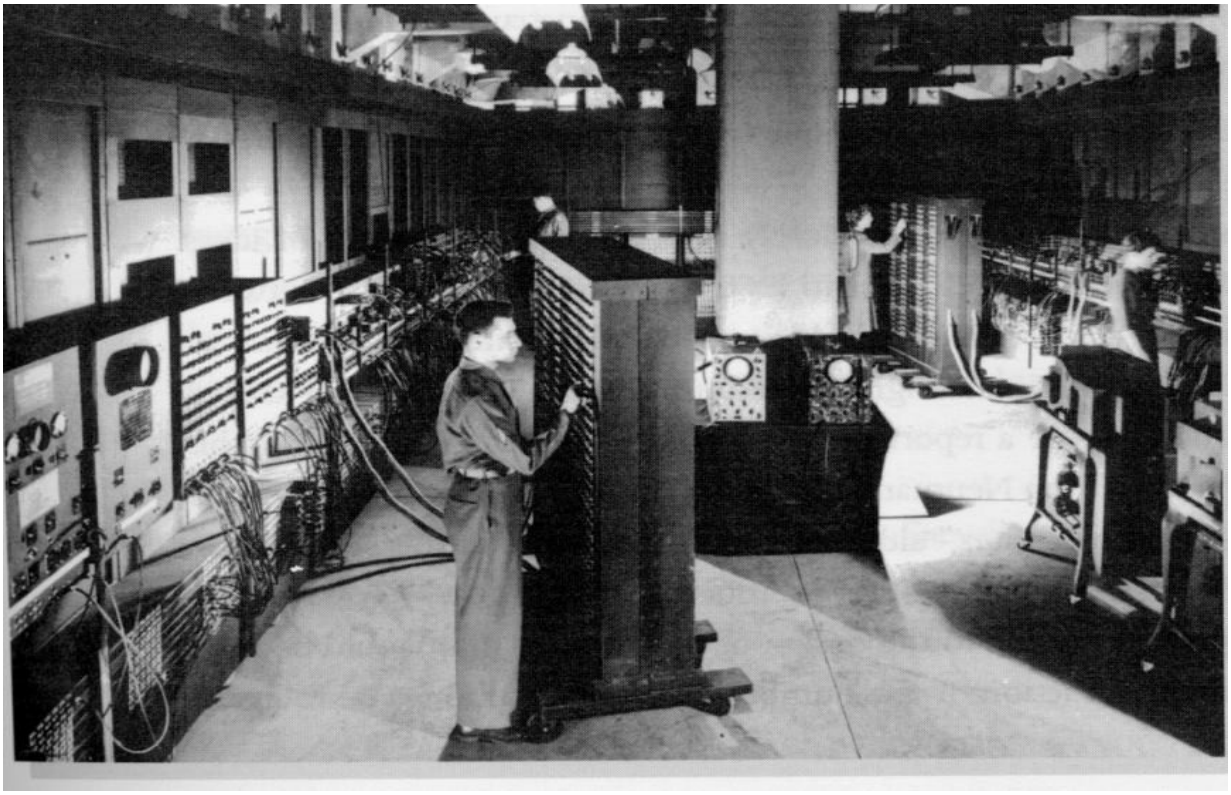
Chương I: Tổng quan về kiến trúc máy tính

1. Các mốc lịch sử phát triển công nghệ máy tính

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

a. Thế hệ đầu tiên (1946-1957)



Hình I.1: Máy tính ENIAC

ENIAC (Electronic Numerical Integrator and Computer) là máy tính điện tử số đầu tiên do Giáo sư Mauchly và người học trò Eckert tại Đại học Pennsylvania thiết kế vào năm 1943 và được hoàn thành vào năm 1946. Đây là một máy tính khổng lồ với thể tích dài 20 mét, cao 2,8 mét và rộng vài mét. ENIAC bao gồm: 18.000 đèn điện tử, 1.500

công tắc tự động, cân nặng 30 tấn, và tiêu thụ 140KW giờ. Nó có 20 thanh ghi 10 bit (tính toán trên số thập phân). Có khả năng thực hiện 5.000 phép toán cộng trong một giây. Công việc lập trình bằng tay bằng cách đấu nối các đầu cắm điện và dùng các ngắt điện.

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

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

b. Thế hệ thứ hai (1958-1964)

Công ty Bell đã phát minh ra transistor vào năm 1947 và do đó thế hệ thứ hai của máy tính được đặc trưng bằng sự thay thế các đèn điện tử bằng các

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

c. Thế hệ thứ ba (1965-1971)

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

Mạch in nhiều lớp xuất hiện, bộ nhớ bán dẫn bắt đầu thay thế bộ nhớ bằng xuyến từ.

Máy tính đa chương trình và hệ điều hành chia thời gian được dùng.

d. Thế hệ thứ tư (1972 - nay)

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

Với sự xuất hiện của bộ vi xử lý (microprocessor) chứa cả phần thực hiện và phần điều khiển của một bộ xử lý, sự phát triển của công nghệ bán dẫn các máy vi tính đã được chế tạo và khởi đầu cho các thế hệ máy tính cá nhân.

Các bộ nhớ bán dẫn, bộ nhớ cache, bộ nhớ ảo được dùng rộng rãi.

Các kỹ thuật cải tiến tốc độ xử lý của máy tính không ngừng được phát triển: kỹ thuật ống dẫn, kỹ thuật vô hướng, xử lý song song mức độ cao,…

e. Khuynh hướng hiện tại

Việc chuyển từ thế hệ thứ tư sang thế hệ thứ 5 còn chưa rõ ràng. Người Nhật đã và đang đi tiên phong trong các chương trình nghiên cứu để cho ra đời thế hệ thứ 5 của máy tính, thế hệ của những máy tính thông minh, dựa trên các ngôn ngữ trí tuệ nhân tạo như LISP và PROLOG,... và những giao diện người - máy thông minh. Đến thời điểm này, các nghiên cứu đã cho ra các sản phẩm bước đầu và gần đây nhất (2004) là sự ra mắt sản phẩm người máy thông minh gần giống với con người nhất: ASIMO (Advanced Step Innovative Mobility: Bước chân tiên tiến của đổi mới và chuyển động). Với hàng trăm nghìn máy móc điện tử tối tân đặt trong cơ thể, ASIMO có thể lên/xuống cầu thang một cách uyển chuyển, nhận diện người, các cử chỉ hành động, giọng nói và đáp ứng một số mệnh lệnh của con người. Thậm chí, nó có thể bắt chước cử động, gọi tên người và cung cấp thông tin ngay sau khi bạn hỏi, rất gần gũi và thân thiện. Hiện nay có nhiều công ty, viện nghiên cứu của Nhật thuê Asimo tiếp khách và hướng dẫn khách tham quan như: Viện Bảo tàng Khoa học năng lượng và Đổi mới quốc gia, hãng IBM Nhật Bản, Công ty điện lực Tokyo. Hãng Honda bắt đầu nghiên cứu ASIMO từ năm 1986 dựa vào

nguyên lý chuyển động bằng hai chân. Cho tới nay, hãng đã chế tạo được 50 robot ASIMO.

Các tiến bộ liên tục về mật độ tích hợp trong VLSI đã cho phép thực hiện các mạch vi xử lý ngày càng mạnh (8 bit, 16 bit, 32 bit và 64 bit với việc xuất hiện các bộ xử lý RISC năm 1986 và các bộ xử lý siêu vô hướng năm 1990). Chính các bộ xử lý này giúp thực hiện các máy tính song song với từ vài bộ xử lý đến vài ngàn bộ xử lý. Điều này làm các chuyên gia về kiến trúc máy tính tiên đoán thế hệ thứ 5 là thế hệ các máy tính xử lý song song.

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

Bảng 1: Các thế hệ máy tính

2. Thông tin và sự mã hóa thông tin

a. Khái niệm thông tin

Hình I.2: 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.

b. Lượng thông tin và sự mã hoá thông tin

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

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

Trong đó: I: là lượng thông tin tính bằng bit

N: là số trạng thái có thể có

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

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

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

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

Trạng thái	**X2**	**X1**	**X0**
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Bảng 2: Tám trạng thái khác nhau ứng với 3 số nhị phân

c. Biểu diễn các số

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

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

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}

$541.25_{10} = 5 * 102 + 4 * 101 + 1 * 100 + 2 * 10^{-1} + 5 * 10^{-2}$

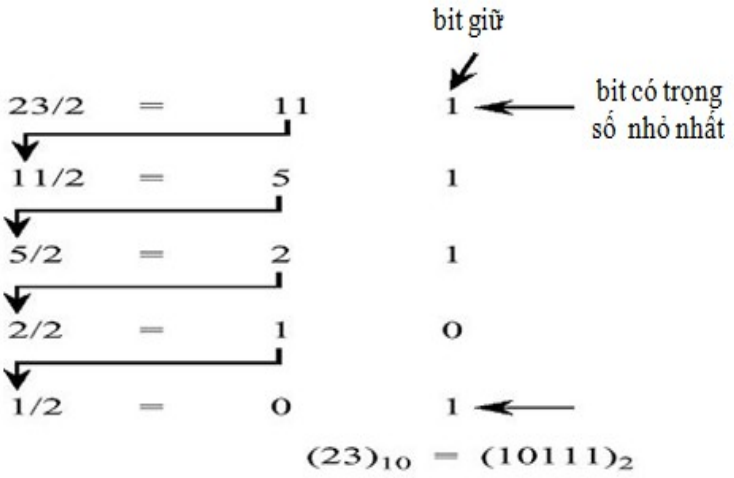
$= (500)_{10} + (40)_{10} + (1)_{10} + (2/10)_{10} + (5/100)_{10}$

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.

