UBND THÀNH PHỐ ĐÀ NẴNG
TRƯỜNG CAO ĐẲNG NGHỀ ĐÀ NẴNG

GIÁO TRÌNH
Môn học/ mô dun: Cấu trúc máy tính
NGHỀ: QUẢN TRỊ MẠNG
TRÌNH ĐỘ: TRUNG CẤP/CAO ĐẲNG

(Ban hành kèm theo Quyết định số: /2011/QĐ-……của ……

TUYÊN BỐ BẢN QUYỀN:

MÃ TÀI LIỆU:

MỤC LỤC

MÔN HỌC ĐÀO TẠO CẤU TRÚC MÁY TÍNH

Mã môn học/mô đun:MH 09

* VỊ TRÍ, TÍNH CHẤT, Ý NGHĨA VÀ VAI TRÒ CỦA MÔ ĐUN

- Vị trí: Môn học Cấu trúc máy tính được bố trí học sau các môn học chung, các môn tin học đại cương, tin học văn phòng, kỹ thuật điện-điện tử và học cùng với mô đun lắp ráp cài đặt máy tính.
- Tính chất: Là môn học kỹ thuật cơ sở thuộc môn học đào tạo nghề bắt buộc.
- Ý nghĩa : đây là môn cơ sở, cung cấp cho sinh viên các kiến thức về máy tính của nghề Quản trị mạng

* MỤC TIÊU MÔ ĐUN:

- Trình bày được lịch sử của máy tính, các thế hệ máy tính và cách phân loại máy tính.
- Mô tả 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.
- Trình bày được 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.
- Trình bày được chức năng và nguyên lý hoạt động của các loại bộ nhớ.
- Trình bày phương pháp lưu trữ dữ liệu đối với bộ nhớ ngoài.
- Cài đặt được chương trình và các lệnh điều khiển cơ bản trong Assembly để thực hiện bài toán theo yêu cầu.
- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.

+ Nội dung chính của môn học /mô đun (danh sách các chương mục/bài học...):

Số TT	Tên chương, mục	Thời gian			
		Tổng số	Lý thuyết	Thực hành, Bài tập	Kiểm tra * (LT hoặcTH)
I	**Tổng quan về kiến trúc máy tính** Các mốc lịch sử phát triển công nghệ máy tính Thông tin và sự mã hóa thông tin Đặc điểm của các thế hệ máy tính điện tử Kiến trúc và tổ chức máy tính Các mô hình kiến trúc máy tính	**10**	**5**	**5**	

II	**Kiến trúc tập lệnh của máy tính** Các thành phần cơ bản của một máy tính Kiến trúc các tập lệnh CISC(complex instruction set computer) và RISC (Reduced instruction set computer) Mã lệnh	**13**	**8**	**4**	**1**
III	**Bộ xử lý** Sơ đồ khối của bộ xử lý Đường dẫn dữ liệu Bộ điều khiển Tiến trình thực hiện lệnh máy Kỹ thuật ống dẫn lệnh Kỹ thuật siêu ống dẫn lệnh Các chướng ngại của ống dẫn lệnh Các loại ngắt	**9**	**4**	**4**	**1**
IV	**Bộ nhớ** Phân loại bộ nhớ Các loại bộ nhớ bắn dẫn Hệ thống nhớ phân cấp Kết nối bộ nhớ với bộ xử lý Các tổ chức cache	**13**	**8**	**4**	**1**
V	**Thiết bị nhớ ngoài** Các thiết bị nhớ trên vật liệu từ Thiết bị nhớ quang học Các loại thẻ nhớ An toàn dữ liệu trong lưu trữ	**15**	**5**	**10**	
VI	**Các loại bus** Định nghĩa bus, bus hệ thống Bus đồng bộ và không đồng bộ Hệ thống bus phân cấp Các loại bus sử dụng trong các hệ thống vi xử lý	**10**	**5**	**5**	

VII	**Ngôn ngữ Assembly** Tổng quan Cấu trúc chương trình Các lệnh điều khiển Ngăn xếp và các thủ tục	**20**	**10**	**9**	**1**
Cộng		**90**	**45**	**41**	**4**

YÊU CẦU VỀ ĐÁNH GIÁ HOÀN THÀNH MÔN HỌC/MÔ ĐUN

- Về kiến thức: Được đánh giá kiến thức qua bài kiểm tra viết, trắc nghiệm đạt được các yêu cầu sau:
  + Biết 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.
  + Hiểu các tập lệnh cơ bản trong Assembly.

- Về kỹ năng**:** Đánh giá kỹ năng thực hành của học sinh:
  + Hiệu chỉnh được các thông số để máy tính đạt hiệu xuất cao nhất.
  + Thực hiện được các phương pháp an toàn dữ liệu trên thiết bị lưu trữ.
  + Viết được các chương trình cơ bản bằng ngôn ngữ Assembly và thực thi chúng.
- Về thái độ: Cẩn thận, thao tác nhanh chuẩn xác, tự giác trong học tập.

LỜI NÓI ĐẦU

Chìa khóa để hướng tới một xã hội thông tin là phát triển công nghệ thông tin (CNTT), tuy nhiên để phát triển CNTT lâu dài và bền vững, không phải chỉ đào tạo những kiến thức mới nhất, mà trong nội dung đào tạo cũng phải trang bị học sinh sinh viên những kiến thức nền tảng, trên cơ sở đó tạo cho học sinh sinh viên phát huy tính sáng tạo, chủ động trong việc tiếp thu nghiên cứu, ứng dụng CNTT. Do đó, trong các trường đào tạo, học sinh sinh viên phải được trang bị các kiến thức nền tảng về CNTT và trong đó thể thiếu là môn học Cấu trúc máy tính.

Hiện nay có nhiều giáo trình cấu trúc máy tính, tuy nhiên hầu hết các giáo trình chỉ đáp ứng các đối tượng là sinh viên đại học. Giáo trình này viết chủ yếu cho đối tượng là học sinh sinh viên các trường dạy nghề.

Giáo trình cung cấp cho học sinh sinh viên những kiến thức cơ bản về cấu trúc máy tính, về tổ chức và hoạt động bộ vi xử lý, các thành phần phần trong hệ thống máy tính và các biện pháp kĩ thuật cơ bản. Cấu trúc máy tính là môn học cơ sở để học sinh sinh viên có thể thực hành bảo trì hệ thống máy tính.

Giáo trình bao gồm 7 chương:

Chương 1: Tổng quan về cấu trúc máy tính

Chương 2: Kiến trúc phần mềm bộ xử lý

Chương 3: Tổ chức bộ vi xử lý

Chương 4: Hệ thống nhớ

Chương 5: Thiết bị nhập xuất

Chương 6: Các loại bus

Chương 7: Ngôn ngữ assembly

Trong mỗi chương đều có giới thiệu mục tiêu, nội dung và các câu hỏi bài tập. Giáo trình có thể xem là nguồn tài liệu cung cấp thông tin cho các giáo viên giảng dạy, đồng thời cũng là tài liệu học tập cho sinh viên.

Nhân đây ban biên soạn cũng xin cảm ơn các lãnh đạo và đồng nghiệp của chúng tôi tại trường Cao đẳng nghề Đà Nẵng đã tạo mọi điều kiện giúp đỡ, cũng cho chúng tôi ý kiến quý báu trong quá trình biên soạn giáo trình này.

Vì thời gian có hạn và đây cũng là lần đầu tiên giáo trình được soạn thảo nên không thể tránh khỏi thiếu sót. Rất mong nhận ý kiến đóng góp bạn đọc.

Đà nẵng, ngày 20 tháng 7 năm 2012

Tham gia biên soạn

1. Trương Văn Hiền

2. Nguyễn Xuân Diệu

3. Nguyễn Thị Trường Giang

CHƯƠNG 1:TỔNG QUAN VỀ CẤU TRÚC MÁY TÍNH

Mã chương:..M1

Mục tiêu:

- Giúp sinh viên hiểu lịch sử phát triển của máy tính
- Biết được các thành tựu của máy tính
- Hiểu được khái niệm về thông tin
- Hiểu 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 mốc lịch sử phát triển công nghệ máy tính

Mục tiêu: *sinh viên hiểu được lịch sử phát triển của máy tính*

30 năm trước, 5150 ra đời đã phá vỡ mọi quan điểm trước đó về máy tính. Lần đầu tiên, máy tính được nhìn nhận như một thiết bị có kích thước vừa phải, hợp túi tiền và được công chúng chú ý nhiều hơn.

1982: Franklin Ace 100

Đây là chiếc máy tính gây ra vụ kiện về bản quyền phần mềm đầu tiên trong lịch sử. Acer bị Apple kiện vì vi phạm nhãn hiệu hàng hóa khi sao chép phần cứng và phần mềm của máy tính Apple II cho Franklin Ace 100 . Trong vụ kiện này, phần thắng thuộc về Apple.

1982: Commodore 64

Có thể coi Commodore là máy tính dành cho hộ gia đình nổi tiếng nhất. Từ năm 1982 tới năm 1993, gần 30 triệu máy Commodore 64 đã được bán ra trên toàn thế giới.

XT là bản nâng cấp máy tính cá nhân 5150 đầu tiên của IBM. XT có ổ cứng trong 10 MB. Sản phẩm này của IBM sau đó nhanh chóng trở thành máy tính tiêu chuẩn.

1983: Apple Lisa

Lisa là máy tính tiêu dùng đầu tiên có giao diện đồ họa. Tuy nhiên, cái giá 10.000 USD trở thành rào cản đưa sản phẩm đến với người tiêu dù.

1984: Macintosh

Macintosh thu được thành công vang dội tới mức 30 năm đó, các sản phẩmmáy tính hiện nay của Apple vẫn được coi là hậu duệ trực tiếp của Macintosh. Macintosh cũng có giao diện đồ họa như Lisa nhưng mức giá "mềm" hơn rất nhiều giúp sản phẩm này dễ tiêu thụ hơn.

1990: NeXT

Máy tính NeXT được sản xuất bởi công ty riêng của Steve Jobs thành lập sau khi ông rời Apple vào năm 1985. Tuy nhiên, chiếc máy tính này trở nên quan trọng vì 1 lý do khác: đây là mẫu máy tính đầu tiên được Tim Berners-Lee dùng làm máy chủ World Wide Web.

1996: Deep Blue

Năm 1994, máy tính Deep Thought của IBM bị kiện tướng cờ vua Garry Kasparov đánh bại một cách dễ dàng. Tháng 2 năm 1996, máy tính Deep Blue đánh thắng Garry Kasparov trong hiệp đấu đầu tiên. Đây là lần đầu tiên một đương kim vô địch thế giới thất bại trong một ván cờtrước đối thủ máy tính.Tuy nhiên, các hiệp sauđó Deepđã bị Garry Kasparov chinh phục. Sau lần thất bại này, các kỹ sư IBM ra sức nghiên cứu nâng cấp Deep Blue và trở lại "phục thù", đánh bại kiện tướng cờ vua vào năm 1997, trình diễn khả năng xử lý chưa từng thấy trong lịch sử trước đó.

1998: iMac

iMac đã xóa đi hình ảnh nhàm chán của những chiếc máy tính cá nhân màu xám cục mịch. Apple đã cách mạng hóa hình ảnh máy tính với những mẫu iMacnhiều màu sắc sặc sỡ.

Hiện tại là iPad?

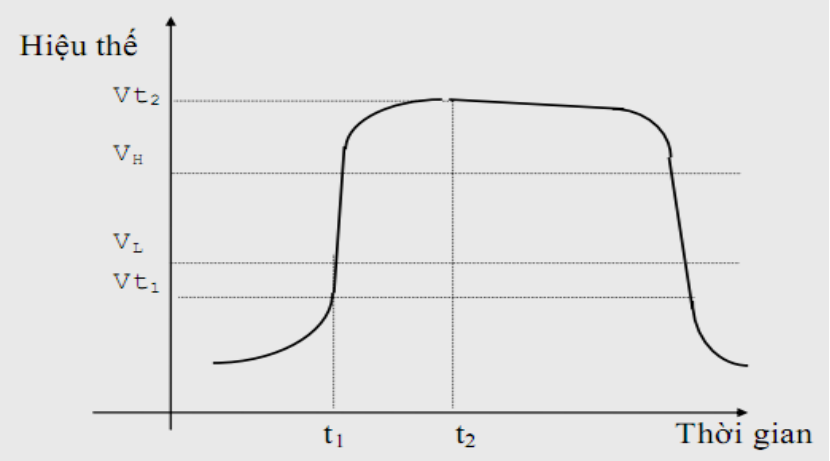
Loại "máy tính di động" này vẫn còn gây ra nhiều tranh cãi vớimối nghi ngờ liệumáy tính bảng nói chung và iPad nói riêng có phải chỉ là "mốt nhất thời". Dù sao hãy thử xem trong vòng 5 năm, máy tính sẽ thay đổi như thế nào nữa với iPad.

2.Thông tin và sự mã hóa thông tin

Mục tiêu:*nắm được thông tin là gì.Cách thức mã hóa thông tin*

2.1. Khái niệm thông tin và lượng thông tin

Khái niệm thông tin



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.

Lượng 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:

$$I = Log_2(N)$$

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 16 trạng thái có thể ứng với một lượng thông tin là:

$$I = Log_2(16) = 4 \text{ bit}$$

Tám trạng thái được ghi nhận nhờ 4 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.

Ví dụ : *Tám trạng thái khác nhau ứng với 3 số nhị phân*

Trạng thái	A0	A1	A2
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

2.2. Sự mã hóa thông tin

2.2.1. Mã và mã hóa là gì?

Mã hóa là phương pháp để biến thông tin (phim ảnh, văn bản, hình ảnh...) từ định dạng bình thường sang dạng thông tin không thể hiểu được nếu không có phương tiện giải mã.

Ví dụ một quy tắc mã hóa đơn giản:

Tất cả các ký tự đều bị thay thế bằng ký tự thứ 4 phía trước nó trong bảng chữ cái.

Bảng chữ cái gồm: "ABCDEFGHIJKLMNOPQRSTUVXYZ "

Vậy với câu: KY THUAT MA HOA CO BAN

Theo quy tắc trên, K => G, Y => T, " " => V ...

Sau khi mã hóa sẽ có được chuỗi: GYVPDQXPVIXVDKXVZKVYXJ

Rõ ràng đọc chuỗi này bạn sẽ không hiểu được nội dung là gì nếu không có khóa để giải mã. Khóa đó chính là số 4 ký tự mà bạn dịch.

Khi nhận được chuỗi này, bạn chỉ cần dịch ngược trở về bằng cách thay ký tự bằng ký tự thứ 4 phía sau nó. G => K, T => Y, ...

Với ví dụ trên,

Tất cả các ký tự đều bị thay thế bằng ký tự thứ 4 phía trước nó trong bảng chữ cái là mã hóa

thay ký tự bằng ký tự thứ 4 phía sau nó là giải mã.

2.2.2. Biểu diễn số trong máy tính

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:

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

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}

$541.25_{10} = 5 * 10^2 + 4 * 10^1 + 1 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$

$= (500)_{10} + (40)_{10} + (1)_{10} + (2/10)_{10} + (5/100)_{10}$

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 23.375_{10} sang nhị phân. Chúng ta sẽ chuyển đổi phần nguyên dùng phương thức số dư:

23	:	2	=	11	Dư	1 ↑
11	:	2	=	5	Dư	1
5	:	2	=	2	Dư	1
2	:	2	=	1	Dư	0
1	:	2	=	0	Dư	1

Kết quả: $(23)_{10} = (10111)_2$

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

0.375 x 2 =	0.75	Phần nguyên = 0 ↓
0.75 x 2 =	1.5	Phần nguyên = 1
0.5 x 2 =	1.0	Phần nguyên = 1

Kết quả: $(0.375)_{10} = (0.011)_2$

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. Ví dụ: 0.2.

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.

Thông thường, người ta nhóm 4 bit trong hệ nhị phân hệ để biểu diễn số dưới dạng thập lục phân (Hexadecimal), nhóm 3 bit để biểu diễn số dưới dạng bát phân (Octal).

Hệ thập phân	Hệ nhị phân	Hệ bát phân	Hệ thập lục phân
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

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:

$101010_2 = (101_2)(010_2) = 52_8$

$01101101_2 = (0110_2)(1101_2) = 6D_{16}$

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.

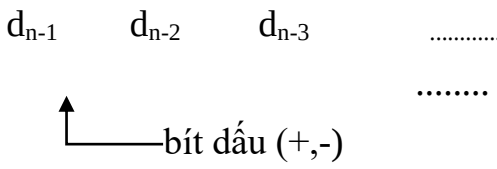
$$N = \sum_{i=0}^{n-1} d_i \, 2^i$$

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.

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*.



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 bằng số bù 1 và số bù 2

+ Số bù 1:

Trong cách biểu diễn này, số âm -N được có bằng cách thay các số nhị phân d_i của số dương 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).

+ Số bù 2:

Để có số bù 2 của một số nào đó, người ta lấy số bù 1 rồi cộng thêm 1.

Ví dụ: $+25_{10} = 00011001_2$

Số bù 1 của 25 là 11100110

\+ 1

Số bù 2 của 25 là 11100111

Vậy $-25_{10} = 11100111_2$

Chỉ có một giá trị 0: $+0 = 00000000_2$, $-0 = 00000000_2$

c) Cách biểu diễn số nguyên bằng mã BCD (Binary Coder Decimal)

Dùng 4 bit để mã hóa cho các chữ số thập phân từ 0 đến 9.

0 → 0000 5 → 0101

1 → 0001 6 → 0110

2 → 0010 7 → 0111

3 → 0011 8 → 1000

4 → 0100 9 → 1001

Có 6 tổ hợp không sử dụng (từ 10 đến 15) : 1010, 1011, 1100, 1101, 1110, 1111.

Ví dụ :

35 → 0011 0101 BCD

61 → 0101 0001 BCD

29 → 0010 1001 BCD

Các kiểu lưu trữ số BCD

- BCD không gói (Unpacked BCD) : mỗi số BCD 4 bit được lưu trữ trong 4 bit thấp của mỗi byte. Ví dụ : số 35 được lưu trữ như sau :

	0011		0101

- BCD gói (Packed BCD): hai số BCD được lưu trữ trong 1 byte.

Ví dụ : số 35 được lưu trữ như sau :

0011	0101

Cách biểu diễn số với dấu chấm động

- Tổng quát : một số thực X được biểu diễn theo kiểu số dấu chấm động như sau :

$X = M * R^E$

M là phần định trị (Mantissa)

R là cơ số (Radix)

E là phần mũ (Exponent)

- Chuẩn IEEE 754: Có nhiều cách biểu diễn dấu chấm động, trong đó cách biểu diễn theo chuẩn IEEE 754 được dùng rộng rãi trong khoa học máy tính hiện nay. Chuẩn IEEE 754 định nghĩa hai dạng biểu diễn số chấm động:

+ Dạng 32 bit :

1 bit 8 bit 23 bit

S	E	M

S là bit dấu, S = 0 là số dương, S = 1 là số âm.

e (8 bit) là mã excess-127 của phần mũ E

$e = E + 127 \rightarrow E = e - 127$

giá trị 127 được gọi là độ lệch (bias)

m (23 bit) là phần lẻ của phần định trị M

M = 1.m

Công thức xác định giá trị của số thực :

$X = (-1)^S * 1.m*2^{e-127}$

Ví dụ 1: xác định giá trị của số thực được biểu diễn bằng 32 bit như sau :

1100 0001 0101 0110 0000 0000 0000 0000

- S =1 → số âm

- $e = 1000\ 0010_2 = 130 \rightarrow E = 130 - 127 = 3$

Vậy $X = -1.10101100*2^3 = -1101.011 = -13.375$

Ví dụ 2: Biểu diễn số thực X = 83.75 về dạng dấu chấm động IEEE 754 32 bit.

$X = 83.75_{10} = 0101\ 0011.11_2 = 1.01001111*2^6$

Ta có: S = 0 vì đây là số dương

$e - 127 = 6 \rightarrow E = 127 + 6 = 133_{10} = 1000\ 0101_2$

Vậy X = 0 100 0010 1 010 0111 1000 0000 0000 0000

S E M

Các qui ước đặc biệt:

- Các bit của e bằng 0, các bit của m bằng 0, thì $X = \pm 0$
- Các bit của e bằng 1, các bit của m bằng 0, thì $X = \pm \infty$
- Các bit của e bằng 1, còn m có ít nhất một bit bằng 1, thì nó không biểu diễn cho số nào cả (NaN - Not a number)
- Phạm vi biểu diễn: 2^{-127} đến 2^{+127}, 10^{-38} đến 10^{+38}

+ Dạng 64 bit:

1bit 11 bit 52 bit

S	e	M

- S là bit dấu
- e (11 bit) là mã excess - 1023 của phần mũ E: E = e - 1023
- m (52 bit) là phần lẻ của phần định trị M
- Giá trị của số thực:

$X = (-1)^S * 1.m * 2^{e-1023}$

- Giải giá trị biểu diễn là: 10^{-308} đến 10^{+308}

3. Đặc điểm của các thế hệ máy tính điện tử

Mục tiêu:- *nắm được các đặc trưng của các thế hệ máy tính*

- qua mỗi thế hệ thấy được sự phát triển của máy tính điện tử

3.1. Thế hệ thứ nhất: (1945-1955)

Máy tính được xây dựng trên cơ sở đèn điện tử mà mỗi đèn tượng trưng cho 1 bit nhị phân. Do đó máy có khối lượng rất lớn, tốc độ chậm và tiêu thụ điện năng lớn. Như máy ENIAC bao gồm 18000 đèn điện tử, 1500 rơ-le, nặng 30 tấn, tiêu thụ công suất 140KW. Về kiến trúc nó có 20 thanh ghi, mỗi thanh ghi chứa 1 số thập phân 10 chữ số. Chiếc máy được lập trình bằng cách đặt vị trí (set) của 6000 chuyển mạch (switch) - mỗi cái có nhiều vịt trí và nối vô số ổ cắm (socket) với một "rừng" đầu cắm (jumper).

Cùng thời kì này, 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 số học và logic (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.

3.2. Thế hệ thứ hai: (1955-1965).

Máy tính được xây dựng trên cơ sở là các đèn bán dẫn (transistor), Công ty Bell đã phát minh ra transistor vào năm 1948 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. Máy tính đầu tiên thế hệ này có tên là TX-0 (transistorized experimental computer 0).

3.3. Thế hệ thứ ba: (1965-1980).

Máy tính dùng mạch tích hợp (còn gọi là mạch vi điện tử - IC) cho phép có thể đặt hàng chục transistor trong một vỏ(chip) , nhờ đó người ta có thể chế tạo các máy tính nhỏ hơn, nhanh hơn và rẻ hơn các máy tính dùng Transistor ra đời trước nó. Điển hình là thế hệ máy System/360 của IBM. Thế hệ máy tính này có những bước đột phá mới như sau:

- Tính tương thích cao: Các máy tính trong cùng một họ có khả năng chạy các chương trình, phần mềm của nhau.

- Đặc tính đa chương trình: Tại một thời điểm có thể có vài chương trình nằm trong bộ nhớ và một trong số đó được cho chạy trong khi các chương trình khác chờ hoàn thành các thao tác vào/ra.

- Không gian địa chỉ rất lớn (2^{24} byte = 16Mb).

3.4. Thế hệ thứ tư: (1980- nay)

Máy tính được xây dựng trên các vi mạch cỡ lớn (LSI) và cực lớn (VLSI).

Đây là thế hệ máy tính số ngày nay, nhờ công nghệ bán dẫn phát triển vượt bậc, mà người ta có thể chế tạo các mạch tổ hợp ở mức độ cực lớn. Nhờ đó máy tính ngày càng nhỏ hơn, nhẹ hơn, mạnh hơn và giá thành rẻ hơn. Máy tính cá nhân bắt đầu xuất hiện và phát triển trong thời kỳ này.

Dựa vào kích thước vật lý, hiệu suất và lĩnh vực sử dụng, hiện nay người ta thường chia máy tính số thế hệ thứ tư thành 5 loại chính, các loại có thể trùm lên nhau một phần:

- Microcomputer: Còn gọi là PC (personal computer), là những máy tính nhỏ, có 1 chip vi xử lý và một số thiết bị ngoại vi. Thường dùng cho một người, có thể dùng độc lập hoặc dùng trong mạng máy tính.

- Minicomputer: Là những máy tính cỡ trung bình, kích thước thường lớn hơn PC. Nó có thể thực hiện được các ứng dụng mà máy tính cỡ lớn thực hiện. Nó có khả năng hỗ trợ hàng chục đến hàng trăm người làm việc. Minicomputer được sử dụng rộng rãi trong các ứng dụng thời gian thực, ví dụ trong điều khiển hàng không, trong tự động hoá sản xuất.

- Supermini: Là những máy Minicomputer có tốc độ xử lý nhanh nhất trong họ Mini ở những thời điểm nhất định. Supermini thường được dùng trong các hệ thống phân chia thời gian, ví dụ các máy quản gia của mạng.

- Mainframe: Là những máy tính cỡ lớn, có khả năng hỗ trợ cho hàng trăm đến hàng ngàn người sử dụng. Thường được sử dụng trong chế độ các công việc sắp xếp theo lô lớn (Large-Batch-Job) hoặc xử lý các giao dịch (Transaction Processing), ví dụ trong ngân hàng.

- Supercomputer: Đây là những siêu máy tính, được thiết kế đặc biệt để đạt tốc độ thực hiện các phép tính dấu phẩy động cao nhất có thể được. Chúng thường có kiến trúc song song, chỉ hoạt động hiệu quả cao trong một số lĩnh vực.

4. Kiến trúc và tổ chức máy tính

Mục tiêu:*nắm được các khái niệm kiến trúc máy tính và tổ chức máy tính, phân biệt được hai khái niệm đó.*

4.1. Khái niệm kiến trúc máy tính

Kiến trúc máy tính là khoa học về việc lựa chọn và kết nối các thành phần phần cứng để tạo ra các máy tính đạt được các yêu cầu về chức năng (functionality), hiệu năng (performance) và giá thành (cost).
Yêu cầu chức năng đòi hỏi máy tính phải có thêm nhiều tính năng phong phúvà hữu ích; yêu cầu hiệu năng đòi hỏi máy tính phải đạt tốc độ xử lý cao hơn và yêu cầu giá thành đòi hỏi máy tính phải càng ngày càng rẻ hơn.

Để đạt được cả ba yêu cầu về chức năng,hiệu năng và giá thành là rất khó khăn. Tuy nhiên, nhờ có sự phát triển rất mạnh mẽ của côngnghệ vi xử lý, các máy tính ngày nay có tính năng phong phú, nhanh hơn và rẻ hơn so vớimáy tính các thế hệ trước.

Kiến trúc máy tính được cấu thành từ 3 thành phần con: (i)
Kiến trúc tập lệnh (Instruction SetArchitecture), (ii)
Vi kiến trúc (Micro Architecture) và *Thiết kế hệ thống* (System Design).

