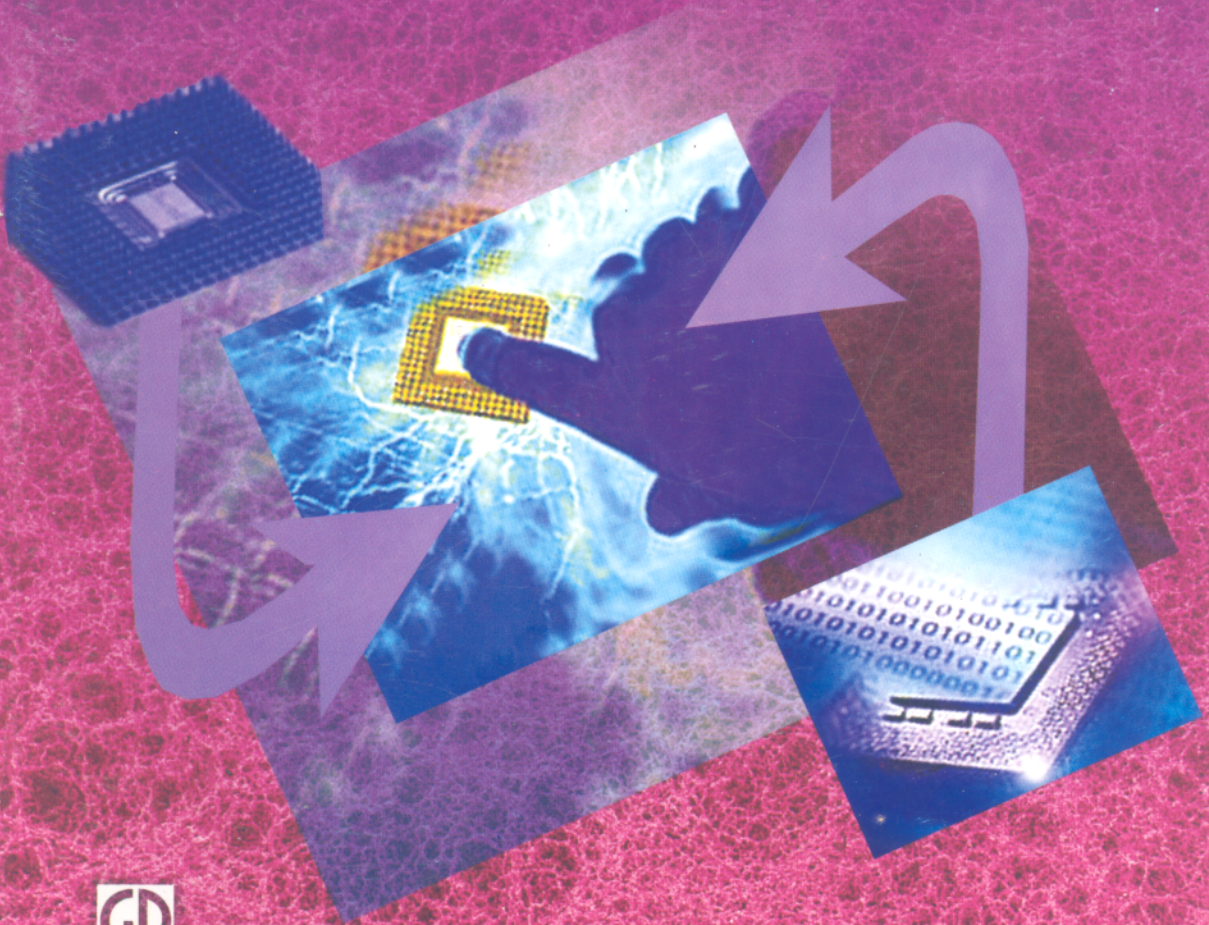
NGÔ DIÊN TẬP (Chủ biên)
PHẠM XUÂN KHÁNH - VŨ TRUNG KIÊN - KIỀU XUÂN THỰC

GIÁO TRÌNH
VI XỬ LÝ VÀ CẤU TRÚC MÁY TÍNH



NHÀ XUẤT BẢN GIÁO DỤC

NGÔ DIÊN TẬP (Chủ biên)
PHẠM XUÂN KHÁNH – VŨ TRUNG KIÊN
KIỀU XUÂN THỰC

GIÁO TRÌNH
VI XỬ LÝ VÀ CẤU TRÚC MÁY TÍNH

(Dùng cho sinh viên Cao đẳng kỹ thuật)

NHÀ XUẤT BẢN GIÁO DỤC

192 – 2007/CXB/3 – 411/GD

Mã số: 7B678M7 – DAI

Lời nói đầu

Ngày nay, nhu cầu học tập và nghiên cứu ứng dụng công nghệ Vi xử lý ngày càng tăng trưởng mạnh mẽ. Các công trình nghiên cứu và ứng dụng công nghệ Vi xử lý đều rất phong phú và đa dạng. Vì vậy, nội dung của giáo trình này cũng được bổ sung và cập nhật những kiến thức mới của công nghệ Vi xử lý cho phù hợp với nhu cầu của người học.

Giáo trình dùng cho sinh viên chuyên ngành Điện tử viễn thông và Tự động hoá. Nội dung của cuốn sách đề cập đến các khái niệm cơ bản về vi xử lý. Một số khái niệm mới, quan trọng đã được phát triển, nhưng các khái niệm cơ bản, đại cương hầu như vẫn được giữ nguyên và được trình bày một cách tinh tế hơn, cô đọng hơn để nội dung trở nên dễ hiểu, dễ tiếp thu, có tính sư phạm và nhấn mạnh những khái niệm quan trọng. Ngoài ra, giáo trình còn cung cấp một lượng ví dụ, bài tập đáng kể, trợ giúp sinh viên trong quá trình ôn tập tự kiểm tra kiến thức. Điều này là rất cần thiết vì chúng đóng vai trò quan trọng trong việc minh hoạ, giải thích các khái niệm cơ sở. Để nắm được nội dung cuốn sách cũng như giải được các bài tập cuối chương, độc giả cần nắm được các kiến thức cơ bản về toán cao cấp

Do thời gian biên soạn ngắn và thời lượng có hạn nên mặc dù có nhiều cố gắng, cuốn *Giáo trình Linh kiện điện tử* chắc chắn còn nhiều vấn đề cần bổ sung, hoàn thiện. Mong bạn đọc góp ý xây dựng, mọi ý kiến xin gửi về địa chỉ: Công ty Cổ phần Sách Đại học – Dạy nghề, 25 Hàn Thuyên, Hà Nội.

CÁC TÁC GIẢ

Chương 1. TỔNG QUAN VỀ VI XỬ LÝ VÀ MÁY TÍNH

1.1. CÁC HỆ ĐẾM VÀ VIỆC MÃ HOÁ THÔNG TIN

1.1.1. Các hệ đếm

1.1.1.1. Hệ thập phân

Hàng ngày, chúng ta thường dùng hệ thập phân (hệ đếm cơ số mười) để biểu diễn các giá trị số. Trong hệ này chúng ta dùng các số trong phạm vi từ 0 đến 9 để biểu diễn các giá trị. Hệ thập phân là một hệ đếm phụ thuộc vị trí, có nghĩa là mỗi chữ số gắn liền với một luỹ thừa 10 với số mũ phụ thuộc vào vị trí của con số đó trong số được biểu diễn. Ví dụ số 1234 sẽ bằng 1 nghìn, 2 trăm, 3 chục và 4 đơn vị:

$1234 = 1.10^3 + 2.10^2 + 3.10^1 + 4.10^0$

Nhưng trong máy tính các vi mạch chỉ có thể xử lý thông tin dưới dạng mã nhị phân nên chúng ta cần phải xem xét cách biểu diễn các số dưới dạng mã nhị phân như thế nào.

1.1.1.2. Hệ nhị phân

Trong hệ nhị phân (hệ đếm cơ số hai), cơ số đếm là 2 nên chỉ sử dụng 2 số là 0 và 1 để biểu diễn các giá trị số.

Ví dụ, chuỗi số nhị phân 101011 dùng để biểu diễn số:

$101011 = 1.2^5 + 0.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 43$ trong hệ thập phân.

Bảng 1.1 chỉ ra tương quan giữa số thập phân và số nhị phân:

Bảng 1.1. Tương quan giữa số thập phân và số nhị phân

Hệ thập phân	Hệ nhị phân	Hệ thập phân	Hệ nhị phân
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Mỗi số 0 và 1 được gọi là một bit (viết tắt của binary digit – số nhị phân); một số ở hệ nhị phân gồm các bit được kết thúc bởi chữ B để phân biệt với các hệ khác. Một cụm 4 bit gọi là một nibble, một cụm 8 bit gọi là một byte, cụm 16 bit gọi là một word, cụm 32 bit gọi là một double word...

Bit đầu tiên bên trái được gọi là bit có ý nghĩa lớn nhất (MSB: most significant bit), còn bit cuối cùng ở bên phải được gọi là bit có ý nghĩa nhỏ nhất (LSB: least significant bit).

1.1.1.3. Hệ thập lục phân

Ta thấy rằng một số biểu diễn ở hệ nhị phân thì rất dài và khó nhớ nên trong thực tế người ta thường sử dụng hệ thập lục phân. Hệ thập lục phân là hệ đếm cơ số 16 nên người ta sử dụng các số từ 0 đến 9 và các chữ cái từ A đến F để biểu diễn các số. Bảng sau chỉ ra tương quan giữa số hệ thập phân và hệ thập lục phân:

Bảng 1.2. Tương quan giữa hệ thập phân và hệ thập lục phân

Hệ thập phân	Hệ thập lục phân	Hệ thập phân	Hệ thập lục phân
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

1.1.2. Chuyển đổi giữa các hệ đếm

Vì con người quen sử dụng các số ở hệ thập phân, trong khi đó các bộ vi xử lý chỉ thao tác với các số ở hệ nhị phân nên để đảm bảo việc giao tiếp giữa người và máy thì phải có phép chuyển đổi giữa hai hệ này.

* *Chuyển từ hệ nhị phân sang hệ thập phân*

Để đổi một số từ hệ nhị phân sang hệ thập phân ta áp dụng công thức sau:

$$(b_n b_{n-1} \ldots b_1 b_0)\mathbf{B} = b_n \times 2^n + b_{n-1} \times 2^{n-1} + \ldots + b_1 \times 2^1 + b_0 \times 2^0$$

Ví dụ:

$100011B = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 35$

** Chuyển từ hệ thập phân sang hệ nhị phân*

Để chuyển từ hệ thập phân sang hệ nhị phân ta sử dụng phương pháp đơn giản sau: Lấy số cần chuyển chia cho 2 và ghi nhớ phần dư, tiếp theo lấy thương của phép chia trước đó chia cho 2 và ghi nhớ phần dư... cứ tiếp tục cho đến khi thương bằng 0. Kết quả của phép chuyển đổi chính là dãy các số dư lấy theo thứ tự đảo ngược.

Ví dụ: Đổi số 25 sang hệ nhị phân.

25	1	25 chia 2 được 12 dư 1
12	0	12 chia 2 được 6 dư 0
6	0	6 chia 2 được 3 dư 0
3	1	3 chia 2 được 1 dư 1
1	1	1 chia 2 được 0 dư 1
0		

Vậy kết quả là ***11001***.

Ví dụ: Đổi số 42 sang hệ nhị phân.

42	0	42 chia 2 được 21 dư 0
21	1	21 chia 2 được 10 dư 1
10	0	10 chia 2 được 5 dư 0
5	1	5 chia 2 được 2 dư 1
2	0	2 chia 2 được 1 dư 0
1	1	1 chia 2 được 0 dư 1
0		

Vậy kết quả là ***101010***.

1.1.3. Các phép toán số học với số hệ nhị phân

** Phép cộng*

Phép cộng với các số hệ nhị phân được thực hiện tương tự như với các số hệ thập phân theo quy tắc sau:

$0 + 0 = 0$

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 nhớ 1

Ví dụ:

0010	100011
+ 1011	+ 011111
= 1101	= 1000010

** Phép trừ*

Phép trừ với các số hệ nhị phân được thực hiện tương tự như với các số hệ thập phân.

Ví dụ:

1111	1000010
– 1011	– 01010
= 0100	= 111000

Trong máy tính người ta thực hiện phép trừ bằng phép cộng với số bù hai.

** Phép nhân*

Phép nhân hai số hệ nhị phân được thực hiện tương tự như nhân hai số hệ thập phân. Quy tắc thực hiện như sau:

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

Ví dụ:

```
      1001
  ×   1101
-----------
      1001
  +   0000
     1001
    1001
-----------
  = 1110101
```

1.1.4. Các mã thường dùng trong máy tính

** Mã BCD*

Mã BCD thường được sử dụng để mã hoá các chữ số từ 0 đến 9 trong vi xử lý, PLC...

Bảng 1.3. Mã BCD của các chữ số từ 0 đến 9

Hệ thập phân	Mã BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Ví dụ: mã BCD của 5 là 0101, của 9 là 1001.

Trong khi làm toán với các số BCD ta thường kết hợp hai số BCD tạo thành một byte, dạng biểu diễn này gọi là dạng BCD chuẩn hay BCD đóng gói (packed BCD). Ví dụ, mã BCD đóng gói (BCD chuẩn) của 59 là 0101 1001B, của 62 là 0110 0010B.

Số BCD không gói là số dài một byte trong đó bốn bit cao bằng 0 còn bốn bit thấp là số BCD chuẩn mã hoá số cần biểu diễn.

Ví dụ, mã BCD không gói của 5 là 0000 0101B, của 4 là 0000 0100B.

** Mã ASCII*

Mã ASCII (American Standard Code for Internationnal Interchange – Bộ mã chuẩn của Mỹ dùng cho trao đổi thông tin) là bộ mã rất thông dụng trong máy tính và truyền thông.

Trong bảng mã ASCII tiêu chuẩn người ta sử dụng 7 bit để mã hoá các ký tự thông dụng, như vậy bảng này sẽ có 128 ký tự ứng với mã từ 0 đến 127.

Trong bảng mã ASCII mở rộng, người ta bổ sung thêm 128 ký tự đặc biệt với mã từ 128 đến 255.

Ví dụ mã ASCII của 'A' là 65 (01000001), của 'a' là 97 (01100001).

1.2. LỊCH SỬ PHÁT TRIỂN CỦA MÁY TÍNH VÀ BỘ VI XỬ LÝ

** Thế hệ máy tính cơ khí*

Ý tưởng về một hệ thống tính toán đã có từ rất lâu, khoảng 500 năm trước Công nguyên, người Babylon đã chế tạo được máy tính đầu tiên có tên là Abacus. Năm 1643, Blaise Pascal chế tạo thành công một máy tính tạo bởi các bánh răng, trong đó số răng của bánh nọ gấp 10 lần số răng của bánh kia, nguyên lý này sau được sử dụng để chế tạo các đồng hồ đo quãng đường của motor, đồng hồ đo nước...

** Thế hệ máy tính cơ điện – điện tử*

Năm 1889, Herman Holerith phát minh ra card đục lỗ dùng để lưu trữ dữ liệu, sau đó ông chế tạo thành công máy tính cơ khí được điều khiển bởi một motor điện, nó có thể thực hiện được các phép đếm, sắp xếp và so sánh thông tin lưu trong card đục lỗ.

Năm 1942, nhà phát minh người Đức Konrad Zure chế tạo ra máy tính điện tử Z3 dùng cho không quân Đức. Năm 1943, Alan Turing phát minh ra hệ thống máy tính điện tử có tên là Collossus được thiết kế từ các đèn điện tử chân không, đây là một máy tính chuyên dụng thực hiện theo một chương trình cố định để giải mã các bí mật quân sự của Đức quốc xã.

Máy tính điện tử đa dụng đầu tiên – hệ máy tính khả trình – được phát triển bởi Đại học Pennsylvania có tên là ENIAC (Electronics Numerical Integrator And Calculator). Đây là một máy tính lớn chứa hơn 17000 đèn điện tử, nặng khoảng 30 tấn và có thể thực hiện được 100000 thao tác trong một giây. ENIAC được lập trình bằng cách nối lại mạch điện, công việc này được thực hiện bởi các công nhân và mất rất nhiều thời gian. Ngoài ra, việc bảo dưỡng cũng phải được thực hiện thường xuyên vì tuổi thọ của các đèn điện tử thấp.

** Thế hệ máy tính dùng vi xử lý*

Năm 1948, transistor được phát minh, đến năm 1958, Jack Kilby phát minh ra mạch tổ hợp – đây là cơ sở để phát triển các vi mạch số. Năm 1971, Marcian T.Hoff một kỹ sư của Intel đã thiết kế ra bộ vi xử lý 4004 – mở đầu cho thời kỳ sử dụng vi xử lý trong máy tính. 4004 là bộ vi xử lý 4 bit, bên trong nó gồm 2300 transistor, có thể quản lý được bộ nhớ có 4096 (≈4K) ô nhớ, mỗi ô gồm 4 bit, tập lệnh của 4004 gồm 45 lệnh khác nhau, nó được

chế tạo theo công nghệ MOSFET kênh P có tốc độ xử lý là 50KIPS (Kilo Instruction Per Second – nghìn lệnh/giây). 4004 được dùng để thiết kế các hệ thống video game, hệ thống điều khiển nhỏ dùng vi xử lý. Trên cơ sở 4004, hãng Intel sản xuất bộ vi xử lý 4040, đây cũng là bộ vi xử lý 4 bit nhưng có tốc độ cao hơn 4004.

Sau năm 1971, Intel sản xuất bộ vi xử lý 8 bit tên là 8008, nó có thể quản lý được 16KB bộ nhớ, tập lệnh gồm 48 lệnh khác nhau. 8008 được Computer Terminal Corporation sử dụng trong các thiết bị đầu cuối Datapoint 2200. Năm 1973, hãng Intel giới thiệu bộ vi xử lý 8 bit hiện đại đầu tiên có tên là 8080, tốc độ xử lý là 500KIPS, quản lý được 64KB bộ nhớ. Năm 1977, Intel phát triển bộ vi xử lý 8085 (tương thích với Z80 của hãng Zilog) là bộ vi xử lý 8 bit đa dụng nhanh hơn 8080.

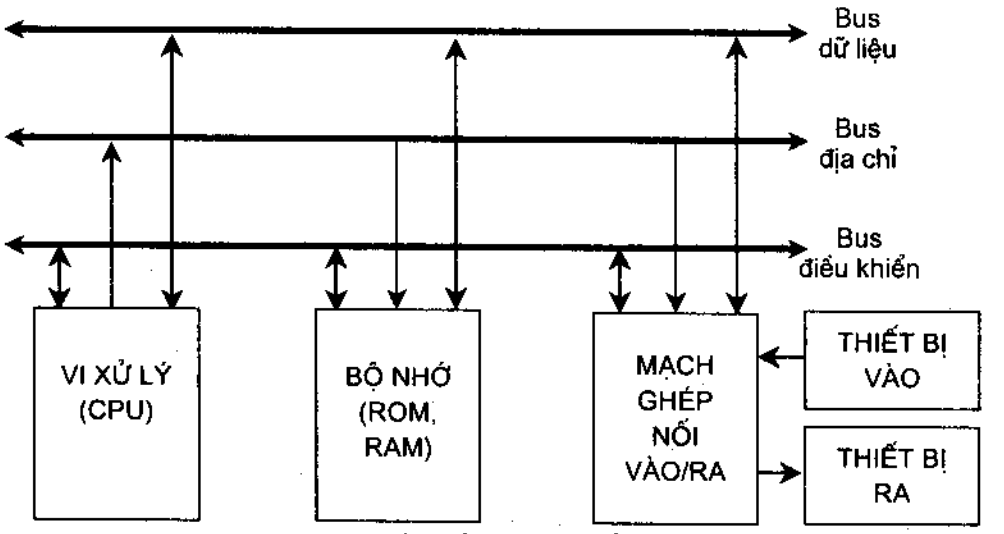
Năm 1978, Intel giới thiệu bộ vi xử lý 16 bit tên là 8088 có tốc độ xử lý là 2,5MIPS (Millions Instruction Per Second – triệu lệnh/giây) và có thể quản lý được 1MB bộ nhớ. Bộ vi xử lý 80286 cũng là bộ vi xử lý 16 bit nhưng nó có thể quản lý được tới 16MB bộ nhớ.

Các ứng dụng đòi hỏi tốc độ ngày càng cao, dung lượng bộ nhớ và độ rộng bus dữ liệu (số chân của vi xử lý để truyền/nhận dữ liệu) ngày càng lớn. Năm 1986, Intel giới thiệu bộ vi xử lý 32 bit có tên là 80386 có thể quản lý được 4GB bộ nhớ. Năm 1989, Intel giới thiệu bộ vi xử lý 80486 có thể thực hiện được 50MIPS. Năm 1993, Intel giới thiệu bộ vi xử lý 80585 (còn gọi là Pentium hay P5) có thể thực hiện được 110MIPS và có thể thực hiện được đồng thời một lúc hai lệnh (hai lệnh độc lập nhau) vì bên trong có tới hai bộ xử lý số nguyên (công nghệ superscalar). Đầu năm 2003, Intel đã xuất xưởng các bộ vi xử lý Pentium IV có tốc độ 3,06MHz với kiến trúc đa luồng (HT – hyper threading – đa luồng) để nâng cao hiệu suất sử dụng các khối chức năng bên trong vi xử lý, với công nghệ đa luồng, một bộ vi xử lý vật lý tương đương như hai bộ vi xử lý logic. Việc tăng tần số làm việc của các bộ vi xử lý để tăng tốc độ làm việc của máy tính cũng đã đến mức tới hạn nên các hãng sản xuất đã chuyển từ hướng tăng tần số làm việc sang thiết kế các bộ vi xử lý nhiều lõi, mỗi bộ vi xử lý lúc này thực chất là nhiều bộ vi xử lý (multi core – nhiều lõi) đóng vỏ chung trong một vi mạch. Đầu năm 2006, Intel đã xuất xưởng bộ vi xử lý hai nhân, gọi tắt là duo (viết tắt của dual core – hai lõi), hãng này cũng đang chuẩn bị cho ra mắt các bộ vi xử lý bốn nhân (quad core), tám nhân...

1.3. CẤU TRÚC CỦA MÁY TÍNH DÙNG VI XỬ LÝ (HỆ VI XỬ LÝ)

Vi xử lý là một thành phần cơ bản không thể thiếu của máy tính, ngoài ra để tạo ra một hệ hoàn chỉnh cần phải có các bộ phận khác như bộ nhớ, các thiết bị vào/ra như bàn phím, màn hình...

Hình 1.1 giới thiệu sơ đồ khối tổng quát của một máy tính sử dụng vi xử lý:



Hình 1.1. Sơ đồ khối cấu trúc của máy tính

Trong sơ đồ này ta thấy một máy tính (hay hệ vi xử lý) bao gồm các khối chức năng sau:

+ Bộ vi xử lý (CPU).

+ Bộ nhớ bán dẫn (ROM, RAM).

+ Mạch ghép nối vào/ra.

+ Bus hệ thống để truyền thông tin giữa các khối, bus hệ thống gồm bus điều khiển, bus địa chỉ và bus dữ liệu.

Sau đây chúng ta sẽ tìm hiểu nguyên lý làm việc của máy tính bằng cách xem xét chức năng của các khối đó.

1.3.1. Bộ vi xử lý

Bộ vi xử lý (microprocessor) hay còn được gọi là CPU (Central Processing Unit – đơn vị xử lý trung tâm) đóng vai trò là bộ não của máy tính. Đây là một vi mạch số với mức độ tích hợp cực lớn (VLSI) bên trong nó bao gồm nhiều khối chức năng khác nhau như đơn vị số nguyên để thao tác

tính toán với các số nguyên, đơn vị xử lý dấu phảy động để thực hiện các phép tính với số thực... Khi hoạt động, nó đọc mã lệnh (mã lệnh được ghi dưới dạng chuỗi các bit 0, 1) từ bộ nhớ, đưa vào trong vi xử lý để giải mã thành các vi lệnh, đây là những xung điều khiển để điều khiển hoạt động của các đơn vị chức năng bên trong vi xử lý.

Các thông số quan trọng của một bộ vi xử lý gồm:

+ Tần số làm việc: là tần số xung nhịp (clock) cung cấp cho vi xử lý, tần số này quyết định đến tốc độ làm việc của vi xử lý.

+ Độ rộng bus dữ liệu m: là số đường dây dùng để truyền dữ liệu ký hiệu từ D_0 đến D_{m-1}. Các giá trị của m thường là 4, 8, 16, 32 và 64.

+ Độ rộng bus địa chỉ n: quyết định đến dung lượng bộ nhớ cực đại mà vi xử lý có thể quản lý được. Một bộ vi xử lý có n đường địa chỉ từ A_0 đến A_{n-1} có thể quản lý được 2^n ô nhớ (mỗi ô nhớ thường là một byte). Các giá trị của n thường là 16, 20 và 32.

1.3.2. Bộ nhớ

Bộ nhớ (hay còn gọi là bộ nhớ trong, bộ nhớ chính) được tạo từ các vi mạch nhớ ROM và RAM là nơi chứa các chương trình cần thực thi.

ROM (Read Only Memory – bộ nhớ chỉ đọc), nội dung bên trong của ROM không bị mất khi mất nguồn nuôi, dùng để chứa các chương trình điều khiển hệ thống như chương trình để kiểm tra các thiết bị mỗi khi bật nguồn, chương trình khởi động máy, các chương trình điều khiển trao đổi tin với các thiết bị ngoại vi như bàn phím, màn hình,...

RAM (Random Access Memory – bộ nhớ truy cập ngẫu nhiên) là bộ nhớ có thể ghi/đọc được, có nghĩa là ta có thể đọc thông tin từ bộ nhớ, xoá thông tin cũ trong bộ nhớ hoặc ghi thông tin mới vào bộ nhớ; nội dung thông tin ghi trong bộ nhớ RAM sẽ bị mất khi mất nguồn cung cấp. RAM được dùng để lưu trữ mã lệnh, toán hạng và kết quả của chương trình khi nó đang được thực hiện. Trong máy tính, bộ nhớ RAM là các module dạng thanh cắm trên bảng mạch chính của máy, trên mỗi module thường gắn nhiều vi mạch RAM và thường có dung lượng là 1, 2, 4, 8, 16, 32, 64, 128, 256 hoặc 512MB.

1.3.3. Mạch ghép nối vào/ra

Mạch ghép nối vào/ra có nhiệm vụ tạo ra khả năng giao tiếp giữa hệ vi xử lý với thế giới bên ngoài. Các thiết bị vào/ra (hay còn gọi là thiết bị ngoại vi) bao gồm các thiết bị vào (bàn phím, chuột, máy quét...), thiết bị ra (màn

hình, máy in, máy vẽ...), các thiết bị lưu trữ (còn gọi là bộ nhớ ngoài) dùng để lưu thông tin với khối lượng lớn như ổ đĩa cứng, ổ đĩa CD, ổ đĩa DVD...

Mạch ghép nối vào/ra có thể là một vi mạch cỡ nhỏ như 8255 để ghép nối song song, 8251 để ghép nối nối tiếp... Tuy nhiên trong các máy tính hiện nay mạch ghép nối vào/ra là những vi mạch cỡ lớn (VLSI) được gọi là chipset, ví dụ chipset 848P của Intel cho phép ghép nối giữa vi xử lý với cổng đồ hoạ tốc độ cao AGP 8x, với ổ đĩa cứng kiểu SATA, với hệ thống âm thanh 5.1, với các thiết bị vào/ra qua cổng USB 2.0...

1.3.4. Bus hệ thống

Bus hệ thống là tập hợp tất cả các đường dây dùng để liên lạc giữa các khối chức năng trong hệ thống. Dựa vào chức năng của các đường dây người ta chia chúng làm 3 nhóm:

+ Bus điều khiển là các đường dây mang các tín hiệu điều khiển hoạt động hoặc phản ánh trạng thái của các khối như /RD (read – đọc bộ nhớ hoặc thiết bị vào), /WR (write – ghi dữ liệu vào bộ nhớ hoặc xuất dữ liệu ra thiết bị ra), INT (interrupt – ngắt vi xử lý để trao đổi dữ liệu)...

+ Bus dữ liệu là các đường dây mang số liệu mà vi xử lý đang trao đổi với bộ nhớ hoặc thiết bị vào/ra.

+ Bus địa chỉ mang thông tin về địa chỉ của ô nhớ hay một thiết bị vào/ra mà vi xử lý đang trao đổi tin. Thông tin về địa chỉ là do vi xử lý phát ra để chọn ra một ô nhớ hoặc một thiết bị vào/ra mà nó cần trao đổi tin.

1.4. KHÁI NIỆM VỀ PHẦN CỨNG, PHẦN MỀM

1.4.1. Phần cứng

Phần cứng (hardware) là thuật ngữ dùng để chỉ toàn bộ những thiết bị cơ khí, điện tử tạo nên máy tính như các ổ đĩa, màn hình, bàn phím...

1.4.2. Phần mềm

Phần mềm (software) là thuật ngữ dùng để chỉ các chương trình máy tính, nó được thực thi trên phần cứng bằng cách điều khiển sự hoạt động của phần cứng. Phần mềm được chia thành các loại sau:

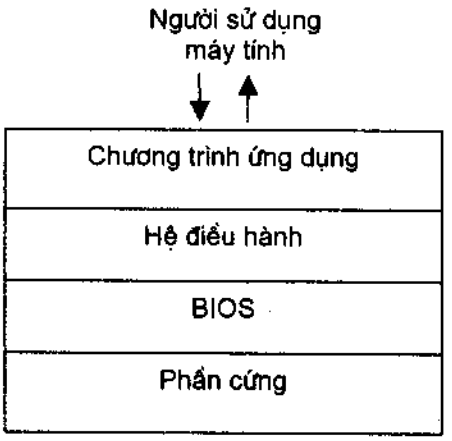
– Hệ điều hành (operating system) như DOS, Windows, Linux...

– Trình tiện ích như NC, NU, BKAV...

– Chương trình ứng dụng như MS Word, MultiSIM, Protel...

– Ngôn ngữ lập trình như Pascal, C, C++, Java... Các ngôn ngữ lập trình được các lập trình viên sử dụng để viết ra các chương trình ứng dụng, thậm chí cả hệ điều hành (chẳng hạn như C được sử dụng để viết hệ điều hành Unix).

Ngoài ra, người ta còn sử dụng khái niệm phần dẻo (firmware) hay phần sụn để chỉ những chương trình tồn tại lâu dài cùng với phần cứng như các chương trình lưu trong bộ nhớ ROM của máy tính, chương trình giải nén của đầu đĩa VCD/MP3... Hình 1.2 sẽ giúp chúng ta phân biệt khái niệm, chức năng của phần cứng, phần mềm.



Hình 1.2. Sơ đồ phân cấp phần cứng – phần mềm

Người sử dụng máy tính sử dụng các chương trình ứng dụng để tương tác với máy tính. Họ cũng có thể trực tiếp sử dụng các dịch vụ của hệ điều hành như định dạng ổ đĩa, tạo/xoá thư mục...

Hệ điều hành cung cấp cho chương trình ứng dụng các dịch vụ (như tạo/xoá thư mục, định dạng ổ đĩa, xoá tập tin...), các hàm chức năng như các hàm API... Các lập trình viên khi lập trình sẽ viết các câu lệnh gọi các dịch vụ, hàm API... của hệ điều hành để thực hiện các chức năng mà chương trình ứng dụng yêu cầu.

Hệ thống các chương trình vào/ra cơ bản – BIOS (Basic Input Output System): cung cấp các hàm cho hệ điều hành để thực hiện các chức năng vào/ra cơ bản như đọc phím ấn của bàn phím, xuất dữ liệu lên màn hình, đọc ổ đĩa từ... Các lập trình viên cũng có thể sử dụng ngôn ngữ Assembly để gọi các chương trình này trong khi viết ra các chương trình ứng dụng.

Phần cứng là nơi thực thi các yêu cầu của người sử dụng. Vi xử lý là thiết bị cuối cùng phải thực thi các nhiệm vụ mà người sử dụng đưa ra thông qua việc chạy các lệnh trong tập lệnh của nó.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 1

1. Nêu sự khác nhau cơ bản giữa ROM và RAM.
2. Thế nào là mã BCD đóng gói, mã BCD không đóng gói.
3. Tìm quan hệ giữa mã BCD và mã ASCII
4. Đổi các số sau ở hệ thập phân sang hệ thập lục phân: 252; 65534; 16632.
5. Đổi các số sau sang hệ nhị phân và hệ thập phân: 001FH; 2FE0H; 0FFFFH.

Chương 2. HỌ VI XỬ LÝ 80×86 CỦA INTEL

2.1. BỘ VI XỬ LÝ 8086 CỦA INTEL

Bộ vi xử lý 8086, được giới thiệu năm 1978, là bộ vi xử lý 16 bit đầu tiên của Intel, mở đầu cho họ vi xử lý x86. Bên trong 8086 gồm 29000 transistor, được sản xuất bằng công nghệ NMOS hoặc CMOS với ba phiên bản:

– 8086 hoạt động ở tần số 4,77MHz.

– 8086–8 hoạt động ở tần số 8MHz.

– 8086–10 hoạt động ở tần số 10MHz.

Cả ba phiên bản đều được đóng gói dạng DIP 40 chân, điện áp nuôi là 5V.

Sau đây chúng ta sẽ đi sâu nghiên cứu cấu trúc bên trong và hoạt động của 8086.

2.1.1. Sơ đồ khối của 8086

Cấu trúc bên trong của 8086 được mô tả một cách đơn giản thông qua sơ đồ khối trên hình 2.1.

Có thể coi 8086 gồm hai khối chính:

2.1.1.1. Khối thực hiện lệnh

Khối thực hiện lệnh (EU – Execution Unit) là nơi giải mã và thi hành các lệnh. EU bao gồm:

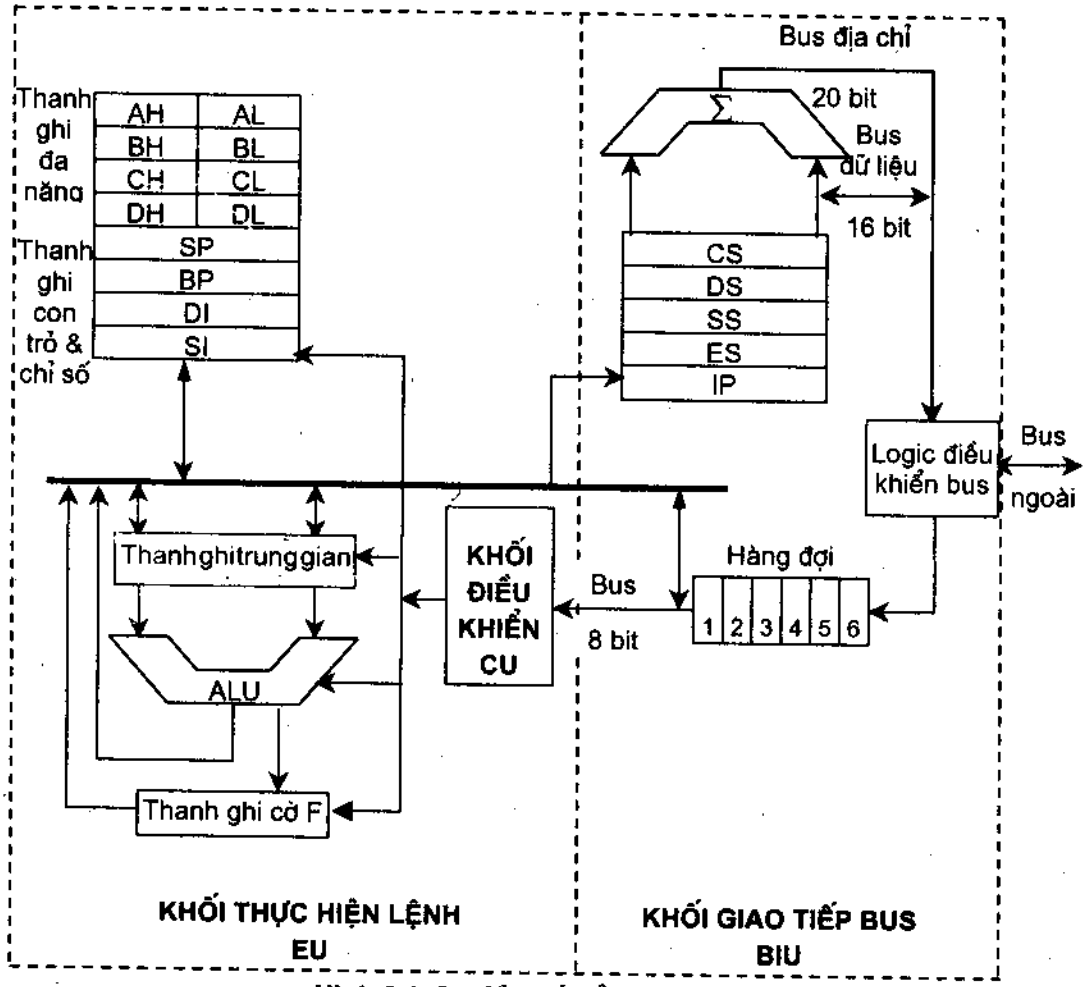
– Bộ xử lý số học và logic (ALU – Arithmatic Logical Unit) là nơi thực hiện các lệnh số học và lệnh logic.

– Các thanh ghi 16 bit chứa toán hạng.

– Thanh ghi cờ F.

– Khối điều khiển (CU – Control Unit) có nhiệm vụ tạo các tín hiệu điều khiển các bộ phận bên trong và bên ngoài CPU. Bên trong khối điều khiển này có mạch giải mã lệnh. Mã lệnh được đọc vào từ bộ nhớ và đưa đến đầu vào của bộ giải mã, các thông tin thu được từ đầu ra của mạch giải mã sẽ được đưa đến mạch tạo xung điều khiển, kết quả là ta thu được các dãy xung

khác nhau (tuỳ theo mã lệnh) để điều khiển hoạt động của các bộ phận bên trong và bên ngoài CPU.



Hình 2.1. Sơ đồ khối của 8086

2.1.1.2. Khối giao tiếp bus

Khối giao tiếp bus (BIU – Bus Interface Unit) có nhiệm vụ đảm bảo việc trao đổi thông tin giữa 8086 với các linh kiện bên ngoài (các vi mạch nhớ, vi mạch ghép nối vào/ra). BIU bao gồm:

– Một bộ cộng để tạo địa chỉ vật lý 20 bit từ các thanh ghi 16 bit.

– Bốn thanh ghi đoạn 16 bit gồm CS, DS, SS và ES để giúp 8086 truy cập tới các đoạn trên bộ nhớ.

– Thanh ghi con trỏ lệnh IP: IP được gọi là con trỏ lệnh vì nó kết hợp với CS để tạo thành địa chỉ của lệnh tiếp theo mà 8086 sẽ thi hành.

– Mạch logic điều khiển bus có nhiệm vụ đảm bảo giao tiếp giữa 8086 với các thiết bị bên ngoài (thông qua bus ngoài).

– Hàng đợi lệnh (hay còn gọi là bộ đệm lệnh) có độ dài 6 byte là nơi chứa các mã lệnh đọc được nằm sẵn để chờ EU xử lý. Đây là một cấu trúc mới được Intel đưa vào bộ vi xử lý 8086 để thực hiện kỹ thuật xử lý xen kẽ liên tục dòng mã lệnh (kỹ thuật pipeline) nhằm cải thiện tốc độ xử lý của CPU.

Thông thường trong các bộ vi xử lý ở các thế hệ trước, hoạt động của CPU để thực hiện một lệnh gồm 3 giai đoạn:

– Đọc mã lệnh (fetch, viết tắt là F).

– Giải mã lệnh (decode, viết tắt là D).

– Thực hiện lệnh (execute, viết tắt là E).

Với giả thiết không gặp lệnh nhảy hoặc lệnh gọi chương trình con thì hoạt động của CPU diễn ra tuần tự: đọc mã lệnh 1, giải mã lệnh 1, thi hành lệnh 1, đọc mã lệnh 2, giải mã lệnh 2, thi hành mã lệnh 2, đọc lệnh mã 3,... Trong một thời điểm nhất định, CPU chỉ có thể thực hiện một trong ba công việc nói trên và vì vậy tuỳ theo từng giai đoạn sẽ có những bộ phận nhất định của CPU ở trạng thái nhàn rỗi. Để khắc phục nhược điểm này, trong bộ vi xử lý 8086, Intel sử dụng cơ chế xử lý xen kẽ liên tục dòng mã lệnh. CPU được chia thành hai khối chức năng và có sự phân chia công việc cho từng khối. Việc đọc mã lệnh là do khối BIU đảm nhiệm, việc giải mã và thi hành lệnh do khối EU thực hiện. Các khối chức năng này có khả năng làm việc đồng thời và các bus sẽ liên tục được sử dụng. Trong khi EU lấy mã lệnh từ bộ hàng đợi lệnh để giải mã hoặc thực hiện các thao tác nội bộ thì BIU vẫn có thể đọc mã lệnh của lệnh tiếp theo từ bộ nhớ chính rồi đặt chúng vào hàng đợi lệnh. Hàng đợi lệnh này làm việc theo kiểu FIFO (First In First Out, vào trước – ra trước), nghĩa là byte nào được cất vào trước sẽ được lấy ra xử lý trước. Nếu có sự vào/ra liên tục của dòng mã lệnh trong hàng đợi lệnh này thì có nghĩa là có sự phối hợp hoạt động hiệu quả giữa hai khối EU và BIU, kết quả là tốc độ của CPU được cải thiện.

2.1.2. Các thanh ghi của 8086

Bên trong bộ vi xử lý 8086 có các thanh ghi 16 bit nằm trong cả hai khối BIU và EU. Ngoài ra, cũng có một số thanh ghi 8 bit hoặc 16 bit tại EU. Chúng ta sẽ lần lượt xem xét các thanh ghi nói trên cùng chức năng chính của chúng.

2.1.2.1. Các thanh ghi đa năng

Trong khối EU có 4 thanh ghi đa năng 16 bit là AX, BX, CX và DX. Điều đặc biệt là khi cần chứa các dữ liệu 8 bit thì mỗi thanh ghi này có thể tách ra thành hai thanh ghi 8 bit làm việc độc lập, đó là các cặp thanh ghi AH và AL, BH và BL, CH và CL, DH và DL (trong đó H chỉ phần cao, L chỉ phần thấp). Mỗi thanh ghi có thể được dùng một cách vạn năng để chứa các loại dữ liệu khác nhau. Nhưng cũng có những công việc đặc biệt nhất định chỉ thao tác với một vài thanh ghi nào đó, chính vì vậy các thanh ghi thường được gán cho những cái tên đặc biệt và có ý nghĩa riêng.

- AX (Accumulator register): thanh ghi chứa, các kết quả của các thao tác thường được chứa ở đây. Nếu kết quả là 8 bit thì thanh ghi AL được sử dụng.

- BX (Base register): thanh ghi cơ sở, thường chứa địa chỉ cơ sở của một bảng khi dùng lệnh XLAT.

- CX (Count register): thanh ghi đếm, CX thường được dùng để chứa số lần lặp trong trường hợp lệnh LOOP (lặp), còn CL thường chứa một số lần dịch hoặc quay trong các lệnh dịch hoặc quay thanh ghi.

- DX (Data register): thanh ghi dữ liệu, DX cùng AX tham gia vào các thao tác của phép nhân hoặc chia các số 16 bit. DX còn dùng để chứa địa chỉ của các cổng trong các lệnh vào/ra dữ liệu trực tiếp.

2.1.2.2. Các thanh ghi đoạn

Khối BIU đưa ra trên bus địa chỉ 20 bit địa chỉ, như vậy 8086 có khả năng phân biệt ra được 2^{20}= 1048576 = 1M ô nhớ, hay 1Mbyte. Trong không gian 1Mbyte này bộ nhớ cần được chia thành các vùng khác nhau dành riêng để:

- Chứa mã chương trình.

- Chứa dữ liệu và kết quả trung gian của chương trình.

- Tạo ra một vùng nhớ đặc biệt gọi là ngăn xếp dùng vào việc quản lý các thông số của bộ vi xử lý.

Trong thực tế, bộ vi xử lý 8086 có các thanh ghi 16 bit liên quan đến địa chỉ đầu của các vùng kể trên và chúng được gọi là thanh ghi đoạn. Đó là thanh ghi đoạn mã CS (code segment), thanh ghi đoạn dữ liệu DS (data segment), thanh ghi đoạn ngăn xếp SS (stack segment) và thanh ghi đoạn dữ liệu phụ ES (extra segment). Các thanh ghi đoạn 16 bit này chỉ ra địa chỉ đầu của 4 đoạn trong bộ nhớ, dung lượng lớn nhất của mỗi đoạn nhớ này là 64Kbyte.

Nội dung các thanh ghi đoạn sẽ xác định địa chỉ của ô nhớ nằm ở đầu đoạn. Địa chỉ này còn gọi là địa chỉ cơ sở. Địa chỉ của các ô nhớ khác nằm trong đoạn tính được bằng cách cộng thêm vào địa chỉ cơ sở một giá trị gọi là địa chỉ lệch hay độ lệch. Độ lệch này được xác định bởi một thanh ghi 16 bit khác đóng vai trò thanh ghi lệch. Để xác định địa chỉ vật lý 20 bit của một ô nhớ nào đó trong một đoạn bất kỳ thì 8086 phải dùng đến hai thanh ghi 16 bit. Từ nội dung của hai thanh ghi này nó tạo ra địa chỉ vật lý theo công thức:

Địa chỉ vật lý = Nội dung thanh ghi đoạn × 16 + Nội dung thanh ghi lệch

Việc dùng 2 thanh ghi để ghi nhớ thông tin về địa chỉ thực chất tạo ra một loại địa chỉ gọi là địa chỉ logic và được ký hiệu:

thanh ghi đoạn: thanh ghi lệch hay ***segment: offset***

2.1.2.3. Thanh ghi con trỏ và chỉ số

Trong 8086 còn có ba thanh ghi con trỏ và hai thanh ghi chỉ số 16 bit là:

– Thanh ghi con trỏ lệnh IP (Instruction Pointer): luôn trỏ vào lệnh tiếp theo sẽ được thực hiện nằm trong đoạn mã CS. Địa chỉ đầy đủ của lệnh tiếp theo này là CS: IP.

– Thanh ghi con trỏ cơ sở BP (Base Pointer): luôn trỏ vào một phần tử dữ liệu nằm trong đoạn ngăn xếp SS. Địa chỉ đầy đủ của một phần tử trong đoạn ngăn xếp SS là SS: BP.

– Thanh ghi con trỏ ngăn xếp SP (Stack Pointer): luôn chỉ vào đỉnh hiện thời của ngăn xếp nằm trong đoạn ngăn xếp SS. Địa chỉ đầy đủ của đỉnh ngăn xếp là SS: SP.

– Thanh ghi chỉ số nguồn SI (Source Index): SI chỉ vào dữ liệu trong đoạn dữ liệu DS mà địa chỉ đầy đủ là DS: SI.

– Thanh ghi chỉ số đích DI (Destination Index): chỉ vào dữ liệu trong đoạn dữ liệu DS mà địa chỉ đầy đủ là DS: DI.

Ngoài các chức năng trên, các thanh ghi này đều có thể được dùng như các thanh ghi đa năng.

2.1.2.4. Thanh ghi cờ F

Đây là thanh ghi khá đặc biệt trong CPU, mỗi bit của nó được dùng để phản ánh một trạng thái nhất định của kết quả phép toán do ALU thực hiện hoặc một trạng thái hoạt động của EU. Dựa vào các cờ này người lập trình có thể có các lệnh thích hợp tiếp theo cho bộ vi xử lý. Thanh ghi cờ gồm 16 bit nhưng người ta chỉ dùng hết 9 bit của nó để làm các bit cờ.

×	×	×	×	O	D	I	T	S	Z	×	A	×	P	×	C

×: Không được định nghĩa

Hình 2.2. Sơ đồ thanh ghi cờ của bộ vi xử lý 8086

Các cờ cụ thể:

– C hoặc CF (Carry Flag): Cờ nhớ, CF = 1 khi có nhớ hoặc mượn.

– P hoặc PF (Parity Flag): Cờ chẵn lẻ, phản ánh tính chẵn lẻ của tổng số bit 1 có trong kết quả. Cờ PF = 1 khi tổng số bit 1 trong kết quả là chẵn.

– A hoặc AF (Auxiliary carry Flag): Cờ nhớ phụ có ý nghĩa khi ta làm việc với các số BCD; AF = 1 khi có nhớ hoặc mượn từ một số BCD thấp (ở 4 bit thấp) sang một số BCD cao (ở 4 bit cao).

– Z hoặc ZF (Zero Flag): Cờ rỗng (cờ không), ZF = 1 khi kết quả bằng 0.

– S hoặc SF (Sign Flag): Cờ dấu, SF = 1 khi kết quả âm.

– OF (Overflow Flag): Cờ tràn, OF = 1 khi kết quả là một số bù 2 vượt ra ngoài giới hạn biểu diễn dành cho nó.

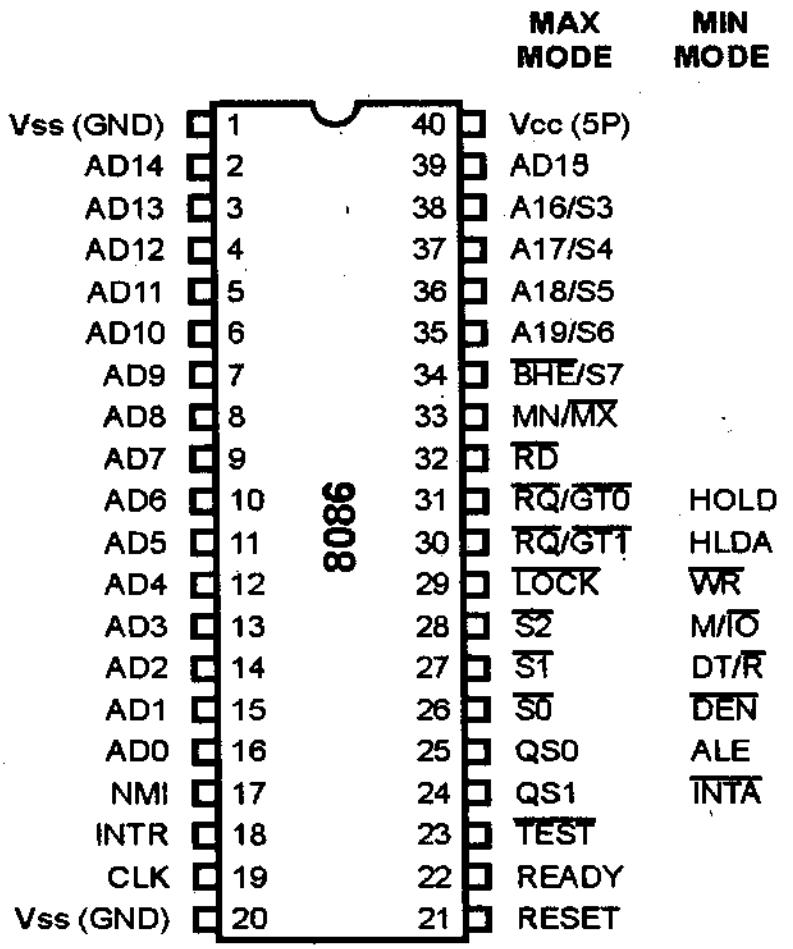
Trên đây là 6 bit cờ trạng thái phản ánh các trạng thái khác nhau của kết quả sau một thao tác nào đó. Trong đó, 5 bit cờ đầu thuộc byte thấp của thanh ghi cờ là các cờ giống bộ vi xử lý 8085 của Intel. Ngoài ra bộ vi xử lý 8086 còn có các cờ điều khiển sau:

– T hoặc TF (Trap Flag): Cờ bẫy, TF = 1 thì CPU làm việc ở chế độ chạy từng lệnh một.

– I hoặc IF (Interrupt Flag): Cờ cho phép ngắt, nếu IF = 1 thì CPU cho phép các yêu cầu ngắt (che được) được tác động, nếu IF = 0 thì CPU cấm các yêu cầu ngắt (che được) tác động trừ ngắt ở chân NMI (Non Maskable Interrupt: ngắt không che được).

– D hoặc DF (direction flag): Cờ hướng (cờ lùi), IF = 1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải qua trái.

2.1.3. Sơ đồ chân và chức năng các chân của 8086



Hình 2.3. Sơ đồ chân của 8086

Vi xử lý 8086 được thiết kế để hoạt động ở một trong hai chế độ, tuỳ thuộc vào mức điện áp đặt ở chân số 33 (chân MN/$\overline{MX}$):

– Chế độ tối thiểu (chế độ MIN) được thiết lập nếu điện áp chân số 33 là 5V. Chế độ tối thiểu là chế độ trong hệ thống chỉ có 8086 và các vi mạch nhớ, các vi mạch ghép nối vào/ra.

– Chế độ tối đa (chế độ MAX) được thiết lập nếu điện áp chân số 33 là 0V. Chế độ tối đa là chế độ áp dụng cho hệ thống đa xử lý (nhiều vi xử lý

8086), đồng xử lý (8086 và bộ đồng xử lý toán học 8087). Trong chế độ tối đa, bộ vi xử lý có thể sử dụng vi mạch điều khiển bus bên ngoài để giải mã các tín hiệu trạng thái /S0 /S1 và /S2 và cung cấp tất cả các tín hiệu điều khiển bus.

Tùy thuộc vào chế độ hoạt động được thiết lập mà các chân từ số 24 đến số 31 có chức năng xác định khác nhau; chẳng hạn như chân số 25 ở chế độ MAX có tên là QS0, còn ở chế độ MIN nó có tên là ALE. Sau đây chúng ta sẽ xem xét chức năng các chân của 8086.

** Các chân mang thông tin địa chỉ:*

Vi xử lý 8086 có 20 đường địa chỉ bao gồm từ A_0 đến A_{19} trong đó 16 đường dây địa chỉ thấp từ A_0 đến A_{15} được ghép kênh với các đường dây dữ liệu từ D_0 đến D_{15} trên các chân từ AD_0 đến AD_{15}; còn 4 đường dây địa chỉ cao nhất từ A_{16} đến A_{19} của 8086 cũng được ghép kênh, nhưng trong trường hợp này nó được ghép kênh với các tín hiệu trạng thái từ S_3 đến S_6 trên các chân từ A_{16}/S_3 đến A_{19}/S_6. Do đó tại chu kỳ bus địa chỉ các đường dây này được sử dụng để mang thông tin địa chỉ đưa đến bộ nhớ hoặc các cổng vào/ra. Như vậy bus địa chỉ của 8086 có độ rộng 20 bit bao gồm các đường dây từ A_0 đến A_{19}, tuy nhiên chỉ có 16 đường dây địa chỉ từ A_0 đến A_{15} được sử dụng khi truy cập các cổng vào/ra. Điều này tạo cho 8086 một không gian địa chỉ vào /ra độc lập 64Kbyte.

** Các chân mang thông tin về dữ liệu:*

Vi xử lý 8086 có 16 đường dây dữ liệu từ D_0 đến D_{15} được ghép kênh với 16 đường địa chỉ thấp từ A_0 đến A_{15}. Khi hoạt động ở chu kỳ bus dữ liệu thì các đường dây này mang thông tin về dữ liệu – là dữ liệu đọc ra hay viết vào bộ nhớ hay thiết bị vào/ra, hay các mã về các loại ngắt từ bộ điều khiển ngắt 8259.

** Các chân tín hiệu trạng thái:*

+ 4 đường dây địa chỉ cao nhất từ A_{16} đến A_{19} của 8086 cũng được ghép kênh, nhưng trong trường hợp này nó được ghép kênh với các tín hiệu trạng thái từ S_3 đến S_6. Các bit trạng thái này được đưa ra cùng thời điểm với các dữ liệu được truyền trên các chân $AD_0 – AD_{15}$.

Bit S3 và S4 kết hợp cùng nhau tạo ra 2 bit mã nhị phân để xác định thanh ghi đoạn nào được sử dụng để tạo ra địa chỉ vật lý được đưa lên bus địa chỉ trong chu kỳ bus hiện tại theo bảng 2.1.

Bảng 2.1. Mối quan hệ giữa mã của S_4, S_3 với thanh ghi được sử dụng

S_4	S_3	Tên thanh ghi đoạn được sử dụng
0	0	ES
0	1	SS
1	0	CS
1	1	DS

Đường dây trạng thái S_5 phản ánh trạng thái của một tính chất khác bên trong vi xử lý, nó là mức logic của bit cờ cho phép ngắt bên trong (IEF).

+ Tín hiệu READY: có thể được cung cấp bởi thiết bị phát xung bên ngoài và có thể được cung cấp bởi bộ nhớ hoặc hệ thống vào/ra để báo cho CPU khi nó sẵn sàng cho phép dữ liệu truyền được hoàn thành. Tín hiệu này có thể được sử dụng để chèn thêm các trạng thái chờ vào chu kỳ bus, do đó mà có thể kéo dài thêm chu kỳ đồng hồ. Điều này rất có ích khi cần trao đổi tin giữa 8086 với một thiết bị ngoài có tốc độ chậm hơn nó.

