VIỆN ĐẠI HỌC MỞ HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

KIẾN TRÚC MÁY TÍNH

PGS. TS. VŨ CHẤN HƯNG

HÀ NỘI - 2012
(Lưu hành nội bộ)

LỜI MỞ ĐẦU

Máy tính điện tử đã và đang được sử dụng trong hầu hết các lĩnh vực hoạt động của đời sống con người. Cuốn sách này trình bầy về tổ chức, kiến trúc, chức năng và nguyên lý hoạt động của hệ thống máy tính hiện đại; tổ chức, kiến trúc, chức năng, nguyên lý hoạt động và định hướng thiết kế những thành phần cơ bản tạo nên hệ thống máy tính. Cuốn sách có 10 chương và được chia làm hai phần. Phần I là về tổ chức và kiến trúc của máy tính điện tử, phân tích cấu trúc, chức năng, nguyên lý hoạt động và định hướng thiết kế các thành phần cơ bản của hệ thống máy tính như đơn vị xử lý trung tâm, bộ nhớ, các thiết bị vào-ra và hệ thống bus. Phần II là về tổ chức và kiến trúc của một dòng máy tính đang được sử dụng phổ biến hiện nay là máy vi tính PC.

Phần I gồm 5 chương, trình bầy về tổ chức và kiến trúc máy tính.

Chương 1 mô tả kiến trúc cơ bản và nguyên lý hoạt động chung của máy tính điện tử số.

Chương 2 trình bầy về đơn vị xử lý trung tâm. Chương 2 mô tả kiến trúc, phân tích chức năng và định hướng thiết kế tập lệnh, định hướng thiết kế ở mức vi kiến trúc những thành phần tạo nên đơn vị xử lý trung tâm là khối xử lý dữ liệu và đơn vị điều khiển, bao gồm cả hai loại đơn vị điều khiển cứng hóa và vi lập trình.

Chương 3 trình bầy về tổ chức hệ thống và phương pháp truyền thông tin trên bus.

Chương 4 trình bầy về tổ chức bộ nhớ máy tính, thiết kế bộ nhớ RAM kích thước lớn, tổ chức bộ nhớ cache và các phương pháp quản lý bộ nhớ.

Chương 5 trình bầy về hệ thống vào-ra và các phương pháp vào-ra dữ liệu.

Phần II gồm 5 chương, trình bầy về tổ chức và kiến trúc máy vi tính PC.

Chương 6 trình bầy về kiến trúc máy vi tính PC và đơn vị xử lý trung tâm, trong đó có kiến trúc của một họ vi xử lý hiện đại là kiến trúc Core của Intel.

Chương 7 nói về các tổ chức DDR DRAM.

Chương 8 trình bầy về các thiết bị điều khiển và giao diện vào–ra chuẩn, bao gồm cả chuẩn giao diện tuần tự vạn năng USB.

Chương 9 là về các thiết bị vào ra cơ bản.

Chương 10 trình bầy về tổ chức và lưu trữ thông tin trên thiết bị đĩa từ, bao gồm tổ chức của hai hệ thống tập tin là FAT và NTFS.

Bạn đọc và sinh viên ngành Công nghệ thông tin và các ngành kỹ thuật có liên quan có thể sử dụng cuốn sách này làm tài liệu học tập, nghiên cứu hoặc tham khảo.

Hà Nội, tháng 1 năm 2012

TÁC GIẢ

PGS.TS. Vũ Chấn Hưng

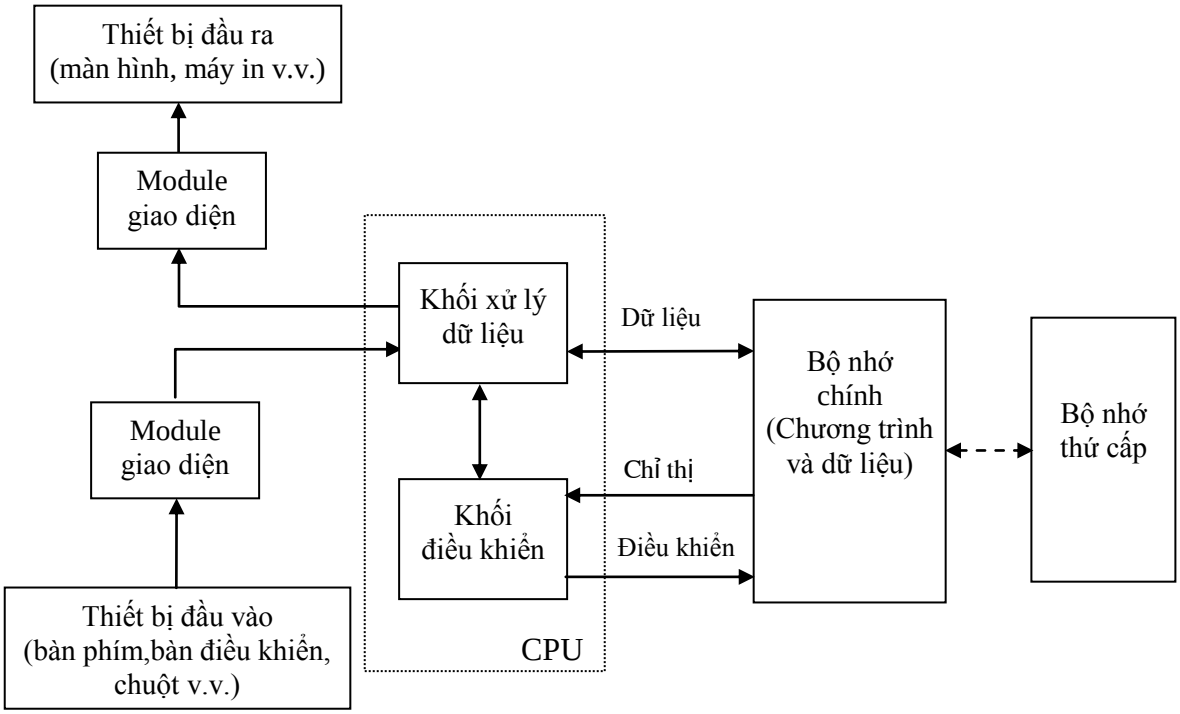
Phần I
KIẾN TRÚC MÁY TÍNH

Chương 1
KIẾN TRÚC CƠ BẢN CỦA MÁY TÍNH ĐIỆN TỬ

1.1. NHỮNG THÀNH PHẦN CƠ BẢN CỦA MÁY TÍNH ĐIỆN TỬ

Chức năng cơ bản của máy tính điện tử (nói gọn là máy tính) là thực hiện chương trình. Chương trình là một chuỗi các chỉ thị được đặt trong bộ nhớ. Chỉ thị và các dữ liệu khác trong máy tính đều được thể hiện dưới dạng các con số nhị phân.

Máy tính điện tử có các thành phần chính như sau (Hình 1).



Hình 1

- Đơn vị xử lý trung tâm (CPU- Central Processing Unit)

  Đơn vị xử lý trung tâm gồm 2 phần chính: khối điều khiển và khối xử lý dữ liệu.

  Đơn vị xử lý trung tâm thực hiện các chức năng sau:

  - Điều khiển ghi/đọc thông tin lên bộ nhớ.
  - Hiểu và thực hiện được một tập hữu hạn các chỉ thị (lệnh máy) được thể hiện dưới dạng số nhị phân.
  - Tuần tự nhập các chỉ thị từ bộ nhớ chính và thực thi (Chức năng thực hiện chương trình đang có trong bộ nhớ chính).
  - Điều khiển quá trình nhập thông tin từ thiết bị đầu vào và điều khiển quá trình xuất thông tin qua thiết bị đầu ra

- Bộ nhớ chính

Chức năng của bộ nhớ chính là lưu trữ chương trình đang được thực hiện và các dữ liệu có liên quan. Chương trình và dữ liệu được biểu diễn dưới dạng các con số nhị phân.

- Thiết bị đầu vào

Thiết bị đầu vào thực hiện chức năng nhập các thông tin nguyên thủy cho máy tính. Thiết bị đầu vào có chức năng mã hoá các thông tin đầu vào dưới dạng số nhị phân, để CPU có thể nhập và xử lý được. Thiết bị đầu vào có thể là bàn phím, chuột hoặc bàn điều khiển v.v.

- Thiết bị đầu ra

Thiết bị đầu ra được dùng để đưa các thông tin ra từ máy tính, ở dạng con người có thể hiểu được. Thiết bị đầu ra có thể là màn hình hiển thị, máy in, thiết bị âm thanh v.v.

Các thiết bị đầu vào/đầu ra (được gọi chung là các *thiết bị ngoại vi*) không được kết nối trực tiếp với đơn vị xử lý trung tâm CPU mà phải thông qua các *module vào-ra* (còn gọi là *module giao diện*). Sự có mặt của module giao diện là do có sự khác biệt rất lớn về dạng thức biểu diễn và truyền tải thông tin giữa đơn vị xử lý trung tâm và các thiết bị ngoại vi. Bên trong máy tính con số nhị phân được sử dụng làm phương tiện biểu diễn thông tin, còn ở thế giới bên ngoài máy tính thông tin lại được truyền tải và biểu diễn dưới dạng các ký tự, ánh sáng, âm thanh v.v. Đơn vị xử lý trung tâm CPU xử lý thông tin với tốc độ rất cao, các thiết bị bên ngoài máy tính xử lý thông tin với tốc độ chậm hơn nhiều. Do vậy module giao diện thực hiện chức năng ghép nối và giao diện giữa đơn vị xử lý trung tâm và các thiết bị ngoại vi, qua đó tạo khả năng để đơn vị xử lý trung tâm thực hiện trao đổi thông tin với các thiết bị ngoại vi và thế giới bên ngoài máy tính.

1.2. HỆ ĐẾM NHỊ PHÂN VÀ PHƯƠNG PHÁP BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH ĐIỆN TỬ

Trong quá trình phát triển, loài người đã có nhu cầu đếm các sự vật, sự việc. Hai bàn tay là một công cụ tự nhiên được sử dụng cho việc đếm. Hệ đếm thập phân (hệ cơ số 10) là kết quả tự nhiên của hệ thống đếm này.

Hệ đếm nhị phân là hệ đếm được dùng trong máy tính. Điều này xuất phát từ khả năng của công nghệ chế tạo các thiết bị lưu trữ và xử lý thông tin dạng nhị phân (bộ giải mã, bộ đếm, thiết bị thực hiện các phép tính số học và logic, bộ nhớ v.v.)

Hệ đếm nhị phân được dùng để mang tải và thể hiện thông tin. Hệ đếm nhị phân được xây dựng trên cơ sở các chữ số nhị phân "0" và "1".

Máy tính điện tử được xây dựng trên cơ sở các mạch điện. Hai trạng thái của mạch điện được dùng để thể hiện các con số nhị phân 1 và 0 là:

- Có hoặc không có dòng điện trong mạch
- Điện áp cao hoặc điện áp thấp

Ví dụ: con số 1 được thể hiện bằng trạng thái trong mạch có dòng điện hoặc có hiệu điện thế trên một điện kháng, con số 0 được thể hiện bằng trạng thái ngược lại.

1.2.1. Bit

Bit (Binary Digit) là một chữ số của hệ nhị phân, được biểu diễn bằng con số "0" hoặc "1". Bit là đơn vị nhỏ nhất biểu diễn dữ liệu (đơn vị nhỏ nhất mang thông tin) trong máy tính. Mỗi bit chỉ mang được một thông tin.

Ví dụ: con số "1" mô tả mạch điện được đóng, đèn sáng, trạng thái "cho phép" v.v. Con số "0" mô tả mạch điện hở, đèn tắt, trạng thái "cấm" v.v.

1.2.2. Biểu diễn dữ liệu số trong máy tính

a) Hệ thống đếm, cơ số và trọng số

Để biểu diễn số lượng bằng con số thì có thể dùng các hệ đếm khác nhau.

Thông thường ta sử dụng các ký hiệu (chữ số) 0 1 2 3 4 5 6 7 8 9 để ghi lại các số lượng đếm được. Muốn ghi lại các trị lớn hơn phải sử dụng ký pháp vị trí. Trong ký pháp này vị trí của mỗi chữ số xác định gía trị của nó trong con số.

Ký pháp vị trí tổng quát của một số là:

$a_{n-1}a_{n-2}... a_1a_0.b_1b_2... b_{m-1}b_m = a_{n-1}*E^{n-1} + a_{n-2}*E^{n-2} +... + a_1*E^1 + a_0*E^0 + b_1*E^{-1} + b_2*E^{-2} +... + b_{m-1}*E^{-(m-1)} + b_m*E^{-m}$

trong đó:

E là cơ số

$E^{n-1,}$ ..., E^2, $E^{1,}$ E^0, E^{-1}, E^{-2},..., $E^{-(m-1)}$, E^{-m} là các trọng số

$a_{n-1}, a_{n-2}, ..., a_1, a_0, b_1, b_2, . ., b_{m-1}, b_m$ là các số nằm ở vị trí tương ứng với trọng số.

b) Hệ đếm thập phân (Hệ cơ số 10)

Hệ thập phân dùng mười chữ số từ "0" đến "9" và ký pháp vị trí để thể hiện số đếm.

Ví dụ: con số 126.5 có ý nghĩa là Một trăm Hai chục Sáu đơn vị và Năm phần mười đơn vị.

Con số 126.5 là thể hiện của giá trị:

$$126.5_{10} = 1*10^2 + 2*10^1 + 6*10^0 + 5*10^{-1}$$

trong đó

10 là cơ số

$10^2, 10^1, 10^0, 10^{-1}$ là các trọng số

1, 2, 6, 5 là các chữ số được đặt ở vị trí tương ứng với trọng số.

c) Hệ đếm nhị phân (Hệ cơ số 2)

Hệ nhị phân dùng hai chữ số "0" và "1" và ký pháp vị trí để thể hiện số đếm với

2 là cơ số

$2^2, 2^1, 2^0, 2^{-1}, 2^{-2}$, v.v. là các trọng số

Các chữ số "0" và "1" được đặt ở vị trí tương ứng với trọng số

Ví dụ: một giá trị 126.5 hệ thập phân sẽ được biểu diễn ở hệ nhị phân dưới dạng:

$$1111110.1_2 = 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1}$$

d) Byte

Một byte là tổ hợp của 8 bit, vị trí mỗi bit tương ứng với trọng số của bit đó. Một byte dữ liệu có thể mang tải 1 trong 256 thông tin khác nhau, với những thể hiện như:

00000000
00000001

11111111

e) Các phép tính với số nhị phân

- Phép cộng:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 dư 1

Ví dụ:
```
  1 1 0
+ 0 1 1
  1 1 0      phần dư
1 0 0 1
```

- Phép trừ:

```
0 - 0 = 0
0 - 1 = 1      nợ 1
1 - 0 = 1
1 - 1 = 0
```

Ví dụ:
```
  1 1 0
- 0 1 1
  1 1        phần nợ
  0 1 1
```

f) Chuyển đổi hệ cơ số 10 sang hệ cơ số 2

Quy tắc chuyển đổi:

Để chuyển một số ở hệ cơ số 10 sang hệ cơ số 2 trước hết cần tách số đó làm hai phần, phần nguyên và phần sau dấu thập phân và thực hiện chuyển đổi từng phần theo hai cách khác nhau, sau đó cộng hai con số kết quả tạm đó lại thành con số nhị phân kết quả của chuyển đổi.

Quy tắc chuyển đổi phần nguyên:

Lấy số phần nguyên chia cho 2, được phần thương số và phần dư. Phần dư được dùng làm kết quả chuyển đổi, phần thương số được dùng để tiếp tục chia 2. Quá trình này được thực hiện cho đến khi thương số bằng 0 thì dừng.

Quy tắc chuyển đổi phần sau dấu thập phân:

Thực hiện nhân đôi phần sau dấu thập phân, ở tích số nếu có số 1 xuất hiện ở bên trái dấu thập phân thì thêm 1 vào bên phải của số nhị phân, nếu có số 0 xuất hiện ở bên trái dấu thập phân thì thêm 0 vào bên phải của số nhị phân. Quá trình này được lặp lại cho đến khi tích số thập phân là 1 thì dừng.

Ví dụ : chuyển đổi con số 19.625_{10} ở hệ cơ số 10 sang hệ cơ số 2.

Con số này được tách ra làm hai phần. Phần nguyên là 19, phần sau dấu thập phân là 0.625.

Chuyển đổi phần nguyên:

	Thương số	Dư
19 ÷ 2 =	9	1
9 ÷ 2 =	4	1
4 ÷ 2 =	2	0
2 ÷ 2 =	1	0
1 ÷ 2 =	0	1

Kết quả chuyển đổi phần nguyên: $19_{10} = 10011$

Chuyển đổi phần sau dấu thập phân:

	Tích số
2 x 0,625 = 1,25	0.1
2 x 0,25 = 0,50	0.10
2 x 0,50 = 1,0	0.101

Kết quả chuyển đổi phần sau dấu thập phân: $0,625_{10} = 0.101_2$

Kết quả chuyển đổi toàn phần:

$19,625_{10} = 10011 + 0.101 = 10011.101_2$

g) Biểu diễn số âm

Số âm trong hệ cơ số 10 được biểu diễn bằng cách thêm ký hiệu " – " vào trước số đó, ví dụ: âm 88 là -88.

Trong máy tính các con số được biểu diễn bởi các chữ số nhị phân, ví dụ 1111110_2 là một con số nhị phân 7 bit. Con số nhị phân 7 bit biểu diễn được các giá trị trong khoảng 0000000 đến 1111111. Để biểu diễn được các giá trị trong khoảng từ 0000000 đến -1111111 cần thêm bit thứ tám, gọi là *bit dấu*. Bit dấu nằm ở vị trí bit cao nhất trong số nhị phân. Theo quy ước chung số dương có bit dấu là 0, số âm có bit dấu là 1. Ví dụ -126_{10} khi biểu diễn ở hệ cơ số 2 là 11111110_2.

Trên thực tế còn một cách biểu diễn số âm khác nữa, đó là cách biểu diễn bằng *số bù*. Kỹ thuật biểu diễn số âm bằng số bù có thể được dùng cho cả hai hệ thống cơ số 10 và cơ số 2. Với cách biểu diễn số âm bằng số bù, máy tính có thể thực hiện cả hai phép tính cộng và trừ chỉ bằng một phép tính cộng.

- Ở hệ cơ số 10 một số âm có thể biểu diễn qua *số bù 10*. Số bù 10 của một số được tạo ra bằng cách lấy 9 trừ mỗi chữ số và cộng 1 vào chữ số có ý nghĩa thấp nhất của số vừa được tạo ra, ví dụ số bù 10 của 88 là 12.

Để thực hiện phép trừ một số dương (số bị trừ) cho một số dương (số trừ), người ta thực hiện phép cộng số bị trừ với số bù 10 của số trừ. Nếu có nhớ ở chữ số cao nhất thì chữ số đó bị bỏ đi và kết quả là dương. Nếu không nhớ thì kết quả là âm, để nhận được kết quả thật cần lấy bù 10 của số đó và thêm dấu " – " vào đằng trước.

Ví dụ:

Phép trừ bình thường	Phép trừ bù 10
97	97
- 88	+12
09	109
	Bỏ phần có nhớ, Kết quả thật là số dương 09
53	53
- 67	+33
- 14	86
	Không nhớ, là số âm và cần lấy bù 10. Kết qủa thật là -14

- Ở hệ cơ số 2 một số âm có thể được biểu diễn qua *số bù 2*. Số bù 2 của một số nhị phân được tạo ra bằng cách lấy 1 trừ mỗi chữ số và cộng 1 vào chữ số có ý nghĩa thấp nhất của số vừa được tạo ra. Phương pháp tạo số bù 2 của một số trên thực tế được thực hiện bằng cách đổi số 0 thành số 1, đổi số 1 thành số 0 và cộng 1 vào chữ số có ý nghĩa thấp nhất của số vừa được tạo ra. Ví dụ số bù 2 của 11000 là 01000.

Với hệ thống bù 2, các phép tính cộng và trừ hai số nhị phân được thực hiện qua chỉ một phép tính cộng. Trong hệ thống bù 2 khi thực hiện cộng hai số có thể có bốn tình huống xảy ra:

- Khi cả hai số là dương. Khi thực hiện phép cộng thì cộng từng số từ phải qua trái, kể cả bit dấu.

  Ví dụ cộng hai số dương 4 bit:

Ký pháp bình thường	Dạng dữ liệu trong máy tính (Sử dụng hệ thống bù 2, thêm một bit dấu. Bit có gạch chân là bit dấu)	
+1000	01000	
+0101	00101	
+1101	01101	Kết quả là số dương
		Kết quả thật là +1101

- Khi một số là dương và một số là âm, số dương có trị tuyệt đối lớn hơn. Bù 2 của số âm được tạo và thêm bit dấu vào vị trí cao nhất.

  Khi thực hiện phép cộng, một bit nhớ sẽ được tạo ra từ bit dấu. Cần bỏ bit nhớ đi. Kết quả là một số dương.

  Ví dụ cộng hai số khác dấu 4 bit:

Ký pháp bình thường	Dạng dữ liệu trong máy tính (Sử dụng hệ thống bù 2, thêm một bit dấu. Bit có gạch chân là bit dấu)
+1000	01000
- 0101	11011
+0011	100011
	└── Bit nhớ được bỏ
	Kết quả là số dương, kết quả thật là +0011

- Khi một số là dương và một số là âm, số âm có trị tuyệt đối lớn hơn.

  Khi thực hiện phép cộng, không có nhớ từ bit dấu. Kết quả là một số âm ở dạng bù 2. Cần bỏ bit dấu ở kết quả và phải tạo số bù 2 với số đó để nhận được kết quả thật.

  Ví dụ cộng hai số khác dấu 4 bit:

Ký pháp bình thường	Dạng dữ liệu trong máy tính (Sử dụng hệ thống bù 2, thêm một bit dấu. Bit có gạch chân là bit dấu)	
+ 0101	00101	
- 1000	11000	
- 0011	11101	Không có nhớ Kết quả là số âm dạng bù 2. Kết quả thật là - 0011

- Khi cả hai số là âm.
  Khi thực hiện phép cộng, có nhớ được tạo ra từ bit dấu. Cần bỏ bit nhớ và kết quả là một số âm ở dạng bù 2. Phải bỏ bit dấu và tạo số bù 2 với số đó để nhận được kết quả thật.
  Ví dụ cộng hai số âm 4 bit:

Ký pháp bình thường	Dạng dữ liệu trong máy tính (Sử dụng hệ thống bù 2, thêm một bit dấu. Bit có gạch chân là bit dấu)	
-0101	11011	
-1000	11000	
-1101	110011	Bit nhớ được bỏ Kết quả là số âm dạng bù 2. Kết quả thật là - 1101

h) Biểu diễn số thực

Trong các hệ đếm, để mô tả các giá trị có phần nhỏ hơn 1 người ta dùng một dấu để phân cách hai phần nguyên và phần phân số. Trong khoa học thường phải tính toán với số cực lớn hoặc cực nhỏ, trong trường hợp này các nhà khoa học thường sử dụng ký pháp dấu chấm động (*số dấu chấm động*):

$$\pm\text{Định trị} * \text{Cơ số}^{\pm\text{số mũ}}$$

Ví dụ: $490000 = 0.49*10^{6}$

$0.00023 = 0.23*10^{-3}$

Các phép tính với số dấu chấm động được thực hiện theo quy tắc:

- Phép nhân: $x = a*E^{n}$, $y = b*E^{m}$

  $x*y = a*b*E^{n+m}$
- Phép chia: $x = a*E^{n}$, $y = b*E^{m}$

  $x/y = (a/b)*E^{n-m}$
- Phép cộng: $x = a*E^{n}$, $y = b*E^{m}$, $n < m$

  Để cộng hai số x và y cần biến đổi *n* thành *m* và *a* thành *a'*

$x+y = (a'+b) * E^{m}$

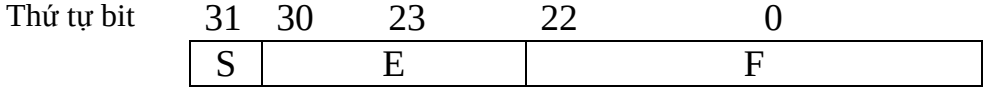
Quá trình biến đổi này gọi là quá trình tỷ lệ hoá (scaling).

Quá trình tỷ lệ hóa được tiến hành tự động trong máy tính nhờ phần cứng hoặc bằng phần mềm.

Trong máy tính, để biểu diễn các giá trị cực lớn hoặc cực nhỏ người ta cũng dùng cách biểu diễn theo dấu chấm động. Giá trị dấu chấm động được biểu diễn ở dạng số nhị phân với cơ số 2 và theo chuẩn IEEE 754. Có hai khuôn dạng số dấu chấm động: khuôn dạng đơn giản và khuôn dạng độ chính xác gấp đôi.

- Khuôn dạng đơn giản

Con số dấu chấm động được lưu trữ dưới khuôn dạng dữ liệu 32 bit:



Độ dài 1bit 8bit 23 bit

Dữ liệu này biểu diễn giá trị: $V = (-1)^{S} * 2^{E-127} * 1.F$

V : giá trị của số dấu chấm động

S : bit dấu

E : số nguyên nhị phân 8 bit

F : Phần sau dấu chấm nhị phân (dạng số nhị phân).

Phần định trị có dạng 1. F

Phạm vi biểu diễn của số dấu chấm động 32 bit chuẩn hoá khác 0, gồm hai vùng:

- Vùng số âm: $-(1.F)\ 2^{E-127}$ với $0< E < 255$
- Vùng số dương: $+(1.F)\ 2^{E-127}$ với $0< E < 255$

Các kết quả xuất hiện nằm ngoài hai vùng này khi thực hiện các phép tính với số dấu chấm động, sẽ được hiểu như sau:

- Giá trị với số mũ bằng 0 cùng với phần định trị bằng 0 biểu diễn gía trị -0 hoặc +0, tuỳ thuộc bit dấu.
- Giá trị với số mũ bằng 0 cùng với phần định trị khác 0 biểu diễn con số phi chuẩn hoá
- Giá trị với tất cả các bit trong số mũ bằng 1 (số mũ bằng 255) cùng với phần định trị bằng 0 biểu diễn gía trị dương vô cùng hoặc âm vô cùng.
- Giá trị với tất cả các bit trong số mũ bằng 1 (số mũ bằng 255) cùng với phần định trị khác 0 cho giá trị NaN (Not a Number), được dùng để chỉ báo các điều kiện ngoại lệ.

- Dạng độ chính xác gấp đôi

Giá trị dấu phảy động độ chính xác gấp đôi được lưu trữ dưới dạng dữ liệu 64 bit:

Thứ tự bit	63	62 ... 52	51 ... 0
	S	E	F

Dữ liệu này biểu diễn giá trị:

$$V = (-1)^S * 2^{E-1023} * 1.F$$

Phạm vi biểu diễn của số dấu chấm động 64 bit được hiểu theo cách tương tự như với số chấm động 32 bit.

1.3. KIẾN TRÚC CƠ BẢN CỦA MÁY TÍNH ĐIỆN TỬ

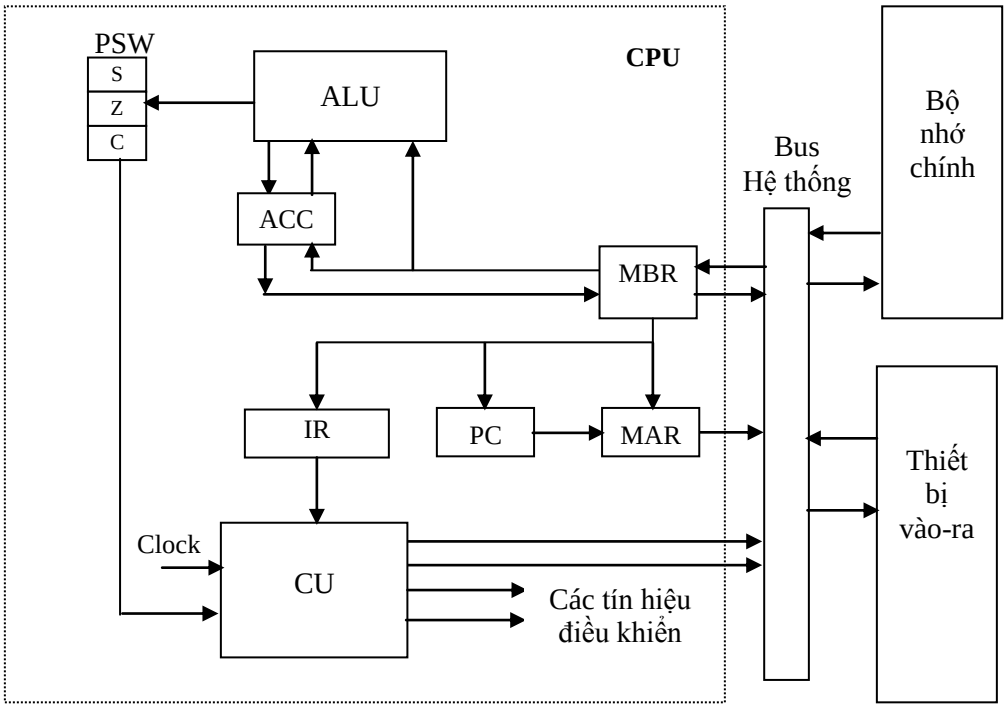
Chức năng cơ bản của máy tính điện tử là thực hiện chương trình, tức là thực thi một chuỗi các lệnh máy (các chỉ thị) trong chương trình. Chương trình này được chứa trong bộ nhớ chính. Đơn vị xử lý trung tâm CPU là thành phần có chức năng thực thi các lệnh máy (Hình 2).

Đơn vị xử lý trung tâm của máy tính hiểu và thực hiện được các *lệnh máy* (*chỉ thị*) có khuôn dạng sau:

Mã thao tác	Phần địa chỉ

Một lệnh máy bao gồm hai phần: mã thao tác (operation code - OP) và phần địa chỉ:

- Mã thao tác, là một số nhị phân, mang thông tin về những việc hoặc những thao tác mà đơn vị xử lý trung tâm cần phải thực hiện.
- Phần địa chỉ của lệnh, là số nhị phân, có thể gồm một hoặc nhiều thành phần, chứa thông tin về vị trí (địa chỉ) nơi chứa toán hạng (dữ liệu) nguồn và toán hạng kết quả.



Hình 2

1.3.1. Đơn vị xử lý trung tâm

Đơn vị xử lý trung tâm (Central Processing Unit - CPU) gồm hai khối chức năng chính là khối điều khiển và khối xử lý dữ liệu.

Khối điều khiển bao gồm các khối chức năng sau:

- Bộ đếm chương trình PC (Program Counter)

Bộ đếm chương trình PC có chức năng tuần tự tạo ra địa chỉ ô nhớ chứa lệnh máy CPU cần nhập. PC xác định địa chỉ của ô nhớ chứa lệnh máy tiếp theo mà CPU sẽ nhập và thực thi. PC thực hiện vai trò một con trỏ trỏ đến ô nhớ tiếp theo chứa lệnh sẽ được nhập, vai trò con trỏ lệnh.

Giá trị của PC tăng tuần tự khi nhập lệnh, tạo ra địa chỉ các ô nhớ chứa lệnh CPU cần nhập.

Khi nhập xong một lệnh máy thì giá trị của PC tự động tăng lên và trỏ đến ô nhớ chứa lệnh tiếp theo (trỏ đến lệnh tiếp theo).

Khi CPU thực hiện các lệnh điều khiển rẽ nhánh thì giá trị của PC thay đổi đột biến.

- Thanh ghi lệnh (thanh ghi chỉ thị) IR (Instruction register)

Thanh ghi lệnh IR thực hiện chức năng chứa mã thao tác của lệnh mà CPU đang thực thi.

- Đơn vị điều khiển CU (Control Unit)

Đơn vị điều khiển CU có chức năng giải mã lệnh và từ đó tạo ra các tín hiệu điều khiển hoạt động của các đơn vị chức năng khác ở bên trong và bên ngoài đơn vị xử lý trung tâm CPU, nhằm thực thi được lệnh hiện hành.

CU thực hiện công việc này bằng cách, thông qua các tín hiệu điều khiển và theo nhịp của xung đồng hồ hệ thống, điều khiển thực hiện chuỗi các vi thao tác cần thiết để thực thi lệnh.

- Thanh ghi địa chỉ bộ nhớ MAR (Memory Address Register)

Thanh ghi địa chỉ bộ nhớ MAR thực hiện chức năng chứa địa chỉ của ô nhớ chứa lệnh hoặc toán hạng CPU cần truy nhập.

Khối xử lý dữ liệu bao gồm các khối chức năng:

- Thanh ghi MBR (Memory Buffer Register)

Thanh ghi MBR là thanh ghi đệm, chứa dữ liệu CPU đọc từ bộ nhớ hoặc ghi ra bộ nhớ.

- Đơn vị số học - logic ALU (Arithmetic - Logic Unit)

ALU thực hiện các phép tính số học, logic và các phép xử lý dữ liệu khác.

- Thanh chứa ACC (Accumulator)

Thanh chứa ACC chứa một toán hạng của lệnh hoặc kết quả thực hiện lệnh.

- Thanh ghi trạng thái PSW

Thanh ghi trạng thái PSW (Program Status Word) chứa các bit mang thông tin về trạng thái thực hiện lệnh. Các bit mang thông tin trạng thái thường được gọi là các bit cờ (flags).

Trạng thái của việc thực hiện lệnh hiện hành có thể được sử dụng và có thể tác động đến quá trình thực hiện lệnh máy tiếp theo. Qua PSW có thể tạo ra mối quan hệ logic giữa các lệnh máy trong một chương trình.

Ví dụ, thanh ghi PSW thường có các bit cờ sau:

Bit cờ Z - cờ Zero: nếu kết quả phép tính là 0 thì bit cờ Z được đặt lên 1

Bit cờ S - cờ Sign: nếu kết quả phép tính là âm thì bit cờ S được đặt lên 1

Bit cờ C- cờ Carry: nếu kết quả phép tính có nhớ thì bit cờ C được đặt lên 1

1.3.2. Bộ nhớ

Bộ nhớ là tập hợp các ô nhớ theo một trật tự nhất định, mỗi ô nhớ có một địa chỉ. Địa chỉ là con số xác định vị trí của ô nhớ trong bộ nhớ. Chức năng của bộ nhớ là chứa thông tin (chương trình và các dữ liệu có liên quan).

Để đọc/ghi một ô nhớ nào đó cần phải xác định địa chỉ chọn ô nhớ đó và cần một tín hiệu điều khiển đọc/ghi bộ nhớ.

Thao tác đọc bộ nhớ:

- CU đưa địa chỉ của ô nhớ cần đọc qua MAR ra bus địa chỉ
- CU đưa ra tín hiệu điều điều khiển đọc RD đến bộ nhớ
- Nội dung ô nhớ được đọc từ bộ nhớ vào MBR của đơn vị xử lý trung tâm

Thao tác ghi ô nhớ:

- CU đưa địa chỉ của ô nhớ cần ghi qua MAR ra bus địa chỉ
- CU đưa dữ liệu qua MBR ra bus dữ liệu
- CU đưa ra tín hiệu điều khiển ghi WR đến bộ nhớ, dữ liệu từ đơn vị xử lý trung tâm được ghi vào bộ nhớ

1.3.3. Thiết bị vào-ra (nhập-xuất) dữ liệu

Thiết bị vào-ra thực chất được xây dựng trên cơ sở hai module thành phần là module vào-ra và các modul đầu vào hoặc đầu ra (các thiết bị ngoại vi). Các module vào-ra một mặt kết nối và giao diện với CPU, mặt kia kết nối và giao diện với các thiết bị ngoại vi, do vậy các module vào-ra còn được gọi là các module giao diện. Thông qua các module giao diện CPU có thể trao đổi thông tin với các thiết bị ngoại vi và thế giới bên ngoài.

1.3.4. Tập lệnh của đơn vị xử lý trung tâm CPU

Đơn vị xử lý trung tâm của máy tính có khả năng thực thi được một tập hữu hạn các lệnh máy (các chỉ thị). Các lệnh này thường gồm các nhóm như: nhóm lệnh chuyển dữ liệu, nhóm lệnh xử lý dữ liệu (trong đó có các lệnh số

học-logic), nhóm lệnh điều khiển rẽ nhánh, nhóm lệnh vào-ra (nhập-xuất) dữ liệu v.v.

Giả định ta khảo sát một CPU có tập gồm 10 lệnh khuôn dạng địa chỉ đơn, được biểu diễn dưới dạng mã nhị phân. Ở loại CPU có tập lệnh địa chỉ đơn, phần địa chỉ của lệnh chỉ có một thành phần, xác định địa chỉ của một toán hạng. Để thực thi được các lệnh cần hai toán hạng nguồn, ví dụ như lệnh cộng ADD và lệnh trừ SUB, thì toán hạng kia ngầm định là phải nằm (trước) ở trong thanh ghi ACC. Thanh ghi ACC cũng là nơi chứa kết quả.

Giả định mỗi một lệnh sẽ có độ dài 8 bit, gồm hai phần: 4 bit cao là mã thao tác, 4 bit thấp là phần địa chỉ của toán hạng. Để thuận lợi hơn cho việc mô tả, diễn đạt và lập trình, người ta thường mô tả lệnh máy dưới dạng ngôn ngữ gợi nhớ (mnemonic) như sau:

Lệnh máy (chỉ thị)	Mã thao tác	Thao tác thực hiện lệnh
LOAD xxxx	0001	ACC <- M_{xxxx}
STORE xxxx	0010	M_{xxxx} <- ACC
ADD xxxx	0011	ACC <- ACC + M_{xxxx}
SUB xxxx	0100	ACC <- ACC - M_{xxxx}
JMP xxxx	0101	PC <- (Phần địa chỉ $xxxx_)$
JZ xxxx	0110	Nếu Z ≡ 1 thì PC <- (Phần địa chỉ $xxxx_)$
JC xxxx	0111	Nếu C ≡ 1 thì PC <- (Phần địa chỉ $xxxx_)$
JS xxxx	1000	Nếu S ≡ 1 thì PC <- (Phần địa chỉ $xxxx_)$
IN xxxx	1001	ACC <- (Thiết bị vào có địa chỉ xxxx)
OUT xxxx	1010	Thiết bị ra có địa chỉ xxxx <- ACC

Trong đó ký hiệu:

xxxx là con số 4 bit, xác định địa chỉ của ô nhớ chứa toán hạng hoặc địa chỉ thiết bị

M_{xxxx} là nội dung của ô nhớ có địa chỉ xxxx trong bộ nhớ chính

1.3.5. Hoạt động của máy tính

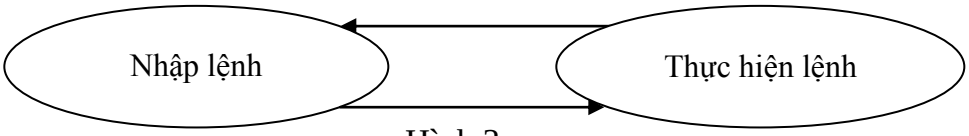
a) Hoạt động của máy tính và chu kỳ lệnh

Chức năng cơ bản của máy tính là thực hiện chương trình. Việc thực hiện chương trình diễn ra như sau:

- CPU tuần tự nhập các lệnh từ ô nhớ PC trỏ tới và dữ liệu (nếu cần) từ bộ nhớ chính vào các thanh ghi bên trong CPU.

- CPU thực thi (tuần tự) các lệnh được nhập. Nơi chứa kết quả được xác định trên lệnh.
- Nếu cần, CPU chuyển kết quả từ các thanh ghi CPU ra bộ nhớ chính.

Việc thực hiện chương trình thực chất là sự lặp lại quá trình nhập lệnh và thực hiện lệnh (Hình 3). Lệnh được nhập từ nơi con trỏ lệnh PC trỏ đến.

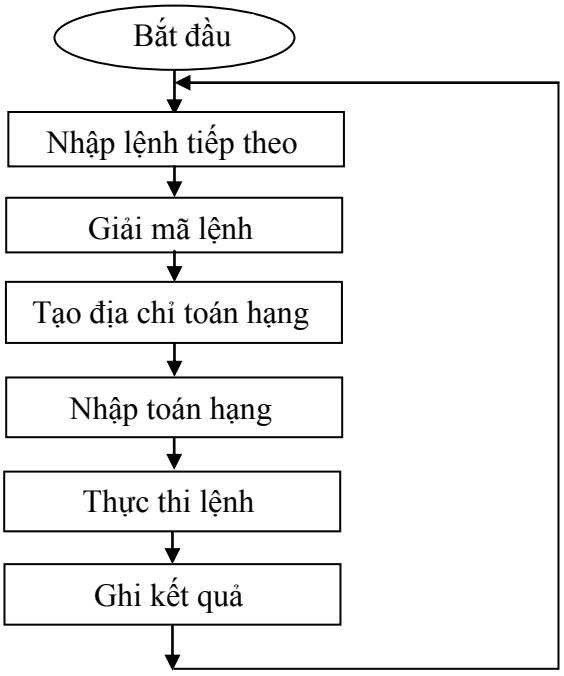


Hình 3

Mỗi một lệnh được thực hiện trong một chu kỳ lệnh. Chu kỳ lệnh là khoảng thời gian từ khi nhập lệnh đến khi thực hiện xong lệnh. Các chu kỳ lệnh không nhất thiết phải dài bằng nhau. Mỗi một chu kỳ lệnh xảy ra trên nhiều chu kỳ nhịp đồng hồ của CPU. Mỗi một chu kỳ lệnh có thể bao gồm nhiều tiểu chu kỳ, phụ thuộc loại lệnh và hành trạng của CPU, trong đó luôn có tiểu chu kỳ nhập lệnh và tiểu chu kỳ thực hiện lệnh. Trong tiểu chu kỳ thực hiện lệnh CPU thực hiện các thao tác giải mã lệnh, tạo địa chỉ toán hạng và nhập toán hạng (nếu cần), thực thi lệnh, ghi kết quả.

Nói chung việc thực hiện một lệnh thường trải qua các giai đoạn: *nhập lệnh*, *giải mã lệnh*, *tạo địa chỉ toán hạng và nhập toán hạng* (nếu cần), *thực thi lệnh*, *ghi kết quả vào bộ nhớ*, trừ khi kết quả nằm ở các thanh ghi của CPU.

Hình 4 là lưu đồ trạng thái của một chu kỳ lệnh.



Hình 4

b) Quá trình thực hiện lệnh

Ta khảo sát hoạt động của máy tính khi thực hiện một lệnh của một chương trình. Giả sử máy tính đang thực hiện chương trình. Tại thời điểm khảo sát, nội dung của PC là 0100, ô nhớ có địa chỉ 0100 đang chứa lệnh ADD 1000, ô nhớ có địa chỉ 1000 đang chứa giá trị 0111, thanh ghi ACC đang chứa giá trị 0101. Lệnh ADD 1000 sẽ thực hiện phép cộng giá trị đang có trong ACC với nội dung của ô nhớ có địa chỉ 1000.

Địa chỉ bộ nhớ	Nội dung bộ nhớ	Nội dung bộ nhớ ở dạng ngôn ngữ gợi nhớ
0100	00111000	ADD 1000
... .	... .	... .
... .	... . .	... .
1000	0111	0111
... .	... .	... .

ACC
0101

Kết quả phép cộng là 1100

Đơn vị xử lý trung tâm bắt đầu nhập lệnh từ ô nhớ có địa chỉ 0100, nơi đang chứa lệnh ADD xxxx và thực hiện lệnh này.

Quá trình thực hiện lệnh ADD xxxx diễn ra như sau:

(1) Nhập lệnh:

PC ≡ 0100

- MAR <- PC ; (MAR) ≡ 0100

CU đưa ra tín hiệu điều khiển đọc bộ nhớ RD

- MBR <- M_{0100} ; MBR ≡ 00111000
- IR <- MBR (phần mã thao tác OP) ; IR ≡ 0011
- PC <- PC + 1 ; PC ≡ 0101

(2) Giải mã lệnh:

- ID <- IR

Bộ giải mã lệnh ID (Instruction Decoder) trong CU thực hiện giải mã lệnh, từ đây xác định được cần thực hiện các thao tác sau:

- nhập toán hạng thứ 2 của phép cộng từ ô nhớ có địa chỉ 1000
- thực thi phép cộng

(3) Tạo địa chỉ toán hạng:

- MAR <- MBR (phần địa chỉ); MAR ≡ 1000

(4) Nhập toán hạng:

CU đưa ra tín hiệu điều khiển đọc bộ nhớ RD

- MBR <- M_{1000} ; MBR ≡ 0111

(5) Thực thi lệnh và ghi kết quả:

- ACC <- ACC + MBR

Kết quả của việc thực hiện lệnh ADD là ACC ≡ 1100

Đơn vị xử lý trung tâm tiếp tục nhập lệnh từ ô nhớ trỏ bởi PC ≡ 0101 và thực hiện lệnh tiếp theo này.

Lệnh ADD xxxx được thực hiện qua hai tiểu chu kỳ nhập lệnh và thực thi lệnh. Sau giải mã lệnh, CPU thực hiện trong tiểu chu kỳ thực hiện lệnh các thao tác như: tạo địa chỉ toán hạng, nhập toán hạng, thực thi lệnh, ghi kết quả. Để thực hiện các thao tác này CPU phải thực hiện tuần tự hoặc đồng thời các *vi thao tác*. Vi thao tác là hoạt động cơ bản nhất của đơn vị xử lý trung tâm. Các hoạt động cơ bản này được kích hoạt dưới sự điều khiển của đơn vị điều khiển CU.

Chương 2
ĐƠN VỊ XỬ LÝ TRUNG TÂM

Đơn vị xử lý trung tâm có những chức năng cơ bản sau: điều khiển quá trình lưu trữ dữ liệu ở bộ nhớ, hiểu và thực thi được một tập hữu hạn các chỉ thị (lệnh máy), tuần tự nhập các chỉ thị từ bộ nhớ và thực thi, điều khiển vào-ra dữ liệu với thế giới bên ngoài.

2.1. TẬP LỆNH CỦA MÁY TÍNH

Hoạt động của đơn vị xử lý trung tâm (ĐVXLTT) được xác định bởi tập lệnh mà nó thực hiện. Các lệnh này được gọi là *lệnh máy tính*.

2.1.1. Các tính chất đặc trưng của lệnh máy tính

Đơn vị xử lý trung tâm có thể thực hiện nhiều chức năng khác nhau, điều này được phản ánh qua các lệnh được định nghĩa cho nó. Mỗi một lệnh phải chứa các thông tin yêu cầu về các thao tác ĐVXLTT phải thực hiện. Điều này xác định cấu trúc của một lệnh. Một lệnh máy cần gồm các thành phần sau

- *Mã thao tác* (operation code - OP): thành phần này xác định thao tác xử lý dữ liệu cần thực hiện (vd. ADD, SUB v.v.). Mã thao tác được thể hiện ở dạng số nhị phân.
- *Địa chỉ toán hạng nguồn*: thao tác có thể đòi hỏi một hoặc hai toán hạng nguồn làm đầu vào cho thao tác.
- *Địa chỉ toán hạng kết quả*.
- *Địa chỉ lệnh tiếp theo*: thành phần này báo đơn vị xử lý trung tâm nơi nhập lệnh tiếp theo. Trong phần lớn các trường hợp, lệnh tiếp theo cần nhập nằm ngay sau lệnh hiện hành. Trong trường hợp này không cần có thành phần địa chỉ tường minh cho lệnh tiếp theo.

Toán hạng nguồn và kết quả có thể nằm ở một trong ba vị trí sau: bộ nhớ chính hoặc bộ nhớ ảo, thanh ghi của ĐVXLTT, thiết bị vào-ra (nhập-xuất).

Các loại lệnh

Tập lệnh của máy thường bao gồm các nhóm lệnh thực hiện các thao tác xử lý dữ liệu cơ bản sau:

- Nhóm lệnh chuyển dữ liệu: các lệnh chuyển dữ liệu giữa ĐVXLTT và bộ nhớ
- Nhóm lệnh xử lý dữ liệu: các lệnh số học và logic
- Nhóm lệnh điều khiển chương trình: các lệnh kiểm tra và điều khiển rẽ nhánh
- Nhóm lệnh vào-ra (nhập-xuất) dữ liệu: các lệnh vào-ra dữ liệu giữa ĐVXLTT và thiết bị bên ngoài.

Bảng 1 mô tả các lệnh điển hình trong bốn nhóm lệnh trên.

Bảng 1

Loại lệnh	Tên lệnh	Diễn giải
Chuyển dữ liệu	LOAD	Sao chép một từ dữ liệu từ bộ nhớ vào thanh ghi CPU
	STORE	Sao chép một từ dữ liệu từ thanh ghi CPU vào bộ nhớ
	MOVE	Sao chép một khối dữ liệu từ nguồn đến đích
	PUSH	Chuyển một từ dữ liệu từ nguồn đến đỉnh ngăn xếp
	POP	Chuyển một từ dữ liệu từ đỉnh ngăn xếp đến đích
Số học	ADD	Tính tổng của hai toán hạng
	ADD WITH CARRY	Tính tổng của hai toán hạng và bit nhớ
	SUBTRACT	Tính hiệu của hai toán hạng
	MULTIPLY	Tính tích của hai toán hạng
	DIVIDE	Tính thương số của hai toán hạng và phần dư
	NEGATE	Thay đổi dấu toán hạng
	INCREMENT	Cộng thêm 1 vào toán hạng
	DECREMENT	Trừ bớt 1 từ toán hạng

Logic	AND	Thực hiện phép VÀ logic mức bit
	OR	Thực hiện phép HOẶC logic mức bit
	NOT	Thực hiện phép ĐẢO logic mức bit
	EXCLUSIVE OR	Thực hiện phép EXCLUSIVE OR logic mức bit
	LOGICAL SHIFT	Dịch toán hạng sang trái (phải), thêm 0 vào cuối
	ROTATE	Quay vòng trái (phải) toán hạng
Điều khiển chương trình	JUMP	Rẽ nhánh chương trình không điều kiện. Nạp PC một địa chỉ xác định
	JUMP CONDITIONAL	Kiểm tra điều kiện. Nếu thoả mãn, nạp PC một địa chỉ xác định
	CALL	Gọi chương trình con. Cất giữ nội dung PC (và các thông tin trạng thái), nạp PC địa chỉ chương trình con
	RETURN	Khôi phục nội dung PC và các thông tin trạng thái

Khuôn dạng lệnh

Mỗi một lệnh được thể hiện bằng một *chuỗi các bit*. Lệnh được chia thành các trường tương ứng với các thành phần của lệnh. Kiểu sắp xếp này được gọi là khuôn dạng lệnh.

Ví dụ

Mã thao tác	Địa chỉ toán hạng	Địa chỉ toán hạng

Phần lớn các tập lệnh đều có nhiều khuôn dạng lệnh khác nhau. Tập lệnh máy thể hiện các thao tác cơ bản, các phép tính cơ bản, bao gồm cả việc chuyển dữ liệu giữa các thanh ghi của đơn vị xử lý trung tâm CPU. Việc trình bầy và phân tích về lệnh máy tính (còn gọi là *ngôn ngữ máy*) trên cơ sở mô tả lệnh ở dạng mã nhị phân là rất khó, nên người ta thường sử dụng kiểu *mô tả tượng trưng* có tính gợi nhớ, còn gọi là *ngôn ngữ máy tượng trưng*, ví dụ:

ADD lệnh cộng
SUB lệnh trừ
MUL lệnh nhân
LOAD lệnh đọc dữ liệu từ bộ nhớ vào CPU

Các toán hạng cũng được mô tả theo cách này, ví dụ: ADD ACC, M

Số lượng trường địa chỉ

Các lệnh số học và logic yêu cầu nhiều toán tử nhất, một hoặc hai toán tử. Với loại lệnh này cần tối đa hai địa chỉ quy chiếu đến các toán tử. Kết quả của phép tính cũng cần được lưu lại, cần đến địa chỉ thứ ba. Sau khi thực hiện xong lệnh hiện thời, lệnh tiếp theo sẽ được nhập và cũng cần địa chỉ cho việc này. Trong thực tế phần lớn các ĐVXLTT được thiết kế có một, hai và ba địa chỉ. Địa chỉ lệnh tiếp theo được thể hiện không tường minh, nhận được từ bộ đếm chương trình (con trỏ lệnh) PC. Dạng lệnh ba địa chỉ cũng ít được sử dụng, vì nó yêu cầu khuôn dạng lệnh tương đối dài. Có một số lệnh máy khi mô tả ở dạng ngôn ngữ tượng trưng không có phần địa chỉ, khi đó vị trí nơi chứa toán hạng là ngầm định và thường là các thanh ghi của ĐVXLTT.

Cách sử dụng địa chỉ với các khuôn dạng lệnh khác nhau như sau:

Ký hiệu: OP mã thao tác

A, B, C địa chỉ nơi chứa toán hạng

ACC thanh ghi ACC

Số trường địa chỉ	**Mô tả lệnh (dạng tượng trưng)**	**Diễn giải**
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	ACC ← ACC OP A

Việc xác định số lượng trường địa chỉ trong lệnh ảnh hưởng cơ bản đến việc thiết kế đơn vị xử lý trung tâm.

Loại lệnh có nhiều trường địa chỉ sẽ cần nhiều thanh ghi dùng chung. Điều này cho phép nhiều phép tính chỉ thực hiện trên thanh ghi và do quy chiếu đến thanh ghi nhanh hơn bộ nhớ nên tốc độ thực hiện lệnh cũng tăng lên.

Loại lệnh có ít trường địa chỉ có độ dài ngắn hơn sẽ chỉ thực hiện được các thao tác đơn giản hơn. Chương trình xử lý dữ liệu sẽ phức tạp hơn, do vậy thời gian tính toán sẽ dài hơn. Có thể thấy điều này qua ví dụ sau.

Ví dụ: cần tính biểu thức K = (X+Y)*(M-N)

Các chương trình thực hiện tính biểu thức này trên cơ sở các dạng lệnh 2 trường địa chỉ và 1 trường địa chỉ như sau:

Ký hiệu

K, X, Y, M, N là địa chỉ bộ nhớ.

ACC, B là các thanh ghi của ĐVXLTT

a) Lệnh có 2 trường địa chỉ

MOV ACC, X	ACC ← X
ADD ACC, Y	ACC ← ACC + Y
MOV B, M	B ← M
SUB B, N	B ← B - N
MUL ACC, B	ACC ← ACC * B
MOV K, ACC	K ← ACC

b) Lệnh có 1 trường địa chỉ

LOAD X	ACC ← X
ADD Y	ACC ← ACC + Y
STORE K	K ← ACC
LOAD M	ACC ← M
SUB N	ACC ← ACC - N
MUL K	ACC ← ACC * K
STORE K	K ← ACC

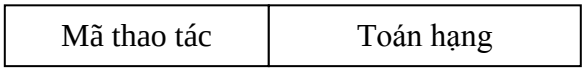
2.1.2. Các kiểu xác định địa chỉ toán hạng

Các kiểu xác định địa chỉ (định vị) toán hạng là các phương thức xác định vị trí chứa toán hạng, thông qua trường địa chỉ trong lệnh. Trường địa chỉ trong khuôn dạng lệnh có kích thước hạn chế. Thông qua phần địa chỉ nằm trong lệnh và kiểu định vị toán hạng có thể xác định rõ và tìm đến được các vị trí chứa toán hạng khác nhau trong máy tính.

Có nhiều kỹ thuật xác định địa chỉ toán hạng. Dưới đây là những kỹ thuật phổ biến nhất:

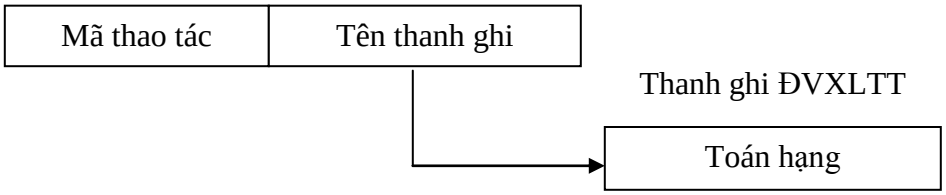
- Định vị tức thời
- Định vị thanh ghi
- Định vị trực tiếp
- Định vị gián tiếp
- Định vị gián tiếp thanh ghi
- Định vị cơ sở

- Định vị tức thời



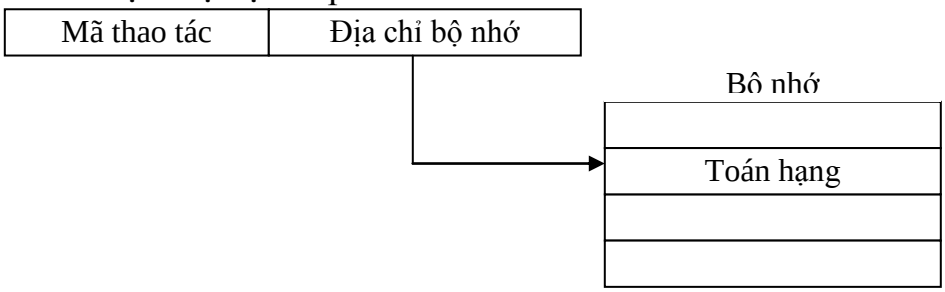
Định vị tức thời là chế độ định vị trong đó giá trị toán hạng nằm ngay trên lệnh.

- Định vị thanh ghi



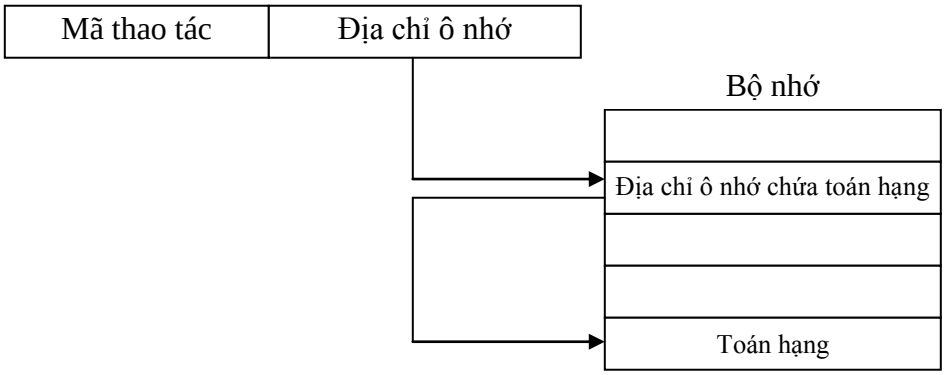
Ở kiểu định vị thanh ghi, các thanh ghi của đơn vị xử lý trung tâm là nơi chứa toán hạng. Tên thanh ghi chứa toán hạng được chỉ rõ trong lệnh.

- Định vị trực tiếp



Ở kiểu định vị trực tiếp, địa chỉ của ô nhớ chứa toán hạng nằm ngay trên lệnh.

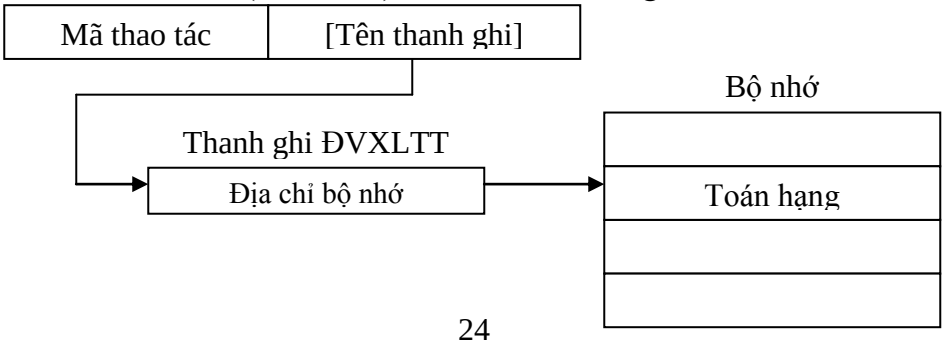
- Định vị gián tiếp



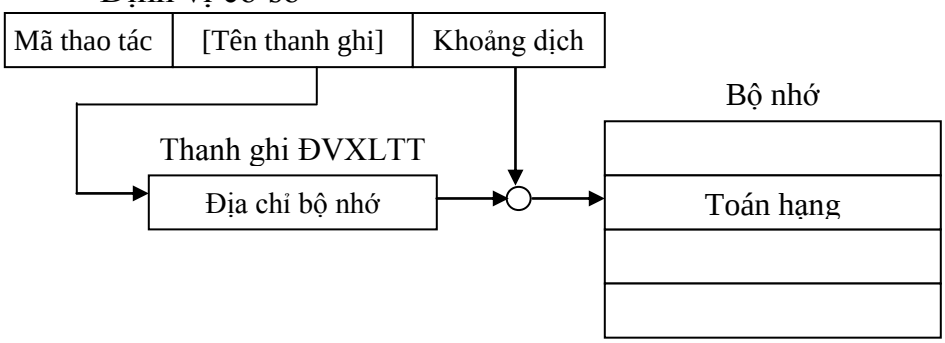
Phương thức xác định địa chỉ này được dùng khi địa chỉ toán hạng quá dài, vượt quá khả năng chứa của trường địa chỉ trên lệnh. Khi đó phần địa chỉ trên lệnh được dùng để xác định *vị trí nơi chứa địa chỉ* của toán hạng.

- Định vị gián tiếp thanh ghi

Kiểu định vị gián tiếp thanh ghi dùng các thanh ghi của đơn vị xử lý trung tâm làm nơi chứa (xác định) địa chỉ toán hạng.



- Định vị cơ sở



Trong kiểu định vị cơ sở, các thanh ghi của đơn vị xử lý trung tâm được dùng để chứa (xác định) địa chỉ nền vùng nhớ chứa toán hạng. Tên thanh ghi này được chỉ rõ trên lệnh. Phần địa chỉ lệch (địa chỉ offset), so với địa chỉ nền, của ô nhớ chứa toán hạng được chỉ rõ ngay trên lệnh. Có nhiều kiểu định vị khác được xây dựng trên kiểu định vị cơ sở.

2.1.3. CISC và RISC

CISC (Complex Instruction Set Computer).

Khi công nghệ mạch tích hợp cỡ lớn VLSI phát triển, giá thành phần cứng máy tính trở nên rẻ hơn, các nhà thiết kế máy tính đã đẩy mạnh thiết kế các máy tính có tập lệnh phức tạp (CISC). Với một tập lệnh phức tạp thì tổng số lệnh phải thực hiện trong mỗi chương trình sẽ giảm đi, do một lệnh phức tạp đơn có thể thay được nhiều lệnh đơn giản. Ví dụ lệnh nhân có thể thay cho một chương trình con thực hiện phép nhân, trong đó gồm nhiều lần lặp lại lệnh cộng và lệnh dịch. Việc giảm tổng số lệnh cũng dẫn đến giảm thời gian nhập lệnh và những thao tác khác liên quan đến bộ nhớ. Công nghệ VLSI tiên tiến cũng cho phép tạo ra các bộ vi xử lý mới, có khả năng thực hiện các lệnh mới, các loại dữ liệu mới và các kiểu xác định địa chỉ mới. Lý do cơ bản để thiết kế đơn vị xử lý trung tâm và máy tính có tập lệnh phức tạp là:

- Tận dụng hiệu năng làm việc của đơn vị xử lý trung tâm, do các phép tính phức tạp có thể được thực hiện bằng chuỗi các vi lệnh của đơn vị xử lý trung tâm.
- Đơn giản hóa nhiệm vụ của người viết ch- ơng trình dịch.
- Cung cấp hỗ trợ cho cả các ngôn ngữ lập trình cấp cao và phức tạp.

Các máy tính dòng IBM System 360-370, VAX 11/780, Motorola 680X0... là những máy tính loại CISC. Các bộ vi xử lý 80x86/Pentium của Intel và những máy tính được thiết kế trên cơ sở các vi xử lý 80x86/Pentium của Intel là những minh chứng điển hình cho xu hướng thiết kế máy tính CISC. Ban đầu, chip vi xử lý 8086 của Intel chỉ xử lý được dữ liệu 16 bit và không có lệnh xử lý số dấu chấm động, đến nay các vi xử lý họ Pentium đã có

khả năng xử lý các từ dữ liệu 32 bit hoặc 64 bit, với một tập lệnh đến trên 240 lệnh, trong đó có đầy đủ các lệnh dấu chấm động.

Tập lệnh CISC có một số đặc điểm như:

- Số lượng lệnh lớn. Ví dụ, tập lệnh IBM 370/168 có 208 lệnh, tập lệnh của Pentium có 242 lệnh.
- Khuôn dạng và kích thước lệnh thay đổi. Kích thước lệnh của IBM 370/168 từ 2 đến 6 byte, kích thước lệnh của Pentium từ 1 đến 12 byte.
- Tập các thanh ghi dùng chung hạn chế. Đơn vị xử lý trung tâm của IBM 370/168 có 16 thanh ghi dùng chung, các bộ vi xử lý họ Pentium có 8 thanh ghi dùng chung.
- Đơn vị điều khiển được thiết kế chủ yếu theo kiểu vi lập trình.
- Các lệnh khác nhau được thực hiện với số chu kỳ nhịp khác nhau.

Xu hướng thiết kế kiến trúc và tổ chức máy tính trong suốt nhiều năm là tăng độ phức tạp của đơn vị xử lý trung tâm theo hướng nhiều lệnh hơn, nhiều kiểu xác định địa chỉ hơn, nhiều thanh ghi chức năng đặc biệt hơn...

RISC (Reduced Instruction Set Computer)

Những mặt hạn chế của CISC được John Cocke và các đồng nghiệp ở IBM nhận ra lần đầu tiên vào giữa những năm 1970. Họ đã đề xuất một hướng thiết kế kiến trúc và tổ chức máy tính khác, đối lập với thiết kế CISC. Mục đích của họ là tạo ra một máy tính có tập lệnh tương đối nhỏ với các lệnh đơn giản, có thể thực hiện lệnh cực nhanh. Chiếc máy tính đầu tiên được chế tạo theo hướng này là máy tính IBM 801. Các máy tính được thiết kế về sau theo hướng này được gọi là máy tính loại RISC (Reduced Instruction Set Computer).

Kiến trúc RISC có những đặc điểm như sau:

- Tương đối ít lệnh và kiểu xác định địa chỉ. Ví dụ, đơn vị xử lý trung tâm MIPS R4000 có 94 lệnh và 1 kiểu xác định địa chỉ, Motorola 88000 có 51 lệnh và 4 kiểu xác định địa chỉ.
- Khuôn dạng lệnh cố định và dễ giải mã. MIPS R4000 có khuôn dạng lệnh cố định 32 bit, Motorola 88000 có khuôn dạng lệnh cố định 4 byte. Các lệnh của bộ vi xử lý ARM6 có kích thước cố định 4 byte.
- Đơn vị điều khiển cứng hóa.
- Thực hiện lệnh trong một chu kỳ nhịp.
- Truy cập bộ nhớ giới hạn ở các lệnh Load và Store.
- Sử dụng chương trình dịch để tối ưu hóa hiệu năng làm việc.

Tập lệnh có kích thước tương đối nhỏ và khuôn dạng lệnh cố định làm đơn giản hóa việc thiết kế đơn vị điều khiển, do vậy đơn vị điều khiển thường được thiết kế cứng hóa. Điều này lại tạo thuận lợi cho việc thực hiện lệnh nhanh trong một chu kỳ nhịp.

Mã chương trình được dịch từ máy tính RISC thường có nhiều lệnh hơn so với mã chương trình CISC, nhưng có thể được thực hiện hiệu quả hơn. Tuy nhiên với những chương trình có tần số xuất hiện các phép tính phức tạp cao thì các máy tính CISC lại xử lý hiệu quả hơn so với RISC.

Tranh luận và đánh giá về ưu nhược điểm của hai công nghệ CISC và RISC hiện vẫn chưa kết thúc.

2.2. KHỐI XỬ LÝ DỮ LIỆU

Khối xử lý dữ liệu, còn được gọi là đơn vị đường dữ liệu (Datapath Unit) hoặc đơn vị thực hiện (Execution Unit), có các thành phần chính là đơn vị số học-logic ALU và các thanh ghi dữ liệu, cùng các đường kết nối và truyền dữ liệu.

Đơn vị số học-logic ALU (Arithmetic-Logic Unit) thực hiện các thao tác số học và logic trên dữ liệu đầu vào. Các phép tính cộng, trừ, nhân, chia số nguyên và các thao tác cơ bản khác như cộng logic, nhân logic, dịch bit v.v. được thực hiện ở đây. Đơn vị điều khiển điều khiển các thao tác của ALU thông qua các tín hiệu điều khiển (là các tín hiệu điện).

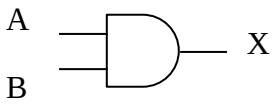
Khối xử lý dữ liệu có thành phần lưu trữ dữ liệu là tập các thanh ghi dữ liệu, được tạo thành từ các mạch lật và mạch logic. Các thanh ghi này gồm thanh tích luỹ (accumulator) và các thanh ghi dùng chung khác (general registers). Các thanh ghi này được dùng để tạm chứa các toán hạng và kết quả phép tính trong khi đang thực hiện lệnh và thực hiện chương trình.

2.2.1. Đơn vị logic

Đơn vị logic thực hiện các thao tác xử lý logic. Các thao tác logic được thực hiện trên từng bit của mỗi từ dữ liệu. Đơn vị logic được xây dựng trên cơ sở các mạch logic cơ bản như mạch AND (nhân logic), OR (cộng logic), XOR (cộng modulo 2), NOT (mạch đảo).

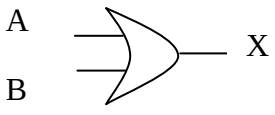
Mạch AND (nhân logic)

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



Mạch OR (cộng logic)

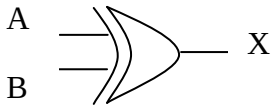
A	B	A OR B
0	0	0
0	1	1



1	0	1
1	1	1

Mạch XOR (cộng modulo 2)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

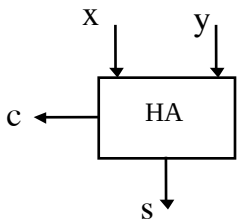


2.2.2. Đơn vị số học

a) Bộ cộng half-adder (HA)

Bộ cộng half-adder HA là một phần tử cơ bản của ALU.

Vào		Ra	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



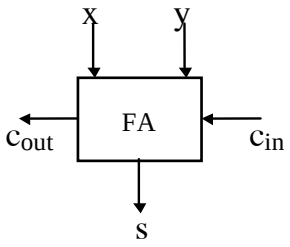
x, y - các bit đầu vào

s - tổng

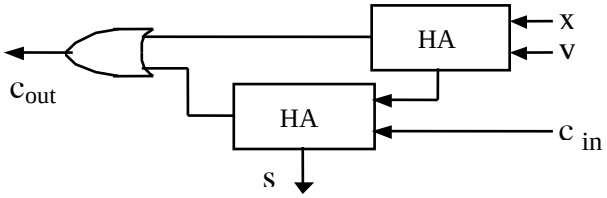
c - bit nhớ

b) Bộ cộng full-adder (FA) 1-bit

Vào			Ra	
x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



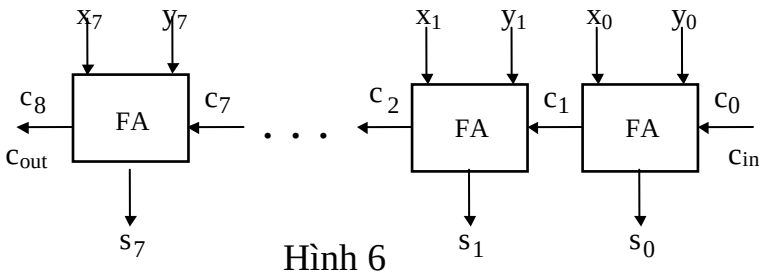
Bộ cộng FA-1-bit được xây dựng trên cơ sở các bộ HA-1-bit (Hình 5).



Hình 5

c) Bộ cộng FA-n-bit

Bộ cộng FA-n-bit được xây dựng trên cơ sở *n* bộ FA-1-bit (Hình 6).

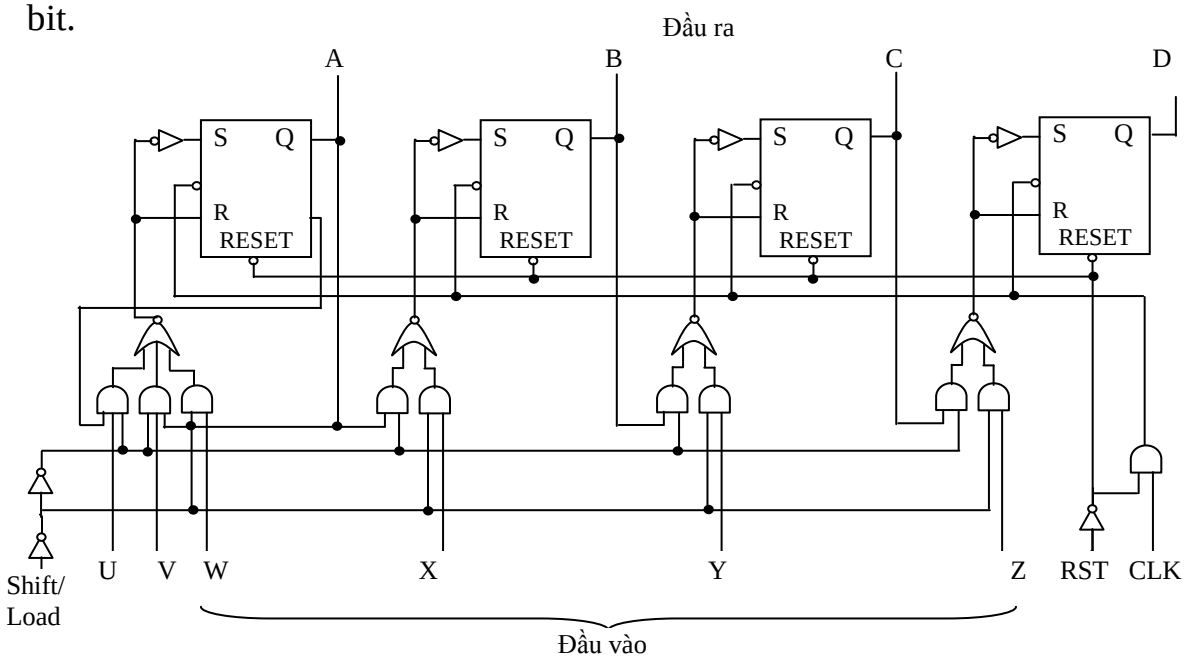


Hình 6

Bộ cộng FA-n-bit loại này được gọi là Ripple Carry Adder (ripple-gợn sóng), do phép cộng chỉ hoàn thành khi bit nhớ *c* từ bit thấp nhất lan truyền đến được vị trí bit cao nhất.