Kiến trúc tập lệnh là hình ảnh của một hệ thống máy tính ở mức ngôn ngữ máy.Kiến trúctập lệnh bao gồm các thành phần: tập lệnh, các chế độ địa chỉ, các thanh ghi, khuôn dạngđịa chỉ và dữ liệu.

Vi kiến trúc là mô tả mức thấp về các thành phần của hệ thống máy tính, phối ghép vàviệc trao đổi thông tin giữa chúng.Vi kiến trúc giúp trả lời hai câu hỏi (1) Các thành phần phần cứng của máy tính kết nối với nhau như thế nào?
và (2) Các thành phần phần cứngcủa máy tính tương tác với nhau như thế nào để thực thi tập lệnh?

Thiết kế hệ thống: bao gồm tất cả các thành phần phần cứng của hệ thống máy tính, bao gồm: Hệ thống phối ghép (các bus và các chuyển mạch), Hệ thống bộ nhớ , Các cơ chế giảm tải cho CPU (như truy nhập trực tiếp bộ nhớ) và Các vấn đề khác(như đa xử lý và xử lý song song).

4.2. Khái niệm tổ chức máy tính

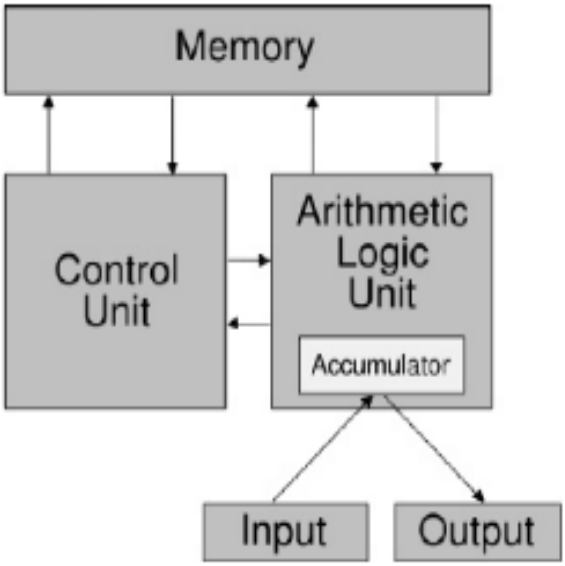
Tổ chức máy tính hay cấu trúc máy tính là khoa học nghiên cứu về các bộ phận của máy tínhvà phương thức làm việc của chúng.Với định nghĩa như vậy, tổ chức máy tính khá gần gũi với vi kiến trúc – một thành phần của kiến trúc máy tính. Như vậy, có thể thấy rằng, kiến trúc máy tính và khái niệm rộng hơn, nó bao hàm cả tổ chức hay cấu trúc máy tính.

5. Các mô hình kiến trúc máy tính

Mục tiêu:*Hiểu được các mô hình kiến trúc Von Neuman và Havard*

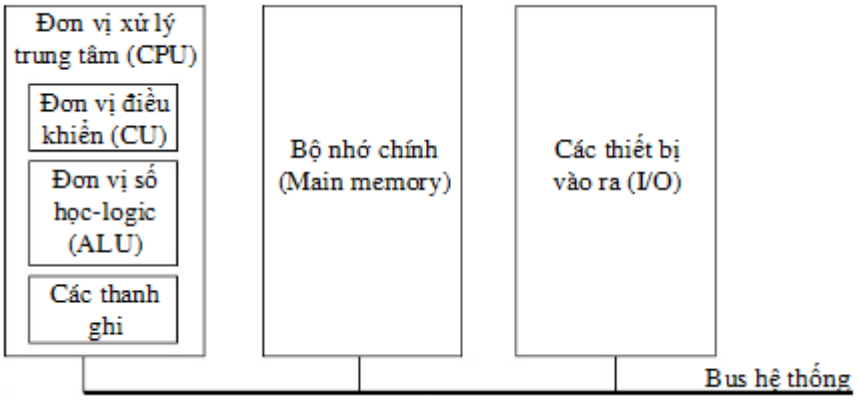
5.1. Mô hình kiến trúc Von Neumann

Kiến trúc máy tính von-Neumann được nhà toán học John von-Neumann đưa ra vào năm1945 trong một báo cáo vềmáy tính EDVACnhư minh hoạ trên



Kiến trúc máy tínhvon- Neumann nguyên thuỷ.

Các máy tính hiện đại ngày nay sử dụng kiến trúc máy tính von-Neumann cải tiến – còn gọi là kiến trúc máy tính von-Neumann hiện đại,như minh hoạ trên hình bên dưới.



Kiến trúcmáy tính von-Neumann hiện đại

Các đặc điểm của kiến trúc von-Neumann

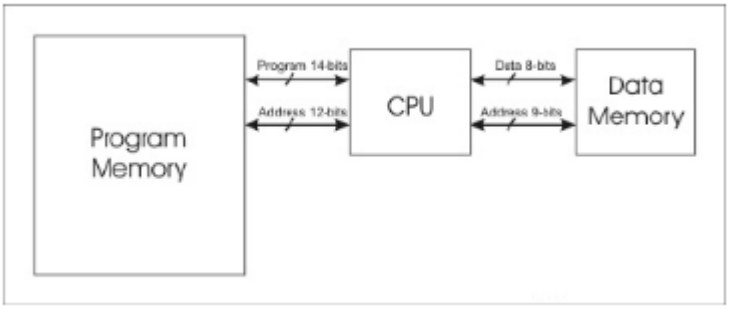
Kiến trúc von- Neumann dựa trên 3 khái niệm cơ sở:(1)Lệnh và dữ liệu được lưu trữ trong bộnhớ đọc ghi chia sẻ-một bộ nhớ duy nhất được sử dụng để lọu trữ cả lệnh và dữ liệu, (2)Bộnhớ được đánh địa chỉ theo vùng, không phụ thuộc vào nội dung nó lưu trữ và (3)Các lệnh của một chương trình được thực hiện tuần tự.

Quá trìnhthực hiện lệnh được chia thành3 giai đoạn (stages) chính:(1)
CPU đọc (fetch) lệnh từ bộ nhớ , (2)CPU giải mã và thực hiện lệnh;nếu lệnh yêu cầu dữ liệu, CPU đọc dữ liệu từ bộ nhớ;và (3)CPU ghi kết quả thực hiện lệnh vào bộ nhớ (nếu có).

5.2. Mô hình kiến trúc Havard

Kiến trúc máy tính Harvard là một kiến trúc tiên tiến như minh hoạ trên

hình.



Kiến trúc máy tính Harvard

Kiến trúc máy tính Harvard chia bộ nhớ trong thành hai phần riêng rẽ: Bộ nhớ lưu chươngtrình (Program Memory) và Bộ nhớ lưu dữ liệu (Data Memory). Hai hệ thống bus riêng được sử dụng để kết nối CPU với bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu. Mỗi hệ thống bus đều có đầy đủ ba thành phần để truyền dẫn các tín hiệu địa chỉ, dữ liệu và điều khiển.

Máy tính dựa trên kiến trúc Harvard có khả năng đạt được tốc độ xử lý cao hơn máy tính dựa trên kiến trúc von-Neumann do kiến trúc Harvard hỗ trợ hai hệ thống bus độc lập với băngthông lớn hơn. Ngoài ra, nhờ có hai hệ thống bus độc lập, hệ thống nhớ trong kiến trúc Harvard hỗ trợ nhiều lệnh truy nhập bộ nhớ tại một thời điểm, giúp giảm xung đột truy nhập bộ nhớ, đặc biệt khi CPU sử dụng kỹ thuật đường ống (pipeline).

CÂU HỎI VÀ BÀI TẬP

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?

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:
    b. Số bù 1.
    c. Số bù 2.
13. Đổi các số sau đây:
    a. $(011011)_2$ ra số thập phân.

b. $(55.875)_{10}$ ra số nhị phân.

14. Biểu diễn số thực (31.75)10 dưới dạng số có dấu chấm động chính xác đơn 32 bit

CHƯƠNG 2: KIẾN TRÚC PHẦN MỀM BỘ XỬ LÝ

Mã chương:..M2

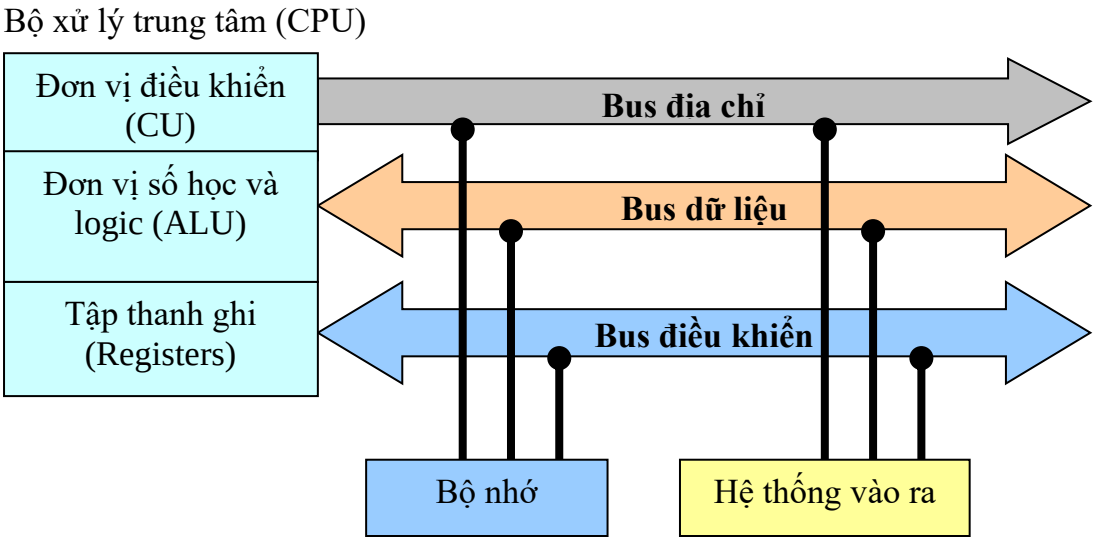
Mục tiêu

- Biết tổng quát tập lệnh của các 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
- Hiểu được kiến trúc RISC (Reduced Instruction Set Computer)

1. Các thành phần cơ bản của máy tính

Mục tiêu: *Hiểu được các thành phần cơ bản của một máy vi tính*

Thành phần cơ bản của một bộ máy tính gồm: bộ xử lý trung tâm (CPU: Central Processing Unit), bộ nhớ, 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.



Hình 2.1: Cấu trúc 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).

Thành phần cơ bản của một máy tính bao gồm :

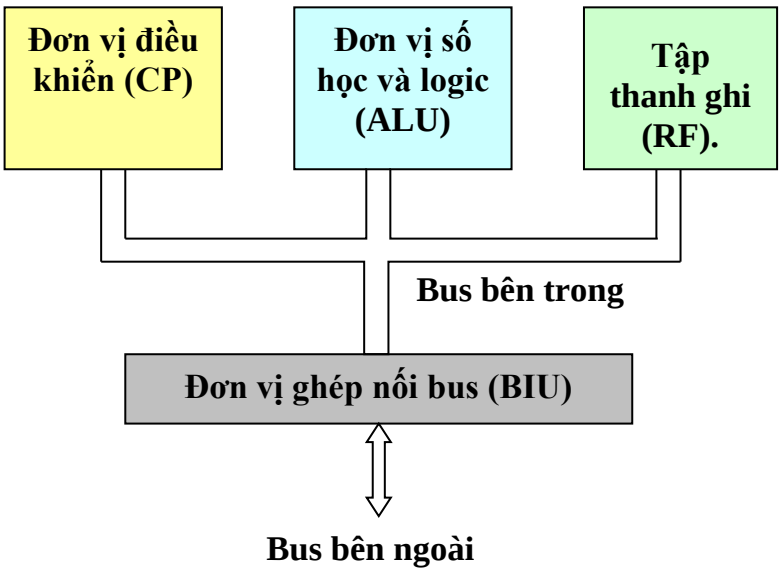
1.1 Bộ xử lý trung tâm (CPU)

+ Chức năng:

- Điều khiển hoạt động của máy tính .
- Xử lý dữ liệu .

+ Nguyên tắc hoạt động cơ bản: CPU hoạt động theo chương trình nằm trong bộ nhớ chính.

Cấu trúc cơ bản của CPU:



Hình 2.2: Cấu trúc cơ bản của CPU

Các thành phần cơ bản của CPU

- Đơn vị điều khiển (Control Unit – CU): điều khiển hoạt động của máy tính theo chương trình đã định sẵn.

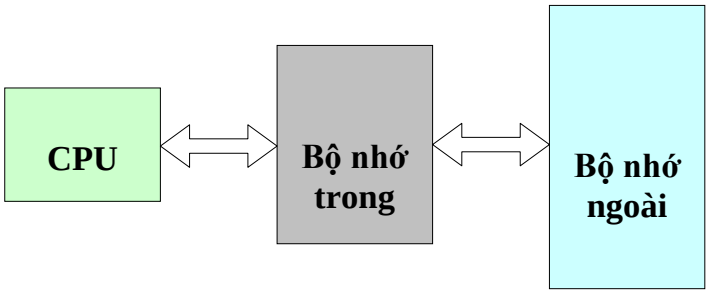
- Đơn vị số học và logic (Arithmetic and Logic Unit – ALU): thực hiện các phép toán số học và các phép toán logic trên các dữ liệu cụ thể.

- Tập thanh ghi (Register File - RF): lưu giữ các thông tin tạm thời phục vụ cho hoạt động của CPU.

- Đơn vị nối ghép bus (Bus interface Unit - BIU): kết nối và trao đổi thông tin giữa bus bên trong (internal bus) và bus bên ngoài (external bus).

1.2 Bộ nhớ máy tính

- + Chức năng: lưu trữ chương trình và dữ liệu.
- + Các thao tác cơ bản với bộ nhớ:
  - Đọc (Read)
  - Ghi (Write)
- + Các thành phần chính:
  - Bộ nhớ trong (Internal Memory)
  - Bộ nhớ ngoài (External Memory)



Hình 2.3: Bộ nhớ máy tính

➢ Bộ nhớ trong (Internal memory)

- Chức năng và đặc điểm:
  + Chứa các thông tin mà CPU có thể trao đổi trực tiếp.
  + Tốc độ rất nhanh.
  + Dung lượng không lớn.
  + Sử dụng bộ nhớ bán dẫn: ROM, RAM.

- Các loại bộ nhớ trong: Bộ nhớ chính, Bộ nhớ cache (bộ nhớ đệm nhanh).

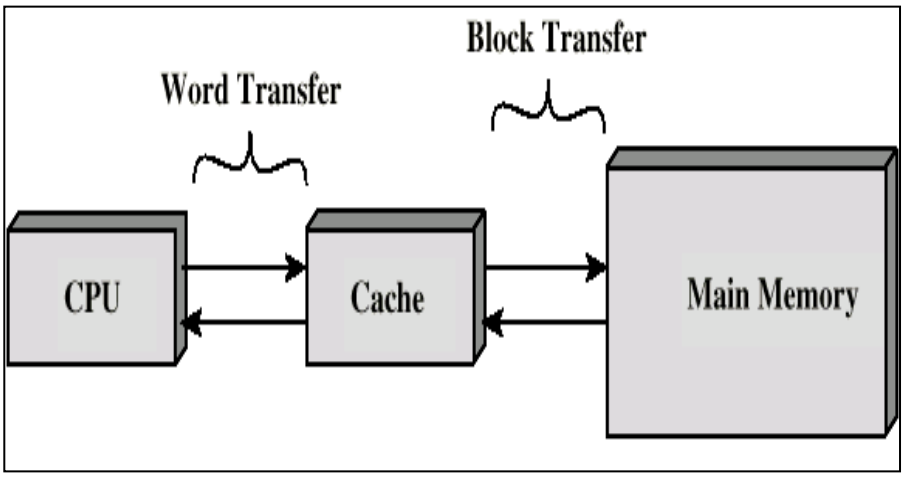
▪ Bộ nhớ chính (Main memory)

- Chứa các chương trình và dữ liệu đang được CPU sử dụng.
- Tổ chức thành các ngăn nhớ được đánh địa chỉ.
- Ngăn nhớ thường được tổ chức theo byte.
- Nội dung của ngăn nhớ có thể thay đổi, song địa chỉ vật lý của ngăn nhớ luôn cố định.

▪ Bộ nhớ đệm nhanh (Cache memory)

- Bộ nhớ có tốc độ nhanh được đặt đệm giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy nhập bộ nhớ.
- Dung lượng nhỏ hơn bộ nhớ chính
- Tốc độ nhanh hơn
- Cache thường được chia thành một số mức
- Cache có thể được tích hợp trên chip vi xử lý.

- Cache có thể có hoặc không



Hình 2.4: Bộ nhớ đệm Cache

➢ Bộ nhớ ngoài (External memory)

- Chức năng và đặc điểm:
  + Lưu giữ tài nguyên phần mềm của máy tính.
  + Được kết nối với hệ thống dưới dạng các thiết bị vào-ra.
  + Dung lượng lớn.
  + Tốc độ chậm.
- Các loại bộ nhớ ngoài:
  + Bộ nhớ từ: đĩa cứng, đĩa mềm.
  + Bộ nhớ quang: đĩa CD, DVD.
  + Bộ nhớ bán dẫn: Flash disk, memory card.

1.3 Hệ thống vào - ra

- Chức năng: Trao đổi thông tin giữa máy tính với thế giới bên ngoài.
- Các thao tác cơ bản: + Vào dữ liệu (Input)
  + Ra dữ liệu (Output)
- Các thành phần chính:

+ Các thiết bị ngoại vi (Peripheral Devices): chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính.

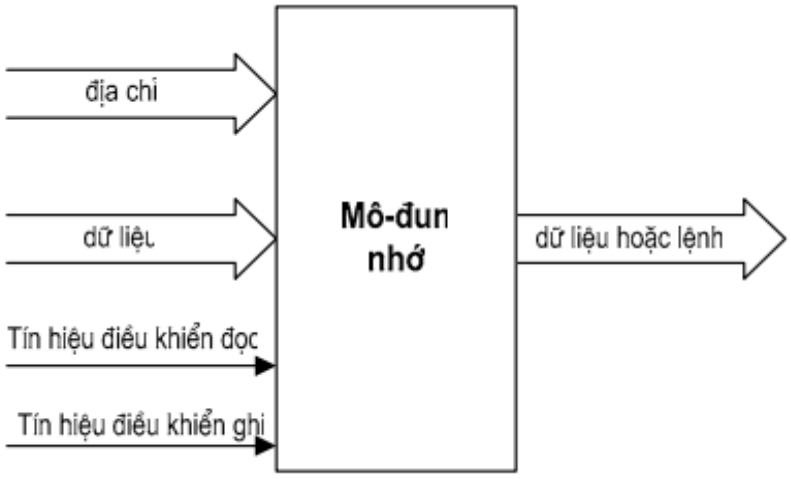
Thiết bị vào: bàn phím, chuột, máy quét ...

Thiết bị ra: màn hình, máy in ...

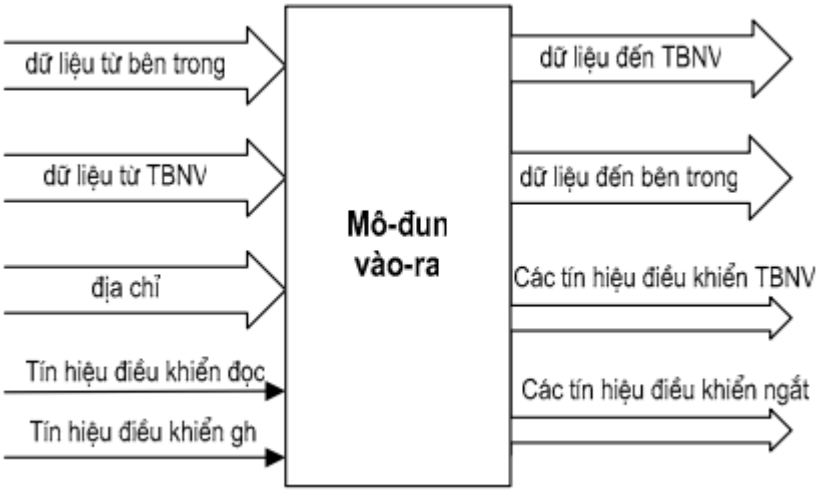
+ Các mô-đun vào ra (IO Modules): nối ghép các thiết bị ngoại vi với máy tính.

1.4 Liên kết hệ thống

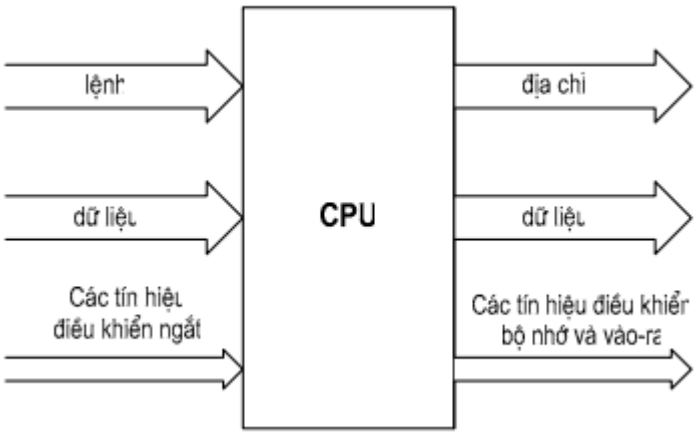
Luồng thông tin trong máy tính trong đó có các mô đun trong máy tính như CPU, mô đun nhớ, mô đun vào ra cần được kết nối với nhau.



Hình 2.5 Kết nối mô đun nhớ



Hình 2.6 Kết nối mô đun vào ra



Hình 2.7 Kết nối CPU

Kết nối và vận chuyển thông tin giữa các thành phần với nhau.Để thực hiện được điều đó chúng ta có khái niệm bus.Bus là đường truyền tín hiệu điện chung nối các thiết bị khác nhau trong một hệ thống máy tính. Bus thường bao gồm 50 đến 100 dây dẫn được gắn chặt với mainboard, trên các dây này có các đường nối đưa ra, các đầu này được sắp xếp và cách nhau một khoảng quy định để có thể cắm vào đó các bảng mạch điều khiển vào ra hoặc bộ nhớ. Chúng ta sẽ tìm hiểu kĩ hơn hệ thống bus ở chương 6 trong giáo trình.

2. Kiến trúc các tập lệnh CISC và RISC

Mục tiêu:*Hiểu được kiến trúc tập lệnh Cisc và Risc*

2.1. Kiến trúc tập lệnh CISC

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 này 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 2.1 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.

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	0	

<u>Bảng 2.1:</u> Đặc tính của một vài máy CISC

2.2. Kiến trúc tập lệnh RISC

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 2.2 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.

Bộ xử lý	IBM 801	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 bit)	32	32	32
Kỹ thuật chế tạo	ECL MSI	NMOS VLSI	NMOS VLSI
Cách thực hiện lệnh	Thanh ghi-thanh ghi	Thanh ghi-thanh ghi	Thanh ghi-thanh ghi

Bảng 2.2 : Đặc tính của ba mẫu đầu tiên máy RISC

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.

3. Mã lệnh

Mục tiêu:*nắm được lệnh máy,hiểu được mã lệnh*

3.1 Khái niệm lệnh máy, mã lệnh

Một lệnh mô tả bằng mã nhị phân có thể dài từ 1 đến 6 byte. Cấu trúc chung của một lệnh bao gồm:

Mã lệnh Toán hạng

Mã lệnh nhằm xác định tương ứng với lệnh là hoạt động hay thao tác nào cần được thực hiện.

Các toán hạng là đối tượng được xử lý bởi lệnh.

Ví dụ: Trong lệnh MOV AX,BX thì MOV là mã lệnh, xác định đây là lệnh chuyển dữ liệu. AX,BX xác định toán hạng được xử lý bởi lệnh.

Tùy theo từng lệnh mà độ dài của nó có sự khác nhau.

3.2 Tập lệnh

Mỗi bộ xử lý có một tập lệnh xác định. Tập lệnh thường có hàng chục đến hàng trăm lệnh. Mỗi lệnh là một chuỗi nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định

Các lệnh được mô tả bằng kí hiệu gợi nhớ.

3.2.1 Các kiểu thao tác

* Gán trị

Việc gán trị, gồm cả gán trị cho biểu thức số học và logic, được thực hiện nhờ một số lệnh mã máy. Cho các kiến trúc RISC, ta có thể nêu lên các lệnh sau :

- Lệnh bộ nhớ

LOAD Ri, M (địa chỉ) ; M[địa chỉ] ← Ri : nạp dữ liệu.

STORE Ri, M(địa chỉ); Ri ← M[địa chỉ] : Cất dữ liệu.

Địa chỉ được tính tuỳ theo kiểu định vị được dùng.

- Lệnh tính toán số học: tính toán số nguyên trên nội dung của hai thanh ghi Ri, Rj và xếp kết quả vào trong Rk:

ADD (cộng)

ADDD (cộng số có dấu chấm động, chính xác kép)

SUB (trừ)

SUBD (trừ số có dấu chấm động, chính xác kép)

MUL (nhân)

DIV (chia)

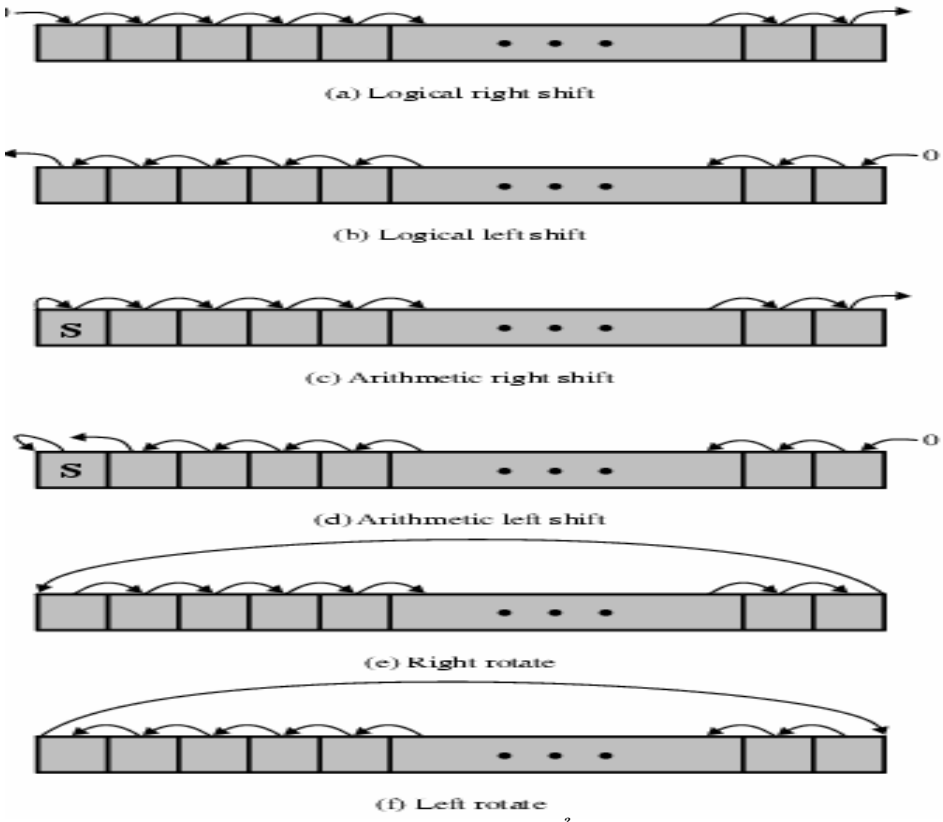
- Lệnh logic: thực hiện phép tính logic cho từng bit một.

