (INAL PORT hay INAL, DX).

*Cờ:* Không bị tác động.

46. **POP** (POP - lấy ra).

*Cú pháp*: POP Đích

Toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau trừ thanh ghi đoạn mã lệnh CS. Toán hạng Gốc ẩn là đỉnh của ngăn xếp.

*Thao tác:* Dữ liệu tại đỉnh của ngăn xếp (địa chỉ là SP) được lấy ra hay trả lại Đích đã gửi vào do lệnh PUSH. Dữ liệu tại đỉnh của ngăn xếp không thay đổi còn địa chỉ của ngăn xếp (SP) tự tăng lên 2. Đích ← {SP} và SP ← SP+2.

*Cờ:* Không bị tác động.

47. **POPF** (POP Flag - lấy Flag ra).

*Cú pháp:* POPF (toán hạng ẩn với Gốc là đỉnh ngăn xếp có địa chỉ SP và Đích là thanh ghi Flag FL).

*Thao tác:* Dữ liệu tại đỉnh ngăn xếp (địa chỉ SP) được lấy ra và ghi vào Flag. Dữ liệu trên chính là giá trị cũ của Flag mà ta đã gửi vào bởi lệnh PUSHF. FL ← {SP} và SP ← SP+2.

Cũng giống lệnh trên, dữ liệu của {SP} và SS không thay đổi.

*Cờ:* Các bit cờ bị tác động.

48. **PUSH** (PUSH - đẩy vào).

*Cú pháp:* PUSH Gốc (Đích ẩn là ngăn xếp).

Gốc được tính theo các phương pháp địa chỉ khác nhau.

*Thao tác:* Địa chỉ của đỉnh của ngăn xếp giảm đi 2 rồi dữ liệu từ Gốc (các thanh ghi thông dụng, thanh ghi đoạn hoặc ô nhớ) được gửi vào đỉnh ngăn xếp để lưu trữ tạm thời (trước khi nhảy tới chương trình con). Dữ liệu trên sẽ được lấy ra bởi lệnh POP Đích sau khi kết thúc chương trình con để trở về chương trình chính. SP ← SP-2 (địa chỉ ngăn xếp từ đỉnh bị giảm đi 2) và {SP} ← Gốc .

*Cờ:* Không bị tác động.

49. **PUSHF** (PUSH Flag - chuyển thanh ghi cờ Flag).

*Cú pháp:* PUSHF (toán hạng Đích ẩn là ngăn xếp có địa chỉ là SP, toán hạng Gốc ẩn là Flag).

*Thao tác:* Địa chỉ của ngăn xếp (SP) giảm đi 2. Thanh ghi cờ (Flag) được cất vào đỉnh ngăn xếp. Dữ liệu của ngăn xếp và thanh ghi SS không thay đổi.

SP ← SP-2 và {SP} ← FL.

Lệnh này được sử dụng trước và có hành động ngược với lệnh POPF, dùng để lưu trữ tạm thời Flag trước khi nhảy tới chương trình con.

50. **RCL** (Rotate through Carry to the Left - quay qua carry về trái).

*Cú pháp:* RCL Đích, 1 hay RCL Đích, CL.

Toán hạng Đích có thể tìm theo các địa chỉ khác nhau.

Toán hạng nguồn ẩn là thanh ghi AL (8 bit) hoặc AX (16 bit).

Ở đây 1 hay CL (tối đa là 32) là số lần thực hiện lệnh quay trái.

*Thao tác:* Dịch chuyển theo một vòng tròn kín gồm các bit của AL (AX) và bit cờ CF (ở tận cùng bên trái) rồi gửi kết quả vào Đích. Nếu số lần quay là:

- 1 thì:
  + Các bit của AX dịch về phía trái 1 vị trí (bit $d_{n-2}$ thay chỗ cho bit $d_{n-1}$) .
  + Giá trị mới của CF = giá trị của bit có vịt trí cao nhất (bit tận cùng bên trái) của AX.
  + Giá trị mới của bit thấp nhất của AX = giá trị cũ của CF.
- CL thì:
  + Các bit của AX dịch về phía trái CL vị trí.
  + Giá trị cũ của CF nằm ở bit thứ CL của AX kể từ phải sang trái.
  + Nếu CL = 9 và lệnh quay cho AL (8 bit) thì kết quả là AL vẫn giữ nguyên giá trị cũ.

*Cờ:* Chỉ có CF và OF bị tác động.

Lệnh này được dùng để kiểm tra một bit nào đó của một thanh ghi mà lại muốn duy trì giá trị cũ của nó bằng cách dịch sang CF rồi dùng lệnh nhảy hay gọi chương trình con.

51. **RCR** (Rotate through Carry to the Right - quay phải thông qua cờ nhớ CF).

*Cú pháp:* RCR Đích, 1 hay RCR Đích, CL.