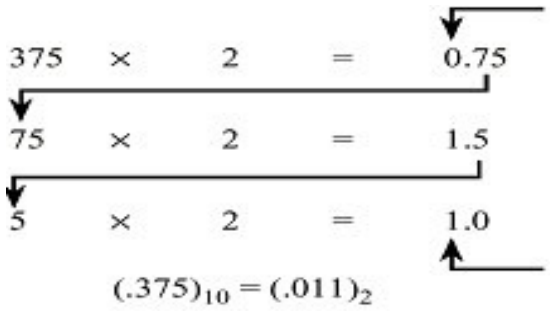
Để biến đổi một số hệ thập phân sang nhị phân, ta có hai phương thức biến đổi:

=Phương thức số dư để biến đổi phần nguyên của số thập phân sang nhị phân.

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



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

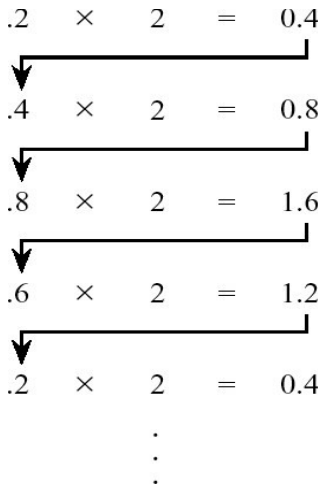


nhất

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:



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.

Binary (Base 2)	**Octal (Base 8)**	**Decimal (Base 10)**	**Hexadecimal (Base 16)**
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C

1101	15	13	D
1110	16	14	E
1111	17	15	F

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

Như vậy, dựa vào cách biến đổi số trong bảng nêu trên, chúng ta có ví dụ về cách biến đổi các số trong các hệ thống số khác nhau theo hệ nhị phân:

- $1011_2 = (10_2)(11_2) = 23_4$
- $23_4 = (2_4)(3_4) = (10_2)(11_2) = 1011_2$
- 1010102 = (1012)(0102) = 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 −1

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

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.

d. Số nguyên có dấu

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

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

d_{n-1} d_{n-2} d_{n-3} . . . . d_2 d_1 d_0

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} .

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.

$$\mathbf{N} = (-1)dn-1 \sum_{i=0}^{n-2} di\ 2i$$

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).

Cách biểu diễn hằng số bù 1

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

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

=Một Byte cho phép biểu diễn tất cả các số có dấu từ -127 ($1000\ 0000_2$) đến 127 ($0111\ 1111_2$)

=Có hai cách biểu diễn cho 0 là 0000 0000 (+0) và 1111 1111 (-0).

Cách biểu diễn bằng số bù 2

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

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

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

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

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

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

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

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

Bảng 3: Số 4 bit có dấu theo cách biểu diễn số âm bằng số bù 2

Cách biểu diễn bằng số thừa K

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

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

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

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

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

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

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

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

e. Cách biểu diễn số với dấu chấm động

Trước khi đi vào cách biểu diễn số với dấu chấm động, chúng ta xét đến cách biểu diễn một số dưới dạng dấu chấm xác định.

Ví dụ:

- Trong hệ thập phân, số 254_{10} có thể biểu diễn dưới các dạng sau:

0 1 2 3 4

254 * 10 ; 25.4 * 10 ; 2.54 * 10 ; 0.254 * 10 ; 0.0254 * 10 ; …

- Trong hệ nhị phân, số $(0.00011)_2$ (tương đương với số 0.09375_{10}) có thể biểu diễn dưới các dạng :

^{0} 0.00011; 0.00011 * 2 ; 0.0011 * 2^{-1}; 0.011 * 2^{-2}; 0.11 * 2^{-3}; 1.1 * 2^{-4}

Các cách biểu diễn này gây khó khăn trong một số phép so sánh các số. Để dễ dàng trong các phép tính, các số được chuẩn hoá về một dạng biểu diễn:

± 1. fff...f x $2^{\pm E}$

Trong đó: f là phần lẻ; E là phần mũ

Số chấm động được chuẩn hoá, cho phép biểu diễn gần đúng các số thập phân rất lớn hay rất nhỏ dưới dạng một số nhị phân theo một dạng qui ước. Thành phần của số chấm động bao gồm: phần dấu, phần mũ và phần định trị. Như vậy, cách này cho phép biểu diễn gần đúng các số thực, tất cả các số đều có cùng cách biểu diễn.

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. Trong cách biểu diễn này, phần định trị có dạng 1,f với số 1 ẩn tăng và f là phần số lẽ.

Chuẩn IEEE 754 định nghĩa hai dạng biểu diễn số chấm động:

- Số chấm động chính xác đơn với định dạng được định nghĩa: chiều dài số: 32 bit được chia thành các trường: dấu S (Sign bit - 1 bit), mũ E (Exponent - 8 bit), phần lẻ F (Fraction - 23 bit).

Số này tương ứng với số thực $(-1)^S * (1,f_1 f_2 ..... f_{23}) * 2^{(E - 127)}$

bit 31 30 23 22 bit 1 bit 0

S	E	f_1	f_2	.....	f22	f23

<u>Hình I.3:</u> Biểu diễn số có dấu chấm động chính xác đơn với 32 bit

- Số chấm động chính xác kép với định dạng được định nghĩa: chiều dài số: 64 bit được chia thành các trường: dấu S (Sign bit - 1 bit), mũ E (Exponent - 11 bit), phần lẻ F (Fraction - 52 bit)

Số này tương ứng với số thực $(-1)^S * (1,f_1\ f_2\ .....\ f_{52}) * 2^{(E\ -\ 1023)}$

bit 63 62 52 51 bit 1 bit 0

S	E	f_1	f_2	.....	f51	f52

<u>Hình I.4:</u> Biểu diễn số có dấu chấm động chính xác kép với 64 bit

Để thuận lợi trong một số phép tính toán, IEEE định nghĩa một số dạng mở rộng của chuẩn IEEE 754:

Tham số	Chính xác đơn	Mở rộng chính xác đơn	Chính xác kép	Mở rộng chính xác kép
Chiều dài (bit)	**32**	**≥ 43**	**64**	**≥ 79**
Chiều dài trường mũ (E)	**8**	**≥ 11**	**11**	**≥ 15**
Số thừa	**127**	**-**	**1023**	**-**
Giá trị mũ tối đa	**127**	**≥ 1023**	**1023**	**≥ 16383**
Giá trị mũ tối thiểu	**-126**	**≤ - 1022**	**-1022**	**≤ -16382**
Chiều dài trường lẻ F	**23**	**≥ 31**	**52**	**≥63**

(bit)				

Chuẩn IEEE 754 cho phép biểu diễn các số chuẩn hoá (các bit của E không cùng lúc bằng 0 hoặc bằng 1), các số không chuẩn hoá (các bit của E không cùng lúc bằng 0 và phần số lẻ f1 f2 ... khác không), trị số 0 (các bit của E không cùng lúc bằng 0 và phần số lẻ bằng không), và các ký tự đặc biệt (các bit của E không cùng lúc bằng 1 và phần lẻ khác không).

Ví dụ các bước biến đổi số thập phân -12.625_{10} sang số chấm động chuẩn IEEE 754 chính xác đơn (32 bit):

- Bước 1: Đổi số -12.625_{10} sang nhị phân: $-12.625_{10} = -1100.101_2$.
- Bước 2: Chuẩn hoá: $-1100.101_2 = -1.100101_2 \times 2^3$ (Số 1.100101_2 dạng 1.f)
- Bước 3: Điền các bit vào các trường theo chuẩn:

Số âm: bit dấu S có giá trị 1.

Phần mũ E với số thừa K=127, ta có: E-127=3

$\Rightarrow E = 3 + 127 = 130\ (1000\ 0010_2)$.

Kết quả nhận được 32 bit

S E F

f. Biểu diễn các số thập phân

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

Số thập phân	$d_3d_2d_1d_0$			Số thập phân	$d_3d_2d_1d_0$		
0	0 0 0 0			5	0 1 0 1		
1	0 0 0 1			6	0 1 1 0		
2	0 0 1 0			7	0 1 1 1		
3	0 0 1 1			8	1 0 0 0		
4	0 1 0 0			9	1 0 0 1		

Bảng 4: Số thập phân mă bằng nhị phân

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

Ví dụ: biểu diễn số $+079_{10}$ bằng số BCD: 0000 0111 1001

Bù 9 1001 0010 0000

+1

Bù 10 1001 0010 0001

Vây, ta có: Số - 079_{10} trong cách biểu diễn số BCD: $1001\ 0010\ 0001_{BCD}$.

