BỘ LAO ĐỘNG THƯƠNG BINH - XÃ HỘI
TỔNG CỤC DẠY NGHỀ

-----□□ □ □□-----

GIÁO TRÌNH
KIẾN TRÚC MÁY TÍNH
NGHỀ: KỸ THUẬT LẮP RÁP & SỬA CHỮA MÁY TÍNH

TRÌNH ĐỘ: CAO ĐẲNG

(Ban hành theo Quyết định số: 120/QĐ-TCDN ngày 25 tháng 02 năm 2013 của Tổng cục trưởng Tổng cục dạy nghề)

(mặt sau trang bìa)

TUYÊN BỐ BẢN QUYỀN:

LỜI GIỚI THIỆU

Kiến trúc máy tính là một mảng kiến thức không thể thiếu đối với sinh viên chuyên ngành điện tử viễn thông và công nghệ thông tin. Đây là nền tảng để nghiên cứu chuyên sâu trong chuyên ngành này. Chúng ta đều biết rằng không có kiến thức cơ sở vững vàng sẽ không có phát triển ứng dụng vì vậy tài liệu này sẽ giúp cho sinh viên trang bị cho mình những kiến thức căn bản nhất, thiết thực nhất. Cuốn sách này không chỉ hữu ích đối với sinh viên ngành viễn thông và công nghệ thông tin, mà còn cần thiết cho cả các cán bộ kỹ thuật đang theo học các lớp bổ túc hoàn thiện kiến thức của mình.

Môn học Kiến trúc máy tính là một môn học chuyên môn của học viên ngành sửa chữa máy tính và quản trị mạng. Môn học này nhằm trang bị cho học viên các trường công nhân kỹ thuật và các trung tâm dạy nghề những kiến thức về Kiến trúc máy tính. Với các kiến thức này học viên có thể áp dụng trực tiếp vào lĩnh vực sản xuất cũng như đời sống. Môn học này cũng có thể làm tài liệu tham khảo cho các cán bộ kỹ thuật, các học viên của các nghành khác quan tâm đến lĩnh vực này.

Mặc dù đã có những cố gắng để hoàn thành giáo trình theo kế hoạch, nhưng do hạn chế về thời gian và kinh nghiệm soạn thảo giáo trình, nên tài liệu chắc chắn còn những khiếm khuyết. Rất mong nhận được sự đóng góp ý kiến của các thầy cô trong Khoa cũng như các bạn sinh viên và những ai sử dụng tài liệu này.

Hà Nội, 2013
Tham gia biên soạn
Khoa Công Nghệ Thông Tin
Trường Cao Đẳng Nghề Kỹ Thuật Công Nghệ
Địa Chỉ: Tổ 59 Thị trấn Đông Anh – Hà Nội
Tel: 04. 38821300
Chủ biên: Phùng Sỹ Tiến

Mọi góp ý liên hệ: Phùng Sỹ Tiến – Trưởng Khoa Công Nghệ Thông Tin
Mobible: 0983393834
Email: tienphungktcn@gmail.com – tienphungktcn@yahoo.com

MỤC LỤC

MÔN HỌC: KIẾN TRÚC MÁY TÍNH

Mã môn học: MH12

Vị trí, ý nghĩa, vai trò môn học:

- Vị trí:
  + Môn học được bố trí sau khi sinh viên học xong các môn học chung, các môn học cơ sở chuyên ngành đào tạo chuyên môn nghề.
- Tính chất:
  + Là môn học chuyên ngành.
- Ý nghĩa và vai trò của môn học
  + Là môn học không thể thiếu của nghề Sửa chữa lắp ráp máy tính. Môn Kiến trúc máy tính cung cấp cho học sinh, sinh viên cấu tạo, nguyên lý hoạt động của toàn bộ linh kiện máy tính phục vụ chính cho học tập và công việc của học sinh, sinh viên của nghề này.

Mục tiêu của môn học:

- Biết về lịch sử của máy tính, các thế hệ máy tính và cách phân loại máy tính.
- Hiểu các thành phần cơ bản của kiến trúc máy tính, các tập lệnh. Các kiểu kiến trúc máy tính: mô tả kiến trúc, các kiểu định vị.
- Hiểu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý. Mô tả diễn tiến thi hành một lệnh mã máy và một số kỹ thuật xử lý thông tin: ống dẫn, siêu ống dẫn, siêu vô hướng.
- Hiểu chức năng và nguyên lý hoạt động của các cấp bộ nhớ.
- Hiểu phương pháp an toàn dữ liệu trên thiết bị lưu trữ ngoài.
- Lập trình được trên các tập lệnh cơ bản của Assembly.
- Tự tin khi tiếp cận những công nghệ phần cứng mới.

Nội dung của môn học:

Mã môn học	Tên chương mục	Thời gian			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra*
MH12-01	**Tổng quan** Các thế hệ máy tính Phân loại máy tính Thành quả của máy tính Thành quả của máy tính	**4**	**4**	**0** 0	
MH12-02	**Kiến trúc phần mềm bộ xử lý** Thành phần cơ bản của một máy	**12**	**8**	**2**	**2**

	tính Định nghĩa kiến trúc máy tính Tập lệnh Thủ tục Toán hạng				
MH12-03	**Tổ chức bộ xử lý** Đường đi dữ liệu Bộ điều khiển Diễn tiến thi hành lệnh mã máy Ngắt Kỹ thuật ống dẫn Ống dẫn, siêu ống dẫn, siêu vô hướng	**12**	**6**	**4**	**2**
MH12-04	**Bộ nhớ** Các loại bộ nhớ Các cấp bộ nhớ Cách truy xuất dữ liệu trong bộ nhớ Hiểu về bộ nhớ Cache và cách tổ chức bộ nhớ Cache trong CPU	**16**	**11**	**4**	**1**
MH12-05	**Thiết bị nhập xuất** Đĩa từ Đĩa quang Các loại thẻ nhớ Băng từ Các chuẩn về BUS An toàn dữ liệu trong lưu trữ	**16**	**11**	**4**	**1**
MH12-06	**Ngôn ngữ Assembly** Tổng quan Các lệnh cơ bản Các lệnh điều khiển Ngăn xếp và các thủ tục	**30**	**16**	**12**	**2**
	Cộng	**90**	**56**	**26**	**8**

CHƯƠNG 1:TỔNG QUAN

Mã chương: MH12 – 01.

Mục đích:

- *Giới thiệu lịch sử phát triển của máy tính, các thế hệ máy tính và cách phân loại máy tính. Giới thiệu các cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự.*

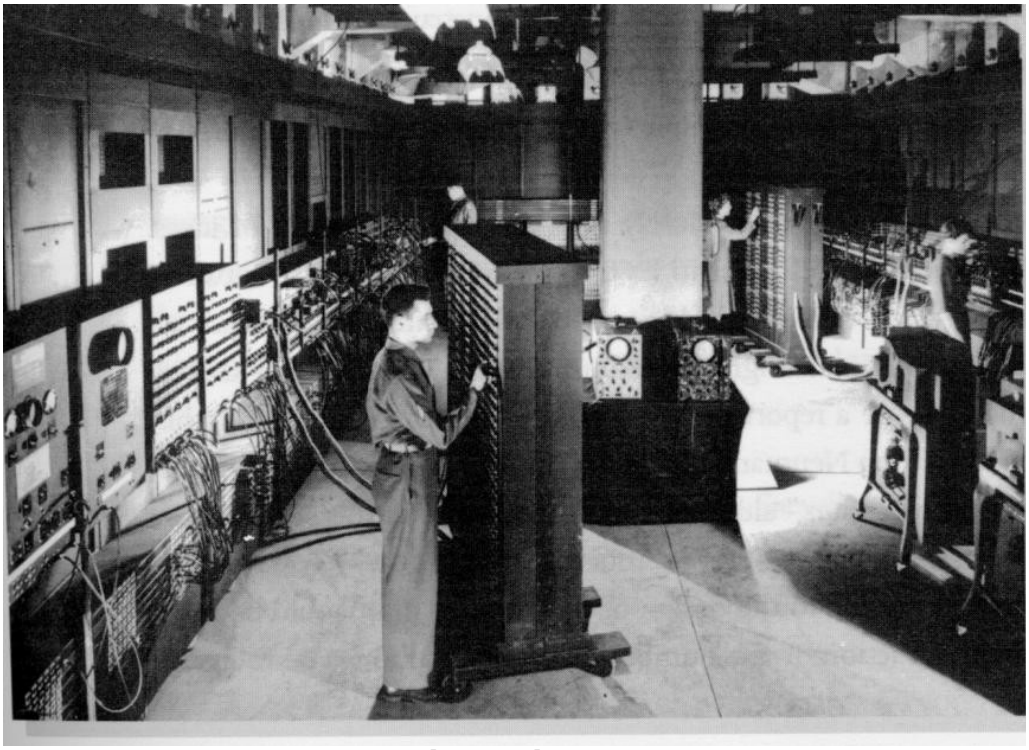
1. Các thế hệ máy tính

Mục đích:

- *Giới thiệu lịch sử phát triển của máy tính*
- *Trình bày được các thế hệ máy tính*

Sự phát triển của máy tính được mô tả dựa trên sự tiến bộ của các công nghệ chế tạo các linh kiện cơ bản của máy tính như: bộ xử lý, bộ nhớ, các ngoại vi,…Ta có thể nói máy tính điện tử số trải qua bốn thế hệ liên tiếp. Việc chuyển từ thế hệ trước sang thế hệ sau được đặc trưng bằng một sự thay đổi cơ bản về công nghệ.

1.1 Thế hệ đầu tiên (1946-1957)



Hình 1- 1. Thế hệ đầu tiên (1946-1957)

ENIAC (Electronic Numerical Integrator and Computer) là máy tính điện tử số đầu tiên do Giáo sư Mauchly và người học trò Eckert tại Đại học Pennsylvania thiết kế vào năm 1943 và được hoàn thành vào năm 1946. Đây là một máy tính khổng lồ với thể tích dài 20 mét, cao 2,8 mét và rộng vài mét. ENIAC bao gồm: 18.000 đèn điện tử, 1.500 công tắc tự động, cân nặng 30 tấn, và tiêu thụ 140KW giờ. Nó có 20 thanh ghi 10 bit (tính toán trên số thập phân). Có khả năng thực hiện 5.000 phép toán cộng trong một giây. Công việc lập trình bằng tay bằng cách đấu nối các đầu cắm điện và dùng các ngắt điện.

Giáo sư toán học John Von Neumann đã đưa ra ý tưởng thiết kế máy tính IAS (Princeton Institute for Advanced Studies): chương trình được lưu trong bộ nhớ, bộ điều khiển sẽ lấy lệnh và biến đổi giá trị của dữ liệu trong phần bộ nhớ, bộ làm toán và luận lý (ALU: Arithmetic And Logic Unit) được điều khiển để tính toán trên dữ liệu nhị phân, điều khiển hoạt động của các thiết bị vào ra. Đây là một ý tưởng nền tảng cho các máy tính hiện đại ngày nay. Máy tính này còn được gọi là *máy tính Von Neumann.*

Vào những năm đầu của thập niên 50, những máy tính thương mại đầu tiên được đưa ra thị trường: 48 hệ máy UNIVAC I và 19 hệ máy IBM 701 đã được bán ra.

1.2 Thế hệ thứ hai (1958-1964)

Công ty Bell đã phát minh ra *transistor* vào năm 1947 và do đó thế hệ thứ hai của máy tính được đặc trưng bằng sự thay thế các đèn điện tử bằng các transistor lưỡng cực. Tuy nhiên, đến cuối thập niên 50, máy tính thương mại dùng transistor mới xuất hiện trên thị trường. Kích thước máy tính giảm, rẻ tiền hơn, tiêu tốn năng lượng ít hơn. Vào thời điểm này, mạch in và bộ nhớ bằng xuyến từ được dùng. Ngôn ngữ cấp cao xuất hiện (như FORTRAN năm 1956, COBOL năm 1959, ALGOL năm 1960) và hệ điều hành kiểu tuần tự (Batch Processing) được dùng. Trong hệ điều hành này, chương trình của người dùng thứ nhất được chạy, xong đến chương trình của người dùng thứ hai và cứ thế tiếp tục.

1.3 Thế hệ thứ ba (1965-1971)

Thế hệ thứ ba được đánh dấu bằng sự xuất hiện của các *mạch kết* (mạch tích hợp - IC: Integrated Circuit). Các mạch kết độ tích hợp mật độ thấp (SSI: Small Scale Integration) có thể chứa vài chục linh kiện và kết độ tích hợp mật độ trung bình (MSI: Medium Scale Integration) chứa hàng trăm linh kiện trên mạch tích hợp.

Mạch in nhiều lớp xuất hiện, bộ nhớ bán dẫn bắt đầu thay thế bộ nhớ bằng xuyến từ. Máy tính đa chương trình và hệ điều hành chia thời gian được dùng.

1.4 Thế hệ thứ tư (1972)

Thế hệ thứ tư được đánh dấu bằng các *IC có mật độ tích hợp cao* (LSI: Large Scale Integration) có thể chứa hàng ngàn linh kiện. Các IC mật độ tích hợp rất cao (VLSI: Very Large Scale Integration) có thể chứa hơn 10 ngàn linh kiện trên mạch. Hiện nay, các chip VLSI chứa hàng triệu linh kiện.

Với sự xuất hiện của bộ vi xử lý (microprocessor) chứa cả phần thực hiện và phần điều khiển của một bộ xử lý, sự phát triển của công nghệ bán dẫn các máy vi tính đã được chế tạo và khởi đầu cho các thế hệ máy tính cá nhân. Các bộ nhớ bán dẫn, bộ nhớ cache, bộ nhớ ảo được dùng rộng rãi. Các kỹ thuật cải tiến tốc độ xử lý của máy tính không ngừng được phát triển: kỹ thuật ống dẫn, kỹ thuật vô hướng, xử lý song song mức độ cao,…

1.5 Khuynh hướng hiện tại

Việc chuyển từ thế hệ thứ tư sang thế hệ thứ 5 còn chưa rõ ràng. Người Nhật đã và đang đi tiên phong trong các chương trình nghiên cứu để cho ra đời thế hệ thứ 5 của máy tính, thế hệ của những máy tính thông minh, dựa trên các ngôn ngữ trí tuệ nhân tạo như LISP và PROLOG,... và những giao diện người - máy thông minh. Đến thời điểm này, các nghiên cứu đã cho ra các sản phẩm bước đầu và gần đây nhất (2004) là sự ra mắt sản phẩm người máy thông minh gần giống với con người nhất: ASIMO (*Advanced Step Innovative Mobility: Bước chân tiên tiến của đổi mới và chuyển động*). Với hàng trăm nghìn máy móc điện tử tối tân đặt trong cơ thể, ASIMO có thể lên/xuống cầu thang một cách uyển chuyển, nhận diện người, các cử chỉ hành động, giọng nói và đáp ứng một số mệnh lệnh của con người. Thậm chí, nó có thể bắt chước cử động, gọi tên người và cung cấp thông tin ngay sau khi bạn hỏi, rất gần gũi và thân thiện. Hiện nay có nhiều công ty, viện nghiên cứu của Nhật thuê Asimo tiếp khách và hướng dẫn khách tham quan như: Viện Bảo tàng Khoa học năng lượng và Đổi mới quốc gia, hãng IBM Nhật Bản, Công ty điện lực Tokyo.

Hãng Honda bắt đầu nghiên cứu ASIMO từ năm 1986 dựa vào nguyên lý chuyển động bằng hai chân. Cho tới nay, hãng đã chế tạo được 50 robot ASIMO. Các tiến bộ liên tục về mật độ tích hợp trong VLSI đã cho phép thực hiện các mạch vi xử lý ngày càng mạnh (8 bit, 16 bit, 32 bit và 64 bit với việc xuất hiện các bộ xử lý RISC năm 1986 và các bộ xử lý siêu vô hướng năm 1990). Chính các bộ xử lý này giúp thực hiện các máy tính song song với từ vài bộ xử lý đến vài ngàn bộ xử lý. Điều này làm các chuyên gia về kiến trúc máy tính tiên đoán thế hệ thứ 5 là thế hệ các máy tính xử lý song song.

Bảng 1-1. Các thế hệ máy tính

Thế hệ	Năm	Kỹ thuật	Sản phẩm mới	Hãng sản xuất và máy tính
1	1946-1957	Đèn điện tử	Máy tính điện tử tung ra thị trường	IBM 701. UNIVAC
2	1958-1964	Transistors	Máy tính rẻ tiền	Intel,Burroughs 6500, NCR, CDC 6600, Honeywell
3	1965-1971	Mach IC	Máy tính mini	50 hãng mới: DEC PDP-11, Data general ,Nova
4	1972	LSI - VLSI	Máy tính cá nhân và trạm làm việc	Apple II, IBM-PC, Appolo DN 300, Sun 2
5		Xử lý song song	Máy tính đa xử lý. Đa máy tính	Sequent … Thinking Machine Inc. Honda, Casio

2. Phân loại máy tính

Mục đích:

- *Trình bày được cách phân loại máy tính.*

Thông thường máy tính được phân loại theo tính năng kỹ thuật và giá tiền.

2.1 Các siêu máy tính (Super Computer):

Là các máy tính đắt tiền nhất và tính năng kỹ thuật cao nhất. Giá bán một siêu máy tính từ vài triệu USD. Các siêu máy tính thường là các máy tính vectơ hay các máy tính dùng kỹ thuật vô hướng và được thiết kế để tính toán khoa học, mô phỏng các hiện tượng. Các siêu máy tính được thiết kế với kỹ thuật xử lý song song với rất nhiều bộ xử lý (hàng ngàn đến hàng trăm ngàn bộ xử lý trong một siêu máy tính).

2.2 Các máy tính lớn (Mainframe):

Là loại máy tính đa dụng. Nó có thể dùng cho các ứng dụng quản lý cũng như các tính toán khoa học. Dùng kỹ thuật xử lý song song và có hệ thống vào ra mạnh. Giá một máy tính lớn có thể từ vài trăm ngàn USD đến hàng triệu USD.

2.3 Máy tính mini (Minicomputer):

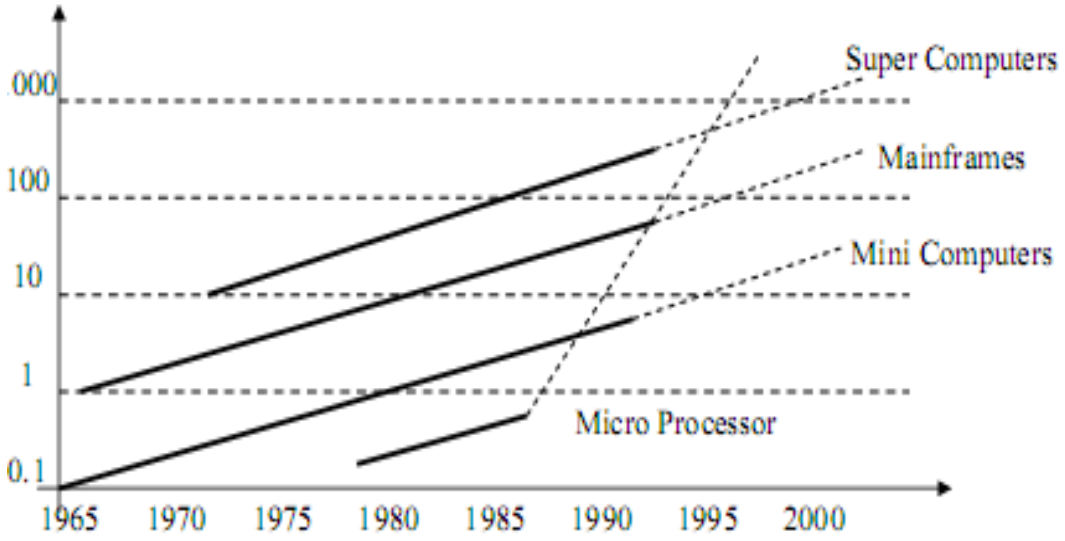
Là loại máy cỡ trung, giá một máy tính mini có thể từ vài chục USD đến vài trăm ngàn USD.

2.4 Máy vi tính (Microcomputer)

Là loại máy tính dùng bộ vi xử lý, giá một máy vi tính có thể từ vài trăm USD đến vài ngàn USD.

3. Thành quả của máy tính, qui luật Moore về sự phát triển của máy tính

Hình 1-2 cho thấy diễn biến của thành quả tối đa của máy tính. Thành quả này tăng theo hàm số mũ, độ tăng trưởng các máy vi tính là 35% mỗi năm, còn đối với các loại máy khác, độ tăng trưởng là 20% mỗi năm. Điều này cho thấy tính năng các máy vi tính đã vượt qua các loại máy tính khác vào đầu thập niên 90.



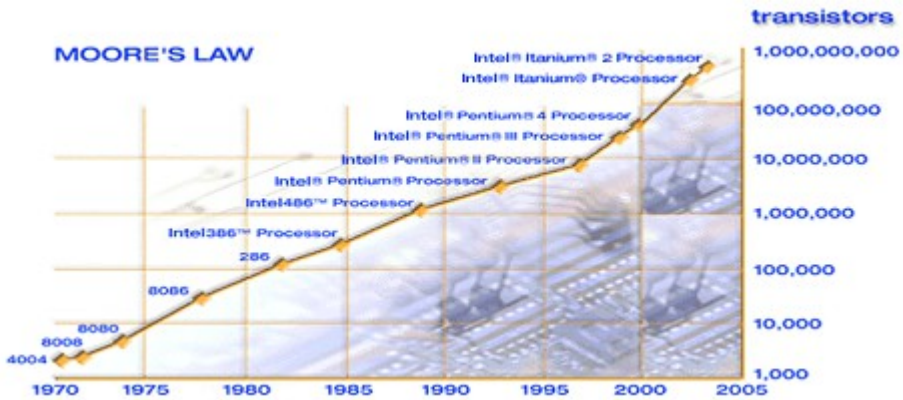
Hình 1- 2. Đánh giá thành quả của máy tính

Máy tính dùng thật nhiều bộ xử lý song song rất thích hợp khi phải làm tính thật nhiều.

Sự tăng trưởng theo hàm số mũ của công nghệ chế tạo transistor MOS là nguồn gốc của thành quả các máy tính.

Hình 1-4 cho thấy sự tăng trưởng về tần số xung nhịp của các bộ xử lý MOS. Độ tăng trưởng của tần số xung nhịp bộ xử lý tăng gấp đôi sau mỗi thế hệ và độ trì hoãn trên mỗi cổng / xung nhịp giảm 25% cho mỗi năm .

Sự phát triển của công nghệ máy tính và đặc biệt là sự phát triển của bộ vi xử lý của các máy vi tính làm cho các máy vi tính có tốc độ vượt qua tốc độ bộ xử lý của các máy tính lớn hơn.



Hình 1-3. Sự phát *triển của bộ xử lý Intel*

Bảng 1-2 Sự phát triển của bộ xử lý Intel
dựa vào số lượng transistor trong một mạch tích hợp theo qui luật Moore

Bộ xử lý Intel	Năm sản xuất	Số lượng transistor tích hợp
4004	1971	2.250
8008	1972	2.500
8080	1974	5.000
8086	1978	29.000
80286	1982	120.000
Intel 386 ™processor	1985	275.000
Intel 486 ™processor	1989	1.180.000
Intel ®Pentium ® processor	1993	3.100.000
Intel ®Pentium ® II processor	1997	7.500.000
Intel ®Pentium ® III processor	1999	24.000.000

Intel ®Pentium ® 4 processor	2000	42.000.000
Intel ® Itanium ® processor	2002	220.000.000
Intel ® Itanium ® 2 processor	2003	410.000.000

Từ năm 1965, Gordon Moore (đồng sáng lập công ty Intel) quan sát và nhận thấy số transistor trong mỗi mạch tích hợp có thể tăng gấp đôi sau mỗi năm, G. Moore đã đưa ra dự đoán: ***Khả năng của máy tính sẽ tăng lên gấp đôi sau 18 tháng với giá thành là như nhau.***

Kết quả của quy luật Moore là:

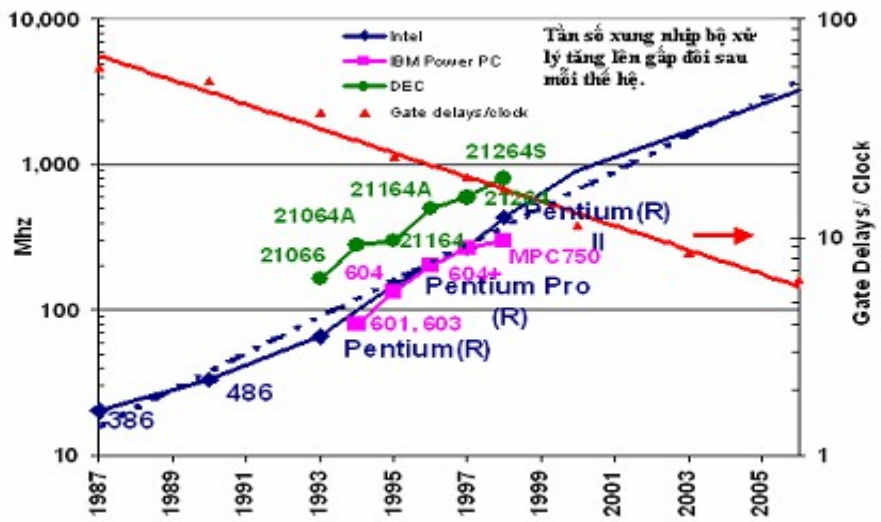
+ Chi phí cho máy tính sẽ giảm.

+ Giảm kích thước các linh kiện, máy tính sẽ giảm kích thước

+ Hệ thống kết nối bên trong mạch ngắn: tăng độ tin cậy, tăng tốc độ .

+ Tiết kiệm năng lượng cung cấp, toả nhiệt thấp.

+ Các IC thay thế cho các linh kiện rời.



Hình 1-4. Xung nhịp các bộ xử lý MOS

Một số khái niệm liên quan:

+ Mật độ tích hợp là số linh kiện tích hợp trên một diện tích bề mặt tấm silicon cho sẵn, cho biết số nhiệm vụ và mạch có thực hiện.

+ Tần số xung nhịp bộ xử lý cho biết tần số thực hiện các nhiệm vụ.

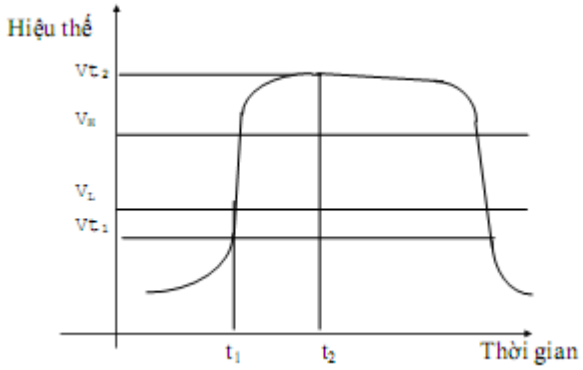
+ Tốc độ xử lý của máy tính trong một giây (hay công suất tính toán của mỗi mạch): được tính bằng tích của mật độ tích hợp và tần số xung nhịp. Công suất này cũng tăng theo hàm mũ đối với thời gian.

4. Thông tin và sự mã hóa thông tin

Mục đích:

- *Giới thiệu các cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự.*

4.1 Khái niệm thông tin



Hình 1-5. Thông tin về 2 trạng thái có ý nghĩa của hiệu điện thế

Khái niệm về thông tin gắn liền với sự hiểu biết một trạng thái cho sẵn trong nhiều trạng thái có thể có vào một thời điểm cho trước.

Trong hình này, chúng ta quy ước có hai trạng thái có ý nghĩa: trạng thái thấp khi hiệu điện thế thấp hơn V_L và trạng thái cao khi hiệu điện thế lớn hơn V_H. Để có thông tin, ta phải xác định thời điểm ta nhìn trạng thái của tín hiệu. Thí dụ, tại thời điểm t_1 thì tín hiệu ở trạng thái thấp và tại thời điểm t_2 thì tín hiệu ở trạng thái cao.

4.2 Lượng thông tin và sự mã hoá thông tin

Thông tin được đo lường bằng đơn vị thông tin mà ta gọi là *bit*. Lượng thông tin được định nghĩa bởi công thức:

$$\mathbf{I = Log_2(N)}$$

Trong đó I: là lượng thông tin tính bằng bit

N: là số trạng thái có thể có

Vậy một bit ứng với sự hiểu biết của một trạng thái trong hai trạng thái có thể có. Thí dụ, sự hiểu biết của một trạng thái trong 8 trạng thái có thể ứng với một lượng thông tin là:

$$I = Log_2(8) = 3 \text{ bit}$$

Tám trạng thái được ghi nhận nhờ 3 số nhị phân (mỗi số nhị phân có thể có giá trị 0 hoặc 1).

Như vậy lượng thông tin là số con số nhị phân cần thiết để biểu diễn số trạng thái có thể có. Do vậy, một con số nhị phân được gọi là một bit. Một từ n bit có thể tượng trưng một trạng thái trong tổng số 2^n trạng thái mà từ đó có thể tượng trưng. Vậy một từ n bit tương ứng với một lượng thông tin n bit.

Bảng 1-3. Tám trạng thái khác nhau ứng với 3 số nhị phân

Trạng thái	X2	X1	X0
0	0	0	0
1	0	0	1
2	0	1	0

3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

4.3 Biểu diễn các số:

Khái niệm hệ thống số: Cơ sở của một hệ thống số định nghĩa phạm vi các giá trị có thể có của một chữ số. Ví dụ: trong hệ thập phân, một chữ số có giá trị từ 0-9, trong hệ nhị phân, một chữ số (một bit) chỉ có hai giá trị là 0 hoặc 1.

Dạng tổng quát để biểu diễn giá trị của một số:

$$V_k = \sum_{i=-m}^{n-1} b_i \cdot k^i$$

Trong đó:
V_k: Số cần biểu diễn giá trị
m: số thứ tự của chữ số phần lẻ
(phần lẻ của số có m chữ số được đánh số thứ tự từ -1 đến -m)
n-1: số thứ tự của chữ số phần nguyên
(phần nguyên của số có n chữ số được đánh số thứ tự từ 0 đến n-1)
b_i: giá trị của chữ số thứ i
k: hệ số (k=10: hệ thập phân; k=2: hệ nhị phân;...).

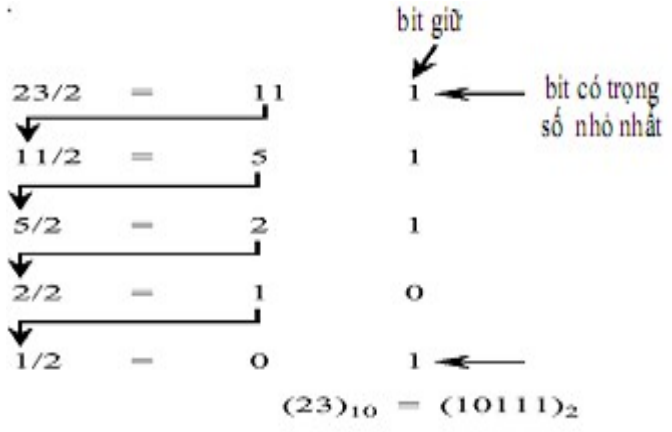
Ví dụ: biểu diễn số 541.25_{10}

$541.25_{10} = 5 * 10^2 + 4 * 10^1 + 1 * 10^0 + 2 * 10\text{-}1 + 5 * 10^{-2}$
$= (500)_{10} + (40)_{10} + (1)_{10} + (2/10)_{10} + (5/100)_{10}$

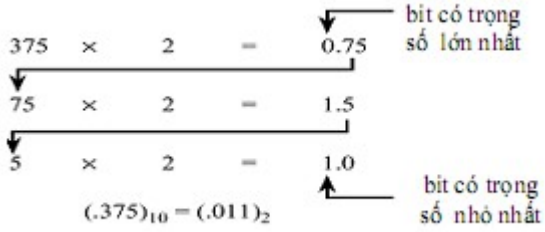
Một máy tính được chủ yếu cấu tạo bằng các mạch điện tử có hai trạng thái. Vì vậy, rất tiện lợi khi dùng các số nhị phân để biểu diễn số trạng thái của các mạch điện hoặc để mã hoá các ký tự, các số cần thiết cho vận hành của máy tính.

Để biến đổi một số hệ thập phân sang nhị phân, ta có hai phương thức biến đổi:

Phương thức số dư để biến đổi phần nguyên của số thập phân sang nhị phân. Ví dụ: Đổi 23.37510 sang nhị phân. Chúng ta sẽ chuyển đổi phần nguyên dùng phương thức số dư:



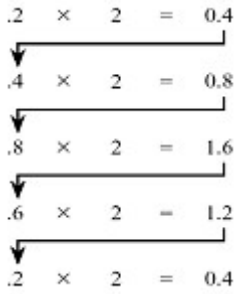
- Phương thức nhân để biến đổi phần lẻ của số thập phân sang nhị phân



Kết quả cuối cùng nhận được là: $23.375_{10} = 10111.011_2$

Tuy nhiên, trong việc biến đổi phần lẻ của một số thập phân sang số nhị phân theo phương thức nhân, có một số trường hợp việc biến đổi số lặp lại vô hạn bit có trọng số lớn nhất bit có trọng số nhỏ nhất

Ví dụ



Trường hợp biến đổi số nhị phân sang các hệ thống số khác nhau, ta có thể nhóm một số các số nhị phân để biểu diễn cho số trong hệ thống số tương ứng.

(Base 2)	Octal (Base 8)	Decimal (Base 10)	Hexadecimal (Base 16)
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4

(Base 2)	Octal (Base 8)	Decimal (Base 10)	Hexadecimal (Base 16)
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Thông thường, người ta nhóm 4 bit trong hệ nhị phân hệ để biểu diễn số dưới dạng thập lục phân (Hexadecimal).
Như vậy, dựa vào cách biến đổi số trong bảng nêu trên, chúng ta có ví dụ về cách biến đổi các số trong các hệ thống số khác nhau theo hệ nhị phân:

- $1011_2 = (10_2)(11_2) = 23_4$
- $23_4 = (2_4)(3_4) = (10_2)(11_2) = 1011_2$
- $101010_2 = (101_2)(010_2) = 52_8$
- $01101101_2 = (0110_2)(1101_2) = 6D_{16}$

Một từ n bit có thể biểu diễn tất cả các số dương từ 0 tới 2^n-1. Nếu d_i là một số nhị phân thứ i, một từ n bit tương ứng với một số nguyên thập phân.

$$N = \sum_{i=0}^{n-1} d_i 2^i$$

Một Byte (gồm 8 bit) có thể biểu diễn các số từ 0 tới 255 và một từ 32 bit cho phép biểu diễn các số từ 0 tới 4294967295.

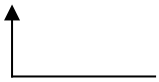
4.4 Số nguyên có dấu

Có nhiều cách để biểu diễn một số n bit có dấu. Trong tất cả mọi cách thì bit cao nhất luôn tượng trưng cho dấu.

Khi đó, bit dấu có giá trị là 0 thì số nguyên dương, bit dấu có giá trị là 1 thì số nguyên âm. Tuy nhiên, cách biểu diễn dấu này không đúng trong trường hợp số được biểu diễn bằng số thừa K mà ta sẽ xét ở phần sau trong chương này (bit dấu có giá trị là 1 thì số nguyên dương, bit dấu có giá trị là 0 thì số nguyên âm).

d_{n-1} d_{n-2} d_{n-3} … d_2 d_1 d0

…



Số nguyên có bit d_{n-1} là bit dấu và có trị số tượng trưng bởi các bit từ d_0 tới d_{n-2}.

a. Cách biểu diễn bằng trị tuyệt đối và dấu

Trong cách này, bit d_{n-1} là bit dấu và các bit từ d_0 tới d_{n-2} cho giá trị tuyệt đối. Một từ n bit tương ứng với số nguyên thập phân có dấu.

$$N = (-1)^{d_{n-1}} \sum_{i=0}^{n-2} d_i 2^i$$

Ví dụ: $+25_{10} = 00011001_2$

$-25_{10} = 10011001_2$

- Một Byte (8 bit) có thể biểu diễn các số có dấu từ -127 tới +127.
- Có hai cách biểu diễn số không là 0000 0000 (+0) và 1000 0000 (-0).

b. Cách biểu diễn hằng số bù 1

Trong cách biểu diễn này, số âm -N được có bằng cách thay các số nhị phân d_i của số đương N bằng số bù của nó (nghĩa là nếu $d_i = 0$ thì người ta đổi nó thành 1 và ngược lại).

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 11100110_2$

- Một Byte cho phép biểu diễn tất cả các số có dấu từ -127 ($1000\ 0000_2$) đến 127 ($0111\ 1111_2$)
- Có hai cách biểu diễn cho 0 là 0000 0000 (+0) và 1111 1111 (-0).

c. Cách biểu diễn bằng số bù 2

Để có số bù 2 của một số nào đó, người ta lấy số bù 1 rồi cộng thêm 1. Vậy một từ n bit (d_{n-1} ..... d_0) có trị thập phân.

$$N = - d_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} d_i 2^i$$

Một từ n bit có thể biểu diễn các số có dấu từ -2^{n-1} đến $2^{n-1} - 1$. Chỉ có một cách duy nhất để biểu diễn cho số không là tất cả các bit của số đó đều bằng không.

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 11100111_2$

- Dùng 1 Byte (8 bit) để biểu diễn một số có dấu lớn nhất là +127 và số nhỏ nhất
là –128.
- Chỉ có một giá trị 0: $+0 = 00000000_2$, $-0 = 00000000_2$

Bảng 1-4. Số 4 bit có dấu theo cách biểu diễn số âm bằng số bù 2

d_3	d_2	d_1	d_0	N		d_3	d_2	d_1	d_0	N
0	0	0	0	0		1	0	0	0	-8
0	0	0	1	1		1	0	0	1	-7
0	0	1	0	2		1	0	1	0	-6
0	0	1	1	3		1	0	1	1	-5
0	1	0	0	4		1	1	0	0	-4
0	1	0	1	5		1	1	0	1	-3
0	1	1	0	6		1	1	1	0	-2
0	1	1	1	7		1	1	1	1	-1

d. Cách biểu diễn bằng số thừa K

Trong cách này, số dương của một số N có được bằng cách "*cộng thêm vào*" số thừa K được chọn sao cho tổng của K và một số âm bất kỳ luôn luôn dương. Số âm -N của số N có được bằng cách lấy K-N (hay lấy bù hai của số vừa xác định).

Ví dụ: (số thừa K=128, số "cộng thêm vào" 128 là một số nguyên dương. Số âm là số lấy bù hai số vừa tính, bỏ qua số giữ của bit cao nhất) :

$+25_{10} = 10011001_2$ $-25_{10} = 01100111_2$

- Dùng 1 Byte (8 bit) để biểu diễn một số có dấu lớn nhất là +127 và số nhỏ nhất là (âm) –128.

- Chỉ có một giá trị 0: $+0 = 10000000_2$, $-0 = 10000000_2$

Cách biểu diễn số nguyên có dấu bằng số bù 2 được dùng rộng rãi cho các phép tính số nguyên. Nó có lợi là không cần thuật toán đặc biệt nào cho các phép tính cộng và tính trừ, và giúp phát hiện dễ dàng các trường hợp bị tràn.

Các cách biểu diễn bằng "*dấu , trị tuyệt đối*" hoặc bằng "*số bù 1*" dẫn đến việc dùng các thuật toán phức tạp và bất lợi vì luôn có hai cách biểu diễn của số không. Cách biểu diễn bằng *"dấu , trị tuyệt đối"* được dùng cho phép nhân của số có dấu chấm động.

Cách biểu diễn bằng số thừa K được dùng cho số mũ của các số có dấu chấm động. Cách này làm cho việc so sánh các số mũ có dấu khác nhau trở thành việc so sánh các số nguyên dương.