2.2.3. Bộ ghi dịch

Thao tác dịch bit thực hiện dịch chuyển các chữ số nhị phân trong một thanh ghi đến một vị trí khác trong thanh ghi đó. Có hai thao tác dịch chuyển là dịch trái và dịch phải. Bộ ghi dịch (Shifter) được dùng trong các phép nhân, chia và khi thực hiện các lệnh dịch bit. Hình 7 là một ví dụ về bộ ghi và dịch 4 bit.

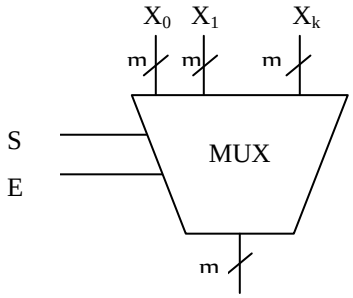


Hình 7

2.2.4. Bộ dồn kênh

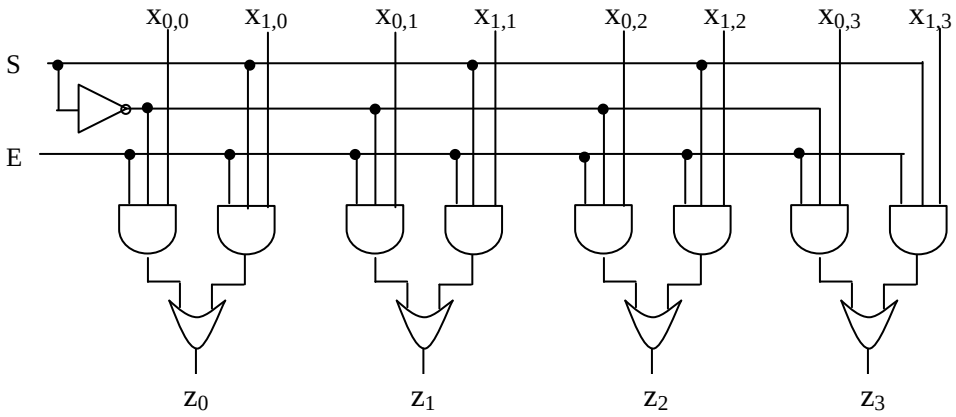
Bộ dồn kênh (Multiplexer - MUX) là thiết bị chọn một dữ liệu từ một vài nguồn (đầu vào) chuyển đến một đích chung (đầu ra). Nếu số đường dữ liệu

vào là *k*, trong đó mỗi đường dữ liệu là *m-bit*, thì đây là bộ dồn kênh *k-đầu vào, m*-bit (Hình 8).



Hình 8

Hình 9 là một ví dụ về bộ dồn kênh 2-đầu vào, 4-bit

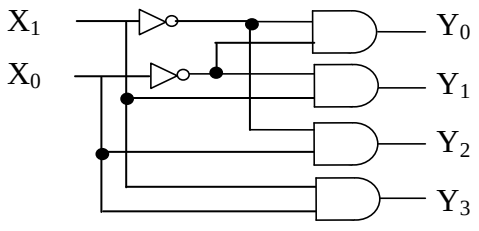


Hình 9

2.2.5. Bộ giải mã

Bộ giải mã (Decoder) là mạch tổ hợp có n-đầu vào/m-đầu ra, trong đó chỉ có một trong số các đầu ra tương ứng với một giá trị cụ thể của đầu vào. Bộ giải mã được dùng để giải mã thông tin mang tải bởi các con số. Một trong các ứng dụng bộ giải mã là từ con số địa chỉ ô nhớ, qua bộ giải mã xác định được vị trí vật lý của ô nhớ trong bộ nhớ.

Hình 10 dưới đây là bộ giải mã 2-đầu vào/4-đầu ra.

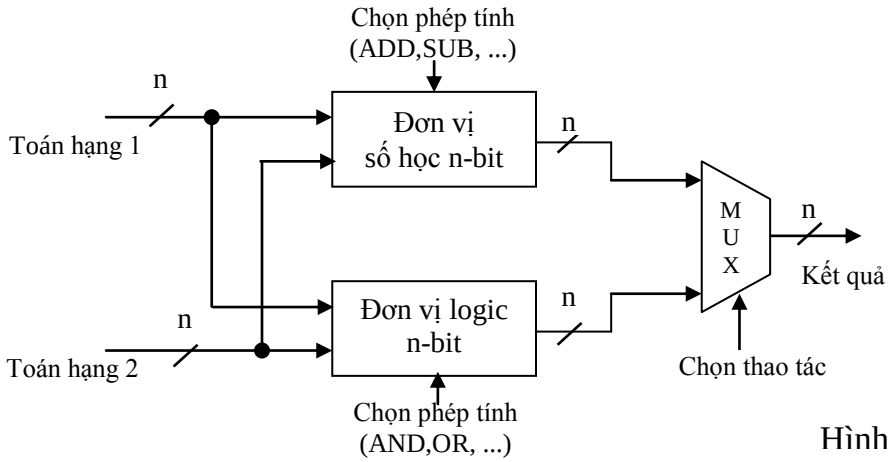


Hình 10

X_1	X_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

2.2.6. Đơn vị số học-logic ALU

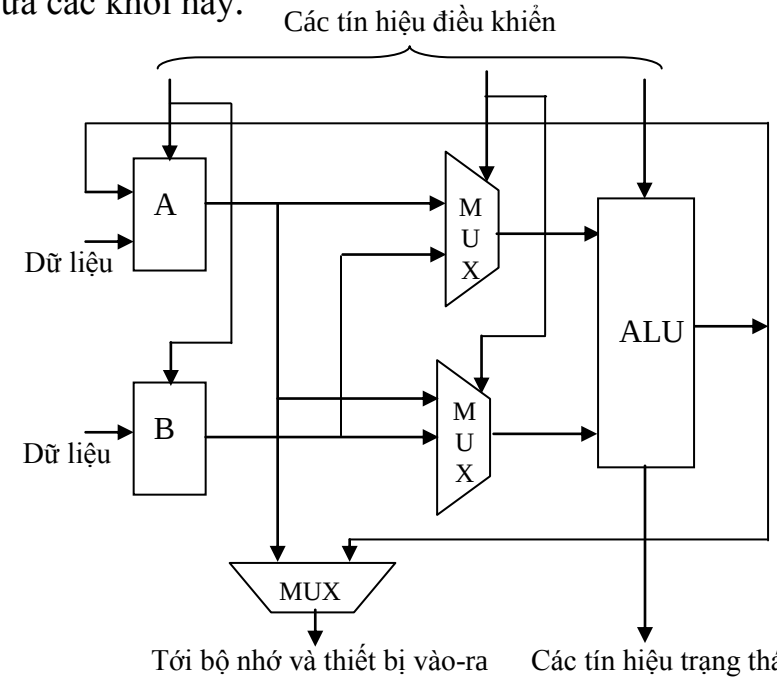
Đơn vị số học-logic ALU (Hình 11) thực hiện các thao tác số học và logic trên dữ liệu đầu vào. Các phép tính cộng, trừ, nhân, chia số nguyên và các thao tác cơ bản khác như cộng logic, nhân logic, dịch bit v.v. được thực hiện ở đây. Đơn vị số học-logic ALU cơ bản được xây dựng trên các mạch logic, bộ cộng, bộ dịch chuyển bit, bộ dồn kênh và một số mạch chức năng khác. Với cách biểu diễn số âm qua số bù 2, có thể thực hiện các phép tính trừ, nhân và chia chỉ qua các phép cộng và phép dịch bit. Để nâng cao tốc độ xử lý, các phép tính trừ, nhân và chia số nguyên cũng có thể được thực hiện bằng phần cứng, khi đó ALU còn chứa cả các bộ trừ, nhân và chia số nguyên.



Hình 11

2.2.7. Khối xử lý dữ liệu

Hình 12 mô tả cấu trúc của khối xử lý dữ liệu với các thành phần cơ bản là đơn vị số học-logic ALU, tập các thanh ghi A, B và các đường truyền dữ liệu giữa các khối này.



Hình 12

Hoạt động chuyển dữ liệu đến tập thanh ghi, từ tập thanh ghi đến ALU, việc thực hiện các phép tính số học và logic trên ALU, việc chuyển kết quả phép tính từ ALU đến thanh ghi tích lũy A đều được điều khiển bởi các tín hiệu từ đơn vị điều khiển. Sơ đồ khối xử lý dữ liệu cũng mô tả các đường tín hiệu điều khiển từ đơn vị điều khiển CU tới khối xử lý dữ liệu, các tín hiệu trạng thái từ ALU đến CUvà các đường truyền dữ liệu từ khối xử lý dữ liệu đến tới các thiết bị bên ngoài CPU như bộ nhớ và các thiết bị vào-ra.

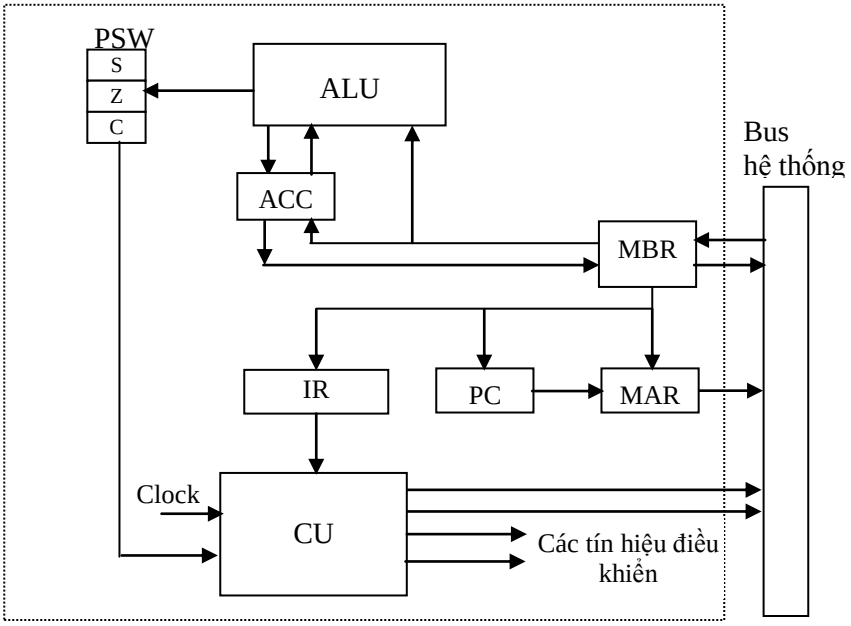
2.3. ĐƠN VỊ ĐIỀU KHIỂN

2.3.1. Chu kỳ lệnh

Chức năng của máy tính là thực hiện chương trình. Thực hiện chương trình thực chất là thực hiện một chuỗi lệnh máy kế tiếp nhau. Mỗi một lệnh máy được thực hiện trong một *chu kỳ lệnh*. Mỗi chu kỳ lệnh bao gồm các tiểu chu kỳ như *chu kỳ nhập lệnh*, *chu kỳ gián tiếp*, *chu kỳ thực hiện lệnh* và *chu kỳ ngắt*, trong đó chu kỳ thực hiện lệnh bao gồm các giai đoạn giải mã lệnh, tạo địa chỉ toán hạng và nhập toán hạng (nếu cần), thực thi lệnh và ghi kết quả. Mỗi một tiểu chu kỳ lại bao gồm các thao tác nhỏ hơn, được gọi là *vi thao tác*. Vi thao tác là thao tác cơ bản nhất của đơn vị xử lý trung tâm. *Đơn vị điều khiển có chức năng gây ra chuỗi các vi thao tác này*.

- Chu kỳ nhập lệnh

Chu kỳ nhập lệnh là tiểu chu kỳ đầu tiên trong chu kỳ lệnh. Lệnh được nhập trong chu kỳ này và từ nơi PC trỏ đến. Trong đơn vị xử lý trung tâm CPU (Hình 13) có 4 thanh ghi tham gia trong chu kỳ này là PC, MAR, MBR và IR.



Hình 13

Chu kỳ nhập lệnh gồm ba bước và bốn vi thao tác, mỗi vi thao tác thực hiện chuyển một dữ liệu vào hoặc ra một thanh ghi.

t1: MAR ← PC
t2: MBR ← bộ nhớ
t3: PC ← PC + 1
IR ← MBR

Mỗi một vi thao tác được thực hiện trong một khoảng thời gian, xác định bằng một chu kỳ nhịp đồng hồ hệ thống, gọi là đơn vị thời gian hệ thống *t*. Ký hiệu t1, t2, t3 mô tả các đơn vị thời gian kế tiếp nhau. Trong chu kỳ nhập lệnh, nội dung PC được đưa ra MAR, xác định vị trí ô nhớ chứa lệnh. Nội dung ô nhớ chứa lệnh (lệnh máy) được nhập và chuyển đến thanh ghi lệnh IR của đơn vị xử lý trung tâm. Đồng thời nội dung của con trỏ lệnh PC tăng thêm 1, trỏ đến ô chứa lệnh tiếp theo, chuẩn bị cho chu kỳ nhập lệnh tiếp theo.

- Chu kỳ gián tiếp

Giả định ta đang khảo sát việc thực hiện loại lệnh có một phần địa chỉ. Trong trường hợp này sẽ có hai kiểu xác định địa chỉ toán hạng có thể được dùng, là trực tiếp và gián tiếp. Nếu mã lệnh xác định kiểu địa chỉ gián tiếp thì chu kỳ gián tiếp sẽ được thực hiện. Chu kỳ gián tiếp sẽ gồm các vi thao tác sau:

t1: MAR ← (phần địa chỉ của lệnh)
t2: MBR ← bộ nhớ
t3: MAR ← MBR

Trong chu kỳ gián tiếp, phần địa chỉ của lệnh được dùng để nhập địa chỉ của toán hạng. Sau khi địa chỉ toán hạng được nhập vào và chuyển đến MAR, toán hạng sẽ được nhập. Chu kỳ tiếp theo sẽ là chu kỳ thực hiện lệnh.

- Chu kỳ thực hiện lệnh

Chu kỳ thực hiện lệnh bao gồm nhiều vi thao tác như giải mã lệnh, tạo địa chỉ toán hạng, nhập toán hạng, thực thi lệnh và ghi kết quả. Trình tự và loại vi thao tác được thực hiện phụ thuộc vào từng lệnh cụ thể.

Sau đây là một vài ví dụ về các vi thao tác trong chu kỳ thực hiện lệnh.

Ví dụ 1: ADD X ;

Lệnh ADD X thực hiện cộng nội dung ô nhớ có địa chỉ X với ACC, kết quả chứa vào ACC, với giả thiết toán hạng thứ nhất đã được nhập vào ACC.

Sau giải mã lệnh, các vi thao tác được tuần tự thực hiện như sau:

t1: MAR ← (phần địa chỉ của lệnh - X)
t2: MBR ← M(MAR) ; Ký pháp M(MAR) mô tả ô nhớ M cần truy nhập có địa chỉ xác định bởi MAR
t3: ACC ← ACC + MBR

Ví dụ 2: STORE X ;

Lệnh STORE X thực hiện sao lưu nội dung ACC ra ô nhớ có địa chỉ X. Sau giải mã lệnh, các vi thao tác được tuần tự được thực hiện như sau:

t1: MAR ← (phần địa chỉ của lệnh - X)

t2: MBR ← ACC

t3: M(MAR) ← MBR

- Chu kỳ ngắt

Cuối chu kỳ thực hiện lệnh, đơn vị xử lý trung tâm (ĐVXLTT) luôn kiểm tra xem có sự kiện báo ngắt nào không. Nếu có báo ngắt thì chu kỳ ngắt được thực hiện. Trong chu kỳ ngắt một chuỗi các vi thao tác được thực hiện, nhằm bảo vệ địa chỉ trở về và chuyển sang chương trình con phục vụ ngắt.

t1: MBR ← PC

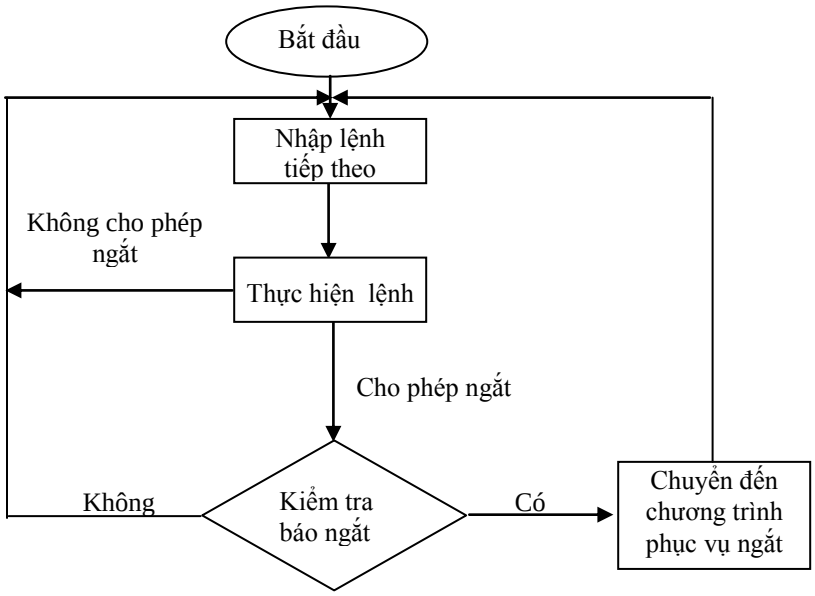
t2: MAR ← Địa chỉ của nơi cất giữ thông tin (địa chỉ đỉnh ngăn xếp)

PC ← Địa chỉ chương trình con phục vụ ngắt

t3: Nơi cất giữ thông tin (đỉnh ngăn xếp) ← MBR

Sau vi thao tác ở bước *t3* ĐVXLTT sẵn sàng bắt đầu chu kỳ nhập lệnh tiếp theo, từ chỗ PC trỏ đến. Đây là nơi chứa lệnh đầu tiên của chương trình con phục vụ ngắt.

Dưới đây là lưu đồ một chu kỳ lệnh (Hình 14)



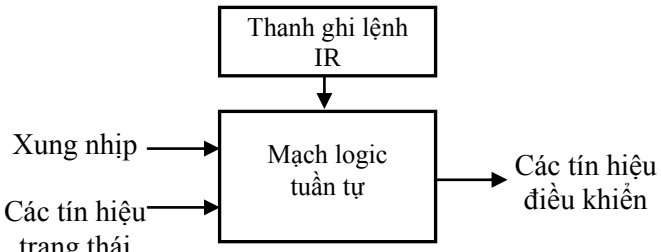
Hình 14

2.3.2. Phân loại đơn vị điều khiển

Chức năng của *đơn vị điều khiển* (*control unit* - CU) là giải mã lệnh, sinh ra chuỗi các tín hiệu điều khiển và điều phối hoạt động của các khối chức năng khác nhau trong máy tính, nhằm thực hiện được lệnh máy. Lệnh máy được thực hiện bởi một tập hợp nhiều nhóm vi thao tác. Mỗi một vi thao tác liên quan đến một tín hiệu điều khiển. Các tín hiệu điều khiển này được hoạt hoá theo một trình tự quy định để kích hoạt các vi thao tác. Ví dụ, việc chuyển nội dung bộ đếm chương trình PC vào thanh ghi MAR là một vi thao tác. Việc tăng nội dung PC thêm 1 cũng là một vi thao tác. Mỗi một vi thao tác đều được kích hoạt bởi một tín hiệu điều khiển được phát ra từ đơn vị điều khiển CU, bằng cách CU phát tín hiệu này ở mức *tích cực* lên đường truyền.

Có hai phương pháp thiết kế và tạo đơn vị điều khiển. Cách thứ nhất là xem đơn vị điều khiển như là một mạch logic tuần tự, nơi tạo ra chuỗi các tín hiệu điều khiển ở đầu ra của mạch logic tuần tự tương ứng với mã lệnh ở đầu vào mạch logic tuần tự (Hình 15).

Phương pháp thiết kế này được áp dụng khi mục tiêu là tốc độ thao tác của đơn vị điều khiển. Một khi đơn vị điều khiển đã được thiết kế và chế tạo thì khó có thể thay đổi hoặc thêm bớt chức năng của nó, vì nếu muốn thế thì phải thực hiện quy trình thiết kế lại từ đầu. Chính vì vậy cách tiếp cận này được gọi là *cứng hoá*.



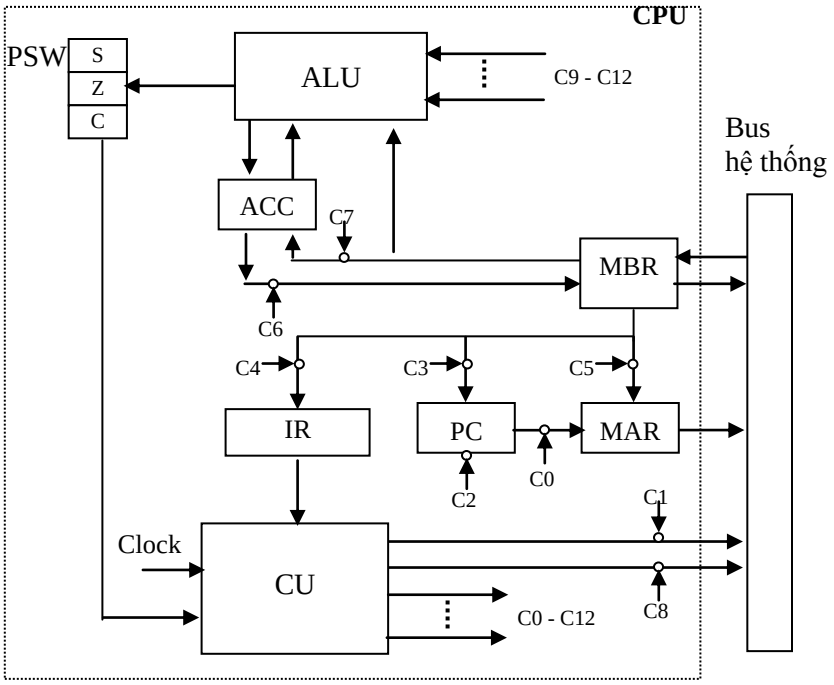
Hình 15

Một cách tiếp cận khác trong thiết kế đơn vị điều khiển là *vi lập trình* (*microprogramming*). Thành phần cốt lõi của *đơn vị điều khiển được vi lập trình* (*microprogrammed control unit*) là một *bộ nhớ điều khiển (control memory - CM)*, trong đó chứa tập các vi chương trình được thiết kế để thực hiện hành vi của lệnh. Mỗi một lệnh máy làm kích hoạt một đoạn vi chương trình và từ đó các tín hiệu điều khiển tương ứng được tạo ra. Cách tiếp cận theo vi lập trình làm cho việc thiết kế đơn vị điều khiển có tính hệ thống hơn, nhờ việc tổ chức tập các tín hiệu điều khiển thành các *vi lệnh*. Các tín hiệu điều khiển được nhúng (biểu hiện) trong các vi lệnh. Chuỗi các tín hiệu điều khiển được sinh ra từ *vi chương trình*. Với cách tiếp cận *vi lập trình*, việc thay

đổi chức năng và thiết kế của đơn vị điều khiển có thể được thực hiện khá dễ dàng bằng việc đổi nội dung của bộ nhớ điều khiển. Đơn vị điều khiển được vi lập trình cũng có nhược điểm là giá thành cao và có xu hướng chậm đi do cần có thời gian đọc vi lệnh từ bộ nhớ điều khiển.

2.3.3. Đơn vị điều khiển cứng hoá

Việc thiết kế đơn vị điều khiển CU cứng hóa khá phức tạp. Ở đây ta sẽ khảo sát quá trình thiết kế một đơn vị điều khiển CU cho đơn vị xử lý trung tâm sau (Hình 16).



Hình 16

Đơn vị xử lý trung tâm được thiết kế để thực hiện 10 lệnh (Bảng 2). Để đơn giản hóa việc thiết kế đơn vị điều khiển CU, các lệnh ADD, SUB, AND và OR được chọn là loại không có phần địa chỉ tường minh. Các toán hạng của các lệnh ADD, SUB, AND và OR được nhập vào trước bằng các lệnh LOAD và MOV1.

Bảng 2.

Loại lệnh	Lệnh dạng ngôn ngữ tượng trưng	Chức năng của lệnh
Chuyển dữ liệu	LOAD X	Nạp nội dung ô nhớ có địa chỉ X vào ACC
	STORE X	Cất giữ ACC vào ô nhớ có địa chỉ X
	MOV1 MBR, ACC	Sao chép nội dung ACC vào MBR
	MOV2 ACC, MBR	Sao chép nội dung MBR vào ACC

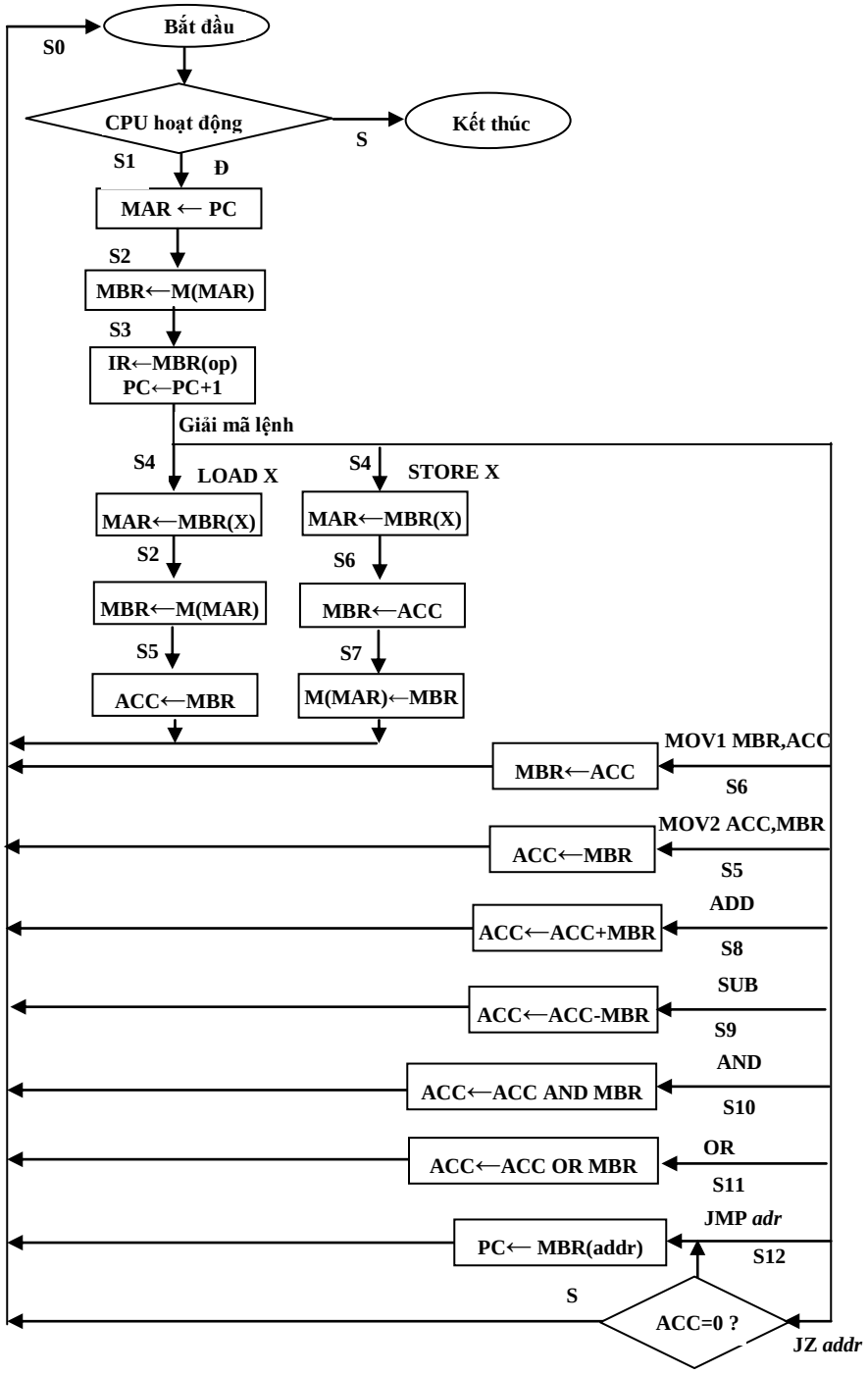
Xử lý dữ liệu	ADD SUB AND OR	Cộng MBR với ACC, kết quả cất giữ vào ACC Trừ MBR với ACC, kết quả cất giữ vào ACC AND MBR và ACC, kết quả cất giữ vào ACC OR MBR và ACC, kết quả cất giữ vào ACC
Điều khiển chương trình	JMP addr JZ addr	Nhảy đến lệnh có địa chỉ addr Nhảy đến lệnh có địa chỉ addr, nếu ACC=0

Quá trình thực hiện lệnh trải qua hai giai đoạn là nhập lệnh và thực hiện lệnh. Vào thời điểm bắt đầu (hoặc tái khởi động), con trỏ lệnh PC=0. Giai đoạn nhập lệnh sao chép nội dung PC sang MAR, sau đó thao tác đọc bộ nhớ được thực hiện, nhập lệnh từ ô nhớ có địa chỉ được xác định bởi MAR vào MBR. Sau đó phần mã thao tác được chuyển vào thanh ghi lệnh IR và được giải mã. Cùng lúc đó nội dung PC được tăng thêm 1, để trỏ dến ô nhớ chứa lệnh tiếp theo.

Hành trạng của đơn vị xử lý trung tâm CPU khi thực hiện các lệnh máy được mô tả trong lưu đồ trên hình 17. Lưu đồ hình 17 mô tả các bước thao tác (các vi thao tác) thực hiện lệnh. Các bước thao tác này xác định các tín hiệu điều khiển cần phải có (Bảng 3) và các điểm điều khiển Ci tương ứng (Hình 16) trong đơn vị xử lý trung tâm CPU. Ký hiệu "← " mô tả thao tác chép dữ liệu hoặc thông tin từ nguồn sang đích. Lưu đồ này cũng cho thấy, tiểu chu kỳ nhập lệnh cần 3 chu kỳ xung nhịp và tiểu chu kỳ thực hiện lệnh cần từ 1 đến 3 chu kỳ xung nhịp, tùy thuộc vào từng lệnh cụ thể.

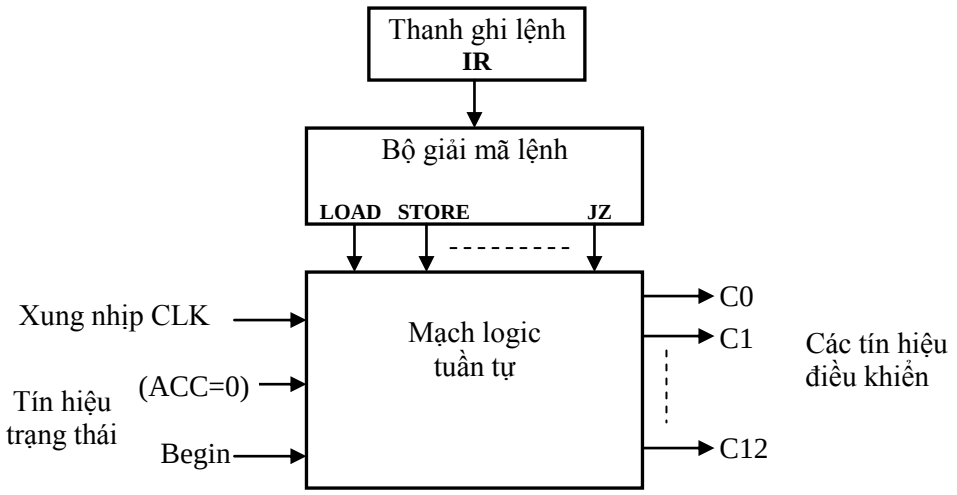
Bảng 3

Tín hiệu điều khiển	**Thao tác được điều khiển**
C0	MAR ← PC
C1	MBR ← M(MAR)
C2	PC ← PC+1
C3	PC ← MBR(addr)
C4	IR ← MBR(op)
C5	MAR ← MBR(addr)
C6	MBR ← ACC
C7	ACC ← MBR
C8	M(MAR) ← MBR
C9	ACC ← ACC + MBR
C10	ACC ← ACC - MBR
C11	ACC ← ACC AND MBR
C12	ACC ← ACC OR MBR



Hình 17

Đơn vị điều khiển cứng hoá, nơi sinh ra các tín hiệu điều khiển việc thực thi lệnh, có dạng sau (Hình 18):



Hình 18

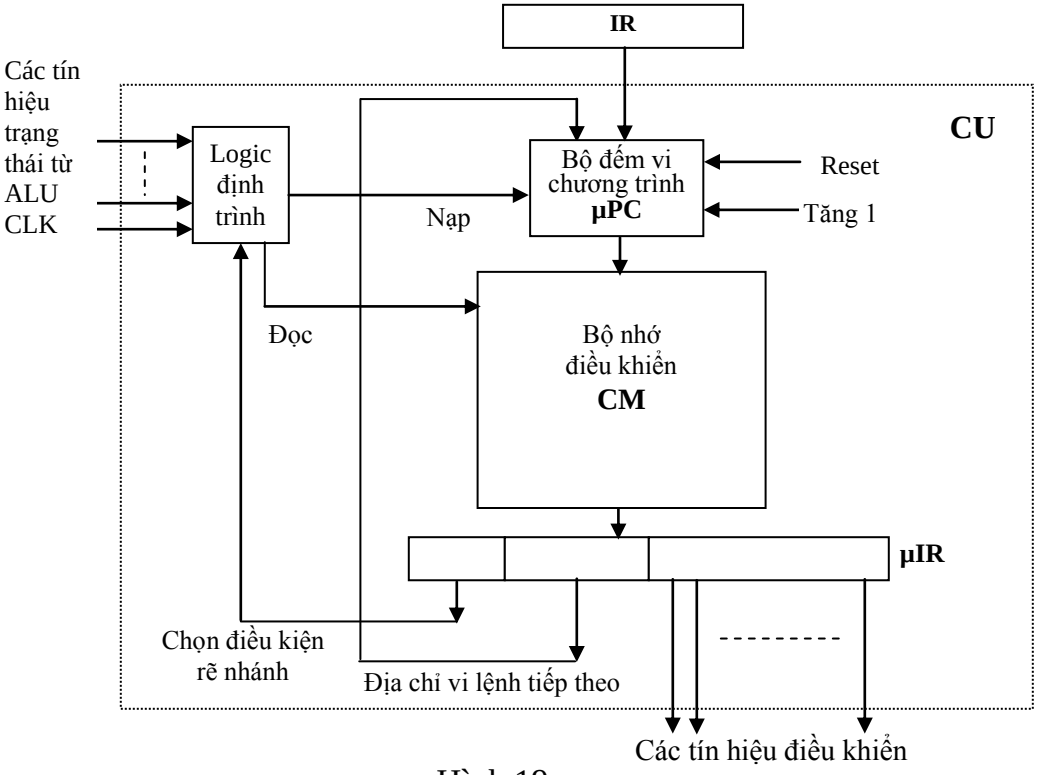
Mạch logic tuần tự trong đơn vị điều khiển có 12 đầu vào, trong đó 10 đầu vào là từ bộ giải mã 10 lệnh và 2 đầu vào mang thông tin trạng thái từ ALU và trạng thái khởi động. Đầu ra của mạch logic tuần tự là 13 tín hiệu điều khiển từ C0 đến C12. Số lượng trạng thái của mạch logic tuần tự cũng được xác định qua lưu đồ thực hiện lệnh (Hình 18), trong đó sau mỗi một trạng thái mạch logic Si là một vi thao tác của CPU. Do vậy mạch logic tuần tự này sẽ có 13 trạng thái từ S0 đến S12.

2.3.4. Đơn vị điều khiển vi lập trình

Việc *thực hiện lệnh máy* trong CPU được tiến hành qua việc *thực hiện chuỗi các vi thao tác*. Các *tín hiệu điều khiển* xác định chuỗi các vi thao tác thực hiện lệnh máy được tạo ra bởi phần cứng là mạch logic tuần tự. Kiểu thiết kế đơn vị điều khiển loại cứng hóa này được sử dụng khá rộng rãi và thích hợp với các đơn vị xử lý trung tâm CPU có chức năng không quá phức tạp. Khi số lượng lệnh máy và các đường điều khiển lên đến hàng trăm thì việc thiết kế đơn vị điều khiển cứng hoá sẽ rất khó khăn. Hơn nữa các thiết kế kiểu này thiếu tính linh hoạt, việc thay đổi thiết kế khá khó khăn khi ta muốn bổ xung các lệnh mới hoặc sửa lỗi.

Có một phương pháp thiết kế đơn vị điều khiển kiểu khác, có tính hệ thống và tính linh hoạt cao, đó là phương pháp *vi lập trình*. *Vi lập trình* là phương pháp thiết kế đơn vị điều khiển, trong đó tập các tín hiệu điều khiển và thông tin về trình tự vi thao tác được thể hiện trong các *vi lệnh (microinstruction)* và được chứa trong bộ nhớ ROM. Bộ nhớ này được gọi là *bộ nhớ điều khiển (control memory - CM)*. Các *tín hiệu điều khiển* (gây ra các vi thao tác thực thi lệnh máy) được xác định bởi *giá trị các bit trong vi lệnh*. Các tín hiệu điều khiển *được hoạt hoá* (có mức tích cực) bởi các bit tương ứng, khi vi lệnh được đọc ra từ bộ nhớ điều khiển CM. Như vậy *ở đơn vi điều*

khiển CU vi lập trình (Hình 19), *các tín hiệu điều khiển được tạo ra bởi đầu ra của bộ nhớ điều khiển CM*, thay cho việc tạo ra bởi các mạch logic như ở các đơn vị điều khiển cứng hoá. Tập hợp chuỗi các vi lệnh được sắp xếp có chủ đích tạo nên một *vi chương trình*. Trong đơn vị xử lý trung tâm vi lập trình (*microprogrammed CPU*), mỗi một lệnh máy sẽ được thực thi bởi một *đoạn vi chương trình*, hoạt động như một *bộ biên dịch* lệnh máy thời gian thực. Việc xác định chuỗi các vi thao tác để thực hiện một chức năng hoặc một lệnh máy nào đó được gọi là *vi lập trình* (*microprogramming*). Người xác định (viết) chuỗi các vi thao tác được gọi là *người viết vi chương trình (microprogrammer)*, sử dụng ngôn ngữ lập trình riêng gọi là *vi hợp ngữ (microassembly languages)* với các *vi lệnh (microinstruction)*. Trình biên dịch vi ngôn ngữ *(microassembler)* sẽ biên dịch vi chương trình thành chương trình khả thi để nạp vào CM.



Hình 19

Khuôn dạng đơn giản nhất của một vi lệnh gồm hai phần: *phần chứa thông tin điều khiển (trường điều khiển)* xác định các tín hiệu điều khiển cần được hoạt hoá và *phần chứa địa chỉ (trường địa chỉ)* xác định địa chỉ ô nhớ chứa vi lệnh tiếp theo trong CM. Trong *trường điều khiển*, cần *một bit cho mỗi tín hiệu điều khiển*, như vậy CU cần tối thiểu bao nhiêu tín hiệu điều khiển thì sẽ cần bấy nhiêu bit tương ứng trong vi lệnh. Trong vi lệnh với khuôn dạng phức tạp hơn còn có một số bit chỉ thị kiểm tra điều kiện rẽ nhánh vi chương trình (*trường điều kiện*). Trường điều kiện chứa các bit xác định điều kiện

ngoài (giá trị các bit từ thanh ghi trạng thái PSW – thanh ghi cờ FLAGS...) cần kiểm tra trong trường hợp thực hiện rẽ nhánh.
Khuôn dạng chung của một vi lệnh như sau:

Địa chỉ vi lệnh tiếp theo	Các bit xác định điều kiện rẽ nhánh	Các bit xác định tín hiệu điều khiển

Vi lệnh được biên dịch như sau:

- Các đường dây điều khiển được chỉ thị bằng bit 1 sẽ được đặt lên mức tích cực, các đường dây điều khiển được chỉ thị bằng bit 0 sẽ được đặt xuống mức không tích cực.
- Nếu điều kiện được chỉ định bởi các bit điều kiện sai, sẽ thực hiện vi lệnh ở ô kế tiếp.
- Nếu điều kiện được chỉ định bởi các bit điều kiện đúng, vị trí vi lệnh tiếp theo được chỉ định trong trường địa chỉ.

Tổ chức tổng quát của một vi chương trình trong bộ nhớ điều khiển CM có dạng sau:

Các vi lệnh (Thủ tục chu kỳ Nhập lệnh) Nhảy đến thủ tục Thực hiện lệnh hoặc Gián tiếp	Thủ tục chu kỳ Nhập lệnh
Các vi lệnh (Thủ tục chu kỳ Gián tiếp) Nhảy đến thủ tục Thực hiện lệnh	Thủ tục chu kỳ Gián tiếp
Các vi lệnh (Thủ tục chu kỳ Ngắt) Nhảy đến thủ tục Nhập lệnh	Thủ tục chu kỳ Ngắt
Giải mã lệnh máy và nhảy đến thủ tục Thực hiện lệnh máy	Bắt đầu chu kỳ Thực hiện lệnh
Các vi lệnh (Thủ tục Thực hiện lệnh ADD) Nhảy đến thủ tục Nhập lệnh hoặc Ngắt	Thủ tục Thực hiện lệnh ADD
Các vi lệnh (Thủ tục Thực hiện lệnh SUB) Nhảy đến thủ tục Nhập lệnh hoặc Ngắt	Thủ tục Thực hiện lệnh SUB
... .	**... ...**

Bộ nhớ điều khiển CM trong đơn vị điều khiển (Hình 19) là thành phần cơ bản trong đơn vị điều khiển CU vi lập trình. Bộ nhớ điều khiển CM chứa vi

chương trình mô tả hành trạng của đơn vị điều khiển.

Bộ đếm vi chương trình (con trỏ vi lệnh) μPC (Hình 19) được dùng để xác định vị trí (trỏ tới) ô nhớ chứa vi lệnh cần đọc trong CM. Vai trò của μPC tương tự như của bộ đếm chương trình (con trỏ lệnh) PC ở mức lệnh máy. Do chỉ có vi lệnh được đọc từ bộ nhớ CM, nên μPC cũng được dùng như một thanh ghi địa chỉ bộ nhớ điều khiển CMAR (CM Address Register).

Việc thực hiện vi chương trình (việc xác định trình tự đọc các vi lệnh từ CM) được thực hiện theo cơ chế tương tự như cơ chế thực hiện chương trình mức lệnh máy.

Đơn vị điều khiển vi lập trình hoạt động như sau: khởi đầu μPC=0.

- Logic định trình phát tín hiệu điều khiển đọc tới CM.
- Nội dung ô nhớ CM, có địa chỉ xác định bởi μPC, được đọc ra thanh ghi vi lệnh μIR.
- Nội dung μIR tạo ra các tín hiệu điều khiển và thông tin về địa chỉ vi lệnh tiếp theo cho đơn vị logic định trình.
- Logic định trình nạp địa chỉ CM mới vào μPC trên cơ sở thông tin địa chỉ vi lệnh tiếp theo từ μIR. Thao tác sau cùng này được thực hiện như thế nào phụ thuộc vào nội dung μIR và giá trị của các bit cờ trạng thái từ ALU. Có ba khả năng có thể xảy ra:

  + μPC tăng thêm 1. Vi lệnh trong ô nhớ tiếp theo sẽ được đọc.
  + Nạp μPC trên cơ sở mã thao tác OP trong IR. Thao tác này đóng vai trò giải mã lệnh máy, chuyển mã thao tác OP thành địa chỉ vi thủ tục thực thi lệnh máy này. Vi lệnh của vi thủ tục thực thi lệnh máy sẽ được đọc.
  + Nạp trường địa chỉ trong μIR vào μPC. Nhảy đến vi thủ tục (đoạn vi lệnh) mới.

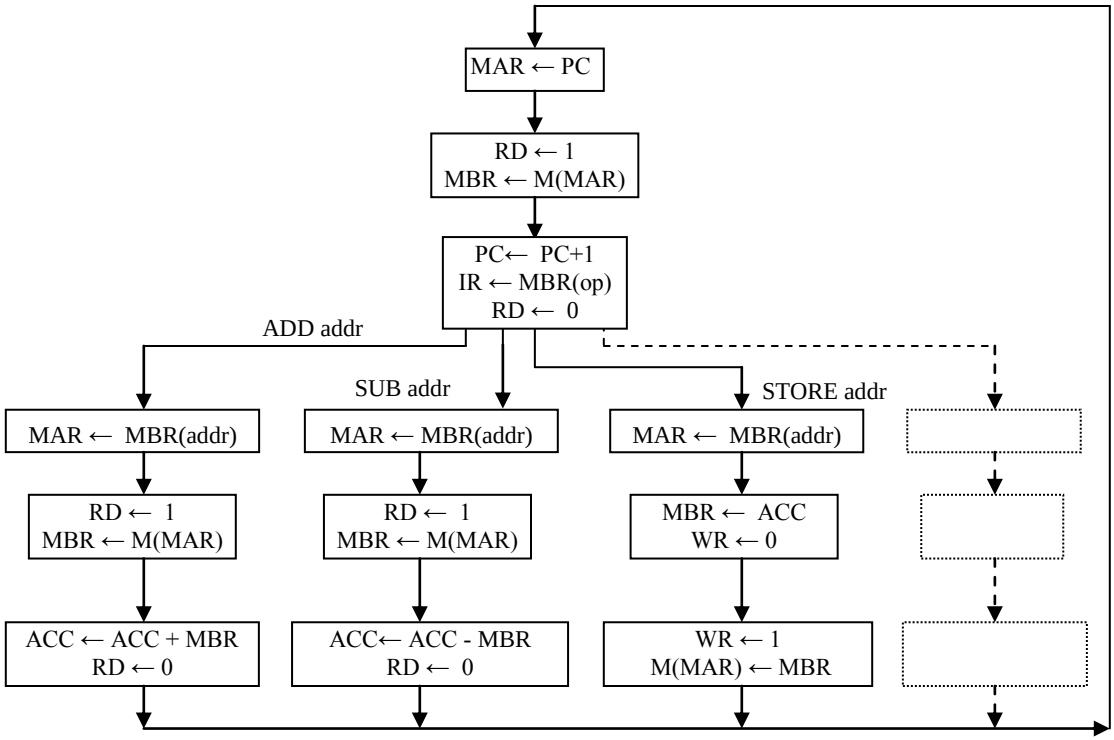
Toàn bộ các hoạt động trên diễn ra trong một xung nhịp đồng hồ hệ thống.

Ta khảo sát quá trình thiết kế và hoạt động của đơn vị điều khiển vi lập trình qua một ví dụ. Giả sử cần thiết kế một đơn vị điều khiển CU vi lập trình có khả năng điều khiển thực thi các lệnh máy trong hệ lệnh của CPU loại 16 bit như trên Hình 2 (Chương I).

Hệ lệnh máy có khuôn dạng gồm một trường mã thao tác và một trường địa chỉ. Trường mã thao tác OP nằm ở các bit cao nhất và có độ dài 5 bit, phần còn lại trong lệnh máy là trường địa chỉ. Ví dụ, mã lệnh ADD là 00000, mã lệnh SUB là 00001, mã lệnh STORE là 00010 v.v.

Quá trình nhập và thực hiện các lệnh máy của CPU được mô tả trong lưu đồ Hình 20. Để thực hiện được chuỗi các vi thao tác trong quá trình thực hiện

lệnh máy thì đơn vị điều khiển CU cần phải tạo ra các tín hiệu điều khiển tương ứng (xem Bảng 4).



Hình 20

Ta xây dựng vi lệnh có khuôn dạng đơn giản và có độ dài 27 bit. Trường địa chỉ trong vi lệnh có 7 bit, từ C0 đến C6, được dùng để xác định địa chỉ vi lệnh tiếp theo trong trường hợp cần thực hiện rẽ nhánh trong vi chương trình. Các bit còn lại từ C7 đến C26 là các bit điều khiển hoạt hoá các tín hiệu điều khiển (Bảng 4).

Bảng 4.

Vi thao tác	**Tín hiệu điều khiển và vị trí bit tương ứng trong vi lệnh**
PC ← 0	C7
MAR← PC	C8
RD←1; MBR←M(MAR)	C9
PC ← PC + 1	C10
PC ← MBR(addr)	C11
IR ← MBR(op)	C12
MAR ← MBR(addr)	C13
MBR ← ACC	C14

ACC← MBR	C15
WR←1; M(MAR) ← MBR	C16
ACC ← ACC + MBR	C17
ACC ← ACC - MBR	C18
ACC ← ACC AND MBR	C19
ACC ← ACC OR MBR	C20
RD ← 0	C21
WR ← 0	C22
µPC ← 0	C23
µPC ← µPC + 1	C24
µPC ← C_{0-6}	C25
µPC ← µPC + IR	C26

Ta tìm hiểu hoạt động của đơn vị điều khiển vi lập trình có bộ nhớ điều khiển CM chứa vi chương trình sau (xem Bảng 5). Khởi đầu, bộ đếm vi chương trình µPC được xác lập giá trị 0. Vi lệnh được đọc ra µIR. Các tín hiệu điều khiển được sinh ra sẽ kích hoạt các vi thao tác tương ứng. Kết qủa của việc thực hiện vi lệnh ở ô có địa chỉ 0 trong CM: Giá trị của PC, xác định vị trí ô nhớ chứa lệnh máy cần nạp, được chép sang MAR. Kết qủa của việc thực hiện vi lệnh ở ô có địa chỉ 1 trong CM: Lệnh máy được nạp từ bộ nhớ chính vào MBR. Kết qủa của việc thực hiện vi lệnh ở ô có địa chỉ 2 trong CM: Phần mã thao tác OP (5 bit đầu tiên trong lệnh máy) được nạp vào IR. Kết qủa của việc thực hiện vi lệnh ở ô có địa chỉ 3 trong CM: Giá trị OP này được cộng với µPC và cộng thêm 1, để xác định vị trí vi lệnh tiếp theo. Ví dụ, nếu lệnh máy được nhập là ADD và mã thao tác của lệnh ADD là 00000 thì vị trí vi lệnh tiếp theo được đọc là vị trí 4. Giả thiết ta đặt đoạn vi chương trình gây ra các thao tác thực thi lệnh ADD bắt đầu từ vị trí 30 trong CM, khi đó các bit C0-C6 (trường chứa thông tin về địa chỉ vi lệnh tiếp theo) của vi lệnh nằm tại vị trí 4 trong CM cần chứa giá trị là 30. Nếu lệnh máy được nhập là SUB và mã thao tác của lệnh SUB là 00001 thì vị trí vi lệnh tiếp theo được đọc là vị trí 5. Giả thiết ta đặt đoạn vi chương trình gây ra các thao tác thực thi lệnh SUB bắt đầu từ vị trí 33, khi đó các bit C0-C6 của vi lệnh nằm tại vị trí 5 trong CM cần chứa giá trị là 33 v.v. Như vậy lệnh ADD sẽ gây ra điều khiển rẽ nhánh đến vị trí 30, lệnh SUB sẽ gây ra điều khiển rẽ nhánh đến vị trí 33 trong vi chương trình v.v. Các thao tác nói trên thực chất đóng vai trò giải mã các lệnh máy.

Bằng cách tương tự, các vi lệnh trong CM tuần tự được đọc ra µIR. Từ đây các tín hiệu điều khiển được sinh ra và kích hoạt các vi thao tác thực thi lệnh máy.

Bảng 5

Vị trí ô nhớ trong CM	**Vi chương trình (Vi lệnh và các vi thao tác)**	**Ý nghĩa**
0	MAR ← PC ; µPC ← µPC + 1	Địa chỉ của lệnh máy tiếp theo được đặt vào MAR ; Tăng µPC
1	RD ← 1; MBR ← M(MAR) ; µPC ← µPC + 1;	Lệnh máy được nhập (đọc) từ bộ nhớ chính vào MBR ; Tăng µPC
2	PC ← PC+1; IR ←MBR(op) RD ← 0 ; µPC ← µPC + 1;	PC tăng thêm 1, trỏ đến lệnh máy tiếp theo. Phần mã lệnh OP được nạp vào IR ; Tăng µPC
3	µPC ← IR + µPC + 1;	*Giải mã lệnh máy*: Cộng mã thao tác OP trong lệnh máy với µPC + 1, xác định địa chỉ vi lệnh tiếp theo được sử dụng
4	µPC ← C_{0-6} ; // Nếu lệnh máy là ADD thì C_{0-6} có giá trị 30	*Điều khiển rẽ nhánh*. C_{0-6} xác định địa chỉ vi lệnh tiếp theo. Nếu lệnh máy là ADD thì nhảy đến vị trí thủ tục thực hiện lệnh ADD (ô nhớ 30 trong CM)
5	µPC ← C_{0-6} ; // Nếu lệnh máy là SUB thì C_{0-6} có giá trị 33	*Điều khiển rẽ nhánh*. C_{0-6} xác định địa chỉ vi lệnh tiếp theo. Nếu lệnh là SUB thì nhảy đến vị trí thủ tục thực hiện lệnh SUB (ô nhớ 33 trong CM)
6	µPC ← C_{0-6} ; // Nếu lệnh máy là STORE thì C_{0-6} có giá trị 36	*Điều khiển rẽ nhánh*. C_{0-6} xác định địa chỉ vi lệnh tiếp theo. Nếu lệnh là STORE thì nhảy đến vị trí thủ tục thực hiện lệnh STORE (ô nhớ 36 trong CM)
7	Các vi lệnh điều khiển rẽ nhánh khác	*Điều khiển rẽ nhánh*. Xác định địa chỉ vi lệnh tiếp theo. (Xác định vị trí thủ tục thực hiện lệnh máy đã được giải mã)
8		
⋮		

30	MAR ← MBR(addr) ; µPC ← µPC + 1;	*Bắt đầu Thủ tục gây ra các vi thao tác thực thi lệnh ADD*. Xác định địa chỉ toán hạng thứ 2 ; Tăng µPC
31	RD ← 1; MBR ← M(MAR) ; µPC ← µPC + 1;	Nhập toán hạng thứ 2 ; Tăng µPC
32	ACC ← ACC + MBR ; RD ← 0 ; µPC ← 0	Thực thi thao tác *cộng* và ghi kết quả vào ACC ; Nhảy về thủ tục Nhập lệnh (vị trí 0 trong CM)
33	MAR ← MBR(addr) µPC ← µPC + 1	*Bắt đầu Thủ tục gây ra các vi thao tác thực thi lệnh SUB*. Xác định địa chỉ toán hạng thứ 2 ; Tăng µPC
34	RD ← 1; MBR ← M(MAR) ; µPC ← µPC + 1	Nhập toán hạng thứ 2 ; Tăng µPC
35	ACC←ACC-MBR ; RD ← 0 ; µPC ← 0	Thực thi thao tác *trừ* và ghi kết quả vào ACC ; Nhảy về thủ tục Nhập lệnh (vị trí 0 trong CM)
36	MAR ← MBR(addr) ; µPC ← µPC + 1	*Bắt đầu Thủ tục gây ra các vi thao tác thực thi lệnh STORE*. Xác định địa chỉ ô nhớ dữ liệu cần lưu. Tăng µPC
37	MBR ← ACC ; WR ← 0 ; µPC ← µPC + 1	Sao chép ACC sang MBR ; Tăng µPC
38	WR ← 1 ; M(MAR) ← MBR ; µPC ← 0	Ghi dữ liệu vào bộ nhớ chính, tại vị trí ô nhớ xác định bởi MAR ; Nhảy về thủ tục Nhập lệnh (vị trí 0 trong CM)
39	*Các đoạn vi lệnh khác*	*Các thủ tục gây ra các vi thao tác thực thi các lệnh máy khác...*
⋮	⋮	

- **Định thời vi thao tác**

Cho đến giờ ta vẫn biết vi lệnh kích hoạt một tập các tín hiệu điều khiển. Một tín hiệu xung nhịp đơn đồng bộ hoá các tín hiệu điều khiển này và thời gian chu kỳ của tín hiệu xung nhịp này bằng với thời gian chu kỳ vi lệnh. Kiểu điều khiển này có tên gọi là *điều khiển đơn pha*. Có thể giảm số lượng các vi lệnh xác định các vi thao tác bằng cách chia chu kỳ vi lệnh thành các *các pha* nhỏ hơn. Các tín hiệu điều khiển được kích hoạt trong từng pha này. Kiểu điều khiển này được gọi là *điều khiển đa pha*. Thao tác đa pha cho phép một vi lệnh có thể xác định một chuỗi có trình tự các vi thao tác khác nhau trong một chu kỳ vi lệnh.

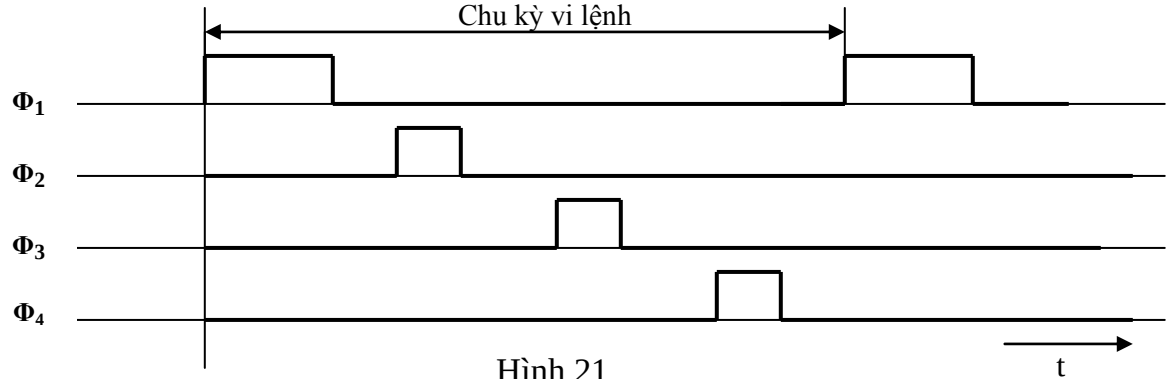
Ví dụ, một vi lệnh diều khiển thao tác sau:

$$ACC \leftarrow ACC + MBR$$

Thao tác này được thực hiện trong 4 pha như sau:

- Pha Φ_1: Đọc vi lệnh từ CM
- Pha Φ_2: Chuyển nội dung ACC và MBR đến đầu vào của ALU
- Pha Φ_3: ALU thực hiện phép tính
- Pha Φ_4: Chuyển kết quả từ ALU đến thanh ghi đích ACC

Hình 21 mô tả các tín hiệu định thời liên quan tới 4 pha này. Các tín hiệu định thời này được tạo ra từ tín hiệu nhịp chu kỳ vi lệnh bằng cách tạo các khoảng thời gian trễ xác định.



Hình 21

2.4. CHƯƠNG TRÌNH CON

Có một công cụ hữu ích cho việc thiết kế các chương trình máy tính, đó là *chương trình con* (*thủ tục*). Chương trình con là một module mã lệnh (module chương trình) độc lập có thể được gọi (kích hoạt) từ chương trình chính và việc này có thể được thực hiện nhiều lần từ các vị trí khác nhau trong chương trình chính.

Để bắt đầu thực hiện bất kỳ một chương trình nào, kể cả chương trình con, thì bộ đếm chương trình (con trỏ lệnh) PC phải trỏ đến (phải chứa địa chỉ) ô nhớ đầu tiên chứa mã lệnh của chương trình đó. Tuy nhiên quá trình thực hiện chương trình con còn đòi thực hiện một thao tác đặc thù sau khi thực

hiện xong chương trình con, đó là trở về được chương trình gọi chương trình con đó, để tiếp tục thực hiện chương trình này. Để làm được việc này cần có một vùng nhớ được quản lý đặc biệt để có thể dễ dàng lưu trữ và khôi phục địa chỉ trở về chương trình chính, vùng nhớ đó được gọi là ngăn xếp.

2.4.1. Ngăn xếp và con trỏ ngăn xếp

Ngăn xếp là một vùng nhớ dữ liệu được quản lý theo kiểu dữ liệu đưa Vào sau cùng sẽ được lấy Ra đầu tiên (kiểu LIFO, Last In-First Out). Dữ liệu đầu tiên được đặt vào ô đầu tiên của ngăn xếp, ô này được gọi là *đáy ngăn xếp*. Đáy ngăn xếp có địa chỉ cao nhất trong ngăn xếp. Nếu có dữ liệu tiếp theo thì nó sẽ được được đưa vào ô tiếp theo có địa chỉ thấp hơn, ô này được gọi là *đỉnh ngăn xếp và* cứ thế tiếp tục. Khi đọc ngăn xếp thì dữ liệu được lấy từ đỉnh ngăn xếp. Đỉnh ngăn xếp luôn dịch chuyển khi ta ghi hoặc đọc ngăn xếp.

Ngăn xếp được dùng trong trường hợp gọi chương trình con để:

- Lưu trữ địa chỉ trở về chương trình gọi chương trình con đó.
- Lưu trữ, bảo vệ nội dung các thanh ghi của CPU.
- Lưu trữ tham số thực, biến cục bộ của chương trình con và các dữ liệu khác trong trường hợp cần thiết.

Để quản lý việc truy nhập ngăn xếp người ta sử dụng một thanh ghi có tên c*on trỏ ngăn xếp* SP. Con trỏ ngăn xếp *luôn chứa địa chỉ* (trỏ đến) *ô đỉnh hiện thời* của ngăn xếp.

Các đơn vị xử lý trung tâm đều có các lệnh truy nhập ngăn xếp kiểu như PUSH, POP, CALL, RET, IRET. Các lệnh này khi được thực hiện sẽ làm thay đổi nội dung (giá trị) của con trỏ ngăn xếp và kích thước của ngăn xếp.

Khi sử dụng lệnh PUSH hoặc CALL, các lệnh này sẽ làm nội dung của con trỏ ngăn xếp SP (phản ảnh địa chỉ ô đỉnh hiện thời của ngăn xếp) giảm đi, dữ liệu được đưa vào ngăn xếp và kích thước ngăn xếp tăng lên.

Khi sử dụng lệnh POP, RET, hoặc IRET, các lệnh này sẽ làm nội dung của con trỏ ngăn xếp SP (phản ảnh địa chỉ ô đỉnh hiện thời của ngăn xếp) tăng lên, dữ liệu được đọc từ ngăn xếp và kích thước ngăn xếp giảm đi.

- Trạng thái của con trỏ ngăn xếp và ngăn xếp khi CPU thực hiện hai lệnh PUSH và POP:

-Với lệnh PUSH:

lệnh PUSH được dùng để đưa nội dung một thanh ghi hoặc cặp thanh ghi CPU vào ngăn xếp, lên trên vị trí đỉnh hiện thời. Lệnh PUSH thường được dùng để bảo vệ (lưu giữ) nội dung các thanh ghi của CPU trước khi cpu chuyển sang thực hiện chương trình con.

PUSH reg16

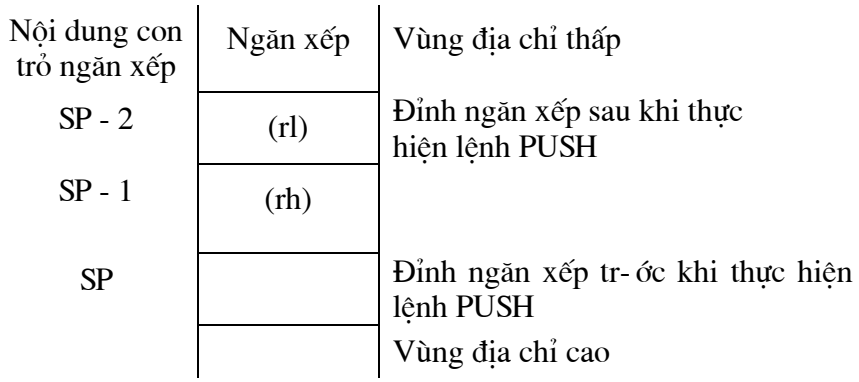
reg16: ký hiệu của thanh ghi 16 bit hoặc cặp thanh ghi của CPU, gồm hai phần 8 bit *rh* vµ *rl*

Quá trình thực hiện lệnh push được mô tả qua ký pháp và hình vẽ sau (hình 22).

Giả thiết trước khi thực hiện lệnh push con trỏ ngăn xếp có giá trị là SP.

PUSH reg16

(SP - 1) <- rh	Nội dung 8 bit cao của thanh ghi reg16 được đưa vào ngăn xếp, tại ô có địa chỉ SP-1
(SP - 2) <- rl	Nội dung 8 bit thấp của thanh ghi reg16 được đưa vào ngăn xếp, tại ô có địa chỉ SP-2
SP <- SP - 2	Sau khi thực hiện lệnh PUSH con trỏ ngăn xếp có giá trị là SP-2 . Đỉnh ngăn xếp hiện thời có địa chỉ giảm đi 2 so với trước đó



Hình 22

- Với lệnh POP:

Lệnh POP được dùng để đọc nội dung của ngăn xếp, từ vị trí đỉnh hiện thời. lệnh POP thường được dùng để khôi phục nội dung các thanh ghi của CPU khi CPU rời khỏi chương trình con trở về chương trình đã gọi nó.

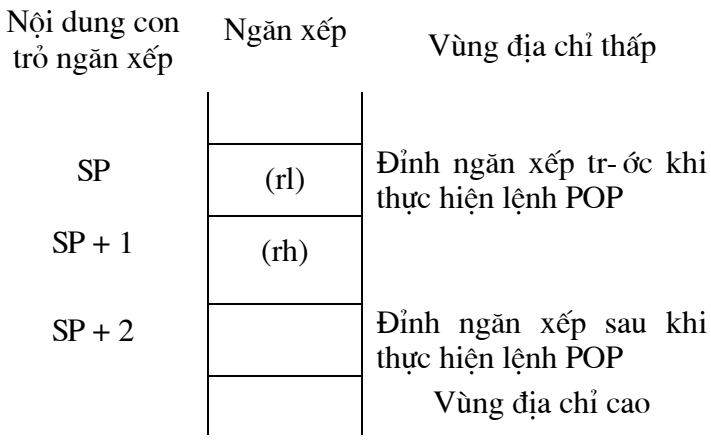
POP reg16

reg16 ký hiệu của thanh ghi 16 bit hoặc cặp thanh ghi của CPU, gồm hai phần 8 bit *rh* và *rl*

Quá trình thực hiện lệnh POP được mô tả qua ký pháp và hình sau (hình 23). Giả thiết trước khi thực hiện lệnh POP con trỏ ngăn xếp có giá trị là SP.

POP reg16

rl <- (SP)	Nội dung ngăn xếp tại ô có địa chỉ SP được đưa trở lại 8 bit thấp của thanh ghi reg16.
rh <- (SP + 1)	Nội dung ngăn xếp tại ô có địa chỉ SP+1 được đưa trở lại 8 bit cao của thanh ghi reg16.
SP <- SP + 2	Sau khi thực hiện lệnh POP con trỏ ngăn xếp có giá trị là SP+2 . Đỉnh ngăn xếp hiện thời có địa chỉ tăng lên 2 so với trước đó.



Hình 23

2.4.2. Cơ chế gọi chương trình con

Việc thực hiện chương trình con xảy ra khi CPU thực hiện lệnh CALL (lệnh gọi chương trình con). Để thực hiện chương trình con thì bộ đếm chương trình (con trỏ lệnh) PC phải trỏ đến (phải chứa địa chỉ) ô nhớ đầu tiên chứa chương trình con đó. Sau khi thực hiện xong chương trình con, CPU cần trở về được chương trình chính tại nơi vừa gọi chương trình con này. Việc này được thực hiện khi CPU gặp lệnh RET. Ngăn xếp được sử dụng để chứa địa chỉ trở về chương trình chính khi CPU gọi chương trình con.

Đơn vị xử lý trung tâm thực hiện chương trình con khi gặp lệnh CALL (gọi chương trình con):

CALL xxxxH

trong đó xxxxH là địa chỉ của chương trình con.

Con số xxxxH là ký pháp thập lục phân (Hexadecimal), trong đó mỗi chữ số thập lục phân (từ 0 đến F) tương ứng với 4 chữ số nhị phân.

Khi thực hiện lệnh CALL thì CPU phải thực hiện hai thao tác cơ bản:

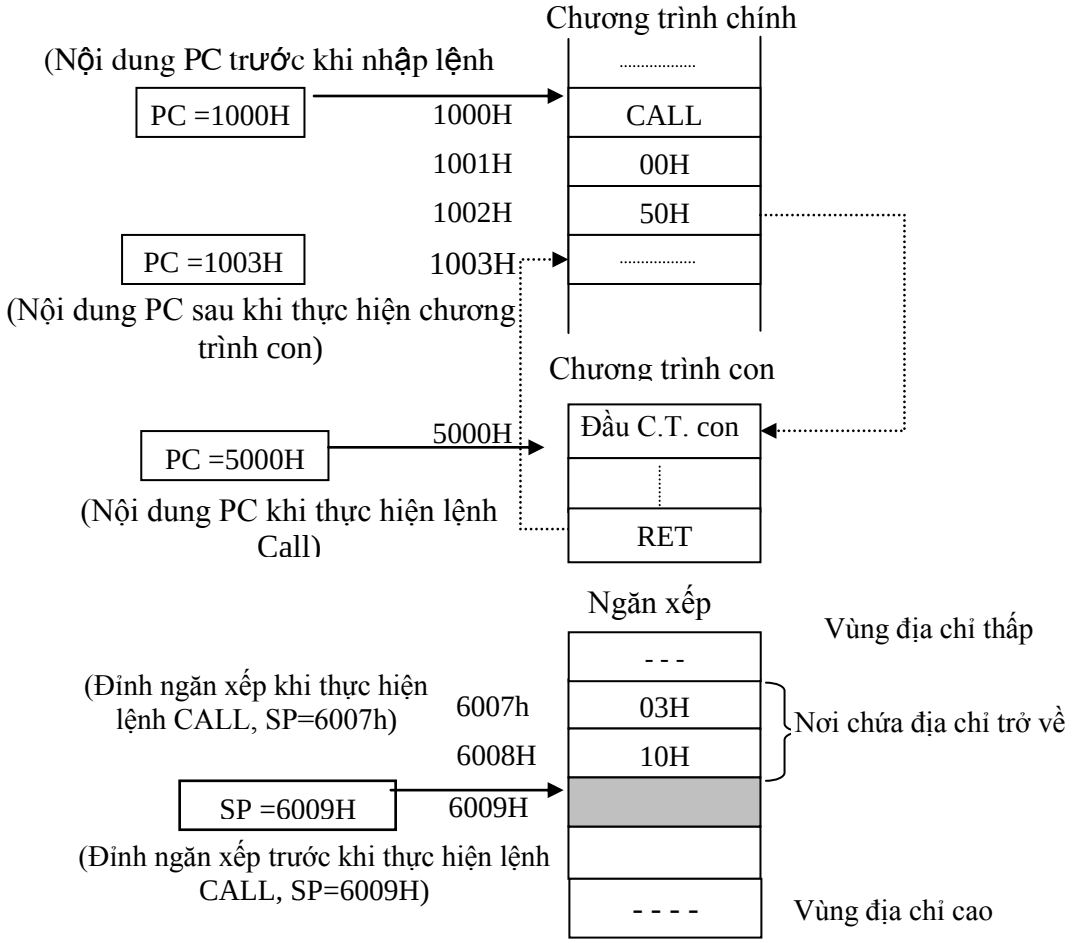
- Lưu địa chỉ của câu lệnh tiếp theo ngay sau lệnh CALL vào ngăn xếp. Địa chỉ này được gọi là *địa chỉ trở về* chương trình gọi chương con đó. Coi thanh ghi PC 16 bit được cấu thành từ hai phần 8 bit PCH và PCL. Giá trị của địa chỉ trở về thực chất là nội dung của con trỏ lệnh PC khi nó đang trỏ đến câu lệnh nằm sau lệnh CALL.
- Nạp địa chỉ của chương trình con vào bộ đếm chương trình (con trỏ lệnh) PC và thực hiện chương trình con từ vị trí PC trỏ đến.

Cơ chế thực hiện chương trình con được thể hiện qua ký pháp và hình vẽ (Hình 24). Giả thiết trước khi thực hiện lệnh CALL con trỏ ngăn xếp có giá trị là SP.