Cách tính toán trên tương đương với cách sau:

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

g. Biểu diễn các ký tự

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

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

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

Bảng mã ASCII

Bảng mã

EBCDIC

- **EBCDIC is an 8-bit code.**

00	NUL	20	DS	40	SP	60	–	80		A0		C0	{	E0	\
01	SOH	21	SOS	41		61	/	81	a	A1	~	C1	A	E1	
02	STX	22	FS	42		62		82	b	A2	s	C2	B	E2	S
03	ETX	23		43		63		83	c	A3	t	C3	C	E3	T
04	PF	24	BYP	44		64		84	d	A4	u	C4	D	E4	U
05	HT	25	LF	45		65		85	e	A5	v	C5	E	E5	V
06	LC	26	ETB	46		66		86	f	A6	w	C6	F	E6	W
07	DEL	27	ESC	47		67		87	g	A7	x	C7	G	E7	X
08		28		48		68		88	h	A8	y	C8	H	E8	Y
09		29		49		69		89	i	A9	z	C9	I	E9	Z
0A	SMM	2A	SM	4A	¢	6A	‘	8A		AA		CA		EA	
0B	VT	2B	CU2	4B		6B	,	8B		AB		CB		EB	
0C	FF	2C		4C	<	6C	%	8C		AC		CC		EC	
0D	CR	2D	ENQ	4D	(	6D	_	8D		AD		CD		ED	
0E	SO	2E	ACK	4E	+	6E	>	8E		AE		CE		EE	
0F	SI	2F	BEL	4F	|	6F	?	8F		AF		CF		EF	
10	DLE	30		50	&	70		90		B0		D0	}	F0	0
11	DC1	31		51		71		91	j	B1		D1	J	F1	1
12	DC2	32	SYN	52		72		92	k	B2		D2	K	F2	2
13	TM	33		53		73		93	l	B3		D3	L	F3	3
14	RES	34	PN	54		74		94	m	B4		D4	M	F4	4
15	NL	35	RS	55		75		95	n	B5		D5	N	F5	5
16	BS	36	UC	56		76		96	o	B6		D6	O	F6	6
17	IL	37	EOT	57		77		97	p	B7		D7	P	F7	7
18	CAN	38		58		78		98	q	B8		D8	Q	F8	8
19	EM	39		59		79		99	r	B9		D9	R	F9	9
1A	CC	3A		5A	!	7A	:	9A		BA		DA		FA	|
1B	CU1	3B	CU3	5B	$	7B	#	9B		BB		DB		FB	
1C	IFS	3C	DC4	5C	·	7C	@	9C		BC		DC		FC	
1D	IGS	3D	NAK	5D	)	7D	'	9D		BD		DD		FD	
1E	IRS	3E		5E	;	7E	=	9E		BE		DE		FE	
1F	IUS	3F	SUB	5F	¬	7F	"	9F		BF		DF		FF	

STX	Start of text	RS	Reader Stop	DC
DLE	Data Link Escape	PF	Punch Off	DC
BS	Backspace	DS	Digit Select	DC
ACK	Acknowledge	PN	Punch On	CU
SOH	Start of Heading	SM	Set Mode	CU
ENQ	Enquiry	LC	Lower Case	CU
ESC	Escape	CC	Cursor Control	SY
BYP	Bypass	CR	Carriage Return	IF
CAN	Cancel	EM	End of Medium	EC
RES	Restore	FF	Form Feed	ET
SI	Shift In	TM	Tape Mark	NA
SO	Shift Out	UC	Upper Case	SM
DEL	Delete	FS	Field Separator	SC
SUB	Substitute	HT	Horizontal Tab	IG
NL	New Line	VT	Vertical Tab	IR
LF	Line Feed	UC	Upper Case	IU

Bảng mã UNICODE

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

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

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

*Thế hệ đầu tiên (1938-1953): Dòng đèn điện tử.

Máy tính điện tử tương đồng(Analog computer) đầu tiên được chế tạo năm

1983. Dòng máy tính này dùng các mạch điện có đặc tính giống như phép tính đang được tiến hành để thực hiện các tính tóan trong máy.

Máy tính điện tử số(Electronic Digital Computer) đầu tiên được chế tạo năm 1946. Chúng ta có thể gọi một cách đơn giản là máy tính. Máy tính đầu tiên này là máy ENIAC(Electronic Numerial Integretor and Computer).Máy này dài 30m, cao 2,8m, rộng tới vài mét, nặng khỏang 30 tấn,tiêu thụ 150kW giờ và giá của nó cũng rất cao. Đây cũng là nền tảng cho các thế hệ máy sau này.

* Thế hệ thứ hai (1952-1963)òng Transistar

Nối tiếp thế hệ thứ nhất công ty Bill đã phát minh ra Transistor năm 1948 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.Kích thước máy tính được giảm lại,mạch in và bộ nhớ băng xuyến từ bắt đầu được dùng,ngôn ngữ cấp cao xuất hiện.

* Thế hệ thứ ba (1962-1975): Dòng IC.

Thế hệ này được đánh dấu bằng sự xuất hiện các mạch kết IC (Integration Cireuit). Mạch in nhiều lớp xuất hiện, bộ nhớ bán dẫn bắt đầu thay thế bộ nhớ băng xuyến từ,giá thành cũng giảm đôi chút.

* Thế hệ thứ tư (1972-19??): Dòng IC tích hợp cao(dòng máy chúng ta đang sử dụng)

Thế hệ này được đánh dấu bằng việc dùng các mạch có độ tích hợp cao LSI(Large scale integration). Bộ nhớ bán dẫn CMOS, bộ nhớ cache và bộ nhớ ảo được dùng rộng rãi. Máy tính dùng kĩ thuật ống dẫn(Pipeline) , máy tính song song hoặc song song mức độ cao đã xuất hiện và không ngừng cải tiến.

* Thế hệ tương lai:

Thế hệ này cũng còn chưa được rõ ràng lắm. Chỉ biết là trong tương lai sẽ

xuất hiện một thế hệ máy tính thông minh,có nhiều chức năng, có thể giao tiếp với con người một cách dễ dàng.

4. Kiến trúc và tổ chức máy tính

4.1 Khái niệm kiến trúc máy tính

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

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

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

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

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

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

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

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

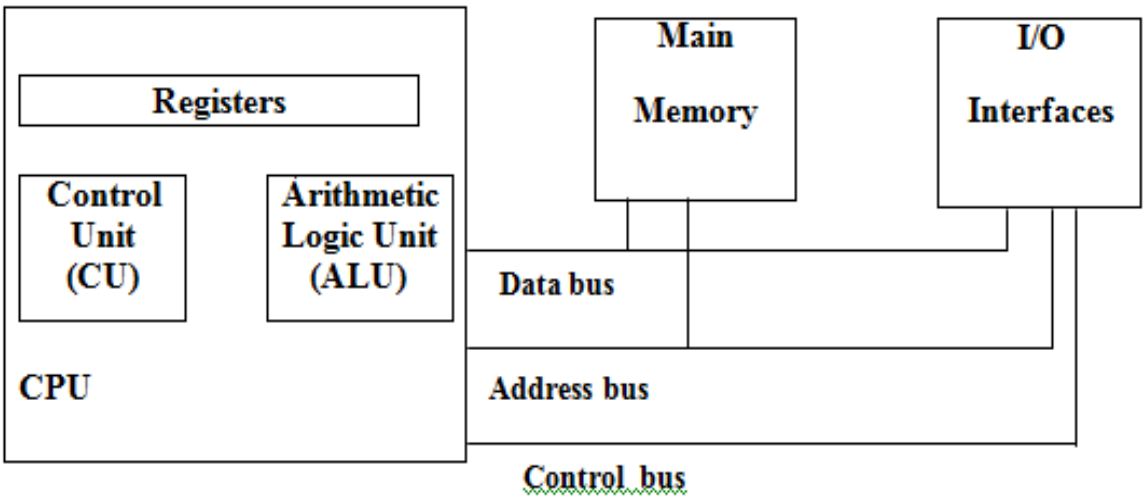
4.2 Khái niệm tổ chức máy tính

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

5. Các mô hình kiến trúc máy tính

5.1 Mô hình kiến trúc Von Neumann

a. **Mô hình chung**



Hình 5.1 Mô hình tuần tự Von Neumann

Registers: Các thanh ghi
Control Unit: đơn vị điều khiển
Arithmetic Logic Unit: Đơn vị số học và logic
CPU: Đơn vị xử lý trung tâm
Main Memory: Bộ nhớ chính
I/O interfaces: các giao diện vào/ra
Data Bus: Bus dữ liệu
Address bus: bus địa chỉ
Cotrol bus: bus điều khiển

Theo mô hình kiến trúc Von Neumann, một máy tính gồm có 3 đơn vị cơ bản là: đơn vị xử lý trung tâm, bộ nhớ và các giao diện vào/ra. Các đơn vị này được kết nối với nhau thông qua hệ thống bus

1) **Đơn vị xử lý trung tâm – CPU (Central Processing Unit)**

Là đơn vị quan trọng nhất của máy tính,điều khiển mọi hoạt động của máy tính và thực hiện các chức năng xử lý dữ liệu.