4.5 Cách biểu diễn số thập phân

Một vài ứng dụng, đặc biệt ứng dụng quản lý, bắt buộc các phép tính thập phân phải chính xác, không làm tròn số. Với một số bit cố định, ta không thể đổi một cách chính xác số nhị phân thành số thập phân và ngược lại. Vì vậy, khi cần phải dùng số thập phân, ta dùng cách biểu diễn số thập phân mã bằng nhị phân (BCD: Binary Coded Decimal) theo đó mỗi số thập phân được mã với 4 số nhị phân (bảng I.6).

Bảng 1-5. Số thập phân mã bằng nhị phân

Số thập phân	d_3	d_2	d_1	d_0	Số thập phân	d_3	d_2	d_1	d_0
0	0	0	0	0	5	0	1	0	1
1	0	0	0	1	6	0	1	1	0
2	0	0	1	0	7	0	1	1	1
3	0	0	1	1	8	1	0	0	0
4	0	1	0	0	9	1	0	0	1

Để biểu diễn số BCD có dấu, người ta thêm số 0 trước một số dương cần tính, ta có số âm của số BCD bằng cách lấy bù 10 số cần tính.

Ví dụ: biểu diễn số $+079_{10}$ bằng số BCD: 0000 0111 1001
Bù 9 1001 0010 0000
+1
Bù 10 1001 0010 0001
Vậy, ta có: Số -079_{10} trong cách biểu diễn số BCD: *1001 0010 0001*$_{BCD}$.
Cách tính toán trên tương đương với cách sau:

- Trước hết ta lấy số bù 9 của số 079 bằng cách: 999 - 079 = 920.
- Cộng 1 vào số bù 9 ta được số bù 10: 920 + 1 = 921.
- Biểu diễn số 921 dưới dạng số BCD, ta có: $1001\ 0010\ 0001_{BCD}$

4.6 Biểu diễn các ký tự

Tuỳ theo các hệ thống khác nhau, có thể sử dụng các bảng mã khác nhau: ASCII, EBCDIC, UNICODE,....Các hệ thống trước đây thường dùng bảng mã ASCII (American Standard Codes for Information Interchange) để biểu diễn các chữ, số và một số dấu thường dùng mà ta gọi chung là ký tự. Mỗi ký tự được biểu diễn bởi 7 bit trong một Byte. Hiện nay, một trong các bảng mã thông dụng được dùng là Unicode, trong bảng mã này, mỗi ký tự được mã hoá bởi 2 Byte.

00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	´	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

Bảng mã ASCII

0000	NUL	0020	SP	0040	@	0060	`	0080	Ctrl	00A0	NBS	00C0	À	00E0	à
0001	SOH	0021	!	0041	A	0061	a	0081	Ctrl	00A1	¡	00C1	Á	00E1	á
0002	STX	0022	"	0042	B	0062	b	0082	Ctrl	00A2	¢	00C2	Â	00E2	â
0003	ETX	0023	#	0043	C	0063	c	0083	Ctrl	00A3	£	00C3	Ã	00E3	ã
0004	EOT	0024	$	0044	D	0064	d	0084	Ctrl	00A4	¤	00C4	Ä	00E4	ä
0005	ENQ	0025	%	0045	E	0065	e	0085	Ctrl	00A5	¥	00C5	Å	00E5	å
0006	ACK	0026	&	0046	F	0066	f	0086	Ctrl	00A6	¦	00C6	Æ	00E6	æ
0007	BEL	0027	'	0047	G	0067	g	0087	Ctrl	00A7	§	00C7	Ç	00E7	ç
0008	BS	0028	(	0048	H	0068	h	0088	Ctrl	00A8	¨	00C8	È	00E8	è
0009	HT	0029	)	0049	I	0069	i	0089	Ctrl	00A9	©	00C9	É	00E9	é
000A	LF	002A	*	004A	J	006A	j	008A	Ctrl	00AA	ª	00CA	Ê	00EA	ê
000B	VT	002B	+	004B	K	006B	k	008B	Ctrl	00AB	«	00CB	Ë	00EB	ë
000C	FF	002C	´	004C	L	006C	l	008C	Ctrl	00AC	¬	00CC	Ì	00EC	ì
000D	CR	002D	-	004D	M	006D	m	008D	Ctrl	00AD	–	00CD	Í	00ED	í
000E	SO	002E	.	004E	N	006E	n	008E	Ctrl	00AE	®	00CE	Î	00EE	î
000F	SI	002F	/	004F	O	006F	o	008F	Ctrl	00AF	¯	00CF	Ï	00EF	ï
0010	DLE	0030	0	0050	P	0070	p	0090	Ctrl	00B0	°	00D0	Ð	00F0	¶
0011	DC1	0031	1	0051	Q	0071	q	0091	Ctrl	00B1	±	00D1	Ñ	00F1	ñ
0012	DC2	0032	2	0052	R	0072	r	0092	Ctrl	00B2	²	00D2	Ò	00F2	ò
0013	DC3	0033	3	0053	S	0073	s	0093	Ctrl	00B3	³	00D3	Ó	00F3	ó
0014	DC4	0034	4	0054	T	0074	t	0094	Ctrl	00B4	´	00D4	Ô	00F4	ô
0015	NAK	0035	5	0055	U	0075	u	0095	Ctrl	00B5	µ	00D5	Õ	00F5	õ
0016	SYN	0036	6	0056	V	0076	v	0096	Ctrl	00B6	¶	00D6	Ö	00F6	ö
0017	ETB	0037	7	0057	W	0077	w	0097	Ctrl	00B7	·	00D7	×	00F7	÷
0018	CAN	0038	8	0058	X	0078	x	0098	Ctrl	00B8	¸	00D8	Ø	00F8	ø
0019	EM	0039	9	0059	Y	0079	y	0099	Ctrl	00B9	¹	00D9	Ù	00F9	ù
001A	SUB	003A	:	005A	Z	007A	z	009A	Ctrl	00BA	º	00DA	Ú	00FA	ú
001B	ESC	003B	;	005B	[	007B	{	009B	Ctrl	00BB	»	00DB	Û	00FB	û
001C	FS	003C	<	005C	\	007C	|	009C	Ctrl	00BC	1/4	00DC	Ü	00FC	ü
001D	GS	003D	=	005D	]	007D	}	009D	Ctrl	00BD	1/2	00DD	Ý	00FD	Þ
001E	RS	003E	>	005E	^	007E	~	009E	Ctrl	00BE	3/4	00DE	ý	00FE	þ
001F	US	003F	?	005F	_	007F	DEL	009F	Ctrl	00BF	¿	00DF	§	00FF	ÿ

NUL	Null	SOH	Start of heading	CAN	Cancel	SP	Space
STX	Start of text	EOT	End of transmission	EM	End of medium	DEL	Delete
ETX	End of text	DC1	Device control 1	SUB	Substitute	Ctrl	Control
ENQ	Enquiry	DC2	Device control 2	ESC	Escape	FF	Form feed
ACK	Acknowledge	DC3	Device control 3	FS	File separator	CR	Carriage return
BEL	Bell	DC4	Device control 4	GS	Group separator	SO	Shift out
BS	Backspace	NAK	Negative acknowledge	RS	Record separator	SI	Shift in
HT	Horizontal tab	NBS	Non-breaking space	US	Unit separator	DLE	Data link escape
LF	Line feed	ETB	End of transmission block	SYN	Synchronous idle	VT	Vertical tab

Bảng mã UNICODE

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 1

1. Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ?
2. Đặc trưng cơ bản của các máy tính thế hệ thứ nhất?
3. Đặc trưng cơ bản của các máy tính thế hệ thứ hai?
4. Đặc trưng cơ bản của các máy tính thế hệ thứ ba?
5. Đặc trưng cơ bản của các máy tính thế hệ thứ tư?
6. Khuynh hướng phát triển của máy tính điện tử ngày nay là gì?
7. Việc phân loại máy tính dựa vào tiêu chuẩn nào?
8. Khái niệm thông tin trong máy tính được hiểu như thế nào?
9. Lượng thông tin là gì ?
10. Sự hiểu biết về một trạng thái trong 4096 trạng thái có thể có ứng với lượng thông tin là bao nhiêu?
11. Điểm chung nhất trong các cách biểu diễn một số nguyên n bit có dấu là gì?
12. Số nhị phân 8 bit $(11001100)_2$, số này tương ứng với số nguyên thập phân có dấu là bao nhiêu nếu số đang được biểu diễn trong cách biểu diễn:
    a. Dấu và trị tuyệt đối.
    b. Số bù 1.
    c. Số bù 2.
13. Đổi các số sau đây:
    a. $(011011)_2$ ra số thập phân.
    b. $(-2005)_{10}$ ra số nhị phân 16 bits.
    c. $(55.875)_{10}$ ra số nhị phân.
13. chuyển các số sau sang hệ nhị phân:
    a. 15,25
    b.134,357
    c.124

Chương 2: GIAO TIẾP VẬT LÝ

Mã chương: MH12 – 02.

Mục đích: Giới thiệu các thành phần cơ bản của một hệ thống máy tính, khái niệm về kiến trúc máy tính, tập lệnh. Giới thiệu các kiểu kiến trúc máy tính, các kiểu định vị được dùng trong kiến trúc, loại và chiều dài của toán hạng, tác vụ mà máy tính có thể thực hiện. Kiến trúc RISC (Reduced Instruction Set Computer): mô tả kiến trúc, các kiểu định vị. Giới thiệu tổng quát tập lệnh của các kiến trúc máy tính.

Yêu cầu :Sinh viên có kiến thức về các thành phần cơ bản của một hệ thống máy tính, khái niệm về kiến trúc máy tính, tập lệnh. Nắm vững các kiến thức về các kiểu kiến trúc máy tính, các kiểu định vị được dùng trong kiến trúc, loại và chiều dài của toán hạng, tác vụ mà máy tính có thể thực hiện. Phân biệt được hai loại kiến trúc: CISC (Complex Instruction Set Computer), RISC (Reduced Instruction Set Computer). Các kiến thức cơ bản về kiến trúc RISC, tổng quát tập lệnh của các kiến trúc máy tính.

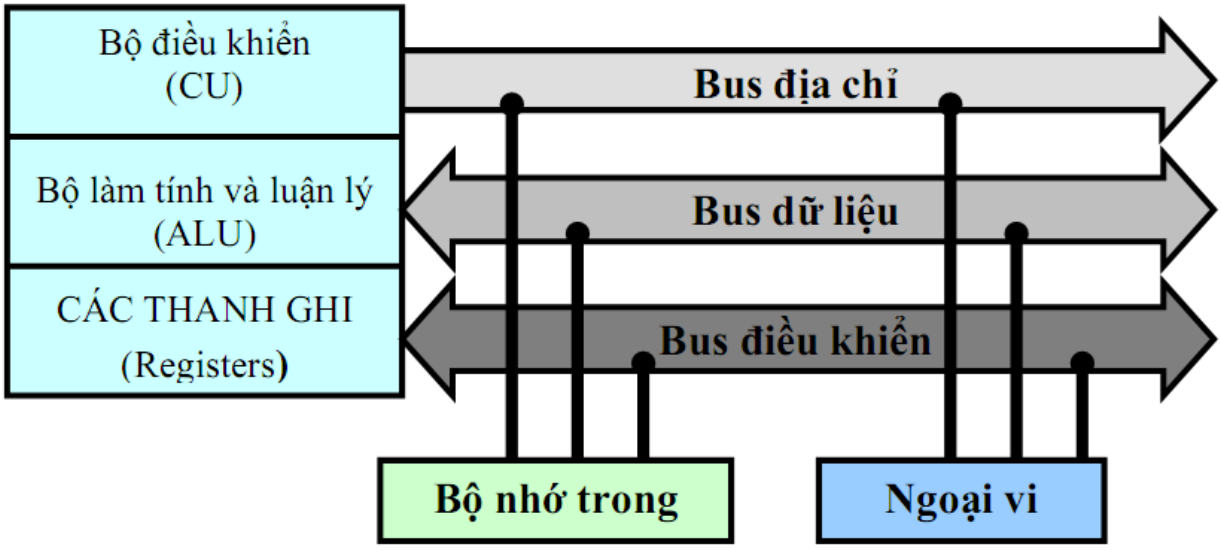
1. Các thành phần cơ bản của máy tính

Mục đích:

- *Giới thiệu các thành phần cơ bản của một hệ thống máy tính*

Thành phần cơ bản của một bộ máy tính gồm: bộ xử lý trung tâm (CPU: Central Processing Unit), bộ nhớ trong, các bộ phận nhập-xuất thông tin. Các bộ phận trên được kết nối với nhau thông qua các hệ thống bus. Hệ thống bus bao gồm: bus địa chỉ, bus dữ liệu và bus điều khiển. Bus địa chỉ và bus dữ liệu dùng trong việc chuyển dữ liệu giữa các bộ phận trong máy tính. Bus điều khiển làm cho sự trao đổi thông tin giữa các bộ phận được đồng bộ. Thông thường người ta phân biệt một bus hệ thống dùng trao đổi thông tin giữa CPU và bộ nhớ trong (thông qua cache), và một bus vào-ra dùng trao đổi thông tin giữa các bộ phận vào-ra và bộ nhớ trong.

1.1 Bộ xử lý trung tâm (CPU)



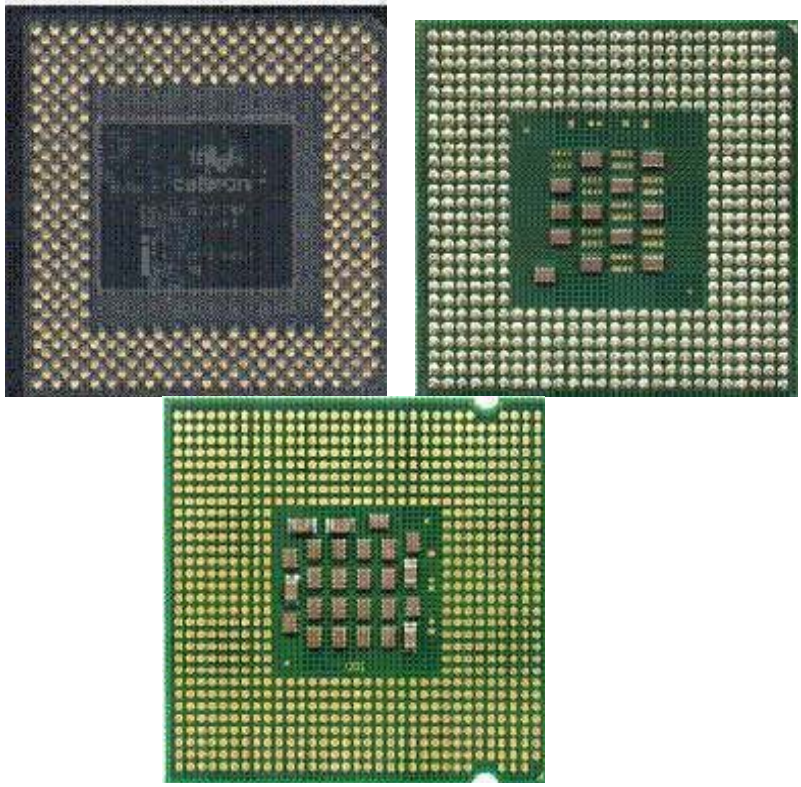
Hình 2-1. Cấu trúc bộ xử lý trung tâm của một hệ máy tính đơn giản

Một chương trình sẽ được sao chép từ đĩa cứng vào bộ nhớ trong cùng với các thông tin cần thiết cho chương trình hoạt động, các thông tin này được nạp vào bộ nhớ trong từ các bộ phận cung cấp thông tin (ví dụ như một bàn phím hay

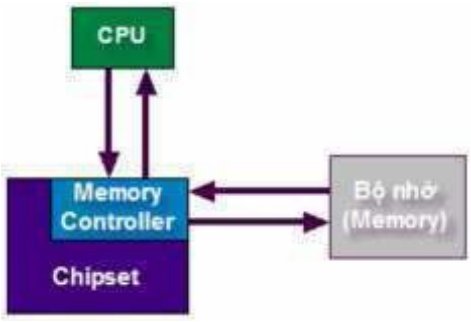
một đĩa từ). Bộ xử lý trung tâm sẽ đọc các lệnh và dữ liệu từ bộ nhớ, thực hiện các lệnh và lưu các kết quả trở lại bộ nhớ trong hay cho xuất kết quả ra bộ phận xuất thông tin (màn hình hay máy in).

Bộ xử lý trung tâm (CPU) là bộ phận thi hành lệnh. CPU lấy lệnh từ bộ nhớ trong và lấy các số liệu mà lệnh đó xử lý. Bộ xử lý trung tâm gồm có hai phần: phần thi hành lệnh và phần điều khiển. Phần thi hành lệnh bao gồm bộ làm toán và luận lý (ALU: Arithmetic And Logic Unit) và các thanh ghi. Nó có nhiệm vụ làm các phép toán trên số liệu. Phần điều khiển có nhiệm vụ đảm bảo thi hành các lệnh một cách tuần tự và tác động các mạch chức năng để thi hành các lệnh.

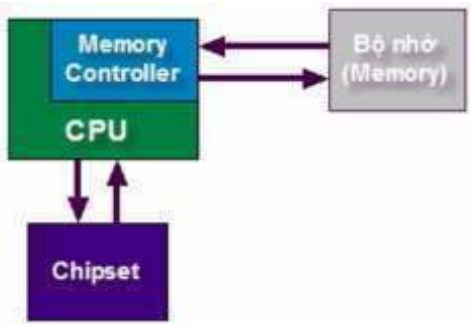
- ALU- bộ xử lý số học, thực hiện các phép tính số học, như phép cộng (+) trừ (-), nhân, chia và các phép logic như logic AND, OR, NOT, XOR.



Hình 2-2: Hình ảnh một số loại CPU Petium4



Hình 2-3. Bố trí memory kiểu Intel



Hình 2-4. Bố trí memory kiểu AMD

1.2 Bo mạch chủ (Mainboard)

Mainboard là trung tâm điều khiển mọi hoạt động của một máy tính và đóng vai trò là trung gian giao tiếp giữa CPU và các thiết bị khác của máy tín. Bản mạch chính là nơi để chứa đựng (cắm) những linh kiện điện tử và những chi tiết quan trọng nhất của một máy tính như: CPU (bộ vi xử lý Central Processing Unit), hệ thống BUS, Bộ nhớ (RAM), các thiết bị lưu trữ (đĩa cứng, ổ CD, …), các Card (card màn hình, card mạng, card âm thanh) và các vi mạch hỗ trợ.

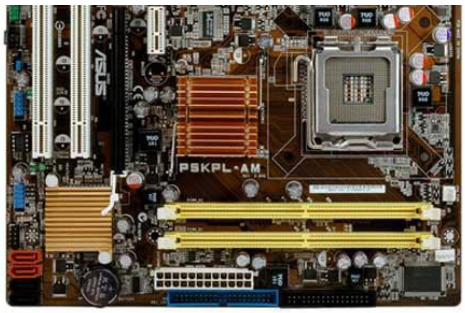
Form factor

Đặc tính này qui định kích thước của mainboard cũng như cách bố trí nó trong thân máy (case). Chuẩn thống trị hiện nay trên máy tính để bàn nói chung chính là ATX (Advanced Technology Extended) 12V, được thiết kế bởi Intel vào năm 1995 và đã nhanh chóng thay thế chuẩn AT, việc kích hoạt chế độ bật được thực hiện qua công tắc có bốn điểm tiếp xúc điện thì với bộ nguồn ATX ta có thể bật tắt bằng phần mềm hay chỉ cần nối mạch hai chân cắm kích nguồn. Các nguồn ATX chuẩn luôn có công tắc tổng để có thể ngắt hoàn toàn dòng điện ra khỏi máy tính. Ngoài ra còn có micro ATX có kích thước nhỏ hơn ATX. Hình 2.6 cho thấy một dạng của 2 loại mainboard này

P5KPL-AM

ASUS P5KPL-AM -Intel G31 chipset (Core 2 Duo & Quadcore)

BIOSTAR G31D-M7 - Intel G31 chipset (Core 2 Quad) - 2 x DDR2 800



Hình 2-5.Hình ảnh một số loại main hiện nay

BTX – vào năm 2004, Intel bắt đầu sản xuất loại mainboard BTX (Balanced Technology Extended). BTX và thùng máy mới sẽ sử dụng ít quạt hơn nên máy tính chạy êm hơn và có khả năng nhiệt độ cũng thấp hơn những hệ thống dùng chuẩn ATX (Advanced Technology Extended) hiện nay. Do vậy, bo mạch BTX có nhiều thay đổi đáng kể trong cách bố trí các thành phần và thiết kế tản nhiệt.

+ Gắn kết các thành phần trên một hệ thống máy tính lại với nhau
+ Điều khiển thay đổi tố độ BUS cho phụ hợp với các thành phần khác nhau
+ Quản lý nguồn cấp cho các thành phần trên Main

+ Cung cấp xung nhịp chủ (xung Clock) để đồng bộ sự hoạt động của toàn hệ thống

Chính vì những chức năng quan trọng trên mà khi Main có sự cố thì máy tính không thể hoạt động được.

Ví dụ: Mainboard :ASUS Intel 915GV P5GL-MX, Socket 775/ s/p

3.8Ghz/ Bus 800/ Sound& Vga, Lan onboard/PCI Express 16X/Dual 4DDR400/ 3 PCI/ 4 SATA/ 8 USB 2.0. có nghĩa là

- ASUS Intel 915GV P5GL-MX, đơn giản đây chỉ là tên của loại bo mạch chỉ của hãng Asus.

- Socket 775 như đã nói ở trên, là loại khe cắm cho CPU

- S/p 3.8 Ghz đó chính là tốc độ xung đồng hồ tối đa của CPU mà bo mạch chủ hỗ trợ.

- BUS 800, chỉ tần số hoạt động tối đa của đường giao tiếp dữ liệu của CPU mà bo mạch chủ hỗ trợ. Thường thì bus tốc độ cao sẽ hỗ trợ luôn các CPU chạy ở bus thấp hơn.

- PCI Express 16X là tên của loại khe cắm card màn hình và mạch chủ. Con số 16 thể hiện một cách tương đối băng thông giao tiếp qua khe cắm, so với AGP 8X, 4X mà ta có thể thấy trên một số bo mạch cũ. Tuy băng thông trên lý thuyết là gấp X lần , thế nhưng tốc độ hoạt động thực tế không phải như vậy mà còn phụ thuộc vào rất nhiều yếu tố khác như dung lượng RAM, loại CPU…

- Sound & VGA, LAN onboard: bo mạch này đã được tích hợp sẵn card âm thanh, card màn hình, và card mạng

- 3PCI, 4SATA, 8 USB 2.0: trên bo mạch chủ này có 3 khe cắm PCI dành để lắp thêm các thiêt bị giao tiêp với máy tính như card âm thanh, modem gắn trong... 4SATA là 4 khe cắm SATA, một loại chuẩn

giao tiếp dành cho đĩa cứng. SATA thì nhanh hơn và ổn định hơn so với chuẩn IDE. 8 cổng cắm USB 2.0 được hỗ trợ trên bo mạch chủ.

1.3 Bộ nhớ trong

Bộ nhớ trong là một tập hợp các ô nhớ, mỗi ô nhớ có một số bit nhất định và chức một thông tin được mã hoá thành số nhị phân mà không quan tâm đến kiểu của dữ liệu mà nó đang chứa. Các thông tin này là các lệnh hay số liệu. Mỗi ô nhớ của bộ nhớ trong đều có một địa chỉ. Thời gian thâm nhập vào một ô nhớ bất kỳ trong bộ nhớ là như nhau. Vì vậy, bộ nhớ trong còn được gọi là bộ nhớ truy cập ngẫu nhiên (RAM: Random Access

Memory). Độ dài của một từ máy tính (Computer Word) là 32 bit (hay 4 byte), tuy nhiên dung lượng một ô nhớ thông thường là 8 bit (1 Byte).

1.4 Thiết bị lưu trữ

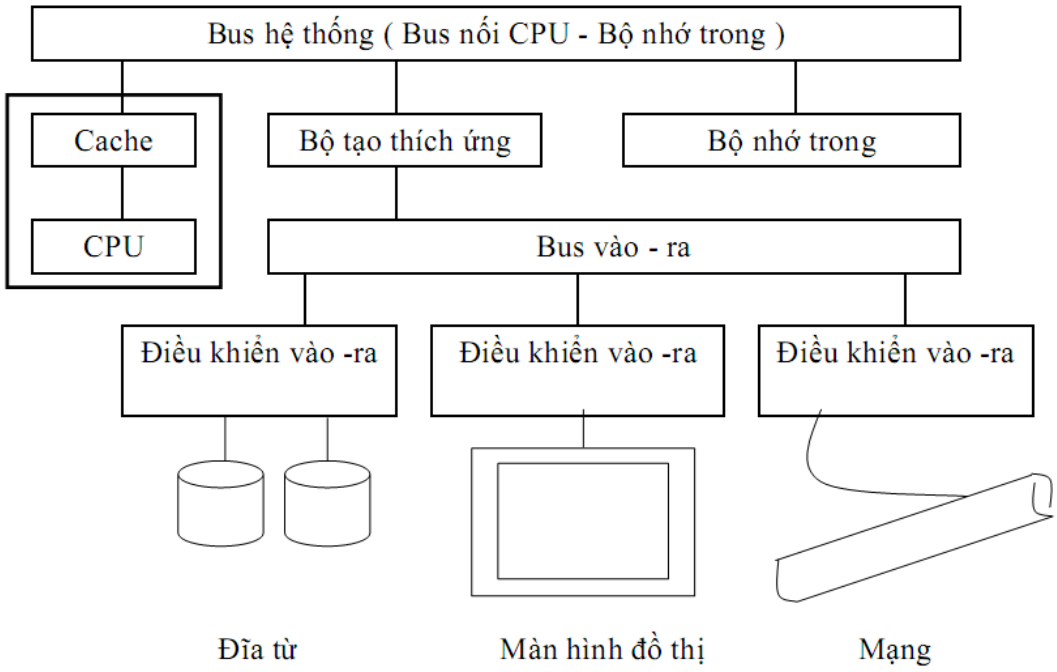
Dùng để lưu trữ thông tin hay còn gọi là bộ nhớ ngoài, thiết bị lưu trữ điển hình nhất là:

+ Đĩa mềm (floppy disk)
+ Đĩa cứng (hard disk)
+ Băng từ (magnetic tape)
+ Thẻ nhớ
+ USB

1.5 Thiết bị nhập xuất

Bộ phận vào – ra là bộ phận xuất nhập thông tin, bộ phận này thực hiện sự giao tiếp giữa máy tính và người dùng hay giữa các máy tính trong hệ thống

mạng (đối với các máy tính được kết nối thành một hệ thống mạng). Các bộ phận xuất nhập thường gặp là: bộ lưu trữ ngoài, màn hình, máy in, bàn phím, chuột, máy quét ảnh, các giao diện mạng cục bộ hay mạng diện rộng...Bộ tạo thích ứng là một vi mạch tổng hợp (chipset) kết nối giữa các hệ thống bus có các tốc độ dữ liệu khác nhau.



Hình 2-6: Sơ đồ mô tả hoạt động điển hình của một máy tính

2. *Định nghĩa kiến trúc máy tính*

Mục tiêu:

- *Khái niệm về kiến trúc máy tính, tập lệnh.*
- *Giới thiệu các kiểu kiến trúc máy tính, các kiểu định vị được dùng trong kiến trúc, loại và chiều dài của toán hạng, tác vụ mà máy tính có thể thực hiện.*

Kiến trúc máy tính bao gồm ba phần: *Kiến trúc phần mềm, tổ chức của máy tính* và *lắp đặt phần cứng.*

- Kiến trúc phần mềm của máy tính chủ yếu là kiến trúc phần mềm của bộ xử lý, bao gồm: tập lệnh, dạng các lệnh và các kiểu định vị.

+ Trong đó, tập lệnh là tập hợp các lệnh mã máy (mã nhị phân) hoàn chỉnh có thể hiểu và được xử lý bởi bộ xử lý trung tâm, thông thường các lệnh trong tập lệnh được trình bày dưới dạng hợp ngữ. Mỗi lệnh chứa thông tin yêu cầu bộ xử lý thực hiện, bao gồm: mã tác vụ, địa chỉ toán hạng nguồn, địa chỉ toán hạng kết quả, lệnh kế tiếp (thông thường thì thông tin này ẩn).

+ Kiểu định vị chỉ ra cách thức thâm nhập toán hạng.

Kiến trúc phần mềm là phần mà các lập trình viên hệ thống phải nắm vững để việc lập trình hiểu quả, ít sai sót.

- Phần tổ chức của máy tính liên quan đến cấu trúc bên trong của bộ xử lý, cấu trúc các bus, các cấp bộ nhớ và các mặt kỹ thuật khác của máy tính. Phần này sẽ được nói đến ở các chương sau.

- Lắp đặt phần cứng của máy tính ám chỉ việc lắp ráp một máy tính dùng các linh kiện điện tử và các bộ phận phần cứng cần thiết. Chúng ta không nói đến phần này trong giáo trình.

Ta nên lưu ý rằng một vài máy tính có cùng kiến trúc phần mềm nhưng phần tổ chức là khác nhau (VAX- 11/780 và VAX 8600). Các máy VAX- 11/780 và VAX- 11/785 có cùng kiến trúc phần mềm và phần tổ chức gần giống nhau. Tuy nhiên việc lắp đặt phần cứng các máy này là khác nhau. Máy VAX- 11/785 đã dùng các mạch kết hiện đại để cải tiến tần số xung nhịp và đã thay đổi một ít tổ chức của bộ nhớ trong.

3. Tập lệnh

Mục tiêu:

- *Giới thiệu tổng quát tập lệnh của các kiến trúc máy tính.*
- *Trình bày được các thanh ghi của bộ vi xử lý 8086*

3.1 Tập các thanh ghi (của bộ vi xử lý 8086)

a. *Các thanh ghi dữ liệu*

Mặc dù bộ vi xử lý có thể thao tác với dữ liệu bộ nhớ nhưng một lệnh như vậy sẽ được thực hiện nhanh hơn (cần ít chu kỳ đồng hồ hơn), nếu dữ liệu được lưu trong các thanh ghi. Đó cũng là nguyên nhân tại sao ngày nay các bộ vi xử lý được sản xuất với xu hướng có nhiều thanh ghi hơn.

Với các thanh ghi dữ liệu các byte thấp và byte cao có thể truy nhập một cách riêng biệt, sử dụng từng 8 bít một cách riêng rẽ. Byte cao của thanh ghi **AX** được gọi là **AH** (Height) và byte thấp được gọi là **AL** (Lower). Tương tự cho các thanh ghi **BX, CX, DX** có **BH, BL, CH, CL, DH, DL**

Chức năng chuyên biệt của từng thanh ghi dữ liệu:

* Thanh ghi AX (thanh ghi chứa – Accumulator register)

AX là thanh ghi được sử dụng nhiều nhất trong các lệnh số học, logic, và chuyển dữ liệu bởi vì việc sử dụng chúng tạo ra mã máy ngắn nhất.

Trong các phép toán nhân chia một trong các số hạng tham gia phải được chứa trong thanh ghi **AX** (nếu là 16 bit) và **AL** (nếu là 8 bit). Các thao tác vào ra cũng sử dụng thanh ghi **AX** hoặc **AL.**

* Thanh ghi **BX** (thanh ghi cơ sở - Base register)

Thanh ghi này ngoài việc thao tác dữ liệu nó thường chứa địa chỉ cơ sở của một bảng dùng cho lệnh **XLAT** (dịch **AL** thành 1 giá trị trong bảng **BX**)

* Thanh ghi **CX** (thanh ghi đếm - Count register)

Việc xây dựng một chương trình lặp được thực hiện dễ dàng bằng cách sử dụng thanh ghi **CX**, trong đó **CX** đóng vai trò bộ đếm vòng lặp (**REP, LOOP**). **CL** được dùng làm bộ đếm trong các lệnh dịch và quay bit.

* Thanh ghi **DX** (thanh ghi dữ liệu - Data register)

DX và **AX** cùng được sử dụng trong các thao tác của phép nhân hoặc chia các số 16 bit. **DX** còn được sử dụng để chứa địa chỉ của các cổng trong các lệnh vào/ ra dữ liệu trực tiếp (**IN/OUT**).

b. *Các thanh ghi đoạn:* ***SC, DS, ES, SS***

Khối BIU đưa ra trên BUS địa chỉ 20 bit địa chỉ, như vậy 8088 có khả năng phân biệt được 2^{20}=1.048.576 = 1 Mbyte ô nhớ. Nói cách khác không gian địa chỉ của 8088 là 1 Mb. Trong không gian 1 Mb bộ nhớ này cần chia thành nhiều đoạn khác nhau:

Đoạn chứa chương trình

Đoạn chứa dữ liệu và kết quả trung gian của chương trình tạo ra vùng nhớ đặc biệt gọi là ngăn xếp

Trong thực tế bộ vi xử lý 8088 có các thanh ghi 16 bit liên quan đến địa chỉ đầu của các đoạn trên và chúng được gọi là các thanh ghi đoạn (Segment Register):**SC, DS, ES, SS.**

Các thanh ghi đoạn này chỉ ra địa chỉ đầu của 4 đoạn trong bộ nhớ dung lượng lớn nhất của 4 đoạn này là 64 Kb. Các đoạn có thể nằm cách nhau hoặc trùm lên nhau.

Nội dung của thanhg ghi sẽ xác định địa chỉ của ô nhớ đầu tiên của đoạn, địa chỉ này gọi là địa chỉ cơ sở. Địa chỉ của các ô nhs khác nhau trong cùng đoạn được tính bằng cách cộng thêm vào địa chỉ cơ sở một giá trị gọi là địa chỉ lệch hay độ lệch (offset)

Địa chỉ vật lý (20bit) của một ô nhớ được xác định như sau:

Địa chỉ vật lý = Địa chỉ đoạn * 10h + thanh ghi lệch (hay offset)

Và địa chỉ logic trong máy tính luôn được biểu diễn dưới dạng: Segment: Offset. Tại mọi thời thì chỉ những ô nhớ được xác định địa chỉ bởi 4 đoạn trên mới được truy cập.

c. *Các thanh ghi con trỏ và chỉ số:* **SI, DI, SP, BP**

Trong 8088 có 3 thanh ghi con trỏ và 2 thanh ghi chỉ số 16 bit. Các thanh ghi này (trừ **IP**), đều có thể được dùng như thanh ghi đa năng, nhưng ứng dụng chính của mỗi thanh ghi là chúng được ngầm định như thanh ghi lệch cho các đoạn tương ứng. Cụ thể như sau:

* Thanh ghi **BP**: (con trỏ cơ sở - Base Pointer)

BP luôn trỏ vào một dữ liệu nằm trong đoạn ngăn xếp **SS.** Địa chỉ cụ thể **SS:BP** được xác định như trên.

* Thanh ghi **SP**: (con trỏ ngăn xếp - Stack Pointer)

Được sử dụng kết hợp với SS để truy nhập vào đoạn ngăn xếp.**S**P luôn trỏ vào đỉnh hiện thời của một ngăn xếp trong đoạn ngăn xếp **SS.** Địa chỉ cụ thể **SS:SP.**

* Thanh ghi **SI** (chỉ số nguồn – Source Index)

SI chỉ vào dữ liệu trong đoạn dữ liệu **DS** mà địa chỉ cụ thể tương ứng với **DS:SI**. Bằng cách tăng nội dung của **SI** chúng ta có thể truy nhập dễ dàng đến ô nhớ liên tiếp.

*. Thanh ghi **DI** (chỉ số đích – Destination Index)

DI chỉ vào dữ liệu trong đoạn dữ liệu **DS** mà địa chỉ cụ thể tương ứng với **DS:DI**. Có một số lệnh gọi là thao tác chuỗi sử dụng **DI** để truy cập đến các ô nhớ được định địa chỉ bởi **ES**.

d. *Thanh ghi con trỏ lệnh*: **IP**

Các thanh ghi bộ nhớ chúng ta vừa trình bày dùng để truy cập dữ liệu, để truy nhập đến các lệnh, 8088 sử dụng các thanh ghi **CS** và **IP**. Thanh ghi **CS** chứa địa chỉ của lệnh tiếp theo còn **IP** chứa địa chỉ offset của lệnh đó. Thanh ghi **IP** được cập nhật mỗi khi có lệnh được thực hiện.

e. Thanh ghi cờ

Đây là thanh ghi 16, mỗi bit được sử dụng để htể hiệnc một trạng thái của bộ vi xử lý tại một thời điểm nhất định trong quá trình thực hiện chương trình. Mới chỉ có 9 bit được sử dụng và người gọi mỗi bít là một cờ

x x x x **OF DF IF TF SF ZF** x **AF** x **PF** x **CF**

* Các cờ trạng thái

CF (Carry Flag): được thiết lập khi phép toán thực hiện có nhớ hoặc có vay mượn

PF (Parity Flag): được thiết lập khi kết quả của phép toán có tổng số bit có giá trị 1 là một số chẵn (ở phần thấp của kết quả)

AF (Auxiliary Flag): được thiết lập khi có nhớ từ *"bít có trọng số lớn nhất ở phần thấp"* sang *" bít có trọng số nhỏ nhất ở phần cao"*.

ZF (Zero Flag): được thiết lập khi kết quả bằng 0

SF (Sign Flag): được thiết lập khi bit MSB (bit dấu) của kết quả có giá trị 1

OF (Overflow Flag): được thiết lập khi kết quả nằm ngoài giới hạn cho phép

* Các cờ điều khiển

TF (Trace Flag) : nếu bit này có giá trị 1 thì bộ vi xử lý cho phép thực hiện từng bước chương trình.

IF (Interrupt Flag):nếu bit này có giá trị 1 thì bộ vi xử lý cho phép ngắt cứng có thể thực hiện.

DF (Direction Flag):nếu bit này có giá trị 1 thì bộ vi xử lý cho phép duyệt chuỗi từ phải sang trái hoặc từ địa chỉ cao đến địa chỉ thấp.

1.1. Tập lệnh

Tập lệnh của bộ vi xử lý là thành phần cơ bản nhất để máy tính có thể thực hiện các yêu cầu của người sử dụng. Tất cả các thao tác, chương trình do người dùng lập ra đều được bộ vi xử lý thực hiện bằng việc ánh xạ chúng dưới dạng mã máy, mã lệnh riêng của bộ vi xử lý.

Thông thường, với các lập trình viên, các lệnh của bộ vi xử được hiểu dưới góc độ là lệnh gợi nhớ (Mnemonic). Với mã lệnh gợi nhớ thì một lệnh của bộ vi xử lý bao gồm các thành phần sau:

[mã lệnh] [các toán hạng]

Trong đó:

Mã lệnh: cơ bản trường này chứa mã lệnh dưới dạng mã gợi nhớ (tên lệnh)

Các toán hạng: là các thành phần mà các lệnh sử dụng để thực hiện lệnh.

Ví dụ:

ADD AL, [BX]

Giải thích:

ADD là mã lệnh (lệnh thực hiện phép cộng)

Al, [BX] là 2 toán hạng với quy định nếu tên thanh ghi hoặc một giá trị hằng nằm trong dấu ngoặc vuông [ ] thì đó là địa chỉ offset của ô nhớ chứa dữ liệu cần thao tác.

Trong tập lệnh của bộ vi xử lý có chứa rất nhiều lệnh, mỗi lệnh thực hiện một nhiệm vụ cụ thể nào đó. Song, trong giới hạn nhất định, chúng ta có thể nghiên cứu một vài lệnh cơ bản.