CALL Địa chỉ chương trình con	
(SP - 1) <- PCH	Nội dung 8 bit cao của thanh ghi PC đ- ợc đ- a vào ngăn xếp, tại ô có địa chỉ SP-1
(SP - 2) <- PCL	Nội dung 8 bit thấp của thanh ghi PC đ- ợc đ- a vào ngăn xếp, tại ô có địa chỉ SP-2
SP <- SP-2	Đỉnh ngăn xếp hiện thời có địa chỉ giảm đi 2
PC <- Địa chỉ chương trình con	Địa chỉ ch- ơng trình con đ- ợc nạp vào PC

Ví dụ: CALL 5000H, trong đó 5000H là địa chỉ chương trình con.

Giả định trong chương trình chính lệnh CALL 5000H nằm ở các ô 1000H, 1001H, 1002H. Địa chỉ của lệnh tiếp theo (địa chỉ trở về) là 1003H.



Hình 24

Khi thực hiện lệnh RET thì CPU đọc nội dung ô đỉnh ngăn xếp (ô chứa địa chỉ trở về chương trình chính) và nạp vào PC, nội dung của SP tự động tăng thêm 2. Sau khi thực hiện lệnh RET thì PC trỏ trở lại vào ô nhớ của lệnh tiếp theo trong chương trình chính, chương trình chính tiếp tục được thực hiện.

2.5. KỸ THUẬT XỬ LÝ LỆNH KIỂU ĐƯỜNG ỐNG DẪN VÀ XỬ LÝ SONG SONG MỨC LỆNH

2.5.1. Kỹ thuật xử lý lệnh kiểu đường ống dẫn

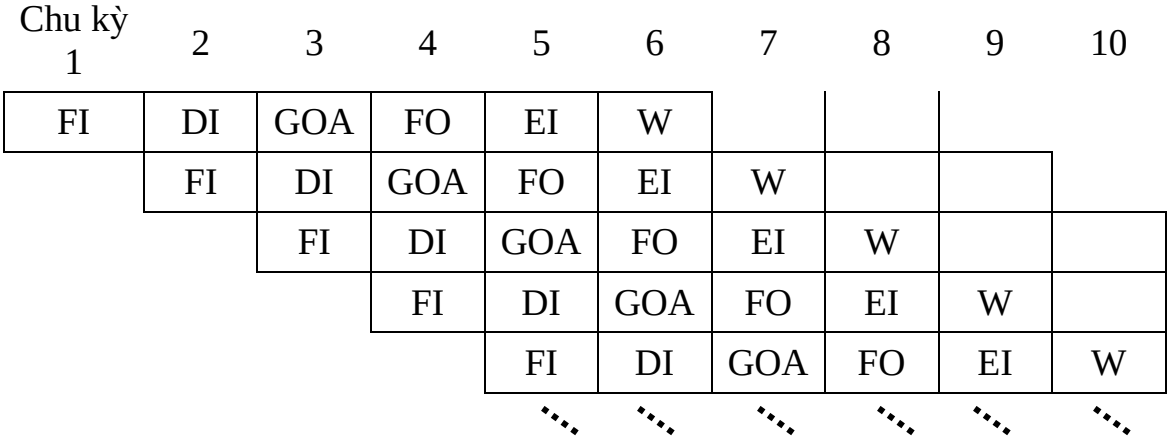
Một lệnh thường được xử lý qua sáu giai đoạn:

1_ Nhập lệnh (FI-Fetch the Instruction)
2_ Giải mã lệnh (DI-Decode the Instruction)
3_ Tạo địa chỉ toán hạng (GOA-Generate Operand Address)
4_ Nhập toán hạng (FO-Fetch Operands)
5_ Thực hiện lệnh (EI-Execute Instruction)
6_ Ghi kết quả (Write Back/Store Result)

Với kỹ thuật xử lý lệnh thông thường, đơn vị xử lý trung tâm phải tuần tự thực hiện xong tất cả các giai đoạn thực hiện một lệnh, thường là sau 6 chu kỳ máy, rồi mới chuyển sang nhập và thực hiện lệnh tiếp theo. Để tăng tốc độ xử lý lệnh mà không nhất thiết phải tăng tần số nhịp của máy, người ta sử dụng cỏc kỹ thuật xử lý lệnh khác, như kỹ thuật xử lý lệnh kiểu *đường ống dẫn* (*Pipelining*) và kỹ thuật *ILP*.

Đường ống tương tự như các dây chuyền sản xuất trong các xí nghiệp. Ở dây chuyền sản xuất, mỗi công đoạn thực hiện một thao tác sản xuất và nhiều công đoạn được thực hiện đồng thời. Sản phẩm được hình thành dần sau mỗi công đoạn sản xuất và được chuyển đến công đoạn sau, cho đến khi được hoàn thành ở công đoạn cuối cùng. Trong kỹ thuật xử lý lệnh theo kiểu đường ống, việc thực hiện lệnh cũng được thực hiện qua nhiều giai đoạn xử lý, giai đoạn nọ tiếp sau giai đoạn kia, cho đến khi thực hiện xong lệnh. Với một đường ống 6-giai đoạn, tại mỗi chu kỳ máy có 6 bộ dữ liệu thuộc 6 giai đoạn xử lý được gửi vào đường ống và 6 thao tác xử lý lệnh được thực hiện đồng thời. Nếu tính từ khi thực thi xong lệnh đầu tiên cứ sau mỗi chu kỳ máy lại có một lệnh được hoàn thành và một lệnh mới được nhập (Hình 25). Như vậy kỹ thuật xử lý lệnh kiểu đường ống làm tăng tốc độ thực hiện lệnh của CPU lên nhiều lần mà không cần tăng tần số nhịp. Có một vài tình huống thực hiện lệnh có thể làm cho kỹ thuật đường ống không được suôn sẻ, ví dụ như khi CPU thực hiện lệnh điều khiển rẽ nhánh, khi đó dòng lệnh trong đường ống sẽ bị ngắt quãng để chuyển sang lệnh khác v.v. Để hạn chế ảnh hưởng của các tình huống như trên, cần phải thực hiện việc phát hiện sớm lệnh điều khiển rẽ nhánh trong đường ống, để xác định địa chỉ cần nhảy tới và nạp vào PC ngay

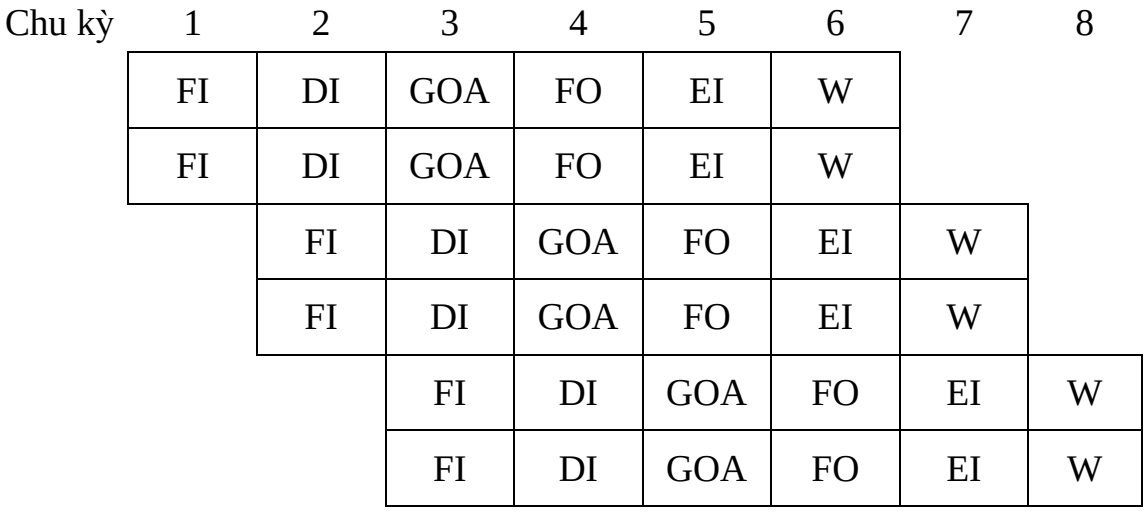
khi có thể. Quá trình này được giải quyết nhờ việc đưa vào cấu trúc của CPU *bộ dự báo rẽ nhánh*.



Hình 25

2.5.2. Kỹ thuật xử lý song song mức lệnh ILP

Kỹ thuật xử lý song song mức lệnh ILP (Instruction Level Parallelism) là kỹ thuật thiết kế đơn vị xử lý trung tâm và chương trình dịch nhằm làm tăng tốc độ thao tác máy bằng cách thực hiện song song các lệnh. Trong các kỹ thuật ILP có kỹ thuật *superscalar*, trong đó tại một chu kỳ máy *n* lệnh được nhập và được thực hiện đồng thời trên *n* kênh khác nhau. Để thực hiện được *n* lệnh đồng thời thì CPU cần phải có *n* đơn vị xử lý thao tác. Ví dụ, để thực hiện 2 lệnh số nguyên đồng thời thì cần 2 kênh thực hiện lệnh, trong đó có 2 đơn vị số học-logic ALU. Quá trình thực hiện lệnh theo kỹ thuật ILP như sau (Hình 26).



Hình 26

Chương 3
BUS VÀ VẤN ĐỀ TRUYỀN THÔNG TIN TRONG MÁY TÍNH

Các thành phần của hệ thống máy tính như đơn vị xử lý trung tâm, bộ nhớ, các module vào-ra và các thành phần khác phải được kết nối với nhau. Việc các thành phần này được kết nối và thực hiện trao đổi thông tin với nhau như thế nào sẽ ảnh hưởng mạnh đến hiệu năng của hệ thống máy tính. Ở các máy tính thế hệ đầu thì bộ nhớ và các thiết bị vào-ra được kết nối trực tiếp với đơn vị xử lý trung tâm. Kiểu kết nối này có nhược điểm là phải sử dụng rất nhiều cáp nối cùng các thiết bị giao diện tương ứng, cồng kềnh và có giá thành cao. Để giảm giá thành hệ thống kết nối và chuẩn hoá được các thiết bị giao diện kết nối, đồng thời cũng làm nhỏ gọn hệ thống, người ta sử dụng một kỹ thuật kết nối khác gọi là bus.

3.1. BUS VÀ TỔ CHỨC HỆ THỐNG MÁY TÍNH

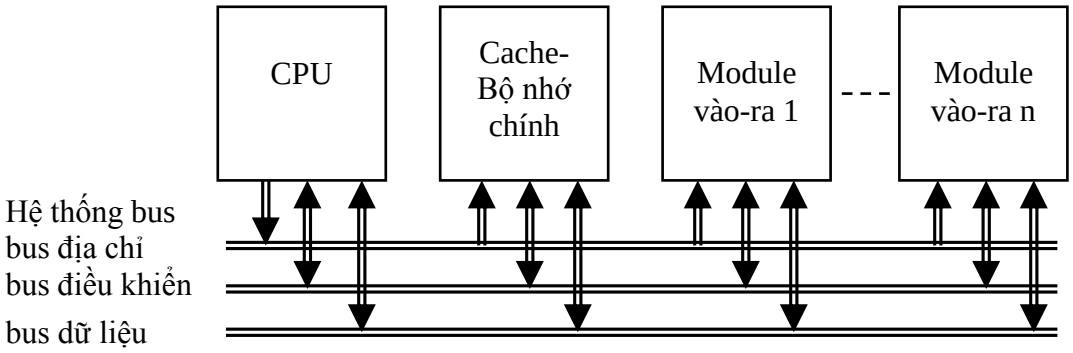
Bus là đường truyền thông tin trong máy tính. Mỗi đường bus truyền tất cả các bit của một từ dữ liệu cùng một lúc, từ nguồn dữ liệu đến đích.

Về mặt vật lý, bus là tập hợp của các đường dây truyền tín hiệu điện, mỗi một đường dây truyền được một bit thông tin tại một thời điểm.

Trong máy tính, để giảm giá thành, bus được sử dụng như là một *môi trường truyền thông tin chung* giữa nhiều thiết bị. Khi các thiết bị được kết nối lên bus, tín hiệu được phát ra bởi một thiết bị nguồn có thể được nhận bởi nhiều thiết bị đích khác đang được kết nối (về mặt điện) lên bus. Bus có thể là một chiều, tức là chỉ truyền theo một hướng, hoặc là hai chiều. Việc sử dụng bus làm một môi trường truyền tin chung làm giảm giá thành xây dựng và sử dụng bus, nhưng lại đòi hỏi một cơ chế điều khiển bus phức tạp hơn.

Trong máy tính, đơn vị xử lý trung tâm CPU thực hiện kết nối và trao đổi thông tin với các đơn vị chức năng khác thông qua hệ thống bus (Hình 27).

Hệ thống bus của máy tính gồm 3 loại bus, mỗi loại bus truyền một loại thông tin.



Hình 27

+ Bus địa chỉ:

Bus địa chỉ là bus được dùng để truyền địa chỉ của ô nhớ hoặc thiết bị mà CPU lựa chọn và muốn truy nhập. Mỗi một ô nhớ hay một thiết bị trong máy tính đều được gán một địa chỉ duy nhất. Để lựa chọn ô nhớ hoặc thiết bị, CPU sẽ phát ra địa chỉ tương ứng. Địa chỉ này được truyền trên bus địa chỉ đến nơi CPU cần truy nhập. Bus địa chỉ là loại bus một chiều. Độ rộng của bus địa chỉ (số lượng đường dây truyền tín hiệu địa chỉ) xác định dung lượng nhớ vật lý tối đa có thể của bộ nhớ chính và thiết bị trong máy tính. Có nhiều loại bus địa chỉ, từ 16 bit đến 32 hoặc 64 bit. Các bit địa chỉ cao được dùng để chọn các module được kết nối lên hệ thống bus. Các bit địa chỉ thấp được dùng để chọn các ô nhớ hoặc vị trí cổng vào ra trong module.

+ Bus điều khiển:

Bus điều khiển là tập hợp của các đường dây, mỗi một đường dây truyền một tín hiệu điều khiển. Các tín hiệu điều khiển do CPU hoặc các thiết bị phát ra để điều khiển các quá trình trao đổi dữ liệu trong máy tính. Bus điều khiển là loại bus hai chiều. Bus điều khiển thường gồm các đường truyền tín hiệu điều khiển sau:

- Tín hiệu đọc bộ nhớ: tín hiệu này làm cho dữ liệu từ bộ nhớ, từ ô nhớ có địa chỉ xác định, được đưa lên bus dữ liệu.
- Tín hiệu ghi bộ nhớ: tín hiệu này làm cho dữ liệu trên bus dữ liệu được ghi vào bộ nhớvào ô nhớ có địa chỉ xác định.
- Tín hiệu đọc cổng vào/ra: tín hiệu này làm cho dữ liệu từ thanh ghi của một module vào-ra có địa chỉ xác định được đưa lên bus dữ liệu.
- Tín hiệu ghi cổng vào/ra: tín hiệu này làm cho dữ liệu trên bus dữ liệu được ghi vào một thanh ghi của module vào/ra có địa chỉ xác định.
- Tín hiệu yêu cầu ngắt: tín hiệu này được phát từ thiết bị vào-ra, báo có yêu cầu ngắt đang chờ.
- Tín hiệu trả lời ngắt: tín hiệu này được phát từ CPU trả lời yêu cầu ngắt đã được ghi nhận.
- Tín hiệu yêu cầu bus: tín hiệu này báo có chủ bus khác đòi quyền điều khiển hệ thống bus.

- Tín hiệu trả lời chấp nhận chuyển quyền điều khiển bus.
- Tín hiệu xung nhịp đồng hồ hệ thống: được dùng để đồng bộ hoá các hoạt động trong hệ thống.

+ Bus dữ liệu:

Bus dữ liệu được dùng để truyền dữ liệu. Bus dữ liệu có thể gồm từ 8 đến 64 đường dây. Do vậy với loại bus dữ liệu 64 đường truyền, tại một thời điểm có thể truyền được từ 1 đến 8 byte dữ liệu. Bus dữ liệu là loại bus hai chiều. Dữ liệu có thể do CPU phát ra hay CPU nhận về từ bộ nhớ hoặc các thiết bị.

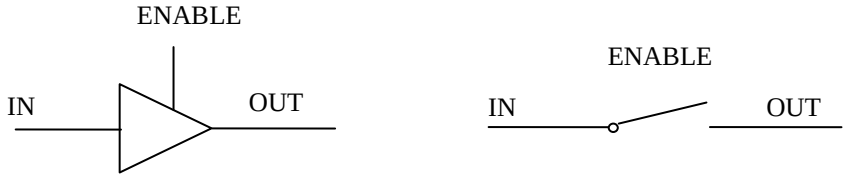
Tại mỗi thời điểm, CPU chỉ làm việc hoặc với bộ nhớ chính hoặc với một thiết bị. Khi CPU muốn trao đổi thông tin với đối tượng nào thì CPU phát ra địa chỉ của đối tượng đó lên bus địa chỉ. Đối tượng (bộ nhớ hoặc thiết bị) có địa chỉ trùng với địa chỉ do CPU phát ra sẽ được kết nối (về mặt điện) lên bus dữ liệu. Quá trình truyền dữ liệu sẽ được thực hiện và được điều khiển bởi các tín hiệu điều khiển.

3.2. GIAO DIỆN BUS VÀ THIẾT BỊ BA TRẠNG THÁI

Thiết bị ba trạng thái là phương tiện giúp cho việc điều khiển kết nối (về mặt điện) CPU, bộ nhớ và các thiết bị lên hệ thống bus.

Việc chỉ dùng một đường bus để kết nối và truyền thông tin giữa các thiết bị khác nhau đòi hỏi tại một thời điểm chỉ một thiết bị được phép phát tín hiệu (mang tải thông tin) lên bus. Nếu tại một thời điểm lại có nhiều thiết bị được kết nối và cùng phát tín hiệu lên bus thì sẽ gây ra xung đột trên bus này.

Để làm chủ được việc kết nối (về mặt điện) các thiết bị lên bus, người ta sử dụng một thiết bị điện tử có tên thiết bị ba trạng thái (tri-state driver). Dạng logic và mạch điện tương đương của thiết bị ba trạng thái như sau (Hình 28).



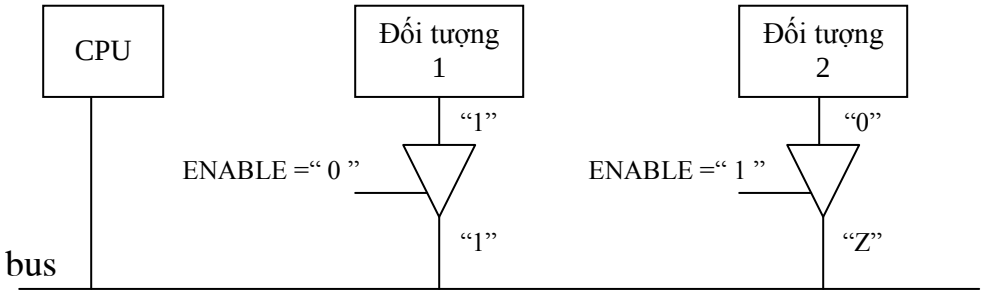
Hình 28

Bảng 6 mô tả chức năng của thiết bị ba trạng thái

Bảng 6

ENABLE	**IN**	**OUT** (có 3 trạng thái)
0	0	0
0	1	1
1	0	Z (trở kháng cao)
1	1	Z (trở kháng cao)

Thiết bị ba trạng thái cho phép CPU hoặc các đơn vị chủ bus khác làm chủ được việc kết nối các đối tượng khác nhau lên bus. Khả năng làm chủ việc kết nối (về mặt điện) sẽ cho phép tránh sự kiện có hai thiết bị cùng được kết nối vào bus dữ liệu và cùng phát dữ liệu trong cùng một thời điểm, đảm bảo không để xảy ra xung đột. Khi cần chọn đối tượng nào CPU phát ra địa chỉ của đối tượng đó cùng tín hiệu điều khiển ghi/đọc tương ứng. Các đối tượng thực hiện giải mã địa chỉ này, kết hợp với tín hiệu điều khiển để tạo ra tín hiệu cho phép ENABLE. Ở đối tượng có địa chỉ đúng với địa chỉ do CPU phát ra, tín hiệu sau giải mã địa chỉ sẽ được kết hợp với tín hiệu điều khiển để tạo ra tín hiệu Enable có mức tích cực, cho phép đối tượng này, thông qua thiết bị ba trạng thái, được kết nối lên bus. Ở các đối tượng có địa chỉ không phù hợp thì tín hiệu Enable được tạo ra sẽ có mức không tích cực, khi đó thiết bị ba trạng thái của nó sẽ có trạng thái trở kháng cao, nên thiết bị không kết nối được lên bus (Hình 29).



Hình 29

Việc sử dụng thiết bị 3 trạng thái trong thiết kế bus chung còn đem lại nhiều lợi ích khác như: cho phép rất nhiều thiết bị được nối lên cùng một đường dây, hỗ trợ việc kết nối thiết bị và điều khển truyền tin theo hai chiều trên cùng một bus tại các thời điểm khác nhau.

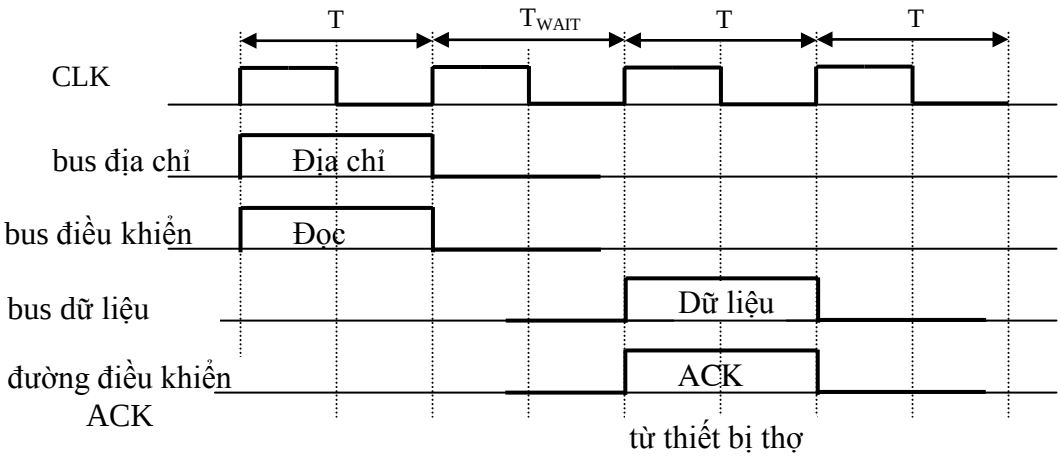
3.3. T RỌNG TÀI BUS

Trên hệ thống bus chung, ngoài CPU là thiết bị điều khiển quá trình truyền thông tin trên bus còn có các thiết bị khác có thể và cần điều khiển hệ thống bus. Các thiết bị có khả năng điều khiển hệ thống bus được gọi là thiết bị chủ bus hay là thiết bị điều khiển bus. Trên hệ thống bus tại một thời điểm có thể một vài thiết bị chủ cùng có yêu cầu truy nhập bus, nhưng tại một thời điểm chỉ cho phép một thiết bị được truyền thông tin trên bus, nên cần có phương pháp điều phối quyền sử dụng bus. Kỹ thuật cho phép điều phối quyền sử dụng bus được gọi là kỹ thuật trọng tài bus hoặc phân xử bus.

3.4. ĐỊNH THỜI

Định thời là cách để phối hợp các sự kiện ở trên hệ thống bus. Có hai phương pháp để định thời quá trình truyền thông tin trên bus, đó là định thời đồng bộ và không đồng bộ.

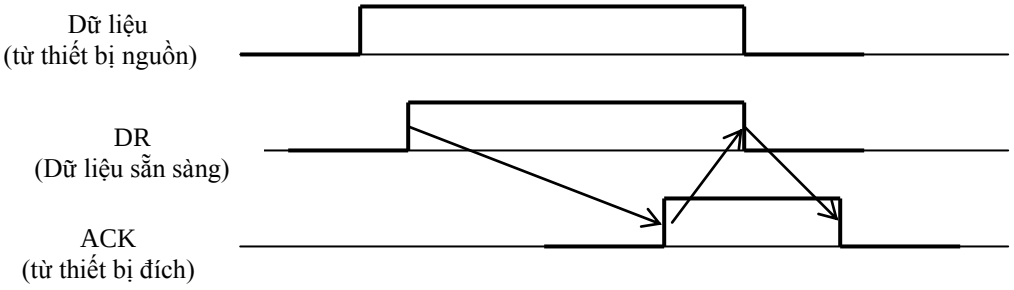
Đặc điểm của phương pháp định thời đồng bộ là các sự kiện xuất hiện trên bus theo nhịp xung đồng hồ. Quá trình truyền tin giữa thiết bị chủ và thợ được đồng bộ hoá theo xung nhịp do thiết bị chủ phát ra (Hình 30).



Hình 30

Ta khảo sát quá trình truyền thông tin kiểu đồng bộ giữa thiết bị chủ và các thiết bị thợ qua thao tác đọc dữ liệu. Quá trình truyền thông tin được diễn ra theo một thủ tục xác định. Đầu tiên thiết bị chủ bus phát tín hiệu điều khiển xác định kiểu thao tác truyền, ví dụ đọc dữ liệu từ thiết bị thợ là bộ nhớ hoặc thiết bị vào-ra, thông qua tín hiệu điều khiển đọc. Đồng thời thiết bị chủ bus đưa địa chỉ của đối tượng được lựa chọn lên bus địa chỉ. Tất cả các thiết bị thợ đang ở trên bus sẽ kiểm tra xem có phải là địa chỉ của mình. Thiết bị thợ có địa chỉ trùng sẽ kết nối và phát từ dữ liệu lên bus dữ liệu trễ sau một chu kỳ nhịp. Thiết bị thợ cũng có thể phát thông tin trạng thái ACK cho chủ. Tín hiệu ACK chỉ được phát ra khi thiết bị thợ đã hoàn thành việc đưa dữ liệu lên bus. Thiết bị chủ chờ cho đến khi nhận được ACK mới bắt đầu một chu kỳ truyền tin khác. Như vậy tín hiệu ACK có thể được thiết bị thợ dùng để trì hoãn đáp ứng từ một đến nhiều chu kỳ nhịp. Thủ tục ghi dữ liệu từ thiết bị chủ đến thợ cũng diễn ra tương tự như vậy. Kỹ thuật truyền tin đồng bộ phù hợp cho việc trao đổi thông tin giữa các thiết bị có tốc độ truyền tin tương đương với nhau. Hệ thống truyền tin bên trong máy tính sử dụng kỹ thuật truyền tin đồng bộ.

Kỹ thuật truyền tin không đồng bộ không sử dụng tín hiệu xung nhịp, mà sử dụng cơ chế "bắt tay" để đồng bộ hoá quá trình truyền tin, trong đó cả hai thiết bị nguồn và đích đều phát tín hiệu định thời (Hình 31). Thiết bị nguồn đưa dữ liệu lên bus. Sau một thời gian trễ xác định, thiết bị nguồn phát tín hiệu báo dữ liệu sẵn sàng DR (Data Ready). Thiết bị nguồn duy trì dữ liệu trên bus cho tới khi nhận tín hiệu trả lời ACK từ thiết bị đích. Thiết bị đích đặt ACK tích cực sau khi đã nhận được dữ liệu. Thủ tục truyền một từ dữ liệu hoàn toàn kết thúc với mức không tích cực của ACK. Kỹ thuật truyền tin không đồng bộ phù hợp cho việc trao đổi thông tin giữa các thiết bị có tốc độ truyền tin khác nhau. Các hệ thống máy tính thường sử dụng kỹ thuật truyền không đồng bộ để trao đổi dữ liệu với các thiết bị bên ngoài.



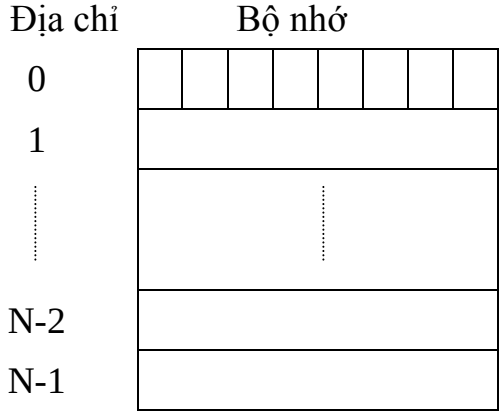
Hình 31

Chương 4
TỔ CHỨC BỘ NHỚ

4.1. TỔ CHỨC HỆ THỐNG BỘ NHỚ MÁY TÍNH

Bộ nhớ là một thành phần cơ bản của máy tính. Chức năng của bộ nhớ là chứa thông tin (chứa các chương trình và dữ liệu có liên quan).

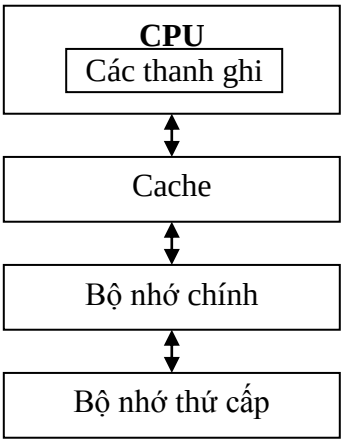
Bộ nhớ là tập hợp có thứ tự của các ô nhớ, mỗi ô nhớ được gán một địa chỉ. Địa chỉ ô nhớ xác định vị trí ô nhớ trong bộ nhớ. Mỗi ô nhớ có khả năng lưu trữ một từ dữ liệu độ dài cố định. Độ dài một từ dữ liệu có thể là 8 bit, 16 bit, 32 hoặc 64 bit. Từ dữ liệu dài 8 bit được gọi là byte. Ô nhớ được xây dựng trên các phần tử nhớ cơ bản. Mỗi phần tử nhớ cơ bản có khả năng lưu trữ 1 bit thông tin.



Một trong những hoạt động của đơn vị xử lý trung tâm (CPU) là nhập lệnh và ghi/đọc dữ liệu lên bộ nhớ chính. CPU thực hiện các thao tác truy nhập bộ nhớ với tốc độ rất cao. Các thiết bị nhớ có thời gian đáp ứng tương đương tốc độ của CPU thường có giá rất cao, nên nếu chỉ sử dụng một loại thiết bị nhớ để xây dựng bộ nhớ sẽ khó thoả mãn đồng thời cả hai yêu cầu về tốc độ truy cập cao và giá thành. Do vậy bộ nhớ của máy tính thường được xây dựng trên nhiều loại thiết bị nhớ có thời gian truy nhập và giá thành khác nhau, được tổ chức thành một hệ thống có nhiều mức và được quản lý như một hệ thống thống nhất (Hình 32).

Hệ thống bộ nhớ nhiều mức bao gồm các thanh ghi CPU, bộ nhớ chính, bộ nhớ thứ cấp và bộ nhớ cache. Cơ sở cho việc xây dựng hệ thống bộ nhớ nhiều mức là dựa trên nguyên lý quy chiếu phân vùng. Ý tưởng chính ở đây là

trong một thời khoảng thì các lệnh và dữ liệu được sử dụng thường nằm ở một khu vực tương đối nhỏ trong bộ nhớ và các vùng này luôn chuyển dịch khi CPU thực hiện chương trình.



Hình 32

Các thanh ghi CPU là loại thiết bị nhớ tốc độ cao, nằm trong CPU và có tốc độ truy cập tương đương với tốc độ hoạt động của CPU. Số lượng thanh ghi trong CPU rất hạn chế. Các thanh ghi CPU thực hiện chức năng lưu trữ tạm thời lệnh máy, các dữ liệu và các thông tin phục vụ việc thực hiện lệnh.

Bộ nhớ chính chứa chương trình và dữ liệu đang được CPU xử lý. Bộ nhớ chính nằm ngoài CPU, có dung lượng lớn và có tốc độ truy nhập cao. CPU truy cập trực tiếp bộ nhớ chính bằng cách xác định địa chỉ ô nhớ và phát tín hiệu điều khiển ghi-đọc khi nhập lệnh máy hoặc thực hiện các lệnh có thao tác ghi và đọc dữ liệu lên bộ nhớ. Bộ nhớ chính còn được gọi là bộ nhớ RAM (Random Access Memory), do CPU có thể truy cập trực tiếp bất kỳ ô nhớ nào trong bộ nhớ và với thời gian truy cập như nhau, không phụ thuộc vào vị trí ô nhớ trong bộ nhớ. Thời gian truy cập bộ nhớ chính thường dài hơn vài chu kỳ xung nhịp của CPU.

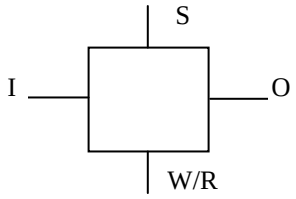
Bộ nhớ thứ cấp được đưa vào hệ thống bộ nhớ do nhu cầu ngày càng tăng về dung lượng nhớ. Bộ nhớ thứ cấp có dung lượng nhớ lớn hơn bộ nhớ chính nhiều lần, nhưng có tốc độ truy cập chậm hơn nhiều so với bộ nhớ chính. Bộ nhớ thứ cấp có khả năng lưu giữ thông tin ngay cả khi không được cung cấp nguồn nuôi. Ở mức vật lý, CPU không trực tiếp truy cập bộ nhớ thứ cấp như đối với bộ nhớ chính, mà thực hiện ghi hoặc đọc nó thông qua các lệnh vào-ra dữ liệu.

Bộ nhớ cache. Cache là loại bộ nhớ tốc độ cao. Bộ nhớ cache được đưa vào hệ thống bộ nhớ như một giải pháp tăng tốc độ truy cập bộ nhớ của CPU. Bộ nhớ cache có thời gian truy cập thấp, nhưng lại có giá thành cao và dung lượng bé hơn nhiều so với bộ nhớ chính.

4.2. TỔ CHỨC BỘ NHỚ CHÍNH (RAM)

Mỗi ô nhớ trong bộ nhớ vật lý được tạo trên cơ sở các phần tử nhớ cơ bản (Hình 33), trong đó S là tín hiệu lựa chọn phần tử, W/R là tín hiệu điều khiển ghi/đọc, I/O là đầu vào/ra dữ liệu.

Phần tử nhớ cơ bản là một thiết bị vật lý có các đặc tính sau: có hai trạng thái xác lập để thể hiện một trong hai chữ số nhị phân 0 và 1 được lưu giữ, có thể lựa chọn riêng rẽ từng phần tử để thực hiện thao tác ghi xác lập trạng thái, mỗi phần tử đều có khả năng duy trì ổn định trạng thái được xác lập, có thể thực hiện thao tác đọc để nhận biết trạng thái này. Thông tin xác định lựa chọn phần tử nhớ và điều khiển ghi đọc có thể được gộp lại dưới dạng một tín hiệu điều khiển duy nhất.

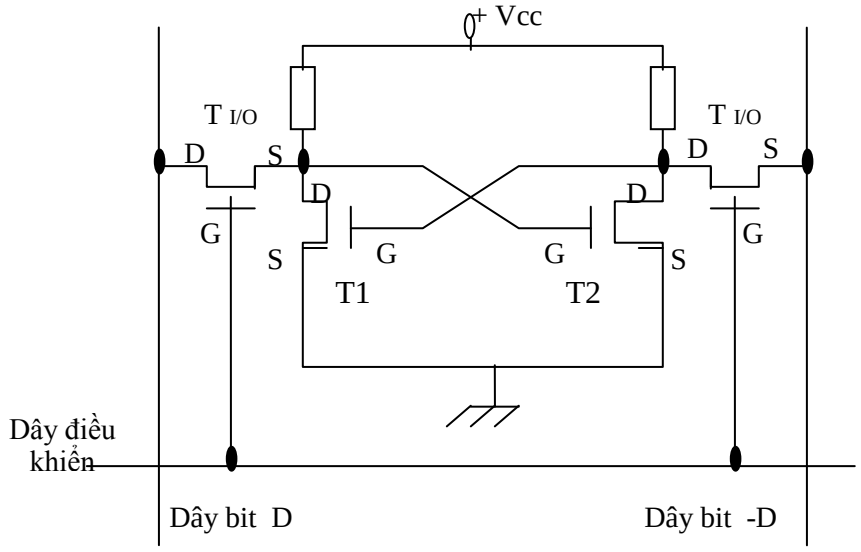


Hình 33

Các phần tử nhớ cơ bản có thể được chế tạo theo các công nghệ khác nhau, do vậy nó cũng có các thuộc tính vật lý như tốc độ truy cập, khả năng lưu trữ và giá thành khác nhau.

4.2.1. Phần tử nhớ tĩnh 1 bit SRAM

Phần tử nhớ tĩnh SRAM (Static Random Access Memory) dạng điện tử là một mạch lật (Hình 34). Thông tin được lưu trữ nhờ trạng thái của mạch lật.



Hình 34

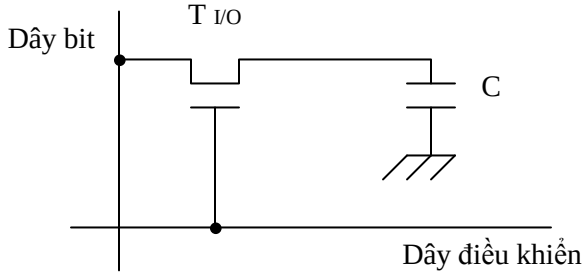
Cơ chế hoạt động của phần tử nhớ tĩnh:

- *Quá trình ghi và nhớ*: Dữ liệu, ví dụ có giá trị "1", được đưa vào dây bit D. Dây điều khiển được đặt lên "1" => $T_{I/O}$ thông => cực G_{T2} = "1" => T_2 thông => cực D_{T2}= "0" => G_{T1}="0" => T1 không thông => cực D_{T1}="1". Trạng thái này của T_1 được duy trì ngay cả khi dữ liệu đầu vào mất đi, giá trị của dữ liệu vào được ghi nhớ.
- *Quá trình đọc dữ liệu*: Dây điều khiển = "1" => $T_{I/O}$ thông => Trạng thái hiện thời của mạch, chính là giá trị của dữ liệu đang được lưu giữ, được đưa ra dây bit D.
- *Quá trình ghi dữ liệu mới*: Khi đưa dữ liệu mới, ví dụ là "0", lên dây bit D và đặt dây điều khiển = "1" => $T_{I/O}$ thông => cực D_{T1} ="0" => cực G_{T2}="0" => T2 không thông => cực D_{T2}= "1" => G_{T1}="1" => T_1 thông => cực D_{T1} = "0". Trạng thái này của T_1 được duy trì ngay cả khi dữ liệu đầu vào mất đi, giá trị của dữ liệu vào mới được ghi nhớ.

Đặc điểm của phần tử nhớ tĩnh: có tốc độ truy nhập cao, thông tin được lưu giữ ổn định chừng nào còn có nguồn nuôi. Tuy nhiên phần tử nhớ tĩnh có cấu trúc phức tạp, nên có mật độ dung lượng nhớ không cao và có giá thành cao.

4.2.2. Phần tử nhớ động 1 bit DRAM

Cấu trúc của một phần tử nhớ động DRAM (Dynamic Random Access Memory) được mô tả trên hình 35. Thành phần cơ bản của phần tử nhớ động là tụ điện C. Bóng bán dẫn $T_{I/O}$ đóng vai trò một chuyển mạch điện tử. Thông tin cần lưu giữ được thể hiện qua trạng thái điện thế của tụ điện C.



Hình 35

- *Quá trình ghi dữ liệu*: Dữ liệu, ví dụ có giá trị "1", được đưa vào dây bit D. Dây điều khiển được đặt lên "1" => $T_{I/O}$ thông => tụ điện C được nạp với mức logic "1". Giá trị dữ liệu vào được ghi nhớ qua trạng thái điện thế của tụ điện C, ngay cả khi dữ liệu đầu vào mất đi.
- *Quá trình đọc dữ liệu*: Dây điều khiển được đặt lên "1" => $T_{I/O}$ thông => trạng thái mức điện áp trên tụ điện C, thể hiện nội dung dữ liệu, được đưa ra dây bit D.

Đặc điểm của phần tử nhớ động (dynamic): có cấu trúc đơn giản nên có thể tạo mật độ phân bố cao và có giá thành thấp, nhưng có tốc độ truy xuất thấp hơn so với phần tử nhớ tĩnh. Hiện tượng tụ điện C tự phóng điện qua $T_{I/O}$ ngay

cả khi vẫn có nguồn nuôi làm mất đi thông tin cần lưu giữ, do vậy đòi hỏi phải có biện pháp làm tươi thông tin.

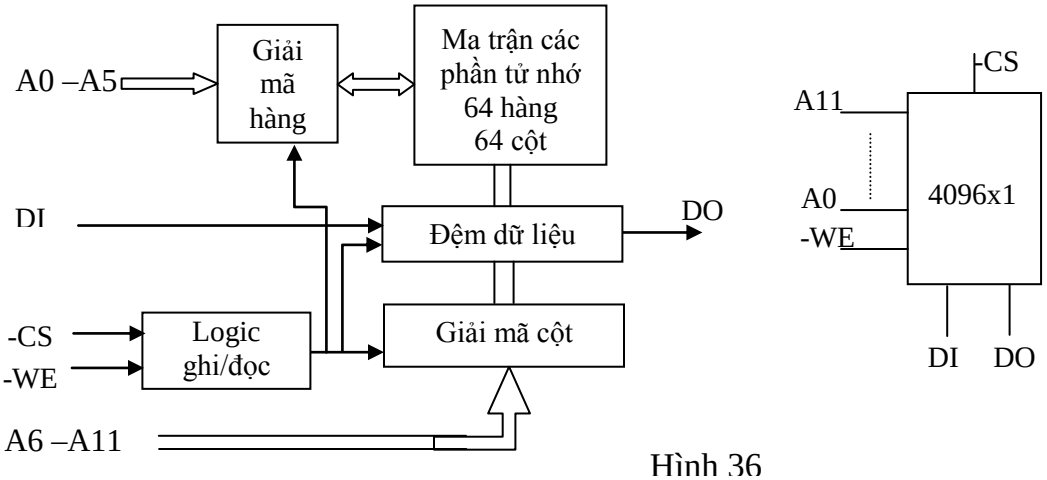
4.2.3. Xây dựng bộ nhớ RAM từ các chip nhớ

Bộ nhớ RAM được xây dựng trên các chip nhớ bán dẫn. Các chip nhớ RAM (SRAM hoặc DRAM) thường có các ô nhớ loại 1 bit, 4 bit hoặc 8 bit. Từ các chíp nhớ này có thể xây dựng nên bộ nhớ với ô nhớ chứa được từ dữ liệu rộng N bit.

4.2.3.1 Xây dựng bộ nhớ với các chip SRAM

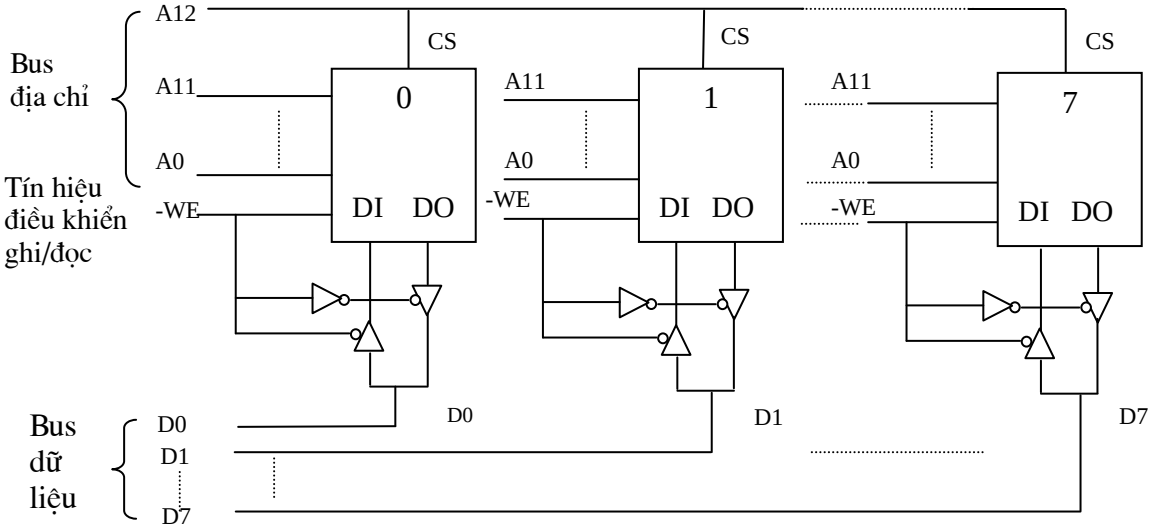
Giả sử cần xây dựng một bộ nhớ kích thước 4096 x 8 bit trên cơ sở các chíp SRAM loại 4096 x 1bit.

Cấu trúc của chip SRAM 4096x1bit (Hình 36):



Hình 36

Băng nhớ SRAM 4096 ô nhớ, trong đó mỗi ô nhớ chứa được một từ dữ liệu 8 bit, được xây dựng trên cơ sở 8 chip SRAM loại 4096x1bit. Để làm được điều này người ta sắp đặt 8 chip SRAM loại 4096x1bit sao cho mỗi chip tại một vị trí xác định sẽ đảm nhiệm lưu trữ bit dữ liệu có vị trí tương ứng trong từ dữ liệu. Các chip SRAM được kết nối theo hình sau (Hình 37).



Hình 37

Các đường tín hiệu:

A_{11} - A_0 các đường địa chỉ xác định vị trí ô nhớ trong chip và vùng nhớ

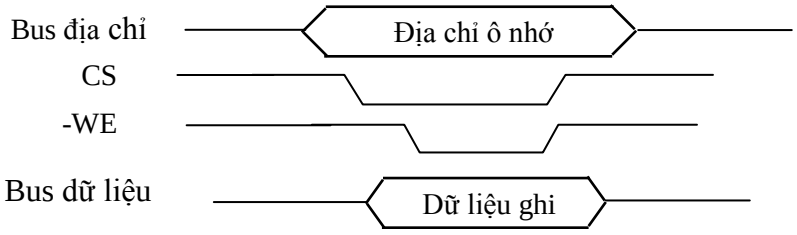
CS tín hiệu chọn chip. Nếu CS = 0 thì truy nhập được chip

WE tín hiệu điều khiển ghi/đọc bộ nhớ. WE=0 điều khiển ghi

D_7 - D_0 các đường truyền các bit dữ liệu từ D_7 đến D_0.

A_{12} đường địa chỉ đóng vai trò chọn vùng nhớ 4096 ô

Quá trình ghi bộ nhớ SRAM được thực hiện như sau (Hình 38).



Hình 38

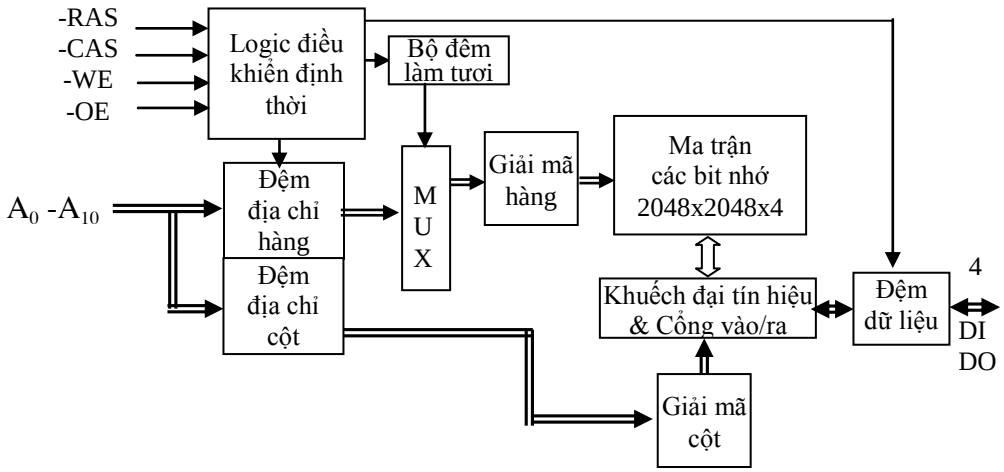
4.2.3.2 Chip nhớ loại DRAM

Chip DRAM được xây dựng từ các phần tử nhớ động. Cấu trúc điển hình của chip DRAM như sau (Hình 39).

Các tín hiệu điều khiển:

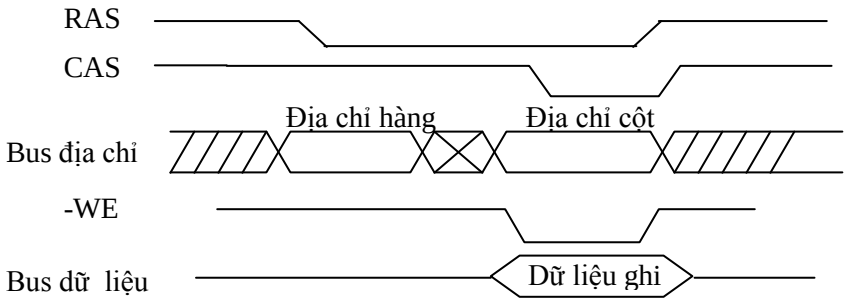
- RAS: khi tín hiệu RAS (Row Address Strobe) tích cực thì địa chỉ hàng được nạp (chốt lại).
- CAS: khi tín hiệu CAS (Column Address Strobe) tích cực thì địa chỉ cột được nạp (chốt lại).
- WE: tín hiệu điều khiển ghi
- OE: tín hiệu điều khiển đọc

Hình 39 mô tả chip DRAM 16 Mbit. Chip DRAM này được tổ chức như một bộ nhớ 4Mx4 bit. Các phần tử nhớ trong chip được bố trí theo ma trận 2048 hàng x 2048 cột x 4 bit. Mỗi một lần sẽ ghi hoặc đọc 4 bit cùng lúc.



Hình 39

Với kích thước bộ nhớ là 4Mx4 bit, mỗi vị trí nhớ cần được xác định bởi địa chỉ 22 bit, nhưng để tiết kiệm số chân, nên chỉ có 11 chân địa chỉ trên chip. Điều này yêu cầu phải có bộ dồn kênh đặt ngoài chip, để đưa được địa chỉ 22 bit lên 11 chân địa chỉ ở trên chip. DRAM dùng phương pháp dồn kênh để nạp lần lượt địa chỉ hàng và địa chỉ cột vào đệm địa chỉ. 11 dây địa chỉ được dùng để chọn 1 trong 2048 hàng. 11 dây địa chỉ khác được dùng để chọn 1 trong 2048 cột. Ví dụ, quá trình ghi bộ nhớ DRAM được thực hiện như sau (Hình 40).

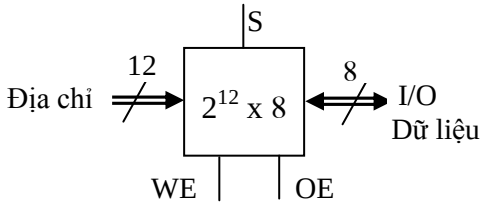


Hình 40

Đầu ra từ logic làm tươi được tuần tự đưa đến bộ giải mã hàng và dây RAS được kích hoạt. Điều này làm các phần tử nhớ trên hàng được làm tươi.

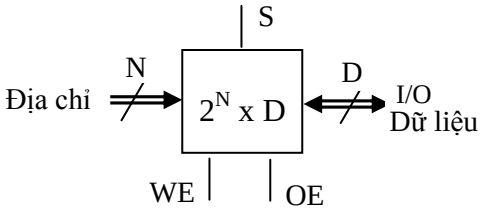
4.2.4. Tổ chức bộ nhớ RAM

Bộ nhớ RAM vật lý dung lượng lớn thường được xây dựng từ các băng nhớ RAM có dung lượng nhỏ hơn. Để làm rõ điều này, băng nhớ RAM kích thước 4Kx8 bit đã được thiết kế trong mục 3.2.3.1 được mô tả lại ở dạng sau (Hình 41).



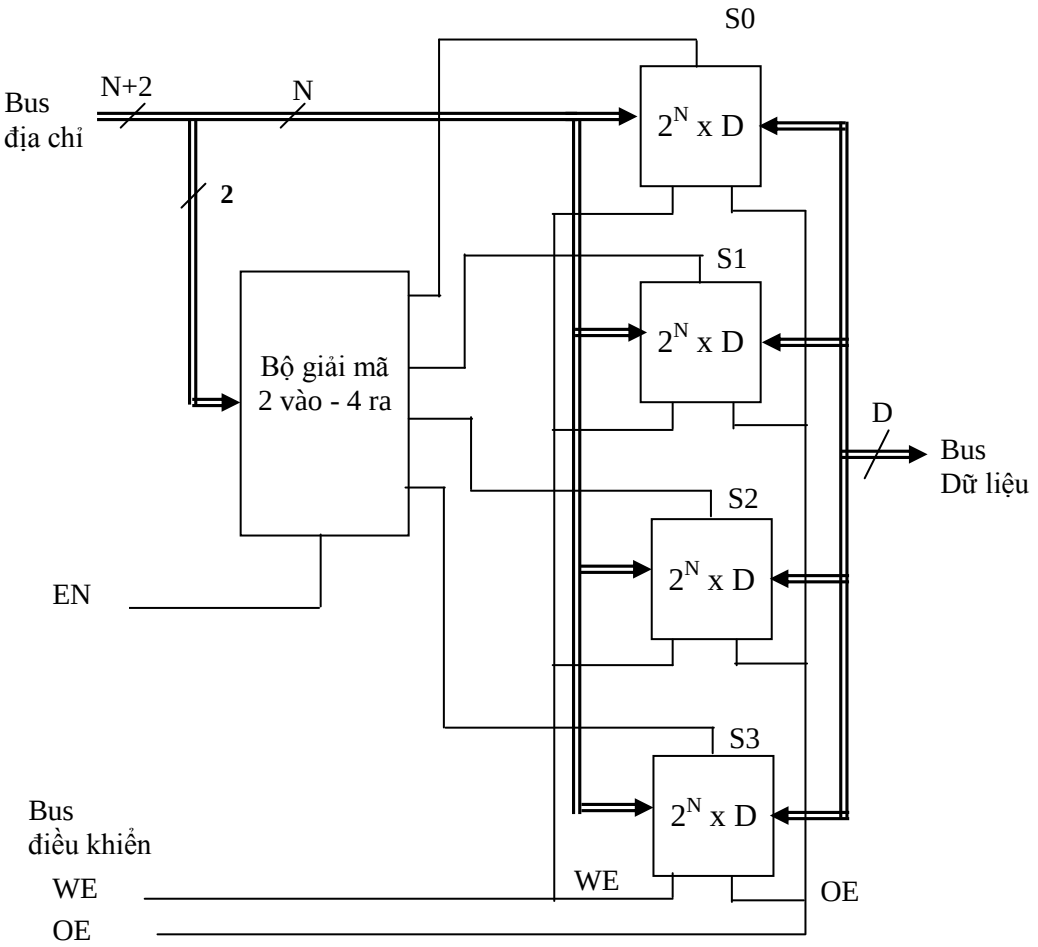
Hình 41

Theo cách tương tự, trên cơ sở các chip nhớ RAM 2^Nxd bit có thể thiết kế và xây dựng băng nhớ RAM dung lượng 2^NxDbit, mỗi ô nhớ chứa được từ dữ liệu dài D bit, trong đó trong đó D là bội số của d. Ký hiệu băng nhớ RAM dung lượng 2^NxDbit như hình 42.



Hình 42

Trên cơ sở các băng nhớ 2^N x Dbit ta có thể thiết kế và xây dựng bộ nhớ RAM kích thước lớn, có dung lượng nhớ là bội của 2^N. Hình 43 là một ví dụ về thiết kế bộ nhớ RAM dung lượng lớn $4x2^N$ ô nhớ D bit.



Hình 43

Bus địa chỉ rộng N+2 bit. Bộ giải mã 2 vào / 4 ra đóng vai trò giải mã chọn vùng nhớ, với đầu vào là 2 dây địa chỉ cao, đầu ra là 4 tín hiệu chọn vùng nhớ (băng nhớ) S0-S3. D dây dữ liệu của các băng nhớ được nối song song với nhau, tạo nên bus dữ liệu D bit.

4.3. THIẾT BỊ ĐĨA TỪ

Thiết bị đĩa từ thực hiện chức năng là bộ nhớ ngoài. Thiết bị đĩa từ có dung lượng lưu trữ thông tin cực lớn. Trong hệ thống bộ nhớ nhiều mức, thiết bị đĩa từ đóng vai trò bộ nhớ thứ cấp.

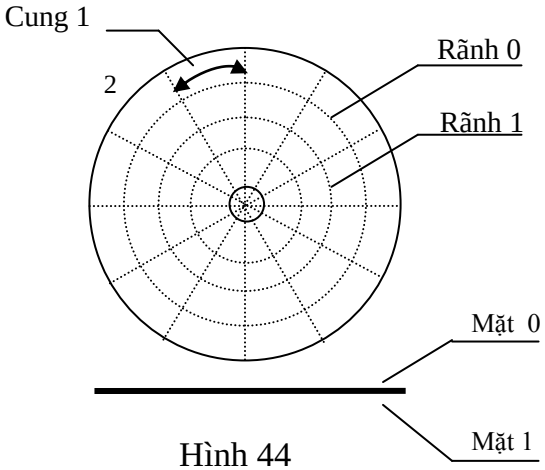
- **Tổ chức lưu trữ thông tin trên đĩa từ ở mức vật lý**

Để có thể sử dụng đĩa từ làm bộ nhớ ngoài cho máy tính cần phân chia không gian lưu trữ của đĩa từ ra thành nhiều đơn vị lưu trữ thông tin và gán địa chỉ cho các đơn vị lưu trữ này. Quá trình phân chia và gán địa chỉ này được gọi là quá trình định dạng cấp thấp (Low Level Formatting) đĩa từ.

Mỗi ổ đĩa đều có một hay nhiều lá đĩa, mỗi lá đĩa có hai mặt (side) và được đánh số tuần tự 0, 1, 2, v.v. từ trên xuống dưới.

Mỗi mặt đĩa được chia thành các vòng tròn đồng tâm cách đều nhau, mỗi vòng tròn được gọi là một rãnh (track). Thông tin được lưu trữ trên các rãnh này. Với đĩa mềm mỗi mặt gồm 80 rãnh. Với đĩa cứng số rãnh quy chuẩn trên một mặt có thể lên tới 1024. Các rãnh được đánh số thứ tự từ ngoài vào trong. Rãnh ngoài cùng (đường kính lớn nhất) có số thứ tự là 0, rãnh tiếp theo là 1, v.v.

Mỗi rãnh được chia thành các cung (sector). Cung là đơn vị lưu trữ thông tin nhỏ nhất trên đĩa. Mỗi một cung tiêu chuẩn chứa được 512 byte dữ liệu. Số cung trên một rãnh tuỳ thuộc vào loại đĩa từ. Với đĩa mềm một rãnh chứa được 18 cung. Với đĩa cứng số cung quy chuẩn trên một rãnh là 63. Các cung trên một rãnh được đánh số thứ tự từ 1 đến tăng dần, theo hướng ngược chiều quay kim đồng hồ (Hình 44).



Hình 44

Các rãnh có cùng đường kính tạo thành một trụ (cylinder). Số thứ tự của rãnh cũng là số thứ tự của trụ.

Việc phân chia rãnh thành các cung với số lượng đều như nhau trên tất các rãnh từ ngoài vào trong (như hình vẽ) chỉ thích hợp cho các đĩa từ có mật độ lưu trữ thấp như đĩa mềm. Kiểu ghi dữ liệu chuẩn này gây lãng phí không gian lưu trữ, đặc biệt là đối với đĩa cứng, vì các rãnh bên ngoài có chu vi lớn hơn và cũng có nghĩa là không gian lưu trữ lớn hơn các rãnh ở trong. Đối với đĩa cứng, để tăng dung lượng lưu trữ thông tin trên đĩa, người ta áp dụng kỹ thuật ghi theo vùng, mỗi vùng có nhiều rãnh. Các vùng kế tiếp nhau từ tâm đĩa ra ngoài sẽ có số cung/rãnh tăng dần lên từ trong ra ngoài. Số lượng cung trên mỗi rãnh trong cùng một vùng là giống nhau. Các ổ đĩa cứng thường có 10 vùng rãnh trở lên.

Thao tác định dạng cấp thấp ghi các byte thông tin định dạng cung và rãnh lên từng rãnh của đĩa từ. Mỗi cung có hai trường chứa thông tin: trường địa chỉ và trường dữ liệu. Trường địa chỉ chứa địa chỉ xác định vị trí mặt, vị trí rãnh và vị trí của cung trên rãnh đó. Bộ ba con số này là duy nhất đối với mỗi

một cung trên đĩa từ và là *địa chỉ vật lý của cung*. Trường dữ liệu được điền bởi các byte dữ liệu giả khi thao tác định dạng cấp thấp được thực hiện.

Đối với đĩa cứng số lượng cung trên một rãnh phụ thuộc vào ổ đĩa và thiết bị điều khiển giao diện trên ổ. Điều đáng lưu ý là để tương thích với BIOS trong máy vi tính, các ổ đĩa cứng phiên dịch thông tin về mật độ thật của rãnh (track) trên mặt đĩa và mật độ thật của số cung (sector) trên một rãnh thành dạng tiêu chuẩn 63 cung/rãnh và cung cấp cho BIOS.

Thiết bị đĩa từ nói riêng và các thiết bị nhớ ngoài khác có đặc điểm chung là có dung lượng lưu trữ thông tin cực lớn. Tuy nhiên các thiết bị nhớ ngoài truyền thống này thường là các thiết bị cơ điện tử, cơ chế truy cập bộ nhớ ngoài được dùng là cơ chế tuần tự, nên tốc độ truy cập thông tin ở bộ nhớ ngoài thấp so với bộ nhớ RAM.

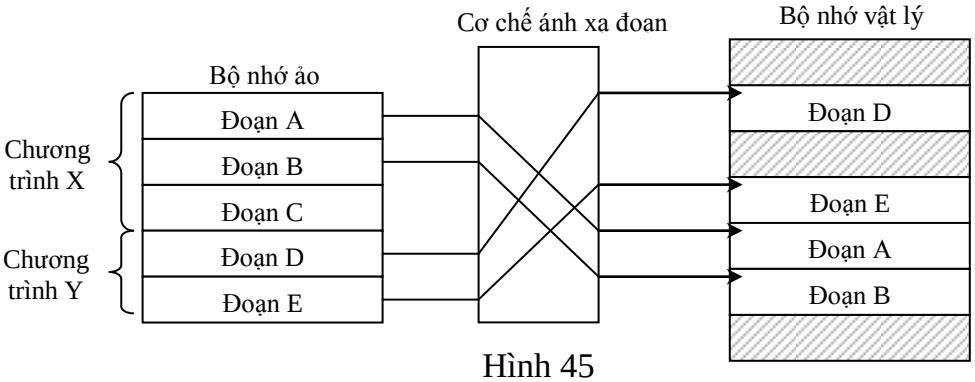
4.4. QUẢN LÝ BỘ NHỚ

Bộ nhớ máy tính được xây dựng trên nhiều loại công nghệ và thiết bị nhớ khác nhau. Bộ nhớ máy tính được tổ chức thành một hệ thống có phân cấp, nhằm đáp ứng được yêu cầu về một bộ nhớ có tốc độ truy nhập cao, dung lượng nhớ lớn và có giá thành thấp. Trong hệ thống bộ nhớ nhiều mức, bộ nhớ chính và bộ nhớ thứ cấp nằm ở hai mức khác nhau, được xây dựng trên cơ sở các thiết bị công nghệ khác nhau, nhưng khi cấp cho người sử dụng thì lại ở dưới dạng một bộ nhớ đơn nhất. Hệ điều hành của máy tính cung cấp cho người sử dụng một bộ nhớ có dung lượng lớn, đơn nhất và có địa chỉ tuyến tính. Kiểu bộ nhớ này còn được gọi là *bộ nhớ ảo* (*vitual memory*). Địa chỉ của ô nhớ trong bộ nhớ ảo gọi là *địa chỉ ảo* hoặc *địa chỉ logic*. Lý do của việc tổ chức, xây dựng và quản lý bộ nhớ ảo là để giải phóng chương trình của người sử dụng khỏi việc phải quản lý trực tiếp vị trí vật lý của các ô nhớ và không gian nhớ, làm cho việc viết và thực hiện chương trình không bị phụ thuộc vào kích thước của bộ nhớ vật lý thực và cấu tạo của nó. Bộ nhớ nhiều mức được quản lý thống nhất còn cho phép tạo ra một bộ nhớ với không gian địa chỉ (ảo) rất lớn, tạo khả năng sử dụng không gian nhớ một cách tối ưu, cũng như cho phép bảo vệ các vùng nhớ trước những truy cập trái phép từ bên trong một quá trình hoặc từ các quá trình khác. Việc quản lý bộ nhớ được thực hiện bởi hệ điều hành của máy tính. Có hai kỹ thuật được sử dụng để quản lý bộ nhớ ảo, đó là kỹ thuật *phân đoạn* (*segmentation*) và kỹ thuật *phân trang* (*paging*).

4.4.1. Quản lý bộ nhớ theo phân đoạn

Cơ chế quản lý bộ nhớ ảo theo phân đoạn (cơ chế phân đoạn) là cơ chế trong đó bộ nhớ được chia nhỏ, được định vị và giám sát theo đoạn. Phân đoạn (segmentation) cho phép người viết chương trình nhìn bộ nhớ (ảo) như là tập hợp của các đoạn (segment), do vậy người viết chương trình có thể tổ chức, quản lý chương trình và dữ liệu của mình một cách khá thuận tiện, như tổ

chức chương trình thành các module chương trình chính, chương trình con và các module dữ liệu. Trong cơ chế phân đoạn mỗi một chương trình không được xem là một chuỗi liên tục của mã lệnh và dữ liệu mà được chia thành các module mã lệnh, dữ liệu, ngăn xếp v.v. Hệ điều hành cấp một vùng nhớ (một vùng địa chỉ) trong bộ nhớ ảo cho mỗi module mã lệnh và dữ liệu của chương trình cần được thực thi, các vùng nhớ đó được gọi là đoạn (segment). Mỗi một đoạn được hệ điều hành gán một tên hoặc một địa chỉ, gọi là địa chỉ đoạn (Hình 45). Quản lý bộ nhớ theo phân đoạn cho phép gán các thuộc tính như mức đặc quyền và quyền truy nhập cho đoạn v.v., qua đó tạo khả năng quản lý các đoạn chương trình và dữ liệu đó. Việc phân đoạn còn cho phép hệ điều hành quản lý nhiều chương trình cùng chạy đồng thời trong bộ nhớ và sử dụng tối ưu không gian nhớ (Hình 45).

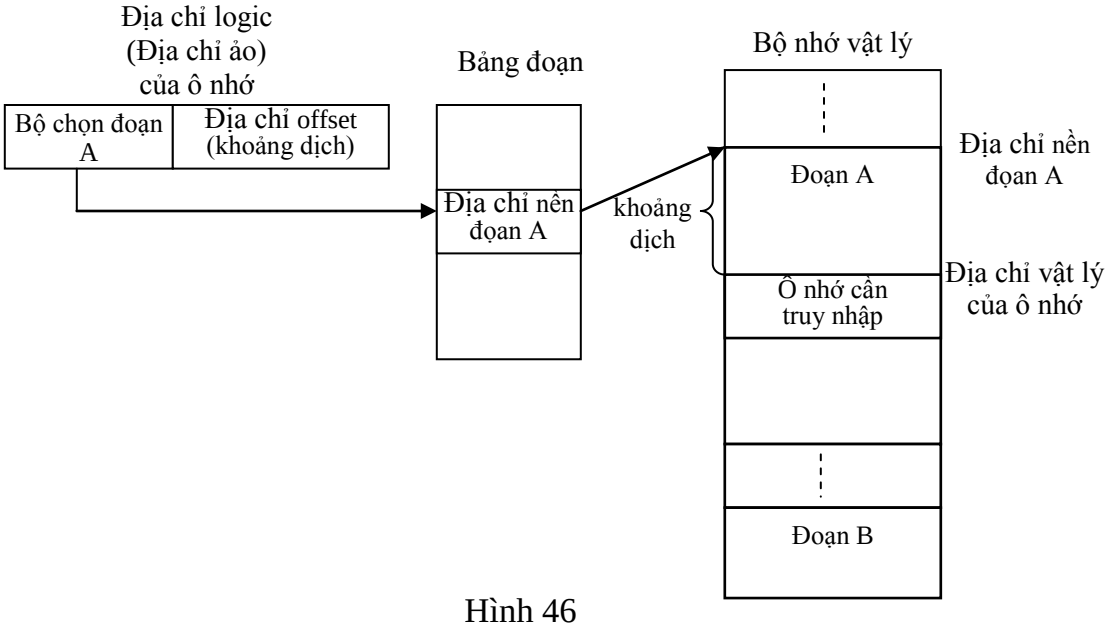


Hình 45

Mỗi một ô nhớ trong mỗi đoạn được định vị bằng một địa chỉ ảo, địa chỉ ảo còn được gọi là *địa chỉ logic* của ô nhớ. Địa chỉ logic là một cặp con số: *địa chỉ đoạn* và *địa chỉ offset*. Địa chỉ đoạn xác định vị trí của đoạn trong bộ nhớ ảo. Địa chỉ offset là con số xác định vị trí của ô nhớ (khoảng dịch-displacement) so với nền đoạn chứa nó. Về mặt vật lý, chương trình cần được thực thi được chứa ở bộ nhớ thứ cấp. Khi chương trình được thực hiện, nó được hệ điều hành nạp từ bộ nhớ thứ cấp vào bộ nhớ chính. Việc chuyển chương trình từ bộ nhớ thứ cấp vào bộ nhớ chính (bộ nhớ vật lý) và truy nhập bộ nhớ chính được thực hiện theo cơ chế ánh xạ địa chỉ (cơ chế ánh xạ đoạn). Việc ánh xạ không gian nhớ ảo sang bộ nhớ vật lý thực và chuyển địa chỉ ảo sang địa chỉ vật lý thực có thể được thực hiện theo nhiều cơ chế. Địa chỉ thực của ô nhớ vật lý có thể được tính từ địa chỉ nền đoạn vật lý và địa chỉ offset, trong đó địa chỉ nền đoạn vật lý có thể được tính trực tiếp từ địa chỉ đoạn (ảo), hoặc được xác định qua *bộ chọn đoạn* và *bảng đoạn*. Bảng đoạn chứa địa chỉ nền vật lý của các đoạn (Hình 46), nằm ở bộ nhớ chính. Trong trường hợp sử dụng bảng đoạn, địa chỉ logic của ô nhớ là một cặp con số: *bộ chọn đoạn* và *địa chỉ offset*. Bộ chọn đoạn là con số chỉ ra nơi chứa địa chỉ vật lý nền đoạn trong bảng đoạn (Hình 46).

Kiểu tổ chức và quản lý bộ nhớ theo phân đoạn mang lại hàng loạt lợi ích:

- Do mỗi cấu trúc dữ liệu được gán một đoạn riêng (có địa chỉ và kích thước riêng), nên hệ điều hành có thể co giãn đoạn nếu cần, ví dụ như với trường hợp cấu trúc dữ liệu kiểu ngăn xếp. Nhờ vậy mà người viết chương trình không buộc phải dự đoán về sự thay đổi của kích thước dữ liệu để đăng ký sử dụng bộ nhớ.
- Cho phép người lập trình có thể thay đổi chương trình và dịch lại từng phần chương trình mà không cần phải dịch và liên kết lại toàn bộ chương trình.
- Phương pháp quản lý bộ nhớ thông qua bảng đoạn tạo khả năng quản lý không gian nhớ ảo rất lớn.
- Cho phép hệ điều hành sử dụng hiệu quả các vùng nhớ được giải phóng khi một quá trình nào đó kết thúc.
- Cho phép gán các thuộc tính bảo vệ hoặc quyền truy nhập cho đoạn, qua đó có thể thực hiện cơ chế bảo vệ đoạn chứa chương trình hoặc dữ liệu.