AND (lệnh VÀ)

OR (lệnh HOẶC)

XOR (lệnh HOẶC LOẠI)

NEG (lệnh lấy số bù 1)



Hình 2.5: Minh hoạ lệnh dịch chuyển và quay vòng

- Các lệnh dịch chuyển số học hoặc logic (SHIFT), quay vòng (ROTATE) có hoặc không có số giữ ở ngã vào, sang phải hoặc sang trái. Các lệnh này được thực hiện trên một thanh ghi và kết quả lưu giữ trong thanh ghi khác. Số lần dịch chuyển (mỗi lần dịch sang phải hoặc sang trái một bit) thường được xác định trong thanh ghi thứ ba. Hình 2.5 minh hoạ cho các lệnh này.

Cho các kiến trúc kiểu RISC, ta có :

SLL (shift left logical : dịch trái logic)
SRL (shift right logical : dịch phải logic)
SRA (shift right arithemtic : dịch phải số học)

* Lệnh có điều kiện

Lệnh có điều kiện có dạng :

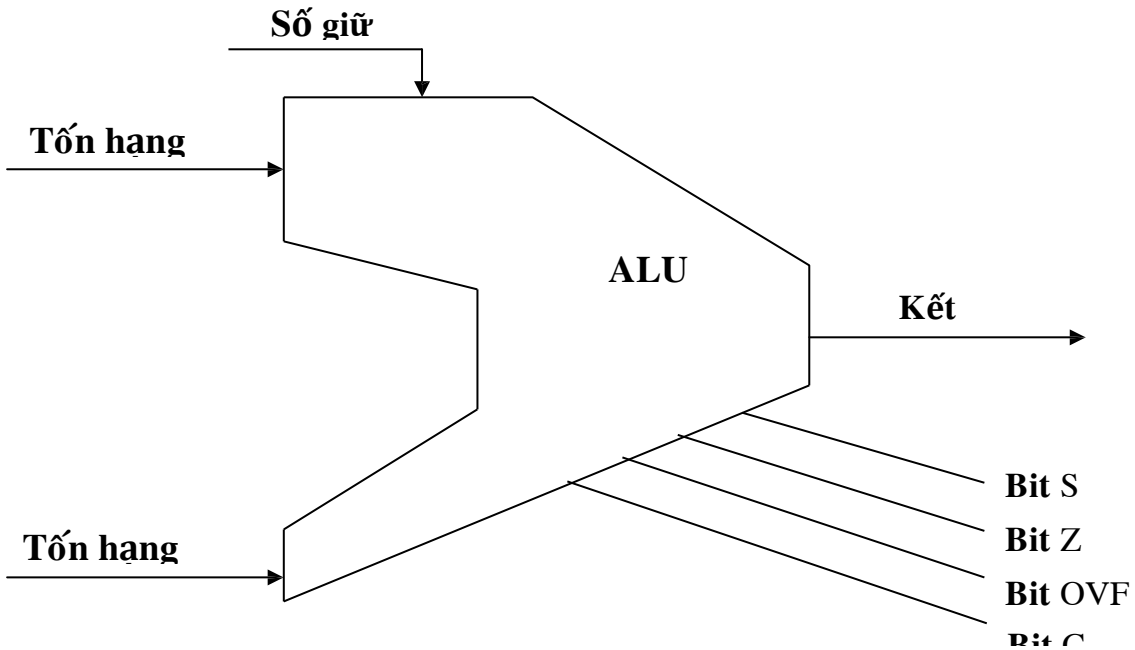
Nếu <điều kiện> thì <chuỗi lệnh 1> nếu không <chuỗi lệnh 2>
(IF <condition> THEN <instructions1> ELSE <instructions2>)

Lệnh này buộc phải ghi nhớ điều kiện và nhảy vòng nếu điều kiện được thoả.

a) Ghi nhớ điều kiện .

Bộ làm tính ALU cung cấp kết quả ở ngã ra tuỳ theo các ngã vào và phép tính cần làm. Nó cũng cho một số thông tin khác về kết quả dưới dạng các bit trạng thái. Các bit này là những đại lượng logic *ĐÚNG* hoặc *SAI* (Hình 2.6).

Trong các bit trạng thái ta có bit dấu *S* (Sign - Đúng nếu kết quả âm), bit trắc nghiệm zero *Z* (Zero - Đúng nếu kết quả bằng không), bit tràn *OVF* (Overflow) ĐÚNG nếu phép tính số học làm thanh ghi không đủ khả năng lưu trữ kết quả, bit số giữ *C* (carry) ĐÚNG nếu số giữ ở ngã ra là 1 .... Các bit trên thường được gọi là bit mã điều kiện.



Hình 2.6: Bit trạng thái mà ALU tạo ra

Có hai kỹ thuật cơ bản để ghi nhớ các bit trạng thái:

Cách thứ nhất, ghi các trạng thái trong một thanh ghi đa dụng.

Ví dụ lệnh *CMP Rk, Ri, Rj*

Lệnh trên sẽ làm phép tính trừ Ri - Rj mà không ghi kết quả phép trừ, mà lại ghi các bit trạng thái vào thanh ghi *Rk*. Thanh ghi này được dùng cho một lệnh nhảy có điều kiện. Điểm lợi của kỹ thuật này là giúp lưu trữ nhiều trạng thái sau nhiều phép tính để dùng về sau. Điểm bất lợi là phải dùng một thanh ghi đa dụng để ghi lại trạng thái sau mỗi phép tính mà số thanh ghi này lại bị giới hạn ở 32 trong các bộ xử lý hiện đại.

Cách thứ hai, là để các bit trạng thái vào một thanh ghi đặc biệt gọi là *thanh ghi trạng thái*. Vấn đề lưu giữ nội dung thanh ghi này được giải quyết bằng nhiều cách. Trong kiến trúc SPARC, chỉ có một số giới hạn lệnh được phép thay đổi thanh ghi trạng thái ví dụ như lệnh ADDCC, SUBCC (các lệnh này thực hiện các phép tính cộng ADD và phép tính trừ SUB và còn làm thay đổi thanh ghi trạng thái). Trong kiến trúc PowerPC, thanh ghi trạng thái được phân thành 8 trường, mỗi trường 4 bit, vậy là thanh ghi đã phân thành 8 thanh ghi trạng thái con.

b) Nhảy vòng

Các lệnh nhảy hoặc nhảy vòng có điều kiện, chỉ thực hiện lệnh nhảy khi điều kiện được thoả. Trong trường hợp ngược lại, việc thực hiện chương trình được tiếp tục với lệnh sau đó. Lệnh nhảy xem xét thanh ghi trạng thái và chỉ nhảy nếu điều kiện nêu lên trong lệnh là đúng.

Chúng ta xem một ví dụ thực hiện lệnh nhảy có điều kiện.

Giả sử trạng thái sau khi bộ xử lý thi hành một tác vụ, được lưu trữ trong thanh

ghi, và bộ xử lý thi hành các lệnh sau :

1. *CMP R4, R1, R2* : So sánh R1 và R2 bằng cách trừ R1 cho R2 và lưu giữ trạng thái trong R4
2. *BGT R4, +2* : Nhảy bỏ 2 lệnh nếu R1 > R2
3. *ADD R3, R0, R2* : R0 có giá trị 0. Chuyển nội dung của R2 vào R3
4. *BRA +1* : nhảy bỏ 1 lệnh
5. *ADD R3, R0, R1* : chuyển nội dung R1 vào R3
6. *Lệnh kế*

Nếu R1 > R2 thì chuỗi lệnh được thi hành là 1, 2, 5, 6 được thi hành, nếu không thì chuỗi lệnh 1, 2, 3, 4, 6 được thi hành.

Chuỗi các lệnh trên , trong đó có 2 lệnh nhảy, thực hiện công việc sau đây :

Nếu R1 > R2 thì R3 = R1 nếu không R3 = R2

Các lệnh nhảy làm tốc độ thi hành lệnh chậm lại, trong các CPU hiện đại dùng kỹ thuật ống dẫn. Trong một vài bộ xử lý người ta dùng lệnh di chuyển có điều kiện để tránh dùng lệnh nhảy trong một vài trường hợp. Thí dụ trên đây có thể được viết lại :

1. *CMP R4, R1, R2* : So sánh R1 và R2 và để các bit trạng thái trong R4.
2. *ADD R3, R0, R2* : Di chuyển R2 vào R3
3. *MGT R4, R3, R1* : (MGT : Move if greater than). Nếu R1 > R2 thì di chuyển R1 vào R3

* Vòng lặp

Các lệnh vòng lặp có thể được thực hiện nhờ lệnh nhảy có điều kiện mà ta đã nói ở trên. Trong trường hợp này, ta quản lý số lần lặp lại bằng một bộ đếm vòng lặp, và người ta kiểm tra bộ đếm này sau mỗi vòng lặp để xem đã đủ số vòng cần thực hiện hay chưa.

Bộ xử lý PowerPC có một lệnh quản lý vòng lặp

BNCT Ri, độ dời

Với thanh ghi Ri chứa số lần lặp lại. Lệnh này làm các công việc sau:

Ri := Ri -1

Nếu Ri <> 0, PC := PC + độ dời. Nếu không thì tiếp tục thi hành lệnh kế.

* Thâm nhập bộ nhớ 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 lấy 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
```

* Các thủ tục

Các thủ tục được gọi từ bất cứ nơi nào của chương trình nhờ lệnh gọi thủ tục CALL. Để khi chấm dứt việc thi hành thủ tục thì chương trình gọi được tiếp tục bình thường, ta cần lưu giữ địa chỉ trở về tức địa chỉ của lệnh sau lệnh gọi thủ tục CALL. Khi chấm dứt thi hành thủ tục, lệnh trở về RETURN nạp địa chỉ trở về vào PC.

Trong các kiến trúc CISC (VAX 11, 80x86, 680x0), địa chỉ trở về được giữ ở ngăn xếp. Trong các kiến trúc RISC, một thanh ghi đặc biệt (thường là thanh ghi R31) được dùng để lưu giữ địa chỉ trở về.

Lệnh gọi thủ tục là một lệnh loại JMPL Ri, lệnh này làm các tác vụ: R31 := PC ; để địa chỉ trở về trong R31

PC := Ri ; nhảy tới địa chỉ của thủ tục nằm trong thanh ghi Ri

Lệnh trở về khi chấm dứt thủ tục là JMP R31, vì thanh ghi R31 chứa địa chỉ trở về.

Việc dùng một thanh ghi đặc biệt để lưu trữ địa chỉ trở về là một giải pháp chỉ áp dụng cho các thủ tục cuối cùng, nghĩa là cho thủ tục không gọi thủ tục nào cả. Để có thể cho các thủ tục có thể gọi một thủ tục khác, ta có hai giải pháp:

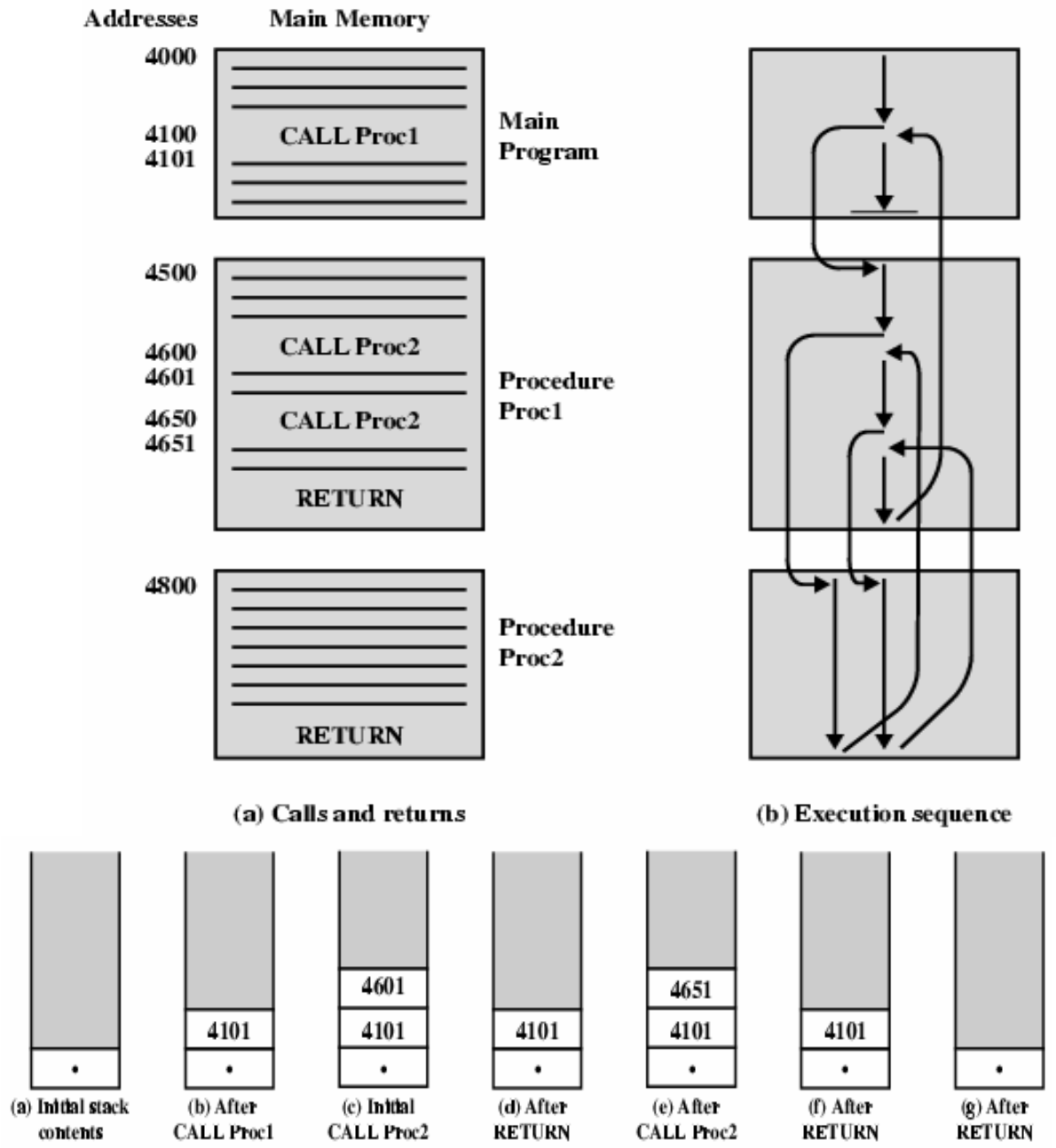
Giải pháp 1: có nhiều thanh ghi để lưu trữ địa chỉ trở về
Giải pháp 2: lưu giữ địa chỉ trở về ở ngăn xếp.

Việc gọi thủ tục có thể được thực hiện bằng chuỗi lệnh sau đây :

```
ADDI  R30, R30,4    ; R30 là con trỏ ngăn xếp
STORE  R31, (R30) ; lưu giữ địa chỉ trở về
JMPL  Ri                ;  gọi thủ tục
```

Người ta dùng chuỗi lệnh sau đây để trở về chương trình gọi :