*Thao tác:* Lệnh này tương tự lệnh RCL nhưng chỉ khác là quay theo chiều bên phải và bit CF giờ nằm ở tận cùng bên phải. Do đó, nếu thực hiện lệnh RCR Đích,1 thì:

- Các bit của AX dịch về bên phải 1 vị trí (bit $d_{n-1}$ thay chỗ cho bit thứ $d_{n-2}$).
- Bit có vị trí cao nhất được thay bằng giá trị của CF.
- Bit CF mới được thay bằng giá trị của bit $d_0$ trước khi quay.

*Cờ:* Chỉ có CF và OF bị tác động.

Lệnh này được dùng để kiểm tra một bit nào đó của một thanh ghi mà lại muốn duy trì giá trị cũ của nó bằng cách dịch sang CF rồi dùng lệnh nhảy hay gọi chương trình con.

52. **REP** (REPeat - lặp lại).

Đây chỉ là tiếp đầu ngữ cho các lệnh thao tác chuỗi (String) có **đuôi S.** Do đó theo sau tiếp đầu ngữ **REP** này là tên lệnh chuỗi (ví dụ MOVS) và lệnh sẽ thực hiện lặp lại CX lần (được nạp vào trước khi viết REP lệnh chuỗi). Sau mỗi lần thực hiện lệnh, CX tự động giảm đi 1 và khi CX = 0 việc thực hiện lệnh bị ngừng. So với lệnh LOOP thực hiện nhiều lệnh trong vòng lặp thì lệnh lặp với tiếp đầu ngữ REP chỉ lặp lại mỗi một lệnh duy nhất đứng liền sau tiếp đầu ngữ. Cách viết này đơn giản, tránh phải viết lại CX lần chc một lệnh.

Cũng có tiếp đầu ngữ lặp lại lệnh (tương tự lệnh LOOP) có điều kiện. Đó là:

- REPE/REPZ (lặp lại nếu bằng/ lặp lại nếu zero, tương tự LOOPE/LOOPZ): lặp lại nếu CX ≠ 0 và ZF ≠ 1 tức quá trình lặp lại bị ngừng khi CX = 0 hoặc ZF = 0). Lệnh này thường dùng để phát hiện hai chuỗi khác nhau (ZF = 0) trong tập hợp các chuỗi nói chung giống nhau bằng việc so sánh (quét) các phần tử tương ứng của hai chuỗi trong hai tập hợp chuỗi trên. Việc so sánh bị ngừng khi quét hết chuỗi (CX = 0) hay khi phát hiện có một phần tử của cặp chuỗi khác nhau (ZF = 0).

- REPNE/REPNZ (lặp lại nếu không bằng/lặp lại nếu không bằng zero, tương tự LOOPNE/LOOPNZ): lặp lại nếu CX ≠ 0 hoặc ZF = 0. Lệnh lặp này có điều kiện trái với điều kiện của lệnh REPE/REPZ. Lệnh này được sử dụng để tìm chuỗi có các phần tử giống nhau bằng cách lặp lại việc so sánh hai chuỗi trong hai tập hợp nói chung khác nhau vì quá trình quét chuỗi bị ngừng khi đếm hết (CX = 0) hoặc khi đã tìm thấy hai chuỗi cần tìm có phần tử giống nhau (ZF = 1).

53. **RET** (RETurn - trở về).

*Cú pháp:* RET (không có toán hạng).

Gốc ẩn là ngăn xếp với địa chỉ là SP, còn Đích ẩn là các thanh ghi CS: IP (trở về chương trình chính từ chương trình con ở xa vì cả hai giá trị CS, IP đều thay đổi) hay IP (trở về từ chương trình con ở gần vì cùng một đoạn CS). Xin xem lệnh CALL (gọi chương trình con).

*Thao tác:* Lệnh này luôn nằm ở cuối chương trình con (thủ tục) để chuẩn bị cho vi xử lý trở về chỗ bị ngắt ở chương trình chính. Muốn vậy:

- Các thanh ghi mã lệnh của vi xử lý (CS:IP) phải được nạp trả lại giá trị của chúng (bởi lệnh POP) khi bị ngắt chương trình phải gửi tạm vào ngăn xếp (bởi lệnh PUSH).

- Thanh ghi cờ Flag phải được nạp trả lại giá trị của chúng (bởi lệnh POPF) khi bị ngắt chương trình phải gửi tạm vào ngăn xếp (bởi lệnh PUSHF).

54. **RET n** (RETurn n - trở về n).

*Cú pháp:* RET n (n là một số nguyên dương).

*Thao tác:* Trở về với n. Lệnh này tương tự lệnh RET, cũng là sự trở về từ chương trình con, nhưng:

- Sử dụng ở cuối một chương trình con phục vụ ngắt khi có lệnh ngắt mềm INT n của hệ điều hành MS-DOS.