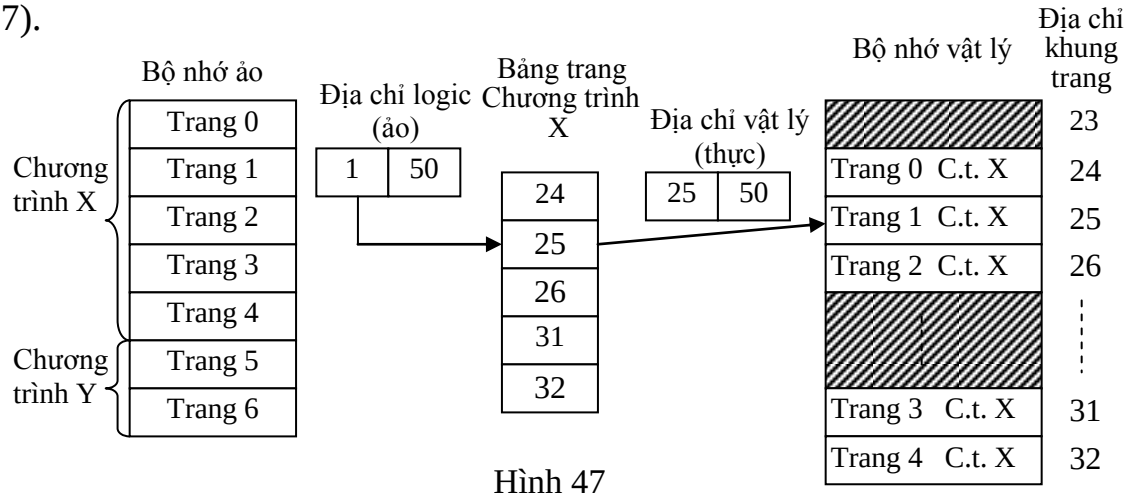
Hình 46

4.4.2. Quản lý bộ nhớ theo phân trang

Kiểu quản lý bộ nhớ ảo theo phân đoạn rất thích hợp cho việc sử dụng tối ưu bộ nhớ và quản lý tiến trình và cũng thuận tiện cho cả người lập trình. Tuy nhiên khi quản lý một bộ nhớ có nhiều mức, hệ điều hành còn luôn phải thực hiện thao tác hoán đổi dữ liệu giữa các mức nhớ, ví dụ như hoán đổi dữ liệu giữa bộ nhớ chính (RAM) và bộ nhớ thứ cấp (thiết bị đĩa từ v.v.). Phương pháp quản lý bộ nhớ phù hợp ở đây là phương pháp *phân trang* (*paging*), trong đó bộ nhớ ảo được chia thành các trang. Đơn vị thông tin cơ bản được

dùng trong việc hoán đổi là một khối có kích thước cố định, ví dụ 4 Kbyte, gọi là *trang* (*page*). Mỗi một chương trình hoặc dữ liệu khi được thực hiện sẽ được cấp một vùng nhớ trong bộ nhớ ảo, gồm các trang (ảo) cùng địa chỉ trang tương ứng và người viết chương trình không nhìn thấy điều này. Địa chỉ của ô nhớ trong bộ nhớ ảo, còn gọi là *địa chỉ ảo* hoặc *địa chỉ logic*, là một cặp con số: *địa chỉ trang* và *khoảng dịch*. Để phù hợp với cách quản lý bộ nhớ theo phân trang, bộ nhớ chính vật lý cũng được chia thành các khung trang (*page frames*). Khung trang có kích thước giống như trang của bộ nhớ ảo.

Chương trình cần được thực hiện nằm ở bộ nhớ thứ cấp, được coi như là tập hợp của các trang chương trình, sẽ được nạp vào bộ nhớ chính theo từng trang khi có yêu cầu và từ bộ nhớ chính các lệnh máy mới được nhập vào CPU để thực thi. Việc chuyển các trang chương trình từ bộ nhớ thứ cấp vào bộ nhớ chính (bộ nhớ vật lý) và truy nhập bộ nhớ chính được thực hiện theo cơ chế ánh xạ địa chỉ (cơ chế chuyển địa chỉ). Quá trình chuyển địa chỉ ảo và địa chỉ trang ảo thành địa chỉ khung trang và địa chỉ vật lý được thực hiện nhờ một *bảng trang* (Hình 47). Bảng trang là tập hợp của các *mục bảng trang*, trong đó mỗi mục bảng trang chứa một địa chỉ vật lý của khung trang. Bảng trang, có một tên gọi khác là *bảng địa chỉ bộ nhớ*, nằm ở bộ nhớ vật lý. Địa chỉ trang (ảo) xác định vị trí của mục bảng trang, nơi chứa địa chỉ vật lý của khung trang tương ứng. Địa chỉ vật lý của ô nhớ cần truy cập trong bộ nhớ vật lý được tính bằng cách cộng địa chỉ khung trang vật lý với khoảng dịch (Hình 47).



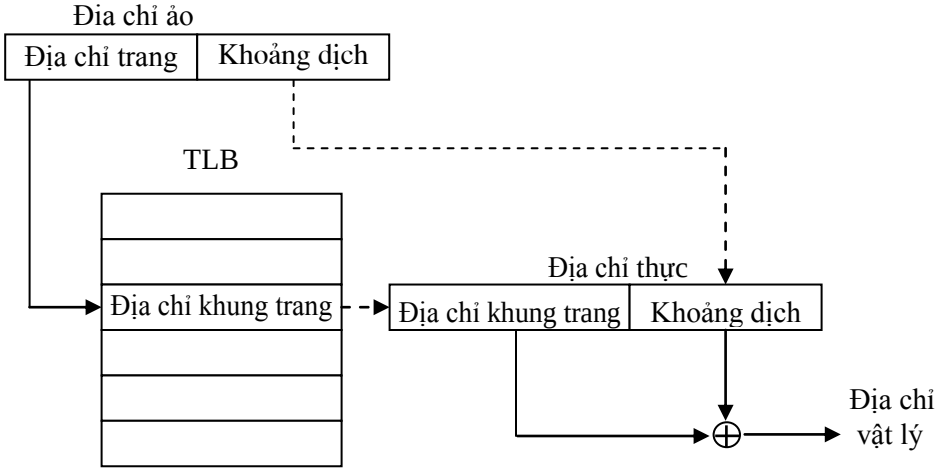
Hình 47

Mỗi một phương pháp quản lý bộ nhớ, kiểu phân đoạn hay phân trang, đều có những ưu điểm riêng, do vậy trên nhiều hệ thống máy tính người ta đã kết hợp hai phương pháp này với nhau để tạo nên một phương pháp kết hợp quản lý bộ nhớ kiểu phân đoạn-phân trang. Máy vi tính PC sử dụng hệ điều hành Windows là một ví dụ về cách quản lý bộ nhớ theo kiểu kết hợp này.

Bộ đệm TLB.

Với cơ chế quản lý theo phân trang, mỗi một lần quy chiếu bộ nhớ cần hai lần truy cập bộ nhớ vật lý, một lần để đọc mục bảng trang và một lần để truy cập dữ liệu trong bộ nhớ vật lý. Nếu như vậy cơ chế quản lý này sẽ làm tăng gấp đôi thời gian truy cập bộ nhớ. Để giảm thời gian truy cập bộ nhớ, một phần của bảng trang, gồm các mục bảng trang được truy cập thường xuyên nhất, được đặt trong đơn vị quản lý bộ nhớ MMU (Memory Management Unit) của đơn vị xử lý trung tâm CPU. Phần chứa thông tin này được gọi là bộ đệm TLB (*Translation Look-aside Buffer*) (Hình 48).

Khi cần truy cập bộ nhớ vật lý, bộ đệm TLB được đọc đầu tiên và địa chỉ trang ảo (trong địa chỉ logic) được dùng để chỉ ra mục bảng trang (nơi chứa địa chỉ khung trang vật lý) ở trong TLB. Địa chỉ khung trang này được cộng với khoảng dịch để tạo ra địa chỉ vật lý của ô nhớ.



Hình 48

4.5. TỔ CHỨC BỘ NHỚ CACHE

Yêu cầu về một bộ nhớ máy tính có dung lượng lớn, có thời gian truy nhập đáp ứng được đòi hỏi của CPU nhưng lại phải có giá thành không đắt, chỉ có thể được giải quyết bằng một bộ nhớ kiểu nhiều mức, trong đó các bộ nhớ vật lý có thời gian truy nhập ngắn hơn được đặt ở mức cao hơn. Bộ nhớ vật lý có thời gian truy nhập đáp ứng được đòi hỏi của CPU được đặt ngay sau CPU, trước bộ nhớ chính, là bộ nhớ SRAM (Hình 37, mục 3.2). Bộ nhớ SRAM có thời gian truy nhập ngắn hơn nhiều so với bộ nhớ chính được xây dựng từ các chip DRAM. Bộ nhớ SRAM tuy có thời gian truy nhập rất thấp nhưng lại đắt tiền, nên chỉ được sử dụng với một dung lượng nhớ hợp lý. Bộ nhớ SRAM được dùng ở đây với tính chất một bộ nhớ đệm và được gọi là *bộ nhớ cache* (*bộ nhớ ẩn*). Bộ nhớ cache được sử dụng để lưu trữ các lệnh và dữ liệu thường được gọi đến trong quá trình thực hiện chương trình. Việc quy chiếu (truy nhập) đến bộ nhớ ở mức vật lý thực chất là truy cập vào bộ nhớ cache. Khi CPU phát ra địa chỉ quy chiếu tới bộ nhớ chính, địa chỉ này được

dùng để quy chiếu tới cache. Việc quy chiếu đến cache được gọi là "trúng" (hit) nếu ô nhớ ứng với địa chỉ quy chiếu có trong cache và gọi là "trượt" (miss) nếu không tìm thấy ô nhớ cần quy chiếu ở trong cache, khi đó phải truy cập đến bộ nhớ chính. Việc truy nhập đến bộ nhớ chính chỉ xảy ra khi không tìm thấy thông tin cần có trong cache.

Cache ánh xạ trực tiếp

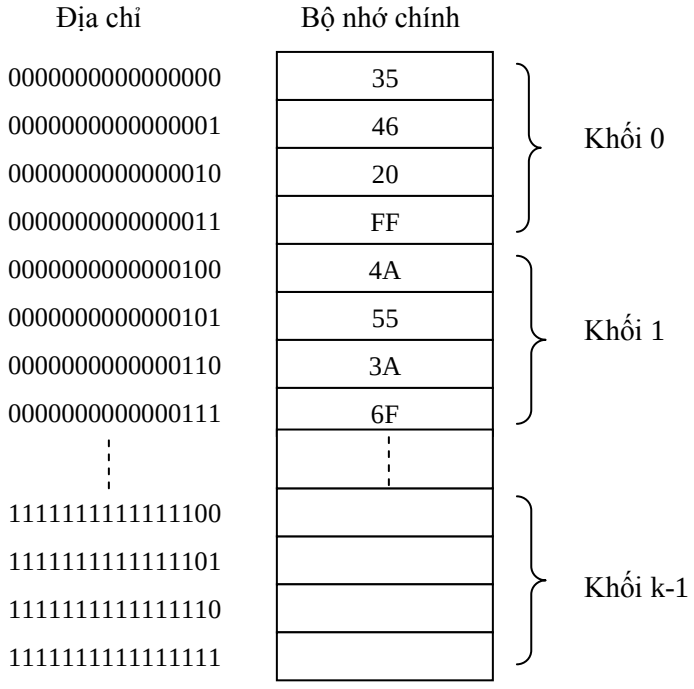
Phương pháp đơn giản nhất để tổ chức bộ nhớ cache là phương pháp ánh xạ trực tiếp, trong đó mỗi khối nhớ trong bộ nhớ chính được ánh xạ vào một dòng cache nhất định.

Bộ nhớ chính được chia thành nhiều khối nhớ, mỗi khối nhớ gồm 1 hoặc nhiều ô nhớ liền nhau (Hình 49).

Bộ nhớ cache gồm nhiều khối, gọi là *khối cache* hoặc *dòng cache*. Mỗi khối cache chứa các thông tin như: số hiệu thẻ, bit cờ và bản thân khối dữ liệu, trong đó khối dữ liệu trong cache là bản sao của khối dữ liệu trong bộ nhớ chính. Cấu trúc một khối cache như sau:

Số thẻ	F	Khối dữ liệu

Địa chỉ bộ nhớ chính được coi là gồm nhiều phần, trong đó có một phần bit địa chỉ sẽ đóng vai trò địa chỉ khối cache. Giả sử bộ nhớ chính có địa chỉ 16 bit và cache có 16 dòng (khối) cache, mỗi dòng cache chứa một khối dữ liệu 4 byte. Khi đó 2 bit thấp nhất trong địa chỉ xác định vị trí byte trong khối, 4 bit cao hơn tiếp theo xác định vị trí khối cache, 10 bit cao nhất xác định số thẻ. Số thẻ ở đây được dùng để đánh dấu sự khác biệt của các khối nhớ có cùng vị trí ánh xạ (vị trí khối cache) trên bộ nhớ cache.



Hình 49

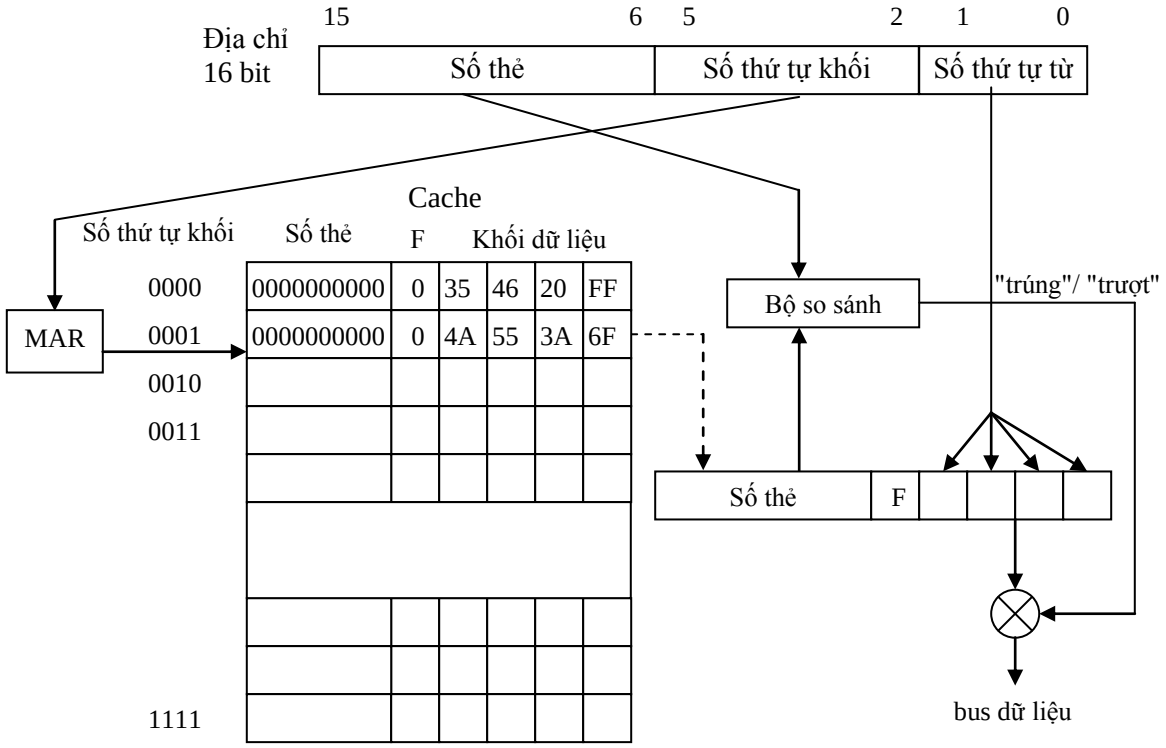
Tổ chức cache ánh xạ trực tiếp được thực hiện theo cách từng vùng khối nhớ được ánh xạ (sao chép) vào cache, trong đó mỗi khối nhớ được sao chép lên một dòng cache nhất định (Bảng 7) cùng với số thẻ tương ứng. Với bộ nhớ cache có *c* dòng (khối) cache, khối nhớ thứ *j* trong bộ nhớ chính sẽ được đặt tại dòng cache *i* theo công thức

$$i = j \bmod c$$

Bảng 7

Số thứ tự khối nhớ của bộ nhớ chính				**Số thứ tự dòng (khối) cache**
0	c	... .	k - c	0
1	c+1	... .	k - c+1	1
2	c+2	... .	k - c+2	2
c - 1	2c -1	... .	k - 1	c - 1

Quá trình truy nhập cache được tiến hành như sau (Hình 50).



Hình 50

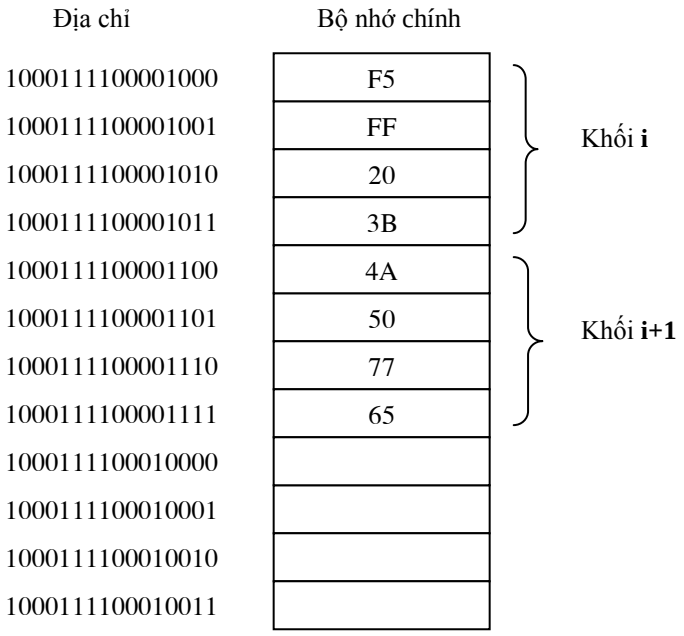
Khi CPU phát một địa chỉ quy chiếu tới bộ nhớ, bộ điều khiển cache sẽ tách phần các bit xác định số khối ra và theo đó truy cập vào vị trí khối cache tương ứng. Tiếp đó bộ điều khiển cache sẽ thực hiện so sánh phần bit địa chỉ xác định số thẻ với số thẻ của chính khối cache này. Nếu so sánh thấy trùng có nghĩa truy nhập "trúng", khi đó phần bit địa chỉ xác định vị trí byte dữ liệu được dùng để truy nhập dữ liệu trong khối cache. Nếu thao tác truy nhập bộ nhớ là đọc, thì dữ liệu được đưa ra cho CPU, nếu là ghi bộ nhớ, thì dữ liệu sẽ được chuyển từ CPU vào cache. Nếu phần bit địa chỉ xác định số thẻ không trùng với bất kỳ số thẻ nào trong cache thì có nghĩa là ô nhớ cần truy nhập không có trong cache, truy nhập "trượt", khi đó cần phải truy nhập tới bộ nhớ chính.

Tổ chức cache theo kiểu ánh xạ trực tiếp đơn giản và không đắt, nhưng có nhược điểm là mỗi khối nhớ chỉ có một vị trí cố định trong cache. Nếu xảy ra hiện tượng chương trình quy chiếu nhiều lần đến cùng vị trí khối cache (dòng cache), khi đó sẽ phải liên tục thực hiện thao tác hoán đổi các khối, điều này làm cho thời gian truy cập trung bình lớn lên.

Cache ánh xạ liên kết.

Một kỹ thuật tổ chức cache khác là *kỹ thuật ánh xạ liên kết*, trong đó mỗi *khối của bộ nhớ chính* được ánh xạ (sao chép) vào *bất kỳ dòng nào* trong cache.

Bộ nhớ chính, được chia thành nhiều khối nhớ, mỗi khối nhớ gồm 1 hoặc nhiều ô nhớ liền nhau. Giả sử bộ nhớ có địa chỉ 16 bit. Có thể chia bộ nhớ thành các khối, mỗi khối 4 byte. Mỗi khối nhớ có một địa chỉ khối, xác định bởi 14 bit địa chỉ cao, vị trí của từng ô trong khối được xác định bởi 2 bit địa chỉ thấp (Hình 51).



Hình 51

Cache bao gồm nhiều khối cache (dòng cache). Cấu trúc một khối cache như sau:

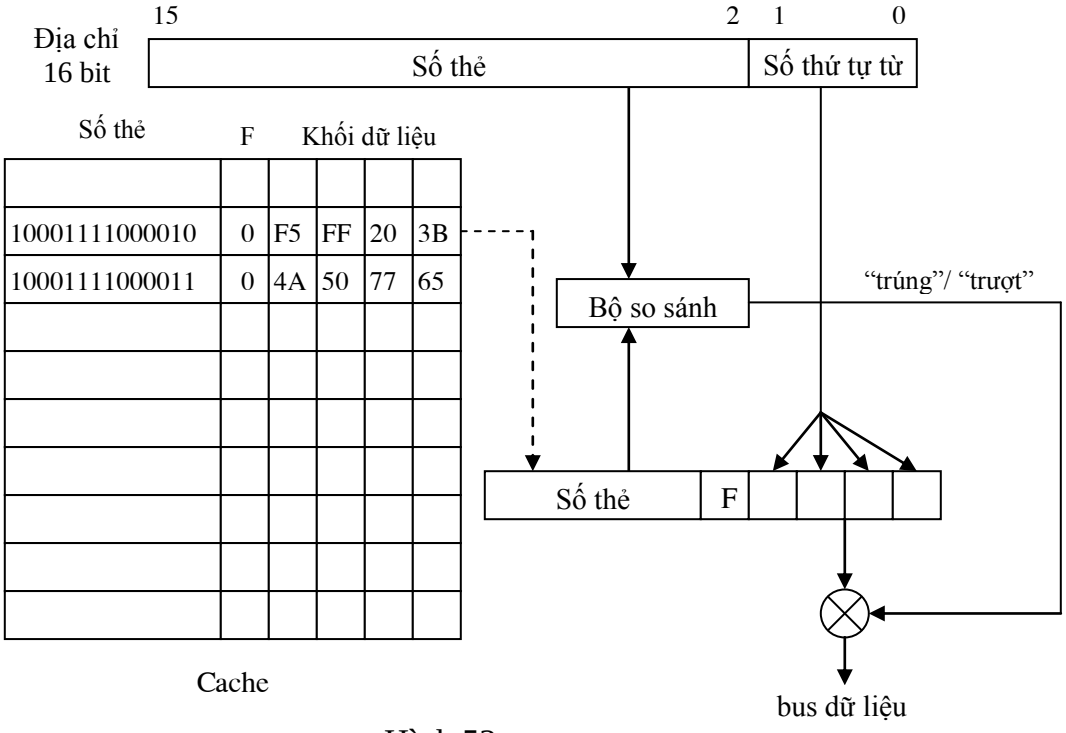
Số thẻ	F	Khối dữ liệu

Số thẻ ở đây mang thông tin về vị trí (địa chỉ) của khối dữ liệu (khối nhớ) trong bộ nhớ chính. Thực chất số thẻ là phần các bit địa chỉ cao trong địa chỉ của khối nhớ (đóng vai trò địa chỉ khối) trong bộ nhớ chính. Khối dữ liệu trong cache là bản sao của khối dữ liệu trong bộ nhớ chính. Bit cờ F được dùng trong việc quản lý ghi-đọc cache.

Đơn vị quản lý bộ nhớ sao chép các khối nhớ trong bộ nhớ chính vào cache theo cách: sao nội dung của từng khối nhớ vào phần dữ liệu trong khối cache, đồng thời tạo ra số thẻ tương ứng trong khối cache.

Quá trình truy nhập cache được tiến hành như sau (Hình 52).

Địa chỉ của ô nhớ cần truy cập được gửi ra bus địa chỉ. Bộ điều khiển cache tách lấy phần địa chỉ khối là các bit địa chỉ cao, ví dụ ở đây là 14 bit địa chỉ cao, so sánh với các số thẻ trong cache. Nếu tìm thấy số thẻ trùng với địa chỉ khối, truy nhập "trúng", khi đó 2 bit địa chỉ thấp được dùng để truy nhập đến từ dữ liệu trong cache, thao tác truy nhập cache được tiến hành. Nếu phần địa chỉ khối nhớ không trùng với bất kỳ số thẻ nào, có nghĩa là ô nhớ cần truy cập không có trong cache, việc truy cập được gọi là "trượt". Khi đó cần phải truy cập tới bộ nhớ chính. Khối dữ liệu ứng với địa chỉ cần truy cập được đưa từ bộ nhớ chính vào cache, đồng thời từ dữ liệu tương ứng được gửi đến CPU hoặc được chuyển từ CPU đến cache.



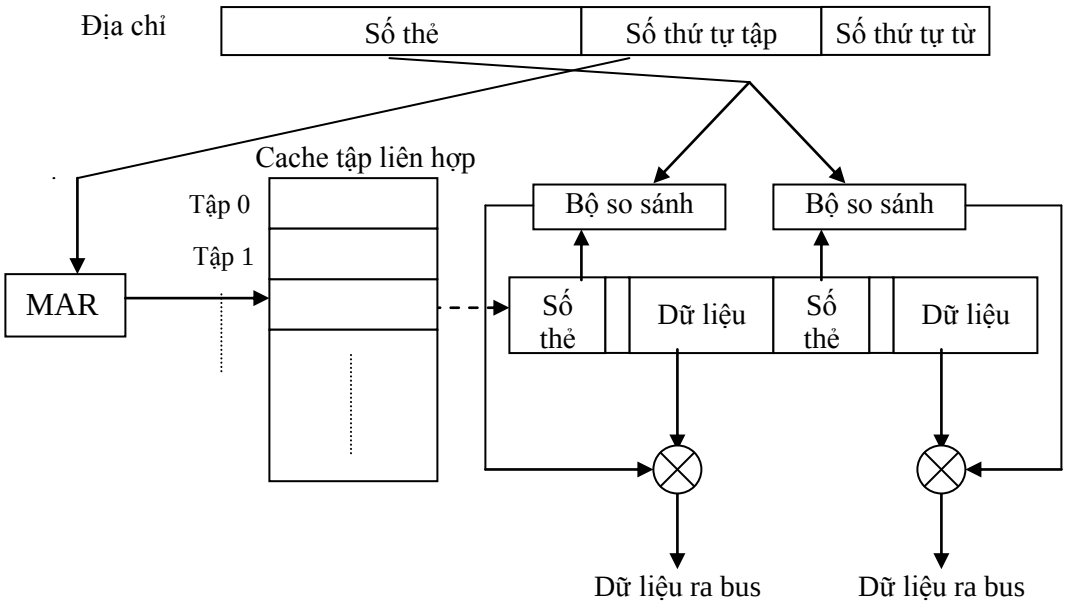
Hình 52

Ưu điểm của phương pháp ánh xạ liên kết là cho phép nạp linh hoạt từng khối nhớ trong bộ nhớ chính vào cache. Nhược điểm của phương pháp này là cần có một mạch phần cứng thực hiện được việc kiểm tra so sánh đồng thời tất cả các thẻ trong cache khi truy nhập. Điều này làm cho cache có giá thành cao.

Cache tập liên kết

Kỹ thuật tập liên kết kết hợp được các thế mạnh của hai kỹ thuật ánh xạ trực tiếp và kỹ thuật ánh xạ liên kết. Cache tập liên kết được chia thành nhiều tập (*dòng tập*), mỗi tập gồm 2 hoặc 4 dòng cache. Các dòng cache này tạo thành một tập liên kết. Mỗi khối dữ liệu trong bộ nhớ chính có thể được ánh xạ vào bất kỳ dòng tập nào. Cách tổ chức này còn giải quyết được tình huống có thể xảy ra với kỹ thuật ánh xạ trực tiếp, khi chương trình thường xuyên quy chiếu đến các lệnh hoặc dữ liệu cùng chung một dòng cache.

Địa chỉ bộ nhớ chính được coi là tập hợp của 3 trường bit địa chỉ: số thẻ, số thứ tự (dòng) tập và số thứ tự từ dữ liệu trong tập. Khi địa chỉ ô nhớ được CPU phát ra, bộ điều khiển cache tách lấy phần số tập để truy nhập cache. Khi dòng tập trong cache được truy nhập thì cả 2 hay nhiều thẻ trong dòng tập này cùng được kiểm tra, để xác định việc truy nhập vào khối cache. Các bit địa chỉ xác định vị trí từ dữ liệu sẽ được dùng để truy nhập từ dữ liệu (Hình 53).



Hình 53

Kỹ thuật ghi bộ nhớ

Kỹ thuật ghi bộ nhớ có cache cần đảm bảo dữ liệu được ghi đến ô nhớ đích ở bộ nhớ chính, nhưng với thời gian ngắn nhất. Có một vài kỹ thuật được dùng để ghi dữ liệu vào bộ nhớ có cache, là *kỹ thuật ghi xuyên* (*write through*) và *kỹ thuật sao lưu* (*copy back*).

Với kỹ thuật ghi xuyên (cache ghi xuyên) dữ liệu được ghi lên cả cache lẫn bộ nhớ chính cùng một lúc, không dùng đến bit cờ F. Kỹ thuật này làm thời gian ghi bộ nhớ tăng lên.

Với kỹ thuật sao lưu (cache sao lưu), mỗi khối cache có thêm một bit cờ F đánh dấu sự thay đổi nội dung dữ liệu. Khi khối dữ liệu lần đầu tiên được đặt vào cache, cờ F được đặt bằng 0. Khi thực hiện ghi bộ nhớ, dữ liệu chỉ được ghi vào cache và bit cờ F được lập (F = 1). Giá trị F=1 ghi nhận nội dung cache khác với nội dung ở bộ nhớ chính. Sau đó, nếu khối dữ liệu trong cache cần được thay thế bằng khối dữ liệu khác từ bộ nhớ chính trong trường hợp truy nhập "trượt", thì bit cờ F được kiểm tra để xác định xem có cần thực hiện thao tác sao lưu dữ liệu trên cache vào bộ nhớ chính hay không. Nếu F = 1 thì cần thực hiện sao lưu, nếu F = 0 thì không cần sao lưu. Kỹ thuật này làm tăng tốc độ truy cập bộ nhớ và được gọi là *kỹ thuật sao lưu có dựng cờ*.

4.6. RAID

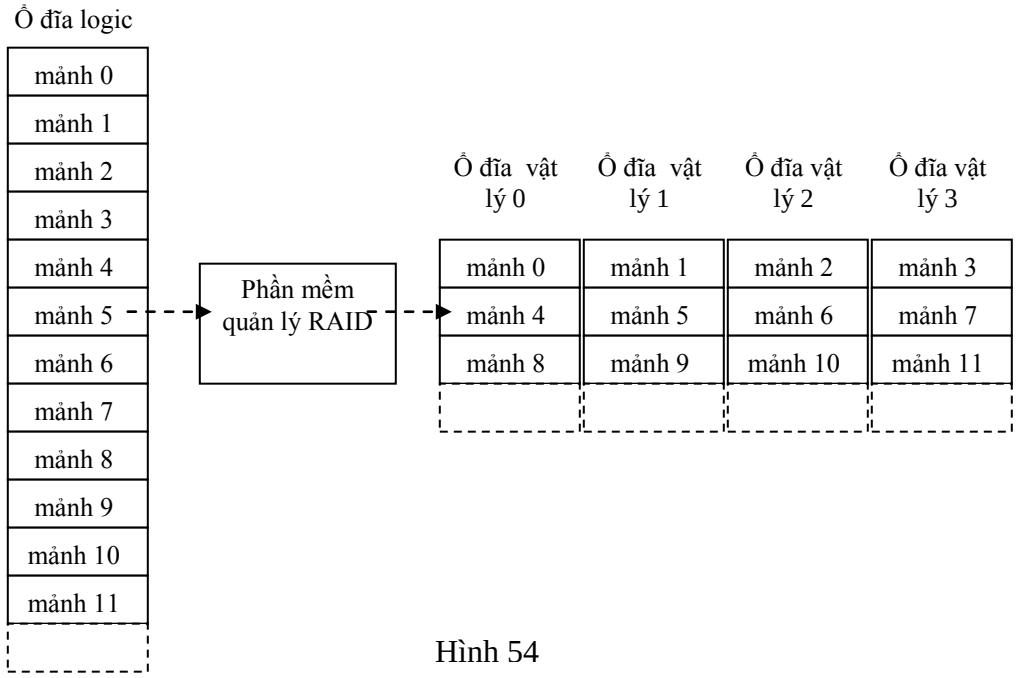
Các thiết bị đĩa từ được dùng làm bộ nhớ thứ cấp trong hệ thống bộ nhớ. Các thiết bị này cung cấp khả năng lưu trữ dữ liệu cao với một giá thành thấp, nhưng cũng có một vài nhược điểm. Tốc độ truy nhập dữ liệu của thiết bị đĩa từ, dù luôn được cải thiện theo sự phát triển của công nghệ chế tạo, luôn thấp so với yêu cầu của CPU. Do chúng là những thiết bị cơ điện tử, nên nguy cơ có lỗi hoặc hỏng hóc qua một thời gian sử dụng là có thể xảy ra. Để nâng cao hiệu xuất làm việc và nâng cao độ tin cậy của bộ nhớ-đĩa từ, các nhà nghiên cứu ở trường đại học California đã đề xuất một thiết kế bộ nhớ-đĩa từ có tên RAID (Redundant Array of Inexpensive Disks). Ý tưởng chính của hệ thống RAID là thay một ổ đĩa có dung lượng lớn bằng nhiều ổ đĩa có dung lượng nhỏ hơn làm việc song song và phân bố dữ liệu lưu trữ theo cách sao cho có thể truy nhập dữ liệu này ở đồng thời tất cả các ổ đĩa này. Bằng cách này, tốc độ truy nhập dữ liệu vào *n* ổ đĩa hoạt động song song sẽ nhanh hơn so với vào 1 ổ đĩa lớn. Cũng nhờ kiểu tổ chức lưu trữ dữ liệu phân tán trên nhiều ổ đĩa song hành mà có thể nâng cao được độ tin cậy của việc lưu trữ.

Sơ đồ RAID có nhiều mức, từ RAID-0 đến RAID-5. Các mức RAID khác nhau cung cấp các cơ chế lưu trữ, khả năng dự phòng và sửa lỗi dữ liệu khác nhau. Ở RAID mức 0, *n* ổ đĩa được kết hợp hoạt động song song để tăng hiệu suất làm việc của bộ nhớ, không có dự phòng hoặc tự sửa lỗi ở mức này. Ở RAID mức1, có *2n* ổ đĩa hoạt động song song, trong đó một dữ liệu được ghi đồng thời lên 2 ổ đĩa chạy song hành. Cơ chế này tạo tính dự phòng cho dữ liệu. Bốn tổ chức RAID từ RAID-2 đến RAID-5 ít tính dự phòng hơn, nhưng lại tạo khả năng tự kiểm tra và sửa lỗi dữ liệu.

- **RAID-0**

Tổ chức RAID mức 0 sử dụng nhiều ổ đĩa hoạt động song hành. Một khối dữ liệu sẽ được phân mảnh và phân bố trên tất cả các ổ đĩa. Kiểu lưu trữ này tạo khả năng truy nhập dữ liệu nhanh hơn hơn so với truy nhập trên một ổ đĩa dung lượng lớn. Mặt khác, nếu có hai yêu cầu truy nhập đến hai khối dữ liệu khác nhau, thì nhiều khả năng các khối dữ liệu này nằm trên các ổ dĩa khác nhau. Như vậy cả hai yêu cầu có thể được đáp ứng đồng thời. Điều này giúp giảm thời gian xếp hàng đợi phục vụ và như vậy cũng làm tăng hiệu năng truy nhập bộ nhớ.

Ở mức RAID-0, một khối dữ liệu sẽ được phân mảnh và phân bố trên tất cả các ổ đĩa vật lý. Dữ liệu được nhìn như được chứa trong một ổ đĩa logic. Ổ logic được chia thành các mảnh logic liền nhau, các mảnh này có thể là các khối vật lý, các cung. Các mảnh được ánh xạ lên ổ vật lý theo kiểu tuần tự vòng tròn (Hình 54).

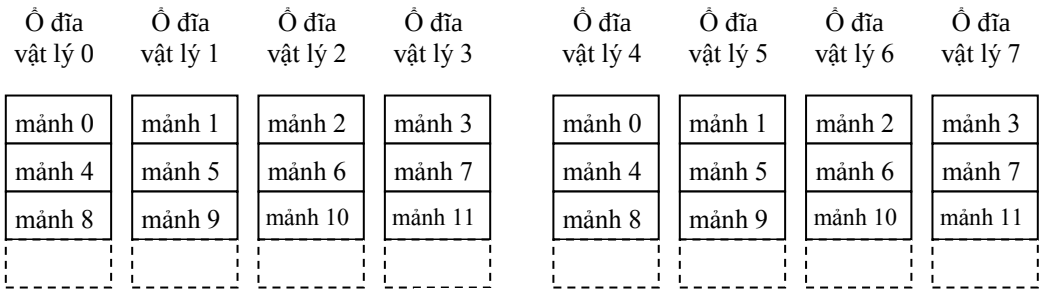


Hình 54

Phần mềm quản lý RAID thực hiện ánh xạ giữa không gian ổ đĩa logic các không gian đĩa vật lý.

- **RAID-1**

RAID-1 là tổ chức RAID có dự phòng. Tổ chức dự phòng được thực hiện một cách đơn giản là tăng gấp đôi việc ghi và lưu trữ dữ liệu. Ở đây phải sử dụng 2n ổ đĩa vật lý (Hình 55). Ưu điểm của tổ chức RAID-1 là việc khôi phục dữ liệu bị lỗi khá đơn giản, vì khi một ổ đĩa nào đó bị lỗi, dữ liệu vẫn còn ở ổ đĩa dự phòng còn lại. Nhược điểm của RAID-1 là giá thành hệ thống cao, do đòi hỏi lượng ổ đĩa tăng gấp đôi.



Hình 55

- **RAID-2**

Tính dự phòng của RAID-2 ít hơn RAID-1, do dùng ít ổ đĩa hơn. RAID-2 là tổ chức RAID có khả năng tự sửa lỗi. RAID-2 sử dụng mã sửa lỗi. Mã sửa lỗi được tính từ các bit trong từ dữ liệu chứa ở hệ thống ổ nhớ chính và được chứa ở vị trí tương ứng trên một ổ đĩa chứa mã sửa lỗi. Khi đọc dữ liệu, tất cả các ổ đĩa đồng thời được truy nhập. Dữ liệu và mã sửa lỗi tương ứng được cấp cho bộ điều khiển. Nếu thấy lỗi, bộ điều khiển phát hiện và sửa lỗi ngay rồi cung cấp cho người yêu cầu. Tổ chức RAID-2 rẻ hơn RAID-1, nhưng số lượng ổ đĩa phục vụ dự phòng và sửa lỗi tỷ lệ thuận với lượng ổ đĩa chứa dữ liệu.

- **RAID-3**

RAID-3 được tổ chức như RAID-2, nhưng chỉ yêu cầu một ổ đĩa dự phòng. Thay vì phải tính mã sủa lỗi như ở RAID-2, RAID-3 chỉ tính một bit parity đơn giản cho mỗi dữ liệu. Một ổ đĩa dự phòng sẽ chứa tất cả các bit parity. Giả sử có một từ dữ liệu $b_{i,\,n-1}b_{i,\,n-2}\ldots\ldots b_{i,\,1}\ b_{i,\,0}\ p_i$. Khi đó bit parity p_i được tính theo công thức sau:

$$p_i = b_{i,\,n-1} \oplus b_{i,\,n-2} \oplus \ldots\ldots \oplus b_{i,\,j} \oplus \ldots\ldots \oplus b_{i,\,0}$$

Nếu một lỗi được phát hiện trên ổ thứ j thì bit lỗi $b_{i,\,j}$ trên ổ j sẽ được khôi phục từ các ổ còn lại như sau:

$$b_{i,\,j} = b_{i,\,n-1} \oplus b_{i,\,n-2} \oplus \ldots\ldots \oplus b_{i,\,j\,-1} \oplus b_{i,\,j\,+1} \oplus \ldots\ldots \oplus b_{i,\,0} \oplus p_i$$

- **RAID-4 và RAID-5**

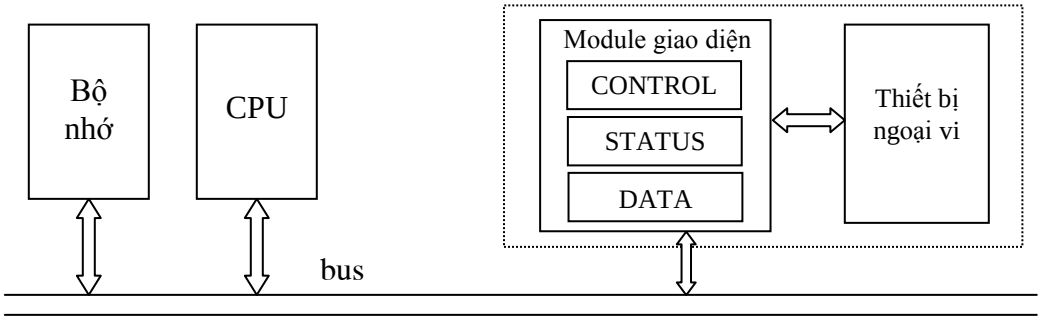
Tổ chức RAID-4 và RAID-5 tương tự như RAID-3, ngoại trừ trên RAID-4 các mảnh dữ liệu có kích thước tương đối lớn và parity được tính trên các mảnh ở từng ổ đĩa dữ liệu, nhưng vẫn được chứa trên một ổ đĩa riêng. Ở RAID-5 thì các mảnh parity được phân bổ trên tất cả các ổ đĩa.

Chương 5
HỆ THỐNG VÀ CÁC PHƯƠNG PHÁP VÀO-RA DỮ LIỆU

5.1. HỆ THỐNG VÀO-RA DỮ LIỆU TRONG MÁY TÍNH

Chức năng chính của máy tính là thực hiện chương trình, qua đó xử lý dữ liệu. Đơn vị xử lý trung tâm CPU nhập lệnh và dữ liệu từ bộ nhớ chính, xử lý và đưa kết quả ra bộ nhớ. Chương trình và các dữ liệu nguyên thuỷ được nhập từ ngoài vào, kết quả xử lý dữ liệu được đưa ra ngoài cho người sử dụng thông qua hệ thống vào-ra (nhập -xuất) dữ liệu. Các thiết bị đầu vào thực hiện chức năng nhập và mã hoá thông tin đầu vào, các thiết bị đầu ra thực hiện chức năng thể hiện thông tin ở dạng thích hợp đối với người sử dụng. Các thiết bị như vậy được gọi chung là các thiết bị ngoại vi.

CPU thực hiện trao đổi thông tin với các thiết bị ngoại vi và thế giới bên ngoài thông qua các *module vào-ra*, các module này còn được gọi là các *module giao diện* (Hình 56). Sự cần thiết của các module giao diện trong hệ thống vào-ra dữ liệu đã được trình bày ở Chương 1.

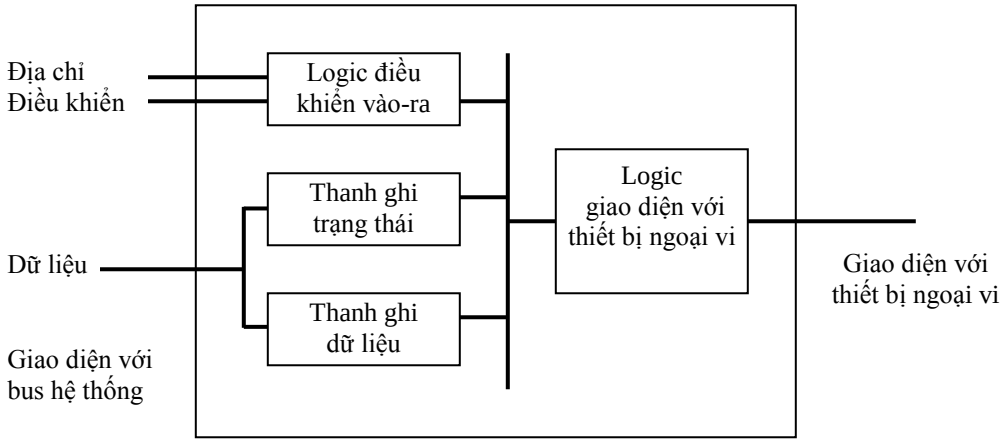


Hình 56

Module giao diện là loại thiết bị khả trình. Cấu trúc của module giao diện khả trình có thể bị sửa đổi nhờ các từ điều khiển để tương hợp được với các thiết bị ngoại vi khác nhau. Các module giao diện có thể có cấu tạo vật lý khác nhau, tuỳ thuộc vào chức năng và kiểu kết nối với thiết bị ngoại vi, nhưng đều có cấu trúc cơ bản như sau (Hình 57).

Mỗi một module giao diện có ba loại thanh ghi, là thanh ghi điều khiển (control), thanh ghi trạng thái (status) và thanh ghi dữ liệu (data). Mỗi một thanh ghi đều được gán một địa chỉ xác định, gọi là *địa chỉ cổng*.

Các thanh ghi điều khiển (thanh ghi CONTROL) nhận và chứa các từ điều khiển xác lập chế độ làm việc của module. Các thanh ghi trạng thái (thanh ghi STATUS) chứa thông tin phản ảnh trạng thái làm việc của module giao diện và thiết bị ngoại vi. Các thanh ghi dữ liệu (thanh ghi DATA) thực hiện chức năng bộ đệm tạm chứa dữ liệu được nhập/xuất.



Hình 57

CPU thực hiện thao tác vào/ra (nhập/xuất) dữ liệu với thiết bị ngoại vi khi nó thực hiện các lệnh vào (INPUT) hoặc ra (OUTPUT) với module giao diện của thiết bị này.

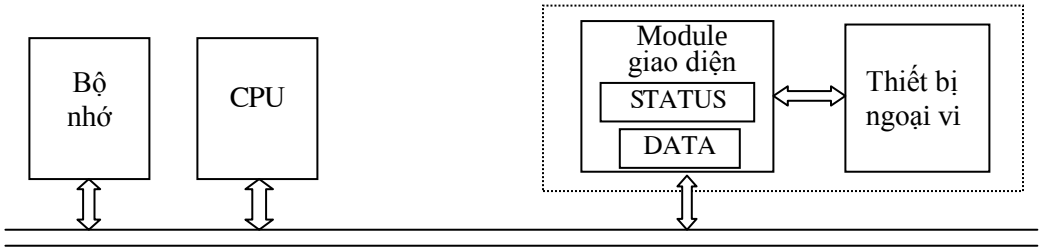
Để xuất một dữ liệu ra ngoài, CPU phải thực hiện lệnh OUT tới một cổng vào-ra có địa chỉ xác định. Thực chất CPU phải đưa dữ liệu ra thanh ghi dữ liệu của một module giao diện, module giao diện này sẽ chuyển dữ liệu thành dạng thích hợp với thiết bị ngoại vi và chuyển ra ngoài cho thiết bị ngoại vi. Khi thiết bị ngoại vi gửi một dữ liệu cho máy tính, dữ liệu này được đưa vào thanh ghi dữ liệu của module giao diện. CPU nhập dữ liệu từ ngoài vào bằng cách đọc thanh ghi dữ liệu này.

5.2. CÁC PHƯƠNG PHÁP VÀO-RA DỮ LIỆU

Module giao diện chỉ giúp CPU kết nối một cách thích hợp về mặt vật lý với các thiết bị bên ngoài, nhưng chưa đảm bảo tính tin cậy của quá trình trao đổi thông tin. Điều này xuất phát từ một thực tế khách quan là nhịp làm việc và tốc độ làm việc của CPU khác xa với nhịp và tốc độ làm việc của các thiết bị ngoại vi. Để CPU có thể thực hiện trao đổi thông tin với các thiết bị ngoại vi với độ tin cậy cao cần phải áp dụng các phương pháp thích hợp trong quá trình vào/ra dữ liệu, các phương pháp này được gọi là *các phương pháp vào-ra dữ liệu*. Việc vào/ra dữ liệu theo các phương pháp vào-ra dữ liệu được thực hiện qua chương trình điều khiển vào-ra dữ liệu. Thực chất phương pháp vào-ra là cách để CPU đồng bộ hoá được với nhịp làm việc của thiết bị ngoại vi.

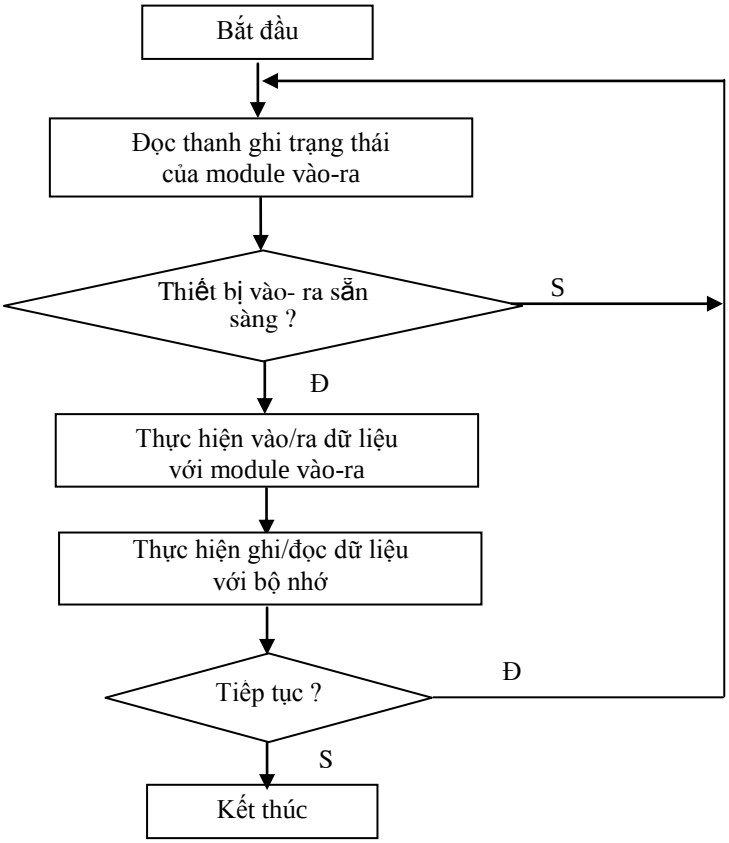
5.2.1. Phương pháp vào-ra theo thăm dò

Trong mỗi module giao diện thường có ít nhất một thanh ghi trạng thái chứa thông tin phản ảnh trạng thái làm việc của thiết bị ngoại vi và của chính module này (Hình 58). Phương pháp vào-ra theo thăm dò là phương pháp trong đó CPU luôn thực hiện kiểm tra trạng thái sẵn sàng làm việc của thiết bị trước khi thực hiện thật sự việc vào-ra dữ liệu.



Hình 58

Quá trình vào/ra dữ liệu thực hiện theo phương pháp thăm dò diễn ra như sau (Hình 59).



Hình 59

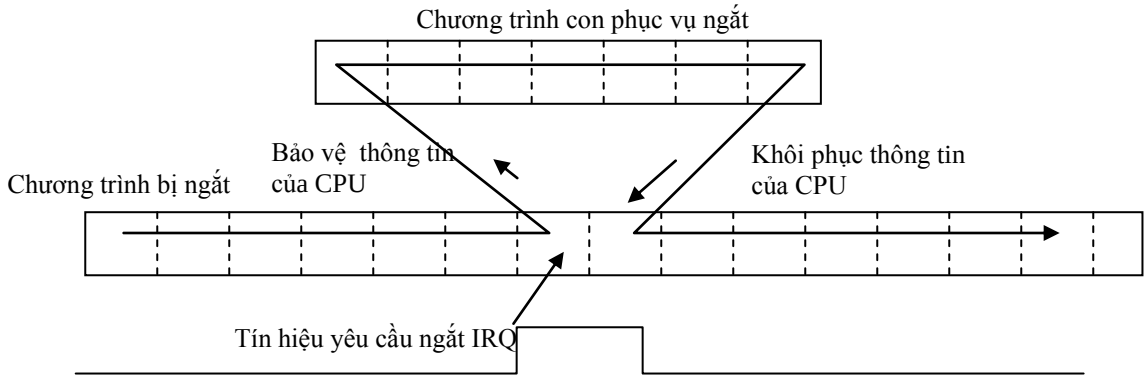
Trước khi thực hiện thao tác nhập/xuất dữ liệu qua thanh ghi dữ liệu của module giao diện, thao tác kiểm tra trạng thái làm việc của thiết bị ngoại vi và của chính module giao diện luôn được thực hiện. Các thao tác trên được thực hiện thông qua việc CPU thực hiện các lệnh vào (INPUT) và ra (OUTPUT) trong chương trình điều khiển vào-ra. Việc kiểm tra (thăm dò) này là cần thiết. Do tốc độ thực hiện lệnh máy nói chung và lệnh vào/ra nói riêng của CPU rất cao, trong khi tốc độ chuyển dữ liệu của thiết bị ngoại vi là rất thấp, thao tác đọc và kiểm tra (liên tục) trạng thái sẵn sàng làm việc (trạng thái có dữ liệu đến hoặc trạng thái sẵn sàng nhận dữ liệu từ CPU) của thiết bị vào-ra sẽ giúp CPU phát hiện được thời điểm thiết bị sẵn sàng chuyển dữ liệu, để thực hiện vào/ra dữ liệu với nó.

Ưu điểm của phương pháp thăm dò là, do CPU luôn kiểm tra trạng thái sẵn sàng làm việc của thiết bị trước khi thực hiện vào/ra dữ liệu nên quá trình vào/ra dữ liệu kiểu này có độ tin cậy cao.

Nhược điểm của phương pháp này là, do CPU luôn tốn thời gian kiểm tra trạng thái sẵn sàng của thiết bị vào-ra cho mỗi lần vào/ra dữ liệu, nên hiệu suất làm việc, xét về mặt xử lý dữ liệu, sẽ bị giảm đáng kể.

5.2.2. Phương pháp vào-ra theo ngắt

Ngắt (interrupt) là sự kiện CPU bị tạm dừng tiến trình đang thực hiện để chuyển sang thực hiện quá trình phục vụ ngắt (thực hiện chương trình vào/ra dữ liệu hoặc các dịch vụ khác theo yêu cầu) (Hình 60). CPU phải được thiết kế để nhận tín hiệu báo ngắt INT (Interrupt) hoặc IRQ (Interrupt Request) từ thiết bị bên ngoài và phản ứng với tín hiệu này một cách thích hợp.



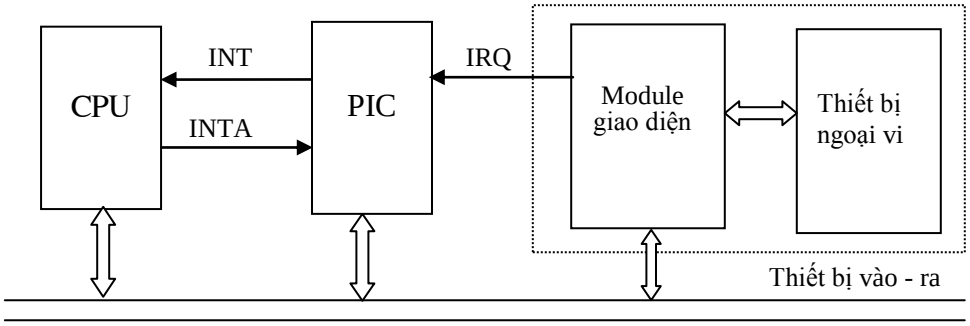
Hình 60

Ngắt còn được hiểu là phương pháp vào-ra dữ liệu, trong đó thiết bị vào-ra chủ động khởi động quá trình (chương trình) vào/ra dữ liệu nhờ hệ thống ngắt cứng. Để thực hiện được vào/ra dữ liệu theo cơ chế ngắt, các thiết bị vào-

ra có một đường dây mang thông tin về yêu cầu được phục vụ IRQ (Interrupt Request) kết nối với CPU. Đường truyền yêu cầu riêng này giúp CPU không phải thực hiện thủ tục kiểm tra trạng thái của thiết bị vào-ra mà vẫn phục vụ các thiết bị vào-ra kịp thời và hiệu quả.

Thông thường quá trình vào/ra dữ liệu theo cơ chế ngắt cứng được trợ giúp bởi thiết bị điều khiển ngắt PIC (Programmable Interrupt Controller). PIC có chức năng thay CPU trực tiếp ghi nhận các yêu cầu ngắt IRQ từ các thiết bị vào-ra, sau đó phát tín hiệu báo ngắt INT (Interrupt) cho CPU. PIC có một chức năng quan trọng khác là cung cấp cho CPU địa chỉ hoặc số ngắt đại diện cho địa chỉ của chương trình con phục vụ ngắt ứng với thiết bị phát ra yêu cầu ngắt IRQ. Các địa chỉ hoặc số ngắt này còn được gọi là *véc tơ ngắt*.

Cấu trúc của hệ thống vào-ra dữ liệu theo ngắt cứng như sau (Hình 61).



Hình 61

Điều kiện để hệ thống phục vụ ngắt hoạt động là bộ điều khiển ngắt PIC được xác lập chế độ làm việc, trong đó có việc cài đặt địa chỉ hoặc số ngắt đại diện cho địa chỉ của chương trình phục vụ ngắt ứng với tín hiệu yêu cầu ngắt IRQ phát ra từ thiết bị.

Quá trình phục vụ ngắt và vào/ra dữ liệu theo phương pháp ngắt cứng diễn ra như sau:

- CPU đang thực hiện chương trình hiện hành.
- Thiết bị vào-ra có yêu cầu phục vụ phát ra tín hiệu IRQ cho PIC.
- Thiết bị PIC phát ra tín hiệu INT cho CPU, đòi CPU phục vụ.
- CPU hoàn thành nốt lệnh đang thực hiện.
- CPU phát tín hiệu INTA (Interrupt Acknowledge) trả lời PIC, báo sẵn sàng phục vụ quá trình ngắt. Bit cờ IF trong thanh ghi trạng thái PSW được đặt xuống 0 khi ngắt được phục vụ.
- CPU cất giữ nội dung con trỏ lệnh PC và nội dung thanh ghi trạng thái PSW vào ngăn xếp (để còn khôi phục lại khi rời khỏi chương trình phục vụ ngắt).
- PIC phát ra địa chỉ (hoặc số ngắt đại diện cho địa chỉ) của chương trình con phục vụ ngắt tương ứng với tín hiệu IRQ, cho CPU.

- Dựa trên địa chỉ hoặc số ngắt này CPU nạp giá trị mới cho con trỏ lệnh PC và bắt đầu thực hiện chương trình con phục vụ ngắt, thực hiện vào/ra dữ liệu với thiết bị vào-ra.
- Chương trình con phục vụ ngắt được kết thúc bằng lệnh IRET. Khi CPU nhập và thực hiện lệnh IRET, CPU khôi phục lại nội dung con trỏ lệnh PC và thanh ghi trạng thái PSW. Bằng cách đó CPU quay lại tiếp tục thực hiện chương trình bị ngắt.

Ưu điểm của phương pháp vào/ra dữ liệu theo phương pháp ngắt cứng là:

- CPU thực hiện việc vào/ra dữ liệu ngay khi có yêu cầu từ thiết bị vào-ra. Điều này làm cho quá trình vào/ra dữ liệu có độ tin cậy rất cao.
- CPU chỉ phục vụ thiết bị vào-ra khi có yêu cầu, khi thiết bị vào/ra đã sẵn sàng cho việc truyền dữ liệu. Do vậy hiệu năng làm việc của CPU tăng lên rất nhiều.

Do những ưu điểm này mà phương pháp vào-ra dữ liệu theo ngắt cứng được dùng để thực hiện vào/ra dữ liệu với hầu hết các thiết bị ngoại vi chuẩn của máy tính. Tuy nhiên với phương pháp này quá trình chuyển dữ liệu giữa thiết bị vào-ra và bộ nhớ vẫn phải qua CPU và quá trình vào/ra dữ liệu vẫn do CPU thực hiện, nên đây chưa phải là phương pháp vào-ra nhanh nhất.

5.2.3. Phương pháp vào-ra kiểu truy nhập trực tiếp bộ nhớ

(Phương pháp DMA- Direct Memory Access)

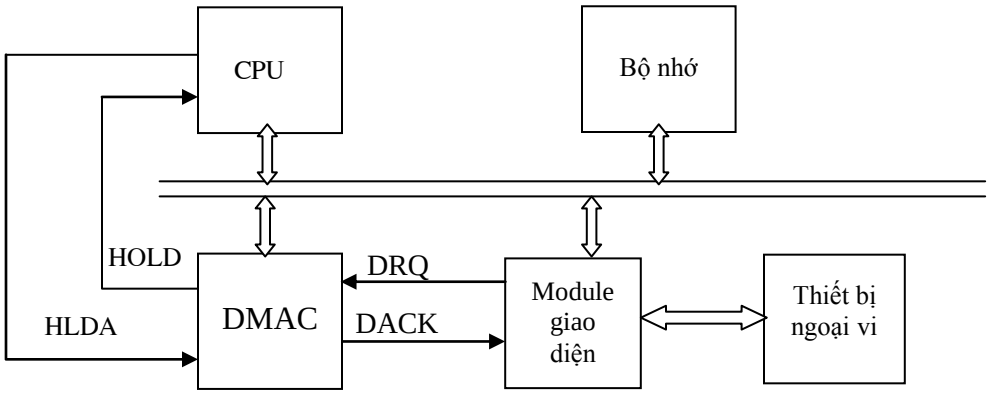
Có nhiều thiết bị ngoại vi đòi hỏi tốc độ vào/ra dữ liệu nhanh hơn khả năng của các phương pháp vào/ra dữ liệu bằng chương trình đã trình bày ở trên, ví dụ quá trình chuyển dữ liệu từ các ổ đĩa vào bộ nhớ và ngược lại. Có một phương pháp vào/ra dữ liệu đáp ứng được yêu cầu cao về tốc độ vào/ra, đó là phương pháp truy nhập trực tiếp bộ nhớ (phương pháp DMA). Quá trình vào/ra dữ liệu trực tiếp giữa bộ nhớ và thiết bị ngoại vi không qua CPU được gọi là quá trình DMA. Trong quá trình DMA việc chuyển dữ kiệu không được điều khiển bởi CPU mà bởi một thiết bị phần cứng là bộ điều khiển DMAC (Direct Memory Access Controller). Thiết bị DMAC thay thế CPU đóng vai trò chủ bus, điều khiển quá trình chuyển dữ liệu trực tiếp giữa thiết bị vào-ra và bộ nhớ chính qua bus hệ thống.

Các thiết bị vào-ra được phục vụ theo cơ chế DMA có một đường dây riêng truyền thông tin về yêu cầu được phục vụ DRQ (DMA Request). Bằng tín hiệu này thiết bị yêu cầu CPU nhường quyền điều khiển bus hệ thống cho DMAC, để DMAC điều khiển quá trình chuyển dữ liệu qua bus.

Bộ điều khiển DMAC có các thanh ghi chức năng sau: thanh ghi đệm dữ liệu IODR chứa dữ liệu cần truyền, thanh ghi địa chỉ AR với chức năng chứa (xác định) địa chỉ nền một vùng nhớ trong bộ nhớ chính dùng cho việc truyền

dữ liệu, bộ đếm DC với chức năng chứa con số xác định khối lượng từ dữ liệu cần truyền.

Cấu trúc hệ thống vào-ra dữ liệu theo kiểu truy nhập trực tiếp bộ nhớ (DMA) như sau (Hình 62).



Hình 62

Quá trình DMA được thực hiện như sau:

- CPU thực hiện các lệnh vào-ra nạp các giá trị ban đầu cho hai thanh ghi AR và DC, đồng thời xác định hướng truyền dữ liệu giữa bộ nhớ và thiết bị vào-ra cho DMAC. DMAC sẵn sàng điều khiển truyền dữ liệu.
- Thiết bị vào-ra phát tín hiệu DRQ cho DMAC.
- DMAC phát tín hiệu HOLD = 1 cho CPU, đòi CPU đáp ứng.
- CPU thực hiện nốt chu kỳ máy.
- CPU phát tín hiệu HLDA trả lời cho DMAC và tự tách ra khỏi hệ thống bus. Quyền điều khiển hệ thống bus thuộc về DMAC.
- DMAC phát tín hiệu DACK trả lời thiết bị yêu cầu, phát địa chỉ ô nhớ lên bus địa chỉ, phát các tín hiệu điều khiển ghi/đọc thiết bị vào-ra và phát tín hiệu điều khiển đọc/ghi bộ nhớ, qua đó điều khiển quá trình chuyển dữ liệu trực tiếp giữa thiết bị vào-ra và bộ nhớ. Sau mỗi từ dữ liệu được truyền, giá trị của AR tăng lên và giá trị của DC giảm đi.
- Nếu DC còn chưa giảm về 0 và thiết bị vào-ra chưa sẵn sàng gửi hoặc nhận dữ liệu tiếp theo, DMAC trả lại quyền điều khiển bus cho CPU bằng cách đặt HOLD = 0. CPU nắm lại quyền điều khiển bus.
- Khi DC giảm về 0, DMAC kết thúc quá trình DMA bằng việc phát tín hiệu HOLD = 0 cho CPU, trả quyền điều khiển bus cho CPU. CPU nắm lại quyền điều khiển bus và tiếp tục làm việc bình thường.

5.3. BỘ XỬ LÝ VÀO-RA

Các phương pháp vào-ra dữ liệu đã tiến hoá theo sự tiến hoá của hệ thống máy tính. *Bộ xử lý vào-ra IOP* (Input-Output Processor) là một dạng mở rộng của các phương pháp điều khiển vào-ra dữ liệu.

Thời kỳ đầu CPU trực tiếp điều khiển các thiết bị ngoại vi. Sau đó các module giao diện được đưa vào hệ thống máy tính và CPU điều khiển quá trình vào-ra qua các module này bằng chương trình. CPU được thiết kế để phản ứng được với cơ chế ngắt và hệ thống máy tính được trang bị bộ điều khiển ngắt. Quá trình vào-ra dữ liệu theo cơ chế ngắt, dù thực chất vẫn được CPU thực hiện bằng chương trình, nhưng giúp CPU không phải tốn thời gian kiểm tra và chờ đợi thiết bị vào-ra khi thực hiện vào/ra dữ liệu, nhờ đó hiệu năng của CPU tăng lên. Quá trình vào-ra dữ liệu kiểu DMA được thực hiện dưới sự điều khiển của bộ điều khiển DMAC, không cần đến CPU. Trong cơ chế vào-ra theo DMA, dữ liệu được chuyển trực tiếp giữa thiết bị vào-ra và bộ nhớ chính và CPU chỉ phải thực hiện một vài thao tác khởi đầu và kết thúc quá trình DMA.

Một phát triển logic tiếp theo của các phương pháp điều khiển vào-ra là module giao diện trở thành *bộ xử lý vào-ra với một tập các chỉ thị chuyên biệt cho điều khiển vào-ra dữ liệu*. CPU ra lệnh cho bộ xử lý vào-ra thực hiện chương trình điều khiển vào-ra. Bộ xử lý vào-ra nhập và thực hiện lệnh của chương trình điều khiển vào-ra, không cần đến sự can thiệp của CPU.

Điều khiển vào-ra dữ liệu bằng bộ xử lý vào-ra chuyên dụng là phương pháp thường được dùng trong thiết kế hệ thống vào-ra dữ liệu ở các hệ thống máy tính lớn có nhiều thiết bị vào-ra và nhiều người sử dụng, như các máy tính mainframe.

Phần II
MÁY VI TÍNH PC

Chương 6
KIẾN TRÚC MÁY VI TÍNH PC VÀ ĐƠN VỊ XỬ LÝ TRUNG TÂM

6.1. KIẾN TRÚC MÁY TÍNH PC

Dựa trên sự phát triển mạnh mẽ của công nghệ bán dẫn và công nghệ vi điện tử, ngành công nghiệp máy tính đã tạo ra một lớp máy tính đa năng mới gọi là máy tính cá nhân (Personal Computer-PC), thoạt đầu hướng tới từng người sử dụng đơn lẻ. Máy tính cá nhân được xây dựng trên kiến trúc của J. Von Neuman và trên cơ sở các bộ vi xử lý, các chip RAM và các thiết bị vào-ra tương ứng, nên còn được gọi là *máy vi tính* (*microcomputer*). Máy vi tính IBM PC/AT là dòng máy thành công nhất, tổ chức và kiến trúc của nó trên thực tế đã trở thành chuẩn cho các máy vi tính khác. Sau IBM đã có nhiều nhà sản xuất khác chế tạo ra hàng loạt các máy vi tính theo kiến trúc của IBM PC/AT, tạo nên một dòng máy tính được sử dụng phổ biến nhất hiện nay trên thế giới, gọi là *máy vi tính PC*.

Hình 63 mô tả kiến trúc điển hình của máy vi tính PC. Trong kiến trúc của máy vi tính PC, bộ vi xử lý kết nối với bộ nhớ chính RAM và hệ thống vào-ra qua bus hệ thống. Bus vào-ra kết nối trực tiếp với các thiết bị điều khiển, các module giao diện vào-ra và một số thiết bị phụ trợ khác.

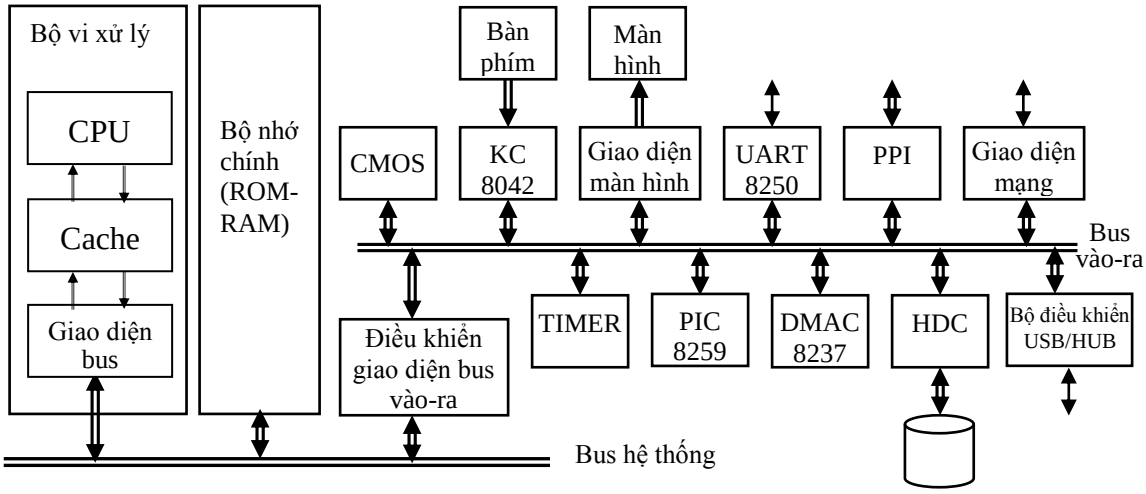
- Đơn vị xử lý trung tâm CPU Intel 80x86

Máy vi tính PC được xây dựng trên cơ sở các bộ vi xử lý họ x86 của Intel, bắt đầu từ CPU 16 bit I80286 trên máy PC/AT đầu tiên cho đến các CPU 32 bit như 80386, 80486, Pentium và Core cho các máy vi tính PC sau này.

CPU x86 có khả năng làm việc ở hai chế độ:

- Chế độ thực. Chế độ thực là chế độ hoạt động giống như CPU 8086. Trong chế độ thực CPU hoạt động ở chế độ đơn nhiệm.

- Chế độ bảo vệ. Chế độ bảo vệ được xác lập bằng phần mềm. Chế độ bảo vệ cung cấp cơ chế bảo vệ bộ nhớ và công cụ hỗ trợ quản lý bộ nhớ ảo. Trong chế độ này CPU quản lý được bộ nhớ vật lý dung lượng tới đến 4 Gbyte. Ở chế độ bảo vệ CPU có thể hoạt động theo cơ chế đa nhiệm, chạy nhiều chương trình khác nhau, trong đó các mã lệnh và dữ liệu của nhiệm vụ này được bảo vệ khỏi sự truy nhập trái phép từ nhiệm vụ hoặc ứng dụng khác.



Hình 63

- Bộ đồng xử lý toán học

Bộ đồng xử lý toán học x87 trợ giúp CPU xử lý các phép tính số học có giá trị là dấu phẩy động, thực hiện các phép tính logarit, lượng giác với tốc độ nhanh và với độ chính xác cao. Từ đời CPU 80486 DX đến sau này, bộ đồng xử lý toán học được tích hợp vào trong chip của CPU.

- Bộ nhớ chính:

Bộ nhớ chính có hai phần ROM và RAM.

- Bộ nhớ ROM: Bộ nhớ ROM là bộ nhớ chỉ đọc. Dữ liệu trong ROM được duy trì ngay cả khi không có nguồn điện nuôi. Bộ nhớ ROM chiếm một vùng địa chỉ vật lý trong bộ nhớ chính. Bộ nhớ ROM trong máy tính PC chứa hai nhóm phần mềm cơ bản. Nhóm phần mềm thứ nhất trong ROM gồm:

  - Chương trình POST (Power On Self Test). Chương trình POST kiểm tra phần cứng hệ thống và xác lập cấu hình hệ thống, khởi động tất cả các thành phần cần xác lập chế độ làm việc như: các thanh ghi CPU, Chipset, các thiết bị điều khiển PIC, DMAC, các thiết bị PCI, các module giao diện, các thiết bị ngoại vi như bàn phím, chuột, ổ đĩa v.v.

- Chương trình quét ROM mở rộng (tại các địa chỉ C000: 0000, C800: 0000). Nếu tìm thấy bất kỳ ROM nào có chương trình thì chương trình ở đó sẽ được thực thi.
- Chương trình kiểm tra giá trị tại ô nhớ địa chỉ 0000: 0472 để xác định là đang khởi động nguội hay khởi động nóng. Nếu là khởi động nóng thì bỏ qua quá trình kiểm tra bộ nhớ RAM.
- Chương trình đọc cung khởi động (mặt 0, rãnh 0, cung vật lý 1) của ổ đĩa khởi động mặc định. BIOS thường cho phép chọn ổ đĩa khởi động mặc định. Nếu khởi động từ ổ đĩa cứng thì cung khởi động chủ (Master Boot Record-MBR) của ổ đĩa cứng thứ nhất được đọc. Nội dung cung này được nạp vào bộ nhớ RAM tại địa chỉ 0000: 7C00. CPU chạy chương trình khởi động hệ điều hành tại địa chỉ này.

Nhóm phần mềm thứ hai trong ROM là các chương trình điều khiển thiết bị vào-ra cơ bản *BIOS (Basic Input/Output System)* của máy tính. BIOS là cầu nối giữa phần cứng và phần mềm trong hệ thống máy tính. Các chương trình điều khiển này luôn gắn với cấu hình phần cứng của máy tính. Thông qua các chương trình này hệ điều hành và người sử dụng có thể thực hiện giao diện ở mức thấp và truy cập được tới phần cứng của máy tính như module giao diện bàn phím, máy in, cổng vào-ra nối tiếp, ổ đĩa v.v. Các chương trình BIOS thực hiện vào-ra dữ liệu bằng xác định địa chỉ trực tiếp (ở mức vật lý) các thanh ghi và thực hiện nhập-xuất dữ liệu qua các thanh ghi của module giao diện.