CPU làm việc với bộ nhớ, các giao diện vào /ra thong qua các bus

Cấu tạo của CPU gồm các bộ phận chính sau

+ Đơn vị điều khiển – CU (Control Unit)

+ Đơn vị số học và logic – ALU (Arithmetic Logic Unit)

+ Các thanh ghi (Registers)

Đơn vị điều khiển: Có các chức năng chính như sau

=====Xác dịnh thứ tự thực hiện các lệnh

=====Lấy lệnh từ bộ nhớ chính,giải mã lệnh

=====Điều khiển ALU và tất cả các thành phần khác để thực hiện các lệnh

=====Thực hiện đồng bộ và phối hợp hoạt động giữa các thành phần trong máy tính

Đơn vị số học và Logic: Thục hiện hầu hết các phép tính cơ bản, quan trong của hệ thống như

=====Thực hiện các phép tính số học: Cộng, trừ, nhân, chia,…

=====Thực hiện các phép tính Logic: and, or, not, xor, các phép so sánh: bắng nhau, khác nhau, lớn hơn, nhỏ hơn,…

Các thanh ghi: là những bộ nhớ có kích thước nhỏ nhưng tốc độ truy cập dữ liệu rất nhanh, dung làm nhiệm vụ lưu trữ lệnh và dữ liệu trung gian, phục vụ cho quá trình xử lý lệnh và dữ liệu tại CPU

2) **Bộ nhớ chính.**

Là một trong những thành phần quan trọng của máy tính, là nơi lưu trữ các lệnh chương trình và dữ liệu. bộ nhớ được tổ chức từ nhiều ô nhớ(hay còn gọi là các từ nhớ) . Mỗi ô nhớ được gán một địa chỉ để CPU quản lý truy cập; địa chỉ này được gọi là địa chỉ bộ nhớ.

Bộ nhớ chính gồm hai loại như sau:

+ Bộ nhớ chỉ đọc- ROM(real only memory)

+ Bộ nhớ truy cập ngẫu nhiên – RAM (Random access memory)

Bộ nhớ chỉ đoc: thường dung để lưu trữ các thong số của máy tính, chứa các chương trình cơ bản phục vụ quá trình khởi động máy tính….

Đặc điểm chính

═══Tốc độ truy cập dữ liệu chậm ơn RAM,

═══Chỉ cho phép đọc dữ liệu, không cho phép ghi dữ liệu

═══Khi bị mất điện dữ liệu lưu trong ROM không bị mất

Bộ nhớ truy cập ngẫu nhiên: Thường dùng để luuw trữ tạm thời các chương trình và dữ liệu

Đặc điểm chính

═══Tốc độ truy cập dữ liệu nhanh hơn ROM

═══Cho phép đọc và ghi dữ liệu

═══Khi bị mất nguồn điện, dữ liệu lưu trong RAM sẽ bị mất đi

3) **Các giao diện vào/ra**

Dùng để kết nối CPU, bộ nhớ chính với các thiết bị ngoại vi như: đĩa cứng, đĩa mềm, bàn phím , màn hình, máy in,…

Cũng giống như các ô nhớ trong bộ nhớ chính, mỗi thiết bị ngoại vi cũng được gán một địa chỉ để CPU quản lý truy cập, để phân biệt người ta thường gọi địa chỉ này là địa chỉ cổng vào ra (I/O port)

4) **Bus**

CPU, bộ nhớ, các giao diện vào/ra được kết nối với nhau thông qua các bus. Bus là tập hợp các đường dẫn song song để truyền tín hiệu điện giữa CPU, bộ nhớ và các giao diện vào /ra. CPU được nối với các thành phần khác bằng bus hệ thống nghĩa là sẽ có nhiều thiết bị cùng dung chung một hệ thống dây dẫn để trao đổi dữ liệu. Để hệ

thống không bị xung đột , CPU phải xử lý sao cho trong một thời điểm , chỉ có một thiết bị hay ô nhớ đãchỉ định mới có thể chiếm lĩnh hệ thống. Do mục đích này bus hệ thống bao gồm 3 loại : bus dữ liệu, bus địa chỉ và bus điều khiển.

b. **Nguyên lý hoạt động**

Nguyên lý:

Một tập hợp các lệnh được sắp xếp theo một trật tự nhất định được gọi là một chương trình. Theo mô hình kiến trúc VonNeumann, chương trình và dữ liệu đuọc lưu trữ trong cùng một bộ nhớ (thường lưu trữ trong bộ nhớ RAM).

CPU sẽ thực hiện lần lượt từng lệnh của chương trình theo quy trình sau: CPU lấy lệnh và dữ liệu từ địa chỉ được lưu trong thanh ghi PC (PC : program Counter – Bộ đếm chương trình); sau khi CPU lấy lệnh và dữ liệu xong, PC tự động tăng lên một giá trị tiếp theo; CPU thực hiện lệnh xong lại tiếp tục lấy lệnh và dữ liệu kế tiếp từ địa chỉ được lưu trong PC

Quy trình thực hiện lệnh

Cấu trúc tổng quát của một lệnh gồm có trường mã lệnh và trường các toán hạng

Mã lệnh	Các toán hạng

(Opcode)	(Operands)

Trường mã lệnh: dung các tham số cần thiết để thực hiện

Trường các toán hạng: chứa các tham số cần thiết để thực hiện lệnh được mô tả trong trường mã lệnh

Chương trình được nạp vào bộ nhớ chính, sau đó CPU sẽ thực hiện tuần tự từng lệnh. Mỗi lệnh thường đươc thực hiện theo các giai đoạn sau:

IF-> ID -> OF -> EX -> WB

Giai đoạn 1: Nạp lênh IF (Ínstruction Fetch)

Giá trị hiện thời của bộ đếm chương trình (PC: Program Counter) cho biết địa chỉ của ô nhớ chúa lệnh cần phải thực hiện. Căn cứ vào giá trị địa chỉ này , CPU lấy nội dung lệnh chương trình cần thực hiện từ bộ nhớ chính nạp vào thanh ghi lênh IR (Ínstruction Register). Sau đó đơn vị điều khiển CU sẽ thưcj hiện tăng giá trị bộ đếm chương trình PC lên một đơn vị để đến địa chỉ của lệnh tiếp theo.

Giai đoạn 2: Giải mã lệnh ID (Ínstruction Decode)

Ở giai đoạn này đơn vị điều khiển tiến hành giải mã lệnh, xác địn loại lệnh vừa nạp yêu cầu CPU thực hiện phép tính, phép xử lý gì.

Giai đoạn 3: nạp toán hạng lên OF (Operands Fetch)

Nếu lệnh cần them dữ liệu trong bộ nhớ thì đơn vị điều khển sẽ xác định địa chỉ nơi chứa dữ liệu; tìm và nạp dữ liệu vào các thanh ghi trong CPU (giai đoạn này có thể có hoặc không có tùy thuộc vào số toán hạng của lệnh)

Giai đoạn 4: Thực hiện lệnh EX (Excutive)

Giai đoạn này thực hiện lệnh sau khi đã giải mã lệnh và nạp toán hạng

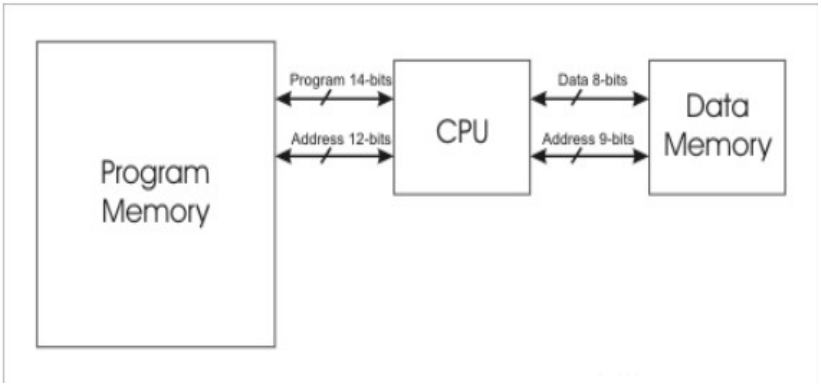
+ nếu lệnh là các phép toán số học và logic thì đơn vị điều khiển sẽ điều khiển bộ ALU thực hiện lệnh này

+ Nếu lệnh là các lệnh điều khiển thì đơn vị điều khiển sẽ sinh ra các tín hiệu điều khiển tương ứng (đọc bộ nhớ, ngắt chương trình, …)

Giai đoạn 5: Lưu trữ kết quả thực hiện lệnh WB (Write Back)

Kết quả của giai đoạn thực hiện lệnh sẽ được ghi vào toán hạng đích, cụ thể giai đoạn này sẽ viết các kết quả, các dữ liệu vào các thanh ghi hoặc bộ nhớ tùy theo yêu cầu của lệnh.