** Các chân tín hiệu điều khiển:*

Các tín hiệu điều khiển được cung cấp để giúp cho 8086 giao tiếp với bộ nhớ và thiết bị vào/ra, chúng được 8086 đưa ra bus điều khiển bên ngoài để điều khiển hoạt động của các linh kiện khác.

+ Tín hiệu ALE (Address Latch Enable – cho phép chốt địa chỉ) là một xung nâng lên mức 1 để báo cho mạch ngoài biết có một địa chỉ hợp lệ ở trên bus. Tín hiệu này được đưa vào điều khiển một vi mạch mạch chốt ở bên ngoài để tách các bit địa chỉ từ A_0 đến A_{15} ra khỏi đường dây ghép kênh địa chỉ – dữ liệu từ AD_0 đến AD_{15}.

+ Tín hiệu $\overline{DEN}$ (Data Enable – cho phép dữ liệu) dùng để báo có dữ liệu hợp lệ ở trên bus bởi mức logic 0. $\overline{DEN}$ thường được đưa tới điều khiển vi mạch đệm dữ liệu trên bus dữ liệu.

+ Tín hiệu M/$\overline{IO}$ báo cho các mạch điện bên ngoài biết bộ nhớ hoặc thiết bị vào/ra đang được nối với bus. Mức logic 0 tại đầu ra này báo cho các mạch ngoài biết thiết bị vào/ra đang được sử dụng bus, mức logic 1 báo rằng bộ nhớ đang sử dụng bus.

+ Tín hiệu DT/$\overline{R}$ dùng để báo hướng truyền dữ liệu trên bus. Khi DT/$\overline{R}$ =1 trong suốt một phần truyền dữ liệu của bus dữ liệu thì bus dữ liệu ở chế độ truyền. Do đó dữ liệu có thể ghi vào bộ nhớ hoặc đưa ra thiết bị

vào/ra. Ngược lại, mức logic 0 tại chân DT/$\overline{R}$ báo hiệu rằng bus đang ở chế độ nhận. Điều này có nghĩa là dữ liệu đang được đọc ra từ bộ nhớ hoặc dữ liệu được đưa vào từ một cổng vào.

+ Tín hiệu $\overline{BHE}$ (Bank High Enable – cho phép băng cao) dùng để báo rằng đang truy cập băng cao hay băng thấp của bộ nhớ. $\overline{BHE}$ = 0 báo hiệu đang truy cập băng cao của bộ nhớ, ngược lại mức logic 1 ở chân này báo hiệu đang truy cập băng thấp của bộ nhớ.

+ Tín hiệu $\overline{RD}$ (Read) và $\overline{WR}$ (Write) báo rằng một chu kỳ đọc hoặc ghi đang được tiến hành. CPU chuyển $\overline{WR}$ xuống mức logic 0 để báo hiệu cho các thiết bị ngoài rằng dữ liệu ghi hợp lệ hoặc dữ liệu đưa ra đang ở trên bus. CPU chuyển $\overline{RD}$ xuống mức logic 0 để báo hiệu rằng CPU đang đọc dữ liệu từ bộ nhớ hoặc nhận dữ liệu từ một cổng vào ra.

* *Các chân tín hiệu ngắt:*

+ Tín hiệu INTR là một đầu vào của vi xử lý 8086 và có thể được sử dụng bởi một thiết bị ngoài để báo rằng nó đang cần được phục vụ. Logic 1 tại chân này đại diện cho một yêu cầu ngắt tích cực. Khi yêu cầu ngắt được nhận biết bởi CPU thì nó báo sự xác nhận này cho thiết bị bên ngoài với mức logic 0 tại đầu ra $\overline{INTA}$ (Interrupt Acknowledge).

+ Tín hiệu vào $\overline{TEST}$ cũng có quan hệ với giao diện ngắt bên ngoài. Nếu $\overline{TEST}$ có mức logic 1 thì CPU treo hoạt động của mình và nó chuyển đến trạng thái mà được hiểu như trạng thái nghỉ. Khi ở trạng thái này CPU không thực hiện lệnh mà thay vào đó nó kiểm tra lại mức logic ở đầu vào $\overline{TEST}$ và cho đến khi nó trở lại mức 0. Nếu $\overline{TEST}$ chuyển đến logic 0, quá trình lại được tiếp tục với lệnh tiếp theo trong chương trình. Đặc điểm này có thể được sử dụng để đồng bộ hoạt động của 8086 với các thiết bị phần cứng bên ngoài.

Lệnh WAIT (chờ để test) khi thực hiện sẽ tạo ra trạng thái chờ cho bộ vi xử lý ở đầu vào /TEST. Nếu /TEST chuyển xuống mức 0 thì mới tiếp tục quá trình xử lý.

+ Đầu vào ngắt không che được NMI (Non Maskable Interrupt). Khi mức logic trên NMI chuyển từ 0 lên 1, điều khiển được chuyển đến chương trình con phục vụ ngắt không che được tại thời điểm hoàn thành sự thực hiện của lệnh đang chạy. NMI là yêu cầu ngắt có mức logic ưu tiên cao nhất và nó không thể che được bằng phần mềm.

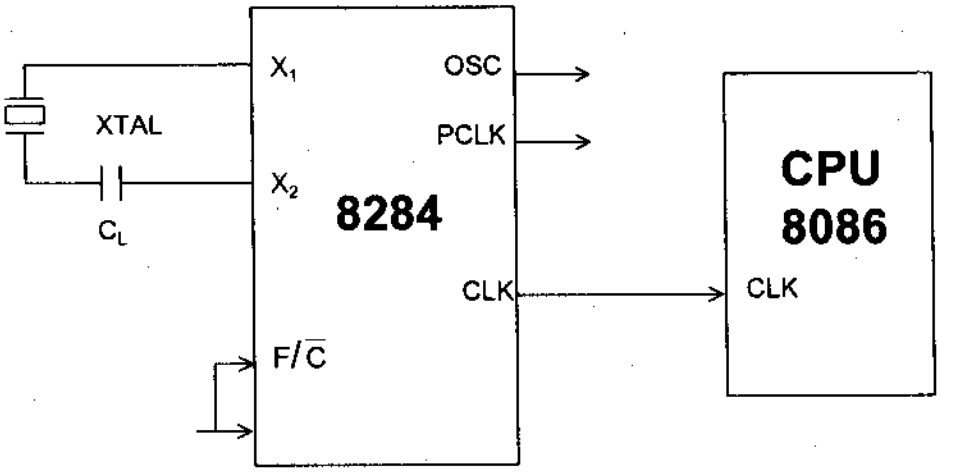
+ Đầu vào RESET được sử dụng để thiết lập lại phần cứng cho CPU. Chuyển RESET xuống mức logic 0 dùng để khởi tạo các thanh ghi nội của vi xử lý và khởi tạo chương trình con phục vụ thiết lập hệ thống.

** Các chân mang tín hiệu phục vụ DMA (Direct Memory Access – truy cập bộ nhớ trực tiếp):*

Giao diện truy cập bộ nhớ trực tiếp của 8086 ở chế độ MIN bao gồm hai tín hiệu HOLD và HLDA. Khi một thiết bị ngoài muốn giành quyền điều khiển bus hệ thống để thực hiện truy cập bộ nhớ trực tiếp, nó báo yêu cầu này cho CPU bằng cách chuyển HOLD lên mức logic 1. Sau đó CPU chuyển sang trạng thái cô lập sau khi chu kỳ bus hiện tại thực hiện xong. Khi nó ở trạng thái cô lập, các đường dây tín hiệu AD_0 – AD_{15}, A_{16}/S_3 – A_{19}/S_6, BHE/S_7, $M/\overline{IO}$, $DT/\overline{R}$, $\overline{WR}$, $\overline{DEN}$ và INTR đều đặt trên trạng thái trở kháng cao. Vi xử lý 8086 báo cho các thiết bị ngoài rằng nó đang ở trạng thái cô lập bằng cách chuyển đầu ra HLDA của nó lên mức logic 1 và lúc này quyền điều khiển bus được chuyển cho vi mạch điều khiển DMA (vi mạch 8237).

** Đồng hồ hệ thống:*

Thời gian cơ sở cho hoạt động đồng bộ bên trong và bên ngoài của vi xử lý trong máy tính được cung cấp bởi đầu vào CLK ở chân số 19. Tín hiệu CLK được tạo ra ở bên ngoài bằng bộ tạo xung 8284 như hình 2.4.

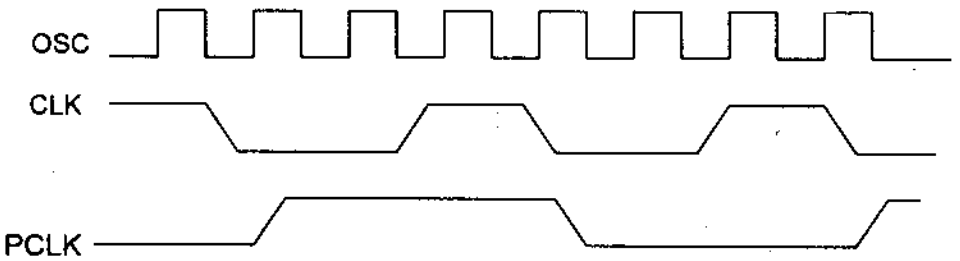


Hình 2.4. Mạch tạo xung đồng hồ cho 8086

Hai đầu vào X_1, X_2 của 8284 được nối với thạch anh có tần số dao động gấp ba lần tốc độ của 8086. Tần số dao động của thạch anh được chia 3 bên trong 8284 để tạo tần số đúng bằng tần số hoạt động của 8086. Tần số này

được đệm ở bên trong và được đưa ra tại chân CLK, đầu ra này được nối trực tiếp đến đầu vào CLK của 8086.

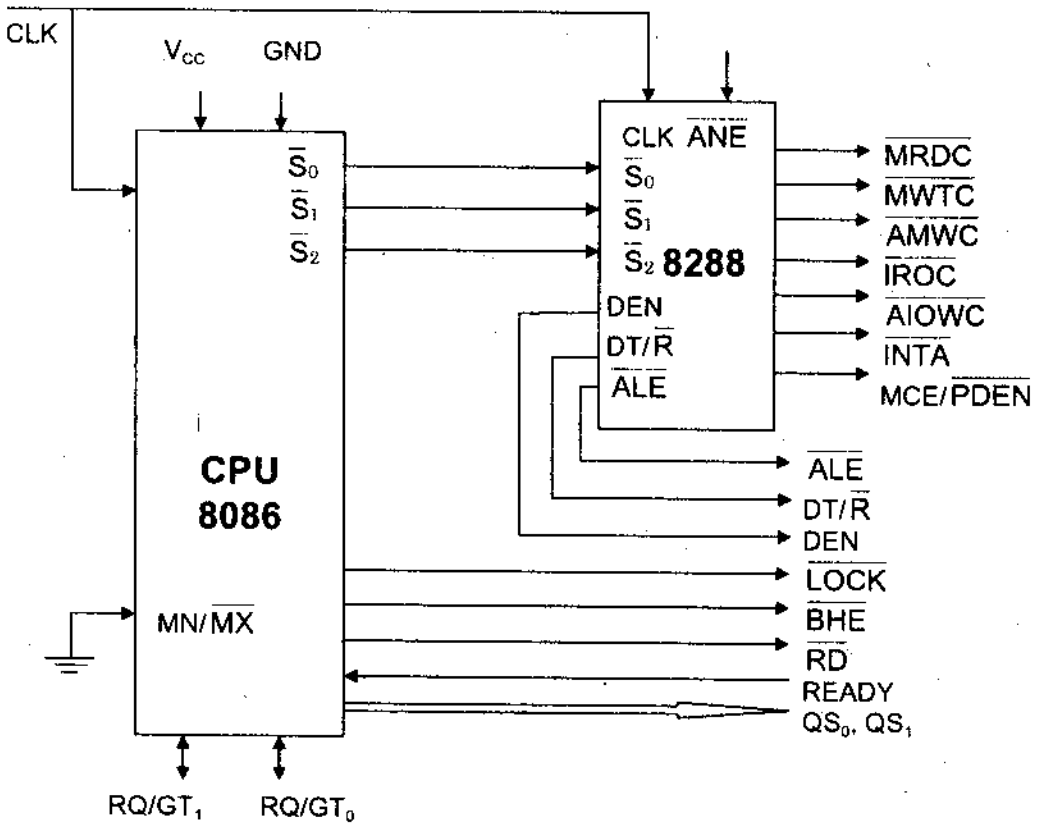
Từ sơ đồ trên ta thấy ngoài tín hiệu đồng hồ CLK cấp cho 8086 thì 8284 còn tạo ra hai tín hiệu xung nhịp nữa là OSC (OSCillator clock) và PCLK (Peripheral CLocK). Hai tín hiệu này dùng để kích cho các IC bên ngoài. Tần số của hai tín hiệu tại đầu ra của PCLK bằng một nửa tần số của tín hiệu tại đầu ra CLK. Tần số ở đầu ra OSC đúng bằng tần số của dao động thạch anh. Vi mạch 8284 cũng có thể được kích một nguồn xung đồng hồ ở bên ngoài được cung cấp tới chân EFI (External Frequency Input) của 8284. Đầu vào F/$\overline{C}$ được cung cấp để lựa chọn nguồn xung clock cung cấp cho 8284 là từ thạch anh hay nguồn xung đồng hồ bên ngoài ở chân EFI. Khi nó bị giữ ở mức 0 thạch anh giữa X1 và X2 được sử dụng. Cung cấp mức logic 1 tại chân F/$\overline{C}$ để lựa chọn EFI là nguồn xung.



Hình 2.5. Mối quan hệ giữa CLK, PCLK và OSC

** Chức năng các chân ở chế độ MAX:*

Khi 8086 được thiết lập cấu hình ở chế độ MAX, nó tạo ra các tín hiệu để thực hiện môi trường đa xử lý, đồng xử lý. Môi trường đa vi xử lý có thể hiểu rằng có nhiều hơn một vi xử lý tồn tại trong hệ thống và mỗi vi xử lý thực hiện chương trình riêng của mình. Thường thì trong môi trường này một số tài nguyên hệ thống dùng chung cho tất cả các vi xử lý. Các tài nguyên này được gọi là tài nguyên toàn cục hay tài nguyên dùng chung. Cũng có các tài nguyên được dành cho các vi xử lý xác định, các tài nguyên này được hiểu là các tài nguyên cục bộ hay tài nguyên dùng riêng. Trong hệ thống hoạt động ở chế độ MAX thì sự phân phát tài nguyên của hệ thống được thực hiện dễ dàng bằng cách chuyển quyền điều khiển bus cho các vi xử lý khác để chia sẻ bus hệ thống. Sơ đồ khối của 8086 làm việc ở chế độ MAX như hình 2.6.



Hình 2.6. Sơ đồ ghép nối 8086 ở chế độ MAX với bộ điều khiển bus 8288

Nhìn vào sơ đồ 2.6 ta thấy ở chế độ MAX, 8086 không trực tiếp cung cấp tất cả các tín hiệu cần thiết để điều khiển bộ nhớ và thiết bị vào/ra, giao diện ngắt. Đặc biệt các tín hiệu $\overline{WR}$, M/$\overline{IO}$, DT/$\overline{R}$, $\overline{DEN}$, ALE và $\overline{INTA}$ không được tạo ra bởi 8086. Thay vào đó, bộ điều khiển bus 8288 tạo ra các tín hiệu này dựa vào các đầu vào $\overline{S}_2$, $\overline{S}_1$ và $\overline{S}_0$ lấy từ 8086.

– Tín hiệu khoá LOCK (Lock signal)

Để thực hiện một hệ thống đa vi xử lý, một tín hiệu gọi là LOCK được cung cấp, tín hiệu này có nghĩa là nếu đầu ra của nó có mức logic 0 mỗi khi một vi xử lý muốn khoá các vi xử lý khác đang sử dụng bus. Tín hiệu LOCK là phù hợp với multibus, đây là một chuẩn công nghiệp cho giao diện các vi xử lý trong một môi trường đa vi xử lý.

– Các tín hiệu trạng thái hàng đợi (Queue status signals) QS_1 và QS_0

Hai tín hiệu này kết hợp với nhau tạo ra hai bit mã trạng thái hàng đợi (queue). Mã này báo cho các mạch bên ngoài biết kiểu thông tin nào vừa được đưa ra từ queue trong suốt chu kỳ đồng hồ lần trước. Bảng bên dưới cho biết 4 mã trạng thái queue khác nhau:

Bảng 2.2. Giải mã QS_1, QS_0

QS_1	$QS0$	Ý nghĩa
0	0	Không lấy dữ liệu khỏi hàng đợi.
0	1	Byte dữ liệu vừa lấy khỏi hàng đợi là byte đầu tiên của một lệnh.
1	0	Hàng đợi rỗng.
1	1	Byte dữ liệu vừa lấy khỏi hàng đợi là byte tiếp theo của một lệnh (không phải byte đầu tiên).

– Các tín hiệu điều khiển bus cục bộ

Trong cấu hình ở chế độ MAX thì hai tín hiệu HOLD và HLDA của hai chế độ MIN cũng được thay đổi. Hai tín hiệu được thay đổi bởi tín hiệu Request/Grant RQ/GT_0 và RQ/GT_1. Chúng cung cấp quyền ưu tiên sử dụng bus cục bộ.

2.2. CÁC BỘ VI XỬ LÝ TIÊN TIẾN CỦA INTEL

2.2.1. Các bộ vi xử lý x86

– Bộ vi xử lý 80186 (1982) còn gọi là iAPX 186, được sử dụng chủ yếu trong những ứng dụng nhúng. Có hai phiên bản của 80186 là 10 và 12MHz.

– Bộ vi xử lý 80286 (1982) được biết đến với tên gọi 286, là bộ vi xử lý đầu tiên của Intel có thể chạy được tất cả ứng dụng viết cho các bộ vi xử lý trước đó, được dùng trong các PC của IBM và các PC tương thích. 286 sử dụng công nghệ 1,5 micromet, 134000 transistor, bộ nhớ mở rộng tới 16MB. Các phiên bản của 286 gồm 6, 8, 10, 12,5, 16, 20 và 25MHz.

– Bộ vi xử lý 80386 gồm các họ 386DX, 386SX và 386SL. Intel386DX là bộ vi xử lý 32 bit đầu tiên Intel giới thiệu vào năm 1985, được dùng trong các PC của IBM và PC tương thích. Intel386 là một bước nhảy vọt so với các bộ vi xử lý trước đó. Đây là bộ vi xử lý 32 bit có khả năng xử lý đa nhiệm, nó có thể chạy nhiều chương trình khác nhau cùng một thời điểm. 80386 sử dụng các thanh ghi 32 bit, có thể truyền 32 bit dữ liệu cùng lúc trên bus dữ liệu và dùng 32 bit để xác định địa chỉ. Cũng như bộ vi xử lý 80286, 80386

hoạt động ở 2 chế độ: real mode và protect mode. 386DX sử dụng công nghệ 1,5 micromet, tạo bởi 275000 transistor, bộ nhớ mở rộng tới 4GB. Các phiên bản của 386DX gồm 16, 20, 25 và 33MHz. Vi xử lý 386SX (năm 1988) sử dụng công nghệ 1,5 micromet, 275000 transistor, kiến trúc 32 bit bên trong, bus dữ liệu ngoài 16 bit, bus địa chỉ 24 bit, bộ nhớ mở rộng 16MB; gồm các phiên bản 16, 20, 25 và 33MHz.

Vi xử lý 386SL (1990) được thiết kế cho thiết bị di động, sử dụng công nghệ 1 micromet, 855000 transistor, bộ nhớ mở rộng 4GB; gồm các phiên bản 16, 20 và 25MHz.

– Bộ vi xử lý 80486DX ra đời năm 1989 với cấu trúc bus dữ liệu 32 bit. 486DX có bộ nhớ sơ cấp (L1 cache) 8KB để giảm thời gian chờ dữ liệu từ bộ nhớ đưa đến, bộ đồng xử lý toán học được tích hợp bên trong. Ngoài ra, 486DX được thiết kế hàng đợi lệnh (pipeline) để có thể xử lý một chỉ lệnh trong một xung nhịp. 486DX sử dụng công nghệ 1 micromet, gồm 1,2 triệu transistor, bộ nhớ mở rộng 4GB; gồm các phiên bản 25MHz, 35MHz và 50MHz. 486SL được sản xuất năm 1992, đây là bộ vi xử lý đầu tiên dành cho máy tính xách tay. 486SL được sản xuất theo công nghệ 0,8 micromet, 1,4 triệu transistor, bộ nhớ mở rộng 4GB; gồm các phiên bản 20, 25 và 33MHz.

– Bộ vi xử lý 80586 (còn gọi là Pentium), là bộ vi xử lý kế tiếp 80486 ra đời năm 1993. Cải tiến lớn nhất của Pentium là thiết kế hai hàng đợi lệnh, do đó Pentium có thể có tốc độ xử lý gấp đôi so với 80486DX. Bộ nhớ sơ cấp 16KB gồm 8KB chứa dữ liệu và 8KB khác để chứa lệnh. Bộ đồng xử lý toán học được cải tiến giúp tăng khả năng tính toán đối với các trình ứng dụng. Pentium sử dụng công nghệ 0,8 micromet chứa 3,1 triệu transistor, có các tốc độ 60 và 66MHz. Các phiên bản 75, 90, 100, 120MHz sử dụng công nghệ 0,6 micromet chứa 3,3 triệu transistor; các phiên bản 133, 150, 166, 200 sử dụng công nghệ 0,35 micromet chứa 3,3 triệu transistor.

– Pentium MMX (năm 1996), phiên bản cải tiến của Pentium với công nghệ MMX được Intel phát triển để đáp ứng nhu cầu về ứng dụng đa phương tiện và truyền thông. MMX kết hợp với SIMD (Single Instruction Multiple Data) cho phép xử lý nhiều dữ liệu trong cùng chỉ lệnh, làm tăng khả năng xử lý trong các tác vụ đồ hoạ, đa phương tiện. Pentium MMX sử dụng công nghệ 0,35 micromet chứa 4,5 triệu transistor, có các tốc độ 166, 200, 233MHz. Nối tiếp sự thành công của dòng Pentium, Pentium Pro được Intel giới thiệu vào tháng 9 năm 1995, sử dụng công nghệ 0,6 và 0,35 micromet chứa 5,5 triệu transistor. Điểm nổi bật của Pentium Pro là bus hệ

thống 60 hoặc 66MHz, bộ nhớ đệm L2 (L2 cache) 256KB hoặc 512KB (trong một số phiên bản). Pentium Pro có các tốc độ 150, 166, 180, 200MHz.

Pentium II (năm 1997), phiên bản cải tiến từ Pentium Pro được sử dụng trong những dòng máy tính cao cấp, máy trạm (workstation) hoặc máy chủ (server). Pentium II có bộ nhớ đệm L1 32KB, L2 512KB, tích hợp công nghệ MMX được cải tiến giúp việc xử lý dữ liệu video, audio và đồ hoạ hiệu quả hơn.

Bộ vi xử lý Pentium II đầu tiên, tên mã Klamath, sản xuất trên công nghệ 0,35 micromet, có 7,5 triệu transistor, bus hệ thống 66MHz, gồm các phiên bản 233, 266, 300MHz.

Pentium II, tên mã Deschutes, sử dụng công nghệ 0,25 micromet; 7,5 triệu transistor, gồm các phiên bản 333MHz (bus hệ thống 66MHz), 350, 400, 450MHz (bus hệ thống 100MHz).

Celeron (năm 1998) được "rút gọn" từ kiến trúc Pentium II, dành cho dòng máy cấp thấp. Phiên bản đầu tiên, tên mã Covington, không có bộ nhớ đệm L2 nên tốc độ xử lý khá chậm, không gây được ấn tượng với người dùng. Phiên bản sau, tên mã Mendocino, đã khắc phục khuyết điểm này với bộ nhớ đệm L2 128KB. Covington sử dụng công nghệ 0,25 micromet, có 7,5 triệu transistor, bộ nhớ đệm L1 32KB, bus hệ thống 66MHz, đế cắm 242 chân Slot 1 SEPP (Single Edge Processor Package), tốc độ 266, 300MHz. Mendocino cũng sử dụng công nghệ 0,25 micromet có đến 19 triệu transistor, bộ nhớ đệm L1 32KB, L2 128KB, bus hệ thống 66MHz, đế cắm Slot 1 SEPP hoặc socket 370 PPGA, tốc độ 300, 333, 366, 400, 433, 466, 500, 533MHz.

Pentium III (năm 1999) bổ sung 70 lệnh mới (Streaming SIMD Extensions – SSE) giúp tăng hiệu suất hoạt động trong các tác vụ xử lý hình ảnh, audio, video và nhận dạng giọng nói. Pentium III gồm các tên mã Katmai, Coppermine và Tualatin. Katmai sử dụng công nghệ 0,25 micromet, bên trong có 9,5 triệu transistor, bộ nhớ đệm L1 32KB, L2 512KB, tốc độ 450, 500, 550, 533 và 600MHz (bus 100MHz), 533, 600MHz (bus 133MHz). Coppermine sử dụng công nghệ 0,18 micromet, bên trong có 28,1 triệu transistor, bộ nhớ đệm L2 256KB được tích hợp bên trong nhằm tăng tốc độ xử lý, có các tốc độ như 500, 550, 600, 650, 700, 750, 800, 850MHz (bus 100MHz), 533, 600, 667, 733, 800, 866, 933, 1000, 1100 và 1133MHz (bus 133MHz). Tualatin áp dụng công nghệ 0,13 micromet có 28,1 triệu transistor, bộ nhớ đệm L1 32KB, L2 256KB hoặc 512KB tích hợp bên trong BXL, bus hệ thống 133MHz. Có các tốc độ như 1133, 1200, 1266,

1333, 1400MHz. Celeron Coppermine (năm 2000) được "rút gọn" từ kiến trúc Pentium III Coppermine, còn gọi là Celeron II, được bổ sung 70 lệnh SSE. Sử dụng công nghệ 0,18 micromet có 28,1 triệu transistor, bộ nhớ đệm L1 32KB, L2 256KB tích hợp bên trong BXL, có các tốc độ như 533, 566, 600, 633, 667, 700, 733, 766, 800MHz (bus 66MHz), 850, 900, 950, 1000, 1100, 1200, 1300MHz (bus 100MHz).

Tualatin Celeron (Celeron S) (năm 2000) được "rút gọn" từ kiến trúc Pentium III Tualatin, áp dụng công nghệ 0,13 micromet, bộ nhớ đệm L1 32KB, L2 256KB tích hợp, bus hệ thống 100MHz, gồm các tốc độ 1,0; 1,1; 1,2; 1,3 và 1,4GHz.

- Pentium IV là bộ vi xử lý thế hệ thứ 7 dòng x86 phổ thông, được giới thiệu vào tháng 11 năm 2000. Pentium IV sử dụng vi kiến trúc NetBurst có thiết kế hoàn toàn mới so với các bộ vi xử lý cũ (PII, PIII và Celeron sử dụng vi kiến trúc P6). Một số công nghệ nổi bật được áp dụng trong vi kiến trúc NetBurst như Hyper Pipelined Technology – mở rộng số hàng đợi lệnh xử lý, Execution Trace Cache – tránh tình trạng lệnh bị chậm trễ khi chuyển từ bộ nhớ đến CPU, Rapid Execution Engine – tăng tốc bộ đồng xử lý toán học, bus hệ thống (system bus) 400MHz và 533MHz; các công nghệ Advanced Transfer Cache, Advanced Dynamic Execution, Enhanced Floating Point và Multimedia Unit, Streaming SIMD Extensions 2 (SSE2) cũng được cải tiến nhằm tạo ra những bộ vi xử lý tốc độ cao hơn, khả năng tính toán mạnh hơn, xử lý đa phương tiện tốt hơn. Pentium IV đầu tiên (tên mã Willamette) xuất hiện cuối năm 2000 đặt dấu chấm hết cho "triều đại" Pentium III. Willamette sản xuất trên công nghệ 0,18 micromet, có 42 triệu transistor (nhiều hơn gần 50% so với Pentium III), bus hệ thống (system bus) 400MHz, tích hợp bộ nhớ đệm L2 256KB. Pentium IV Willamette có một số tốc độ như 1,3; 1,4; 1,5; 1,6; 1,7; 1,8; 1,9; 2,0GHz. (Chú ý: tần số xung thực của Pentium IV là 100MHz nhưng với công nghệ Quad Data Rate cho phép truyền 4 bit dữ liệu trong 1 chu kỳ, nên bus hệ thống của nó là 400MHz).

Pentium IV Northwood xuất hiện vào tháng 1 năm 2002, được sản xuất trên công nghệ 0,13 micromet, có khoảng 55 triệu transistor, bộ nhớ đệm tích hợp L2 512KB. Northwood có 3 dòng gồm Northwood A (system bus 400MHz), tốc độ 1,6; 1,8; 2,0; 2,2; 2,4; 2,5; 2,6 và 2,8GHz. Northwood B (system bus 533MHz) tốc độ 2,26; 2,4; 2,53; 2,66; 2,8 và 3,06GHz (riêng phiên bản 3,06GHz có hỗ trợ công nghệ siêu phân luồng Hyper Threading – HT). Northwood C (system bus 800MHz, tất cả hỗ trợ HT), gồm 2,4; 2,6; 2,8; 3,0; 3,2; 3,4GHz. Pentium IV Prescott (năm 2004) là bộ vi xử lý đầu tiên của Intel sản xuất theo công nghệ 90 nanomet, kích thước vi mạch

giảm 50% so với Pentium IV Willamette. Điều này cho phép tích hợp nhiều transistor hơn trên cùng kích thước (125 triệu transistor so với 55 triệu transistor của Pentium IV Northwood), tốc độ chuyển đổi của transistor nhanh hơn, tăng khả năng xử lý, tính toán. Dung lượng bộ nhớ đệm tích hợp L2 của Pentium IV Prescott gấp đôi so với Pentium IV Northwood (1MB so với 512KB). Ngoài tập lệnh MMX, SSE, SSE2, Prescott được bổ sung tập lệnh SSE3 giúp các ứng dụng xử lý video và game chạy nhanh hơn.

Các bộ vi xử lý Celeron Pentium IV được thiết kế với mục tiêu dung hoà giữa công nghệ và giá cả, đáp ứng các yêu cầu phổ thông như truy cập Internet, email, chat, xử lý các ứng dụng văn phòng. Celeron Willamette 128 (2002), bản "rút gọn" từ Pentium IV Willamette, sản xuất trên công nghệ 0,18 micromet, bộ nhớ đệm L2 là 128KB, bus hệ thống 400MHz. Celeron Willamette 128 hỗ trợ tập lệnh MMX, SSE, SSE2. Một số BXL thuộc dòng này như Celeron 1.7 (1,7GHz) và Celeron 1.8 (1,8GHz).

Celeron NorthWood 128, "rút gọn" từ Pentium IV Northwood, công nghệ 0,13 micromet, bộ nhớ đệm tích hợp L2 128KB, bus hệ thống 400MHz. Celeron NorthWood 128 cũng hỗ trợ các tập lệnh MMX, SSE, SSE2, gồm Celeron 1.8A, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8 tương ứng với các tốc độ từ 1,8GHz đến 2,8GHz.

Celeron D (Presscott 256) được xây dựng từ nền tảng Pentium IV Prescott, sản xuất trên công nghệ 90 nanomet, bộ nhớ đệm tích hợp L2 256KB (gấp đôi dòng Celeron NorthWood), bus hệ thống 533MHz, Socket 478 và 775LGA. Ngoài các tập lệnh MMX, SSE, SSE2, Celeron D hỗ trợ tập lệnh SSE3, một số phiên bản sau có hỗ trợ tính toán 64 bit. Celeron D gồm 310, 315, 320, 325, 325J, 326, 330, 330J, 331, 335, 335J, 336, 340, 340J, 341, 345, 345J, 346, 350, 351, 355 với các tốc độ tương ứng từ 2,13GHz đến 3,33GHz.

Pentium IV Extreme Edition (P4EE) xuất hiện vào tháng 9 năm 2003, được xây dựng từ bộ xử lý Xeon dành cho máy chủ và trạm làm việc. Ngoài công nghệ HT "đình đám" thời bấy giờ, điểm nổi bật của P4EE là bổ sung bộ nhớ đệm L3 2MB. Phiên bản đầu tiên của P4 EE (nhân Gallatin) sản xuất trên công nghệ 0,13 micromet, bộ nhớ đệm L2 512KB, L3 2MB, bus hệ thống 800MHz, gồm P4 EE 3.2 (3,2GHz), P4 EE 3.4 (3,4GHz).

2.2.2. Các bộ vi xử lý kiến trúc NetBurst và Core của Intel

Vi kiến trúc NetBurst 64bit (Extended Memory 64 Technology – EM64T) đầu tiên được Intel sử dụng trong bộ xử lý Pentium IV Prescott

(tên mã Prescott 2M). Prescott 2M cũng sử dụng công nghệ 90nm, bộ nhớ đệm L2 2MB, bus hệ thống 800MHz, socket 775LGA. Ngoài các tập lệnh MX, SSE, SSE2, SSE3, công nghệ HT và khả năng tính toán 64 bit, Prescott 2M (trừ BXL 620) có hỗ trợ công nghệ Enhanced SpeedStep để tối ưu tốc độ làm việc nhằm tiết kiệm điện năng. Các bộ xử lý 6x2 có thêm công nghệ ảo hoá (Virtualization Technology). Prescott 2M có một số tốc độ như P4 HT 620 (2,8GHz), 630 (3,0GHz), 640 (3,2GHz), 650 (3,4GHz), 660, 662 (3,6GHz) và 670, 672 (3,8GHz).

Prescott Cedar Mill (năm 2006) hỗ trợ các tập lệnh và tính năng tương tự Prescott 2M nhưng không tích hợp Virtualization Technology. Cedar Mill được sản xuất trên công nghệ 65nm nên tiêu thụ điện năng thấp hơn, toả nhiệt ít hơn các dòng trước, gồm 631 (3,0GHz), 641 (3,2GHz), 651 (3,4GHz) và 661 (3,6GHz).

– Pentium D (năm 2005) mã Smithfield 8xx, là bộ xử lý lõi kép (dual core) đầu tiên của Intel, được cải tiến từ Pentium IV Prescott nên cũng gặp một số hạn chế như hiện tượng thắt cổ chai do băng thông ở mức 800MHz (400MHz cho mỗi lõi), điện năng tiêu thụ cao, toả nhiều nhiệt. Smithfield được sản xuất trên công nghệ 90nm, có 230 triệu transistor, bộ nhớ đệm L2 2MB (2x1MB, mỗi lõi dùng riêng 1MB), bus hệ thống 533MHz (Smithfield 805) hoặc 800MHz, socket 775LGA. Ngoài các tập lệnh MMX, SSE, SSE2, SSE3, Smithfield được trang bị tập lệnh mở rộng EMT64 hỗ trợ đánh địa chỉ nhớ 64 bit, công nghệ Enhanced SpeedStep (830, 840). Một số bộ xử lý thuộc dòng này như Pentium D 805 (2,66GHz), 820 (2,8GHz), 830 (3,0GHz), 840 (3,2GHz).

Cũng sử dụng vi kiến trúc NetBurst, Pentium D mã Presler 9xx được Intel thiết kế mới trên công nghệ 65nm, 376 triệu transistor, bộ nhớ đệm L2 4MB (2x2MB), hiệu năng cao hơn, nhiều tính năng mới và ít tốn điện năng hơn Smithfield. Pentium D 915 và 920 có tốc độ 2,8GHz, 925 và 930 (3,0GHz), 935 và 940 (3,2GHz), 945 và 950 (3,4GHz), 960 (3,6GHz). Presler dòng 9x0 có hỗ trợ công nghệ ảo hoá Virtualization Technology.

– Pentium Extreme Edition (năm 2005) là bộ xử lý lõi kép dành cho game thủ và người dùng cao cấp. Pentium EE sử dụng nhân Smithfield, Presler của Pentium D trong đó Pentium EE Smithfield sử dụng công nghệ 90nm, bộ nhớ đệm L2 được mở rộng đến 2MB (2×1MB), hỗ trợ tập lệnh MMX, SSE, SSE2, SSE3, công nghệ HT, Enhanced Intel SpeedStep Technology (EIST) và EM64T. Pentium 840 EE (3,20GHz, bus hệ thống 800MHz, socket 775LGA) là một trong những BXL thuộc dòng này. Pentium EE Presler sử dụng công nghệ 65nm, bộ nhớ đệm L2 được mở rộng

đến 4MB (2×2MB), hỗ trợ tập lệnh MMX, SSE, SSE2, SSE3, công nghệ HT, Enhanced Intel SpeedStep Technology (EIST), EM64T và Virtualization Technology. Một số bộ xử lý thuộc dòng này là Pentium EE 955 (3,46GHz) và Pentium EE 965 (3,73GHz) có bus hệ thống 1066MHz, Socket 775.

– Các bộ xử lý 64 bit, kiến trúc Core: Tại diễn đàn IDF đầu năm 2006, Intel đã giới thiệu kiến trúc Intel Core với 5 cải tiến quan trọng là khả năng mở rộng thực thi động (Wide Dynamic Execution), tính năng quản lý điện năng thông minh (Intelligent Power Capability), chia sẻ bộ nhớ đệm linh hoạt (Advanced Smart Cache), truy xuất bộ nhớ thông minh (Smart Memory Access) và tăng tốc phương tiện số tiên tiến (Advanced Digital Media Boost). Những cải tiến này sẽ tạo ra những BXL mạnh hơn, khả năng tính toán nhanh hơn và giảm mức tiêu thụ điện năng, toả nhiệt ít hơn so với kiến trúc NetBurst.

Intel Core 2 Duo là bộ xử lý lõi kép sản xuất trên công nghệ 65nm, hỗ trợ SIMD instructions, công nghệ Virtualization Technology cho phép chạy cùng lúc nhiều hệ điều hành, tăng cường bảo vệ hệ thống trước sự tấn công của virus (Execute Disable Bit), tối ưu tốc độ nhằm tiết kiệm điện năng (Enhanced Intel SpeedStep Technology), quản lý máy tính từ xa (Intel Active Management Technology). Ngoài ra, còn hỗ trợ các tập lệnh MMX, SSE, SSE2, SSE3, SSSE3. Core 2 Duo (tên mã Conroe) có 291 triệu transistor, bộ nhớ đệm L2 4MB, bus hệ thống 1066MHz. Một số bộ xử lý thuộc dòng này là E6600 (2,4GHz), E6700 (2,66GHz). Core 2 Duo (tên mã Allendale) E6300 (1,86GHz), E6400 (2,13GHz) có 167 triệu transistor, bộ nhớ đệm L2 2MB, bus hệ thống 1066MHz.

Core 2 Extreme là bộ xử lý lõi kép dành cho game thủ sử dụng kiến trúc Core, có nhiều đặc điểm giống với bộ xử lý Core 2 như công nghệ sản xuất 65nm, hỗ trợ các công nghệ mới Enhanced Intel SpeedStep Technology, Intel x86–64, Execute Disable Bit, Intel Active Management, Virtualization Technology, Intel Trusted Execution Technology... các tập lệnh MMX, SSE, SSE2, SSE3, SSSE3. Core 2 Extreme (tên mã Conroe XE) xuất hiện tháng 7 năm 2006 với đại diện X6800 tốc độ 2,93GHz, bộ nhớ đệm L2 đến 4MB, bus hệ thống 1066MHz, Socket 775LGA. Cuối năm 2006, con đường phía trước của BXL tiếp tục rộng mở khi Intel giới thiệu các bộ xử lý bốn nhân (quad core) như Core 2 Extreme QX6700, Core 2 Quad Q6300, Q6400, Q6600.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 2

1 . Một bộ vi xử lý có độ rộng bus địa chỉ là m = 32 có thể quản lý bộ nhớ với dung lượng tối đa là bao nhiêu?

2 . Tìm địa chỉ vật lý tạo bởi cặp thanh ghi CS:IP biết nội dung CS là 001FH; nội dung IP là 0006H.

3 . Cho địa chỉ logic của một ô nhớ là 02F0H:0006H. Tìm địa chỉ vật lý tương ứng.

4 . Nêu sự khác nhau cơ bản giữa ngắt không che được và ngắt che được.

5 . Giải thích cơ chế hoạt động của công nghệ HT.

6 . Sử dụng vi mạch chốt dữ liệu 74HC573, vẽ mạch tách địa chỉ ra khỏi bus đa hợp địa chỉ/dữ liệu.

7 . Nêu ý nghĩa các chân INTR, TEST, HOLD, HLDA của 8086.

Chương 3. LẬP TRÌNH HỢP NGỮ

3.1. TỔNG QUAN VỀ LẬP TRÌNH HỢP NGỮ

Lập trình hợp ngữ (Assembly) là một ngôn ngữ lập trình bậc thấp, ngôn ngữ này sử dụng chính các câu lệnh trong tập lệnh của bộ vi xử lý tương ứng để viết nên một chương trình. Lập trình hợp ngữ có thể tìm thấy trong các chương trình viết cho các hệ thống đo lường, điều khiển sử dụng các bộ vi xử lý nhỏ.

Ưu điểm của lập trình hợp ngữ là tính hiệu quả vì chương trình viết bằng hợp ngữ chạy nhanh và chiếm dung lượng bộ nhớ nhỏ hơn so với các chương trình viết bằng các ngôn ngữ bậc cao do trình dịch chỉ cần qua một bước là có thể biên dịch các chương trình này ra mã máy, và lập trình hợp ngữ cho phép đọc hoặc ghi trực tiếp vào các ô nhớ, thanh ghi hay các cổng vào/ra một cách dễ dàng.

Người lập trình thường sử dụng ngôn ngữ bậc cao để viết chương trình hơn là sử dụng hợp ngữ vì một chương trình viết bằng hợp ngữ dài nên khó kiểm soát lỗi, khó bảo trì hơn so với một chương trình tương tự viết bằng ngôn ngữ bậc cao. Chương trình viết bằng hợp ngữ chỉ thực thi được trên hệ thống máy tương ứng, không thể thực thi trên hệ thống máy có kiến trúc và tập lệnh khác.

Hiện nay, rất nhiều ngôn ngữ bậc cao cho phép nhúng các chương trình con bằng hợp ngữ. Chỉ khi nghiên cứu hợp ngữ, bạn mới thực sự hiểu được máy tính.

Chương này sẽ giới thiệu lập trình hợp ngữ cho máy IBM PC hoặc tương thích với máy IBM PC vì máy IBM PC có cấu trúc khá tiêu biểu của một hệ vi xử lý. Sau khi đọc xong chương này bạn có thể dễ dàng nghiên cứu hợp ngữ trên các loại máy khác.

3.1.1. Cú pháp của hợp ngữ

Khi sử dụng bất kỳ một ngôn ngữ lập trình nào ta đều phải nắm được cú pháp của ngôn ngữ đó. Các chương trình viết bằng hợp ngữ nếu viết đúng cú pháp sẽ được trình dịch hợp ngữ (MASM) dịch ra mã máy, do đó chúng phải được viết sao cho phù hợp với các khuôn mẫu của trình biên dịch. Hợp ngữ không phân biệt chữ hoa, chữ thường.

Một chương trình hợp ngữ bao gồm các dòng lệnh, mỗi lệnh được viết trên một dòng. Một dòng lệnh có thể là lệnh thật hoặc hướng dẫn biên dịch.

Lệnh thật sẽ được hợp ngữ dịch ra mã máy còn hướng dẫn biên dịch thì không được dịch ra mã máy mà để chỉ dẫn cho trình dịch thực hiện công việc, chẳng hạn dành chỗ cho một biến nhớ hay khai báo một chương trình con.

Mỗi một lệnh bao gồm có bốn trường:

```
Tên: Mã_lệnh Toán_hạng; Chú_giải
```

Các trường phải được cách nhau ít nhất một ký tự trống hoặc TAB. Một lệnh không nhất thiết phải có đầy đủ cả bốn trường, ví dụ trường tên hoặc trường chú giải có thể bỏ qua nhưng các trường phải xuất hiện theo đúng theo thứ tự như trên.

Ví dụ một lệnh:

```
NHAP: MOV AX,0; Chuyển 0 vào thanh ghi AX
```

Trong ví dụ này, trường tên là NHAP, mã lệnh là MOV (lệnh dịch chuyển dữ liệu), toán hạng là AX và 0, chú giải là Chuyển 0 vào thanh ghi AX.

3.1.1.1. Trường tên

Trường tên được sử dụng làm nhãn lệnh, tên các thủ tục và tên biến. Tên sẽ được chương trình dịch gán bằng các địa chỉ cụ thể của ô nhớ. Các quy định khi đặt tên:

- Không được dài quá 31 ký tự.
- Bao gồm các chữ cái, chữ số và một số ký tự đặc biệt (! @ % _ $).
- Không được đứng đầu bằng một chữ số.
- Không chứa khoảng trống hoặc dấu gạch ngang.
- Nếu có dấu (.) thì phải đặt ở vị trí đầu tiên của tên.
- Không phân biệt chữ thường, chữ hoa.

Trường tên cách với trường mã lệnh bởi dấu (:)

Ví dụ một số tên hợp lệ:

```
BAITAP1
BAITAP_1
@BAITAP
$100
.BAITAP
BAI?
```

Ví dụ một số tên không hợp lệ:

```
BAI TAP          ; Chứa khoảng trống
BAITAP-1         ; Chứa dấu gạch ngang
1BAI             ; Bắt đầu bằng một chữ số
BAI.1            ; Dấu chấm không đứng đầu
BAI&1            ; Chứa một ký tự không hợp lệ
```

3.1.1.2. Trường mã lệnh

Trường mã lệnh là các ký hiệu gợi nhớ biểu thị chức năng thao tác như MOV, ADD, SHL và được dịch ra mã máy.

3.1.1.3. Trường toán hạng

Trong mỗi lệnh, toán hạng xác định dữ liệu sẽ được các lệnh tác động lên. Một lệnh có thể không có, có một hoặc có hai toán hạng. *Ví dụ:*

```
NOP              ; Không có toán hạng nào
DEC CX           ; Có một toán hạng là CX
MOV AX,1         ; Có hai toán hạng là AX và 1
```

Chú ý: Trong lệnh có hai toán hạng thì hai toán hạng phải cách nhau bởi dấu phẩy (,). Toán hạng đầu là toán hạng đích, nơi chứa kết quả, toán hạng thứ hai là toán hạng nguồn, các lệnh thường không làm thay đổi toán hạng nguồn. Các toán hạng của lệnh phải có cùng độ lớn hay cùng kiểu thì mới hợp lệ. *Ví dụ:*

```
MOV AX,BX        ; Hai toán hạng có cùng kiểu
MOV AX,BL        ; Hai toán hạng không cùng kiểu
```

3.1.1.4. Trường chú giải

Trường chú giải được sử dụng để giải thích mục đích sử dụng lệnh của chương trình giúp người đọc dễ hiểu. Lời giải thích cách với trường toán hạng dấu chấm phẩy (;). Chương trình dịch bỏ qua khi dịch đến dấu chấm phẩy, vì thế người ta hay sử dụng dấu này để bỏ qua một dòng lệnh nào đó như:

```
; MOV CX,DX
```

Khi viết chương trình ta không nên đưa ra một lời chú giải như:

```
MOV CX,5         ; Chuyển 5 vào thanh ghi CX
```

Giả sử muốn khởi tạo cho một vòng lặp thực hiện năm lần, ta chú giải như sau:

```
MOV CX,5          ; Khởi tạo đếm bằng 5
```

3.1.2. Dữ liệu chương trình

Trong một chương trình hợp ngữ, dữ liệu có thể biểu diễn dưới dạng nhị phân, thập phân, hexa hoặc ký tự. Chương trình dịch sẽ dịch tất cả các dạng dữ liệu khác nhau thành các số nhị phân. Dữ liệu ở hệ nào thì phải được viết kèm đuôi ở hệ đó.

Một số thuộc hệ nhị phân là một chuỗi các bit kết thúc bằng chữ 'B' hay 'b'.

Ví dụ: 10110100B.

Một số thuộc hệ hexa là một chuỗi các số bắt đầu là một số thập phân và kết thúc bằng chữ 'H' hay 'h'. Ví dụ: 0AH hoặc 12C5H. Nếu một số bắt đầu bằng một chữ cái từ A đến F hay từ a đến f thì phải thêm 0 vào trước số đó để chương trình dịch không nhầm sang một tên hay một nhãn.

Riêng đối với một số thuộc hệ thập phân có thể kết thúc số đó bằng chữ 'D' hay 'd' (hoặc không có). Ví dụ: 12 hay 12D là như nhau.

Các ký tự hoặc chuỗi ký tự phải được đặt trong dấu nháy đơn hoặc nháy kép. Ví dụ: "a", "Chao ban". Các ký tự sẽ được dịch ra mã ASCII của chúng, chương trình dịch sẽ không phân biệt giữa "A" với 41H, 1000001B hoặc 65.

3.1.3. Biến và hằng

Cũng giống như các ngôn ngữ bậc cao, trong hợp ngữ mỗi biến có một kiểu dữ liệu và được chương trình gán cho một địa chỉ nhất định trong bộ nhớ. Để định nghĩa các kiểu dữ liệu ta thường dùng các lệnh giả được liệt kê dưới đây. Mỗi toán tử giả có thể dùng để thiết lập một hay nhiều dữ liệu của kiểu đã được đưa ra.

DB	Định nghĩa 1 byte
DW	Định nghĩa 1 từ
DD	Định nghĩa từ kép
DQ	Định nghĩa 4 từ
DT	Định nghĩa 10 byte

3.1.3.1. Biến kiểu byte

Một biến kiểu byte được định nghĩa như sau:

```
Tên DB giá_trị_khởi_tạo
```

Ví dụ:
```
X1  DB      0AH
```

Ví dụ trên định nghĩa biến byte có tên là X1 và dành 1 byte trong bộ nhớ để chứa giá trị của biến, biến có giá trị khởi đầu là 0AH.

Nếu đặt dấu chấm hỏi (?) ở vị trí giá trị khởi tạo của biến X2 thì biến X2 cũng được dành một byte trong bộ nhớ nhưng không được gán giá trị khởi đầu.

Giá trị khởi tạo cho biến kiểu byte từ –128 đến 127 với kiểu có dấu và từ 0 đến 255 với kiểu không dấu.

3.1.3.2. Biến kiểu từ

Khi khai báo, một biến kiểu từ sẽ được dành hai byte nhớ để chứa giá trị của biến.

```
Tên DB giá_trị_khởi_tạo
```

Ví dụ:
```
W1      DW      0FFFH
W2      DW       ?
```

Ví dụ trên định nghĩa một biến kiểu từ có tên là W1 và giá trị khởi đầu là 0FFFH.

Nếu đặt dấu chấm hỏi (?) ở vị trí giá trị khởi tạo của biến W2 thì biến đó không được gán giá trị khởi đầu. Giá trị khởi tạo cho biến kiểu từ trong khoảng – 32768 đến 32767 với kiểu có dấu và trong khoảng 0 đến 65535 đối với kiểu không dấu.

3.1.3.3. Biến kiểu mảng

Biến mảng là một tập hợp nhiều phần tử có cùng một kiểu giá trị. Trong lập trình hợp ngữ, mảng chỉ là chuỗi các byte nhớ hoặc các từ nhớ.

Ví dụ để định nghĩa một mảng bốn byte có tên là MANG và có giá trị lần lượt là 0, 1, 2, 3 ta khai báo như sau:

```
MANG      DB    0, 1, 2, 3
```

Trong ví dụ này, tên MANG được gán cho byte đầu tiên có giá trị là 0, MANG+1 được gán cho byte thứ hai có giá trị là 1, MANG+2 được gán cho byte thứ ba có giá trị là 2 và MANG+3 được gán cho byte thứ tư có giá trị là 3.

Một mảng kiểu từ cũng có thể được định nghĩa tương tự, *ví dụ:*

```
MANG      DW   0, 1, 2, 3
```

Nếu phần tử đầu tiên của mảng bắt đầu tại địa chỉ 0200H trong bộ nhớ thì các phần tử còn lại của mảng nằm trong các ô nhớ có địa chỉ như sau:

Phần tử	Địa chỉ	Giá trị (nội dung)
MANG	0200H	0
MANG+1	0202H	1
MANG+2	0204H	2
MANG+3	0206H	3

Khi chúng ta muốn khởi đầu cho các phần tử của mảng cùng một giá trị ta dùng toán tử DUP. *Ví dụ:* `M1 DW 100 DUP(0)`

Ví dụ trên định nghĩa một biến mảng có tên là M1 được dành 200 byte trong bộ nhớ để chứa 100 phần tử, mỗi phần tử của mảng được khởi tạo giá trị đầu là 0.

Biến kiểu chuỗi ký tự có thể coi là một trường hợp đặc biệt của biến mảng. Một chuỗi ký tự cũng có thể được khởi tạo bằng một biến mảng.

Ví dụ: `M2 DB 'Hello'`

tương đương với: `M2 DB 48H,65H,6CH,6CH,6FH`

3.1.3.4. Hằng có tên

Các hằng trong hợp ngữ có thể là kiểu số hay kiểu ký tự và thường được gán tên. Khi gán tên cho hằng người ta sử dụng toán tử giả EQU (equate), cú pháp như sau:

```
        Tên EQU hằng_số
Ví dụ:  LF  EQU  0AH
        CR  EQU  0DH
```

Trong ví dụ trên, mã ASCII của ký tự xuống dòng (0AH) và ký tự trở về đầu dòng (0DH) được gán tên là LF và CR. Hằng số cũng có thể là một chuỗi ký tự.

Ví dụ hai lệnh sau: `TRUYEN EQU "ME VANG NHA"`

`TR DB TRUYEN`

tương đương với lệnh: `TR DB "ME VANG NHA"`

3.1.4. Khung của một chương trình hợp ngữ

Một chương trình hợp ngữ dịch ra đuôi *.exe có dạng như sau:

```
.model  kiểu_bộ_nhớ
.stack dung_lượng_ngăn_xếp
.data
 ; khai báo các hằng, biến số
.code
main proc
; các lệnh chương trình chính
main endp
; các hàm và thủ tục
end main
```

3.1.4.1. Khai báo kiểu bộ nhớ (.model)

Các chương trình khác nhau đòi hỏi có kích thước bộ nhớ khác nhau. Một chương trình dài sẽ chiếm dung lượng bộ nhớ lớn còn chương trình ngắn sẽ chiếm dung lượng bộ nhớ nhỏ. Bộ nhớ thường được tổ chức thành các vùng nhớ khác nhau để chứa dữ liệu, chứa mã lệnh và vùng nhớ dùng làm ngăn xếp. Bảng 3.1 mô tả đầy đủ các kiểu bộ nhớ:

Bảng 3.1. Mô tả khai báo các kiểu bộ nhớ

Kiểu kích thước	Mô tả
Tiny	Mã lệnh và dữ liệu nằm trong một đoạn
Small	Mã lệnh nằm trong một đoạn, dữ liệu nằm trong một đoạn
Medium	Mã lệnh không nằm trong một đoạn, dữ liệu nằm trong một đoạn
Compact	Mã lệnh nằm trong một đoạn, dữ liệu không nằm trong một đoạn
Large	Mã lệnh không nằm trong một đoạn, dữ liệu không nằm trong một đoạn, không có mảng nào lớn hơn 64KB.
Huge	Mã lệnh không nằm trong một đoạn, dữ liệu không nằm trong một đoạn, các mảng có thể lớn hơn 64KB.

3.1.4.2. Khai báo kích thước ngăn xếp (.stack)

Kích thước khi khai báo ngăn xếp sẽ quyết định số byte dành cho ngăn xếp.

Ví dụ: Nếu khai báo: `.stack  100H` ta sẽ có 256 byte dành cho ngăn xếp. Nếu không khai báo thì chương trình sẽ tự gán cho ngăn xếp dung lượng là 1Kbyte.

3.1.4.3. Khai báo đoạn dữ liệu (.data)

Đoạn dữ liệu chứa toàn bộ định nghĩa cho các biến số, hằng số của chương trình. Ví dụ ta có thể khai báo các biến trong đoạn dữ liệu như sau:

```
.data
   M2     DB     'Hello!$'
   LF     EQU    0AH
   CR     EQU    0DH
```

3.1.4.4. Khai báo đoạn mã (.code)

Đoạn mã chứa toàn bộ các câu lệnh của chương trình chính và các chương trình con. Các lệnh trong đoạn mã phải được tổ chức một cách hợp lý, đúng cú pháp và được chương trình dịch chuyển sang ngôn ngữ máy.

Dưới đây là ví dụ về một chương trình hiển thị dòng chữ "Hello world, I'm learning Assembly !!!" lên màn hình:

```
.model small
.stack 100H
.data
message db "Hello world, I'm learning Assembly !!!$"
.code
main  proc
      mov    AX,@data
      mov    DS,AX
      mov    AH,9
      mov    DX,message
      int    21H
      mov    AH,4CH
      int    21H
main  endp
end   main
```

3.1.5. Cách tạo và chạy một chương trình hợp ngữ

Để tạo và chạy một chương trình hợp ngữ trên máy IBM PC ta thực hiện các bước như sau:

Bước 1: Soạn thảo chương trình sử dụng các phần mềm soạn thảo: NCedit, SK hoặc các chức năng soạn thảo của Turbo Pascal để tạo ra một chương trình gốc. File chương trình này được gán đuôi .ASM.