Để dễ hiểu chúng ta có thể chia chúng thành các nhóm lệnh sau:

a. Nhóm lệnh di chuyển dữ liệu

Trong nhóm này ta quan tâm đến một số lệnh cơ bản sau: **MOV, MOVSB, MOVSW, XCHG, PUSH, POP**

b. Nhóm lệnh số học

Một số lệnh cơ bản trong nhóm lệnh số học: **ADD, ADC, INC, SUB, SBB, DEC MUL (IMUL), DIV (IDIV)**

c. Nhóm lệnh logic

Một số lệnh cơ bản trong nhóm lệnh logic: **AND, OR, NOT, XOR.**

d. Nhóm lệnh dịch chuyển và quay

Một số lệnh cơ bản trong nhóm lệnh dịch chuyển và quay: **SHL, SHR, ROL,ROR**

e. Nhóm lệnh rẽ nhánh

Một số lệnh cơ bản trong nhóm lệnh rẽ nhánh: lệnh nhảy có điều kiện và lệnh nhảy không điều kiện (JMP nhãn_đích)

f. Nhóm lệnh vào ra cổng

Một số lệnh cơ bản trong nhóm lệnh vào ra cổng: **IN, OUT**

g. Nhóm lệnh điều khiển

Một số lệnh cơ bản trong nhóm lệnh điều khiển: **CALL, INT, HLT, NOP**

Cụ thể các cú pháp các lệnh trong từng nhóm chúng ta sẽ nghiên cứu tại chương 6: Ngôn ngữ Assembly

4. Kiến trúc RISC

Mục tiêu:

- *Kiến trúc RISC (Reduced Instruction Set Computer): mô tả kiến trúc, các kiểu định vị.*

4.1 Giới thiệu

Các kiến trúc với tập lệnh phức tạp CISC (Complex Instruction Set Computer) được nghĩ ra từ những năm 1960. Vào thời kỳ này, người ta nhận thấy các chương trình dịch khó dùng các thanh ghi, rằng các vi lệnh được thực hiện nhanh hơn các lệnh và cần thiết phải làm giảm độ dài các chương trình. Các đặc tính nầy khiến người ta ưu tiên chọn các kiểu ô nhớ - ô nhớ và ô nhớ - thanh ghi, với những lệnh phức tạp và dùng nhiều kiểu định vị. Điều này dẫn tới việc các lệnh có chiều dài thay đổi và như thế thì dùng bộ điều khiển vi chương trình là hiệu quả nhất.

Bảng II.6 cho các đặc tính của vài máy CISC tiêu biểu. Ta nhận thấy cả ba máy đều có điểm chung là có nhiều lệnh, các lệnh có chiều dài thay đổi. Nhiều cách thực hiện lệnh và nhiều vi chương trình được dùng.

Tiến bộ trong lãnh vực mạch kết (IC) và kỹ thuật dịch chương trình làm cho các nhận định trước đây phải được xem xét lại, nhất là khi đã có một khảo sát định lượng về việc dùng tập lệnh các máy CISC.

Bảng 2-1. Đặc tính một vài máy CISC

Bộ xử lý	IBM 370/168	DEC 11/780	iAPX 432
Năm sản xuất	1973	1978	1982
Số lệnh	208	303	222
Bộ nhớ vi chương trình	420 KB	480 KB	64 KB
Chiều dài lệnh (tính bằng bit)	16 - 48	16 - 456	6 - 321
Kỹ thuật chế tạo	ECL - MSI	TTl - MSI	NMOS VLSI
Cách thực hiện lệnh	Thanh ghi- thanh ghi Thanh ghi - bộ nhớ Bộ nhớ - bộ nhớ	Thanh ghi - thanh ghi Thanh ghi - bộ nhớ Bộ nhớ - bộ nhớ	Ngăn xếp Bộ nhớ- bộ nhớ
Dung lượng cache	64 KB	64 KB	0

Ví dụ, chương trình dịch đã biết sử dụng các thanh ghi và không có sự khác biệt đáng kể nào khi sử dụng ô nhớ cho các vi chương trình hay ô nhớ cho các chương trình. Điều này dẫn tới việc đưa vào khái niệm về một máy tính với tập lệnh rút gọn RISC vào đầu những năm 1980. Các máy RISC dựa chủ yếu trên một tập lệnh cho phép thực hiện kỹ thuật ống dẫn một cách thích hợp nhất bằng cách thiết kế các lệnh có chiều dài cố định, có dạng đơn giản, dễ giải mã. Máy RISC dùng kiểu thực hiện lệnh thanh ghi - thanh ghi. Chỉ có các lệnh ghi hoặc đọc ô nhớ mới cho phép thâm nhập vào ô nhớ. Bảng II.7 diễn tả ba mẫu máy RISC đầu tiên: mẫu máy của IBM (IBM 801) của Berkeley (RISC1 của

Patterson) và của Stanford (MIPS của Hennessy). Ta nhận thấy cả ba máy đó đều có bộ điều khiển bằng mạch điện (không có ô nhớ vi chương trình), có chiều dài các lệnh cố định (32 bits), có một kiểu thi hành lệnh (kiểu thanh ghi - thanh ghi) và chỉ có một số ít lệnh.

Bảng 2-2. Đặc tính của ba mẫu đầu tiên máy RISC

Bộ xử lý	IBM801	RISC1	MIPS
Năm sản xuất	1980	1982	1983
Số lệnh	120	39	55
Dung lượng bộ nhớ vi chương trình	0	0	0
Độ dài lệnh (tính bằng bít)	32	32	32
Kỹ thuật chế tạo	ECL MSI	NMOS VLSI	NMOS VLSI
Cách thực hiện	Thanhghi-Thanh ghi	Thanh ghi– Thanh ghi	Thanhghi-Thanh ghi

Tóm lại, ta có thể định nghĩa mạch xử lý RISC bởi các tính chất sau:

- Có một số ít lệnh (thông thường dưới 100 lệnh).
- Có một số ít các kiểu định vị (thông thường hai kiểu: định vị tức thì và định vị gián tiếp thông qua một thanh ghi).
- Có một số ít dạng lệnh (một hoặc hai)
- Các lệnh đều có cùng chiều dài.
- Chỉ có các lệnh ghi hoặc đọc ô nhớ mới thâm nhập vào bộ nhớ.
- Dùng bộ tạo tín hiệu điều khiển bằng mạch điện để tránh chu kỳ giải mã các vi lệnh làm cho thời gian thực hiện lệnh kéo dài.
- Bộ xử lý RISC có nhiều thanh ghi để giảm bớt việc thâm nhập vào bộ nhớ trong.

Ngoài ra các bộ xử lý RISC đầu tiên thực hiện tất cả các lệnh trong một chu kỳ máy.

Bộ xử lý RISC có các lợi điểm sau :

- Diện tích của bộ xử lý dùng cho bộ điều khiển giảm từ 60% (cho các bộ xử lý CISC) xuống còn 10% (cho các bộ xử lý RISC). Như vậy có thể tích hợp thêm vào bên trong bộ xử lý các thanh ghi, các cổng vào ra và bộ nhớ cache .....
- Tốc độ tính toán cao nhờ vào việc giải mã lệnh đơn giản, nhờ có nhiều thanh ghi (ít thâm nhập bộ nhớ), và nhờ thực hiện kỹ thuật ống dẫn liên tục và có hiệu quả (các lệnh đều có thời gian thực hiện giống nhau và có cùng dạng).
- Thời gian cần thiết để thiết kế bộ điều khiển là ít. Điều này góp phần làm giảm chi phí thiết kế.
- Bộ điều khiển trở nên đơn giản và gọn làm cho ít rủi ro mắc phải sai sót mà ta gặp thường trong bộ điều khiển.

Trước những điều lợi không chối cãi được, kiến trúc RISC có một số bất lợi:

+ Các chương trình dài ra so với chương trình viết cho bộ xử lý CISC. Điều này do các nguyên nhân sau :

+ Cấm thâm nhập bộ nhớ đối với tất cả các lệnh ngoại trừ các lệnh đọc và ghi vào bộ nhớ. Do đó ta buộc phải dùng nhiều lệnh để làm một công việc nhất định.

+ Cần thiết phải tính các địa chỉ hiệu dụng vì không có nhiều cách định vị.

+ Tập lệnh có ít lệnh nên các lệnh không có sẵn phải được thay thế bằng một chuỗi lệnh của bộ xử lý RISC.

+ Các chương trình dịch gặp nhiều khó khăn vì có ít lệnh làm cho có ít lựa chọn để diễn dịch các cấu trúc của chương trình gốc. Sự cứng nhắc của kỹ thuật ống dẫn cũng gây khó khăn.

+ Có ít lệnh trợ giúp cho ngôn ngữ cấp cao.

Các bộ xử lý CISC trợ giúp mạnh hơn các ngôn ngữ cao cấp nhờ có tập lệnh phức tạp. Hãng Honeywell đã chế tạo một máy có một lệnh cho mỗi động từ của ngôn ngữ COBOL. Các tiến bộ gần đây cho phép xếp đặt trong một vi mạch, một bộ xử lý RISC nền và nhiều toán tử chuyên dùng.

Thí dụ, bộ xử lý 860 của Intel bao gồm một bộ xử lý RISC, bộ làm tính với các số lẻ và một bộ tạo tín hiệu đồ hoạ.

4.2 Các kiểu định vị trong các bộ xử lý

Trong bộ xử lý RISC, các lệnh số học và logic chỉ được thực hiện theo kiểu thanh ghi và tức thì, còn những lệnh đọc và ghi vào bộ nhớ là những lệnh có toán hạng bộ nhớ thì được thực hiện với những kiểu định vị khác.

a. Kiểu định vị thanh ghi

Đây là kiểu định vị thường dùng cho các bộ xử lý RISC, các toán hạng nguồn và kết quả đều nằm trong thanh ghi mà số thứ tự được nêu ra trong lệnh. Hình II.5 cho vài ví dụ về kiểu thanh ghi và dạng các lệnh tương ứng trong một vài kiến trúc RISC.

Bảng 2-3 Dạng lệnh trong kiểu định vị thanh ghi – thanh ghi cho một số CPU RISC

MIPS	Opcode 6		nguồn 1 5		nguồn 2 5		Đích 5		Dịch chuyển 5		Hàm 6			
SPARC	Opcode 2	Đích 5		Opcode 6		nguồn 1 5		0 1	Khoảng trống khác 8			nguồn2 5		
Power PC	Opcode 6		Đích 5		nguồn 1 5		nguồn 2 5		Opcode mở rộng 10					0 1

ALPHA	Opcode 6	nguồn 1 5	nguồn 2 5	3	01	Opcode mở rộng 7	Đích 5

b. Kiểu định vị tức thì

Trong kiểu này, toán hạng là một số có dấu, được chứa ngay trong lệnh. Hình II.6 cho ta vài ví dụ về dạng lệnh kiểu tức thì.

Bảng 2-4. Dạng lệnh trong kiểu định vị thanh ghi - tức thì cho một số CPU RISC

MIPS	Opcode 6	Thanh ghi nguồn 5	Thanh ghi đích 5	Số có dấu (toán hạng tức thì) 16			
SPARC	Opcode 2	Thanh ghi đích 5	Opcode 6	Thanh ghi nguồn 5	11	Toán hạng tức thì có dấu 13	
ALPHA	Opcode 6	Thanh ghi nguồn 5	Toán hạng tức thì>0 8	11	Opcode mở rộng 7	Thanh ghi đích 5	
Power PC	Opcode 6	Thanh ghi đích 5	Thanh ghi nguồn 5	Toán hạng tức thì có dấu 16			

c. Kiểu định vị trực tiếp

Trong kiểu này địa chỉ toán hạng nằm ngay trong lệnh (hình II.6). Ví dụ, kiểu định vị trực tiếp được dùng cho các biến của hệ điều hành, người sử dụng không có quyền thâm nhập các biến này.

Bảng 2-5. Dạng lệnh thâm nhập bộ nhớ trong của một số kiến trúc RISC

MIPS	Opcode 6	Thanh ghi địa chỉ 5	Thanh ghi số liệu 5	độ dờicó dấu 16		
SPARC	Opcode	Thanh ghi số	Opcode 6	Thanh ghi địa chỉ	11	độ dời có dấu

	2	liệu 5		5		13
ALPHA	Opcode 6	Thanh ghi số liệu 5	Thanh ghi địa chỉ 5	Độ dời có dấu 16		
Power PC	Opcode 6	Thanh ghi đích 5	Thanh ghi nguồn 5	Độ dời có dấu 16		

d. Kiểu định vị gián tiếp bằng thanh ghi + độ dời

Đây là kiểu đặc thù cho các kiến trúc RISC. Địa chỉ toán hạng được tính như sau :

Địa chỉ toán hạng = Thanh ghi (địa chỉ) + độ dời. Ta để ý rằng kiểu định vị trực tiếp chỉ là một trường hợp đặc biệt của kiểu này khi thanh ghi (địa chỉ) = 0. Trong các bộ xử lý RISC, một thanh ghi (R0 hoặc R31) được mắc vào điện thế thấp (tức là 0) và ta có định vị trực tiếp khi dùng thanh ghi đó như là thanh ghi địa chỉ.

e. Kiểu định vị tự tăng

Một vài bộ xử lý RISC, ví dụ bộ xử lý PowerPC, dùng kiểu định vị này.

5. Toán hạng

Kiểu của toán hạng thường được đưa vào trong mã tác vụ của lệnh. Có bốn kiểu toán hạng được dùng trong các hệ thống:

- Kiểu địa chỉ.
- Kiểu dạng số: số nguyên, dấu chấm động,...
- Kiểu dạng chuỗi ký tự: ASCII, EBIDEC,...
- Kiểu dữ liệu logic: các bit, cờ,...

Tuy nhiên một số ít máy tính dùng các nhãn để xác định kiểu toán hạng.

Thông thường loại của toán hạng xác định luôn chiều dài của nó. Toán hạng thường có chiều dài là byte (8 bit), nữa từ máy tính (16 bit), từ máy tính (32 bit), từ đôi máy tính (64 bit). Đặc biệt, kiến trúc PA của hãng HP (Hewlet Packard) có khả năng tính toán với các số thập phân BCD. Một vài bộ xử lý có thể xử lý các chuỗi ký tự.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 2

*****

1. Các thành phần của một hệ máy tính đơn giản
2.Nêu chức năng của các thanh ghi cơ sở và thanh ghi cờ.
3. Chỉ ra giá trị của các cờ trạng thái sau khi thực hiện các lệnh sau:

```
MOV  AL,3Fh;
     MOV  DL,D7h;
     ADD AL,DL;
```

4. Nêu định nghĩa kiến trúc máy tính

5. Mô tả các kiểu thi hành lệnh của một máy tính. Tại sao kiểu thi hành lệnh thanh ghi – thanh ghi được dùng nhiều hiện tại?

6. Mô tả mỗi kiểu định vị trong các kiểu định vị mà một CPU có thể có. Cho CPU RISC, các kiểu định vị nào thường được dùng nhất?

7. Sự khác biệt giữa CPU RISC và CPU CISC?

Chương 3: TỔ CHỨC BỘ XỬ LÝ

Mã chương: MH12 – 03

Mục đích: Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý: đường đi của dữ liệu, bộ điều khiển tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra. Mô tả diễn tiến thi hành một lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật ống dẫn, siêu ống dẫn, siêu vô hướng,...Một số kỹ thuật xử lý thông tin: ống dẫn, siêu ống dẫn

Yêu cầu: Sinh viên phải nắm vững cấu trúc của bộ xử lý trung tâm và diễn tiến thi hành một lệnh mã máy, vì đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật xử lý thông tin trong máy tính.

1. Đường đi của dữ liệu

Mục đích:

- *Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý: đường đi của dữ liệu tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra.*

Phần đường đi của dữ liệu gồm có bộ phận làm tính và luận lý (ALU: Arithmetic and Logic Unit), các mạch dịch, các thanh ghi và các đường nối kết các bộ phận trên. Phần này chứa hầu hết các trạng thái của bộ xử lý. Ngoài các thanh ghi tổng quát, phần đường đi dữ liệu còn chứa:

Thanh ghi đếm chương trình (PC: Program Counter),

Thanh ghi trạng thái(SR: Status Register),

Thanh ghi đệm TEMP (temporary),

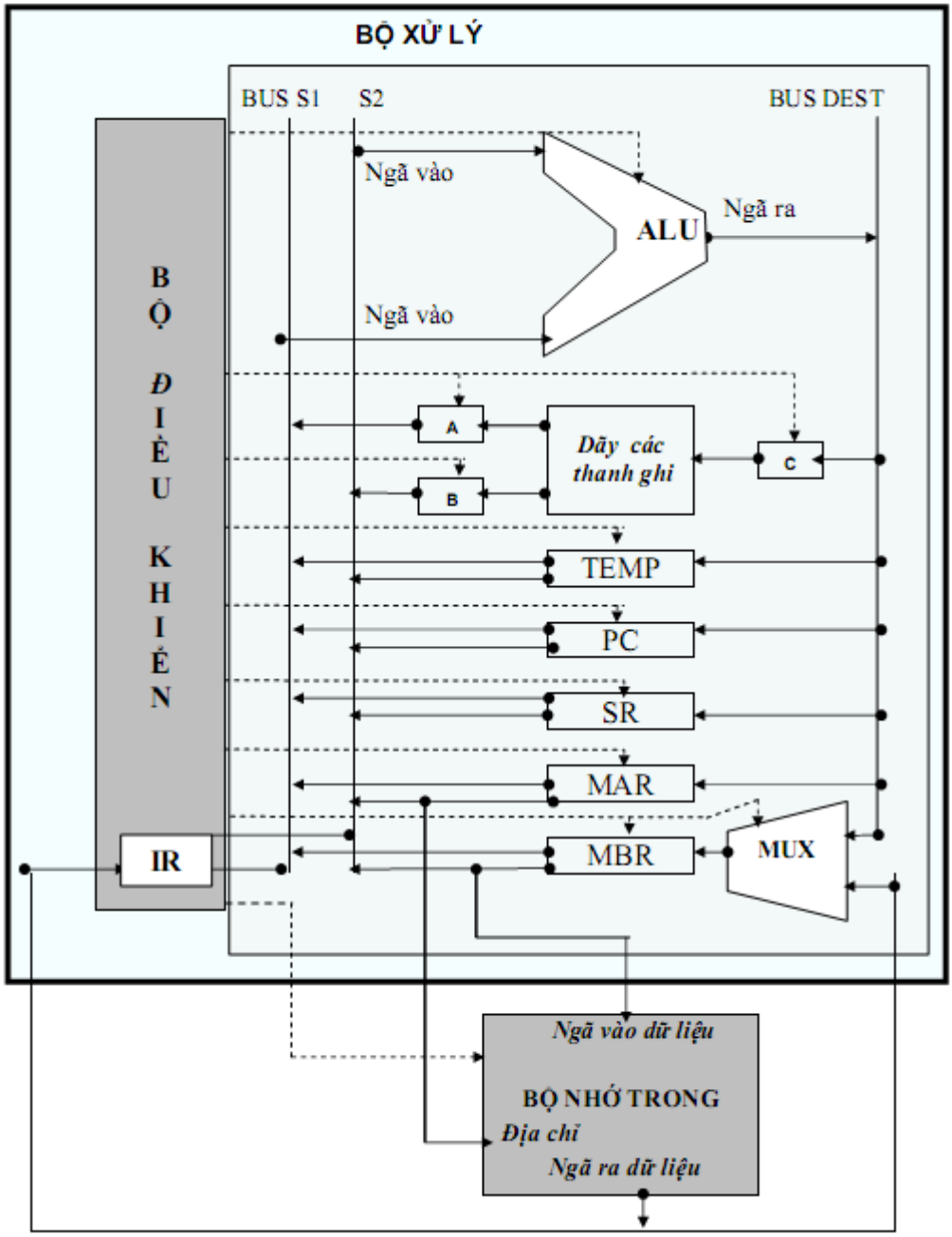
Các thanh ghi địa chỉ bộ nhớ (MAR : Memory Address Register),

Thanh ghi số liệu bộ nhớ (MBR: Memory Buffer Register).

Bộ đa hợp (MUX: Multiplexor) đây là điểm cuối của kênh dữ liệu – CPU và bộ nhớ, với nhiệm vụ lập thời biểu truy cập bộ nhớ từ CPU và các kênh dữ liệu, hệ thống BUS nguồn (S1, S2) và bus kết quả (Dest).

Nhiệm vụ chính của phần đường đi dữ liệu là đọc các toán hạng từ các thanh ghi tổng quát, thực hiện các phép tính trên toán hạng này trong bộ làm tính và luận lý ALU và lưu trữ kết quả trong các thanh ghi tổng quát. Ở ngã vào và ngã ra các thanh ghi tổng quát có các mạch chốt A, B, C. Thông thường, số lượng các thanh ghi tổng quát 32.

Phần đường đi của dữ liệu chiếm phân nửa diện tích của bộ xử lý nhưng là phần dễ thiết kế và cài đặt trong bộ xử lý.



Hình 3-1. Tổ chức của một bộ xử lý điển hình

*(các đường không liên tục là các đường điều khiển)*ALU (Arthmetical and Logical Unit): bộ phận tính toán số học và logic.

PC: Program Counter :thanh ghi đếm chương trình.

SR: Status Register : thanh ghi trạ

Temp: temporary :thanh ghi đệm

MAR: Memory address Register :thanh ghi địa chỉ nhớ

MBR: Memory Buffer Register :thanh ghi số liệu bộ nhớ

MUX: Multiplexor : Bộ đa hợp

BUS S1,S2 : hệ thống nguồn

Interrupt Request): yêu cầu ngắt, một đường hoặc môt tín hiệu được kích hoạt bởi thiết bị ngoại vi để phát ra một ngắt cứng tới CPU.

2. Bộ điều khiển

Mục đích:

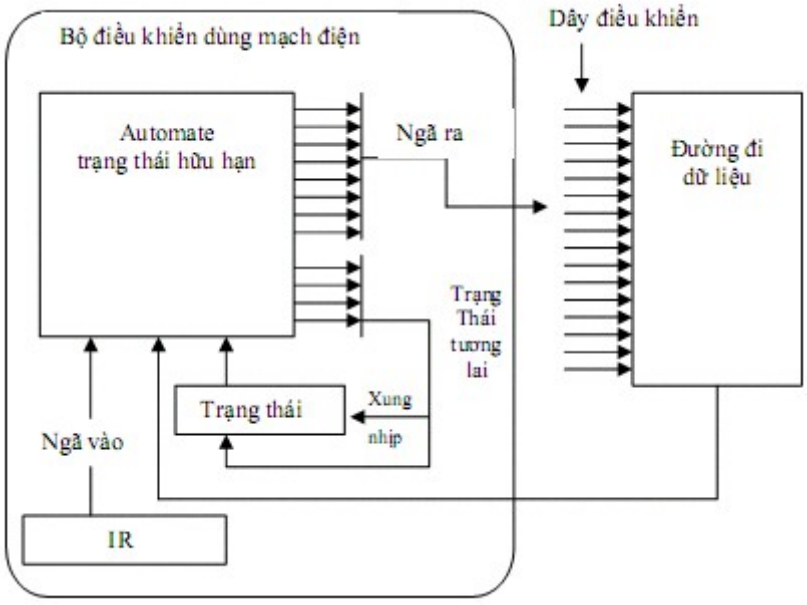
- *Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý: đường đi của dữ liệu, bộ điều khiển tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra.*

Bộ điều khiển tạo các tín hiệu điều khiển di chuyển số liệu (tín hiệu di chuyển số liệu từ các thanh ghi đến bus hoặc tín hiệu viết và vào các thanh ghi, điều khiển các tác vụ mà các bộ phận chức năng phải làm (điều khiển ALU, điều khiển đọc và viết vào bộ nhớ trong…). Bộ điều khiển cũng tạo các tín hiêu giúp lệnh được thực hiện một cách tuần tự.

Việc cài đặt bộ điều khiển có thể dùng một trong 2 cách sau: dùng mạch điện tử hoặc dùng vi chương trình (microprogram)

2.1 Bộ điều khiển mạch điện tử.

Để hiểu được vận hành của bộ điều khiển mạch điện tử, chúng ta xét đến mô tả về Automate trạng thái hữu hạn: có nhiều hệ thống hay nhiều thành phần mà ở mỗi thời điểm xem xét đều có một trạng thái (state). Mục đích của trạng thái là ghi nhớ những gì có liên quan trong quá trình hoạt động của hệ thống. Vì chỉ có một số trạng thái nhất định nên nói chung không thể ghi nhớ hết toàn bộ lịch sử của hệ thống, do vậy nó phải được thiết kế cẩn thận để ghi nhớ những gì quan trọng. Ưu điểm của hệ thống (chỉ có một số hữu hạn các trạng thái) đó là có thể cài đặt hệ thống với một lượng tài nguyên cố định. Chẳng hạn, chúng ta có thể cài đặt Automate trạng thái hữu hạn trong phần cứng máy tính ở dạng mạch điện hay một dạng chương trình đơn giản, trong đó, nó có khả năng quyết định khi chỉ biết một lượng giới hạn dữ liệu hoặc bằng cách dùng vị trí trong đoạn mã lệnh để đưa ra quyết định.

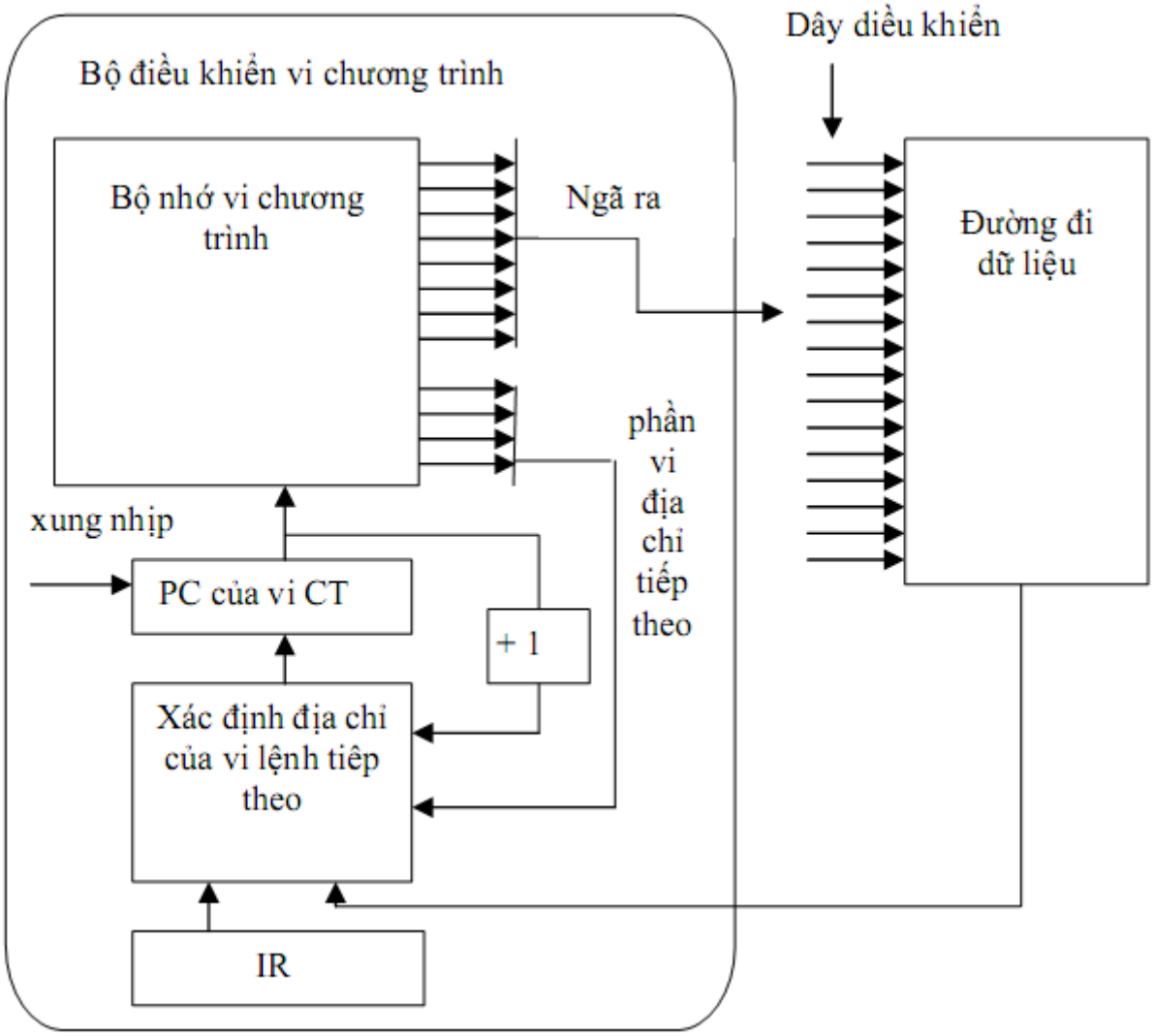


Hình 3-2. Nguyên tắc vận hành của bộ điều khiển dùng mạch điện

Hình 3-2 cho thấy nguyên tắc của một bộ điều khiển bằng mạch điện. Các đường điều khiển của phần đường đi số liệu là các ngã ra của một hoặc nhiều Automate trạng thái hữu hạn. Các ngã vào của Automate gồm có thanh ghi lệnh, thanh ghi này chứa lệnh phải thi hành và những thông tin từ bộ đường đi số liệu. Ứng với cấu hình các đường vào và trạng thái hiện tại, Automate sẽ cho trạng thái tương lai và các đường ra tương ứng với trạng thái hiện tại. Automate được cài đặt dưới dạng là một hay nhiều mạch mảng logic lập trình được (PLA: Programmable Logic Array) hoặc các mạch logic ngẫu nhiên.

Kỹ thuật điều khiển này đơn giản và hữu hiệu khi các lệnh có chiều dài cố định, có dạng thức đơn giản. Nó được dùng nhiều trong các bộ xử lý RISC.

2.2 Bộ điều khiển vi chương trình:



Hình 3-3: Nguyên tắc vận hành của bộ điều khiển vi chương trình

Sơ đồ nguyên tắc của bộ điều khiển dùng vi chương trình được trình bày ở hình 3-3. Trong kỹ thuật này, các đường dây điều khiển của bộ đường đi dữ liệu ứng với các ngã ra của một vi lệnh nằm trong bộ nhớ vi chương trình. Việc điều khiển các tác vụ của một lệnh mã máy được thực hiện bằng một chuỗi các vi lệnh. Một vi máy tính nằm bên trong bộ điều khiển thực hiện từng lệnh của vi chương trình này. Chính vi máy tính này điều khiển việc thực hiện một cách tuần tự các vi lệnh để hoàn thành tác vụ mà lệnh mã máy phải thực hiện. Các tác vụ của lệnh mã máy cũng tuỳ thuộc vào trạng thái của phần đường đi dữ liệu.

Bộ điều khiển bằng vi chương trình được dùng rộng rãi trong các bộ xử lý CISC. Bộ xử lý này có tập lệnh phức tạp với các lệnh có chiều dài khác nhau và có dạng thức phức tạp. Trong các bộ xử lý CISC, người ta cài đặt một lệnh mã máy bằng cách viết một vi chương trình. Như vậy công việc khá đơn giản và rất hữu hiệu. Các sai sót trong thiết kế automat điều khiển cũng dễ sửa đổi.

3. Diễn tiến thi hành lệnh mã máy

Mục đích:

- *Mô tả diễn tiến thi hành một lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật ống dẫn, siêu ống dẫn, siêu vô hướng,...*

Việc thi hành một lệnh mã máy có thể chia thành 5 giai đoạn:

+ *Đọc lệnh* (IF: Instruction Fetch)
+ *Giải mã lệnh* (ID: Instruction Decode)
+ *Thi hành lệnh* (EX: Execute)
+ *Thâm nhập bộ nhớ trong hoặc nhảy* (MEM: Memory access)
+ *Lưu trữ kết quả* (RS: Result Storing).

Mỗi giai đoạn được thi hành trong một hoặc nhiều chu kỳ xung nhịp.

a. Đọc lệnh:

MAR ← PC

IR ← M[MAR]

Bộ đếm chương trình PC được đưa vào MAR . Lệnh được đọc từ bộ nhớ trong, tại các ô nhớ có địa chỉ nằm trong MAR và được đưa vào thanh ghi lệnh IR.

b. Giải mã lệnh và đọc các thanh ghi nguồn:

A ← Rs1

B ← Rs2

PC ← PC + 4

Lệnh được giải mã. Kế đó các thanh ghi Rs1 và Rs2 được đưa vào A và B. Thanh ghi PC được tăng lên để chỉ tới lệnh kế đó.

Để hiểu rõ giai đoạn này, ta lấy dạng thức của một lệnh làm tính tiêu biểu sau đây:

	Mã lệnh	Thanh ghi Rs1	Thanh ghi Rs2	Thanh ghi Rd	Tác vụ
bit	*6*	*5*	*5*	*5*	*11*

Các thanh ghi nguồn Rs1 và Rs2 được sử dụng tuỳ theo tác vụ, kết quả được đặt trong thanh ghi đích Rd.

Ta thấy việc giải mã được thực hiện cùng lúc với việc đọc các thanh ghi Rs1 và Rs2 vì các thanh ghi này luôn nằm tại cùng vị trí ở trong lệnh.

c. Thi hành lệnh:

Tuỳ theo loại lệnh mà một trong ba nhiệm vụ sau đây được thực hiện:

- Liên hệ tới bộ nhớ

MAR ← Địa chỉ do ALU tính tuỳ theo kiểu định vị (Rs2).

MBR ← Rs1

Địa chỉ hiệu dụng do ALU tính được đưa vào MAR và thanh ghi nguồn Rs1 được đưa vào MBR để được lưu vào bộ nhớ trong.

- Một lệnh của ALU

Ngã ra ALU ← Kết quả của phép tính

ALU thực hiện phép tính xác định trong mã lệnh, đưa kết quả ra ngã ra.

- Một phép nhảy

Ngã ra ALU ← Địa chỉ lệnh tiếp theo do ALU tính.

ALU cộng địa chỉ của PC với độ dời để làm thành địa chỉ đích và đưa địa chỉ này ra ngã ra. Nếu là một phép nhảy có điều kiện thì thanh ghi trạng thái được đọc quyết định có cộng độ dời vào PC hay không.

d. Thâm nhập bộ nhớ trong hoặc nhảy lần cuối

Giai đoạn này thường chỉ được dùng cho các lệnh nạp dữ liệu, lưu giữ dữ liệu và lệnh nhảy.

- Tham khảo đến bộ nhớ:

MBR ← M[MAR] hoặc M[MAR] ← MBR

Số liệu được nạp vào MBR hoặc lưu vào địa chỉ mà MAR trỏ đến.

- Nhảy:

If (điều kiện), PC ← ngả ra ALU

Nếu điều kiện đúng, ngã ra ALU được nạp vào PC. Đối với lệnh nhảy không điều kiện, ngả ra ALU luôn được nạp vào thanh ghi PC.

e. Lưu trữ kết quả

Rd ← Ngã ra ALU hoặc Rd ← MBR

Lưu trữ kết quả trong thanh ghi đích.

4. Ngắt (INTERRUPT)

Mục tiêu: *Trình bày được khái niệm ngắt, khi nào thì gọi ngắt.*

Ngắt quãng là một sự kiện xảy ra một cách ngẫu nhiên trong máy tính và làm ngưng tính tuần tự của chương trình (nghĩa là tạo ra một lệnh nhảy). Phần lớn các nhà sản xuất máy tính (ví dụ như IBM, INTEL) dùng từ ngắt quãng để ám chỉ sự kiện này, tuy nhiên một số nhà sản xuất khác dùng từ "ngoại lệ", "lỗi", "bẩy" để chỉ định hiện tượng này.

Bộ điều khiển của CPU là bộ phận khó thực hiện nhất và ngắt quãng là phần khó thực hiện nhất trong bộ điều khiển. Để nhận biết được một ngắt quãng lúc đang thi hành một lệnh, ta phải biết điều chỉnh chu kỳ xung nhịp và điều này có thể ảnh hưởng đến hiệu quả của máy tính.

Người ta đã nghỉ ra "ngắt quãng" là để nhận biết các sai sót trong tính toán số học, và để ứng dụng cho những hiện tượng thời gian thực. Bây giờ, ngắt quãng được dùng cho các công việc sau đây:

+ Ngoại vi đòi hỏi nhập hoặc xuất số liệu.
+ Người lập trình muốn dùng dịch vụ của hệ điều hành.
+ Cho một chương trình chạy từng lệnh.
+ Làm điểm dừng của một chương trình.
+ Báo tràn số liệu trong tính toán số học.
+ Trang bộ nhớ thực sự không có trong bộ nhớ.
+ Báo vi phạm vùng cấm của bộ nhớ.
+ Báo dùng một lệnh không có trong tập lệnh.
+ Báo phần cứng máy tính bị hư.
+ Báo điện bị cắt.

Dù rằng ngắt quãng không xảy ra thường xuyên nhưng bộ xử lý phải được thiết kế sao cho có thể lưu giữ trạng thái của nó trước khi nhảy đi phục vụ ngắt quãng. Sau khi thực hiện xong chương trình phục vụ ngắt, bộ xử lý phải khôi phục trạng thái của nó để có thể tiếp tục công việc. Để đơn giản việc thiết kế, một vài bộ xử lý chỉ chấp nhận ngắt sau khi thực hiện xong lệnh đang chạy. Khi một ngắt xảy ra, bộ xử lý thi hành các bước sau đây:

1. Thực hiện xong lệnh đang làm.

2. Lưu trữ trạng thái hiện tại.

3. Nhảy đến chương trình phục vụ ngắt

4. Khi chương trình phục vụ chấm dứt, bộ xử lý khôi phục lại trạng thái cũ của nó và tiếp tục thực hiện chương trình mà nó đang thực hiện khi bị ngắt.

5. Kỹ thuật ống dẫn (PIPELINE)

Mục đích:

- Giới thiệu một số kỹ thuật xử lý thông tin: ống dẫn.

5.1 Ống dẫn

Đây là một kỹ thuật làm cho các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc.

Ví dụ: Chúng ta có những lệnh đều đặn, mỗi lệnh được thực hiện trong cùng một khoản thời gian. Giả sử, mỗi lệnh được thực hiện trong 5 giai đoạn và mỗi giai đoạn được thực hiện trong 1 chu kỳ xung nhịp. Các giai đoạn thực hiện một lệnh là: lấy lệnh (IF: Instruction Fetch), giải mã (ID: Instruction Decode), thi hành (EX: Execute), thâm nhập bộ nhớ (MEM: Memory Access), lưu trữ kết quả (RS: Result Storing).

Bảng 3-1 cho thấy chỉ trong một chu kỳ xung nhịp, bộ xử lý có thể thực hiện một lệnh (bình thường lệnh này được thực hiện trong 5 chu kỳ).

Bảng 3-1.Các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc

Chuỗi lệnh	Chu kỳ xung nhịp								
	1	2	3	4	5	6	7	8	9
Lệnh thứ i	IF	ID	EX	MEM	RS				
Lệnh thứ i+1		IF	ID	EX	MEM	RS			
Lệnh thứ i+2			IF	ID	EX	MEM	RS		
Lệnh thứ i+3				IF	ID	EX	MEM	RS	
Lệnh thứ i+4					IF	ID	EX	MEM	RS

So sánh với kiểu xử lý tuần tự thông thường, 5 lệnh được thực hiện trong 25 chu kỳ xung nhịp, thì xử lý lệnh theo kỹ thuật ống dẫn thực hiện 5 lệnh chỉ trong 9 chu kỳ xung nhịp.

Như vậy kỹ thuật ống dẫn làm tăng tốc độ thực hiện các lệnh. Tuy nhiên kỹ thuật ống dẫn có một số ràng buộc:

- Cần phải có một mạch điện để thi hành mỗi giai đoạn của lệnh vì tất cả các giai đoạn của lệnh được thi hành cùng lúc. Trong một bộ xử lý không dùng kỹ thuật ống dẫn, ta có thể dùng bộ làm toán ALU để cập nhật thanh ghi PC, cập nhật địa chỉ của toán hạng bộ nhớ, địa chỉ ô nhớ mà chương trình cần nhảy tới, làm các phép tính trên các toán hạng vì các phép tính này có thể xảy ra ở nhiều giai đoạn khác nhau.