Khi hệ thống được cấp điện hoặc được khởi động nóng lại, đơn vị xử lý trung tâm nhập lệnh đầu tiên từ địa chỉ FFFFFFF0H và thực hiện. Lệnh đầu tiên này thường là một *lệnh nhảy xa* tới chương trình POST. Tiếp đó chương trình POST và các chương trình khác tuần tự được thực hiện, dẫn đến việc nạp và hoạt động của hệ điều hành máy tính.

- RAM: RAM là bộ nhớ thao tác (bộ nhớ chính). Bộ nhớ RAM có chức năng chứa các phần mềm của hệ thống (hệ điều hành), phần mềm và dữ liệu của một hoặc nhiều chương trình đang hoạt động (chương trình ứng dụng).

Phân bổ địa chỉ của các vùng chức năng tại vùng nhớ 1Mbyte đầu tiên trong bộ nhớ chính như sau (Bảng 8).

Bảng 8

Địa chỉ vùng nhớ (Hexadecimal)	**Tên và chức năng vùng nhớ**
000000 - 09FFFF	Vùng nhớ quy ước 640 Kb dành cho hệ diều hành và chương trình ứng dụng.
0A0000 - 0BFFFF	Vùng nhớ dành cho hiển thị (VIDEO RAM).

0C0000 - 0DFFFF	Vùng nhớ dự phòng cho ROM-BIOS mở rộng (C000, C800, D000, D800).
0E0000 - 0EFFFF	Vùng nhớ dự phòng cho ROM mở rộng hoặc RAM.
0F0000 - 0FFFFF	Vùng nhớ 64Kb ROM BIOS.

- CMOS:

Thiết bị CMOS gồm hai bộ phận chính: đồng hồ thời gian thực và RAM-CMOS. Các thiết bị này được nuôi bằng nguồn điện độc lập.

Đồng hồ thời gian thực cung cấp thông tin thời gian thực: năm-tháng-ngày, giờ-phút-giây. Các thông tin này được chứa ở 14 byte đầu tiên trong RAM-CMOS. Đồng hồ thời gian thực được dùng để hiệu chỉnh đồng hồ hệ thống khi khởi động máy vi tính.

RAM-CMOS: phần còn lại của RAM-CMOS chứa thông tin về cấu hình hệ thống (kích thước bộ nhớ RAM, các thông tin về các ổ đĩa từ v.v.). Việc truy nhập RAM-CMOS được thực hiện thông qua các cổng với địa chỉ 070H và 071H, trong đó cổng địa chỉ 70H được dùng để xác định địa chỉ ô nhớ trong RAM CMOS, cổng địa chỉ 71H được dùng để ghi/đọc nội dung ô nhớ được chọn.

- Bộ định thời TIMER

Bộ định thời TIMER thực hiện chức năng tạo ra các dãy xung nhịp với các tần số khác nhau, như xung nhịp hệ thống, xung nhịp cho các thiết bị trong máy tính v.v.

Với các máy PC/AT đời đầu, bộ định thời TIMER 8254 có 3 bộ đếm nhị phân 16 bit. Đây là các bộ đếm khả trình. Các bộ đếm 16 bit trong TIMER 8254 nhận xung nhịp tần số 1190MHz từ mạch tạo xung chuẩn. TIMER 8254 tạo ra xung nhịp cho đồng hồ hệ thống, xung nhịp làm tươi DRAM và xung âm thanh cho loa:

- Bộ đếm # 0: tạo xung nhịp chu kỳ 55ms (≈ 18. 2 xung /1giây) cho đồng hồ hệ thống.
- Bộ đếm # 1: tạo xung nhịp kích hoạt hệ thống làm tươi DRAM.
- Bộ đếm # 2: cho phép người sử dụng lập trình tạo xung âm thanh cho loa.

- Bộ điều khiển ngắt PIC 8259 (Programmable Interrupt Controller)

PIC 8259 là bộ điều khiển ngắt khả trình. PIC 8259 hỗ trợ CPU thực hiện cơ chế ngắt cứng, cho phép CPU phản ứng tức thời với yêu cầu của các thiết bị vào-ra dữ liệu và các thiết bị khác. Trong máy vi tính dòng PC/AT có hai thiết bị PIC 8259, mỗi thiết bị ghi nhận được 8 yêu cầu ngắt IRQ

- Bộ điều khiển truy nhập trực tiếp bộ nhớ DMAC 8237 (Direct Memory Access Controller)

DMAC 8237 là thiết bị điều khiển DMA khả trình. DMAC 8237 điều khiển quá trình vào/ra dữ liệu trực tiếp giữa bộ nhớ và các thiết bị ngoại vi (ví dụ: thiết bị đĩa từ), gọi tắt là quá trình DMA (Direct Memory Access). Trong máy vi tính dòng AT có hai thiết bị DMAC 8237, mỗi thiết bị phục vụ 4 kênh DMA.

- Module giao diện bàn phím KC 8042 (Keyboard Controller)

KC 8042 thực hiện chức năng giao diện giữa bàn phím và đơn vị xử lý trung tâm CPU. KC 8042 thực hiện giao diện kiểu nối tiếp đồng bộ giữa CPU và bàn phím, phát các chỉ thị và nhận mã phím từ bàn phím.

- Module giao diện màn hình

Thiết bị giao diện màn hình thực hiện chức năng giao diện giữa CPU và màn hình hiển thị. Nó nhận lệnh và dữ liệu từ CPU, từ đó tạo ra tín hiệu ảnh và các tín hiệu đồng bộ điều khiển hiển thị trên màn hình. Thiết bị giao diện màn hình có thể làm việc ở hai chế độ văn bản và đồ họa màu.

- Module giao diện nối tiếp UART 8250 (Universal Asynchronous Receiver Transmitter)

UART 8250 thực hiện chức năng giao diện kiểu nối tiếp không đồng bộ giữa CPU và các thiết bị ngoại vi theo chuẩn RS 232. UART 8250 thực hiện chức năng nhận và phát dữ liệu dạng nối tiếp, tạo và nhận các tín hiệu bắt tay với thế giới bên ngoài theo chuẩn RS 232. Trong máy vi tính dòng PC/AT có hai thiết bị UART 8250, với các đầu kết nối với thiết bị ngoại vi DB9.

- Module giao diện song song chuẩn PPI (Programmable Peripheral Interface)

Mạch PPI thực hiện quá trình giao diện song song giữa CPU và các thiết bị ngoại vi. PPI thực hiện chức năng nhận và phát dữ liệu dạng song song, tạo và nhận các tín hiệu bắt tay với các thiết bị ngoại vi (ví dụ như máy in v.v.).

- Module giao diện đĩa mềm FDC (Floppy Disk Controller)

FDC thực hiện chức năng giao diện giữa CPU và thiết bị đĩa mềm. FDC nhận lệnh và dữ liệu từ CPU hoặc bộ nhớ, chuyển thành dạng thích hợp đưa ra thiết bị đĩa mềm và ngược lại.

- Module giao diện đĩa cứng HDC (Hard Disk Controller)

HDC thực hiện chức năng giao diện giữa CPU và thiết bị đĩa cứng theo chuẩn ATA IDE. HDC nhận dữ liệu từ bộ nhớ chính, chuyển thành dạng thích hợp đưa ra thiết bị đĩa cứng và ngược lại theo chuẩn ATA IDE. Ở các máy vi

tính PC hiện đại, module giao diện HDC được gắn cùng với ổ đĩa cứng, tạo thành các thiết bị đĩa cứng chuẩn PATA và SATA.

- Bộ điều khiển chủ USB/HUB gốc

Bộ điều khiển chủ USB tích hợp với HUB gốc tạo kết nối và giao diện với các thiết bị ngoại vi USB theo chuẩn USB.

Trong các máy vi tính PC hiện đại, các mạch chức năng như 8254, 8259, 8237, CMOS-RAM, 8250, PPI, 8042, điều khiển đĩa mềm, giao diện IDE, cầu PCI v.v. hiện nay đã được tích hợp thành các mạch tích hợp cỡ lớn, gọi là các *chipset.*

Các vi xử lý dòng Intel sử dụng một không gian địa chỉ vào-ra riêng, tách rời khỏi không gian địa chỉ bộ nhớ. Kích thước không gian địa chỉ vào-ra (IO Address) là 64K địa chỉ. Địa chỉ các cổng vào-ra (port addresses) ở trên máy tính dòng PC/AT được phân bổ như sau (Bảng 9).

Bảng 9

Vùng địa chỉ (Hexadecimal)	**Thiết bị**
000 – 01F	Bộ điều khiển DMAC-1 8237
020 – 03F	Bộ điều khiển ngắt PIC-1 8259, chủ
040 – 043	Bộ định thời 8254
060 – 064	Module giao diện bàn phím KC 8042
070 –071	CMOS-RAM - Đồng hồ thời gian thực
080 – 08F	Thanh ghi trang DMA
0A0 – 0A1	Bộ điều khiển ngắt PIC-2 8259, thợ
0C0 – 0DF	Bộ điều khiển DMAC-2 8237
0F8 – 0FF	Bộ đồng xử lý 80x87
170 – 177	Module giao diện IDE-1
1F0 – 1F7	Module giao diện IDE-0
200 – 207	Module giao diện điều khiển trò chơi
278 – 27F	Module giao diện song song 2 (cổng máy in LPT2)
2F8 – 2FF	Module giao diện nối tiếp 2(giao diện nối tiếp COM 2)

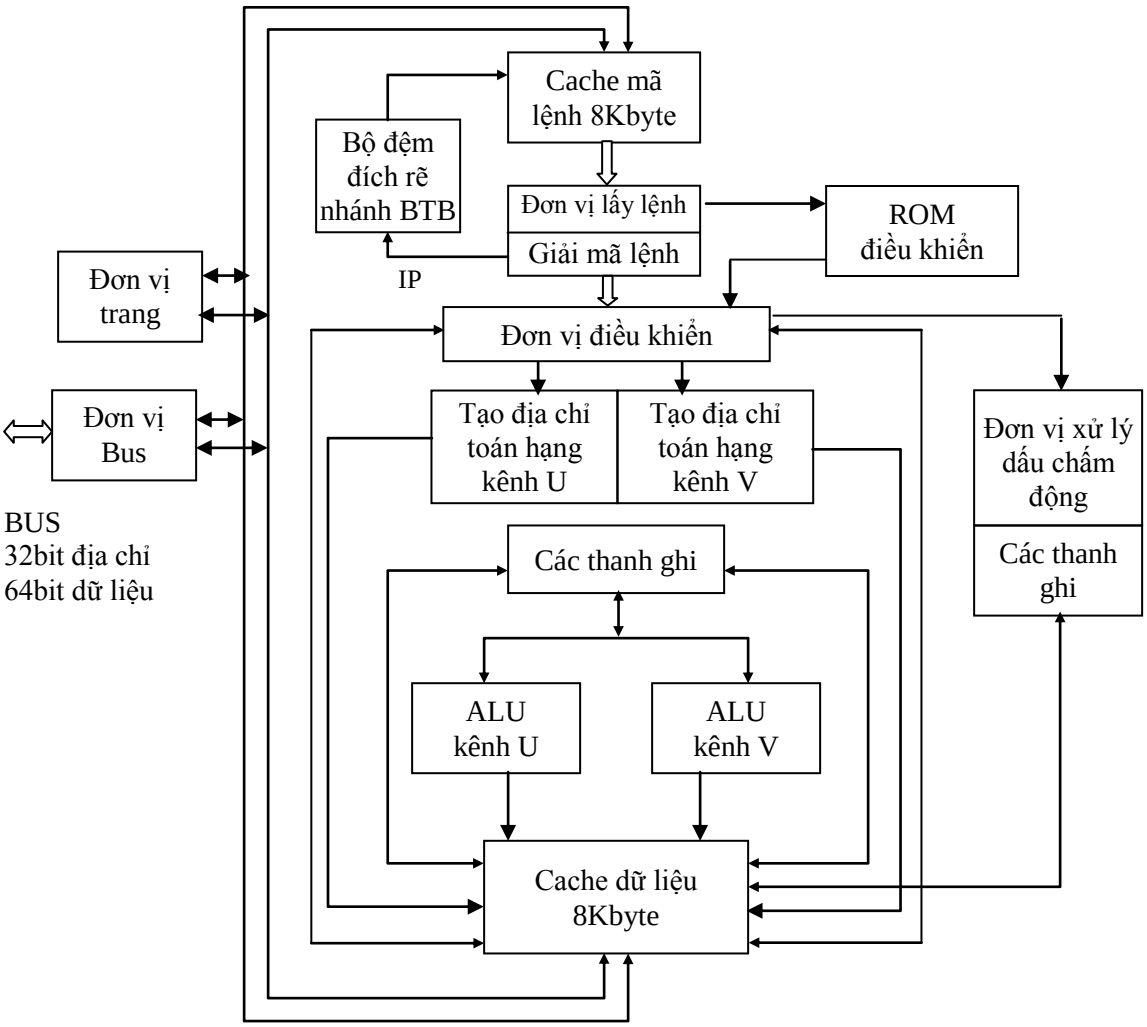
Vùng địa chỉ (Hexadecimal)	**Thiết bị**
300 – 31F	Card thử nghiệm
378 – 37F	Module giao diện song song 1 (cổng máy in LPT 1)
380 – 38F	Dự phòng
3A0 – 3AF	Dự phòng
3B0 – 3BF	Thiết bị giao diện màn hình đơn sắc
3C0 – 3CF	Dự phòng
3D0 – 3DF	Thiết bị giao diện màn hình VGA
3E8 – 3EF	Module giao diện nối tiếp 3 (cổng giao diện nối tiếp COM 3)
3F0 – 3F7	Thiết bị giao diện đĩa mềm FDC
3F8 – 3FF	Module giao diện nối tiếp 1(giao diện nối tiếp COM 1)
CF8 – CFF	Các thanh ghi cấu hình PCI

6.2. ĐƠN VỊ XỬ LÝ TRUNG TÂM

6.2.1. Đơn vị xử lý trung tâm Pentium

Pentium là đơn vị xử lý trung tâm 32 bit và là đơn vị xử lý trung tâm loại CISC (Complex Instruction Set Computer) với các đặc điểm: hệ lệnh phức tạp, nhiều kiểu xác định địa chỉ, nhiều khuôn dạng lệnh và nhiều kích thước lệnh khác nhau. Pentium sử dụng kỹ thuật xử lý song song mức lệnh, kỹ thuật ILP (Instruction Level Pararellism).

Pentium (Hình 64) có bus địa chỉ trong và ngoài 32 bit, bus dữ liệu trong và ngoài 64 bit. Pentium có hai cache 8Kbyte độc lập: một cache 8Kbyte 2 cổng dành cho dữ liệu và một cache 8Kbyte chứa lệnh. Mỗi cache có một bảng TLB riêng, cho phép CPU nhập 2 lệnh và truy nhập toán hạng đồng thời. Pentium là loại đơn vị xử lý trung tâm được thiết kế theo kỹ thuật *superscalar*, trong đó hai lệnh được nhập và giải mã đồng thời (ở giai đoạn nhập lệnh và giai đoạn giải mã lệnh 1). Pentium có hai kênh thực hiện lệnh song song U và V với hai đơn vị số học-logic ALU.



Hình 64

Tập lệnh của Pentium gồm hai nhóm lệnh: nhóm các lệnh đơn giản và nhóm các lệnh phức tạp. Đường ống thực hiện lệnh của Pentium gồm 5 giai đoạn:

1- Nhập lệnh. Lệnh được nhập từ cache và đưa vào vùng đệm trước giải mã.
2- Giải mã 1. Lệnh được giải mã để tạo ra từ điều khiển. Từ điều khiển đơn giản sẽ gây ra việc thực hiện trực tiếp lệnh. Từ điều khiển phức tạp sẽ được xử lý tiếp. Dự báo rẽ nhánh cũng được thực hiện trong giai đoạn này.
3- Giải mã 2. Từ điều khiển phức tạp được giải mã tiếp để tạo ra các vi mã thao tác (micro-ops) dùng cho giai đoạn thực thi lệnh. Địa chỉ toán hạng cũng được tạo ra ở giai đoạn này.
4- Thực thi lệnh. Lệnh được thực thi ở ALU. Nếu cần, cache dữ liệu cũng được truy nhập ở giai đoạn này.
5- Ghi kết quả. Kết quả tính toán được ghi trở lại vào các thanh ghi CPU.

Pentium có bộ dự báo rẽ nhánh và bộ đệm đích rẽ nhánh BTB. Bộ dự báo rẽ nhánh dự đoán trước sự có mặt của các lệnh nhảy, lệnh gọi chương trình con và lệnh trở về, cho phép đoán trước hướng đi của chương trình, nhờ đó đường ống lệnh luôn được làm đầy. Đơn vị nhập lệnh tính con trỏ lệnh theo giá trị từ bộ đệm đích rẽ nhánh BTB (Branch Target Buffer). Có nhiều mã thao tác được giải mã thành một vi lệnh đơn. Một số mã thao tác khác lại được chuyển thành ba hoặc bốn vi mã thao tác. Bộ giải mã có thể tạo ra nhiều vi mã trong một chu kỳ nhịp, do trong bộ giải mã có ba bộ giải mã con, trong đó hai bộ giải mã con cho mã lệnh đơn giản và một bộ giải mã con cho mã lệnh phức tạp.

Điều kiện cơ bản để thực hiện đồng thời hai lệnh trong Pentium là cả hai lệnh phải là loại đơn giản. Các lệnh đơn giản là các lệnh: chuyển dữ liệu tức thời từ thanh ghi hoặc từ bộ nhớ tới thanh ghi, chuyển giá trị tức thời hoặc từ thanh ghi tới bộ nhớ, lệnh số học nguyên, lệnh tăng giảm, lệnh bảo vệ và khôi phục thanh ghi từ ngăn xếp, lệnh nhảy không điều kiện và nhảy gần có điều kiện, lệnh gọi chương trình con, lệnh lấy địa chỉ hiệu dụng v.v. Các lệnh đơn giản này được thực hiện bằng phần cứng và cơ bản là được thực hiện trong một chu kỳ nhịp.

Quá trình xử lý song song mức lệnh có thể được thể hiện qua thuật toán sau:

Giải mã đồng thời hai lệnh I1 và I2

Nếu những điều sau đúng:

I1 và I2 là các lệnh đơn giản

I1 không phải là lệnh nhảy

Đích của I1 không phải là nguồn của I2

Đích của I1 không phải là đích của I2

thì đưa I1 vào kênh U và I2 vào kênh V

các trường hợp khác thì đưa I1 vào kênh U

Nếu lệnh đầu tiên trong hai lệnh là lệnh nhảy, nó được đưa trước vào kênh U để thực hiệnvà không lệnh nào được đưa vào kênh V.

Với các lệnh dấu chấm động, hai giai đoạn đầu tiên nhập và giải mã lệnh được thực hiện trên kênh U, sau đó được chuyển sang FPU. Không lệnh dấu chấm động nào được thực hiện song song.

Pentium có một bộ đồng xử lý dấu chấm động FPU được tích hợp trong chíp và một đường ống riêng thực hiện các lệnh dấu chấm động. FPU có các bộ cộng, nhân, chia dấu chấm động và một hệ thống các thanh ghi riêng. Cache dữ liệu trong Pentium là loại hai cổng, cho phép hai kênh U và V truy nhập đồng thời.

Pentium hỗ trợ quản lý bộ nhớ theo phân trang. Kích thước trang thông thường là 4 Kbyte, nhưng cũng có tuỳ chọn cho trang 4 Mbyte. Mỗi một cache trong Pentium có một bảng TLB riêng. Với cache dữ liệu có bảng TLB cho cả hai loại trang 4 Kbyte và 4 Mbyte. Với cache mã lệnh chỉ có bảng TLB cho trang 4 Kbyte. Intel cũng cung cấp các chip phụ điều khiển cache L2 512 Kbyte nằm ngoài chip CPU.

6.2.2. Tập các thanh ghi cơ bản

- Các thanh ghi đa năng EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP

EAX, EBX, ECX, EDX là các thanh ghi đa năng 32 bit, được dùng để chứa các toán hạng, kết quả của phép tính hoặc các toán hạng địa chỉ. Có thể truy nhập từng thanh ghi này như là các thanh ghi 16 bit AX, BX, CX, DX hoặc như các thanh ghi 8 bit AH, AL, BH, BL, CH, CL, DH, DL.

31 16	15 0	
	AH (AX) AL	EAX
	BH (BX) BL	EBX
	CH (CX) CL	ECX
	DH (DX) DL	EDX
	BP	EBP
	SI	ESI
	DI	EDI
	SP	ESP

- EAX (Accumulator) là thanh ghi đa năng. Khi gọi các hàm chức năng của hệ điều hành qua lệnh INT, thanh ghi AX hoặc AH được dùng để chứa con số xác định một chức năng cụ thể cần thực hiện của hàm này.

- EBX (Base) là thanh ghi đa năng. Khi thực hiện các lệnh truy nhập đoạn dữ liệu, EBX được dùng để xác định địa chỉ offset hoặc địa chỉ offset nền của một vùng nhớ trong đoạn.

- ECX (Counter) là thanh ghi đa năng. Khi thực hiện các lệnh LOOP hoặc lệnh có tiền tố REP, ECX đóng vai trò một bộ đếm (chứa số đếm), xác định số lần thực hiện lặp.

- EDX (Data) là thanh ghi đa năng. EDX (DX) còn được dùng để chứa 4 byte (2 byte) cao kết quả phép nhân 32 bit (16 bit) hoặc phần dư của phép chia

64 bit (32 bit). Khi truy nhập các cổng vào/ra có địa chỉ lớn hơn 255, thanh ghi DX được dùng để chứa địa chỉ (xác định địa chỉ) các cổng này.

EBP, ESI, EDI và ESP là các thanh ghi 32 bit.

- EBP (Base Pointer). EBP thường được dùng để xác định địa chỉ offset hoặc địa chỉ offset nền khi truy nhập đoạn ngăn xếp.

- ESI (Source Index). ESI thường được dùng để chứa địa chỉ offset của xâu ký tự trong đoạn dữ liệu trỏ bởi DS khi thực hiện các lệnh thao tác xâu ký tự.

- EDI (Destination Index). EDI thường được dùng để chứa địa chỉ offset của xâu ký tự trong đoạn mở rộng trỏ bởi ES khi thực hiện lệnh thao tác xâu ký tự.

- ESP (Stack Pointer). Con trỏ ngăn xếp ESP chứa địa chỉ offset của ô nhớ trong đoạn ngăn xếp. ESP luôn trỏ đến đỉnh ngăn xếp.

- Con trỏ lệnh EIP (Instruction Pointer). Con trỏ lệnh EIP là một thanh ghi 32 bit. EIP chứa địa chỉ offset của ô nhớ chứa lệnh tiếp theo trong đoạn mã lệnh. Khi nhập lệnh thì EIP tự động tăng dần và khi CPU nhập xong một lệnh thì EIP trỏ đến ô nhớ chứa lệnh tiếp theo trong đoạn mã lệnh. Nội dung EIP bị thay đổi bất thường khi CPU thực lệnh nhảy, lệnh gọi chương trình con hoặc bởi các cơ chế ngắt cứng và mềm.

- Thanh ghi cờ trạng thái EFLAGS. EFLAGS là thanh ghi 32 bit. Thanh ghi trạng thái EFLAGS chứa các bit mang thông tin phản ảnh trạng thái của bộ xử lý và một số bit điều khiển.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										ID	VIP	VIF	AC	VM	RF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NT	IOPL		OF	DF	IF	TF	SF	ZF		AF		PF		CF

Các bit 1, 3, 5, 15 và từ bit 22 đến bit 31 trong EFLAGS không được định nghĩa.

Có thể tác động vào một số bit cờ qua các lệnh thao tác cờ như CLC, STC, CLI, STI, CLD, STD v.v.

- CF (Carry Flag) - cờ nhớ. Nếu có nhớ từ bit cao nhất của kết quả khi thực hiện phép tính thì CF = 1. Các lệnh CLC, STC và các lệnh dịch bit, quay vòng ảnh hưởng tới cờ này.

- PF (Parity Flag)- cờ kiểm tra chẵn lẻ. Nếu lượng các bit 1 trong 8 bit thấp của kết qủa là chẵn thì PF = 1.
- AF (Auxiliary Flag)- cờ nhớ phụ. Cờ AF được sử dụng trong các phép tính số học với các số BCD (Binary Coded Decimal).
- ZF (Zero Flag)- cờ zero. Nếu tất cả các bit của kết quả phép tính là 0 thì ZF = 1.
- SF (Sign Flag)- cờ dấu. Nếu kết quả âm (bit cao nhất của kết quả có giá trị 1) thì SF = 1.
- TF (Trap Flag)- cờ bẫy. Nếu TF = 1, CPU được đặt ở chế độ chạy từng lệnh, phục vụ cho hiệu chỉnh chương trình (debug). Trong chế độ này mỗi lệnh khi được thực hiện xong sẽ gây ra một ngoại lệ kích hoạt chương trình hiệu chỉnh.
- IF (Interrupt Flag)- cờ ngắt. Nếu IF = 1, cho phép CPU phản ứng với tín hiệu báo ngắt INT. Nếu IF = 0 CPU sẽ không phản ứng với tín hiệu báo ngắt INT. Các lệnh STI và CLI làm thay đổi giá trị cờ IF.
- DF (Direction Flag)- cờ hướng. DF xác định hướng tăng hoặc giảm của các thanh ghi ESI và EDI khi thực hiện các lệnh xử lý xâu ký tự. Nếu DF = 1 thì ESI và EDI giảm. Nếu DF = 0 thì ESI và EDI tăng. Các lệnh STD và CLD làm thay đổi giá trị cờ DF.
- OF (Overflow Flag)- cờ tràn. OF = 1 nếu kết quả quá lớn hoặc quá nhỏ, vượt quá khả năng biểu diễn của máy tính (bị tràn).
- IOPL (I/O Privilege Level). Giá trị của IOPL (0, 1, 2 hoặc 3) chỉ thị mức đặc quyền cao nhất được phép truy nhập vào không gian địa chỉ cổng vào-ra.
- NT (Nested Task). Nếu NT=1, là thông báo nhiệm vụ hiện hành đang nằm trong một nhiệm vụ khác.
- RF (Resume Flag). Khi RF=1, tạm thời loại bỏ lỗi hiệu chỉnh.
- VM (Virtual Mode). Nếu VM=1, CPU sẽ được đặt sang chế độ 8086 ảo, giả lập môi trường lập trình của bộ vi xử lý 8086.
- AC (Alignment Check). Việc đặt AC=1 và bit AM trong thanh ghi điều khiển CR0 cho phép kiểm tra việc quy chiếu bộ nhớ.
- VIF (Virtual Interrupt Flags). Bit cờ VIF là ảnh ảo của cờ IF.
- VIP (Virtual Interrupt Pending Flags). Bit cờ VIP cùng với VIF cho phép chương trình ứng dụng ảo hoá cờ IF của hệ thống trong môi trường đa nhiệm.
- ID (Indetification Flag). Chương trình có thể đặt hoặc xoá cờ ID để chỉ thị rằng bộ xử lý hỗ trợ lệnh nhận dạng CPU.

- Các thanh ghi điều khiển 32 bit CR0, CR1, CR2, CR3, CR4

CR0
CR1
CR2
CR3
CR4

- Thanh ghi CR0. Thanh ghi CR0 chứa các bit cờ điều khiển hệ thống, các cờ này điều khiển kiểu thao tác hoặc chỉ thị trạng thái hệ thống. Chỉ có các bit 0-5, 16, 18, 29-31 đang được sử dụng. Ví dụ: cờ PG (bit 31) Paging Enable, nếu PG=1 cho phép quản lý theo phân trang, PG=0 quản lý không theo phân trang; cờ TS (bit 3) Task Switch, nếu thao tác chuyển nhiệm vụ được thực hiện thì TS=1; cờ PE (bit 0) Protection Enable, nếu PE được đặt (PE=1) thì cho phép cơ chế bảo vệ v.v.

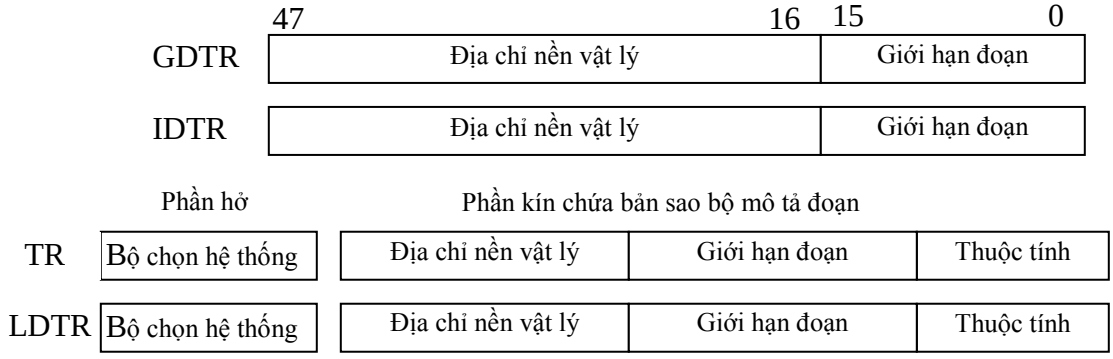
- Thanh ghi CR1. Chưa sử dụng.

- Thanh ghi CR2. CR2 chứa địa chỉ tuyến tính 32 bit của trang có lỗi (gây lỗi trang). Ví dụ, khi nhiệm vụ quy chiếu đến trang, nhưng nó chưa có trong bộ nhớ vật lý (chưa được nạp từ bộ nhớ thứ cấp vào bộ nhớ vật lý), khi đó CPU phát hiện ra trạng thái lỗi trang. Địa chỉ của trang có lỗi được ghi trong CR2.

- Thanh ghi CR3. CR3 còn được gọi là Thanh ghi nền thư mục trang PDBR. 20 bit cao (bit 31 đến 12) của CR3 chứa *20 bit cao nhất của địa chỉ nền thư mục trang, 12 bit thấp của địa chỉ nền thư mục trang mặc định bằng 0*. Trong 12 bit còn lại của CR3 chỉ có bit 3 và bit 4 được dùng để chỉ thị cho phép hoặc cấm cache trong chip và kỹ thuật ghi cache.

- Thanh ghi CR4. CR4 chứa các bit cho phép mở rộng kiến trúc của CPU. Ví dụ: bit PSE (bit 4) Page Size Extention, đặt PSE=1 cho phép định nghĩa mở rộng trang đến 4 Mbyte. Khi trang 4 Mbyte được sử dụng, chỉ còn một mức bảng trang được dùng trong chuyển hoá địa chỉ tuyến tính sang địa chỉ vật lý.

- Các thanh ghi quản lý bộ nhớ GDTR, LDTR, TR, IDTR



Các thanh ghi GDTR (Global Descriptor Table Register), LDTR (Local Descriptor Table Register), TR (TSS Register) và IDTR (Interupt Descriptor Table Register) được sử dụng để xác định vị trí các cấu trúc dữ liệu dùng trong quản lý bộ nhớ theo phân đoạn ở chế độ bảo vệ (xem mục 5.2.6).

6.2.3. Tập lệnh

Để thuận tiện cho mô tả và lập chương trình với các lệnh máy, tập lệnh máy thường được thể hiện dưới dạng ngôn ngữ assembly với khuôn dạng tổng quát như sau:

Tên lệnh toán hạng, toán hạng

Lệnh máy có thể thao tác trên 0 toán hạng, 1 toán hạng hoặc 2 toán hạng. Các toán hạng có thể là 8 bit, 16 bit hoặc 32 bit. Toán hạng có thể nằm ngay trên lệnh, có thể nằm ở các thanh ghi của CPU hoặc nằm ở bộ nhớ. Các chương trình viết dưới dạng ngôn ngữ assembly cần phải được dịch sang dạng ngôn ngữ máy (mã lệnh máy) trước khi nó được CPU thực hiện.

Lệnh máy có thể thực hiện thao tác giữa các đối tượng sau:

Thanh ghi <- Tức thời
Bộ nhớ <- Tức thời
Thanh ghi <-> Thanh ghi
Bộ nhớ <-> Thanh ghi

Tập lệnh của Pentium và các CPU Intel x86 gồm các *nhóm lệnh* sau:

- Các lệnh chuyển dữ liệu, gồm 21 lệnh, ví dụ như: MOV, MOVSX, PUSH, POP, LEA, LDS, LSS, IN, OUT v.v.
- Các lệnh số học, gồm 21 lệnh, ví dụ như: ADD, ADC, SUB, SBB, MUL, IMUL, DIV, IDIV, INC, DEC, CMP v.v.
- Các lệnh logic, dịch, quay gồm 15 lệnh, ví dụ như: AND, OR, XOR, NOT, SAL, SAR, SHL, SHR, RCL, RCR v.v.
- Các lệnh thao tác xâu, gồm 31 lệnh, ví dụ như: MOVS, MOVSB, CMPS, CMPSB, LODS, LODSB, STOS, STOSB, REP, INS, OUTS v.v
- Các lệnh thao tác bit, gồm 6 lệnh, ví dụ như: BT, BTC v.v.
- Các lệnh điều khiển rẽ nhánh, gồm 46 lệnh, ví dụ như: JMP, CALL, RET, JC, JNC, JS, JNS, JZ, JNZ, JO, JNO, JE, JG, LOOP, INT, IRET v.v.
- Các lệnh hỗ trợ cơ chế bảo vệ, gồm 16 lệnh, ví dụ như: LGDT, SGDT, LLDT, SLDT, LTR, STR, LIDT, SIDT, ARPL v.v.
- Các lệnh điều khiển bộ xử lý gồm 8 lệnh, ví dụ như: HLT, NOP, XLAT v. v

- Các lệnh dấu chấm động, gồm 74 lệnh, ví dụ như: FADD, FSUB, FMUL, FDIV, FABS, FCOM, FCOS, FSIN, FPTAN, FLDLG2, FLDL2T v.v.
- Các lệnh hỗ trợ ngôn ngữ bậc cao HLL (High-Level Language), gồm 4 lệnh, như ENTER, LEAVE, BOUND, SET.

Pentium có một số lệnh như ENTER, LEAVE hỗ trợ trực tiếp việc gọi thủ tục và chương trình con trong ngôn ngữ lập trình bậc cao có cấu trúc. Lệnh ENTER có hai tham số, tham số đầu tiên xác định số byte cần cho lưu trữ động (chứa các con trỏ đến khung ngăn xếp trước, chứa các biến tự động) trong ngăn xếp. Tham số thứ hai xác định mức độ sâu (mức làm tổ - nesting level) của chương trình con. Nó xác định có bao nhiêu tập con trỏ khung ngăn xếp được sao lưu vào khung ngăn xếp mới từ khung ngăn xếp trước đó, danh sách này được gọi là *hiển thị* (*display*). Lệnh ENTER tạo một hiển thị cho một chương trình con, dự trữ vùng nhớ cho các biến cục bộ và giảm ESP đi một lượng đúng bằng số lượng byte dăng ký ở tham số thứ nhất. Giá trị mới của ESP được dùng cho các thao tác ngăn xếp của chương trình con mới. Thanh ghi EBP được dùng để trỏ đến điểm đầu của khung ngăn xếp mới. Lệnh LEAVE được dùng ở cuối chương trình con, để phục hồi trạng thái trước lệnh ENTER. Lệnh ENTER và LEAVE làm đơn giản hoá việc vào và ra khỏi chương trình con.

6.2.4. Các kiểu xác định địa chỉ toán hạng

Các *kiểu xác định địa chỉ toán hạng* **(***các chế độ định vị toán hạng***)** cho phép xác định nơi chứa toán hạng. Nơi chứa toán hạng có thể là ngay trên lệnh, là thanh ghi của CPU hoặc bộ nhớ. Mặc định bộ nhớ được hiểu là đoạn dữ liệu hiện thời, trỏ bởi DS. Trong các trường hợp khác cần chỉ rõ đoạn nhớ cần truy nhập, ví dụ ES: [địa chỉ offset].

Xác định địa chỉ toán hạng là *xác định địa chỉ offset hiệu dụng của ô nhớ chứa toán hạng* trong đoạn đó.

Hình thức tổng quát của một lệnh ở dạng ngôn ngữ assembly như sau:

Tên lệnh toán hạng, toán hạng

trong đó thành phần ngay sau tên lệnh xác định nơi chứa kết quả của phép toán hoặc toán hạng nguồn, thành phần tiếp theo, được phân cách bằng một dấu phảy, là nơi xác định toán hạng nguồn.

Có các kiểu xác định địa chỉ toán hạng sau:

- Kiểu xác định địa chỉ Tức thời

  Dữ liệu nằm ngay trong câu lệnh.

  Ví dụ: MOV EAX, FF00H ; đưa giá trị FF00H vào EAX

- Kiểu xác định địa chỉ Thanh ghi

Các thanh ghi đa năng hoặc các thanh ghi đoạn của CPU là nơi chứa dữ liệu.

Ví dụ: MOV EAX, EBX ; chuyển (sao chép) giá trị nằm ở EBX vào EAX

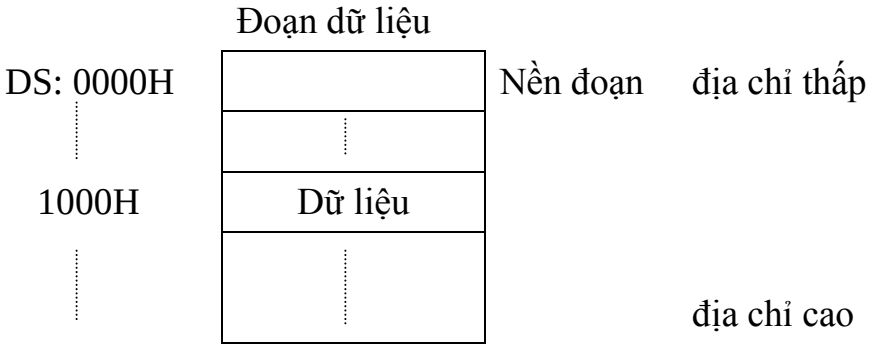
- Kiểu xác định địa chỉ Trực tiếp

Địa chỉ offset của ô nhớ chứa toán hạng nằm trực tiếp ngay trong lệnh.

Địa chỉ offset ≡ [giá trị cụ thể]

Ví dụ: MOV EAX, [1000H] ; Đưa nội dung ô nhớ có địa chỉ DS: 1000H vào EAX

MOV EAX, ALFA ; ALFA là tên biến đã được khai báo



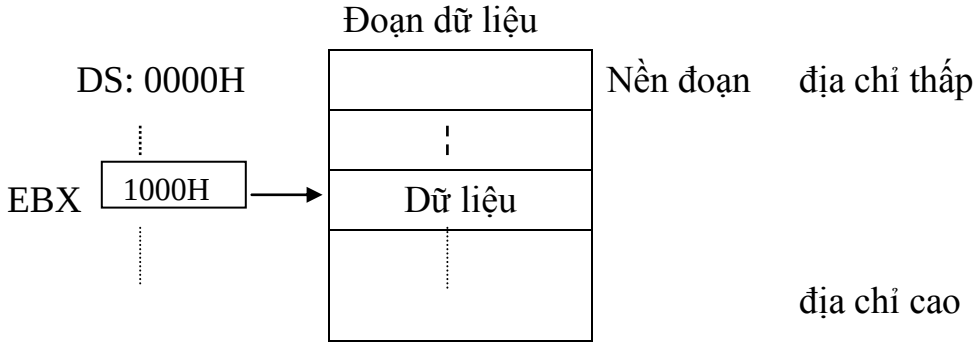
- Kiểu xác định địa chỉ Gián tiếp thanh ghi

Với hệ 32 bit, địa chỉ ô nhớ được xác định bởi bất kỳ thanh ghi đa năng nào: EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP

Với hệ 16 bit, địa chỉ ô nhớ được xác định bởi các thanh ghi BX, BP, SI, DI, SP.

Ví dụ: MOV EAX, [EBX] ; trong đó [EBX] ≡ 1000H

lệnh MOV thực hiện sao chép nội dung ô nhớ được trỏ bởi EBX vào EAX, địa chỉ ô nhớ là DS: 1000H.



- Kiểu xác định địa chỉ Cơ sở + khoảng dịch

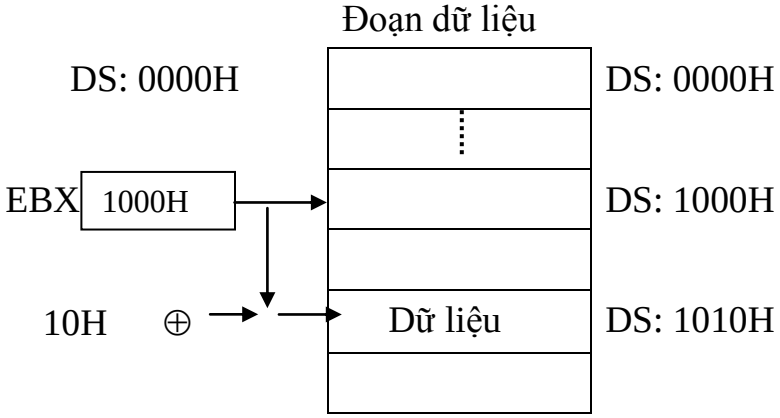
  Địa chỉ offset ≡ [EyX + khoảng dịch]

  trong đó: với hệ 32 bit, EyX là bất kỳ thanh ghi đa năng nào
  với hệ 16 bit, EyX là BX, BP, SI, DI

Phương pháp này thường được dùng trong các thao tác với phần tử mảng dữ liệu hoặc với các cấu trúc dữ liệu tương đương.

Ví dụ: MOV EAX, [EBX + 10H] ; trong đó (EBX) ≡ 1000H

lệnh MOV thực hiện sao nội dung ô nhớ địa chỉ DS: 1010H vào AX



- Kiểu xác định địa chỉ Chỉ số + khoảng dịch

  Địa chỉ offset ≡ [EzX + khoảng dịch]

  với hệ 32 bit, EzX là bất kỳ thanh ghi đa năng nào, trừ ESP
  với hệ 16 bit, EzX là SI, DI

Điều khác biệt giữa phương pháp định vị Chỉ số+khoảng dịch so với phương pháp định vị Cơ sở+khoảng dịch *bắt đầu từ hệ 32 bit* là có thể viết (ESI* bội số) và (EDI*bội số), trong đó bội số là 1, 2, 4, 8.

- Kiểu xác định địa chỉ Cơ sở + chỉ số

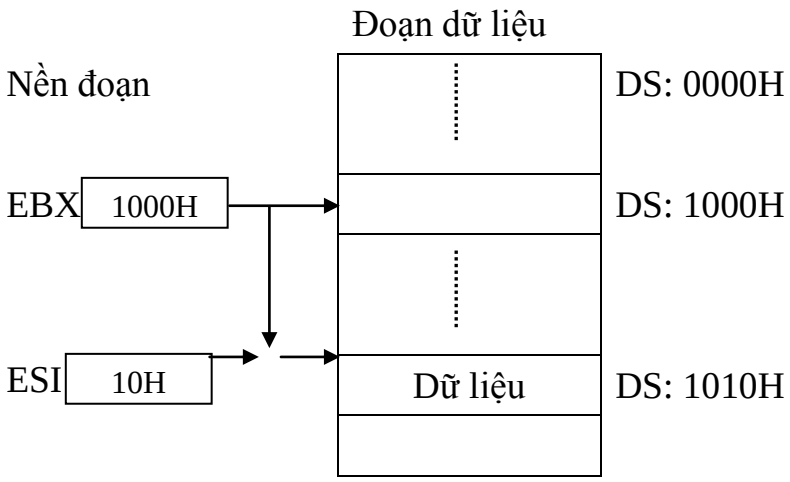
  Địa chỉ offset ≡ [EyX + EzX]

  trong đó với hệ 32 bit, EyX là bất kỳ thanh ghi đa năng nào và EzX là bất kỳ thanh ghi đa năng nào, trừ ESP.

  với hệ 16 bit, EyX là BX, BP và EzX là SI và DI

Ví dụ: MOV EAX, [EBX + ESI]; trong đó (EBX) ≡ 1000H, (ESI) ≡ 10H

lệnh MOV thực hiện sao chép nội dung ô nhớ có địa chỉ DS: 1010H vào EAX



Phương pháp định vị cơ sở chỉ số thường được dùng trong các thao tác với mảng dữ liệu hoặc với các cấu trúc dữ liệu tương đương.

- Kiểu xác định địa chỉ Cơ sở + chỉ số + khoảng dịch

  Địa chỉ offset ≡ [EyX + EzX + khoảng dịch]

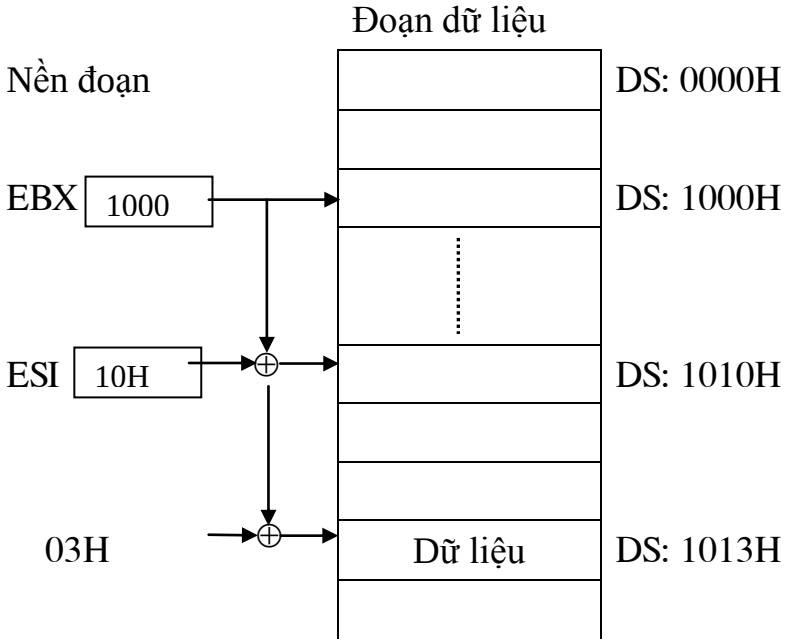
  trong đó:

  với hệ 32 bit, EyX là bất kỳ thanh ghi đa năng nào

  EzX là bất kỳ thanh ghi đa năng nào, trừ ESP.

  Có thể viết (ESI* bội số) và (EDI*bội số), trong đó bội số là 1, 2, 4, 8.

  với hệ 16 bit, EyX là BX, BP và EzX là SI và DI



Phương pháp định vị này thường được dùng trong các thao tác với mảng dữ liệu hai chiều hoặc với các cấu trúc dữ liệu phức tạp.

6.2.5. CHẾ ĐỘ THỰC VÀ CHẾ ĐỘ 8086 ẢO

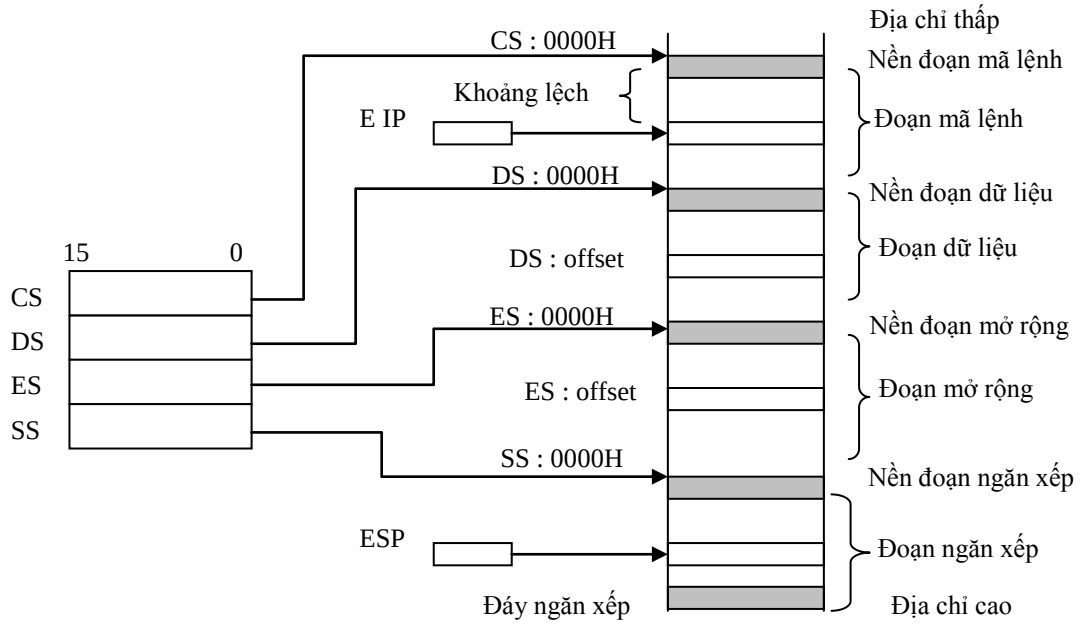
Bộ vi xử lý 32 bit Pentium cho phép thực hiện các chương trình ứng dụng 8086 ở hai chế độ: *chế độ thực và chế độ 8086 ảo*. CPU ở chế độ thực có kiến trúc cơ bản giống như 8086, nhưng tập các thanh ghi là 32 bit. Khi bộ xử lý được cấp nguồn hoặc được khởi động lại, nó được khởi động vào chế độ thực. Chế độ 8086 ảo cho phép thực hiện các ứng dụng 8086 (16 bit), nhưng đồng thời cho phép người thiết kế hệ thống sử dụng các tính năng của chế độ bảo vệ. Chế độ 8086 ảo thực chất là một nhiệm vụ chạy trong chế độ bảo vệ. Khi hệ điều hành chuyển đến nhiệm vụ 8086 ảo, CPU mô phỏng chế độ hoạt động của 8086. Chế độ 8086 ảo khác chế độ thực chỉ ở chỗ vẫn dùng một số dịch vụ của chế độ bảo vệ như phân trang, ngắt trong chế độ bảo vệ và xử lý ngoại lệ.

Trong chế độ thực, bộ nhớ được quản lý theo cơ chế phân đoạn. Cơ chế phân đoạn đã được trình bầy ở Chương 4, mục 4. 4. 1. Trong chế độ thực, mỗi đoạn có kích thước 64 Kb. Hệ điều hành có thể đặt các module của chương trình ở bất kỳ địa chỉ nào là bội của 16 trong bộ nhớ vật lý, khi đó đoạn nhớ sẽ mang tên của loại module nó chứa. Có 4 loại đoạn khác nhau: đoạn mã lệnh (code segment) chứa mã lệnh của chương trình, đoạn dữ liệu (data segment) chứa dữ liệu của chương trình, đoạn ngăn xếp (stack segment) chứa các thông tin và dữ liệu phục vụ chương trình con, đoạn mở rộng (extra segment) chứa dữ liệu mở rộng. Mỗi một đoạn có một địa chỉ gọi là *địa chỉ đoạn*.

Pentium có phần cứng hỗ trợ việc quản lý và truy nhập các đoạn, đó là các *thanh ghi đoạn* 16 bit CS, DS, SS, ES, FS, GS:

CS	Thanh ghi đoạn mã lệnh
DS	Thanh ghi đoạn dữ liệu
ES	Thanh ghi đoạn dữ liệu
SS	Thanh ghi đoạn ngăn xếp
FS	Thanh ghi đoạn dữ liệu
GS	Thanh ghi đoạn dữ liệu

Trong chế độ thực thanh ghi đoạn chứa địa chỉ đoạn. Thanh ghi CS chứa địa chỉ đoạn mã lệnh, thanh ghi DS, FS và GS chứa địa chỉ các đoạn dữ liệu, thanh ghi ES chứa địa chỉ đoạn mở rộng, thanh ghi SS chứa địa chỉ đoạn ngăn xếp (Hình 65). Địa chỉ đoạn xác định vị trí của đoạn trong bộ nhớ. Địa chỉ nền của đoạn vật lý (địa chỉ vật lý của ô nhớ đầu tiên của đoạn) được xác định bằng cách dịch trái địa chỉ đoạn đi 4 bit và điền các số 0 vào 4 bit thấp nhất.



Hình 65

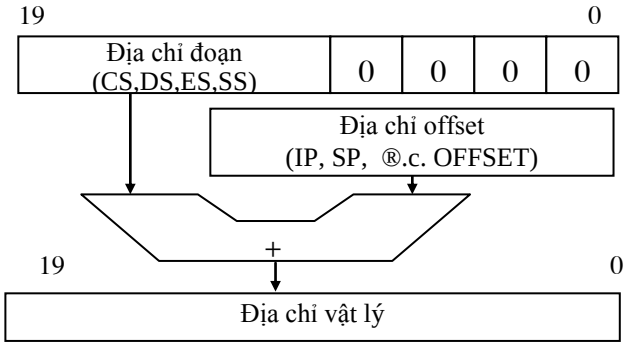
Mỗi một ô nhớ trong mỗi đoạn được định vị bằng một cặp con số: *địa chỉ đoạn và địa chỉ offset (địa chỉ lệch)*, trong đó địa chỉ offset là con số xác định vị trí của ô nhớ so với nền đoạn. Cặp số này được gọi là *địa chỉ logic* (địa chỉ ảo) của ô nhớ. Địa chỉ logic của một ô nhớ thường được biểu diễn ở dạng:

Địa chỉ đoạn: Địa chỉ offset

Thanh ghi IP chứa địa chỉ offset của ô nhớ chứa lệnh CPU sẽ nhập trong đoạn mã lệnh. Thanh ghi SP chứa địa chỉ offset của ô đỉnh ngăn xếp trong đoạn ngăn xếp. Khi truy nhập các đoạn dữ liệu, người lập trình cần xác định địa chỉ offset của các ô nhớ theo một trong các kiểu xác định địa chỉ toán hạng đã được quy định. Các thanhg ghi BX, BP, SI và DI được dùng để định vị toán hạng.

Trong chế độ thực, việc chuyển đổi địa chỉ logic sang địa chỉ vật lý được thực hiện như sau (Hình 66):

Địa chỉ vật lý = Địa chỉ đoạn*16 + Địa chỉ offset



Hình 66

6.2.6. Chế độ bảo vệ

Chế độ bảo vệ (Protected Mode) được thiết kế cho các bộ vi xử lý Intel từ CPU 80286 đến các CPU 32 bit như 80386, 80486 và Pentium v.v. sau này. Chế độ bảo vệ được thiết kế để hỗ trợ hệ điều hành đa nhiệm, cách ly và bảo vệ hệ điều hành khỏi những truy nhập trái phép của các chương trình ứng dụng, cách ly và bảo vệ chương trình ứng dụng này khỏi sự truy nhập trái phép của chương trình ứng dụng khác.

6.2.6.1 Các mức đặc quyền và luật về quyền truy nhập

- **Các mức đặc quyền**. Trong chế độ bảo vệ, mỗi đoạn nhớ được gắn một *mức đặc quyền* và được bảo vệ nhờ cơ chế về *quyền truy nhập*.

Các mức đặc quyền được thiết kế để hỗ trợ hoạt động của hệ điều hành đa nhiệm nhằm cách ly và bảo vệ hệ điều hành khỏi các truy nhập trái phép của chương trình ứng dụng, cách ly và bảo vệ chương trình ứng dụng này khỏi sự truy nhập trái phép của chương trình ứng dụng khác. Dựa vào mức đặc quyền và luật về quyền truy nhập mà CPU sẽ quyết định cho phép hay không cho phép truy nhập đoạn nhớ yêu cầu.

Các mức đặc quyền (ký hiệu là PL – Privilege Level) nằm trong một hệ thống các mức đặc quyền 4 cấp:

- Đặc quyền mức PL = 0, mức đặc quyền cao nhất. Mức đặc quyền PL = 0 được gán cho các chương trình quản lý thiết bị và quản lý bộ nhớ.
- Đặc quyền mức PL = 1. Mức đặc quyền PL = 1 được gán cho các chương trình thiết lập mức ưu tiên giữa các nhiệm vụ, chương trình hoán đổi dữ liệu giữa bộ nhớ chính và bộ nhớ thứ cấp (đĩa từ), chương trình quản lý các cổng vào/ra và các dịch vụ hệ thống khác.
- Đặc quyền mức PL = 2. Mức đặc quyền PL = 2 được gán cho các chương trình quản lý tệp, thư mục và các chức năng mở rộng của hệ điều hành.
- Đặc quyền mức PL = 3, mức thấp nhất. Mức đặc quyền PL = 3 được gán cho các chương trình ứng dụng.

- **Các luật về quyền truy nhập**.

Luật về quyền truy nhập xác định quy tắc truy nhập đoạn nhớ.

Luật 1: Dữ liệu được lưu trữ trong đoạn nhớ có mức đặc quyền PL = P chỉ có thể bị truy nhập bởi mã lệnh có mức đặc quyền bằng hoặc cao hơn P (CPL ≤ DPL), trong đó CPL là mức đặc quyền của mã lệnh thuộc nhiệm vụ đang thực hiện, DPL là mức đặc quyền của đoạn dữ liệu bị truy nhập).

Luật 2: Đoạn mã lệnh có mức đặc quyền PL = P có thể bị gọi hoặc truy nhập bởi mã lệnh thuộc nhiệm vụ có mức đặc quyền bằng hoặc thấp hơn P (CPL ≥ DPL). Đoạn mã lệnh có mức đặc quyền thấp có thể gọi hoặc truy nhập đoạn mã lệnh có mức đặc quyền cao hơn thông qua cổng gọi.

Theo các luật về quyền truy nhập thì chương trình đang thực hiện có thể truy nhập tự do vào các đoạn mã lệnh và đoạn dữ liệu có cùng mức đặc quyền. Một chương trình có thể truy nhập vào một đoạn dữ liệu có mức đặc quyền thấp hơn, nhưng nếu truy nhập hoặc gọi đoạn mã lệnh có mức đặc quyền cao hơn thì phải thông qua cổng gọi.

6.2.6.2 Quản lý bộ nhớ theo phân đoạn trong chế độ bảo vệ

Các đoạn nhớ trong chế độ bảo vệ được quản lý theo 3 thông số: địa chỉ nền đoạn, giới hạn đoạn và quyền truy nhập.

Do thông tin về mỗi đoạn khá lớn nên không thể chứa trong thanh ghi đoạn mà được chứa trong các *bộ mô tả đoạn*. Các bộ mô tả đoạn nằm trong *bảng bộ mô tả*.

Có ba loại bảng bộ mô tả:

- Bảng bộ mô tả toàn cục GDT (Bảng GDT - Global Descriptor Table). Bảng GDT quản lý các đoạn (các vùng nhớ) chứa các chương trình của hệ điều hành và chứa dữ liệu của hệ thống (các vùng nhớ chứa các thông tin có tính chất toàn cục, thuộc không gian nhớ toàn cục). Các chương trình ứng dụng có thể truy nhập vùng nhớ này.
- Bảng bộ mô tả cục bộ LDT (Bảng LDT - Local Descriptor Table). Mỗi Bảng LDT quản lý các vùng nhớ (đoạn nhớ) thuộc một nhiệm vụ. Các đoạn nhớ chứa mã lệnh và dữ liệu của một nhiệm vụ lại thuộc không gian nhớ cục bộ, do các thông tin này có tính chất cục bộ. Mã lệnh và dữ liệu của một nhiệm vụ đang chạy sẽ được bảo vệ trước sự truy nhập trái phép của các nhiệm vụ khác. Bản thân các Bảng LDT lại nằm trong không gian nhớ toàn cục.
- Bảng bộ mô tả ngắt (Bảng IDT - Interrupt Descriptor Table). Bảng IDT chứa các bộ mô tả trỏ đến 256 chương trình phục vụ ngắt. Bảng IDT đóng vai trò bảng véc tơ ngắt, trong đó mỗi véc tơ ngắt là một bộ mô tả.

Tất cả các bảng bộ mô tả (mà thuật ngữ chung gọi là bảng đoạn) đều nằm trong bộ nhớ chính.

a) Bộ chọn đoạn 16 bit

Trong chế độ bảo vệ các thanh ghi đoạn CS, DS, ES, SS, FS, GS không được dùng để xác định địa chỉ nền đoạn như trong chế độ thực, mà được dùng để *chọn bộ mô tả đoạn* trong các bảng bộ mô tả, tức là thực hiện chức năng *bộ*

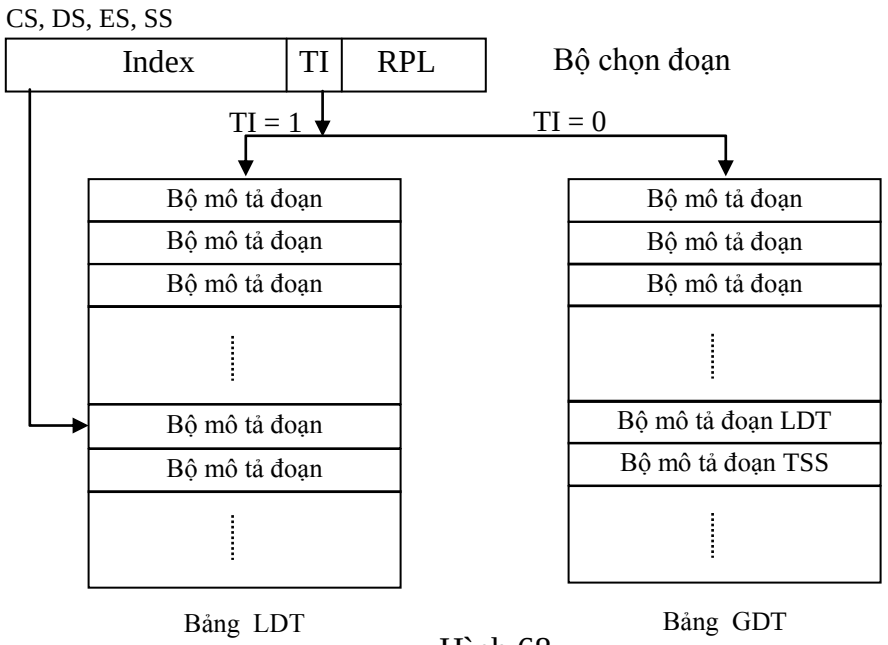
chọn đoạn. Mỗi bộ chọn đoạn có một phần cache (phần kín) chứa bản sao của bộ mô tả đoạn tương ứng (Hình 67).

	Phần hở	Phần kín chứa bản sao bộ mô tả đoạn		
CS	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập
DS	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập
ES	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập
SS	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập
FS	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập
GS	Bộ chọn đoạn	Địa chỉ nền vật lý	Giới hạn đoạn	Quyền truy nhập

Hình 67

Bộ chọn đoạn được dùng để xác định vị trí của bộ mô tả đoạn trong bảng bộ mô tả. Người lập trình phải nạp bộ chọn đoạn vào thanh ghi đoạn tương ứng khi muốn truy nhập một đoạn nào đó.

Bộ chọn đoạn có 3 phần (Hình 68).



Hình 68

- Phần Index 13 bit, dùng để xác định vị trí của bộ mô tả đoạn, tính từ nền của Bảng bộ mô tả.
- TI. TI xác định loại bảng bộ mô tả cần truy nhập:
  TI = 1 Truy nhập các bảng LDT
  TI = 0 Truy nhập bảng GDT
- RPL (Requestor Privilege Level)-Mức đặc quyền của người yêu cầu. RPL là mức đặc quyền PL của người cung cấp bộ chọn đoạn.

b) Bộ mô tả đoạn

Bộ mô tả đoạn chứa các thông tin quản lý một đoạn, bao gồm địa chỉ nền đoạn, kích thước (giới hạn) đoạn và quyền truy nhập đoạn. Bộ mô tả đoạn được hệ điều hành, trình biên dịch hoặc trình nạp bộ nhớ tạo ra.

Bộ mô tả đoạn gồm 8 byte:

<table>
<tr><td colspan="5">A31-A24</td><td>1 byte</td></tr>
<tr><td>G</td><td>D</td><td>0</td><td>AV
L</td><td>L19-L16</td><td>1 byte</td></tr>
<tr><td colspan="5">Quyền truy nhập</td><td>1 byte</td></tr>
<tr><td colspan="5">Địa chỉ nền đoạn A23 – A0</td><td>3 byte</td></tr>
<tr><td colspan="5">Giới hạn đoạn L15 – L0</td><td>2 byte</td></tr>
</table>

- Địa chỉ nền đoạn (A31 – A0) xác định địa chỉ nền của đoạn.
- Giới hạn đoạn (L19 – L0) xác định kích thước của đoạn.

  G - (Granularity Bit). Nếu G=0 giới hạn đoạn tối đa là 1M.

  Nếu G=1 giới hạn đoạn tối đa là 4G. Giới hạn đoạn tối đa được mô tả với L19-L0 = FFFFFH và với 12 bit thấp là số 0.

  D - (Default Operation Size). Chỉ có ý nghĩa với bộ mô tả đoạn mã lệnh
- Byte quyền truy nhập xác định mức đặc quyền và các thuộc tính khác của đoạn:

<table>
<tr><td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D0</td></tr>
<tr><td>P</td><td colspan="2">DPL</td><td>DT</td><td colspan="2">Kiểu bộ mô tả</td></tr>
</table>

P - (Present). Nếu P = 1 đoạn đang tồn tại trong bộ nhớ vật lý. Nếu P = 0, tức là đoạn này chưa được ánh xạ vào bộ nhớ vật lý. Nếu có yêu cầu truy nhập đoạn, CPU sẽ tạo ra một ngoại lệ "không tồn tại đoạn" để nạp đoạn tương ứng vào bộ nhớ vật lý.

DPL - (Descriptor Privilege Level). DPL xác định mức đặc quyền của bộ mô tả (mức đặc quyền của đoạn)

DT - (Descriptor Type). DT xác định loại bộ mô tả.

DT = 1 Bộ mô tả đoạn mã lệnh hoặc dữ liệu

DT = 0 Bộ mô tả đoạn hệ thống hoặc cổng giao dịch

Kiểu bộ mô tả - Cấu trúc của trường này phụ thuộc vào loại bộ mô tả. Có các loại bộ mô tả sau: bộ mô tả đoạn dữ liệu, bộ mô tả đoạn mã lệnh, bộ mô tả đoạn hệ thống. Bộ mô tả đoạn hệ thống (DT=0) có 2 loại: bộ mô tả LDT và bộ mô tả TSS. Bộ mô tả cổng giao dịch (DT=0) (còn gọi là cổng giao dịch) được dùng để truy nhập vào các đoạn mã lệnh.

- Cấu trúc của byte quyền truy nhập trong bộ mô tả đoạn dữ liệu:

7	6	5	4	3	2	1	0
P	DPL		1	0	ED	W/R	A

ED (Expansion Direction): xác định hướng truy nhập đoạn (hướng tiến triển của địa chỉ)

ED = 1: hướng địa chỉ giảm, đoạn dữ liệu là loại ngăn xếp

ED = 0: hướng địa chỉ tăng

W/R (Write/Read): xác định quyền ghi/đọc

W/R = 1: cho đọc/ghi đoạn dữ liệu

W/R = 0: cấm ghi đoạn dữ liệu

A (Accessed):

A = 1 đoạn đã bị truy nhập

- Cấu trúc của byte quyền truy nhập trong bộ mô tả đoạn mã lệnh:

7	6	5	4	3	2	1	0
P	DPL		1	1	C	R	A

C (Conforming):

C = 0 chương trình con sẽ thực hiện với mức đặc quyền PL = DPL

C = 1 chương trình sẽ thực hiện với mức đặc quyền PL bằng mức đặc quyền của đoạn chứa chương trình gọi chương trình con này.

R (Read):

R = 0: Đoạn mã lệnh thực hiện được

R = 1: Đoạn mã lệnh thực hiện được và đọc được

A (Accessed):

A = 1 đoạn mã lệnh đã bị truy nhập

- Cấu trúc của byte quyền truy nhập trong bộ mô tả đoạn hệ thống:

Bộ mô tả đoạn hệ thống (bộ mô tả TSS, bộ mô tả LDT) quy chiếu đến (trỏ đến) các đoạn chứa thông tin hệ thống.

7	6	5	4	3	2	1	0
P	DPL		0	0	0	Kiểu đoạn	

Kiểu đoạn:

- Kiểu=1: Bộ mô tả quy chiếu đến đoạn trạng thái nhiệm vụ TSS, nhiệm vụ này không ở trạng thái đang thực hiện
- Kiểu=2: Bộ mô tả quy chiếu đến đoạn chứa bảng LDT
- Kiểu=3: Bộ mô tả quy chiếu đến đoạn trạng thái nhiệm vụ TSS của nhiệm vụ đang thực hiện.

c) Bộ mô tả cổng giao dịch (cổng giao dịch)

Bộ mô tả cổng giao dịch (cổng giao dịch) được dùng để truy nhập vào các đoạn mã lệnh. Cổng giao dịch cung cấp phương tiện chuyển giao điều khiển giữa chương trình nguồn và chương trình đích, ví dụ qua các lệnh CALL. Cổng giao dịch cung cấp địa chỉ của chương trình được gọi thông qua bộ chọn (trỏ đến bộ mô tả đoạn mã lệnh) và địa chỉ offset.

Cổng giao dịch có cấu trúc như sau:

Địa chỉ offset A31-A16				2 byte
Quyền truy nhập				1 byte
0	0	0	Bộ đếm	1 byte
Bộ chọn				2 byte
Địa chỉ offset A15-A0				2 byte

- Bộ chọn, trỏ đến bộ mô tả đoạn mã lệnh đích
- Địa chỉ offset của mã lệnh đích
- Bộ đếm (WC - word count): xác định số từ cần sao chép từ ngăn xếp của chương trình gọi sang chương trình được gọi. Thông số WC chỉ có ở cổng giao dịch kiểu gọi (cổng gọi).
- Byte quyền truy nhập trong bộ mô tả cổng giao dịch (cổng giao dịch):

7	6 5	4	3 2 1 0
P	DPL	0	Kiểu cổng giao dịch

Kiểu cổng giao dịch: có 4 loại cổng giao dịch.

- Kiểu = 4: cổng giao dịch kiểu gọi (cổng gọi)
- Kiểu = 5: cổng giao dịch kiểu nhiệm vụ (cổng nhiệm vụ)
- Kiểu = 6: cổng giao dịch kiểu ngắt (cổng ngắt)
- Kiểu = 7: cổng giao dịch kiểu bẫy (cổng bẫy)

Bộ mô tả cổng giao dịch kiểu gọi (cổng gọi) thường được dùng để chuyển điều khiển từ chương trình nguồn có mức đặc quyền thấp hơn tới chương trình đích có mức đặc quyền cao hơn.

Bộ mô tả cổng giao dịch kiểu nhiệm vụ (cổng nhiệm vụ) được sử dụng khi có sự thay đổi nhiệm vụ trong nhiệm vụ hiện hành. Bộ mô tả cổng giao dịch kiểu nhiệm vụ quy chiếu đến (trỏ tới) đoạn TSS. Trong trường hợp này trường địa chỉ offset không được dùng.

Bộ mô tả cổng giao dịch kiểu ngắt và kiểu bẫy (cổng ngắt và cổng bẫy) cung cấp bộ chọn trỏ đến bộ mô tả đoạn mã lệnh chứa chương trình con phục vụ ngắt và địa chỉ offset xác định vị trí của chương trình con phục vụ ngắt bên trong đoạn mã lệnh đó.

d) Khả năng quản lý bộ nhớ theo phân đoạn trong chế độ bảo vệ

Trong chế độ bảo vệ, một đoạn có thể bắt đầu từ bất kỳ địa chỉ nào trong bộ nhớ. Mỗi ô nhớ được định vị bằng một địa chỉ logic (địa chỉ ảo) gồm hai thành phần

Bộ chọn đoạn (16 bit): Địa chỉ offset (32 bit)

Bộ chọn đoạn 16 bit được dùng để trỏ đến bộ mô tả đoạn, qua đó xác định được địa chỉ nền của đoạn. Địa chỉ offset xác định vị trí của ô nhớ bên trong đoạn.