Bước 2: Dùng trình dịch MASM để dịch file .ASM ra mã máy dưới dạng file .OBJ. Nếu trong bước dịch có lỗi cú pháp ta phải quay lại bước 1 để sửa lại chương trình gốc.

Bước 3: Dùng chương trình LINK để nối một hay nhiều tệp OBJ lại với nhau thành một tệp chương trình với đuôi .exe.

Bước 4: Cho chạy chương trình vừa dịch.

3.2. CÁC CHẾ ĐỘ ĐỊA CHỈ CỦA HỌ VI XỬ LÝ 80×86

Chế độ địa chỉ (addressing mode) là cách để vi xử lý tìm thấy toán hạng của lệnh khi hoạt động. Mỗi bộ vi xử lý có thể có nhiều chế độ địa chỉ, các chế độ địa chỉ đó được xác định ngay từ khi chế tạo bộ vi xử lý. Họ vi xử lý 80x86 có 7 chế độ địa chỉ.

3.2.1. Chế độ địa chỉ thanh ghi (Register Addressing Mode)

Chế độ địa chỉ thanh ghi là chế độ địa chỉ mà các toán hạng của lệnh là các thanh ghi chứa dữ liệu cần thao tác, vì thế khi thực hiện các lệnh ở chế độ này có thể đạt tốc độ truy cập nhanh hơn so với các lệnh truy cập tới bộ nhớ. *Ví dụ:*

```
MOV AX,BX ;Chuyển nội dung của BX vào AX
MOV DL,AL ;Chuyển nội dung của AL vào DL
MOV SI,DX ;Chuyển nội dung của DX vào SI
MOV SP,BP ;Chuyển nội dung của BP vào SP
ADD AL,BH ;Cộng nội dung của BH với AL,kết quả lưu trong AL
```

Trong các chế độ địa chỉ, toán hạng đích và nguồn phải có cùng độ lớn. Ví dụ lệnh sau đây toán hạng nguồn là một thanh ghi 16 bit, toán hạng đích là một thanh ghi 8 bit:

```
MOV CL,BX ;Lệnh này sẽ báo lỗi vì đích và nguồn không cùng kiểu
```

3.2.2. Chế độ địa chỉ tức thời (Immediate Addressing Mode)

Trong chế độ địa chỉ này, toán hạng đích là một thanh ghi hay ô nhớ, toán hạng nguồn là hằng số. Chế độ địa chỉ này cho phép nạp dữ liệu cần thao tác vào bất kỳ thanh ghi nào (trừ thanh ghi đoạn) hoặc ô nhớ nào nằm trong đoạn dữ liệu DS. *Ví dụ:*

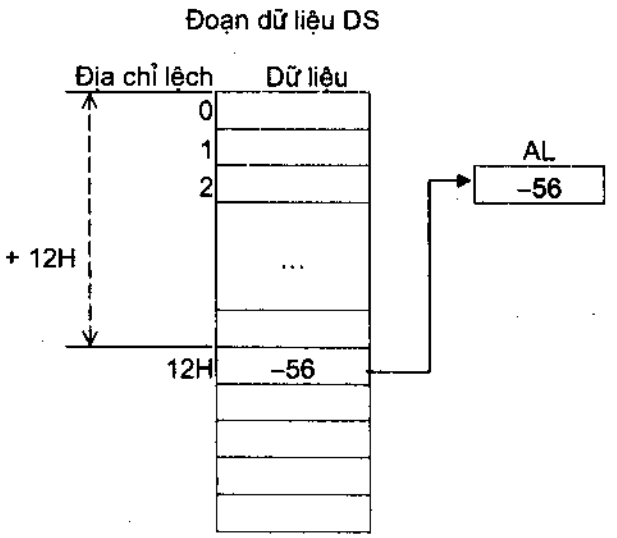
```
MOV AH,12 ;chuyển 12 vào thanh ghi AH
MOV DL,0AH ;chuyển 10 (0AH) vào thanh ghi DL
MOV AH,01000010B ; chuyển 01000010B (66) vào thanh ghi AH
MOV CL,'A' ; chuyển 'A' (65) vào thanh ghi CL
MOV [BX],100 ; chuyển 100 vào ô nhớ có địa chỉ DS:BX
```

Hằng số có thể được biểu diễn dưới dạng thập phân, nhị phân, hexa hoặc dưới dạng ký tự.

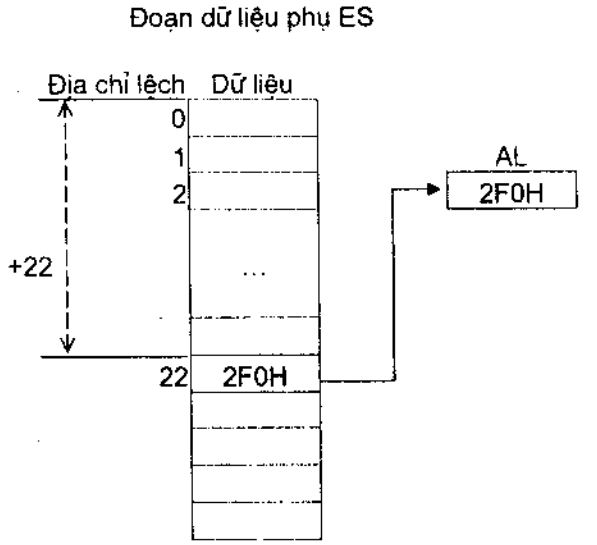
3.2.3. Chế độ địa chỉ trực tiếp (Direct Addressing Mode)

Chế độ địa chỉ trực tiếp là chế độ địa chỉ mà một toán hạng của lệnh là thanh ghi, toán hạng còn lại chứa địa chỉ lệch của ô nhớ chứa dữ liệu. *Ví dụ:*

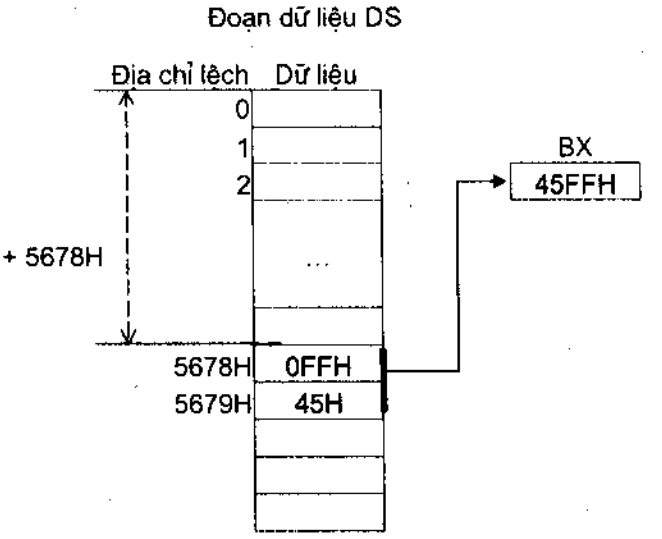
```
MOV AL, [012H] ;chuyển nội dung ô nhớ DS:[012H] vào AL
MOV AL, ES:[22] ;chuyển nội dung ô nhớ ES:[012H] vào AL
MOV [16H], AX ;chuyển nội dung của AX vào hai ô nhớ
              ;liên tiếp có địa chỉ DS:[16H] và DS:[17H]
MOV BX,[5678H] ;chuyển nội dung của hai ô nhớ có địa
               ;chỉ DS:[5678H] và DS:[5679h] vào BX
```



Hình 3.1. Mô tả lệnh MOV AL,[012H]



Hình 3.2. Mô tả lệnh MOV AL,ES:[22]



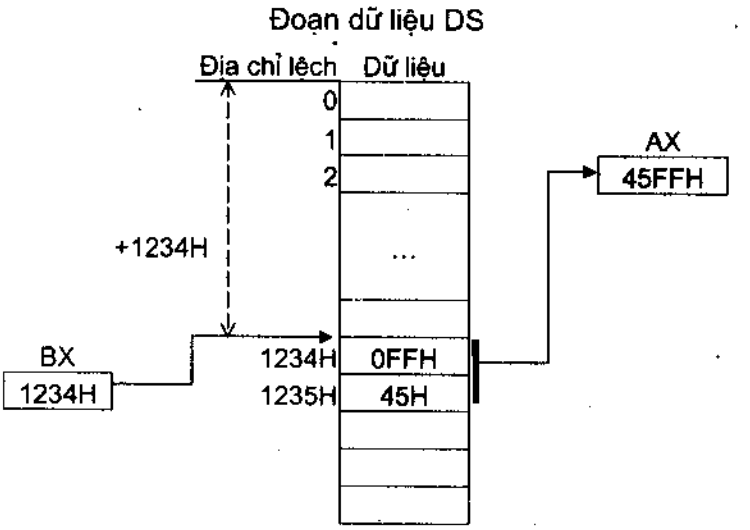
Hình 3.3. Mô tả lệnh MOV BX,[5678H]

3.2.4. Chế độ địa chỉ gián tiếp qua thanh ghi (Register Direct Addressing Mode)

Trong chế độ địa chỉ này một toán hạng phải là thanh ghi chứa dữ liệu, toán hạng kia là thanh ghi chứa địa chỉ lệch của ô nhớ chứa dữ liệu.

Ví dụ:
```
MOV AX,[BX] ; Chuyển nội dung của hai ô nhớ có địa
            ; chỉ DS:[BX] và DS:[BX+1] vào AX
```

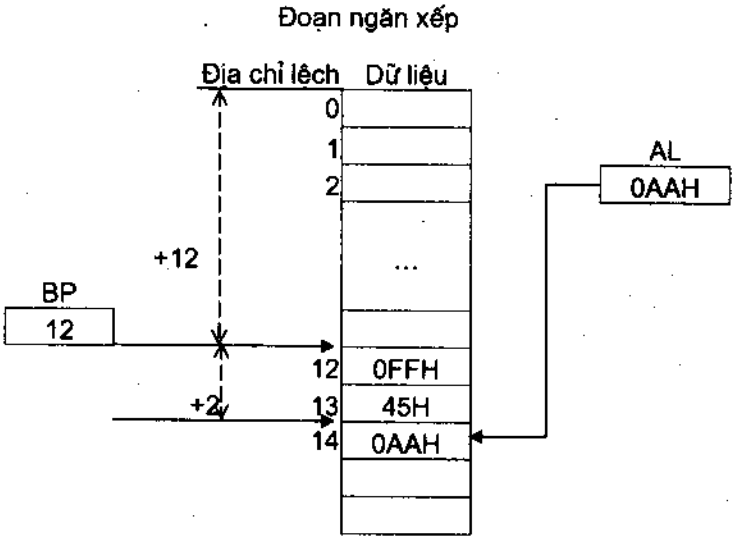
```
MOV  AH,[ BX] ;  Chuyển  nội  dung  của  ô  nhớ  có  địa  chỉ
              ;  DS:[ BX]  vào  AL
MOV  [ BP] ,AL;  Chuyển  nội  dung  của  AL  vào  ô  nhớ  có  địa
              ;  chỉ  SS:[ BP]
```



Hình 3.4. **Mô tả lệnh MOV AX,[BX]**

3.2.5. Chế độ địa chỉ tương đối cơ sở (Based Relative Addressing Mode)

Trong chế độ địa chỉ này một toán hạng là thanh ghi, toán hạng còn lại là địa chỉ lệch của ô nhớ chứa dữ liệu. Địa chỉ lệch này được lưu bởi một thanh ghi cơ sở BX hoặc BP và các hằng số biểu diễn giá trị dịch chuyển.



Hình 3.5. **Mô tả lệnh MOV [BP+2],AL**

Cú pháp lệnh:

```
MOV [ BP+2] ,AL ; Chuyển nội dung của AL vào ô nhớ có địa
                ; chỉ SS:[ BP+2]
MOV AX,[ BX+4] ; Chuyển nội dung của hai ô nhớ có địa chỉ
               ; DS:[ BX+4] và DS:[ BX+5] vào AX
```

Lệnh này có thể viết như sau: `MOV AX,4[ BX]`

`MOV AX,[ BX] +4`

3.2.6. Chế độ địa chỉ tương đối chỉ số (Indexed Relative Addressing Mode)

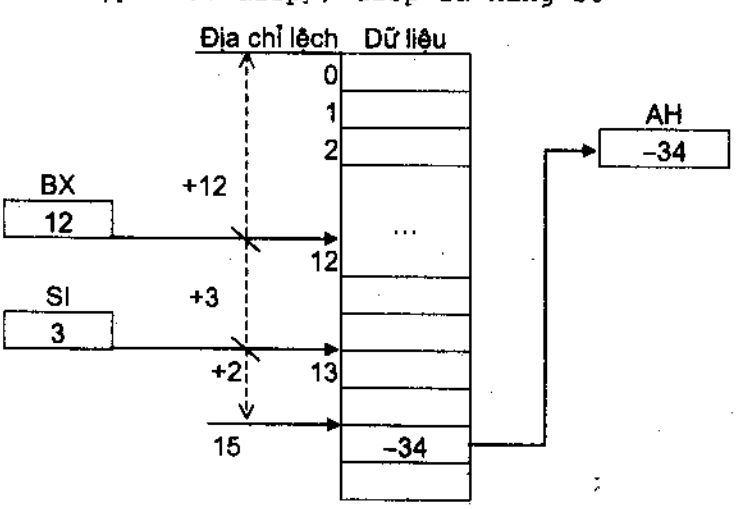
Trong chế độ địa chỉ này, một toán hạng là thanh ghi, toán hạng còn lại là địa chỉ lệch của ô nhớ chứa dữ liệu. Địa chỉ lệch này được lưu bởi một thanh ghi chỉ số SI hoặc DI và các hằng số biểu diễn giá trị dịch chuyển.

Ví dụ:
```
MOV [ SI+2] ,AL ; chuyển nội dung của AL vào ô nhớ
                ; có địa chỉ:[ BP+2]
MOV AX,[ DI+4] ; chuyển nội dung của hai ô nhớ có địa chỉ
               ; DS:[ BX+4] và DS:[ BX+5] vào AX
```

3.2.7. Chế độ địa chỉ tương đối chỉ số cơ sở (Based Indexed Relative Addressing Mode)

Trong chế độ địa chỉ này, một toán hạng là thanh ghi, toán hạng còn lại là địa chỉ lệch của ô nhớ chứa dữ liệu. Địa chỉ lệch này được lưu bởi một thanh ghi chỉ số và một thanh ghi cơ sở cộng thêm với các hằng số để biểu diễn giá trị dịch chuyển.

Ví dụ: `MOV AX,[ BX+SI+disp] ; disp là hằng số`



Hình 3.6. **Mô tả lệnh MOV AH,[BX+SI+2]**

3.3. MỘT SỐ HÀM CỦA NGẮT 21H

Phần này sẽ giới thiệu một số hàm cơ bản như: hàm nhập vào ký tự từ bàn phím, hàm hiển thị ký tự lên màn hình hoặc hiển thị chuỗi ký tự.

3.3.1. Lệnh INT

Lệnh ngắt được dùng để gọi các hàm của DOS và BIOS. Cú pháp lệnh ngắt như sau: `int N`

Trong đó N là số hiệu ngắt, ví dụ: `int 21H` là lệnh để gọi rất nhiều các hàm của DOS. Mỗi một hàm được thực hiện bằng cách đưa vào thanh ghi AH số hiệu của hàm sau đó gọi `int 21H`.

3.3.2. Hàm 1

Hàm 1 đợi đọc 1 ký tự từ thiết bị vào chuẩn (bàn phím), sau đó lưu mã ASCII của ký tự nhập vào trong AL. Hàm 1 được mô tả như sau:

Vào: AH = 1

Ra: AL = Mã ASCII của ký tự nhập vào.

AL = 0 nếu một phím điều khiển hay một phím chức năng được nhấn.

Ví dụ về một chương trình nhập vào một ký tự từ bàn phím:

```
.model small
.stack
.code
main proc
; nhập vào một ký tự
mov AH,1
int 21H
mov BL,AL ; cất mã ASCII của ký tự được nhập vào BL
; trở về DOS
mov AH,4CH
int 21H
main endp
end main
```

3.3.3. Hàm 4CH

Hàm 4CH kết thúc chương trình hiện tại và trả điều khiển cho chương trình gọi nó.

Vào: AH = 4CH

Ra: Không.

3.3.4. Hàm 2

Hàm hiển thị ký tự lên màn hình hoặc thi hành các chức năng điều khiển.

Vào: AH = 2

DL = Mã ASCII của ký tự muốn hiển thị hoặc điều khiển.

Ra: AL = Mã ASCII của ký tự vừa hiển thị hoặc điều khiển.

Muốn hiển thị một ký tự, ta phải đưa mã ASCII của nó trong DL, ví dụ để hiển thị chữ A lên màn hình ta dùng đoạn lệnh sau đây.

```
mov AH,2
mov DL,'A'
int 21H
```

Sau khi ký tự hiển thị, con trỏ sẽ dịch sang vị trí tiếp theo cạnh ký tự đó. Nếu thanh ghi DL chứa mã ASCII của ký tự điều khiển, hàm 2 sẽ thi hành các chức năng điều khiển. Dưới đây là mã ASCII của một số ký tự điều khiển quan trọng, chúng ta có thể tra thêm trong bảng mã ASCII.

Mã ASCII	Ký hiệu	Chức năng
7	Bel	Phát tiếng bíp
8	BS	Xoá lùi
9	HT	Tab
0AH	LF	Xuống dòng
0DH	CR	Về đầu dòng

Muốn phát ra tiếng bíp của máy tính ta sử dụng các lệnh sau:

```
mov AH,2
mov DL,7
int 21H
```

Ví dụ: Viết chương trình nhập vào một ký tự và hiển thị ký tự đó ở đầu dòng tiếp theo.

```
.model small
.stack 100H
.data
.code
main proc
;nhập vào 1 ký tự
mov AH,1                    ; hàm nhập một ký tự
int 21H                     ; thực hiện nhập
mov BL,AL                   ; cất ký tự vào BL
mov AH,2
mov DL,0AH                  ; thực hiện xuống dòng
int 21H
mov DL,0DH                  ; thực hiện về đầu dòng
int 21H
mov DL,BL                   ; lấy ký tự trong BL
int 21H                     ; và đặt vào DL
mov AH,4CH                  ; hiển thị ký tự đó
int 21H                     ;về DOS
main endp
end main
```

Sau khi nhập vào một ký tự từ bàn phím ta phải cất ký tự vào trong một thanh ghi khác vì AL sẽ chứa mã ASCII của ký tự điều khiển hoặc ký tự hiển thị sau khi sử dụng hàm 2.

3.3.5. Hàm 9

Hàm 9 hiển thị chuỗi ký tự lên màn hình.

Vào: AH = 9

DS:DX = địa chỉ của chuỗi ký tự.

Ra: Không.

Hàm 9 yêu cầu DS:DX chứa địa chỉ của chuỗi ký tự. Để thực hiện điều này chúng ta sẽ dùng lệnh sau:

```
LEA đích, nguồn
```

Trong đó, đích là thanh ghi công dụng chung, nguồn là một ô nhớ. Lệnh LEA lấy địa chỉ và chép từ nguồn sang đích. Ví dụ để hiển thị chuỗi có tên MANG lên màn hình ta dùng các lệnh:

```
mov AH,9
LEA DX,MANG
int 21H
```

Lệnh `LEA DX,MANG` sẽ nhập địa chỉ tương đối của biến `MANG` vào `DX`. Ký tự '$' đánh dấu kết thúc chuỗi ký tự và nó không được hiển thị. Nếu chuỗi ký tự chứa mã ASCII của ký tự điều khiển thì chức năng điều khiển sẽ được thi hành. Để chạy đúng, một chương trình có chứa đoạn dữ liệu sẽ bắt đầu với hai lệnh sau:

```
mov AX,@data
mov DS,AX
```

`@data` là tên đoạn dữ liệu được định nghĩa bởi đoạn `.data` chương trình dịch sẽ dịch `@data` thành địa chỉ.

Ví dụ sau đây sẽ cho phép nhập vào một ký tự thường sau đó đổi thành chữ hoa và hiển thị ở dòng dưới với hai dòng thông báo:

```
"moi ban nhap vao mot ky tu thuong: a"
"doi thanh chu hoa va hien thi: A"
```

Khi khai báo chuỗi thứ hai ta dùng mã ASCII của ký tự xuống dòng và về đầu dòng để khi chuỗi hai được hiển thị bằng hàm 9 thì chức năng điều khiển này sẽ được thi hành.

```
.model small
.stack 100H
.data
bien1  DB  "moi ban nhap vao mot ky tu thuong:$"
bien1  DB  10,13,"doi thanh chu hoa va hien thi:$"
.code
main proc
;khởi tạo thanh ghi DS
```

```
mov AX,@data
mov DS,AX
;hiển thị chuỗi thứ nhất
mov AH,9
LEA DX,bien1
int 21H
;nhập vào 1 ký tự và đổi thành chữ hoa
mov AH,1
int 21H
mov BL,AL
sub BL,AL
;hiển thị chuỗi thứ hai ở dòng dưới
mov AH,9
LEA DX,bien2
int 21H
;hiển thị dạng chữ hoa của ký tự ban đầu
mov AH,2
mov DL,BL
int 21H
;trở về DOS
mov AH,4CH
int 21H
main endp
end main
```

3.4. CÁC NHÓM LỆNH CỦA 8086

3.4.1. Nhóm lệnh dịch chuyển dữ liệu

Phần này sẽ giới thiệu một số lệnh cơ bản hay sử dụng nhất cho việc dịch chuyển dữ liệu giữa các thanh ghi bên trong bộ vi xử lý, giữa bộ nhớ hoặc thiết bị ngoài với bộ vi xử lý.

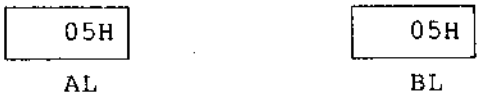
3.4.1.1. Lệnh MOV (MOVE)

Dạng lệnh: `MOV đích, nguồn`

Lệnh MOV được sử dụng để chuyển dữ liệu từ toán hạng nguồn vào toán hạng đích. *Ví dụ:* `MOV AL,BL`

Giả sử trước khi thực hiện lệnh MOV, thanh ghi AL và BL có nội dung:

Sau khi thực hiện lệnh MOV, nội dung thanh ghi AL và BL:



Một bản sao của thanh ghi BL sẽ được chuyển vào thanh ghi AL còn nội dung của BL vẫn giữ nguyên. Giá trị trước đó của AL bị viết chèn lên.

Khả năng kết hợp giữa toán hạng nguồn với toán hạng đích được chỉ ra ở bảng 3.2.

Bảng 3.2. Khả năng kết hợp giữa các toán hạng của lệnh MOV

Toán hạng đích / Toán hạng nguồn	Thanh ghi công dụng chung	Thanh ghi đoạn	Ô nhớ	Hằng số
Thanh ghi công dụng chung	Có thể	Có thể	Có thể	Không thể
Thanh ghi đoạn	Có thể	Không thể	Có thể	Không thể
Ô nhớ	Có thể	Có thể	Không thể	Không thể
Hằng số	Có thể	Không thể	Có thể	Không thể

Một vài ví dụ về các lệnh không hợp lệ:

```
MOV   DS,ES       ;chuyển nội dung từ thanh ghi đoạn đến
                  ;thanh ghi đoạn
MOV   DS,0AH      ;chuyển hằng số vào thanh ghi đoạn
MOV   [BX],0AH    ;chuyển hằng số vào ô nhớ
```

3.4.1.2. Lệnh OUT

Dạng lệnh: `OUT cổng,thanh_chứa`

Thao tác: Lệnh OUT xuất dữ liệu từ thanh ghi (AL) ra cổng. Cổng có thể là một hằng số (cổng cố định) hay DX (dữ liệu thay đổi).

Cờ: Không bị ảnh hưởng.

3.4.1.3. Lệnh IN

Dạng lệnh: `IN thanh_chứa, cổng`

Thao tác: Lệnh IN đọc dữ liệu từ cổng vào thanh ghi (AL). Cổng có thể là một hằng số (cổng cố định) hay DX (dữ liệu thay đổi).

Cờ: Không bị ảnh hưởng.

Ví dụ: Giả sử các cổng PA, PB đã được định nghĩa hai lệnh sau đây thực hiện đọc dữ liệu từ cổng vào thanh ghi và xuất dữ liệu từ thanh ghi ra cổng.

```
IN  AL,PA       ;đọc dữ liệu từ PA vào AL
OUT PB,AL       ;xuất dữ liệu từ AL ra PB
```

3.4.1.4. Lệnh XCHG (exchange)

Dạng lệnh: `XCHG đích, nguồn`

Thao tác: Toán hạng đích và toán hạng nguồn được đổi lẫn cho nhau.

Cờ: Không bị ảnh hưởng.

Giả sử thanh ghi AL và BL có nội dung như sau:

Sau khi thực hiện lệnh `XCHG AL, BL` nội dung AL và BL:

3.4.1.5. Lệnh XLAT (translate byte to AL)

Dạng lệnh: `XLAT nhãn_nguồn`

Thao tác: BX phải chứa địa chỉ offset của bảng nguồn có chiều dài tối đa là 256 byte. AL phải chứa chỉ số của các phần tử bảng. Lệnh sẽ thay thế nội dung của AL bằng nội dung của các phần tử bảng được định địa chỉ bằng (BX + AL).

Cờ: Không bị ảnh hưởng.

3.4.1.6. Lệnh LAHF (Load AH from Flags)

Dạng lệnh: `LAHF`

Thao tác: 8 bit thấp của thanh ghi cờ được chuyển vào AH.

Cờ: Không bị ảnh hưởng.

3.4.1.7. Lệnh LDS (Load Data Segment register)

Nạp vào thanh ghi DS địa chỉ đoạn và thanh ghi công dụng chung địa chỉ lệch để truy cập dữ liệu.

Dạng lệnh: `LDS đích, nguồn`

Thao tác: Toán hạng nguồn là một ô nhớ có độ dài 2 từ. Từ thấp được đặt vào thanh ghi đích còn từ cao đặt vào thanh ghi DS.

Cờ: Không bị ảnh hưởng.

3.4.1.8. Lệnh LES (Load Etra Segment register)

Nạp vào thanh ghi ES địa chỉ đoạn và một thanh ghi công dụng chung địa chỉ lệch để có thể truy cập dữ liệu.

Dạng lệnh: `LES đích, nguồn`

Thao tác: Toán hạng nguồn là ô nhớ có độ dài 2 từ. Từ thấp được đặt vào thanh ghi đích, từ cao đặt vào thanh ghi ES.

Cờ: Không bị ảnh hưởng.

3.4.1.9. Lệnh SAHF (Store AH in Flag register)

Dạng lệnh: `SAHF`

Thao tác: Lưu 5 bit thấp của AH vào byte thấp của thanh ghi cờ. Chỉ có bit tương ứng với cờ mới được chuyển đổi. Các cờ trong bit thấp của thanh ghi cờ là SF = bit 7, ZF = bit 6, AF = bit 4, PF = bit 2, CF = bit 0.

Cờ: Bị ảnh hưởng AF, PF, ZF, SF, CF.

3.4.2. Nhóm lệnh số học

3.4.2.1. Lệnh ADD, SUB

Các lệnh ADD, SUB được sử dụng để cộng hoặc trừ nội dung của hai toán hạng.

Dạng lệnh:
```
ADD đích, nguồn
SUB đích, nguồn
```

Lệnh ADD và SUB thực hiện cộng hoặc trừ nội dung của toán hạng đích với toán hạng nguồn, kết quả lưu vào toán hạng đích, nguồn không thay đổi. *Ví dụ:* `ADD AL,BL`

Giả sử trước khi thực hiện lệnh MOV, AL và BL có nội dung như sau:

Sau khi thực hiện lệnh MOV, nội dung AL và BL:

Ví dụ: `SUB AL,BL`

Giả sử trước khi thực hiện lệnh MOV, AL và BL có nội dung như sau:

Sau khi thực hiện lệnh MOV, nội dung AL và BL:

Khả năng kết hợp giữa toán hạng nguồn với toán hạng đích trong các phép cộng và phép trừ được chỉ ra ở bảng 3.3.

Bảng 3.3. Khả năng kết hợp giữa các toán hạng của lệnh ADD, SUB

Toán hạng đích / Toán hạng nguồn	Thanh ghi công dụng chung	Ô nhớ	Hằng số
Thanh ghi công dụng chung	Có thể	Có thể	Không thể
Ô nhớ	Có thể	Không thể	Không thể
Hằng số	Có thể	Có thể	Không thể

3.4.2.2. Lệnh ADC (ADD with Carry)

Cờ nhớ được cộng vào tổng của toán hạng nguồn và toán hạng đích.

Dạng lệnh: `ADC đích,nguồn`

Thao tác: Nếu CF = 1 thì đích = nguồn + đích + 1.

Nếu CF = 0 thì đích = nguồn + đích.

Cờ: Bị ảnh hưởng: AF, CF, OF, SF, ZF, PF.

3.4.2.3. Lệnh CBW (Convert Byte to Word)

Đổi số có dấu 8 bit trong AL thành số có dấu 16 bit trong AX.

Dạng lệnh: `CBW`

Thao tác: Nếu bit 7 của AL bằng 1 thì AH = FFH.

Nếu bit 7 của AL bằng 0 thì AH = 00H.

Cờ: Không ảnh hưởng.

3.4.2.4. Lệnh DEC (DECrement)

Thực hiện phép cộng có nhớ.

Dạng lệnh: `Dec đích`

Thao tác: Giảm toán hạng đích đi 1.

Cờ: Bị ảnh hưởng AF, OF, PF, SF, ZF.

Lệnh này tương đương với lệnh `ADD đích,1`.

3.4.2.5. DIV (DIVide)

Thực hiện phép chia không dấu.

Dạng lệnh: `Div nguồn`

Thao tác: Số chia là toán hạng nguồn có thể là một ô nhớ hay thanh ghi. Trong phép chia cho byte (toán hạng nguồn 8 bit), số bị chia là một từ mặc định chứa trong AX, kết quả thương số chứa trong AL và số dư trả về trong AH. Khi thực hiện phép chia cho word (toán hạng nguồn 16 bit), số bị chia 32 bit chứa trong DX:AX (16 bit cao trong DX, 16 bit thấp trong AX). Kết quả thương số chứa trong AX và số dư trả về trong DX.

Cờ: Không xác định AF, OF, CF, ZF, SF, PF.

Ví dụ: Giả sử AX chứa 20, DX chứa 0, BX chứa 6. Khi thực hiện phép chia cho byte hoặc cho word ta sẽ được kết quả như sau:

Kết quả / Lệnh	AH	AL	AX	DX
Div BL	2	3	0203H	0
Div BX	0	3	0003H	2

Hiện tượng tràn số có thể xảy ra khi số bị chia quá lớn so với số chia. Nếu thương số lớn hơn 8 bit trong phép chia cho byte (16 bit trong phép chia cho word) hoặc số chia bằng 0 thì ngắt INT O được tạo ra.

3.4.2.6. Lệnh IDIV (Integer DIVide)

Thực hiện phép chia có dấu.

Dạng lệnh: `IDIV nguồn`

Thao tác: Số chia là toán hạng nguồn có thể là một ô nhớ hay một thanh ghi. Trong phép chia cho byte (toán hạng nguồn 8 bit) số bị chia chứa trong AX, còn trong phép chia cho word (toán hạng nguồn 16 bit) số chia chứa trong DX:AX. Thương số chứa trong AL (AX trong trường hợp chia cho word) và số dư (luôn cùng dấu với số bị chia) trả về trong AH (DX cho phép chia cho word). Nếu thương số lớn hơn 127 hoặc nhỏ hơn –127 trong phép chia cho byte (lớn hơn 32767 hoặc nhỏ hơn –32767 trong phép chia cho word) hoặc số chia bằng 0 thì ngắt INT O được tạo ra.

Cờ: Không xác định AF, OF, CF, ZF, PF.

3.4.2.7. Lệnh IMUL (Integer MULtiplycation)

Thực hiện phép nhân có dấu giữa hai toán hạng.

Dạng lệnh: `IMUL nguồn`

Thao tác: Lệnh Imul nguồn thực hiện nhân nội dung của AL với toán hạng nguồn nếu toán hạng nguồn là một byte, tích mặc định chứa trong AX, hoặc nhân nội dung của AX với toán hạng nguồn nếu toán hạng nguồn là một từ, tích mặc định chứa trong DX:AX. Các cờ CF, OF = 0 nếu nửa cao của tích là phần mở rộng dấu của nửa thấp. Toán hạng nguồn không được là hằng số.

Cờ: Bị ảnh hưởng CF, OF.

3.4.2.8. Lệnh MUL (MULtiply)

Thực hiện phép nhân không dấu.

Dạng lệnh: `MUL nguồn`

Thao tác: Lệnh `MUL nguồn` thực hiện nhân nội dung của AL với toán hạng nguồn nếu toán hạng nguồn là một byte, tích mặc định chứa trong AX, hoặc nhân nội dung của AX với toán hạng nguồn nếu toán hạng nguồn là một từ, tích mặc định chứa trong DX: AX. Các cờ CF, OF = 0 nếu nửa cao

của tích là phần mở rộng dấu của nửa thấp. Toán hạng nguồn không được là hằng số.

Cờ: Bị ảnh hưởng CF, OF.

Ví dụ: Giả sử A, B, C, D là các biến không dấu kiểu byte. Để thực hiện phép toán: `AX (chứa kết quả)= A*B + C*D`

Đoạn mã như sau:

```
mov AL,A        ; AL chứa A
mov Bl,B        ; BL chứa B
MUL BL          ; AX chứa A*B
mov DX,AX       ; cất A*B vào DX
mov AL,C        ; AL chứa C
mov Bl,D        ; BL chứa D
MUL BL          ; AX chứa C*D
ADD AX,DX       ; AX chứa A*B + C*D
```

3.4.2.9. Lệnh NEG (NEGate)

Lấy số bù 2.

Dạng lệnh: `NEG đích`

Thao tác: Toán hạng đích bị trừ đi từ số toàn chữ số 1 (0FFH với byte và 0FFFFH với word). Sau đó kết quả chứa trong toán hạng đích được cộng thêm 1.

Cờ: Bị ảnh hưởng AF, OF, CF, SF, ZF.

Ví dụ: Giả sử thanh ghi DH chứa 2, kết quả DH sau khi thực hiện lệnh `NEG DH` như sau:

```
DH =    (0FFH - 2) + 1
   =    256 - 2
   =    -2 (hay 0FEH)
```

Nếu thanh ghi DX chứa 2, sau khi thực hiện lệnh `NEG DX` kết quả `DX = 0FFFEH`.

3.4.2.10. Lệnh SBB (SuBtract with Borrow)

Trừ có nhớ.

Dạng lệnh: `SBB đích,nguồn`

Thao tác: Trừ toán hạng đích cho toán hạng nguồn và nếu CF = 1 thì trừ kết quả nhận được đi 1. Kết quả chứa trong toán hạng đích.

Cờ: Bị ảnh hưởng AF, CF, OF, PF, SF, ZF.

3.4.3. Nhóm lệnh điều khiển rẽ nhánh

Trong lập trình hợp ngữ, để thực hiện cấu trúc chọn lựa và cấu trúc lặp ta phải sử dụng các lệnh nhảy (jump) và lệnh lặp (loop).

Cú pháp chung của các lệnh nhảy như sau: `Mã_lệnh Nhãn_lệnh`

3.4.3.1. Nhóm lệnh nhảy có điều kiện

Với các lệnh nhảy có điều kiện thì khi gặp lệnh nhảy, nếu như điều kiện của lệnh nhảy được thoả mãn thì `Nhãn_lệnh` được thi hành còn nếu ngược lại thì thực hiện lệnh ngay sau lệnh nhảy. `Nhãn_lệnh` có thể đặt trước hoặc sau lệnh nhảy.

Phạm vi của lệnh nhảy có điều kiện trong khoảng 128 byte trước lệnh nhảy và 127 byte sau lệnh nhảy. Ví dụ về một lệnh nhảy có điều kiện:

a) Lệnh JNZ (Jump if Not Zero)

Điều kiện của JNZ (nhảy nếu khác 0) là kết quả của lệnh trước đó khác 0.

Dạng lệnh: `JNZ nhãn_đích`

Thao tác: Nếu kết quả của lệnh trước đó khác 0 thì lệnh có tên là nhãn_đích được thi hành còn nếu ngược lại thì thực hiện lệnh ngay sau lệnh nhảy.

Còn rất nhiều các lệnh nhảy có điều kiện được chỉ ra ở bảng 3.4. Một số lệnh nhảy có hai mã lệnh như: JL (nhảy nếu nhỏ hơn) và JNGE (nhảy nếu không lớn hơn hay bằng), khi mã hoá kết quả của hai lệnh này là giống nhau.

Bảng 3.4. Các lệnh nhảy có điều kiện

Lệnh	Nhảy nếu	Điều kiện	Mã lệnh
JA	Above	CF = 0 AND ZF = 0	77 Disp
JAE	Above or Equal	CF = 0	73 Disp
JB	Below	CF = 1	72 Disp
JBE	Below or Equal	CF = 1 OR ZF = 1	76 Disp

JC	Carry	CF = 1	72 Disp
JCXZ	CX=0	(CF OR ZF) = 0	E3 Disp
JE	Equal	ZF = 1	74 Disp
JG	Greater	ZF = 0 AND SF = OF	7F Disp
JGE	Greater Equal	ZF = OF	7D Disp
JL	Less	(SF XOR OF) = 1	7C Disp
JLE	Less or Equal	(SF XOR OF) OR ZF = 1	7E Disp
JNA	Not Above	CF = 1 OR ZF = 1	76 Disp
JNAE	Not Above or Equal	CF = 1	72 Disp
JNB	Not Below	CF = 0	73 Disp
JNBE	Not Below or Equal	CF = 0 OR ZF = 0	77 Disp
JNC	Not Carry	CF = 0	73 Disp
JNE	Not Equal	ZF = 0	75 Disp
JNG	Not Greater	(SF XOR OF) OR ZF = 1	7E Disp
NGE	Not Greater or Equal	(SF XOR OF) = 1	7C Disp
JNL	Not Less	SF = OF	7D Disp
JNLE	Not Less or Equal	ZF = 0 Và SF = OF	7F Disp
JNO	Not Over flow	OF = 0	71 Disp
JNP	Not Parity	PF = 0	7B Disp
JNS	Not Sign	SF = 0	79 Disp
JNZ	Not Zero	ZF = 0	75 Disp
JO	Overflow	OF = 1	70 Disp
JP	Parity	PF = 1	7A Disp
JPE	Parity Even	PF = 1	7A Disp
JPO	Parity Odd	PF = Ò	7B Disp
JS	Sign	SF = 1	78 Disp
JZ	Zero	ZF = 1	74 Disp

Thông thường các lệnh nhảy được cung cấp bởi lệnh so sánh (compare).

b) Lệnh CMP (CoMPare)

Lệnh CMP so sánh hai toán hạng bằng cách thực hiện phép trừ nhưng không lưu lại kết quả.

Dạng lệnh: `CMP đích, nguồn`

Thao tác: Trừ toán hạng đích cho toán hạng nguồn và thiết lập các cờ theo kết quả của phép tính, các toán hạng vẫn không thay đổi.

Cờ: Bị ảnh hưởng AF, CF, OF, PF, EF, ZF.

Ví dụ đoạn chương trình sau đây hiển thị ra màn hình 9 ký tự từ liên tiếp '0' đến '8' trên một dòng:

```
        mov AH,2      ;Hàm hiển thị ký tự
        mov DL,30H    ;DL chứa mã ASCII của ký tự hiển thị
H_thi:  int 21H       ;giảm mã ASCII
        inc DL        ;so sánh với ký tự '9'
        CMP DL,39H    ;lặp lại nếu không phải '9'
        JNZ H_thi     ;nếu bằng '9' thì thoát
Thoát:
```

Nếu thay lệnh JNZ trong đoạn mã trên bằng lệnh JLE (nếu nhỏ hơn hay bằng) thì đoạn hiển thị ra màn hình 10 ký tự liên tiếp từ '0' đến '9' trên một dòng:

```
        mov AH,2        ;Hàm hiển thị ký tự
        mov DL,30H      ;DL chứa mã ASCII của ký tự hiển thị
H_thi:  int 21H
        inc DL          ;giảm mã ASCII
        CMP DL,39H      ;so sánh với ký tự '9'
        JLE H_thi       ;lặp lại nếu nhỏ hơn hoặc bằng 9
Thoát:                  ;nếu lớn hơn '9' thì thoát
```

3.4.3.2. Lệnh nhảy không điều kiện (JuMP)

Dạng lệnh: `JMP nhãn_đích`

Thao tác: Nếu gặp lệnh JMP thì chương trình chuyển ngay tới lệnh nhãn_đích để thi hành mà không cần điều kiện gì.

Cờ: Không bị ảnh hưởng.

Ví dụ:

```
        mov AH,2      ;hàm hiển thị ký tự
        mov DL,30H    ;DL chứa mã ASCII của ký tự hiển thị
H1:     int 21H
        inc DL        ;giảm mã ASCII
        CMP DL,39H    ;so sánh với ký tự '9'
        JLE H2        ;nhảy đến h2 nếu (<= 9)
        JMP thoat     ;nếu lớn hơn '9' thì thoát
H2:     JMP H1        ;nhảy không điều kiện về H1
Thoát:
```

3.4.3.3. Các cấu trúc lập trình

a) Cấu trúc tuần tự

Cấu trúc tuần tự là cấu trúc thông dụng và đơn giản nhất. Trong cấu trúc này, chương trình là dãy các lệnh được sắp xếp và thực hiện một cách tuần tự từ trên xuống dưới.

Cú pháp:

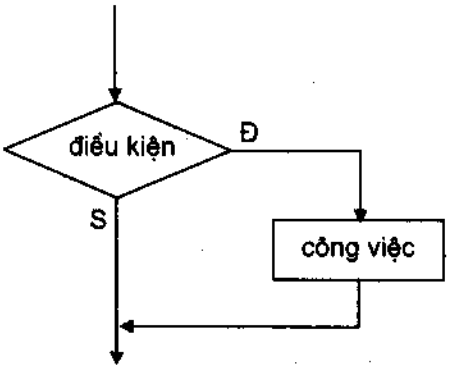
```
Begin
công_việc_1
công_việc_2
...
công_việc_n
end
```

b) Cấu trúc rẽ nhánh

*if then

Cú pháp:

```
if điều_kiện
then
   công_việc
end_if
```



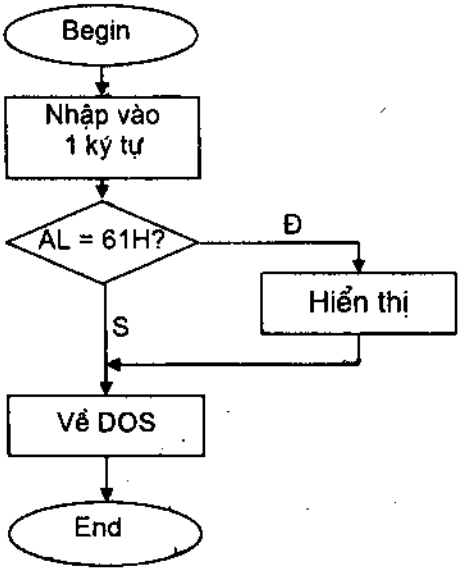
Hình 3.7. Lưu đồ thuật toán cú pháp if then

Nếu điều kiện thoả mãn thì thực hiện công việc, nếu không thì bỏ qua sau đó thực hiện các lệnh kế tiếp.

Ví dụ 1: Viết chương trình nhập vào một ký tự từ bàn phím, nếu là 'a' thì hiển thị. Nếu không phải thì về DOS.

Giải:

Lưu đồ thuật toán hình 3.8.



Hình 3.8. Lưu đồ thuật toán ví dụ 1

Mã hoá ví dụ 1 bằng chương trình sau:

```
.model small
.stack 100H
.data
.code
main proc

    mov AH,1          ; nhập một ký tự
    int 21H
    mov BL,AL
    cmp AL,61H        ; so sánh ký tự với 'a'
    jne thoat
```

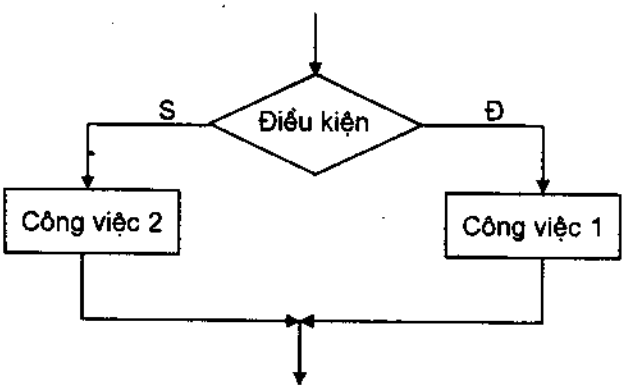
```
        mov AH,2            ; nếu không phải 'a' thì thoát
        mov DL,BL           ; nếu là 'a' thì hiển thị
        int 21H
thoat:
        mov AH,4CH
        int 21H
main endp
end main
```

Trong đoạn chương trình trên, để so sánh ký tự nhập vào xem có phải 'a' không ta dùng lệnh CMP AL,61H, sau đó dùng lệnh nhảy JNE để bỏ qua bước hiển thị nếu ký tự đó không phải 'a'.

*if then else

Cú pháp:

```
if điều_kiện
then
    công_việc_1
else
    công_việc_2
end_if
```



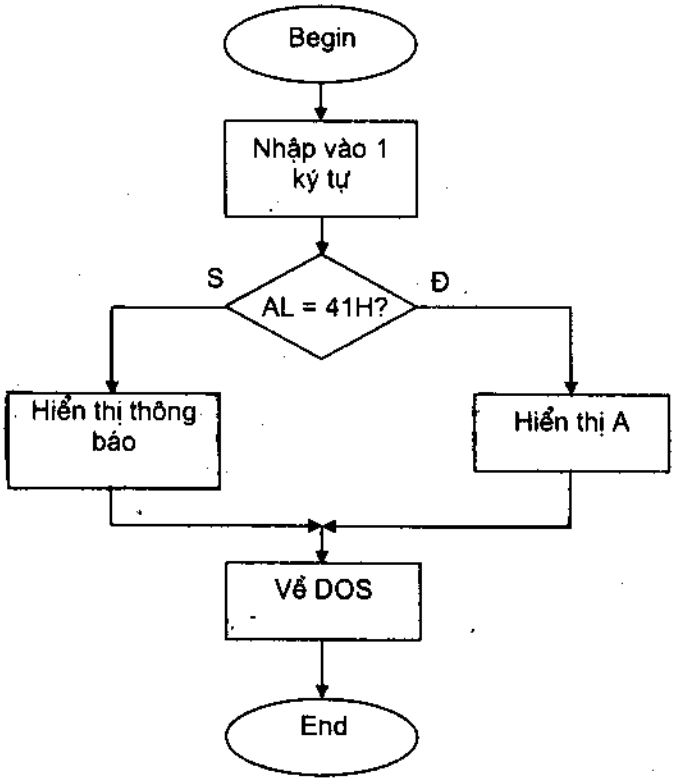
Hình 3.9. Lưu đồ thuật toán cú pháp if then else

Nếu điều kiện đúng thì thực hiện công việc 1, nếu điều kiện sai thì thực hiện công việc 2.

Ví dụ 2: Nhập vào một ký tự từ bàn phím, nếu là chữ A thì hiển thị, nếu không phải, hiển thị dòng thông báo "Ky tu do khong phai chu A".

Giải:

Lưu đồ thuật toán:



Hình 3.10. Lưu đồ thuật toán ví dụ 2

Mã hoá ví dụ 2 bằng đoạn chương trình sau:

```
        mov AH,1                ; nhập một ký tự
        int 21H
        mov BL,AL
        cmp AL,41H              ; so sánh ký tự với 'A'
        je ht_kytu              ; nếu là 'A' thì hiển thị ký tự
        mov AH,9                ; nếu không phải 'A' thì
        lea DX,thongbao
        int 21H                 ; hiển thị thông báo
        jmp veDOS               ; nhảy về DOS
ht_kytu:
        mov AH,2
        mov DL,BL
        int 21H                 ; hiển thị ký tự
```

```
veDOS:
        mov AH,4CH
        int 21H
main endp
end main
```

Đôi khi điều kiện của if thường có thêm các điều kiện AND hoặc điều kiện OR.

Cú pháp:

Điều kiện AND:	Điều kiện OR:
if điều_kiện_1 **and** điều_kiện_2 **then** công_việc_1 **else** công_việc_2 **end_if**	**if** điều_kiện_1 **or** điều_kiện_2 **then** công_việc_1 **else** công_việc_2 **end_if**

Điều kiện AND chỉ đúng khi đồng thời hai điều kiện điều kiện 1 và điều kiện 2 cùng đúng. Ngược lại, chỉ cần một trong hai điều kiện sai là biểu thức sai.

Điều kiện OR đúng khi một trong hai điều kiện đúng, và chỉ sai khi cả hai điều kiện sai.

Ví dụ 3: Nhập một ký tự, nếu là các con số từ 0 đến 9 thì hiển thị.

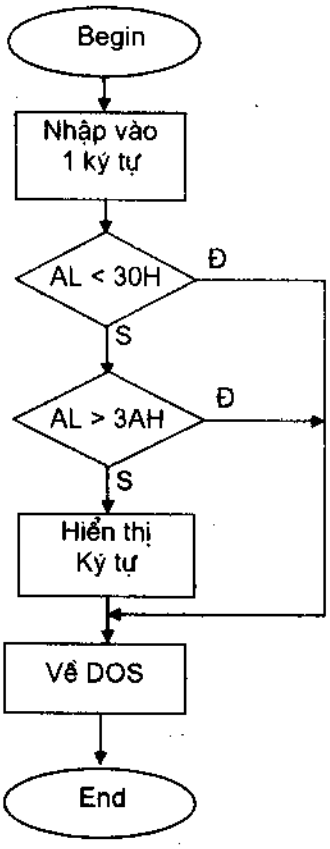
Giải:

Trong ví dụ này, sau khi nhập vào một ký tự ta phải kiểm tra nếu ký tự đó đứng trước '0' thì kết thúc. Nếu ký tự đó đứng sau '0' cũng chưa được hiển thị ngay ta phải kiểm tra tiếp với 9. Nếu ký tự đó sau 9 thì kết thúc, còn nếu đứng trước '9' mới được hiển thị.

Lưu đồ thuật toán và đoạn mã chương trình được viết như sau:

```
MOV AH,1
INT 21H
CMP AL,30H
JB END_IF
```

```
CMP AL,39H
JA END_IF
MOV DL,AL
MOV AH,2
INT 21H
END_IF:
```

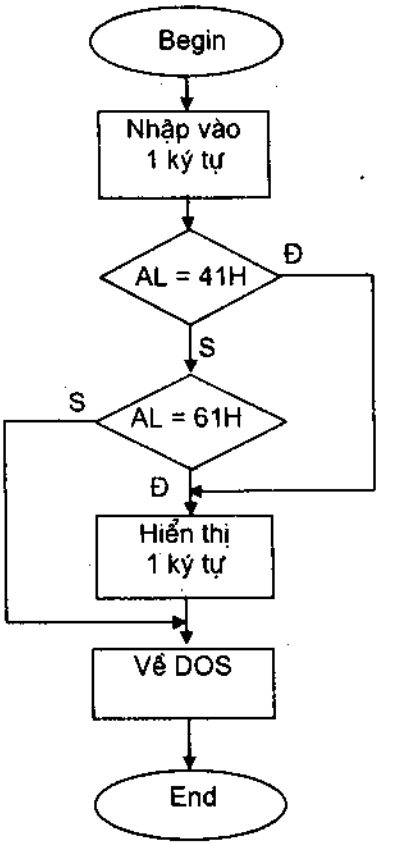


Hình 3.11. Lưu đồ thuật toán ví dụ 3

Ví dụ 4: Viết chương trình nhập một ký tự, nếu là 'A' hay 'a' thì hiển thị.

Trong ví dụ này, khi nhập vào một ký tự ta phải kiểm tra xem ký tự đó có phải 'A' không, nếu ký tự đó không phải 'A' thì so sánh tiếp với 'a' nếu cũng không phải thì thoát. Còn nếu là 'A' hoặc 'a' thì hiển thị. Sau đây là lưu đồ thuật toán và mã lệnh.

```
   MOV AH,1
   INT 21H
   CMP AL,41H
   JE hienthi
   CMP AL,61H
   JNE thoat
hienthi:
   MOV DL,AL
   MOV AH,2
   INT 21H
thoat:
   MOV AH,4CH
   INT 21H
   END_IF:
```

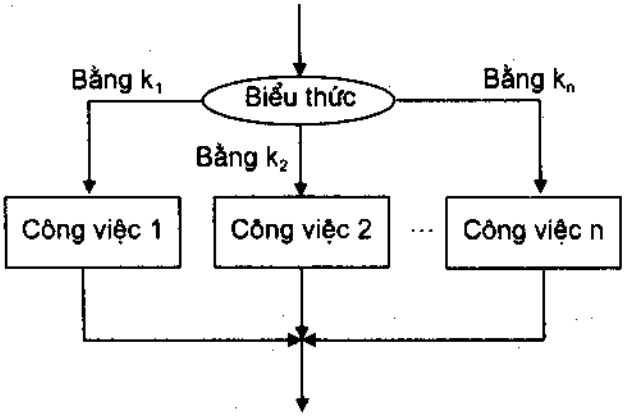


Hình 3.12. Lưu đồ thuật toán ví dụ 4

*case

Cú pháp:

```
case biểu_thức
     k1: công_việc_1
     k2: công_việc_2
          ...
     kn: công_việc_n
end_case
```



Hình 3.13. Lưu đồ thuật toán cấu trúc case

Case là một cấu trúc lựa chọn một trong nhiều nhánh. Nếu giá trị của biểu thức bằng k_i thì thực hiện công việc thứ i.

Ví dụ 5: Nếu thanh ghi AX chứa số âm, hãy nhập –1 vào thanh ghi BX, nếu AX chứa 0 thì thay BX = 0, nếu AX dương thì thay BX bằng 1.

Giải: Mã lệnh giả và đoạn mã bằng hợp ngữ như sau:

```
Case AX
< 0: BX = -1
= 0: BX = 0
> 0: BX = 1
end_case
```

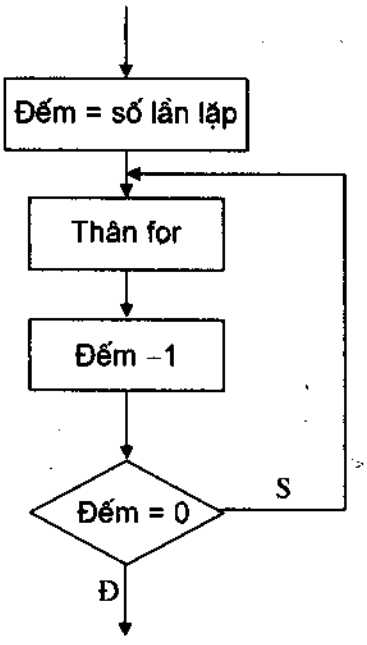
```
CMP AX,0
JG duong
JL am
JE khong
duong:
MOV BX,1
JMP ketthuc
am:
MOV BX,-1
JMP ketthuc
khong:
MOV BX,0
ketthuc:
```

c) Cấu trúc lặp

* for

Cú pháp:

```
for số_lần_lặp
do
thân for
end_for
```



Hình 3.14. Lưu đồ thuật toán cấu trúc lặp for

Vòng lặp for sẽ thực hiện công việc với số lần lặp đã biết trước.

Để thực hiện cấu trúc lặp for bằng hợp ngữ, ta sử dụng lệnh lặp loop với cú pháp như sau: `Loop Nhãn_đích`

Hoạt động của lệnh `loop` như sau: Số lần lặp được nạp vào thanh ghi CX, mỗi lần thực hiện lệnh `loop` thì CX tự động giảm đi 1 rồi so sánh CX với 0, nếu CX khác 0 thì lặp lại thân vòng lặp. Nếu CX bằng 0 thì vòng lặp kết thúc. `Nhãn_đích` phải đặt trước lệnh lặp không quá 126 byte. Cú pháp vòng lặp như sau:

```
; khởi tạo CX bằng số lần lặp
Lable:
; Thân vòng lặp
        loop Lable
```

Chú ý: Nếu trước vòng lặp CX bằng 0, sau khi thực hiện thân vòng lặp một lần thì CX tự động giảm đi 1 nên nội dung trong thanh ghi CX là 0FFFFH vì thế vòng lặp được thực hiện thêm 65535 lần nữa. Ta có thể sử dụng lệnh JCXZ (Jump if CX is Zero) được đặt trước vòng lặp để nếu CX bằng 0 trước vòng lặp thì sẽ không thực hiện vòng lặp. Cú pháp vòng lặp như sau:

```
; khởi tạo CX bằng số lần lặp
JCXZ exit
Lable:
; Thân vòng lặp
LOOP Lable
EXIT:
```

Ví dụ 6: Sử dụng vòng lặp viết chương trình hiển thị 10 dấu '*' trên 10 dòng.