5.2 Mô hình kiến trúc Havard



Hình 5.2: Mô hình kiến truc Havard

Kiến trúc máy tính Havard chia bộ nhớ trong thành hai phần riêng rẽ , bộ nhớ lưu chương trình (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 bú 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 truc Havard có khả năng đạt đượctốc độ xử lý cao hownmays tính dựa trên kiến truc Von Neumann do kiến truc Havard hỗ trợ hai hệ thống bú độc lập và băng thông lớn hơn. Ngoài ra, nhờ có hai hệ thống bus độc lập , hệ thống nhớ trong kiến trúc Havard hỗ trợ nhiều lện truy nhập bộ nhớ tại một thời điểm, giúp giảm xung đột truy nhaapjbooj nhớ, đặc biệt khi CPU sử dụng kỹ thuật đường ống (pipeline)

HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG I

*****

Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ?

1. Đặc trưng cơ bản của các máy tính thế hệ thứ nhất?

2. Đặc trưng cơ bản của các máy tính thế hệ thứ hai?

3. Đặc trưng cơ bản của các máy tính thế hệ thứ ba?

4. Đặc trưng cơ bản của các máy tính thế hệ thứ tư?

5. Khuynh hướng phát triển của máy tính điện tử ngày nay là gì?

6. Việc phân loại máy tính dựa vào tiêu chuẩn nào?

7. Khái niệm thông tin trong máy tính được hiểu như thế nào?

8. Lượng thông tin là gì ?

9. Sự hiểu biết về một trạng thái trong 4096 trạng thái có thể có ứng với lượng thông tin là bao nhiêu?

10. Điểm chung nhất trong các cách biểu diễn một số nguyên n bit có dấu là gì?

11. Số nhị phân 8 bit $(11001100)_2$, số này tương ứng với số nguyên thập phân có dấu là bao nhiêu nếu số đang được biểu diễn trong cách biểu diễn:

a. Dấu và trị tuyệt đối.

b. Số bù 1.

c. Số bù 2.

12. Đổi các số sau đây:

a. $(011011)_2$ ra số thập phân.

b. $(-2005)_{10}$ ra số nhị phân 16 bits.

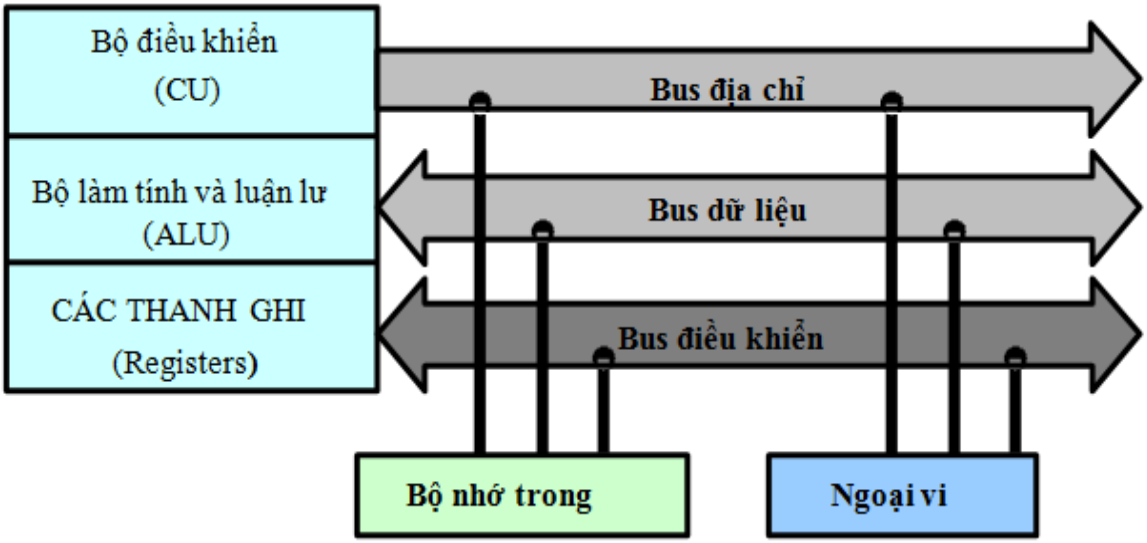
Chương II : Kiến trúc tập lệnh của máy tính

1. Thành phần cơ bản của một máy tính

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

giữa các bộ phận vào-ra và bộ nhớ trong.

Bộ xử lý trung tâm (CPU)



Hình II.1: Cấu trúc của một hệ máy tính đơn giản

Một chương trnh sẽ được sao chép từ đĩa cứng vào bộ nhớ trong cùng với các thông tin cần thiết cho chương trnh 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).

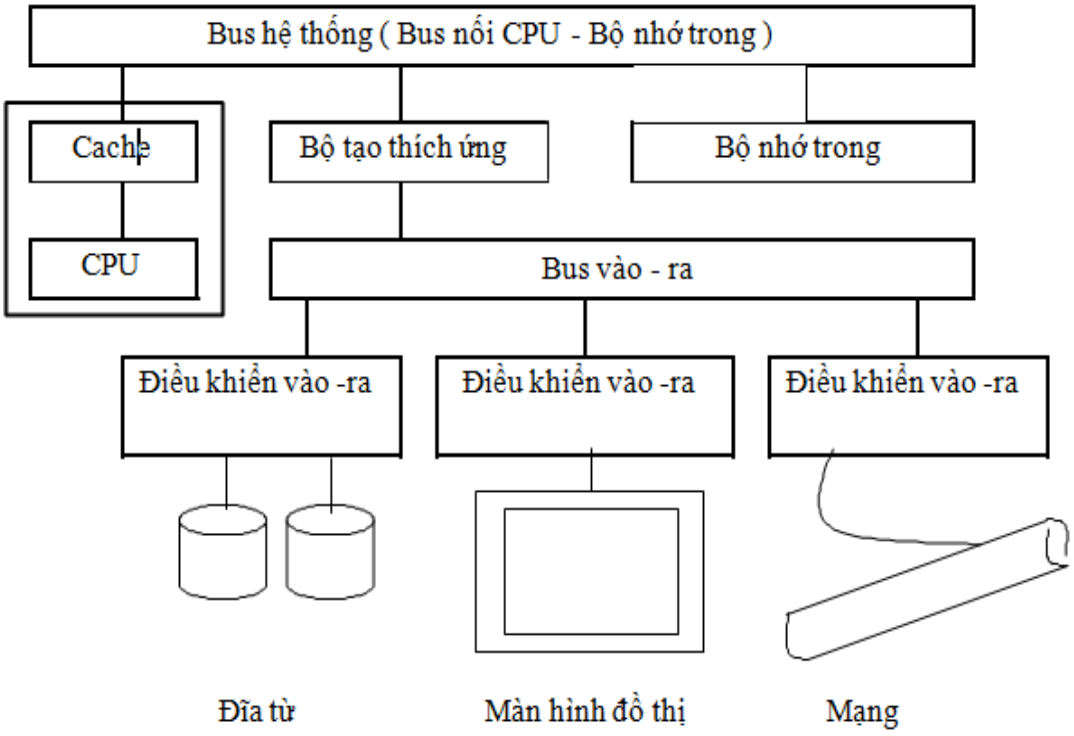
Thành phần cơ bản của một máy tính bao gồm :

- **Bộ nhớ trong:** Đây là một tập hợp các ô nhớ, mỗi ô nhớ có một số bit nhất định và chức một thông tin được mã hoá thành số nhị phân mà không quan tâm đến kiểu của dữ liệu mà nó đang chứa. Các

thông tin này là các lệnh hay số liệu. Mỗi ô nhớ của bộ nhớ trong đều có một địa chỉ. Thời gian thâm nhập vào một ô nhớ bất kỳ trong bộ nhớ là như nhau. Vì vậy, bộ nhớ trong còn được gọi là bộ nhớ truy cập ngẫu nhiên (RAM: Random Access Memory). Độ dài của một từ máy tính (Computer Word) là 32 bit (hay 4 byte), tuy nhiên dung lượng một ô nhớ thông thường là 8 bit (1 Byte).

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

- **Bộ phận vào - ra:**đây là bộ phận xuất nhập thông tin, bộ phận này thựchiện sự giao tiếp giữa máy tính và người dùng hay giữa các máy tính trong hệ thống mạng (đối với các máy tính được kết nối thành một hệ thống mạng). Các bộ phận xuấtnhập thường gặp là: bộ lưu trữ ngoài, màn hình, máy in, bàn phím, chuột, máy quét ảnh, các giao diện mạng cục bộ hay mạng diện rộng...Bộ tạo



thích ứng là một vi mạch tổnghợp (chipset) kết nối giữa các hệ thống bus có các tốc độ dữ liệu khác nhau.