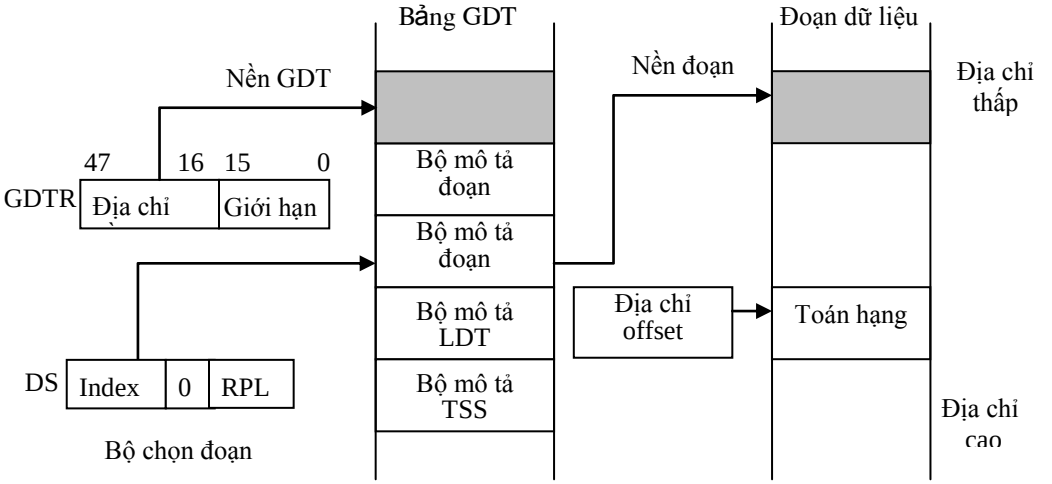
Mỗi một nhiệm vụ có thể có tới $2x2^{13}$ = 16K bộ chọn đoạn, mỗi bộ chọn đoạn trỏ tới 1 bộ mô tả đoạn. Mỗi bộ mô tả đoạn có thể quản lý được một đoạn có kích thước giới hạn tối đa là 2^{32} = 4 Gbyte. Như vậy mỗi nhiệm vụ có thể có một không gian địa chỉ cực đại là

$$2^{14}*2^{32} = 2^{46} = 64 \text{ Tera byte}$$

Nói cách khác, với cơ chế quản lý bộ nhớ qua bộ mô tả, Pentium và các CPU 32 bit khác của Intel có khả năng quản lý một không gian địa chỉ (ảo) tới 64 Tera byte.

e) Cơ chế truy nhập bộ nhớ (ô nhớ) qua bảng GDT

Cơ chế truy nhập bộ nhớ (ô nhớ) qua bảng GDT được mô tả qua hình sau (Hình 69). Bảng GDT được hệ điều hành tạo ra khi khởi động hệ thống. CPU quản lý bảng GDT qua thanh ghi GDTR. Thanh ghi GDTR chứa hai thông tin về bảng GDT: địa chỉ nền bảng và kích thước (giới hạn) bảng.



Hình 69

Khi có yêu cầu truy nhập đoạn, người yêu cầu cung cấp bộ chọn đoạn. CPU thực hiện thao tác kiểm tra quyền truy nhập đoạn trước khi cho truy nhập. Ví dụ, đối với việc truy nhập đoạn dữ liệu quá trình kiểm tra được tiến hành theo quy tắc:

$$EPL = \max(CPL, RPL) \leq DPL$$

trong đó:

- CPL là mức đặc quyền của nhiệm vụ đang thực hiện. Thông thường CPL có giá trị bằng mức đặc quyền của đoạn chứa mã lệnh đang chạy. Có thể thay đổi mức đặc quyền của nhiệm vụ bằng cách chuyển điều khiển thông qua cổng giao dịch.
- RPL là mức đặc quyền yêu cầu, xác định mức đặc quyền của bộ chọn. Mức đặc quyền RPL được sinh ra bởi người nạp bộ chọn đoạn.
- EPL là mức đặc quyền hiệu dụng.
- DPL là mức đặc quyền của đoạn bị truy nhập.

Nếu điều kiện trên không được thoả mãn thì sẽ sinh ra một ngoại lệ và CPU không cho truy nhập đoạn. Nếu điều kiện về quyền truy nhập được thoả mãn thì CPU cho truy nhập đoạn. Việc truy nhập từng ô nhớ trong đoạn được thực hiện thông qua địa chỉ nền đoạn (có được từ bộ mô tả đoạn vừa được chọn) và địa chỉ offset của ô nhớ đó.

Đối với việc truy nhập đoạn mã lệnh, quá trình kiểm tra được tiến hành theo quy tắc:

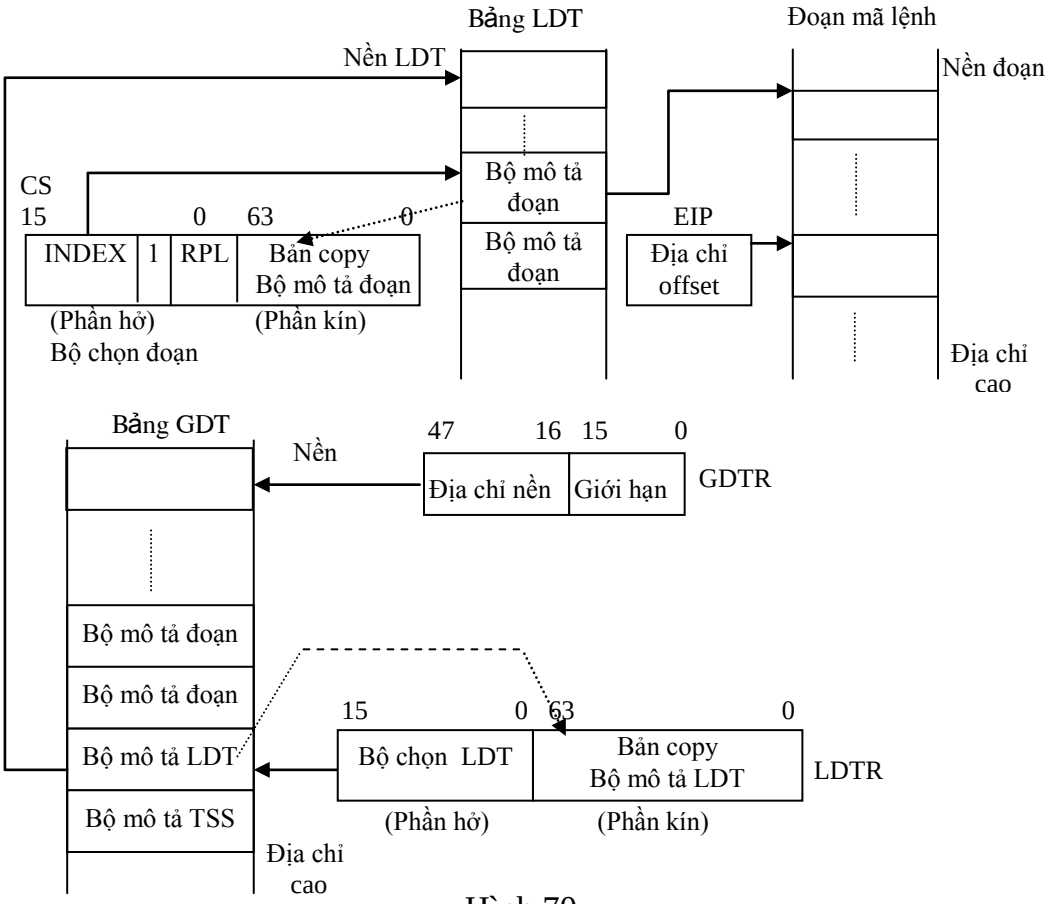
$$EPL = \max(CPL, RPL) \geq DPL$$

trong đó việc truy nhập một đoạn mã lệnh có mức đặc quyền cao hơn (EPL > DPL) phải thực hiện thông qua cổng gọi.

f) Cơ chế truy nhập bộ nhớ (ô nhớ) qua bảng LDT

Bảng LDT được hệ điều hành tạo ra khi nạp một chương trình ứng dụng vào bộ nhớ hoặc khi thực hiện một nhiệm vụ. Mỗi bảng LDT quản lý các đoạn của một nhiệm vụ. Việc quản lý các đoạn (các vùng nhớ) thuộc một nhiệm vụ được tổ chức như sau.

Mỗi một đoạn nhớ được quản lý bởi một bộ mô tả đoạn. Các bộ mô tả đoạn của một nhiệm vụ được chứa trong một bảng LDT riêng biệt của nhiệm vụ đó. Nói cách khác, mỗi bảng LDT quản lý các đoạn nhớ của một nhiệm vụ. Mỗi bảng LDT được quản lý bởi một bộ mô tả LDT. Bộ mô tả LDT chứa địa chỉ nền bảng LDT, kích thước bảng, quyền truy nhập bảng (quyền truy nhập nhiệm vụ). Các bộ mô tả LDT của các nhiệm vụ được chứa trong bảng GDT. Bảng GDT được quản lý bởi thanh ghi hệ thống GDTR (Hình 70).



Hình 70

Khi một nhiệm vụ được thực hiện, hệ điều hành sẽ nạp bộ chọn LDT vào thanh ghi hệ thống LDTR. Thanh ghi LDTR trỏ đến bộ mô tả LDT trong bảng GDT, từ đây CPU thông qua bảng LDT quản lý được các đoạn của nhiệm vụ đó và bắt đầu (hoặc tiếp tục) thực hiện nhiệm vụ này. Để truy nhập các đoạn trong nhiệm vụ, người yêu cầu cần nạp bộ chọn đoạn vào thanh ghi đoạn tương ứng. CPU thực hiện thao tác kiểm tra quyền truy nhập đoạn. Nếu điều kiện về quyền truy nhập được thoả mãn thì CPU cho truy nhập đoạn. Nếu điều kiện trên không được thoả mãn thì CPU sẽ sinh ra một ngoại lệ và không cho truy nhập đoạn. Việc truy nhập từng ô nhớ trong đoạn được thực hiện thông qua địa chỉ nền đoạn (có được từ bộ mô tả đoạn vừa được chọn) và địa chỉ offset của ô nhớ đó.

g) Cơ chế chuyển điều khiển và gọi chương trình con trong chế độ bảo vệ

Việc chuyển điều khiển xảy ra khi thực hiện các lệnh nhảy (lệnh JMP) hoặc lệnh gọi chương trình con (lệnh CALL).

Trường hợp thực hiện lệnh nhảy hoặc lệnh gọi *trong cùng đoạn mã lệnh* của nhiệm vụ đang chạy (*lệnh nhảy gần*, *lệnh gọi gần*) xảy ra như sau:

- Khi thực hiện lệnh nhảy gần (*near jump*), con trỏ lệnh IP được nạp giá trị mới. Chương trình tiếp tục được thực hiện từ vị trí mới do IP trỏ đến.
- Khi thực hiện lệnh gọi gần (NEAR CALL), đơn vị xử lý trung tâm thực hiện các thao tác sau:
  + Cất giá trị hiện thời của IP vào ngăn xếp
  + Nạp địa chỉ offset của chương trình con được gọi (đích) vào IP
  + Thực hiện chương trình con (đích).

Trường hợp chuyển điều khiển đến những đoạn mã lệnh khác khi thực hiện lệnh gọi xa (FAR CALL) thì có hai tình huống:

- Đoạn mã lệnh đích có mức đặc quyền thấp hơn hoặc bằng mức đặc quyền của đoạn mã lệnh nguồn hiện tại. Khi đó đơn vị xử lý trung tâm thực hiện các thao tác sau:
  + Cất giá trị hiện thời của CS và EIP vào ngăn xếp
  + Nạp bộ chọn đoạn mã lệnh chứa chương trình con (đích) vào CS
  + Nạp địa chỉ offset của chương trình con (đích) vào EIP
  + Thực hiện chương trình đích

  Lệnh RET cho phép rời khỏi chương trình con để trở về chương trình gọi nó. Lệnh này khôi phục lại nội dung bộ chọn đoạn mã lệnh nguồn (nội dung thanh ghi CS), nội dung con trỏ lệnh (nội dung thanh ghi EIP) và tiếp tục thực hiện chương trình đã gọi chương trình con này.
- Đoạn mã lệnh chương trình con (đích) có mức đặc quyền cao hơn mức đặc quyền của đoạn mã lệnh nguồn hiện tại. Khi đó việc gọi chương trình con (đích) phải thực hiện qua cổng gọi. Bộ chọn đoạn lúc này không trỏ đến bộ mô tả đoạn mã lệnh chứa chương trình con (đích), mà trỏ đến cổng gọi (bộ mô tả cổng gọi). Cổng gọi trỏ đến bộ mô tả đoạn mã lệnh của chương trình con (đích), cổng gọi cũng chứa địa chỉ offset bắt đầu chương trình con (đích), qua đó gọi được chương trình con (đích). Bộ xử lý trung tâm thực hiện quá trình này như sau:
  + Tạm lưu giữ nội dung CS, EIP, SS, ESP hiện thời (thuộc chương trình nguồn)
  + Nạp bộ chọn cổng gọi và kiểm tra quyền truy nhập
  + Cất giữ giá trị tạm lưu của SS và ESP nguồn vào ngăn xếp đích

+ Chuyển các tham số từ ngăn xếp nguồn sang ngăn xếp đích
+ Cất giữ giá trị tạm lưu của CS và EIP nguồn vào ngăn xếp đích
+ Nạp bộ chọn bộ mô tả đoạn mã lệnh đích và địa chỉ offset (lấy từ cổng gọi), qua đó nạp bộ mô tả đoạn mã lệnh đích
+ Thực hiện chương trình con (đích)

Khi bộ xử lý trung tâm gặp lệnh RET thì việc trở về chương trình nguồn được thực hiện bắt đầu bằng việc kiểm tra quyền truy nhập, sau đó là khôi phục nội dung các thanh ghi CS, EIP, SS, ESP theo một trình tự ngược lại.

6.2.6.3 Cơ chế chuyển nhiệm vụ

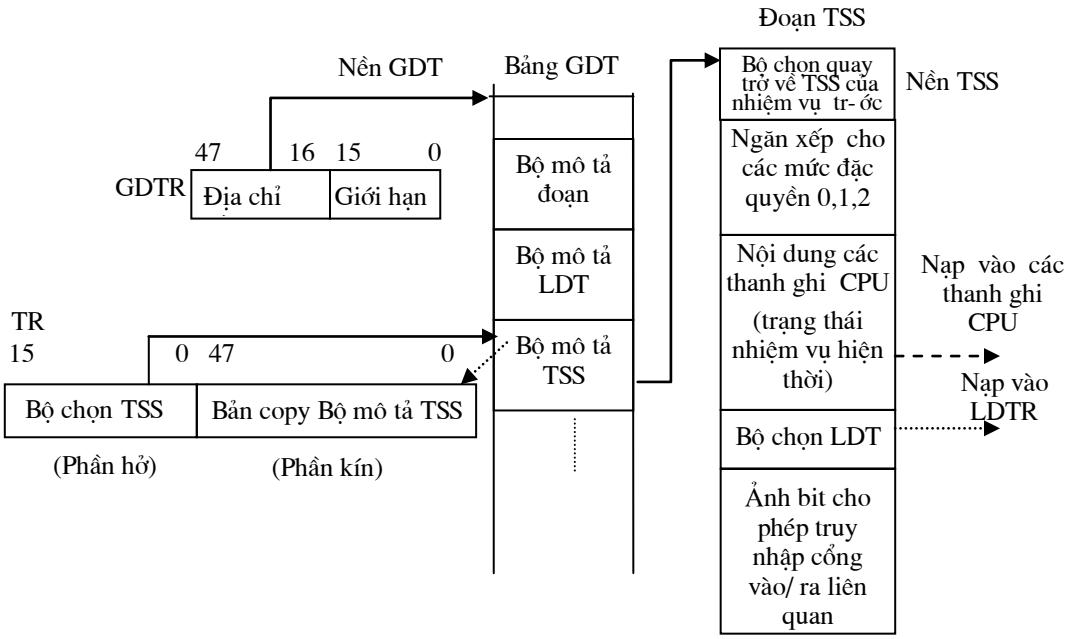
Nhiệm vụ được định nghĩa như là một thực hiện của chương trình nào đó. Pentium có phần cứng hỗ trợ quản lý và hỗ trợ thao tác chuyển nhiệm vụ. Thao tác chuyển nhiệm vụ thực hiện lưu và bảo vệ toàn bộ trạng thái hoạt động của nhiệm vụ hiện thời (bao gồm nội dung toàn bộ các thanh ghi của CPU, không gian địa chỉ có liên quan và bộ chọn LDT của nhiệm vụ đang chạy) vào đoạn trạng thái của nhiệm vụ TSS (Task State Segment) của nhiệm vụ này, sau đó nạp trạng thái của nhiệm vụ tiếp theo từ đoạn TSS tương ứng vào CPU, kiểm tra quyền truy nhập và bắt đầu thực hiện nhiệm vụ mới.

Thao tác chuyển nhệm vụ được thực hiện bởi lệnh CALL hoặc lệnh JMP giữa các đoạn. Khi các lệnh này được thực hiện sẽ quy chiếu tới đoạn trạng thái nhiệm vụ TSS hoặc bộ mô tả cổng nhiệm vụ trong GDT hoặc LDT. Bộ mô tả cổng nhiệm vụ chứa bộ chọn TSS. Các ngắt cứng, lệnh INT n, ngoại lệ hoặc bẫy cũng có thể gây ra thao tác chuyển nhiệm vụ, nếu bộ mô tả cổng nhiệm vụ được cài trong bảng IDT tương ứng.

Mỗi một nhiệm vụ có một đoạn trạng thái nhiệm vụ (đoạn TSS) chứa toàn bộ trạng thái của nhiệm vụ đó như: nội dung các bộ chọn đoạn, thanh ghi EIP, EFLAGS, các thanh ghi đa năng, con trỏ ngăn xếp, các thanh ghi địa chỉ bảng trang CR, thanh ghi LDTR v.v. Mỗi đoạn TSS được quản lý (trỏ) bởi một bộ mô tả TSS nằm trong bảng GDT.

Đoạn trạng thái nhiệm vụ TSS hiện thời được quản lý bởi một thanh ghi đặc biệt tên là TR (TSS Register) trong CPU. Thanh ghi TR có hai phần, phần hở là bộ chọn hệ thống 16 bit và phần kín 32 bit, chứa bản sao của bộ mô tả TSS. Bộ chọn TSS trỏ đến bộ mô tả TSS quản lý nhiệm vụ hiện thời. Khi các lệnh CALL hoặc JMP được thực hiện, bộ chọn TSS được nạp vào TR, nó trỏ tới bộ mô tả TSS tương ứng trong GDT. Bộ mô tả TSS (nội dung của nó được sao chép vào phần kín của TR) trỏ tới đoạn TSS của nhiệm vụ được chuyển tới.

Thao tác chuyển nhiệm vụ (Hình 71) được tiến hành theo các bước sau:



Hình 71

+ Toàn bộ trạng thái hoạt động của nhiệm vụ hiện thời (bao gồm nội dung toàn bộ các thanh ghi của CPU, các địa chỉ có liên quan và nội dung thanh ghi LDTR của nhiệm vụ hiện thời) được lưu vào đoạn trạng thái nhiệm vụ TSS của nhiệm vụ hiện thời.
+ Nạp bộ chọn TSS nhiệm vụ mới vào thanh ghi TR. Thanh ghi TR trỏ đến bộ mô tả TSS quản lý đoạn TSS của nhiệm vụ mới.
+ Bộ mô tả TSS của nhiệm vụ mới được nạp vào phần kín của TR. Qua bộ mô tả TSS truy nhập đoạn TSS của nhiệm vụ mới, nạp trạng thái nhiệm vụ mới vào các thanh ghi của CPU, trong đó có thanh ghi LDTR.
+ Thực hiện kiểm tra quyền truy nhập.
+ Thực hiện nhiệm vụ mới.

Việc quay trở lại nhiệm vụ vừa bị dừng được thực hiện bằng lệnh IRET. Quá trình quay trở lại diễn ra như sau:

+ Trạng thái của nhiệm vụ hiện thời được cất giữ vào đoạn TSS của nó.
+ Bộ chọn TSS quay trở về nhiệm vụ trước được nạp vào TR, cho phép truy nhập đoạn TSS của nhiệm vụ trước, qua đó khôi phục trạng thái nhiệm vụ trước đó vào CPU.

6.2.6.4 Quản lý bộ nhớ theo phân trang trong chế độ bảo vệ

Trong chế độ bảo vệ, Pentium có phần cứng hỗ trợ quản lý bộ nhớ theo cả hai cơ chế phân đoạn và phân trang. Các cơ chế quản lý bộ nhớ theo phân đoạn và phân trang tổng quát đã được trình bày ở chương 4. Trong phần này ta chỉ trình bầy cơ chế quản lý bộ nhớ theo phân trang trong CPU Pentium và các CPU 32 bit khác của Intel.

Đơn vị quản lý bộ nhớ chuyển *địa chỉ logic* (*địa chỉ ảo*) gồm *địa chỉ nền đoạn 32 bit* và *địa chỉ offset 32 bit* thành *địa chỉ tuyến tính*.

Nếu chọn cách quản lý bộ nhớ thuần phân đoạn thì địa chỉ tuyến tính sẽ là địa chỉ vật lý, địa chỉ này được dùng để truy nhập bộ nhớ (ô nhớ) vật lý.

Nếu chọn cách quản lý bộ nhớ kiểu hỗn hợp *phân đoạn phân trang* thì đơn vị quản lý bộ nhớ sẽ chuyển địa chỉ tuyến tính thành địa chỉ vật lý theo cơ chế phân trang. Trong cơ chế quản lý bộ nhớ kiểu hỗn hợp, cơ chế phân đoạn được dùng để xác định địa chỉ logic (địa chỉ ảo) của ô nhớ. Cơ chế phân đoạn được dùng (và rất phù hợp) để quản lý (bảo vệ) việc truy nhập bộ nhớ (ảo) và quản lý nhiệm vụ. Tiếp đó, cơ chế phân trang được dùng (và rất phù hợp) cho việc quản lý bộ nhớ ảo theo nghĩa cung cấp các khung trang vật lý ứng với các trang nhớ (ảo), chuyển hoá địa chỉ ảo thành địa chỉ vật lý (thực) và thực hiện truy nhập bộ nhớ (ô nhớ) vật lý.

Trong cơ chế quản lý theo trang, chương trình được chia thành các khúc nhỏ có kích thước xác định, được gọi là *trang*. Mỗi trang chương trình được gán vào từng đoạn nhỏ của bộ nhớ, mỗi đoạn nhỏ chứa trang được gọi là một *khung trang*.

Pentium sử dụng hai mức bảng trang để chuyển địa chỉ tuyến tính thành địa chỉ vật lý, đó là *thư mục trang và bảng trang*.

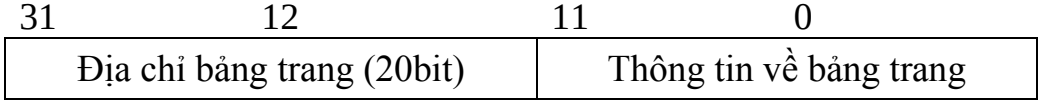
Cấu trúc của hệ thống quản lý trang gồm 3 phần: *thư mục trang* (Page Directory), *bảng trang* (Page Table) và *khung trang* (Page Frame). Thư mục trang, bảng trang và khung trang đều có cùng một kích thước là 4Kbyte.

+ Thư mục trang:

Thư mục trang có kích thước 4Kbyte. Thư mục trang có thể chứa tới 1024 *mục thư mục trang* PDE (Page Directory Entry). 10 bit cao nhất (bit 31 đến bit 22) trong địa chỉ tuyến tính 32 bit được dùng để chọn PDE trong thư mục trang.

PDE, có kích thước 32 bit, được dùng để trỏ tới bảng trang ở mức tiếp theo. Mỗi một thư mục trang có thể quản lý được tới 1024 bảng trang.

Cấu trúc một PDE trong thư mục trang:



- Phần Địa chỉ bảng trang chứa 20 bit cao của địa chỉ nền bảng trang (địa chỉ bắt đầu một bảng trang), 12 bit thấp của địa chỉ nền bảng trang luôn bằng 0.
- Phần Thông tin về bảng trang (12 bit) chứa các thông tin về quyền truy nhập bảng trang và một số thuộc tính khác của trang. Bit 0 (bit P - Present) chỉ thị có thể dùng PDE để chuyển hoá địa chỉ. Nếu P=1 thì có thể dùng PDE này. Các bit 1 (R/W bit) và bit 2 (U/S bit) cung cấp mức đặc quyền và quyền ghi/đọc trang của tất cả các trang trỏ bởi PDE này.

Trong cơ chế quản lý bộ nhớ theo trang, 10 bit cao nhất (từ bit 31 đến bit 22) của địa chỉ tuyến tính được dùng để xác định vị trí của PDE trong thư mục trang, vị trí so với nền thư mục trang.

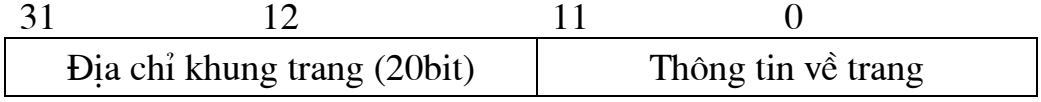
+ Bảng trang:

Bảng trang chỉ ra vị trí khung trang (xác định địa chỉ nền khung trang) của mỗi trang chương trình. CPU dựa vào địa chỉ nền khung trang và địa chỉ offset của ô nhớ trong trang để tính ra địa chỉ vật lý của ô nhớ.

Bảng trang có kích thước 4 Kbyte. Bảng trang chứa 1024 *mục bảng trang* PTE (Page Table Entry).

PTE có kích thước 32 bit. Mỗi *PTE chứa địa chỉ nền* của một *khung trang* 4Kbyte. Một bảng trang có thể quản lý được tới 1024 khung trang.

Cấu trúc một PTE trong bảng trang:



- Phần Địa chỉ khung trang chứa 20 bit cao của địa chỉ vật lý nền khung trang (địa chỉ vật lý bắt đầu một khung trang), 12 bit thấp của địa chỉ vật lý nền khung trang luôn bằng 0.
- Phần Thông tin về trang chứa các thông tin về quyền truy nhập trang và một số thuộc tính khác của trang. Phần Thông tin về trang (12 bit), tương tự như ở PDE, chứa các thông tin về quyền truy nhập trang và một số thuộc tính khác của trang. Bit 0 (bit P - Present) chỉ thị có thể dùng PTE để chuyển hoá địa chỉ. Nếu P=1 thì có thể dùng PTE này. Các bit 1 (R/W bit) và bit 2 (U/S bit) cung cấp mức đặc quyền và quyền ghi/đọc trang của tất cả các trang trỏ bởi PTE này.

Trong cơ chế quản lý bộ nhớ theo trang, 10 bit của địa chỉ tuyến tính (từ bit 21 đến bit 12) được dùng để xác định vị trí của PTE trong bảng trang, vị trí so với nền bảng trang.

Có hai mức đặc quyền được dùng để bảo vệ trang: mức chương trình ứng dụng (tương đương mức 3 trong quản lý theo đoạn) và mức chương trình giám sát (tương đương mức 0, 1, 2 trong quản lý theo đoạn).

Mức thấp nhất trong hai mức của PDE và PTE được lấy làm mức đặc quyền và quyền ghi/đọc của từng trang.

Địa chỉ vật lý của ô nhớ trong trang được tạo ra tính bằng cách kết hợp 20 bit cao (nằm ở phần Địa chỉ khung trang trong PTE) với 12 bit thấp nhất (phần đóng vai trò địa chỉ offset) trong địa chỉ tuyến tính.

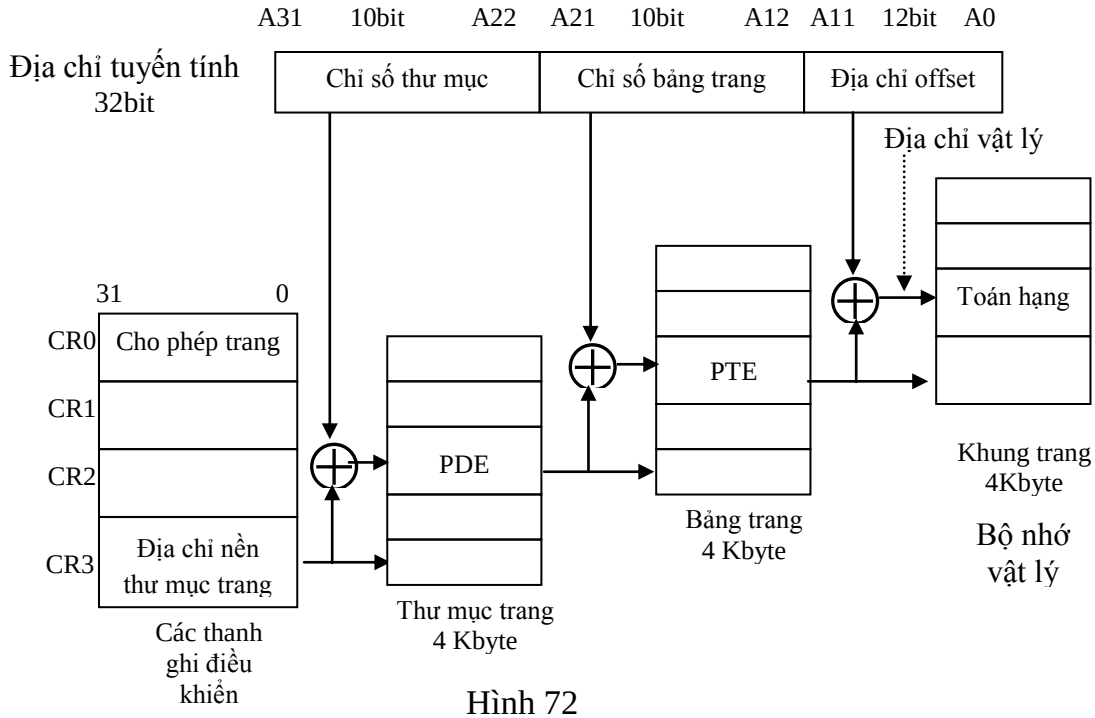
- **Cơ chế truy nhập bộ nhớ theo phân trang**

(Cơ chế chuyển địa chỉ tuyến tính sang địa chỉ vật lý)

Đơn vị quản lý phân trang nhận địa chỉ tuyến tính 32 bit từ đơn vị quản lý phân đoạn.

Địa chỉ tuyến tính được tách thành 3 phần (Hình 72):

+ $A_{31} \div A_{22}$: 10 bit chỉ số PDE, xác định vị trí PDE trong thư mục trang.
+ $A_{21} \div A_{12}$: 10 bit chỉ số PTE, xác định vị trí PTE trong bảng trang.
+ $A_{11} \div A_0$: là 12 bit địa chỉ offset của ô nhớ trong trang.



Hình 72

Đơn vị quản lý phân trang lấy 20 bit cao của địa chỉ này so sánh với các mục trong TLB để tìm địa chỉ trùng hợp (PTE trùng). *Nếu trùng* thì địa chỉ vật lý 32 bit được tính và việc truy nhập bộ nhớ vật lý được thực hiện. Nếu *không trùng*, PTE cần tìm không có trong TLB, thì đơn vị quản lý phân trang sẽ đọc PDE (Hình 80). Trong PDE, nếu thấy P=1, có nghĩa bảng trang đang có trong

bộ nhớ. Đơn vị quản lý phân trang đọc PTE trong bảng trang. Nếu trong PTE thấy P=1, có nghĩa trang cần truy nhập đang có trong bộ nhớ, khi đó địa chỉ vật lý được tính và việc truy nhập bộ nhớ vật lý được thực hiện; tiếp đó PTE này được đọc vào TLB để dùng cho việc truy nhập bộ nhớ trong các lần sau và bit 5 (bit A-Accesed) được đặt lên 1. Nếu P=0 trong PDE hoặc PTE thì CPU sẽ sinh ra ngoại lệ báo lỗi trang, khi đó trang lỗi sẽ được đọc từ bộ nhớ thứ cấp vào bộ nhớ chính.

Địa chỉ vật lý của ô nhớ (trong khung trang) được tính bằng cách kết hợp 20 bit địa chỉ nằm ở Phần địa chỉ nền khung trang trong PTE (đóng vai trò 20 bit địa chỉ cao), với 12 bit thấp từ A11 đến A0 trong địa chỉ tuyến tính này (đóng vai trò là địa chỉ offset).

6.2.7. Ngắt và ngoại lệ

Các vấn đề chung về ngắt và phục vụ ngắt đã được trình bầy ở chương 5. Mục này chỉ trình bầy một số điểm riêng liên quan đến kiến trúc của CPU Pentium và các CPU dòng x86 nói chung.

Ngoại lệ cũng gây ngắt bộ xử lý và làm CPU chạy chương trình con phục vụ ngoại lệ. Phản ứng của CPU với ngắt và ngoại lệ tương tự nhau. Điểm khác giữa ngoại lệ và ngắt là ở chỗ, ngắt được gây ra bởi thiết bị bên ngoài CPU, còn ngoại lệ được gây ra bởi các lỗi bên trong bộ xử lý hoặc các ngoại lệ (ví dụ, tình huống lỗi khi thực hiện chia cho 0, hoặc nhập mã lệnh sai, hoặc phát hiện truy nhập trái phép trong cơ chế bảo vệ bộ nhớ v.v.); xử lý ngắt xoá cờ ngắt IF còn xử lý ngoại lệ không xoá cờ ngắt IF.

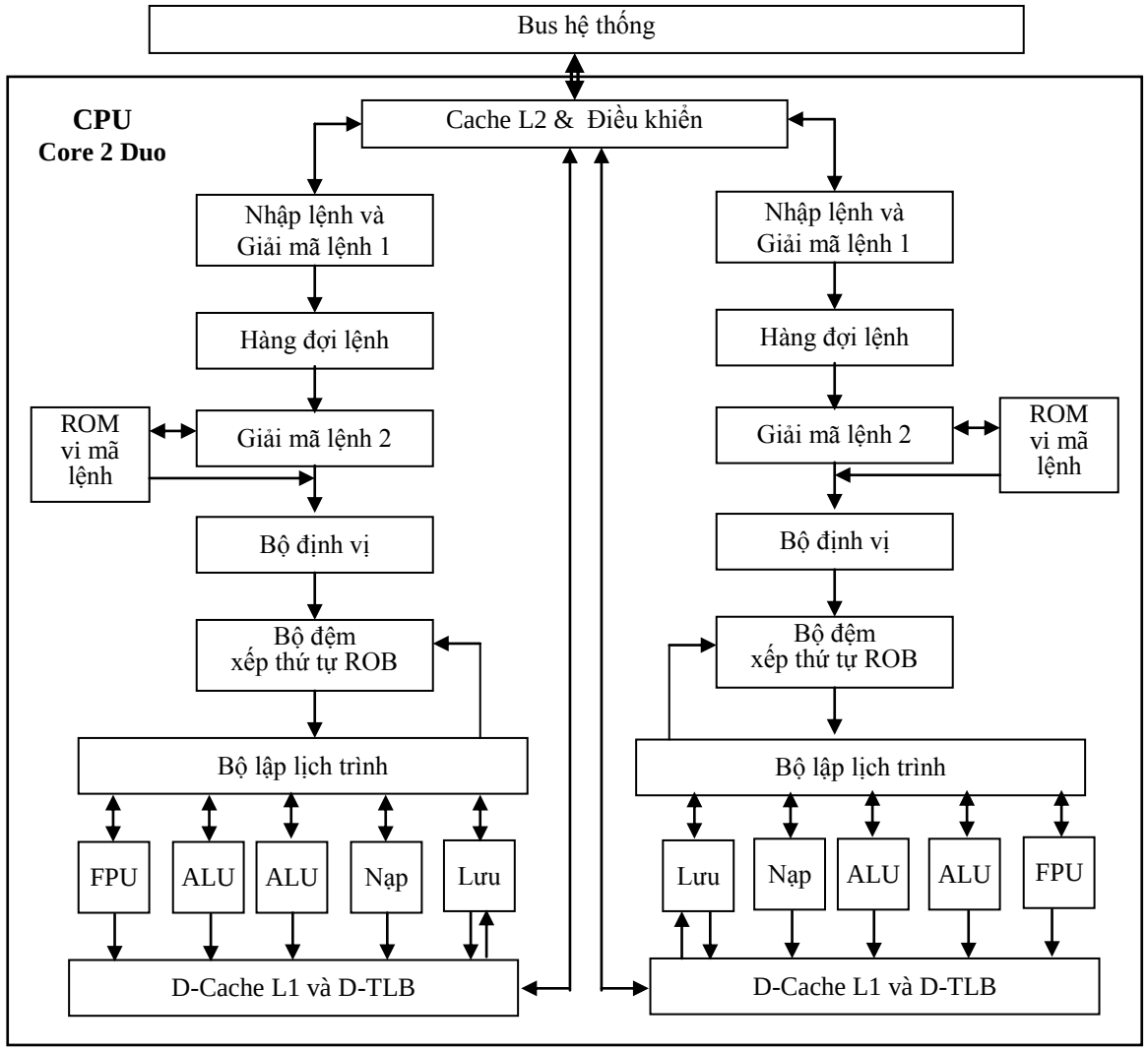
Cơ chế phục vụ ngắt (và tương tự là ngoại lệ) đã được trình bầy ở chương 5, mục 5.2.2. Địa chỉ bắt đầu chương trình phục vụ ngắt được gọi là *véc tơ ngắt*. Mỗi véc tơ ngắt dài 8 byte. Các véc tơ ngắt được chứa trong Bảng véc tơ ngắt trong bộ nhớ chính. CPU xác định véc tơ ngắt *qua số ngắt 8 bit*. Số ngắt được cung cấp cho CPU bằng nhiều con đường khác nhau: ngoại lệ cung cấp số ngắt từ bên trong CPU, thiết bị điều khiển ngắt (gây ngắt cứng) cung cấp số ngắt từ bên ngoài CPU, hoặc lệnh INT n cung cấp số ngắt n (ngắt mềm).

Pentium và các CPU dòng Intel x86 có hai đầu nhận tín hiệu báo ngắt từ ngoài (tín hiệu ngắt cứng) là: INTR- Interrupt Request (loại có thể che chắn) và NMI-Nonmaskable Interrupt (loại không che chắn). Có thể lập che chắn để CPU không nhận tín hiệu báo ngắt INTR hoặc bỏ che chắn thông qua việc xác lập giá trị của bit cờ ngắt IF trong thanh ghi cờ EFLAGS.

Dòng vi xử lý Intel x86 có thể phục vụ đến 256 ngắt và ngoại lệ. CPU Intel x86 dành riêng 32 ngắt và ngoại lệ cho hệ thống, 224 ngắt còn lại là cho người sử dụng.

6.3. KIẾN TRÚC CORE CỦA INTEL VÀ BỘ VI XỬ LÝ CORE 2 DUO

Kiến trúc Core của Intel xuất phát từ kiến trúc NetBurst của CPU Pentium 4. Để nâng cao hiệu năng làm việc của CPU, Intel không giải quyết theo cách nâng cao tốc độ xung nhịp mà bằng tổ chức hai hệ thống xử lý song song và điều khiển hoạt động của các hệ thống đó. Kiến trúc Core được áp dụng để xây dựng các CPU như Core 2 Duo (Hình 73) và Core 2 Extreme, những CPU hiệu năng cao đang được sử dụng phổ biến trong các máy tính để bàn và máy tính xách tay trên thế giới hiện nay.



Hình 73

Bộ vi xử lý dòng Core có kiến trúc kiểu đường ống, một đường ống 14 giai đoạn. Kiến trúc Core có 4 kênh và có thể thực hiện ít nhất 4 lệnh cùng một lúc. Khối đầu của bộ xử lý nhập lệnh, thực hiện tiền phân tích và tái xây dựng lại chúng. Bộ vi xử lý loại Core có một tập vi mã thao tác riêng. Lệnh

được giải mã, được dịch thành các vi mã thao tác (micro-ops), trong đó một số được tổ hợp lại thành các thao tác đơn giản. Ở những nơi có thể, bộ xử lý gộp hai vi thao tác lại với nhau và coi nó như là một. Cách xử lý này làm giảm các bước xử lý đi được khoảng 10 phần trăm trong mỗi trường hợp. Ở giai đoạn cuối cùng của quá trình giải mã, bộ định vị (Allocator) đưa các bit cờ trạng thái vào vi mã thao tác nếu cần. Các lệnh sau giải mã được gửi đến bộ đệm xếp thứ tự ROB (Reorder Buffer) trước khi đi vào bộ lập lịch trình (Schedulers). ROB giống như một cuốn nhật ký, ghi lại tất cả các thông tin cốt yếu về mỗi lệnh. Chức năng ban đầu của ROB là đảm bảo để các lệnh được thực thi theo cách tối ưu nhất, ngay cả khi không đúng theo trật tự lệnh trong chương trình và sắp xếp lại để đảm bảo các lệnh được kết thúc theo đúng như trình tự lệnh trong chương trình. Các kết quả thực hiện lệnh được ghi vào tập thanh ghi CPU theo một trình tự thích hợp. Bộ lập lịch trình điều khiển đơn vị thực hiện lệnh. Đơn vị thực hiện lệnh gồm hai ALU và một FPU, thực hiện được hai lệnh số nguyên và một lệnh dấu chấm động đồng thời.

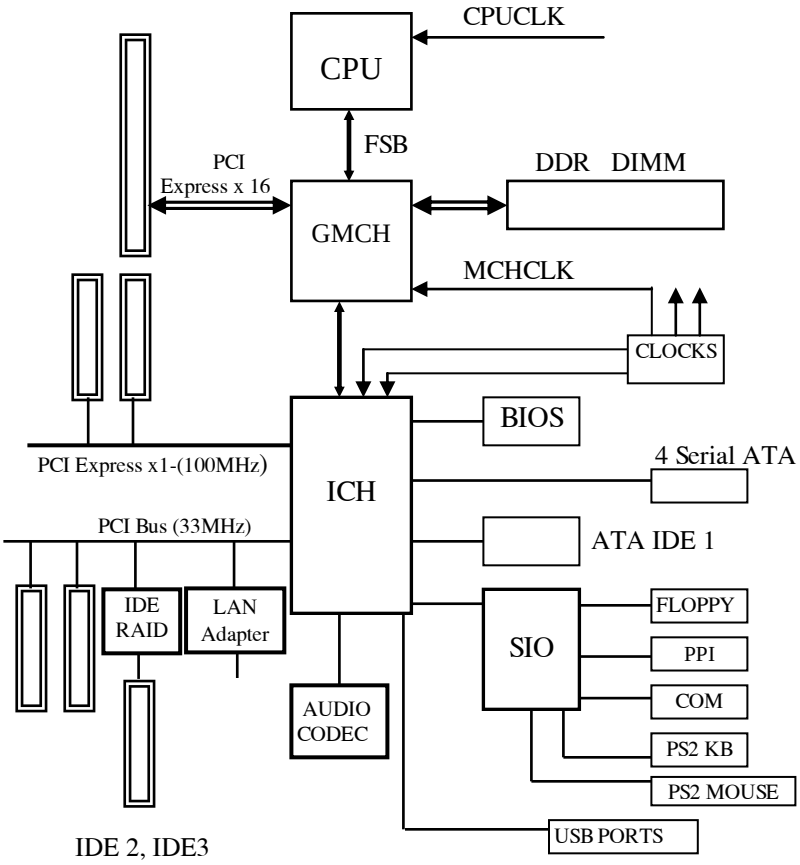
Core 2 Duo có hai nhân và các nhân làm việc độc lập với nhau. Quá trình thực hiện lệnh nói ở trên xảy ra riêng rẽ trong từng nhân riêng biệt. Sự khác biệt của Core 2 Duo so với các CPU dòng Core trước đó là hai nhân cùng chia sẻ bộ cache mức 2 (L2) và theo một cách tối ưu. Khi cả hai nhân cùng thao tác trên một vùng nhớ, ngay lập tức bản sao của dữ liệu ở đó được đưa vào cache. Hiệu suất thực hiện lệnh nhờ đó được tăng lên. Cache L2 được định vị động, phù hợp với yêu cầu sử dụng của từng nhân, do vậy dung lượng cache được sử dụng tối ưu.

Một đặc tính quan trọng khác của vi xử lý dòng Core, mà một trong số đó là Core 2 Duo, là có một bộ quản lý năng lượng động. Bộ quản lý năng lượng động giám sát để khu vực nào trong chip không được sử dụng ở bất kỳ điểm nào đều sẽ được tách khỏi nguồn nuôi, khu vực nào không làm việc hết năng suất thì được chuyển sang chế độ năng lượng thấp. Quản lý năng lượng động cũng được áp dụng cho cả các bus và cache. Nhờ vậy mà năng lượng tiêu thụ của chip được giảm xuống đáng kể, trong khi hiệu năng làm việc của CPU vẫn giữ nguyên.

6.4. SƠ ĐỒ MỘT MÁY VI TÍNH PC HIỆN ĐẠI

Hình 74 là sơ đồ khối điển hình của một máy vi tính PC hiện đại. Các CPU thông dụng có trên máy PC hiện đại ngày nay thường thuộc loại 32 bit (dòng Pentium) với tần số xung nhịp từ 1, 2 GHz đến 3, 8 GHz, hoặc loại 64 bit (dòng Core) với tần số xung nhịp từ 1, 06 GHz đến 3, 3 GHz. Máy tính PC hiện đại được xây dựng trên cơ sở bộ vi xử lý CPU, các chipset (loại mạch tích hợp chức năng cỡ lớn), bộ nhớ, các đường bus tốc độ cao và các thiết bị

vào ra. Ở giai đoạn đầu của ngành công nghiệp chế tạo máy vi tính, các mạch điện tử tạo nên các khối chức năng của máy tính được chế tạo dưới dạng các con chip đơn lẻ. Ngày nay các mạch chức năng đó được tích hợp lại trong một vài con chip điện tử tích hợp cỡ lớn gọi là chipset. Các loại chipset gồm chipset cầu bắc, chipset cầu nam, chipset siêu vào-ra và một vài chipset chức năng riêng khác. Chipset cầu bắc kết nối CPU với các thiết bị tốc độ rất cao như bộ nhớ RAM, các bộ điều khiển đồ hoạ và với chipset cầu nam. Chipset cầu nam kết nối với các bus ngoại vi tốc độ thấp hơn và với chipset siêu vào-ra. Các chipset được thiết kế và chế tạo để phù hợp tối ưu với các bộ vi xử lý và các bo mạch chủ cụ thể.



Hình 74

Chipset cầu bắc GMCH (Graphics and Memory Control Hub) kết nối với CPU qua bus FSB tốc độ từ 400 MHz đến 1600 MHz. Chipset GMCH điều khiển bộ nhớ DRAM và giao diện đồ họa PCI Express x16. GMCH có cổng giao diện với chipset cầu nam ICH.

Chipset cầu nam ICH (Input-output Control Hub) thường chứa các bộ điều khiển và giao diện sau:

- Giao diện GMCH
- Bộ điều khiển ngắt PIC 8259, 16 kênh IRQ
- Bộ điều khiển DMAC 8237
- Kết nối ROM BIOS
- Giao diện cho hai loại thiết bị đĩa cứng chuẩn PATA và SATA
- Giao diện PCI (Peripheral Component Interconnect) bus. Bus PCI 32 bit, với tần số nhịp đồng bộ bus 33, 33 MHz, tốc độ truyền dữ liệu đạt đến (33. 33 MHz x 32 bits)/ 8 bits = 133 MB/s
- Giao diện PCI Express x1 bus. Tần số nhịp đồng bộ bus là 100 MHz
- Bộ điều khiển chủ USB và HUB gốc
- Hỗ trợ kết nối với mạch CODEC (Coder-Decoder) xử lý tín hiệu số-giao diện âm thanh chất lượng cao.
- Hỗ trợ kết nối với mạch giao diện mạng LAN

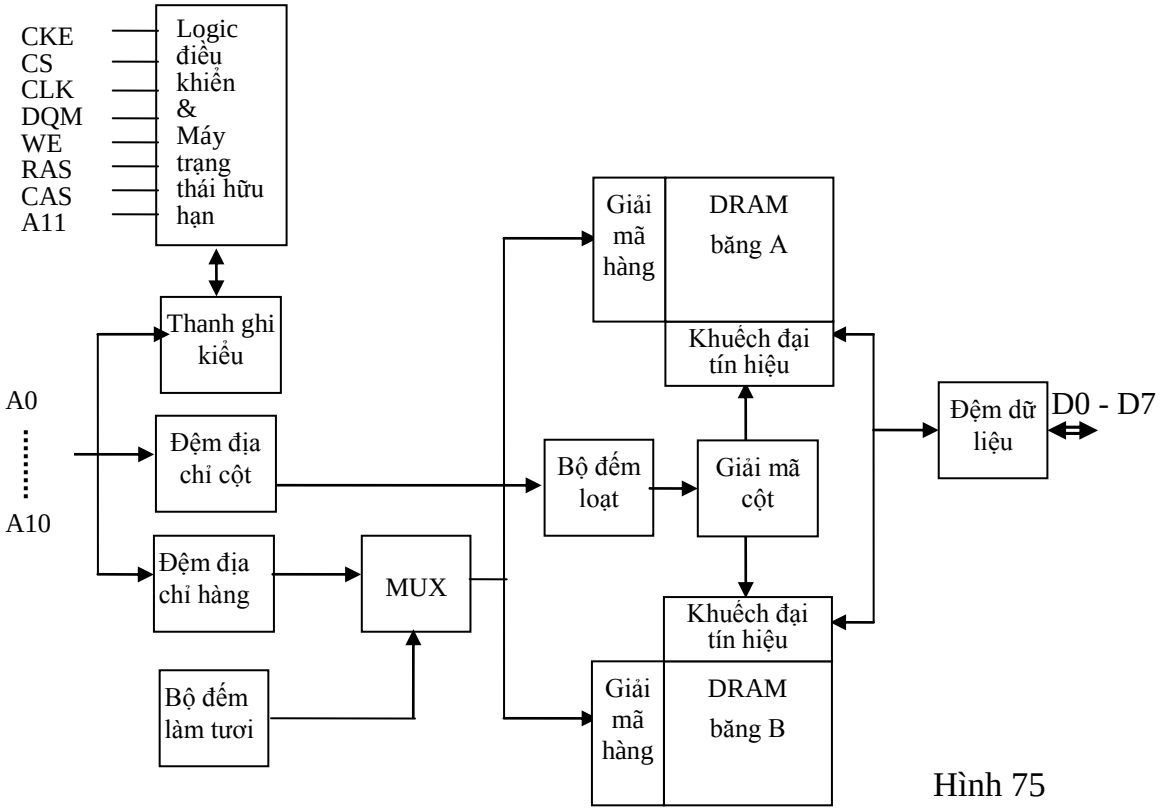
Chipset siêu vào-ra (Super IO) chứa các mạch giao diện vào-ra dữ liệu cơ bản nhất:

- Giao diện vào-ra nối tiếp chuẩn UART (2xCOM port)
- Giao diện song song chuẩn (LPT port)
- Giao diện với bàn phím và chuột chuẩn PS/2
- Giao diện điều khiển đĩa mềm FDC
- Đồng hồ thời gian thực RTC
- Thiết bị quản lý nguồn điện

Chương 7
CÁC TỔ CHỨC DRAM TIÊN TIẾN

7.1. BỘ NHỚ SDRAM

SDRAM (Synchronous Dynamic Random Access Memory) là DRAM, nhưng quá trình trao đổi dữ liệu với CPU được đồng bộ hóa bởi xung nhịp ngoài và hoạt động với tốc độ của bus, không có trạng thái đợi (Hình 75).



Hình 75

SDRAM có kiến trúc băng đôi tạo khả năng truy nhập song song. SDRAM sử dụng chế độ *burst* để hạn chế thời gian xác lập địa chỉ hàng và cột sau lần truy nhập đầu tiên. Ở chế độ burst, một chuỗi các bit có thể được đưa ra ngay sau khi bit đầu tiên được truy cập.

7.2. BỘ NHỚ DDR SDRAM

DDR SDRAM là SDRAM, nhưng làm việc với tốc độ dữ liệu gấp đôi (DDR-Double Data Rate). DDR SDRAM sử dụng cùng một tín hiệu điều khiển như SDRAM cho một chu kỳ, nhưng đọc hoặc ghi hai từ dữ liệu trong một chu kỳ nhịp, trong đó việc truyền dữ liệu được thực hiện ở sườn lên và cả sườn xuống của tín hiệu xung nhịp. Với dữ liệu 64 bit trong một lần truyền, DDR SDRAM có tốc độ truyền là:

[(tần số xung nhịp bộ nhớ) x 2 (tốc độ gấp đôi) x 64 (số bit trong một lần truyền)]/8 [Byte/giây]

Ví dụ, nếu tần số xung nhịp bộ nhớ là 100 MHz thì DDR SDRAM cho tốc độ truyền tối đa là 1600 MByte/giây.

DDR SDRAM sử dụng nguồn nuôi điện thế 2, 5 V. Tốc độ xung nhịp điển hình của DDR là 133, 166 hoặc 200 MHz (chu kỳ nhịp là 7, 5 ns, 6ns và 5 ns), nhưng thường được mô tả như là DDR 266, DDR 333 và DDR 400. DDR SDRAM sử dụng chân cắm kiểu DIMM 184-chân, được biết với các tên PC-2100 (2100 MByte/giây), PC-2700 (2700 MByte/giây) và PC-3200 (3200 MByte/giây)

7.3. BỘ NHỚ DDR2 SDRAM

Bộ nhớ DDR2 SDRAM cũng là bộ nhớ SDRAM có tốc độ truyền gấp đôi, nhưng nó không dùng các đặc tả kỹ thuật nguyên thủy của DDR và *không tương thích với DDR SDRAM*. Ngoài việc tăng gấp đôi tốc độ bus, bằng sử dụng cả sườn lên và sườn xuống của xung nhịp, DDR2 tận dụng bộ đệm dữ liệu vào-ra trong bộ nhớ và bus dữ liệu để có thể hoạt động với tốc độ gấp đôi tốc độ xung nhịp bus. Sử dụng kết hợp cả hai yếu tố này DDR2 truyền được tới 4 dữ liệu trong một chu kỳ bộ nhớ. Với dữ liệu 64 bit trong một lần truyền, DDR2 SDRAM có tốc độ truyền là:

[(tần số xung nhịp bộ nhớ) x 2(nhân đôi tần số bus)x 2 (tốc độ gấp đôi) x 64 (số bit trong một lần truyền)]/8 [Byte/giây]

Ví dụ, nếu tần số xung nhịp bộ nhớ là 100 MHz thì DDR2 SDRAM cho tốc độ truyền tối đa là 3200 MByte/giây.

DDR2 SDRAM sử dụng nguồn nuôi điện thế 1, 8 V. Tốc độ xung nhịp điển hình của DDR2 là 200, 266, 333 hoặc 400 MHz, nhưng thường được mô tả như là DDR2 400, DDR2 533 và DDR2 667 và DDR2 800. DDR2 SDRAM sử dụng chân cắm kiểu DIMM 240-chân, được biết với các tên từ PC2-3200 (3200 Mbyte/giây) đến PC2-8500 (8533 MByte/giây).

7.4. BỘ NHỚ DDR3 SDRAM

Bộ nhớ DDR3 SDRAM là bộ nhớ SDRAM, nhưng không tương thích với DDR2. DDR3 SDRAM có tốc độ truyền gấp hai lần DDR2. Tốc độ truyền của DDR3 SDRAM với dữ liệu 64 bit trong một lần truyền là:

[(tần số xung nhịp bộ nhớ) x 4 (nhân bốn lần tần số bus)x 2 (tốc độ gấp đôi) x 64 (số bit trong một lần truyền)]/8 [Byte/giây]

Ví dụ, nếu tần số xung nhịp bộ nhớ là 100 MHz thì DDR3 SDRAM cho tốc độ truyền tối đa là 6400 MByte/giây.

DDR3 SDRAM sử dụng nguồn nuôi điện thế 1, 5 V. Tốc độ xung nhịp điển hình của DDR3 là 400, 533, 667 hoặc 800 MHz. DDR3 SDRAM sử dụng chân cắm kiểu DIMM 240-chân, giống như DDR2, nhưng không tương thích về điện và chỗ vết khía. DDR3 có các tên gọi từ PC3-6400 (6400 Mbyte/giây) đến PC3-12800 (12800 MByte/giây).

DDR4 SDRAM sẽ kế tục DDR3 SDRAMvà vẫn còn đang được thiết kế. Dự tính DDR4 được nuôi với nguồn 1, 2 V. Tốc độ xung nhịp dự tính đạt đến 1600 MHz.

Chương 8
CÁC THIẾT BỊ ĐIỀU KHIỂN VÀ GIAO DIỆN VÀO-RA DỮ LIỆU CHUẨN

8.1. NGẮT VÀ BỘ ĐIỀU KHIỂN NGẮT PIC 8259

8.1.1. Phân loại ngắt

Ngắt là sự kiện đơn vị xử lý trung tâm CPU bị tạm dừng việc thực hiện quá trình hiện hành và chuyển sang thực hiện quá trình phục vụ ngắt. Ngắt cứng là phương pháp vào-ra dữ liệu, trong đó thiết bị vào-ra (bàn phím, máy in, đồng hồ nhịp thời gian v.v.) chủ động khởi động quá trình vào-ra. Đơn vị xử lý trung tâm được thiết kế để nhận tín hiệu báo ngắt INT (interrupt) từ bên ngoài và phản ứng theo cơ chế phục vụ ngắt.

Thuật ngữ "ngắt" (interrupt) xuất phát từ kỹ thuật ngắt cứng. Khi nói đến ngắt cứng, ngắt mềm hoặc ngắt logic (hoặc ngoại lệ) là hàm ý nói đến các chương trình con phục vụ hoạt động của hệ thống máy tính và nói đến cách kích hoạt các chương trình con này. Tất cả các chương trình phục vụ ngắt đều có chung đặc điểm: thứ nhất là hầu hết đã được viết sẵn (là các chương trình của hệ điều hành) và được phép sử dụng; thứ hai là địa chỉ của các chương trình con này phải được đặt ở một vùng xác định là *Bảng véc tơ ngắt*, nằm trong bộ nhớ chính. Các chương trình con phục vụ *ngắt cứng* thường được dùng để điều khiển quá trình vào-ra ở mức vật lý với các thiết bị vào-ra chuẩn. Các chương trình con phục vụ ngắt cứng được kích hoạt bởi các *tín hiệu vật lý IRQ* đến từ thiết bị vào-ra. Các chương trình con phục vụ *ngắt mềm* là các chương trình hệ thống thực hiện các thao tác vào-ra cơ bản ở *mức logic* và các hoạt động khác của hệ thống. Các chương trình con phục vụ ngắt mềm được kích hoạt bởi *lệnh INT n* trong hệ lệnh của CPU. Các chương trình con phục vụ ngắt logic (phục vụ ngoại lệ) cũng phục vụ cho hoạt động của hệ thống, nhưng chúng chỉ được kích hoạt khi CPU hoạt động hoặc thực hiện lệnh và phát sinh một *ngoại lệ* nào đó cần xử lý.

- Bảng véc tơ ngắt

Bảng véc tơ ngắt là bảng chứa địa chỉ của các chương trình phục vụ ngắt. Bảng này có 256 ô, các ô được đánh số thứ tự lần lượt từ 00H, 01H,.., 08H,.., 0FH, 10H,.., FFH. Số thứ tự của từng ô trong bảng được gọi là *số ngắt*. Mỗi ô chứa địa chỉ logic của một chương trình phục vụ ngắt xác định, các địa chỉ này còn được gọi là *véc tơ ngắt*.

Cấu trúc của Bảng véc tơ ngắt ở chế độ thực (Bảng 10).

Bảng 10

	Số t.t.	Số ngắt	Chức năng của chương trình phục vụ ngắt
Địa chỉ đoạn Địa chỉ offset	00H	00H	Xử lý chia cho 0
Địa chỉ đoạn Địa chỉ offset	01H	01H	Thực hiện gỡ rối (debug)
⋮			⋮
Địa chỉ đoạn Địa chỉ offset	08H	08H	Đồng hồ hệ thống
Địa chỉ đoạn Địa chỉ offset	09H	09H	Phục vụ bàn phím
⋮	⋮	⋮	⋮
Địa chỉ đoạn Địa chỉ offset	21H	21H	Thực hiện các dịch vụ của hệ điều hành
⋮	⋮	⋮	⋮
Địa chỉ đoạn Địa chỉ offset	FFH	FFH	Dự phòng

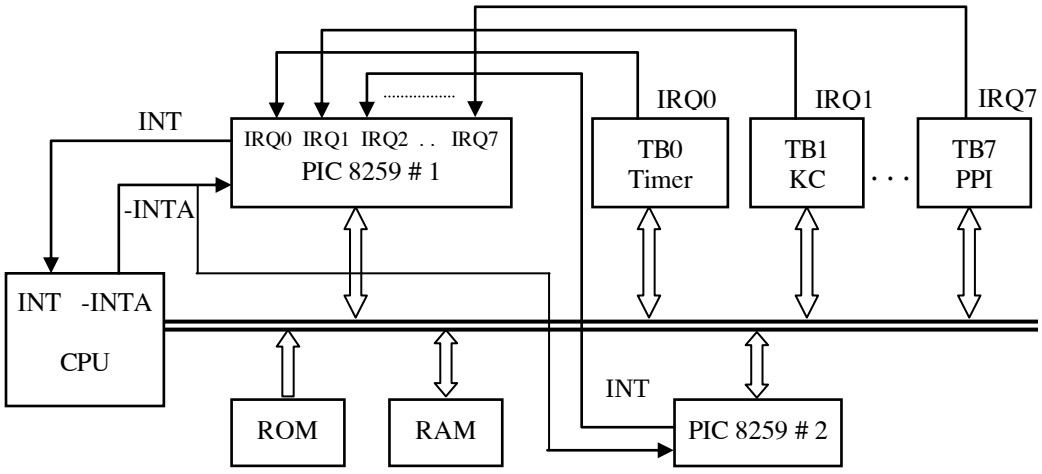
8.1.2. Hệ thống ngắt cứng trong máy tính PC

CPU được thiết kế để đáp ứng được với các quá trình phục ngắt cứng. CPU có một đầu vào nhận tín hiệu ngắt INT, khi nhận được tín hiệu này CPU sẽ phản ứng theo cơ chế ngắt cứng. Trong thực tế có nhiều thiết bị ngoại vi yêu cầu được phục vụ theo phương pháp ngắt cứng (bàn phím, đồng hồ hệ thống, máy in, v.v.) và sinh ra nhiều yêu cầu ngắt, do vậy cần có một bộ điều khiển giúp CPU quản lý và phục vụ các yêu cầu ngắt, đó là bộ điều khiển ngắt PIC (Programmable Interrupt Controller).

PIC 8259 là một mạch điện tử tích hợp khả trình được thiết kế để giúp CPU phục vụ quá trình ngắt cứng. PIC 8259 thực hiện các chức năng sau:

+ Ghi nhận được 8 yêu cầu ngắt IRQ_i, i=0, 1, ..., 7.
+ Cho phép chọn và phục vụ các yêu cầu ngắt theo mức ưu tiên.
+ Cung cấp cho CPU số ngắt tương ứng với yêu cầu ngắt IRQ_i. Số ngắt là con số đại diện cho địa chỉ của chương trình con phục vụ thiết bị đưa ra yêu cầu ngắt IRQ_i. Từ số ngắt CPU đến bảng véc tơ ngắt để nhận được địa chỉ chương trình phục vụ ngắt tương ứng.
+ Cho phép hoặc không cho phép các yêu cầu ngắt IRQ_i kích hoạt hệ thống ngắt.

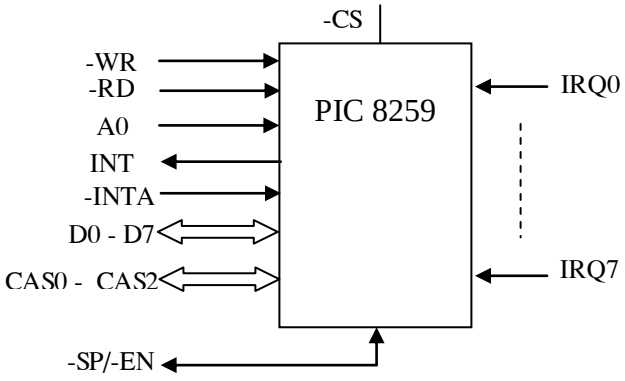
Hệ thống ngắt cứng của máy tính PC được xây dựng trên cơ sở 2 bộ điều khiển ngắt PIC 8259, mỗi PIC 8259 có thể nhận 8 tín hiệu yêu cầu ngắt IRQ_i, i=0, 1,...., 7, từ thiết bị vào-ra. Hai PIC này được kết nối với nhau theo kiểu ghép tầng, kết hợp hoạt động để có thể phục vụ được 16 yêu cầu ngắt. Cấu trúc của hệ thống ngắt cứng trong máy tính PC có dạng sau (Hình 76).



Hình 76

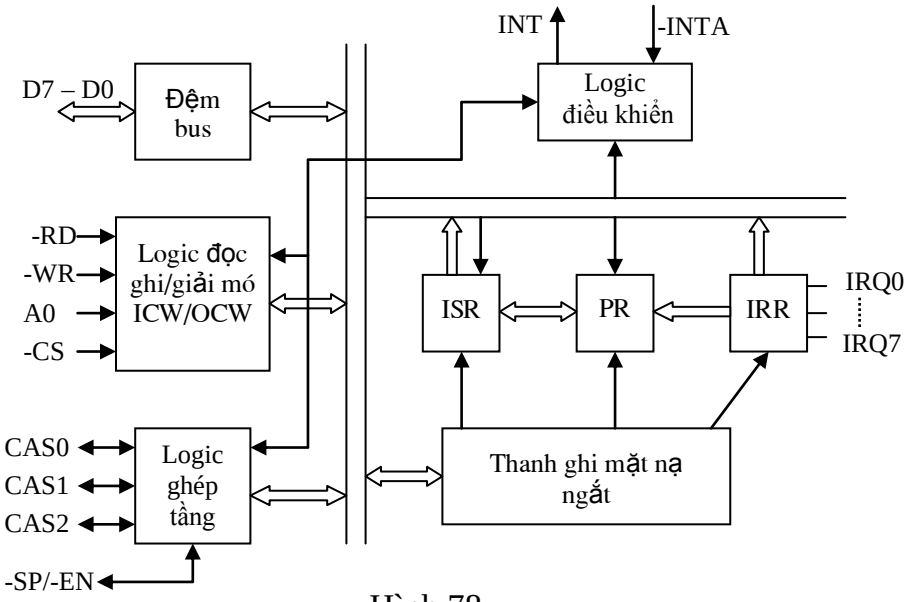
8.1.3. Bộ điều khiển ngắt khả trình PIC 8259 và cơ chế phục vụ ngắt cứng

Cấu trúc bên ngoài của PIC 8259 (Hình 77).



Hình 77

Hình 78 mô tả cấu trúc bên trong của PIC 8259. PIC 8259 có các thành phần chức năng sau:



Hình 78

+ Thanh ghi yêu cầu ngắt IRR (Interrupt Request Register): là thanh ghi 8 bit, IRR chứa (ghi nhận) tất cả các yêu cầu ngắt IRQ_i đòi phục vụ. Nếu tín hiệu IRQ_i = "1" thì bit IRR_i tương ứng được đặt bằng "1".

+ Bộ giải quyết ưu tiên PR (Priority Resolver): là thanh ghi 8 bit, PR xác định mức ưu tiên của các yêu cầu ngắt. Yêu cầu ngắt có ưu tiên cao nhất trong số các IRQ_i cùng xuất hiện sẽ được phục vụ và được ghi nhớ bằng bit tương ứng trong ISR ở chu kỳ INTA.

+ Thanh ghi ngắt đang được phục vụ ISR (In Service Register): là thanh ghi 8 bit, ISR ghi nhận các ngắt đang được phục vụ. Yêu cầu ngắt IRQ_i nào đang được phục vụ thì bit ISR_i tương ứng được đặt bằng "1".

+ Khối logic điều khiển: khối logic điều khiển đưa ra tín hiệu INT, được nối thẳng với chân nhận tín hiệu báo ngắt INT của CPU. Khối logic điều khiển nhận tín hiệu INTA từ CPU. Khi nhận được tín hiệu INTA, PIC 8259 sẽ cung cấp số ngắt ra bus dữ liệu cho CPU.

+ Khối đệm bus: là loại 8 bit, 2 hướng, 3 trạng thái. Các từ điều khiển ICW, OCW được đưa vào PIC 8259 qua khối này để xác lập chế độ hoạt động của 8259. Số ngắt và trạng thái hoạt động của PIC cũng được đưa ra bus dữ liệu qua khối này.

+ Khối ghép tầng: PIC 8259 có cơ cấu cho phép nối ghép tầng các PIC 8259 với nhau và phối hợp hoạt động của các PIC này. Tầng thứ nhất có đầu ra INT nối trực tiếp với CPU, gọi là *PIC 8259-chủ*. Đầu vào IRQ_i của PIC chủ được nối với đầu ra INT của PIC 8259 thứ hai. PIC này được gọi là *PIC*

8259-thợ. Cơ chế ghép tầng cho phép xây dựng một hệ thống ngắt cứng quản lý được đến 64 yêu cầu ngắt IRQ.

+ Khối logic ghi/đọc và giải mã: thực hiện giải mã các từ điều khiển khởi động ICW(Initialization Command Word) và từ điều khiển hoạt động OCW (Operation Command Word). Qua hai loại từ điều khiển này người sử dụng có thể lập trình xác lập chế độ hoạt động cho PIC.

+ Thanh ghi IMR: là thanh ghi 8 bit, cho phép đặt/xóa mặt nạ ngắt.

+ Bảng các tín hiệu CS, A0, RD, WR và cách ghi/đọc PIC 8259 (Bảng 11).

Bảng 11

CS	A0	RD	WR	D4	D3	Hướng thông tin
0	0	0	1	X	X	IRR, ISR => BUS
0	1	0	1	X	X	0CW1 => BUS
0	0	1	0	0	0	BUS => 0CW2
0	0	1	0	0	1	BUS => 0CW3
0	0	1	0	1	X	BUS => ICW1
0	1	1	0	X	X	BUS => ICW2, ICW3, ICW4, 0CW1

a) Cơ chế phục vụ ngắt cứng

Điều kiện ban đầu: PIC 8259 cần được lập trình khởi động qua các từ điều khển ICW. Sau khi các từ điều khiển ICW được nạp thì PIC 8259 sẵn sàng hoạt động.

- Một hoặc nhiều thiết bị vào-ra có yêu cầu được phục vụ phát tín hiệu IRQ_i = "1" (mức tích cực) cho PIC. PIC ghi nhận các yêu cầu ngắt IRQ_i này bằng cách đặt các bit IRR_i tương ứng lên "1".
- PIC 8259 chọn IRQ_i có mức ưu tiên cao nhất trong số các yêu cầu để phục vụ. PIC gửi tín hiệu INT cho CPU, đòi CPU phục vụ.
- CPU thực hiện các thao tác sau:
    + Thực hiện nốt lệnh của quá trình hiện hành.
    + Lưu địa chỉ trở về (nội dung của các thanh ghi CS, IP) và thanh ghi cờ EFLAGS vào ngăn xếp.
    + Gửi hai tín hiệu trả lời ngắt INTA cho PIC.
- Khi PIC 8259 nhận được tín hiệu INTA thứ 1: bit ISR_i ứng với IRQ_i có mức ưu tiên cao nhất được thiết lập (ISR_i=1) và bit IRR_i tương ứng bị xóa (IRR_i=0). Trong chu kỳ INTA thứ nhất này PIC 8259 không gửi gì cho CPU qua bus dữ liệu.
- Khi PIC 8259 nhận được tín hiệu INTA thứ 2: PIC 8259 gửi số ngắt tương

ứng với IRQ_i đang được phục vụ qua bus dữ liệu cho CPU.

- CPU nhận số ngắt và trên cơ sở số ngắt này vào vị trí tương ứng trong Bảng véc tơ ngắt để xác định địa chỉ của chương trình phục vụ ngắt. CPU nạp địa chỉ chương trình phục vụ ngắt vào các thanh ghi CS và EIP và bắt đầu thực hiện chương trình phục vụ ngắt này.
- Khi thực hiện xong chương trình phục vụ ngắt (khi CPU thực hiện lệnh IRET) thì quá trình phục vụ ngắt của CPU cũng kết thúc. CPU khôi phục địa chỉ trở về vào các thanh ghi CS, EIP, khôi phục nội dung thanh ghi EFLAGS và tiếp tục thực hiện quá trình vừa bị ngắt.

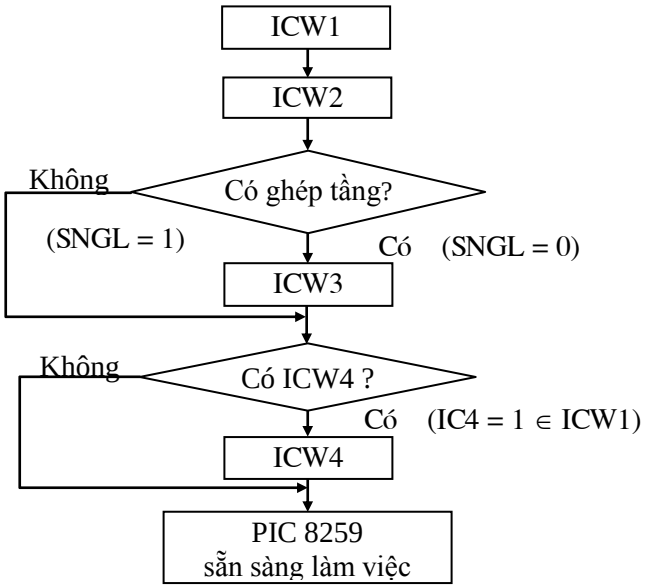
Hệ thống ngắt cứng có thể kết thúc phục vụ ngắt hiện thời theo hai chế độ:

+ Kết thúc ngắt bình thường EOI (End Of Interrupt): khi PIC được đặt chế độ kết thúc ngắt bình thường EOI thì CPU phải phát lệnh báo kết thúc ngắt EOI (qua từ điều khiển OCW2) cho PIC trước khi rời khỏi chương trình con phục vụ ngắt. Khi đó bit ISR_i của ngắt đang được phục vụ sẽ được đặt xuống 0.
+ Kết thúc ngắt tự động AEOI (Automatic EOI): khi PIC được đặt chế độ kết thúc ngắt tự động AEOI thì tại chu kỳ INTA thứ 2 bit ISR_i của ngắt đang được phục vụ sẽ được đặt xuống 0.