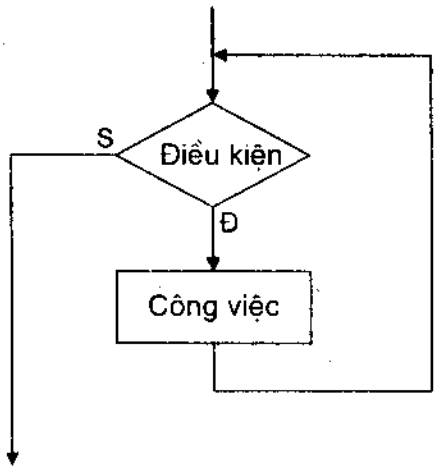
Giải: Mã lệnh giả và đoạn mã bằng hợp ngữ như sau:

```
                        MOV AH,2
for 10 lần              MOV CX,10
do                      LAP:
                        MOV DL,'*'
; hiển thị              INT 21H
                        MOV DL,0dh
; xuống dòng            INT 21H
                        MOV DL,0ah
; về đầu dòng           INT 21H
end_for                      LOOP LAP
```

*while

Cú pháp:

```
while điều_kiện
do
  công_việc
end_while
```



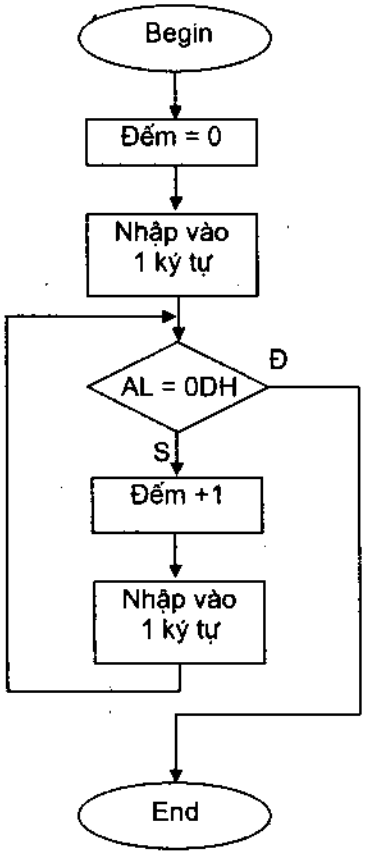
Hình 3.15. Lưu đồ thuật toán cấu trúc while

Vòng lặp **while** sẽ kiểm tra điều kiện trước, nếu kiều kiện lặp được thoả mãn thì thân vòng lặp được thực hiện, sau đó điều kiện lặp lại được kiểm tra và thực hiện cho đến khi điều kiện lặp không thoả mãn. Công việc trong vòng lặp không được thực hiện lần nào nếu điều kiện sai ngay từ đầu tiên.

Ví dụ 7: Viết chương trình để đếm số ký tự trên một dòng được nhập từ bàn phím.

Giải: Lưu đồ thuật toán và đoạn mã chương trình như sau:

```
MOV CX,0
MOV AH,1
INT 21H
tieptuc:
CMP AL,0DH
JE ketthuc
INC CX
INT 21H
JMP tieptuc
ketthuc:
```

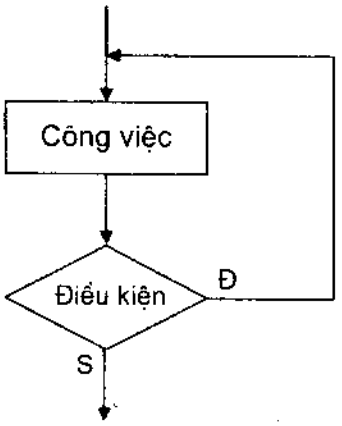


Hình 3.16. Lưu đồ thuật toán ví dụ

*do while

Cú pháp:

```
do
    công_việc
while điều_kiện
end_do_while
```



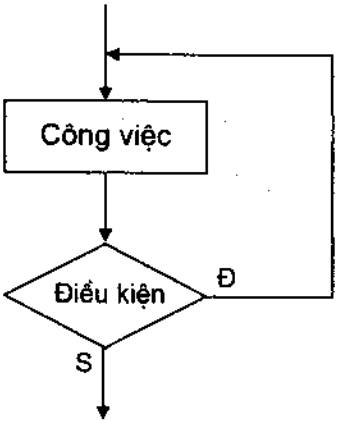
Hình 3.17. **Lưu đồ thuật toán cấu trúc lặp do while**

Vòng lặp **do while** sẽ kiểm tra điều kiện sau, tức là thân vòng lặp được thực hiện trước sau đó mới tiến hành kiểm tra điều kiện lặp, nếu điều kiện sai thì vòng lặp kết thúc, nếu điều kiện đúng thì vòng lặp tiếp tục được thực hiện. Công việc trong vòng lặp được thực hiện ít nhất một lần nếu điều kiện sai ngay từ vòng đầu tiên.

*repeat until

Cú pháp:

```
repeat
        công_việc
until điều_kiện
```



Hình 3.18. Lưu đồ thuật toán cấu trúc lặp repeat until

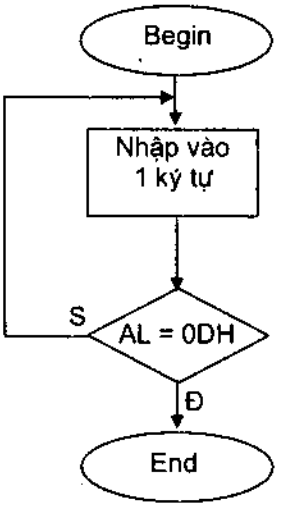
Vòng lặp **repeat until** thực hiện thân vòng lặp trước sau đó mới tiến hành kiểm tra điều kiện lặp, nếu điều kiện đúng thì vòng lặp kết thúc, nếu điều kiện sai thì vòng lặp tiếp tục được thực hiện. Công việc trong vòng lặp được thực hiện ít nhất một lần nếu điều kiện đúng ngay từ vòng đầu tiên.

Ví dụ 8: Viết chương trình đếm số ký tự trên một dòng được nhập từ bàn phím.

Giải: Lưu đồ thuật toán và đoạn mã bằng hợp ngữ của ví dụ này ta có thể sử dụng vòng lặp **do while** và **repeat until** như sau:

Cách 1: Sử dụng vòng lặp **do while**.

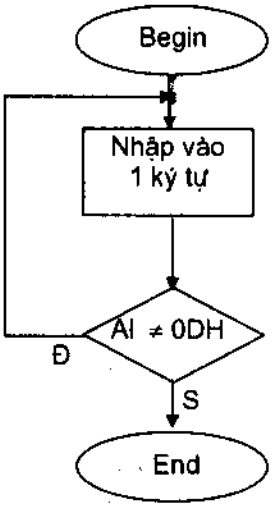
```
MOV AH,1
tieptuc:
INT 21H
CMP AL,0DH
JnE tieptuc
ketthuc:
mov AH,4CH
int 21H
```



Hình 3.19. Lưu đồ thuật toán ví dụ 8 sử dụng vòng lặp do while

Cách 2: Sử dụng vòng lặp **repeat until**.

```
MOV AH,1
tieptuc:
INT 21H
CMP AL,0DH
JE ketthuc
Jmp tieptuc
ketthuc:
mov AH,4CH
int 21H
```



Hình 3.20. Lưu đồ thuật toán ví dụ 8 sử dụng vòng lặp repeat until

3.4.4. Nhóm lệnh xử lý bit

Các lệnh logic, dịch và quay có thể tác động đến từng bit trong một byte hoặc một từ, nhóm lệnh này được dùng để thiết lập, và kiểm tra từng bit trong thanh ghi hay ô nhớ mà không ảnh hưởng đến các bit khác.

3.4.4.1. Các lệnh logic

a) Các lệnh AND, OR, XOR và TEST

Dạng lệnh:

```
AND     đích, nguồn
OR      đích, nguồn
XOR     đích, nguồn
TEST    đích, nguồn
```

Thao tác: Mỗi bit toán hạng nguồn được lấy logic với bit tương ứng của toán hạng đích theo bảng chân lý 3.5. Kết quả được lưu trong toán hạng đích, cờ CF và OF bị xoá.

Cờ: Bị ảnh hưởng: PF, SF, ZF.

Cờ AF không xác định.

Bảng 3.5. Bảng chân lý của các phép toán logic

	NGUỒN	ĐÍCH	ĐÍCH (KẾT QUẢ)		NGUỒN	ĐÍCH	ĐÍCH (KẾT QUẢ)
AND	1	1	1	XOR	1	1	0
	1	0	0		1	0	1
	0	1	0		0	1	1
	0	0	0		0	0	0
OR	1	1	1	NOT		0	1
	1	0	1			0	1
	0	1	1			1	0
	0	0	0			1	0

Lệnh TEST thực hiện thao tác giống như lệnh AND nhưng không lưu lại kết quả trong toán hạng đích, thường TEST được sử dụng làm điều kiện cho các lệnh nhảy.

Ta có thể dùng các lệnh logic để thay đổi có chọn lọc các bit của toán hạng đích. Để làm điều đó chúng ta tạo lên một mẫu bit được gọi là mặt nạ (MASK). Các bit của mặt nạ được chọn sao cho các bit tương ứng của toán hạng đích được thay đổi đúng như ta mong muốn khi lệnh logic được thực hiện.

Để tìm được mặt nạ, từ bảng trên rút ra một số kết luận sau:

– Lệnh AND có thể sử dụng để xoá các bit nhất định của toán hạng đích trong khi giữ nguyên các bit còn lại. Bit 0 của mặt nạ sẽ xoá bit tương ứng, còn bit 1 của mặt nạ giữ nguyên bit tương ứng của toán hạng đích.

– Lệnh OR có thể được dùng để thiết lập các bit xác định của toán hạng đích trong khi vẫn giữ nguyên những bit còn lại. Bit 1 của mặt nạ sẽ thiết lập bit tương ứng trong khi bit 0 của nó giữ nguyên bit tương ứng của toán hạng đích.

– Lệnh XOR có thể dùng để đảo các bit xác định của toán hạng đích trong khi vẫn giữ nguyên các bit còn lại. Bit 1 của mặt nạ làm đảo bit tương ứng còn bit 0 giữ nguyên bit tương ứng của toán hạng đích. XOR còn có thể xoá nội dung của thanh ghi hay ô nhớ.

Dưới đây là một số ví dụ minh hoạ:

+ Để xoá bit dấu của AL trong khi giữ nguyên các bit còn lại. Ta sử dụng lệnh AND, mặt nạ tìm được là 01111111B (7FH) như sau:

```
AND AL,7FH
```

+ Để thiết lập bit có trọng số cao nhất và bit có trọng số thấp nhất của thanh ghi AH trong khi giữ nguyên các bit còn lại. Ta sử dụng lệnh OR với mặt nạ 10000001B = 81H như sau:

```
OR AH,81H
```

+ Đổi từ mã ASCII của số sang số

Từ bảng mã ASCII ta thấy mã ASCII của các chữ số từ '0' đến '9' là từ 30H đến 39H. Như vậy, mã ASCII của chữ số hơn chính số đó là 30H. Ta có thể sử dụng lệnh AND hoặc SUB để đổi từ mã ASCII sang số. Ví dụ: nếu ta nhập vào từ bàn phím chữ số '7' thì nội dung của thanh ghi AL là 37H, để nội dung chứa trong AL chính là giá trị 7 nhập vào thì ta phải sử dụng một trong hai lệnh sau:

```
and AL,0FH ; xoá 4 bit cao
```

Hoặc

```
sub AL,30H
```

+ Đổi chữ thường thành chữ hoa hoặc chữ hoa thành chữ thường

Mã ASCII của các chữ thường từ 'a' đến 'z' nằm từ 61H đến 7AH, mã ASCII của các chữ hoa từ 'A' đến 'Z' nằm từ 41H đến 5AH. Như vậy muốn chuyển mã ASCII của chữ thường sang hoa chỉ cần trừ đi 20H (00100000B). Nếu so sánh mã ASCII dạng nhị phân của các chữ thường và chữ hoa tương ứng sẽ thấy:

Ký tự	Mã ASCII	Ký tự	Mã ASCII
a	01100001	A	01000001
b	01100010	B	01000010
...	...	...	...
z	01111010	Z	01011010

Để chuyển chữ thường thành hoa ta chỉ việc xoá bit 5 bằng cách thực hiện lệnh AND với mặt nạ 11011111B (0DFH). Ví dụ AL chứa mã ASCII của chữ thường, muốn chuyển thành chữ hoa ta dùng lệnh sau:

```
and AL,0DFH ; xoá bit 5 của AL
```

Hoặc

```
sub AL,20H
```

Ngược lại, muốn đổi chữ hoa thành thường chỉ cần thiết lập bit thứ 5, giữ nguyên các bit khác bằng cách thực hiện lệnh OR với mặt nạ 00100000B (20H). Giả sử AL chứa mã ASCII của chữ hoa để chuyển sang chữ thường ta dùng lệnh sau: `or AL, 20H ; thiết lập bit 5 của AL`

Hoặc: `add AL,20H`

+ Xoá một thanh ghi

Để xoá một thanh ghi ta có thể dùng một trong các cách sau:

```
MOV CX,0
SUB CX,CX
XOR CX,CX
```

CX sẽ được xoá về 0 nếu ta sử dụng một trong các lệnh trên.

+ Kiểm tra một thanh ghi hay ô nhớ chứa giá trị chẵn hay lẻ

Để kiểm tra một thanh ghi hay một ô nhớ chứa số chẵn hay lẻ, ta có thể sử dụng lệnh AND với mặt nạ là 1. Nếu kết quả là 0 thì là số chẵn, ngược lại là số lẻ. Ví dụ kiểm tra AL là chẵn hay lẻ ta có thể sử dụng lệnh TEST hoặc AND như sau:

```
TEST AL,01H
And  Al,01H
```

Nếu ZF = 0 thì AL là số chẵn, ngược lại AL là số lẻ.

b) Lệnh NOT (logical NOT)

Dạng lệnh:

```
NOT     đích
```

Thao tác: Tạo dạng bù 1 của toán hạng, xem bảng 3.5.

Cờ: Không bị ảnh hưởng.

Ví dụ nội dung của thanh ghi AL bằng 11110000B, sau khi thực hiện lệnh:

```
NOT     AL
```

nội dung của thanh ghi AL bằng 00001111B.

3.4.4.2. Các lệnh dịch và quay

a) Các lệnh dịch

Các lệnh dịch sẽ dịch các bit trong toán hạng sang trái hoặc sang phải một hay nhiều vị trí. Các bit bị dịch ra ngoài sẽ được đưa vào cờ nhớ.

+ Lệnh SAL/SHL (Shift Arithmethic Left/SHift logical Left)

Dạng lệnh:

```
SHL/SAL     đích,1
```

Hoặc:

```
SHL/SAL     đích,cl
```

Thao tác: Dịch các bit trong toán hạng đích sang trái một hay nhiều vị trí. Bit bên trái nhất sẽ được dịch vào cờ nhớ, bit bên phải nhất sẽ được đưa 0 vào.

Nếu dịch trái một vị trí ta dùng lệnh thứ nhất, để dịch nhiều hơn một lần số lần dịch được đặt trong CL. Khi bộ đếm CL = 1 và MSB của toán

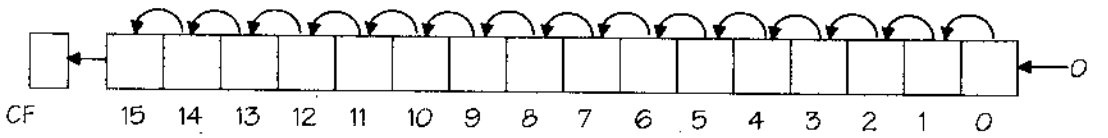
hạng đích bị thay đổi so với trước khi quay thì cờ OF được thiết lập 1, nếu chúng bằng nhau thì cờ OF bị xoá. Nếu bộ đếm khác 1, OF không xác định, CL không thay đổi khi lệnh thực hiện xong.

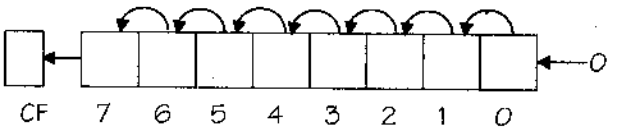
Lệnh SAL và SHL tạo ra cùng một mã máy.

Cờ: Bị ảnh hưởng CF, OF, PF, SF, ZF.

Không xác định AF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:

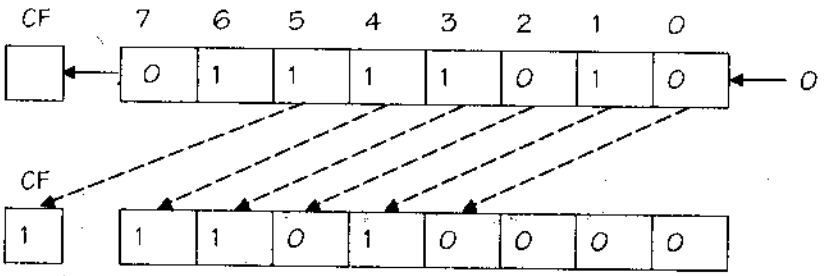




Ví dụ 1: Giả sử thanh ghi AH chứa 7AH (01111010B), nếu thực hiện lệnh SHL AH,1, nội dung trong AH nhận được bằng cách xoá đi 1 bit phần cao của AH và thêm 1 bit 0 vào phần thấp. Kết quả là AH = 11110100B, cờ CF = 0. Nếu muốn dịch trái AH ba vị trí ta thực hiện các lệnh như sau (giả sử thanh ghi AH chứa 7AH):

```
mov CL,3          ; CL chứa số lần dịch
SHL AH,CL
```

Sau khi thực hiện lệnh `SHL AH,CL`, nội dung trong AH nhận được bằng cách xoá đi ba bit phần cao của AH và thêm ba bit 0 vào phần thấp. Lệnh dịch nhiều bit thực chất là thực hiện nhiều lệnh dịch một bit vì thế có thể thay hai lệnh trên bằng ba lệnh dịch một bit liên tiếp, cờ CF chứa bit cuối cùng dịch ra khỏi toán hạng.



Kết quả là AH = 11010000B, cờ CF chứa 1.

Khi dịch toán hạng sang trái một bit sẽ tương ứng nhân số đó với 2. Ví dụ AH = 00000011B (3), dịch trái một bit ta được 00000110B (6) nếu dịch trái tiếp ta sẽ được 00001100B (12). Như vậy, khi dịch trái toán hạng n vị trí tương ứng nhân toán hạng đó với 2^n. Hiện tượng tràn có thể xảy ra khi chúng ta sử dụng lệnh dịch trái để thực hiện phép nhân.

+ Lệnh SHR (SHift logical Right)

Dạng lệnh: `SHR  đích,1`

Hoặc: `SHR  đích,CL`

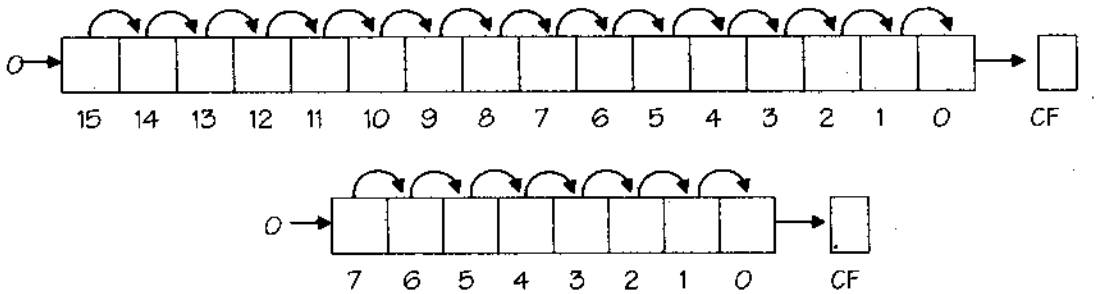
Thao tác: Dịch các bit trong toán hạng đích sang phải một hay nhiều vị trí. CF nhận được bit LSB và 0 được dịch vào bit MSB.

Nếu dịch một vị trí ta dùng lệnh thứ nhất, để dịch nhiều hơn một lần số lần dịch được đặt trong CL. Khi bộ đếm bằng 1 và 2 bit trái nhất của toán hạng đích bằng nhau thì cờ OF bị xoá. Nếu chúng khác nhau, OF thiết lập 1. Bộ đếm khác 1, OF không xác định, CL không thay đổi.

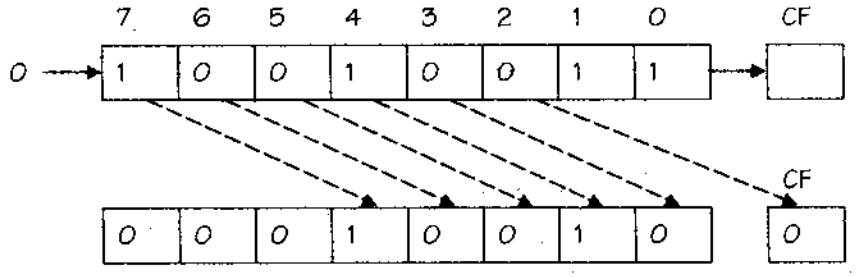
Cờ: Bị ảnh hưởng CF, OF, PF, SF, ZF.

Không xác định AF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:



Ví dụ 2: Giả sử CL chứa 3, AH chứa 93H, sau khi thực hiện lệnh SHL AH, CL nội dung trong AH nhận được bằng cách xoá đi ba bit phần thấp và thêm ba bit 0 vào phần cao của AH.



Kết quả là AH = 00001111B, cờ CF chứa bit cuối cùng dịch ra khỏi toán hạng.

Ngược lại với phép dịch trái, khi dịch phải toán hạng một bit sẽ tương ứng chia số đó cho 2, khi dịch phải toán hạng n vị trí tương ứng chia toán hạng đó cho 2^n. Ví dụ AH = 10100000B (160), dịch phải một bit ta được 01010000B (80) nếu dịch phải tiếp ta sẽ được 00001100B (40). Nhưng điều này chỉ đúng với các số chẵn, còn đối với các số lẻ, dịch phải sẽ chia đôi nó và làm tròn xuống số nguyên gần nhất.

+ Lệnh SAR (Shift Arithmetically Right)

Dạng lệnh: `SAR đích,1`

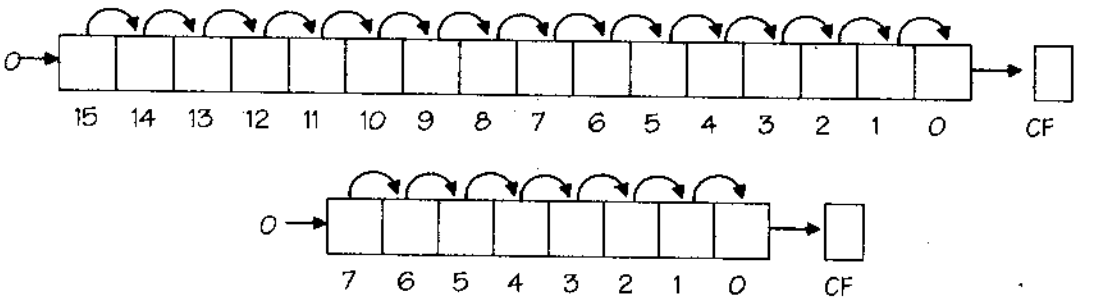
Hoặc: `SAR đích,CL`

Thao tác: Dạng đầu dịch toán hạng đích 1 lần, CF nhận được bit LSB và bit MSB được giữ nguyên. Để dịch nhiều hơn 1 lần, số lần dịch được đặt trong CL. OF luôn bằng 0, CL không thay đổi.

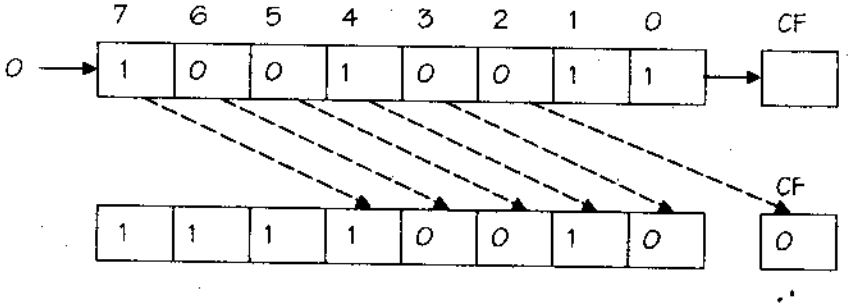
Cờ: Bị ảnh hưởng CF, OF, PF, SF, ZF.

Không xác định AF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:



Ví dụ 3: Giả sử CL chứa 3, AH chứa 10010011B (93H), sau khi thực hiện lệnh SAR AH, CL nội dung trong AH nhận được bằng cách xoá đi ba bit phần thấp và giữ nguyên bit cao nhất (bit 7) sau mỗi lần dịch. Bit MSB của AH là 1 nên nội dung AH được thêm ba bit 1 vào phần cao.



Khi chia bằng phép dịch phải sẽ có sự khác biệt giữa phép chia với các số không dấu và phép chia với các số có dấu. Nếu ta thực hiện phép chia với các số không dấu ta sử dụng lệnh SHL, còn khi thực hiện phép chia với các số có dấu ta phải sử dụng SAR vì lệnh này giữ nguyên dấu, sử dụng lệnh SHR sẽ làm mất dấu của toán hạng.

Ví dụ 4: Giả sử CL chứa –10 (11110101B), sau khi thực hiện lệnh SAR CL,1 nội dung trong CL nhận được bằng –5 (11111010B).

Lệnh dịch chỉ có thể thực hiện được phép nhân và chia với các số là luỹ thừa của 2, muốn nhân với các số bất kỳ ta phải sử dụng lệnh nhân và lệnh chia trong phần sau.

b) Các lệnh quay

Lệnh quay thực hiện dịch tất cả các bit của toán hạng sang trái hoặc phải một hay nhiều vị trí, bit dịch ra khỏi toán hạng sẽ được đưa vào cờ nhớ đồng thời đưa trở lại.

+ Lệnh ROL (ROtate Left)

Dạng lệnh: `ROL  đích,1`

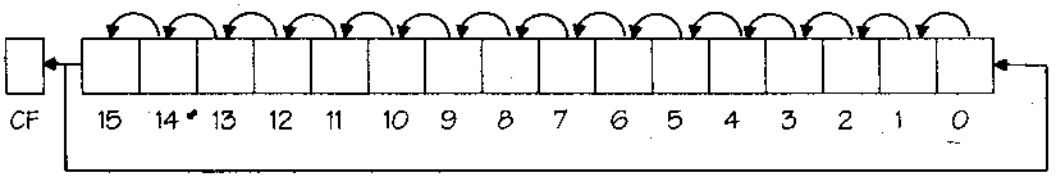
Hoặc: `ROL  đích,CL`

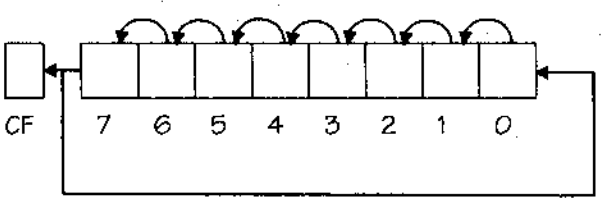
Thao tác: Lệnh ROL thực hiện dịch tất cả các bit của toán hạng đích sang trái một hay nhiều vị trí, bit MSB dịch ra khỏi toán hạng sẽ được đưa vào cờ nhớ đồng thời đưa trở lại vị trí bit LSB.

Nếu quay trái một vị trí ta dùng lệnh thứ nhất, nếu quay nhiều hơn một lần, số lần quay được đặt trong CL. Khi bộ đếm bằng 1 và MSB của toán hạng đích bị thay đổi so với trước khi quay thì cờ OF được thiết lập 1. Nếu bộ đếm khác 1, OF không xác định, CL không thay đổi.

Cờ: Bị ảnh hưởng, ngoại trừ OF, CF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:





+ Lệnh ROR (ROtate Right)

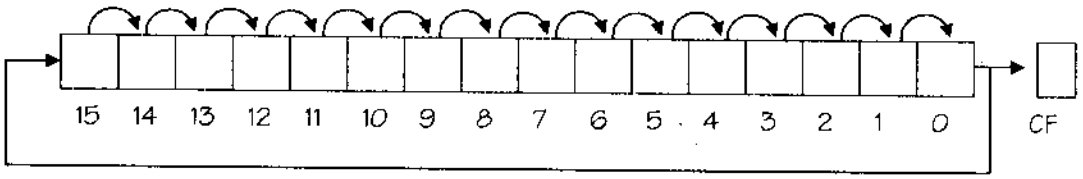
Dạng lệnh: ROR đích,1

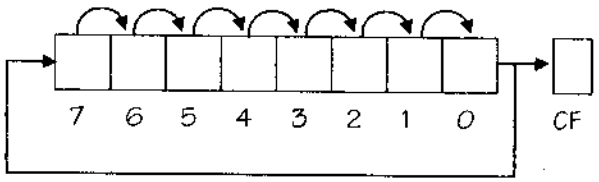
Hoặc: ROR đích,CL

Thao tác: Ngược với lệnh ROL, ROR thực hiện dịch tất cả các bit của toán hạng đích sang phải một hay nhiều vị trí, bit LSB dịch ra khỏi toán hạng sẽ được đưa vào cờ nhớ đồng thời đưa trở lại vị trí bit MSB.

Cờ: Bị ảnh hưởng, ngoại trừ OF, CF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:





Ví dụ 5: Hiển thị từng bit trong một toán hạng mà không làm thay đổi nội dung của chúng. Giả sử hiển thị các bit của thanh ghi AX.

```
        MOV CX,16       ;đếm số bit cần hiển thị
        MOV AH,2        ;hàm hiển thị
top:                    ;CF chứa bit MSB dịch đến
        ROL BX,1        ;CF = 1 xuống hiển thị '1'
        JC sol          ;CF = 0
        MOV DL,'0'      ;hiển thị 0
        INT 21H
        JMP thoat       ;bỏ qua hiển thị '1'
sol:
        MOV DL,'1'
        INT 21H         ;hiển thị '1'
thoat:  Loop top        ;lặp lại 16 lần
```

Muốn hiển thị từng bit trong một toán hạng thì ta phải lần lượt đưa được các bit của toán hạng đó ra cờ nhớ. Sử dụng lệnh nhảy JC hoặc JNC, nếu có nhớ (JC = 1) thì bit dịch ra là bit 1, khi đó hiển thị '1', ngược lại hiển thị '0'. Thực hiện lặp 16 lần ta in ra được 16 bit của toán hạng. Lệnh ROL sau khi thực hiện xong vòng lặp giá trị của toán hạng vẫn giữ nguyên.

Để ngắn gọn hơn, ta có thể dùng lệnh ADC như sau:

```
        MOV CX,16       ; lặp 16 lần
        MOV AH,2        ; hàm hiển thị
        MOV DL,30H
top:
        ROL BX,1
        ADC DL,0        ; DL = 30H + 0 + CF
        INT 21H         ; hiển thị '1' hoặc '0'
        loop top        ; lặp lại nếu CX = 0
```

+ Lệnh RCL (Rotate through Carry Left)

Dạng lệnh:

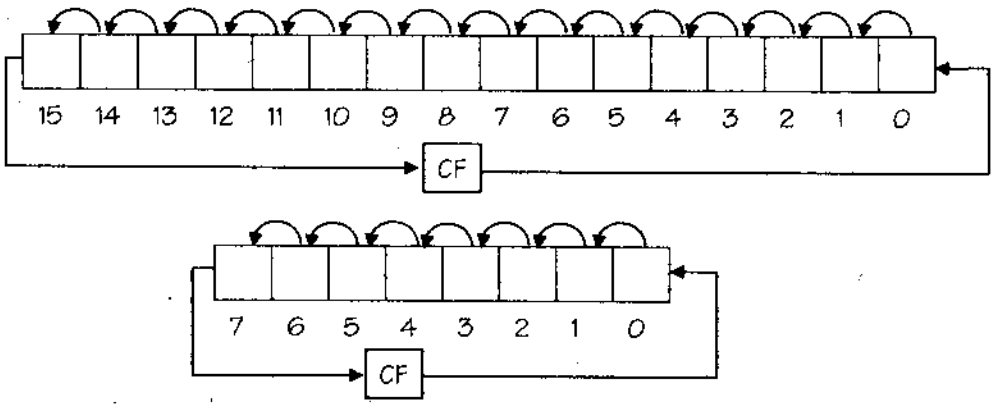
```
RCL   đích,1
```

Hoặc

```
RCL   đích,cl
```

Thao tác: Quay các bit của toán hạng đích sang trái qua cờ CF một hoặc nhiều lần.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:



+ Lệnh RCR (Rotate through Carry Right)

Dạng lệnh:

```
RCR     đích,1
```

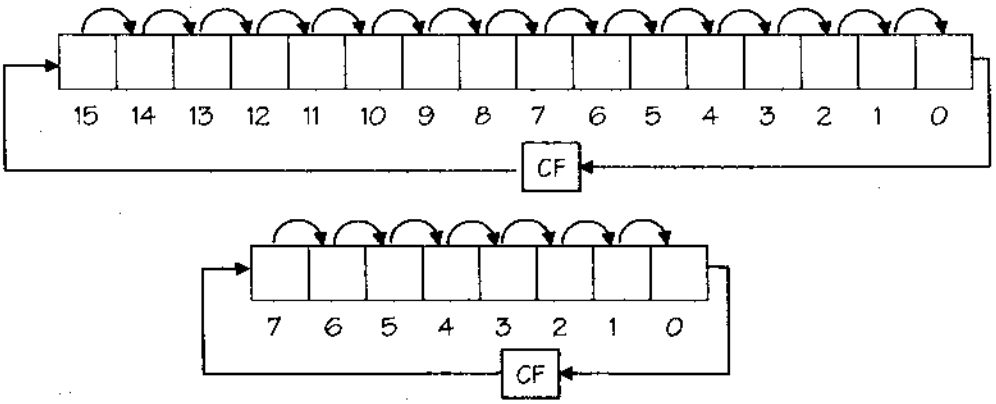
Hoặc:

```
RCR     đích,cl
```

Thao tác: Quay các bit của toán hạng đích sang phải qua cờ CF một hoặc nhiều lần. Bit LSB của toán hạng đích được đặt vào CF và nội dung cũ của CF được chuyển vào bit MSB. Lệnh quay nhiều bit thực chất là thực hiện nhiều lệnh dịch một bit, số lần dịch chứa trong CL.

Cờ: Bị ảnh hưởng: CF, OF.

Mô tả: Với toán hạng là một từ hoặc một byte như sau:

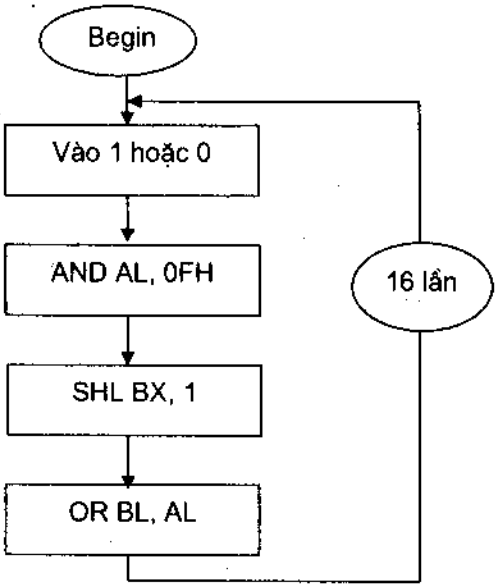


Ví dụ 6: Để đảo các mẫu bit trong một byte hoặc một từ, ta có thể sử dụng kết hợp cả lệnh quay và lệnh dịch. Dùng lệnh dịch trái SHL để dịch các bit phía bên trái ra khỏi toán hạng, sau đó dùng lệnh RCR để đưa vào một thanh ghi khác. Đoạn chương trình như sau sẽ đảo ngược mẫu bit trong thanh ghi BX:

```
        MOV CX,16       ; lặp 16 lần
top:
        SHL BX,1        ; dịch bit MSB sang cờ CF
        ROR AX,1        ; quay bit trong CF vào AX
        loop top        ; lặp lại nếu CX = 0
        MOV BX,AX       ; BX chứa mẫu bit đảo ngược
```

3.4.4.3. Ứng dụng của các lệnh thao tác bit

a) Vào/ra số nhị phân



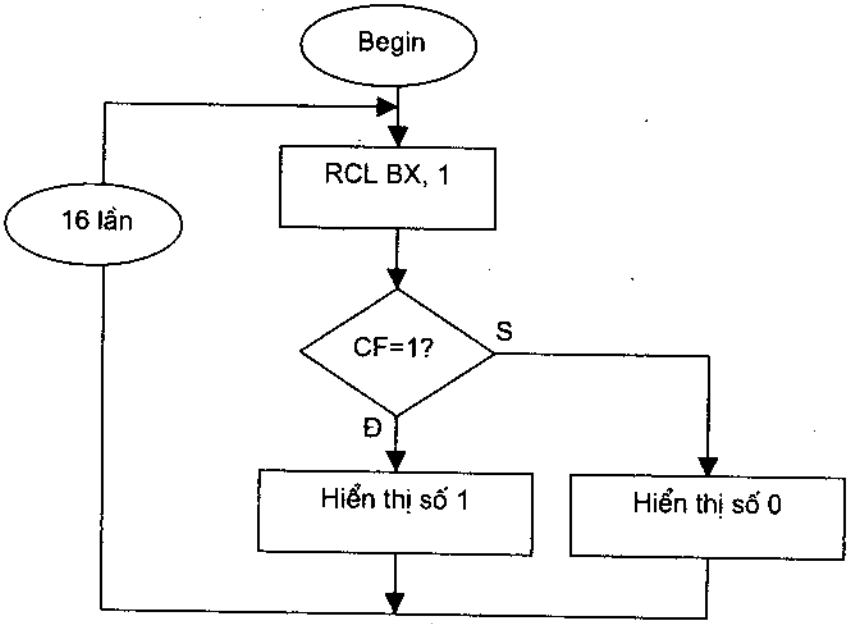
Hình 3.21. Thuật toán vào một số nhị phân 16 bit từ bàn phím

Để nhập vào từ bàn phím một số nhị phân (10110100B) và lưu số đó trong thanh ghi BX, thực chất ta phải nhập liên tiếp một dãy các ký tự '0' và '1'. Mỗi một ký tự '0' hoặc '1' được nhập thì AL chứa 30H hoặc 31H, ta phải đổi chúng thành giá trị của từng bit rồi dịch các bit vào trong thanh ghi.

Đoạn mã sau thực hiện nhập một số nhị phân vào từ bàn phím và lưu vào BX.

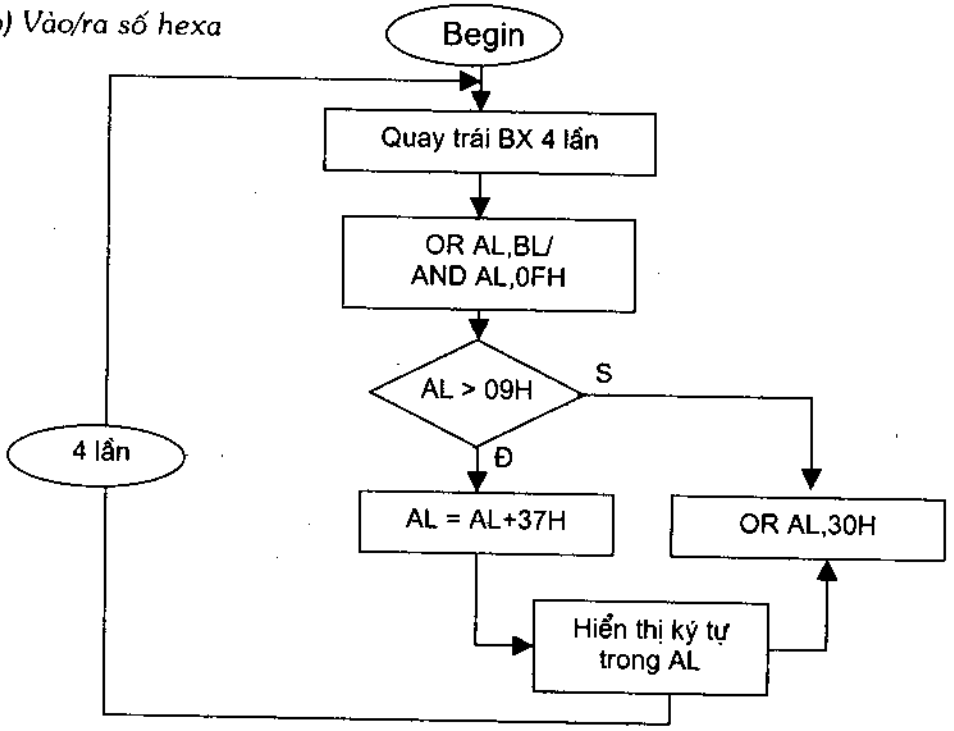
```
        mov BX,0                ;đếm số bit cần hiển thị
        mov CX,16               ;hàm hiển thị
      lap:                      ;CF chứa bit MSB dịch đến
        mov AH,1                ;CF = 1 xuống hiển thi '1'
        int 21H                 ;CF = 0
        and AL,0FH              ;hiển thị 0
        shl BX,1
        or BL,AL                ;bỏ qua hiển thị '1'
        loop lap                ;hiển thị '1'
      thoat:                    ;lặp lại 16 lần
```

Thuật toán sau hiển thị đưa ra nội dung thanh ghi BX dưới dạng số nhị phân (đoạn mã chương trình đã được trình bày trong ví dụ 5, mục 3.4.3.3).



Hình 3.22. Thuật toán đưa ra một số nhị phân 16 bit từ thanh ghi BX

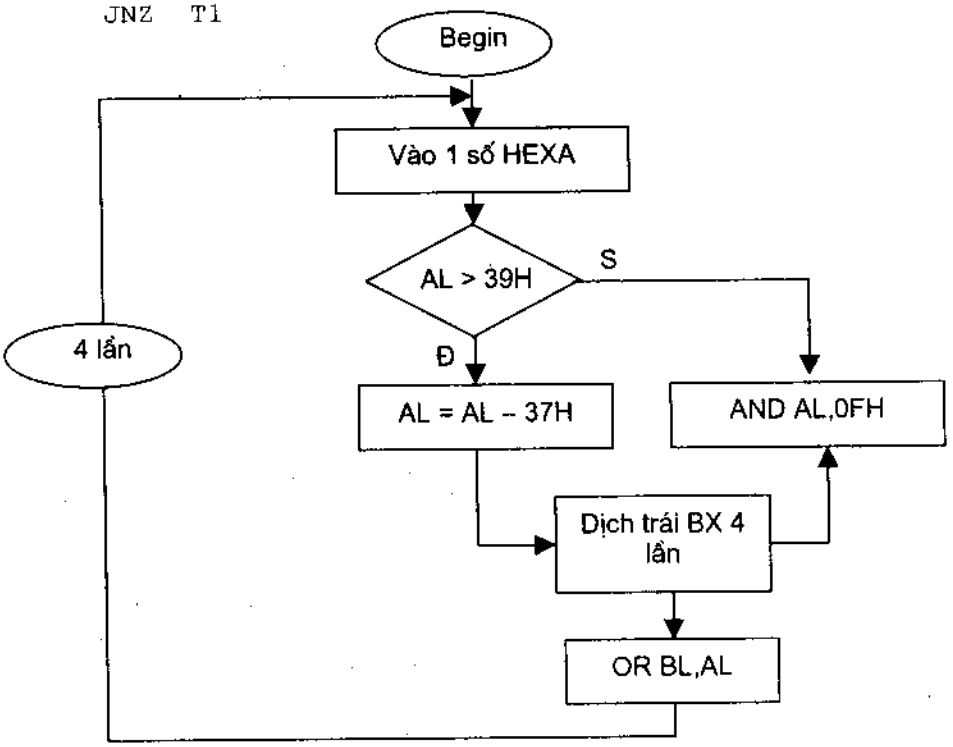
b) Vào/ra số hexa



Hình 3.23. Thuật toán đưa ra một số hexa

Đoạn mã chương trình sau đây hiển thị nội dung của thanh ghi BX dưới dạng số hexa.

```
        MOV   DH,4
        MOV   AH,02
        MOV   CL,04
T1:     ROR   BX,CL
        MOV   DL,BL
        AND   DL,0FH
        ADD   DL,30H
        CMP   DL,39H
        JNG   T2
        ADD   DL,07H
T2:     INT   21
        DEC   DH
        JNZ   T1
```



Hình 3.24. Thuật toán vào một số hexa từ bàn phím

Đoạn mã chương trình sau đây cho phép nhập một số hexa vào thanh ghi BX.

```
        XOR   BX,BX
        MOV   DH,4
T1:
        MOV   AH,1
        INT   21H
        SUB   AL,30H
        CMP   AL,9H
        JNG   T2
T2:     SUB   DL,07H

        MOV   CL,04
        SHL   BX,CL
        OR    BL,AL
        DEC   DH
        JNZ   T1
```

3.4.5. Nhóm lệnh xử lý điều khiển

3.4.5.1. ESC (ESCape)

Cho phép các bộ vi xử khác, chẳng hạn 8087, thực hiện các thao tác của chúng. Bộ vi xử lý 8086 không thực hiện thao tác ngoại trừ việc lấy các toán hạng bộ nhớ cho các bộ xử lý khác.

Dạng lệnh: `ESC extenal_opcode, soure`

Cờ: Không.

3.4.5.2. Lệnh HLT (HaLT)

Đưa bộ vi xử lý vào trạng thái dừng để chờ ngắt ngoài.

Dạng lệnh: `HLT`

Cờ: Không.

3.4.5.3. Lệnh LOCK (khoá bus)

Khoá bus trong môi trường có nhiều bộ vi xử lý.

Dạng lệnh: LOCK

Thao tác: Lock có thể dùng như một tiền tố đặt trước bất cứ một lệnh nào. Bus sẽ bị khoá trong thời gian thực hiện lệnh để ngăn các bộ vi xử lý khác truy cập bộ nhớ.

Cờ: Không ảnh hưởng.

3.4.5.4. Lệnh NOP (No OPeration)

Dạng lệnh: NOP

Thao tác: Không thực hiện một thao tác nào.

Cờ: Không bị ảnh hưởng.

3.4.5.5. Lệnh STI (SeT Interrupt flag)

Dạng lệnh: STI

Thao tác: IF được thiết lập 1.

Cờ: Bị ảnh hưởng IF.

3.4.5.6. Lệnh WAIT

Dạng lệnh: WAIT

Thao tác: Bộ vi xử lý ở trạng thái chờ cho đến khi được kích hoạt bằng một lệnh ngắt ngoài.

Cờ: Không bị ảnh hưởng.

3.4.6. Nhóm lệnh xử lý chuỗi

3.4.6.1. Lệnh MOVS/MOVSB/MOVSW (MOVeS/MOVeS Byte/ MOVeS Word)

Chuyển dữ liệu trong bộ nhớ tại địa chỉ DS: SI vào ô nhớ có địa chỉ ES: DI bằng cách dùng tiền tố REP, có thể chuyển đồng thời nhiều byte hay word.

Dạng lệnh:

```
        MOVS      chuỗi_đích, chuỗi_nguồn
hoặc    MOVSB
hoặc    MOVSW
```

Thao tác: Chuỗi byte (hay word) được chuyển tới toán hạng đích. Cả DI và SI đều tăng lên 1 (hoặc 2 đối với word) nếu DF = 0; giảm đi 1 (hoặc 2 đối với word) nếu DF = 1.

Cờ: Không bị ảnh hưởng.

3.4.6.2. Lệnh CMPS/CMPSB/CMPSW (COMPareS/ COMPareS Byte/ COMPareS Word)

So sánh hai toán hạng là một chuỗi byte hay word.

Dạng lệnh:

```
        CMPS      chuỗi_đích, chuỗi_nguồn
hoặc    CMPSB
hoặc    CMPSW
```

Thao tác: Trừ các thành phần của chuỗi nguồn (được chỉ số bằng DS: SI) cho các thành phần chuỗi đích (được chỉ số bằng ES: SI). Các cờ trạng thái được thiết lập theo kết quả của phép tính. Nếu cờ định hướng DF = 0 thì cả SI và DI đều tăng sau mỗi phép tính (1 khi so sánh byte và 2 khi so sánh word) trong trường hợp còn lại cả SI và DI đều giảm (1 khi so sánh byte và 2 khi so sánh word).

Cờ: Bị ảnh hưởng, ngoại trừ AF, CF, SF, ZF, PF, OF.

3.4.6.3. Lệnh LODS/LODSB/LODSW (LOaDS/ LOaDS Byte/ LOaDS Word)

Chuyển nội dung của byte hay word từ bộ nhớ được chỉ ra bởi SI vào thanh chứa.

Dạng lệnh:

```
        LODS      chuỗi_nguồn
hoặc    LODSB
hoặc    LODSW
```

Thao tác: Byte (hay word) được nạp vào AL (hay AX). SI tăng lên 1 hay 2 nếu DF = 0; ngược lại SI giảm đi 1 (hay 2) nếu DF = 1.

Cờ: Không bị ảnh hưởng.

3.5. NGĂN XẾP VÀ THỦ TỤC

3.5.1. Ngăn xếp

Ngăn xếp là nơi lưu trữ dữ liệu tạm thời. Ngăn xếp hoạt động theo kiểu 'vào trước, ra sau' (LIFO: Last In, First Out). Phần tử cuối cùng cất vào ngăn xếp sẽ là đỉnh của ngăn xếp và phần tử này sẽ được lấy ra đầu tiên.

3.5.1.1. Lệnh PUSH (PUSH word to stack)

Cất một từ dữ liệu vào ngăn xếp.

Dạng lệnh:

```
PUSH   nguồn
```

Thao tác: Lệnh PUSH nguồn thực hiện giảm SP đi 2, sau đó chuyển bản sao của toán hạng nguồn ngăn xếp có địa chỉ SS: SP.

Cờ: Không bị ảnh hưởng.

3.5.1.2. Lệnh PUSHF (PUSH Flag in to stack)

Chuyển thanh ghi cờ vào ngăn xếp.

Dạng lệnh:

```
PUSHF
```

Thao tác: Giảm SP đi và chuyển các bit cờ vào đỉnh ngăn xếp.

Cờ: Không bị ảnh hưởng.

3.5.1.3. Lệnh POP (Pop word of stack to destinnation)

Lấy dữ liệu ra khỏi ngăn xếp và đưa vào toán hạng đích.

Dạng lệnh:

```
POP   đích
```

Thao tác: Nội dung của toán hạng đích được thay vào bằng từ nằm ở đỉnh ngăn xếp, con trỏ ngăn xếp tăng lên 2.

Cờ: Không bị ảnh hưởng.

3.5.1.4. Lệnh POPF (POP word from top of stack to Flag register)

Dạng lệnh:

```
POPF
```

Thao tác: Chuyển nội dung của 2 byte từ đỉnh của ngăn xếp vào thanh ghi cờ, sau đó tăng con trỏ ngăn xếp lên 2.

Cờ: Tất cả các cờ bị ảnh hưởng.

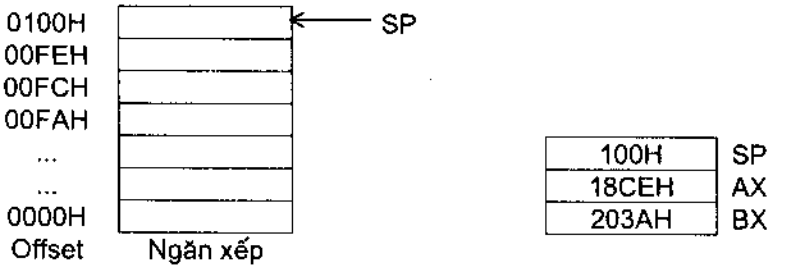
Chú ý: Tất cả các lệnh PUSH, POP đều thao tác với các từ là ô nhớ hay thanh ghi, nếu ta dùng với các byte sẽ không hợp lệ, *Ví dụ:*

```
PUSH    AL              ; không hợp lệ
POP     BL              ; không hợp lệ
PUSH    2000            ; không hợp lệ
```

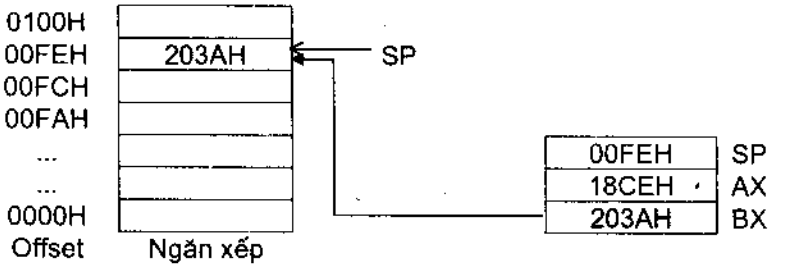
Mỗi chương trình phải được dành ra một khối lượng bộ nhớ để làm ngăn xếp, công việc đó được thực hiện bằng thủ tục khai báo:

```
.STACK  100H
```

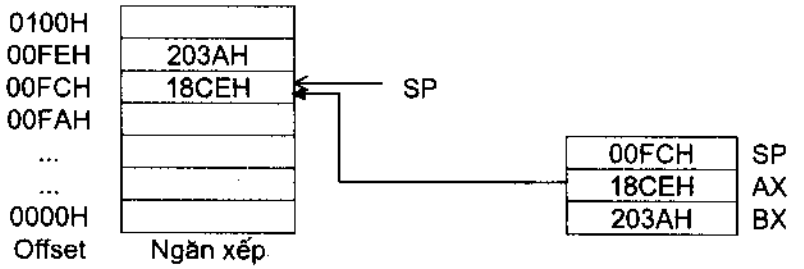
Trong khai báo trên, con trỏ SP được khởi tạo bằng 100H, đây chính là đỉnh hiện thời của ngăn xếp khi ngăn xếp chưa được sử dụng. Dưới đây mô tả hoạt động của ngăn xếp.



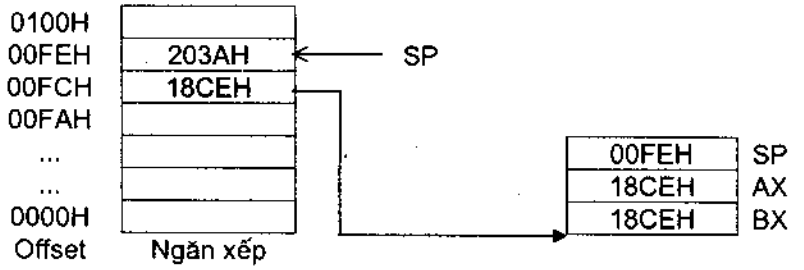
Hình 3.25. Ngăn xếp khi chưa sử dụng



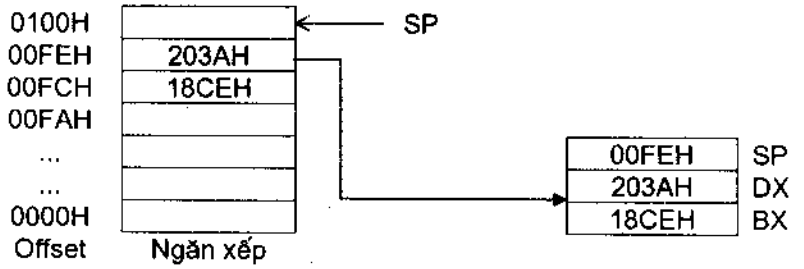
Hình 3.26a. Ngăn xếp sau lệnh PUSH BX



Hình 3.26b. Ngăn xếp sau lệnh PUSH AX (trước lệnh POP BX)



Hình 3.27a. Ngăn xếp sau lệnh POP BX



Hình 3.27b. Ngăn xếp sau lệnh POP DX

Ví dụ: Nhập vào 4 số tự nhiên từ 0 đến 9, tính tổng của chúng, chứa kết quả trong BL.

Dưới đây là chương trình đầy đủ:

```
.model small
.stack 100H
.data
.code
main proc
```

```
XOR AX,AX                ; xoá AX
MOV CX,4                 ; vòng lặp CX = 4 lần nhập
top1:
MOV AH,1                 ; hàm nhập một ký tự
INT 21H                  ; thực hiện nhập
AND AL,0FH               ; đổi mã ký tự thành số
PUSH AX                  ; cất vào ngăn xếp
loop top1
XOR AX,AX                ; xoá AX
MOV CX,4                 ; 4 lần nhập
top2:
POP AX                   ; lấy ký tự trong ngăn xếp
ADD BL,AL
loop top2                ; BL đã chứa tổng của 4 số
main endp
end main
```

3.5.2. Thủ tục

Hàm và thủ tục trong hợp ngữ có thể coi là chương trình con trong các ngôn ngữ bậc cao hơn. Từ chương trình chính có thể dùng lệnh CALL để gọi các hàm, thủ tục. Kết thúc các hàm, thủ tục phải có các lệnh RET hoặc IRET để trở về chương trình chính.

Cú pháp khai báo một thủ tục như sau:

```
name    proc    type
; các lệnh trong thân của thủ tục
ret
name endp
```

Trong đó NAME là tên thủ tục. TYPE là kiểu thủ tục, có hai kiểu thủ tục là kiểu NEAR và kiểu FAR. Nếu các dòng lệnh gọi thủ tục ở cùng một đoạn với thủ tục đó thì sẽ là kiểu NEAR, ngược lại sẽ là FAR. Nếu bỏ qua TYPE thì sẽ ngầm định là kiểu NEAR.