- Sau khi hồi phục CS:IP thì SP ← SP + n tức nhảy qua n địa chỉ mà không lấy lại các thông số khác còn lại trong ngăn xếp.

55. **ROL** (Rotate to the Left - quay về bên trái).

*Cú pháp :* ROL Đích, CL (Gốc ẩn là AX)

Toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau.

CL là số lần quay, được nạp vào trước khi viết lệnh ROL Đích, CL và có thể CL = 1 nên viết trực tiếp số 1 sau dấu phảy.

*Thao tác:* Quay hay dịch về bên trái theo một vòng kín các bit của thanh ghi tích lũy AX đi CL lần rồi gửi vào Đích. Mọi hành động của lệnh này giống lệnh RCL nhưng không quay qua bit cờ CF (xem lệnh RCL).

Lệnh này được dùng để:

- Nhân một số nhị phân với $2^n$ trong đó n = CL tức số lần dịch hay quay.

- Kiểm tra một bit nào đó của một thanh ghi mà lại muốn duy trì giá trị cũ của nó.

*Cờ :* Chỉ có CF và OF bị tác động.

56. **ROR** (Rotate to the Right - quay về bên phải).

*Cú pháp:* ROR Đích, CL (Gốc ẩn là AX).

Toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau.

CL là số lần quay, được nạp vào trước khi viết lệnh ROR Đích, CL và có thể CL = 1 nên viết trực tiếp số 1 sau dấu phẩy.

*Thao tác:* Quay hay dịch về bên phải theo một vòng kín các bit của thanh ghi tích lũy AX đi CL lần rồi gửi vào Đích. Mọi hành động của lệnh này giống lệnh RCR nhưng không quay qua bit cờ CF (xem lệnh RCR).

Lệnh này được dùng để:

- Chia một số nhị phân cho $2^n$ trong đó n = CL tức số lần dịch hay quay.
- Kiểm tra một bit nào đó của một thanh ghi mà lại muốn duy trì giá trị cũ của nó.

*Cờ:* Chỉ có CF và OF bị tác động.

57. **SAHF** (Store AH into Flag- tích AH vào byte thấp của Flag).

*Cú pháp:* SAHF (không có toán hạng).

Toán hạng Gốc ẩn là AH, còn toán hạng Đích ẩn là byte thấp của FL.

*Thao tác:* Cất thanh ghi AH vào Flag. Lệnh này có hành động trái ngược với lệnh LAHF dùng để trả lại giá trị cho FL sau khi đã gửi vào AH để kiểm tra bởi lệnh LAHF.

*Cờ:* Tất cả các bit bị tác động.

58. **SAL/SHL** (Shift Arithmetic to the Left/Shift logic to the Left - dịch số học sang bên trái/dịch logic sang bên trái).

*Cú pháp:* SAL Đích, CL
SHL Đích, CL.

Gốc ẩn là AX. Toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau. CL là số lần dịch được nạp vào từ trước khi viết lệnh, có thể viết 1 thay cho CL = 1.

Hai lệnh trên là tương đương vì không có lệnh riêng rẽ SAL như SAR dưới đây.

*Thao tác:* Dịch trái số học hay dịch trái logic làm các bit của AX dịch sang trái CL lần với đặc điểm sau:

- Bit tận cùng bên trái là bit cờ nhớ CF.
- Bit tận cùng bên phải luôn luôn được nạp vào giá trị 0.

Lệnh này có tác dụng tương tự lệnh ROL (cũng dịch sang trái qua CF) nhưng:

+ Không có sự quay vòng giữa bit CF và bit cuối cùng thấp nhất mà bit thấp nhất luôn luôn được nạp số 0 sau mỗi lần dịch.

+ Các bit thấp của AX dần dần bị xóa về 0 sau mỗi lần dịch. Nói khác đi, việc dịch trái này luôn luôn kèm động tác xóa bit thấp đã bị dịch, nội dung thanh ghi bị xóa sau số lần dịch bằng số bit của nó.

Lệnh này có hành động ngược lại với lệnh SHR.

*Cờ:* Tất cả các bit bị tác động chỉ có AF không xác định.

59. **SAR** (Shift Arithmetically Right - dịch phải số học).

*Cú pháp:* SAR Đích, CL

Gốc ẩn là AX. Toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau. CL là số lần dịch được nạp vào từ trước khi viết lệnh, có thể viết 1 thay cho CL = 1.

*Thao tác:* Dịch phải số học làm các bit của AX dịch sang phải CL lần với đặc điểm sau:

- Bit tận cùng bên phải là bit cờ nhớ CF.
- Bit tận cùng bên trái luôn luôn được nạp lại giá trị của chính nó (quay vòng chính nó).