```
LOAD  R31, (R30)   ;  phục hồi địa chỉ trở về
SUBI  R30, R30,4    ;  cập nhật con trỏ  ngăn xếp
JMP R31                 ;  trở về chương trình gọi
```



Hình 2.7: Gọi thủ tục và trở về khi thực hiện xong thủ tục

Việc truyền tham số từ thủ tục gọi đến thủ tục bị gọi có thể thực hiện bằng cách dùng các thanh ghi của bộ xử lý hoặc dùng ngăn xếp. Nếu số tham số cần truyền ít, ta dùng các thanh ghi.

3.2.2 Tập lệnh của bộ xử lý 8086/8088

* Chip 8086

Chip 8086 bắt đầu được phát triển từ năm 1978 và đưa ra thị trường năm 1980. Đây là bộ xử lý 16 bit, các thanh ghie bên trong 16 bit và nó xử lý 16 bit số liệu cùng một lúc.

* chip 8088

Ra đời sau chip 8086, nó có cấu trúc bên trong và tập lệnh hoàn toàn giống 8086, chỉ khác ở kênh truyền số liệu với thế giới bên ngoài.

Tập lệnh của các bộ xử lý 8086/8088 chỉ có thể làm các phép tính số học với số nguyên, không thực hiện được các phép tính số học với số dấu phẩy động (số thực) một cách trực tiếp. Khi cần làm các phép tính số học với số dấu phẩy động thì có thể sử dụng giải pháp phần mềm, bằng cách lập các chương trình con.Đây là giải pháp được dùng nhiều nhất, chương trình con có thể tạo ra các kết quả dấu phẩy động bởi các phép tính logic và số học trên các số nguyên.

CÂU HỎI VÀ BÀI TẬP

1. Các thành phần của một máy tính.
2. Sự khác biệt giữa CPU RISC và CPU CISC?
3. Nêu khái niệm mã lệnh, lệnh máy.
4. Cho ví dụ minh họa lời gọi thủ tục.
5. Trình bày tập lệnh của bộ xử lý 8086/8088.

CHƯƠNG 3: TỔ CHỨC BỘ VI XỬ LÝ

Mã chương:..M3

Mục tiêu của bài

- *Hiểu được nhiệm vụ và cách tổ chức đường đi của dữ liệu trong bộ xử lý*
- *Hiểu nguyên tắc vận hành của bộ điều khiển mạch điện tử*
- *Hiểu nguyên tắc vận hành của bộ điều khiển vi chương trình*
- *Hiểu nhiệm vụ của ngắt*
- *Hiểu được tiến trình thi hành lệnh mã máy*
- *Biết một số kỹ thuật xử lý thông tin: ống dẫn, siêu ống dẫn*

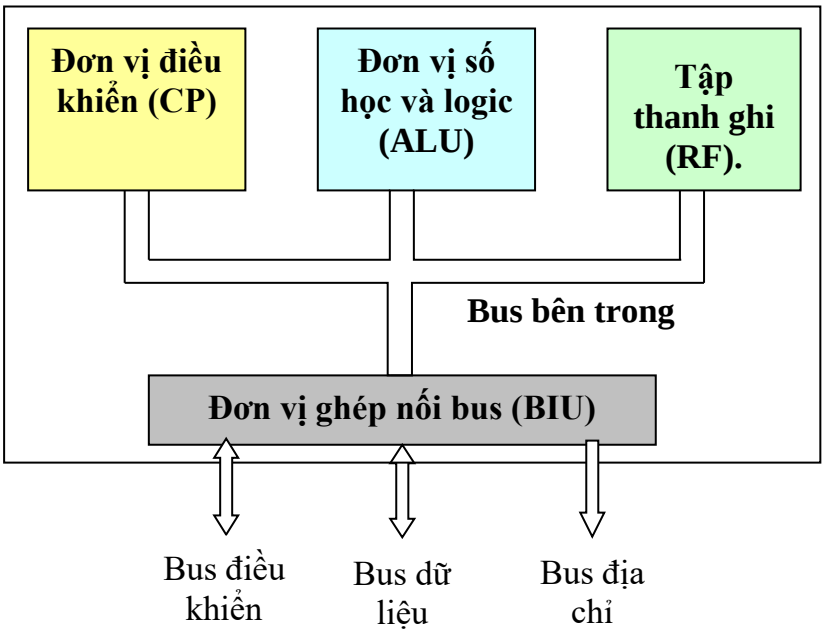
1.Sơ đồ khối của bộ xử lý

Mục tiêu:*nắm được sơ đồ khối các phần thành bên trong bộ xử lý*

Một bộ phận không thể thiếu đối với bất kỳ máy tính nào đó là bộ vi xử lý, bộ phận điều khiển mọi hoạt động của máy tính. Lịch sử phát triển các thế hệ máy vi tính gắn liền với sự phát triển các vi xử lý như: 8088, 80286, 80386, 80486, Pentium… Trong các máy tính cá nhân thường chỉ sử dụng 1 vi xử lý, đây chính là trung tâm xử lý thông tin và phát ra các tín hiệu điều khiển. Do vậy, chức năng của vi xử lý ở đây cũng chính là CPU (Control Processing Unit).

Trước khi tìm hiểu cấu trúc của một CPU, chúng ta hãy xem xét kỹ hơn các bước mà CPU cần làm khi thực hiện một lệnh. Đó là:

- Nhận lệnh: CPU nhận lệnh từ bộ nhớ.
- Dịch lệnh: Là bước để CPU giải mã xem công việc cụ thể phải thực hiện là gì để tương ứng với lệnh đã được nhận.
- Nhận dữ liệu: Bước này là cần thiết khiyeeu cầu của lệnh phải xử lý dữ liệu từ nơi khác như bộ nhớ hay các Module I/O (để kết nối với các thiết bị bên ngoài).
- Xử lý dữ liệu: Bước này được thực hiện khi yêu cầu của lệnh phải xử lý dữ liệu nhận được.
    - Ghi dữ liệu: Bước này thường là cần thiết để CPU lưu kết quả thực hiện ra bộ nhớ hay Module I/O.
    - Với một loạt nhiệm vụ mà CPU cần phải thực hiện ở trên, chúng ta thấy rằng bên trong CPU phải có đơn vị điều khiển CU (Control Unit) để điều khiển hoạt động chung của nó. Bên cạnh đó, phải có bộ phận xử lý dữ liệu để thực hiện các phép tính số học và lôgic, đây chính là đơn vị số học và lôgic ALU (Arithmetic and Logic Unit). Một bộ phận không thể thiếu trong CPU là tập các thanh ghi bên trong (registers), nơi lưu trữ và xử lý thông tin. Xuất phát từ đó mà các CPU đều có sơ đồ khối biểu diễn như hình dưới đây:



Hình 3.1: Sơ đồ khối chung của CPU

2. Đường dẫn dữ liệu

Mục tiêu: *Hiểu được nhiệm vụ và cách tổ chức đường đi của dữ liệu trong bộ xử lý*

2.1. Các thành phần đường dẫn dữ liệu

Phần đường dẫn dữ liệu gồm:

- Đơn vị số học và lôgic (ALU: Arithmetic and Logic Unit).
- Các mạch dịch
- Các thanh ghi
- Các đường nối kết các bộ phận trên.

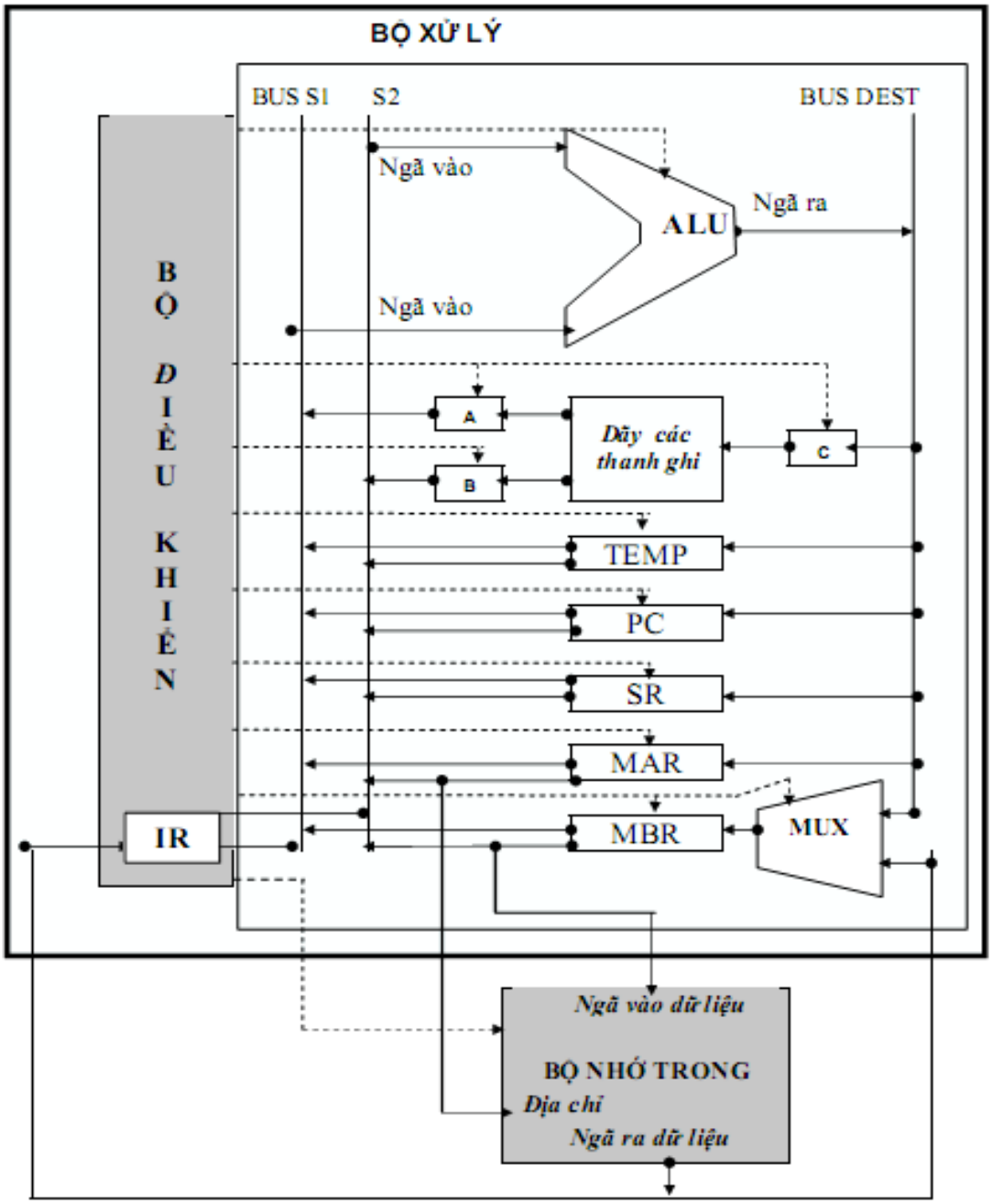
Phần này chứa hầu hết các trạng thái của bộ xử lý. Ngoài các thanh ghi tổng quát, phần đường dẫn dữ liệu còn chứa thanh ghi đếm chương trình (PC: Program Counter), thanh ghi trạng thái (SR: Status Register), thanh ghi đệm TEMP (Temporary), các thanh ghi địa chỉ bộ nhớ (MAR: Memory Address Register), thanh ghi số liệu bộ nhớ (MBR: Memory Buffer Register), bộ đa hợp (MUX: Multiplexor), đây là điểm cuối của các 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).

2.2. Nhiệm vụ của đường dẫn dữ liệu

Nhiệm vụ chính của phần đường dẫn 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 là 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 3.2: Tổ chức của một xử lý điển hình
(Các đường không liên tục là các đường điều khiển)

3. Bộ điều khiển

Mục tiêu: *Hiểu nguyên tắc vận hành của bộ điều khiển mạch điện tử*
Hiểu nguyên tắc vận hành của bộ điều khiển vi chương trình

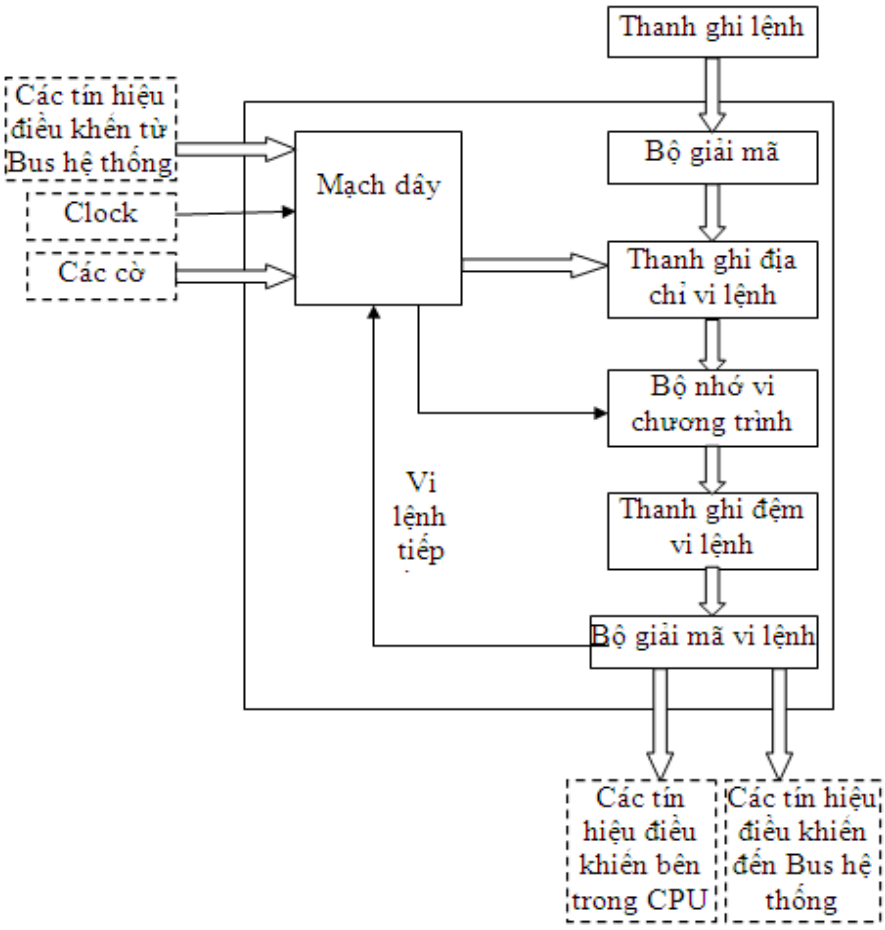
3.1. Chức năng bộ điều khiển

- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh
- Tăng nội dung của PC để trỏ sang lệnh kế tiếp
- Giải mã lệnh đã được nhận để xác định thao tác mà lệnh yêu cầu
- Phát ra các tín hiệu điều khiển thực hiện lệnh
- Nhận các tín hiệu yêu cầu từ bus hệ thống và đáp ứng với các yêu cầu đó.

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 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 hiệu giúp các lệnh được thực hiện một cách tuần tự.

3.2. Các phương pháp thiết kế bộ điều khiển

3.2.1 Bộ điều khiển vi chương trình (Microprogrammed Control Unit)

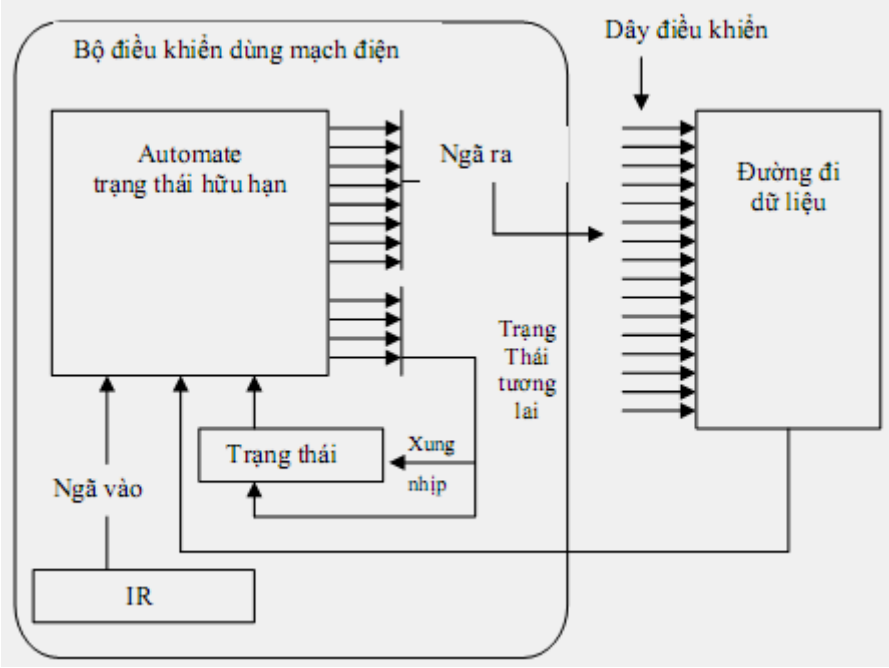


Hình 3.3: Nguyên tắc vận hành của bộ điều khiển dùng vi chương trình

– Bộ nhớ vi chương trình (ROM) lưu trữ các vi chương trình (microprogram)
– Một vi chương trình bao gồm các vi lệnh (microinstruction)
– Mỗi vi lệnh mã hoá cho một vi thao tác (microoperation)
– Để hoàn thành một lệnh cần thực hiện một hoặc một vài vi chương trình
– Tốc độ chậm

3.2.2. Bộ điều khiển dùng 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 3.4: Nguyên tắc vận hành của bộ điều khiển dùng mạch điện tử

Hình 3.4 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.

4. Tiến trình thực hiện lệnh máy

- ***Mục tiêu:*** *Hiểu được tiến trình thi hành lệnh mã máy*

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).

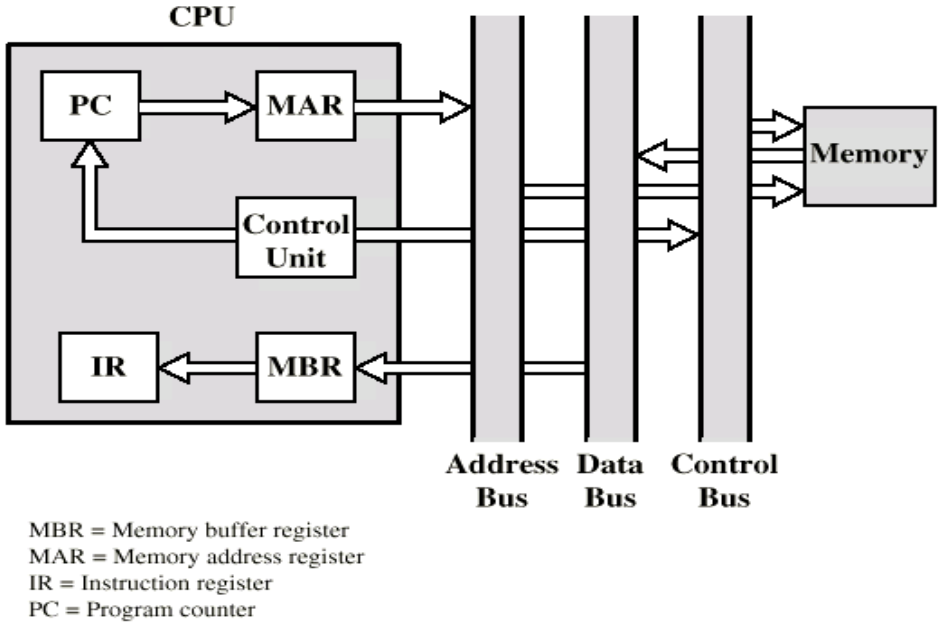
Mỗi giai đoạn được thi hành trong một hoặc nhiều chu kỳ xung nhịp.

4.1. Đọc lệnh

MAR ← PC

IR ← M[MAR]

- Thanh ghi PC chứa địa chỉ lệnh tiếp theo
- Địa chỉ chuyển vào thanh ghi MAR
- Địa chỉ đưa lên bus địa chỉ
- Đơn vị điều khiển yêu cầu đọc bộ nhớ
- Kết quả đưa lên data bus, sao chép vào thanh ghi MBR, đưa vào thanh ghi IR



Hình 3.5:Sơ đồ mô tả quá trình đọc lệnh

4.2. Giải mã lệnh

– Lệnh từ thanh ghi lệnh IR được đưa đến đơn vị điều khiển
– Đơn vị điều khiển tiến hành giải mã lệnh để xác định thao tác phải thực hiện
– Giải mã lệnh xảy ra bên trong CPU

4.3. Thi hành lệnh

– Có nhiều dạng tuỳ thuộc vào lệnh
– Có thể là:
  ✓ Đọc/Ghi bộ nhớ
  ✓ Vào/Ra
  ✓ Chuyển giữa các thanh ghi
  ✓ Thao tác số học/logic
  ✓ Chuyển điều khiển (rẽ nhánh)
  ✓ ...

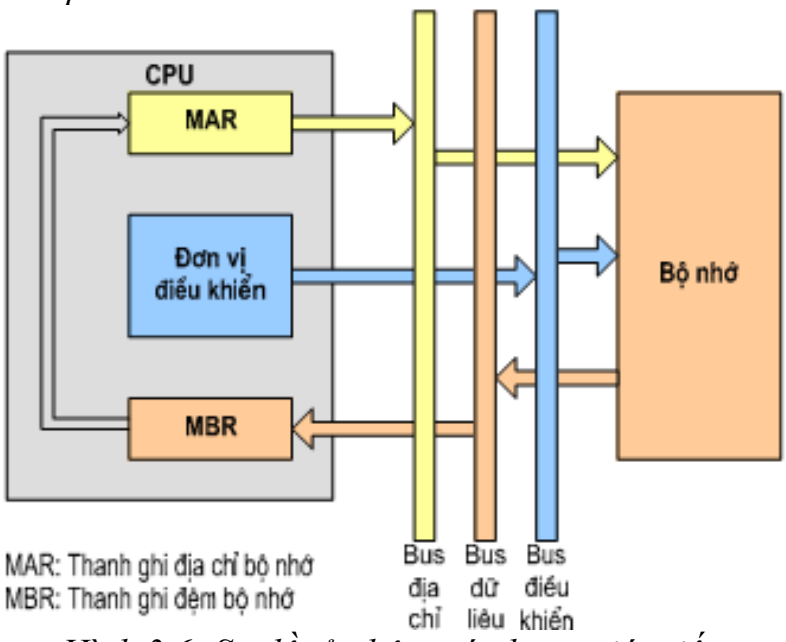
4.4. Thâm nhập bộ nhớ trong

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

Nhận dữ liệu trực tiếp:

– CPU đưa địa chỉ của toán hạng ra bus địa chỉ
– CPU phát tín hiệu điều khiển đọc
– Toán hạng được đọc vào CPU
– Tương tự như nhận lệnh

Nhận dữ liệu gián tiếp:



Hình 3.6: Sơ đồ tả nhận toán hạng gián tiếp

Quá trình nhận dữ liệu gián tiếp:

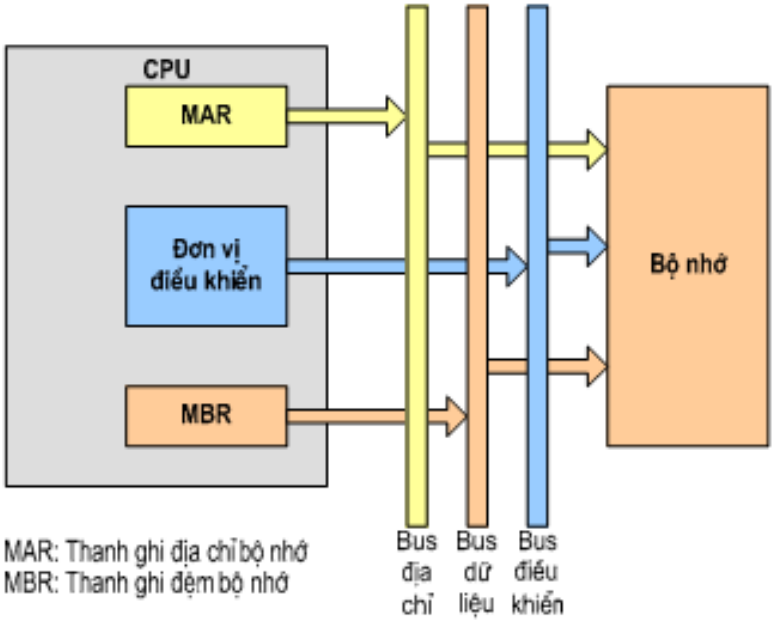
– CPU đưa địa chỉ ra bus địa chỉ

- CPU phát tín hiệu điều khiển đọc
- Nội dung ngăn nhớ được đọc vào CPU, đó chính là địa chỉ của toán hạng
- Địa chỉ này được CPU phát ra bus địa chỉ để tìm ra toán hạng
- CPU phát tín hiệu điều khiển đọc
- Toán hạng được đọc vào CPU

4.5 Lưu trữ kết quả

Rd ← Ngã ra ALU hoặc Rd ← MBR

- CPU đưa địa chỉ ra bus địa chỉ
- CPU đưa dữ liệu cần ghi ra bus dữ liệu
- CPU phát tín hiệu điều khiển ghi
- Dữ liệu trên bus dữ liệu được copy đến vị trí xác định → Lưu trữ kết quả trong thanh ghi đích.



Hình 3.7: Sơ đồ mô tả quá trình lưu kết quả

5. Kỹ thuật ống dẫn lệnh

Mục tiêu: *Biết kỹ thuật xử lý thông tin: 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 khoảng 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).

Hình 3.8 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).

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

Hình 3.8: Các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc

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:

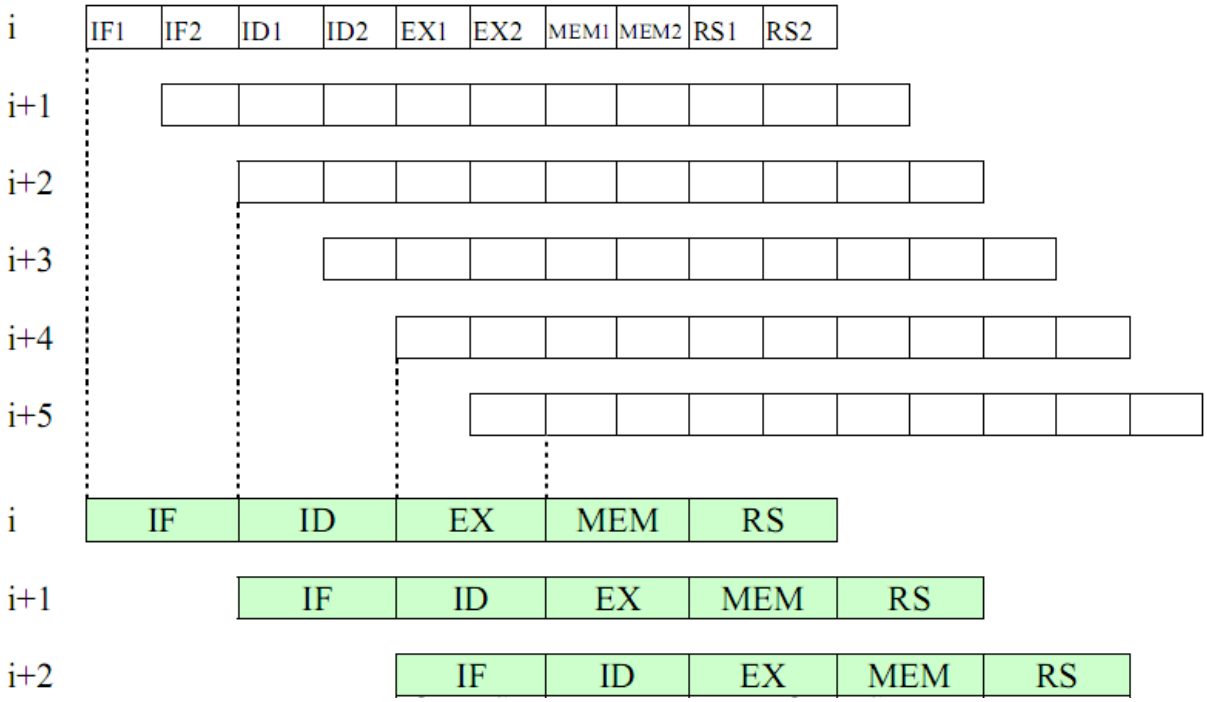
- Cần phải có một mạch điện để thi hành mỗi giai đoạn của lệnh vì tất cả các giai đoạn của lệnh được thi hành cùng lúc. Trong một bộ xử lý không dùng kỹ thuật ống dẫn, ta có thể dùng bộ làm toán ALU để cập nhật thanh ghi PC, cập nhật địa chỉ của toán hạng bộ nhớ, địa chỉ ô nhớ mà chương trình cần nhảy 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 3.8, 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.

6. Kỹ thuật siêu ống dẫn lệnh

Mục tiêu: *Biết 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 3.9 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 làm 1 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 3.9, 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 3.9: 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 trong máy có kỹ thuật ống dẫn đơn giản.

7. Các chướng ngại của ống dẫn lệnh

***Mục tiêu**: biết được các khó khăn khi sử dụng kĩ thuật ống dẫn và cách khắc phục*

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 (chướng ngại): *khó khăn do cấu trúc, khó khăn do số liệu* và *khó khăn do điều khiển*.

7.1. Chướng ngại do cấu trúc

Xung đột cấu trúc xảy ra khi có 2 lệnh cùng cố gắng sử dụng cùng một

nguồn tại cùng một thời điểm, có thể là cùng ghi kết quả vào một thanh ghi, cùng truy cập vào một ô nhớ, cùng yêu cầu một bộ tính toán số học, hoặc khi việc nạp lệnh và đọc dữ liệu từ bộ nhớ diễn ra cùng một lúc.

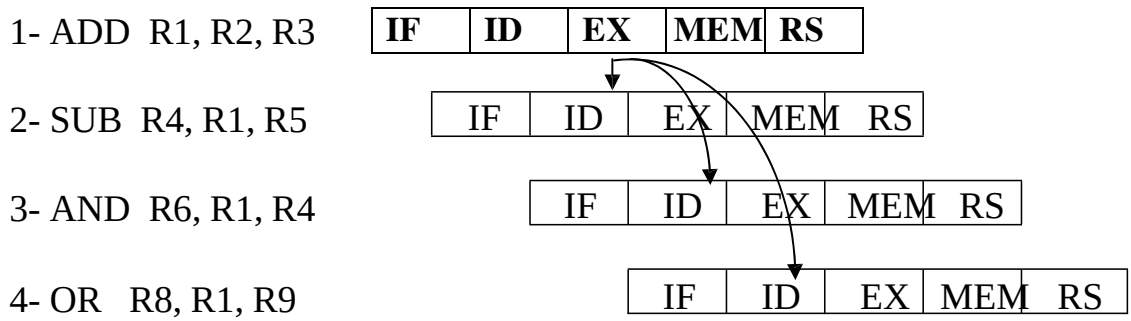
Để khắc phục xung đột kiểu này ta thường sử dụng cách chèn trễ vào giữa các chu kì lệnh hoặc tổ chức lại các lệnh.

7.2. Chướng ngại do dữ liệu

Lấy ví dụ trường hợp các lệnh liên tiếp sau:

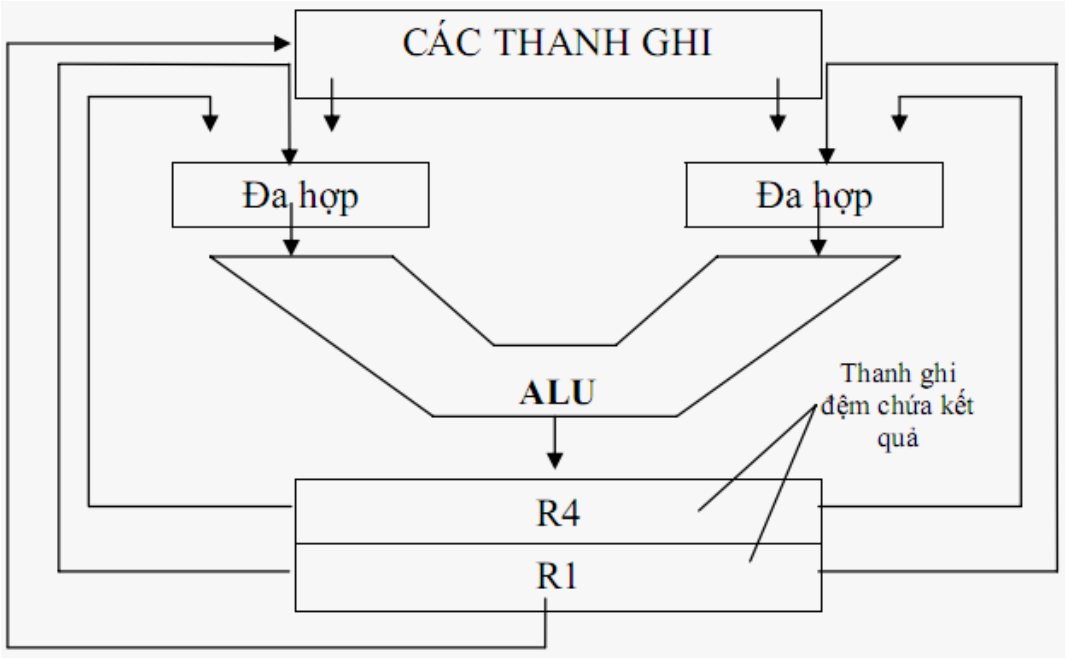
Lệnh 1: *ADD R1, R2, R3*
Lệnh 2: *SUB R4, R1, R5*
Lệnh 3: *AND R6, R1, R7*
Lệnh 4: *OR R8, R1, R9*

Hình 3.10 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 3.10: 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 3.11.



Hình 3.11: 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.

Ta có thể hạn chế xung đột dữ liệu bằng cách sử dụng đường dữ liệu nội đặc biệt. Ngoài ra có thể kết hợp với cả việc tổ chức lại các lệnh trong chương trình.

7.3. Chướng ngại 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 nhảy tới chỉ được bắt đầu ở chu kỳ C+2. Ngoài ra, phải biết địa chỉ cần nhảy đến mà ta có 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.

8. Các loại ngắt

Mục tiêu: *Hiểu nhiệm vụ của ngắt, phân biệt được các loại ngắt.*

8.1. Ngắt

Khái niệm chung về ngắt: Ngắt là cơ chế cho phép CPU tạm dừng chương trình đang thực hiện để chuyển sang thực hiện một chương trình khác, gọi là chương trình con phục vụ ngắt.

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.

8.2. Các loại ngắt

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:

- Ngắt do lỗi khi thực hiện chương trình, ví dụ: tràn số, chia cho 0.
- Ngắt do lỗi phần cứng, ví dụ lỗi bộ nhớ RAM.
- Ngắt do mô-đun vào-ra phát tín hiệu ngắt đến CPU yêu cầu trao đổi dữ liệu.
- Người lập trình muốn dùng dịch vụ của hệ điều hành.
- 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 điện bị cắt.

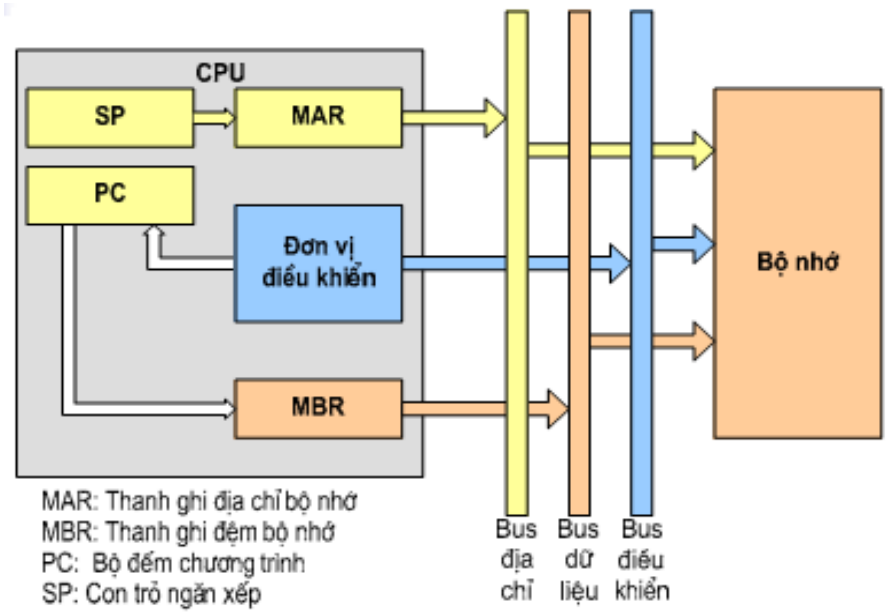
Dù rằng ngắt quãng không xảy ra thường xuyên nhưng bộ xử lý phải được thiết kế sao cho có thể lưu giữ trạng thái của nó trước khi nhảy đi phục vụ ngắt quãng. Sau khi thực hiện xong chương trình phục vụ ngắt, bộ xử lý phải khôi phục trạng thái của nó để có thể tiếp tục công việc.

8.3. Hoạt động của ngắt

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.

Sơ đồ mô tả hoạt động ngắt



Hình 3.12: Sơ đồ mô tả chu trình ngắt

- Nội dung của bộ đếm chương trình PC (địa chỉ trở về sau khi ngắt) được đưa ra bus dữ liệu.
- CPU đưa địa chỉ (thường được lấy từ con trỏ ngăn xếp SP) ra bus địa chỉ.
- CPU phát tín hiệu điều khiển ghi bộ nhớ.
- Địa chỉ trở về trên bus dữ liệu được ghi ra vị trí xác định (ở ngăn xếp).
- Địa chỉ lệnh đầu tiên của chương trình con điều khiển ngắt được nạp vào PC.

CÂU HỎI VÀ BÀI TẬP

1. Các thành phần và nhiệm vụ của đường đi dữ liệu?
2. Thế nào là ngắt quãng? Các giai đoạn thực hiện ngắt quãng của CPU.
3. Vẽ hình để mô tả kỹ thuật ống dẫn.
4. Các khó khăn trong kỹ thuật ống dẫn và cách giải quyết khó khăn này.

CHƯƠNG 4: HỆ THỐNG NHỚ

Mã chương:..M4

Mục tiêu

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.Phân loại bộ nhớ

Mục tiêu: *Hiểu được các cấp bộ nhớ và cách thức vận hành*

1.1. Phân loại bộ nhớ theo phương pháp truy nhập

Bộ nhớ chứa chương trình, nghĩa là chứa lệnh và số liệu. Người ta phân biệt bộ nhớ theo truy nhập như sau:

- Bộ nhớ truy nhập ngẫy nhiên: Đây là loại bộ nhớ mà khi ta muốn truy nhập đến một phần tử bất kỳ của nó, không cần phải truy nhập lần lượt qua tất cả các phần tử đứng trước nó. Chính vì vậy mà thời gian truy nhập đến các phần tử nhớ trong trường hợp này không phụ thuộc vào vị trí của các phần tử nhớ (đĩa cứng,...).
- Bộ nhớ truy nhập tuần tự: Đây là loại bộ nhớ mà khi chúng ta muốn truy nhập đến một phần tử bất kỳ của nó thì phải truy nhập lần lượt qua tất cả các phần tử nhớ trước nó.

1.2.Phân loại theo đọc ghi của bộ nhớ

Tùy theo chức năng mà bộ nhớ có thể có những khả năng đọc/ghi thông tin khác nhau.

- Có những loại bộ nhớ chỉ có thể đọc thông tin từ chúng mà không thể ghi thông tin ra chúng thường gọi là ROM (Read Only Memory).
- Có những loại bộ nhớ vừa có thể đọc thông tin lại vừa có thể ghi thông tin ra chúng, thường gọi là RAM (Random Access Memory).

2. Các loại bộ nhớ bán dẫn

Mục tiêu: *Nắm được đặc điểm và các loại bộ nhớ bán dẫn. Biết tổ chức chíp nhớ và cách tăng dung lượng bộ nhớ.*

2.1.ROM (Read Only Memory)

2.1.1. Đặc điểm ROM

Bộ nhớ chỉ đọc ROM cũng được chế tạo bằng công nghệ bán dẫn. Bộ nhớ mà các phần tử nhớ của nó có trạng thái cố định, thông tin lưu giữ trong ROM cũng cố định và thậm chí không bị mất ngay cả khi mất điện. Chương trình trong ROM được viết vào lúc chế tạo nó. ROM là bộ nhớ không khả biến .
Lưu trữ các thông tin sau:

- Thư viện các chương trình con
- Các chương trình điều khiển hệ thống (BIOS)
- Các bảng chức năng
- Vi chương trình

2.1.2. Các loại ROM

- ROM mặt nạ: thông tin được ghi khi sản xuất, rất đắt.
- PROM (Programmable ROM): Cần thiết bị chuyên dụng để ghi bằng chương trình → chỉ ghi được một lần.
- EPROM (Erasable PROM): Cần thiết bị chuyên dụng để ghi bằng chương trình → ghi được nhiều lần. Trước khi ghi lại, xóa bằng tia cực tím.
- EEPROM (Electrically Erasable PROM): Có thể ghi theo từng byte, xóa bằng điện.
- Flashmemory (Bộ nhớ cực nhanh): Ghi theo khối, xóa bằng điện.

2.2.RAM (Random Access Memory)

2.2.1. Đặc điểm

RAM là một loại bộ nhớ chính của máy tính. RAM được gọi là bộ nhớ truy cập ngẫu nhiên vì nó có đặc tính: thời gian thực hiện thao tác đọc hoặc ghi đối với mỗi ô nhớ là như nhau, cho dù đang ở bất kỳ vị trí nào trong bộ nhớ. Mỗi ô nhớ của RAM đều có một địa chỉ. Thông thường, mỗi ô nhớ là một byte (8 bit); tuy nhiên hệ thống lại có thể đọc ra hay ghi vào nhiều byte (2, 4, 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).

Mục đích: Máy vi tính sử dụng RAM để lưu trữ mã chương trình và dữ liệu trong suốt quá trình thực thi. Đặc trưng tiêu biểu của RAM là có thể truy cập vào những vị trí khác nhau trong bộ nhớ và hoàn tất trong khoảng thời gian tương tự, ngược lại với một số kỹ thuật khác, đòi hỏi phải có một khoảng thời gian trì hoãn nhất định.

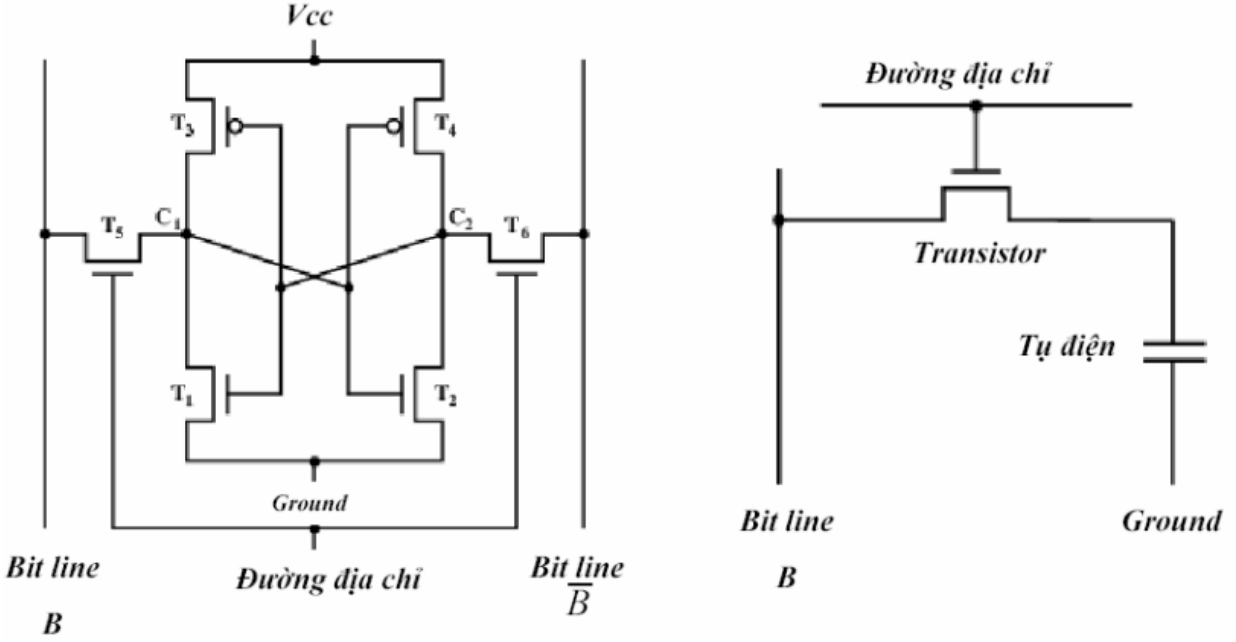
2.2.2. Các loại RAM

Tuỳ theo công nghệ chế tạo, người ta phân biệt RAM tĩnh (SRAM: Static RAM) và RAM động (Dynamic RAM).

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. Bộ nhớ DRAM chậm nhưng rẻ tiền hơn SRAM.



Hình 4.1: SRAM và DRAM

Các loại DRAM

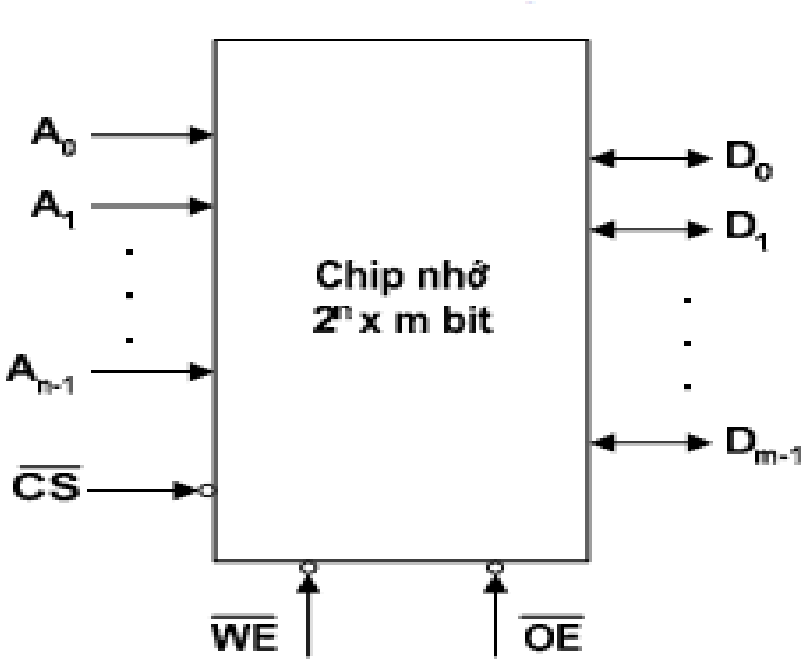
- SDRAM (Viết tắt từ Synchronous Dynamic RAM) được gọi là DRAM đồng bộ. SDRAM gồm 3 phân loại: SDR, DDR, và DDR2.
  - SDR SDRAM (Single Data Rate SDRAM), thường được giới chuyên môn gọi tắt là "SDR". Có 168 chân, có tốc độ 33Mhz, 66Mhz, 100Mhz và 133Mhz. Được dùng trong các máy vi tính cũ, bus speed chạy cùng vận tốc với clock speed của memory chip, nay đã lỗi thời.

- DDR SDRAM (Double Data Rate SDRAM), thường được giới chuyên môn gọi tắt là "DDR". Có 184 chân. DDR SDRAM là cải tiến của bộ nhớ SDR với tốc độ truyền tải gấp đôi SDR (200Mhz, 266Mhz, 333Mhz, 400Mhz,...) nhờ vào việc truyền tải hai lần trong một chu kỳ bộ nhớ. Đã được thay thế bởi DDR2. Hầu hết các mainboard đời mới đều hỗ trợ DDR (và không hỗ trợ SDRAM).
- DDR2 SDRAM (Double Data Rate 2 SDRAM), Thường được giới chuyên môn gọi tắt là "DDR2". Là thế hệ thứ hai của DDR với 240 chân, lợi thế lớn nhất của nó so với DDR là có bus speed cao gấp đôi clock speed.

❖ RDRAM (Viết tắt từ Rambus Dynamic RAM), thường được giới chuyên môn gọi tắt là "Rambus". Đây là một loại DRAM được thiết kế 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 được thực hiện giữa các mạch điều khiển và từng chip 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 Rambus đạt từ 400-800 MHz Rambus tuy không nhanh hơn SDRAM là bao nhưng lại đắt hơn rất nhiều nên có rất ít người dùng. RDRAM phải cắm thành cặp và ở những khe trống phải cắm những thanh RAM giả (còn gọi là C-RIMM) cho đủ.

2.3. Thiết kế môdun nhớ bán dẫn

- Tổ chức chíp nhớ



Hình 4.2 Tổ chức chíp nhớ

Các tín hiệu của chíp nhớ.

- Các đường địa chỉ:An-1->A0:có 2^n từ nhớ
- Các đường dữ liệu:Dn-1 ->D0: độ dài từ nhớ bằng m bit
- Dung lượng chip nhớ:
  - 2^n x m bit
- Các đường điều khiển:
  - Tín hiệu chọn chip CS (Chip Select)
  - Tín hiệu điều khiển đọc OE(output Enable)
  - Tín hiệu điều khiển ghi wE(write enable)

Dung lượng chip nhớ=2^n xm bit

- Cần thiết kế để tăng dung lượng:
  - Thiết kế tăng độ dài từ nhớ
  - Thiết kế tăng số lượng từ nhớ
  - Thiết kế kết hợp

* Tăng độ dài từ nhớ

Ví dụ :

Cho chip nhớ SRAM 4K x 4 bit

Thiết kế môdun nhớ 4K x 8 bit

Giải:

Dung lượng chip nhớ=2^{12} x 4 bit

Chip nhớ có 12 chân địa chỉ, 4 chân dữ liệu

Môdun nhớ cần có:12 chân địa chỉ, 8 chân dữ liệu

* Tăng số lượng từ nhớ

Ví dụ:

Cho chip nhớ SRAM 4K x8 bit

Thiết kế môdun nhớ 8K x 8 bit

Giải:

Dung lượng chip nhớ = 2^{12} x 8 bit

Chip nhớ có:12 chân địa chỉ, 8 chân dữ liệu

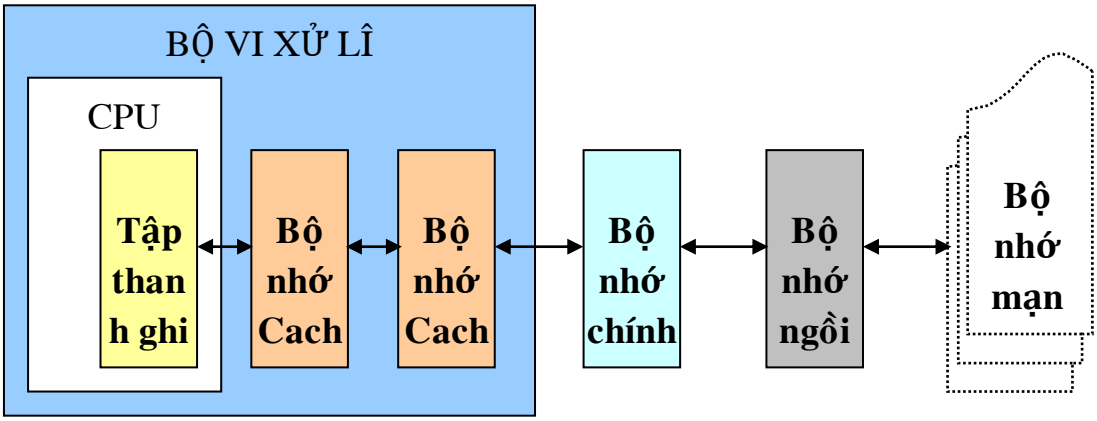
Dung lượng môdun nhớ:2^{13} x 8 bit

13 chân địa chỉ, 8 chân dữ liệu

3. Hệ thống nhớ phân cấp

Mục tiêu: *Nhận xét được các cấp bộ nhớ về dung lượng, tốc độ.*

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 4.3: Các cấp bộ nhớ

Ta nhận thấy rằng từ trái sang phải: dung lượng tăng dần, tốc độ giảm dần, giá thành/1bit giảm dần.

Máy tính lưu trữ dữ liệu cũng theo cấu trúc phân cấp tương tự. Khi các ứng dụng khởi động, dữ liệu và lệnh được chuyển từ đĩa cứng tốc độ chậm sang bộ nhớ chính (RAM động hay DRAM), nơi mà CPU có thể truy xuất nhanh hơn. DRAM hoạt động như là một vùng đệm cho đĩa.

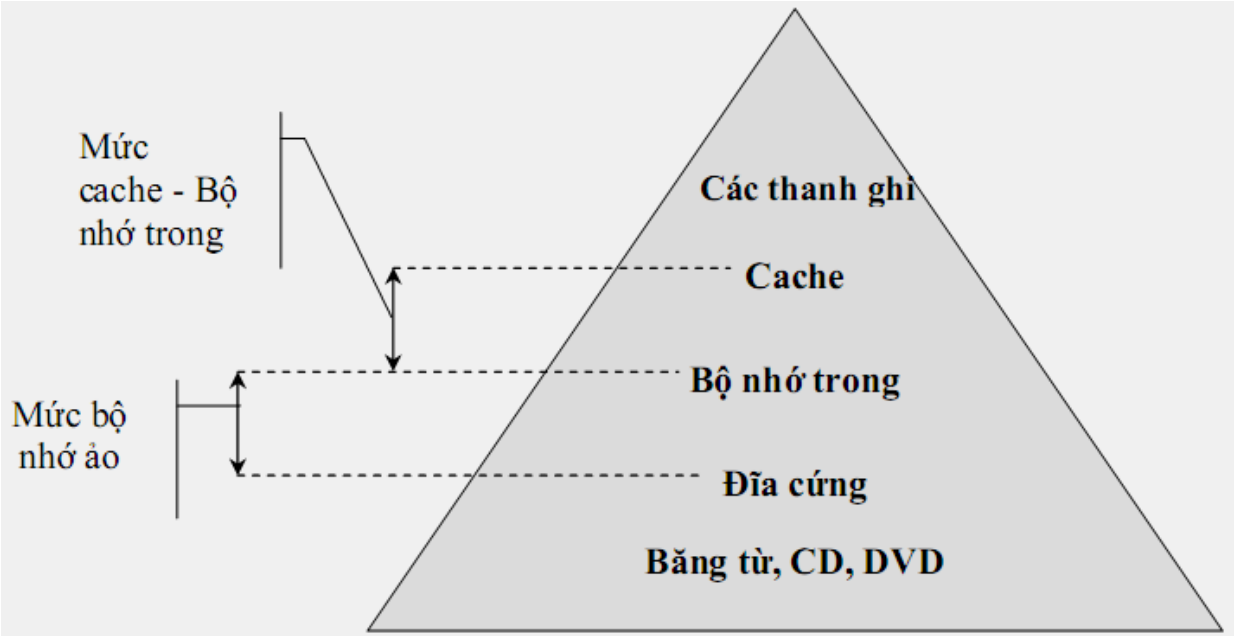
Mặc dù DRAM nhanh hơn đĩa cứng, nó vẫn còn bị hạn chế. Vì thế, dữ liệu thường dùng đến sẽ được chuyển lên một loại bộ nhớ nhanh hơn gọi là bộ nhớ đệm cấp 2 (L2). Loại bộ nhớ này có thể nằm trên RAM tĩnh cạnh bên CPU, nhưng những CPU loại mới thường kết hợp bộ nhớ đệm L2 ngay trên chip bộ xử lý.

Ở cấp cao nhất, thông tin thường sử dụng nhất, ví dụ lệnh của các vòng lặp thực thi lặp đi lặp lại, được lưu trực tiếp trong một vùng đặc biệt ngay trên bộ xử lý gọi là bộ nhớ đệm cấp 1 (L1). Đây là loại bộ nhớ nhanh nhất.

Bộ nhớ đệm L2 nằm trên CPU có tốc độ truy xuất nhanh gấp bốn lần so với trường hợp nó nằm trên chip riêng.

Khi bộ xử lý cần thực thi một câu lệnh nào đó, đầu tiên nó sẽ tìm kiếm trong thanh ghi dữ liệu của riêng nó. Nếu dữ liệu cần thiết không có ở đó, nó sẽ tìm trên bộ nhớ đệm L1 và sau đó là L2, và nếu trong bộ nhớ đệm cũng không có nó sẽ gọi đến bộ nhớ chính RAM. Cuối cùng, nếu dữ liệu vẫn không có thì hệ thống sẽ phải lấy dữ liệu này từ đĩa cứng.

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) . 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 4.4: 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.

4. Kết nối bộ nhớ với bộ xử lý

Mục tiêu: *Hiểu được nguyên tắc kết nối bộ nhớ với bộ xử lý,cách thức thâm nhập bộ nhớ.*

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.

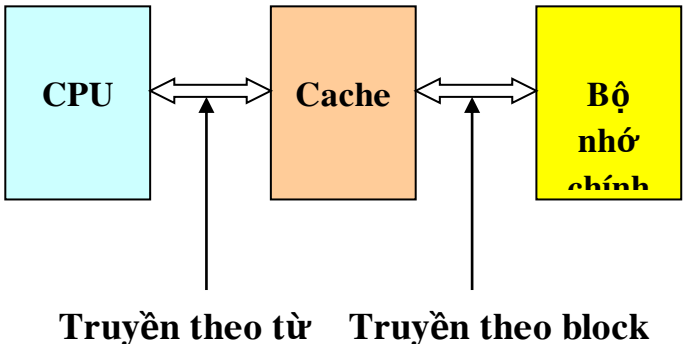
Tổ chức các cấp bộ nhớ sao cho các lệnh và dữ liệu thường dùng được nằm trong bộ nhớ cache, điều này làm tăng hiệu quả của máy tính một cách đáng kể.

5.Các tổ chức cache

Mục tiêu: *Hiểu được đặc điểm bộ nhớ cache,tổ chức bộ nhớ cache.*
Vận dụng được các phương pháp ánh xạ địa chỉ

5.1. Cache (bộ nhớ đệm nhanh)

- Cache có tốc độ nhanh hơn bộ nhớ chính
- Cache được đặt giữa CPU và bộ nhớ chính
- Nhằm tăng tốc độ truy cập bộ nhớ của CPU
- Cache có thể được đặt trên chip CPU



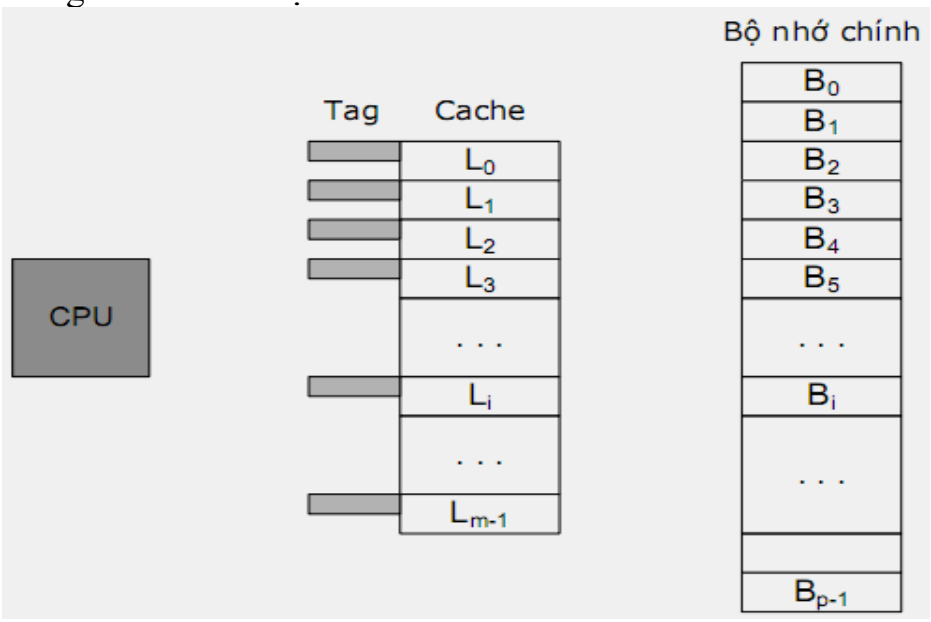
Hình 4.5: Bộ nhớ Cache

+ Ví dụ về thao tác của cache:

- CPU yêu cầu nội dung của ngăn nhớ.
- CPU kiểm tra trên cache với dữ liệu này.
- Nếu có, CPU nhận dữ liệu từ cache (nhanh).
- Nếu không có, đọc block nhớ chứa dữ liệu từ bộ nhớ chính vào cache.
- Tiếp đó chuyển dữ liệu từ cache vào CPU.

5.2. Tổ chức cache

+ Cấu trúc chung của cache / bộ nhớ chính



Hìn 4.6 Cấu trúc cache và bộ nhớ

- Bộ nhớ chính có 2^N byte nhớ
  - Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
  - Bộ nhớ chính: $B_0, B_1, B_2, ..., B_{p-1}$ (p Blocks)
  - Bộ nhớ cache: $L_0, L_1, L_2, ..., L_{m-1}$ (m Lines)
  - Kích thước của Block = 8, 16, 32, 64, 128 byte
- Một số Block của bộ nhớ chính được nạp vào các Line của cache.
- Nội dung Tag (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó.
- Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:
  - Từ nhớ đó có trong cache (cache hit)
  - Từ nhớ đó không có trong cache (cache miss)

Vì số line của cache ít hơn số block của bộ nhớ chính nên cần có một thuật giải ánh xạ thông tin trong bộ nhớ chính và cache.

5.3. Các phương pháp ánh xạ địa chỉ

a. Ánh xạ trực tiếp

Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:

$B_0 \rightarrow L_0$

$B_1 \rightarrow L_1$

.....

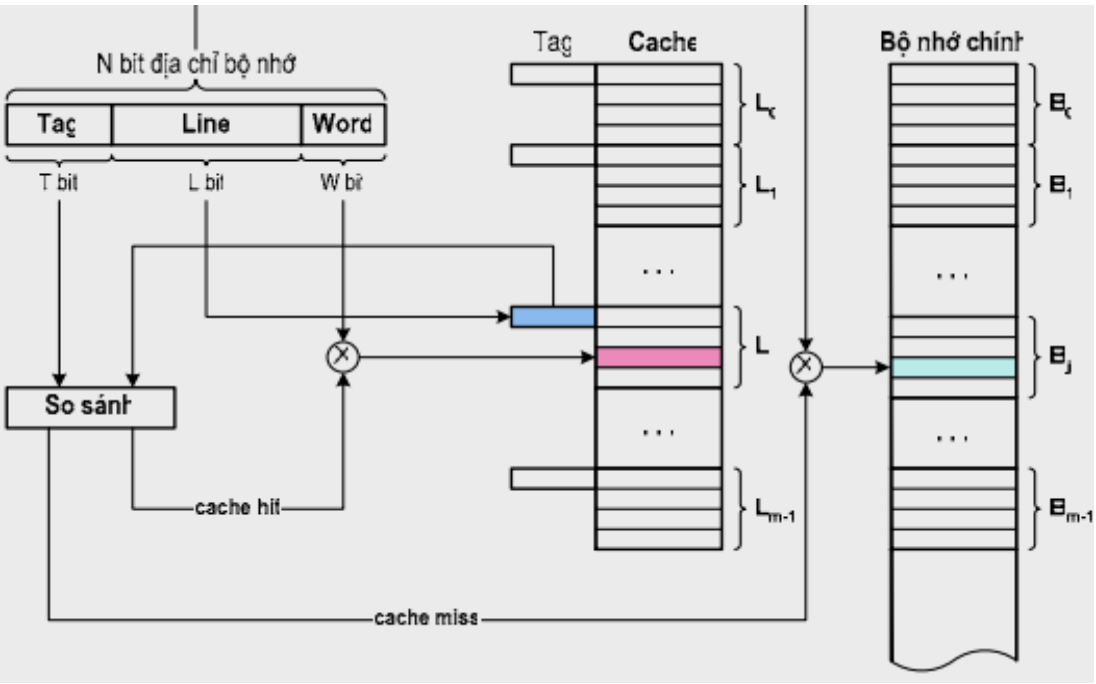
$B_{m-1} \rightarrow L_{m-1}$

$B_m \rightarrow L_0$

$B_{m+1} \rightarrow L_1$

Tổng quát:

B_j chỉ có thể nạp vào $L_{j \bmod m}$, m là số Line của cache.



Hình 4.7: Sơ đồ ánh xạ trực tiếp

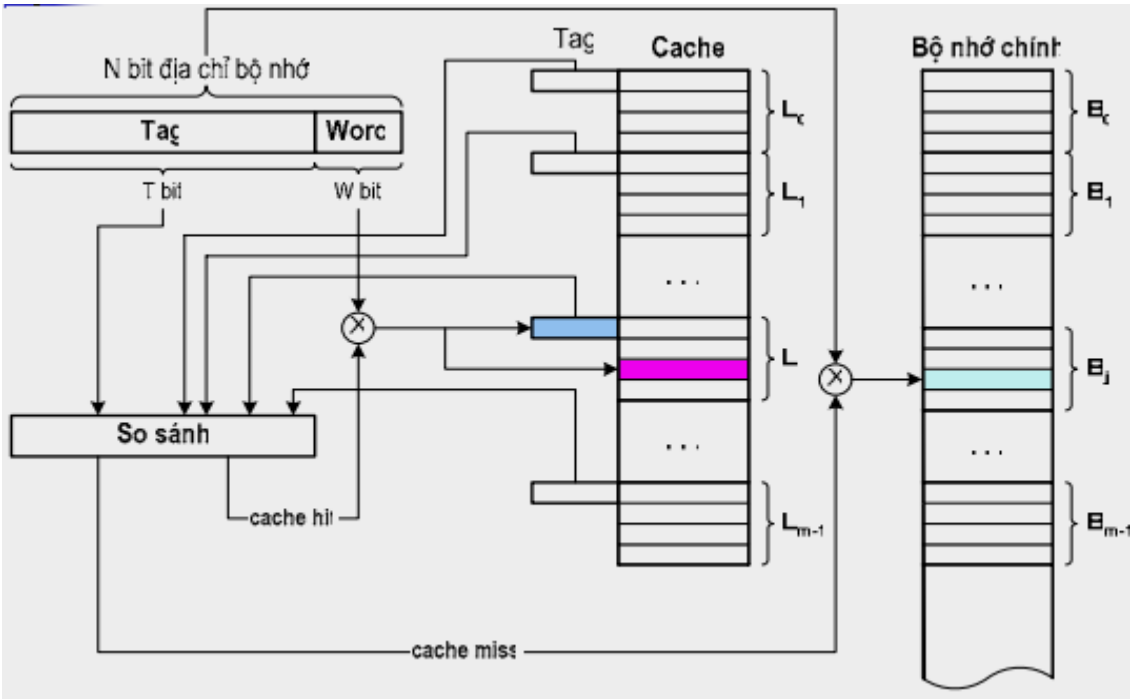
Đặc điểm của ánh xạ trực tiếp:

+ Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:
  - Trường Word gồm W bit xác định một từ nhớ trong Block hay Line: 2^W = kích thước của Block hay Line
  - Trường Line gồm L bit xác định một trong số các Line trong cache: 2^L = số Line trong cache = m
  - Trường Tag gồm T bit: T = N - (W+L)

+ Bộ so sánh đơn giản
+ Xác suất cache hit thấp

b. Ánh xạ liên kết toàn phần

- Mỗi Block có thể nạp vào bất kỳ Line nào của cache.
- Địa chỉ của bộ nhớ chính bao gồm hai trường:
  - Trường Word giống như trường hợp ở trên.
  - Trường Tag dùng để xác định Block của bộ nhớ chính.
- Tag xác định Block đang nằm ở Line đó



Hình 4.8: Sơ đồ ánh xạ toàn phần

Đặc điểm của ánh xạ liên kết toàn phần:
- So sánh đồng thời với tất cả các Tag → mất nhiều thời gian
- Xác suất cache hit cao.
- Bộ so sánh phức tạp.

c. Ánh xạ liên kết tập hợp

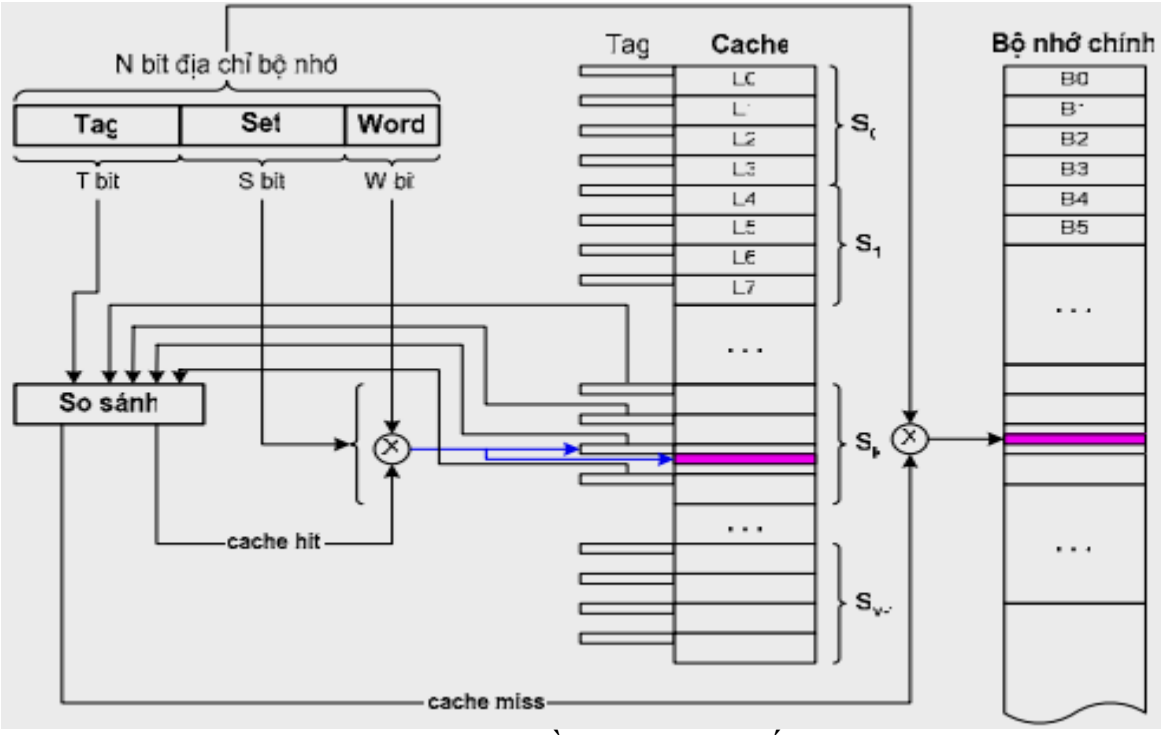
- Cache đươc chia thành các Tập (Set)
- Mỗi một Set chứa một số Line
- Vídụ: 4 Line/Set → 4-way associative mapping
- Ánh xạ theo nguyên tắc sau:

  B_0 → S_0

  B_1 → S_1

B_2 → S_2

.....



Hình 4.9: Sơ đồ ánh xạ liên kết tập hợp

Đặc điểm của ánh xạ liên kết tập hợp:

- Kích thước Block = 2^W Word
- Trường Set có S bit dùng để xác định một trong số V = 2^S Set
- Trường Tag có T bit: T = N -(W+S)
- Tổng quát cho cả hai phương pháp trên
- Thông thường 2,4,8,16 Lines/Set

* *Các giải thuật thay thế block trong cache*

+ Thuật giải thay thế (1): Ánh xạ trực tiếp
  - Không phải lựa chọn
  - Mỗi Block chỉ ánh xạ vào một Line xác định
  - Thay thế Block ở Line đó

+ Thuật giải thay thế (2): Ánh xạ liên kết
  - Được thực hiện bằng phần cứng (nhanh)
  - Random: Thay thế ngẫu nhiên
  - FIFO (First In First Out): Thay thế Block nào nằm lâu nhất ở trong Set đó
  - LFU (Least Frequently Used): Thay thế Block nào trong Set có số lần truy nhập ít nhất trong cùng một khoảng thời gian.
  - LRU (Least Recently Used): Thay thế Block ở trong Set tương ứng có thời gian lâu nhất không được tham chiếu tới → Tối ưu nhất.

* *Phương pháp ghi dữ liệu khi cache hit*

- Ghi xuyên qua (Write-through): Ghi cả cache và cả bộ nhớ chính, tốc độ chậm.

- Ghi trả sau (Write-back): Chỉ ghi ra cache, tốc độ nhanh, khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính.

** Cache trên các bộ xử lý Intel*

- 80386: Không có cache trên chip
- 80486: 8KB cache L1 trên chip
- Pentium: có 2 cache L1 trên chip
  - Cache lệnh = 8KB
  - Cache dữ liệu = 8KB
- Pentium4: hai mức cache L1 và L2 trên chip

- Cache L1: Mỗi cache 8KB, Kích thước Line = 64byte, ánh xạ liên kết tập hợp 4 đường.

- Cache L2: 256KB, Kích thước Line = 128byte, ánh xạ liên kết tập hợp 8 đường.

** Các mức Cache*

Việc dùng cache trong có thể làm cho sự cách biệt giữa kích thước và thời gian thâm nhập giữa cache trong và bộ nhớ trong càng lớn. Người ta đưa vào nhiều mức cache:

• Cache mức một (L1 cache): thường là cache trong (on-chip cache; nằm bên trong CPU)

• Cache mức hai (L2 cache) thường là cache ngoài (off-chip cache; cache này nằm bên ngoài CPU).

CÂU HỎI VÀ BÀI TẬP

1. Sự khác nhau giữa SRAM và DRAM? Trong máy tính chúng được dùng ở đâu?
2. Mục tiêu của các cấp bộ nhớ?
3. Sự khác biệt giữa cache và bộ nhớ ảo?

CHƯƠNG 5: THIẾT BỊ NHẬP XUẤT

Mục tiêu

- *Biết được cấu tạo và các vận hành của các loại thiết bị lưu trữ*
- *Hiểu các phương pháp để đảm bảo an toàn dữ liệu lưu trữ*

1.Các thiết bị nhớ trên vật liệu từ

Mục tiêu : *Biết được cấu tạo và các vận hành của các loại thiết bị nhớ vật liệu từ.*

Các thiết bị nhớ thông dụng là:

- Các đĩa từ, băng từ, đĩa quang, các loại thẻ nhớ là những bộ phận lưu trữ thông tin trữ lượng lớn.

Trong chương này chúng ta tập trung nói đến các bộ phận lưu trữ số liệu có trữ lượng cao (đĩa từ, đĩa quang, băng từ) và sự kết nối các bộ phận này vào máy tính.