3.5.2.1. Lệnh CALL (gọi thủ tục)

Dạng lệnh:

```
CALL name
```

Thao tác: Địa chỉ offset của lệnh ngay sau lệnh CALL là địa chỉ trở về của chương trình được cất vào ngăn xếp. Điều khiển được chuyển cho thủ tục, IP chứa địa chỉ offset của lệnh đầu tiên trong thủ tục.

Cờ: Không bị ảnh hưởng.

3.5.2.2. Lệnh IRET (Interrupt RETurn)

Dạng lệnh:

```
IRET
```

Thao tác: Lấy ra khỏi ngăn xếp nội dung của CS, IP và thanh ghi flags.

Cờ: Tất cả các cờ đều bị ảnh hưởng.

3.5.2.3. Lệnh RET (RETurn from procedure)

Trả lại điều khiển khi thủ tục được thực hiện xong.

Dạng lệnh: RET

Thao tác: Nếu lệnh RET một thủ tục NEAR, nó được dịch thành một lệnh trở về trong cùng đoạn, lệnh này cập nhật thanh ghi IP bằng cách lấy một từ ra khỏi ngăn xếp.

Cờ: Không bị ảnh hưởng.

Sau đây là một ví dụ dùng thủ tục:

```
.model small
.stack 100H
.data
x_dong DB 13,10
.code
     main proc
mov AX,@data          ; khởi tạo thanh ghi DS
mov DS,AX
call nhap             ; nhập số thứ nhất vào BX
push BX               ; cất số thứ nhất vào ngăn xếp
```

```
mov AH,9                ; xuống dòng về đầu dòng
lea DX,x_dong
int 21H
call nhap               ; nhập số thứ hai vào BX
pop AX                  ; lấy số thứ nhất đặt vào AX
add BX,AX               ; cộng hai số lưu kết quả ở BX
mov AH,9                ; xuống dòng về đầu dòng
lea DX,x_dong
int 21H
mov CX,16               ; hiển thị kết quả dưới dạng
mov AH,2                ; nhị phân
mov DL,30H
lap2:
rol BX,1
adc AL,0
int 21H
loop lap2
     mov AH,4CH
     int 21H
     main endp
nhap proc  near         ; chương trình con nhập vào
     mov BX,0000H       ; một số nhị phân
     mov CX,10H
lap1:
     mov AH,1
     int 21H
     cmp AL,0DH
     jne thoat
     and AL,0FH
     shl BX,1
```

```
        or BL,AL
        loop lap1
thoat:
        ret
nhap endp
end main
```

BÀI TẬP CHƯƠNG 3

1. Cho biết kết quả của thanh ghi AX khi vi xử lý thực hiện xong đoạn chương trình sau:

a)
```
MOV BH,0
MOV AX,0
T:
ADD BH,1
MUL BH
CMP BH,45
JNE T
```

c)
```
MOV AH,1
INT 21H
MOV BL,AL
MOV AH,2
MOV DL,0DH
INT 21H
MOV DL,0AH
INT 21H
MOV DL,BL
```

b)
```
MOV AX,1
MOV CX,5
TOP:
MUL CL
   LOOP TOP
```

d)
```
MOV AH,1
INT 21H
MOV CL,AL
CMP AL,61H
JNZ T
MOV AH,2
MOV DL,CL
INT 21H
JMP TH
T:CMP AL,41H
JNZ TH
MOV AH,2
MOV DL,CL
INT 21H
INT 21H
TH:
```

2 . Viết chương trình tìm MAX của 2 số không bằng nhau, cất vào ngăn xếp số lớn, giả sử 2 số đang nằm trong AL và AH.

3 . Viết chương trình thực hiện phép toán cộng và hiển thị kết quả dưới dạng số nhị phân.

AX = 0 + 1 + 2 +...+ 255.

4 . Viết chương trình thực hiện phép toán cộng và hiển thị kết quả dưới dạng số hexa.

AX = 1 + 5 + 10 +...+ 100.

5 . Viết chương trình thực hiện phép toán AX = 7!, sau đó hiển thị kết quả dưới dạng số nhị phân hoặc số thập phân.

6 . Viết chương trình thực hiện phép toán AX = n! (0 < n < 10), n được nhập vào từ bàn phím, sau đó hiển thị kết quả dưới dạng số thập phân.

7 . Viết chương trình thực hiện nhập liên tiếp một dãy các ký tự khi nào gặp enter thì kết thúc. Hiển thị dãy ký tự theo chiều ngược lại.

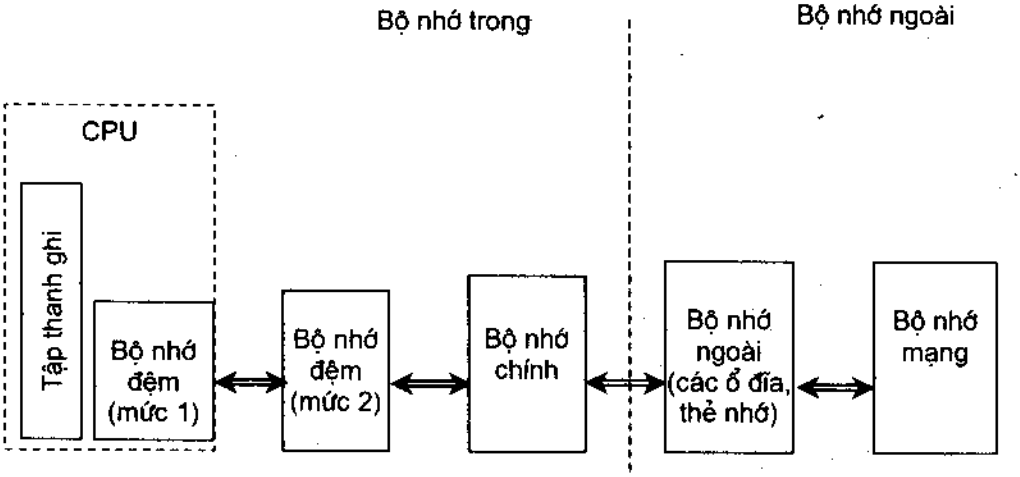
8 . Viết chương trình thực hiện nhập vào hai số hexa từ bàn phím, tính tổng của hai số và hiển thị tổng dưới dạng số hexa.

9 . Viết chương trình thực hiện nhập vào hai số hexa từ bàn phím, tính tích của hai số và hiển thị tích dưới dạng số hexa.

Chương 4. BỘ NHỚ VÀ HỆ THỐNG LƯU TRỮ

4.1. NGUYÊN TẮC TỔ CHỨC BỘ NHỚ

Bộ nhớ máy tính (memory) là nơi lưu trữ chương trình và dữ liệu, gồm bộ nhớ trong và bộ nhớ ngoài. Bộ nhớ trong (internal memory) là các vi mạch nhớ bán dẫn có tốc độ truy cập cao, năng lượng tiêu thụ thấp nhưng dung lượng nhỏ. Vì vậy, máy tính cần phải có thêm các bộ nhớ ngoài (external memory) để chứa chương trình, dữ liệu có dung lượng lớn của người sử dụng. Trong máy tính, bộ nhớ được phân chia thành 6 cấp như hình 4.1.



Hình 4.1. Phân cấp hệ thống nhớ trong máy tính

– ***Cấp 0:*** Tệp các thanh ghi (registers) bên trong bộ vi xử lý là mức nhớ thấp nhất.

– ***Cấp 1:*** Cache sơ cấp L1 (primary cache) là bộ nhớ có tốc độ truy cập nhanh gần bằng tốc độ truyền dữ liệu trong bộ vi xử lý nhưng dung lượng nhỏ và được tích hợp ngay trên chip đối với những bộ vi xử lý hiện đại.

– ***Cấp 2:*** Cache thứ cấp L2 (secondary cache) là bộ nhớ truy cập nhanh nhưng dung lượng nhỏ hơn bộ nhớ chính và thường nằm bên ngoài chip vi xử lý.

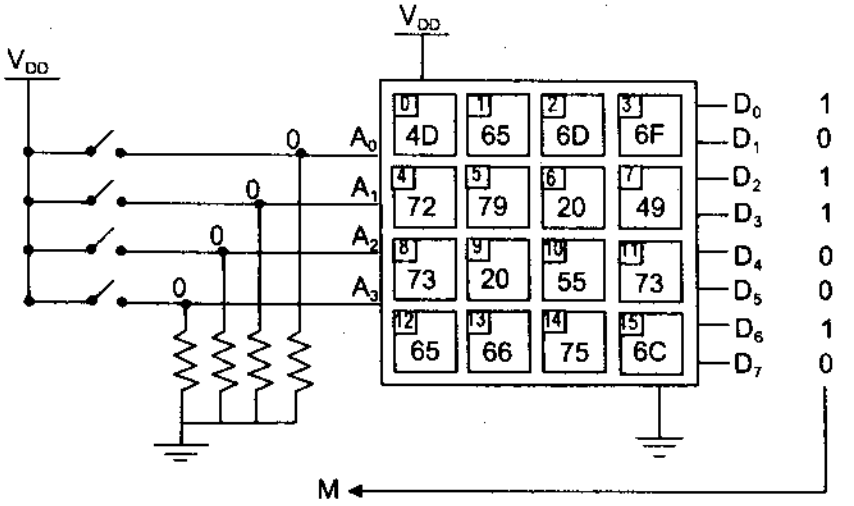
– ***Cấp 3:*** Bộ nhớ chính (main memory), được bộ vi xử lý đánh địa chỉ trực tiếp, chứa dữ liệu và các chương trình đang được sử dụng hiện hành, đây chính là ROM và RAM.

– ***Cấp 4:*** Bộ nhớ ngoài (external memory), không được bộ vi xử lý đánh địa chỉ trực tiếp, bộ nhớ ngoài có dung lượng rất lớn, lớn hơn rất nhiều lần so với bộ nhớ chính nhưng tốc độ truy cập lại chậm hơn: Có thể kể ra một số thiết bị thông dụng thuộc về bộ nhớ ngoài như: ổ đĩa cứng (Hard Disk Drive–HDD), ổ đĩa mềm (Floppy Disk Drive–FDD). Đĩa CDROM (Compact Disk Read Only Memory), đĩa DVD (Digital Versatile Disk), Flash Disk, Memory Card...

– ***Cấp 5:*** Bộ nhớ mạng là bộ nhớ mà một máy tính có thể truy cập tới bộ nhớ của một máy khác trong mạng máy tính.

4.2. BỘ NHỚ TRONG

Bộ nhớ trong được cấu tạo từ các vi mạch nhớ bán dẫn (ROM, RAM). Mỗi một vi mạch nhớ lại được chia thành các ô nhớ nhỏ có địa chỉ riêng, xác định gọi là *địa chỉ ô nhớ*. Trong máy tính, mỗi một địa chỉ nhớ chứa một byte chứa dữ liệu, hoặc có thể là word (2 byte), hoặc doubleword (4 byte). Hình 4.2 là ví dụ về bộ nhớ ROM (16×8bit). Có 16 ô nhớ được phân biệt bởi bốn tín hiệu địa chỉ A_0, A_1, A_2, A_3. Vì các dữ liệu trong máy tính được biểu diễn dưới dạng số nhị phân nên địa chỉ của bộ nhớ cũng được biểu diễn dưới dạng số nhị phân. Mỗi ô nhớ có 8 bit (D_0, D_1, D_2,... D_7) chứa một byte dữ liệu có giá trị từ 00000000B đến 11111111B. Để lưu được một bit thông tin có thể dùng các phần tử nhớ như diode, tụ điện, transistor, các flip–flop,...

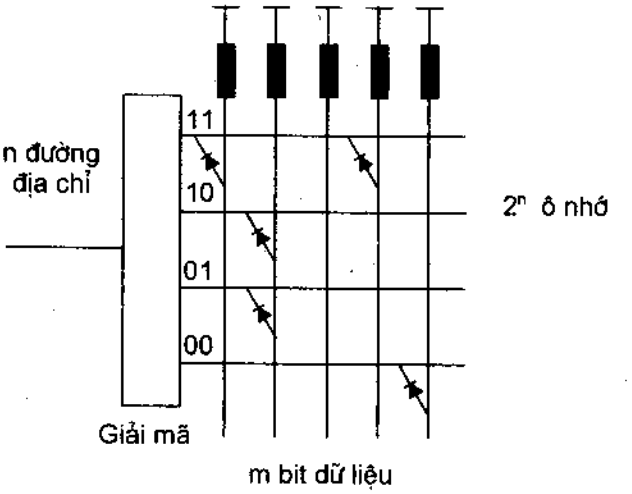


Hình 4.2. ROM (16×8bit)

4.2.1. Bộ nhớ ROM (Read Only Memory)

ROM là bộ nhớ chỉ đọc, đây là bộ nhớ rất quan trọng đối với bất kỳ một hệ thống vi xử lý nào. Dữ liệu lưu trong ROM không bị mất khi mất nguồn điện. Do đặc trưng này nên ROM thường được dùng chứa các chương trình quản lý, điều khiển phần cứng của hệ thống (thường được gọi là các chương trình hệ thống), ví dụ như trong hệ thống máy tính, ROM chứa BIOS, trong các thiết bị điện tử gia dụng, ROM chứa chương trình điều khiển sự hoạt động của các thiết bị đó...

4.2.1.1. Nguyên tắc cấu tạo và hoạt động



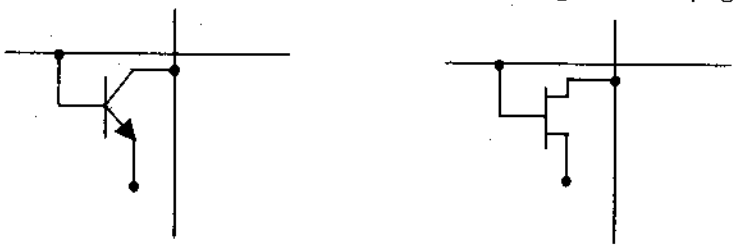
A_1A_0	$D_4D_3D_2D_1D_0$
11	01101
10	10111
01	10111
00	11110

Hình 4.3. ROM dùng ma trận diode

– Nguyên tắc cấu tạo:

ROM có thể được chế tạo bằng công nghệ lưỡng cực hoặc MOS, ROM lưỡng cực có thời gian truy cập nhanh hơn, tính năng kích hoạt tốt hơn, ROM công nghệ MOS có thành phần chủ yếu là các transistor trường.

Trên hình 4.3 mô tả cấu tạo của một bộ nhớ ROM đơn giản sử dụng diode cho các bit nhớ, mỗi một bit nhớ có diode mang giá trị logic 0, bit nhớ không có diode mang giá trị logic 1.



Hình 4.4. ROM dùng ma trận transistor

Các diode cũng

có thể được thay bằng các transistor, khi đó sẽ có các vi mạch *ROM dùng ma trận transistor* (hình 4.4).

– Nguyên tắc hoạt động:

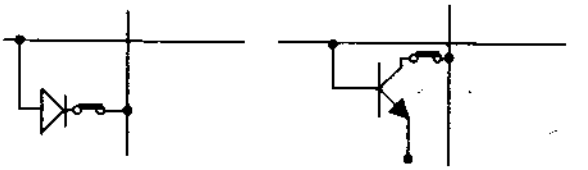
Khi hàng nào nối với 5V (được chọn) thì diode giao giữa hàng đó và cột sẽ thông và thông tin đọc được trên cột đó là 1, cột nào không có diode nối với hàng đó thì sẽ mang thông tin 0.

Hiện nay ROM có nguyên tắc cấu tạo và hoạt động như trên ít được nhắc tới, thay vào đó là các loại PROM, EPROM và EEPROM có dung lượng cao, thuận tiện trong việc lập trình hơn nhiều. Chính vì vậy, khái niệm "ROM là bộ nhớ chỉ đọc" không còn thích hợp nữa, thay vào đó là khái niệm "ROM là bộ nhớ không bị mất thông tin khi mất nguồn điện".

4.2.1.2. Phân loại

a) Bộ nhớ PROM (Programmable ROM)

Là một loại bộ nhớ ROM có thể lập trình được một lần. Khi xuất xưởng, PROM chưa lưu thông tin, khi cần lưu thông tin, PROM sẽ được "lập trình". Công việc lập trình sẽ tạo ra các xung dòng điện đủ lớn qua một số bit nhớ chọn lọc, xung điện này sẽ làm đứt các cầu chì (hình 4.5), nhờ việc làm đứt (hoặc không làm đứt) các các cầu chì, các bit trong PROM sẽ có giá trị bằng "0" (hoặc bằng "1"). Khi đã lập trình, nội dung của PROM không thể thay đổi được.



Hình 4.5. PROM

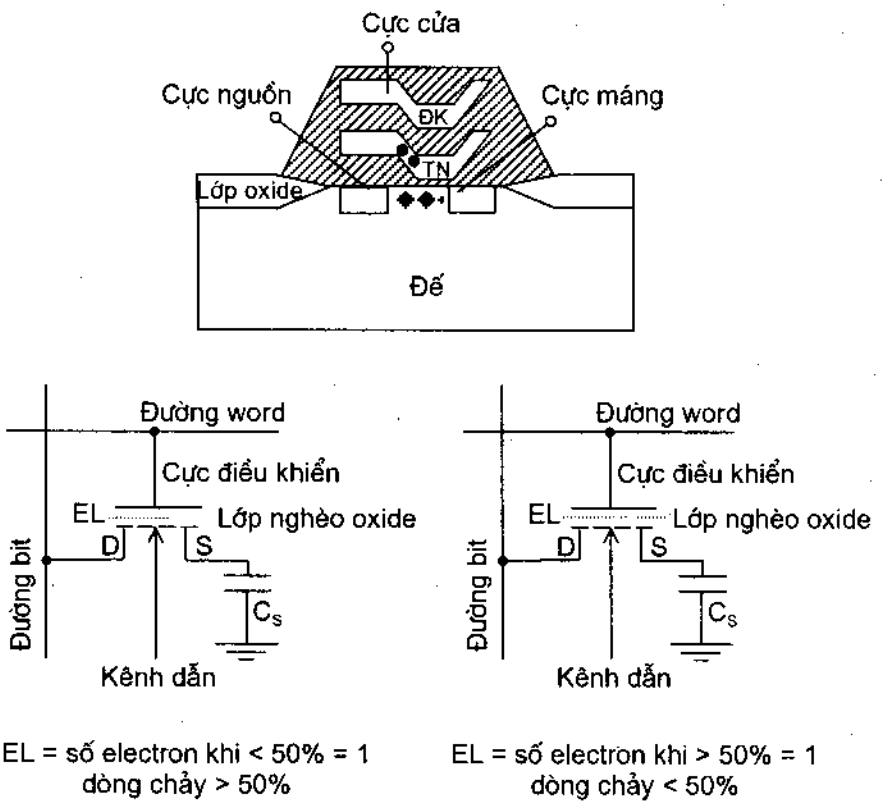
b) EPROM (Erasable Programming ROM)

EPROM là loại ROM được xoá bằng nguồn tia cực tím (UV–Ultraviolet light) chiếu trên các transistor và được lập trình bằng xung điện thế cao. Các giá trị logic được ghi ngược với giá trị xoá.

Nguyên tắc cấu tạo và hoạt động (hình 4.6).

Cực cửa transistor trường được đặt trong lớp cách điện. Khi đặt điện áp –35V giữa cực nguồn và cực máng thì cực cửa sẽ có điện tích cảm ứng. Điện tích này sẽ lưu mãi ở đó vì không có đường thoát. Điện tích này đến lượt nó lại tạo ra kênh dẫn trong MOS khi bỏ nguồn –35V đi. Việc đặt nguồn –35V vào transistor MOS là việc nạp cho EPROM. EPROM bằng cách chiếu tia

cực tím vào transistor thì điện tích trên cực cửa sẽ bị tiêu tan đi do tia cực tím làm cho lớp oxide cách điện trở thành dẫn điện.



Hình 4.6. EPROM

Nhiều hãng sản xuất ROM đã tìm cách tạo ra ROM ghi xoá được bằng điện áp. Các ROM này được gọi là EEPROM. Về nguyên tắc chung thì nó giống như EPROM nhưng lớp cách điện giữa cực cửa và kênh dẫn được chia thành hai lớp khác nhau và thêm cực cửa điều khiển. Khi đặt điện áp đủ lớn vào cực cửa điều khiển thì làm cho các điện tử có thể đi qua được lớp cách điện bằng hiệu ứng đường hầm, như vậy ta đã xoá được điện tích ở trên cực cửa hay xoá được thông tin nhớ trong ROM.

c) Bộ nhớ EEPROM (Electrically EPROM)

Là bộ nhớ có thể ghi, xoá bằng xung điện. Số lần có thể ghi, xoá cho một vi mạch nhiều hơn nhiều so với các bộ nhớ ROM khác (vài chục lần đến vài chục nghìn lần). Việc nạp cho các vi mạch nhớ này nhờ các mạch nạp, các

mạch nạp sẽ đưa các địa chỉ của các ô nhớ lên BUS địa chỉ đồng thời đưa dữ liệu và xung tích cực thích hợp vào chân cho phép nạp, khi đó dữ liệu sẽ được nạp vào EEPROM.

Về nguyên tắc chung, EEPROM giống như EPROM nhưng lớp cách điện giữa cực cửa và kênh dẫn được chia thành hai lớp khác nhau và thêm cực cửa điều khiển. Khi đặt điện áp đủ lớn vào cực cửa điều khiển thì các hạt điện tử có thể đi qua được lớp cách điện bằng hiệu ứng đường hầm, điều này đồng nghĩa với việc đã xoá được điện tích ở trên cực cửa hay xoá được thông tin nhớ trong EEPROM nhờ xung điện thay vì tia cực tím.

d) Bộ nhớ flash ROM

Hiện nay bộ nhớ *flash ROM* xuất hiện rất nhiều thay thế cho đĩa mềm, đĩa cứng. Flash hoạt động như bộ nhớ RAM nhưng lại không mất dữ liệu khi mất nguồn điện. Bộ nhớ này cho phép hệ thống thay đổi được nội dung sau khi thiết bị được sản xuất.

Bộ nhớ EEPROM và flash ROM thường được ứng dụng rộng trong các lĩnh vực đo lường, điều khiển dùng vi xử lý.

4.2.2. RAM (Random Access Memory)

RAM là bộ nhớ truy cập ngẫu nhiên, thông tin lưu trong RAM sẽ bị mất khi mất nguồn điện.

Trong máy tính, bao giờ cũng chỉ có một không gian bộ nhớ được điều chỉnh và không gian đó được gọi là bộ nhớ chính của vi xử lý. Bộ nhớ chính của máy tính bao gồm bộ nhớ ROM và bộ nhớ RAM. Bộ nhớ ROM có dung lượng nhỏ hơn rất nhiều so với RAM vì ROM chỉ chứa các chương trình vào/ra cơ sở *BIOS* như chương trình khởi động máy tính, chương trình phục vụ cho việc xâm nhập ổ đĩa card đồ hoạ... Mọi chương trình và dữ liệu của người sử dụng được lưu trữ ở ổ đĩa ngoài, khi máy tính hoạt động nó sẽ được đưa vào bộ nhớ RAM để thực thi, RAM đóng vai trò là nơi chứa thông tin tạm thời cho CPU khi CPU hoạt động. Các lệnh của chương trình được đưa vào bộ nhớ bằng quá trình ghi bộ nhớ, khi chạy chương trình thì bộ vi xử lý sẽ đọc những lệnh đó ra. RAM là một hệ thống bộ nhớ mà trong đó các vùng nhớ đều có thể truy cập đến dễ dàng như nhau, vi xử lý dùng cách truy cập trực tiếp vào RAM.

4.2.2.1. Cấu tạo và nguyên tắc hoạt động

a) RAM tĩnh (SRAM – Static RAM)

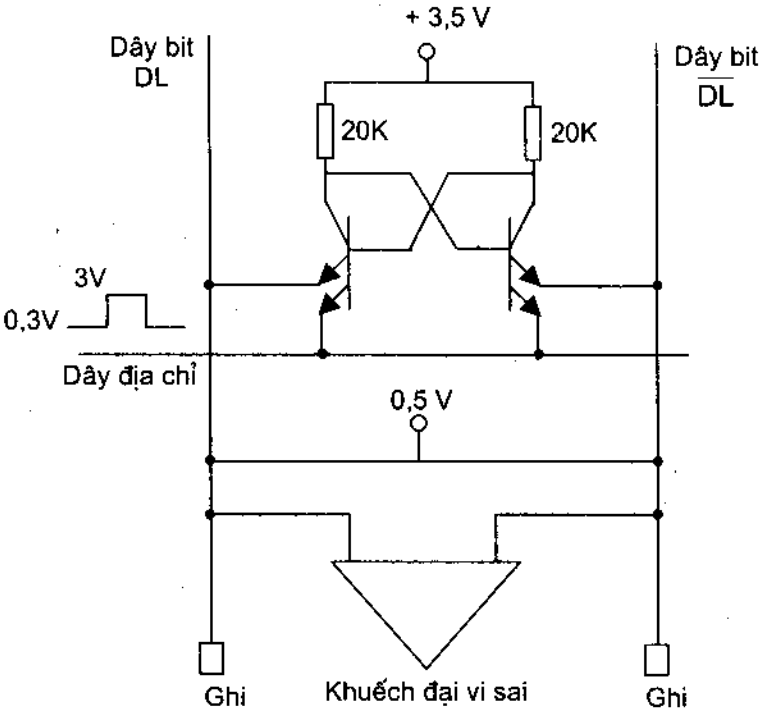
SRAM có đặc điểm là tốc độ nhanh, dung lượng nhỏ, giá thành đắt, thích hợp dùng để chế tạo bộ nhớ cache nhằm cải thiện tốc độ của hệ thống. Bộ nhớ tĩnh được chế tạo bằng công nghệ lưỡng cực hoặc công nghệ MOS với thời gian truy cập cỡ vài chục ns.

Hình 4.7 là cấu tạo của 1 ô nhớ SRAM, mỗi bit nhớ của SRAM là một mạch lật hai trạng thái ổn định với các transistor MOS.

Nguyên tắc hoạt động:

Dây địa chỉ là đường tín hiệu cho phép chọn địa chỉ, nó đưa tín hiệu địa chỉ đến phần tử để có thể đọc hoặc ghi thông tin đối với phần tử đó. Hai dây DL và $\overline{DL}$ là hai dây cho phép xác định trạng thái của phần tử để có thể đọc hoặc ghi thông tin đối với phần tử.

– Ở trạng thái nhớ: Trên đường dây địa chỉ có điện áp 0,3V thấp hơn 0,5V trên hai dây bit làm cho hai emitter nối với hai dây bit không dẫn. Trạng thái của mạch lật được duy trì bởi hai emitter nối với địa chỉ. Lúc này trong mạch sẽ luôn ở trạng thái cố định có một transistor thông và một transistor tắt.



Hình 4.7. Cấu tạo của một ô nhớ SRAM

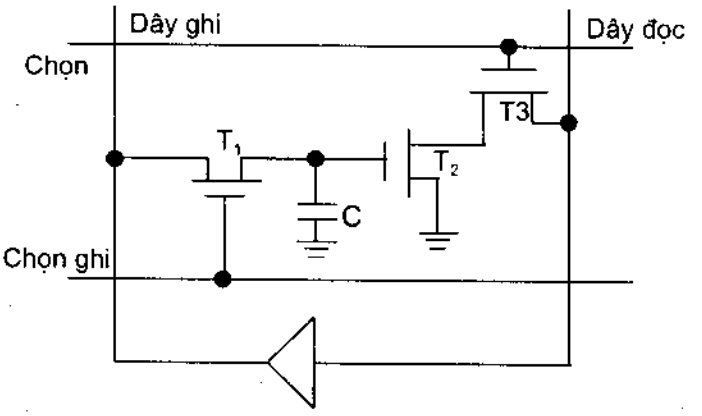
– Khi cần ghi hoặc đọc phần tử ta cần thực hiện chọn phần tử bằng cách nâng điện áp trên dây địa chỉ lên 3V làm cho hai emitter nối với dây địa chỉ sẽ không dẫn nữa. Lúc này trên hai emitter nối với hai dây bit sẽ có một dây có dòng emitter nếu transistor nối với nó thông làm điện thế của nó cao hơn dây kia. Việc so sánh điện áp trên hai dây bit sẽ cho ta biết transistor nào đang thông hay biết được trạng thái của phần tử. Đây chính là nguyên tắc đọc bộ nhớ.

– Để ghi thông tin vào phần tử thì cùng với việc nâng cao điện áp trên đường dây địa chỉ nhằm liên kết hai emiter nối với nó, ta nhận thấy rằng nếu đặt một điện thế đủ lớn vào hai emitter nối với dây bit của transistor đang thông thì transistor này tắt làm cho phần tử nằm ở trạng thái mà ta muốn, tức là đã ghi được thông tin.

b) RAM động (DRAM – Dynamic RAM)

Mỗi một phần tử nhớ của DRAM là các tụ điện. DRAM dùng cách lưu trữ thông tin bằng cách nạp hoặc không nạp điện tích lên các tụ điện, vì thế sau một khoảng thời gian, bộ nhớ cần phải được làm tươi bằng cách ghi đọc, nếu không điện tích trên các tụ bị phóng và dẫn đến mất thông tin. Vì lý do này mà các RAM động phải có các mạch phụ thực hiện công việc "làm tươi".

Ưu điểm của loại RAM này là có dung lượng lớn (có thể tích hợp được số lượng lớn các phần tử trên một đơn vị diện tích) nhưng tốc độ chậm và thường được sử dụng làm bộ nhớ chính.



Hình 4.8. Cấu tạo của một ô nhớ DRAM

Nguyên tắc hoạt động:

– Ở trạng thái nhớ dữ liệu: Tín hiệu chọn T_1 và T_3 không có, tụ C được tích điện.

– Ghi thông tin: Khi truyền điện đến dây ghi tín hiệu cần ghi và tín hiệu chọn T_1 thì tụ C được nạp qua T_1 đang thông. Như vậy, ta đã ghi được thông tin vào phần tử.

– Đọc thông tin: Đưa tín hiệu chọn T_3, do đó T_3 thông, nên trên đường dây đọc có điện thế tương đương điện thế của tụ C.

– Làm tươi bộ nhớ: Nếu có tín hiệu chọn đồng thời đến T_1 và T_3 thì thông tin đọc được sau khi được khuếch đại đưa vào để ghi lại cho tụ C.

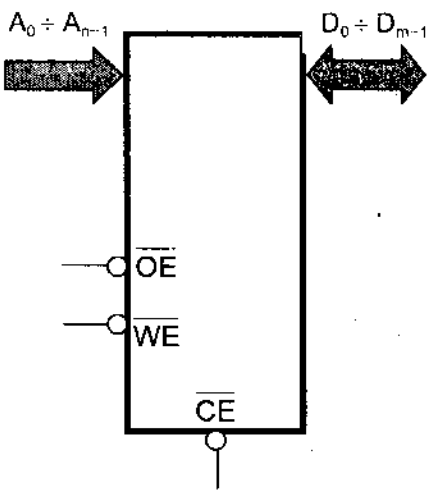
4.2.2.2. Phân loại RAM

Về cơ bản, RAM được phân thành hai loại chính như đã trình bày ở trên. Tuy nhiên hiện nay, do tốc độ hoạt động của các bộ vi xử lý ngày càng cao nên RAM luôn được nghiên cứu, cải thiện không ngừng cả về phần cứng cũng như phần mềm. Các chip nhớ DRAM thường được lắp ráp thành các thanh RAM cắm ngay trên các khe cắm trên bản mạch chính gần bộ vi xử lý. DRAM được chia thành một số loại: SDRAM (Synchronus DRAM), SDR SDRAM (Single Data Rate SDRAM), DDR SDRAM (Double Data Rate SDRAM), DRDRAM (Direct Rambus DRAM).

4.2.3. Ghép nối 8086 với các vi mạch ROM27xxx và RAM62xxx

4.2.3.1. Cấu trúc chung của một vi mạch nhớ

Một chip nhớ được chế tạo bao gồm: Một ma trận nhớ, các mạch logic giải mã địa chỉ cho ô nhớ, các mạch logic cho phép đọc ra hay ghi vào ô nhớ, các mạch vào/ra, mở rộng địa chỉ.



Hình 4.9. Sơ đồ cấu trúc chung của mạch nhớ RAM

Dung lượng vi mạch nhớ là số bit thông tin tối đa mà bộ nhớ có thể lưu trữ nó. Đối với bộ nhớ trong máy tính, đơn vị nhớ được tính theo byte còn với chip nhớ, đơn vị nhớ được tính theo bit. Một vi mạch nhớ thường được biểu diễn dưới dạng sơ đồ cấu trúc như hình 4.9, bao gồm có các nhóm tín hiệu sau:

– *Nhóm tín hiệu địa chỉ:* Bao gồm n đường địa chỉ, số lượng đường địa chỉ có liên quan tới dung lượng của bộ nhớ theo công thức sau:

Dung lượng (byte) = 2^n .

Các ô nhớ kề nhau sẽ có địa chỉ kế tiếp nhau. Nếu có n bit để biểu diễn cho một địa chỉ thì số địa chỉ tối đa có thể có là 2^n, từ địa chỉ 0 đến $2^n - 1$. Tín hiệu địa chỉ được biểu diễn dưới dạng nhị phân là $A_{n-1} A_{n-2} \ldots A_2 A_1 A_0$.

– *Nhóm tín hiệu dữ liệu:* Gồm m đường dữ liệu, số lượng đường dữ liệu có liên quan tới độ dài của ô nhớ (chính bằng số bit trong một ô nhớ được biểu diễn dưới dạng nhị phân là $D_{m-1} D_{m-2} \ldots D_2 D_1 D_0$). Ví dụ, chip nhớ 27128 là vi mạch nhớ ROM có dung lượng 128Kbit hay bằng 32K×8 bit (32K ô nhớ, mỗi ô nhớ 8 bit), bộ nhớ ROM này có số đường địa chỉ n = 15, số đường dữ liệu là m = 8.

– *Nhóm tín hiệu điều khiển:* Gồm có tín hiệu cho phép đọc $\overline{OE}$ (Output Enable) để cho phép dữ liệu được đưa ra, đối với vi mạch nhớ RAM còn có tín hiệu cho phép ghi $\overline{WE}$ (Write Enable). Ngoài ra, còn có thể có các tín hiệu chốt địa chỉ hàng $\overline{RAS}$ (Row Access Stroble), tín hiệu chốt địa chỉ cột $\overline{CAS}$ (Column Access Stroble) đối với RAM động. Các chân tín hiệu này đều được tích cực ở mức thấp.

– *Nhóm tín hiệu chọn vi mạch:* Gồm có tín hiệu chọn chip $\overline{CS}$ (Chip Select) hay cho phép vi mạch $\overline{CE}$ (Chip Enable). Các tín hiệu chọn vi mạch thường được nối với đầu ra của mạch giải mã địa chỉ. Khi một vi mạch nhớ không được chọn thì bus dữ liệu của nó bị treo.

4.2.3.2. Giải mã địa chỉ cho bộ nhớ

Mỗi một ô nhớ của vi mạch nhớ khi ghép nối với CPU đều được gán một địa chỉ xác định bằng mạch giải mã bên trong vi mạch nhớ, với mỗi một vi mạch nhớ phải được gán bằng một vùng địa chỉ riêng, xác định nằm trong không gian tổng thể của bộ nhớ. Việc gán địa chỉ cho vi mạch nhớ được thực hiện nhờ một xung lấy từ *mạch giải mã địa chỉ.*

Mạch giải mã địa chỉ luôn phải đảm bảo đồng thời cả hai điều kiện:

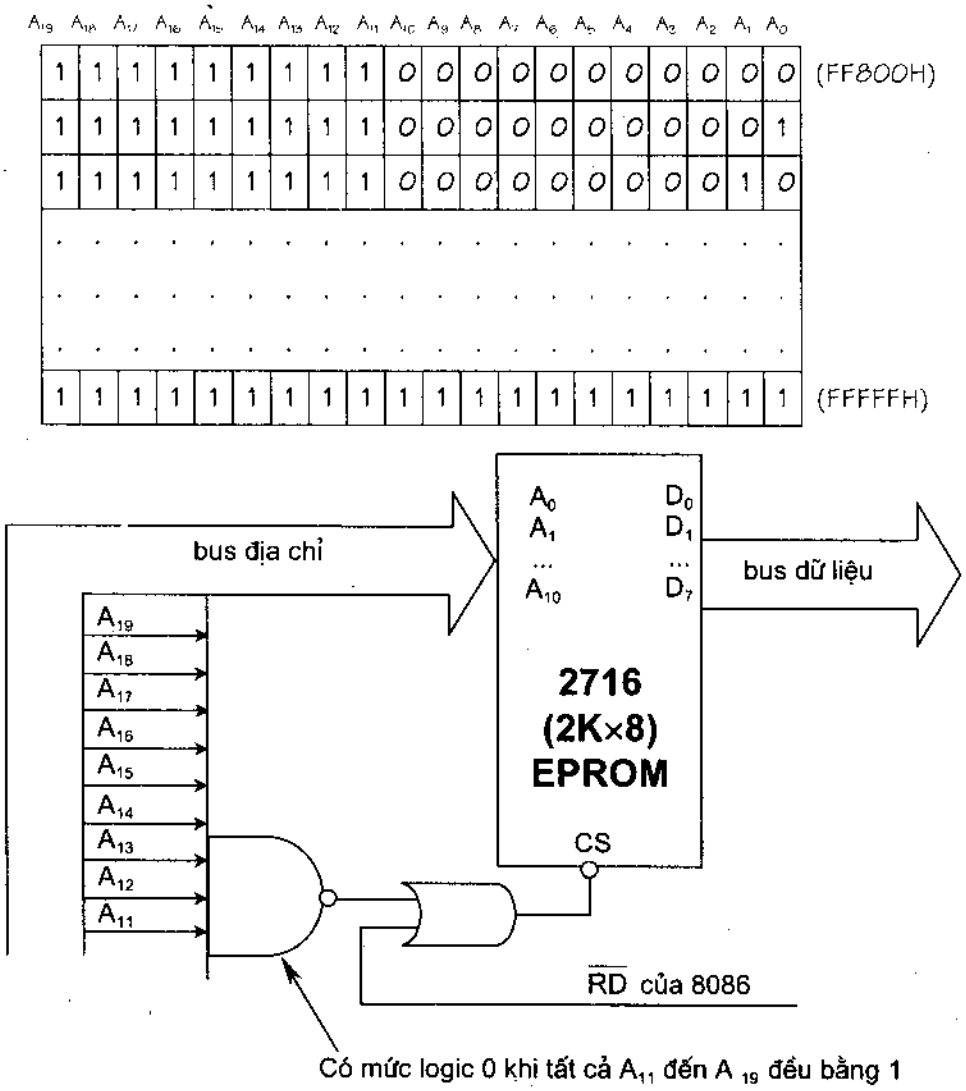
– Nếu vi xử lý đưa ra địa chỉ của một ô nhớ nằm trong vi mạch nhớ nào đó đã được giải mã thì phải truy cập đọc/ghi được ô nhớ đó, tức là chân tín hiệu $\overline{CE}$ của vi mạch nhớ đó phải được tích cực.

– Nếu vi xử lý đưa ra địa chỉ của một ô nhớ nằm ngoài vi mạch nhớ nào đó đã được giải mã thì sẽ không truy cập được tới ô nhớ đó, chân tín hiệu $\overline{CE}$ của vi mạch nhớ đó không được tích cực... (điều kiện này để tránh xung đột bộ nhớ).

a) Giải mã địa chỉ dùng các cổng logic

Ví dụ 1: Xây dựng mạch giải mã địa chỉ cho IC 2716 có địa chỉ đầu là FF800H.

Bộ vi xử lý 8086 sử dụng 20 bit địa chỉ nên có thể quản lý tối đa bộ nhớ có dung lượng 1Mbyte. Tuy nhiên, IC 2716 là vi mạch nhớ ROM có dung lượng là 16Kbit = 2KB×8 bit (11 đường địa chỉ và 20 đường dữ liệu) nên ta cần 11 đường địa chỉ để quản lý từng ô nhớ (A_0 – A_{10}). Số còn lại (9 đường địa chỉ A_{14} - A_{19}) tổ hợp để đưa vào chân cho phép vi mạch $\overline{CE}$.



Hình 4.10. Sơ đồ mạch giải mã ví dụ 1

Vùng địa chỉ yêu cầu cần giải mã nằm trong khoảng địa chỉ từ FF800H đến FFFFFH.

Nhìn vào các địa chỉ của vi mạch nhớ ta thấy các bit địa chỉ từ A_0 – A_{10} thay đổi theo từng ô nhớ còn các bit A_{11} – A_{19} cố định bằng 1, các bit này sẽ được tổ hợp để tạo ra tín hiệu có mức logic thấp đưa tới chân $\overline{CE}$. Mạch giải mã được thiết kế như hình 4.10.

Ví dụ 2: Thiết kế mạch ghép nối giữa 8086 và các IC nhớ SRAM 4KB×8 bit để thành bộ nhớ 4 KB×16 bit. Địa chỉ đầu của bộ nhớ là: 7C000H.

Giải:

– Xác định số IC cần mắc song song:

$$m = \frac{\text{Độ rộng dữ liệu mong muốn}}{\text{Độ rộng dữ liệu của 1 IC}} = \frac{16}{8} = 2.$$

Như vậy cần 2 IC mắc song song.

– Xác định dải địa chỉ bộ nhớ:

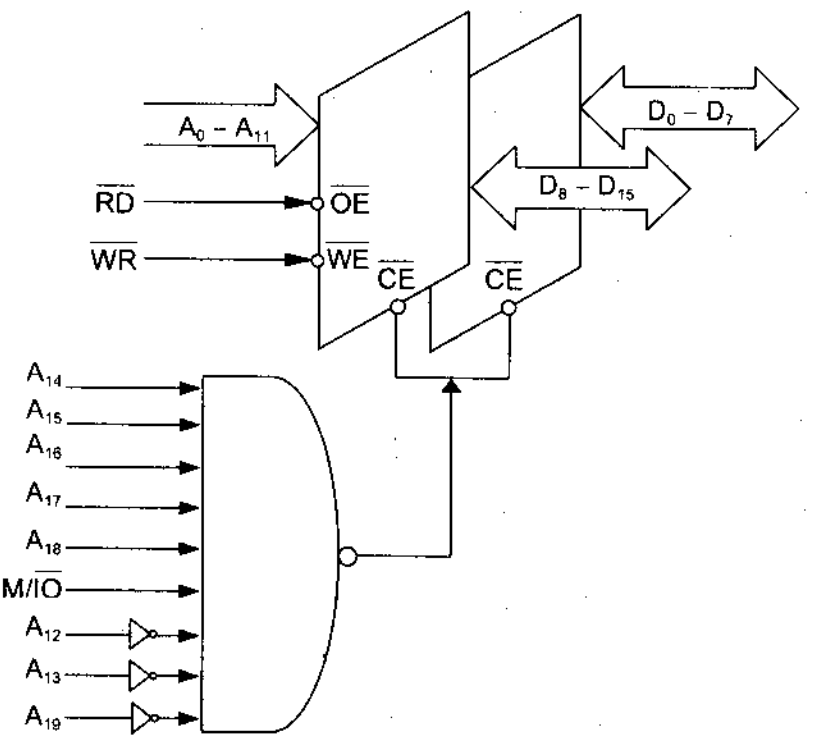
Địa chỉ bắt đầu: 7C000H đến 7CFFFH.

+ Địa chỉ bắt đầu ở dạng nhị phân là: 0111 1100 0000 0000 0000

+ Địa chỉ kết thúc ở dạng nhị phân là: 0111 1100 1111 1111 1111

– Xác định các tín hiệu địa chỉ đưa trực tiếp vào các IC nhớ:

+ Phần thay đổi trong dải địa chỉ trên sẽ đưa trực tiếp vào các IC nhớ SRAM (A_0 đến A_{11}).



Hình 4.11. Sơ đồ mạch giải mã ví dụ 2

+ Phần địa chỉ

thay đổi sẽ đưa vào mạch giải mã để tạo ra tín hiệu chọn các vi mạch SRAM (A_{12} đến A_{19}).

– Xây dựng mạch giải mã dùng mạch NAND.

+ Tín hiệu đưa trực tiếp vào mạch NAND là: A_{14} đến A_{18}, M/$\overline{IO}$.

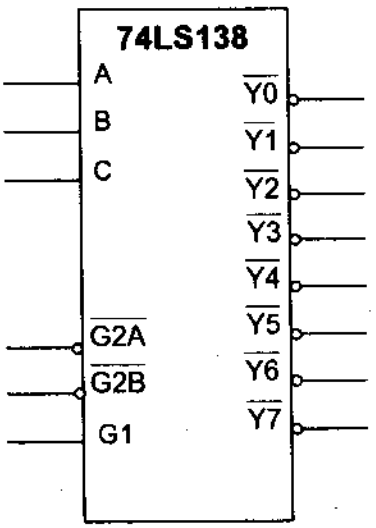
+ Tín hiệu đưa vào mạch NAND thông qua mạch đảo là: A_{12}, A_{13}, A_{19}.

+ Vẽ sơ đồ mạch như hình 4.11.

b) Giải mã địa chỉ dùng các vi mạch giải mã 74LS139, 74LS138

Mạch giải mã địa chỉ dùng các cổng logic sẽ trở nên cồng kềnh khi cần giải mã cho nhiều vi mạch nhớ. Vì vậy người ta ít dùng cổng logic mà thường dùng các vi mạch giải mã có sẵn.

Đầu vào						Đầu ra							
/G2A	/G2B	G1	C	B	A	/Y0	/Y1	/Y2	/Y3	/Y4	/Y5	/Y6	/Y7
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0



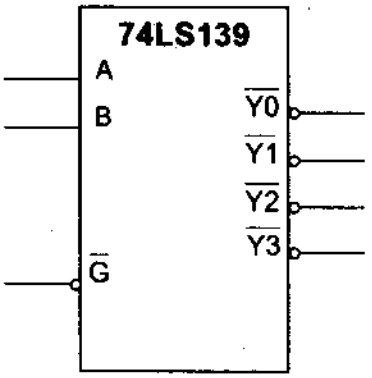
Hình 4.12. Vi mạch giải mã 74LS138

Ví dụ 3: Xây dựng mạch giải mã địa chỉ cho 4 vi mạch nhớ RAM(4KB×8bit) kề nhau, địa chỉ đầu của vi mạch thứ nhất là E0000H.

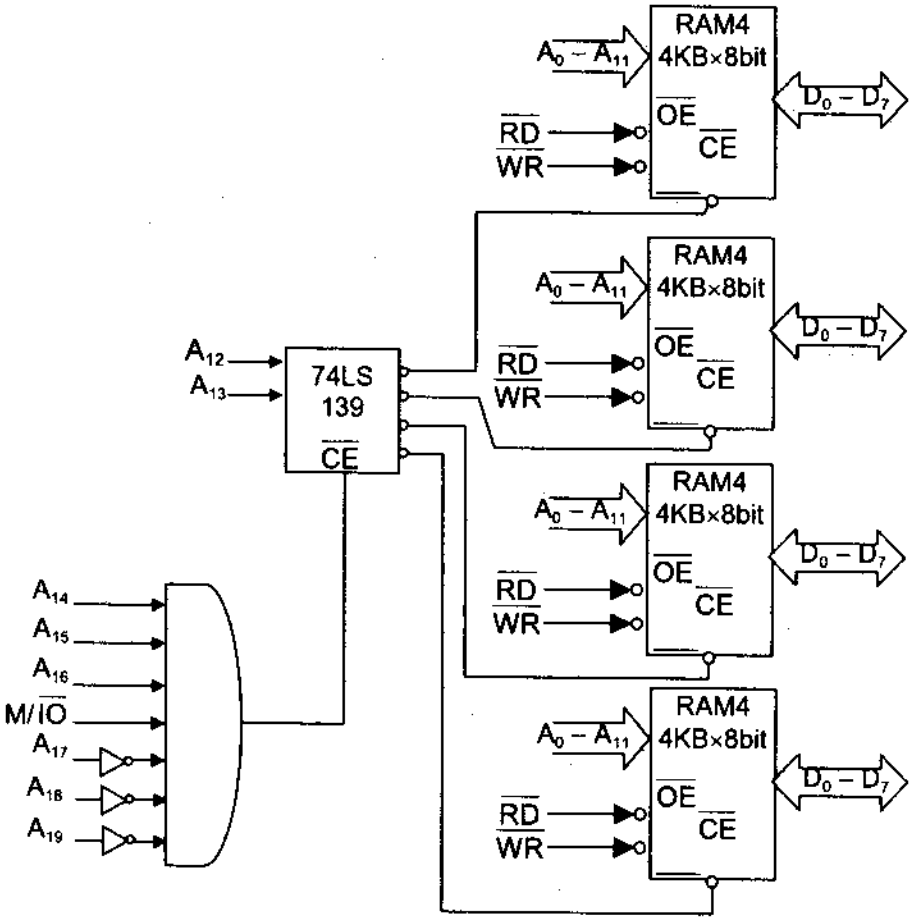
Trong ví dụ này chúng ta có 4 vi mạch nhớ, số lượng này trùng khớp với số lượng đầu ra mà vi mạch 74LS139 có, vì vậy chúng ta sử dụng vi mạch này.

Cách tính toán địa chỉ hoàn toàn tương tự như trong ví dụ 1, hình 4.14 là mạch giải mã hoàn chỉnh:

/G	A	B	/Y0	/Y1	/Y2	/Y3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



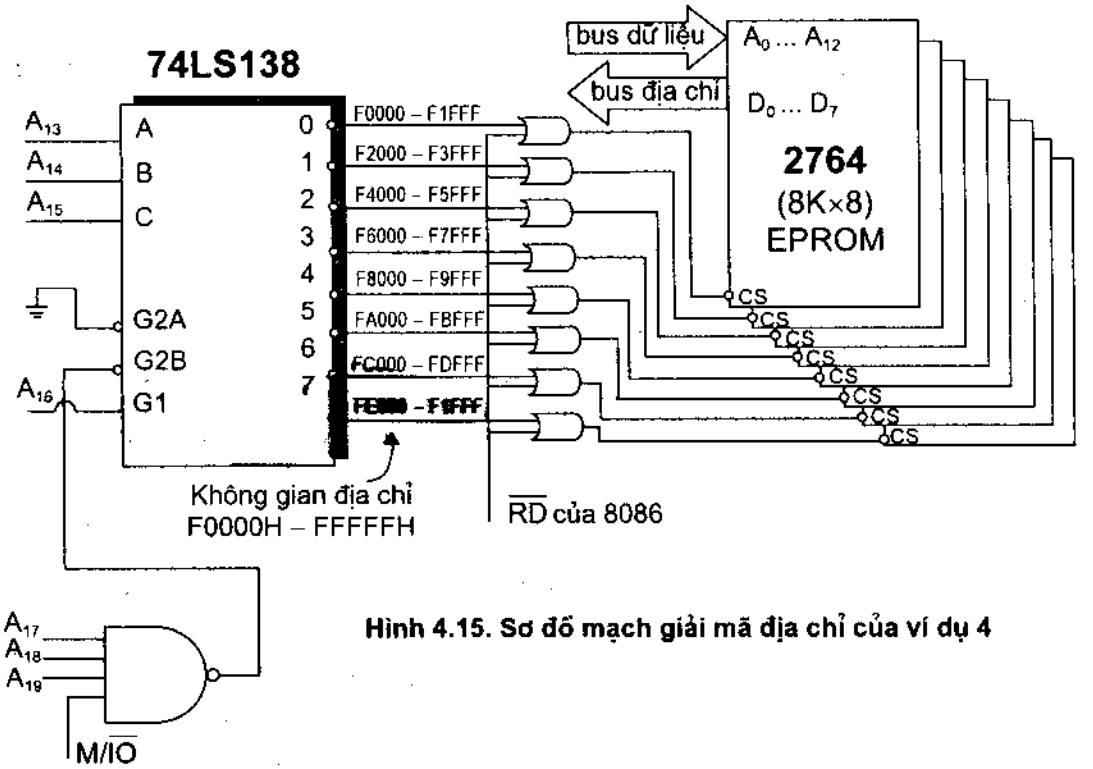
Hình 4.12. Vi mạch giải mã 74LS139



Hình 4.14. Sơ đồ mạch giải mã địa chỉ của ví dụ 3

Ví dụ 4: Xây dựng mạch giải mã địa chỉ cho 8 vi mạch nhớ RAM (16KB×8bit) kề nhau, địa chỉ đầu của vi mạch thứ nhất là F0000H.

Từ A_{13} đến A_{15} chọn một IC2764. Từ A_{16} đến A_{19} chọn mạch giải mã.



Hình 4.15. Sơ đồ mạch giải mã địa chỉ của ví dụ 4

4.3. HỆ THỐNG LƯU TRỮ NGOÀI

Các hệ thống lưu trữ nói chung có thể hiểu là các thiết bị lưu trữ ngoài, chúng là cấp thấp nhất trong phân cấp của hệ thống nhớ máy tính.

4.3.1. Một số phần tử lưu trữ điển hình

Dựa theo theo tính chất vật lý của việc ghi/đọc thông tin người ta phân chia các thiết bị lưu trữ thành 3 nhóm chính:

+ Thiết bị nhớ từ tính.

+ Thiết bị nhớ quang.

+ Thiết bị nhớ flash.

4.3.1.1. Đĩa cứng (Hard Disk)

Cấu tạo của một ổ cứng gồm các khối sau:

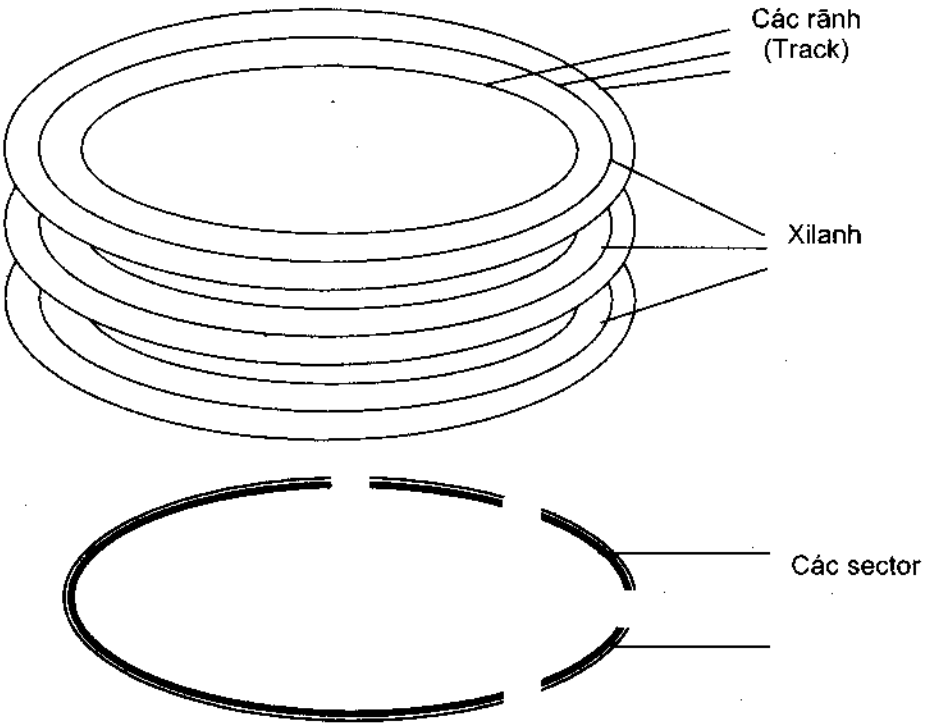
– Khối đĩa và hộp đĩa.

– Khối đầu từ trong định vị đầu từ.

- Khối điện tử điều khiển.

a) Khối đĩa và hộp đĩa

Đĩa cứng thường chứa nhiều đĩa đặt chồng lên nhau trên một trục tạo thành chồng đĩa. Các đĩa cách nhau một khoảng cách vài mm để đầu từ chuyển động. Như vậy, có rất nhiều đầu từ, mỗi đầu từ đọc/ghi cho một mặt (trừ hai mặt ngoài của đĩa dưới cùng và đĩa trên cùng). Trên mặt mỗi đĩa người ta chia thành các rãnh là các vòng tròn đồng tâm (track). Thông thường có từ 5000 đến 30000 track/1 mặt, trên các rãnh này lại được chia thành các cung gọi là các sector (có từ 100 đến 500 sector/1 track), sector là đơn vị nhỏ nhất của ổ đĩa cứng, mỗi sector chứa 512 byte dữ liệu. Các rãnh có cùng chỉ số trên các đĩa gọi là 1 xilanh. Thông thường một ổ đĩa gồm 1 đến 12 đĩa quay với tốc độ 3600 đến 15000 vòng/phút.



Hình 4.16. Cách bố trí đĩa, phân chia đĩa thành các đơn vị nhỏ

b) Khối đầu từ trong định vị đầu từ

Số đầu từ = ((số đĩa) * 2) – 2.

(vì hai đĩa ngoài cùng để bảo vệ đầu từ).

c) Khối điện tử điều khiển: phần điện tử gồm bộ phận điều khiển quay đĩa, điều khiển motor và điều khiển đầu từ ghi/đọc.