- Phải có nhiều thanh ghi khác nhau dùng cho các tác vụ đọc và viết. Trên

hình 3-4, tại một chu kỳ xung nhịp, ta thấy cùng một lúc có 2 tác vụ đọc (ID, MEM) và 1 tác vụ viết (RS).

- Trong một máy có kỹ thuật ống dẫn, có khi kết quả của một tác vụ trước đó, là toán hạng nguồn của một tác vụ khác. Như vậy sẽ có thêm những khó khăn mà ta sẽ đề cập ở mục tới.

- Cần phải giải mã các lệnh một cách đơn giản để có thể giải mã và đọc các toán hạng trong một chu kỳ duy nhất của xung nhịp.

- Cần phải có các bộ làm tính ALU hữu hiệu để có thể thi hành lệnh số học dài nhất, có số giữ, trong một khoảng thời gian ít hơn một chu kỳ của xung nhịp.

- Cần phải có nhiều thanh ghi lệnh để lưu giữ lệnh mà chúng ta phải xem xét cho mỗi giai đoạn thi hành lệnh.

- Cuối cùng phải có nhiều thanh ghi bộ đếm chương trình PC để có thể tái tục các lệnh trong trường hợp có ngắt quãng.

5.2 Khó khăn trong kỹ thuật ống dẫn

Khi thi hành lệnh trong một máy tính dùng kỹ thuật ống dẫn, có nhiều trường hợp làm cho việc thực hiện kỹ thuật ống dẫn không thực hiện được như là: thiếu các mạch chức năng, một lệnh dùng kết quả của lệnh trước, một lệnh nhảy.

Ta có thể phân biệt 3 loại khó khăn: *khó khăn do cấu trúc, khó khăn do số liệu* và *khó khăn do điều khiển.*

a. Khó khăn do cấu trúc:

Đây là khó khăn do thiếu bộ phận chức năng, ví dụ trong một máy tính dùng kỹ thuật ống dẫn phải có nhiều ALU, nhiều PC, nhiều thanh ghi lệnh IR ... Các khó khăn này được giải quyết bằng cách thêm các bộ phận chức năng cần thiết và hữu hiệu.

b. Khó khăn do số liệu:

Lấy ví dụ trường hợp các lệnh liên tiếp sau:

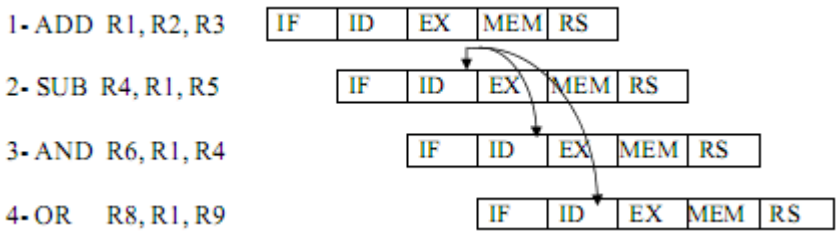
Lệnh 1: ***ADD R1, R2, R3***

Lệnh 2: ***SUB R4, R1, R5***

Lệnh 3: ***AND R6, R1, R7***

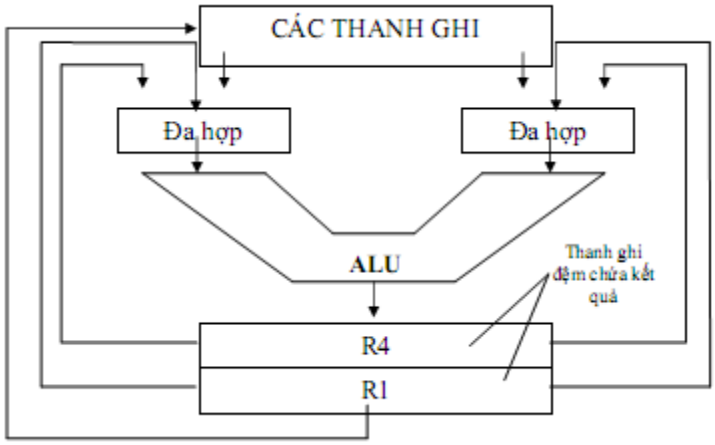
Lệnh 4: ***OR R8, R1, R9***

Hình 3-4 cho thấy R1, kết quả của lệnh 1 chỉ có thể được dùng cho lệnh 2 sau giai đoạn MEM của lệnh 1, nhưng R1 được dùng cho lệnh 2 vào giai đoạn EX của lệnh 1. Chúng ta cũng thấy R1 được dùng cho các lệnh 3 và 4.



Hình 3-4. Chuỗi lệnh minh hoạ khó khăn do số liệu.

Để khắc phục khó khăn này, một bộ phận phần cứng được dùng để đưa kết quả từ ngã ra ALU trực tiếp vô một trong các thanh ghi ngã vào như trong hình 3-5



Hình 3-5. ALU với bộ phận phần cứng đưa kết quả tính toán trở lại ngã vào

Khi bộ phận phần cứng nêu trên phát hiện có dùng kết quả của ALU làm toán hạng cho liệt kê, nó tác động vào mạch đa hợp để đưa ngã ra của ALU vào ngã vào của ALU hoặc vào ngã vào của một đơn vị chức năng khác nếu cần.

c. Khó khăn do điều khiển:

Các lệnh làm thay đổi tính thi hành các lệnh một cách tuần tự (nghĩa là PC tăng đều đặn sau mỗi lệnh), gây khó khăn về điều khiển. Các lệnh này là lệnh nhảy đến một địa chỉ tuyệt đối chứa trong một thanh ghi, hay lệnh nhảy đến một địa chỉ xác định một cách tương đối so với địa chỉ hiện tại của bộ đếm chương trình PC. Các lệnh nhảy trên có thể có hoặc không điều kiện.

Trong trường hợp đơn giản nhất, tác vụ nhảy không thể biết trước giai đoạn giải mã (xem hình 3-4). Như vậy, nếu lệnh nhảy bắt đầu ở chu kỳ C thì lệnh mà chương trình ở cuối giai đoạn giải mã ID. Trong lệnh nhảy tương đối, ta phải cộng độ dời chứa trong thanh ghi lệnh IR vào thanh ghi PC. Việc tính địa chỉ này chỉ được thực hiện vào giai đoạn ID với điều kiện phải có một mạch công việc riêng biệt.

Vậy trong trường hợp lệnh nhảy không điều kiện, lệnh mà chương trình nhảy đến bắt đầu thực hiện ở chu kỳ C+2 nếu lệnh nhảy bắt đầu ở chu kỳ C.

Cho các lệnh nhảy có điều kiện thì phải tính toán điều kiện. Thông thường các kiến trúc RISC đặt kết quả việc so sánh vào trong thanh ghi trạng thái, hoặc vào trong thanh ghi tổng quát. Trong cả 2 trường hợp, đọc điều kiện tương đương với đọc thanh ghi. Đọc thanh ghi có thể được thực hiện trong phân nửa chu kỳ cuối giai đoạn ID.

Một trường hợp khó hơn có thể xảy ra trong những lệnh nhảy có điều kiện. Đó là điều kiện được có khi so sánh 2 thanh ghi và chỉ thực hiện lệnh nhảy khi kết quả so sánh là đúng. Việc tính toán trên các đại lượng logic không thể thực hiện được trong phân nửa chu kỳ và như thế phải kéo dài thời gian thực

hiện lệnh nhảy có điều kiện. Người ta thường tránh các trường hợp này để không làm giảm mức hữu hiệu của máy tính.

Vậy trường hợp đơn giản, người ta có thể được địa chỉ cần nhảy đến và điều kiện nhảy cuối giai đoạn ID. Vậy có chậm đi một chu kỳ mà người ta có thể giải quyết bằng nhiều cách.

Cách thứ nhất là đóng băng kỹ thuật ống dẫn trong một chu kỳ, nghĩa là ngưng thi hành lệnh thứ i+1 đang làm nếu lệnh thư i là lệnh nhảy. Ta mất trắng một chu kỳ cho mỗi lệnh nhảy.

Cách thứ hai là thi hành lệnh sau lệnh nhảy nhưng lưu ý rằng hiệu quả của một lệnh nhảy bị chậm mất một lệnh. Vậy lệnh theo sau lệnh nhảy được thực hiện trước khi lệnh mà chương trình phải nhảy tới được thực hiện. Chương trình dịch hay người lập trình có nhiệm vụ xen vào một lệnh hữu ích sau lệnh nhảy.

Trong trường hợp nhảy có điều kiện, việc nhảy có thể được thực hiện hay không thực hiện. Lệnh hữu ích đặt sau lệnh nhảy không làm sai lệch chương trình dù điều kiện nhảy đúng hay sai.

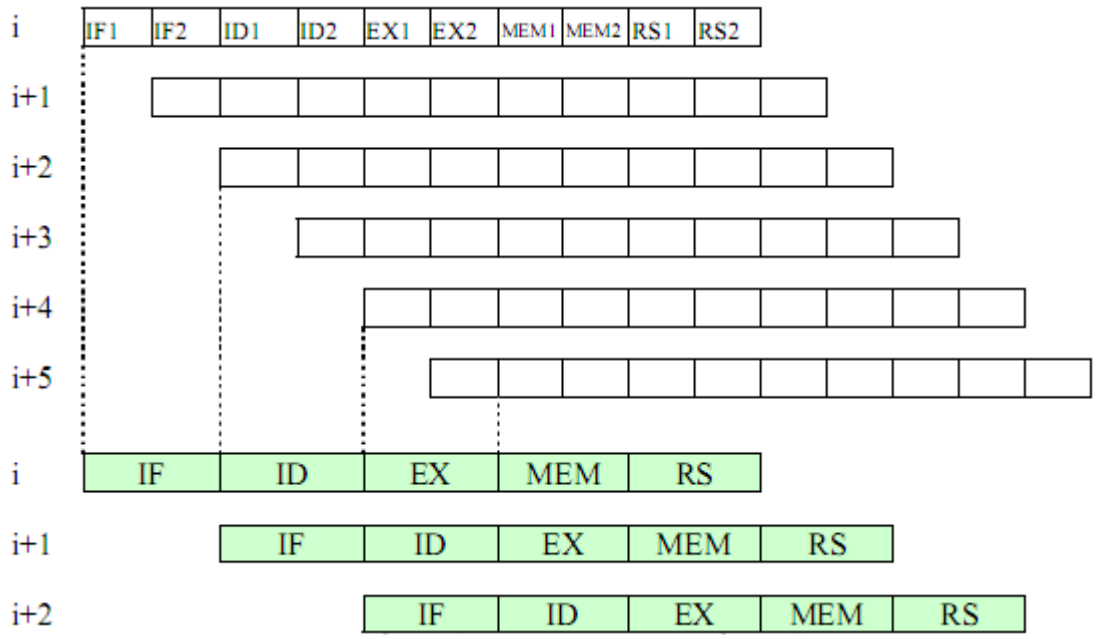
Bộ xử lý RISC SPARC có những lệnh nhảy với huỷ bỏ. Các lệnh này cho phép thi hành lệnh sau lệnh nhảy nếu điều kiện nhảy đúng và huỷ bỏ thực hiện lệnh đó nếu điều kiện nhảy sai.

5. Siêu ống dẫn

Mục đích:

- Giới thiệu một số kỹ thuật xử lý thông tin: siêu ống dẫn

Máy tính có kỹ thuật siêu ống dẫn bậc n bằng cách chia các giai đoạn của kỹ thuật ống dẫn đơn giản, mỗi giai đoạn được thực hiện trong khoản thời gian Tc, thành n giai đoạn con thực hiện trong khoản thời gian Tc/n. Độ hữu hiệu của kỹ thuật này tương đương với việc thi hành n lệnh trong mỗi chu kỳ Tc. Hình 3-6 trình bày thí dụ về siêu ống dẫn bậc 2, có so sánh với siêu ống dẫn đơn giản. Ta thấy trong một chu kỳ Tc, máy dùng kỹ thuật siêu ống dẫn làm 2 lệnh thay vì làm1 lệnh trong máy dùng kỹ thuật ống dẫn bình thường. Trong máy tính siêu ống dẫn, tốc độ thực hiện lệnh tương đương với việc thực hiện một lệnh trong khoảng thời gian Tc/n. Các bất lợi của siêu ống dẫn là thời gian thực hiện một giai đoạn con ngắn Tc/n và việc trì hoãn trong thi hành lệnh nhảy lớn. Trong ví dụ ở hình 3-6, nếu lệnh thứ i là một lệnh nhảy tương đối thì lệnh này được giải mã trong giai đoạn ID, địa chỉ nhảy đến được tính vào giai đoạn EX, lệnh phải được nhảy tới là lệnh thứ i+4, vậy có trì trệ 3 lệnh thay vì 1 lệnh trong kỹ thuật ống dẫn bình thường.



Hình 3-6. Siêu ống dẫn bậc 2 so với siêu ống dẫn đơn giản. Trong khoảng thời gian Tc, máy có siêu ống dẫn làm 2 lệnh thay vì 1 lệnh như ống có kỹ thuật ống dẫn đơn giản

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 3

*****

1. Các thành phần và nhiệm vụ của đường đi dữ liệu?
2. Thế nào là ngắt? Các giai đoạn thực hiện ngắt của CPU.
3. Vẽ hình để mô tả kỹ thuật ống dẫn. Kỹ thuật ống dẫn làm tăng tốc độ CPU lên bao nhiêu lần (theo lý thuyết)? Tại sao trên thực tế sự gia tăng này lại ít hơn?
4. Các điều kiện mà một CPU cần phải có để tối ưu hoá kỹ thuật ống dẫn. Giải thích từng điều kiện.
5. Các khó khăn trong kỹ thuật ống dẫn và cách giải quyết khó khăn này.

Chương 4: BỘ NHỚ

Mã chương: MH12 – 04.

Mục đích: Chương này giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache: nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên lý vận hành của bộ nhớ ảo.

Yêu cầu: Sinh viên phải hiểu được các cấp bộ nhớ và cách thức vận hành của các loại bộ nhớ được giới thiệu để có thể đánh giá được hiệu năng hoạt động của các loại bộ nhớ.

1. Các loại bộ nhớ

Mục đích:

- *Giới thiệu về các loại bộ nhớ máy tính, bộ nhớ trong.*

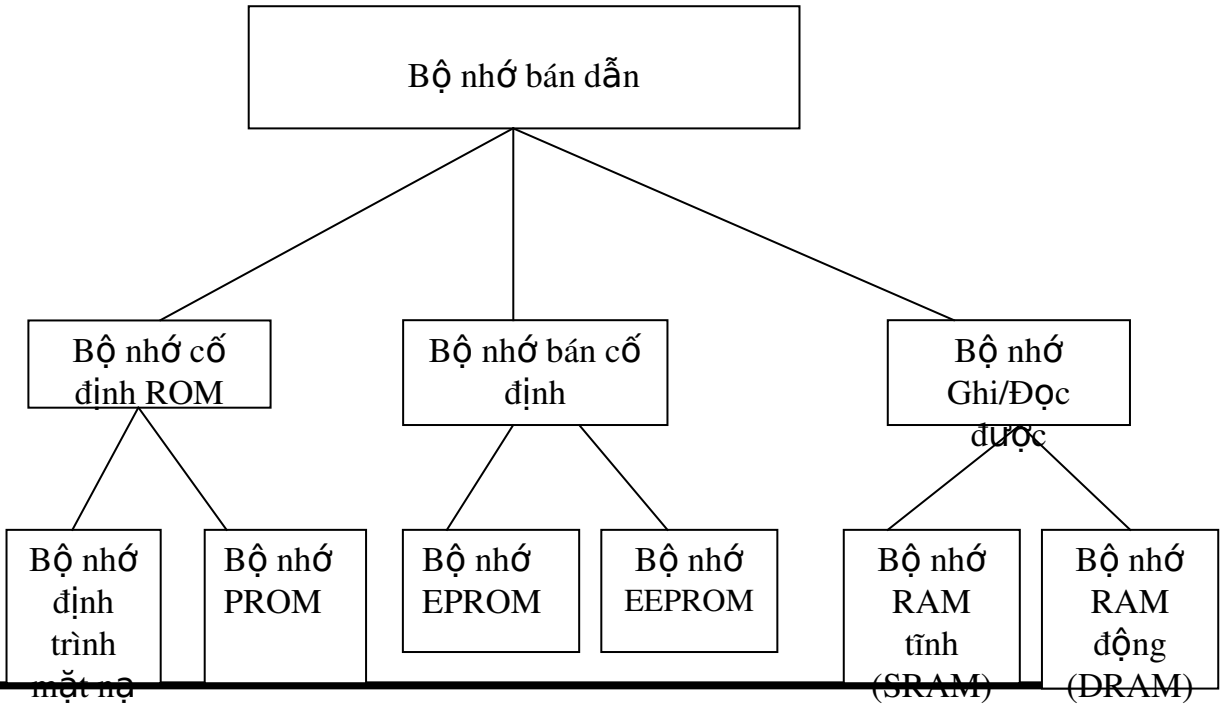
Bộ nhớ chứa chương trình, nghĩa là chứa lệnh và số liệu. Người ta phân biệt các loại bộ nhớ: Bộ nhớ trong (RAM-Bộ nhớ vào ra ngẫu nhiên), được chế tạo bằng chất bán dẫn; bộ nhớ chỉ đọc (ROM) cũng là loại bộ nhớ chỉ đọc và bộ nhớ ngoài bao gồm: đĩa cứng, đĩa mềm, băng từ, trống từ, các loại đĩa quang, các loại thẻ nhớ,...

Bộ nhớ RAM có đặc tính là các ô nhớ có thể được đọc hoặc viết vào trong khoảng thời gian bằng nhau cho dù chúng ở bất kỳ vị trí nào trong bộ nhớ. Mỗi ô nhớ có một địa chỉ, thông thường, mỗi ô nhớ là một byte (8 bit), nhưng hệ thống có thể đọc ra hay viết vào nhiều byte (2,4, hay 8 byte). Bộ nhớ trong (RAM) được đặc trưng bằng dung lượng và tổ chức của nó (số ô nhớ và số bit cho mỗi ô nhớ), thời gian thâm nhập (thời gian từ lúc đưa ra địa chỉ ô nhớ đến lúc đọc được nội dung ô nhớ đó) và chu kỳ bộ nhớ (thời gian giữa hai lần liên tiếp thâm nhập bộ nhớ).

1.1 Bộ nhớ trong

a. Phân loại

Có thể phân loại các vi mạch nhớ bán dẫn thành bộ nhớ cố định, bán cố định và bộ nhớ ghi/đọc như sơ đồ (hình 4-1)



Hình 4-1.Phân loại các bộ nhớ bán dẫn

Bộ nhớ có nội dung ghi sẵn một lần khi chế tạo gọi là bộ nhớ cố định và được ký hiệu là ROM (Read Only Memory). Sau khi đã viết (bằng mặt nạ) từ nhà máy thì ROM loại này không viết lại được nữa. PROM một dạng khác, các bit có thể ghi bằng thiết bị ghi của người sử dụng trong một lần đầu (Programmable ROM). Bộ nhớ có thể đọc/ghi nhiều lần gọi là RAM (Random Access Memory) gồm có 2 loại: RAM tĩnh là SRAM (Static RAM) thường được xây dựng trên các mặt lật điện tử, Ram động là DRAM (Dinamic RAM) được xây dựng trên cơ sở nhớ các điện tích ở tụ điện. Bộ nhớ DRAM phải được phục hồi nội dung thường xuyên.

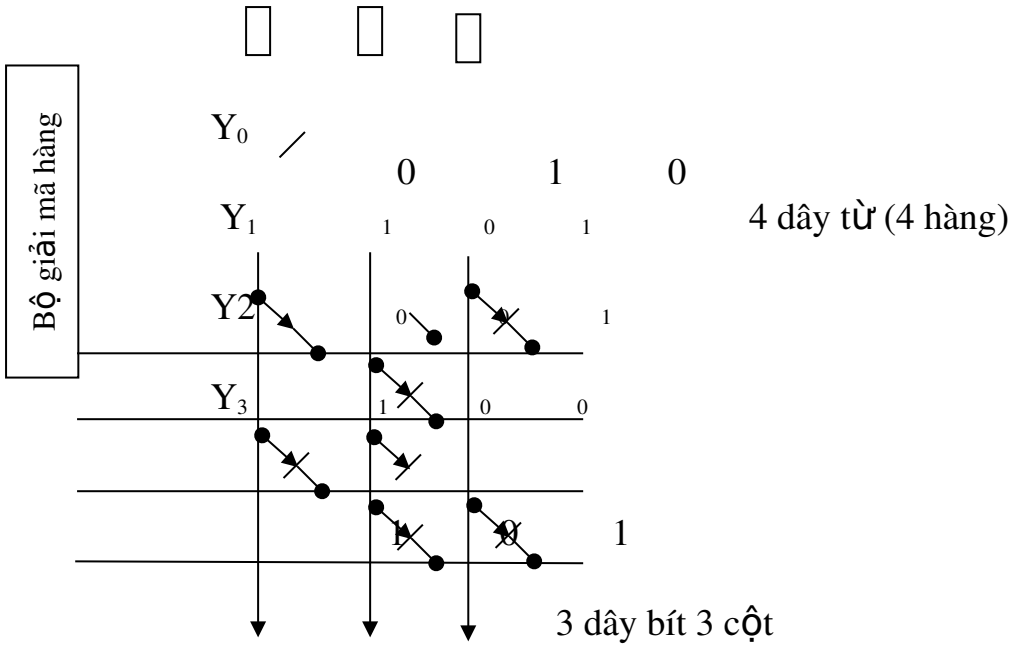
Giữa ROM và RAM có một lớp các bộ nhớ được gọi là bộ nhớ bán cố định. Trong đó có bộ nhớ EPROM (Erasable Programmable ROM) có thể ghi được bằng các xung điện nhưng cũng xóa được bằng tia cực tím, EEPROM (Elictric Erasable Programmable ROM) lại có thể xóa được bằng dòng điện. Các bộ nhớ DRAM thường thỏa mãn những yêu cầu khi cần bộ nhớ có dung lượng lớn; trong khi cần có tốc độ truy xuất nhanh thì lại phải dùng các bộ SRAM. Nhưng cả 2 loại này đều bị mất thông tin khi nguồn điện nuôi bị mất đi. Do vậy các chương trình dùng cho việc khởi động máy vi tính như BIOS phải nạp trên các bộ nhớ ROM gọi là ROM BIOS.

b. Nguyên lý hoạt động của các linh kiện nhớ bán dẫn

*** ROM**

ROM (Read Only Memory) là các chíp nhớ mà khi đến tay người dùng chỉ có thể đọc được. Đó là loại chíp nhớ có nội dung được viết sẵn một lần khi chế tạo và được giữ mãi cố định (non- volatile). ROM lập trình kiểu mặt nạ được chế tạo trên một phiến silicon nhằm tạo ra những tiếp giáp bán dẫn điện theo một chiều như diode tại các điểm vắt chéo nhau trên một ma trận các dây dẫn hàng (từ số liệu) và cột (bit số liệu) như ví dụ trong (hình 4.2)

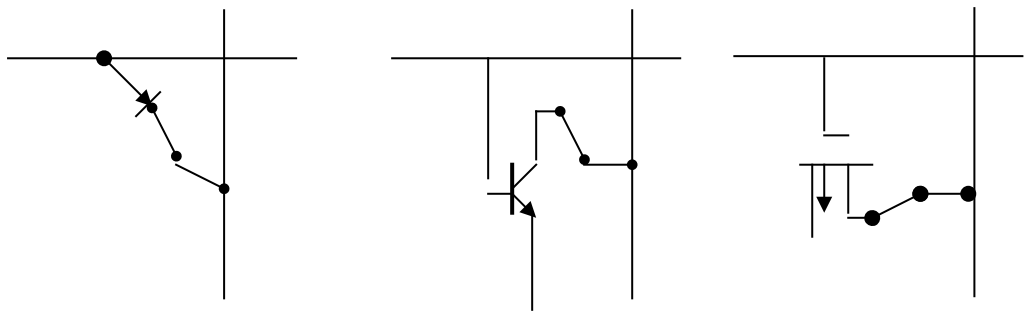
Tại đó các điểm vắt chéo chứa diode sẽ mang lại thông tin là 0, các điểm còn lại mang thông tin là 1. Khi lối ra bộ giải mã địa chỉ ở mức thấp chọn một hàng thì thế lối ra của các dây bit phản ánh các giá trị được lưu trữ trong chip nhớ tại hàng đó. Trên hình 4.2 là trường hợp chip lưu trữ 4 từ dữ liệu, mỗi từ 3 bit: 010,101,001 và 100. Khi bộ giải mã chọn địa chỉ hàng Y_1 như hình trên thì lối ra của chip nhớ sẽ xuất hiện từ dữ liệu là 101.



Hình 4-2. Bộ nhớ ROM diode

*** PROM**

PROM (Progammable ROM) là các ROM khả trình cho phép người dùng có thể ghi thông tin được một lần. Đó là loại ROM mà khi sản xuất, tất cả các điểm vắt chéo đều được đặt các diode hoặc transistor nối tiếp với một cầu chì, khi cần thông tin với mức logic "1" ở điểm vắt chéo nào thì chỉ việc cho dòng điện đủ lớn đi qua và làm cháy đứt cầu chì tương ứng ở điểm đó và điểm vắt chéo đó coi như không có transistor hoặc diode. Rõ ràng loại PROM này chỉ ghi thông tin được một lần mà không xóa được.



Hình 4-3.Cầu chì trong các điểm vắt chéo với diode, transistor lưỡng cực và transistor trường

*** EPROM**

EPROM (Erasable PROM) là các chip nhớ PROM có thể xoá được, nó cho phép ghi và xoá thông tin nhiều lần và được chế tạo theo nguyên tắc khác. Trong các chip này, mỗi bit nhớ là một transistor MOS có cửa nổi được chế tạo theo công nghệ FAMOST (Floating gate avalanche injection MOS transistor). Số liệu có thể được viết vào bằng các xung điện có độ dài cỡ 50ms và độ lớn +20V khi đặt vào giữa cực cửa và máng của transistor.Do cửa nổi được cách điện cao với xung quanh nên sau khi hết xung điện, các điện tử giữ vai trò là những phần tử mang thông tin không còn đủ năng lượng để có thể vượt ra ngoài lớp cách điện đó nữa.

Vì vậy thông tin được giữ cố định cả sau khi ngừng cấp điện cho chip trong một thời gian rất dài (ít nhất là 10 năm). Để xoá thông tin tức là làm mất các điện tích trong vùng cửa nổi, phải chiếu ánh sáng tử ngoại đủ mạnh và chip nhớ. Những điện tử ở đây lúc này hấp thụ năng lượng nhảy lên mức năng lượng cao hơn, chúng sẽ rời cửa nổi như cách thâm nhập vào đó. Vì vậy trong chip EPROM có một cửa sổ làm bằng thuỷ tinh thạch anh chỉ để cho ánh sáng tử ngoại đi qua khi cần xoá số liệu trong bộ nhớ.

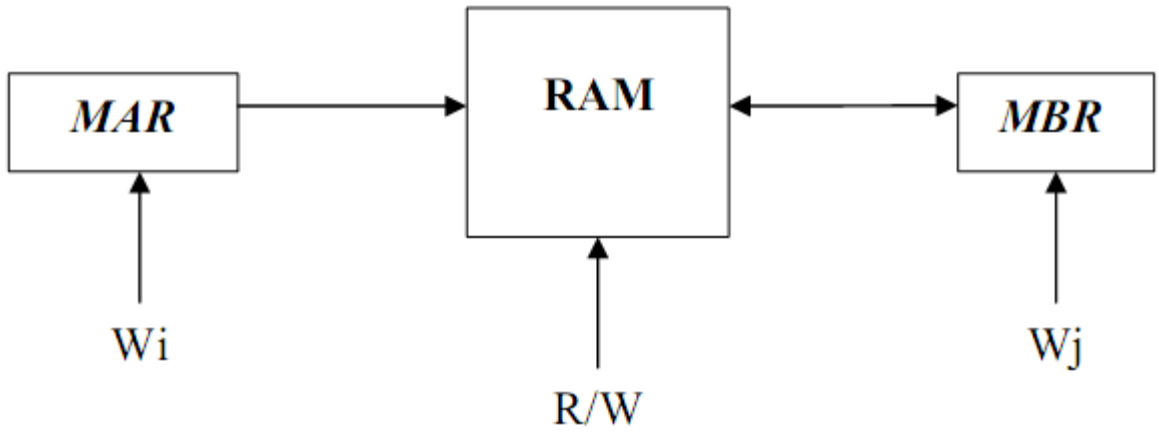
*** EEPROM**

EEPROM (Electrically EPROM) là loại EPROM xoá được bằng phương pháp điện. Việc nạp các điện tử cho cửa nổi được thực hiện như cách ở EPROM. Để xoá EEPROM

,có một lớp kênh màng mỏng ôxít giữa vùng cửa nổi trải xuống dưới đế và cực máng giữ vai trò quan trọng. Các lớp cách điện không thể là lý tưởng được, các điện tích mang có thể thấm qua lớp phân cách với một sác xuất thấp. Sác xuất này tăng lên khi bề dày của lớp giảm đi và hiệu điện thế giữa 2 điện cực ở 2 mặt lớp cách điện tăng lên. Muốn phóng các điện tích trong vùng cửa nổi, một điện thế (-20V) được đặt vào cực cửa điều khiển và cực máng. Lúc này các điện tử âm trong cửa nổi được chảy về cực máng qua kênh màng mỏng ôxit và số liệu lưu giữ được xoá đi.

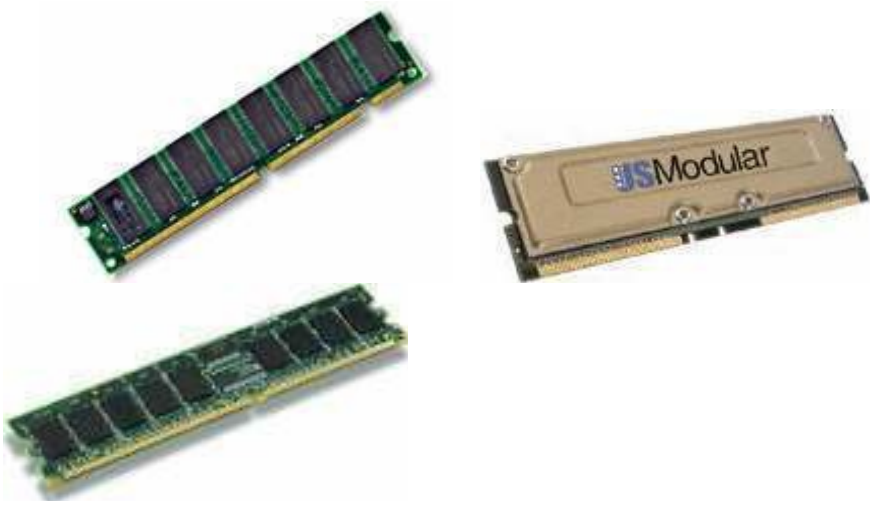
*** RAM**

RAM (Random Acess Memory) là loại bộ nhớ có thể ghi/đọc được. Đây là loại chíp nhớ mà thông tin lưu trữ thông tin trong nó sẽ được sẽ bị mất đi khi bị cắt nguồn điện nuôi. RAM có 2 loại: RAM động, được viết tắt là DRAM (dinamic RAM) và Ram tĩnh được viết tắt là SRAM (Static RAM). Cấu trúc đơn giản



Hình 4-4. Vận hành của bộ nhớ RAM
(Wi, Wj, R/W là các tín hiệu điều khiển)

Slot để cắm RAM



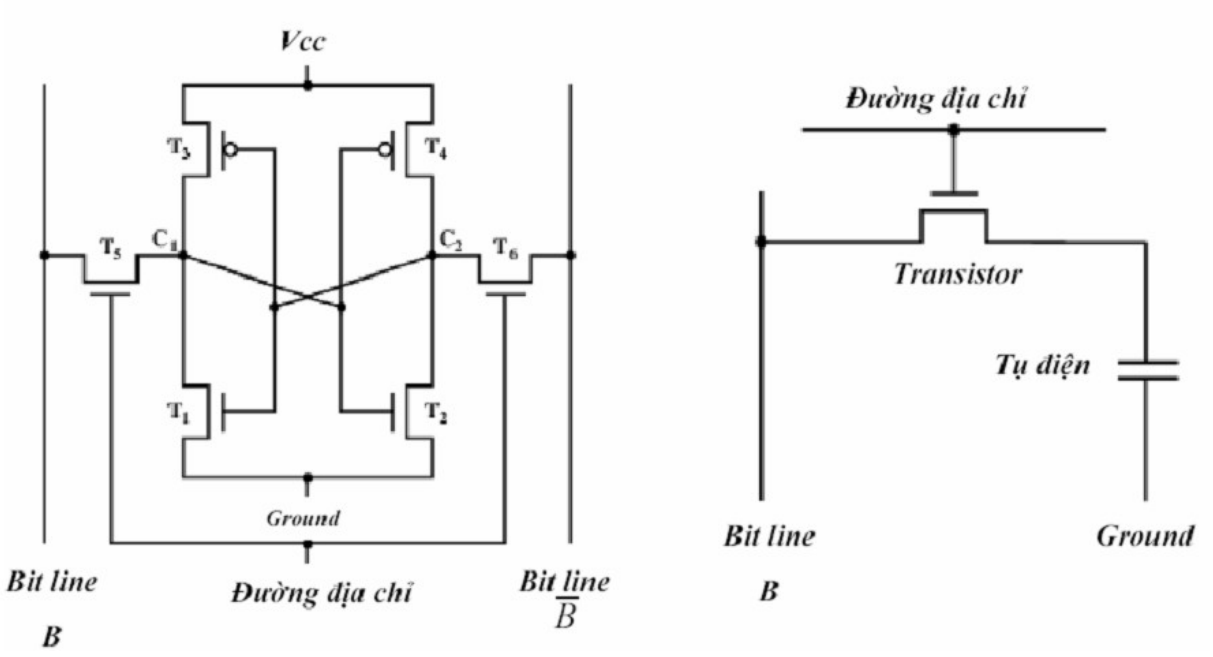
Hình 4-5.Hình dáng bên ngoài một số loại RAM

Tuỳ theo công nghệ chế tạo, người ta phân

RAM tĩnh được chế tạo theo công nghệ ECL (CMOS và BiCMOS). Mỗi bit nhớ gồm có các cổng logic với độ 6 transistor MOS, việc nhớ một dữ liệu là tồn tại nếu bộ nhớ được cung cấp điện. SRAM là bộ nhớ nhanh, việc đọc không làm huỷ nội dung của ô nhớ và thời gian thâm nhập bằng chu kỳ bộ nhớ.

RAM động dùng kỹ thuật MOS. Mỗi bit nhớ gồm có một transistor và một tụ điện. Cũng như SRAM, việc nhớ một dữ liệu là tồn tại nếu bộ nhớ được

cung cấp điện. Việc ghi nhớ dựa vào việc duy trì điện tích nạp vào tụ điện và như vậy việc đọc một bit nhớ làm nội dung bit này bị huỷ. Vậy sau mỗi lần đọc một ô nhớ, bộ phận điều khiển bộ nhớ phải viết lại ô nhớ đó nội dung vừa đọc và do đó chu kỳ bộ nhớ động ít nhất là gấp đôi thời gian thâm nhập ô nhớ. Việc lưu giữ thông tin trong bit nhớ chỉ là tạm thời vì tụ điện sẽ phóng hết điện tích đã nạp vào và như vậy phải làm tươi bộ nhớ sau mỗi 2μs. Làm tươi bộ nhớ là đọc ô nhớ và viết lại nội dung đó vào lại ô nhớ. Việc làm tươi được thực hiện với tất cả các ô nhớ trong bộ nhớ. Việc làm tươi bộ nhớ được thực hiện tự động bởi một vi mạch bộ nhớ.



Hình 4-6.SRAM và DRAM

SDRAM (Synchronous DRAM – DRAM đồng bộ), một dạng DRAM đồng bộ bus bộ nhớ. Tốc độ SDRAM đạt từ 66-133MHz (thời gian thâm nhập bộ nhớ từ 75ns-150ns).

DDR SDRAM (Double Data Rate SDRAM) là cải tiến của bộ nhớ SDRAM với tốc độ truyền tải gấp đôi SDRAM nhờ vào việc truyền tải hai lần trong một chu kỳ bộ nhớ. Tốc độ DDR SDRAM đạt từ 200-400MHz

RDRAM (Rambus RAM) là một loại DRAM được thiết kế với kỹ thuật hoàn toàn mới so với kỹ thuật SDRAM. RDRAM hoạt động đồng bộ theo một hệ thống lặp và truyền dữ liệu theo một hướng. Một kênh bộ nhớ RDRAM có thể hỗ trợ đến 32 chip DRAM. Mỗi chip được ghép nối tuần tự trên một module gọi là RIMM (Rambus Inline Memory Module) nhưng việc truyền dữ liệu giữa các mạch điều khiển và từng chíp riêng biệt chứ không truyền giữa các chip với nhau. Bus bộ nhớ RDRAM là đường dẫn liên tục đi qua các chip và module trên bus, mỗi module có các chân vào và ra trên các đầu đối diện. Do đó, nếu các khe

cắm không chứa RIMM sẽ phải gắn một module liên tục để đảm bảo đường truyền được nối liền.Tốc độ RDRAM đạt từ 400-800MHz

Bộ nhớ chỉ đọc ROM cũng được chế tạo bằng công nghệ bán dẫn. Chương trình trong ROM được viết vào lúc chế tạo nó. Thông thường, ROM chứa chương trình khởi động máy tính, chương trình điều khiển trong các thiết bị điều khiển tự động,...