Lệnh này có tác dụng tương tự lệnh ROR (cũng dịch sang phải qua CF) nhưng:

- Không có sự quay vòng giữa bit CF và bit ở vị trí cao nhất .
- Bit vị trí cao nhất luôn luôn được nạp giá trị của chính nó sau mỗi lần dịch.

Lệnh này được sử dụng để kiểm tra từng bit (sang CF nhưng vẫn duy trì bit cao nhất. Nếu bit cao chỉ dấu của số nguyên thì lệnh này luôn duy trì dấu - hoặc +).

*Cờ:* Tất cả các bit bị tác động chỉ có AF không xác định.

60. **SBB** (Substract with Borrow - trừ có mượn).

*Cú pháp:* SBB Đích, Gốc.

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa cùng số bit và không đồng thời là hai ô nhớ hay thanh ghi đoạn.

*Thao tác:* Trừ Đích cho Gốc và cờ nhớ rồi gửi kết quả vào Đích.

Đích Đích - Gốc CF.

Nếu CF = 0 thì lệnh này trùng với lệnh SUB Đích, Gốc.

*Cờ:* Tất cả các bit bị tác động chỉ có AF không xác định.

61. **SCAS/SCAB/SCAW** (Scan a String/ Scan a String byte/ Scan a String word - quét chuỗi/ quét chuỗi byte/ quét chuỗi từ).

*Cú pháp:* SCAS chuỗi Đích (chuỗi Gốc ẩn là AX).
SCASB chuỗi Đích byte
SCASW chuỗi Đích từ.

Lệnh này cũng có đặc điểm của lệnh chuỗi (về địa chỉ của chuỗi trong bộ nhớ về DF, SI, DI, DS, ES) như MOVS, CMPS v.v...

Lệnh này được dùng để so sánh hay tìm chuỗi hay xâu mong muốn trong một vùng nhớ, còn chuỗi cần tìm được nạp vào AX.

62. **SHR** (Shift Right logically - dịch phải logic).

*Cú pháp:* SHR Đích, CL.

Gốc ẩn là AX còn toán hạng Đích được tính theo các phương pháp địa chỉ khác nhau. CL là số lần dịch được nạp vào từ trước khi viết lệnh, có thể viết 1 thay cho CL = 1.

*Thao tác:* Dịch phải logic làm các bit của AX dịch sang phải CL lần với đặc điểm sau:

- Bit tận cùng bên phải là bit cờ nhớ CF.
- Bit tận cùng bên trái luôn luôn được nạp giá trị 0.

Lệnh này có tác dụng tương tự lệnh ROR (cũng dịch sang phải qua CF) nhưng:

- Không có sự quay vòng giữa bit CF và bit ở vị trí cao nhất .
- Bit vị trí cao nhất luôn luôn được nạp giá trị 0 sau mỗi lần dịch.
- Các bit cao của AX dần dần bị xóa về 0 sau mỗi lần dịch. Nói khác đi, việc dịch phải này luôn luôn kèm động tác xóa bit cao đã bị dịch, nội dung thanh ghi bị xóa sau số lần dịch bằng số bit của nó.

Lệnh này có hành động ngược lại với lệnh SAL/SHL.

Lệnh này được sử dụng để kiểm tra từng bit (sang CF nhưng xóa dần các bit cao nhất.

*Cờ:* Tất cả các bit bị tác động chỉ có AF không xác định.

63. **STC** (SeT Carry flag - xác lập Carry).

*Cú pháp:* STC (không có toán hạng, toán hạng ẩn là bit CF của flag).

*Thao tác :* Xác lập bit CF của flag lên 1. CF 1.

Lệnh này hành động trái ngược với lệnh CLC. Thường được sử dụng để cưỡng bức CF lên 1 rồi nhảy chương trình có điều kiện CF = 1.

*Cờ:* Không tác động tới cờ khác ngoài CF.

64. **STD** (SeT Direction Flag - xác lập bit cờ hướng DF).

*Cú pháp:* STD (không có toán hạng, toán hạng ẩn là bit DF của flag).

*Thao tác :* Xác lập bit DF của flag lên 1,DF ← 1.

Lệnh này hành động trái ngược với lệnh CLD. Thường được sử dụng để SI và DI tự động giảm đi 1 đơn vị (cho chuỗi theo byte) hay 2 đơn vị (cho chuỗi theo từ).

*Cờ:* Không tác động tới cờ khác ngoài DF.

65. **STI** (SeT Interrupt Flag - xác lập bit cờ cho phép ngắt IF).

*Cú pháp:* STI (không có toán hạng, toán hạng ẩn là bit IF của flag).

*Thao tác :* Xác lập bit IF của flag lên 1, IF ← 1.