Bằng các cách nói trên hệ thống ngắt cứng có thể tiếp tục phục vụ yêu cầu ngắt này ở những lần tiếp theo.

b) Lập trình khởi động PIC 8259 và các từ điều khiển khởi động ICW

Cần xác lập chế độ làm việc của PIC 8259 trước khi sử dụng. Quá trình này được gọi là lập trình khởi động thiết bị. Việc lập trình khởi động PIC 8259 được thực hiện qua các từ điều khiển ICW và theo lưu đồ sau (Hình 79):



Hình 79

- ICW1:

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	1	LTIM	ADI	SNGL	IC4

Các bit D5 – D7 không dùng cho CPU x86.

+ IC4 (bit D0): Cho biết có cần ICW4 ?
  IC4 = 0 : không cần ICW4.
  IC4 = 1 : có ICW4.
+ SNGL (bit D1): cho biết hệ thống ngắt chỉ có một PIC hay có nhiều PIC ghép tầng.
  SNGL = 0 có ghép tầng
  SNGL = 1 chỉ có một PIC 8259
+ ADI (bit D2): không dùng cho hệ CPU x86
+ LTIM: xác định dạng tín hiệu IRQ
  LTIM = 1 IRQ phải là tín hiệu mức TTL
  LTIM = 0 IRQ phải là tín hiệu dạng sườn xung.
+ D4 = 1
+ D5 = D6 = D7 = 0

- ICW2:

ICW2 định nghĩa số ngắt nền cho 7 số ngắt còn lại.

D7	D6	D5	D4	D3	D2	D1	D0
T7	T6	T5	T4	T3	x	x	x

+ Các bit T7 - T3 là 5 bit cao của số ngắt, 3 bit còn lại liên quan đến các đầu vào IRQ_i
+ Năm bit cao T7 - T3 (do người sử dụng tùy chọn) cùng với 3 bit thấp nhất bằng 0 xác định số ngắt nền. Dựa trên số ngắt nền ứng với IRQ_0 này, PIC 8259 tự tạo ra các số ngắt tiếp theo tương ứng với các IRQ_1 đến IRQ_7.

Ví dụ: ở hệ thống ngắt cứng của máy vi tính PC, các số ngắt do PIC 8259-chủ cung cấp như sau:

0	0	0	0	1	0	0	0	ứng với IRQ 0
0	0	0	0	1	0	0	1	ứng với IRQ 1
.....								.....
0	0	0	0	1	1	1	1	ứng với IRQ 7

- ICW3: liên quan đến ghép tầng.

Mạch phần cứng có chân SP/EN xác định chủ/thợ ở chế độ ghép tầng: nếu SP = 1 thì PIC là chủ, nếu SP = 0 thì PIC là thợ.

Có hai loại ICW3

- ICW3 cho PIC chủ: xác định đầu vào IRQ_i nhận tín hiệu INT từ PIC thợ thứ i.

D7	D6	D5	D4	D3	D2	D1	D0
S7	S6	S5	S4	S3	S2	S1	S0

Nếu Si = 1 báo có PIC thợ nối vào chân IRQ_i của chủ

- ICW3 cho PIC thợ: xác định địa chỉ (chỉ thị nhận dạng) của PIC thợ.

0	0	0	0	0	ID2	ID1	ID0

Các bit ID2, ID1, ID0 xác định địa chỉ riêng của các PIC 8259-thợ. Khi nhận được tín hiệu INTA2, PIC 8259-thợ so sánh các tín hiệu CAS0 - CAS2 (phát ra từ PIC 8259-chủ) với ID2 - ID0, nếu chúng giống nhau thì PIC 8259-thợ gửi số ngắt lên bus dữ liệu cho CPU, ngược lại thì không gửi.

- ICW4:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	SFNM	BUFF	M/S	AEOI	μP

+ bit μp: báo cho PIC 8259 biết phải làm việc với họ vi xử lý nào.
  μp = 1: làm việc với họ x86
  μP = 0: làm việc với họ 8085
+ bit AEOI: xác lập chế độ kết thúc ngắt.
  AEOI = 0 : kết thúc bình thường EOI
  AEOI = 1 : kết thúc tự động AEOI
+ bit BUFF: báo chế độ có bộ đệm bus
  BUFF =1: PIC làm việc ở chế độ đệm bus, lúc này tín hiệu SP/EN ở chế độ ra và việc định nghĩa chủ/thợ được xác định bằng bit M/S.
+ bit M/S: xác định chủ/thợ
  M/S=1: PIC là chủ
  M/S=0: PIC là thợ.
  Nếu BUFF=0 thì M/S không có ý nghĩa.
+ bit SFNM: bit này được đặt bằng 0 ngay khi khởi động hệ thống. Kiểu ưu tiên cố định là mặc định, trong đó IRQ_0 có mức ưu tiên cao nhất, IRQ_7 có mức ưu tiên thấp nhất. Có thể thay đổi kiểu ưu tiên bằng từ điều khiển OCW2. Trong kiểu ưu tiên cố định, khi SFNM=0, khi bit

ISR_i =1 thì tất cả các IRQ_i có mức ưu tiên thấp hơn đều bị cấm. Chỉ có các IRQ_i có mức ưu tiên cao hơn được phép gây ngắt chương trình phục vụ ngắt hiện thời.

Các từ điều khiển hoạt động OCW

Các từ điều khiển OCW được dùng để xác lập các chế độ làm việc cụ thể trong quá trình hoạt động của PIC 8259. Có thể gửi các từ OCW này cho PIC 8259 vào bất kỳ lúc nào sau khi khởi động hệ thống ngắt.

- OCW1: cho phép hoặc cấm nhận một yêu cầu ngắt IRQ_i nào đó bằng mặt nạ ngắt.

  Với PIC chủ: địa chỉ thanh ghi chứa OCW1 là 21H
  Với PIC thợ : địa chỉ thanh ghi chứa OCW1 là A1H

D7	D6	D5	D4	D3	D2	D1	D0
M7	M6	M5	M4	M3	M2	M1	M0

  Mỗi bit Mi tương ứng với IRQ_i

  Khi Mi = 1 mặt nạ ngắt được đặt, cấm PIC nhận IRQ_i (cấm IRQ_i gây ngắt)
  Khi Mi = 0 mặt nạ ngắt được xoá, cho phép PIC nhận IRQ_i (cho phép IRQ_i gây ngắt)
  Hệ điều hành đặt mặt nạ che chắn các IRQ mà hệ thống chưa dùng đến.

- OCW2: dùng để đổi kiểu ưu tiên và báo kết thúc ngắt EOI.

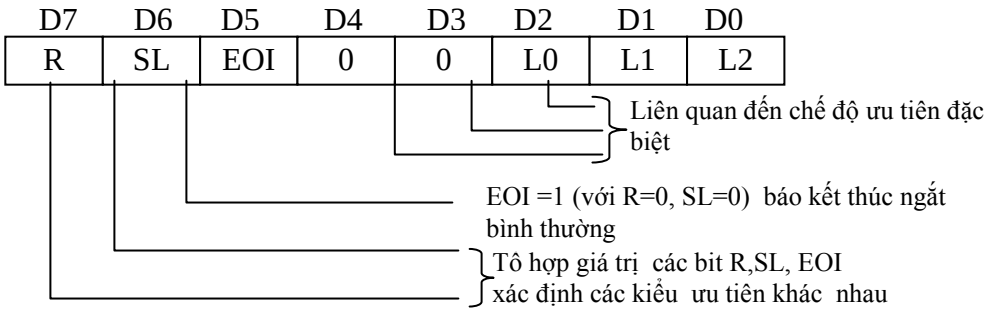
  Địa chỉ thanh ghi chứa OCW2: 20H (PIC chủ), A0H (PIC thợ)

  PIC cho phép chọn một trong ba chế độ ưu tiên:

Ưu tiên cố định: IRQ_0 có mức ưu tiên cao nhất, IRQ_7 có mức ưu tiên thấp nhất. Trong chế độ này IRQ mức cao có quyền ngắt chương trình phục vụ ngắt có mức ưu tiên thấp hơn.

Ưu tiên quay vòng: IRQ_i nào vừa được phục vụ thì bit ISR_i sẽ bị xoá xuống 0 và tự động có mức ưu tiên thấp nhất. Điều này thực tế đã tạo ra các mức ưu tiên bằng nhau.

Ưu tiên đặc biệt: người lập trình có thể thay đổi mức ưu tiên bằng chương trình. Nếu các bit trong OCW2 R=1, SL=1 thì các bit L2-L0 sẽ đặt IRQ_n xuống mức thấp nhất và IRQ_{n+1} lên mức cao nhất.



- OCW3 cho phép đặt/đọc ISR và các thanh ghi khác của PIC 8259.

7	6	5	4	3	2	1	0
1	ESMM	SMM	0	1	P	RR	RISS

  - ESMM =1 và SMM cho phép đặt/xoá chế độ mặt nạ đặc biệt. Chế độ mặt nạ đặc biệt này chỉ cấm một IRQ và cho phép tất cả các IRQ còn lại được yêu cầu ngắt.
  - D4 = 0, D3 = 1
  - Bit P: cho phép PIC 8259 làm việc với CPU ở chế độ hỏi đáp, không cần qua các tín hiệu INT, INTA. Nếu P=1 thì PIC coi tín hiệu điều khiển đọc RD như là tín hiệu INTA.
  - Các bit RR và RIS:

    RR = 1 & RIS = 0: báo sẽ đọc IRR ở lệnh đọc tiếp sau
    RR = 1 & RIS = 1: báo sẽ đọc ISR ở lệnh đọc tiếp sau.

c) ***Phân bố chức năng các yêu cầu ngắt và số ngắt trong máy PC*** (Bảng 12, 13)

- PIC 8259-chủ:

  PIC 8259-chủ chiếm hai địa chỉ cổng: 20H, 21H.

  Bảng 12

IRQ_i	**Số ngắt**	**Thiết bị yêu cầu ngắt**
IRQ_0	08H	Bộ tạo xung nhịp đồng hồ hệ thống
IRQ_1	09H	Module giao diện bàn phím
IRQ_2	0AH	PIC 8259-thợ
IRQ_3	0BH	Module giao diện vào-ra tuần tự 2
IRQ_4	0CH	Module giao diện vào-ra tuần tự 1
IRQ_5	0DH	Dự phòng
IRQ_6	0EH	Module giao diện ổ đĩa mềm FDC
IRQ_7	0FH	Module giao diện vào-ra song song

- PIC 8259-thợ:

  PIC 8259-thợ chiếm hai địa chỉ cổng: A0H, A1H

  Bảng 13

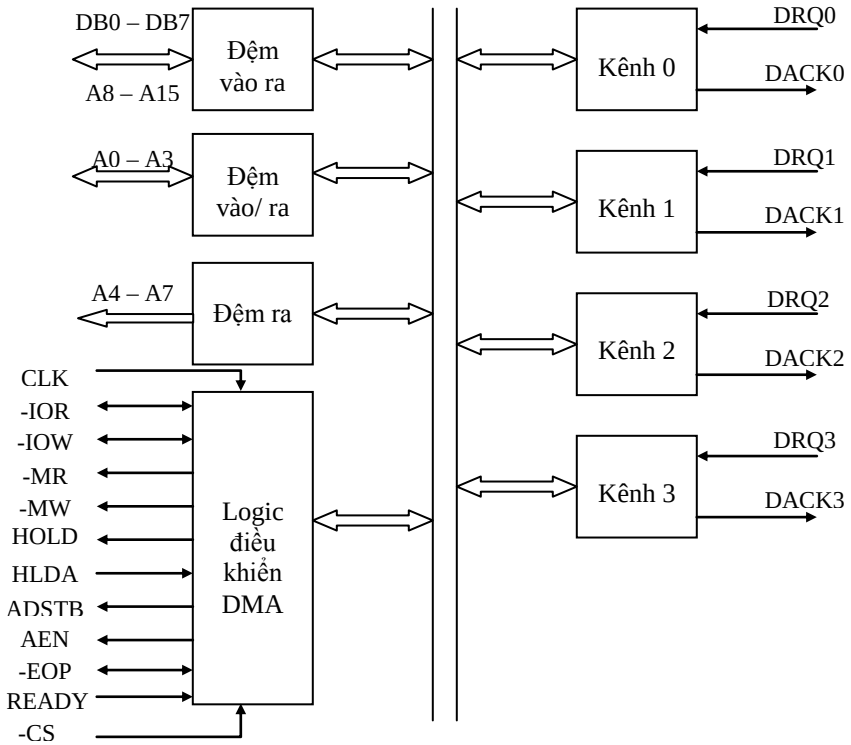
IRQ_i	**Số ngắt**	**Thiết bị yêu cầu ngắt**
IRQ_8	70H	Đồng hồ thời gian thực
IRQ_9	71H	Dự phòng
IRQ_{10}	72H	Card âm thanh

IRQ_{11}	73H	Module giao diện USB
IRQ_{12}	74H	Module giao diện chuột PS/2
IRQ_{13}	75H	Bộ đồng xử lý x87
IRQ_{14}	76H	Bộ điều khiển bus IDE 1
IRQ_{15}	77H	Bộ điều khiển bus IDE 2

8.2. TRUY NHẬP TRỰC TIẾP BỘ NHỚ VÀ BỘ ĐIỀU KHIỂN DMAC 8237

8.2.1. Sơ đồ khối mạch DMAC 8237

Hình 80 mô tả cấu trúc bên trong của DMAC 8237



Hình 80

- DRQ0-DRQ3 (tín hiệu vào): tín hiệu yêu cầu DMA. Bốn dây DRQ nối với 4 thiết bị cần phục vụ kiểu DMA. Khi thiết bị ngoại vi đưa tín hiệu DRQ lên mức tích cực thì quá trình DMA bắt đầu. DRQ0 có mức ưu tiên cao nhất, DRQ3 có mức ưu tiên thấp nhất.
- DACK0-DACK3 (tín hiệu ra): tín hiệu trả lời DMA. DACK được nối vào từng thiết bị ngoài để báo cho thiết bị ngoài biết là DMAC đang phục vụ nó.

- HOLD (tín hiệu ra): dùng để báo cho CPU là DMAC đòi quyền điều khiển bus.
- HLDA (tín hiệu vào): tín hiệu HLDA được gửi từ CPU báo cho DMAC biết là quyền điều khiển bus thuộc về DMAC.
- A0-A3 (tín hiệu vào/ra): 4 dây địa chỉ thấp. Trong chu kì nghỉ được dùng để xác định địa chỉ các thanh ghi trong DMAC khi lập trình hoạt động cho DMAC. Trong chu kỳ tích cực được dùng để cung cấp 4 bit địa chỉ A0-A3 cho bus địa chỉ.
- A4-A7(tín hiệu ra): cung cấp 4 bit địa chỉ cao cho bus địa chỉ trong quá trình DMA.
- DB0-DB7(tín hiệu vào/ra): là bus 8 bit đa năng.

  Trong chu kỳ nghỉ: là bus 8 bit dữ liệu vào truyền dữ liệu xác lập chế độ làm việc cho DMAC và cũng là bus 8 bit dữ liệu ra trong quá trình CPU đọc trạng thái DMAC.

  Trong chu kỳ tích cực (chu kỳ DMA): truyền hai loại thông tin trong 2 thời khoảng:
  + *Thời khoảng đầu*: với sự có mặt của tín hiệu ADSTB thì DB0-DB7 là bus 8 bit địa chỉ cao nhất (A15-A8), 8 bit địa chỉ này cần được chốt lại để kết hợp với 8 bit địa chỉ thấp (A7-A0) thành địa chỉ 16 bit.
  + *Thời khoảng sau*: là bus 8 bit dữ liệu, dành cho truyền dữ liệu trong chế độ DMA.
- MR (tín hiệu ra): là tín hiệu điều khiển đọc bộ nhớ trong chế độ DMA.
- MW(tín hiệu ra): là tín hiệu điều khiển ghi bộ nhớ trong chế độ DMA.
- ADSTB (tín hiệu ra): dùng để điều khiển việc chốt (ghi lại) 8 bit địa chỉ cao (A8 - A15).
- AEN (tín hiệu ra): cho phép/cấm thiết bị chốt địa chỉ hoạt động. AEN cũng được dùng để cấm các thiết bị điều khiển bus khác trong chế độ DMA.
- EOP (tín hiệu vào/ra): khi là tín hiệu vào, tín hiệu EOP buộc DMAC kết thúc quá trình DMA. Là tín hiệu ra trong chế độ hoạt động, tín hiệu EOP do DMAC phát ra báo đã truyền đủ số byte xác định bởi TC.
- IOW (tín hiệu vào/ra): trong chế độ nghỉ là tín hiệu vào, tín hiệu này do CPU cung cấp để điều khiển ghi dữ liệu vào DMAC. Trong chu kỳ tích cực là tín hiệu ra, DMAC tạo tín hiệu điều khiển ghi thiết bị ngoại vi.

8.2.2. Các chế độ hoạt động của DMAC 8237

DMAC 8237 hoạt động trong 2 chu kỳ chính là: chu kỳ nghỉ và chu kỳ hoạt động.

- Chu kỳ nghỉ: khi không có kênh nào yêu cầu thì DMAC đi vào chu kỳ nghỉ. Trong chu kỳ này DMAC "quan sát" các dây DRQ ở từng nhịp

đồng hồ, đồng thời làm việc với CPU. Trong chu kỳ nghỉ của DMAC thì CPU có thể xác lập, thay đổi hoặc kiểm tra nội dung các thanh ghi chế độ làm việc của DMAC. Khi có tín hiệu DRQ tích cực, DMAC gửi ra tín hiệu HOLD và đi vào chu kỳ hoạt động.

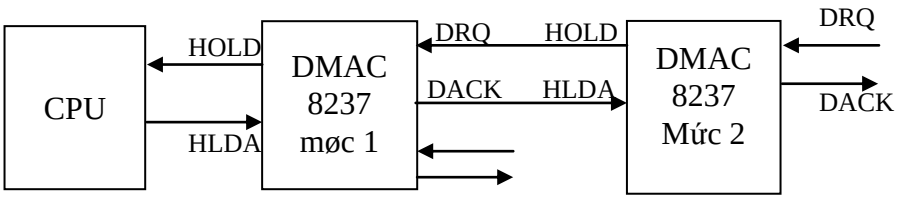
- Chu kỳ hoạt động: Trong chu kỳ hoạt động DMAC 8237 có thể hoạt động theo 1 trong 4 kiểu, tùy theo lập trình:

+ Kiểu truyền đơn lẻ (single transfer mode): trong kiểu truyền này, DMAC được lập trình để thực hiện *chỉ một lần truyền*. Thanh đếm từ TC tự động giảm, địa chỉ tăng hoặc giảm cho đến khi TC=0. Trong kiểu truyền này DRQ phải được giữ ở mức tích cực chừng nào DACK tích cực còn được ghi nhận. Nếu DRQ còn giữ ở mức tích cực quá một lần truyền thì HOLD bị đặt xuống 0 và DMAC trả quyền điều khiển bus cho CPU.

+ Kiểu truyền khối (block transfer mode): trong kiểu truyền khối DMAC bắt đầu thực hiện việc truyền cả khối dữ liệu khi DRQ tích cực và tiếp tục truyền cho đến khi thanh đếm TC đếm xuống 0 hoặc cho đến khi nhận được tín hiệu EOP từ bên ngoài. Tín hiệu DRQ chỉ cần được giữ tích cực cho đến khi DACK trở thành tích cực.

+ Kiểu truyền theo yêu cầu (demand transfer mode): trong kiểu truyền này, dữ liệu được truyền cho đến khi DRQ không tích cực hoặc TC = 0 hoặc nhận được tín hiệu EOP. Việc truyền dữ liệu trực tiếp cho đến khi thiết bị bên ngoài chuyển được hết dữ liệu của nó. Nếu thiết bị chuẩn bị dữ liệu chậm hơn, nó đặt DRQ thành không tích cực. Khi thiết bị chuẩn bị xong thì nó đặt DRQ lên thành tích cực và quá trình DMA lại tiếp tục.

+ Kiểu ghép tầng(Cascade mode) (Hình 81).



Hình 81

Trong kiểu ghép tầng: dây HOLD của DMAC tầng 2 nối với dây DRQ của DMAC tầng 1. Dây DACK của DMAC tầng 1 nối với các dây HLDA của DMAC tầng 2.

Yêu cầu DRQ của DMAC tầng 2 được truyền qua mạch phân định ưu tiên của DMAC tầng 1. Trong trường hợp kênh có nối tầng, DMAC tầng 1 chỉ thực hiện chức năng phân định ưu tiên và không đưa ra các tín hiệu điều khiển bus của riêng nó.

8.2.3. Các thanh ghi của DMAC

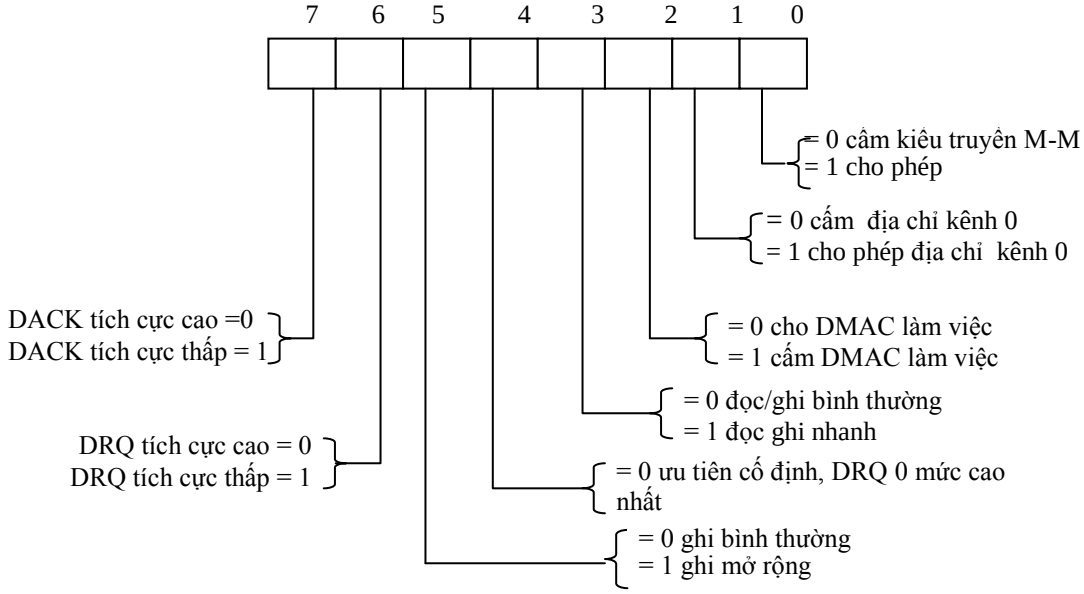
Mỗi kênh DMA có 4 thanh ghi chức năng:

+ Thanh ghi địa chỉ nền (16 bit): lưu giữ địa chỉ nền (địa chỉ đầu) của vùng nhớ cần truy cập.
+ Thanh ghi đếm nền (16 bit): lưu giữ con số xác định lượng từ dữ liệu gốc cần truyền.
+ Thanh ghi địa chỉ hiện thời (16 bit): Thanh ghi này chứa giá trị địa chỉ của ô nhớ đang được truy nhập trong quá trình DMA. Địa chỉ tự động tăng hoặc giảm sau mỗt lần truyền và giá trị tức thời này lập tức được nạp vào đây. Thanh ghi này được ghi/đọc bởi CPU.
+ Thanh ghi đếm từ hiện thời (16 bit): thanh ghi này giữ số lần truyền được thực hiện. Số đếm này giảm sau mỗi lần truyền. Khi giá trị của thanh ghi đếm từ hiện thời giảm xuống 0 thì tín hiệu EOP được tạo ra. Thanh ghi này được ghi/đọc bởi CPU.

Các thanh ghi điều khiển và trạng thái:

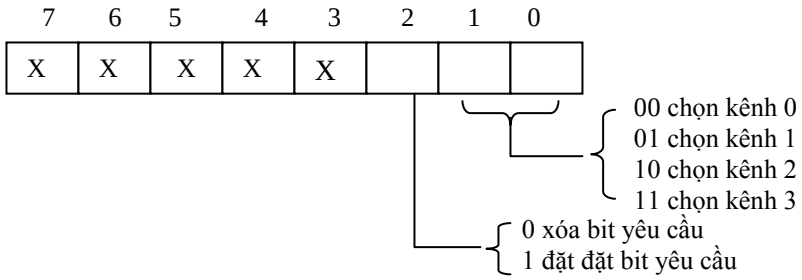
- Thanh ghi lệnh (Command Register)

Thanh ghi lệnh 8 bit này được nạp để xác lập chế độ làm việc cho DMAC: cho phép hoạt động, kiểu ưu tiên, kiểu tốc độ truy nhập, dạng tín hiệu DRQ và DACK.



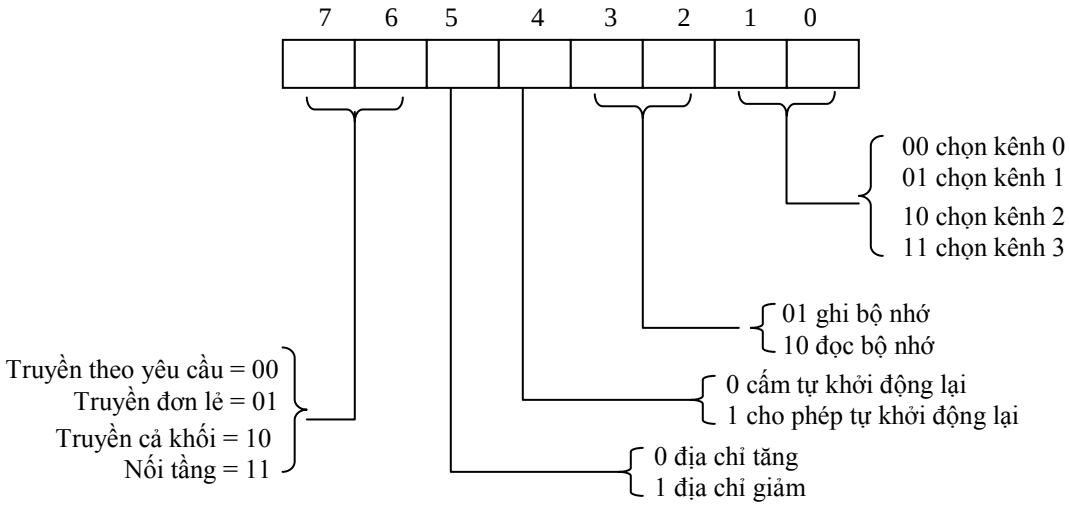
- Thanh ghi yêu cầu (request register)

Thanh ghi yêu cầu 8 bit: cho phép DMAC có thể trả lời DACK cho từng thiết bị yêu cầu và đặt/xóa yêu cầu DRQ.



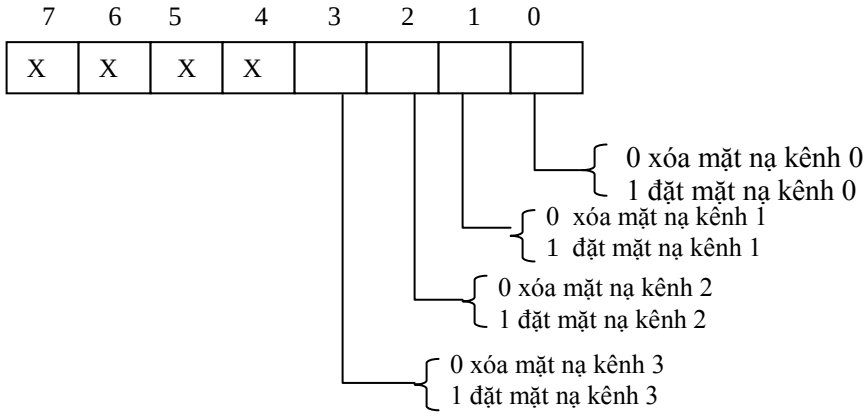
- Thanh ghi kiểu làm việc (8 bit):

Xác lập chế độ và kiểu làm việc cho từng kênh DMA, cho phép chọn: kênh, kiểu truyền, thực hiện DMA để ghi hay đọc, kiểu tăng/giảm địa chỉ, cấm/cho phép tự khởi đầu lại.



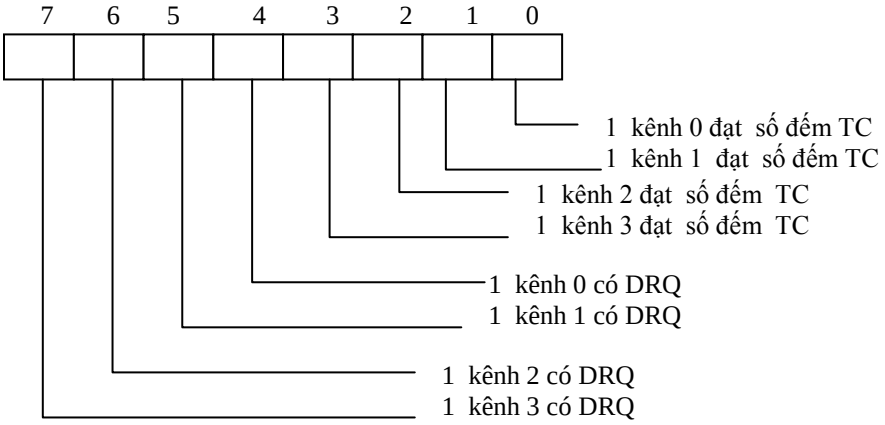
- Thanh ghi mặt nạ

Thanh ghi mặt nạ 8 bit: thanh ghi này cho phép đặt/xoá mặt nạ (nhận/không nhận DRQ) đến mức từng kênh.



- Thanh ghi trạng thái:

Thanh ghi trạng thỏi(8 bit): cho biết trạng thỏi DRQ và TC của từng kênh:



8.2.4. Lập trình chế độ làm việc DMAC 8237

Địa chỉ nền của DMAC 8237 #1 : 000h

Địa chỉ nền của DMAC 8237 #2 : 0C0h

+ Lập trình các thanh ghi điều khiển

Thanh ghi	Thao tác	Địa chỉ thanh ghi			
		A3	A2	A1	A0
Lệnh	Ghi	1	0	0	0
Kiểu làm việc	Ghi	1	0	1	1
Trạng thái	Đọc	1	0	0	0
Đặt /xoá mặt nạ	Ghi	1	0	1	0
Yêu cầu	Ghi	1	0	0	1

+ Lập trình các thanh ghi địa chỉ và đếm

Kênh	Thanh ghi	Thao tác	Địa chỉ thanh ghi			
			A3	A2	A1	A0
0	Byte thấp địa chỉ nền	Ghi	0	0	0	0
	Byte cao địa chỉ nền	Ghi	0	0	0	0
	Byte thấp thanh đếm nền	Ghi	0	0	0	1
	Byte cao thanh đếm nền	Ghi	0	0	0	1

1	Byte thấp địa chỉ nền	Ghi	0	0	1	0
	Byte cao địa chỉ nền	Ghi	0	0	1	0
	Byte thấp thanh đếm nền	Ghi	0	0	1	1
	Byte cao thanh đếm nền	Ghi	0	0	1	1
2	Byte thấp địa chỉ nền	Ghi	0	1	0	0
	Byte cao địa chỉ nền	Ghi	0	1	0	0
	Byte thấp thanh đếm nền	Ghi	0	1	0	1
	Byte cao thanh đếm nền	Ghi	0	1	0	1
3	Byte thấp địa chỉ nền	Ghi	0	1	1	0
	Byte cao địa chỉ nền	Ghi	0	1	1	0
	Byte thấp thanh đếm nền	Ghi	0	1	1	1
	Byte cao thanh đếm nền	Ghi	0	1	1	1

- Trình tự lập trình khởi động DMAC
  + Nạp byte thấp cho thanh ghi địa chỉ nền
  + Nạp byte cao cho thanh ghi địa chỉ nền.
  + Nạp byte thấp cho thanh ghi đếm.
  + Nạp byte cao cho thanh ghi đếm
  + Nạp thanh ghi lệnh, thanh ghi yêu cầu và thanh ghi kiểu làm việc.
  + Nạp thanh ghi địa chỉ trang
  + Xóa mặt nạ kênh cần dùng.
  + Cho phép kênh hoạt động

8.2.5. Quá trình DMA

DMAC được xác lập chế độ làm việc: các thanh ghi lệnh, thanh ghi kiểu làm việc, thanh ghi địa chỉ nền và thanh đếm kích thước khối dữ liệu được nạp, cho phép kênh làm việc.

- Thiết bị vào-ra phát tín hiệu DRQ cho DMAC.
- DMAC phát tín hiệu HOLD = 1 cho CPU, đòi CPU đi vào chế độ DMA.
- CPU thực hiện nốt chu kỳ máy.
- CPU phát tín hiệu HLDA cho DMAC và tách ra khỏi bus hệ thống. Quyền điều khiển bus thuộc về DMAC.
- DMAC làm chủ các bus địa chỉ, bus dữ liệu và bus điều khiển. DMAC phát tín hiệu DACK, phát ra địa chỉ bộ nhớ (16 bit), phát tín hiệu điều khiển ghi/ đọc thiết bị vào-ra, tín hiệu điều khiển đọc/ghi bộ nhớ, một byte dữ liệu được truyền trực tiếp giữa bộ nhớ và thiết bị vào-ra. Địa chỉ

ô nhớ tiếp theo được phát ra và quá trình trên lặp lại cho đến khi thực hiện xong quá trình DMA theo kiểu đã được lập trình (TC = 0).

- Khi quá trình DMA kết thúc DMAC phát tín hiệu HOLD = 0 cho CPU và trả quyền điều khiển hệ thống bus cho CPU. Quá trình DMA cũng có thể được kết thúc từ bên ngoài bằng tín hiệu EOP.
- CPU nắm lại quyền điều khiển bus hệ thống và tiếp tục làm việc bình thường.

8.2.6. Thanh ghi trang và địa chỉ trang

DMAC chỉ tạo được địa chỉ 16 bit nên chỉ cho phép truy nhập được vùng nhớ 64 Kbyte. Để thực hiện quá trình DMA với loại bộ nhớ có bus địa chỉ rộng hơn ở máy PC, người ta tổ chức quản lý bộ nhớ theo trang.

Người ta thiết kế các thanh ghi trang chứa các bit cao của địa chỉ vật lý, kết hợp với một mạch điện tử (74LS612) để tạo các địa chỉ vật lý trong quá trình DMA.

Hệ thống DMA trong máy vi tính PC có 4 thanh ghi trang ứng với 4 kênh DMAC (Bảng 14).

Bảng 14

Thanh ghi trang	Địa chỉ cổng thanh ghi trang
Kênh 0	87H
Kênh 1	83H
Kênh 2	81H
Kênh 3	82H

Các thanh ghi trang chứa địa chỉ đầu của các vùng nhớ, mỗi một vùng nhớ được dùng làm vùng nhớ đệm phục vụ quá trình DMA:

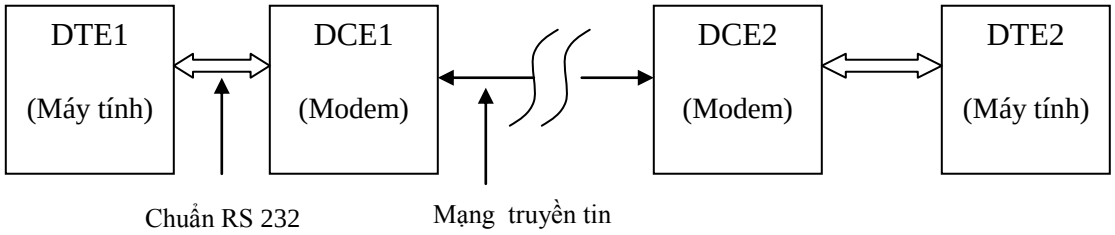
	Thanh ghi trang (chứa các bit cao của địa chỉ vật lý)	Thanh ghi địa chỉ nền trong DMAC 8237 (chứa 16 bit địa chỉ thấp)
Địa chỉ bộ nhớ	Các bit địa chỉ cao	A15... . A0

8.3. VÀO-RA TUẦN TỰ VÀ MODULE GIAO DIỆN VÀO-RA TUẦN TỰ UART

Vào/ra tuần tự là phương pháp kết nối và truyền dữ liệu giữa máy tính và thiết bị ngoại vi, trong đó mã chứa thông tin được truyền tuần tự theo từng bit, bit nọ sau bit kia (truyền tin tuần tự).

8.3.1. Chuẩn truyền tin RS-232

Chuẩn truyền tin RS-232 quy định về phương pháp kết nối và giao diện giữa DTE và DCE (Hình 82).



Hình 82

DTE (Data Terminal Equipment) thiết bị đầu cuối dữ liệu: là thiết bị phát hoặc nhận dữ liệu.

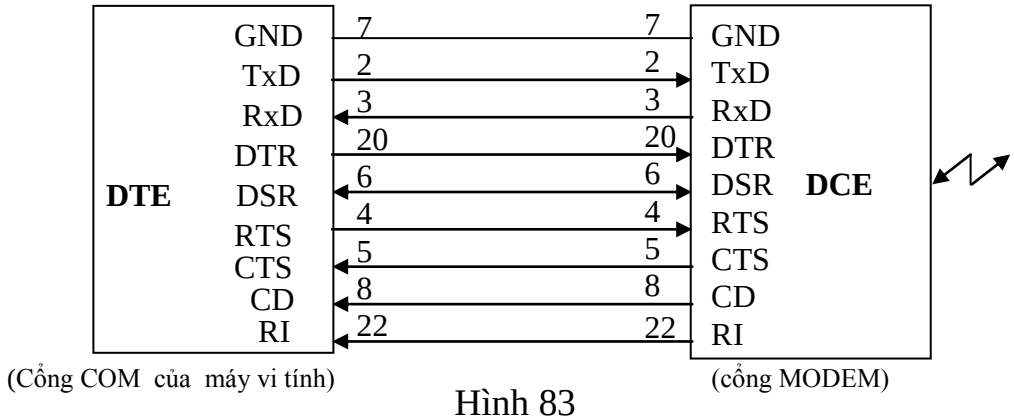
DCE (Data Circuit-terminating Equipment): thiết bị truyền dữ liệu.

Chuẩn RS-232 quy định về:

- Kết nối vật lý: loại đầu nối, số lượng, vị trí và chức năng truyền thông tin của mỗi chân trong đầu nối.
- Mức điện áp tín hiệu.
- Kiểu truyền tin, khuôn dạng dữ liệu và tốc độ truyền.

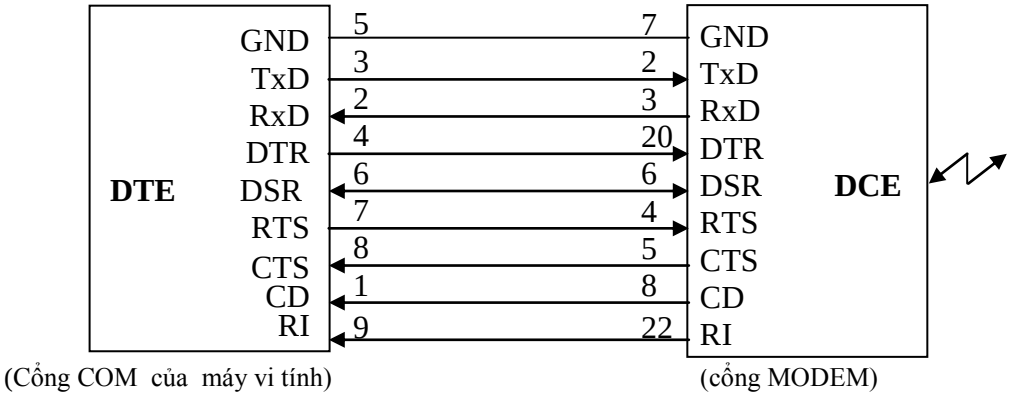
Kết nối vật lý: sử dụng 2 loại đầu nối DB9 9 chân hoặc DB25 25 chân.

- Kết nối DTE & DCE qua đầu nối DB25 (Hình 83):



Hình 83

- Kết nối DTE & DCE qua đầu nối DB9 (Hình 84)



Hình 84

Ý nghĩa của các tín hiệu (nhìn từ phía DTE):

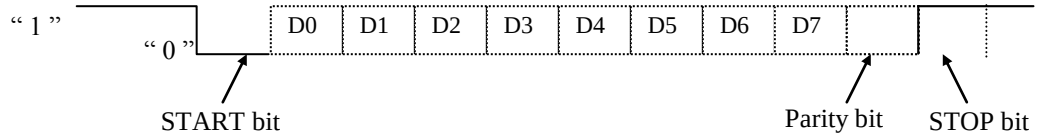
- Các tín hiệu mang dữ liệu:
  TxD: tín hiệu dữ liệu phát
  RxD: tín hiệu dữ liệu nhận
- Các tín hiệu bắt tay: (khi các tín hiệu có mức tích cực)
  Từ DTE (từ cổng COM của máy vi tính):
  DTR: DTE báo sẵn sàng làm việc
  RTS : DTE báo có dữ liệu muốn gửi
  Đến DTE (đến cổng COM máy vi tính)
  DSR: DCE báo sẵn sàng làm việc
  CTS: DCE sẵn sàng nhận dữ liệu để gửi đi
  CD : DCE báo phát hiện được sóng mang của phía bên kia.
  RI : DCE bỏo nhận được tớn hiệu rung chuụng.

Mức tín hiệu: +/- 25V
"0": +3V đến +25V
"1": –3V đến -25V

Kích thước cáp nối không quá 20 m.

Phương pháp truyền:

DTE truyền dữ liệu không đồng bộ về pha nhưng đồng bộ trên từng byte dữ liệu bằng bit " START". Dữ liệu khi được truyền có khuôn dạng sau



Khuôn dạng dữ liệu truyền: bên phát và bên nhận phải được chọn cùng một khuôn dạng dữ liệugiống nhau.

Tốc độ truyền: hai bên phát và nhận phải đặt tốc độ truyền bằng nhau.

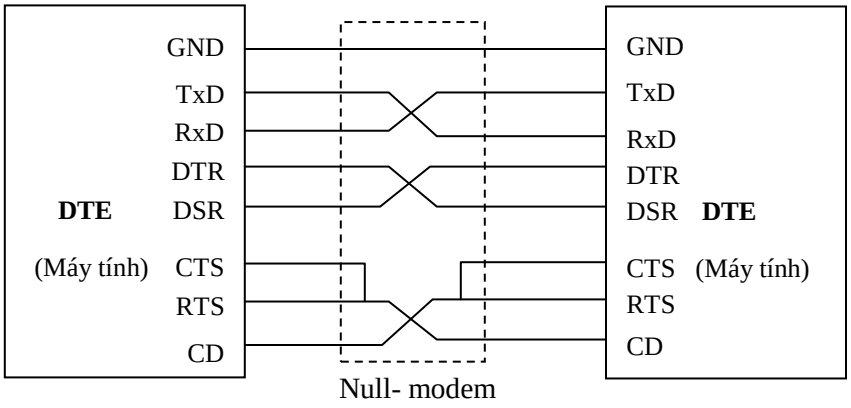
Modem (DCE)

Khi cần thực hiện truyền dữ liệu giữa hai máy tính trên khoảng cách lớn hơn 20 mét, phương pháp hiệu quả nhất là truyền qua hệ thống mạng điện thoại công cộng. Thiết bị kết nối giữa máy tính và hệ thống mạng điện thoại là modem. Modem thực hiện chuyển tín hiệu số nhị phân thành tín hiệu tương tự để hệ thống điện thoại có thể truyền đi được. Modem cũng thực hiện chuyển tín hiệu tương tự nhận được từ hệ thống mạng điện thoại thành tín hiệu số nhị phân cho máy vi tính. Ngoài chức năng điều chế (MOdulate) và giải điều chế (DEModulate), modem còn thực hiện các chức năng giao diện với hệ thống điện thoại và với máy vi tính.

Kết nối kiểu Null-modem

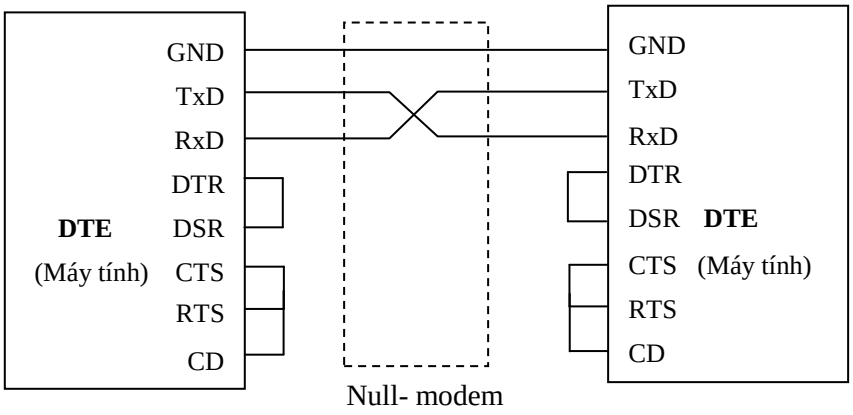
Có thể thực hiện kết nối và truyền tin trực tiếp giữa máy tính với máy tính hoặc giữa máy tính với thiết bị ngoại vi khác qua cổng tuần tự, không cần modem. Phương pháp kết nối trực tiếp DTE-DTE (máy tính với máy tính) không qua modem được gọi là kết nối kiểu Null-Modem. Khi thực hiện kết nối và truyền tin tuần tự không qua modem vẫn cần tuân theo chuẩn RS- 232. Có hai kiểu kết nối Null-modem: kiểu 7 dây và kiểu 3 dây.

Kết nối Null-modem kiểu 7 dây (Hình 85).



Hình 85

Kết nối Null-modem kiểu 7 dây (Hình 86).



Hình 86

8.3.2. Module giao diện vào/ra tuần tự UART

Module giao diện vào/ra tuần tự UART thực hiện hai chức năng chính:

- Chuyển 1 byte dữ liệu dạng 8 bit (do CPU gửi đến) ở thanh ghi đệm phát thành dạng tuần tự, tạo khung dữ liệu dạng tuần tự và phát đi tuần tự từng bit cho đến hết byte dữ liệu. Nhận 1 khung dữ liệu dạng tuần tự, loại bỏ các bit tạo khung (bit START, PARITY, STOP), chuyển thành dạng dữ liệu song song và cất vào thanh ghi đệm nhận.

- Tạo và nhận các tín hiệu bắt tay theo chuẩn RS 232.

Địa chỉ nền của 2 module giao diện vào-ra tuần tự:

Module UART #1 (COM1): 3F8H

Module UART #2 (COM2): 2F8H

*a) **Thanh ghi dữ liệu phát THR***

THR là nơi chứa dữ liệu (hoặc ký tự) cần phát đi.

Địa chỉ: Địa chỉ nền +0, ghi, DLAB=0

D7							D0

Bit D0 là bit thấp nhất, được phát đầu tiên.

b) Thanh ghi dữ liệu nhận RBR:

RBR là nơi chứa dữ liệu (hoặc ký tự) nhận được

Địa chỉ: Địa chỉ nền +0, đọc, DLAB=0

D7							D0

DLAB là bit D7 của thanh ghi LCR.

c) Thanh ghi điều khiển đường truyền LCR

Thanh ghi LCR xác định khuôn dạng dữ liệu phát/nhận và cho phép truy nhập vào các thanh ghi THR, RBR, IER hoặc BRG.

Địa chỉ: Địa chỉ nền +3, ghi

D7							D0

D1 và D0: xác định khích thức dữ liệu truyền.

D1	D0	
0	0	5 bit
0	1	6 bit
1	0	7 bit
1	1	8 bit

D2- xác định số lượng bit stop được tạo và kiểm tra

D2 = 0 1 Bit Stop

D2 =1 2 Bit Stop

D3- cho phép tạo hoặc kiểm tra parity

D3 = 0 không cho phép

D3 = 1 cho phép

D4- chọn kiểu parity

D4 = 0 Số lượng lẻ bit “1” được báo hoặc kiểm tra

D4 = 1 Số lượng chẵn bit “1” được báo hoặc kiểm tra

D5- chọn mức tích cực của bit parity

Nếu D5 = 1 và D4 = 1 thì mức tích cực của bit parity là 0 (Parity chẵn)

Nếu D5 = 1 và D4 = 0 thì mức tích cực của bit parity là 1 (Parity lẻ)

D6- đặt điều khiển nghỉ (BREAK)

Khi D6 = 1 thì SOUT = 0. Chú ý D6 = 0 ! (thường đặt)

D7-DLAB bit: cho phép truy nhập các thanh ghi THR, RBR, IER hoặc BRG

D7 = 1 cho phép truy nhập cặp thanh ghi tốc độ truyền BRG

D7 = 0 cho phép truy nhập các thanh ghi THR, RBR và IER

d) Thanh ghi xác lập tốc độ truyền 16 bit BRG

BRG gồm 2 thanh ghi byte cao và byte thấp, xác định hệ số chia của tốc độ truyền.

Byte thấp:

Địa chỉ: Địa chỉ nền +0, ghi, DLAB=1

D7							D0

Byte cao

Địa chỉ: Địa chỉ nền +1, ghi, DLAB=1

D7							D0

Nội dung thanh ghi BRG (Bảng 15):

Bảng 15

Nội dung thanh ghi (Hexa)	**Tốc độ truyền** (baud)
0900	50
0180	300
00C0	600
0060	1200
0020	3600
⋮	⋮
000C	9600

e) Thanh ghi điều khiển giao diện với modem MCR

MCR điều khiển giao diện với modem và cho phép phát yêu cầu ngắt IRQ.

Địa chỉ: Địa chỉ nền +4, ghi

D7							D0

D0- DTR bit:

D0 = 1 đặt DTR tích cực

D0 = 0 đặt DTR không tích cực

D1-RTS bit:

D1 = 1 đặt RTS tích cực

D1 = 0 đặt RTS không tích cực

D2- OUT1 bit: ảnh hưởng đến đầu ra OUT1

D3- OUT2 bit: ảnh hưởng đến đầu ra OUT2

D3 = 1 đặt OUT 2= 0, cho phép UART phát tín hiệu IRQ

D3 = 0 không cho phép UART phát tín hiệu IRQ

D4: cho phép vào chế độ tự kiểm tra UART bằng cách nối tắt nội bộ các đầu TxD& RxD; DTR & DSR; RTS & CTS….

D5 = D6 = D7 = 0

f) Thanh ghi trạng thái đường truyền LSR

LSR cung cấp thông tin về trạng thái đường truyền tin.

Địa chỉ: Địa chỉ nền +5, đọc

D7							D0

D0- RBR bit:

D0 = 1 báo UART nhận được 1 byte dữ liệu và đã đặt vào thanh ghi RBR, (báo RBR đầy).

D1- Lỗi đè

D1 = 1 báo RBR có dữ liệu nhận nhưng chưa được CPU đọc và bị 1 byte mới nhận tiếp theo đè lên.

D2- Lỗi Parity

D2 = 1 báo có lỗi Parity (lỗi truyền tin).

D3 - Lỗi khuôn dạng

D3 = 1 báo khuôn dạng dữ liệu nhận bị sai.

D4- Lỗi break

D4 = 1 báo đầu nhận dữ liệu có trạng thái logic 0 dài hơn thời gian truyền 1 byte dữ liệu.

D5-THR bit:

D5 = 1 báo UART sẵn sàng nhận 1 byte dữ liệu mới từ CPU để phát đi (thanh ghi THR rỗng).

D6-TSR bit: D6=1 báo thanh ghi TSR rỗng.

D7- không dùng.

g) Thanh ghi trạng thái Modem MSR

MSR cung cấp thông tin về trạng thái các tín hiệu bắt tay từ modem.

Địa chỉ: Địa chỉ nền +6, đọc

D7							D0

D0-D3: báo có sự thay đổi trạng thái của các tín hiệu bắt tay từ modem

D4 - CTS bit:

D4 = 1 báo tín hiệu CTS có mức tích cực

D5 - DSR bit

D5 = 1 báo tín hiệu DSR có mức tích cực

D6 - RI bit:

D6 = 1 báo tín hiệu RI có mức tích cực

D7 - CD-bit:

D7 = 1 báo tín hiệu CD có mức tích cực

h) Thanh ghi chọn nguồn ngắt IER

IER cho phép chọn các nguồn báo ngắt.

Địa chỉ: Địa chỉ nền +1, ghi, DLAB=0

D7							D0

D0 - liên quan đến báo ngắt của thanh ghi dữ liệu nhận RBR.

D0 = 1 cho phép UART báo ngắt khi nó nhận được dữ liệu mới (khi RBR đầy)

D0 = 0 không cho phép

D1- liên quan đến báo ngắt của thanh ghi phát THR

D1 = 1 cho phép UART báo ngắt khi nó sẵn sàng nhận 1 byte dữ liệu mới từ CPU để phát đi (khi THR rỗng)

D1 = 0 không cho phép

D2- liên quan đến báo ngắt của thanh ghi LSR

D2 = 1 cho phép UART báo ngắt khi có thay đổi thông tin về trạng thái truyền tin trong LSR

D2 = 0 không cho phép

D3 - liên quan đến báo ngắt của thanh ghi MSR

D3 = 1 cho phép UART báo ngắt khi có thay đổi thông tin về trạng thái modem trong MSR

D3 = 0 không cho phép báo ngắt

D4 ÷ D7 = 0

i) Thanh ghi nhận dạng nguồn ngắt IIR

Thanh ghi IIR cho phép nhận dạng nguồn gây ngắt và kiểm tra trạng thái báo ngắt.

Địa chỉ: Địa chỉ nền +2, đọc

D7							D0

D3 ÷ D7 = 0

D0 - trạng thái báo ngắt

D0 = 0 còn nguồn báo ngắt

D0 = 1 hết báo ngắt

D1, D2 : tổ hợp giá trị hai bit này chỉ thị nguồn đang báo ngắt (Bảng 16).

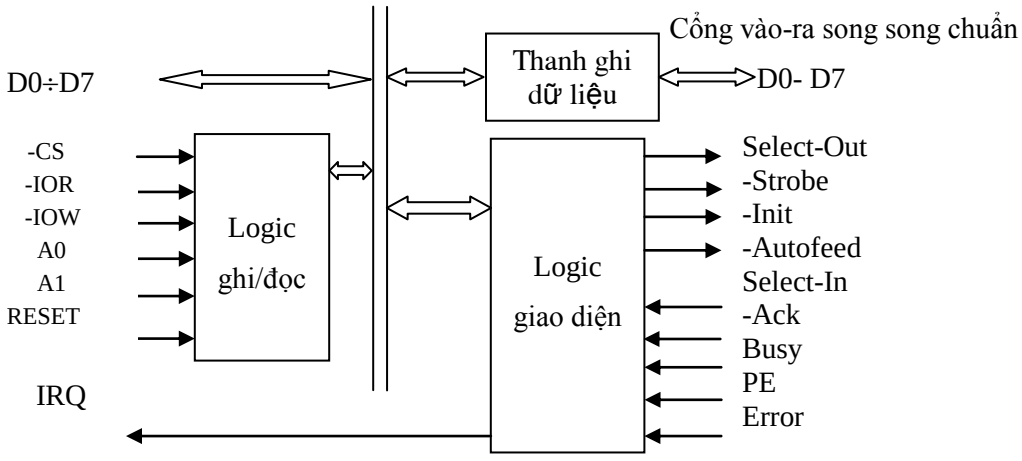
Bảng 16

D2	D1	D0	Nguồn báo ngắt	Mức ưu tiên	Phương pháp xoá báo ngắt
0	0	1	Không có báo ngắt		
1	1	0	LSR	Cao nhất	đọc LSR
1	0	0	RBR	nhì	đọc RBR
0	1	0	THR	ba	ghi THR
0	0	0	MSR	tư	đọc MSR

8.4. MODULE GIAO DIỆN SONG SONG

8.4.1. Module giao diện song song chuẩn

Module giao diện song song chuẩn (Hình 87) thực hiện chức năng giao diện giữa CPU và các thiết bị ngoại vi (trong đó có máy in) theo kiểu song song, trong đó các bit của 1 byte dữ liệu được đưa ra hoặc nhận vào CPU trong cùng một nhịp thời gian.



Hình 87

Các tín hiệu trên đầu nối song song chuẩn DB 25 (Bảng 17).

Bảng 17

Chân số	Tín hiệu
1	-Strobe
2 ÷ 9	D0 ÷ D7
10	-Ack
11	Busy
12	Paper Error
13	Select-In
14	-Autofeed
15	-Error
16	-Init
17	-Select-Out
18 ÷25	GND

8.4.2. Các thanh ghi

Địa chỉ nền của các thanh ghi (cổng giao diện song song): 378H

a) Thanh ghi dữ liệu

Thanh ghi dữ liệu chứa dữ liệu xuất ra hoặc dữ liệu nhập vào qua module giao diện.

Địa chỉ: Địa chỉ nền +0, đọc /ghi

D7							D0

D7-D0: các bit của byte dữ liệu.

b) Thanh ghi điều khiển

Thanh ghi điều khiển chứa các bit tạo tín hiệu bắt tay với thiết bị ngoại vi. Nếu thiết bị ngoại vi là máy in thì đây là các bit tạo tín hiệu bắt tay và điều khiển máy in.

Địa chỉ: Địa chỉ nền + 2, ghi

D7							D0

D0: STROBE-bit

D0 = 1 tạo tín hiệu STROBE với mức tích cực (thấp). Tín hiệu này được dùng để chốt dữ liệu vào thiết bị ngoại vi hoặc máy in.

D1: AUTOFEED-bit

D1 = 1 tạo tín hiệu AUTOFEED tích cực(thấp). Với máy in, tín hiệu này điều khiển dịch một dòng in.

D2: INIT - bit

D2 = 0 tạo tín hiệu xung INIT mức tích cực (thấp) với độ rộng 50 micro giây. Với máy in, tín hiệu này khởi động máy in.

D3: SELECT-OUT-bit

D3 = 1 tạo tín hiệu SELECT-OUT tích cực (thấp). Tín hiệu SELECT-OUT được dùng để "bắt tay" với thiết bị ngoại vi hoặc chọn máy in.

D4: IRQ-bit

D4 = 1 cho phép module giao diện song song chuẩn phát tín hiệu yêu cầu ngắt IRQ7 khi nhận được tín hiệu ACK chuyển từ mức "1" sang mức "0".

D5-D7: không dùng.

c) Thanh ghi trạng thái

Thanh ghi trạng thái chứa các bit thông tin phản ảnh trạng thái của các tín hiệu từ bên ngoài vào (trạng thái của thiết bị ngoại vi hoặc của máy in).

Địa chỉ: Địa chỉ nền + 1, đọc

D7							D0

D0-D2: không dùng

D3: ERROR-bit

D3 = 0 báo tín hiệu ERROR có mức tích cực thấp. Với máy in: báo máy in đang trong trạng thái có lỗi.

D4: SELECT-IN- bit

D4 = 1 báo tín hiệu SELECT-IN tích cực cao. Với máy in: báo máy in đã sẵn sàng làm việc.

D5: PE- bit

D5 = 1 báo máy in gặp lỗi hết giấy, mức tích cực cao.

D6: ACK-bit.

D6=0 báo tín hiệu xung ACK có mức tích cực thấp. Với máy in, tín hiệu xung ACK tích cực báo máy in sẵn sàng nhận một ký tự mới.

D7: BUSY-bit

D7 = 0 báo tín hiệu BUSY có mức tích cực cao. Với máy in, báo máy in đang bận, không thể nhận dữ liệu.

D7 = "1" báo máy in không bận.

8.5. GIAO DIỆN TUẦN TỰ VẠN NĂNG USB

Các máy tính PC hiện nay vẫn còn sử dụng các thiết bị ngoại vi với các phương tiện kết nối và giao diện được thiết kế và sử dụng từ những năm 1980. Các kiểu kết nối và giao diện chuẩn này có những nhược điểm như:

- Số lượng thiết bị ngoại vi có thể được kết nối với máy tính qua các đầu cắm chuẩn bị hạn chế (ví dụ, các đầu cắm chuẩn ở máy tính PC/AT gồm một đầu cắm bàn phím PS/2, một đầu cắm chuột PS/2, hai đầu cắm (cổng COM) cho các thiết bị ngoại vi giao diện kiểu tuần tự, một đầu cắm cho thiết bị ngoại vi giao diện kiểu song song, một đầu cắm cho thiết bị hiển thị v.v.).
- Tốc độ truyền dữ liệu chậm.
- Tài nguyên hệ thống hạn chế, điển hình là vấn đề sử dụng và định vị các yêu cầu ngắt cho các thiết bị vào-ra.
- Về phía người sử dụng, kiểu kết nối và giao diện chuẩn dẫn đến những khó khăn như: nhiều loại đầu nối và cáp nối, phần lớn các thiết bị ngoại vi không thể gắn vào máy tính khi máy đang hoạt động, cần phải khởi động lại hệ thống dể cài đặt phần mềm điều khiển thiết bị, giá thành phụ kiện kết nối cao.

Chuẩn giao diện thiết bị ngoại vi USB (Universal Serial Bus) ra đời là nhằm khắc phục những nhược điểm nói trên. Mục tiêu thiết kế và sử dụng giao diện USB là:

- Chỉ sử dụng một loại đầu nối đơn giản để kết nối các loại thiết bị ngoại vi của máy tính PC.
- Có khả năng kết nối nhiều thiết bị ngoại vi vào cùng một đầu cắm chủ USB trên máy tính (Về mặt lý thuyết, có thể kết nối tối đa 127 thiết bị ngoại vi vào một đầu cắm chủ USB)
- Hỗ trợ kết nối nóng, tự động phát hiện và cấu hình thiết bị. Máy chủ nhận biết ngay khi một thiết bị ngoại vi được kết nối lên bus USB, trong khi máy tính vẫn đang hoạt động. Máy chủ thực hiện thủ tục kiểm tra thiết bị, nhờ đó sẽ biết cài đặt phần mềm điều khiển vào-ra thích hợp với thiết bị được kết nối. Khả năng này làm đơn giản hoá việc kết nối và sử dụng thiết bị ngoại vi, cho phép người dùng chỉ cần cắm thiết bị lên bus USB và sử dụng.
- Hỗ trợ kết nối với các thiết bị ngoại vi mới.
- Sử dụng ít năng lượng.
- Giá thành thiết bị kết nối thấp.

8.5.1. Các hệ thống và cấu hình USB

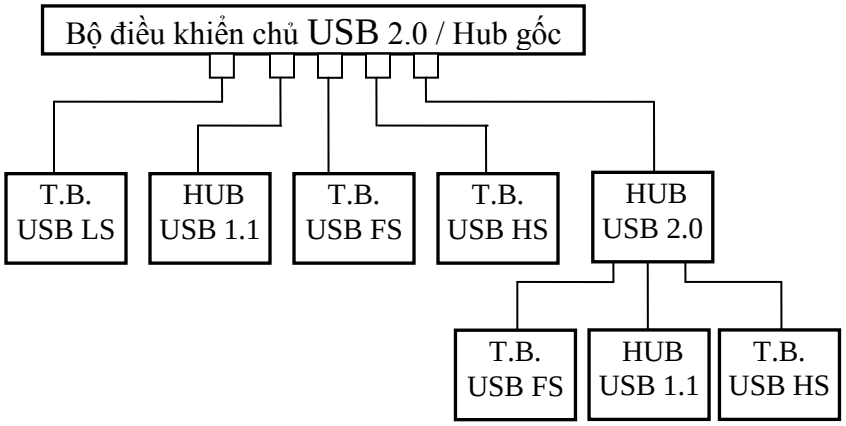
USB là hệ thống truyền tin được tổ chức theo kiểu chủ-tớ. Hệ thống USB được thiết kế để các thiết bị USB có thể kết nối và truyền dữ liệu với máy tính (đóng vai trò là *bộ điều khiển chủ USB*) theo chuẩn USB. Thiết bị USB được hiểu là các thiết bị được thiết kế để giao diện và truyền tin theo chuẩn USB, kể cả thiết bị hub. Hệ thống USB có cấu hình kiểu hình sao, trong đó hub USB

cung cấp điểm kết nối cho các thiết bị USB. Bộ điều khiển chủ USB chứa *hub gốc*.

Có hai hệ thống USB là USB 1. x và USB 2. 0. Hệ thống USB 1. x, ra đời vào năm 1996, hỗ trợ tốc độ truyền 1,5 Mb/s (tốc độ thấp – Low Speed) và 12 Mb/s (tốc độ đầy đủ – Full Speed).

Hệ thống USB 2. 0, ra đời vào năm 2000, được thiết kế để hỗ trợ các thiết bị USB tốc độ thấp LS, tốc độ đầy đủ FS và tốc độ cao HS (High Speed) 480 Mb/s.

Kiến trúc điển hình của hệ truyền tin USB, trong đó các thiết bị USB 1. x và USB 2. 0 kết nối trong hệ thống USB 2. 0 như sau (Hình 88):



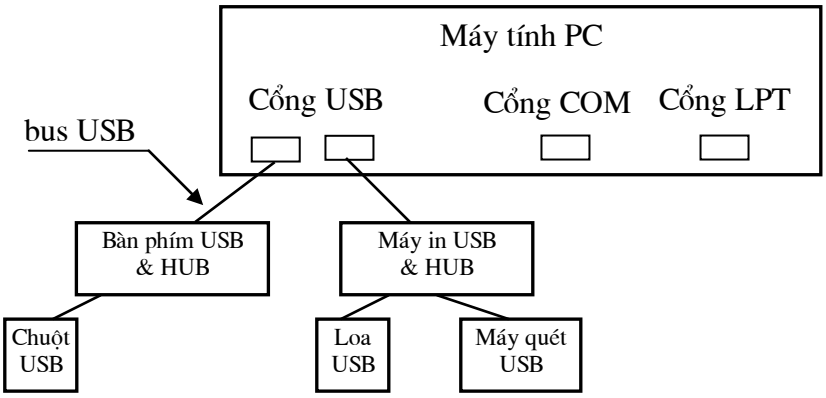
Hình 88

trong đó ký hiệu: T. B. USB LS là thiết bị USB 1. x tốc độ thấp

T. B. USB FS là thiết bị USB 1. x tốc độ đầy đủ

T. B. USB HS là thiết bị USB 2. 0 tốc độ cao

Hình 89 là một ví dụ về việc kết nối các thiết bị ngoại vi USB với máy tính PC qua hệ thống bus USB:



Hình 89

8.5.2. Môi trường truyền tín hiệu

Hệ thống USB sử dụng cáp 4 dây, trong đó 2 dây dùng để cấp nguồn (Vcc = 5VDC và GND, Imax=50mA), 2 dây truyền tín hiệu (+D và -D). Tín hiệu được truyền theo kiểu vi sai.

Chuẩn USB định nghĩa một đầu nối đơn giản để kết nối tất cả các thiết bị ngoại vi USB vào hệ thống chủ. Nhiều thiết bị ngoại vi có cáp USB gắn liền trên thiết bị. Cũng có nhiều thiết bị ngoại vi khác sử dụng cáp USB rời. Để ngăn ngừa cáp USB rời được đồng thời cắm vào hai cổng USB, hai loại đầu cắm cho cáp rời được thiết kế:

- Đầu cắm A được dùng để kết nối thiết bị ngoại vi USB với cổng hub.
- Đầu cắm B được dùng để kết nối cáp USB với thiết bị USB, khi cáp rời được sử dụng. Đầu nối B dạng nhỏ được gọi là đầu nối mini B.

Bảng 18 định nghĩa vị trí chân và ý nghĩa tín hiệu.

Bảng 18

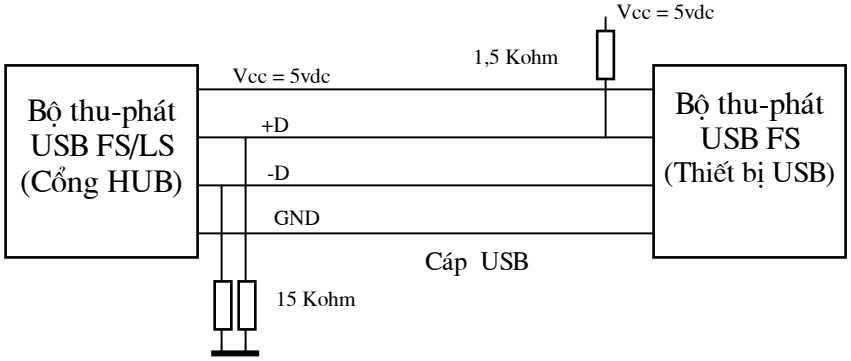
Số chân	Tên tín hiệu	Mầu cáp
1	Vcc	Đỏ
2	- D	Trắng
3	+ D	Xanh lá cây
4	GND	Đen

Cáp USB rời tốc độ thấp 1, 5Mb/s có độ dài tối đa 3m, cặp dây tín hiệu không xoắn, không bọc kim.

Cáp USB rời tốc độ cao 480 Mb/s có độ dài tối đa 5m, cặp dây tín hiệu xoắn, có bọc kim.

- Cơ chế nhận biết thiết bị USB được kết nối vào và gỡ ra khỏi bus USB.

Trước khi truyền thông tin từ/đến thiết bị USB, thiết bị chủ USB cần nhận biết được sự có mặt của thiết bị USB trên bus. Thiết bị chủ USB nhận biết được các sự kiện này nhờ giám sát và kiểm tra mức điện thế của các đường dây tín hiệu trên cổng hub. Các đầu dây tín hiệu +D và -D trên cổng hub luôn được nối qua điện trở 15 Kohm xuống đất (gần có điện thế 0 vdc). Trên thiết bị USB, các đầu dây tín hiệu +D hoặc -D (phụ thuộc tốc độ truyền) luôn được nối qua điện trở 1, 5 Kohm với dây nguồn nuôi (Hình 90).



Hình 90

Khi một thiết bị USB được kết nối (qua cáp) vào cổng USB, điện áp dương sẽ xuất hiện trên dây +D (hoặc -D) do hiệu quả phân áp trên các điện trở này. Thiết bị chủ USB giám sát và kiểm tra thấy có sự thay đổi mức điện thế trên các đường dây tín hiệu, qua đó nhận biết được sự có mặt của thiết bị và cả tốc độ truyền tin của nó, từ đó sẽ bắt đầu thực hiện các thủ tục tiếp theo. Việc gỡ thiết bị ra khỏi bus USB cũng được chủ USB phản ứng theo cách tương tự.

- Tín hiệu vi sai

Hệ thống USB sử dụng tín hiệu vi sai và mã hóa NRZI để truyền thông tin. Tín hiệu vi sai làm giảm thiểu các nhiễu khác nhau trên đường truyền. Thông tin được truyền theo kiểu bán song công, trong đó một đường dây cáp được dùng cho cả hai hướng truyền, nhưng tại một thời điểm thì chỉ truyền được theo một hướng.

Tín hiệu "1" được thể hiện bằng mức D+ > D-. Tín hiệu "0" được thể hiện bằng mức D+ < D- .

Mức điện thế cho bên phát như sau:

"1" = D+ > V_{OHmin} vµ D- < V_{OLmax}

"0" = D+ < V_{OLmax} vµ D- > V_{OHmin}

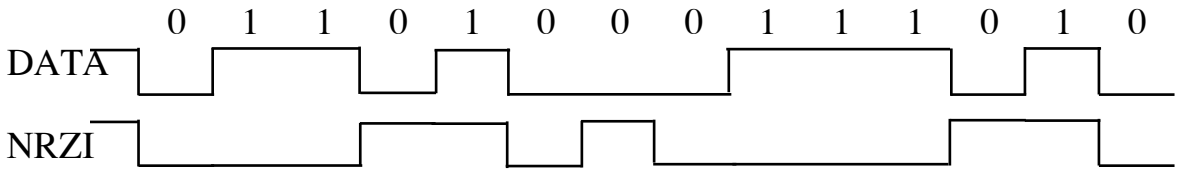
Mức điện thế để bên nhận phát hiện thông tin là:

"1" = D+ - D- > 200 mV vµ D+ > V_{IHmin}

"0" = D- - D+ > 200 mV vµ D- > V_{IHmin}

- Mã hóa tín hiệu

Gói dữ liệu USB được mã hóa theo phương pháp NRZI (Non Return to Zero, Inverted). Bên phát mã hóa dữ liệu theo NRZI, sau đó được gửi đi qua cáp bằng tín hiệu vi sai. Bên nhận nhận tín hiệu vi sai và thực hiện giải mã. Dữ liệu truyền được mã hóa theo NRZI nên không cần đường dây truyền xung đồng bộ riêng. Trong luồng dữ liệu, logic "0" được thể hiện bằng sự chuyển mức tín hiệu từ "1" xuống "0" hoặc ngược lại, logic "1" được thể hiện bằng sự không chuyển mức tín hiệu (Hình 91). Bộ giải mã phải cắt mẫu dòng dữ liệu trong từng khoảng thời gian 1 bit để kiểm tra và phát hiện sự chuyển mức tín hiệu.



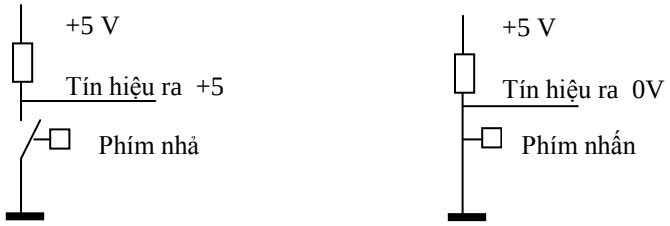
Hình 91

Chương 9
THIẾT BỊ VÀO-RA CƠ BẢN

9.1. BÀN PHÍM

9.1.1. Phím nhấn và phương pháp tạo mã quét

Bàn phím là thiết bị ngoại vi cho phép đưa thông tin vào máy tính dưới dạng mã ký tự. Bàn phím thực hiện chức năng chuyển thông tin dạng lực nhấn phím và vị trí của phím được nhấn thành mã phím và chuyển cho máy tính. Bàn phím gồm hai bộ phận chính là ma trận phím và mạch điện tử quét phím. Ma trận phím là tổ hợp các phím nhấn được sắp xếp theo các hàng và cột. Phím nhấn có cấu trúc cơ điện như sau (Hình 92).