Bảng 4-1. Các kiểu bộ nhớ bán dẫn

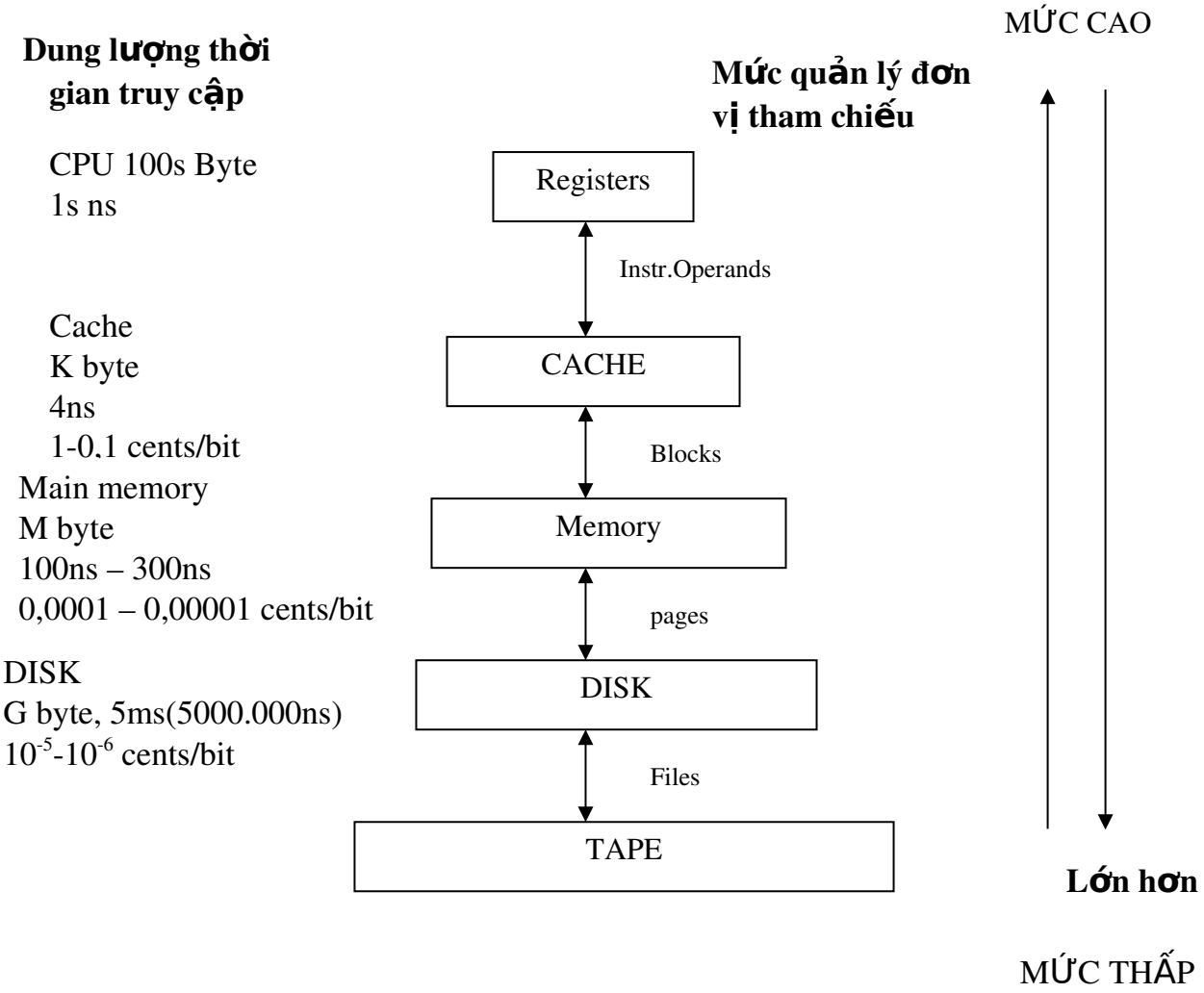
<table>
<tr><th>Kiểu bộ nhớ</th><th>Loại</th><th>Cơ chế xoá</th><th>Cơ chế ghi</th><th>Tính bay hơi</th></tr>
<tr><td>RAM</td><td>đọc/ghi</td><td>bằng điện, mức byte</td><td>bằng điện</td><td>Có</td></tr>
<tr><td>ROM</td><td rowspan="2">chỉ đọc</td><td rowspan="2">Không thể xoá</td><td>Mặt nạ</td><td rowspan="4">Không</td></tr>
<tr><td>Programmable ROM (PROM)</td><td rowspan="3">bằng điện</td></tr>
<tr><td>Erasable PROM</td><td rowspan="3">hầu hết chỉ đọc</td><td>Tia cực tím, mức chip</td></tr>
<tr><td>Electrically Erasable PROM (EEPROM)</td><td>bằng điện, mức byte</td></tr>
<tr><td>Flash Memory</td><td>bằng điện, mức khối</td></tr>
</table>

2. Các cấp bộ nhớ

Mục đích:

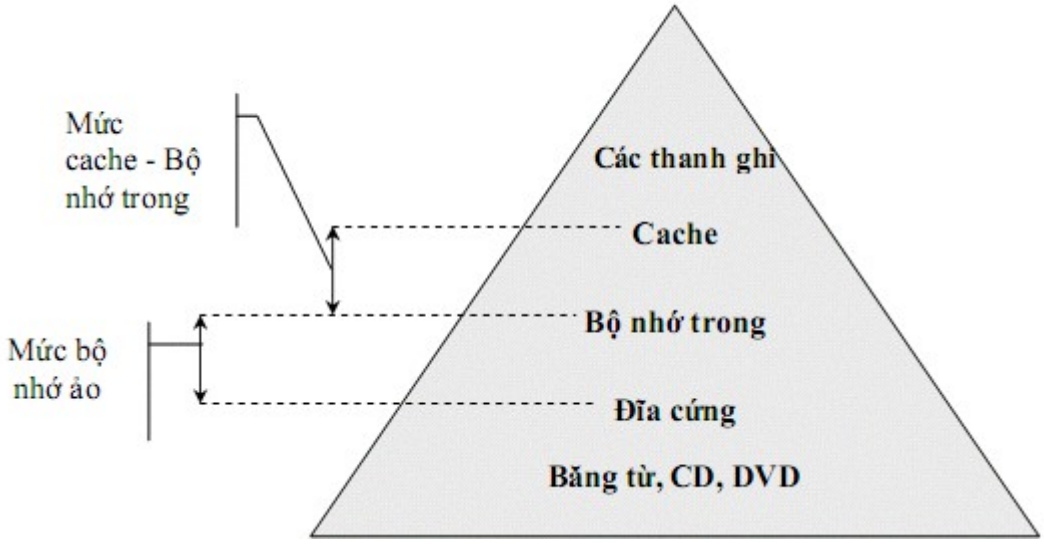
- *Giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính.*

Các đặc tính như lượng thông tin lưu trữ, thời gian thâm nhập bộ nhớ, chu kỳ bộ nhớ, giá tiền mỗi bit nhớ khiến ta phải phân biệt các cấp bộ nhớ: các bộ nhớ nhanh với dung lượng ít đến các bộ nhớ chậm với dung lượng lớn (hình 4-6)



Hình 4-6. Các cấp bộ nhớ

Các đặc tính chính của các cấp bộ nhớ dẫn đến hai mức chính là: *mức cache - bộ nhớ trong* và *mức bộ nhớ ảo* (bao gồm bộ nhớ trong và không gian cấp phát trên đĩa cứng) (hình IV.4). Cách tổ chức này trong suốt đối với người sử dụng. Người sử dụng chỉ thấy duy nhất một không gian định vị ô nhớ, độc lập với vị trí thực tế của các lệnh và dữ liệu cần thâm nhập.



Hình 4-7. Hai mức bộ nhớ

Các cấp bộ nhớ giúp ích cho người lập trình muốn có một bộ nhớ thật nhanh với chi phí đầu tư giới hạn. Vì các bộ nhớ nhanh đắt tiền nên các bộ nhớ được tổ chức thành nhiều cấp, cấp có dung lượng ít thì nhanh nhưng đắt tiền hơn cấp có dung lượng cao hơn. Mục tiêu của việc thiết lập các cấp bộ nhớ là người dùng có một hệ thống bộ nhớ rẻ tiền như cấp bộ nhớ thấp nhất và gần nhanh như cấp bộ nhớ cao nhất. Các cấp bộ nhớ thường được lồng vào nhau. Mọi dữ liệu trong một cấp thì được gặp lại trong cấp thấp hơn và có thể tiếp tục gặp lại trong cấp thấp nhất.

Chúng ta có nhận xét rằng, mỗi cấp bộ nhớ có dung lượng lớn hơn cấp trên mình, ánh xạ một phần địa chỉ các ô nhớ của mình vào địa chỉ ô nhớ của cấp trên trực tiếp có tốc độ nhanh hơn, và các cấp bộ nhớ phải có cơ chế quản lý và kiểm tra các địa chỉ ánh xạ.

3. Truy cập dữ liệu trong bộ nhớ

Mục đích: *Trình bày truy cập dữ liệu trong bộ nhớ và thiết bị vào ra.*

Cache là bộ nhớ nhanh, nó chứa lệnh và dữ liệu thường xuyên dùng đến. Việc lựa chọn lệnh và dữ liệu cần đặt vào cache dựa vào các nguyên tắc sau đây:

Một chương trình mất 90% thời gian thi hành lệnh của nó để thi hành 10% số lệnh của chương trình.

Nguyên tắc trên cũng được áp dụng cho việc thâm nhập dữ liệu, nhưng ít hiệu nghiệm hơn việc thâm nhập lệnh. Như vậy có hai nguyên tắc: *nguyên tắc về không gian* và *nguyên tắc về thời gian*

-***Nguyên tắc về thời gian:*** cho biết các ô nhớ được hệ thống xử lý thâm nhập có khả năng sẽ được thâm nhập trong tương lai gần. Thật vậy, các chương trình được cấu tạo với phần chính là phần được thi hành nhiều nhất và các phần

phụ dùng để xử lý các trường hợp ngoại lệ. Còn số liệu luôn có cấu trúc và thông thường chỉ có một phần số liệu được thâm nhập nhiều nhất mà thôi.

-***Nguyên tắc về không gian:*** cho biết, bộ xử lý thâm nhập vào một ô nhớ thì có nhiều khả năng thâm nhập vào ô nhớ có địa chỉ kế tiếp do các lệnh được sắp xếp thành chuỗi có thứ tự.

Tổ chức các cấp bộ nhớ sao cho các lệnh và dữ liệu thường dùng được nằm trong bộ nhớ cache, điều này làm tăng hiệu quả của máy tính một cách đáng kể.

3.1 Truy nhập bộ nhớ và thiết bị vào/ ra

Thời gian một chù kỳ của xung đồng hồ hệ thống được gọi là một trạng thái. Một trạng thái được tính từ sườn âm của một xung đồng hồ đến sườn âm của xung tiếp theo.

Chu kỳ máy hay **chu kỳ bus** là một quá trình cơ bản của bộ vi xử lý hay đơn vị làm chủ bus thực hiện việc truyền tải dữ liệu trên bus. Một chu kỳ máy gồm hai giai đoạn : gửi địa chỉ lên bus và chuyển dữ liệu đến hay đi. Giai đoạn đầu, gọi là thời gian địa chỉ , trong đó địa chỉ đích được vi xử lý gửi đi cùng với tín hiệu xác định loại chu kỳ bus. Giai đoạn hai, gọi là thời gian số liệu, trong đó bộ xử lý kiểm tra xem đã có tín hiệu sẵn sàng từ đơn vị cần trao đổi thông tin chưa để cấp hoặc nhận dữ liệu.

Có 4 loại chu kỳ máy cơ bản: đọc bộ nhớ, viết bộ nhớ, đọc vào/ra, viết vào/ra.

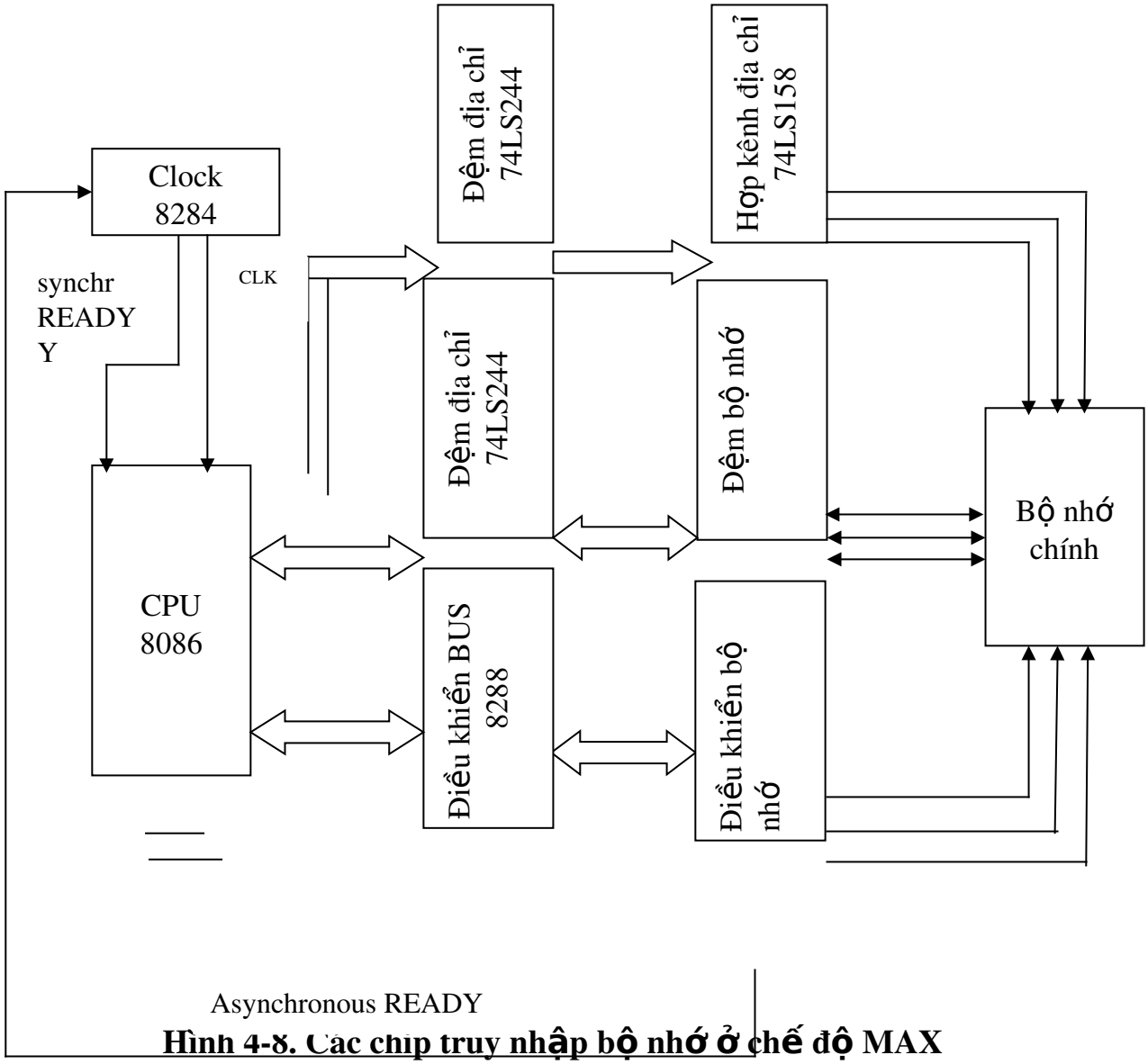
Chu kỳ lệnh là thời gian mà vi xử lý cần để nhận lệnh và thi hành một lệnh. Một chu kỳ lệnh gồm một hay nhiều chu kỳ máy.

Tóm lại, các trạng thái tạo nên một chu kỳ máy và các chu kỳ máy tạo nên một chu kỳ lệnh.

Trạng thái đợi: trong nhiều trường hợp, ví dụ như do tốc độ truy cập của bộ nhớ hay tốc độ xử lý dữ liệu của thiết bị ngoại vi chậm hơn tốc độ của vi xử lý thì phải có cách nhận biết và trì hoãn quá trình trao đổi dữ liệu. Cách giải quyết vấn đề này là một bên thông tin chỉ nhận hay phát tiếp thông tin nếu nhận được một tín hiệu sẵn sàng READY từ bộ nhớ hay ngoại vi rồi mới phát dữ liệu tiếp theo. Nếu bộ nhớ hay ngoại vi chậm, nó sẽ trì hoãn việc phát tín hiệu READY và xử lý không nhận ngay được tín hiệu mà phải trải qua một số nhịp đồng hồ. Mỗi khoảng thời gian ứng với một chu kỳ đồng hồ đợi đó gọi là chu kỳ đợi hay trạng thái đợi. Rõ ràng hệ thống máy tính càng có nhiều trạng thái đời thì hiệu suất xử lý càng chậm.

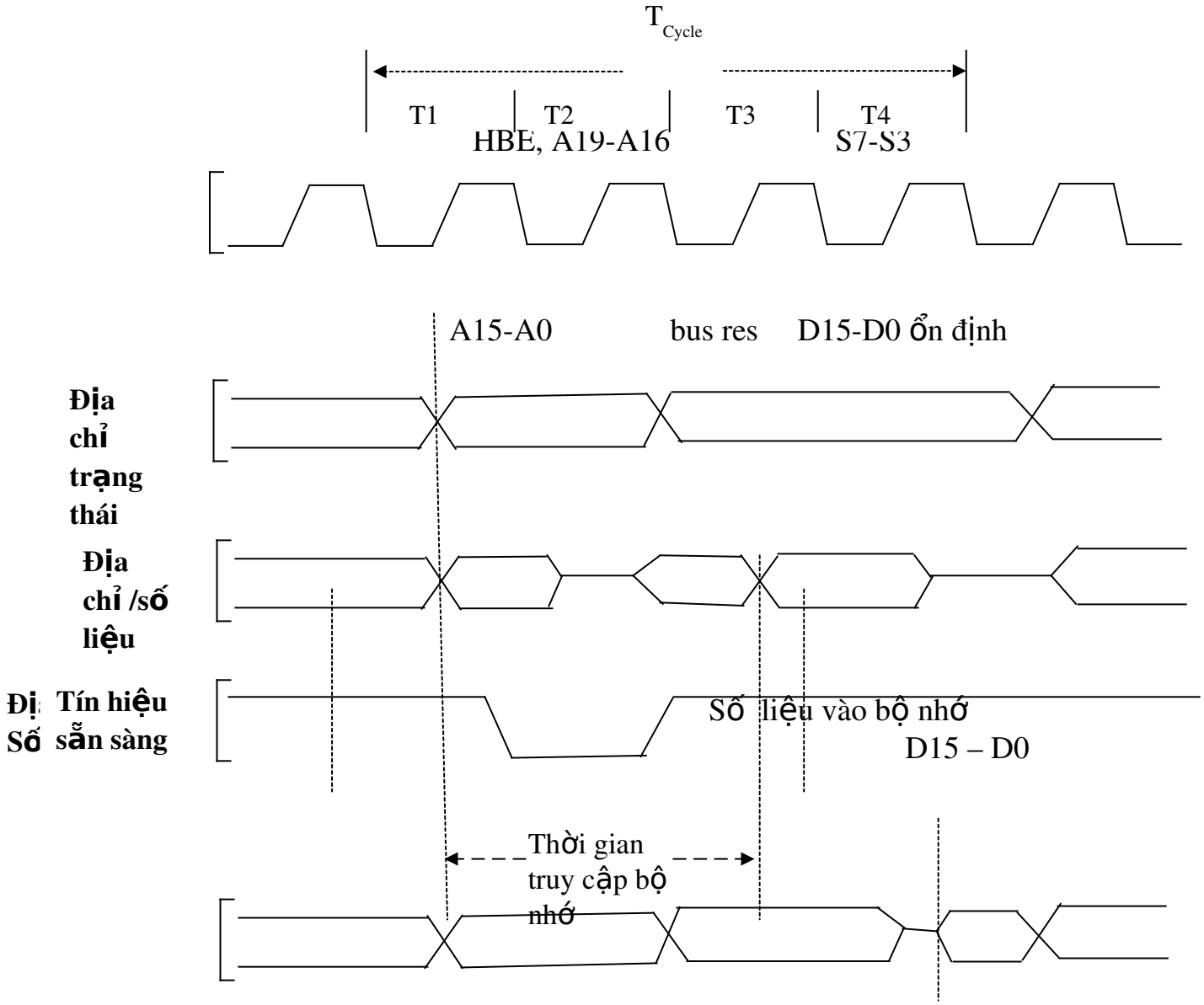
3.2 Truy nhập bộ nhớ chính

Hình 4-8 cho ta sơ đồ kết nối các chip trong quá trình vi xử lý truy nhập ở chế độ MAX. Lúc này 8288 phát ra các tín hiệu điều khiển cho BUS và một vài bộ đệm trong bộ nhớ tạm thời. Bộ điều khiển nhớ điều khiển bộ nhớ để đọc/ viết số liệu tời địa chỉ mong muốn một cách chính xác



Hình 4-8. Các chip truy nhập bộ nhớ ở chế độ MAX

Hình 4-9 là giản đồ xung của các tín hiệu trên bus hệ thống trong một chu kỳ đọc hay viết bộ nhớ. Trên giản đồ xung theo thời gian ta thấy có 4 tín hiệu liên quan đến chu kỳ bus: xung nhịp từ máy phát nhịp đồng hồ CLK, các tín hiệu địa chỉ/trạng thái. Các tín hiệu địa chỉ/số liệu, tín hiệu sẵn sàng READY để chỉ thị đã hoàn thành việc đọc số liệu



Hình 4-9. Các tín hiệu trên bus hệ thống trong một chu kỳ đọc hoặc viết bộ nhớ

a. Chu kỳ đọc bộ nhớ

Nhìn trên hình (a) ta thấy chu kỳ đọc bộ nhớ bao gồm các quá trình xảy ra như sau:

T1: Vi xử lý đưa ra tín hiệu điều khiển S2, S1, S0 tới bộ điều khiển bus, kích bộ đệm số liệu và địa chỉ hoạt động. Tiếp đó địa chỉ trên các chân từ A19 đến A0 được đưa vào bộ đệm địa chỉ. Tín hiệu HBE chỉ thị byte hoặc từ đang được đọc. Tín hiệu READY nhảy xuống mức thấp, nó chỉ nhảy lên cao một khi bộ nhớ đã được cấp xong số liệu.

T2: Chuyển hướng truyền số liệu trên bus. Đường BHE và A19 – A16 chuyển sang thông tin trạng thái. Các đường A15-A0 chuyển từ mode địa chỉ sang mode số liệu.

T3: Chu trình truyền số liệu bắt đầu. Chừng nào số liệu chưa ổn định trên D15-D0 các hiệu trạng thái S7- S3 xuất hiện. Khi toàn bộ số liệu được truyền xong vào bộ nhớ, bộ điều khiển nhớ sẽ nâng mức điện thế ở dây READY phải được đồng bộ trước bằng cách cho nó qua máy nhịp 8284 để phát ra xung READY đồng bộ với xung nhịp đưa vào vi xử lý. Như vậy số liệu được truyền từ bộ đệm bộ nhớ tời bộ đệm số liệu. Vi xử lý lúc này khởi phát việc nhận số liệu từ bộ đệm.

T4: vi xử lý kết thúc việc đọc số liệu vào sau ½ chu kỳ . Lúc này các bộ đệm bị cấm nhưng vi xử lý vẫn liên tục cho ra các tín hiệu trạng thái S7-S3. Sau khi kết thúc T4, bus hệ thống lại một lần nữa trở về trạng thái khởi phát.

b. Chu kỳ viết bộ nhớ

Các tín hiệu trên hình (b) giống như ở chu kỳ đọc, ngoại trừ tín hiệu địa chỉ/ số liệu

T1: Các xử lý giống trên và chỉ có bộ điều khiển bus được tác động vì viết số liệu.

T2: Hướng của bus địa chỉ / số liệu không cần đổi bời vì cả địa chỉ và số liệu đều là hướng ra. Do đó ngay sau khi cấp địa chỉ, vi xử lý có thể phát ra ngay số liệu vào bộ đệm ở xung nhịp đồng hồ trong T2. Bộ đệm số liệu truyền nó tới bộ đệm nhớ. Đồng thời bộ điều khiển nhớ sẽ điều khiển bộ nhớ viết số liệu vào nó.

T3: Sau khi hoàn thành việc viết số liệu bên trong bộ nhớ, bộ điều khiển nhớ sẽ nâng mức điện thế trên dây READY lên cao để chỉ thị tới vi xử lý.

T4: Vi xử lý kết thúc quá trình viết. Các bộ đệm bị cấm nhưng vi xử lý tiếp tục ra các tín hiệu trạng thái S7-S3.

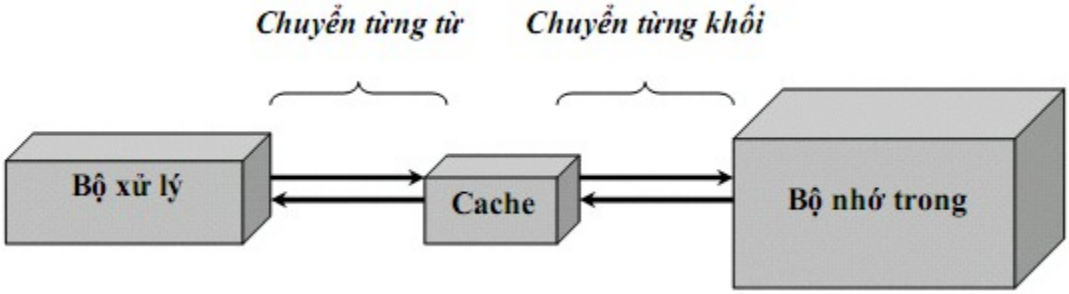
4. Bộ nhớ CACHE

Mục đích:

- *Giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache: nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên lý vận hành của bộ nhớ ảo.*

Mức cache -bộ nhớ trong trong bảng các cấp bộ nhớ có cơ cấu vận hành trong suốt đối với bộ xử lý. Với thao tác đọc bộ nhớ, bộ xử lý gởi một địa chỉ và nhận một dữ liệu từ bộ nhớ trong. Với thao tác ghi bộ nhớ, bộ xử lý viết một dữ liệu vào một ô nhớ với một địa chỉ được chỉ ra trong bộ nhớ. Để cho chương

trình vận hành bình thường thì cache phải chứa một phần con của bộ nhớ trong để bộ xử lý có thể thâm nhập vào các lệnh hoặc dữ liệu thường dùng từ bộ nhớ cache. Do dung lượng của bộ nhớ cache nhỏ nên nó chỉ chứa một phần chương trình nằm trong bộ nhớ trong. Để đảm bảo sự đồng nhất giữa nội dung của cache và bộ nhớ trong thì cache và bộ nhớ trong phải có cùng cấu trúc. Việc chuyển dữ liệu giữa cache và bộ nhớ trong là việc tải lên hay ghi xuống các *khối dữ liệu*. Mỗi khối chứa nhiều từ bộ nhớ tuỳ thuộc vào cấu trúc bộ nhớ cache. Sự lựa chọn kích thước của khối rất quan trọng cho vận hành của cache có hiệu quả.



Hình 4-10. Trao đổi dữ liệu giữa các thành phần CPU

Trước khi khảo sát vận hành của cache, ta xét đến các khái niệm liên quan:

- **Thành công cache (cache hit):** bộ xử lý tìm gặp phần tử cần đọc (ghi) trong cache.

- **Thất bại cache (cache miss):** bộ xử lý không gặp phần tử cần đọc (ghi) trong cache.

- **Trừng phạt thất bại cache (cache penalty):** Thời gian cần thiết để xử lý một thất bại cache. Thời gian bao gồm thời gian thâm nhập bộ nhớ trong cộng với thời gian chuyển khối chứa từ cần đọc từ bộ nhớ trong đến cache. Thời gian này tuỳ thuộc vào kích thước của khối.

Để hiểu được cách vận hành của cache, ta lần lượt xem xét và giải quyết bốn vấn đề liên quan đến các tình huống khác nhau xảy ra trong bộ nhớ trong.

vấn đề 1: ***Phải để một khối bộ nhớ vào chỗ nào của cache (sắp xếp khối)?***

vấn đề 2: ***Làm sao để tìm một khối khi nó hiện diện trong cache (nhận diện khối)?***

vấn đề 3: ***Khối nào phải được thay thế trong trường hợp thất bại cache (thay thế khối)?***

vấn đề 4: ***Việc gì xảy ra khi ghi vào bộ nhớ (chiến thuật ghi)?***

Giải quyết vấn đề 1: ***Phải để một khối bộ nhớ vào chỗ nào của cache (sắp xếp khối)?***

Một khối bộ nhớ được đặt vào trong cache theo một trong ba cách sau:

Kiểu tương ứng trực tiếp: Nếu mỗi khối bộ nhớ chỉ có một vị trí đặt khối duy nhất trong cache được xác định theo công thức:

K= i mod n

Trong đó:

K: vị trí khối đặt trong cache

i: số thứ tự của khối trong bộ nhớ trong

n: số khối của cache

Như vậy, trong kiểu xếp đặt khối này, mỗi vị trí đặt khối trong cache có thể chứa một trong các khối trong bộ nhớ cách nhau ***xn*** khối (x: 0,1,...,m; n: số khối của cache)

Số thứ tự khối cache	Số thứ tự của khối trong bộ nhớ trong
0	0, n, 2n,...mn
1	1,n+1, 2n+1,...,mn+1
…	…
n-1	n-1, 2n-1,...mn-1

Kiểu hoàn toàn phối hợp: trong kiểu đặt khối này, một khối trong bộ nhớ trong có thể được đặt vào vị trí bất kỳ trong cache.

Như vậy, trong kiểu xếp đặt khối này, mỗi vị trí đặt khối trong cache có thể chứa một trong tất cả các khối trong bộ nhớ

Kiểu phối hợp theo tập hợp: với cách tổ chức này, cache bao gồm các tập hợp của các khối cache. Mỗi tập hợp của các khối cache chứa số khối như nhau. Một khối của bộ nhớ trong có thể được đặt vào một số vị trí khối giới hạn trong tập hợp được xác định bởi công thức***:*** **K= i mod s**

Trong đó: K: vị trí khối đặt trong cache

i: số thứ tự của khối trong bộ nhớ trong

s: số lượng tập hợp trong cache.

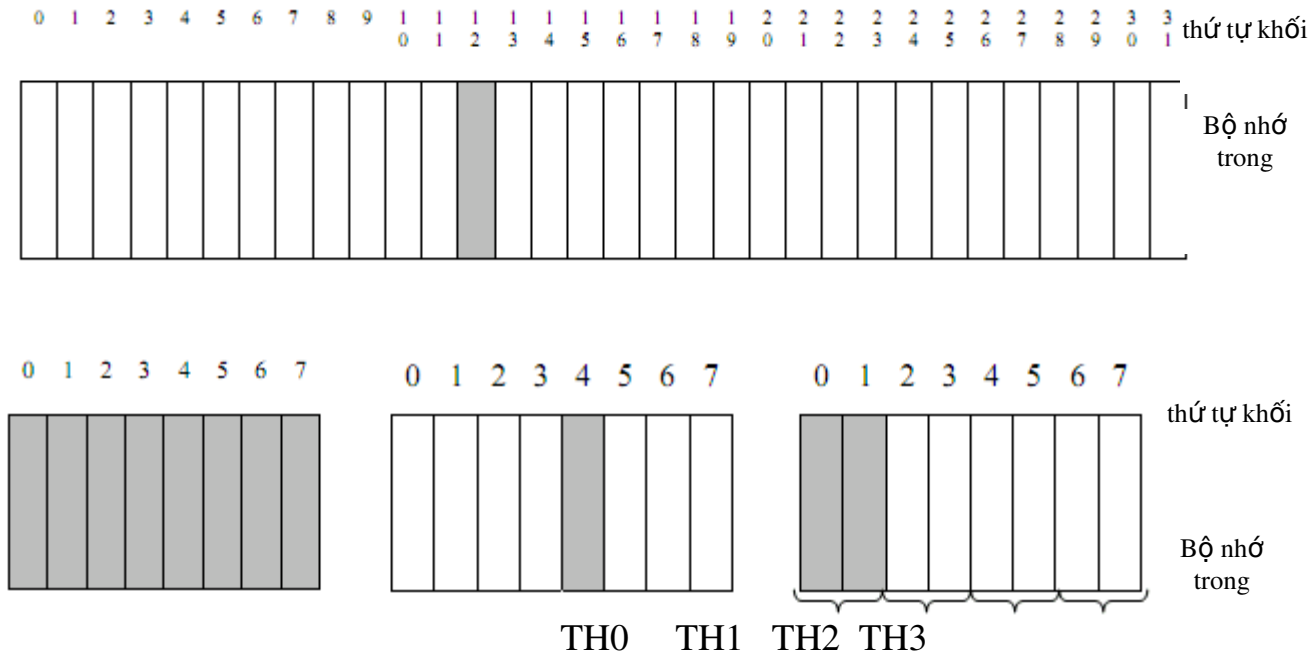
Trong cách đặt khối theo kiểu phối hợp theo tập hợp, nếu tập hợp có ***m*** khối, sự tương ứng giữa các khối trong bộ nhớ trong và các khối của cache được gọi là phối hợp theo tập hợp ***m*** khối.

Nếu m=1 (mỗi tập hợp có 1 khối), ta có kiểu tương ứng trực tiếp.

Nếu m=n (n: số khối của cache), ta có kiểu tương hoàn toàn phối hợp.

Hiện nay, phần lớn các cache của các bộ xử lý đều là kiểu tương ứng trực tiếp hay kiểu phối hợp theo tập hợp (mỗi tập hợp gồm 2 hoặc 4 khối).

Ví dụ: Bộ nhớ trong có 32 khối, cache có 8 khối, mỗi khối gồm 32 byte, khối thứ 12 của bộ nhớ trong được đưa vào cache.



Hoàn toàn phối hợp Tương ứng trực tiếp Phối hợp theo tập hợp

Giải quyết vấn đề 2: ***Làm sao để tìm một khối khi nó hiện diện trong cache (nhận diện khối)?***

Mỗi khối của cache đều có một nhãn địa chỉ cho biết số thứ tự của các khối bộ nhớ trong đang hiện diện trong cache. Nhãn của một khối của cache có thể chứ thông tin cần thiết được xem xét để biết được các khối nằm trong cache có chứa thông tin mà bộ xử lý cần đọc hay không. Tất cả các nhãn đều được xem xét song song (trong kiểu tương ứng trực tiếp và phối hợp theo tập hợp) vì tốc độ là yếu tố then chốt. Để biết xem một khối của của cache có chứa thông tin mà bộ xử lý cần tìm hay không, người ta thêm một bit đánh dấu (valid bit) vào nhãn để nói lên khối đó có chứa thông tin mà bộ xử lý cần tìm hay không.

Như đã mô tả ở phần đầu, với thao tác đọc (ghi) bộ nhớ, bộ xử lý đưa ra một địa chỉ và nhận (viết vào) một dữ liệu từ (vào) bộ nhớ trong. Địa chỉ mà bộ xử lý đưa ra có thể phân tích thành hai thành phần: phần nhận dạng số thứ tự khối và phần xác định vị trí từ cần đọc trong khối.

Tương ứng với ba kiểu lắp đặt khối đã xét, ta có:

a. Căn cứ vào tổ chức số từ trong khối bộ nhớ mà số bit trong địa chỉ xác định vị trí từ cần đọc trong khối. Cách này đúng với cả ba cách xếp đặt khối đã xét.

b. Phần nhận dạng số thứ tự khối sẽ khác nhau tuỳ thuộc vào cách xếp đặt khối, trường chỉ số khối được so sánh với nhãn của cache để xác định khối trong cache.

Dữ liệu được bộ xử lý đọc cùng lúc với việc đọc nhãn. Phần chỉ số khối của khối trong bộ nhớ trong được so sánh với bảng tương quan để xác định khối

có nằm trong cache hay không. Để chắc rằng nhãn chứa thông tin đúng đắn (tức là khối có chứa từ mà bộ xử lý cần đọc-ghi), nếu việc so sánh nhãn của khối cache giống với số thứ tự khối, bit đánh dấu (Valid bit) phải được bật lên. Ngược lại, kết quả so sánh được bỏ qua. Bộ xử lý căn cứ vào phần xác định từ trong khối để đọc (ghi) dữ liệu từ (vào) cache.

- Đối với kiểu tương ứng trực tiếp, phần nhận dạng chỉ số khối được chia thành hai phần:

+ Phần chỉ số khối cache: chỉ ra số thứ tự khối cache tương ứng cần xem xét.

+ Phần nhãn: so sánh tương ứng với nhãn của khối cache được chỉ ra bởi phần chỉ số khối

Chỉ số khối trong bộ nhớ		Địa chỉ từ cần đọc trong khối
Nhãn	Chỉ số khối cache	

Đối với kiểu hoàn toàn phối hợp, phần nhận dạng chỉ số khối trong địa chỉ sẽ được so sánh với nhãn của tất cả các khối cache.

Chỉ số khối	Địa chỉ từ cần đọc trong khối

Đối với kiểu phối hợp theo tập hợp, phần nhận dạng chỉ số khối được chia thành hai phần:

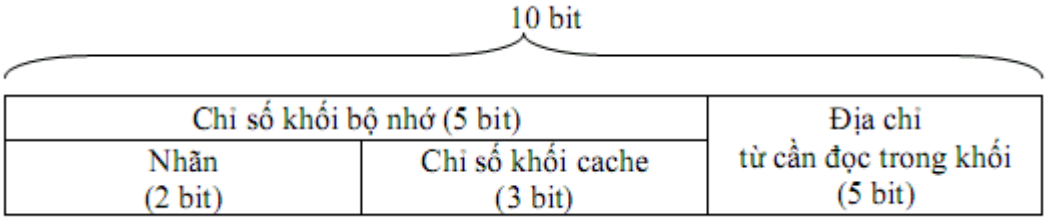
+ Phần chỉ số tập hợp: chỉ ra số thứ tự tập hợp trong cache cần xem xét.

+ Phần nhãn: so sánh tương ứng với nhãn của các khối cache thuộc tập hợp được chỉ ra bởi phần chỉ số tập hợp.

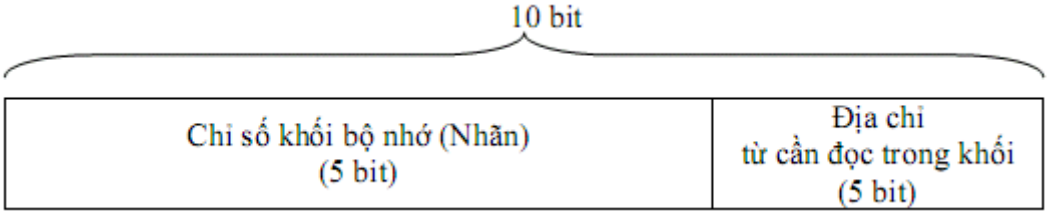
Chỉ số khối bộ nhớ		Địa chỉ từ cần đọc trong khối
Nhãn	Chỉ số tập hợp	

Ví dụ: phân tích địa chỉ một từ trong được cho ở trên, địa chỉ xác định một từ trong bộ nhớ có 10 bit, tuỳ theo cách xếp đặt khối mà ta có thể phân tích địa chỉ này thành các thành phần như sau:

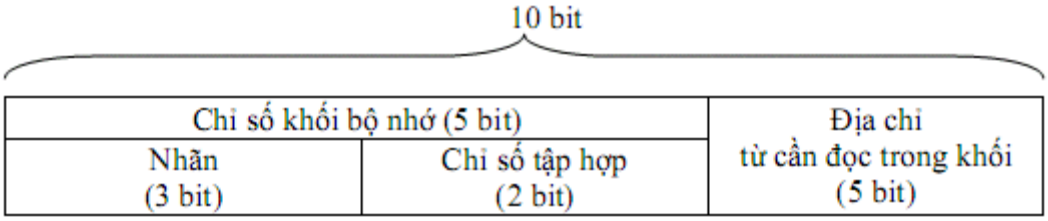
Đối với kiểu tương ứng trực tiếp:



Đối với kiểu hoàn toàn phối hợp:



Đối với kiểu phối hợp theo tập hợp, giả sử cache gồm 4 tập hợp, mỗi tập hợp gồm hai khối:



Giải quyết vấn đề 3: ***Khối nào phải được thay thế trong trường hợp thất bại cache (thay thế khối)?***

Khi có thất bại cache, bộ điều khiển cache thâm nhập bộ nhớ trong và chuyển khối mà bộ xử lý cần đọc (ghi) vào cache. Như vậy, khối nào trong cache sẽ bị thay thế bởi khối mới được chuyển lên. Đối với kiểu tương ứng trực tiếp, vị trí đặt khối không có sự lựa chọn, nó được xác định bởi trường chỉ số khối cache trong địa chỉ của từ cần đọc (ghi). Nếu cache là kiểu hoàn toàn phối hợp hay phối hợp theo tập hợp thì khi thất bại phải chọn lựa thay thế trong nhiều khối. Có bốn chiến thuật chủ yếu dùng để chọn khối thay thế trong cache:

- *Thay thế ngẫu nhiên:* để phân bố đồng đều việc thay thế, các khối cần thay thế trong cache được chọn ngẫu nhiên.
- *Khối xưa nhất* (LRU: Least Recently Used): các khối đã được thâm nhập sẽ được đánh dấu và khối bị thay thế là khối không được dùng từ lâu nhất.
- *Vào trước ra trước* (FIFO: First In First Out): Khối được đưa vào cache đầu tiên, nếu bị thay thế, khối đó sẽ được thay thế trước nhất.
- *Tần số sử dụng ít nhất* (LFU: Least Frequently Used): Khối trong cache được tham chiếu ít nhất

Điều này sử dụng hệ quả của nguyên tắc sử dụng ô nhớ theo thời gian: nếu các khối mới được dùng có khả năng sẽ được dùng trong tương lai gần, khối bị thay thế là khối không dùng trong thời gian lâu nhất.