Lệnh này hành động trái ngược với lệnh CLI. Thường được sử dụng để cho phép *tín hiệu yêu cầu ngắt* (đưa vào chân tín hiệu INTR của vi xử lý) *gây ra ngắt chương trình đang chạy.*

*Cờ:* Không tác động tới cờ khác ngoài IF.

66. **STOS/STOSB/STOSW** (Store AL/AX in String Byte/Word - cất giữ AL/AX vào chuỗi byte/từ).

*Cú pháp*: STOS chuỗi Đích (toán hạng Gốc ẩn là AL/AX)
STOSB chuỗi Đích (toán hạng Gốc ẩn là AL/AX)
STOSW chuỗi Đích (toán hạng Gốc ẩn là AL/AX)

*Thao tác*: Cất giữ AL (cho lệnh STOSB) hay AX (cho lệnh STOSW)

Sau khi cất giữ địa chỉ chuỗi sẽ tự động tăng (đã ghi DF = 0) hay tự động giảm (đã ghi DF = 1) đi 1 (lệnh STOSB) hay 2 (lệnh STOSW).

*Cờ:* Không tác động tới cờ.

67. **SUB** (SUBstract - trừ).

*Cú pháp:* SUB Đích, Gốc

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa cùng số bit và không đồng thời là hai ô nhớ hay thanh ghi đoạn.

*Thao tác:* Trừ Đích cho Gốc rồi gửi kết quả vào Đích.

Đích←Đích - Gốc .

Lệnh SUB Đích, Gốc này tương tự lệnh SBB Đích, Gốc khi CF = 0

*Cờ:* Tất cả các bit bị tác động.

68. **TEST** (TEST - kiểm tra, thử).

*Cú pháp:* TEST Đích, Gốc

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa cùng số bit và không đồng thời là hai ô nhớ hay thanh ghi đoạn.

*Thao tác:* Nhân logic Đích với Gốc. Kết quả không được lưu trữ vào thanh ghi nhưng thể hiện qua các bit cờ, Đích ∧ Gốc.

Lệnh này thường dùng để kiểm tra hay thử giá trị một bit (bit Gốc tương ứng là 1) hay làm mặt nạ che bit (bit Gốc tương ứng là 0). Thường dùng lệnh này thay cho lệnh so sánh CMP Đích, Gốc để kiểm tra bit trạng thái sẵn sàng của thiết bị ngoài để vi xử lý trao đổi tin với thiết bị ngoài khi thiết bị ngoài có yêu cầu hay đã sẵn sàng trao đổi tin (kết quả kiểm tra bit trạng thái = 1).

*Cờ:* CF, OF bị xóa; AF không xác định; các bit còn lại bị tác động.

69. **WAIT** (WAIT - chờ).

*Cú pháp:* WAIT (không có toán hạng).

*Thao tác:* Bắt vi xử lý chờ:

- Tín hiệu của đồng vi xử lý 80 x 87 đưa vào chân TEST của vi xử lý để kiểm tra xem đồng xử lý đã thực hiện xong lệnh chưa.

- Tín hiệu yêu cầu ngắt của thiết bị ngoài đưa vào chân INTR của vi xử lý để gây ra ngắt cứng.

Lệnh WAIT dùng để đồng bộ hoạt động của vi xử lý với môi trường bên ngoài như 80 x87 hay thiết bị ngoài. Lệnh này còn dùng để dừng việc thực hiện chương trình tại vị trí mong muốn bằng cách chèn lệnh trên vào vị trí đó.

*Cờ:* Không tác động đến bit cờ nào.

70. **XCHG** (eXCHanG - trao đổi).

*Cú pháp:*XCHG Đích, Gốc.

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa cùng số bit và không đồng thời là hai ô nhớ hay thanh ghi đoạn.

*Thao tác:* Trao đổi dữ liệu giữa Đích và Gốc, có nghĩa là:

- Dữ liệu cũ của Gốc chuyển cho Đích.

- Dữ liệu cũ của Đích chuyển cho Gốc.

Lệnh này có điểm giống lệnh MOV Đích, Gốc là dữ liệu từ Gốc chuyển cho Đích, nhưng có điểm khác là dữ liệu cũ bị mất, trong khi lệnh XCHG Đích, Gốc còn có thêm hành động chuyển dữ liệu cũ của Đích chuyển cho Gốc. Nếu chỉ dùng lệnh MOV để thực hiện hành động của lệnh XCHG, ta phải dùng thêm một thanh ghi trung gian nào đó và phải tới 3 lệnh MOV như sau:

```
MOV Thanh ghi trung gian, Đích    ; Thanh ghi trung gian ← Đích
MOV Đích , Gốc                    ; Đích ← Gốc
MOV Gốc, Thanh ghi trung gian     ; Gốc ← Thanh ghi trung gian
```

Lệnh XCHG này thường dùng để chuyển dữ liệu cho xử lý nhưng vẫn lưu giữ được dữ liệu cũ và nếu cần sẽ dùng lệnh XCHG một lần nữa thì dữ liệu cũ không xử lý lại trở về vị trí cũ.