1.1Đĩa từ (đĩa cứng, đĩa mềm)

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.

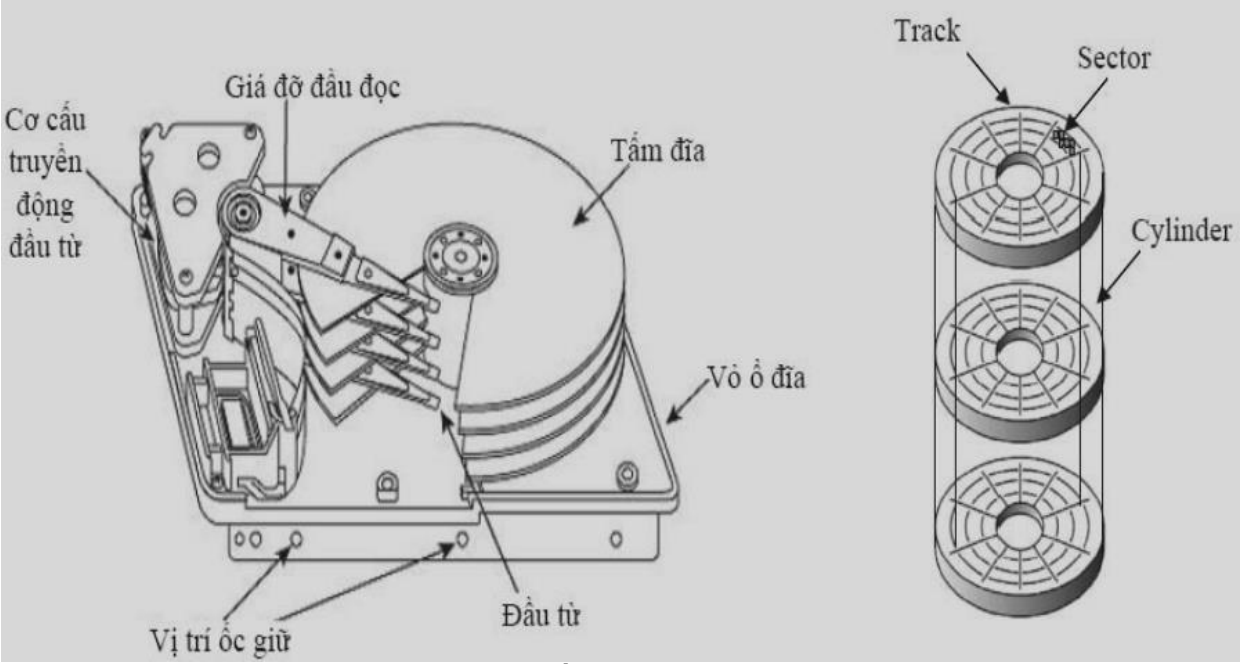
- Lưu trữ dài hạn các tập tin.
- Thiết lập một cấp bộ nhớ bên dưới bộ nhớ trong để làm bộ nhớ ảo lúc chạy chương trình.

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 5.1). 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* (Track). 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.

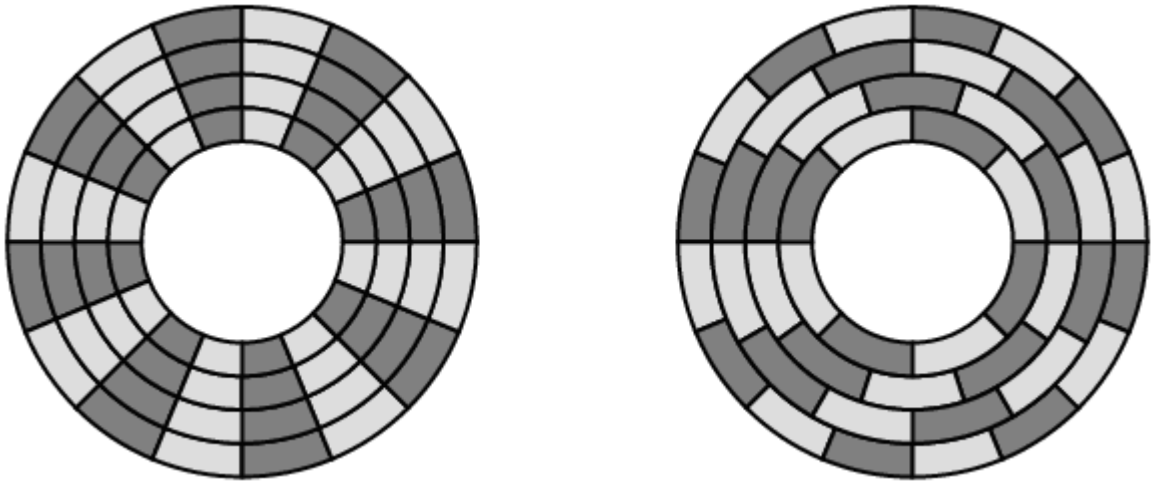
Số sector trên các track là khác nhau từ phần rìa đĩa vào đến vùng tâm đĩa, các ổ đĩa cứng đều chia ra hơn 10 vùng mà trong mỗi vùng có số sector/track bằng nhau.

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

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.



Mật độ ghi đều Mật độ ghi không đều

Hình 5.2: 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.

Băng thông số kỹ thuật đĩa cứng	
Dung lượng tối đa	có thể đạt 500 GB
Số lượng đầu đọc	1 – 8
Số tấm ghi (đĩa)	1 - 4
Cache (bộ đệm)	2 – 16 MB
Số cung (Sectors - 512 bytes/sector)	xxx,xxx,xxx
Tốc độ quay đĩa (RPM)	3600 - 15000
Mật độ	có thể đạt 95 Gb/in^2
Mật độ rãnh (TPI - Max Tracks/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

Bảng 5.1: Thông số kỹ thuật của đĩa cứng

1.2 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).

2.Thiết bị nhớ quang học

Mục tiêu : *Biết được cấu tạo và các vận hành của các loại thiết bị nhớ quang học.*

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).

2.1. CD-ROM, CD-R/W

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.

2.2. DVD-ROM, DVD-R/W

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.

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

Bảng 5.2: So sánh một số thông số của hai loại đĩa CDROM và DVDROM

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.

2.3. Bluray

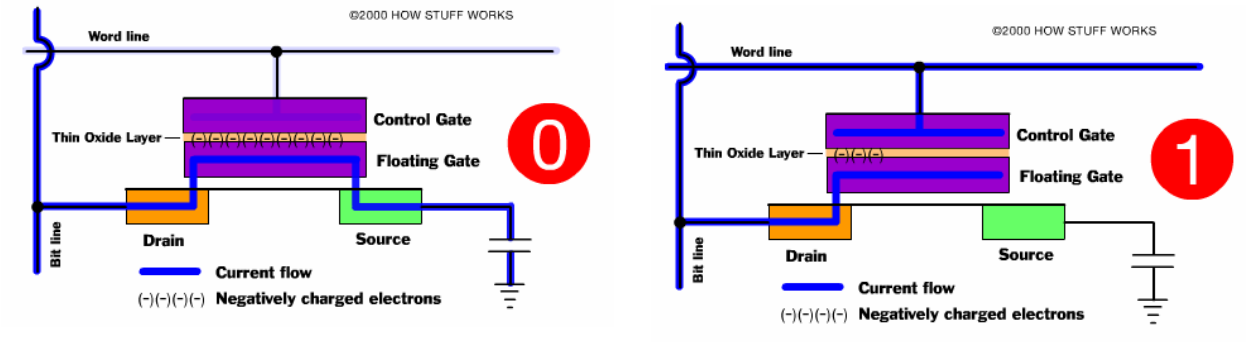
Blu-ray và HD DVD là hai công nghệ DVD có công suất lưu trữ lớn khi ghi nội dung độ phân giải cao, gấp 6 lần so với chuẩn DVD trước đó. Loại đĩa này có 25 GB bộ nhớ ghi trên một mặt của một đĩa đơn 12 cm, cho phép thu hình tới 13 giờ(Ở độ phân giải chuẩn DVD, tức là khoảng 720*480) so với đĩa 4,7 GB trước đó chỉ thu được 2 giờ. Đĩa quang có tên Disque Blu-ray bởi vì nó được áp dụng tia laser màu xanh lam để nạp thông tin vào đĩa.

3.Các loại thẻ nhớ

Mục tiêu : *Biết được cấu tạo và các vận hành của các loại thiết bị nhớ quang học.*

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 5.3: Minh hoạ hai trạng thái của một bit nhớ trong thẻ nhớ

4. An toàn dữ liệu trong lưu trữ

Mục tiêu: *Hiểu các phương pháp để đảm bảo an toàn dữ liệu lưu trữ*

4.1. RAID (Redundant Arrays of Inexpensive Disks)

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:

+ 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.

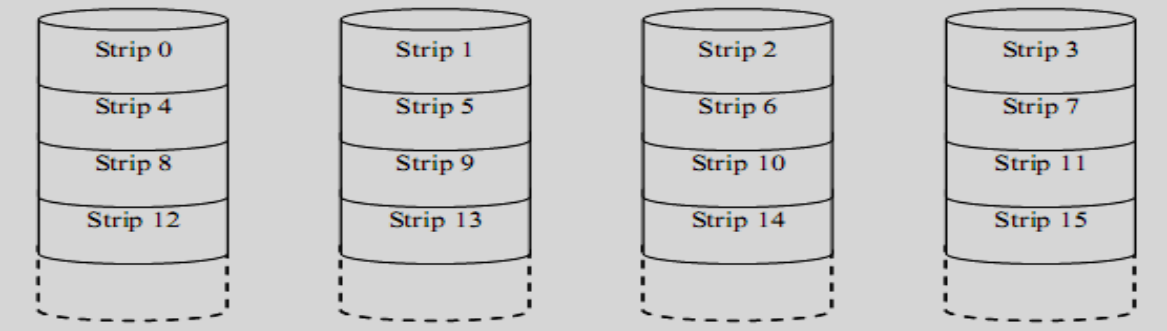
+ Với cơ chế đọc/ghi thông tin diễn ra trên nhiều đĩa (ghi đan chéo hay soi gương).

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 .

4.2. Các loại RAID

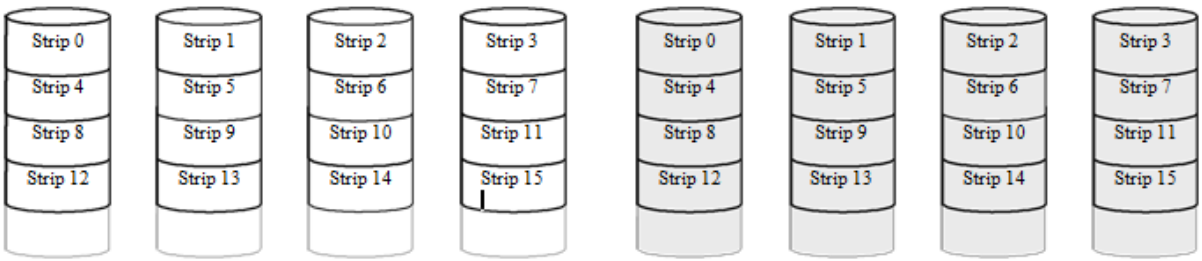
i). *RAID 0*: Thực ra, kỹ thuật này không nằm trong số các kỹ thuật có cơ chế an toàn dữ liệu. Khi mảng được thiết lập theo RAID 0, ổ đĩa logic có được (mà hệ điều hành nhận biết) có dung dượng bằng tổng dung lượng của các ổ đĩa thành viên. Điều này giúp cho người dùng có thể có một ổ đĩa logic có dung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



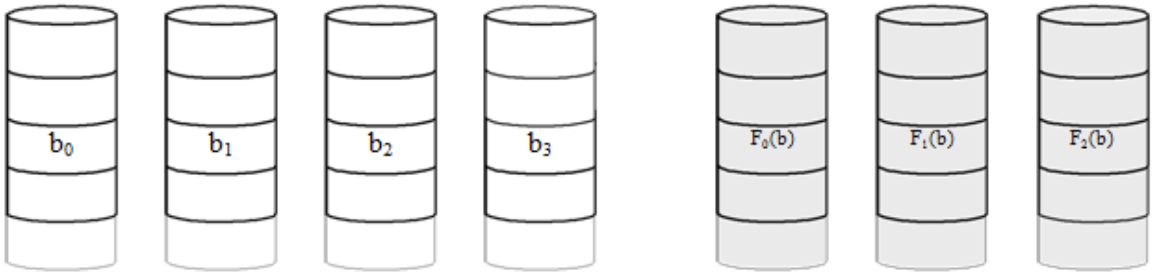
Hình 5.4: RAID 0

ii). *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%).



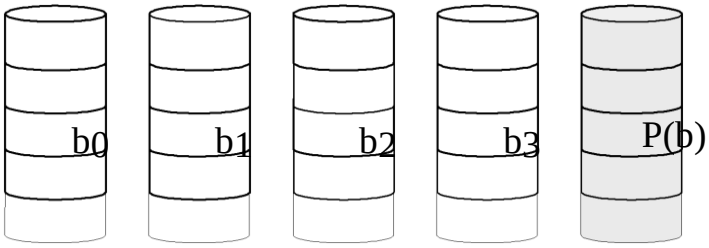
Hình 5.5: RAID 1

iii) *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.



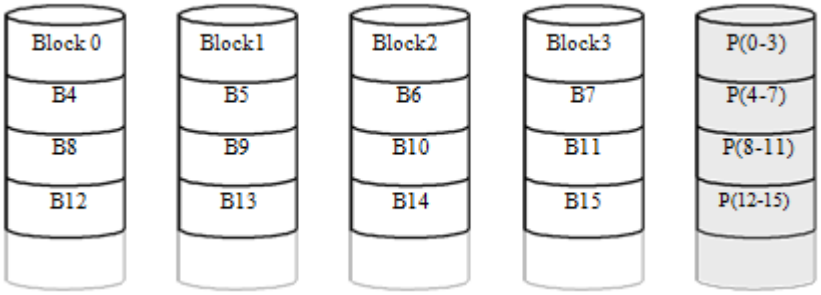
Hình 5.6:RAID 2

iii). *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).



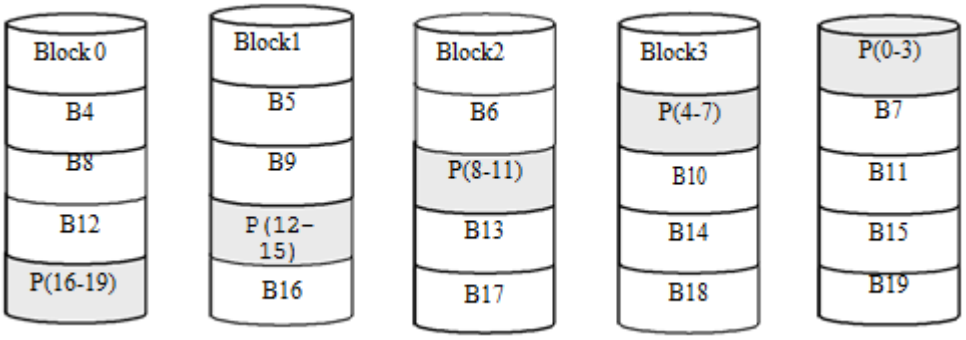
Hình 5.7: RAID 3

iv) *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 5.8: RAID 4

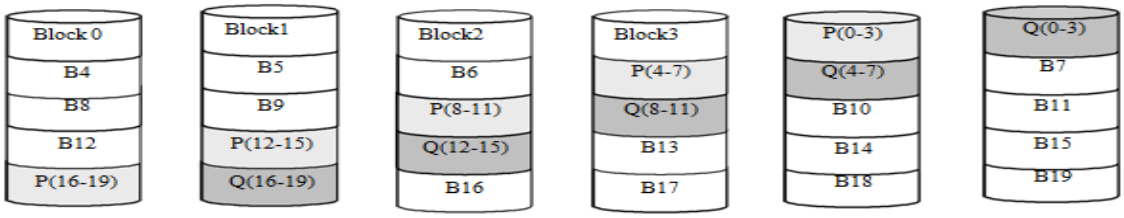
v). *RAID 5:* yêu cầu thiết lập giống như RAID 4, dữ liệu được ghi từng khối trên các đĩa thành viên, các bit chẵn lẻ được tính toán mức độ khối được ghi trải đều lên trên tất cả các ổ đĩa trong mảng. Tương tự RAID 4, khi một đĩa bất kỳ trong mảng bị hư hỏng, hệ thống vẫn hoạt động bình thường. Khi thay 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 5.9: RAID 5

vi). *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 5.10: RAID 6

CÂU HỎI VÀ BÀI TẬP

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.

CHƯƠNG 6: CÁC LOẠI BUS

Mục tiêu:
- *Phân biệt được các hệ thống Bus trong máy tính ;*
- *Trình bày được chức năng của các loại Bus.*
- *Thực hiện các thao tác an toàn với máy tính.*

1. Định nghĩa bus, bus hệ thống

Mục tiêu:*nắm được định nghĩa bus,phân biệt được các hệ thống bus máy tính*

1.1. Định nghĩa bus

Trong máy tính, bộ xử lý và bộ nhớ trong liên lạc với các ngoại vi bằng bus. Bus là một hệ thống các dây cáp nối (khoảng 50 đến 100 sợi cáp riêng biệt) trong đó một nhóm các cáp được định nghĩa chức năng khác nhau bao gồm: các đường dữ liệu, các đường địa chỉ, các dây điều khiển, cung cấp nguồn. (Như vậy bus là tập hợp các đường kết nối dùng để vận chuyển thông tin giữa các thành phần của máy tính với nhau)

Dùng bus có 2 ưu điểm là giá tiền thấp và dễ thay đổi ngoại vi. Người ta có thể gỡ bỏ một ngoại vi hoặc thêm vào ngoại vi mới cho các máy tính dùng cùng một hệ thống bus.

Giá tiền thiết kế và thực hiện một hệ thống bus là rẻ, vì nhiều ngã vào/ra cùng chia sẻ một số đường dây đơn giản. Tuy nhiên, điểm thất lợi chính của bus là tạo ra nghẽn cổ chai, điều này làm giới hạn lưu lượng vào/ra tối đa. Các hệ thống máy tính dùng cho quản lý phải dùng thường xuyên các ngoại vi, nên khó khăn chính là phải có một hệ thống bus đủ khả năng phục vụ bộ xử lý trong việc liên hệ với các ngoại vi.

Một trong những lý do khiến cho việc thiết kế một hệ thống bus khó khăn là tốc độ tối đa của bus bị giới hạn bởi các yếu tố vật lý như chiều dài của bus và số bộ phận được mắc vào bus.

1.2. Bus hệ thống(System bus)

Bus hệ thống : Là hệ thống dẫn đường liên quan các thiết bị quan trọng như: CPU, bộ nhớ và các mạch vào ra.

2. Bus đồng bộ và không đồng bộ

Mục tiêu:*phân biệt được bus đồng bộ và bus không đồng bộ.*

2.1.Bus đồng bộ

là một đường điều khiển bởi bộ dao động thạch anh, tín hiệu trên đường dây này có dạng sóng vuông

+ Bus có tín hiệu Clock
+ Các sự kiện trên bus được xác định bởi xung nhịp Clock.

Bus hệ thống là một bus đồng bộ, nó gồm có một xung nhịp trong các đường dây điều khiển, và một nghi thức cho các địa chỉ và các số liệu đối với xung nhịp. Do có rất ít hoặc không có mạch logic nào dùng để quyết định hành động kế tiếp nào cần thực hiện, nên các bus đồng bộ vừa nhanh, vừa rẻ tiền. Trên bus này, tất cả đều phải vận hành với cùng một xung nhịp.

2.2. Bus không đồng bộ

+ Không có tín hiệu Clock

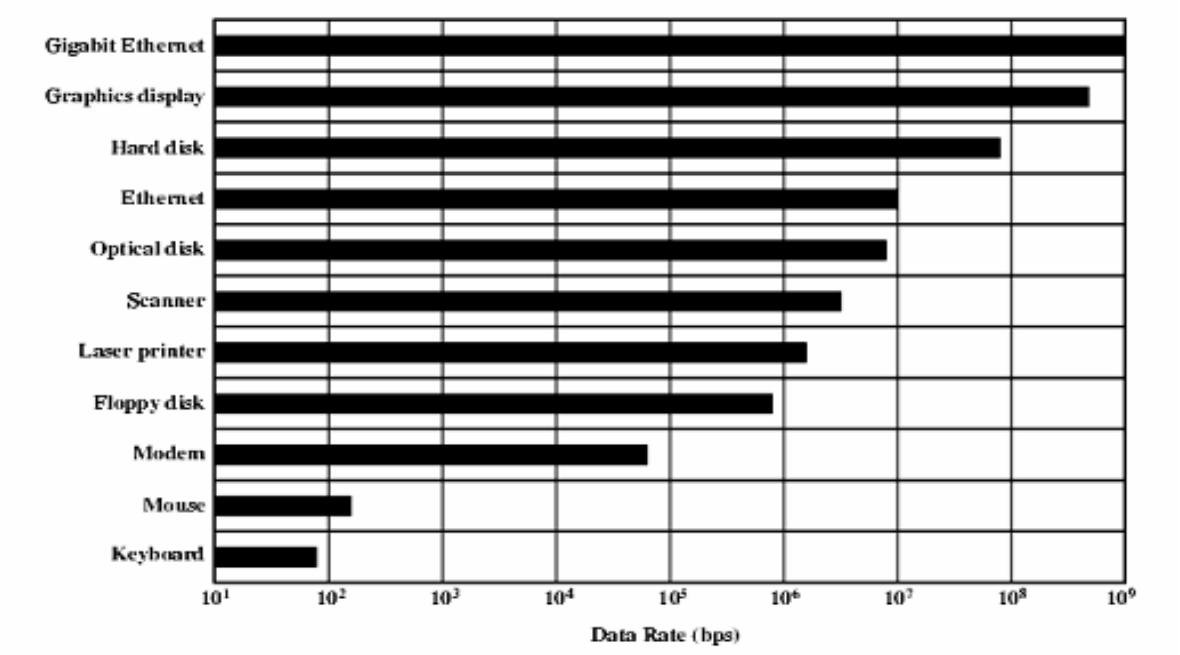
+ Kết thúc một sự kiện trên trên bus sẽ kích hoạt cho một sự kiện tiếp theo.

3. Hệ thống bus phân cấp

Mục tiêu: *Hiểu các kiến thức về hệ thống kết nối cơ bản, các bộ phận bên trong máy tính, cách giao tiếp giữa các thiết bị ngoại vi và bộ xử lý.*

3.1. Bus nối bộ xử lý với bộ nhớ

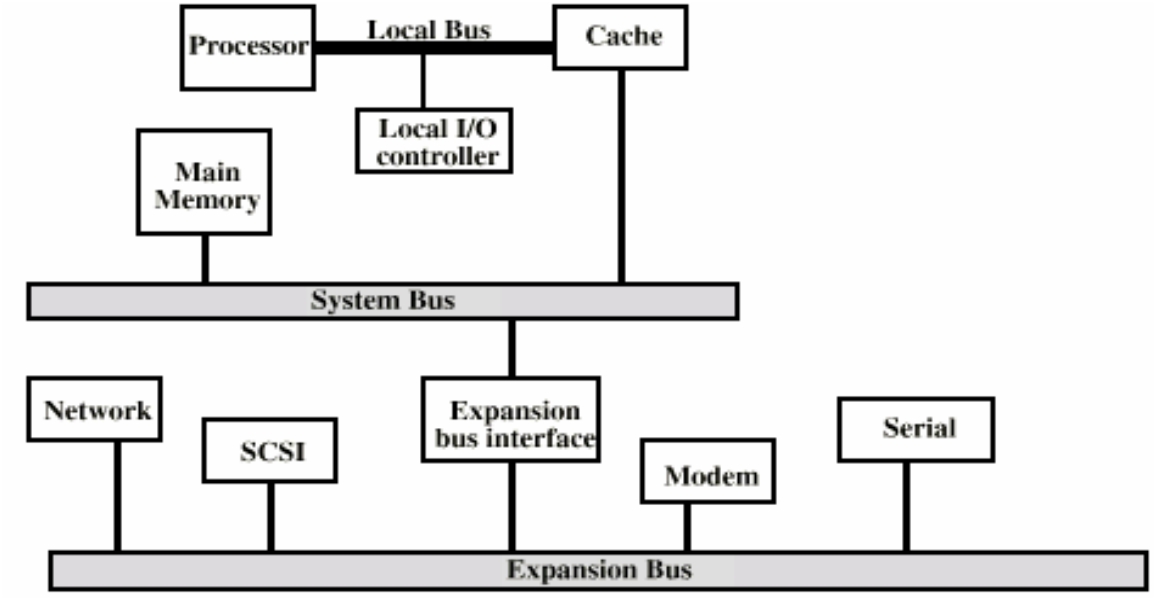
Bus hệ thống nối bộ xử lý với bộ nhớ (system bus, Front Side Bus-FSB). Bus kết nối bộ xử lý với bộ nhớ thì ngắn và thường thì rất nhanh. Trong giai đoạn thiết kế bus kết nối bộ xử lý với bộ nhớ, nhà thiết kế biết trước các linh kiện và bộ phận mà ông ta cần kết nối lại, còn nhà thiết kế bus vào/ra phải thiết kế bus thoả mãn nhiều ngoại vi có mức trì hoãn và lưu lượng rất khác nhau (xem hình 6.1)



Hình 6.1: *Bảng biểu diễn tốc độ dữ liệu của các ngoại vi*

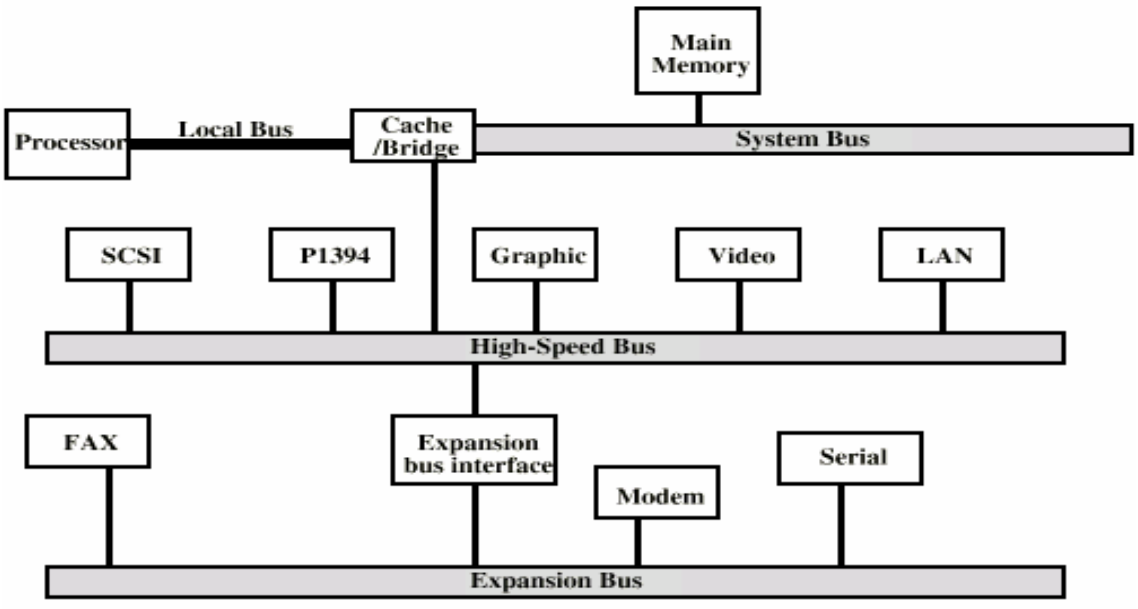
3.2. Bus vào – ra*:(BUS nối ngoại vi)*

Bus vào/ra có thể có chiều dài lớn và có khả năng nối kết với nhiều loại ngoại vi, các ngoại vi này có thể có lưu lượng thông tin khác nhau, định dạng dữ liệu khác nhau (xem hình 6.2)



Hình 6.2: Hệ thống bus trong một máy tính

Hiện nay, trong một số hệ thống máy tính, bus nối ngoại vi được phân cấp thành hai hệ thống bus con. Trong đó, bus tốc độ cao (high-speed bus) hỗ trợ kết nối các thiết bị tốc độ cao như SCSI, LAN, Graphic, Video,…và hệ thống bus mở rộng (expansion bus) được thiết kế để kết nối với các ngoại vi yêu cầu tốc độ thấp như: modem, cổng nối tiếp, cổng song song,…Giữa hai hệ thống bus nối ngoại vi trong tổ chức hệ thống bus phân cấp là một giao diện đệm (hình 6.3).



Hình 6.3: Hệ thống bus phân cấp

Ta có thể có nhiều lựa chọn trong việc thiết kế một bus

Đặc tính của bus	Bus hệ thống	Bus nối ngoại vi
Độ rộng của bus	Đường dây địa chỉ và số liệu khác nhau	Đường địa chỉ và số liệu được đa hợp
Độ rộng bus số liệu	Càng rộng càng nhanh (ví dụ	Càng hẹp càng ít tốn kém (ví dụ 8 bít)
Số từ được chuyền	Chuyển nhiều từ	Chuyển đơn giản mỗi lần một từ
Chủ nhân của bus	Nhiều	Một
Chuyển từng gói	Có. Cần nhiều chủ nhân bus	Không. Kết nối một lần và chuyển hết thông tin
Xung nhịp	Đồng bộ	Bất đồng bộ

Trong bảng trên có khái niệm sau đây liên quan đến các chủ nhân của bus – các bộ phận có thể khởi động một tác vụ đọc hoặc viết trên bus. Ví dụ bộ xử lý luôn là một chủ nhân của bus. Một bus có nhiều chủ nhân khi nó có nhiều bộ xử lý, hoặc khi các ngoại vi có thể khởi động một tác vụ có dùng bus. Nếu có nhiều chủ nhân của bus thì phải có một cơ chế trọng tài để quyết định chủ nhân nào được quyền chiếm lĩnh bus. Một bus có nhiều chủ, có thể cấp một dãi thông rộng (bandwidth) bằng cách sử dụng các gói tin thay vì dùng bus cho từng tác vụ riêng lẻ. Kỹ thuật sử dụng gói tin được gọi là phân chia nhỏ tác vụ (dùng bus chuyển gói). Một tác vụ đọc được phân tích thành một tác vụ yêu cầu đọc (tác vụ này chứa địa chỉ cần đọc), và một tác vụ trả lời của bộ nhớ (chứa thông tin cần đọc). Mỗi tác vụ đều có một nhãn cho biết loại của tác vụ. Trong kỹ thuật phân chia nhỏ tác vụ, trong khi bộ nhớ đọc các thông tin ở địa chỉ đã xác định thì bus được dành cho các chủ khác.

4. Các loại bus sử dụng trong các hệ thống vi xử lý

Mục tiêu: *Nắm được các loại bus trong hệ thống vi xử lý*

Bus thường phân loại theo 3 cách sau:

❖ *Theo tổ chức phần cứng*

Trong thế giới máy tính có rất nhiều loại bus khác nhau được sử dụng.
Các loại bus này nói chung là không tương thích với nhau.
Sau đây là một số loại bus được dùng phổ biến

Tên bus	**Lĩnh vực áp dụng**

IBM PC, PC/AT	Máy tính IBM PC, IBM/PC/AT
Multibus I	Một số hệ thống có VXL 8086, 8088
Multibus II	Một số hệ thống có VXL 80386
Versabus	Một số hệ thống dùng VXL Motorola
Camac	Vật lý hạt nhân

❖ *Theo giao thức truyền thông* (bus đồng bộ và không đồng bộ)

❖ *Theo loại tín hiệu truyền trên bus (bus địa chỉ, bus dữ liệu, bus điều khiển)*

+ **Bus địa chỉ:**

Chức năng : vận chuyển địa chỉ để xác định ngăn nhớ hay cổng vào- ra

+ **Bus dữ liệu:**

Chức năng: vận chuyển lệnh từ bộ nhớ đến CPU, vận chuyển dữ liệu giữa CPU, các môđun nhớ và môđun vào-ra.

+ **Bus điều khiển:**

Chức năng: vận chuyển các tín hiệu điều khiển

Các loại tín hiệu điều khiển:

Các tín hiệu phát ra từ CPU để điều khiển môđun nhớ và môđun vào-ra

Các tín hiệu từ môđun nhớ hay môđun vào-ra gửi đến yêu cầu CPU.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 6

1. Tại sao cần modul vào/ra? Chức năng modul vào/ra
2. Thông thường có bao nhiêu loại bus? Tại sao phải có các chuẩn cho các bus vào ra?
3. Thế nào là chủ nhân của bus? Khi bus có nhiều chủ nhân thì làm thế nào để giải quyết tranh chấp bus?

CHƯƠNG 7: NGÔN NGỮ ASSEMBLY

Mục tiêu :

- *Phân biệt các thành phần cơ bản của Assembly;*
- *Trình bày cấu trúc của 1 chương trình Assembly;*
- *Khai báo biến, toán tử, một số hàm cơ bản và các chế độ địa chỉ;*
- *Vận dụng được cú pháp các lệnh điều khiển để xây dựng bài toán;*
- *Sử dụng được các lệnh cơ bản;*
- *Trình bày được ngăn xếp;*
- *Viết chương trình con và cách truyền tham số cho chương trình con.*

1. Tổng quan

Mục tiêu: *Phân biệt các thành phần cơ bản của Assembly*

❖ **Ngôn ngữ máy:**

Bất kì một chương trình nào viết ở bất kì ngôn ngữ nào muốn thực thi trên máy phải chuyển về dạnh ngôn ngữ máy đó là chuỗi các bít 0, 1. Mỗi họ CPU đều có một ngôn ngữ máy riêng của nó. ở đây chúng ta chỉ xét họ CPU của hãng InTel.

Ví dụ :

01D8 --> ADD AX, BX --> AX := AX + BX
50h --> Push AX --> Đẩy AX vào Stack
80C210 --> ADD DL, 10

* Ghi chú :

Mỗi lệnh sinh ra một mã máy khác nhau. Mỗi lệnh có một độ dài khác nhau.

* Ưu điểm của mã máy :

Thực thi nhanh.
Không cần trình biên dịch.

* Nhược điểm của mã máy :

Mã máy khó nhớ.
Khó sửa

❖ **Hợp ngữ:**

Để khắc phục nhược điểm của mã máy người ta sử dụng từ gợi nhớ gọi là hợp ngữ .

Ví dụ :

Mã máy	Từ gợi nhớ
01D8	ADD AX, BX
50	PUSH AX
80C210	ADD DL, 10

* Ưu điểm :

Dễ nhớ

Khó sửa

* Nhược điểm :

Cần một trình biên dịch để dịch sang mã máy gọi là Assemble (Bộ biên dịch).

2.Cấu trúc chương trình

Mục tiêu: *Trình bày cấu trúc của 1 chương trình Assembly*

Khai báo biến, toán tử, một số hàm cơ bản và các chế độ địa chỉ;

2.1. Cấu trúc chương trình hợp ngữ

❖ **Tập tin .COM**

Đặc điểm:

+ Chạy nhanh hơn tập tin .EXE
+ Các dữ liệu, chương trình và Stack đều dùng chung một đoạn
+ Thích hợp với những chương trình nhỏ
+ Bắt đầu thự c thi ở địa chỉ 100h

Cấu trúc:

```
Code_Seg    SEGMENT
ASSUME    CS:code_seg, DS:code_seg
ORG   100h
Begin
   In các lệnh
   int 20h
   [ chương trình con nếu có]
   ; khai báo dữ liệu
Code_seg   END
              END Begin
```

Ví dụ: Chương trình 'hello.com'

```
Code   SEGMENT
ASSUME   CS:Code,DS:Code
ORG   100h
Begin
      MOV   AH,09h
```

```
                MOV   DX,Offset String
                Int 20h
                String   DB ' hello,how are you ? $ '
        Code Ends
                END Begin
```

❖ **Tập tin .EXE**

Đặc điểm:

+ Dữ liệu, chương trình, stack được dùng ở các đoạn khác nhau
+ Tập tin có thể lớn tùy ý chỉ phụ thuộc vào bộ nhớ RAM của máy

Cấu trúc:

```
Stack_Seg    SEGMENT   Stack
          ; Khai báo stack
          Stack_seg    ENDS
          Data_seg   SEGMENT
          ; khai báo dữ liệu
          Data_seg   ENDS
         Code_seg    SEGMENT
        ASSUME   CS:Code, DS:Data_seg, SS:Stack_seg
        Begin
                MOV   AX ,Data_seg
                MOV   DX,AX
                Các lệnh
                MOV   AX,4C00h
                Int   21h
                ;[Chương trình con nếu có]
        Code_seg   ENDS
        END Begin
```

***** ***Ghi chú:***

Hàm INT 20h là hàm để thoát về DOS chỉ dùng cho tập tin .COM

Một lệnh MOV AX,4C00h
Int 21h

dùng để thoát về DOS cho cả tập tin .COM và .EXE

2.2. Cú pháp lệnh hợp ngữ

Một lệnh của hợp ngữ có cấu trúc như sau:

[nhãn] [<từ gợi nhớ>] [< toán hạng >] [;chú thích]

Một lệnh dài tối đa 128 ký tự

Mỗi phần trên được phân cách với nhau bằng khoảng trắng hay phím tab

Không nhất thiết phải luôn luôn xuất hiện 4 phần trên trong một lệnh mà tùy theo ý nghĩa mỗi lệnh

+ Nhãn (label) :

Là một tên dùng để thay thế địa chỉ trong tập tin nguồn, như vậy những phần khác có thể tham chiếu đến lệnh sau nhãnthông qua tên nhãn.

Mỗi nhãn chỉ được định nghĩa một lần
Không đặt tên nhãn trùng với từ khóa
Không được dùng tên chỉ dẫn để đặt tên nhãn
Sau nhãn có thể có dấu hai chấm hay không tùy lệnh
Nhãn có thể đứng một mình trên một dòng hoặc không
Ví dụ:

addition :	addition : add	AX,BX	
add	AX,BX	<=>	
JMP	addision	JMP	addision

+ Từ gợi nhớ : (Mnemonic)

Xác định hành động của câu lệnh, nó có thể là một chỉ thị hay một chỉ dẫn

Chỉ thị của hợp ngữ: gần giống như chỉ thị của CPU, nó xác định hành động mà CPU sẽ thực hiện. Như vậy , khi dịch Assembler sẽ dịch mỗi chỉ thị ra một mã máy tương ứng Chỉ dẫn là lệnh của hợp ngữ, nó không phải lệnh của CPU. Các chỉ dẫn tuy có xuất hiện trong chương trình nhưng không được biên dịch ra mã máy. Nó chẳng qua là dùng để điều khiển cách dịch của Assembler

Ví dụ :

Begin
End Begin

+Toán hạng : (Operad)

Là dữ liệu mà câu lệnh cần xử lý. Toán hạng có thể là thanh ghi, tên thanh ghi, hằng, biến, nhãn, chương trình con, biểu thức...

* Chú thích :

Được viết sau dấu chấm phẩy ở cuối mỗi lệnh hay ở đầu dòng.

2.3. Các kiểu dữ liệu trong hợp ngữ

2.3.1. Các số

Một số nhị phân là một dãy các bit 0 và 1 và phải kết thúc bằng h hoặc H
Một số thập phân là một dãy các chữ số thập phân và kết thúc bởi d hoặc D (có thể không cần)
Một số hex phải bắt đầu bởi 1 chữ số thập phân và phải kết thúc bởi h hoặc H .
Sau đây là các biểu diễn số hợp lệ và không hợp lệ trong ASM :

Số	**Loại**
10111	**thập phân**
10111b	**nhị phân**
64223	**thập phân**
-2183D	**thập phân**

1B4DH	**hex**
1B4D	**số hex không hợp lệ**
FFFFH	**số hex không hợp lệ**
0FFFFH	**số hex**

2.3.2. Các kí tự

Ký tự và một chuỗi các ký tự phải được đóng giữa hai dấu ngoặc đơn hoặc hai dấu ngoặc kép . Ví dụ 'A' và "HELLO" . Các ký tự đều được chuyển thành mã ASCII bởi ASM . Do đó trong một chương trình ASM sẽ xem khai báo 'A' và 41h (mã ASCII của A) là giống nhau .

2.3.3. Các biến, hằng

Các biến:

Trong ASM biến đóng vai trò như trong ngôn ngữ cấp cao . Mỗi biến có một loại dữ liệu và nó được gán một địa chỉ bộ nhớ sau khi dịch chương trình . Bảng sau đây liệt kê các toán tử giả dùng để định nghĩa các loại số liệu .

❖ **Biến byte:**

Chỉ dẫn của ASM để định nghĩa biến byte có dạng như sau :

tên biến DB giá trị ban đầu

Ví dụ :

ALPHA DB 4

Chỉ dẫn này sẽ gán tên ALPHA cho một byte nhớ trong bộ nhớ mà giá trị ban đầu của nó là 4 . Nếu giá trị của byte là không xác định thì đặt dấu chấm hỏi (?) vào giá trị ban đầu .

Ví dụ :

BETA DB ?

Trong một byte :

Biểu diễn được 1 kí tự .

Biểu diễn từ 0 --> 255 (Số không dấu).

Biểu diễn từ -128 --> 127.

❖ **Biến từ:**

Chỉ dẫn của ASM để định nghĩa một biến từ như sau :

Tên biến DW giá trị ban đầu

Ví dụ :

WRD DW -2

Cũng có thể dùng dấu ? để thay thế cho biến từ có giá trị không xác định .

Trong một word biểu diễn từ 0 --> 65535 (Số không dấu)

Biểu diễn từ - 32768 --> 32767 (Số có dấu)

Ví dụ :

Double word	(Dword)	4 byte
Quad word	(Qword)	6 byte
Ten byte	(Tbyte)	10 byte

❖ **Biến mảng:**

Trong ASM một mảng là một loạt các byte nhớ hoặc từ nhớ liên tiếp
nhau . Ví dụ để định nghĩa một mảng 3 byte gọi là B_ARRAY mà giá trị ban đầu
của nó là 10h,20h và 30h chúng ta có thể viết :
B_ARRAY DB 10h,20h,30h
B_ARRAY là tên được gán cho byte đầu tiên
B_ARRAY+1 là tên của byte thứ hai
B_ARRAY+2 là tên của byte thứ ba

Byte thấp và byte cao của một từ

Đôi khi chúng ta cần truy xuất tới byte thấp và byte cao của một biến từ . Giả sử chúng ta định nghĩa :
WORD1 DW 1234h
Byte thấp của WORD1 chứa 34h , còn byte cao của WORD1 chứa 12h
Ký hiệu địa chỉ của byte thấp là WORD1 còn ký hiệu địa chỉ của byte cao là WORD1+1 .

Các hằng:

Trong một chương trình các hằng có thể được đặt tên nhờ chỉ dẫn EQU (equates) .
Cú pháp của EQU là :
Tên tượng trưng EQU hằng số
ví dụ :
LF EQU 0AH
sau khi có khai báo trên thì LF được dùng thay cho 0Ah trong chương trình . Vì vậy ASM sẽ chuyển các lệnh :
MOV DL,0Ah
và MOV DL,LF
thành cùng một mã máy .

3. Các lệnh điều khiển

- *Vận dụng được cú pháp các lệnh điều khiển để xây dựng bài toán;*
- *Sử dụng được các lệnh cơ bản;*

3.1. Các lệnh cơ bản

3.1.1. Nhóm lệnh chuyển dữ liệu

3.1.1.1. Lệnh MOV

* *Cú pháp :* **MOV < đích > , < nguồn >**

< đích > : chỉ có thể là một thanh ghi hay bộ nhơ

< nguồn > : có thể là thanh ghi ,bộ nhớ hay hằng

Chức năng :

Chuyển nội dung tóan hạng nguồn vào tóan hạng đích .chiều dài dữ liệu có thể 8 hay 16 bit.

* Ghi chú :

- Lệnh MOV không ảnh hưởng đến các thanh ghi cờ hiệu.
- Không được phép chuyển hai tóan hạng bộ nhớ với nhau . Muốn chuyển chúng ta phải dùng thanh ghi trung gian .

Ví dụ :