Khi đĩa không quay tức là không có ghi/đọc tin, đầu từ tỳ lền mặt đĩa. Khi ghi/đọc thông tin, đầu từ tự nâng lên một khoảng 0,2mm đến 1mm khỏi mặt đĩa do áp lực của không khí chuyển động nhanh trong ổ đĩa. Khi đọc/ghi dữ liệu, đầu từ cần phải được di chuyển tới đúng sector có chứa thông tin cần đọc, để làm được điều nầy, các sector đều được định địa chỉ riêng biệt, thời gian di chuyển của đầu từ đến các sector được gọi là thời gian truy cập của đầu từ, khoảng thời gian này cỡ 5 đến 12ms, tuy nhiên thực tế thị thời gian này chỉ bằng 33% trên lý thuyết. Để tìm ra một sector, đầu từ phải quay 1/2 vòng quanh track, thời gian này là thời gian trễ quay. Khi đầu từ dã di chuyển được đến vị trí cần ghi/đọc thì thông số cần quan tâm nữa là thời gian ghi/đọc. Thông số này phụ thuộc vào kích thước đĩa, tốc độ quay, mật độ của các sector trên track và đáp ứng của mạch điện tử kết nối giữa đĩa và máy tính. Một ổ đĩa với kích thước đĩa là 3,5inchs (≈ 0,889m), quay 15000 vòng/phút thì tốc độ đó là 65Mbyte. Thông số thời gian cuối cùng trong tổng thời gian ghi/đọc dữ liệu từ máy tính lên đĩa là thời gian đáp ứng của mạch điều khiển đầu từ khi nhận lệnh từ máy tính.

Ví dụ, một ổ đĩa máy tính có thời gian truy cập của đầu từ trung bình là 5ms, tốc độ ghi/đọc là 40Mbyte, tốc độ quay là 10000 vòng/phút, đáp ứng của bộ điều khiển đầu từ là 0,1ms thì thời gian ghi/đọc cho 1 sector sẽ là:

5ms * 33% + 0,5 vòng / (10000*60) + 0.5KB / (40*1024) + 0,1ms

Trong tất cả các thiết bị lưu trữ thì ổ đĩa cứng luôn được chú trọng phát triển. Hai vấn đề được quan tâm nhất đó là mật độ thông tin có thể lưu trữ trên một diện tích nhất định và giá thành cho một đơn vị lưu trữ. Từ năm 1988, mật độ thông tin đã tăng lên 29%/năm, đến năm 1996 là 60%/năm và đến 2001 là 100%/năm, con số cụ thể là 60 triệu bit/2,5cm^2.

Ngoài ổ đĩa cứng thông dụng ra, còn có ổ đĩa mềm có cấu tạo và hoạt động tương tự ổ đĩa cứng, băng từ... Tuy nhiên các thiết bị này đều không thể so sánh được với ổ đĩa cứng về dung lượng, tốc độ nên nói tới hệ thống lưu trữ từ tính thông thường người ta chỉ nhắc tới ổ đĩa cứng.

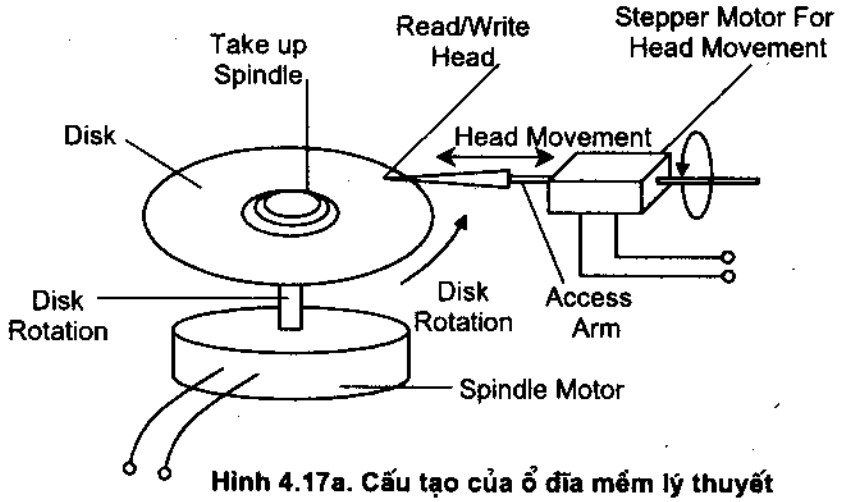
4.3.1.2. Đĩa mềm

Cấu tạo của ổ đĩa mềm (hình 4.17) bao gồm:

– Đầu từ và motor bước điều khiển đầu từ.

– Đĩa từ, motor điều khiển đĩa quay.

– Mạch điều khiển hệ thống motor và hoạt động ghi/đọc thông tin trên đĩa.



Hình 4.17a. Cấu tạo của ổ đĩa mềm lý thuyết

Disk: đĩa từ

Take up Spindle:trục nâng

Motor Shaft: trục motor

Spindle Motor: motor quay

Disk Rotation: hướng quay của đĩa

Access Arm: cần truy xuất

Read/Write Head: đầu từ đọc /ghi

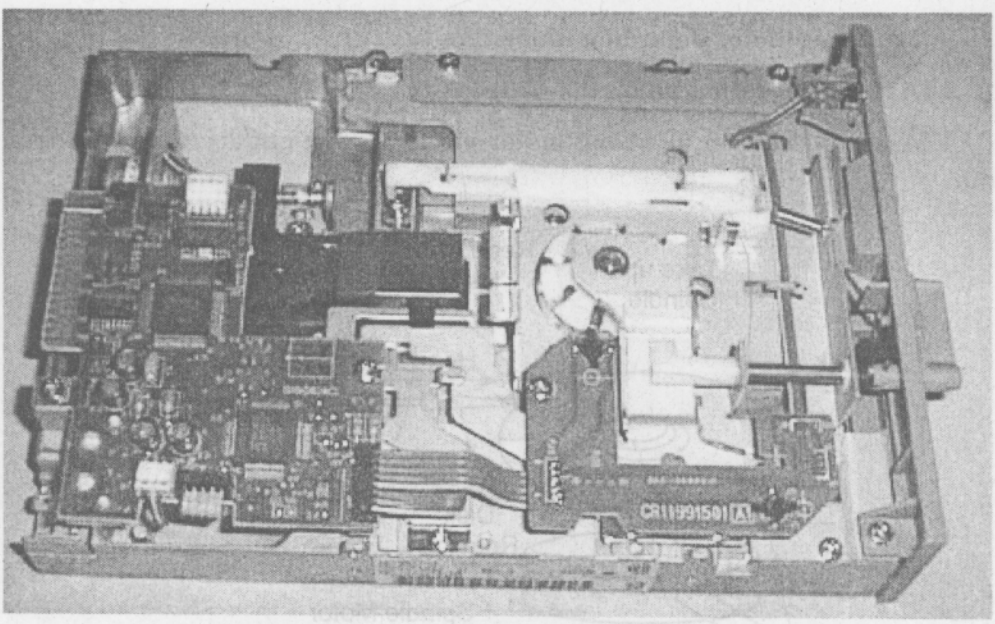
Track: rãnh từ

Head Movement: phương chuyển động của đầu từ

Rotation of stepper Motor: hướng quay của motor bước

Stepper Motor For Head Movement: motor bước điều khiển chuyển động

Khi CPU đọc/ghi dữ liệu, đĩa được quay bởi một motor điều khiển với tốc độ 300 vòng/phút với đĩa có dung lượng 360Kbyte hoặc 360 vòng/phút với các loại đĩa khác. Đĩa mềm có hai mặt nên có hai đầu từ đọc/ghi dữ liệu. Đầu từ được gắn ở đầu cần truy xuất (access arm). Chuyển động quay của một motor bước (stepping motor) sẽ biến thành chuyển động tịnh tiến theo phương bán kính của cần truy xuất qua một cơ cấu bánh răng. Đầu từ có một cuộn dây cảm ứng, khi đọc, sự biến đổi từ thông của phần tử lưu trữ tin được biến thành điện thế cảm ứng ở hai đầu ra của cuộn dây tạo nên tín hiệu dữ liệu (data signal). Khi ghi, cuộn dây sẽ phát ra từ trường qua khe từ để từ hoá các bột sắt từ mạ trên mặt đĩa thành các trạng thái tương ứng với các mức dữ liệu 0 và 1.



Hình 4.17b. Cấu tạo của ổ đĩa mềm thực tế

4.3.1.3. Đĩa quang số

Cấu tạo của ổ đĩa quang số bao gồm:

– Giá quay và motor điều khiển quay đĩa.

– Nguồn laser và các thiết bị để tiêu thụ, thay đổi đường truyền điều chế cường độ sáng và biến đổi quang sang điện.

– Các sơ đồ điện tử điều khiển tốc độ quay, thời điểm phát sáng thu nhận tín hiệu điện, thay đổi từ trường bằng dọc điện.

– Khối điều khiển ghép nối với máy tính.

Tuỳ theo nguyên tắc đọc/ghi quang hay từ quang, ta có hai cấu trúc ghi/đọc khác nhau:

– Cấu trúc ghi/đọc quang.

– Cấu trúc ghi/đọc từ quang.

Chỉ có một nguồn diode laser bán dẫn điều chế cường độ sáng bằng dòng điện gây từ trường, làm quay ánh sáng phân cực, nên cần có kính phân cực để lọc chùm sáng có các cực hỗn loạn thành ánh sáng phân cực.

Nguyên tắc ghi/đọc: Dựa theo đĩa quay hay từ quang ta có hai cách ghi/đọc nhưng đều sử dụng một nguồn sáng laser như nhau.

– Ghi/đọc quang: được tiến hành bởi một chùm tia laser (phát bởi một diode laser). Khi ghi, cường độ ánh sáng mạnh hay yếu đều làm bề mặt đĩa thay đổi trạng thái. Khi đọc, chùm tia laser (cường độ nhỏ hơn lúc ghi 10 lần) sẽ phản xạ khác nhau, ở bề mặt sẽ cho cường độ khác nhau và tuỳ theo trạng thái bề mặt đã bị biến dạng của đĩa lúc ghi.

– Ghi/đọc từ quang: kỹ thuật này dùng cho hai loại đĩa từ quang. Sử dụng phương pháp ghi/đọc chuyển pha và nắn cực vì nó có lợi là cho phép xoá số liệu và tái sử dụng được đĩa.

Các loại đĩa quang số:

– WRAM (Write RAM): loại đĩa này có thể ghi lại nhiều lần nên được dùng như đĩa cứng và đĩa mềm. WRAM là loại từ quang, các đầu đọc cũng có thể đọc được đĩa Worm. Nó là phương tiện rất tốt đối với các tệp CAD/CAM và hình đồ hoạ lớn (chục–trăm GByte).

– WORM (Write Once Read Many) có đường kính 12 hay 30cm, một hay hai mặt cho ghi chỉ một lần (không xoá được) và số lần đọc vô hạn. Nhờ việc tạo thành cung và rãnh, thời gian truy cập trung bình 100ms. Nó như một thiết bị ngoài chuẩn giống đĩa cứng.

– CD–ROM: là loại đĩa có dạng giống như đĩa CD âm thanh, đường kính 12cm. Mỗi đĩa chứa 600 – 660MByte (tương đương với 500 đĩa mềm AT).

4.3.2. Các nguyên tắc tổ chức hệ thống lưu trữ

4.3.2.1. Nguyên tắc tổ chức hệ thống lưu trữ theo kiểu RAID (Redundant Arrays of Inexpensive Disks)

Nhu cầu lưu trữ thông tin ngày một gia tăng dòi hỏi con người phải cho ra đời những hệ thống có khả năng lưu trữ một lượng thông tin lớn, thời gian truy cập thông tin nhanh và đặc biệt là tính an toàn trong việc lưu giữ thông tin. Có ý kiến cho rằng chỉ cần tăng dung lượng của ổ đĩa lên, có ý kiến khác cho rằng bố trí nhiều ổ đĩa thành một mảng lớn trong đó cho phép truy cập đồng thời các đĩa. Ý kiến thứ nhất tỏ ra không hợp lý vì không thể tăng mãi dung lượng của ổ đĩa lên khi kỹ thuật chưa cho phép hơn nữa dung lượng lớn thì thời gian truy cập lại chậm hơn. Ý kiến thứ hai hợp lý hơn vì giải quyết được cả hai vấn đề, đó là dung lượng và thời gian truy cập. Tuy nhiên nếu ta đặt giả thiết là các ổ đĩa có cùng mức độ tin cậy thì hệ thống cho bởi ý kiến thứ hai lại có độ an toàn chỉ bằng 1/n lần hệ thống cho bởi ý kiến thứ nhất (n là số đĩa sử dụng trong mảng). Mặt hạn chế của tổ chức hệ thống lưu trữ theo mảng gồm các đĩa sẽ được khắc phục bằng cách thêm vào các đĩa phụ (đĩa dư thừa), các đĩa này sẽ làm nhiệm vụ

khắc phục các sự cố từ các đĩa khác (các đĩa đang hoạt động), đây chính là nguyên tắc cơ bản của một hệ thống lưu trữ được tổ chức theo kiểu RAID.

Làm thế nào để phát hiện lỗi ở các đĩa trong hệ thống RAID?

Đối với các đĩa từ thì trên mỗi sector đều có ghi các thông tin phụ để phát hiện ra lỗi trên chính sector đó. Khi đọc một sector, phần mạch điều khiển sẽ đọc luôn cả các thông tin phụ đó và phân tích lập tức sẽ phát hiện ra lỗi hoặc mất thông tin.

Một vấn đề khác, làm thế nào để giảm thời gian sửa lỗi trong hệ thống RAID?

Cách phổ biến nhất là thêm vào hệ thống các đĩa phụ, các đĩa này sẽ được đưa vào hoạt động ngay khi có một đĩa trong hệ thống bị lỗi, thông tin của đĩa bị lỗi này sẽ được xây dựng lại từ chính dữ liệu dự trữ của nó chứa trong một đĩa khác trong hệ thống RAID.

Việc thay thế các phần tử lỗi và xây dựng lại dữ liệu từ các phần tử đó có thể thực hiện nhờ vào các phần tử "nóng" luôn được thường trực sẵn sàng. Như vậy hệ thống lưu trữ theo kiểu RAID sẽ giúp hệ thống máy tính nói chung có thể luôn trong trạng thái sẵn sàng.

RAID cấp 0: Trong cấp này của RAID, dữ liệu được lấy ra từ các đĩa trong hệ thống nhưng không có các thông tin phụ đi kèm để phát hiện và khắc khục lỗi khi có sự cố.

RAID cấp 1 và cấp 2: Trong cấp này, các đĩa trong hệ thống được tổ chức thành cặp. Khi thông tin được ghi lên đĩa chính thì cũng đồng thời được ghi lên đĩa phụ. Khi đĩa chính bị sự cố, đĩa phụ sẽ thay thế và có chức năng hoàn toàn như đĩa chính. Hệ thống tổ chức theo kiểu này sẽ cần rất nhiều đĩa dẫn tới lãng phí.

RAID cấp 3: Khác hẳn với cấp 2, ở cấp này, hệ thống chỉ cần một đĩa phụ "Đĩa dư thừa" cho N đĩa khác.

Thông tin ghi vào đĩa phụ không phải là nguyên bản thông tin ở các đĩa khác mà chỉ là tập hợp các thông tin lưu trên các đĩa khác và thông tin kiểm tra khi một đĩa có sự cố. Những thông tin lưu trên đĩa phụ này cho phép khôi phục lại thông tin bị mất trên bất kỳ một đĩa nào trong hệ thống.

RAID cấp 4 và cấp 5: Ở 2 cấp này, tỷ lệ đĩa phụ (đĩa dự phòng)/ đĩa chính cũng giống như ở cấp 3 chỉ khác ở cách truy cập các đĩa. Thông thường hoạt động truy cập đọc diễn ra trên tất cả các đĩa, tuy nhiên hệ thống vẫn cho phép một vài hoạt động truy cập đọc đơn lẻ diễn ra trên một

vài đĩa. Hoạt động này diễn ra trong thời gian các thông tin kiểm soát lỗi được đọc ra trên mỗi sector.

Hoạt động ghi đơn lẻ thì phức tạp hơn nhiều, đầu tiên dữ liệu mới được ghi vào một đĩa nào đó phải được đọc và so sánh với dữ liệu ở tất cả các đĩa còn lại để tìm ra các bit khác nhau, sau đó các bit khác nhau đó được ghi vào đĩa dự phòng và dữ liệu mới được ghi vào đĩa. Theo cách này thì sẽ mất rất nhiều quá trình truy cập, vì vậy để đơn giản hoá người ta làm theo cách khác như sau: Dữ liệu mới cần ghi vào một đĩa nào đó sẽ được so sánh với dữ liệu cũ ở đĩa đó. Sự khác nhau giữa hai dữ liệu này lại được so sánh với dữ liệu ở đĩa dự phòng, kết quả sẽ được các bit khác nhau ghi vào đĩa dự phòng còn dữ liệu mới ghi vào đĩa cần ghi ban đầu.

Sự khác nhau duy nhất giữa RAID 4 và RAID 5 là các đĩa dự phòng (chứa tập hợp các thông tin của các đĩa) được tập trung trong một khối đối với RAID 4 và phân tán trong các khối đối với RAID 5.

RAID cấp 6: Đây là cấp độ có tính an toàn cao nhất đối với dữ liệu được lưu trữ. Thông tin dư thừa có khả năng phát hiện và sửa lỗi cho dữ liệu trên mỗi đĩa được phân tích theo hai cách: Cách thứ nhất giống như các cấp độ 4, 5 của RAID. Đó là có một hệ thống đĩa chẵn lẻ để lưu lại các thông tin dư thừa của mỗi đĩa. Cách thứ hai là phân tích trên toàn bộ cơ sở dữ liệu và thông tin ở các đĩa khác. Theo cách này thì số lượng "đĩa dư thừa" sẽ phải lớn gấp đôi so với ở cấp 4 và 5 của RAID.

4.3.2.2. Kết nối giữa các phần tử lưu trữ và CPU

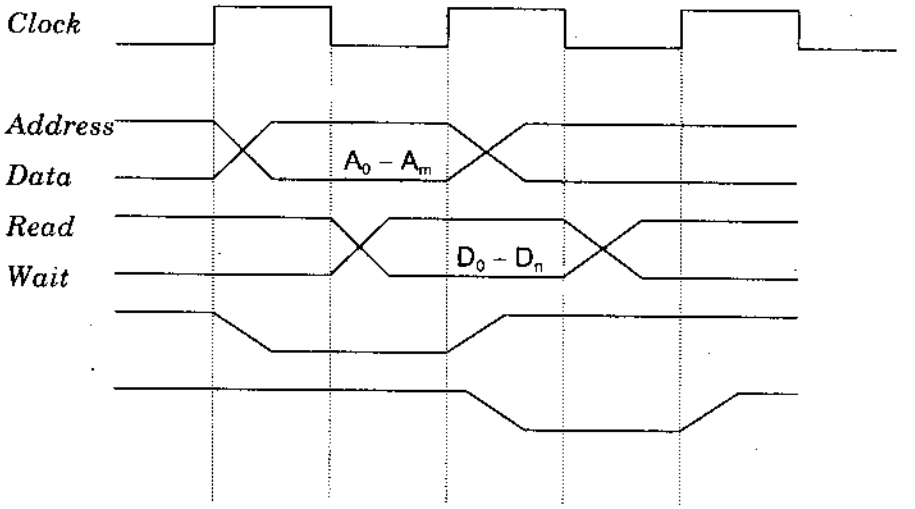
Máy tính gồm nhiều các hệ thống con liên kết với nhau bởi các bus, chức năng cơ bản của bus là truyền thông tin giữa các khối, hệ thống con trong máy tính với nhau.

Ưu điểm của kiến trúc dùng bus nói chung đó là sự linh hoạt trong kết nối, trao đổi thông tin giữa các thành phần, các hệ thống con và chi phí thấp. Nhược điểm là thường tạo ra các điểm "thắt nút cổ chai". Để khắc phục, người ta đưa ra khuyến nghị nên giới hạn lưu lượng thông tin trao đổi trong các hoạt động vào/ra đối với từng hệ thống và một số phần tử vào/ra nên kết nối trực tiếp với các hệ thống lưu trữ không qua bus của máy tính.

Bus trong máy tính thông thường gồm 2 loại chính: bus kết nối CPU và bộ nhớ chính; bus kết nối CPU và các phần tử vào/ra. Ngoài ra cũng có thể phân loại theo chức năng, khi đó chúng ta có bus địa chỉ, bus dữ liệu và bus điều khiển. Trong khi bus kết nối CPU và bộ nhớ chính thường có độ rộng xác định (phụ thuộc vào dung lượng bộ nhớ), tốc độ cao, khoảng cách ngắn

thì bus kết nối CPU và các phần tử vào/ra thường phải được dự trù độ rộng vì các phần tử vào/ra thường không cố định, bus này cũng có tốc độ chậm và dài hơn về mặt vật lý.

Một chu trình của bus thường gồm hai quá trình chính: gửi địa chỉ; đọc hoặc ghi dữ liệu, hình 4.18 minh hoạ cho một chu trình bus đơn giản.



Hình 4.18. Minh hoạ một chu trình BUS đơn giản

Một chu trình đọc bắt đầu bằng việc CPU gửi một địa chỉ cần đọc lên bus địa chỉ cùng với tín hiệu cho phép đọc (/RD). Hệ tín hiệu này tác động lên bộ nhớ và kết quả là bộ nhớ gửi lên bus dữ liệu một hoặc một số byte ứng với địa chỉ CPU đã gửi mà nó đang lưu giữ cùng với một tín hiệu đợi một chu kỳ đồng hồ trước khi chuyển sang một chu trình khác. Chu trình ghi từ CPU vào bộ nhớ hoặc ghi ra các thiết bị ngoài cũng tương tự như vậy.

Vấn đề thiết kế bus như thế nào, chọn kiểu nào, độ rộng, tốc độ... ra sao sẽ có ảnh hưởng rất lớn đến hiệu năng của toàn hệ thống. Trong nội dung của mục này chúng ta chỉ đề cập tới việc thiết kế bus kết nối các phần tử lưu trữ trong hệ thống lưu trữ và CPU.

Hình 4.19 là một ví dụ điển hình về tổ chức bus và kết nối các phần tử lưu trữ với CPU trong các máy tính để bàn. Theo cách tổ chức này, bus được phân cấp theo chức năng, từ trên xuống dưới, bus kết nối giữa CPU và bộ nhớ chính có tốc độ cao nhất còn bus cho các phần tử vào/ra có tốc độ thấp nhất. Thông tin từ các hệ thống lưu trữ ở đây có thể được chuyển tới CPU theo hai cách: Cách thứ nhất là chuyển qua bus I/O và chuyển tới bộ

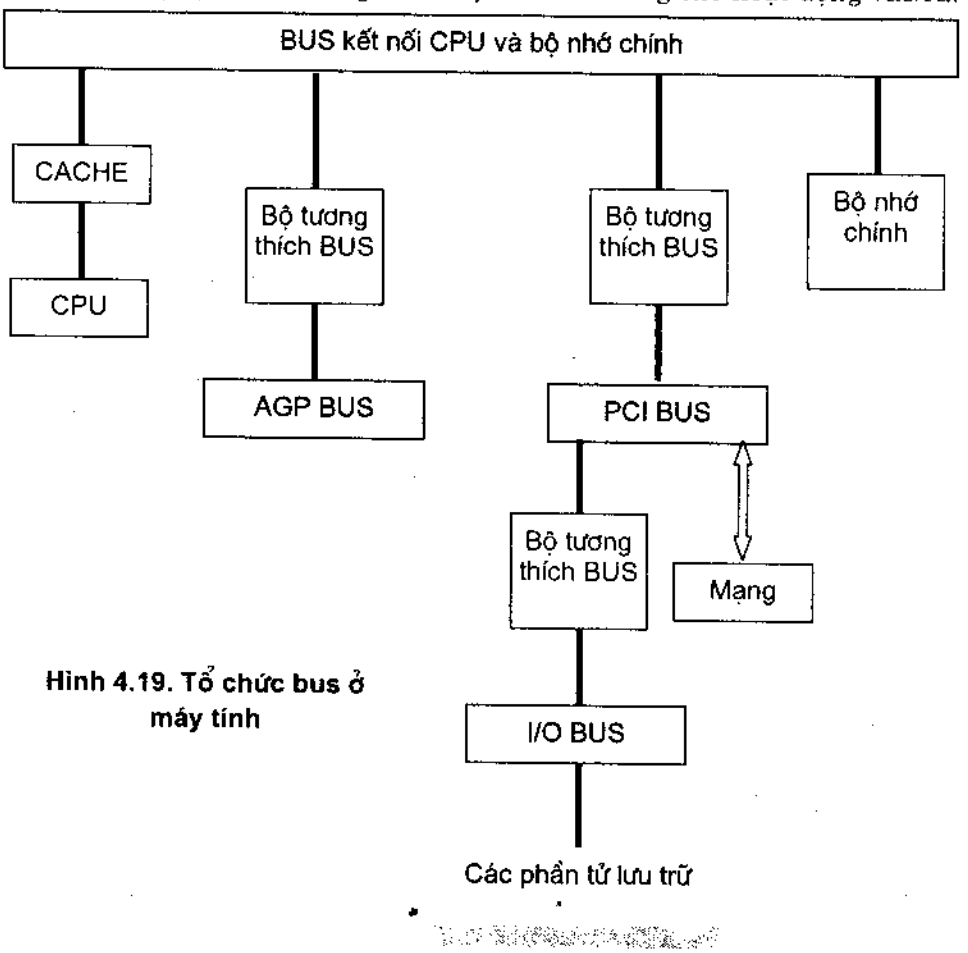
nhớ chính trước khi được đọc vào CPU. Cách thứ hai có thể qua hệ thống mạng rồi cũng được chuyển về bộ nhớ chính.

Một vấn đề đặt ra khi sử dụng bus chung đó là làm thế nào để CPU có thể biết được đâu là các phần tử vào/ra để có thể truy cập đọc/ghi? Có ba cách thức mà tuỳ theo từng bộ vi xử lý khác nhau sẽ lựa chọn cách thức khác nhau:

– Không gian địa chỉ vật lý mà CPU có thể quản lý được chia thành hai phần riêng biệt: một phần cho bộ nhớ chính và một phần cho các phần tử vào/ra.

– CPU có hỗ trợ những tín hiệu đặc trưng cho phép truy cập các phần tử vào/ra (ví dụ như tín hiệu IO/M ở 8086).

– Trong tập lệnh có những nhóm lệnh dành riêng cho hoạt động vào/ra.



Hình 4.19. Tổ chức bus ở máy tính

Trong thực tế nhiều khi ta cần trao đổi thật nhanh với thiết bị ngoại vi như khi đưa dữ liệu ra màn hình hoặc điều khiển đĩa. Trường hợp đó ta cần có được khả năng ghi/đọc trực tiếp không thông qua CPU thì mới đáp ứng được yêu cầu về tốc độ trao đổi. Để làm được điều này các hệ vi xử lý cần có những vi mạch chuyên dụng điều khiển xâm nhập trực tiếp vào bộ nhớ (DMA – Direct Memory Access), điển hình là vi mạch 8237A của Intel có thể điều khiển chuyển 1 byte trong một mảng dữ liệu ra thiết bị ngoại vi chỉ hết 4 chu kỳ đồng hồ trong khi 8086 làm hết khoảng 40 chu kỳ. Trong trường hợp này, CPU ở trạng thái "treo" nhường lại quyền điều khiển bus cho các thiết bị vào/ra. Vấn đề này sẽ được đề cập tới trong chương 7.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4

1. Vì sao phải phân cấp bộ nhớ của máy tính?

2. Vì sao vi xử lý đọc/ghi dữ liệu ở ROM và RAM nhanh hơn ở ổ đĩa cứng?

3. Khi được RESET, 8086 sẽ thực hiện chương trình trong bộ nhớ nào trước tiên, vì sao?

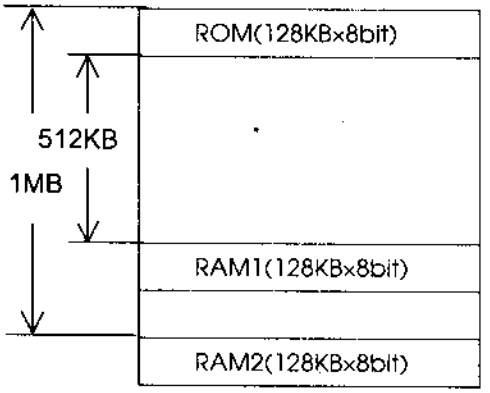
4. Số đường địa chỉ của bộ vi xử lý có liên quan như thế nào với dung lượng của bộ nhớ trên hệ thống máy tính đó?

5. Xây dựng mạch giải mã địa chỉ cho 8 vi mạch nhớ RAM (16KB×8bit) kề nhau, địa chỉ đầu của vi mạch thứ nhất là 80000H.

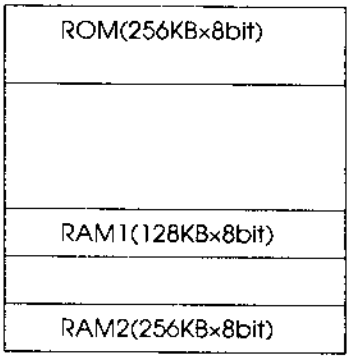
6. Xây dựng mạch giải mã địa chỉ cho bộ nhớ gồm các vi mạch nhớ được bố trí như sau:

ROM(16KB×8bit)	60000H (địa chỉ đầu của ROM)
32KB	
RAM1(16KB×8bit)	
RAM2(16KB×8bit)	
16KB	
RAM3(16KB×8bit)	

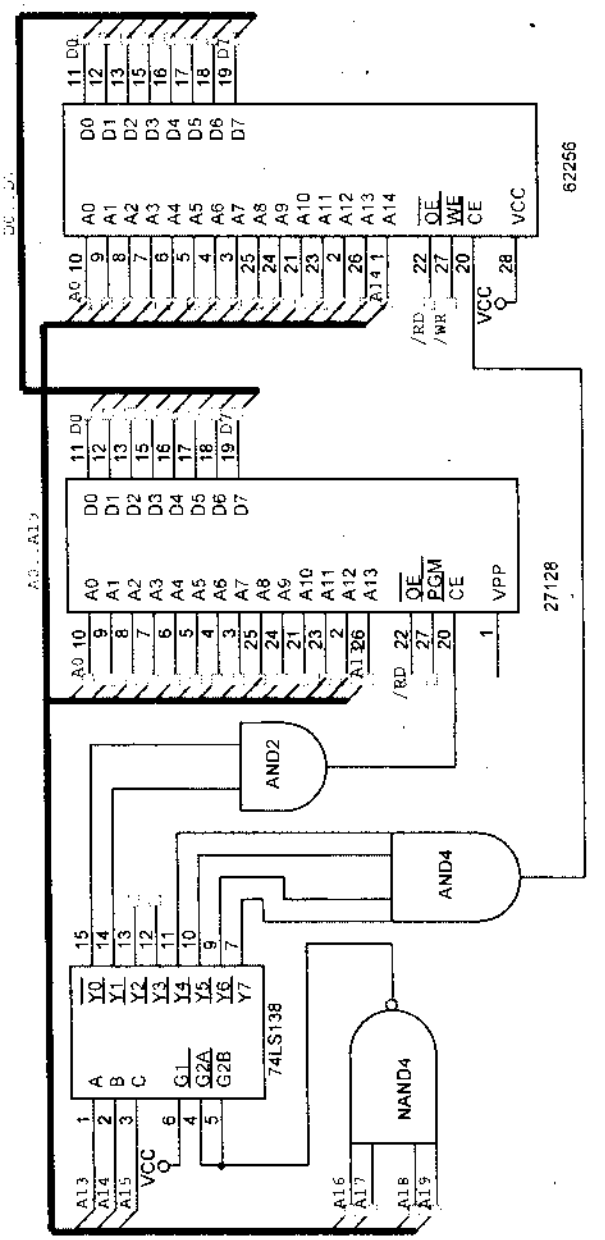
7. Xây dựng mạch giải mã địa chỉ cho bộ nhớ gồm các vi mạch nhớ được bố trí như sau:



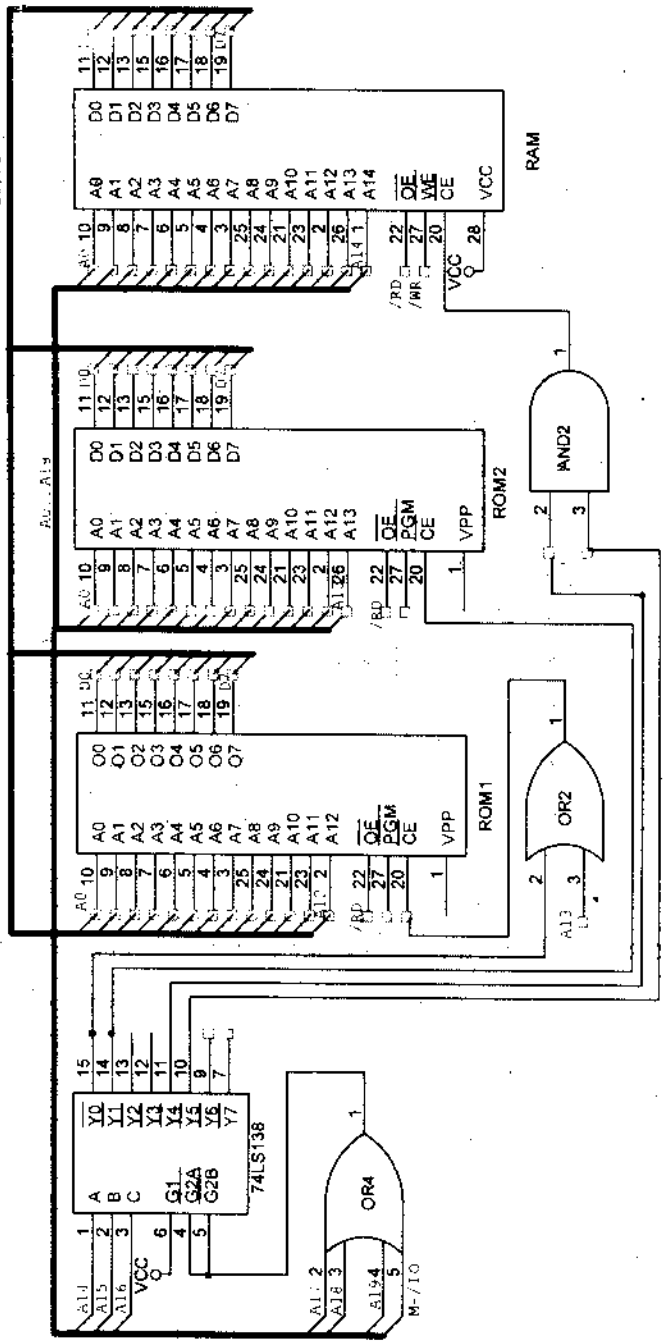
8. Xây dựng mạch giải mã địa chỉ cho bộ nhớ gồm các vi mạch nhớ được bố trí như sau:



9. Cho sơ đồ mạch ghép nối như hình dưới. Xác định địa chỉ đầu, cuối của của các vi mạch nhớ.



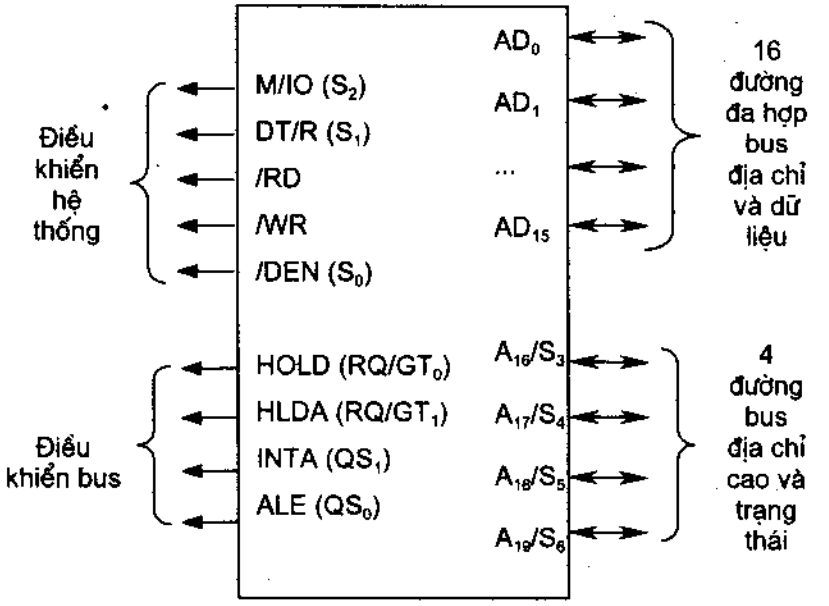
10. Cho sơ đồ mạch ghép nối như hình dưới. Xác định địa chỉ đầu, cuối của của các vi mạch nhớ.



Chương 5. VÀO, RA DỮ LIỆU

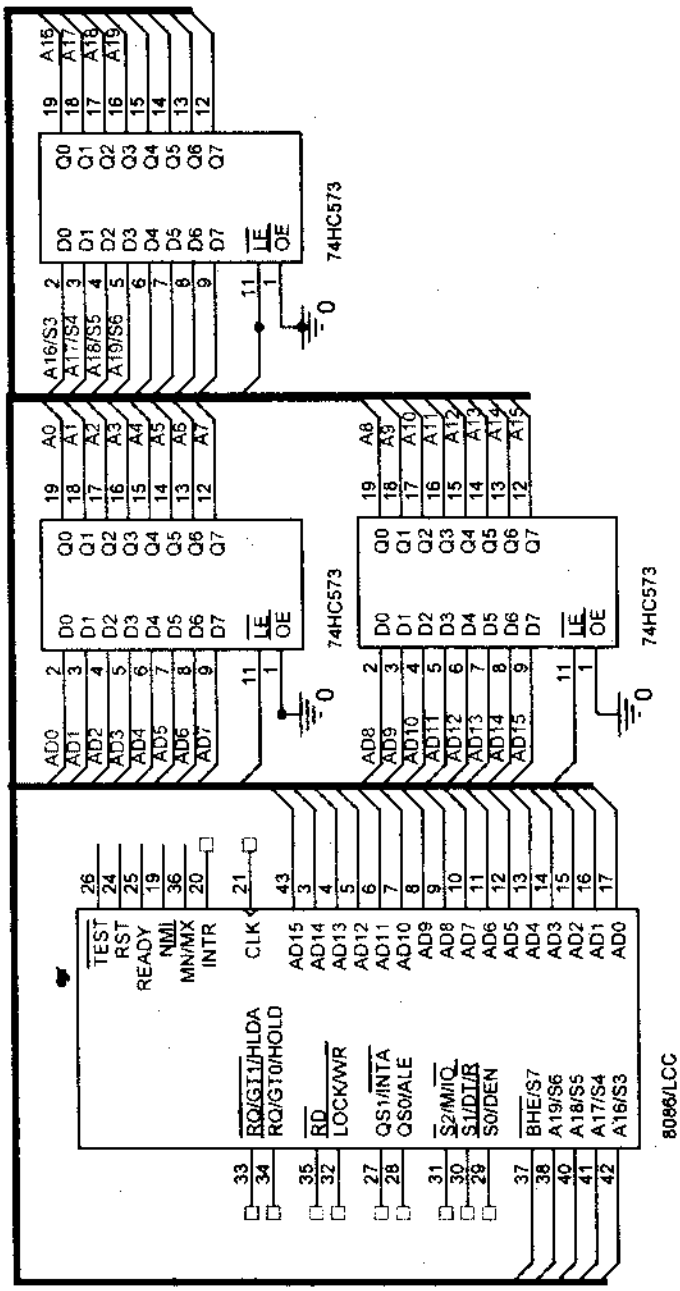
5.1. CÁC TÍN HIỆU PHỤC VỤ TRAO ĐỔI DỮ LIỆU

Trao đổi dữ liệu với thiết bị ngoài thực chất là việc ghi/đọc dữ liệu đến từ thiết bị ngoài. Để thực hiện chức năng này ngoài đường bus dữ liệu, bộ vi xử lý còn cần phải có các đường tín hiệu địa chỉ để quản lý các thiết bị ngoài, các đường tín hiệu điều khiển việc ghi/đọc dữ liệu, các đường tín hiệu trạng thái, cụ thể như sau:



Hình 5.1. Các tín hiệu phục vụ việc trao đổi dữ liệu trên 8086

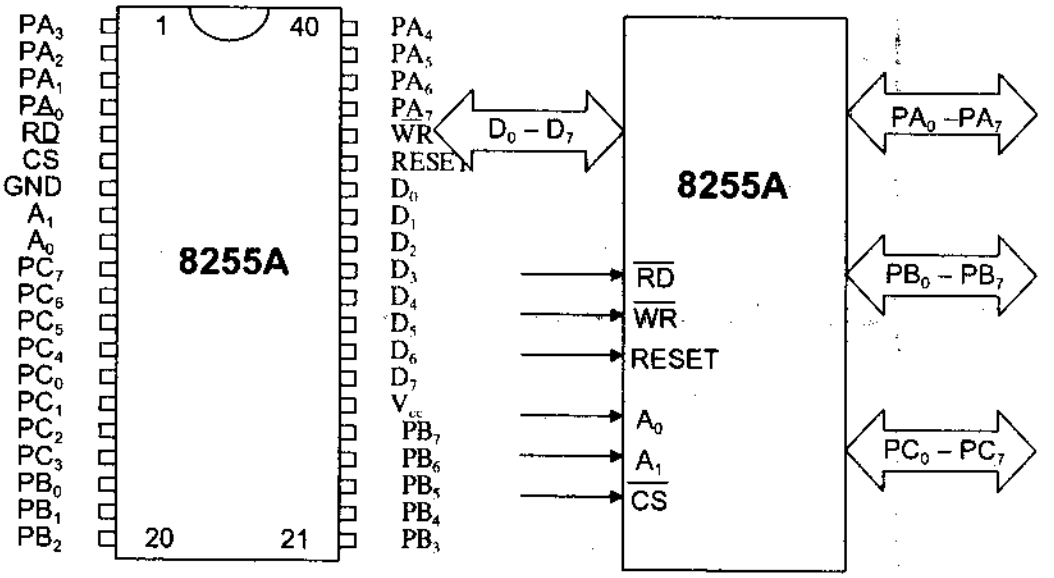
Trong số các đường tín hiệu phục vụ việc trao đổi dữ liệu của 8086 có một số đường là tín hiệu đa hợp, để sử dụng được các đường tín hiệu này đúng mục đích thì cần có một mạch chốt tách riêng các tín hiệu trên các chân đa hợp (hình 5.2).



Hình 5.2. Tách tín hiệu địa chỉ trên các chân đa hợp

5.2. VÀO/RA DỮ LIỆU VỚI 8255A

5.2.1. Sơ đồ cấu trúc, sơ đồ chân vi mạch 8255A



Hình 5.3. Sơ đồ chân, sơ đồ logic của 8255A

$D_0 - D_7$: bus dữ liệu (hai chiều).

Reset: Cho phép khởi tạo trạng thái ban đầu cho vi mạch.

/CS (Chip Select): Tín hiệu chọn vi mạch

/RD (Read): Tín hiệu cho phép đọc

/WR (Write): Tín hiệu cho phép ghi

$A_0 - A_1$: Tín hiệu địa chỉ

$PA_7 - PA_0$: Cổng A

$PB_7 - PB_0$: Cổng B

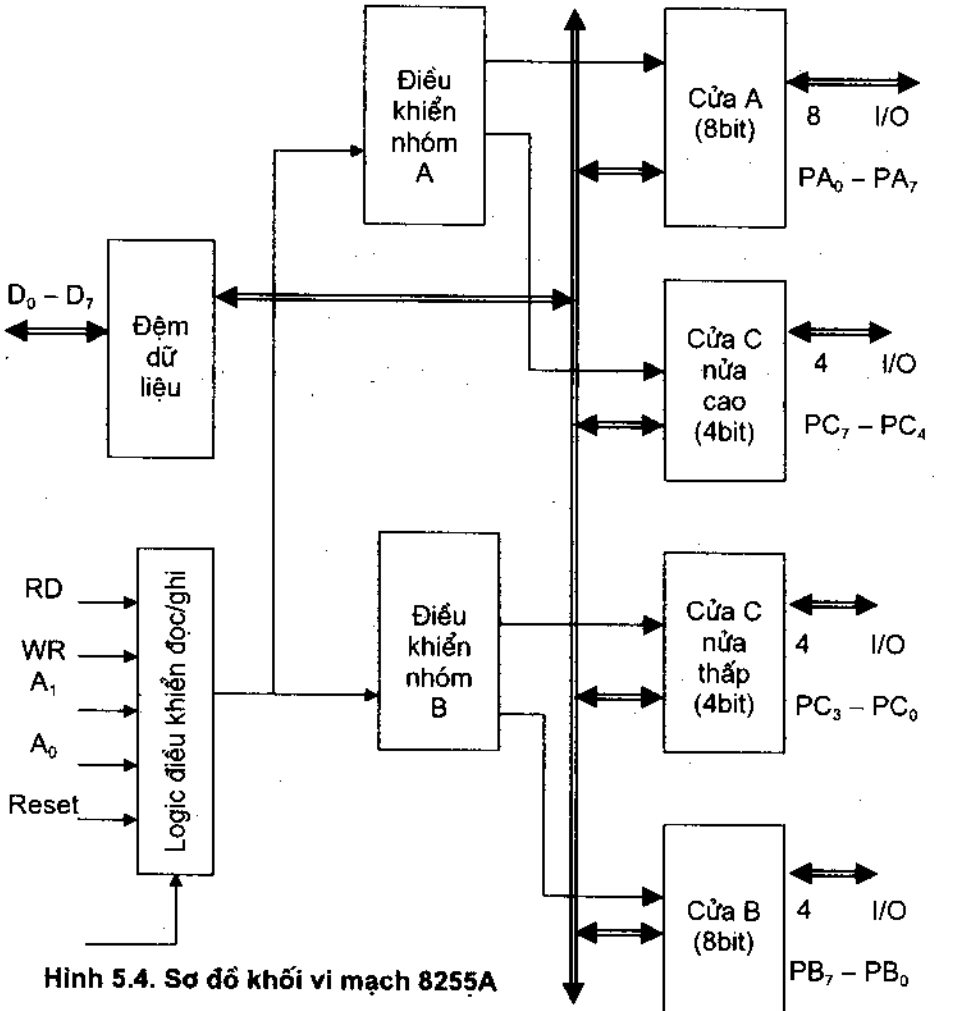
$PC_7 - PC_0$: Cổng C

5.2.1.1. Phần ghép nối với vi xử lý

Phần ghép nối với vi xử lý bao gồm:

– *Bộ đệm dữ liệu* để trao đổi thông tin về dữ liệu hai chiều (vào/ra) giữa đường dây của vi xử lý và đường dây bus nội của 8255A.

– *Bộ logic điều khiển đọc/ghi*: là bộ giải mã địa chỉ lệnh do các thanh ghi đệm và thanh ghi điều khiển.



Hình 5.4. Sơ đồ khối vi mạch 8255A

5.2.1.2. Phần ghép nối với thiết bị ngoài

Phần ghép nối với thiết bị ngoài bao gồm:

– Cổng A: Thanh ghi đệm dữ liệu (8 bit) vào/ra tuỳ theo chương trình khởi phát.

– Cổng B: Thanh ghi đệm số liệu (8 bit) vào/ra tuỳ theo chương trình khởi phát.

– Cổng C: Nửa cao (4 bit).

– Cổng C: Nửa thấp (4 bit).

Tuỳ theo chế độ sử dụng ghi bởi từ điều khiển cổng C có thể được dùng:

– Trao đổi dữ liệu vào hoặc ra.

– Điều khiển hoặc đối thoại với thiết bị ngoài và vi xử lý khi cổng A và B ở chế độ 0 bằng cách xác lập và xoá từng bit PC_i.

– Điều khiển hoặc đối thoại với thiết bị ngoài và vi xử lý khi cổng A và B ở chế độ 1 và 2.

5.2.1.3. Phần các mạch điều khiển nội bộ

Có các khối điều khiển nhóm A, nhóm B; khối điều khiển các cổng A, B và C.

5.2.2. Các lệnh ghi/đọc các cổng và thanh ghi điều khiển

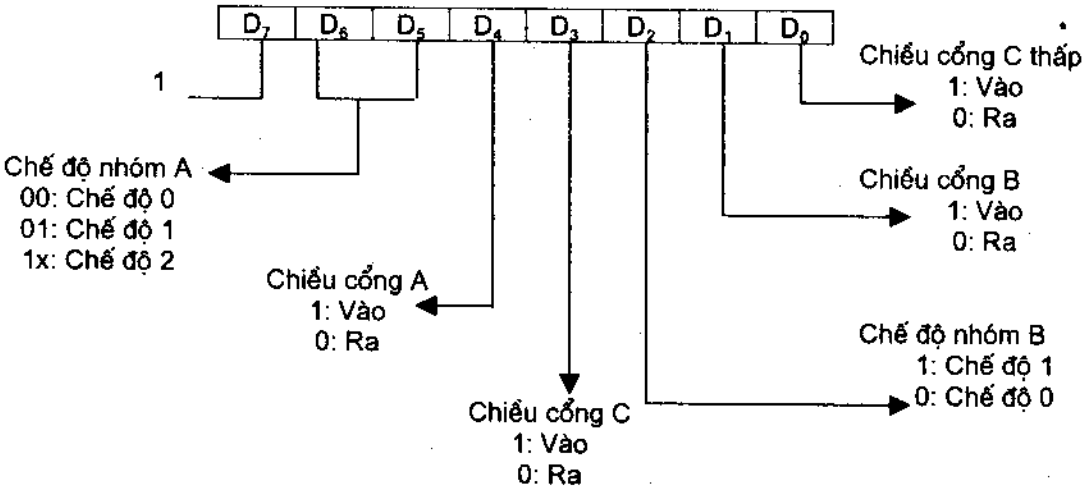
Với tổ hợp của các tín hiệu địa chỉ (A_0A_1), tín hiệu chọn vi mạch (/CS), các tín hiệu đọc (/RD), ghi (/WR) của vi xử lý, ta có thể có các lệnh ghi và đọc khác nhau cho các cổng (A, B, C) và thanh ghi điều khiển tạo ra sự di chuyển dữ liệu giữa đường dây dữ liệu, các cổng và thanh ghi điều khiển. Như vậy, vi mạch 8255A có đặc điểm là không có lệnh đọc thanh ghi trạng thái mà dùng lệnh đọc cổng C khi vi mạch ở chế độ 1 và 2, còn ở chế độ 0, không đọc được trạng thái.

A_1	A_0	/CS	/RD	/WR	Lệnh (của VXL)	Chiều di chuyển dữ liệu
0	0	0	0	1	Đọc cổng A	Cổng A → $D_0 - D_7$
0	1	0	0	1	Đọc cổng B	Cổng B → $D_0 - D_7$
1	0	0	0	1	Đọc cổng C	Cổng C → $D_0 - D_7$
1	1	0	0	1		Không có giá trị
0	0	0	1	0	Ghi cổng A	$D_0 - D_7$ → Cổng A
0	1	0	1	0	Ghi cổng B	$D_0 - D_7$ → Cổng B
1	0	0	1	0	Ghi cổng C	$D_0 - D_7$ → Cổng C
1	1	0	1	0	Ghi thanh ghi điều khiển	$D_0 - D_7$ thanh ghi điều khiển
x	x	1	x	x	Vi mạch ở trạng thái điện trở cao	Không có trao đổi dữ liệu

Bảng 5.1. Logic ghi/đọc của 8255A

5.2.3. Các chế độ hoạt động

Tuỳ giá trị ghi vào thanh ghi điều khiển khi khởi tạo, vi mạch có thể hoạt động ở các chế độ 0, 1, 2 khác nhau, chiều của các cổng A, B, C có thể là ra hoặc vào.



Hình 5.5. Cấu trúc từ điều khiển của 8255A

5.2.3.1. Chế độ 0

Chế độ này còn được gọi là chế độ vào/ra cơ sở vì:

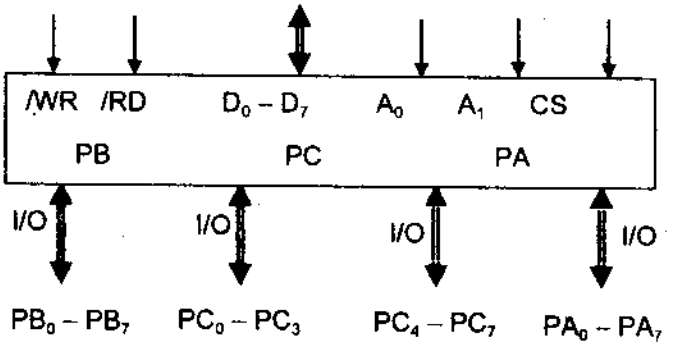
– Các cổng A, B và C được sử dụng độc lập với nhau.

– Các cổng A, B và C có thể vào hoặc ra tuỳ thuộc giá trị của từ điều khiển chế độ ghi vào thanh ghi điều khiển.

– Dữ liệu ra được chốt.

– Dữ liệu vào không được chốt.

– Không có sự đối thoại với thiết bị ngoài.



Hình 5.6. Mô tả hoạt động của 8255A ở chế độ 0

5.2.3.2. Chế độ 1

Chế độ này còn gọi là chế độ vào/ra có đột cửa hay đối thoại với các bit của cổng C. Các cổng A, B, C được chia thành 2 nhóm:

– Nhóm A gồm cổng A để trao đổi dữ liệu và cổng C cao (PC_7 – PC_4) để đối thoại với vi xử lý và thiết bị ngoài.

– Nhóm B gồm cổng B để trao đổi dữ liệu và cổng C thấp (PC_0 – PC_3) để đối thoại với vi xử lý và thiết bị ngoài.

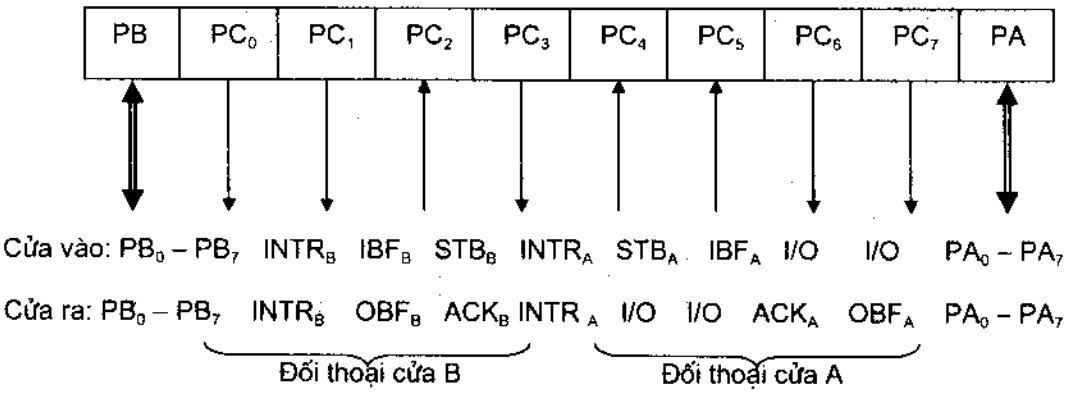
Chiều và chế độ của cổng A, B do từ điều khiển quyết định, còn các tín hiệu đối thoại PC_i phụ thuộc chiều cổng vào hay ra.

Ở chế độ 1 ta thấy:

– PC_0 luôn là tín hiệu $INTR_B$ – tín hiệu yêu cầu ngắt chương trình cho cổng B.

– PC_3 luôn là tín hiệu $INTR_A$ – tín hiệu yêu cầu ngắt chương trình cho cổng A.

– PC_2 luôn là tín hiệu vào, nhận các tín hiệu yêu cầu STB_B và xác nhận ACK_B của thiết bị ngoài cho cổng B tương ứng với chiều vào hay chiều ra. Còn cổng A nếu là cổng vào, PC_4 nhận STB_A của thiết bị ngoài và PC_6 nhận ACK_A của thiết bị ngoài nếu cổng A là cổng ra.



Hình 5.7. Mô tả hoạt động của 8255A ở chế độ 1

5.2.3.3. Chế độ 2

Chế độ này chỉ dùng cho cổng A với vào/ra thuận nghịch và các bit PC_3, PC_4 – PC_7 dùng làm các tín hiệu đối thoại, trong đó:

– PC_3 cho tín hiệu yêu cầu ngắt $INTR_A$ chung cho cả hai chiều và giống chế độ 1.

– PC_4 cho tín hiệu vào STB_A khi cổng A có chiều vào.

– PC_6 cho tín hiệu vào ACK_A khi cổng A có chiều ra.