*Cờ:* Không tác động tới cờ.

71. **XLAT** (TransLATe - chuyển đổi bảng).

*Cú pháp:* XLAT không có toán hạng, toán hạng Gốc ẩn là AL và BX còn toán hạng Đích ẩn là AL.

*Thao tác:* Chuyển nội dung của ô nhớ có địa chỉ là BX + AL sang AL.

$$AL \leftarrow \{AL + BX\}$$

Lệnh này được sử dụng để biến đổi mã từ một bảng Gốc có địa chỉ lệch (offset) là BX và AL chứa chỉ số của các phần tử của bảng (từ 0 ÷ 255), nội dung mới của mã được đọc từ địa chỉ BX + AL của bảng vào AL để xử lý tiếp.

*Cờ :* Không bị tác động.

72. **XOR** (eXclusive OR - hoặc loại trừ).

*Cú pháp:* XOR Đích, Gốc

Các toán hạng Đích, Gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa cùng số bit và không đồng thời là hai ô nhớ hay thanh ghi đoạn.

*Thao tác*: Từng bit tương ứng của Đích và Gốc được thực hiện *phép tính logic hoặc và loại trừ* rồi gửi kết quả vào Đích. Đích←Đích ⊕ Gốc.

Lệnh này thường dùng để:

- Phát hiện hai bit khác nhau trong hai byte hay từ (kết quả = 1) và giống nhau (kết quả = 0).
- Xóa một thanh ghi Rg khi dùng lệnh XOR với chính nó: XOR Rg, Rg.

*Cờ:* Xóa CF, OF; AF không xác định ; còn các bit khác bị tác động.

## B. CÁC LỆNH BỔ SUNG CỦA VI XỬ LÝ INTEL 80186 (DẤU *) VÀ 80286

Các vi xử lý 80186 và 80286 ngoài các lệnh cơ bản của 8086/8088 còn có các lệnh bổ sung bảng dưới đây. Trong các lệnh bổ sung, ta thấy:

- Các lệnh cất giữ và lấy ra tất cả các thanh ghi: PUSHA, POPA.
- Các lệnh vào/ra chuỗi số liệu : INS, OUTS .
- Các lệnh liên quan tới ngôn ngữ bậc cao: BOUND, ENTER, LEAVE.
- Các lệnh thuộc nhóm ở chế độ bảo vệ.

| **Ký hiệu** | **Tên lệnh** (tiếng Anh) | **Tên lệnh** (tiếng Việt) |
|---|---|---|
| *1. Các lệnh dịch chuyển* | | |
| INS/INSB/INSW* | INput String/Byte/Word | Nhập vào chuỗi/Byte/từ |
| OUTS/OUTB/OUTW | OUT String/Byte/Word | Đưa ra chuỗi/Byte/từ |
| PUSHA | PUSH All | Cất giữ tất cả các thanh ghi |
| POPA | POP All | Lấy ra tất cả các thanh ghi |
| *2. Các lệnh ngôn ngữ bậc cao* | | |
| BOUND | BOUND | Kiểm tra biên của trường |
| ENTER | ENTER | Vào - Tạo khối các thông số để vào chương trình con |
| LEAVE | Leave | Rời - ra khỏi chương trình con. |
| *3. Nhóm lệnh ở chế độ bảo vệ* | | |
| ARPL | Adjust Request Privilege Level | Hiệu chỉnh mức đặc quyền yêu cầu |
| LAR | Load Acces Rights | Nạp quyền xâm nhập |
| LSL | Load Segment Limit | Nạp giới hạn mảng |
| LMSW | Load Machine Status Word | Nạp từ trạng thái máy |
| SMSW | Store Machine Status Word | Cất giữ từ trạng thái máy |

| | | |
|---|---|---|
| LGDT | Load Global Descriptor Table | Nạp thanh ghi bảng các bộ mô tả toàn cục |
| SGDT | Store Global Descriptor Table | Cất thanh ghi bảng các bộ mô tả toàn cục |
| LIDT | Load Interrupt Descriptor Table | Nạp thanh ghi bảng các bộ mô tả ngắt. |
| SIDT | Store Interrupt Descriptor Table | Cất thanh ghi bảng các bộ mô tả ngắt |
| LLDT | Load Local Descriptor Table | Nạp thanh ghi bảng các bộ mô tả cục bộ |
| SLDT | Store Local Descriptor Table | Cất thanh ghi bảng các bộ mô tả cục bộ |
| LTR | Load Task Register | Nạp thanh ghi nhiệm vụ |
| STR | Store Task Register | Cất thanh ghi nhiệm vụ |
| VERW | VERryfy Write | Kiểm tra mảng để ghi |
| VERR | VERryfy Read | Kiểm tra mảng để đọc. |