```
MOV   AX , Var 1
MOV   Var 2 , AX
```

- Không thể chuyển trực tiếp giữahai thanh ghi đọan .
- Không thể chuyển một hằng vào thanh ghi đọan, muốn chuyển chúng ta phải dùng thanh ghi trung gian .

Ví dụ :

```
MOV AX ,data
MOV DS , AX
```

- Không dùng thanh ghi CS vào trong tóan hạng của lệnh MOV

Màu :

- Địa chỉ màn hình màu : B800 : 0000
- Mỗi ký tự trên màn hình được biểu diễn bằng 2 byte

Byte thấp : chứa ký tự cần in ra

Byte cao : chứa màu nền và màu ký tự đó

Nó được định dạng như sau :

7	6	5	4	3	2	1	0
B/I	H	G	B	I	R	G	B

B / I : nằm ở bit thứ 7

H : nằm ở bit thứ 6

G : nằm ở bit thứ 5

B : nằm ở bit thứ 4

H, G ,B : màu nền

I : nằm ở bit thứ 3

R : nằm ở bit thứ 2

G : nằm ở bit thứ 1

B: nằm ở bit 0

I ,R ,G ,B : màu chử

I : intensity (cường độ , độ sáng) 0 :tối và 1 : sáng

B : blinking (chớp nháy) 0 : không nháy và 1 :nháy

Bit 7(B / I) : vừa làm nhiệm vụ cho nhấp nháy chữ hay tăng giảm độ sáng cho màu nền.

I	R	G	B	MÀU	I	R	G	B	MÀU

0	0	0	0	đen	1	0	0	0	Đen nhạt
0	0	0	1	Xanh dương	1	0	0	1	Da trời
0	0	1	0	Xanh lá cây	1	0	1	0	Xanh nhạt
0	0	1	1	cyan	1	0	1	1	cyan sáng
0	1	0	0	Đỏ	1	1	0	0	Đỏ sáng
0	1	0	1	Tím	1	1	0	1	Tím cà
0	1	1	0	Nâu	1	1	1	0	Vàng
0	1	1	1	Xám đậm	1	1	1	1	Trắng

Cách nhớ màu:

0	1	2	3	4	5	6	7
Đen	Xanh dương	Xanh lá cây	cyan	Đỏ	Tím	Nâu	Xám

- Muốn chữ sáng : cộng thêm số 8 cho màu chử
- Muốn nền sáng / chử nháy cộng thêm số 8 cho màu nền

CÔNG THỨC :

Màu nền * 16 + màu chữ

Ví dụ :

Chữ A có màu xanh dương trên nền đỏ

A : 65 đổi sang hệ thập phân là : 1000000

Ta lấy 1000000 + 1 = 01000001

Trong đó 0100 là nền đỏ và 0001 là chử A màu xanh dương .

Công thức tính địa chỉ offset của ký tự cần xuất ra màn hình

(Dòng * 18 + cột)*2

Ví dụ :

Viết chương trình in chử A tại dòng 5 cột 10 có màu xanh dương nền đỏ

```
Code   segment
    Assume  CS : code , DS : code
                Org  100h
    Begin
            MOV   AX, 0B800h
            MOV   DS, AX
            MOV   AL, 'A'
            MOV   AH, 65
             dòng = 5
```

```
                cột = 10
                MOV  BX , ( dòng * 80 + cột  )   *  2
                MOV  [ BX ], AX
        Code Ends
                End Begin
```

3.1.1.2.Lệnh XCHG : (Exchange)

* *Cú pháp :*

XCHG < đích > ,< nguồn >

Toán hạng < nguồn > và < đích > là thanh ghi hay bộ nhớ

* *Công dụng :* dùng để hóan chuyển nội dung của hai tóan hạng <nguồn > và < đích >

* Ghi chú :

- Lệnh này không ảnh hưởng đến cờ hiệu
- Không dùng lệnh này với thanh ghi đọan

Ví dụ :

```
A    DW    1234h
B    DW     3456h
MOV   AX , A
XCHG  AX , B
MOV    A , AX
```

3.1.1.3.Lệnh PUSH

* *Cú pháp :* **PUSH < nguồn >**

< Nguồn >: Có thể là một thanh ghi hay bộ nhớ 16 bit

* *Công dụng :* Dùng để nạp nội dung của tóan hạng < nguồn > vào stack (chồng) khi này thanh ghi SP giảm đi hai đơn vị .

3.1.1.4. Lệnh POP

* *Cú pháp :*

POP < đích >

Ngược lại với lệnh PUSH lệnh POP sẽ lấy nội dung đỉnh STACK đưa vào tóan hạng đỉnh khi này thanh SP tăng lên hai đơn vị

* Lưu ý :

Để cất giữ các thanh ghi trong việc sử dụng lệnh PUSH , lệnh POP khi dùng lệnh POP ta phải lấy ngược với lệnh PUSH .

Ví dụ :

```
PUSH  AX
PUSH  BX
PUSH   CX
```

Ta dùng lệng POP ngược lại với PUSH

```
POP CX
POP   BX
POP AX
```

3.1.1.5. Lệnh XLAT

* *Cú pháp :* XLAT

* Công dụng :

Dùng để chuyển nội dung của một số ô nhớ (8 bit) vào thanh ghi AL

* Ghi chú :

DS :BX chứa địa chỉ offset của vùng nhớ AL chứa thứ tự của ô nhớ

3.1.2. Nhóm lệnh tính toán số học

3.1.2.1.Nhóm lệnh xử lý phép cộng

3.1.2.1.1. Lệnh ADD: (Addition)

Cú pháp : **ADD < Dest > , < Source >**

< Dest > = < Dest > + < Source >

Dest : Là thanh ghi hay bộ nhớ

Source : Là thanh ghi hay bộ nhớ hay trực hằng

Chức năng : Dùng để cộng không nhớ giữa toán hạng Dest và toán hạng Source , kết quả lưu vào toán hạng Dest

Lệnh này ảnh hưởng đến các cờ : SF, CF , OF , AF, PF, ZF

3.1.2.1.2. Lệnh ADC (Addition With Carry)

Cú pháp : **ADC < Dest > , < Source >**

< Dest > = < Dest > + < Source > + CF

Tương tự như lênh ADD , lệnh ADC thực hiện phép cộng có nhớ

Lệnh ADC sẽ lấy toán hạng < Dest > + < Source> + cờ CF, kết quả lưu vào toán hạng Dest , lệnh này thường dùng cho phép cộng lớn hơn 16 bít

3.1.2.1.3. Lệnh INC : (Increment)

Cú pháp : **INC < Dest >**

< Dest > = < Dest > + 1

Dest : Có thể là thanh ghi hay bộ nhớ

Lệnh này ảnh hưởng đến 5 cờ : OF , ZF , PF , AF , SF

Ví dụ : Cho hai số A, B dạng double word chứa hai số nguyên không dấu , Viết chương trình cộng hai số A , B . Kết quả đưa vào một biến C , có sử lý trường hợp tràn số

```
Code    SEGMENT
ASSUME     CS : Code , DS : Code
ORG 100h
Begin :
        MOV     AX , Word    PTR  A
        MOV     BX , Word    PTR  A+2
        ADD     AX , Word   PTR  B
        ADC     BX , Word   B+2
        MOV     Word     PTR  C+2 , 0
        Int       20h
        A         DD    7000F000h
        B         DD    90002000h
        C         DB    5    Dup( 0 )
Code    ENDS
```

END Begin

3.1.2.2. Nhóm lệnh xử lý phép trừ

3.1.2.2.1. Lệnh SUB (Subtract)

* *Cú pháp:* **SUB < dest>,<source>**

< Dest >=<Dets> - <source>

Source :có thể là thanh ghi hay bộ nhớ

Lệnh source sẽ lấy nội dung toán hạng dest trừ đi toán hạng source và kết quả lưu vào dest

Ví dụ : Mov AX , 1234h
Sub AX , 0345h;
AX = 1234h - 0345h

Lệnh Sub ảnh hưởng đến sáu cờ 0F , CF , AF , PF , SF ,ZF

3.1.2.2.2. Lệnh SBB (Subtract with Borrow)

* *Cú pháp:* **SBB < dest > , < source >**

< Dest > = < Dest > - < Source > - CF

Lệnh SBB sẽ lấy toán hạng < Dest > - < Source > và trừ thêm cờ CF , kết quả lưu vào toán hạng Dest

3.1.2.2.3. Lệnh DEC (Decrement)

* *Cú pháp:* **DEC < Dest >**

< Dest > = < Dest > -1

Lệnh Dest là lệnh giảm đi 1 đơn vị . Lệnh này ảnh hưởng đến 5 Cờ và Cờ CF không ảnh hưởng tới

Lưu ý :

Lệnh SBB được sử dụng cho trường hợp số lớn hơn 16 bít

Ví dụ : Viết chương trình trừ 2 số :A=70002000h ,
B=90003000h,kết quả lưu vào bộ nhớ C = 5 Byte . Nếu byte cao của C =1thì kết quả là số âm , ngược lại là số dương

```
Code  Segment
      Assume  CS :Code
      ORG   100h
Begin:
      Mov   AX , Word  PTR   A
      Mov   BX , Word   PTR   A+2
      SUB   AX , Word  PTR   B
      SUB   BX , Word  PTR    B+2
      Mov   Word  PTR  C , AX
      Mov   Word  PTR  C+2 , BX
      ADC  Word  PTR  C +4
      Int    20h
      A   DB  70002000h
      B   DB   90003000h
      C   DB   5 Dup  ( ? )
Code    Ends
```

End Begin

3.1.2.3. Nhóm lệnh xử lý phép nhân

Lệnh MUL (Multiply)

* *Cú pháp:* **MUL < Source>**

< Source> có thể là thanh ghi hay bộ nhớ 8 bít hay 16 bít

Trương hợp 1 : Nếu < Source > có độ dài là 8 bít thì lệnh MUL sẽ lấy nội dung của thanh ghi AL nhân với source , kết quả lưu vào thanh ghi AX

Trường hợp 2:Nếu source có độ dài 16 bít thì lệnh MUL sẽ lấy nội dung thanh ghi AX nhân với sourrce , kết quả là 32 bít :16 bít thấp lưu vào AX , 16 bit cao lưu vào DX

Ví dụ :
```
Mov AL , 12h
Mov BL , 10h
MUL BL ; BL = 120h , AX = 0120h
Mov AX ,1234h
Mov BX , 100h
MUL BX
```

3.1.2.4. Nhóm lệnh xử lý phép chia

Lệnh DIV

* *Cú pháp:* **DIV < Source >**

Tương tự như phép nhân < source > có 2 trường hợp

Trường hợp 1:Nếu source có độ dài 8 bít thì lệnh DIV lấy nội dung thanh ghi AX chia cho source , thương số của phép chia được lưu trong thanh ghi AL và dư số được lưu trong AH

Trường hợp 2 :Nếu source là 16 bít thì lệnh DIV sẽ lấy nội dung trong cặp thanh ghi DX : AX chia cho toán hạng source , thương số được lưu trong AX , dư số được lưu trong DX

Ví dụ :
```
Mov AX , 0234h
Mov BL , 10h
DIV BL ; AL = 23h , AH = 04h
Mov AX , 1234h
Mov DX , 0567h
Mov BX , 100h
DIV BX ; AX = 567h , DX = 0234h
```

Ghi chú :

Muốn in một số >= 2 kí số chúng ta chia liên tiếp những số đó cho 10 đến khi thương số = 0 . Sau mỗi phép chia cất dư số vào stack , Sau khi thương số = 0 muốn in ra ta lấy từng gía trị trên đỉnh Stack đưa ra màn hình.

3.2. Các lệnh chuyển điều khiển

3.2.1. So sánh

* *Cú pháp:* **CMP < Left >, < Right >**

Left : Là thanh ghi hay bộ nhớ

Right : Là thanh ghi hay bộ nhớ hay trực hằng

Ví dụ :

CMP AL,0Dh

* *Công dụng :* Dùng để so sánh giữa toán hạng Left và toán hạng Right . Lệnh này Assemble sẽ lấy toán hạng Left trừ toán hạng Right kết quả được lưu trong các cờ mà không thay đổi nội dung của hai toán hạng trên

Kết quả của phép so sánh :

1.1 Đối với số không dấu :

CỜ ZF	CF
Left > Right	
Left = Right	
Left < Right	
0	0
1	0
0	1

1.2 Đối Với Số Có Dấu :

CỜ ZF	OF	SF
Left > Right		
Left = Right		
Left < Right		
0	0/1	0/1
1	0	0
0	1 /0	0/1

Ví dụ :

```
Mov AX , 6000h
CMP  AX , -7000h
```

<=> 6000h - (-7000h)

3.2.2. Lệnh lặp

Lệnh Loop :

* *Cú pháp :* **Loop <Nhãn>**

* *Công dụng :* Dùng để lặp vòng . Nếu thanh ghi CX khác 0 thì lệnh Loop <Nhãn> sẽ chuyển điều khiển đến lệnh kế sau nhãn và giảm thanh ghi CX đi một đơn vị . Nếu CX=0 thì lệnh nằm phía sau lệnh Loop <Nhãn> sẽ được thực thi.

Ví dụ :

```
    Mov  AH , 01h
    Mov  CX , 5
Lặp :
    Loop  lặp
```

Nhóm Lệnh Lặp Có Điều Kiện :

a. LoopE / LoopZ

Lặp nếu CX khác 0 và ZF = 1

b. LoopNE / LoopNZ < Nhãn >

Lặp nếu CX khác 0 và ZF = 0

Ví dụ : Viết chương trình nhập từ bàn phím một chuỗi tối đa 80 kí tự . Sau đó tìm kí tự chữ 'a' trong chuỗi vừa nhập vào , in ra kết quả

```
Code    SEGMENT
ASSUME     CS : Code , DS : Code
ORG     100h
Begin :
        Mov     AH , 09h
        Mov     DX , Offset    mess1
        Int       21h
        Mov     DX , Offset Maxtype
        Mov     AH , 0Ah
        Int       21h
        Mov     BX , Offset Len
        Mov     CL , [ BX ]
        Mov     CH , 0
        Mov     BX , Offset Buffer - 1
        Mov     AL , 'a'
   Lặp:
        INC     BX
        CMP     AL , [ BX ]
        LoopNE    Lặp
        JE     thấy
        JMP     không thấy
  Thấy:
        Mov     AH , 09h
        Mov     DX , Offset mess2
        Int       21h
        JMP     thoat
Không thấy :
        Mov     AH , 09h
        Mov     DX , Offset mess3
        Int       21h
 thoat :
        Int       20h
        mess3     DB ' Không tìm thấy $'
        mess2     DB ' Đã tìm thấy $'
        mess1     DB ' Nhập một chuỗi (< 80 ) kí tự '
        Maxtype     DB    81
        Len           DB    0
        Buffer        DB    81    Dup( ? )
Code   ENDS
        END     Begin
```

3.2.3. Lệnh nhảy

3.2.3.1. Lệnh nhảy không điều kiện

Cú pháp : **JMP < Nhãn >**

3.2.3.2. Lệnh nhảy có điều kiện

3.2.3.2.1 Nhóm Lệnh Nhảy Dùng Cho Số Không Dấu

LỆNH	ĐIỀU KIỆN	Ý NGHĨA
JA (>)	ZF = 0 and CF = 0	Jump if above
JAE (> =)	CF = 0	Jump if above or equal
JNA (< =)		Jump if Not above
JNAE (<)	CF = 1	Jump if Not above or equal
JB (<)	CF = 1 or ZF = 1	Jump if below
JBE (< =)		Jump if below or equal
JNB (> =)		Jump if Not below
JNBE (>)		Jump if Not below or equal

3.2.3.2.2 Nhóm Lệnh Nhảy Dùng Cho Số Có Dấu

LỆNH	ĐIỀU KIỆN	Ý NGHĨA
JG (>)	SF = OF and ZF = 0	Jump if greater
JGE (> =)	SF = OF	Jump if greater or equal
JNG (< =)		Jump if Not greater
JNGE (<)	SF <> OF	Jump if Not greater or equal
JL (<)	SF <> OF or ZF = 1	Jump if less
JLE (< =)		Jump if less or equal
JNL (> =)		Jump if Not less
JNLE (>)		Jump if Not less or equal

3.2.3.2.3 Nhóm Lệnh Nhảy Dùng Cho Cả Số Có Dấu Và Không Dấu

LỆNH	ĐIỀU KIỆN	Ý NGHĨA
JE (=)	ZF = 1	Jump if equal
JNE (< >)	ZF = 0	Jump if Not equal
JZ (=)		Jump if Zero
JNZ (< >)		Jump if Not Zero
JC	CF = 1	Jump if carry
JNC	CF = 0	Jump if Not carry
JCXZ	CX = 0	Jump if CX Zero

4. Ngăn xếp và các thủ tục

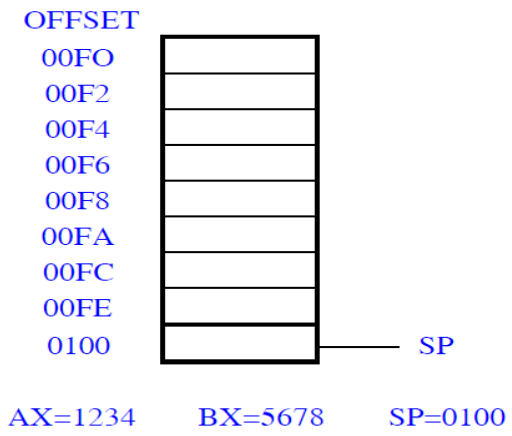
- *Mục tiêu: Trình bày được ngăn xếp;*
- *Viết chương trình con và cách truyền tham số cho chương trình con.*

4.1. Ngăn xếp

Đoạn ngăn xếp (stack segment) trong chương trình được dùng để cất giữ tạm thời số liệu và địa chỉ . Trong phần này chúng ta sẽ xem xét cách tổ chức stack và sử dụng nó để thực hiện các thủ tục (procedure).

Ngăn xếp là cấu trúc dữ liệu 1 chiều. Điều đó có nghĩa là số liệu được đưa vào và lấy ra khỏi stack tại đầu cuối của stack theo nguyên tắc LIFO (last in first out) . Vị trí tại đó số liệu được đưa vào hay lấy ra gọi là đỉnh của ngăn xếp (top of stack) .Có thể hình dung satck như một chồng đĩa. Đĩa đưa vào sau cùng nằm tại đỉnh của chồng đĩa. Khi lấy ra, đĩa trên cùng sẽ được lấy ra trước . Một chương trình phải dành ra một khối nhớ cho ngăn xếp. Chúng ta dùng chỉ dẫn .STACK 100h để khai báo kích thước vùng stack là 256 bytes .

Khi chương trình được dịch và nạp vào bộ nhớ thanh ghi SS (stack segment) sẽ chứa địa chỉ đoạn stack . Còn SP (stack pointer) chứa địa chỉ đỉnh của ngăn xếp . Trong khai báo stack 100h trên đây , SP nhận giá trị 100h . Điều này có nghĩa là stack trống rỗng (empty) như hình 6.1.



Hình 6.1: Stack empty

Lệnh PUSH

Để thêm một từ mới vào stack chúng ta dùng lệnh :

PUSH nguồn; đưa một thanh ghi hoặc từ nhớ 16 bit vào stack

Ví dụ PUSH AX . Khi lệnh này được thực hiện thì :

· SP giảm đi 2

· một bản copy của toán hạng nguồn được chuyển đến địa chỉ SS:SP còn toán hạng nguồn không thay đổi .

Lệnh POP

Để lấy số liệu tại đỉnh stack ra khỏi stack ,chúng ta dùng lệnh :

POP đích ; lấy số liệu tại đỉnh stack ra đích

Đích có thể là 1 thanh ghi hoặc từ nhớ 16 bit . Ví dụ :

POP BX ; Lấy số liệu trong stack ra thanh ghi BX .

Khi thực hiện lệnh POP :

· nội dung của đỉnh stack (địa chỉ SS:SP) được di chuyển đến đích .

· SP tăng 2

4.2. Các thủ tục

Trong các ngôn ngữ cấp cao người ta dùng thủ tục để giải các bài toán con , và chúng ta cũng làm như vậy trong hợp ngữ. Như vậy là một chương trình hợp ngữ có thể được xây dựng bằng các thủ tục .

Một thủ tục gọi là thủ tục chính sẽ chứa nội dung chủ yếu của chương trình .

Để thực hiện một công việc nào đó , thủ tục chính gọi (CALL) một thủ tục con .

Thủ tục con cũng có thể gọi một thủ tục con khác .Khi một thủ tục gọi một thủ tục khác , điều khiển được chuyển tới (controltransfer) thủ tục được gọi và các lệnh của thủ tục được gọi sẽ được thi hành . Sau khithi hành hết các lệnh trong nó , thủ tục được gọi sẽ trả điều khiển (return control) cho thủ tục gọi nó

Cú pháp của lệnh tạo một thủ tục:

tên thủ tục PROC kiểu gọi thủ tục
; các lệnh
RET
tên thủ tục ENDP

- tên thủ tục do người dùng định nghĩa.
- kiểu gọi thủ tục có thể là NEAR (có thể không khai báo) hoặc FAR .
- NEAR có nghĩa là thủ tục được gọi nằm cùng một đoạn với thủ tục gọi . FAR có nghĩa là thủ tục được gọi và thủ tục gọi nằm khác đoạn.
- Lệnh RET trả điều khiển cho thủ tục gọi . Tất cả các thủ tục phải kết thúc
- bởi RET trừ thủ tục chính .

Ví dụ:

Viết chương trình tính tích của 2 số dương A và B bằng thuật toán cộng (ADD) và dịch (SHIFT)

Thuật toán như sau :

```
Product = 0
REPEAT
IF lsb of B is 1
THEN
product=product+A
END_IF
shift left A
shift right B
UNTIL B=0
```

Viết chương trình

```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
; thực hiện bằng DEBUG . Đặt A = AX , B=BX
CALL MULTIPLY
;DX chứa kết qủa
```

```
MOV AH,4CH
INT 21H
MAIN ENDP
MULTIPY PROC
; input : AX=A , BX=B , AX và BX có giá trị trong khoảng 0...FFH
; output : DX= kết qủa
PUSH AX
PUSH BX
XOR DX,DX
REPEAT:
; Nếu lsb của B =1
TEST BX,1 ;lsb=1?
JZ END_IF ; không , nhảy đến END_IF
; thì
ADD DX,AX ; DX=DX+AX
END_IF :
SHL AX,1 ; dịch trái AX 1 bit
SHR BX,1 ;dịch phải BX 1 bit
; cho đến khi BX=0
JNZ REPEAT ; nếu BX chưa bằng 0 thì lặp
POP BX ; lấy lại BX
POP AX ; lấy lại AX
RET ; trả điều khiển cho chương trình chính
MULTIPLY ENDP
END MAIN
```

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 7

Bài 1: Giải thích các câu lệnh sau

a/
```
MOV AL,5Bh
MOV BL,0ADh
ADD AL,BL
```

b/
```
MOV AX,170Fh
MOV BX,80EBh
ADD AX,BX
```

c/
```
MOV AL,41h
MOV BL,50h
CMP AL,BL
```

Bài 2 :Viết chương trình hiển thị ra màn hình một hình chữ nhật gồm các kí tự ' * '

Bài 3 :Viết chương trình hiển thị chuỗi ' * * * * * * * * ' ra màn hình

Bài 4 : Viết chương trình hiển thị ra màn hình 26 chữ cái(A->Z) và có khỏang trắng ở giữa
Bài 5 : Viết chương trình hiển thị chuỗi " Hello , How are you " ra màn hình
Bài 6 : Viết chương trình hiển thị ra màn hình 26 chữ cái(A->Z) không có khỏang trắng ở giữa
Bài 7 : Viết chương trình hiển thị ra màn hình chuỗi '012345678'
Bài 8:Viết chương trình hiển thị ra màn hình một hình vuông gồm các kí tự ' * ' bên trong.
Bài 9 :Viết chương trình nhập vào 2 số và cho biết số trước lớn hớn , nhỏ hơn hay bằng số sau
Bài 10 :Viết chương trình nhập vào một chuỗi (<=80) kí tự , Viết chương trình nhập vào một kí tự . Cho biết kí tự vừa nhập có trong chuỗi trước hay không
Bài 11 : Viết chương trình nhập vào một chuỗi và hiển thị chuỗi đó ra màn hinh ở dạng chữ hoa
Bài 12 : Viết chương trình nhập vào một kí tự thường và hiển thị kí tự vừa nhập ra màn hình ở dạng chữ hoa
Bài 13 : Viết chương trình in ra ngày sản xuất BIOS
Bài 14 : Viết chương trình nhập vào 2 kí tự . Cho biết kí tự đầu lớn hơn , nhỏ hơn hay bằng kí tự sau
Bài15: Viết chương trình đảo ngược chuỗi số '12345678'
Bài 16 : Viết chương trình nhập vào 3 kí tự cho biết kí tự nào là kí tự lớn nhất
Bài 17 : Viết chương trình nhập vào 1 kí tự , cho biết kí tự đó là kí tự thường ,hoa hay là kí tự đặc biệt

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Đình Việt. *Kiến trúc máy tính*. Nhà xuất bản Đại học quốc Gia Hà Nội. 2007.

[2]. Msc. Võ Văn Chín, Th.s. Nguyễn Hồng Vân. *Giáo trình kiến trúc máy tính*. Khoa CNTT Đại học cần thơ. 2009

[3]. Tống Văn On, Hoàng Đức Hải. *Hợp ngữ & Lập trình ứng dụng.* Nhà xuất bản lao động-xã hội. 2004