Giải quyết vấn đề 4: ***Việc gì xảy ra khi ghi vào bộ nhớ (chiến thuật ghi)?***

Thông thường bộ xử lý thâm nhập cache để đọc thông tin. Chỉ có khoảng 15% các thâm nhập vào cache là để thực hiện thao tác ghi (con số này là 33% với các tính toán vectơ-vectơ và 55% đối với các phép dịch chuyển ma trận). Như vậy, để tối ưu hoá các hoạt động của cache, các nhà thiết kế tìm cách tối ưu hoá việc đọc bởi vì các bộ xử lý phải đợi đến khi việc đọc hoàn thành nhưng sẽ không đợi đến khi việc ghi hoàn tất. Hơn nữa, một khối có thể được đọc, so sánh và như thế việc đọc một khối có thể được bắt đầu khi chỉ số khối được biết. Nếu thao tác đọc thành công, dữ liệu ô nhớ cần đọc sẽ được giao ngay cho bộ xử lý. Chú ý rằng, khi một khối được ánh xạ từ bộ nhớ trong vào cache, việc đọc nội dung của khối cache không làm thay đổi nội dung của khối so với khối còn nằm trong bộ nhớ trong.

Đối với việc ghi vào bộ nhớ thì không giống như trên, việc thay đổi nội dung của một khối không thể bắt đầu trước khi nhãn được xem xét để biết có thành công hay thất bại. Thao tác ghi vào bộ nhớ sẽ tốn nhiều thời gian hơn thao tác đọc bộ nhớ. Trong việc ghi bộ nhớ còn có một khó khăn khác là bộ xử lý cho biết số byte cần phải ghi, thường là từ 1 đến 8 byte. Để đảm bảo đồng nhất dữ liệu khi lưu trữ, có hai cách chính để ghi vào cache:

- *Ghi đồng thời*: Thông tin được ghi đồng thời vào khối của cache và khối của bộ nhớ trong. Cách ghi này làm chậm tốc độ chung của hệ thống. Các ngoại vi có thể truy cập bộ nhớ trực tiếp

- *Ghi lại*: Để đảm bảo tốc độ xử lý của hệ thống, thông tin cần ghi chỉ được ghi vào khối trong cache. Để quản lý sự khác biệt nội dung giữa khối của cache và khối của bộ nhớ trong, một bit trạng thái (Dirty bit hay Update bit) được dùng để chỉ thị. Khi một thao tác ghi vào trong cache, bit trạng thái (Dirty bit hay Update bit) của khối cache sẽ được thiết lập. Khi một khối bị thay thế, khối này sẽ được ghi lại vào bộ nhớ trong chỉ khi bit trạng thái đã được thiết lập. Với cách ghi này, các ngoại vi liên hệ đến bộ nhớ trong thông qua cache.

Khi có một thất bại ghi vào cache thì phải lựa chọn một trong hai giải pháp sau:

- *Ghi có nạp*: khối cần ghi từ bộ nhớ trong được nạp vào trong cache như mô tả ở trên. Cách này thường được dùng trong cách ghi lại.

- *Ghi không nạp*: khối được thay đổi ở bộ nhớ trong không được đưa vào cache. Cách này được dùng trong cách ghi đồng thời.

Trong các tổ chức có nhiều hơn một bộ xử lý với các tổ chức cache và bộ nhớ chia sẻ, các vấn đề liên quan đến tính đồng nhất của dữ liệu cần được đảm bảo. Sự thay đổi dữ liệu trên một cache riêng lẻ sẽ làm cho dữ liệu trên các hệ thống cache và bộ nhớ liên quan không đồng nhất. Vấn đề trên có thể được giải quyết bằng một trong các hệ thống cache tổ chức như sau:

+ Mỗi bộ điều khiển cache sẽ theo dõi các thao tác ghi vào bộ nhớ từ các bộ phận khác. Nếu thao tác ghi vào phần bộ nhớ chia sẻ được ánh xạ vào cache của nó quản lý, bộ điều khiển cache sẽ vô hiệu hoá sự thâm nhập này. Chiến lược này phụ thuộc vào cách ghi đồng thời trên tất cả các bộ điều khiển cache.

+ Một vi mạch được dùng để điều khiển việc cập nhật, một thao tác ghi vào bộ nhớ từ một cache nào đó sẽ được cập nhật trên các cache khác.

+ Một vùng nhớ chia sẻ cho một hay nhiều bộ xử lý thì không được ánh xạ lên cache. Như vậy, tất cả các thâm nhập vào vùng nhớ chia sẻ này đều bị thất bại cache.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 4

*****

1. Nêu các loại bộ bán dẫn và đặc điểm của chúng?
2. Mục tiêu của các cấp bộ nhớ?
3. Nêu hai nguyên tắc mà cache dựa vào đó để vận hành.
5. Các nguyên nhân chính gây thất bại cache?
6. Các giải pháp đảm bảo tính đồng nhất dữ liệu trong hệ thống bộ đa xử lý có bộ nhớ chia sẻ dùng chung?
7. Các cách nới rộng dãy thông của bộ nhớ trong?
8. Tại sao phải dùng bộ nhớ ảo?
9. Sự khác biệt giữa cache và bộ nhớ ảo?

*Chương 5:*THIẾT BỊ NHẬP XUẤT

Mã chương: MH12 – 05.

Mục đích: Giới thiệu một số thiết bị lưu trữ ngoài như: đĩa từ, đĩa quang, thẻ nhớ, băng từ. Giới thiệu hệ thống kết nối cơ bản các bộ phận bên trong máy tính. Cách giao tiếp giữa các ngoại vi và bộ xử lý. Phương pháp an toàn dữ liệu trên thiết bị lưu trữ ngoài.

Yêu cầu: Sinh viên phải nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ phận bên trong máy tính, cách giao tiếp giữa các ngoại vi và bộ xử lý. Biết được cấu tạo và các vận hành của các loại thiết bị lưu trữ ngoài và phương pháp an toàn dữ liệu trên đĩa cứng.

1. Đĩa từ

Mục đích:

- *Giới thiệu một số thiết bị lưu trữ ngoài như: đĩa từ.*

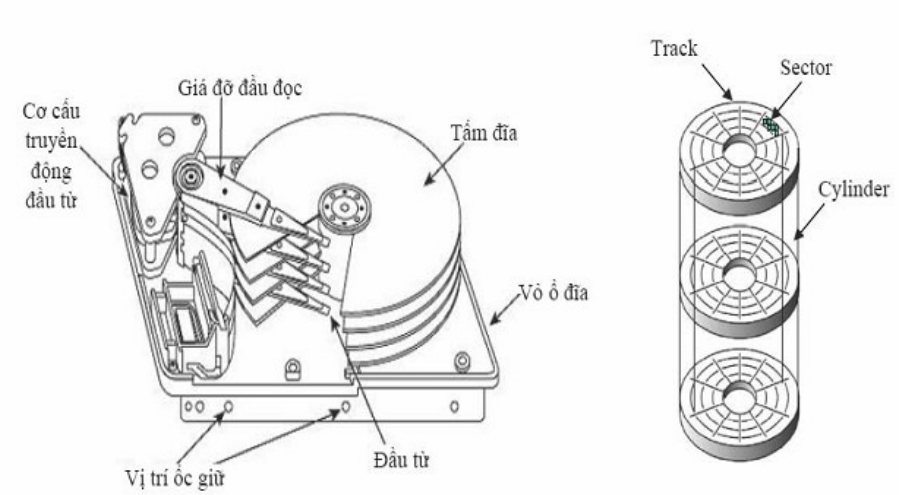
Dù rằng công nghệ mới không ngừng phát minh nhiều loại bộ phận lưu trữ một lượng thông tin lớn nhưng đĩa từ vẫn giữ vị trí quan trọng từ năm 1965. Đĩa từ có hai nhiệm vụ trong máy tính.

- Lưu trữ dài hạn các tập tin.
- Thiết lập một cấp bộ nhớ bên dưới bộ nhớ trong để làm bộ nhớ ảo lúc chạy chương trình.

Do đĩa mềm dần được các thiết bị lưu trữ khác có các tính năng ưu việt hơn nên chúng ta không xét đến thiết bị này trong chương trình mà chỉ nói đến đĩa cứng. Trong tài liệu này mô tả một cách khái quát cấu tạo, cách vận hành cũng như đề cập đến các tính chất quan trọng của đĩa cứng.

Một đĩa cứng chứa nhiều lớp đĩa (từ 1 đến 4) quay quanh một trục khoảng 3.600-15.000 vòng mỗi phút. Các lớp đĩa này được làm bằng kim loại với hai mặt được phủ một chất từ tính (hình V.1). Đường kính của đĩa thay đổi từ 1,3 inch đến 8 inch. Mỗi mặt của một lớp đĩa được chia thành nhiều đường tròn đồng trục gọi là *rãnh*. Thông thường mỗi mặt của một lớp đĩa có từ 10.000 đến gần 30.000 rãnh. Mỗi rãnh được chia thành nhiều *cung* (sector) dùng chứa thông tin. Một rãnh có thể chứa từ 64 đến 800 cung. Cung là đơn vị nhỏ nhất mà máy tính có thể đọc hoặc viết (thông thường khoảng 512 bytes). Chuỗi thông tin ghi trên mỗi cung gồm có: số thứ tự của cung, một khoảng trống, số liệu của cung đó bao gồm cả các mã sửa lỗi, một khoảng trống, số thứ tự của cung tiếp theo.

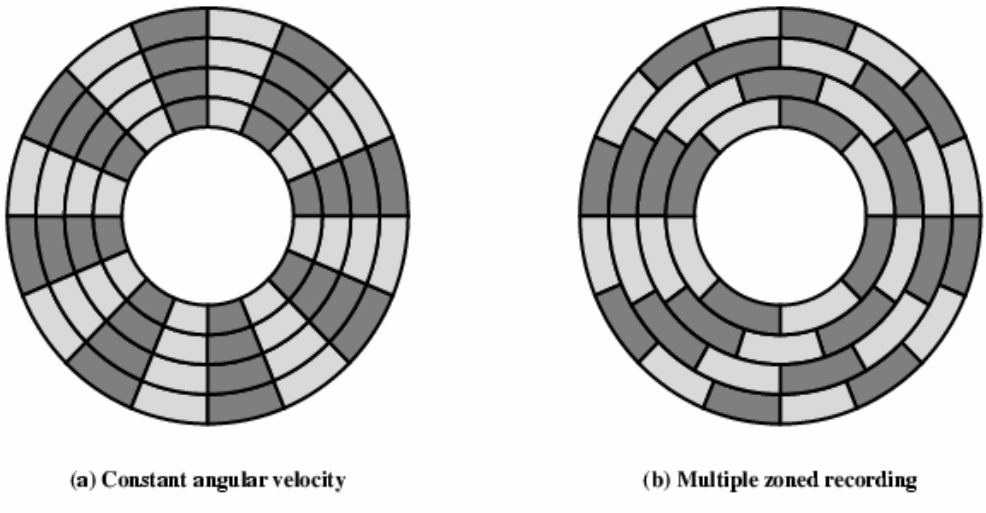
Với kỹ thuật ghi mật độ không đều, tất cả các rãnh đều có cùng một số cung, điều này làm cho các cung dài hơn ở các rãnh xa trục quay có mật độ ghi thông tin thấp hơn mật độ ghi trên các cung nằm gần trục quay.



Hình 5-1. Cấu tạo của một đĩa cứng

Hình 5-2.Hình ảnh bên ngoài của ổ cứng

Với công nghệ ghi với mật độ đều, người ta cho ghi nhiều thông tin hơn ở các rãnh xa trục quay. Công nghệ ghi này ngày càng được dùng nhiều với sự ra đời của các chuẩn giao diện thông minh như chuẩn SCSI.



Hình 5-3. Mật độ ghi dữ liệu trên các loại đĩa cứng

Để đọc hoặc ghi thông tin vào một cung, ta dùng một đầu đọc ghi di động áp vào mỗi mặt của mỗi lớp đĩa. Các đầu đọc/ghi này được gắn chặt vào một thanh làm cho chúng cùng di chuyển trên một đường bán kính của mỗi lớp đĩa và như thế tất cả các đầu này đều ở trên những rãnh có cùng bán kính của các lớp

đĩa. Từ "*trụ*" (cylinder) được dùng để gọi tất cả các rãnh của các lớp đĩa có cùng bán kính và nằm trên một hình trụ.

Người ta luôn muốn đọc nhanh đĩa từ nên thông thường ổ đĩa đọc nhiều hơn số dữ liệu cần đọc; người ta nói đây là cách đọc trước. Để quản lý các phức tạp khi kết nối (hoặc ngưng kết nối) lúc đọc (hoặc ghi) thông tin, và việc đọc trước, ổ đĩa cần có bộ điều khiển đĩa.

Công nghiệp chế tạo đĩa từ tập trung vào việc nâng cao dung lượng của đĩa mà đơn vị đo lường là mật độ trên một đơn vị bề mặt

Bảng 5-1. Thông số kỹ thuật của đĩa cứng

Bảng thông số kỹ thuật đĩa cứng	
Dung lượng tối đa	Có thể đạt 500 GB
Số lượng cần đọc	1-8
Số tấm ghi (đĩa)	1-4
Cache (bộ đệm)	2-16 MB
Số cung (Sectors -512 byte/sector)	Xxx,xxx,xxx
Tốc độ quay đĩa (RPM)	3600-15000
Mật độ	Có thể đạt 95 Gb/in
Mật độ rãnh (TPI-Max Track/Inch)	Có thể đạt 120.000
Mật độ ghi BPI (Max Bits/Inch)	Có thể đạt 702.000
Tốc độ dữ liệu tối đa (internal)	Có thể đạt 900 Mb/s
Tốc độ truyền dữ liệu với ngoại vi	Có thể đạt 320 Mb/s
Thời gian chuyển track R/W	Có thể đạt 15 ms
Thời gian quay nửa vòng	Có thể đạt 6 ms

2. Đĩa quang

Mục đích:

- *Giới thiệu một số thiết bị lưu trữ ngoài như: đĩa quang.*

Các thiết bị lưu trữ quang rất thích hợp cho việc phát hành các sản phẩm văn hoá, sao lưu dữ liệu trên các hệ thống máy tính hiện nay. Ra đời vào năm 1978, đây là sản phẩm của sự hợp tác nghiên cứu giữa hai công ty Sony và Philips trong công nghiệp giải trí. Từ năm 1980 đến nay, công nghiệp đĩa quang phát triển mạnh trong cả hai lĩnh vực giải trí và lưu trữ dữ liệu máy tính. Quá trình đọc thông tin dựa trên sự phản chiếu của các tia laser năng lượng thấp từ lớp lưu trữ dữ liệu. Bộ phận tiếp nhận ánh sáng sẽ nhận biết được những điểm mà tại đó tia laser bị phản xạ mạnh hay biến mất do các vết khắc (pit) trên bề mặt đĩa. Các tia phản xạ mạnh chỉ ra rằng tại điểm đó không có lỗ khắc và điểm này được gọi là điểm nền (land). Bộ nhận ánh sáng trong ổ đĩa thu nhận các tia phản xạ và khuếch tán được khúc xạ từ bề mặt đĩa. Khi các nguồn sáng được thu nhận, bộ vi xử lý sẽ dịch các mẫu sáng thành các bit dữ liệu hay âm thanh.

Các lỗ trên CD sâu 0,12 micron và rộng 0,6 micron (1 micron bằng một phần ngàn mm). Các lỗ này được khắc theo một track hình xoắn ốc với khoảng cách 1,6 micron giữa các vòng, khoảng 16.000 track/inch. Các lỗ (pit) và nền (land) kéo dài khoản 0,9 đến 3,3 micron. Track bắt đầu từ phía trong và kết thúc ở phía ngoài theo một đường khép kín các rìa đĩa 5mm. Dữ liệu lưu trên CD thành từng khối, mỗi khối chứa 2.352 byte. Trong đó, 304 byte chứa các thông tin về bit đồng bộ, bit nhận dạng (ID), mã sửa lỗi (ECC), mã phát hiện lỗi (EDC). Còn lại 2.048 byte chứa dữ liệu. Tốc độ đọc chuẩn của CD-ROM là 75 khối/s hay 153.600 byte/s hay 150KB/s (1X).

Dưới đây là một số loại đĩa quang thông dụng.

CD (Compact Disk): Đĩa quang không thể xoá được, dùng trong công nghiệp giải trí (các đĩa âm thanh được số hoá). Chuẩn đĩa có đường kính 12 cm, âm thanh phát từ đĩa khoảng 60 phút (không dừng).

CD-ROM (Compact Disk Read Only Memory): Đĩa không xoá dùng để chứa các dữ liệu máy tính. Chuẩn đĩa có đường kính 12 cm, lưu trữ dữ liệu hơn 650 MB. Khi phát hành, đĩa CD-ROM đã có chứa nội dung. Thông thường, dĩa CD-ROM được dùng để chứa các phần mềm và các chương trình điều khiển thiết bị.

CD-R (CD-Recordable): Giống như đĩa CD, đĩa mới chưa có thông tin, người dùng có thể ghi dữ liệu lên đĩa một lần và đọc được nhiều lần. Dữ liệu trên đĩa CD-R không thể bị xoá.

CD-RW (CD-Rewritable): Giống như đĩa CD, đĩa mới chưa có thông tin, người dùng có thể ghi dữ liệu lên đĩa, xoá và ghi lại dữ liệu trên đĩa nhiều lần.

DVD (Digital Video Disk - Digital Versatile Disk): Ra đời phục vụ cho công nghiệp giải trí, đĩa chứa các hình ảnh video được số hoá. Ngày nay, DVD được sử dụng rộng rãi trong các ứng dụng công nghệ thông tin. Kích thước đĩa có hai loại: 8cm và 12 cm. Đĩa DVD có thể chứa dữ liệu trên cả hai mặt đĩa, dung lượng tối đa lên đến 17GB. Các thông số kỹ thuật của đĩa DVD-ROM (loại đĩa chỉ đọc) so với CD-ROM. Tốc độ đọc chuẩn (1X) của DVD là 1.3MB/s (1X của DVD tương đương khoảng 9X của CDROM).

DVD-R (DVD-Recordable): Giống như đĩa DVD-ROM, người dùng có thể ghi dữ liệu lên đĩa một lần và đọc được nhiều lần. Đĩa này chỉ có thể ghi được trên một mặt đĩa, dung lượng ghi trên mỗi mặt tối đa là 4.7 GB.

DVD-RW (DVD-Rewritable): Giống như đĩa DVD-ROM, người dùng có thể ghi, xoá và ghi lại dữ liệu lên đĩa nhiều lần.. Đĩa này cũng có thể ghi được trên một mặt đĩa, dung lượng ghi trên mỗi mặt tối đa là 4.7 GB.

Bảng 5-2. So sánh một số thông số của hai loại đĩa CDROM và DVDROM

Đặc trưng	CDROM	DVDROM
Kích thước Pit	0.834 micron	0.4 micron
Khoảng cách rãnh	1.6 micron	0.74 micron
Số lớp dữ liệu trên đĩa	1 lớp	2 lớp
Số mặt đĩa	1 mặt	1 - 2 mặt
Dung lượng	640-700 MB	1.36 – 17 GB
Độ phân giải phim	VCD=320x200	720x640

Với các đặc tính của đĩa quang, giá thành ngày càng thấp, được xem như một phương tiện thích hợp để phân phối các phần mềm cho máy vi tính. Ngoài ra, đĩa quang còn được dùng để lưu trữ lâu dài các dữ liệu thay thế cho băng từ.

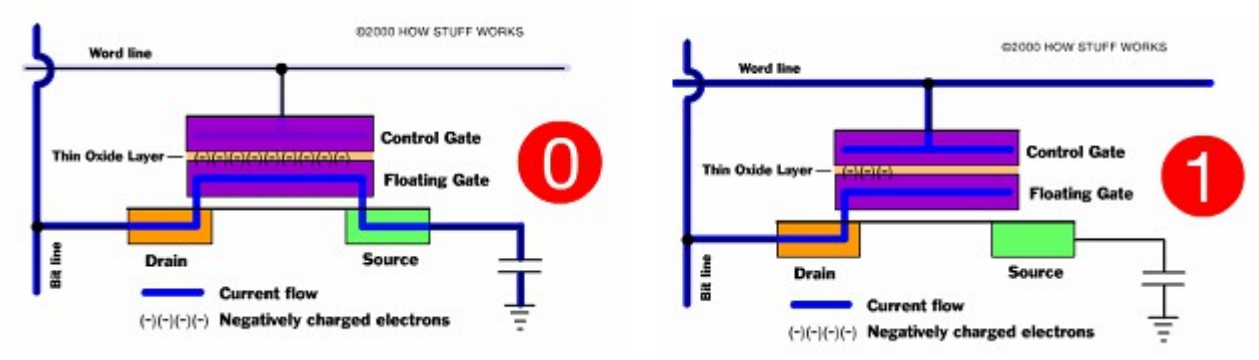
3. Các loại thẻ nhớ

Mục đích:

- *Giới thiệu một số thiết bị lưu trữ ngoài như: thẻ nhớ.*

Hiện nay, thẻ nhớ là một trong những công nghệ mới nhất được dùng làm thiết bị lưu trữ. Thẻ nhớ flash là một dạng bộ nhớ bán dẫn EEPROM(công nghệ dùng để chế tạo các chip BIOS trên các vỉ mạch chính), được cấu tạo bởi các hàng và các cột. Mỗi vị trí giao nhau là một ô nhớ gồm có hai transistor, hai transistor này cách nhau bởi một lớp ô-xít mỏng. Một transistor được gọi là *floating gate* và transistor còn lại được gọi là *control gate*. Floating gate chỉ có thể nối kết với hàng (word line) thông qua control gate. Khi đường kết nối được thiết lập, bit có giá trị 1. Để chuyển sang giá trị 0 theo một qui trình có tên *Fowler-Nordheim tunneling*. Tốc độ, yêu cầu về dòng điện cung cấp thấp và đặc biệt với kích thước nhỏ gọn của các loại thẻ nhớ làm cho kiểu bộ nhớ này được dùng rộng rãi trong công nghệ lưu trữ và giải trí hiện nay.

Hình 5-4.Hình ảnh một số USB



Hình 5-5.Minh hoạ hai trạng thái của một bit nhớ trong thẻ nhớ

4. Băng từ

Mục đích:

- *Giới thiệu một số thiết bị lưu trữ ngoài như: băng từ.*

Băng từ có cùng công nghệ với các đĩa từ nhưng khác đĩa từ hai điểm:

- Việc thâm nhập vào đĩa từ là ngẫu nhiên còn việc thâm nhập vào băng từ là tuần tự. Như vậy việc tìm thông tin trên băng từ mất nhiều thời gian hơn việc tìm thông tin trên đĩa từ.

- Đĩa từ có dung lượng hạn chế còn băng từ gồm có nhiều cuộn băng có thể lấy ra khỏi máy đọc băng nên dung lượng của băng từ là rất lớn (hàng trăm GB). Với chi phí thấp, băng từ vẫn còn được dùng rộng rãi trong việc lưu trữ dữ liệu dự phòng.

Các băng từ có chiều rộng thay đổi từ 0,38cm đến 1,27 cm được đóng thành cuộn và được chứa trong một hộp bảo vệ. Dữ liệu ghi trên băng từ có cấu trúc gồm một số các rãnh song song theo chiều dọc của băng.

Có hai cách ghi dữ liệu lên băng từ:

Ghi nối tiếp: với kỹ thuật ghi xoắn ốc, dữ liệu ghi nối tiếp trên một rãnh của băng từ, khi kết thúc một rãnh, băng từ sẽ quay ngược lại, đầu từ sẽ ghi dữ liệu trên rãnh mới tiếp theo nhưng với hướng ngược lại. Quá trình ghi cứ tiếp diễn cho đến khi đầy băng từ.

Ghi song song: để tăng tốc độ đọc-ghi dữ liệu trên băng từ, đầu đọc - ghi có thể đọc-ghi một số rãnh kề nhau đồng thời. Dữ liệu vẫn được ghi theo chiều dọc băng từ nhưng các khối dữ liệu được xem như ghi trên các rãnh kề nhau. Số rãnh ghi đồng thời trên băng từ thông thường là 9 rãnh (8 rãnh dữ liệu - 1byte và một rãnh kiểm tra lỗi).

5. Các chuẩn về BUS

Mục đích:

- *Giới thiệu hệ thống kết nối cơ bản các bộ phận bên trong máy tính. Cách giao tiếp giữa các ngoại vi và bộ xử lý.*

Số lượng và chủng loại các bộ phận vào/ra không cần định trước trong các hệ thống xử lý thông tin. Điều này giúp cho người sử dụng máy tính dùng bộ phận vào/ra nào đáp ứng được các yêu cầu của họ. Vào/ra là giao diện trên đó các bộ phận (thiết bị) được kết nối vào hệ thống. Nó có thể xem như một bus nới rộng dùng để kết nối thêm ngoại vi vào máy tính. Các chuẩn làm cho việc nối kết các ngoại vi vào máy tính được dễ dàng; bởi vì, trong khi các nhà thiết kế-sản xuất máy tính và các nhà thiết kế-sản xuất ngoại vi có thể thuộc các công ty khác nhau. Sự tồn tại các chuẩn về bus là rất cần thiết. Như vậy, nếu nhà thiết kế máy tính và nhà thiết kế ngoại vi tôn trọng các chuẩn về bus này thì các ngoại vi có thể kết nối dễ dàng vào máy tính. Chuẩn của bus vào/ra là tài liệu quy định cách kết nối ngoại vi vào máy tính.

Các máy tính quá thông dụng thì các chuẩn về bus vào/ra của chúng có thể được xem là chuẩn cho các hãng khác (ví dụ: trước đây, UNIBUS của máy PDP 11, các chuẩn về bus của máy IBM PC, AT và hiện nay là các chuẩn của hãng Intel liên quan đến các máy vi tính). Các chuẩn về bus phải được các cơ quan về chuẩn như ISO, ANSI và IEEE công nhận.

Một máy vi tính có thể có nhiều loại BUS như sau:

BUS bộ xử lý, còn gọi khác là Back side (BSB): là các đường truyền giữa vi xử lý và các mạch đệm trung gian, thường là đường truyền giữa bộ xử lý và bộ nhớ cache ngoại L2 hoặc3. BUS này hoạt động với tốc độ nhanh nhất so với các loại BUS khác và không bị tắt nghẽn. Nó cũng bao gồm các BUS thành phần dữ liệu, địa chỉ và điều khiển. Thí dụ trong hệ thống pentium, Bus xử lý có 64 đường dữ liệu, 32 đường địa chỉ và các đường điều khiển.

BUS hệ thống, còn gọi là front side bus (FSB): được sử dụng để truyền thông tin giữa vi xử lý và bộ nhớ chính RAM cũng như tới các ổ đĩa,vv… bus này hoặc là thành phần của chính bus bộ xử lý hoặc trong nhiều trường hợp được phân cách với bus bộ xử lý bằng các mạch đệm là các chip chuyên dụng. Với các hệ thống chạy ở tốc độ đồng hồ bản mạch chính cao sẽ có một chip điều khiển bộ nhớ cho phép điều khiển sự ghép nối giữa các bus bộ xử lý có tốc độ nhanh hơn và bộ nhớ chính có tốc độ truy xuất chậm hơn. Do đó thông tin truyền trên bus hệ thống được truyền với tốc độ chậm hơn so với thông tin trên bus bộ xử lý.

BUS vào/ra còn gọi là bus mở rộng: cho phép sử lý thông tin được với các thiết bị ngoại vi. Nó cho phép bổ sung vào hệ thống máy tính các thiết bị để ở rông tính năng của máy vi tính. Các khe cắm mở rộng được nối vào bus mở rộng. Các bản mạch ghép nối được cắm vào các khe cắm này. Do đó khi nói về chuẩn cho một loại bus mở rộng nào đó cũng có nghĩa là nói các khe cắm mở rộng và card dùng nó.

Trong các máy vi tính hiện nay, nhiều ngoại vi được tích hợpngay trên bản mạch chính. Thí dụ chúng có ít nhất 2 bộ điều khiển chuẩn ghép nối ổ đĩa IDE (cũ) và hiện nay là chuẩn sata (sơ và thứ cấp)

Cable theo chuẩn SATA

Power cable

Cable IDE

Hình 5-6. Cable dữ liệu và Cable điện

6. An toàn dữ liệu trong lưu trữ

Mục đích:

- *Phương pháp an toàn dữ liệu trên thiết bị lưu trữ ngoài.*

Người ta thường chú trọng đến sự an toàn trong lưu giữ thông tin ở đĩa từ hơn là sự an toàn của thông tin trong bộ xử lý. Bộ xử lý có thể hư mà không làm tổn hại đến thông tin. Ổ đĩa của máy tính bị hư có thể gây ra các thiệt hại rất to lớn.

Một phương pháp giúp tăng cường độ an toàn của thông tin trên đĩa từ là dùng một mảng đĩa từ. Mảng đĩa từ này được gọi là *Hệ thống đĩa dự phòng* (*RAID - Redundant Array of Independent Disks*). Cách lưu trữ dư thông tin làm tăng giá tiền và sự an toàn (ngoại trừ RAID 0). Cơ chế RAID có các đặc tính sau:

1. RAID là một tập hợp các ổ đĩa cứng (vật lý) được thiết lập theo một kỹ thuật mà hệ điều hành chỉ "nhìn thấy" chỉ là một ổ đĩa (logic) duy nhất.

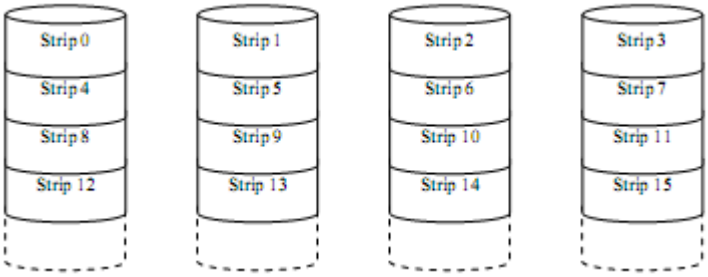
2. Với cơ chế đọc/ghi thông tin diễn ra trên nhiều đĩa (ghi đan chéo hay soi gương).

3. Trong mảng đĩa có lưu các thông tin kiểm tra lỗi dữ liệu; do đó, dữ liệu có thể được phục hồi nếu có một đĩa trong mảng đĩa bị hư hỏng .

Tuỳ theo kỹ thuật thiết lập, RAID có thể có các mức sau:

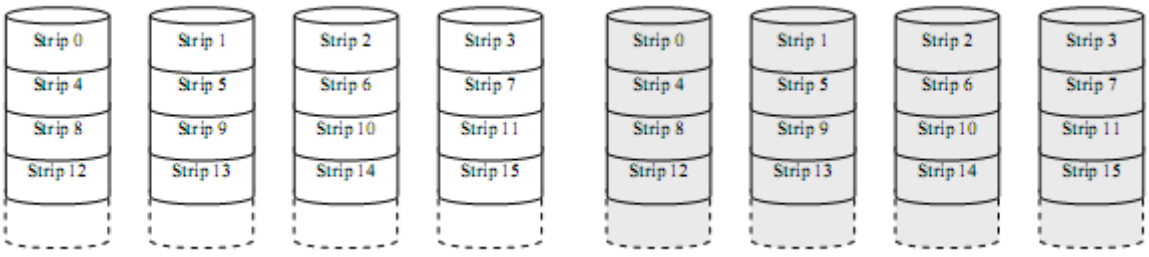
RAID 0: Thực ra, kỹ thuật này không nằm trong số các kỹ thuật có cơ chế an toàn dữ liệu. Khi mảng được thiết lập theo RAID 0, ổ đĩa logic có được (mà hệ điều hành nhận biết) có dung dượng bằng tổng dung lượng của các ổ đĩa thành viên. Điều này giúp cho người dùng có thể có một ổ đĩa logic có dung

lượng lớn hơn rất nhiều so với dung lượng thật của ổ đĩa vật lý cùng thời điểm. Dữ liệu được ghi phân tán trên tất cả các đĩa trong mảng. Đây chính là sự khác biệt so với việc ghi dữ liệu trên các đĩa riêng lẻ bình thường bởi vì thời gian đọc-ghi dữ liệu trên đĩa tỉ lệ nghịch với số đĩa có trong tập hợp (số đĩa trong tập hợp càng nhiều, thời gian đọc – ghi dữ liệu càng nhanh). Tính chất này của RAID 0 thật sự hữu ích trong các ứng dụng yêu cầu nhiều thâm nhập đĩa với dung lượng lớn, tốc độ cao (đa phương tiện, đồ hoạ,…). Tuy nhiên, như đã nói ở trên, kỹ thuật này không có cơ chế an toàn dữ liệu, nên khi có bất kỳ một hư hỏng nào trên một đĩa thành viên trong mảng cũng sẽ dẫn đến việc mất dữ liệu toàn bộ trong mảng đĩa. Xác suất hư hỏng đĩa tỉ lệ thuận với số lượng đĩa được thiết lập trong RAID 0. RIAD 0 có thể được thiết lập bằng phần cứng (RAID controller) hay phần mềm (Stripped Applications)



Hình 5-7. RAID 0

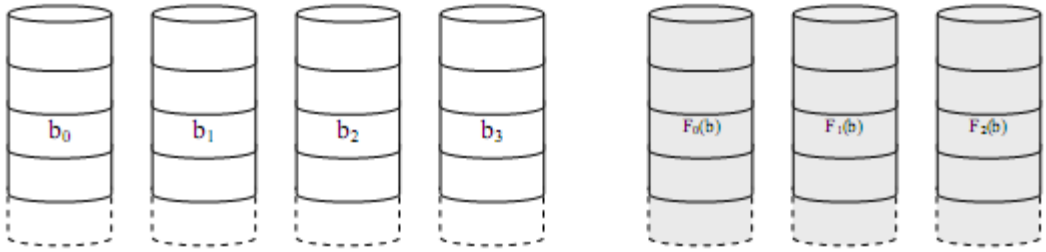
RAID 1 (Mirror - Đĩa gương): Phương cách thông thường tránh mất thông tin khi ổ đĩa bị hư là dùng đĩa gương, tức là dùng 2 đĩa. Khi thông tin được viết vào một đĩa, thì nó cũng được viết vào đĩa gương và như vậy luôn có một bản sao của thông tin. Trong cơ chế này, nếu một trong hai đĩa bị hư thì đĩa còn lại được dùng bình thường. Việc thay thế một đĩa mới (cung thông số kỹ thuật với đĩa hư hỏng) và phục hồi dữ liệu trên đĩa đơn giản. Căn cứ vào dữ liệu trên đĩa còn lại, sau một khoảng thời gian, dữ liệu sẽ được tái tạo trên đĩa mới (rebuild). RAID 1 cũng có thể được thiết lập bằng phần cứng (RAID controller) hay phần mềm (Mirror Applications) với chi phí khá lớn, hiệu suất sử dụng đĩa không cao (50%).



Hình 5-8. RAID 1

RAID 2: Dùng kỹ thuật truy cập đĩa song song, tất cả các đĩa thành viên trong RAID đều được đọc khi có một yêu cầu từ ngoại vi. Một mã sửa lỗi (ECC) được tính toán dựa vào các dữ liệu được ghi trên đĩa lưu dữ liệu, các bit

được mã hoá được lưu trong các đĩa dùng làm đĩa kiểm tra. Khi có một yêu cầu dữ liệu, tất cả các đĩa được truy cập đồng thời. Khi phát hiện có lỗi, bộ điều khiển nhận dạng và sửa lỗi ngay mà không làm giảm thời gian truy cập đĩa. Với một thao tác ghi dữ liệu lên một đĩa, tất cả các đĩa dữ liệu và đĩa sửa lỗi đều được truy cập để tiến hành thao tác ghi. Thông thường, RAID 2 dùng mã Hamming để thiết lập cơ chế mã hoá, theo đó, để mã hoá dữ liệu được ghi, người ta dùng một bit sửa lỗi và hai bit phát hiện lỗi. RAID 2 thích hợp cho hệ thống yêu cầu giảm thiểu được khả năng xảy ra nhiều đĩa hư hỏng cùng lúc.



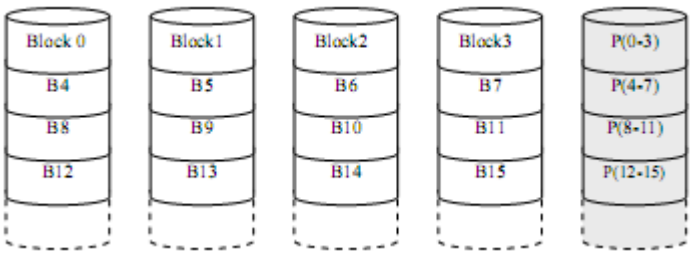
Hình 5-9. RAID 2

RAID 3: Dùng kỹ thuật ghi song song, trong kỹ thuật này, mảng được thiết lập với yêu cầu tối thiểu là 3 đĩa có các thông số kỹ thuật giống nhau, chỉ một đĩa trong mảng được dùng để lưu các thông tin kiểm tra lỗi (parity bit). Như vậy, khi thiết lập RAID 3, hệ điều hành nhận biết được một đĩa logic có dung lượng n-1/n (n: số đĩa trong mảng). Dữ liệu được chia nhỏ và ghi đồng thời trên n-1 đĩa và bit kiểm tra chẵn lẻ được ghi trên đĩa dùng làm đĩa chứa bit parity – chẵn lẻ đan chéo ở mức độ bít. Bít chẵn lẻ là một bít mà người ta thêm vào một tập hợp các bít làm cho số bít có trị số 1 (hoặc 0) là chẵn (hay lẻ). Thay vì có một bản sao hoàn chỉnh của thông tin gốc trên mỗi đĩa, người ta chỉ cần có đủ thông tin để phục hồi thông tin đã mất trong trường hợp có hỏng ổ đĩa. Khi một đĩa bất kỳ trong mảng bị hư, hệ thống vẫn hoạt động bình thường. Khi thay thế một đĩa mới vào mảng, căn cứ vào dữ liệu trên các đĩa còn lại, hệ thống tái tạo thông tin. Hiệu suất sử dụng đĩa cho cách thiết lập này là n-1/n. RAID 3 chỉ có thể được thiết lập bằng phần cứng (RAID controller).

Hình 5-10.RAID 3

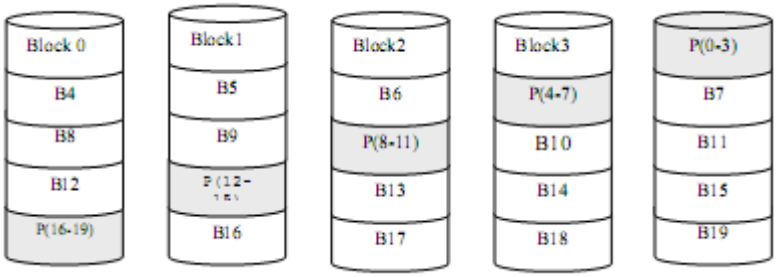
RAID 4: từ RAID 4 đến RAID 6 dùng kỹ thuật truy cập các đĩa trong mảng độc lập. Trong một mảng truy cập độc lập, mỗi đĩa thành viên được truy xuất độc lập, do đó mảng có thể đáp ứng được các yêu cầu song song của ngoại vi. Kỹ thuật này thích hợp với các ứng dụng yêu cầu nhiều ngoại vi là các ứng dụng yêu cầu tốc độ truyền dữ liệu cao. Trong RAID 4, một đĩa dùng để chứa

các bit kiểm tra được tính toán từ dữ liệu được lưu trên các đĩa dữ liệu. Khuyết điểm lớn nhất của RAID 4 là bị nghẽn cổ chai tại đĩa kiểm tra khi có nhiều yêu cầu đồng thời từ các ngoại vi.