Hình II.2: Sơ đồ mô tả hoạt động điển hình của một máy tính

2. Kiến trúc các tập lệnh CISC và RISC

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

Bảng II.3 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.

Bộ xử lý	IBM 370/168	DEC 11/780	iAPX 432

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

Bảng II.3: Đặc tính của một vài máy CISC

Ví dụ, chương trình dịch đã biết sử dụng các thanh ghi và không có sự khác biệt đáng kể nào khi sử dụng ô nhớ cho các vi chương trình hay ô nhớ cho các chương trình. Điều này dẫn tới việc đưa vào khái niệm về một máy tính với tập lệnh rút gọn RISC vào đầu những năm 1980. Các máy RISC dựa chủ yếu trên một tập lệnh cho phép thực hiện kỹ thuật ống dẫn một cách thích hợp nhất bằng cách thiết kế các lệnh có chiều dài cố định, có dạng đơn giản, dễ giải mã. Máy RISC dùng kiểu thực hiện lệnh thanh ghi - thanh ghi. Chỉ có các lệnh ghi hoặc đọc ô nhớ mới cho phép thâm nhập vào ô nhớ. Bảng II.4 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 II.4 : 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 của phần này là dùng các ví dụ trích từ các kiến trúc phần mềm được dùng nhiều nhất, để cho thấy các kỹ thuật ở mức ngôn ngữ máy dùng để thi hành các cấu trúc trong các ngôn ngữ cấp cao.

Để minh hoạ bằng thí dụ, ta dùng cú pháp lệnh trong hợp ngữ sau đây :

Từ gợi nhớ mã lệnh, thanh ghi đích, thanh ghi nguồn 1, thanh ghi nguồn 2.

Từ gợi nhớ mã lệnh mô tả ngắn gọn tác vụ phải thi hành trên các thanh ghi nguồn, kết quả được lưu giữ trong thanh ghi đích.

Mỗi lệnh của ngôn ngữ cấp cao được xây dựng bằng một lệnh mã máy hoặc một chuỗi nhiều lệnh mã máy. Lệnh nhảy (GOTO) được thực hiện bằng các lệnh hợp ngữ về nhảy (JUMP) hoặc lệnh hợp ngữ về vịng. Chúng ta phân biệt lệnh nhảy làm cho bộ đếm chương trình được nạp vào địa chỉ tuyệt đối nơi phải nhảy đến (PC ←địa chỉ tuyệt đối nơi phải nhảy tới), với lệnh vịng theo đó ta chỉ cần cộng thêm một độ dời vào bộ đếm chương trình (PC ← PC + độ dời). Ta lưu ý là trong trường hợp sau, PC chứa địa chỉ tương đối so với địa chỉ của lệnh sau lệnh vịng.

a. 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

STORE Ri, M(địa chỉ) ; Ri ← M[địa chỉ]

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)

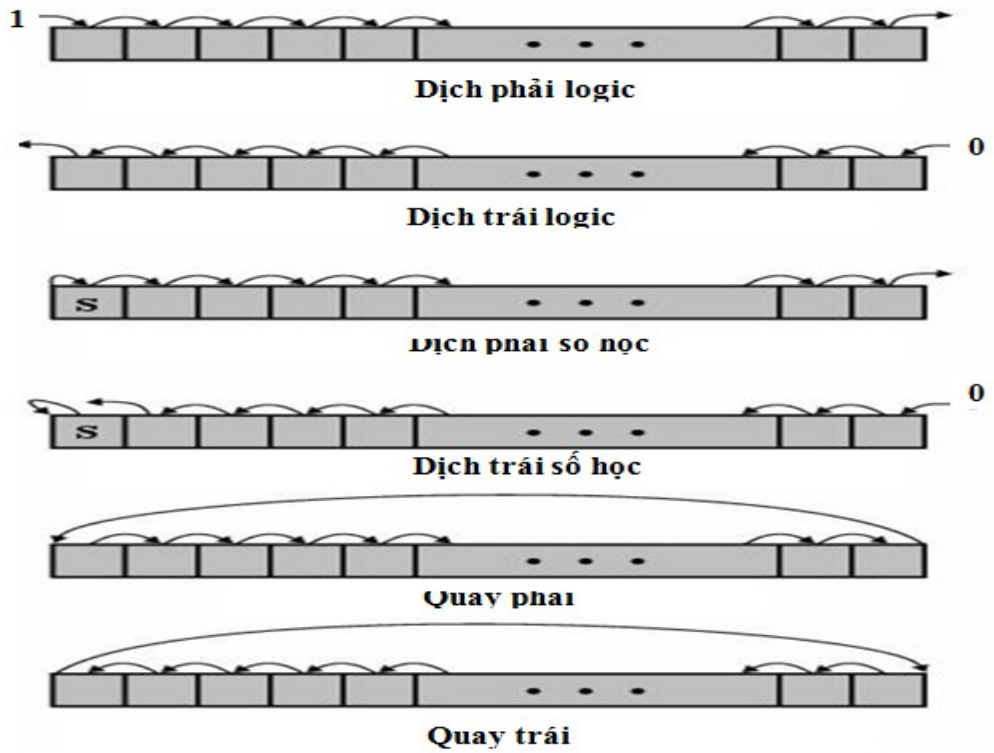
- **Lệnh logic:** thực hiện phép tính logic cho từng bit một.

AND (lệnh VÀ)

OR (lệnh HOẶC)

XOR (lệnh HOẶC LOẠI)

NEG (lệnh lấy số bù 1)



- Các lệnh dịch chuyển số học hoặc logic (SHIFT), quay vịng (ROTATE) có hoặc không có số giữ ở ngă 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 II.7 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)

b. - Lệnh có điều kiện

Lệnh có điều kiện có dạng :

Nếu <điều kiện> thì <chuỗi lệnh 1> nếu không <chuỗi lệnh 2>

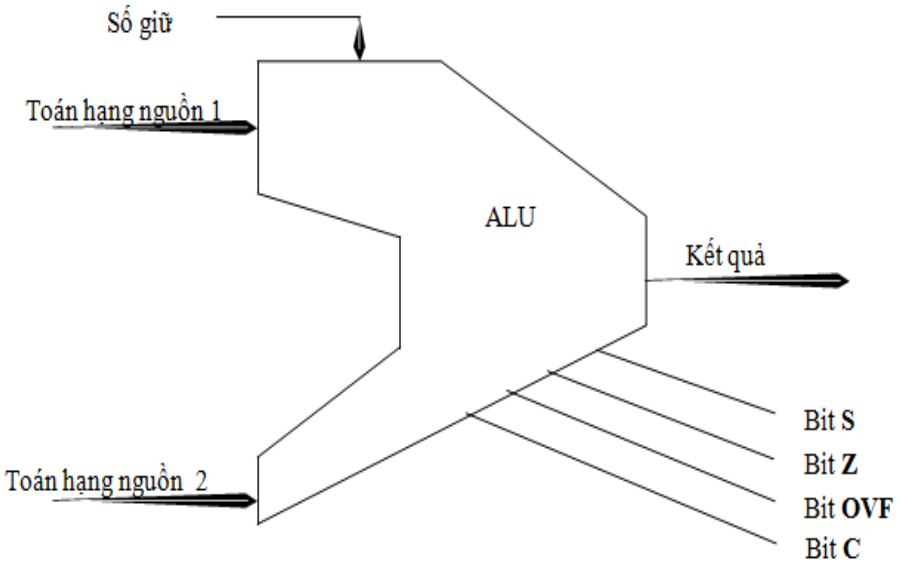
(IF <condition> THEN <instructions1> ELSE <instructions2>)

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

Ghi nhớ điều kiện .

Bộ làm tính ALU cung cấp kết quả ở ngă ra tuỳ theo các ngă 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 II.8).

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 ngă ra là 1 .... Các bit trên thường được gọi là bit mă điều kiện.



Hình II.8 : 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 tr ạng 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, ch ỉ 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.

Nhảy vòng

Các lệnh nhảy hoặc nhảy vọng 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
4. BRA +1	: nhảy bỏ 1 lệnh
5. ADD R3, R0, R1	: chuyển nội dung R1 vào R3
6. Lệnh kế	

Nếu R1 > R2 th´chuỗi lệnh được thi hành là 1, 2, 5, 6 được thi hành, nếu không th´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 thR3 = R1 nếu không R3 = R2

Các lệnh nhảy làm tốc độ thi hành lệnh chậm lại, trong các CPU hiện đại dung 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

c. - 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.

d. 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 lấp ra khỏi ngăn xếp dùng các lệnh bình thường. Ví dụ thanh ghi R30 là con trỏ ngăn xếp thì việc viết vào ngăn xếp được thực hiện bằng các lệnh:

ADDI R30, R30, 4 ; tăng con trỏ ngăn xếp lên 4 vì từ dài 32 bit STORE Ri, (R30) ; Viết Ri vào đỉnh ngăn xếp

Việc lấy ra khỏi ngăn xếp được thực hiện bằng các lệnh :

LOAD Ri, (R30.) ; lấy số liệu ở đỉnh ngăn xếp và nạp vào Ri

SUBI R30, R30,4 ; giảm con trỏ ngăn xếp bớt 4

e. 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.