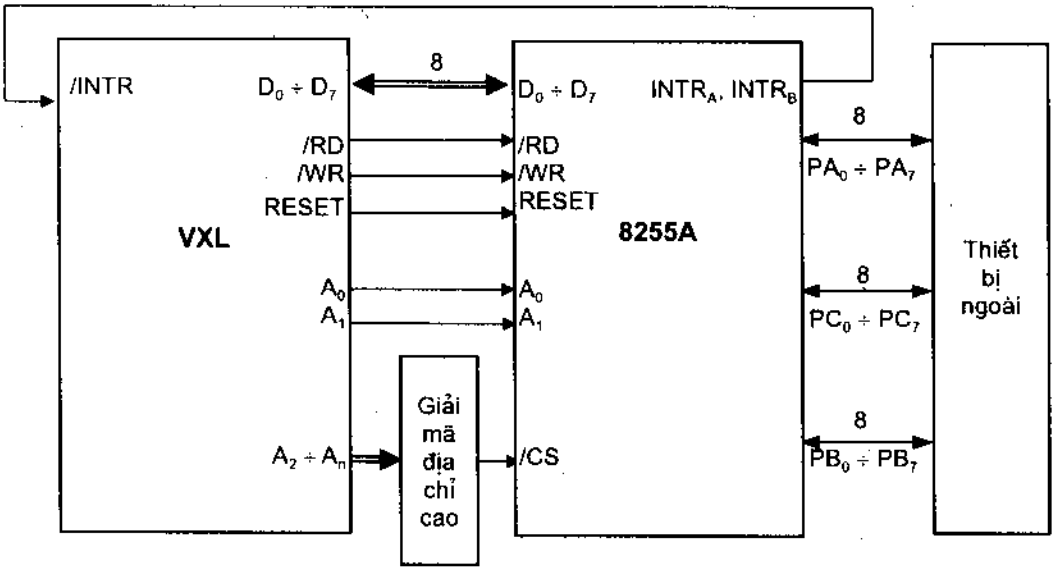
Chung cả hai chế độ 1 và 2 các bit còn lại dùng làm đối thoại của cổng C đều là các tín hiệu ra:

– IBF_A, IBF_B chỉ dữ liệu vào đã ghi đầy vào các cổng A hoặc B để yêu cầu thiết bị ngoài không đưa dữ liệu vào nữa.

– OBF_A, OBF_B chỉ dữ liệu ra đã ghi đầy vào các cổng A hoặc B để yêu cầu thiết bị ngoài đọc dữ liệu.

5.2.4. Ghép nối 8255 với vi xử lý và thiết bị ngoài

8255A là một trong các vi mạch hỗ trợ các bộ vi xử lý thế hệ 8086 trong việc giao tiếp với các thiết bị ngoài. Trong một hệ thống máy tính 8255A là trung gian giữa vi xử lý và các thiết bị ngoài (hình 5.8).



Hình 5.8. Phối ghép 8086 với 8255A

5.2.4.1. Phần ghép nối với vi xử lý

Các tín hiệu đường dây về dữ liệu ($D_0 \div D_7$), địa chỉ thấp ($A_0 A_1$), tín hiệu cho phép đọc /RD, cho phép ghi /WR được nối thẳng với các tín hiệu tương ứng của 8255A.

Tín hiệu /CS của 8255A được nối với bộ giải mã các địa chỉ cao (A_2 – A_n) của vi xử lý.

Các tín hiệu yêu cầu ngắt chương trình $INTR_A$, $INTR_B$ của 8255A được nối vào lối vào INTR của vi xử lý.

5.2.4.2. Phần ghép nối với thiết bị ngoài

Tuỳ thuộc vào thiết bị ngoài (đưa tin vào, nhận tin ra) số bit của đường dây dữ liệu (8, 16 bit) và phương thức trao đổi thông tin mà ta có thể sử dụng 8255A ở các chế độ khác nhau.

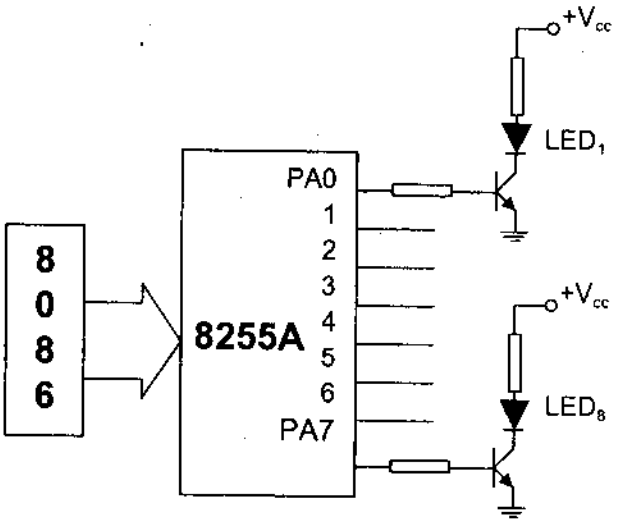
– 8255A ở chế độ 0: Ba cổng A, B và C đều có thể được dùng để trao đổi dữ liệu độc lập với nhau.

– 8255A ở chế độ 1: Chỉ có hai cổng PA, PB là trao đổi dữ liệu độc lập nhau, còn các đường dây PC_i của cổng PC được dùng để đối thoại cho các cổng PA và PB. Các đường dây này có chiều xác định và có vai trò xác định (không thể thay đổi chiều và vai trò sử dụng).

– 8255A ở chế độ 2: Chỉ cho cổng A có thể trao đổi dữ liệu vào/ra, các bit của cổng PC cũng có vai trò và chiều xác định.

5.2.5. Một số ví dụ lập trình với 8255A

Ví dụ 1: Viết chương trình điều khiển các LED sáng tuần tự theo vòng tròn với thời gian trễ là 1s. Giả sử địa chỉ các cổng A, B, C và thanh ghi điều khiển đã được định sẵn là 19H, 1BH, 1DH, 1FH (hình 5.9).



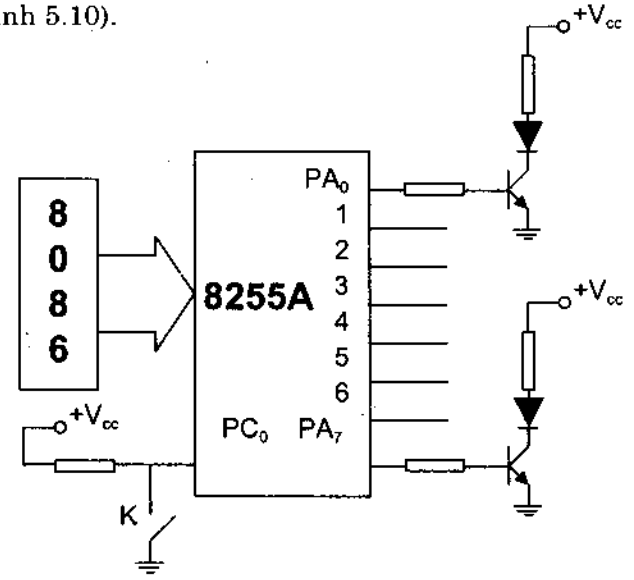
Hình 5.9. 8255A điều khiển các LED sáng tuần tự

Dưới đây là chương trình đầy đủ:

```
CODE SEGMENT
    ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
    ;
CREG EQU 1FH
PC EQU 1DH
PB EQU 1BH
PA EQU 19H
    ;
    ORG 1000H
    MOV AL,10000000B
; từ điều khiển (MODE0 các cổng có chiều ra)
    OUT CREG,AL
    ;
    MOV AL,00000000B ; cấm các cổng B, C
    OUT PB,AL
    OUT PC,AL
    ;
    MOV AL,00000001B ; LED thứ nhất sáng
L:
OUT PA,AL
    CALL TIMER ; trễ 1 giây
    ROL AL,1
    JMP L
    ;
    INT 3
    ;
TIMER:
MOV CX,1
TIMER2:
```

```
PUSH CX
    MOV CX,0
TIMER1:
NOP
    NOP
    NOP
    NOP
    LOOP TIMER1
    POP CX
    LOOP TIMER2
    RET
    ;
CODE ENDS
    END
```

Ví dụ 2: Viết chương trình điều khiển các LED sáng nếu khoá K đóng và tắt nếu khoá K mở (hình 5.10).



Hình 5.10. 8255A điều khiển các LED tắt/mở theo khoá K

```
CODE SEGMENT
    ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
    ;
```

```
CREG EQU 1FH
PC EQU 1DH
PB EQU 1BH
PA EQU 19H
   ;
   ORG 1000H
   MOV AL,10000001B
; từ điều khiển (MODE0,a,b RA,PC, vào)
   OUT CREG,AL
   ;
   MOV AL,00000000B ; cấm cổng B
   OUT PB,AL
   ;
nh:
IN AL,PC ;kiểm tra khoá K
   AND AL,01H
   CMP AL,01H
   JNE ledtat
   MOV AL,11111111B ; LED sáng
   OUT PA,AL
   JMP lap
ledtat:
   MOV AL,00000000B ; LED tat
   OUT PA,AL
lap:
JMP nh
   CODE ENDS
   END
```

5.3. VÀO/RA DỮ LIỆU BẰNG DMA (DIRECT MEMORY ACCESS)

5.3.1. Nguyên tắc trao đổi dữ liệu với thiết bị ngoại vi bằng cách truy cập trực tiếp bộ nhớ

Trong các cách điều khiển việc trao đổi dữ liệu giữa thiết bị ngoại vi và hệ vi xử lý bằng cách thăm dò trạng thái sẵn sàng của thiết bị ngoại vi hay bằng cách ngắt bộ vi xử lý đã được nói đến trong phần trước, dữ liệu thường được chuyển từ bộ nhớ qua bộ vi xử lý rồi từ đó ghi vào thiết bị ngoại vi hoặc ngược lại từ thiết bị ngoại vi vào bộ vi xử lý và chuyển đến bộ nhớ. Theo các cách này, tốc độ trao đổi phụ thuộc rất nhiều vào tốc độ của bộ vi xử lý.

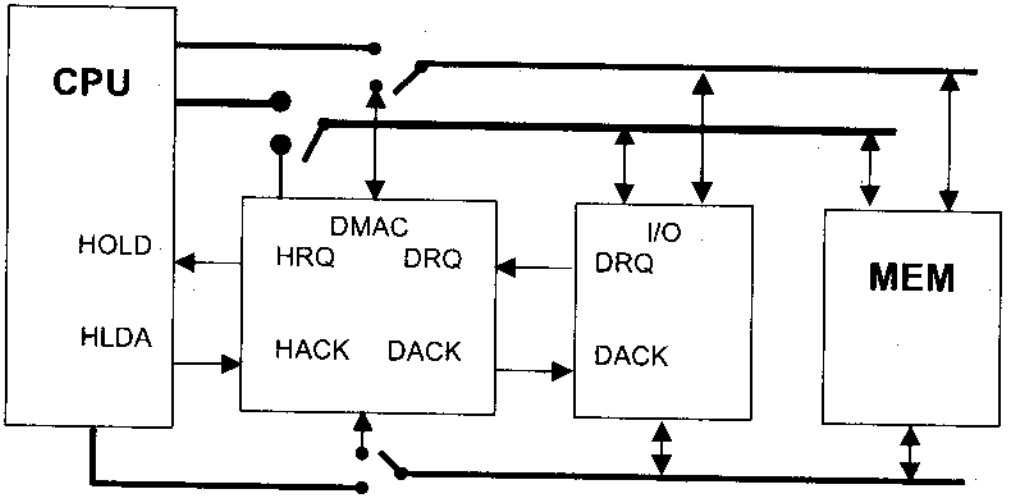
Trong thực tế, nhiều khi ta cần trao đổi nhanh với thiết bị ngoại vi như khi đưa dữ liệu ra màn hình hoặc điều khiển đĩa. Trường hợp đó ta cần có được khả năng ghi/đọc trực tiếp không thông qua CPU thì mới đáp ứng được yêu cầu về tốc độ trao đổi. Để làm được điều này, các hệ vi xử lý cần có những vi mạch chuyên dụng để điều khiển xâm nhập trực tiếp vào bộ nhớ (DMA), điển hình là vi mạch 8237A của Intel có thể điều khiển chuyển 1 byte trong một mảng dữ liệu ra thiết bị ngoại vi chỉ hết 4 chu kỳ đồng hồ trong khi 8086 làm hết khoảng 40 chu kỳ.

```
LAP: MOV AL,[SI]      ; 10 chu kỳ
     OUT PORT,AL      ; 10
      INC SI ; 2
      LOOP LAP ; 17
               ; tổng 39 chu kỳ
```

5.3.2. Sơ đồ khối hệ thống trao đổi kiểu DMA

Để hỗ trợ cho khả năng trao đổi trực tiếp bộ nhớ, các CPU thường có chân yêu cầu treo HOLD. Khi thiết bị ngoại vi có yêu cầu trao đổi trực tiếp bộ nhớ thì thông qua chân này báo cho CPU biết. Đến lượt CPU khi nhận được yêu cầu treo nếu chấp nhận nó sẽ tự treo ra khỏi hệ thống và đưa xung HLDA ra ngoài để cho phép sử dụng bus.

Sơ đồ khối của một hệ thống trao đổi theo kiểu DMA được cho như hình 5.11.



Hình 5.11. Sơ đồ khối một hệ thống trao đổi dữ liệu bằng DMA

Ta nhận thấy trong hệ thống này khi CPU treo thì nó trao quyền sử dụng bus cho DMA. Trong trường hợp này, DMA phải có trách nhiệm quản lý toàn bộ hệ thống. Để làm được điều đó, DMA phải tạo ra được các tín hiệu giống như CPU và bản thân nó phải là một thiết bị lập trình được.

Quá trình hoạt động có thể tóm tắt như sau: Khi thiết bị ngoại vi có yêu cầu trao đổi dữ liệu trực tiếp với bộ nhớ, nó đưa yêu cầu treo DRQ = 1 đến DMAC, DMAC sẽ đưa yêu cầu HRQ = 1 đến chân HOLD của CPU. Nhận được yêu cầu treo CPU sẽ tự treo các bus của mình và trả lời qua tín hiệu HLDA = 1 đến chân HACK của DMAC. DMAC thông báo cho thiết bị ngoại vi qua tín hiệu DACK = 1. Khi quá trình DMA kết thúc thì DMAC đưa ra tín hiệu HRQ = 0.

5.3.3. Các phương thức trao đổi dữ liệu

Trong thực tế có 3 kiểu trao đổi dữ liệu bằng cách xâm nhập trực tiếp bộ nhớ.

** Trao đổi cả mảng dữ liệu*

Trong chế độ này, CPU bị treo trong suốt quá trình trao đổi mảng dữ liệu. Chế độ này được dùng khi ta có nhu cầu trao đổi dữ liệu với ổ đĩa hoặc màn hình, các thủ tục diễn ra như sau:

– CPU phải ghi từ điều khiển và từ chế độ làm việc vào DMAC để quy định cách thức làm việc, địa chỉ đầu của mảng, độ dài của mảng...

– Khi thiết bị ngoại vi có yêu cầu trao đổi dữ liệu, nó đưa DRQ = 1 đến DMAC:

+ DMAC đưa ra tín hiệu HRQ đến chân HOLD của CPU để yêu cầu treo CPU. Tín hiệu HOLD phải ở mức cao trong suốt quá trình trao đổi.

+ Nhận được yêu cầu treo, CPU kết thúc chu kỳ bus hiện tại sau đó nó treo các bus của mình và đưa tín hiệu HLDA báo cho DMAC được toàn quyền sử dụng bus.

+ DMAC đưa ra xung DACK báo cho thiết bị ngoại vi biết là đã có thể trao đổi dữ liệu.

+ DMAC bắt đầu chuyển dữ liệu từ bộ nhớ ra thiết bị ngoại vi bằng cách đưa địa chỉ của byte đầu ra bus địa chỉ và đưa tín hiệu MEMR = 0 để đọc một byte từ bộ nhớ ra bus dữ liệu. Tiếp đó DMAC đưa ra tín hiệu IOW = 0 để ghi dữ liệu ra thiết bị ngoại vi. Sau đó DMAC giảm số byte cần chuyển đi, cập nhật địa chỉ của byte cần đọc tiếp và lặp lại cho đến khi hết mảng dữ liệu.

– Khi quá trình DMA kết thúc, DMAC đưa ra tín hiệu HRQ = 0 để báo cho CPU biết để CPU giành lại quyền điều khiển bus hệ thống.

** Treo CPU để trao đổi từng byte*

Trong cách trao đổi này CPU không bị treo lâu dài nhưng thỉnh thoảng lại bị treo một khoảng thời gian rất ngắn đủ để trao đổi một byte dữ liệu (CPU bị lấy mất một số chu kỳ đồng hồ). Do bị lấy mất một số chu kỳ đồng hồ nên tốc độ xử lý một công việc nào đó bị chậm lại. Cách hoạt động tương tự như lần trước chỉ khác là mỗi lần trao đổi một byte.

** Tận dụng thời gian CPU không dùng bus để trao đổi dữ liệu*

Trong cách trao đổi dữ liệu này ta phải có logic phụ thêm ngoài cần thiết để phát hiện chu kỳ xử lý nội bộ của CPU (không dùng đến bus ngoài) và tận dụng các chu kỳ đó để tiến hành trao đổi. Trong cách này thì DMAC và CPU thay nhau sử dụng bus và kiểu xâm nhập này không ảnh hưởng gì đến hoạt động của CPU.

Chương 6. NGẮT VÀ XỬ LÝ NGẮT

6.1. KHÁI NIỆM VÀ SỰ CẦN THIẾT PHẢI NGẮT CPU

Trong thực tế người ta rất muốn tận dụng khả năng của CPU để làm thêm được nhiều công việc khác nữa, chỉ khi nào cần trao đổi dữ liệu thì mới yêu cầu CPU tạm dừng công việc hiện tại để phục vụ việc trao đổi dữ liệu. Sau khi hoàn thành việc trao đổi dữ liệu thì CPU lại quay về làm tiếp công việc hiện đang bị gián đoạn. Cách làm việc theo kiểu này gọi là *ngắt CPU (gián đoạn hoạt động của CPU)*. Một hệ thống sử dụng *ngắt* có thể đáp ứng rất nhanh các yêu cầu trao đổi dữ liệu trong khi vẫn có thể làm được các công việc khác. Khi hệ thống có sử dụng ngắt thì nó phải được tổ chức sao cho có khả năng thực hiện các *chương trình phục vụ ngắt* tại các địa chỉ xác định của CPU. Khi nghiên cứu các tín hiệu của CPU 8086, ta đã thấy vi mạch này có các chân tín hiệu cho các yêu cầu ngắt che được INTR và không che được NMI. Các chân tín hiệu yêu cầu ngắt này sẽ được sử dụng vào việc đưa các yêu cầu ngắt từ bên ngoài đến CPU.

6.2. TỔ CHỨC NGẮT Ở 8086

Trong hệ vi xử lý 8086 có thể xếp các nguyên nhân gây ra ngắt CPU vào 3 nhóm như sau:

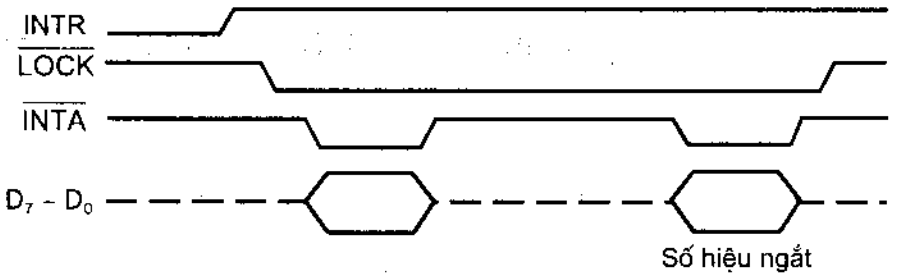
* *Nhóm các ngắt cứng*

Ngắt cứng là các yêu cầu ngắt CPU do các tín hiệu đến từ các chân INTR và NMI.

Ngắt cứng NMI là yêu cầu ngắt không che được tương đương với ngắt mềm INT2. Các lệnh CLI (xoá cờ IF) và STI (lập cờ IF) không có ảnh hưởng đến việc nhận biết tín hiệu yêu cầu ngắt NMI.

Ngắt cứng INTR là yêu cầu ngắt che được. Các lệnh CLI và STI có ảnh hưởng trực tiếp tới trạng thái của cờ IF trong bộ vi xử lý, tức là ảnh hưởng tới việc CPU có nhận biết yêu cầu ngắt tại chân này hay không. Yêu cầu ngắt tại chân INTR có thể có kiểu ngắt N nằm trong khoảng 0 – 0FFH. Kiểu ngắt này phải được đưa bus dữ liệu để CPU có thể đọc được khi có xung INTA trong chu kỳ trả lời chấp nhận ngắt.

Biểu đồ thời gian của các xung liên quan đến quá trình trên được mô tả trên hình 6.1.



Hình 6.1. Chu kỳ trả lời ngắt của CPU 8086

** Nhóm các ngắt mềm*

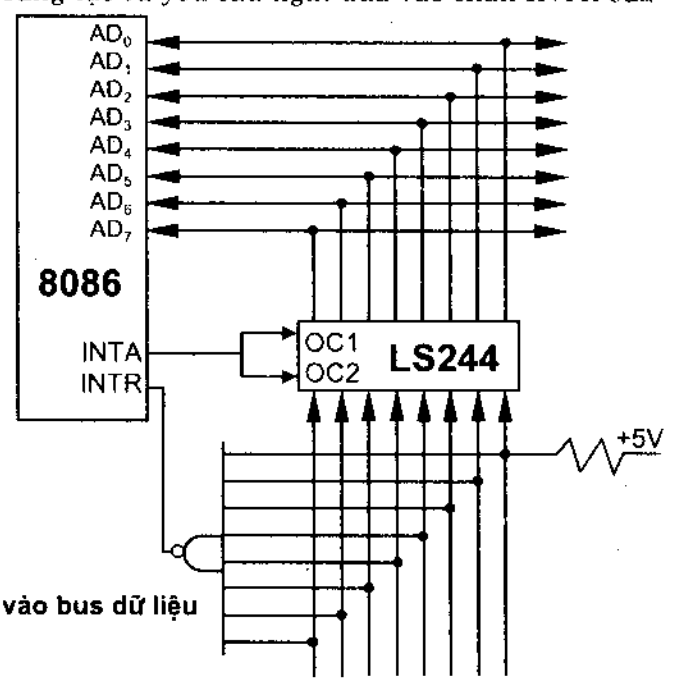
Ngắt mềm là khi CPU thực hiện các lệnh ngắt dạng INT N, trong đó N là số hiệu hay kiểu ngắt nằm trong khoảng 0 -- 225.

** Nhóm các hiện tượng ngoại lệ*

Đó là ngắt do các lỗi nảy sinh trong quá trình hoạt động của CPU như phép chia cho 0, xảy ra tràn khi tính toán.

Yêu cầu ngắt sẽ được CPU kiểm tra thường xuyên tại chu kỳ đồng hồ cuối cùng của mỗi lệnh.

Trên hình 6.2 trình bày một cách đơn giản để đưa được số hiệu ngắt N vào bus dữ liệu trong khi cũng tạo ra yêu cầu ngắt đưa vào chân INTR của bộ vi xử lý 8086.



Hình 6.2. Đưa số hiệu ngắt vào bus dữ liệu

Giả thiết trong một thời điểm nhất định chỉ có một yêu cầu ngắt IR_i được tác động và khi đó ở đầu ra của mạch NAND sẽ có xung yêu cầu ngắt đến CPU. Tín hiệu IR_i được đồng thời đưa qua mạch khuếch đại đệm để tạo ra số hiệu ngắt tương ứng, số hiệu ngắt này sẽ được CPU đọc vào khi nó đưa ra tín hiệu trả lời INTA.

Bảng 6.1. Quan hệ giữa IR_i và số hiệu ngắt N tương ứng

AD_7	IR_6	IR_5	IR_4	IR_3	IR_2	IR_1	IR_0	N
1	1	1	1	1	1	1	0	FEH (254)
1	1	1	1	1	1	0	1	FDH (253)
1	1	1	1	1	0	1	1	FBH (251)
1	1	1	1	0	1	1	1	F7H (247)
1	1	1	0	1	1	1	1	EFH (239)
1	1	0	1	1	1	1	1	DFH (223)
1	0	1	1	1	1	1	1	BFH (191)

6.3. ĐÁP ỨNG CỦA CPU KHI CÓ YÊU CẦU NGẮT

Khi có yêu cầu ngắt kiểu N đến chân CPU và nếu yêu cầu đó được cho phép, CPU thực hiện các công việc sau:

1. SP ← SP–2, {SP} ← ER, trong đó {SP} là ô nhớ do SP chỉ ra.

(chỉ ra đỉnh mới của ngăn xếp, cất thanh ghi cờ vào đỉnh ngăn xếp)

2. IF ← 0, TF ← 0.

(cấm các ngắt khác tác động vào CPU, cho CPU chạy ở chế độ bình thường).

3. SP ← SP–2, {SP} ← CS.

(chỉ ra đỉnh mới của ngăn xếp, cất phần địa chỉ đoạn của địa chỉ trở về vào đỉnh ngăn xếp).

4. SP ← SP–2, {SP} ← IP.

(chỉ ra đỉnh mới của ngăn xếp, cất phần địa chỉ lệch của địa chỉ trở về vào đỉnh ngăn xếp).

5. {N*4} → IP, {N*4+2} → CS.

(lấy lệnh tại địa chỉ mới của chương trình con phục vụ ngắt kiểu N tương ứng trong bảng vectơ ngắt).

6. Tại cuối chương trình phục vụ ngắt, khi gặp lệnh IRET

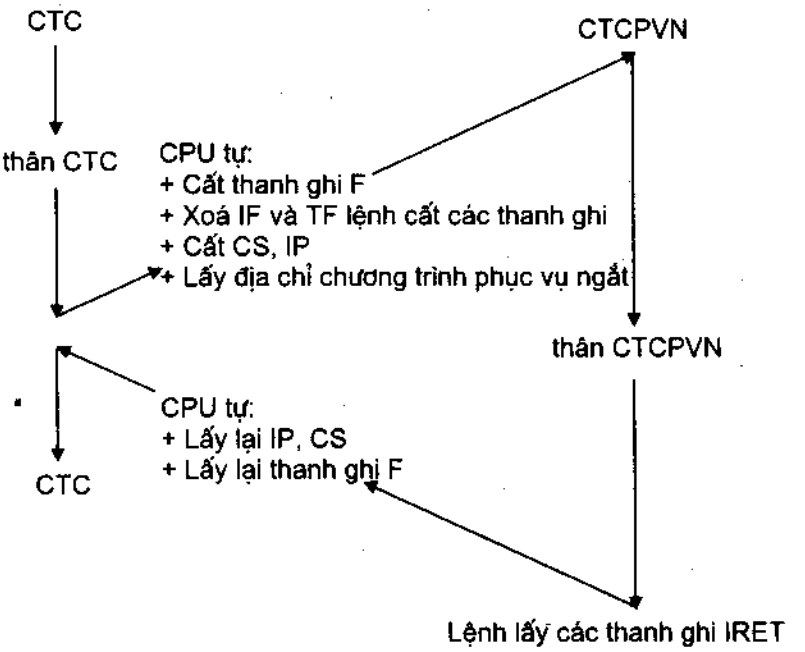
{SP} → IP, SP ← SP + 2

{SP} → CS, SP ← SP + 2

{SP} → FR, SP ← SP + 2

(bộ vi xử lý quay lại chương trình chính tại địa chỉ trở về và với giá trị cũ của thanh ghi cờ được lấy ra từ ngăn xếp).

Về mặt cấu trúc chương trình, khi có ngắt xảy ra thì chương trình chính (CTC) liên hệ với chương trình con phục vụ ngắt (CTCPVN) như mô tả trên hình 6.3.



Hình 6.3. Liên hệ giữa CTC và CTCPVN khi có ngắt

Intel đã quy định một số kiểu ngắt mềm đặc biệt được xếp vào đầu bảng vector ngắt như sau:

+ INT 0: Ngắt mềm do phép chia cho số 0 gây ra.

+ INT 1: Ngắt mềm để chạy từ lệnh ứng với trường hợp cờ TF = 1.

+ INT 2: Ngắt cứng do tín hiệu tích cực tại chân NMI gây ra.

+ INT 3: Ngắt mềm để đặt điểm dừng của chương trình tại một địa chỉ nào đó.

+ INT 4: (hoặc lệnh INT O): ngắt mềm ứng với trường hợp cờ tràn OF = 1.

Các kiểu ngắt khác còn lại thì được dành cho Intel và cho người sử dụng (IBM không hoàn toàn tuân thủ các quy định này khi chế tạo các máy PC/XT và PC/AT):

+ INT 5 – INT 1FH: dành riêng cho Intel trong các bộ vi xử lý cao cấp khác.

+ INT 20H – INT FFH: dành cho người sử dụng.

Các kiểu ngắt N trong INT N đều tương ứng với các địa chỉ xác định của CTCPVN mà ta có thể tra được trong *bảng các vector ngắt.* Intel quy định bảng này nằm trong RAM bắt đầu từ địa chỉ 00000H và dài 1Kbyte (vì 8086 có tất cả 256 kiểu ngắt, mỗi kiểu ngắt ứng với 1 vector ngắt, 1 vector ngắt cần 4 byte để chứa địa chỉ đầy đủ cho CS : IP của CTCPVN). Trên bảng 6.2 giới thiệu một phần bảng vector ngắt của CPU 8086.

Bảng 6.2. Bảng vector ngắt của 8086 tại 1KB RAM đầu tiên

03FEH – 03FFH	CS của CTCPVN INT FFH
03FCH – 03FDH	IP của CTCPVN INT FFH
0082H – 0083H	CS của CTCPVN INT 20H
0080H – 0081H	IP của CTCPVN INT 20H
000AH – 000BH	CS của CTCPVN INT 2
0008H – 0009H	IP của CTCPVN INT 2
0006H – 0007H	CS của CTCPVN INT 1
0004H – 0005H	IP của CTCPVN INT 1
0002H – 0003H	CS của CTCPVN INT O
0000H – 0001H	IP của CTCPVN INT O

Xử lý ưu tiên khi ngắt:

Có một vấn đề rất thực tế đặt ra là nếu tại cùng một thời điểm có nhiều yêu cầu ngắt thuộc các loại ngắt khác nhau cùng đòi hỏi CPU phục vụ thì CPU xử lý các yêu cầu ngắt đó như thế nào? Câu trả lời là CPU xử lý các yêu cầu ngắt theo thứ tự ưu tiên với nguyên tắc ngắt nào có mức ưu tiên cao nhất sẽ được CPU nhận biết và phục vụ trước.

Ngay từ khi được chế tạo (thường gọi là ngầm định), CPU 8086 có khả năng phân biệt các mức ưu tiên khác nhau cho các loại ngắt theo thứ tự từ cao xuống thấp như sau:

Mức ưu tiên:

+ Ngắt nội bộ: INT 0 (phép chia cho 0), INT N, INT 0.

+ Ngắt không che được NMI.

+ Ngắt che được INTR.

+ Ngắt để chạy từng lệnh INT 1.

Để thấy rõ hoạt động của CPU trong cơ chế ngắt ưu tiên ta có thể lấy một ví dụ cụ thể như sau:

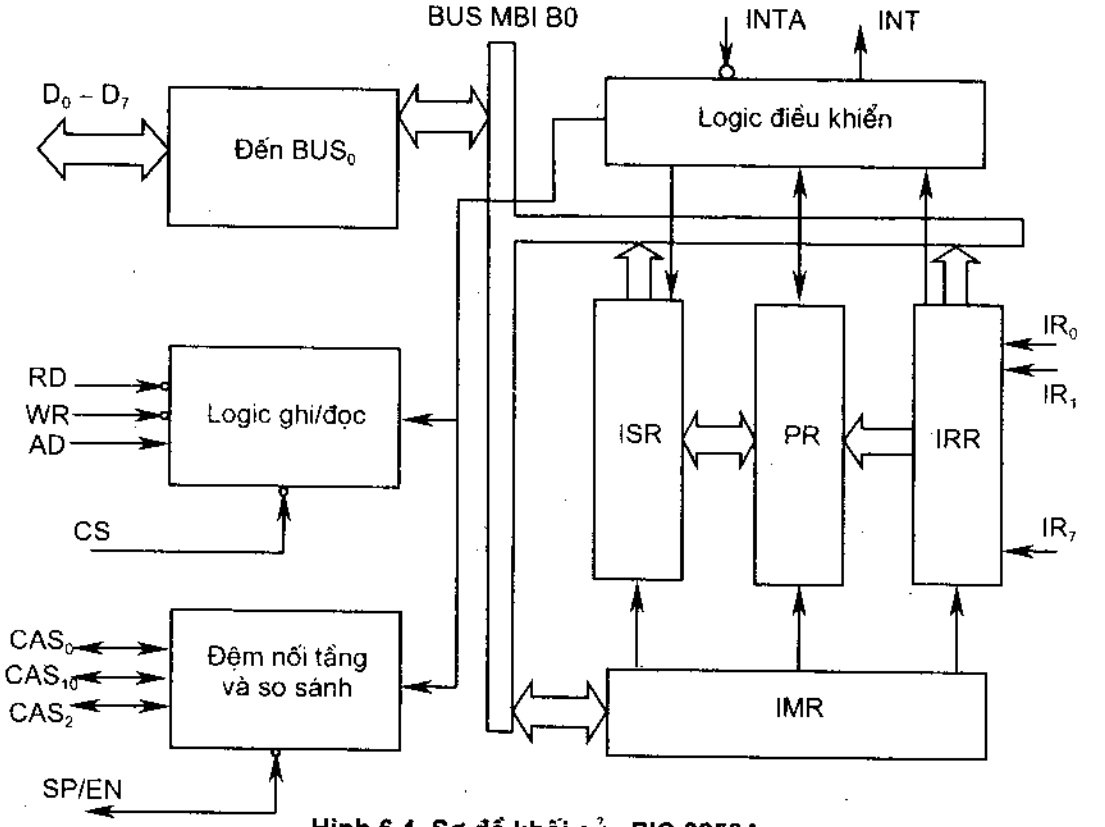
Giả thiết tại một thời điểm nào đó, trong khi CPU (ở trạng thái cho phép ngắt với cờ IF = 1) đang thực hiện phép chia và có lỗi xảy ra do số chia bằng 0, đúng vào lúc đó CPU cũng nhận được yêu cầu ngắt từ đầu vào INTR, CPU sẽ xử lý ra sao trong trường hợp này?

Theo thứ tự ưu tiên ngầm định trong việc xử lý ngắt của CPU 8086 thì INT 0 có mức ưu tiên cao hơn INTR, vì vậy đầu tiên CPU sẽ thực hiện chương trình phục vụ ngắt INT 0 để đáp ứng với lỗi đặc biệt do phép chia cho 0 gây ra và cờ IF bị xoá về 0. Yêu cầu ngắt INTR sẽ tự động bị cấm cho tới khi chương trình phục vụ ngắt INT 0 được hoàn tất và trở về IRET, cờ IF cũ được trả lại. Tiếp theo đó CPU sẽ đáp ứng yêu cầu ngắt INTR bằng cách thực hiện chương trình phục vụ ngắt dành cho INTR.

6.4. VI MẠCH XỬ LÝ NGẮT 8259A

Trong trường hợp có nhiều yêu cầu ngắt che được từ bên ngoài phải phục vụ ta thường dùng vi mạch có sẵn 8259A để giải quyết vấn đề ưu tiên. Mạch 8259A được gọi là mạch điều khiển ngắt ưu tiên (Priority Interrupt Controller – PIC). Đó là một vi mạch cỡ lớn lập trình được, có thể xử lý

trước được 8 yêu cầu ngắt với 8 mức ưu tiên khác nhau để tạo ra một yêu cầu ngắt đưa đến đầu vào INTR (yêu cầu ngắt che được) của CPU 8086. Nếu nối tầng 1 mạch 8259A chủ với 8 mạch 8259A thợ ta có thể nâng tổng số các yêu cầu ngắt với các mức ưu tiên khác nhau lên thành 64.



Hình 6.4. Sơ đồ khối của PIC 8259A

Ghi chú:

$IR_0 - IR_7$: Các yêu cầu ngắt.

IRR: Thanh ghi yêu cầu ngắt. ISR: Thanh ghi các yêu cầu ngắt đang được phục vụ. IMR: Thanh ghi mặt nạ ngắt.

PR: Bộ xử lý ưu tiên.

SP/EN: Slave Prorgam/ENable buffer (lập trình thành mạch thợ/ mở đệm bus dữ liệu).

$CAS_0 - CAS_2$: Tín hiệu nối tầng giữa các PIC 8259A với nhau.

6.4.1. Các khối chức năng chính của PIC 8259A

+ Thanh ghi IRR: ghi nhớ các yêu cầu ngắt có tại đầu vào IR_i.

+ Thanh ghi ISR: ghi nhớ các yêu cầu ngắt đang được phục vụ trong số các yêu cầu ngắt IR_i.

+ Thanh ghi IMR: ghi nhớ mặt nạ ngắt đối với các yêu cầu ngắt IR_i.

+ Logic điều khiển: khối này có nhiệm vụ gửi yêu cầu ngắt tới INTR của 8086 khi có tín hiệu tại các chân IR_i và nhận trả lời chấp nhận yêu cầu ngắt INTA từ CPU để rồi điều khiển việc đưa ra kiểu ngắt trên bus dữ liệu.

+ Đệm bus dữ liệu: dùng để phối ghép 8259A với bus dữ liệu của CPU.

+ Logic điều khiển ghi/đọc: dùng cho việc ghi các từ điều khiển và đọc các từ trạng thái của 8259A.

+ Khối đệm nối tầng và so sánh: ghi nhớ và so sánh số hiệu của các mạch 8259A có mặt trong hệ vi xử lý.

6.4.2. Các tín hiệu của 8259A

Một số tín hiệu trong mạch 8259A có tên giống như các tín hiệu tiêu chuẩn của hệ vi xử lý 8086. Ta có thể thấy rõ và hiểu được ý nghĩa của chúng ngay trên hình 6.4. Ngoài các tín hiệu này ra, còn có một số tín hiệu đặc biệt khác của 8259A đó là:

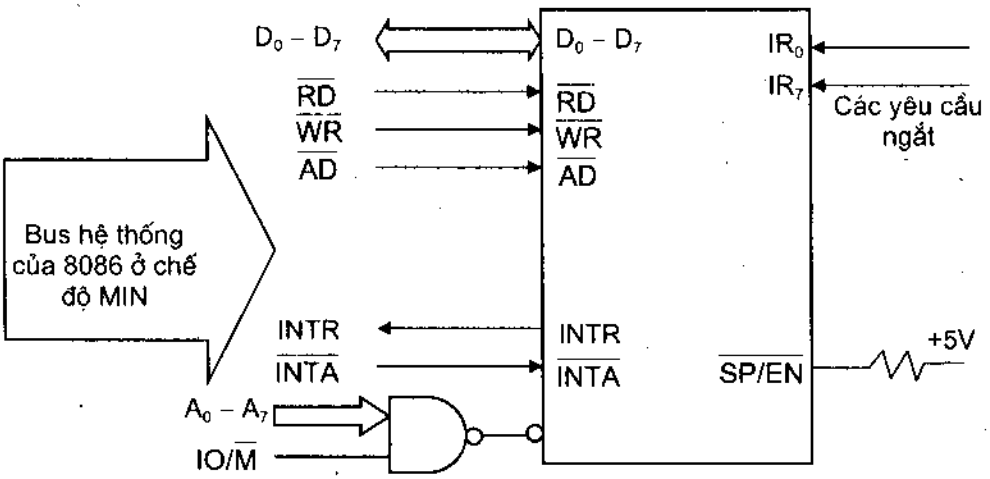
+ CAS_0 – CAS_2 [I, O]: là các đầu vào đối với các mạch 8259A thợ hoặc các đầu ra của mạch 8259A chủ dùng khi cần nối tầng để tăng thêm các yêu cầu ngắt cần xử lý.

+ SP/EN [I, O]: khi 8259A làm việc ở chế độ không có đệm bus dữ liệu thì dây là tín hiệu vào dùng lập trình để biến mạch 8259A thành mạch thợ (SP = 0) hoặc mạch chủ (SP = 1), khi 8259A làm việc trong hệ vi xử lý ở chế độ có đệm bus dữ liệu thì chân này là tín hiệu ra EN dùng mở đệm bus dữ liệu để 8086 và 8259A thông vào bus dữ liệu hệ thống, lúc này việc định nghĩa mạch 8259A là chủ hoặc thợ phải thực hiện thông qua từ điều khiển khởi đầu ICW4.

+ INT [O]: Tín hiệu yêu cầu ngắt đến chân INTR của CPU 8086.

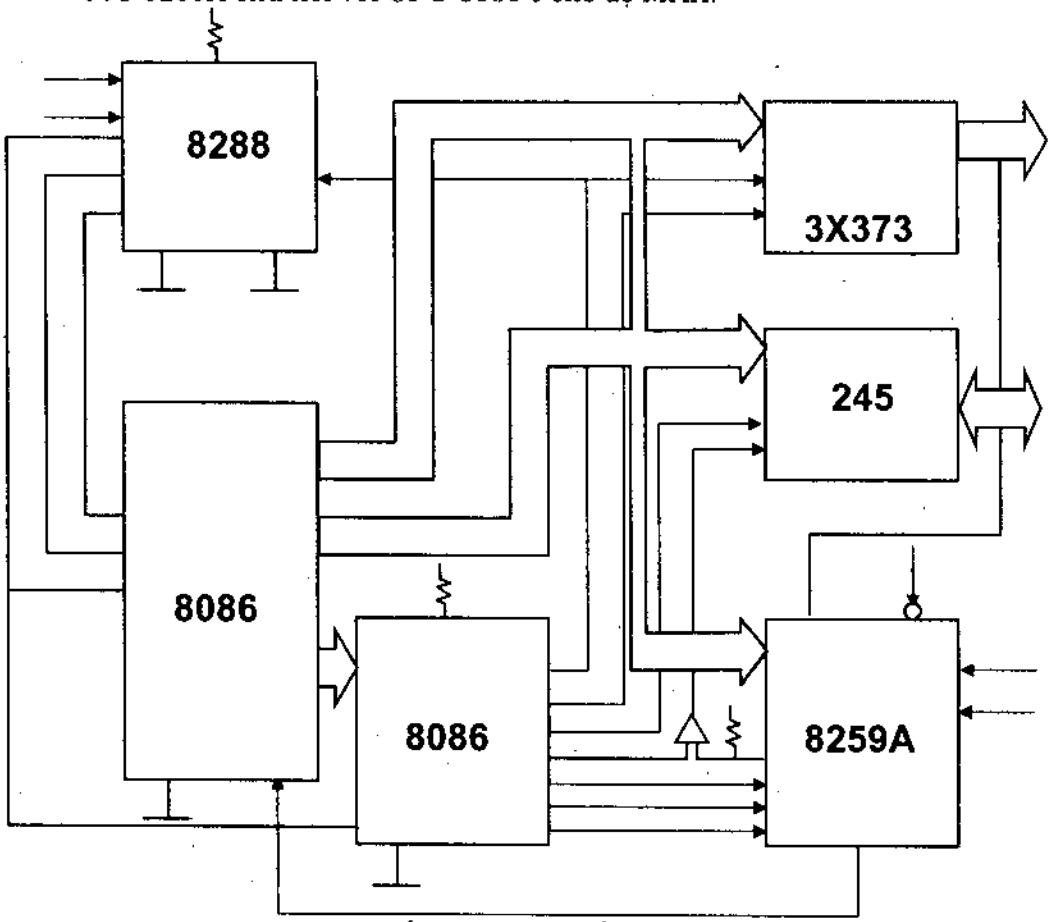
+ INTA [I]: Nối với tín hiệu báo chấp nhận ngắt INTA của CPU.

PIC 8259A chủ (ở chế độ không đệm) nối với CPU 8086 ở chế độ MIN. Trên hình 6.5 là sơ đồ nối mạch PIC 8259A làm việc độc lập (mạch chủ) với bus của CPU 8086 làm việc ở chế độ MIN.



Hình 6.5. Nối bus 8086 chế độ MIN với PIC 8259A (địa chỉ cổng FE–FFH)

PIC 8259A chủ nối với CPU 8086 ở chế độ MAX.



Hình 6.6. Nối CPU 8086 chế độ MAX với PIC 8259A

Nếu hệ vi xử lý 8086 làm việc ở chế độ MAX thường ta phải dùng mạch điều khiển bus 8288 và các đệm bus để cung cấp các tín hiệu thích hợp cho bus hệ thống. Mạch 8259A phải làm việc ở chế độ có đệm để nối được với bus hệ thống này.

Trên hình 6.6 là sơ đồ CPU 8086 làm việc ở chế độ MAX nối với PIC 8259A. Trong mạch này, ta nhận thấy tín hiệu địa chỉ cho 8259A được lấy ra từ bus hệ thống, trong khi đó tín hiệu dữ liệu của nó được nối với bus dữ liệu của vi xử lý và từ đó được thông qua các đệm để nối vào bus hệ thống.

6.4.3. Lập trình cho PIC 8259A

Để mạch PIC 8259A có thể hoạt động được theo yêu cầu, sau khi bật nguồn cấp điện PIC cần phải được lập trình bằng cách ghi vào các thanh ghi (tương đương với các cổng) bên trong nó các *từ điều khiển khởi đầu (ICW)* và tiếp sau đó là các *từ điều khiển hoạt động (OCW).*

Các từ điều khiển khởi đầu dùng để tạo nên các kiểu làm việc cơ bản cho PIC 8259A, còn các từ điều khiển hoạt động sẽ quyết định cách thức làm việc cụ thể của PIC 8259A. Từ điều khiển hoạt động sẽ được ghi khi ta muốn thay đổi hoạt động của PIC 8259A.

6.4.3.1. Các từ điều khiển khởi đầu ICW

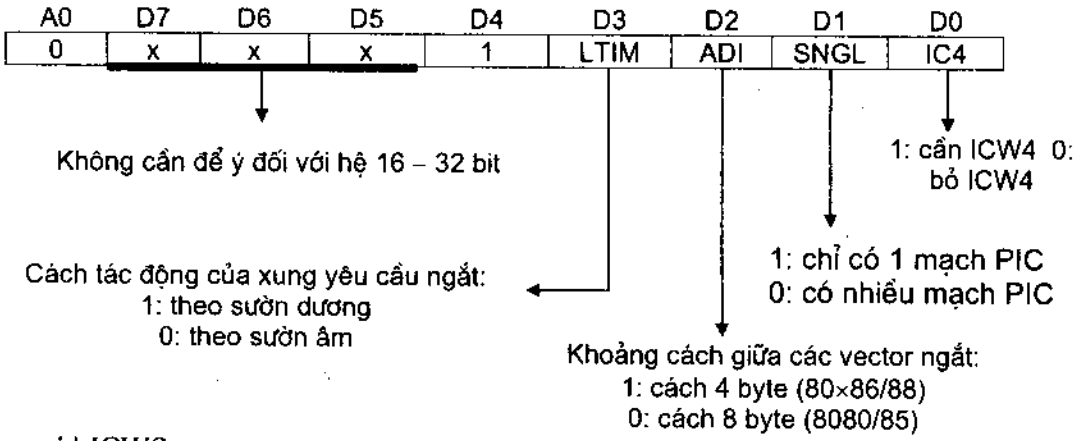
PIC 8259A có tất cả 4 từ điều khiển khởi đầu là ICW1 đến ICW4. Trong khi lập trình cho PIC 8259A không phải lúc nào ta cũng cần dùng cả 4 từ điều khiển đó. Tuỳ theo các trường hợp ứng dụng cụ thể mà có lúc ta cần ghi liên tiếp cả 4 từ điều khiển khởi đầu nhưng có lúc ta chỉ cần ghi vào đó 2 hay 3 từ là đủ.

a) ICW1

Bit D_0 của ICW1 quyết định 8259A sẽ được nối với họ vi xử lý nào. Để làm việc với hệ 16 – 32 bit (8086 hoặc họ 80×86) thì ICW1 nhất thiết phải có IC4 = 1 (tức là ta luôn cần đến ICW4), còn đối với hệ 8 bit như họ 8080/85 thì phải có IC4 = 0 (và như vậy các bit của ICW4 sẽ bị xoá về 0). Các bit còn lại của ICW1 định nghĩa cách thức tác động của xung yêu cầu ngắt (tác động theo sườn hay theo mức) tại các chân yêu cầu ngắt IR của mạch 8259A và việc bố trí các mạch 8259A khác trong hệ làm việc đơn lẻ hay theo chế độ nối tầng.

Các bit được đánh dấu x là không quan trọng và thường được lấy giá trị 0 để lập trình cho các ứng dụng sau này.

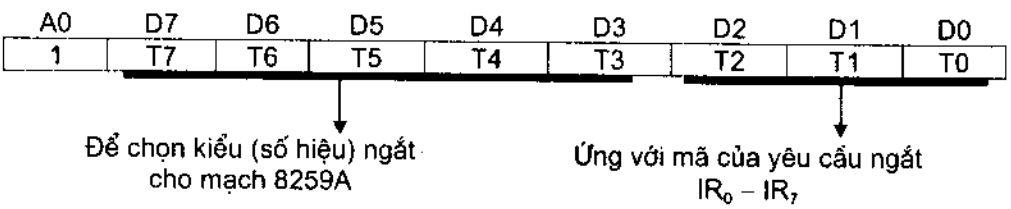
ICW1



b) ICW2

Từ điều khiển khởi đầu này cho phép chọn kiểu ngắt (số hiệu ngắt) ứng với các bit T3 – T7 cho các đầu vào yêu cầu ngắt. Các bit T0 – T2 được 8259A tự động gán giá trị tuỳ theo đầu vào yêu cầu ngắt cụ thể IR_i. Ví dụ nếu ta muốn các đầu vào của mạch 8259A có kiểu ngắt là 40H – 47H ta chỉ cần ghi giá trị 40H vào các bit T3 – T7. Nếu làm như vậy thì IR_0 sẽ có kiểu ngắt là 40H, IR_1 sẽ có kiểu ngắt là 41H.

ICW2



c) ICW3

Từ điều khiển khởi đầu này chỉ dùng đến khi bit SNGL thuộc từ điều khiển khởi đầu ICW1 có giá trị 0, nghĩa là trong hệ có các mạch 8259A làm việc ở chế độ nối tầng. Chính vì vậy tồn tại 2 loại ICW3: 1 cho mạch 8259A chủ và 1 cho mạch 8259A thợ.

ICW3 cho mạch chủ: dùng để chỉ ra đầu vào yêu cầu ngắt IR_i nào của nó có tín hiệu INT của mạch thợ nối vào.

ICW3 cho mạch thợ: dùng làm phương tiện để các mạch này được nhận biết, vì vậy từ điều khiển khởi đầu này phải chứa mã số i ứng với đầu vào IR_i của mạch chủ mà mạch thợ đã cho nối vào. Mạch thợ sẽ so sánh mã số này với mã số nhận được ở CAS2 – CAS0. Nếu bằng nhau thì số hiệu ngắt sẽ được đưa ra bus khi có INTA.

ICW3 chủ

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	S7	S6	S5	S4	S3	S2	S1	S0

1: đầu vào IRi nối với mạch 8259A thợ
0: đầu vào IRi không nối với mạch thợ

ICW3 thợ

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	ID2	ID1	ID0

Mã hoá số hiệu của mạch thợ 0 - 7 nối vào IR_0 – IR_7

d) ICW4

Từ điều khiển khởi đầu này chỉ dùng đến khi trong từ điều khiển ICW1 có bit D0 =1 (IC4).

Bit μPM cho ta khả năng chọn loại vi xử lý để làm việc với 8259A. Bit μPM = 1 cho phép các bộ vi xử lý từ 8086/88 hoặc cao hơn làm việc với 8259A.

Bit SFNM = 1 cho phép chọn *chế độ ưu tiên cố định đặc biệt*. Trong chế độ này, yêu cầu ngắt với mức ưu tiên cao nhất hiện thời từ một mạch thợ làm việc theo kiểu nối tầng sẽ được mạch chủ nhận biết ngay cả khi mạch chủ còn đang phải phục vụ một yêu cầu ngắt ở một mạch thợ khác nhưng với mức ưu tiên thấp hơn. Sau khi các yêu cầu ngắt được phục vụ xong thì chương trình phục vụ ngắt phải có lệnh kết thúc yêu cầu ngắt (EOI) đặt trước lệnh trở về (IRET) đưa đến cho mạch 8259A chủ.

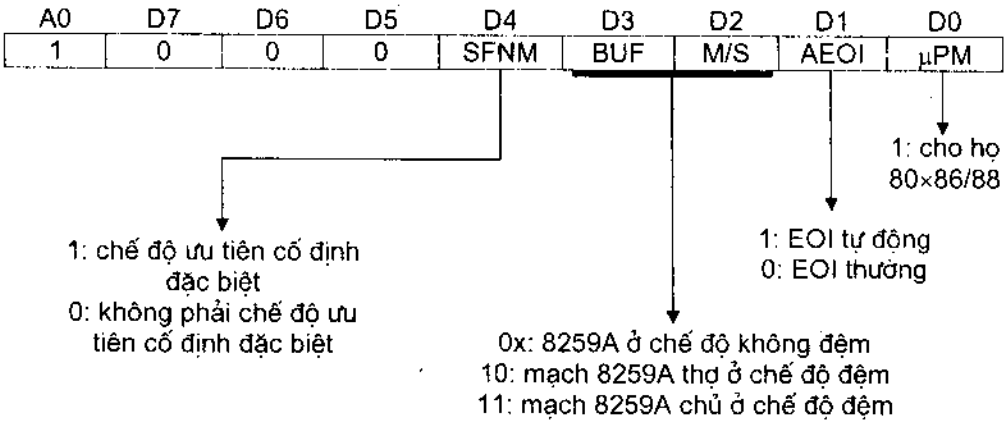
Bit SFNM = 0 thì *chế độ ưu tiên cố định* được chọn (IR_0: mức ưu tiên cao nhất, IR_7: mức ưu tiên thấp nhất). Thực ra đối với mạch 8259A không dùng đến ICW4 thì chế độ này đã được chọn như ngầm định. Trong chế độ ưu tiên cố định, tại một thời điểm chỉ có một yêu cầu ngắt i được phục vụ (bit ISR_i = 1), lúc này tất cả các yêu cầu khác với mức ưu tiên thấp hơn đều bị

cấm, tất cả các yêu cầu khác với mức ưu tiên cao hơn có thể ngắt các yêu cầu khác có mức ưu tiên thấp hơn.

Bit BUF cho phép định nghĩa mạch 8259A để làm việc với CPU trong trường hợp có đệm hoặc không có đệm nối với bus hệ thống. Khi làm việc ở chế độ có đệm (BUF = 1), bit M/S = 1/0 cho phép ta chọn mạch 8259A để làm việc ở chế độ chủ/thợ. SP/EN trở thành đầu ra cho phép mở đệm để PIC và CPU thông với bus hệ thống.

Bit AEOI = 1 cho phép chọn cách *kết thúc yêu cầu ngắt tự động*. Khi AEOI = 1 thì 8259A tự động xoá $ISR_i = 0$ khi xung INTA cuối cùng chuyển lên mức cao mà không làm thay đổi thứ tự ưu tiên. Ngược lại, khi ta chọn cách *kết thúc yêu cầu ngắt thường* (AEOI = 0) thì chương trình phục vụ ngắt phải có thêm lệnh EOI đặt trước lệnh IRET để kết thúc cho 8259A.

ICW4



6.4.3.2. Các từ điều khiển hoạt động OCW

Các từ điều khiển hoạt động OCW sẽ quyết định mạch 8259A sẽ hoạt động như thế nào sau khi nó đã được khởi đầu bằng các từ điều khiển ICW. Tất cả các từ điều khiển hoạt động sẽ được ghi vào các thanh ghi trong PIC khi A0 = 0, trừ OCW1 được ghi khi A0 = 1.

a) OCW1

OCW1 dùng để ghi giá trị của các bit mặt nạ vào thanh ghi mặt nạ ngắt IMR. Khi một bit mặt nạ nào đó của IMR được lập thì yêu cầu ngắt tương ứng với mặt nạ đó sẽ không được 8259A nhận biết nữa (bị che). Từ điều khiển này phải được đưa đến 8259A ngay sau khi ghi ra các ICW của 8259A.

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

Mặt nạ ngắt tại các yêu cầu ngắt
1: có mặt nạ
0: không có mặt nạ

b) OCW2

Các bit R, SL và EOI phối hợp với nhau cho phép chọn ra các cách thức kết thúc yêu cầu ngắt khác nhau. Một vài cách kết thúc yêu cầu ngắt còn tác động các yêu cầu ngắt được chỉ đích danh với mức ưu tiên được mã hoá bởi 3 bit $L_2L_1L_0$.

Giới thiệu các chế độ làm việc của 8259A:

– Chế độ ưu tiên cố định: (như trình bày trong phần ICW4).

– Chế độ quay mức ưu tiên tự động: ở chế độ này sau khi một yêu cầu ngắt được phục vụ xong, 8259A sẽ xoá bit tương ứng của nó trong thanh ghi ISR và gán cho đầu vào của nó mức ưu tiên thấp nhất để tạo điều kiện cho các yêu cầu ngắt khác có thời cơ được phục vụ.

– Chế độ quay mức ưu tiên chỉ đích danh: ở chế độ này ta cần chỉ rõ đầu vào IR_i nào, với i = $L_2L_1L_0$, được gán mức ưu tiên thấp nhất, đầu vào IR_{i+1} sẽ tự động được gán mức ưu tiên cao nhất.