Hình 92

Bình thường phím luôn ở trạng thái nhả, khi phím nhả thì hai tiếp điểm không được nối với nhau, đầu ra có mức điện áp dương, tương ứng với mức logic 1. Khi phím được nhấn thì hai tiếp điểm được nối với nhau qua công tắc phím và đầu ra có mức điện áp bằng 0V tương ứng mức logic 0.

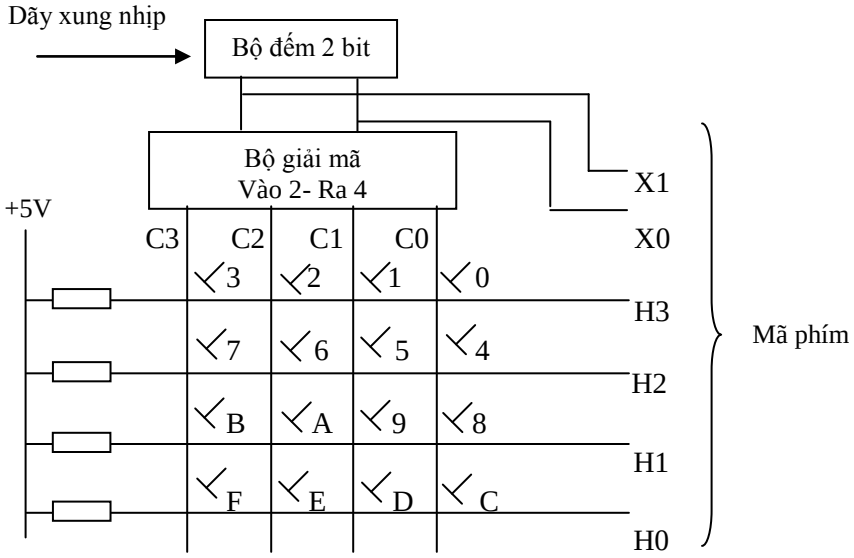
Để mỗi lần nhấn phím có một mã phím tương ứng được sinh ra, cần sắp xếp hệ thống phím dưới dạng ma trận phím. Cấu trúc của ma trận 16 phím cùng với mạch điện tử quét phím có dạng sau (Hình 93).

Ma trận phím gồm các dây hàng và các dây cột giao nhau nhưng không tiếp xúc với nhau. Các phím nhấn được đặt ở chỗ giao của hàng và cột. Hai tiếp điểm của công tắc phím nằm ở trên hàng và cột tại chỗ giao nhau đó. Mỗi khi phím được nhấn thì hai dây hàng và cột được nối với nhau qua hai tiếp điểm và công tắc tại chỗ giao nhau.

Nguyên tắc quét phím và tạo mã quét như sau:

Đầu ra X1X0 của bộ đếm nhị phân 2 bit lần lượt (theo một chu kỳ xác định) cho ra các số nhị phân 2 bit 00, 01, 10, 11, … Số nhị phân này được đưa vào bộ giải mã 2 đầu vào 4 đầu ra. Ở đầu ra C3C2C1C0 của bộ giải mã sẽ lần lượt xuất hiện các giá trị 1110, 1101, 1011, 0111, … Các dây cột Ci của ma trận phím lần lượt có mức logic 0, thao tác này được gọi là thao tác "quét" bàn phím. Mã quét của phím được nhấn hình thành từ việc ghép 2 bit X1X0 ở đầu vào bộ giải mã với 4 bit lấy từ các dây hàng H3H2H1H0, thành mã nhị phân X1X0H3H2H1H0.

Khi không phím nào được nhấn thì tất cả các dây hàng H3H2H1H0 đều có mức logic 1. Khi có một phím nào đó được nhấn thì hai dây hàng và cột được nối với nhau tại chỗ phím đang được nhấn, dây hàng chứa phím được nhấn thông qua công tắc phím được nối với dây cột. Tại thời điểm dây cột có giá trị 0 (được quét qua), một mã X1X0H3H2H1H0 tương ứng với phím được nhấn được sinh ra.



Hình 93

Ví dụ:

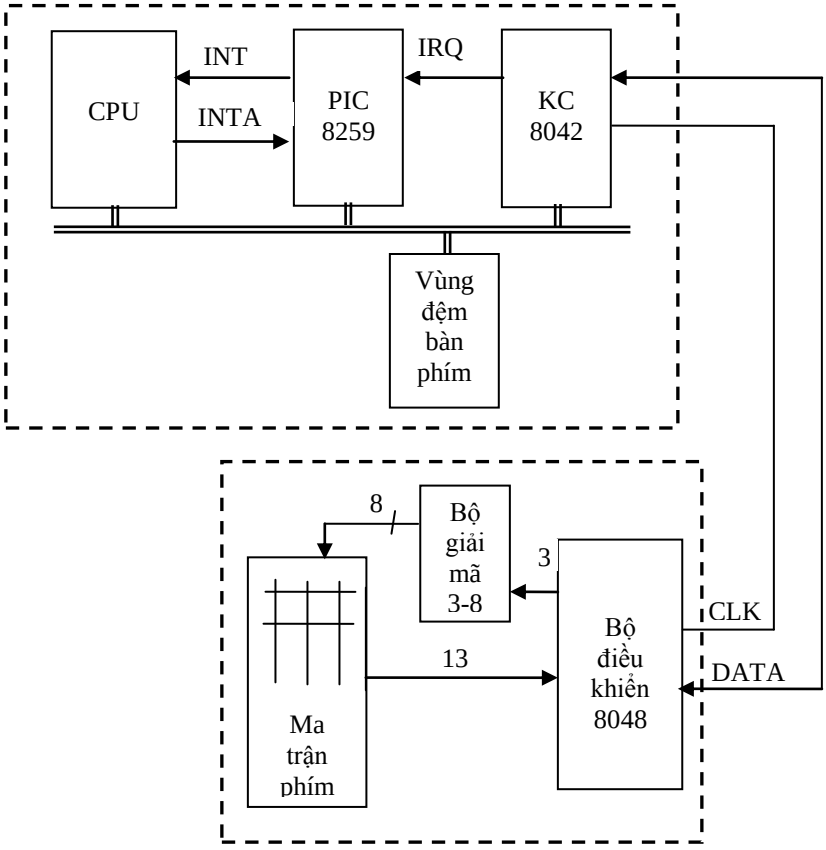
Nếu phím 6 được nhấn, tại thời điểm dây cột chứa phím 6 (dây C2) có mức logic 0 thì dây hàng chứa phím 6 (dây H2), do công tắc 6 đóng, cũng có mức logic 0. Mã đầu ra X1X0H3H2H1H0 = 101011.

Nếu phím A được nhấn, tại thời điểm dây cột chứa phím A (dây C2) có mức logic 0 thì dây hàng chứa phím A (dây H1), do công tắc A đóng, cũng có mức logic 0. Mã đầu ra X1X0H3H2H1H0 = 101101.

Nếu phím 7 được nhấn, tại thời điểm dây cột chứa phím 7 (dây C3) có mức logic 0 thì dây hàng chứa phím số 7 (dây H2), do công tắc 7 đóng, cũng có mức logic 0. Mã đầu ra X1X0H3H2H1H0 = 111011 v.v.

9.1.2. Hệ thống bàn phím của máy vi tính

Hệ thống bàn phím của máy vi tính gồm hai phần bàn phím và thiết bị giao diện bàn phím, được kết nối và trao đổi thông tin theo kiểu "chủ"- "thợ" (Hình 94).



Hình 94

Bàn phím là tổ hợp của ma trận 8x13 phím và mạch điều khiển 8048. Mạch 8048 là một hệ vi xử lý nhỏ được tích hợp trên một đơn chip. Mạch 8048 bao gồm CPU, bộ nhớ ROM chứa chương trình điều khiển quét và tạo mã phím, RAM chứa dữ liệu của chương trình điều khiển, hai cổng vào-ra song song P1và P2, một cổng giao tuần tự. Mạch 8048 tuần tự đưa mã nhị phân 3 bit ra cổng P2, qua bộ giải mã vào 3 - ra 8 tạo ra tín hiệu quét bàn phím. Tại thời điểm mã 3 bit được đưa ra, mạch 8048 thực hiện đọc tín hiệu 13 bit từ ma trận phím vào cổng P1, từ đây tạo ra mã phím (mã quét) của phím được nhấn. Khi phím được nhả, một mã phím (mã quét) cũng được tạo ra bằng cách cộng mã phím nhấn với 80H.

Mạch 8048, được nuôi bằng nguồn từ máy tính, thực hiện trao đổi thông tin với thiết bị giao diện bàn phím KC 8042 theo kiểu nối tiếp đồng bộ. KC 8042 có cấu trúc tương tự mạch 8048. Cơ chế truyền tin giữa mạch 8048 và

KC 8042 được tổ chức theo kiểu chủ-thợ, trong đó KC 8042 đóng vai trò "chủ", mạch 8048 đóng vai trò "thợ". Dữ liệu được truyền qua dây "DATA" theo kiểu nối tiếp đồng bộ. Dữ liệu được truyền trong một khung bao gồm: 1 bit START, 8 bit dữ liệu, 1 bit PARITY, 1 bit STOP. Quá trình trao đổi thông tin giữa 8048 và KC 8042 được đồng bộ bởi tín hiệu đồng bộ trên dây "CLK".

Quá trình truyền dữ liệu được thực hiện như sau:

Mạch 8048 luôn phải kiểm tra trạng thái truyền tin qua hai dây "DATA" và "CLOCK" trước khi phát đi mã phím. Khi KC 8042 đặt "DATA " = 0 và "CLK"=1 thì 8048 phải nhận các chỉ thị từ KC 8042. Khi KC 8042 đặt "DATA " = 1 và "CLOCK" = 1 thì 8048 được quyền truyền mã phím cho máy tính. Quá trình truyền dữ liệu được đồng bộ bằng dãy xung đồng bộ do 8048 phát ra trên dây "CLK".

Khi KC 8042 nhận được mã phím dạng nối tiếp, nó loại bỏ các bit tạo khung dữ liệu, chuyển mã phím vào thanh ghi tạm và phát ra yêu cầu ngắt IRQ1 cho hệ thống ngắt cứng. Hệ thống ngắt cứng sẽ kích hoạt chương trình phục vụ bàn phím 09H (chương trình phục vụ ngắt 09H) nằm ở BIOS. Chương trình phục vụ bàn phím 09H có chức năng dịch mã phím thành mã hai byte và chứa vào vùng đệm bàn phím.

Chương trình phục vụ bàn phím 09H trước hết kiểm tra (mã) các phím trượt (Shift, Alt, Ctrl) và các phím đặc biệt (ScrollLock, NumLock, CapsLock, Insert) trước khi dịch mã phím sang mã hai byte.

Mã hai byte được chương trình phục vụ bàn phím 09h tạo ra có cấu trúc tuỳ thuộc mã phím hoặc tổ hợp mã phím nhận được. Nếu chương trình ngắt 09H nhận được mã của phím ký tự thì byte thấp của mã hai byte chứa mã ASCII của ký tự tương ứng, byte cao chứa mã phím. Khi chương trình phục vụ bàn phím 09H nhận được mã các phím không phải là ký tự thì byte thấp của mã hai byte có giá trị 0, byte cao chứa mã phím điều khiển.

Vùng đệm bàn phím có kích thước 32 byte nằm trên bộ nhớ chính tại địa chỉ 0000H: 041EH. Trạng thái của các phím trượt và các phím đặc biệt được chứa ở hai ô nhớ 0000H: 0417H và 0000H: 0418H. Có thể truy nhập vùng đệm bàn phím để đọc thông tin về bàn phím nhờ chương trình phục vụ ngắt 16H của BIOS.

Chương trình phục vụ bàn phím 09h cũng xử lý các trường hợp đặc biệt như:

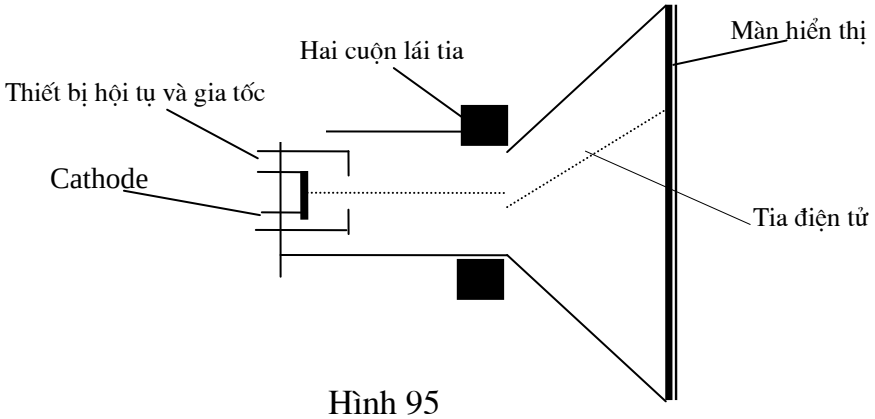
- khi phím được nhấn quá lâu (ví dụ quá 0. 5 giây) và KC 8042 không nhận được mã phím nhả, nó sẽ gửi ra cho đơn vị xử lý trung tâm mã của phím được nhấn.

- khi nhận được tổ hợp các phím Ctrl+Alt+Del nó sẽ khởi động lại máy tính hoặc kết thúc một nhiệm vụ.
- khi nhận được mã phím PrintScreen nó sẽ kích hoạt ngắt 05H của BIOS.
- khi nhận được mã phím Ctrl+Break nó sẽ kích hoạt ngắt 1BH của BIOS.

9.2. MÀN HÌNH

9.2.1. Màn hình CRT

Màn hình ống tia âm cực CRT (Cathode Ray Tube) là thiết bị hiển thị thông dụng hiện nay. Màn hình CRT có cấu tạo như sau (Hình 95):

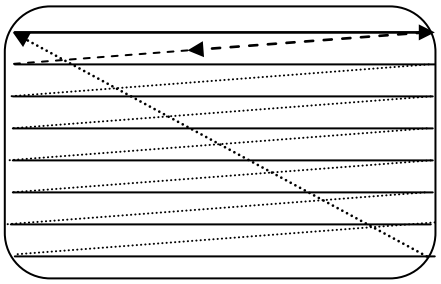


Hình 95

Màn hình CRT là một ống thủy tinh chân không với các bộ phận: cathode phát xạ điện tử, ống phóng tia điện tử, cuộn lái tia và màn hiển thị. Cathode bằng kim loại được nối với điện áp âm, được đốt nóng và tạo ra các điện tử tự do. Màn hiển thị được phủ một lớp chất liệu phát quang và dẫn điện, được nối với điện áp dương và đóng vai trò một anode. Dưới tác dụng của điện trường cường độ cao trong ống phóng, điện tử rời khỏi cathode, được hội tụ thành chùm tia hướng về phía màn hiển thị. Cuộn lái tia có tác dụng lái chùm tia điện tử dịch chuyển theo hai chiều dọc và ngang màn hình. Khi chùm tia điện tử đập vào màn hiển thị sẽ tạo nên một điểm phát sáng. Cường độ điểm sáng phụ thuộc vào cường độ chùm tia và chất liệu phát sáng. Khi chùm tia mất đi hoặc chuyển hướng thì điểm vẫn còn lưu sáng một khoảng thời gian ngắn sau đó, thời gian lưu sáng phụ thuộc vào chất liệu phát sáng và cường độ chùm tia.

Ảnh trên màn hình được tạo từ các điểm ảnh. Điểm ảnh được tạo ra khi cường độ chùm tia điện tử được tăng lên, điểm ảnh không xuất hiện khi chùm tia bị tắt đi. Các điểm ảnh được tạo theo từng dòng, từ trên xuống dưới. Một ảnh hoàn chỉnh được tạo ra trên màn hiển thị bởi các dòng chứa các điểm ảnh. Các điểm ảnh chỉ tồn tại trong một thời gian rất ngắn. Để có thể quan sát được

ảnh cần làm tươi các điểm ảnh theo một chu kỳ xác định. Các điểm ảnh được làm tươi theo từng dòng, bắt đầu từ dòng thứ nhất. Các dòng được làm tươi tuần tự từ trên xuống dưới. Khi dòng cuối cùng được quét xong, quá trình làm tươi được bắt đầu lại từ dòng đầu tiên (Hình 96).

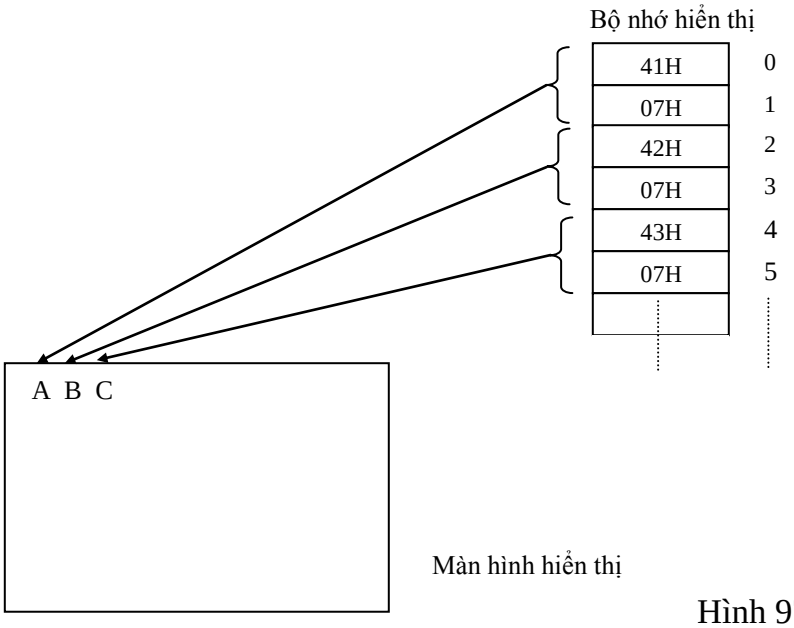


Hình 96

9.2.2. Hiển thị ở chế độ văn bản

Các thiết bị hiển thị được sử dụng ở máy vi tính PC đều là loại ánh xạ bộ nhớ. Bộ nhớ này được cả đơn vị xử lý trung tâm và thiết bị điều khiển màn hình cùng truy nhập và được gọi là bộ nhớ hiển thị. Thông tin cần hiển thị được đưa ra bộ nhớ hiển thị, thiết bị điều khiển màn hình liên tục đọc bộ nhớ này để đưa ra màn hình. Hình 97 minh họa nguyên tắc ánh xạ từ bộ nhớ hiển thị ra màn hình trong chế độ văn bản.

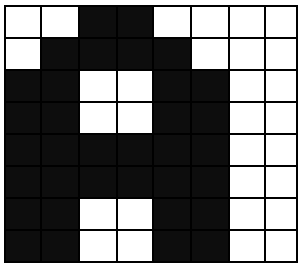
Mỗi một ký tự trên màn hình là một ánh xạ của một ô nhớ hai byte trong bộ nhớ hiển thị. Byte đầu chứa mã ASCII của ký tự, byte thứ hai chứa thuộc tính (màu nền, màu chữ, có/không nhấp nháy) của ký tự. Vị trí của mã ký tự trong bộ nhớ xác định vị trí ký tự trên màn hình. Mã ký tự tại ô nhớ đầu tiên trong bộ nhớ hiển thị (ví dụ: mã 41H) được ánh xạ thành ký tự (ký tự A) lên vị trí hàng đầu tiên và cột đầu tiên trên màn hiển thị, mã ký tự trong ô nhớ tiếp theo được ánh xạ thành ký tự ở vị trí tiếp theo v.v.



Hình 97

Phương pháp ánh xạ bộ nhớ cho phép chương trình máy tính có thể dễ dàng thay đổi nội dung màn hiển thị bằng cách thay đổi nội dung của bộ nhớ hiển thị.

Mỗi ký tự được hiển thị trên màn hình dưới dạng một ma trận 8x8 điểm ảnh sáng/tối như trên hình vẽ (Hình 98)



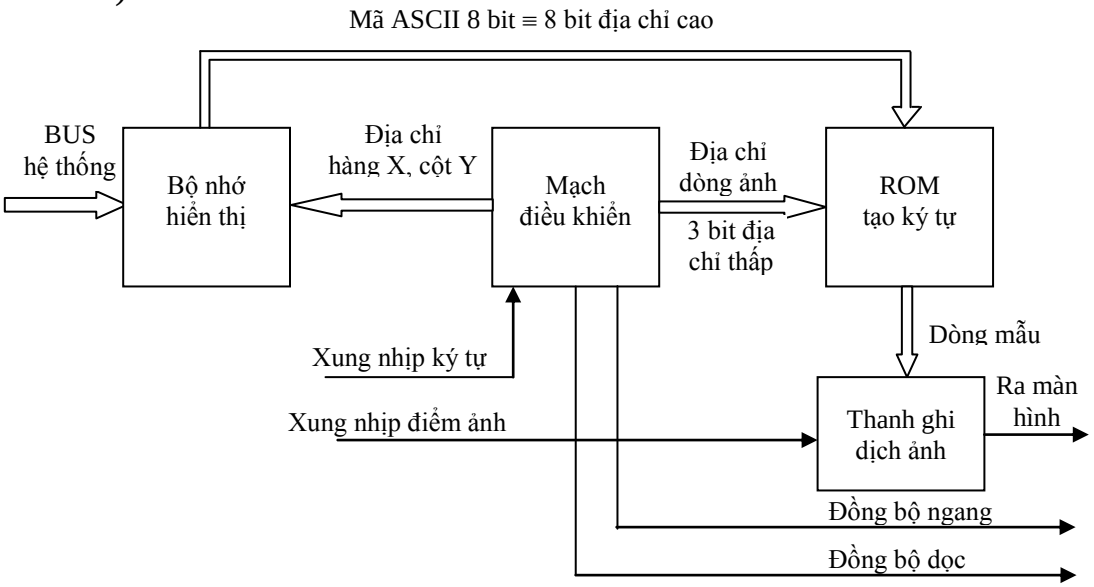
Hình 98

Phương pháp hiển thị ánh xạ bộ nhớ không hoàn toàn phù hợp với việc hiển thị các đối tượng có hình dạng không bình thường và chuyển động nhanh, đáp ứng thời gian thực bị chậm vì cần phải thao tác nhiều điểm ảnh để dịch chuyển đối tượng.

Bộ điều khiển màn hình

Thiết bị giao diện màn hình (bộ điều khiển màn hình) thực hiện việc chuyển mã ký tự trong bộ nhớ hiển thị thành ký tự hiện trên màn hình. Ở chế độ văn bản các mẫu ký tự chỉ được hiển thị ở các vị trí hàng và cột cố định (25 hàng x 80 cột).

Sơ đồ nguyên lý của thiết bị giao diện màn hình ở chế độ văn bản như sau (Hình 99).



Hình 99

Mỗi một ký tự trên màn hình được hiển thị dưới dạng tập hợp các dòng điểm ảnh. Bộ điều khiển màn hình có nhiệm vụ chuyển mỗi mã ASCII trong bộ nhớ hiển thị thành các mẫu điểm ảnh và đưa các mẫu ảnh lên màn hình. Điều này được thực hiện nhờ bộ ROM tạo ký tự. ROM tạo ký tự chứa các hộp mẫu ký tự, mỗi hộp mẫu ký tự có kích thước 8 byte mang thông tin về ma trận điểm ảnh của một ký tự (hình ảnh ký tự ở dạng tập hợp các điểm ảnh). Hình ảnh của ký tự được thể hiện qua các cách sắp đặt các bit 0 và 1, trong đó bit 1 thể hiện có điểm ảnh, bit 0 thể hiện không có điểm ảnh. Ví dụ hộp mẫu ký tự A có dạng sau:

00110000
01111000
11001100
11001100
11111100
11111100
11001100
11001100

Nếu cần hiển thị 256 ký tự ASCII cần một ROM 2 Kbyte, đủ chứa 256 hộp mẫu ký tự, mỗi hộp mẫu chiếm 8 ô nhớ liền nhau. Các hộp mẫu ký tự trong bộ ROM tạo ký tự được định vị bằng địa chỉ 11 bit, trong đó 8 bit địa chỉ cao xác định vị trí của hộp trong ROM, 3 bit địa chỉ thấp xác định vị trí của từng dòng mẫu điểm ảnh trong hộp đó. Các hộp mẫu ký tự được đặt trong ROM ký tự theo trật tự của bảng mã ASCII.

Nguyên lý hoạt động của thiết bị giao diện màn hình trong chế độ văn bản như sau.

Từng hàng ký tự được hiển thị lần lượt từ trên xuống dưới. Mỗi hàng ký tự gồm 8 dòng màn hình, mỗi dòng dòng màn hình được hiển thị lần lượt từ trên xuống dưới. Trên mỗi dòng màn hình, các mẫu dòng ảnh có cùng vị trí trong các hộp mẫu của ký tự thuộc một hàng được hiện lần lượt từ trái qua phải cho đến hết một dòng màn hình.

Giả sử cần hiển thị hai ký tự A và B tại các vị trí hàng 0 - cột 0 và hàng 0 - cột 1 trên màn hình. Mã ASCII của hai ký tự được đặt tại hai vị trí tương ứng trong bộ nhớ hiển thị (xem hình vẽ ở mục 9. 2. 2).

Bộ điều khiển gửi địa chỉ hàng và cột màn hình cho bộ nhớ hiển thị (hàng=0, cột=0). Bộ nhớ hiển thị giải mã này thành địa chỉ ô nhớ của bộ nhớ hiển thị, qua đó truy cập ô nhớ chứa mã ASCII của ký tự, xuất mã ASCII của ký tự (ký tự A) cho ROM. Mã ASCII của ký tự mang thông tin về địa chỉ (vị trí) của hộp mẫu ký tự trong ROM (mã này đóng vai trò là 8 bit địa chỉ cao của địa chỉ 11 bit của ROM ký tự). Tại cùng thời điểm này bộ điều khiển gửi địa chỉ xác định vị trí của dòng mẫu điểm ảnh trong hộp mẫu (đóng vai trò là 3

bit địa chỉ thấp của địa chỉ 11 bit), trường hợp trong ví dụ này là địa chỉ (vị trí) của dòng mẫu điểm ảnh 0, cho ROM. Hai phần địa chỉ (8 bit địa chỉ cao và 3 bit địa chỉ thấp) này được kết hợp lại tạo thành địa chỉ 11 bit, xác định địa chỉ của ô nhớ dòng mẫu điểm ảnh, qua đó cho phép truy nhập vào dòng mẫu điểm ảnh 0 (dòng mẫu đầu tiên) của ký tự (ký tự A) trong ROM, dòng mẫu này được xuất ra thanh ghi dịch ảnh. Từ thanh ghi dịch ảnh, từng bit mẫu ảnh tuần tự được đưa ra màn hình.

Khi tất cả các bit mẫu ảnh từ thanh ghi dịch được đẩy ra màn hình, bộ điều khiển tiếp tục gửi địa chỉ hàng-cột (hàng=0, cột=1) cho bộ nhớ hiển thị để truy cập mã ký tự tiếp theo (ký tự B) trong bộ nhớ hiển thị. Bộ nhớ hiển thị gửi mã ASCII của ký tự (ký tự B) cho ROM. Cùng trong thời gian này, bộ điều khiển vẫn gửi cùng một địa chỉ dòng mẫu điểm ảnh (dòng mẫu điểm ảnh 0) cho ROM. Dòng mẫu điểm ảnh 0 (dòng mẫu đầu tiên) của ký tự tiếp theo (ký tự B) được xuất ra thanh ghi dịch ảnh. Tương tự như thế các dòng mẫu điểm ảnh đầu tiên của tất cả các ký tự thuộc cùng một hàng ký tự được hiển thị, cho đến ký tự cuối cùng trên hàng. Dòng ảnh đầu tiên của tất cả các ký tự trên cùng một hàng được hiển thị.

Bộ điều khiển tiếp tục gửi địa chỉ hàng-cột (hàng=0, cột=0) đến bộ nhớ hiển thị dể truy cập lại từ đầu các mã ký tự thuộc cùng một hàng, nhưng phát ra địa chỉ dòng mẫu điểm ảnh bây giờ là 1(dòng mẫu điểm ảnh 1) cho ROM. Bộ nhớ hiển thị gửi mã ASCII của ký tự A cho ROM, 8 bit mã ASCII của ký tự A bây giờ được kết hợp với 3 bit địa chỉ thấp (có giá trị là 1) thành địa chỉ của dòng mẫu 1 trong hộp mẫu này. ROM xuất ra dòng mẫu điểm ảnh 1 của ký tự A cho màn hình. Bộ điều khiển tiếp tục gửi địa chỉ hàng-cột (hàng=0, cột=1) đến bộ nhớ hiển thị và dòng 1 của ký tự B được xuất ra theo cách tương tự. Các mẫu điểm ảnh tiếp theo của các ký tự tiếp theo trong bộ nhớ hiển thị lần lượt được hiển thị lên màn hình theo cùng cách tương tự, cho đến khi tất cả các dòng điểm ảnh của hàng văn bản đầu tiên (hàng 0) được hiển thị trên màn hình.

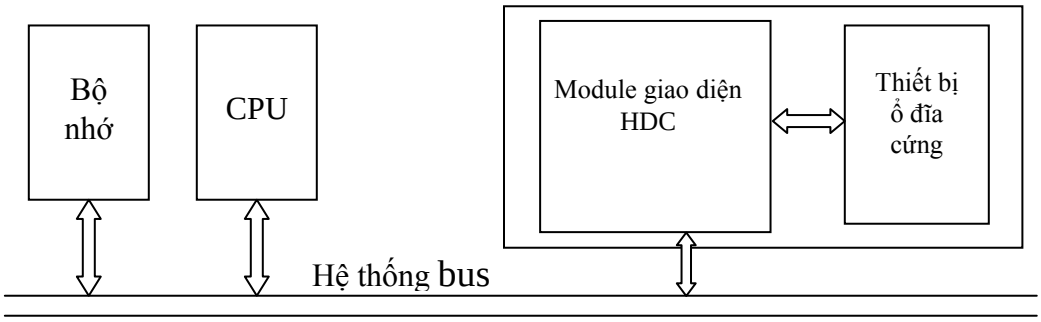
Các hàng ký tự tiếp theo cũng được hiển thị theo phương pháp nói trên.

Trên thực tế hoạt động của bộ điều khiển màn hình phức tạp hơn. Bộ điều khiển màn hình phải có khả năng hiển thị ở chế độ đồ họa. Bộ điều khiển màn hình phải theo dõi thông tin về thuộc tính của ký tự hiển thị, phải tạo ra điểm nháy. Bộ điều khiển màn hình cũng phải tạo ra hai tín hiệu đồng bộ ảnh ngang - dọc và làm tươi màn hình. Tần số làm tươi tối thiểu là 50 Hz (tái tạo 50 khung hình trong 1 giây).

Chương 10
TỔ CHỨC LƯU TRỮ THÔNG TIN TRÊN ĐĨA TỪ

10.1. THIẾT BỊ ĐĨA CỨNG VÀ GIAO DIỆN ATA IDE

Trong hệ thống máy tính, ở cấp độ vật lý thiết bị đĩa cứng là một thiết bị ngoại vi và được đơn vị xử lý trung tâm giao tiếp và truy cập như một thiết bị ngoại vi. Cấu trúc chung của hệ thống vào-ra này như sau (Hình 100).



Hình 100

Thiết bị đĩa cứng là một thiết bị tích hợp gồm phần ổ đĩa và phần thiết bị điện tử điều khiển ổ đĩa HDC (Hard Disk Controller) được gắn ngay trên ổ đĩa. Loại thiết bị này được gọi là thiết bị IDE (Integrated Drive Electronics).

Chức năng của module giao diện đĩa cứng HDC là thực hiện truyền dữ liệu từ hệ thống máy tính đến ổ đĩa và nhận dữ liệu từ ổ đĩa cung cấp cho hệ thống máy tính. Tốc độ truyền dữ liệu với ổ cứng là một thông số quan trọng và ảnh hưởng đến tốc độ hoạt động của máy tính. Tốc độ truyền dữ liệu phụ thuộc nhiều vào cách giao diện giữa ổ cứng và hệ thống. Việc đặt module giao diện, bao gồm các thiết bị tạo các tín hiệu điều khiển ổ đĩa và thiết bị mã hoá/giải mã dạng số/tương tự, ngay trong ổ đĩa làm cho ổ IDE có độ tin cậy cao hơn so với kiểu module giao diện nằm độc lập trên bản mạch chủ và được nối với ổ đĩa bằng cáp tín hiệu (như ở thiết bị đĩa mềm). Việc tích hợp thiết bị giao diện trên ổ đĩa và thực hiện giao diện với hệ thống bus của máy tính theo chuẩn IDE còn cho phép các nhà sản xuất ổ đĩa có thể độc lập thiết kế phát triển thiết bị đĩa cứng có các tính năng ngày càng mạnh hơn. Trước kia có

nhiều chuẩn giao diện IDE được xây dựng. Ngày nay chỉ còn chuẩn ATA IDE được dùng. Thuật ngữ ATA hoặc IDE khi được dùng là cùng chỉ một kiểu giao diện và có thể dùng thay thế cho nhau. Đã có nhiều phiên bản cải tiến của chuẩn giao diện ATA được thiết kế để có tốc độ truyền dữ liệu nhanh hơn và khả năng kết nối mạnh hơn như: ATA 2 (Fast-ATA hoặc EIDE), ATA 3, ATA/ATAPI 4 (Ultra-ATA/33, UDMA/33) đến ATA/ATAPI 7 (Ultra-ATA/133, UDMA/133).

Một chuẩn giao diện kết nối khác cũng đã được thiết kế để thay thế chuẩn giao diện ATA cũ (bây giờ được gọi là PATA-Parallel ATA) là chuẩn SATA (Serial ATA). Chuẩn SATA vẫn sử dụng các lệnh điều khiển mức thấp giống như PATA, nhưng kỹ thuật truyền dữ liệu là theo kiểu nối tiếp. Chuẩn SATA có một số đặc điểm ưu việt hơn PATA như: số đường dây trong cáp nối ít hơn, tốc độ truyền dữ liệu hiệu dụng (MB/giây) nhanh hơn PATA và cho phép kết nối nóng. Bộ điều khiển SATA hiện đại sử dụng giao diện điều khiển AHCI (Advanced Host Cotroller Interface). Nếu bo mạch chủ không hỗ trợ AHCI thì bộ điều khiển SATA sẽ hoạt động ở chế độ "phỏng IDE".

Thiết bị IDE có một tập các thanh ghi cho phép điều khiển và giao diện với ổ đĩa.

Thanh ghi dữ liệu:

D7	D6	D5	D4	D3	D2	D1	D0

Thanh ghi dữ liệu được dùng để gửi lệnh cho thiết bị giao diện, còn các dữ liệu ghi/đọc được chuyển theo chế độ DMA.

Thanh ghi điều khiển chọn điều khiển ổ đĩa và đầu từ:

1	0	1	DRV	H3	H2	H1	H0

DRV chọn ổ đĩa
H3 - H0 chọn đầu từ

Thanh ghi điều khiển phụ:

X	X	X	X	X	RST	INT	X

RST khởi động mềm
INT cho phép báo ngắt. INT=1 cấm thiết bị giao diện báo ngắt.

Ba thanh ghi số cung, địa chỉ cung và địa chỉ rãnh cần ghi/đọc.

Thanh ghi trạng thái:

BSY	RDY	WFT	SK	DTRQ	COR	IDX	ERR

BSY ổ đĩa bận

RDY ổ đĩa sẵn sàng
WFT lỗi ghi
SK trạng thái dịch chuyển đầu từ
DTRQ yêu cầu truyền dữ liệu
COR lỗi dữ liệu chữa được
IDX index
ERR có lỗi. Mã lỗi nằm ở thanh ghi trạng thái lỗi.

Thanh ghi trạng thái lỗi:

NDM	NTR	ABT	X	NID	X	UC	BK

NMD không tìm được vùng DM
NTR không tìm được rãnh
ABT lệnh bị ngắt
NID không tìm được vùng ID
UC lỗi dữ liệu không khôi phục
BK cung có lỗi

Quá trình thực hiện một lệnh trải qua 3 giai đoạn:

- Giai đoạn nạp lệnh: HDC nhận lệnh và các thông số điều khiển từ đơn vị xử lý trung tâm, giải mã để có các thông tin về những thao tác cần thiết khi thực thi lệnh này.
- Giai đoạn thực hiện: HDC thực hiện các thao tác thực thi lệnh.
- Giai đoạn kết quả: sau khi lệnh được thực thi, trạng thái thực hiện lệnh được chuyển về cho đơn vị xử lý trung tâm.

10.2. TỔ CHỨC LƯU TRỮ THÔNG TIN TRÊN ĐĨA TỪ Ở MỨC LOGIC

Địa chỉ vật lý (địa chỉ mặt, rãnh và cung) của một cung vật lý (xem chương 4) là kiểu địa chỉ không tuyến tính và không thích hợp cho việc quản lý và sử dụng cung với tư cách là một đơn vị lưu trữ thông tin của bộ nhớ máy tính. Việc dùng đĩa từ như là một hệ thống lưu trữ tập tin (file) của máy tính đòi hỏi phải tổ chức lại cách quản lý các cung nói riêng và quản lý việc lưu trữ thông tin trên đĩa từ nói chung.

Các hệ điều hành khác nhau có các cách tổ chức lưu trữ và quản lý tập tin trên đĩa từ khác nhau. Hệ điều hành DOS, Windows 9x và Windows XP sử dụng cách tổ chức lưu trữ và quản lý tập tin (tổ chức hệ thống tập tin) kiểu FAT (File Allocation Table), trong khi các hệ điều hành khác như Windows NT, Vista, Windows 7 lại sử dụng hệ thống tập tin NTFS (NT File System).

10.2.1. Bảng phân vùng

Một ổ cứng vật lý có thể được phân chia thành nhiều phân vùng (partition) và từ mỗi phân vùng tạo ra được một volume tách biệt (ổ logic). Việc tạo ra các phân vùng cho phép ta có thể cài đặt nhiều hệ điều hành và các hệ thống tập tin khác nhau trong cùng một ổ cứng vật lý. Các hệ thống tập tin khác nhau (ví dụ như hệ tập tin FAT, FAT32, NTFS v.v.) sẽ sử dụng các phương pháp riêng để phân chia và quản lý không gian lưu trữ tập tin.

Mỗi ổ cứng phải có ít nhất 1 phân vùng và có thể có tối đa 4 phân vùng chính. Quá trình tạo phân vùng có thể thực hiện bằng phần mềm của hệ điều hành hoặc bằng nhiều phầm mềm khác.

Thông tin về cách phân chia ổ đĩa cứng thành các phân vùng được chứa trong *Bảng phân vùng* (Partition Table). Hệ điều hành sử dụng một vùng riêng và cố định trên ổ đĩa cứng để chứa Bảng phân vùng, đó là cung vật lý 1 trên rãnh 0 của mặt 0. Cung vật lý đầu tiên của ổ đĩa cứng (mặt 0, rãnh 0, cung 1) không chứa cung khởi động volume (Volume Boot Record) mà chứa cung khởi động chủ (Master Boot Record- MBR).

- Cung khởi động chủ có cấu trúc như sau (Bảng 19).

Bảng 19

Địa chỉ offset (Hexadecimal)	**NỘI DUNG**
000	Chương trình đọc cung khởi động
1BE	Bảng phân vùng chính
1FE	Chữ ký AA55H

Cung khởi động chủ chứa chương trình đọc Bảng phân vùng chính và cung khởi động volume. Chức năng của chương trình đọc cung khởi động là chuyển điều khiển hệ thống đến chương trình khởi động hệ điều hành, chương trình này nằm trong cung khởi động volume của phân vùng khởi động.

Bảng phân vùng chính chứa các *lối vào phân vùng* (Partition Entry). Bảng phân vùng chính có ít nhất 1 lối vào phân vùng và nhiều nhất 4 lối vào phân vùng. Lối vào phân vùng chứa thông tin về vị trí và kích thước của phân vùng trong ổ đĩa cứng.

- Bảng phân vùng chính có cấu trúc như sau (Bảng 20).

Bảng 20

Địa chỉ offset (Hexadecimal)	**NỘI DUNG**	**Kích thước** (byte)
1BE	Lối vào phân vùng 1	16
1CE	Lối vào phân vùng 2	16

1DE	Lối vào phân vùng 3	16
1EE	Lối vào phân vùng 4	16

Mỗi một lối vào phân vùng mang thông tin về phân vùng tương ứng: vị trí (mặt, rãnh, cung) bắt đầu phân vùng, vị trí (mặt, rãnh, cung) kết thúc phân vùng, kích thước phân vùng (tính theo đơn vị cung) v.v.

- Cấu trúc của một lối vào phân vùng (Bảng 21).

Bảng 21

Địa chỉ offset (Hexadecimal)	**NỘI DUNG**	**Kích thước** (byte)
00	Chỉ thị khởi động	1
01	Địa chỉ đầu phân vùng	3
04	Chỉ thị hệ thống	1
05	Địa chỉ cuối phân vùng	3
08	Số lượng cung trước phân vùng	4
0c	Số lượng cung trong phân vùng	4

- Chỉ thị khởi động:

  80H Phân vùng khởi động (Khởi động hệ điều hành từ phân vùng này).

  00H Phân vùng không tíchcực.

- Chỉ thị hệ thống: cho biết loại hệ thống tập tin được sử dụng trong phân vùng.

  01H FAT 12

  06H, 0EH FAT 16

  07H NTFS

  0BH FAT 32

- Khuôn dạng của trường địa chỉ đầu/địa chỉ cuối phân vùng:

Mặt								Cung		Rãnh													
H7	H6	H5	H4	H3	H2	H1	H0	C9	C8	S5	S4	S3	S2	S1	S0	C7	C6	C5	C4	C3	C2	C1	C0

Địa chỉ mặt là một số nhị phân 8 bit : H7 – H0. Số mặt tối đa có thể là 256
Địa chỉ cung là một số nhị phân 6 bit : S5 – S0. Số cung tối đa có thể là 63
Địa chỉ rãnh là một số nhị phân 10 bit : C9 – C0. Số rãnh tối đa có thể là 1024

Nội dung một lối vào phân vùng có thể như sau:

00 H	01 H	02 H	03 H	04 H	05 H	06 H	07 H	08 H	09 H	0A H	0B H	0C H	0D H	0E H	0F H
80 H	01 H	01 H	00 H	06 H	04 H	E2 H	C4 H	00 H	00 H	00 H	22 H	90 H	32 H	02 H	00 H

Phân vùng này là phân vùng khởi động.

Địa chỉ đầu phân vùng: Mặt 1, rãnh 0, cung vật lý 1.

Địa chỉ cuối phân vùng: Mặt 4, rãnh 964, cung vật lý 34

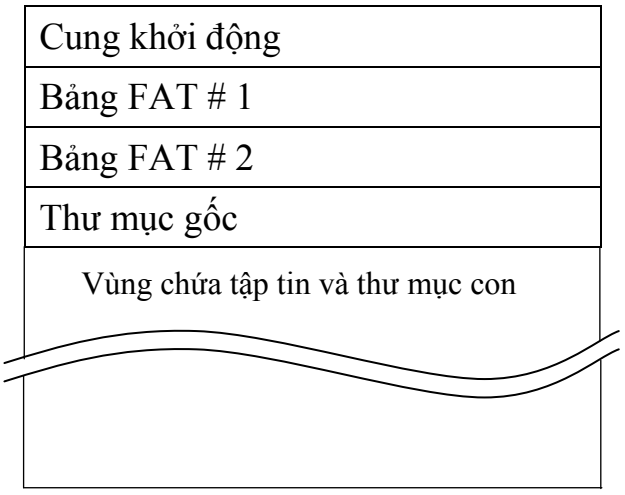
Kích thước phân vùng: 023290h = 164016 cung

Tại mỗi phân vùng có thể thực hiện định dạng (mức cao) để tạo volume và cài đặt hệ điều hành trên các volume đó.

10.2.2. Hệ thống tập tin FAT

Trong cách tổ chức hệ thống tập tin FAT, bộ nhớ ngoài được coi là tập hợp của các volume, một volume có thể là toàn bộ một đơn vị vật lý (như đĩa mềm) hoặc là một phần của ổ cứng. Mỗi volume được xem là một tập hợp liên tục các cung logic. Các cung logic được đánh số thứ tự, bắt đầu là cung logic 0, tiếp đó là cung logic 1 v.v. lần lượt tăng dần cho đến hết volume. Việc tổ chức lại các thiết bị đĩa từ theo volume làm cho việc quản lý các thiết bị này trở nên thống nhất, không phụ thuộc vào cấu trúc vật lý cụ thể của các ổ đĩa. Quá trình thao tác với tập tin lúc này phải thực hiện qua hai bước: trước hết yêu cầu ghi/đọc tập tin từ ứng dụng được chuyển thành ghi/đọc các cung logic (có địa chỉ tuyến tính), sau đó các địa chỉ tuyến tính của các cung được chuyển thành địa chỉ vật lý (địa chỉ mặt, rãnh, cung). Quá trình ghi/đọc tập tin thật sự được thực hiện theo địa chỉ vật lý này.

Mỗi volume được chia ra thành các vùng, mỗi vùng chứa một loại thông tin về cách tổ chức lưu trữ trên volume. Cấu trúc của một volume như sau (Hình 101).



Hình 101

10.2.2.1 Cung khởi động volume (Volume Boot Sector)

Cung khởi động volume (gọi tắt là cung khởi động) nằm ở cung logic 0 của một volume. Đối với đĩa mềm cung logic 0 có địa chỉ vật lý là: mặt 0, rãnh 0, cung 1. Đối với đĩa cứng cung logic 0 của volume đầu tiên có địa chỉ vật lý là: mặt 1, rãnh 0, cung 1.

Cung khởi động volume chứa thông tin về cách phân vùng trên volume, về tổ chức volume ở mức vật lý và có thể chứa chương trình khởi động hệ điều hành (Bảng 22).

Bảng 22

Địa chỉ offset (Hexadecimal)	**Nội dung**	**Kích thước** (byte)
00	Lệnh nhảy đến chương trình khởi động	3
03	Tên nhà sản xuất và thế hệ của HĐH	8
0B	Số lượng byte / một cung	2
0D	Số lượng cung / một liên cung	1
0E	Số lượng cung trước bảng FAT	2
10	Số lượng bảng FAT	1
11	Số lượng lối vào thư mục trong thư mục gốc	2
13	Số lượng cung trong volume	2
15	Byte mô tả loại đĩa	1
16	Số lượng cung / một bảng FAT	2
18	Số lượng cung / một rãnh	2
1A	Số lượng đầu từ	2
1C	Số lượng cung ẩn	2
1E	Số lượng cung trong volume, nếu kích thước volume lớn hơn 32 Mb	4
22	Số thứ tự ổ đĩa	1
23 - 33	Dự phòng cho các thông tin phụ	17
34	Nhận dạng hệ FAT	8
3C – 1FF	Chương trình khởi động HDH	452

Cung khởi động được kết thúc bằng 2 byte chữ ký AA55H. Khi khởi động máy tính, cung khởi động được nạp vào địa chỉ 0000H: 7C00H. BIOS thực hiện kiểm tra chữ ký, nếu không tìm thấy hai byte chữ ký thì sẽ báo lỗi, nếu tìm thấy chữ ký BIOS sẽ cho thực hiện chương trình từ địa chỉ 0000H:

7C00H. Đó cũng là lý do vì sao byte đầu tiên của cung khởi động luôn chứa mã lệnh nhảy không điều kiện.

Cung khởi động chứa các dữ liệu mà từ đó có thể tính ra được vị trí bắt đầu và kích thước (theo đơn vị cung) của các vùng thông tin trên volume.

10.2.2.2 Bảng FAT

Hệ điều hành lưu trữ tập tin trong vùng chứa tập tin. Vùng chứa tập tin nằm sau thư mục gốc và chiếm toàn bộ không gian còn lại của volume. Vùng chứa tập tin được coi là tập hợp của các liên cung (cluster). Liên cung là tập hợp của một hoặc nhiều cung liền nhau. Liên cung là đơn vị lưu trữ thông tin nhỏ nhất mà hệ điều hành dùng để lưu trữ tập tin. Các liên cung được đánh số thứ tự bắt đầu từ 2 và tăng dần đến hết volume.

Bảng FAT nằm ngay sau cung khởi động. Mỗi volume thường có hai bảng FAT #1 và #2, nội dung hai bảng này giống hệt nhau.

Bảng FAT được dùng để quản lý các liên cung. Bảng FAT chứa các lối vào (entry). Kích thước của một lối vào có thể là 12, 16 hoặc 32 bit, tuỳ thuộc vào kích thước của volume và cách tổ chức hệ thống tập tin FAT. Hai lối vào đầu tiên trong bảng FAT được dùng để chỉ ra dạng tổ chức của đĩa. Số lượng lối vào còn lại đúng bằng số lượng các liên cung. Các lối vào này cũng được đánh số thứ tự bắt đầu từ 2 và tăng dần cho đến tận lối vào cuối cùng trong bảng FAT. Mỗi một lối vào, bắt đầu từ lối vào số 2, chứa một thông tin về trạng thái của liên cung có số thứ tự tương ứng. Thông tin về trạng thái của liên cung được thể hiện dưới dạng mã nhị phân như sau (Bảng 23).

Bảng 23

Nội dung của lối vào (Hexadecimal)	**Trạng thái liên cung tương ứng**
00000000	Liên cung rỗng
FFFFFFF0 – FFFFFFF6	Liên cung dự phòng
FFFFFFF7	Liên cung hỏng
FFFFFFF8 – FFFFFFFF	Liên cung cuối cùng của một tập tin tin nào đó
XXXXXXXX	Liên cung tương ứng với lối vào này đang chứa tập tin. Con số XXXXXXXX xác định địa chỉ liên cung tiếp theo của tập tin.

Bảng FAT đóng vai trò một bản đồ về trạng thái các liên cung. Từ bảng FAT có thể tìm ra được một chuỗi danh sách các liên cung thuộc một tập tin nào đó, nhưng còn thiếu thông tin về liên cung đầu tiên của tập tin. Thông tin này gắn với một tập tin cụ thể và nằm ở thư mục.

10.2.2.3 Thư mục gốc (Root Directory)

Mỗi volume có một thư mục gốc. Thư mục gốc nằm sau bảng FAT #2. Thư mục gốc chứa các lối vào thư mục (Directory Entry), mỗi lối vào thư mục có kích thước 32 byte. Thư mục gốc có kích thước hạn chế được xác định trước.

Lối vào thư mục chứa các thông tin (tên tập tin, thuộc tính, kích thước v.v.) của một tập tin cụ thể đang được lưu trữ trên volume và số thứ tự liên cung đầu tiên của tập tin đó.

Lối vào thư mục có cấu trúc như sau (Bảng 24).

Bảng 24

Địa chỉ offset (Hexadecimal)	**Nội dung**	**Kích thước**
00	Tên tập tin hoặc tên thư mục con	8
08	Phần tên mở rộng	3
0B	Thuộc tính tập tin	1
0C – 15	Dự phòng (Với FAT32: hai ô 14h và 15h chứa hai byte cao số thứ tự của liên cung đầu tiên)	10
16	Thời gian tạo hoặc cập nhật tập tin tin	2
18	Ngày tạo hoặc cập nhật tập tin	2
1A	Số thứ tự liên cung đầu tiên của tập	2
1C	Kích thước tập tin	4

- Byte đầu tiên của phần tên tập tin có thể chứa các thông tin đặc biệt sau:

  00H: báo thư mục kết thúc ở đây.

  e5H: tập tin bị xoá

  2eH: lối vào thư mục của thư mục hiện thời. Lối vào này chứa thông tin về vị trí và kích thước của chính thư mục đang chứa lối vào này.

  2eH 2eH: lối vào thư mục mẹ của thư mục hiện thời. Lối vào này chứa thông tin về vị trí và kích thước của thư mục mẹ của thư mục đang chứa lối vào này. Nhờ thông tin này mà từ thư mục hiện thời ta có thể chuyển về thư mục mẹ.

- Trường thời gian được mã hoá như sau (Bảng 25):

Bảng 25

Vị trí bit	**Nội dung**
0 – 4	Số nhị phân của giây chẵn: 0 – 29 tương ứng giây thứ 0 – 58
5 – 10	Số nhị phân của phút: 0- 59
11 – 15	Số nhị phân của giờ : 0- 23

- Trường thời ngày được mã hoá như sau (Bảng 26):

Bảng 26

Vị trí bit	**Nội dung**
0 – 4	Số nhị phân của ngày: 1 – 31
5 – 8	Số nhị phân của tháng: 1- 12
9 – 15	Số nhị phân của năm

- Trường thuộc tính tập tin, khi bit được đặt bằng 1 thì mang thông tin như sau (Bảng 27).

Bảng 27

Vị trí bit	**Nội dung**
0	Chỉ đọc
1	ẩn
2	Hệ thống
3	Nhãn Volume
4	Thư mục con
5	Lưu trữ. Bit này bằng 1 khi tập tin bị thay đổi nội dung
6	Dự phòng
7	Dự phòng

Cấu trúc của thư mục con giống như thư mục gốc. Thư mục con nằm ở vùng chứa tập tin và thư mục con. Thư mục con được quản lý như một tập tin thông thường và có kích thước không hạn chế.

10.2.2.4 Vùng chứa tập tin và thư mục con

Vùng chứa tập tin và thư mục con là toàn bộ vùng còn lại nằm sau thư mục gốc. Vùng này được coi là tập hợp của các liên cung. Các liên cung được đánh số thứ tự bắt đầu từ 2 tăng dần đến liên cung cuối cùng trong volume. Trạng thái của các liên cung được phản ảnh bởi lối vào tương ứng trong bảng FAT. Các thư mục con nằm ở vùng này. Thư mục con được quản lý như một tập tin thông thường và có kích thước không hạn chế.

Một tập tin được lưu trữ trên một hoặc nhiều liên cung khác nhau, tuỳ thuộc kích thước tập tin. Các liên cung chứa dữ liệu của một tập tin không nhất thiết phải liền kề nhau.

Nếu volume chứa hệ thống khởi động hệ điều hành thì các liên cung đầu tiên của vùng này được dành để chứa các tập tin hệ thống của hệ điều hành.

10.2.2.5 Mối quan hệ giữa thư mục, bảng FAT và liên cung

Mối quan hệ giữa bảng FAT và liên cung đã được trình bày ở mục về bảng FAT. Mối quan hệ giữa thư mục, bảng FAT và liên cung trong việc quản lý một tập tin được minh hoạ qua ví dụ sau (Hình 102).

Giả sử có tập tin tên MYFILE. DAT, tên tập tin nằm tại thư mục gốc. Trong thư mục gốc ta có thể tìm thấy và đọc được một lối vào thư mục có nội dung sau:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4DH	59H	46H	49H	4CH	45H	20H	20H	44H	41H	54H	20H	00H	00H	00H	00H
00H	00H	00H	00H	00H	00H	98H	9AH	9bH	0AH	03H	00H	42H	05H	00H	00H

Số thứ tự liên cung đầu tiên của tập tin — Kích thước tập tin

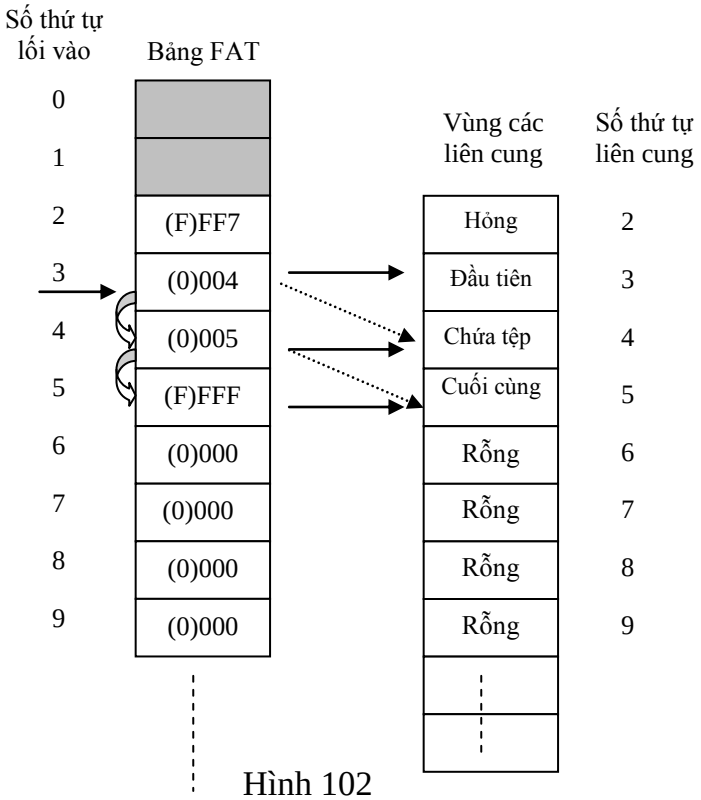
Các byte 00H - 0AH: mã ASCII của tên tập tin
MYFILE. DAT

Byte 0BH: 20H – tập tin lưu trữ

Byte 1AH-1BH: (0)003H – Số thứ tự của liên cung đầu tiên của tập tin. Lối vào đầu tiên của tập tin trên bảng FAT cũng là (0)003H.

Byte 1CH – 1FH: kích thước tập tin 0542H ≡ 1346 byte.

Quá trình truy cập tập tin ở mức thấp bắt đầu từ việc tìm kiếm thư mục chứa tên tập tin. Trong thư mục sẽ tìm được lối vào thư mục có chứa tên tập tin cần truy cập, ví dụ lối vào thư mục có chứa tên tập tin MYFILE. DAT. Từ lối vào thư mục này tìm được địa chỉ (số thứ tự) liên cung đầu tiên của tập tin MYFILE. DAT, ví dụ (0)003H. Con số (0)003H cũng là số thứ tự của lối vào đầu tiên của tập tin này trong bảng FAT. Thao tác tiếp theo là tìm đến bảng FAT. Lối vào đầu tiên cần đọc là (0)003H. Lối vào (0)003H chứa con số (0)004H, con số này phản ánh hai thông tin: một là liên cung số (0)003H đang chứa một phần của tập tin, hai là liên cung tiếp theo của tập tin là (0)004H. Từ đây tiếp tục truy nhập và đọc lối vào số (0)004H v.v. cho đến khi gặp lối vào chứa giá trị (F)FFFH thì dừng tìm kiếm. Ví dụ ở đây ta có danh sách các liên cung chứa tập tin là (0)003H, (0)004H, (0)005H.



Hình 102

Tổng quát mà nói, từ lối vào tại vị trí I ta đọc được nội dung J, nếu J thuộc khoảng giá trị (0)002H - (F)FEFH thì nó phản ảnh hai thông tin: liên cung thứ I đang chứa tập tin và liên cung thứ J là liên cung tiếp theo của tập tin. Quá trình tìm kiếm danh sách các liên cung chứa tập tin dừng khi gặp lối vào chứa giá trị (F)FFFH. Liên cung tương ứng lối vào này là liên cung cuối cùng của tập tin. Trên thực tế không phải bao giờ cũng nhận được chuỗi danh sách các liên cung liền kề nhau.

Để truy nhập tập tin trên đĩa cần phải chuyển danh sách các liên cung chứa tập tin thành danh sách các cung logic. Công thức chuyển đổi địa chỉ (số thứ tự) liên cung thành địa chỉ (số thứ tự) cung logic như sau:

Địa chỉ cung logic=(Địa chỉ liên cung–2)(Số lượng cung trên một liên cung) + Địa chỉ cung logic đầu tiên của vùng dữ liệu.*

Quá trình truy nhập cung khởi động, bảng FAT và tập tin có thể thực hiện nhờ các ngắt 25H và 26H. Khi truy nhập tập tin và đọc vào bộ nhớ cần lưu ý dữ liệu của tập tin chỉ chiếm một phần của liên cung cuối cùng chứ không phải toàn bộ liên cung này. Thông tin về kích thước tập tin giúp xác định chính xác lượng dữ liệu thật của tập tin trên liên cung cuối cùng.

10.2.3. Hệ thống tập tin NTFS

NTFS là hệ thống tổ chức và quản lý tập tin (hệ thống tập tin) hiệu năng cao và có tự sửa lỗi, phù hợp cho các hệ điều hành Windows XP, NT, 2000, Vista và Windows 7. Hệ thống tập tin NTFS hỗ trợ bảo mật tập tin qua phân quyền và mã hóa, hỗ trợ truy cập và xử lý tập tin nén, hỗ trợ đặt hạn ngạch và kiểm soát hạn ngạch sử dụng tập tin. Hệ thống tập tin NTFS có khả năng quản lý volume kích thước lớn và rất lớn. Thuộc tính mới quan trọng nhất của NTFS so với FAT là khả năng mã hóa tập tin và thư mục để bảo vệ các dữ liệu nhậy cảm.

NTFS sử dụng Bảng quản lý tập tin MFT (Master File Table) để quản lý tập tin thay cho FAT. Một volume khi được tạo khuôn dạng (mức cao) theo NTFS, chương trình tạo khuôn dạng sẽ tạo ra một *cung khởi động phân vùng*.

10.2.3.1 Cung khởi động phân vùng

Cung khởi động phân vùng PBS (Partition Boot Sector) nằm ở 16 cung đầu tiên, chứa cung khởi động và chương trình khởi động (bootstrap code) (Bảng 28).

Bảng 28

Địa chỉ offset	**Kích thước**	**Tên trường**
0x00	3 byte	Lệnh nhảy
0x03	8 byte	Nhận dạng nhà sản xuất thiết bị (OEM ID)
0x0B	25 byte	BPB -BIOS Partition Block Cấu trúc dữ liệu mô tả tổ chức volume ở mức vật lý
0x24	48 byte	BPB mở rộng
0x54	426 byte	Chương trình khởi động HĐH
0x01FE	2 byte	Dấu hiệu cuối cung

Trường dữ liệu ngay sau cấu trúc dữ liệu BPB (Bios Partition Block) chuẩn (như trong cung khởi động của FAT) là BPB mở rộng chứa các thông số đặc trưng của NTFS. Dữ liệu trong các trường của cung khởi động PBS cho phép chương trình nạp tìm thấy Bảng quản lý tập tin MFT khi khởi động hệ thống. Trong volume NTFS, MFT không nằm ở một cung được xác định trước như ở FAT. Nhờ đó mà Bảng quản lý tập tin MFT có thể được rời đi nơi khác nếu cung chứa nó bị hỏng. Nếu dữ liệu trong MFT bị hỏng, Windows coi là volume chưa được tạo khuôn dạng. Bản sao của cung khởi động được đặt ở vùng giữa của đĩa.

Nội dung các trường trong BPB như sau (Bảng 29).

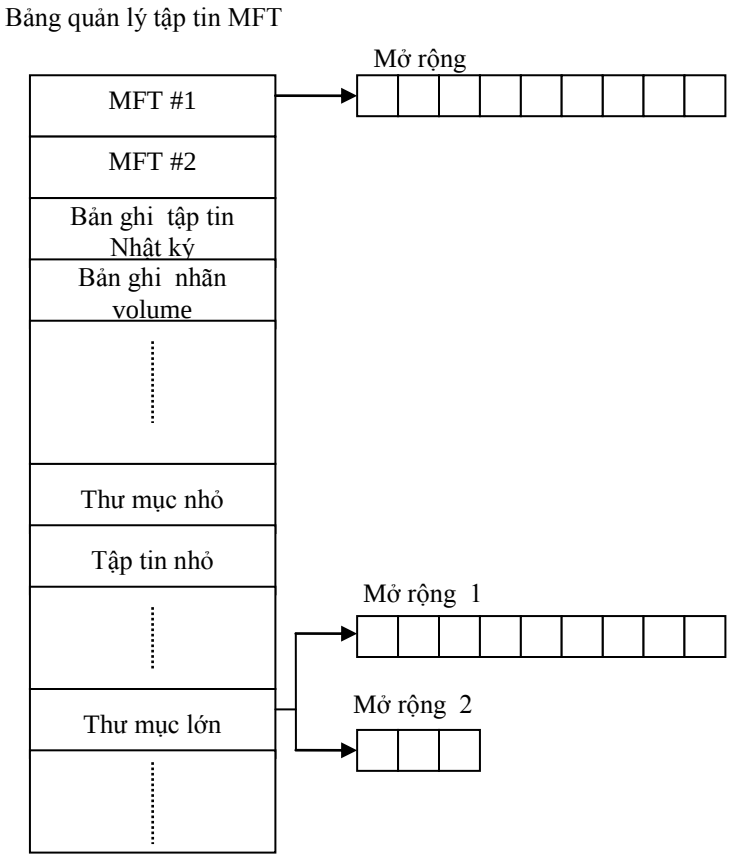
Bảng 29

Địa chỉ offset (Hexadecimal)	**Nội dung**	**Kích thước** (byte)
0B	Số lượng byte / một cung	2
0D	Số lượng cung / một liên cung	1
0E	Số lượng cung dự phòng	2
10	0	3
13	NTFS không dử dụng	2
15	Byte mô tả loại đĩa	1
16	0	2
18	Số lượng cung / một rãnh	2
1A	Số lượng đầu từ	2
1C	Số lượng cung ẩn	4
20	NTFS không dử dụng	4
24	NTFS không dử dụng	4
28	Số lượng cung/volume	8
30	Số liên cung chứa tập tin $MFT	8
38	Số liên cung chứa tập tin $MFTMirr	8
40	Số lượng liên cung/Đoạn bản ghi tập tin	4
44	Số lượng liên cung/Khối chỉ số	4
48	Số loạt volume	8
50	Checksum	4

10.2.3.2 Bảng quản lý tập tin MFT

Mỗi một tập tin trên NTFS được mô tả bằng *một bản ghi* trong một tập tin đặc biệt gọi là *Bảng quản lý tập tin MFT (Master File Table)* (Hình 103).

NTFS dành *16 bản ghi đầu tiên* của bảng MFT để chứa các thông tin đặc biệt.



Hình 103

Bản ghi đầu tiên (bản ghi số 0) của bảng MFT mô tả chính bản thân MFT. Bản ghi MFT này mô tả tất cả các tập tin trên volume, bao gồm tên tập tin, dấu thời gian, danh sách các số liên cung chứa luồng dữ liệu, nhận dạng bảo mật v.v.

Bản ghi thứ hai (bản ghi số 1) là tập tin MFT *phản chiếu* (là bản sao của MFT thứ nhất). Vị trí của đoạn dữ liệu chứa tập tin MFT (MFT #1) và MFT phản chiếu (MFT #2) được ghi trong cung khởi động. Bảng MFT #2 được NTFS sử dụng khi bảng MFT #1 bị lỗi.

Bản ghi thứ ba (bản ghi số 2) của MFT là tập tin nhật ký, chứa các giao dịch được NTFS dùng để khôi phục tập tin.

Bản ghi thứ tư (bản ghi số 3) đến thứ mười sáu (bản ghi số 15) chứa các thông tin như thư mục gốc, các liên cung được sử dụng trong volume, các liên cung hỏng trong volume, mô tả an ninh v.v.

Bản ghi thứ mười bẩy (bản ghi số 16) và các bản ghi tiếp theo của MFT là để chứa tập tin và thư mục (cũng được xem là tập tin) trên volume.

MFT chỉ định một số vùng cho mỗi bản ghi tập tin. Các tập tin và thư mục nhỏ (nhỏ hơn 1500 byte) được chứa toàn bộ trong bản ghi MFT. Hình 104 mô tả cấu trúc của một tập tin/thư mục nhỏ.

Thông tin chuẩn (dấu thời gian, kích thước v. v)	Tên tập tin hoặc thư mục	Mô tả an ninh	Dữ liệu hoặc chỉ số liên cung

Hình 104

Thiết kế này làm việc truy nhập tập tin/thư mục nhanh hơn. Ngay khi ta nhìn thấy tập tin là dùng được ngay, không như ở hệ thống FAT.

Các bản ghi thư mục khu trú trong MFT như bản ghi tập tin. THay vì chứa dữ liệu, nó chứa thông tin về chỉ số liên cung. Các bản ghi thư mục nhỏ nằm trọn trong cấu trúc của MFT.

Các thư mục lớn được tổ chức thành *cây -B+*, có các bản ghi trỏ đến các liên cung chứa các *lối vào thư mục*, các liên cung này không nằm trong cấu trúc MFT (Hình 120).

Hệ thống tập tin NTFS nhìn mỗi một tập tin/thư mục như một *tập các thuộc tính*. Mỗi một thuộc tính được nhận dạng bởi *mã thuộc tính*. Các thành phần của tập tin như: tên tập tin, dấu thời gian, mô tả an ninh, danh sách thuộc tính, dữ liệu (nội dung tập tin) v.v. đều là các thuộc tính tập tin. NTFS tạo thuộc tính *Danh sách thuộc tính* để mô tả vị trí của tất cả các bản ghi thuộc tính của tập tin.

10.2.3.3 Các tập tin hệ thống NTFS và siêu dữ liệu trong MFT

NTFS bao gồm các tập tin hệ thống, tất cả đều là ẩn trên volume. Tập tin hệ thống được NTFS sử dụng để lưu giữ siêu dữ liệu (metadata) của hệ thống và dùng trong hệ thống tập tin. Các tập tin hệ thống được chương trình tạo khuôn dạng đặt trong volume.
Các tập tin hệ thống nằm trong cấu trúc của bảng MFT như sau (Bảng 30).

Bảng 30

Tập tin hệ thống	Tên tập tin	Vị trí bản ghi	Ý nghĩa tập tin
MFT # 1	$MFT	0	Chứa bản ghi tập tin cơ sở cho từng tập tin/thư mục trên volume. MFT mô tả tất cả các tập tin trên volume, bao gồm tên tập tin, dấu thời gian, danh sách các số liên cung, nơi chứa luồng dl số liên cung, nơi chứa luồng dữ liệu, nhận dạng bảo mật v.v.
MFT # 2	$MFTMirr	1	Bản sao của $MFT
Tập tin nhật ký	$LogFile	2	Chứa danh sách các bước giao dịch, được dùng để khôi phục tập tin NTFS
Volume	$Volume	3	Chứa thông tin về volume như nhãn volume v.v.
Định nghĩa thuộc tính	$AttrDef	4	Bảng các tên, số và bộ mô tả thuộc tính
Chỉ số tên thư mục gốc	.	5	Thư mục gốc. dữ liệu thư mục chứa trong thuộc tính $Index_Root
Bitmap liên cung	$Bitmap	6	Chỉ ra các liên cung volume đang sử dụng
Cung khởi động	$Boot	7	Chứa phần BPB và chương trình khởi động, nếu là volume khởi động
Tập tin liên cung hỏng	$BadClus	8	Chỉ ra các liên cung hỏng
Tập tin bảo mật	$Secure	9	Chứa các bộ mô tả bảo mật cho tất cả các tập tin trong volume
Bảng chữ hoa	$Upcase	10	Chuyển ký tự thường thành ký tự hoa
Tập tin mở rộng	$Extend	11	Sử dụng cho các mở rộng tùy chọn như hạn ngạch, nhận dạng đối tượng...
		12-15	Dự phòng

TÀI LIỆU THAM KHẢO

1. John P. Hayes, *Computer Architecture and Organization*. McGraw-Hill, Inc. 1998.
2. William Stallings, *Computer Organization and Architecture*, Prentice-Hall, Inc. 1996.
3. Thomas C. Bartee, *Computer Architecture and Logic Design*. McGraw-Hill, Inc. 1991.
4. Daniel Tabak, *Advanced Microprocessors*. McGraw-Hill, Inc. 1995
5. Ivan Tomek, *The Foundations of Computer Architecture and Organization*. Computer Science Press. 1991.
6. Intel, *Intel Architecture: Basic Architecture*. 1997.
7. Intel, *Pentium Processor Family: Developer's Manuel*. 1997.
8. Intel, *Component Data Catalog*. 1991.
9. Barry B. Brey, *The Intel Microprocessors-Architecture, Programming and Interfacing*. Prentice-Hall, Inc. 1996.
10. Don Andreson. *USB System Architecture*. MindShare, Inc. 2001.
11. Scott Mueller, *Upgrading and Repairing PCs. (Bản dịch tiếng Việt: Cẩm nang sửa chữa và nâng cấp máy tính cá nhân* – Nhà Xuất bản Đà Nẵng, 2002).
12. Larry Jordan, *Communication and Networking for IBM-PC and Compatibles*. Brandy Publishing. 1992.
13. Henri Nussbaumer, C*omputer Communication Systems.* John Wiley&Son. 1990.
14. Nguyễn Văn Tam, *Các bộ vi xử lý thông dụng 16/32 bit*. Nhà xuất bản Thống kê. 1990.
15. Nguyễn Nam Trung, *Cấu trúc máy vi tính & Thiết bị ngoại vi*. Nhà xuất bản Khoa học kỹ thuật. 2000.

MỤC LỤC

Phần II MÁY VI TÍNH PC

Chương 6

KIẾN TRÚC MÁY VI TÍNH PC VÀ ĐƠN VỊ XỬ LÝ TRUNG TÂM

Chương 7

CÁC TỔ CHỨC DRAM TIÊN TIẾN

Chương 8

CÁC THIẾT BỊ ĐIỀU KHIỂN VÀ GIAO DIỆN VÀO-RA DỮ LIỆU CHUẨN