## C. CÁC LỆNH BỔ SUNG CỦA VI XỬ LÝ INTEL 80386

Ngoài các lệnh cơ bản của 8086/8088, 80286 vi xử lý 80386 có thêm các lệnh bổ sung như bảng ở dưới. Các lệnh bổ sung này có các đặc điểm sau:

- Có thêm nhiều lệnh xử lý bit.
- Có thêm một số lệnh hoạt động với hai từ (32 bit, đuôi D).

| **Ký hiệu** | **Tên lệnh** (tiếng Anh) | **Tên lệnh** (tiếng Việt) |
|---|---|---|
| 1. *Nhóm lệnh xử lý bit* | | |
| BSF | Bit Scan Foward | Quét bit về phía trước |
| BSR | Bit Scan | Quét bit về phía sau |
| BT | Bit Test | Kiểm tra bit |
| BTC | Bit Test and Complementary | Kiểm tra bit và đảo bit |
| BTR | Bit Test and Reset | Kiểm tra bit và xóa bit |
| BTS | Bit Test and Set | Kiểm tra bit xác lập bit |
| IBTS | Insert String Bit | Xen vào một chuỗi bit |
| XBTS | eXtra String bit | Trích ra một chuỗi bit |
| 2. *Nhóm lệnh chuyển* | | |
| LFS | LOAD F Segment | Nạp thanh ghi FS |
| LGS | LOAD G Segment | Nạp thanh ghi GS |
| LSS | LOAD S Segment | Nạp thanh ghi SS |
| STOSD | STOre String Double | Cất chuỗi kép |

| | | |
|---|---|---|
| LODSD | LOaD String Double | Nạp chuỗi kép |
| MOVSD | MOVe String Double | Chuyển chuỗi kép |
| MOVSX | MOVe with Signe- eXtend | Chuyển với mở rộng dấu |
| MOVZX | MOVe with Zero eXtend | Chuyển với mở rộng 0 |
| POPFD | POP Flag Double | Lấy ra cờ kép |
| PUSHFD | PUSH Flag Double | Cất giữ cờ kép |
| SCASD | Scan Double | Quét từ kép |
| SHLD | SHift Left Double | Dịch trái từ kép |
| SHRD | SHift Right Double | Dịch phải từ kép |
| INSD | Input String Double | Nhập vào chuỗi lời kép |
| OUTSD | OUTput String Double | Đưa ra chuỗi lời kép. |
| *3.Các lệnh khác* | | |
| CWDE | Convert Word to Extended Double | Biến đổi 1 từ thành 2 từ |
| CDQ | Convert Double to Quad | Biến đổi 2 thành 4 từ |
| CMPSD | COMPare String Double | So sánh chuỗi kép |
| SCASD | SCAN String Double | Quét chuỗi kép |
| SET cc | SET condidion (lớn, nhỏ, bằng, S, O, P) | Xác lập có điều kiện |
| IRETD | Interrupt RETurn Double | Trở về từ ngắt kép. |

## D. CÁC LỆNH BỔ SUNG CỦA VI XỬ LÝ INTEL 80486

Vi xử lý Intel 80486, chính là 80386 + 80387, ngoài các lệnh cơ bản của 8086/8088 và 80386 còn có các lệnh bổ sung như bảng sau:

| **Ký hiệu** | **Tên** (tiếng Anh) | **Tên** (tiếng Việt) |
|---|---|---|
| BSWAP | Byte SWAP | Hoán chuyển hai byte |
| CMPXCHG | CoMPare and eXcHanG | So sánh và hoán chuyển |
| INVD | INValiDe cache | (Làm vô hiệu bộ nhớ ngầm), kê khai những nội dung ghi của nhớ ngầm là không giá trị. |
| INVLPG | INValiDate TLB | (Vô hiệu hoá lối vào TLB). Kê khai những nội dung ghi trong nhớ đệm "Translation Lookaside" của bộ phận nhớ truy cập trực tiếp là không giá trị. |
| INVLPG | INValiDate TLB | (Vô hiệu hoá lối vào TLB). Kê khai những nội dung ghi trong nhớ đệm "Translation Lookaside" của bộ phận nhớ truy cập trực tiếp là không giá trị. |
| XADD | eXchange and ADD | Hoán chuyển và cộng. |
| WBINV | Write-back and INValiDate data cache | Ghi trở lại nội dung từ bộ nhớ trung tâm vào nhớ ngầm và làm mất giá trị của nhớ ngầm dữ liệu. |