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	R	SL	EOI	0	0	L2	L1	L0

Mã hoá mức ưu tiên ngắt bị tác động
000... mức 0
111... mức 7

Kết thúc ngắt (EOI)	0	0	1	Lệnh EOI thường
	0	1	1	Lệnh EOI chỉ đích danh (*)
Đổi mức ưu tiên tự động	1	0	1	Đổi mức ưu tiên khi có EOI thường
	1	0	0	Lập chế độ quay khi có EOI tự động
	0	0	0	Xoá chế độ quay khi có EOI tự động
Đổi mức ưu tiên đích danh	1	1	1	Đổi mức ưu tiên khi có EOI đích danh (*)
	1	1	0	Lệnh lập mức ưu tiên (*)
	0	1	0	Không làm gì

(*): dùng tổ hợp L_2 L_1 L_0

c) OCW3

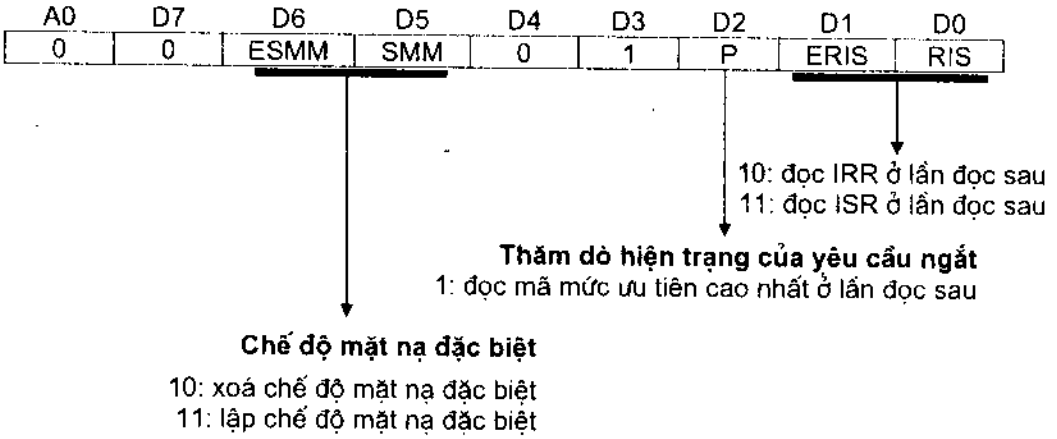
Từ điều khiển hoạt động OCW3 sau khi được ghi vào 8259A cho phép.

+ Chọn ra các thanh ghi để đọc.

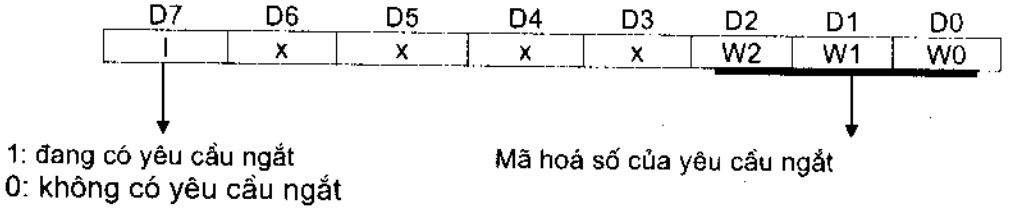
+ Thăm dò trạng thái yêu cầu ngắt bằng cách kiểm tra trạng thái của đầu vào yêu cầu ngắt IR_i với mức ưu tiên cao nhất cùng mã của đầu vào đó.

+ Thao tác với mặt nạ đặc biệt.

OCW3



Bằng việc đưa vào 8259A từ diều khiển OCW3 với bit P = 1, ta có thể dọc được trên bus dữ liệu ở lần đọc tiếp ngay sau đó từ thăm dò, trong đó có các thông tin về yêu cầu ngắt với mức ưu tiên cao nhất đang hoạt động và mã tương ứng với yêu cầu ngắt ấy theo dạng thức như sau:



Bit ESMM = 1 cho phép 8259A thao tác với chế độ mặt nạ đặc biệt, bit SMM = 1 cho phép lập chế độ mặt nạ đặc biệt. Chế độ mặt nạ đặc biệt được dùng để thay đổi thứ tự ưu tiên ngay bên trong chương trình con phục vụ ngắt. Ví dụ, trong trường hợp có một yêu cầu ngắt bị cấm (bị che bởi chương trình phục vụ ngắt với từ lệnh OCW1) mà ta lại muốn cho phép các yêu cầu ngắt với mức ưu tiên thấp hơn so với yêu cầu ngắt bị cấm đó được tác động thì ta sẽ dùng chế độ mặt nạ đặc biệt. Khi đã được lập, chế độ mặt nạ đặc

biệt sẽ tồn tại cho tới khi bị xoá bằng cách ghi vào 8259A một từ lệnh OCW3 khác với bit SMM = 0. Mặt nạ đặc biệt không ảnh hưởng tới các yêu cầu cần ngắt với mức ưu tiên cao hơn.

Cuối cùng để có cái nhìn một cách có hệ thống về hoạt động của hệ vi xử lý với CPU 8086 và PIC 8259A khi có yêu cầu ngắt, ta tóm lược hoạt động của chúng như sau:

1. Khi có yêu cầu ngắt từ thiết bị ngoại vi tác động vào một trong các chân IR của PIC, 8259A sẽ đưa INT = 1 đến chân INTR của 8086.

2. 8086 đưa ra xung INTA đầu đến 8259A.

3. 8259A dùng xung INTA đầu như là thông báo để nó hoàn tất các xử lý nội bộ cần thiết, kể cả xử lý ưu tiên nếu như có nhiều yêu cầu ngắt cùng xảy ra.

4. 8086 đưa ra xung INTA thứ hai đến 8259A.

5. Xung INTA thứ hai khiến 8259A đưa ra bus dữ liệu 1 byte chứa thông tin về số hiệu ngắt của yêu cầu ngắt vừa được nhận biết.

6. 8086 dùng số hiệu ngắt để tính ra địa chỉ của vector ngắt tương ứng.

7. 8086 cất FR, xoá các cờ IF và TF và cất địa chỉ trở về CS : IP vào ngăn xếp.

8. 8086 lấy địa chỉ CS : IP của chương trình phục vụ ngắt từ bảng vector ngắt và thực hiện chương trình đó.

6.4.4. Ví dụ lập trình với 8259A

Lập trình cho 8259A để làm việc với CPU 8086 ở chế độ chủ (đơn lẻ), trong hệ có đệm bus, chế độ ưu tiên cố định và với EOI thường, IR kích theo mức, tín hiệu IR_0 được gán số hiệu ngắt là 50H.

Giải:

ICW1 = 00011011 = 1BH

D0 = 1 cần thêm ICW4

D1 = 1 làm việc đơn lẻ, không cần ICW3

D2 = 0 làm việc với hệ 8086/88

D3 = 1 đầu vào IR ăn theo mức

D4 = 1 bắt buộc với ICW1

D5 = D6 = D7 = 0 gán bằng 0 cho hệ 8086/88

ICW2 = 01010000 = 50H

Vì các bit T_7 – T_3 của ICW2 phải mã hoá trị số 50H để IR_0 – IR_7 được mã hoá tiếp bởi các bit T_2 – T_0 = 000.

ICW3 không cần đến

ICW4 = 00001101 = 0DH

D0 = 1 làm việc với hệ 8086/88

D1 = 0 EOI thường (phải có EOI trước IRET)

D3D2 = 11 làm việc ở chế độ chủ trong hệ có đệm bus

D4 = 0 chế độ ưu tiên cố định

D5 = D6 = D7 = 0 luôn bằng 0 cho ICW4.

Dưới đây là đoạn chương trình con khởi tạo cho 8259A với chế độ hoạt động như trên:

```
INIT PROC NEAR
    ;ICW1
    MOV AL,00010011B
    OUT INTA,AL
    ;ICW2 interrupt vector
MOV AL,50H
    OUT INTA2,AL
    ;ICW4
    MOV AL,00000001B
OUT INTA2,AL
    ;interrupt mask , OCW1
    MOV AL,11111011B
    OUT INTA2,AL
    RET
INIT ENDP
```

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 6

1. Tìm địa-chỉ của chương trình con phục vụ ngắt tương ứng với vector ngắt là 20H.

2. Tìm địa chỉ của chương trình con phục vụ ngắt khi CPU thực hiện lệnh INT 3.

3. Đoạn chương trình sau gây ra ngắt gì?

```
XOR AL,AL
MOV BL,8
MOV AL,8
DIV BL
```

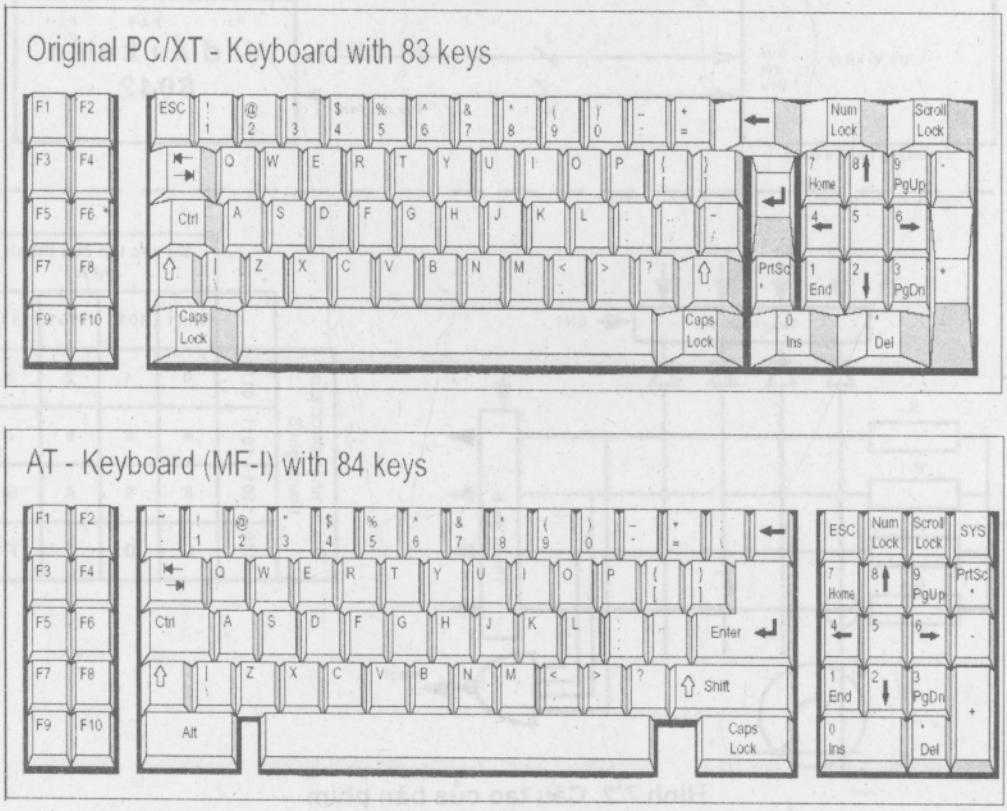
4. Đoạn chương trình sau gây ra ngắt gì?

```
XOR AL,AL
MOV BL,AL
MOV AL,8
DIV BL
```

Chương 7. CÁC THIẾT BỊ NGOẠI VI THÔNG DỤNG

7.1. BÀN PHÍM

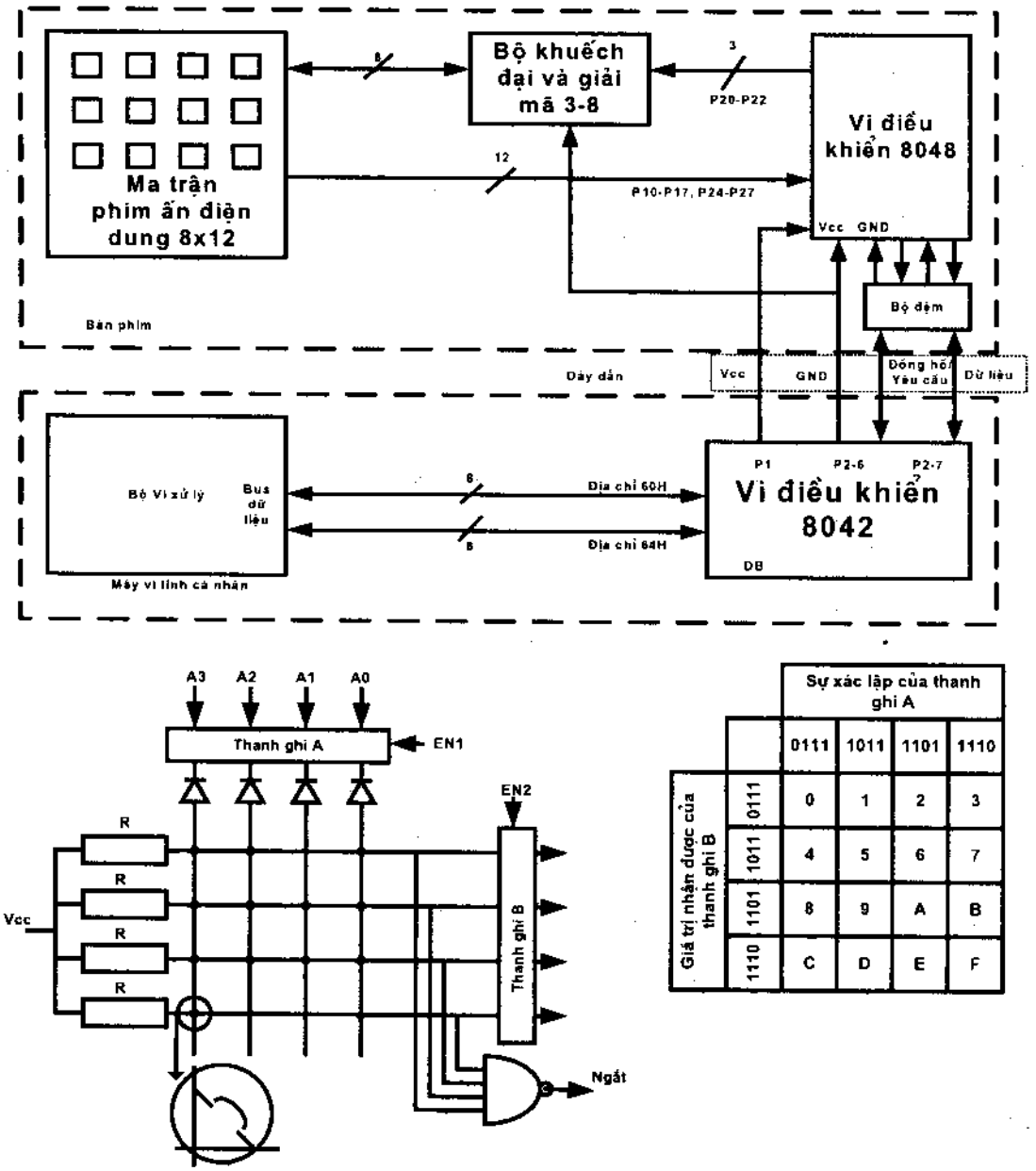
Có thể nói bàn phím và chuột là hai thiết bị đưa tin vào phổ biến nhất. Nếu năm 1963, chuột mới được đưa ra để phục vụ cho việc tăng năng suất và nâng cao hiệu quả của việc dùng máy tính thì bàn phím đã được sinh ra trước đó gần 100 năm, cụ thể là năm 1870 và được đặt tên là QWERTY, theo những chữ cái từ bên trái ở dòng đầu tiên của bàn phím này. Ngoài bàn phím QWERTY còn một loại bàn phím nữa là bàn phím Dvorak do ông August Dvorak và ông William Deay thiết kế vào những năm 1930. Loại bàn phím Dvorak này gõ nhanh hơn do ít phải dịch chuyển tay hơn so với loại bàn phím QWERTY. Tuy nhiên bàn phím QWERTY lại phổ biến hơn so với bàn phím Dvorak do nó ra đời trước nên khi bàn phím Dvorak ra đời thì trên thế giới đã có quá nhiều người quen sử dụng QWERTY.



Hình 7.1. Bàn phím QWERTY cho máy PC/XT và AT

7.1.1. Cấu tạo của bàn phím

Bàn phím được cấu tạo từ hai phần chính là các phím nhấn và bộ tạo mã quét.



Giá trị nhận được của thanh ghi B	Sự xác lập của thanh ghi A: 0111	1011	1101	1110
0111	0	1	2	3
1011	4	5	6	7
1101	8	9	A	B
1110	C	D	E	F

Hình 7.2. Cấu tạo của bàn phím

Các phím nhấn có nhiệm vụ tạo ra sự thay đổi có thể về điện trở, về điện dung hoặc về dòng điện để bộ phận tạo mã quét nhận biết mỗi khi người sử

dụng tác động lên phím. Bộ phận tạo mã quét có nhiệm vụ nhận biết vị trí của phím được tác động, tạo ra mã quét và chuyển về cho máy tính. Hình 7.2 minh hoạ cấu tạo một bàn phím gồm 12x8 phím.

7.1.2. Hoạt động của bàn phím

Vi điều khiển 8048 phát ra 3 bit, giải mã thành 8 bit quét ma trận bàn phím, ma trận phím đưa về cho 8048 12 bit dò. Khi một phím được ấn, 8048 truyền mã yêu cầu cho 8042 để 8042 phát ra yêu cầu ngắt về bộ vi xử lý của máy tính. Đồng thời 8048 căn cứ vào vị trí của phím được nhấn, 8048 sẽ tạo mã quét, mã quét của phím được nhấn để truyền cho 8042. Mã quét này gồm 1 bit trong đó có 1 bit start, 8 bit dữ liệu, 1 bit kiểm tra chẵn/lẻ và 1 bit stop.

Khi 8042 nhận được mã quét từ 8048, 8042 phát ra yêu cầu ngắt IRQ1 gửi tới bộ xử lý ngắt 8259A, 8259A xử lý và truyền cho bộ vi xử lý của máy tính số hiệu ngắt 09H (ngắt BIOS). Chương trình xử lý ngắt sẽ chuyển mã quét thành mã ASCII và đặt vào bộ đệm bàn phím ở trên bộ nhớ chính của máy tính. Các hàm của DOS và BIOS thường lấy mã quét và mã ASCII từ bộ đệm này.

Cả hai vi điều khiển 8048 và 8042 đều có bộ nhớ ROM để chứa các chương trình điều khiển và RAM để chứa các tham số của chương trình. Các chương trình trong các bộ vi điều khiển này cho phép:

7.1.2.1. Xử lý sự kiện ấn phím

Khi ấn một, hai hoặc ba phím đồng thời vi điều khiển của bàn phím tạo ra các mã quét. Các mã này được vi xử lý của máy tính đọc vào để gây ra các ngắt chương trình INT 16H cho bàn phím và thực hiện các chương trình con của BIOS với các hàm chức năng ghi vào AH.

Kết quả thực hiện một số chức năng khi nhận được mã của tổ hợp các phím:

- Ctrl – Alt – Del: Khởi động nóng.

- Ctrl – NumLock: Treo tới khi một phím khác được đánh.

- Tự động lặp phím nếu ấn lâu quá 0,5s, nó sẽ tạo ra động tác lặp phím 10 lần trong 1s.

- Alt – Phím chữ số: Đưa ra mã ASCII và trực tiếp bằng cách đánh chữ số là mã ASCII (dạng cơ số 10)...

7.1.2.2. Tạo ra mã ASCII

Bảng mã được đặt trong một vùng nhớ của ROM-BIOS. Máy vi tính dùng lệnh so sánh để tìm các chữ tương ứng trong bảng mã khi nhận được mã hàng và cột của ma trận bàn phím (tức mã quét). Dựa vào đây người ta có thể cài đặt lại tương ứng các mã quét cho các ký tự của các nước khác không dùng tiếng Anh. Đó chính là công việc cài đặt một bộ font chữ.

7.2. CHUỘT

Chuột được Douglas Englebart phát minh ra vào năm 1964, khi ông đang làm việc ở Viện Nghiên cứu Stanford và Trường Đại học Stanford đã bảo trợ cho ý tưởng của ông. Lúc đầu, chuột được gọi là bộ định vị X-Y cho hệ thống màn hình. Sau này qua nhiều lần phát triển, chuột đã trở thành một thiết bị ngoại vi không thể thiếu trong hệ thống máy tính.

Hiện nay người ta sử dụng hai loại chuột chính là chuột cơ và chuột quang.

7.2.1. Chuột cơ

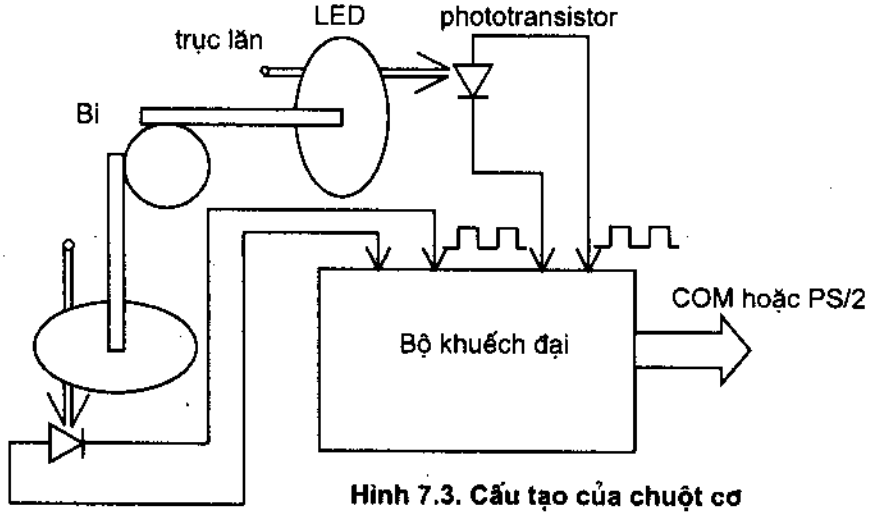
Chuột cơ được cấu tạo từ các thành phần:

+ Phần cơ gồm các phím nhấn và một quả bi bằng thép có phủ cao su để tăng ma sát. Quả bi này tỳ lên hai trục vuông góc với nhau theo toạ độ x, y.

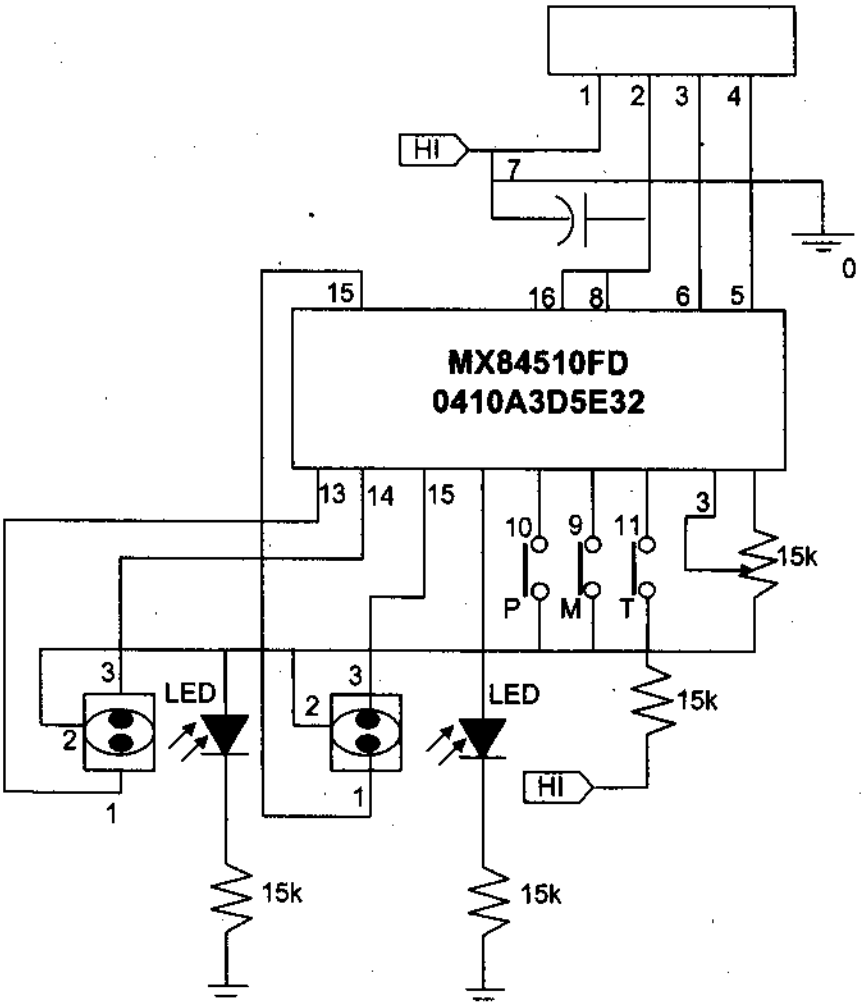
+ Phần điện tử bao gồm:

– Hai bộ cảm biến quang, mỗi bộ cảm biến gồm một diode phát và một phototransistor thu.

– Một bộ khuếch đại tín hiệu từ hai cảm biến quang truyền về máy tính để xử lý.



Hình 7.3. Cấu tạo của chuột cơ



Hình 7.4. Sơ đồ nguyên lý mạch điện tử trong chuột cơ

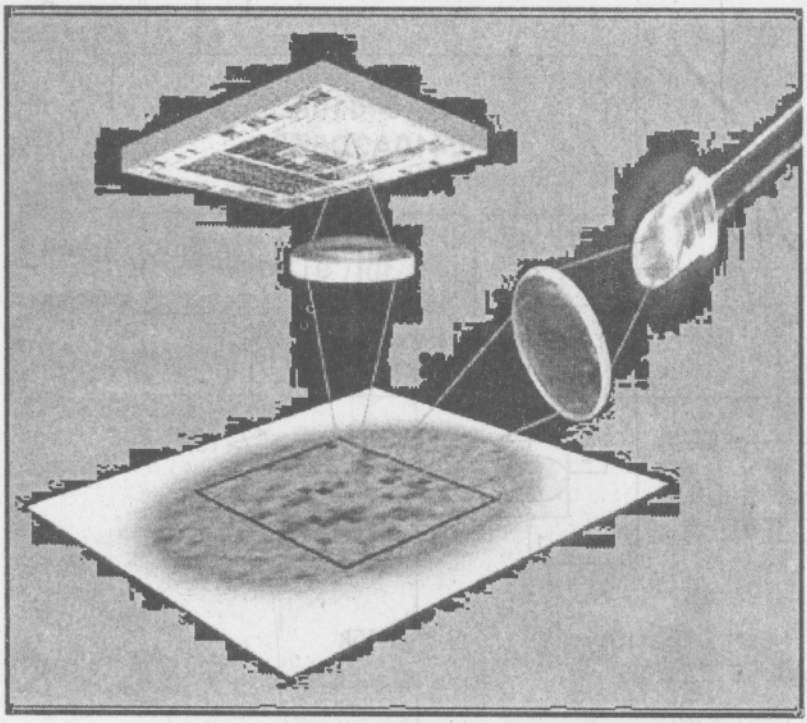
Nguyên lý hoạt động của chuột cơ là chuyển từ sự di chuyển của bi (cơ) sang sự chiếu sáng/che khuất (quang) và từ đó chuyển thành các dãy xung (điện).

Có 2 phototransistor trong mỗi bộ cảm biến quang, 2 phototransistor này được đặt ở vị trí khác nhau nên khi trục được bi tỳ lên quay thuận thì xung thu được từ phototransistor thứ nhất sẽ sớm pha hơn xung thu được từ phototransistor thứ hai và ngược lại, từ đó nhận biết được chuột đang đi lên hay đi xuống. Cũng tương tự với bộ cảm biến quang thứ hai để phát hiện chuột đang đi sang trái hay phải. Kết hợp cả 2 cảm biến này sẽ thu được kết quả là chuột đang di chuyển theo toạ độ nào trên mặt phẳng (x, y).

7.2.2. Chuột quang

Cấu tạo của chuột quang gồm:

- Hệ thống quang (optical system).
- Một chipset.
- Vỏ (case) và hệ thống các phím.



Hình 7.5. Hệ thống quang trong chuột quang

Hệ thống quang (hình 7.5) bao gồm:

- Một cảm biến quang.
- Thấu kính (LENS) được thiết kế đặc biệt để dẫn hướng ánh sáng từ LED chiếu sáng bề mặt rồi phản xạ lên trên cảm biến. Thấu kính được làm bằng plastic đặc biệt.
- Một diode phát ánh sáng đỏ (LED).

Cảm biến quang gồm ba khối chức năng: một hệ thống dọc ảnh (image reading system), một bộ xử lý tín hiệu số, một giao tiếp truyền dữ liệu nối tiếp (serial interface of data transfer).

Từ góc độ xem xét cấu trúc, một cảm biến quang là một vi mạch, ở phía dưới vi mạch có một vật kính rất nhỏ là nơi cho ánh sáng phản xạ từ bề mặt hội tụ vào trong vi mạch để xử lý. Phía trong vật kính là một camera CMOS đơn sắc có nhiệm vụ chụp những ảnh của một vùng bề mặt hình vuông diện tích cỡ một milimet vuông (diện tích này tùy thuộc tham số kỹ thuật của cảm biến).

Bức ảnh camera CMOS thu được thường được gọi là frame. Frame của bề mặt được chia thành những phần nhỏ bằng nhau (gọi là quadrate). Hai frame được chụp khi chuột di chuyển. Với mỗi phần nhỏ đó, giá trị trung bình của độ sáng được tính. Giá trị này có thể thay đổi từ 0 đến 63 (các cảm biến khác nhau có lượng giá trị để mã hoá cho độ sáng của các phần nhỏ là khác nhau), ở đó 0 tương ứng với phần tối đen nhất và 63 ứng với phần nhỏ sáng trắng nhất. Như vậy, ảnh lắp ghép bao gồm nhiều quadrate có độ sáng khác nhau. Một quadrate như thế gọi là một pixel. Độ phân giải của chuột quang được xác định bằng số pixel trên một inch, mỗi inch trên bề mặt (không phải trên ảnh) và được tính bằng đơn vị là cpi (counts per inch) thay cho dpi (dots per inch) như chuột thông thường.

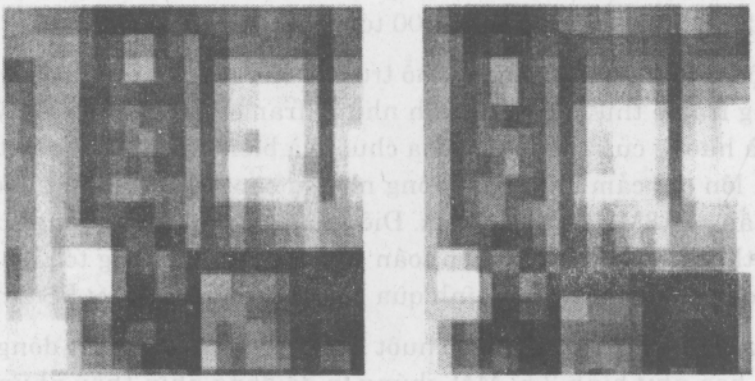
Cảm biến chỉ chụp phần nhỏ của bề mặt trong khi con trỏ màn hình phải di chuyển trơn tru và không bị trì hoãn. Để đạt được mục đích này, bề mặt được chụp với tốc độ từ 1500 tới 2300 ảnh trên một giây.

Bộ xử lý tín hiệu số với sự hỗ trợ của của một giải thuật đặc biệt sẽ xử lý những frame thu được. So sánh những frame thu được bộ xử lý xác định độ lớn và hướng của sự đổi chỗ của chuột và biến đổi dữ liệu này thành toạ độ. Phần lớn các cảm biến hoạt động nhờ sự cấp xung của một dao động thạch anh tần số 18MHz hay 24MHz. Điều này giải thích cho công suất của bộ xử lý số thực hiện 18 triệu phép toán trên giây. Cuối cùng toạ độ đã được tính toán được truyền tới máy tính qua giao tiếp nối tiếp như PS/2 hoặc USB.

Nguyên lý hoạt động của chuột quang: Nguyên lý hoạt động cơ bản được minh hoạ như hình 7.5. Mắt chúng ta dễ dàng nhìn thấy những vị trí khác nhau trên một bề mặt vật chất có cấu trúc (gồ ghề lớn) khác nhau. Do cường độ và năng lượng ánh sáng ở những vị trí có cấu trúc bề mặt khác nhau phản xạ, hội tụ vào võng mạc của mắt là khác nhau. Với một bề mặt vật chất nhẵn bóng, mắt thường chúng ta không thể nhìn thấy những chi tiết nhỏ gồ ghề của nó, nhưng dưới kính hiển vi chúng ta sẽ thấy cấu trúc lổm chổm của bề mặt. Cấu trúc bề mặt lổm chổm rất nhỏ này được chuột quang dùng để tạo ra (bằng phương pháp quang học tinh vi và công nghệ CMOS) một ảnh bề mặt gồm những diểm có độ sáng ứng với cường độ và năng lượng phản xạ của các điểm bề mặt tương ứng.

Một diode phát ánh sáng (LED) làm sáng bề mặt phía dưới đáy của chuột. Ánh sáng từ LED phản ánh những đặc tính kết cấu rất nhỏ (chỉ nhìn thấy dưới kính hiển vi) của bề mặt ra không gian. Một thấu kính bằng nhựa hội tụ ánh sáng được phản xạ từ những điểm rất nhỏ, gần nhau vào cảm biến hình thành một ảnh trên một cảm biến. Nếu chúng ta nhìn bức ảnh, nó sẽ là bức ảnh trắng đen của một phần nhỏ xíu của bề mặt. Như minh hoạ trong hình 7.6, bức ảnh nhỏ xíu này gồm nhiều điểm ảnh bằng nhau nhưng có cường độ sáng hoàn toàn khác nhau nằm giữa độ sáng của màu tối đen và màu trắng sáng, các điểm ảnh có độ sáng khác nhau này là do cấu trúc hiển vi của bề mặt khác nhau tại các điểm hiển vi khác nhau. Cảm biến liên tục thu những bức ảnh khi chuột di chuyển. Cảm biến thu những bức ảnh rất nhanh – cỡ 1500 ảnh trên giây hay nhanh hơn đủ để cho những ảnh liên tiếp trùng khớp (giống nhau) một phần. Những ảnh sau đó được gửi đến Optical Navigation Engine (tạm dịch phương tiện dẫn đường quang) để xử lý.

Nhiệm vụ của Optical Navigation Engine là nhận dạng những cấu trúc, đặc điểm khác nhau giữa những ảnh thu được và theo dấu sự di động của chúng. Hình 7.6 minh hoạ cách làm này:



Ảnh A Ảnh B

Hình 7.6. Mô tả 2 bức ảnh chụp liên tiếp

Optical Navigation Engine nhận dạng những đặc điểm chung trong các ảnh liên tiếp để xác định hướng và lượng di chuyển. Ảnh B được chụp trong khi chuột đang di chuyển, một thời gian ngắn sau khi chụp ảnh A. Hình B giống như hình A nhiều điểm, dễ thấy hình B là hình A mà được dịch xuống và về phía trái.

Hai ảnh được bắt liên tiếp khi chuột được quét sang bên phải và đi lên. Nhiều chỗ trực quan giống nhau có thể được nhận ra dễ dàng trong hai ảnh. Thông qua giải thuật xử lý ảnh, Opticcal Navigation Engine nhận dạng những nét chung giữa hai ảnh này và xác định khoảng cách giữa chúng (khoảng cách không gian giữa điểm chụp được ảnh A và điểm chụp được ảnh B). Thông tin này sau đó được chuyển đổi thành toạ độ di chuyển X (theo phương ngang) và Y (theo phương thẳng đứng) để biểu thị sự di chuyển của chuột. Vị trí con trỏ chuột được định vị bằng cách kết hợp hai giá trị X và Y này.

Bằng trực quan, xem xét kỹ ảnh chụp lúc A và B ta thấy rõ ràng chuột đã di chuyển qua phải và lên trên. Đó là định tính, thực tế giải thuật xử lý ảnh rất tinh vi, được cài đặt trong IC cảm biến sẽ xác định chính xác hướng và khoảng cách di chuyển. Chúng ta sẽ xét các IC cảm biến trong mục sau, nó gồm nhiều bộ phận có chức năng khác nhau.

7.3. CÁC THIẾT BỊ ĐỌC (VISION): MÁY QUÉT, CHỤP,...

Các thiết bị đọc lợi dụng sự phản xạ ánh sáng ở một vật khi chiếu một chùm sáng vào nó để đọc dữ liệu hình ảnh. Hình dáng và độ đậm nhạt của chùm tia phản xạ được một bộ cảm ứng quang học (tế bào quang điện, màn quang điện) biến thành tín hiệu điện để đưa vào máy tính sau khi biến đổi từ dạng tương tự sang dạng số. Các thiết bị đọc thông dụng là máy quét, máy đọc quang, thiết bị nhìn.

7.3.1. Bộ đọc quét

Gồm một nguồn sáng (neon) để chiếu vào tài liệu cần đọc, ánh sáng phản xạ được hội tụ rồi chiếu lên một thanh của các tế bào nhạy ánh sáng CCD. Nguồn sáng và thanh CCD thường xuyên chuyển động quét diện tích tài liệu hoặc chúng đứng yên còn tài liệu chuyển động qua khe hẹp chiếu sáng.

Nếu chiếu tài liệu bằng ba chùm sáng đỏ, lam, lục và đọc tia phản xạ tương ứng bằng ba thanh CCD ta có thể thu được cả màu sắc của tài liệu cần đọc. Có thể chỉ dùng nguồn sáng ba lần liên tiếp nên tài liệu đọc với 3 màu lọc sắc đỏ, xanh và lục, để có ba chùm tia khác nhau. Một điểm màu được lấy mẫu trên 6, 8 hay 11 bit. Với màu 8 bit có thể có 256 sắc thái trên 65000 màu cơ bản và hơn 65000 màu cơ bản và hơn 16 triệu màu nhuộm (leinteo).

7.3.2. Bộ đọc quang và bộ đọc từ

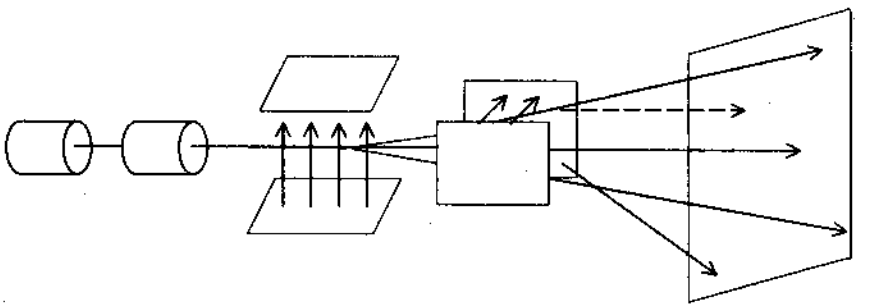
Bộ đọc quang cho phép nhận biết các chữ đã chuẩn hoá nhờ hai chùm sáng song song ở hai vùng riêng của vết mực. Sự có mặt của mực làm thay đổi cường độ sáng của chùm tia phản xạ. Bộ đọc quang được dùng trong các máy đọc bìa đục lỗ, băng đục lỗ để đọc các bit 1 của tin với 8 bit tin và một bit đánh dấu.

Biến thể của bộ đọc quang là bộ đọc từ, bộ đọc từ đọc chữ hay hình ảnh được in hoặc vẽ bằng loại mực từ tính CMC7 với tốc độ cao, thiết bị đọc đơn giản và rẻ tiền hơn đọc bằng phương pháp quang học.

7.4. MÀN HÌNH

Màn hình là thiết bị chuẩn đưa tin ra hiển thị thuận tiện và kinh tế nhất. Cùng với bàn phím (thiết bị đưa tin vào) làm thành một thiết bị đầu cuối (Terminal) là thiết bị ngoài tối thiểu của máy tính. Tùy cấu trúc có các màn hình điện tử (ống tia âm cực CRT) đen trắng, màu và màn hình Plasma, màn hình LCD. Thông tin trên màn hình có thể là dữ liệu – chữ – hình vẽ (đồ hoạ – Graphics) được hiển thị theo các phương pháp khác nhau.

Phương pháp hiện ảnh trên màn hình của monitor máy tính cũng giống như trong máy thu hình thông thường. Hình bên dưới minh hoạ việc hiện ảnh trên màn hình kiểu ống phóng tia âm cực CRT (Cathode Ray Tube).



Hình 7.7. Cấu tạo ống hình CRT

Các điện tử phát xạ từ cathode trong ống được hội tụ thành 1 chùm tia, sau đó được tăng tốc và được làm lệch hướng chuyển động bởi các bộ phận lái tia. Tia này sẽ đập vào màn hình có phủ chất huỳnh quang để tạo thành 1 điểm sáng gọi là 1 điểm ảnh.

Do hiện tượng lưu ảnh trong võng mạc của mắt người nên khi tia điện tử được quét rất nhanh theo chiều ngang từ trái sang phải sẽ tạo nên một vệt

sáng ngang được gọi là dòng quét. Đến cuối một dòng, nó được quét ngược trở về bên trái để quét tiếp dòng thứ hai bên dưới, v.v... Quá trình quét các dòng được dịch dần từ trên xuống dưới cho đến hết chiều dọc của màn hình được gọi là quét dọc.

Độ chói (sáng/ tối) được quyết định bởi cường độ chùm tia đập vào màn hình huỳnh quang và 1 điểm màu tự nhiên được hiện nhờ sự trộn lẫn của 3 màu: đỏ, xanh dương, xanh lá cây theo một tỉ lệ nào đó. Ba màu này được hiện nhờ 3 tia điện tử cùng bắn vào 3 điểm trên màn hình kề cận nhau, mỗi điểm được phủ chất huỳnh quang phát ra các màu tương ứng. 3 chùm tia điện tử đó được phát ra bởi 3 súng điện tử là 3 cathode được xếp đặt bên trong CRT một cách cẩn thận. Có 2 kiểu quét tia điện tử:

– Quét xen kẽ (interlaced): các dòng lẻ được quét trước cho đến hết màn hình theo chiều dọc, gọi là mành lẻ; sau đó các dòng chẵn tạo nên mành chẵn được quét sau. Phương pháp này có ưu điểm là thu hẹp được dải tần số làm việc của thiết bị nhưng có nhược điểm là hình ảnh bị nhấp nháy.

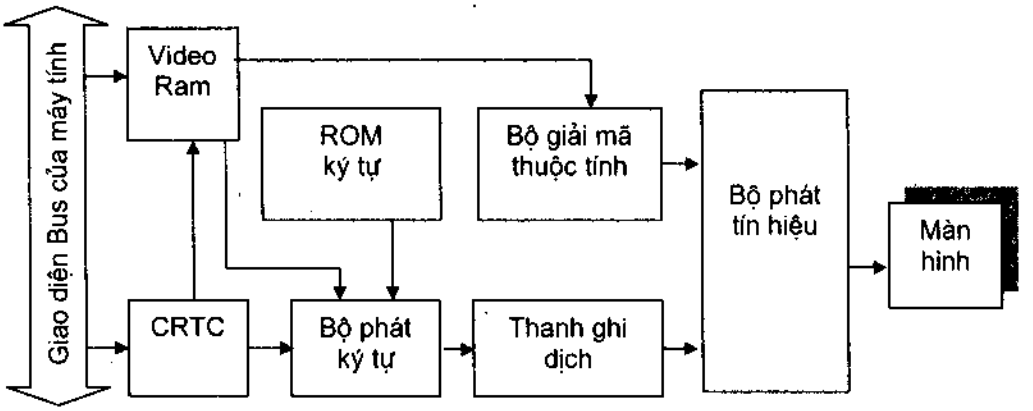
– Quét không xen kẽ (non-interlaced): các dòng quét được thực hiện tuần tự. Ưu điểm là hình ảnh được điều chỉnh chính xác và ổn định nhưng thiết kế mạch điện sẽ khó hơn vì phải giải quyết vấn đề tăng dải tần làm việc.

Hiện nay còn có các monitor dùng màn hình tinh thể lỏng LCD hoặc ống chứa khí được hoạt động theo nguyên lý tương tự như trên nhưng không có tia điện tử quét nên thay vì các điểm ảnh riêng biệt là các phần tử phát sáng được định địa chỉ một cách tuần tự. Do vậy, trên các monitor này hình ảnh cũng được phát ra từng dòng một. Quá trình quét ngược cũng không còn nữa vì ở đây đơn giản chỉ việc thay đổi địa chỉ về phần tử đầu dòng tiếp theo.

Card giao tiếp đồ hoạ:

Để hiện các hình ảnh, ký tự, hay hình vẽ trên màn hình, PC phải thông qua mạch ghép nối màn hình (graphics adapter). Board mạch này thường được cắm trên khe cắm mở rộng của PC. Sơ đồ khối như hình 7.8.

Phần trung tâm là chip điều khiển ống hình CRTC (Cathode Ray Tube Controller). CPU xâm nhập RAM Video qua mạch ghép nối bus để ghi thông tin xác định ký tự hay hình vẽ cần hiển thị. CRTC liên tục phát ra các địa chỉ để RAM video đọc các ký tự trong đó và truyền chúng tới máy phát ký tự (character generator).



Hình 7.8. Sơ đồ khối của bản mạch ghép nối màn hình

Trong chế độ văn bản (text mode), các ký tự được xác định bởi mã ASCII, trong đó có cả các thông tin về thuộc tính của ký tự, ví dụ ký tự được hiện theo cách nhấp nháy hay đảo màu đen trắng... ROM ký tự (character ROM) lưu trữ các hình mẫu điểm ảnh của các ký tự tương ứng để máy phát ký tự biến đổi các mã ký tự đó thành một chuỗi các bit điểm ảnh (pixel bit) và chuyển chúng tới thanh ghi dịch (shift register). Máy phát tín hiệu sẽ sử dụng các bit điểm ảnh này cùng với các thông tin thuộc tính từ RAM video và các tín hiệu đồng bộ từ CRTC để phát ra các tín hiệu cần thiết cho monitor.

Trong chế độ đồ hoạ (graphics mode), thông tin trong RAM video được sử dụng trực tiếp cho việc phát ra các ký tự. Lúc này các thông tin về thuộc tính cũng không cần nữa. Chỉ từ các giá trị bit trong thanh ghi dịch, máy phát tín hiệu sẽ phát các tín hiệu về độ sáng và màu cho monitor.

7.5. MÁY IN

Máy in là thiết bị đưa tin ra và lưu trữ trên giấy. Máy in thay thế máy chữ vì nó có khả năng xử lý các ký tự bằng phần mềm. Các kỹ thuật mới về cơ khí, điện tử, quang học đã đem lại nhiều chủng loại máy in có kích thước nhỏ gọn, in nhanh và hình ảnh đẹp.

Có thể phân loại máy in theo các tiêu chuẩn sau:

- Theo kỹ thuật in:

+ In tiếp xúc: chữ được hiện khi gõ vào một dải băng được tẩm mực.

+ In không tiếp xúc: chữ in hiện không cần va đập cơ lên băng tẩm mực trên giấy.

– Theo chế độ in ký tự:

+ Ký tự có sẵn: chữ được tạo khi có một gõ trên máy.

+ In ma trận: Chữ được tạo bởi một ma trận các kim gõ thành các điểm.

– Theo chế độ in văn bản:

+ Chế độ chữ: những chữ tạo nên văn bản được tạo từng chữ một.

+ Chế độ in dòng: thiết bị in phủ tất cả dòng của văn bản, mỗi dòng chỉ được in một lần.

– Theo chế độ truyền giấy:

+ Ma sát giấy được kéo bởi lực ma sát lên mặt giấy bằng một bánh xe cao su.

+ Kéo giấy bởi bánh xe răng cưa lên các dải đục lỗ của giấy.

7.5.1. Máy in có bộ chữ đúc sẵn (máy in dập)

Máy in có bộ chữ đúc sẵn là loại máy cổ xưa nhất, nó hoạt động giống như một máy đánh chữ, nghĩa là thanh kim loại hoặc nhựa có ký tự nổi trên đó đập ruy băng đã thấm mực vào trang giấy và để lại hình ảnh ký tự đó trên giấy. Trên máy tính cá nhân, loại máy in này có bộ chữ đúc sẵn, chữ cần in được đặt trước một dải băng tẩm mực căng song song trên mặt giấy. Khi có lệnh in, một chiếc búa gõ vào băng, để chữ đè lên giấy, hình nổi của chữ được in do băng mực. Máy tính điều khiển quay chữ mong muốn tới trước mặt băng giấy và ra lệnh để gõ búa.

Tùy theo cấu trúc của giá mang bộ chữ ta có các loại máy in sau:

Máy in quả cầu: bộ chữ trên một quả cầu xoay theo lệnh in.

Máy in bông cúc: bộ chữ nằm trên các cánh của một bánh xe quay giống hình bông hoa cúc, có thể có tới 96 chữ.

Máy in trống: bộ chữ trên một trống quay kim loại dài đặt trước băng mực.

Máy in dây xích: bộ chữ nằm trên một dây xích chuyển động không ngừng nhờ các bánh xe kéo.

Máy in dập có ưu điểm là đơn giản, chất lượng in tốt nhưng có nhược điểm tiếng ồn lớn và khi máy tính điều khiển thay đổi bộ chữ chương trình thì phải thay thế cả bộ chữ khác bằng tay.

7.5.2. Máy in ma trận chấm

Máy in ma trận chấm là loại máy in phổ biến nhất, trong đó một đầu in chứa từ 7 đến 24 kim có thể kích hoạt bằng điện từ và quét qua từng dòng in. Nguyên tắc tạo chữ của máy in giống như cách vẽ chữ hay ký hiệu trên màn hình, mỗi chữ được tạo bởi nhiều chấm mực bố trí theo một ma trận.

Tuỳ cách tạo chấm ta có các loại máy in sau:

– *Máy in ma trận kim dùng băng mực*: đầu in gồm một số kim (1, 7, 9, 14, 18, 24) được điều khiển bởi các búa gõ điện tử. Khi búa gõ, băng mực được ấn vào giấy tạo ra các chấm mực trên giấy. Tuỳ số lượng kim và kích thước búa gõ điện từ có thể bố trí các kim theo một cột (in chậm) hay theo ma trận để gõ đồng thời các búa gõ (in nhanh). Các motor bước sẽ được điều khiển làm quay giấy. Ngoài in chữ, máy in ma trận kim có thể in được hình vẽ nhiều màu (phụ thuộc số băng mực màu), tất nhiên mỗi lần chỉ in được một màu.

– *Máy in ma trận kim dùng nhiệt*: Giống máy in ma trận kim dùng băng mực, nhưng búa cơ điện và băng mực được thay thế bằng các đầu đốt nóng đặt thành ma trận. Đầu đốt nóng được áp sát vào giấy in, giấy in được dùng ở đây là loại giấy đặc biệt bị đổi màu khi bị làm nóng lên. Giấy in này có nhược điểm là đắt tiền và bản in bị đen khi giấy vô tình bị làm nóng (khi để gần vật toả nhiệt hay dưới nắng). Để khắc phục hiện tượng này, hiện nay người ta thường dùng băng tác dụng nhiệt, khi băng bị nung nóng, vết nung nóng đó truyền sang mặt giấy bình thường tạo thành vết đen do bị đốt nóng.

– *Máy in phun mực*: Tia mực được phun ra bề mặt giấy (giống tia điện tử của màn hình) được tích điện khi qua tụ tích điện bị lệch thẳng đứng do tụ lái tia mực rồi dập vào băng giấy. Điều khiển di chuyển đầu in theo chiều ngang, ta in được ma trận chấm của các ký tự và pixel bằng mực lên giấy. Có thể cải tiến thành các loại:

+ Có nhiều lỗ phun theo chiều dọc: in từng dòng điểm của ma trận và di chuyển đầu in theo chiều ngang, ta in được ma trận điểm.

+ Có một lỗ phun, di chuyển đầu in theo chiều ngang (giống quét dòng của màn hình) hoặc di chuyển giấy theo chiều dọc (giống quét mành của màn hình) nhưng tốc độ in chậm. Máy có độ phân giải 300 chấm/inch tương đương máy in laser nhưng hơi mờ hơn vì giọt mực nhỏ.

Ngoài nguyên lý in kiểu phun mực theo tia nhỏ, người ta còn chế tạo ra máy in tạo thành giọt, bọt hay bốc hơi.

7.5.3. Máy in laser

Nguyên tắc in của máy in laser như sau: Chùm tia laser mảnh đi qua bộ điều chế đến mặt trống có phủ lớp nhạy quang bằng selen để tạo nên hình ảnh bằng điện tích. Mực dạng bột đã tích điện được hút và bám vào mặt trống mà độ đậm nhạt tùy thuộc vào cường độ ánh sáng đã chiếu. Bột mực sẽ bám vào giấy tích điện với điện thế cao, ngược dấu với trống tạo nên hình ảnh ngược với hình ảnh bằng điện tích trên trống khi cho giấy đi qua trống quay. Sau đó giấy được đưa qua một trống sấy nóng ở nhiệt độ cao khoảng 180^0C làm hạt mực nóng chảy và bám chắc trên giấy tạo thành hình ảnh cần in. Việc điều chế chùm tia laser để chiếu lên mặt trống được thực hiện bằng một gương quay. Gương quay ở đây làm chùm tia lệch theo một dòng, dọc chiều dài trống. Khi trống quay nó sẽ quét thành các dòng liên tiếp trên mặt trống giống như quét mành trong truyền hình. Có thể thay chùm tia laser trên bằng các LED sắp xếp theo một ma trận để chiếu sáng theo nội dung cần in, đây chính là nguyên tắc hoạt động của các máy in của hãng OKI.

Tài liệu tham khảo

1. Văn Thế Minh. *Kỹ thuật Vi xử lý*. NXB Giáo dục, 1997.

2. Quách Tuấn Ngọc. *Lập trình hợp ngữ và Máy vi tính IBM – PC*. Đại học Bách khoa, Hà Nội, 2004.

3. *The 8086 Microprocessor Intel.*

4. *Microprocessors And Interfacing*. Douglas V. Hall. Source: www.amazon.com/**Microprocessors**–**Interfacing**–Programming–**Douglas**–**Hall**/

5. *MDA – 8086*. Manual Midas Engineering Co., Ltd

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 NGUYỄN QUÝ THAO

Tổ chức bản thảo và chịu trách nhiệm nội dung:
Chủ tịch HĐQT kiêm Giám đốc CTCP Sách ĐH-DN
TRẦN NHẬT TÂN

Biên tập và sửa bản in:
PHẠM THỊ PHƯỢNG

Trình bày bìa:
LƯU CHÍ ĐỒNG

Chế bản:
THUỶ TINH

GIÁO TRÌNH VI XỬ LÝ VÀ CẤU TRÚC MÁY TÍNH

Mã số: 7B678M7- DAI

In 1.500 bản (QĐ65), khổ 16 x 24cm. Tại Nhà in Hà Nam
Số 29 - Đường Lê Hoàn - TX. Phủ Lý - Hà Nam
Số in: 369. Số ĐKKH xuất bản: 192-2007/CXB/3-411/GD
In xong và nộp lưu chiểu tháng 9 năm 2007.

CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO
25 HÀN THUYÊN – HÀ NỘI
Website : www.hevobco.com.vn

VƯƠNG MIỆN KIM CƯƠNG
CHẤT LƯỢNG QUỐC TẾ

TÌM ĐỌC SÁCH THAM KHẢO KỸ THUẬT
CỦA NHÀ XUẤT BẢN GIÁO DỤC

1.	Giáo trình Vật lý đại cương tập một	Lương Duyên Bình
2.	Giáo trình Vật lý đại cương tập hai	Lương Duyên Bình
3.	Bài tập Vật lý đại cương tập một	Lương Duyên Bình
4.	Bài tập Vật lý đại cương tập hai	Lương Duyên Bình
5.	Giáo trình Kỹ thuật xung số	Đặng Văn Chuyết (CB)
6.	Giáo trình Kỹ thuật lập trình C	Nguyễn Linh Giang (CB)
7.	Giáo trình Kỹ thuật mạch điện tử	Đặng Văn Chuyết (CB)
8.	Giáo trình Linh kiện điện tử	Nguyễn Viết Nguyên (CB)
9.	Giáo trình Lý thuyết điều khiển tự động	Phan Xuân Minh (CB)
10.	Giáo trình Xử lý số tín hiệu	Nguyễn Quốc Trung (CB)
11.	Giáo trình Vi xử lý và cấu trúc máy tính	Ngô Diên Tập (CB)
12.	Giáo trình Khí cụ điện	Phạm Văn Chới
13.	Lập trình C# từ cơ bản đến nâng cao	Phạm Công Ngô
14.	Cơ sở tự động điều khiển quá trình	Nguyễn Văn Hoà

Bạn đọc có thể mua tại các Công ti Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng của Nhà xuất bản Giáo dục :

Tại Hà Nội : 25 Hàn Thuyên ; 187B Giảng Võ ; 232 Tây Sơn ; 23 Tràng Tiền ;

Tại Đà Nẵng : Số 15 Nguyễn Chí Thanh ; Số 62 Nguyễn Chí Thanh ;

Tại Thành phố Hồ Chí Minh : 104 Mai Thị Lựu, Quận 1 ; Cửa hàng 451B - 453, Hai Bà Trưng, Quận 3 ; 240 Trần Bình Trọng – Quận 5.

Tại Thành phố Cần Thơ : Số 5/5, đường 30/4 ;

Website : www.nxbgd.com.vn

8 934980 771600

Giá: **20.000 đ**