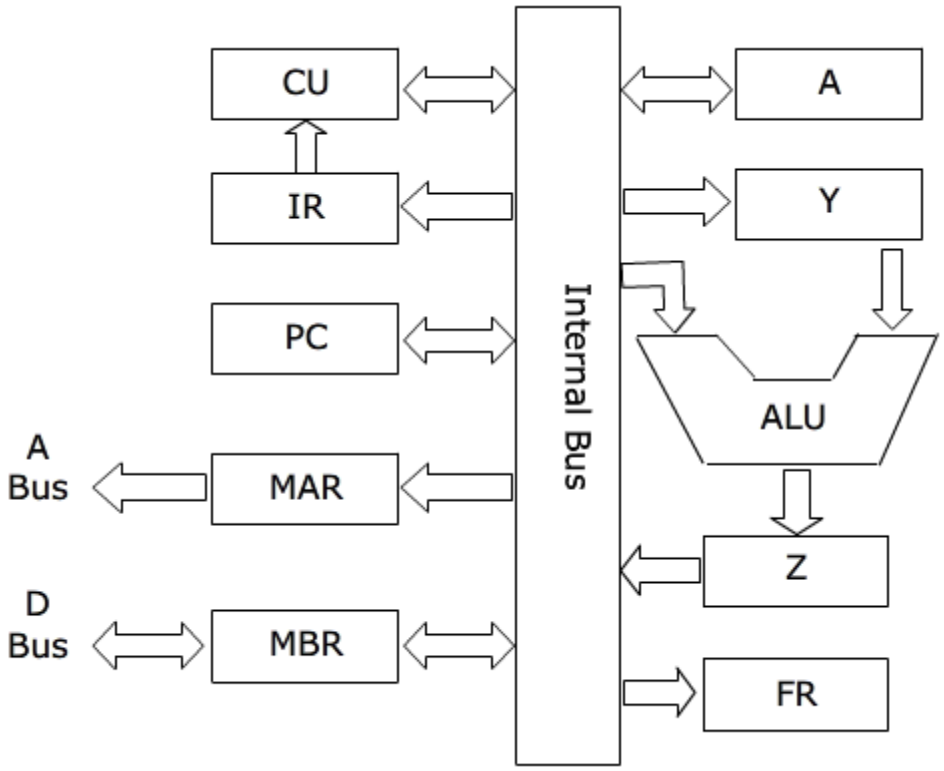
Trong các kiến trúc CISC (VAX 11, 80x86, 680x0), địa chỉ trở về được giữ ở ngăn xếp. Trong các kiến trúc RISC, một thanh ghi đặc biệt (thường là thanh ghi R31) được dùng để lưu giữ địa chỉ trở về.

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

Chương III: Bộ xử lý

1. **Sơ đồ khối của bộ xử lý**



Hinh III.1 Sơ đồ khối tổng quát của CPU

Các thành phần của CPU:

==Bộ điều khiển (Control Unit – CU)

==Bộ tính toán số học và logic (Arithmeric and Logic Unit)

==Bus trong CPU (CPU Internal Bus)

==Các thanh ghi CPU:

+ Thanh ghi tích lũy A (Accummulator)

+ Bộ đếm chương trinh PC (Program Counter)

+ Thanh ghi lệnh IR (Instruction Register)

+ Thanh ghi địa chỉ bộ nhớ MAR (memory Address Register)

+ Thanh ghi đệm dữ liệu MBR (Memory Buffer Register)

+ Các thanh ghi tạm thời Y và Z

+ Thanh ghi cờ FR (Flag Register)

Đơn vị xử lý trung tâm CPU (Central proccessing Unit) gọi tắt là bộ xử lý ,là bộ phận quan trọng nhất của máy tính .Khi được thiết kế chế tạo trên một chop ,nó được gọi là bộ vi xử lý (microprocessor).Như trên đã nói ,nó có nhiệm vụ thực thi các chưng trình chứa sẵn trong bộ nhớ chính bằng cách tìm –nạp (fetch) các lện của chương trình ,giải mã chúng và sau đó thực hiện từng bước các thao tác trong lệnh ,lệnh nọ tiếp theo lệnh kia .Có thể lấy vi xử lý 8086 của hãng Intel làm ví dụ điển hình để nói vấn đề tắt về các đơn vị trong bộ xử lý này .Nó gồm 2 phần chính là đơn vị ghép nối bus BIU (bus interface unit) và đơn vị thực hiện lệnh EU (excute unit)như chỉ ra trên hình 2.17.BIU thực hiện tất cả các nhiệm vụ về bus cho EU .Nó thiết lập khâu nối với thế giới bên ngoài là các bus số liệu , địa chỉ và điều khiển .BIU bao gồm các thanh ghi , đoạn nhớ ,thanh ghi con trỏ lệnh và bộ điều khiển lôgic bus mà ta sẽ đề cập chi tiết về sau

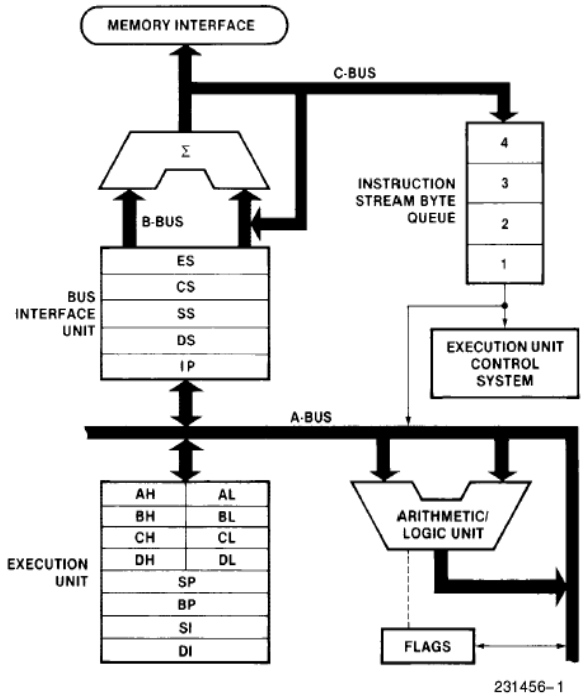


Figure 1. 8088 CPU Functional Block Diagram

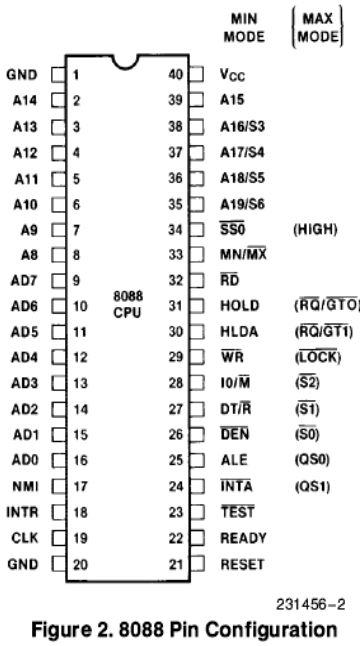


Figure 2. 8088 Pin Configuration

2. Đường dẫn dữ liệu

Phần đường đi dữ liệu gồm có bộ phận làm tính và luận lý (ALU: Arithmetic and Logic Unit), các mạch dịch, các thanh ghi và các đường nối kết các bộ phận trên. Phần này chứa hầu hết các trạng thái của bộ xử lý. Ngoài các thanh ghi tổng quát, phần đường đi dữ liệu còn chứa thanh ghi đếm chương trình (PC: Program Counter), thanh ghi trạng thái (SR: Status Register), thanh ghi đệm TEMP (Temporary), các thanh ghi địa chỉ bộ nhớ (MAR: Memory Address Register), thanh ghi số liệu bộ nhớ (MBR: Memory Buffer Register), bộ đa hợp (MUX: Multiplexor), đây là điểm cuối của 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).