## E. CÁC LỆNH BỔ SUNG CỦA VI XỬ LÝ INTEL 80586

Vi xử lý Pentium 80586 ngoài các lệnh cơ bản của các vi xử lý thuộc thế hệ trước, còn có các lệnh bổ sung như bảng sau:

| **Ký hiệu** | **Tên lệnh** (tiếng Anh) | **Tên lệnh** (tiếng Việt) |
|---|---|---|
| CMPXCHG8B | CoMPare and eXCHanG 8 Byte | So sánh và hoán chuyển 8 Byte |
| CPUID | CPU IDentification | Nhận dạng loại vi xử lý (EAX và các tin khác). |
| RSM | Resume from System Management Mode | Nhận sự thực hiện một chương trình bị ngắt bởi một hệ điều khiển ngắt hệ thống (SMI)và cũng dời chế độ điều khiển hệ thống. |
| RDMSR | ReaD Model Specific Register | Đọc thanh ghi riêng của mẫu (ví dụ thanh ghi kiểm tra v.v...) và tích giá trị của nó vào EDX:EAX. |
| WRMSR | Write Model Specific Register | Ghi thanh ghi riêng của mẫu (ví dụ thanh ghi kiểm tra v.v...) với giá trị trong EDX:EAX. |

# TÀI LIỆU THAM KHẢO


1. Phạm Trương Dần - Lập trình hợp ngữ Turbo Assembler 2.0 - Nhà xuất bản Giáo dục, Hà Nội, 1991.
2. Hoàng Văn Đặng - Assembler for Pascal - Nhà xuất bản Trẻ Thành phố Hồ Chí Minh, 1992
3. Trương Văn Thắng - Bài tập và bài giải hợp ngữ - Nhà xuất bản Đồng nai, 1995.
4. Nguyễn Lê Tín - Hỗ trợ kỹ thuật cho lập trình hệ thống - Nhà xuất bản Thành phố Hồ Chí Minh, 1992.
5. Nguyễn Quang Tiến, Vũ thanh Hiền - Lập trình với hợp ngữ - Nhà xuất bản Thống kê, 1996.
6. Đặng Văn Phú - Turbo Assembler và ứng dụng - Nhà xuất bản Khoa học và Kỹ thuật, 1996.
7. Trần Quang Vinh - Cấu trúc máy vi tính - Nhà xuất bản Giáo dục, 1997.
8. Văn Thế Minh - Kỹ thuật vi xử lý - Nhà xuất bản Giáo dục, 1997.
9. Nguyễn Mạnh Giang - Kỹ thuật ghép nối máy vi tính - Nhà xuất bản Giáo dục, 1997.
10. Alan R. Miller (người dịch: Nguyễn Minh San) - Lập trình Assembler cho DOS - Nhà xuất bản Giáo dục, 1993.
11. Peter Norton (người dịch: Nguyễn Minh San) - Nhập môn Assembler - Nhà xuất bản Giáo dục, 1995.
12. Julio Sanchez, Maria P. Canton (biên dịch: Nguyễn Hữu Lộc, Nguyễn Hữu An) - Sổ tay cho người lập trình PC - Nhà xuất bản Thống kê, 1997.
13. Peter Norton, Richard Wilton - Le guide de Peter Norton du programmeur sur PC & PS/2 - Microsoft Press, 1995.
14. Martin Althaus - PC/XT/AT/386, livre d or - Sybese, 1992.
15. Hans Peter Messmer - Pentium et compagnie - Addition Wesley France SA, 1994.
16. Yu cheng Liu, Glenn Geson - Microcomputer system, the 8086/8088 family - Prentice Hall, 1987.
17. Douglas V. Hall - Microprocessor and interfacing, programming and harward - Glencoe, 1992.
18. Walter A. Tribel - The 80386 DX microprocessor - Prentice Hall, 1991.

# MỤC LỤC

*Chịu trách nhiệm xuất bản :*

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI

Phó Tổng Giám đốc kiêm Tổng biên tập VŨ DƯƠNG THỤY

*Biên tập tái bản :*

DƯƠNG VĂN BẰNG

*Biên tập kỹ thuật :*

NGUYỄN LIÊN HƯƠNG

*Trình bày bìa :*

TRẦN THUÝ HẠNH

*Sửa bản in :*

HOÀNG TRỌNG NGHĨA

*Chế bản :*

ANH ĐỨC

---

**LẬP TRÌNH BẰNG NGÔN NGỮ ASSEMBLY CHO MÁY TÍNH PC-IBM**

**Mã số : 7B467T5 - DAI**

In 1.000 bản, khổ 19 x 27 cm tại Công ty cổ phần In Diên Hồng 187[B] Giảng Võ - Hà Nội. Số in : 188/P. Số XB : 1421/14 - 05. In xong và nộp lưu chiểu quý IV năm 2005.

CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO
Địa chỉ : 25 HànThuyên, Hà Nội

8 934980 542538

*Giá :* **41.500đ**