Hình 5-11: RAID 4

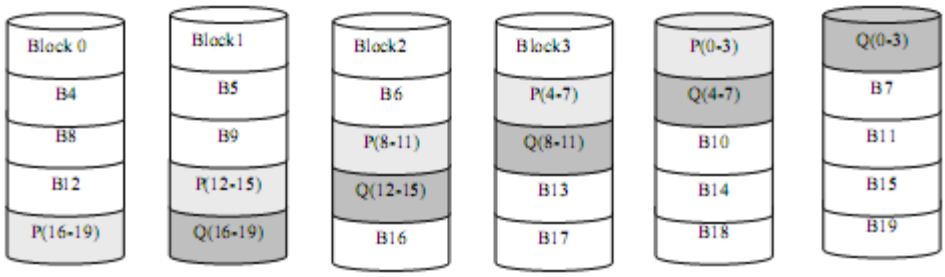
RAID 5: yêu cầu thiết lập giống như RAID 4, dữ liệu được ghi từng khối trên các đĩa thành viên, các bit chẵn lẻ được tính toán mức độ khối được ghi trải đều lên trên tất cả các ổ đĩa trong mảng. Tương tự RAID 4, khi một đĩa bất kỳ trong mảng bị hư hỏng, hệ thống vẫn hoạt động bình thường. Khi thay thế một đĩa mới vào mảng, căn cứ vào dữ liệu trên các đĩa còn lại, hệ thống tái tạo thông tin. Hiệu suất sử dụng đĩa cho cách thiết lập này là n-1/n. RAID 5 chỉ có thể được thiết lập bằng phần cứng (RAID controller). Cơ chế này khắc phục được khuyết điểm đã nêu trong cơ chế RAID 4.



Hình 5-12. RAID 5

RAID 6: Trong kỹ thuật này, cần có n+2 đĩa trong mảng. Trong đó, n đĩa dữ liệu và 2 đĩa riêng biệt để lưu các khối kiểm tra. Một trong hai đĩa kiểm tra dùng cơ chế kiểm tra như trong RAID 4&5, đĩa còn lại kiểm tra độc lập theo một giải thuật kiểm tra. Qua đó, nó có thể phục hồi được dữ liệu ngay cả khi có hai đĩa dữ liệu trong mảng bị hư hỏng.

Hiện nay, RAID 0,1,5 được dùng nhiều trong các hệ thống. Các giải pháp RAID trên đây (trừ RAID 6) chỉ đảm bảo an toàn dữ liệu khi có một đĩa trong mảng bị hư hỏng. Ngoài ra, các hư hỏng dữ liệu do phần mềm hay chủ quan của con người không được đề cập trong chương trình. Người dùng cần phải có kiến thức đầy đủ về hệ thống để các hệ thống thông tin hoạt động hiệu quả và an toàn.



Hình 5-13. RAID 6

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 5

*****

1. Mô tả vận hành của ổ đĩa cứng. Cách lưu trữ thông tin trong ổ đĩa cứng
2. Mô tả các biện pháp an toàn trong việc lưu trữ thông tin trong đĩa cứng.
3. Nguyên tắc vận hành của đĩa quang. Ưu khuyết điểm của các loại đĩa quang.
4. Thông thường có bao nhiêu loại bus? Tại sao phải có các chuẩn cho các bus vào ra?
5. Giải thích việc nới rộng dãi thông bằng cách sử dụng các gói tin.
6. Hệ thống đĩa dự phòng (RAID - Redundant Array of Independent Disks) theo kỹ thuật thiết lập có những mức nào? Trình bày từng mức đó?

Chương 6: NGÔN NGỮ ASSEMBLY

Mã chương: MH12 – 06.

Mục đích:

- Hiểu các thành phần cơ bản của Assembly
- Nắm được cấu trúc của 1 chương trình Assembly
- Hiểu cách khai báo biến, toán tử, một số hàm cơ bản và các chế độ địa chỉ
- Hiểu được cú pháp và sử dụng được các lệnh điều khiển
- Hiểu và sử dụng đượcđược ngăn xếp
- Hiểu được cách viết chương trình con và cách truyền tham số cho chương trình con.
- Tính cách suy luận chặt chẽ, có cơ sở khoa học.

1. Tổng quan

Mục đích:

- *Hiểu các thành phần cơ bản của Assembly*
- *Nắm được cấu trúc của 1 chương trình Assembly*
- *Hiểu cách khai báo biến, toán tử, một số hàm cơ bản và các chế độ địa chỉ*

1.1 Cấu trúc chung của một chương trình

Một lệnh của hợp ngữ dù đơn giản hay phức tạp đều phải có đầy đủ một trong các thành phần sau:

[Ten] [Ma_lenh] [Cac_toan_hang] [;chú thích]

Trong đó:

Ten: có thể là một chương trình con, một macro, một nhãn hoặc một thành phần nào đó.

Ma_lenh: cơ bản trường này chứa mã lệnh dưới dạng mã gợi nhớ, có thể là lệnh thật hoặc lệnh giả

nếu là lệnh thật thì đây là các lệnh của bộ vi xử lý

nếu là lệnh giả thì đây là các hướng dẫn của chương trình dịch ví dụ: PROC, ENDP, ENDM, SEGMENT, ENDS…

Cac_toan_hang: là các thành phần mà các lệnh sử dụng để thực hiện lệnh, nếu là các lệnh thật thì đây là các toán hạng của lệnh, nếu là lệnh giả thì đây là các tham số. số lượng tham số tuỳ thuộc vào việc khai báo các dẫn hướng.

Chú thích: là lời giải thích để người sử dụng hiểu rõ hơn về lệnh. Lời chú thích phải được bắt đầu bởi dấu (;) và chúng sẽ không được chương trình dịch xử lý.

Ví dụ:

Lap: ADD AL,[BX];cộng giá trị trong AL với ô nhớ có địa chỉ là DS:BX kết quả được đưa và thanh ghi AL.

Với ví dụ trên :

Lap: là tên một nhãn và kết thúc bằng dấu (:)

ADD: là lệnh thựchiện phép cộng

AL,[BX] là 2 toán hạng với quy định nếu tên thanh ghi hoặc một giá trị hằng năm trong dấu [] thì đó là địa chỉ offset của ô nhớ chứa dữ liệu cần thao tác

Các thành phần sau dấu (;) chỉ là lới chú thích.

Một chương trình hợp ngữ thể hiện các đoạn dành cho chúng rõ ràng với việc sử dụng kích thước bộ nhớ phù hợp và cú pháp của các lệnh rõ ràng tuân theo một cú pháp chung, mặc dù đó là lệnh thật hoặc lệnh giả. Để thể hiện một chương trình đó, người ta đưa ra 2 dạng cấu trúc, bao gồm:

a. Cấu trúc chương trình để dịch ra tệp *.EXE

Một chương trình EXE thường thể hiện rõ ràng 3 đoạn: mã, dữ liệu, ngăn xếp. Sau khi được biên dịch (Assembler) và liên kết (Linked), chúng có thể được thực thi trực tiếp từ dòng lệnh của DOS như các chương trình khác. Cấu trúc cơ bản của một chương trình dạng này như sau:

```
TITLE  ten_chuong_trinh
.MODEL  <Qui mô sử dụng bộ nhớ>
.STACK  <Kích thước ngăn xếp>
.DATA
        <khai báo các biến, hằng cho chương trình>
. CODE
        Main  PROC
                ;khởi tạo cho DS hoặc/và ES
                MOV AX,@data
                MOV DS,AX
                ;MOV ES,AX
                ; các lệnh của chương trình chính
                ; trở về DOS bằng việc sử dụng chức năng 4Ch của ngắt 21h
                MOV AH,4Ch
                INT 21h
        Main  ENDP
                ; các chương trình con (nếu con)
                END Main
```

Trong cấu trúc trên, tại dòng cuối cùng xuất hiện dẫn hướng chương trình END và MAIN để kết thúc toàn bộ chương trình. Main chính là nơi bắt đầu các lệnh của chương trình trong đoạn mã.

Khi một chương trình *.EXE được nạp vào bộ nhớ, DOS sẽ tạo ra một mảng gồm 256 byte cho PSP (Program Segment Prefix - tiền tố chương trình). PSP được sử dụng để chứa các thông tin liên quan đến chương trình và đặt ngay vào trước phần chứa mã lệnh của chương trình.

Tại dòng .MODEL chúng ta có thể khai báo qui mô sử dụng bộ nhớ phù hợp cho từng chương trình. Có thể là một trong các thành phần sau:

Kiểu kích thước	Mô tả
Tiny (hẹp)	Mã lệnh và dữ liệu gói gọn trong một đoạn
Small (nhỏ)	Mã lệnh gói gọn trong một đoạn, dữ liệu nằm trong một đoạn
Medium (trung bình)	Mã lệnh không gói gọn trong một đoạn, dữ liệu nằm trong một đoạn
Compact (gọn)	Mã lệnh gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn
Larg (rộng)	Mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, không có mảng nào lớn hơn 64Kb.
Huge (đồ sộ)	Mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, mảng có thể lớn hơn 64Kb.

Chương trình dịch sẽ dịch tên @data thành các giá trị số của đoạn dữ liệu và đưa các thông số của dữ liệu vào thanh ghi DS và ES. Để hiểu rõ hơn về cấu trúc chương trình này , chúng ta lấy một ví dụ minh hoạ sau:

Ví dụ: Viết một chương trình hợp ngữ thực hiện in hai chuỗi kí tự trên 2 dòng màn hình

```
Title chuongtrinh
.MODEL  Small
.STACK   100h
.DATA
        Chao DB ‘chao cac ban sinh vien!$’
        Hoi    DB ‘cac ban muon hoc mon kien truc may tinh khong?$’
. CODE
        Main  PROC
                ;khởi tạo cho DS
                MOV AX,@data
                MOV DS,AX
                ; xoá màn hình bằng chức năng 0 của ngắt 10h
                MOV AH,0
        MOV AL,3  ;chế độ text
        INT 10h
        ; in chuỗi thứ nhất ra màn hình
                MOV AH,9
        MOV DX,offset Chao
        INT 21h
        ; đưa con trỏ xuống dòng tiếp theo và về đầu dòng
```

```
        MOV AH,2
    MOV DL,13       ; xuống dòng
    INT 21h
MOV DL,10       ; về đầu dòng
    INT 21h
    ; in chuỗi thứ hai ra màn hình
        MOV AH,9
    MOV DX,offset  Hoi
    INT 21h
    ; kết thúc chương trình về DOS
        MOV AH,4Ch
    INT 21h
Main ENDP
    END main
```

Như vậy, ta thấy chương trình trên là một chương trình nhỏ nên sử dụng quy mô bộ nhớ nhỏ. Kích thước ngăn xếp là 256 byte. Dữ liệu cho chương trình có 2 chuỗi là Chao và Hoi, các chuỗi này đều được khai báo với toán tử DB.Đặc biệt, các lệnh hoặc nhóm lệnh của chương trình trên có giải thích để người đọc và bản thân người lập trình (khi đọc lại) cũng sẽ dễ hiểu hơn.

b. Cấu trúc chương trình để dịch ra tệp *.COM

Một chương trình *.COM thường có đặc điểm khác biệt với các chương trình *.EXE là chúng chỉ có thể sử dụng một đoạn duy nhất của bộ nhớ để chứa mã, dữ liệu và ngăn xếp. Vì vậy, quy mô sử dụng bộ nhớ của các chương trình dạng này thường là Tiny

Cấu trúc cơ bản của dạng chương trình này như sau:

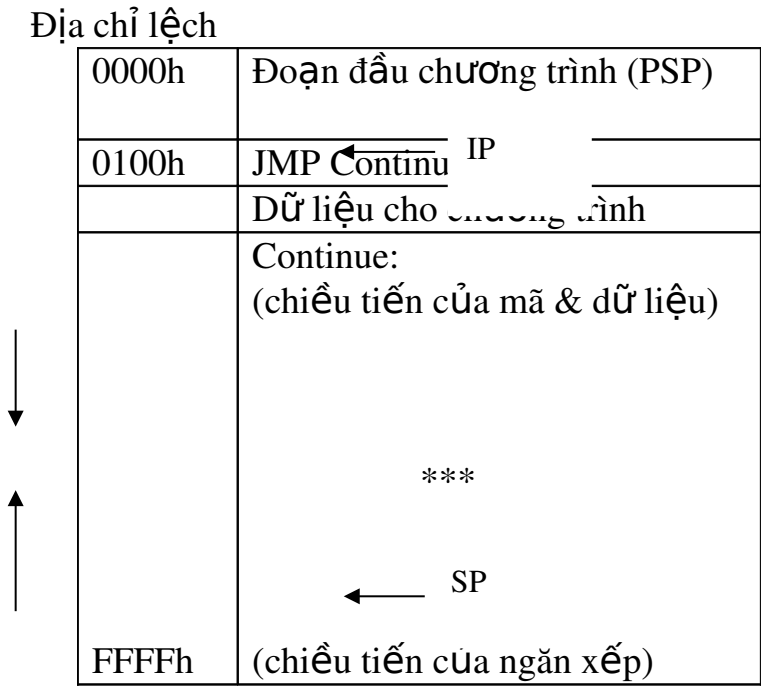
```
Title cautruc_COM
. Model Tiny
.Code
Org 100h
Start:JMP Continue
; định nghĩa các biến, hằng.
Continue:
    Main PROC
        ;các lệnh của chương trình chính
    INT 21h
    Main ENDP
        ;các chương  trình con (nếu có)
END Start
```

Với dạng cấu trúc này, khi dịch chương trình, chương trình dịch sẽ đưa ra thông báo "No Stack Segment".

Trong cấu trúc trên, ta thấy, ở ngay đầu chương trình đoạn mã là lệnh giả ORG (Origin: điểm xuất phát) và lệnh nhảy JMP (nhảy). Lệnh giả ORG 100h

dùng để gán địa chỉ bắt đầu cho chương trình tại địa chỉ 100h (256) trong đoạn mã, chừa lại đoạn bộ nhớ 256 byte cho PSP. Lệnh JMP sau nhãn Start dùng để nhảy qua phần bộ nhớ dành cho việc định nghĩa và khai báo dữ liệu. Đích của lệnh nhảy là phần đầu của chương trình chính.

Hình ảnh của chương trình được thể hiện trong bộ nhớ như trong hình vẽ sau



Như vậy, một chương trình *.COM có một số hạn chế sau:

- Dung lượng cực đại của chương trình chỉ giới hạn trong một đoạn 64K.
- Chương trình chỉ được phép sử dụng ngăn xếp với kích thước nhỏ. Nếu không, nó sẽ chiếm nhiều phần trong đoạn mã lệnh.

Chương *.COM thường được kết thúc bởi lệnh INT 20h. Sau đây chúng ta sẽ xét một ví dụ để hiểu rõ hơn về việc sử dụng cấu trúc trên trong lập trình hợp ngữ.

Chương trình được viết như sau:

```
Title cautruc_COM
        . Model Tiny
        .Code
        Org 100h
        Start:JMP Continue
        Chao  DB ‘chao cac ban sinh vien$’
Hoi     DB ‘cac ban co muon hoc mon kien truc may tinh khong?$’
        ; định nghĩa các biến, hằng.
        Continue:
                Main PROC
                ;xóa màn hình bằng chức năng 0 của ngắt 10h
                        MOV AH,0
```

```
                MOV AL,3
                INT 10h
        ; in chuỗi thứ nhất ra màn hình
                MOV AH,9
                MOV DX, offset chao ;trỏ tới địa chỉ offset của chuỗi
                INT 21h
; đưa con trỏ xuống đầu dòng tiếp theo và về đầu dòng
                MOV AH,2
                MOV DL, 13 ;xuống dòng
                INT 21h
                MOV DL, 10 ;về đầu dòng
                INT 21h
; in chuỗi thứ hai ra màn hình
                MOV AH,9
                MOV DX, offset hoi ;trỏ tới địa chỉ offset của chuỗi
                INT 21h
        Main ENDP
                ;các chương  trình con (nếu có)
    END Start
```

Chúng ta có thể nhận thấy rằng một chương trình *.COM không cần phải nạp dữ liệu vào DS vì chương trình dạng này có mã, dữ liệu và ngăn xếp trong cùng một đoạn.

c. Cách tạo vào chạy một chương trình hợp ngữ

Môt chương trình viết bằng ngôn ngữ bậc cao có thể chạy trực tiếp trong môi trường của chúng. Song với ngôn ngữ lập trình bậc thấp như ASSEMBLY thì việc chạy chương trình phải thông qua việc hợp dịch, liên kết.

Các bước để soạn thảo, dịch và chạy chương trình như sau:

Bước1: Soạn thảo văn bản chương trình (viết các lệnh của chương trình)

Trong bước này có thể thực hiện bằng một trình soạn thảo bất kỳ nào như

Notepad, word, pascal,c nhưng cần phải ghi lại với phần mở rộng là *.ASM.

Bước2: Hợp dịch chương trình nguồn ra các tệp đối tượng *.OBJ. Trong bước này, ta có thể sử dụng trình MASM hoặc TASM theo cú pháp sau:

MASM tentep [.phan_mo_rong]

Hoặc TASM tentep [.phan_mo_rong]

Sau khi dịch nếu có lỗi cú pháp, máy sẽ báo dòng gây lỗi và mã lỗi. khi đó ta có thể quay lai bước1 để sửa tệp nguồn và dịch lại cho đến khi không có lỗi.

Bước 3: Liên kết các tệp đối tượng thành tệp chương trình *.EXE

Có thể sử dụng trình liên kết link hoặc Tlink theo cú pháp sau:

LINK tentep

Hoặc Tlink tentep

Bước 4: Nếu chương trình viết dưới dạng cấu trúc *.COM thì thực hiện dịch sang dạng *.COM bằng cú pháp sau:

EXE2BIN tentep_exe tentep_com.COM

Nếu tệp viết để dịch ra chương trình *.EXE thì bỏ qua bước này

Bước 5: Chạy tệp chương trình vừa dịch

Soạn thảo văn bản chương trình (viết các lệnh của chương trình) (có thể soạn và chạy trực tiếp trên EMU8086)

1.2 Biến và khai báo biến

Khi khai báo dữ liệu cho chương trình, người ta thường sử dụng toán tử DB. Nhưng trên thực tế, các toán tử khác cũng có thể được dùng đến để khai báo các dữ liệu có kiểu khác nhau.

DB – Define Byte : định nghĩa biến kiểu Byte
DW – Define Word : định nghĩa biến kiểu Word
DD – Define Double Word : định nghĩa biến kiểu Double Word
DT – Define Ten Byte : định nghĩa biến kiểu 10 Byte
EQU– Equal (bằng) : Khai báo một hằng, địa chỉ cổng xác định

a. Biến kiểu Byte

Biến kiểu byte sẽ chiếm 1 byte trong bộ nhớ. Hướng dẫn chương trình dịch để định nghĩa biến kiểu byte có dạng tổng quát như sau:

Ten_bien DB Gia_tri_khoi_tao
Ví dụ: so1 DB 3Eh

Trong ví dụ trên so1 là một biến kiểu byte có giá trị khởi tạo là 3Eh, song đôi khi chúng ta không cần thiết phải khởi tạo giá trị cho biến, có thể sử dụng cách khai báo sau:

so1 DB ?

Cũng như các ngôn ngữ lập trình khác, một biến kí tự cũng được coi là một biến kiểu byte với mỗi kí tự được thể hiện bằng mã ASCII của chúng từ 0 – FFh

Giả sử muốn khai báo một biến kí tự và khởi tạo giá trị bằng "A", ta có thể thực hiện một trong 2 cách khai báo sau:

Ch DB 'A' ;khởi tạo trực tiếp giá trị 'A'
hoặc Ch DB 41h ;khởi tạo thông qua mã ASCII của 'A'

b. Biến kiểu Word

Để khai báo một biến kiểu Word, ta có thể sử dụng cú pháp sau:

Ten_bien DW Gia_tri_khoi_tao

Cũng giống như việc khai báo biến kiểu byte, Gia_tri_khoi_tao có thể là dấu hỏi (?)

Ví dụ: w1 DB 3FB4h ;khai báo biến w1 với giá trị khởi tạo là 3FB4h
w2 DB ? ;khai báo biến w2 nhưng không khởi tạo giá trị

c. Biến kiểu mảng

Mảng là một danh sách (dãy) các phần tử có cùng một kiểu với các giá trị (có thể) khác nhau. Vì vậy, để khai báo một mảng ta có thể thực hiện theo cú pháp sau:

Ten_bien DB Danh_sach_gia_tri_khoi_tao;khai báo mảng các phần tử kiểu byte

Ten_bien DW Danh_sach_gia_tri_khoi_tao;khai báo mảng các phần tử kiểu Word

Ví dụ1:

mang1 DB 30,55,73,88,83,90

Giả thiết, mang1 được nạp vào bộ nhớ tại địa chỉ offset 23E5h. Khi đó mảng này sẽ có các phần tử được khởi tạo và hình ảnh của chúng trong bộ nhớ như sau:

Tên phần tử	Giá trị khởi tạo	Địa chỉ bộ nhớ
Mang10	30	23E5h
Mang11	55	23E6h
Mang12	73	23E7h
Mang13	88	23E8h
Mang14	83	23E9h
Mang15	90	23EAh

Ví dụ 2:

mang2 DW 2BA3h,2748h,9843h,1F3Bh

Cũng với giả thiết, mang2 được nạp vào bộ nhớ tại địa chỉ offset 23E5h. Khi đó mảng này sẽ có các phần tử được khởi tạo và hình ảnh của chúng trong bộ nhớ sẽ là:

Tên phần tử	Giá trị khởi tạo	Địa chỉ bộ nhớ
Mang20	2BA3h	23E5h
Mang22	2748h	23E7h
Mang24	9843h	23E9h
Mang26	1F3Bh	23EBh

Chúng ta có thể khởi tạo các giá trị liên tiếp giống nhau. Khi đó, ta cũng có thể sử dụng toán tử DUP.

Ví dụ 3:

mang3 DB 100 DUP(5Ch) ;mang3 là một mảng có 100 phần tử với các giá trị khởi tạo của các phần tử đều là 5Ch

Ví dụ 4:

mang4 DB 3,4,2,10 DUP(0) ;mang4 là một mảng có 13 phần tử, trong đó 3 phần tử đầu lần lượt có giá trị khởi tạo là 3,4,2 các phần tử còn lại (10phần tử) đều có giá trị khởi tạo bằng 0.

Với mảng 2 chiều việc định nghĩa chúng có thể thực hiện theo cú pháp sau:

```
Ten_bien  DB (DW)  gia_tri_khoi_tao_hang_0
                   gia_tri_khoi_tao_hang_1
                   gia_tri_khoi_tao_hang_2
                   ……
                   gia_tri_khoi_tao_hang_n
```

Ví dụ:

```
Mang_2_chieu   DB   10 DUP(0)
                    10 DUP(50)
                    10 DUP(100)
                    10 DUP(150)
```

Khai báo một mảng 2 chiều có 4 hàng 10 cột với giá trị khởi tạo của hàng 0 là 0, hàng 1 là 50, hàng 2 là 100, hàng 3 là 150

d. Biến chuỗi

Biến chuỗi hay xâu kí tự thực chất chỉ là một mảng 1 chiều các phần tử là các kí tự trong đó mỗi kí tự có thể được biểu diễn bằng một số là mã ASCII hoặc là kí tự nằm trong dấu "" hoặc ''. Vì vậy ta có thể khai báo bằng một trong các cách sau:

```
Xau1 DB  'chao'
Xau2 DB 'c', 'h', 'a', 'o'
Xau3 DB  43h,68h,61h,6Fh
```

e. khai báo hằng

```
ten bien   EQU    giatri    (EQU: Equal)
```

Ví dụ:

```
S01  EQU  10        ;biến S01 sẽ có giá trị là 10 và không thể thay đổi được giá trị
của  S01
```

1.3 Các chế độ địa chỉ

Các kiểu định vị chính là các chế độ địa chỉ nó là phương pháp để xác định toán hạng hoặc kiểu toán hạng trong các câu lệnh. Bộ vi xử lý (8086/8088) có tổng số trên 20 chế độ địa chỉ cho các thành phần khác nhau mã lệnh, dữ liệu, ngăn xếp. Nhưng trên thực tế, việc lập trình, phân tích lệnh, người ta chỉ quan tâm đến việc dữ liệu của lệnh được xử lý ra sao. Vì vậy, chúng ta cũng chỉ nghiên cứu về các chế độ địa chỉ dữ liệu của lệnh.

Trong bộ vi xử lý 8086 quy định có 7 chế độ địa chỉ cho toán hạng của lệnh. Cụ thể bao gồm các chế độ sau:

- Chế độ địa chỉ thanh ghi
- Chế độ địa chỉ tức thì
- Chế độ địa chỉ trực tiếp
- Chế độ địa chỉ gián tiếp thanh ghi
- Chế độ địa chỉ tương đối cơ sở

- Chế độ địa chỉ tương đối chỉ số
- Chế độ địa chỉ tương đối chỉ số cơ sở

Trong bộ xử lý RISC, các lệnh số học và logic chỉ được thực hiện theo kiểu thanh ghi và tức thì, còn những lệnh đọc và ghi vào bộ nhớ là những lệnh có toán hạng bộ nhớ thì được thực hiện với những kiểu định vị (chế độ địa chỉ) khác.

a. Chế độ địa chỉ thanh ghi

Trong chế độ này việc trao đổi thông tin diễn ra trực tiếp giữa các thanh ghi. Toán tử chỉ hoàn toàn là các thanh ghi. Loại địa chỉ thanh ghi không cần truy nhập bộ nhớ nên rất nhanh.

Ví dụ: MOV AX, BX ; sao chép nội dung thanh ghi BX sang thanh ghi AX

MOV AL,CL ; sao chép nội dung thanh ghi CL sang thanh ghi AL

ADD AL,CL ; cộng nội dung thanh ghi CL với thanh ghi AL

b. Chế độ địa chỉ tức thì

Trong chế độ này toán hạng đích là một thanh ghi hoặc ô nhớ, còn toán hạng nguồn là một hằng số. Ta có thể dùng chế độ này để nạp dữ liệu vào bất kỳ thanh ghi nào, trừ thanh ghi đoạn và thanh ghi cờ.

Ví dụ: MOV CL,5Fh ;chuyển 5Fh vào thanh ghi CL

MOV AX,0FFh ;chuyển 0FFh vào thanh ghi AX

MOV DS,10 ;chuyển 10 vào ô nhớ tại địa chỉ DS:BX

c. Chế độ trực tiếp

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

Ví dụ: MOV AL,[1053] ;chuyển nội dung ô nhớ địa chỉ DS:1053h vào AL

MOV [5307h],CX ;chuyển nội dung CX vào 2 ô nhớ liên tiếp có địa chỉ ;DS:5307h và DS:5308h

ADD [5607h],AL ;Cộng nội dung AL vào ô nhớ có địa chỉ DS: 5607h

d. Chế độ địa chỉ gián tiếp thanh ghi

Trong chế độ này, một toán hạng là một thanh ghi được sử dụng để chứa địa chỉ lệch (Offset) của ô nhớ chứa dữ liệu, còn toán hạng kia chỉ có thể là thanh ghi mà không được là ô nhớ.

Ví dụ: MOV AL,[BX] ;chuyển nội dung ô nhớ có địa chỉ DS:BX vào AL

MOV [SI],CL ;chuyển nội dung CL vào ô nhớ có địa chỉ DS:SI

MOV [DI],AX ;chuyển nội dung AX vào 2 ô nhớ liên tiếp có địa chỉ

;DS:DI và DS:DI+1

e. Chế độ tương đối cơ sở

Trong chế độ này thanh ghi cơ sở BX và BP là các hằng số biểu diễn các giá trị dịch chuyển (Displacement Values), kết hợp với DS và SS để tính địa chỉ hiệu dụng của toán hạng các vùng nhớ. Sự có mặt của các giá trị dịch chuyển xác định tính tương đối (so với cơ sở) của địa chỉ.

Ví dụ: MOV CX,[BX+10] ;chuyển nội dung 2 ô nhớ liên tiếp có địa chỉ
;DS:(BX +10) vàDS:(BX +11) vào CX

hoặc MOV CX,[BX]+10 ;

MOV CL,[BP]+5 ;chuyển nội dung ô nhớ có địa chỉ DS: (BP+5) vào CL

Chú ý: Trong các ví dụ trên các giá trị 10,5 được gọi là các giá trị dịch chuyển

- (BX+10), (BP+5) được gọi là các địa chỉ hiệu dụng
- DS:(BX +10) và DS: (BP+5) là địa chỉ logic

f. Chế độ địa chỉ tương đối chỉ số

Trong chế độ này, các thanh ghi chỉ số DI và SI và các hằng số biểu diễn giá trị dịch chuyển được dùng để tính địa chỉ của toán hạng trong vùng nhớ DS.

Ví dụ: MOV AX,[SI]+10 ;chuyển nội dung 2 ô nhớ liên tiếp có địa chỉ
;DS:(SI +10) vàDS:(SI+11) vào AX

hoặc MOV AX,[SI]+10 ;

MOV AL,[DI]+5 ;chuyển nội dung ô nhớ có địa chỉ DS: (DI+5) vào AL

g. Chế độ địa chỉ tương đối chỉ số cơ sở

Là sự kết hợp của 2 chế độ địa chỉ, đó là chế độ địa chỉ tương đối chỉ số và chế độ địa chỉ tương đối cơ sở. Trong chế độ này dùng cả 2 thanh ghi cơ sở và 2 thanh ghi chỉ số để tính địa chỉ của toán hạng.

Ví dụ: MOV AX,[BX] +[SI]+8 ;chuyển nội dung 2 ô nhớ liên tiếp có địa chỉ
;DS:(BX+SI +8) vàDS:(BX+SI+9) vào AX

hoặc MOV AX,[BX]+[SI]+8

hoặc MOV AX, [BX+SI+8]

MOV CL,[BP]+[DI]+5 ;chuyển nội dung ô nhớ có địa chỉ DS: ;(BP+DI+5)vào CL

Chú ý:

Trong các chế độ địa chỉ trên, các thanh ghi đoạn và các thanh ghi lệch được ngầm định đi kèm với nhau. Muốn loại bỏ sự ngầm định đó ta có thể viết tường minh địa chỉ của đoạn.

Ví dụ: MOV AL,[BX] ;ngầm định là DS:BX

Muốn bỏ ngầm định là DS ta phải viết

MOV AL,SS:[BX] ; chuyển thành SS:BX

Bảng tóm tắt các chế độ địa chỉ

STT	Chế độ địa chỉ	Toán hạng	Đoạn ngầm định
1	Thanh ghi	Reg	

2	Tức thì	Data	
3	Trực tiếp	[Offset]	
4	Gián tiếp thanh ghi	[BX] [SI] [DI]	
5	Tương đối cơ sở	[BX] + Disp [BP] + Disp	
6	Tương đối chỉ số	[DI] + Disp [SI] + Disp	
7	Tương đối chỉ số cơ sở	[BX] + [DI] + Disp [BX] + [SI] + Disp [BP] + [DI] + Disp	

2. Các Lệnh cơ bản

Mục đích:

- *Hiểu được cú pháp và sử dụng được các lệnh cơ bản*

2.1 Các lệnh tính toán

a. ADD:Addition (cộng 2 toán hạng)

Cú pháp: ADD đích, nguồn

Tác dụng: Cộng toán hạng đích với toán hạng nguồn. Kết quả được chứa trong toán hạng đích

Đích = đích + nguồn

Điều kiện: hai toán hạng phải cùng độ dài, không được là 2 thanh ghi đoạn.

ADD AX,word1

AX = AX + word1

b. SUB: Subtraction (trừ)

Cú pháp: SUB đích, nguồn

Tác dụng: Trừ nội dung của toán hạng đích cho toán hạng nguồn, kết quả chứa trong toán hạng đích.

Ví dụ:

MOV BX,F0h

SUB BX,50h

BX = BX -50h = A0h

c. MUL: Multiplexing (nhân không dấu)

Nhân toán hạng với nội dung chứa trong thanh ghi AX. Tức là nhân 2 toán hạng với nhau nhưng 1 toán hạng phải được chứa trong AX. Hoặc là trong DX và AX

MUL gốc

Tùy thuộc vào độ dài của toán hạng gốc mà xác định kết quả:

Gốc: 8 bit thì số bị nhân trong AL kết quả chứa trong AX

Gốc: 16 bit thì số bị nhân trong AX kết quả chứa trong DX:AX

Ví dụ

MOV AL,10h
MOV BL,5h
MUL BL

Vì toán hạng nguồn là thanh ghi BL, nên kết quả sẽ được lấy ra trong AX. AX=50h. Trong trường hợp muôn nhân số có dấu, ta có thể sử dụng lệnh IMUL có dạng lệnh như lệnh MUL

d. DIV: Unsigned Divide (chia hai số không có dấu)

Cú pháp

DIV nguồn

Nguồn là số 8 bit: AX/nguồn số bị chia phải là số không dấu 16 bit trong AX sau khi chia thương chứa trong AL còn số dư chứa trong AH.

Nguồn là số 16 bit:DX: AX/nguồn số bị chia phải là số không dấu đặt trong cặp DX:AX sau khi chia thương chứa trong AX còn số dư chứa trong DX.

Nguồn =0 (chia cho 0) hoặc kết quả lớn hơn FFh, FFFFh thì gọi ngắt INT 0. Trong trường hợp muốn chia số có dấu, ta có thể sử dụng lệnh IDIV có dạng lệnh như lệnh DIV

2.2 Lệnh nhập và xuất.

a. Lệnh IN: nhập vào từ cổng 1 byte hay 1 word

Cú pháp

IN thanhchứa, cổng

Nếu thanh chứa là AL thì dữ liệu 8 bit được đưa vào có giá trị là địa chỉ cổng.

Nếu thanh chứa là AX thì dữ liệu 16 bit được đưa vào từ cổng có giá trị là địa chỉ cổng +1

Địa chỉ cổng trong khoảng 00h – FFh

b. Lệnh OUT: xuất ra cổng 1 byte hoặc 1 word

Cú pháp

OUT địa_chỉ_cổng, Acc

3. Các lệnh điều khiển

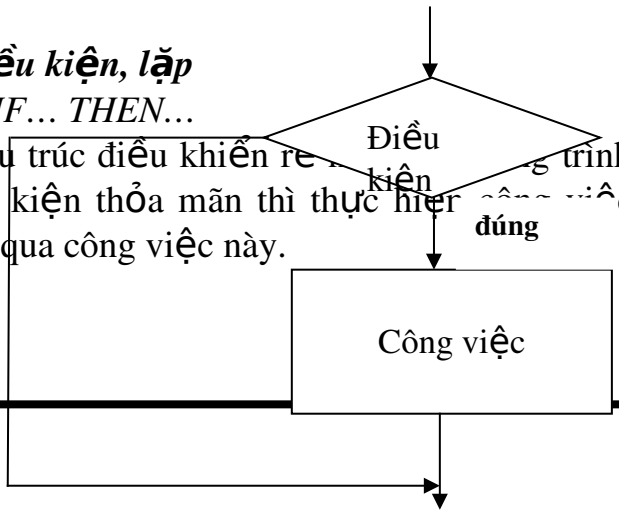
Mục đích:

- *Hiểu được cú pháp và sử dụng được các lệnh điều khiển*

3.1 Các lệnh điều kiện, lặp

a. Cấu trúc IF… THEN…

Đây là cấu trúc điều khiển rẽ nhánh chương trình với việc kiểm tra điều kiện. Nếu điều kiện thỏa mãn thì thực hiện công việc. Nếu điều kiện không thỏa mãn thì bỏ qua công việc này.



Cú pháp:

IF điều_kiện THEN Công_việc

Ví dụ: Viết một đoạn chương trình thực hiện nhập vào một kí tự. Nếu là kí tự 'A' thì hiển thị tại dòng tiếp theo.

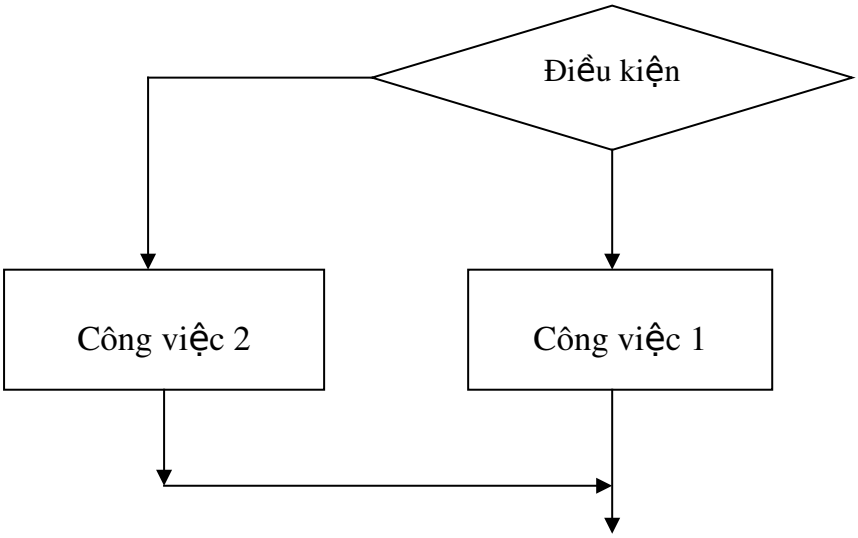
Giải:

```
MOV AH,1 ;nhập vào từ bàn phím một kí tự, kí tự đó nằm trong AL
INT 21h
CMP  AL,41h
JNE  kethuc
PUSH  AX ;cất tạm kí tự này vào ngăn xếp
MOV Ah,2  ;đưa con trỏ
MOV DL,13         ;xuống dòng tiếp theo
INT 21h
MOV DL,10 ;về đầu dòng
INT 21h
POP  DX     ; lấy  kí tự ra và đưa trực tiếp vào DL
INT 21h      ;in ra màn hình
Ketthuc:
MOV AH,4Ch
INT 21h
```

Đoạn chương trình trên sẽ thực hiện cho phép người sử dụng nhập vào từ bàn phím một kí tự. Sau đó, so sánh với mã ASCII của 'A'. Nếu không bằng thì sẽ thực hiện các lệnh sau lệnh "JNE ketthuc" rồi mới kết thúc. Nếu không, máy sẽ bỏ qua các lệnh đó và kết thúc chương trình bằng hàm ngắt 4Ch của ngắt 21h.

b. Cấu trúc IF …THEN …ELSE

Trong thực tế, chúng ta thường đưa ra một điều kiện nào đó. Nếu trường hợp thỏa mãn thì sẽ thực hiện một công việc nào đó. Ngược lại, sẽ không thực hiện một công việc khác.



Cú pháp:

```
IF điều_kiện THEN
        Công_việc_1
ELSE
        Công_việc_2
END_IF
```

Ví dụ: Viết 1 đoạn chương trình hợp ngữ thực hiện nhập vào từ bàn phím một kí tự. Nếu kí tự nhập vào có mã ASCII nhỏ hơn mã ASCII của số 1 thì đưa ra màn hình thông báo "kí tự này đứng trước '1' trong bảng mã", ngược lại, đưa ra màn hình thông báo "kí tự này đứng sau '1' trong bảng mã"

Giải: đoạn chương trình được thể hiện như sau:

*Giả thiết: thông báo "kí tự này đứng trước '1' trong bảng mã" được lưu trong biến *truoc*, thông báo báo "kí tự này đứng sau '1' trong bảng mã" được lưu trong biến *sau*.

Đoạn chương trình được viết như sau:

```
        ; nhập kí tự từ bàn phím
        Mov  al,1
        Int  21h
        ; bắt đầu cấu trúc
        CMP al, '1' ;so sánh kí tự nhập vào với '1'
        JL then
        Mov ah,9
        Lea dx,sau
        Int 21h
        Jmp end_if
Then:
        Mov ah,9
        Lea dx, truoc
```

Int 21h
End_if

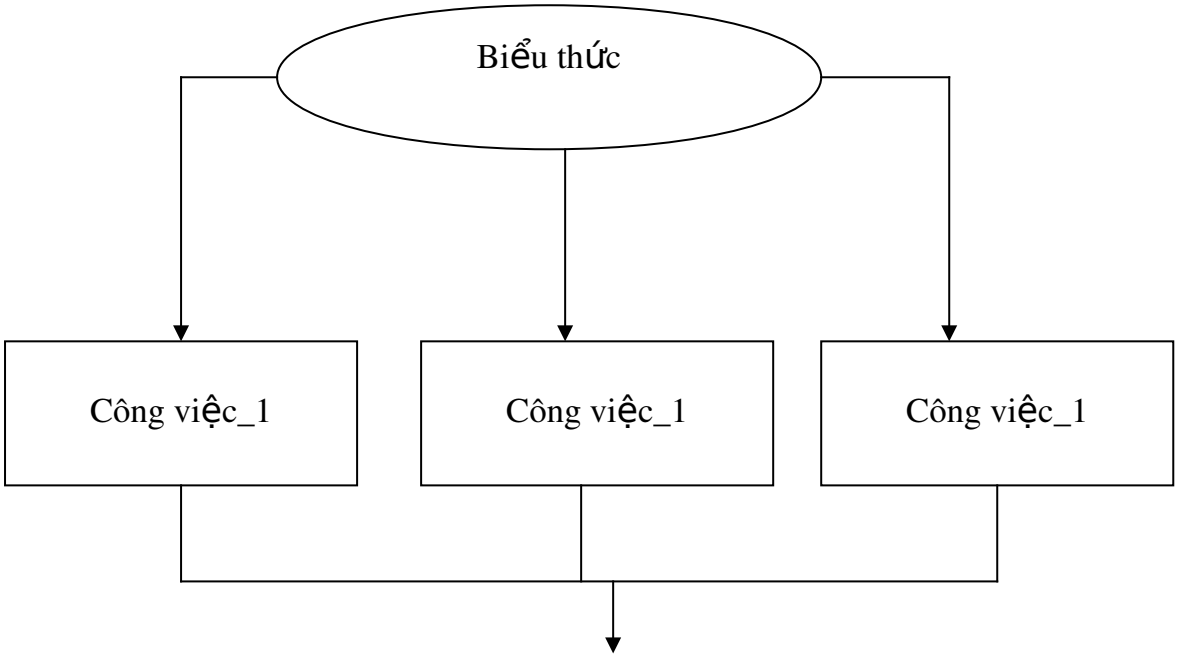
c. *Cấu trúc rẽ nhánh Case…of*

Là một cấu trúc đa nhánh. Nó kiểm tra các thanh ghi, các biến hay các giá trị riêng rẽ trong miền giá trị.

Cú pháp:

```
Case  biểu_thức
        Giá_trị1: công_việc_1
        Giá_trị2 :công_việc_2
        ……
        Giá_trịn :công_việc_n
End case
```

Như vậy,nếu biểu thức bằng giá trị nào, thì công việc tương ứng sẽ được thực hiện.



Như vậy, nếu biểu thức bằng giá trị nào, thì công việc tương ứng sẽ được thực hiện.

Ví dụ: Viết một đoạn chương trình thực hiện nhập vào một kí tự. Nếu kí tự đứng trước ‘A’ trong bảng mã ASCII thì đưa ra thông báo kí tự đứng trước ‘A’. Nếu kí tự nhập vào là ‘A’ thì đưa ra thông báo chính là kí tự ‘A’. Nếu đứng sau ‘A’ thì đưa ra thông báo kí tự đứng sau ‘A’.

Giải: Giải thiết các biến truoc, dung, sau chứa nội dung là các chuỗi thông báo cần đưa ra. Ta có thể thực hiện đoạn chương trình như sau:

```
;nhập vào kì tự
Mov ah,1
Int 21h
;Case …of
CMP AL,’A’
JL L1
JE L2
JG L3
L1:
Mov ah,9
Lea dx, truoc
Int 21h
Jmp end_
```

```
        L2:
Mov ah,9
        Lea dx, dung
        Int 21h
        Jmp end_
        L3:
Mov ah,9
        Lea dx, sau
        Int 21h
        end_:
```

d. Cấu trúc rẽ nhánh với điều kiện kép

Là một dạng cấu trúc rẽ nhánh mà trong đó, điều kiện là một sự kết hợp của hai hay nhiều điều kiện khác nhau. Điều kiện kép có dạng.

* Điều kiện kết hợp AND

Là một dạng cấu trúc rẽ nhánh mà trong đó có nhiều điều kiện kết hợp. Tất cả các điều kiện đều thỏa mãn thì công việc sẽ được thực hiện. Ngược lại, có thể thực hiện một công việc khác hoặc không thực hiện gì.

Ví dụ: Viết một đoạn chương trình thực hiện nhập vào từ bàn phím một kí tự. kiểm tra xem phím nhập vào có phải là số không.

Giải:

```
        Mov ah,1
        Int 21h
        Cmp AL,'0'
        JL END_IF
        Cmp AL,'9'
        JG END_IF
        ;then
        Mov AH,9
        Lea DX,So
        Int 21h
        END_IF:
        MOV Ah,4Ch
        INT 21h
```

* Điều kiện kết hợp OR

Là một dạng cấu trúc rẽ nhánh mà trong đó có nhiều điều kiện kết hợp. Tất cả các điều kiện đều thỏa mãn thì công việc sẽ được thực hiện. Ngược lại, nếu tất cả các điều kiện không được thỏa mãn thì có thể thực hiện một công việc khác hoặc không thực hiện gì.

Ví dụ: Viết một đoạn chương trình thực hiện nhập vào từ bàn phím một kí tự. kiểm tra xem phím nhập vào có phải là 'Y' hoặc 'y' không.

Giải: Giả thiết có 2 biến. yeucau chứa thông báo nhập vào; dung chứa thông báo nhập kí tự nhập vào đúng là 'Y' hoặc 'y'. Đoạn chương trình sẽ được viết ra như sau:

```
Mov ah,9
Lea DX,yeucau
Int 21h
Mov ah,1
Int 21h
Cmp AL,'Y'
JE THEN
Cmp AL,'y'
JE THEN
Jmp END_IF
THEN:
Mov AH,9
Lea DX,dung
Int 21h
END_IF:
MOV AH,4Ch
INT 21h
```

e. Cấu trúc for…to…do

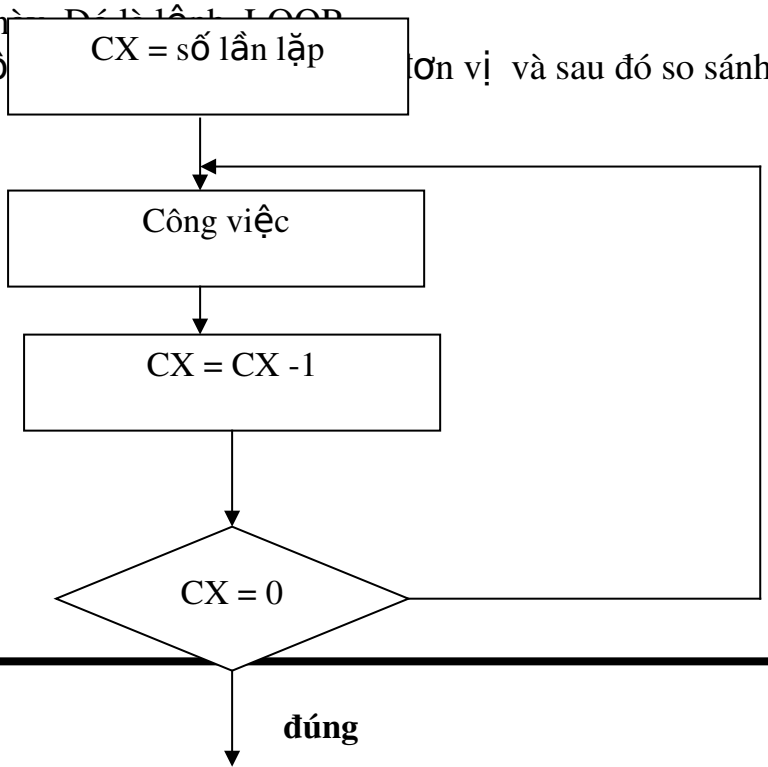
Đây là một dạng cấu trúc lặp với số lần lặp đã được xác định.

Cú pháp:

FOR số_lần_lặp DO công_việc

Đầu tiên, gán cho thanh ghi đếm CX một giá trị bằng số lần lặp. Sau đó, thực hiện công việc. Sau mỗi lần lặp, giảm giá trị trong thanh ghi CX đi 1 đơn vị và kiểm tra nó với 0. Nếu chưa bằng 0 thì tiếp tục thực hiện công việc …cho tới khi CX=0

Trong tập lệnh của bộ vi xử lý 8086 tồn tại một lệnh sử dụng phù hợp trong cấu trúc n
Lệnh này tự độ tơn vị và sau đó so sánh CX với 0.



sai

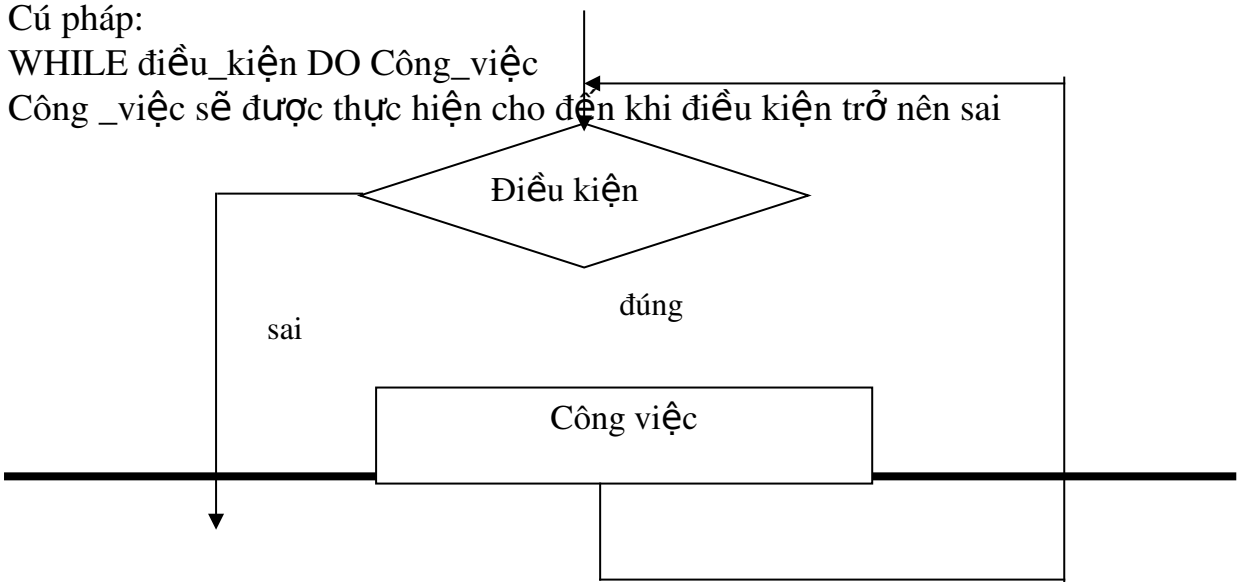
Ví du: Viết một đoạn chương trình thực hiện nhập vào từ bàn phím một kí tự. Sau đó cho hiển thị nó 200 lần trên màn hình.
Giải:

```
MOV AH,9
LEA DX, yeucau
INT 21h
MOV CX,200
MOV AH,1
INT 21h
PUSH AX
MOV AH,2
MOV DL,13
INT 21h
MOV DL,10
INT 21h
POP AX
MOV         AH,2
FOR:
MOV DL,AL
INT 21h
LOOP FOR
```

f. Cấu trúc while…do…

Là một cấu trúc lặp phụ thuộc vào một điều kiện

Cú pháp:
WHILE điều_kiện DO Công_việc
Công _việc sẽ được thực hiện cho đến khi điều kiện trở nên sai



Ví dụ: Viết một đoạn chương trình thực hiện đếm số kí tự nhập vào từ bàn phím
Giải:

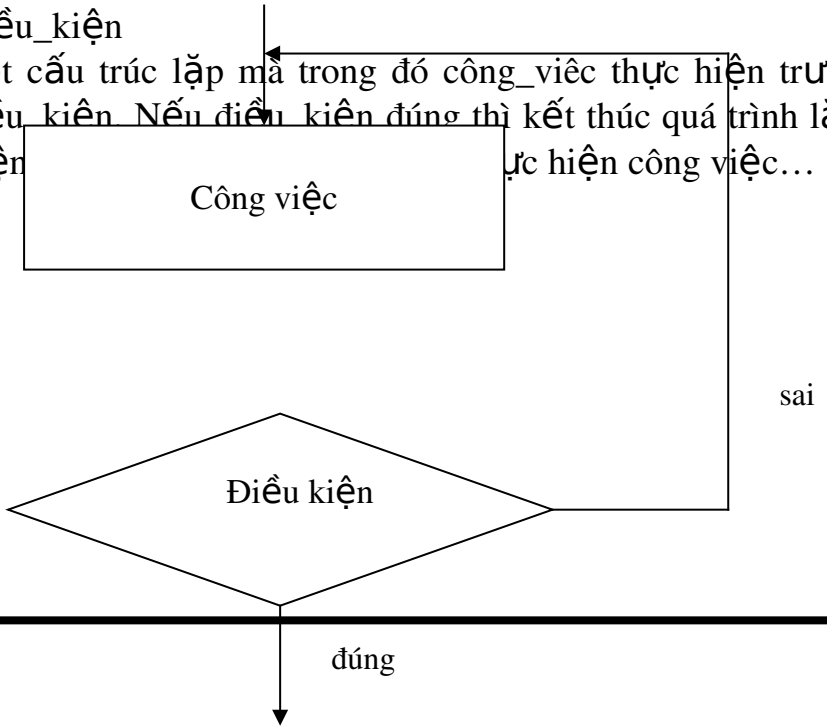
```
XOR CX,CX
MOV AH,1
WHILE:
        INT 21h
        CMP AL,13
        JE END_WHILE
                INC  CX
        JMP WHILE
END_ WHILE:
                MOV AH,4Ch
        INT 21h
```

g. Cấu trúc REPEAT …UNTIL…

Cú pháp:

REPEAT
công_việc
UNTIL điều_kiện

Đây là một cấu trúc lặp mà trong đó công_viêc thực hiện trước, sau đó mới kiểm tra điều_kiện. Nếu điều_kiện đúng thì kết thúc quá trình lặp. Ngược lại, nếu điều_kiện [illegible] ực hiện công việc…



Ví dụ: Viết một đoạn chương trình thực hiện nhập vào từ bàn phím một kí tự kết thúc bằng việc nhấn phím ENTER
Giải:

```
MOV AH,9
LEA DX,yeucau
INT 21h
REPEAT:
        MOV AH,1
        INT 21h
        CMP AL,13
        JNE REPEAT
        MOV AH,4Ch
        INT 21h
```

3.3 Lệnh chuyển hướng chương trình

a. Nhảy có điều kiện

Cú pháp:

Jxxx nhãn_đích

Nếu điều kiện nhảy được thoả mãn thì sẽ nhảy đến nhãn_đích và thi hành lệnh này. Nhãn có thể trước hoặc sau. Trước không quá 126 byte, sau không quá 127 byte.

Bảng các lệnh nhảy

Nhảy có dấu		
Kí hiệu	Chức năng	điều kiện nhảy
Jg/jnle	nhảy nếu lớn hơn nhảy nếu không nhỏ hơn hay bằng	Zf = 0, sf = 0f
Jge/jnl	nhảy nếu lớn hơn hay	Sf = 0f

Nhảy có dấu		
	bằng nhảy nếu không nhỏ hơn	
Jl/jnge	nhảy nếu nhỏ hơn nhảy nếu không lớn hơn hay bằng	Sf <> 0f
Jle/jng	nhảy nếu nhỏ hơn hay bằng nhảy nếu không lớn hơn	Zf = 1 hay sf = 0f
nhảy không dấu		
Ja/jnbe	nhảy nếu lớn hơn nhảy nếu không nhỏ hơn hay bằng	Cf = 0 và zf = 0
Jae/jnb	nhảy nếu lớn hơn hay bằng nhảy nếu không nhỏ hơn	Cf = 0
Jb/jnae	nhảy nếu nhỏ hơn nhảy nếu không lớn hơn hay bằng	Cf = 1
nhảy điều kiện đơn		
Je/jz	nhảy nếu bằng nhảy nếu bằng 0	Zf = 1
Jne/jnz	nhảy nếu không bằng nhảy nếu không bằng 0	Zf = 0

b. Nhảy không điều kiện

Cú pháp:

JMP nhãn_đích

Nhãn đích nằm trong cùng đoạn với JMP, vượt xa 126 byte đối với các lệnh nhảy có điều kiện.

c. Lệnh logic

Trong nhóm này ta quan tâm đến các lệnh sau:AND, OR,XOR,NOT

* AND

Cú pháp

AND đích,nguồn

Đích, nguồn phải có điều kiện:

- cùng độ dài
- không phải đồng thời là 2 ô nhớ, 2 thanh ghi đoạn.

Tác dụng: thường dùng để che đi hay giữ lại một vài bit nào đó của toán hạng đích

Ví dụ: AND AX,0Fh

* OR

Cú pháp

OR đích,nguồn

Đích, nguồn phải có điều kiện:

- cùng độ dài
- không phải đồng thời là 2 ô nhớ, 2 thanh ghi đoạn.

Tác dụng: thường dùng để thiết lập một vài bit nào đó của toán hạng đích bằng cách cộng logic toán hạng đó với toán hạng tức thời mà các bit 1 có vị trí tương ứng với bít cần lập.

Ví dụ: OR AL,BL

OR AL,0F h

* NOT

Cú pháp

NOT toanhang

Tác dụng: lấy phủ định của toán hạng (dùng để dảo bít một toán hay hạng lấy bù1)

* XOR : hoặc (loại trừ toán hạng)

Cú pháp

XOR toanhang, toanhang

Tác dụng: dùng để xóa về 0 một thanh ghi nào đó.

Ví dụ: XOR AX,AX ;xóa thanh ghi AX về 0

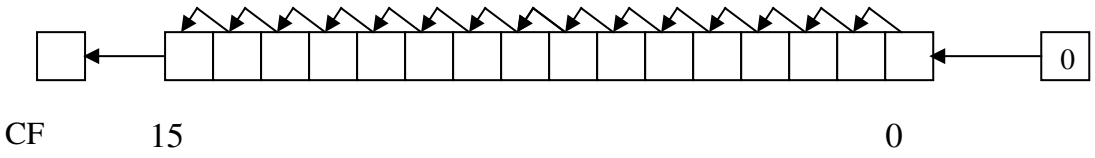
d. Nhóm lệnh dịch chuyển và quay

* SHL : Shift – left (dịch trái)

Cú pháp

SHL đích,CL

Tác dụng: dịch các bit của toán hạng đích sang trái CL vị trí, một giá trị 0 được đưa vào bên phải của toán hạng đích, còn bít MSB được đưa vào CF.

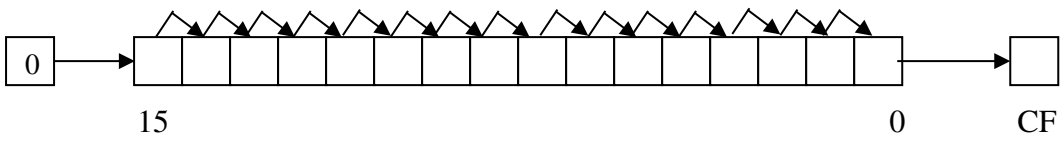


* SHR: Shift – Right; dịch phải

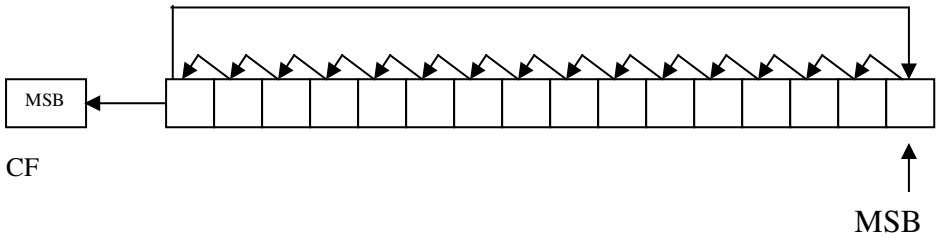
Cú pháp

SHR đích,CL

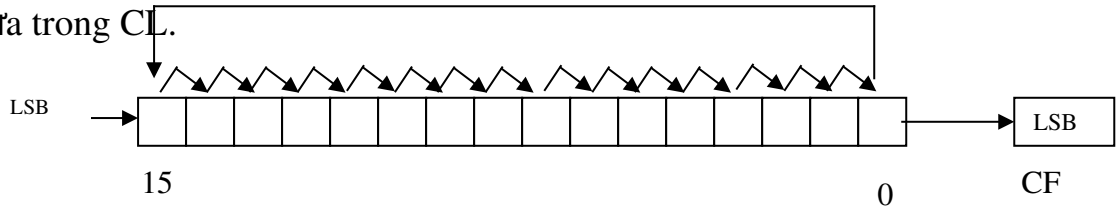
Tác dụng: dịch các bit của toán hạng đích sang phải CL vị trí, một giá trị 0 được đưa vào bít MSB còn giá trị của bit LSB được chuyển vào cờ CF.



* ROL: Rotation Left- quay trái
Cú pháp
ROL đích,CL
Dịch các bit sang bên trái. Bit MSB được đưa và LSB và cờ CF. muốn quay nhiều lần thì chứa trong CL



* ROR: Rotation Right – quay phải
Cú pháp
ROR đích, CL
Dịch sang bên phải. bit LSB được đưa vào MSB và cờ CF. muốn quay nhiều lần thì chứa trong CL.



4. Ngăn xếp và thủ tục

Mục đích:

- *Hiểu và sử dụng đượcđược ngăn xếp*
- *Hiểu được cách viết chương trình con và cách truyền tham số cho chương trình con.*

4.1 Ngăn xếp (stack)

a. Định nghĩa ngăn xếp:

Ngăn xếp là một tổ chức bộ nhớ sao cho ta chỉ có thể đọc một từ ở đỉnh ngăn xếp hoặc viết một từ vào đỉnh ngăn xếp. Địa chỉ của đỉnh ngăn xếp được chứa trong một thanh ghi đặc biệt gọi là con trỏ ngăn xếp SP (Stack Pointer).

Ứng với cấu trúc ngăn xếp, người ta có lệnh viết vào ngăn xếp PUSH và lệnh lấy ra khỏi ngăn xếp POP. Các lệnh này vận hành như sau:

- Cho lệnh PUSH

SP := SP +1

M (SP) := Ri (Ri là thanh ghi cần viết vào ngăn xếp)

- Cho lệnh POP

Ri := M(SP) (Ri là thanh ghi, nhận từ lấy ra khỏi ngăn xếp)

SP := SP -1

Trong các bộ xử lý RISC, việc viết vào hoặc lấp ra khỏi ngăn xếp dùng các lệnh bình thường. Ví dụ thanh ghi R30 là con trỏ ngăn xếp thì việc viết vào ngăn xếp được thực hiện bằng các lệnh:

ADDI R30, R30, 4 ; tăng con trỏ ngăn xếp lên 4 vì từ dài 32 bit
STORE Ri, (R30) ; Viết Ri vào đỉnh ngăn xếp
Việc lấy ra khỏi ngăn xếp được thực hiện bằng các lệnh :
LOAD Ri, (R30) ; lấy số liệu ở đỉnh ngăn xếp và nạp vào Ri
SUBI R30, R30,4 ; giảm con trỏ ngăn xếp bớt 4

b. Đặc tả ngăn xếp

- Che dấu thông tin:khi sử dụng lớp stack chúng ta không cần biết nó được lưu trữ trong bộ nhớ như thế nào và các phương thức của nó được thực hiện ra sao.Đây là vấn đề cho dấu thông tin.

- Tính khả thi và hiệu quả của ứng dụng:Tuy ứng dụng cần phải độc lập với hiện thực của cấu trúc dữ liệu nhưng việc chọn cách hiện thực nào ảnh hưởng đến tính khả thi và hiệu quả của ứng dụng.Chúng ta cần hiểu các ưu nhược điểm của mỗi cách hiện thực của cấu trúc dữ liệu để lựa chọn cho phù hợp với tính chất của ứng dụng.

- Tính trong sáng của chương trình:ưu điểm khác của che giấu thông tin là tính trong sáng của chương trình.Những tên gọi quen thuộc dành cho các thao tác trên cấu trúc dữ liệu giúp chúng ta hình dung rõ ràng giải thuật của chương trình.Chẳng hạn thao tác trên ngăn xếp,người ta thường quen dùng các từ:đầy-push;lấy-pop;

- Thiết kế từ trên xuống:sự tách rời giữa việc sử dụng cấu trúc dữ liệu và cách hiện thực của nó còn giúp chúng ta thực hiên tốt hơn quá trình thiết kế từ trên xuống cả cho cấu trúc dữ liệu và cả cho chương trình ứng dụng.

4.2 Chương trình con

Nhằm mục đích làm cho chương trình ngắn gọn và dễ hiểu, thông thường người ta thực hiện chia nhỏ chương trình thành các Module khác nhau, mỗi Module có thể thực hiện một hoặc một khối công việc nhất định. Mỗi một Module đó được gọi là một chương trình con.

Trong lập trình hợp ngữ thông thường người ta chỉ sử dụng một loại chương trình con thủ tục. Cấu trúc của chương trình con loại này được thực hiện như sau:

a. Cấu trúc của chương trình con

<tên ctc> PROC [kiểu]
; thân chương trình con
;……
RET
<tên ctc> ENDP

Trong đó:

+ tên ctc: là tên chương trình con mà người sử dụng tự đặt theo quy định đặt tên của ASSEMBLY.

+ PROC, ENDP: các lệnh giả được thực hiện để khai báo bắt đầu và kết thúc chương trình con.

+ kiểu: có thể là NEAR hoặc FAR.

NEAR (mặc định) có nghĩa là dòng lệnh gọi thủ tục ở cùng đoạn với thủ tục đó.

FAR có nghĩa là dòng lệnh gọi thủ tục ở trong một đoạn khác.

b. Ví dụ

```
Xoa_mh    PROC
          MOV AH,0
          MOV AL,3
          INT 10h
          RET
Xoa_mh    ENDP
```

c. Một số chú ý:

Tránh trường hợp sau khi thực hiện xong chương trình con, nội dung các thanh ghi có thể bị thay đổi, thường người ta sử dụng lệnh PUSH và POP trong chương trình con để đưa tạm vào ngăn xếp và sau đó lấy lại thanh ghi. Ví dụ trong đoạn chương trình trên, sau khi thực hiện chương trình con xong, nội dung của thanh ghi AX có thể bị thay đổi. Ta có thể viết lại như sau:

```
Xoa_mh    PROC
          PUSH AX    ;đẩy tạm AX vào ngăn xếp
          MOV AH,0
MOV AL,3
          INT 10h
          POP AX     ;lấy lại giá trị cũ từ ngăn xếp cho AX
          RET        ; sau đó trở về chương trình chính
Xoa_mh    ENDP
```

- Để người khác có thể đọc và hiểu rõ thủ tục thực hiện như thế nào thì, người lập trình phải có một đoạn giải thích như sau:

```
; Chức năng thủ tục
; Vào: (lấy thông tin từ chương trình gọi)
;Ra: (trả thông tin đã được xử lý về cho chương trình gọi)
; cách sử dụng (nếu có)
```

Ví dụ: Viết một thủ tục thực hiện nhân 2 số nguyên dương A và B bằng cách cộng và dịch các bit.

Thuật toán:

```
Tich = 0
Repeat
    If  LSB(B) = 1
    Then
```

```
            Tich = tich + A
            End_if
            SHL  A,1
            SHR  B,1
        Until B= 0
```

Đoạn mã:

```
Nhan PROC
; nhan 2 so A,B bang phep dich cong cac bit
; Vao: AX = A; BX = B
; Ra: DX = ketqua
        PUSH  AX   ;đẩy tạm vào AX vào ngăn xếp
        PUSH  BX   ;đẩy tạm vào BX vào ngăn xếp
        XOR DX,DX        ;xoá thanh ghi DX chứa tích
REPEAT:
;if B le
        TEST  BX,1 ; bit LSB của BX bằng 1?
        JZ END_IF  ;khôngbằng 1, dịch trái AX…
;then
        ADD DX,AX        ;tich=tich+AX
END_IF:
        SHL  AX,1  ;dịch trái AX
        SHR  BX,1  ;dịch phải BX
;until B=0
        JNZ REPEAT   ;B<>0, lặp lại
        POP BX      ;khôi phục lại BX
        POP AX      ; và AX từ ngăn xếp
        RET
Nhan ENDP
```

4.3 Truyền tham số cho chương trình con

Sau khi lập được chương trình con, người lập trình chỉ việc gọi chúng ra từ một đoạn chương trình nào đó như sau:

```
CALL <tên_chương _trình_con>
```

Việc gọi chương trình con rất đơn giản, song có một số chú ý khi ta thực hiện với chương trình con.

a. *Chương trình con nằm trong cùng một đoạn với chương trình.*

Nếu chương trình con nằm trong cùng một đoạn với chương trình ta có thể gọi và thực hiện theo mẫu sau:

```
        Title ctchinh
        .MODEL   Small
        .STACK  100h
        .DATA
        ; khai báo dữ liệu cho chương trình
```

```
.CODE
Main PROC
        MOV AX, @data
        MOV  DS,AX
        ; các lệnh của chương trình chính
        CALL Ctc
        ; các lệnh của chương trình chính
Main ENDP
Ctc  PROC
        ;các lệnh của chương trình con
        RET
Ctc ENDP
END main
```

b. Chương trình con được soạn thảo trong một tệp khác

Nếu chương trình con được soạn thảo trong một tệp khác (ví dụ CTC.lib), thì có thể sử dụng lệnh INCLUDE tại vị trí khai báo chương trình con.

Vídụ:

```
Title ctchinh
.MODEL   Small
.STACK  100h
.DATA
; khai báo dữ liệu cho chương trình
.CODE
Main PROC
        MOV AX, @data
        MOV  DS,AX
        ; các lệnh của chương trình chính
        CALL Ctc
        ; các lệnh của chương trình chính
Main ENDP
        INCLUDE Ctc.lib
END mai
```

4.4 Một số hàm của ngắt 21h

Ngắt 21h: chức năng ngắt của DOS

Hàm 01h: vào 1 kí tự từ bàn phím và hiển thị ra màn hình

Mô tả	Ví dụ minh hoạ
Vào: AH=01h	MOV AH,01h
Ra: AL= mã ASCII của kí tự nhậpvào	INT 21h MOV Ktu, AL

Hàm 02h: In một kí tự ra màn hình

Mô tả	Ví dụ minh hoạ

Vào:
AH=02h
DL = mã ASCII của kí tự nhập vào
Ra: không
	MOV AH,02 ;in ra màn hình
	MOV DL, 'A' ; chữ A
	INT 21h

Hàm 08h: và một kí tự từ bà phím không hiển thị ra màn hình

Mô tả	Ví dụ minh hoạ

Vào:
AH=08h
Ra AL = mã ASCII của kí tự nhập vào
	MOV AH,08h
	INT 21h
	MOV Ktu, AL

Hàm 09h: in một chuỗi kí tự ra màn hình

Mô tả	Ví dụ minh hoạ
Vào: AH=09h DS:DX =con trỏ đến chuỗi kết thúc bằng $ Ra: không	MOV AH,09h LEA DX,chuoi INT 21h

Hàm 4Ch: kết thúc chương trình .EXE

Mô tả	Ví dụ minh hoạ
Vào: AH= 4Ch Ra: không	MOV AH,4Ch INT 21h

Hàm 2Ah: Xác định ngày tháng

Mô tả	Ví dụ minh hoạ
Vào: AH= 2Ah Ra: AL = ngày trong tuần (0-6) CX = Năm DH = tháng (1-12) DL=ngày trong tháng	R.AH=$2A INTR($21,R) Ngay-tuan:=R.AL; Nam:=R.CX; Thang:=R.DH; Ngay:=R.DL;

Hàm 2Bh: Cài đặt ngày tháng (cài đặt ngày hệ thống)

Mô tả	Ví dụ minh hoạ
Vào: AH= 2Bh Ra:	R.AH=$2B Nam:=R.CX; Thang:=R.DH;

AL = ngày trong tuần (0-6) CX = Năm DH = tháng (1-12) DL=ngày trong tháng	Ngay:=R.DL; INTR($21,R) IF R>AL=0 then write('ok'); Else write('not ok');

Hàm 2Ch: Xác định thời gian hệ thống

Mô tả	Ví dụ minh hoạ
Vào: AH= 2Ch Ra: CH = giờ (0-23) CL = phút (0-59) DH = giây (0-59) DL=phần trăm giây (0-99)	R.AH=$2C INTR($21,R) Gio:=R.CH; Phut:=R.CL; Giay:=R.DH; Phan_tram:=R.DL

Hàm 2Dh: đặt lại thời gian hệ thống
Hàm 30h: Xác định số phiên bản của DOS
Hàm 36h: Xác định dung lượng còn trống trên đĩa

Mô tả	Ví dụ minh hoạ
Vào: AH= 36h DL = ổ đĩa Ra: BX = số liên cung chưa dùng CX = số byte/liên cung DX = số liên cung / đĩa AX= FFFFh nếu ổ đĩa không hợp lệ = số cung/liên cung (hợp lệ) DH = giây (0-59) DL=phần trăm giây (0-99)	R.DL := 1; INTR ($21,R); Free_cyl:=R.BX; Bps:=R.CX; {byte per sector} Cpd:= R.DX; {cylinder per disk} IF AX=FFFFh then write('no disk') Else Spc:= R.AX; {Sector per cylinder}

b. Một số loại ngắt khác
Ngắt 10h: ngắt màn hình
Hàm 00h: chọn chế độ hiển thị cho màn hình

Mô tả	Ví dụ minh hoạ
Vào: AH= 0h AL = chế độ 03h: text 80*25*16 12h: Grapt 640*480*16 13h: Grapt 320*200*256 Ra: không	R.AH= 0h; R.AL:=mode; INTR($10,R)

Hàm 02h: Dịch chuyển con trỏ

Mô tả	Ví dụ minh hoạ
Vào: AH= 02h BH = trang số DH = hàng DL = cột Ra: không	R.AH= 02h; R.BH:=trang; R.DH:= hang; R.DL:= cot; INTR($10,R)

Hàm 06h: Cuốn màn hình hay cửa sổ lên một số dòng xác định

Mô tả	Ví dụ minh hoạ
Vào: AH= 06h AL=số dòng (=0; toàn bộ) BH= thuộc tính của dòng trống CH,DL = số dòng, cột góc bên trái CL,DL= số dòng,cột góc dưới phải Ra: không	R.AH= 02h; R.BH:=trang; R.DH:= hang; R.DL:= cot; INTR($10,R)

Hàm 07h: Cuốn màn hình hay cửa sổ xuống một số dòng xác định

Mô tả	Ví dụ minh hoạ
Vào: AH= 07h AL=số dòng cuốn (=0; toàn bộ) BH= thuộc tính của dòng trống CH,CL= số dòng,cột góc dưới phải Ra: không	R.AH= 07h; R.AL:=so_dong; R.BH:=thuoc_tinh; R.CH:= dong1; R.CL = cot1 R.DH:= dong2; R.DL=cot2; INTR($10,R);

Hàm 09h: Hiển thị kí tự với thuộc tính tại vị trí con trỏ.

Mô tả	Ví dụ minh hoạ
Vào: AH= 09h AL= mã ASCII của kí tự BH= trang số BL= thuộc tính (text); màu (graph) CX = số lần viết kí tự Ra: không	R.AH= 09h; R.AL:=kitu; R.BH:=0 {trang số 0}; R.BL:= mau; R.CX:= solan; INTR($10,R);

Ngắt 16h: Ngắt bàn phím

Hàm 00h: Đọc kí tự từ bàn phím

Mô tả	Ví dụ minh hoạ
Vào: AH= 00h Ra: AH= mã quét của phím AL= ASCII của kí tự	R.AH= 00h; INTR($16,R); R.AH:= ma_scan; R.AL:= ma_ ASCII;

Hàm 02h: Lấy các cờ bàn phím

Mô tả	Ví dụ minh hoạ
Vào: AH= 02h Ra: AL= các cờ	R.AH= 02h; INTR($16,R); R.AH:= ma_scan; R.AL:= ma_ ASCII;

Ngắt 33: ngắt chuột

Hàm 02h: khởi tạo chuột

Mô tả	Ví dụ minh hoạ
Vào: AX= 00h Ra: AX= FFFFh không nhận chuột	R.AX= 00h; INTR($33,R); If R.AX:= FFFF then Write (‘khong khoi tạo duoc chuot’)

Hàm 01h: hiện trỏ chuột

Mô tả	Ví dụ minh hoạ
Vào: AX= 01h Ra: không	R.AX= 01h; INTR($33,R);

Hàm 02h: ẩn trỏ chuột

Mô tả	Ví dụ minh hoạ
Vào: AX= 02h Ra: không	R.AX= 02h; INTR($33,R);

Hàm 03h: trạng thái nhấn chuột

Mô tả	Ví dụ minh hoạ
Vào: AX= 03h Ra: CX,DX = toạ độ ảo của chuột BX = trạng thái nút chuột nhấn 0: nút trái 1: nút phải	R.AX= 03h; INTR($33,R); X = R.CX shl 3+1; Y = R.DX shl 3+1; If (R.BX and 1) = 1 then Write (‘phim trai chuot!’);

Mô tả	Ví dụ minh hoạ
2: nút giữa	

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 6

1. Giải thích các lệnh sau:

```
MOV  AL,3Fh;
MOV  DL,D7h;
ADD AL,DL;
```

2.Chỉ ra chế độ địa chỉ của các lệnh sau:

a. MOV AL,BL

b. MOV AX,100h

c.MOV AX,[101]

d. ADD AL,[BX]+5

e. ADD AL,[BX]+[SI]+1000

3. Cho biết mã lệnh mã máy của các lệnh trong bài 2

4.Viết chương trình thực hiện in ra màn hình các thông tin về bản thân. Sau mỗi lần hiển thị ra một thông tin, người sử dụng

5.Viết chương trình thực hiện việc nhập vào một kí tự từ bàn phím sau đó in ra màn hình 2 lần kí tự đó tại 2 dòng tiếp theo

6.Viết đoạn chương trình thực hiện việc nhập vào một kí tự từ bàn phím. Nếu là kí tự số thì in ra tại dòng tiếp theo, nếu không là kí tự số thì in ra một dòng thông báo "kí tự bạn vừa nhập không phải là kí tự số"

7.Viết đoạn chương trình thực hiện việc nhập vào một kí tự từ bàn phím. Nếu là chữ hoa thì đổi thành chữ thường và in ra kết quả ra màn hình, nếu là chữ thường thì đổi thành chữ hoa in kết quả ra màn hình, nếu không phải là kí tự thì in ra thông báo "kí tự bạn vừa nhập không phải là chữ"

8.Viết đoạn chương trình thực hiện việc nhập vào 2 kí tự số (0-9) từ bàn phím và thực hiện việc tính tổng hai số đó và in ra màn hình

9.Viết đoạn chương trình thực hiện việc nhập vào 2 kí tự số (0-9) từ bàn phím và thực hiện việc lấy số lớn trừ số nhỏ, in ra màn hình hiệu của 2 số.

10.Viết đoạn chương trình thực hiện việc nhập vào từ bàn phím một kí tự, kiểm tra nếu kí tự đó là chữ thì hiển thị 20 lần trên một dòng, nếu không là chữ thì thoát khỏi chương trình.

TÀI LIỆU THAM KHẢO

- Nguyễn Đình Việt (2000), *Giáo trình Kiến trúc máy tính*, NXB: Đại học Quốc gia Hà nội.

- Msc Võ Văn Chín, Ths Nguyễn Hồng Vân, KS Phạm Hữu Tài, *Giáo trình kiến trúc máy tính(1997)*, Khoa Công nghệ thông tin, Đại học Cần thơ.