Nhiệm vụ chính của phần đường đi dữ liệu là đọc các toán hạng từ các thanh ghi tổng quát, thực hiện các phép tính trên toán hạng này trong bộ làm tính và luận lý ALU và lưu trữ kết quả trong các thanh ghi tổng quát. Ở ngã vào và ngã ra các thanh ghi tổng quát có các mạch chốt A, B, C. Thông thường, số lượng các thanh ghi tổng quát 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 III.2: Tổ chức của một xử lý điển hình

(Các đường không liên tục là các đường điều khiển)

3. Bộ điều khiển

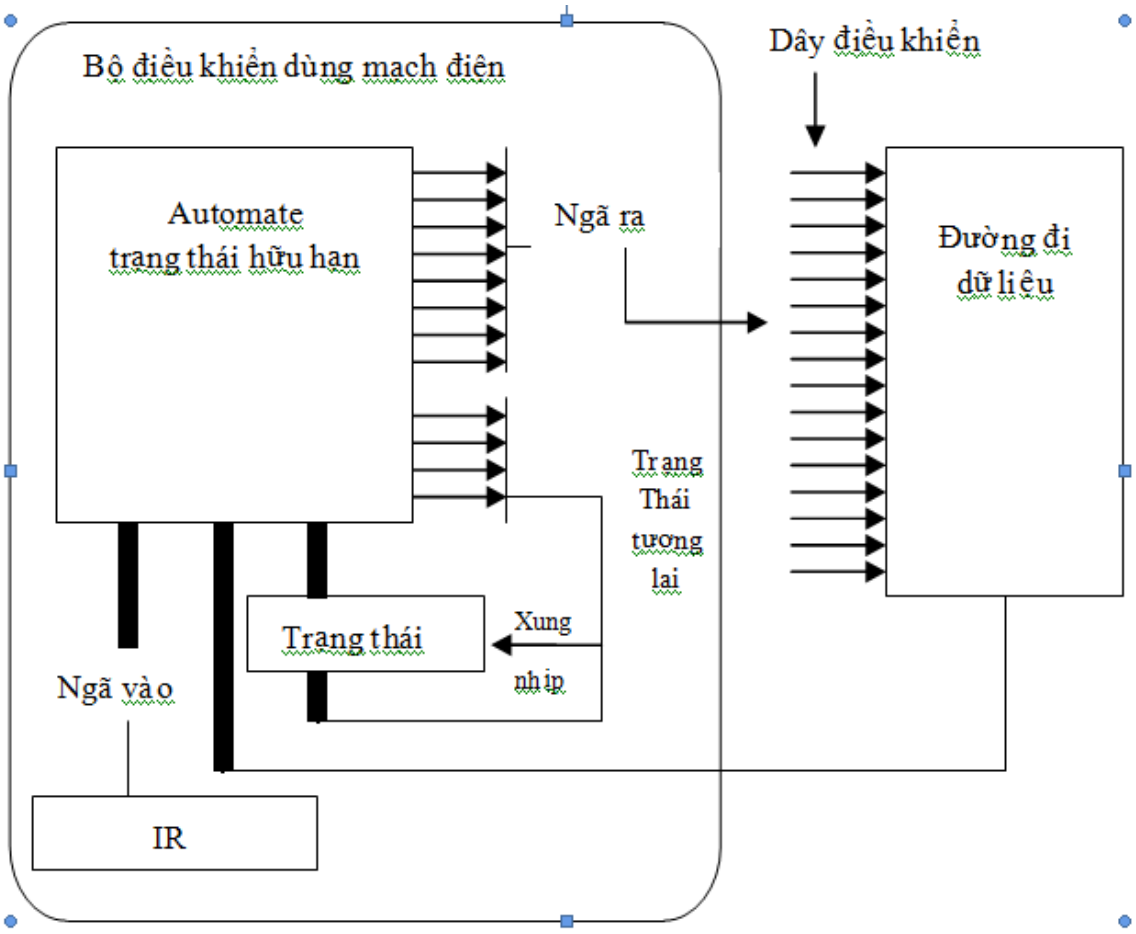
Bộ điều khiển tạo các tín hiệu điều khiển di chuyển số liệu (tín hiệu di chuyển số liệu từ các thanh ghi đến bus hoặc tín hiệu viết vào các thanh ghi), điều khiển các tác vụ mà các bộ phận chức năng phải làm (điều khiển ALU, điều khiển đọc và viết vào bộ nhớ trong...). Bộ điều khiển cũng tạo các tín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

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

a. Bộ điều khiển mạch điện tử

Để hiểu được vận hành của bộ điều khiển mạch điện tử, chúng ta xét đến mô tả về Automate trạng thái hữu hạn: có nhiều hệ thống hay nhiều thành phần mà ở mỗi thời điểm xem xét đều có một trạng thái (state). Mục đích của trạng thái là ghi nhớ những gì có liên quan trong quá trình hoạt động

của hệ thống. Vì chỉ có một số trạng thái nhất định nên nói chung không thể ghi nhớ hết toàn bộ lịch sử của hệ thống, do vậy nó phải được thiết kế cẩn thận để ghi nhớ những gì quan trọng. Ưu điểm của hệ thống (chỉ có một số hữu hạn các trạng thái) đó là có thể cài đặt hệ thống với một lượng tài nguyên cố định. Chẳng hạn, chúng ta có thể cài đặt Automate trạng thái hữu hạn trong phần cứng máy tính ở dạng mạch điện hay một dạng chương trình đơn giản, trong đó, nó có khả năng quyết định khi chỉ biết một lượng giới hạn dữ liệu hoặc bằng cách dùng vị trí trong đoạn mã lệnh để đưa ra quyết định.

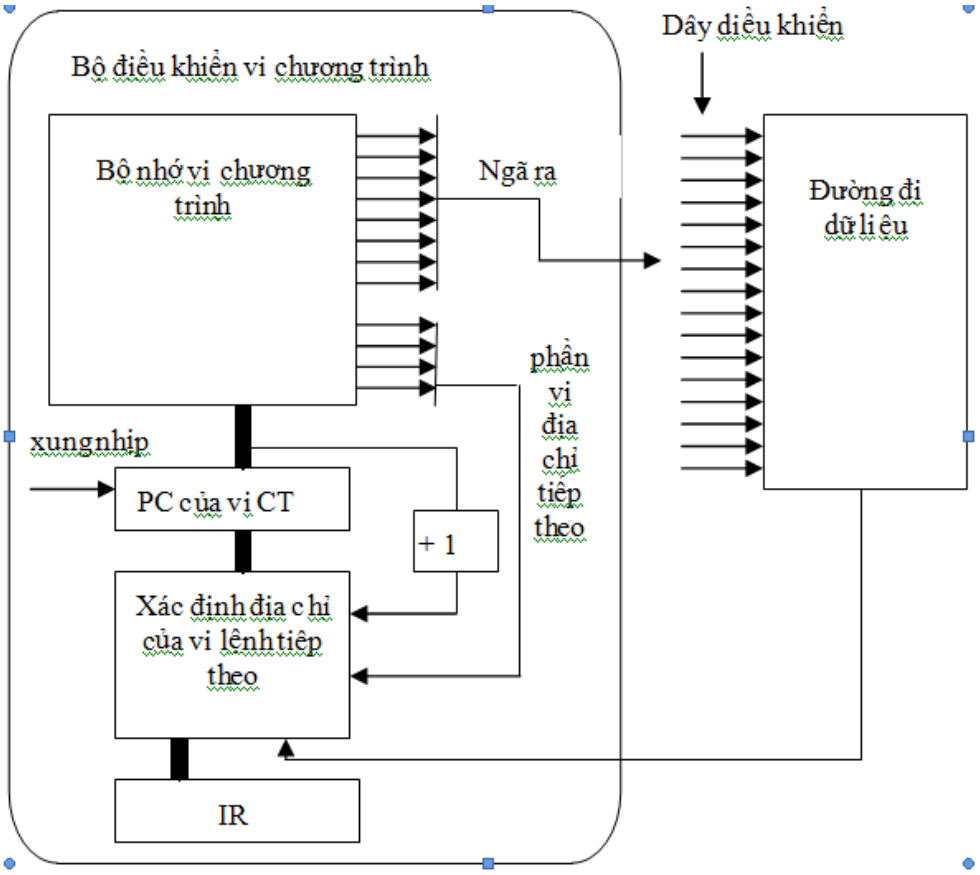


Hình III.3: Nguyên tắc vận hành của bộ điều khiển dùng mạch điện

Hình III.3 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.

b. **Bộ điều khiển vi chương trình:**



<u>Hình III.4</u>: Nguyên tắc vận hành của bộ điều khiển vi chương trình

Sơ đồ nguyên tắc của bộ điều khiển dùng vi chương trình được trình bày ở hình III.4. Trong kỹ thuật này, các đường dây điều khiển của bộ đường đi dữ liệu ứng với các ngã ra của một vi lệnh nằm trong bộ nhớ vi chương trình. Việc điều khiển các tác vụ của một lệnh mã máy được thực hiện bằng một chuỗi các vi lệnh. Một vi máy tính nằm bên trong bộ điều khiển thực hiện từng lệnh của vi chương trình này. Chính vi máy tính này điều khiển việc thực hiện một cách tuần tự các vi lệnh để hoàn thành tác vụ mà lệnh mã máy phải

thực hiện. Các tác vụ của lệnh mã máy cũng tuỳ thuộc vào trạng thái của phần đường đi dữ liệu.

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

4. Tiến trình thực hiện lệnh 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.

Đọc lệnh:

MAR ← PC

IR ← M[MAR]

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

Giải mã lệnh và đọc các thanh ghi nguồn:

a.i.A ← Rs1

a.i.B ← Rs2

PC ← PC + 4

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

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

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

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

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

Thi hành lệnh:

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

- Liên hệ tới bộ nhớ

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

MBR ← Rs1

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

- Một lệnh của ALU

Ngã ra ALU ← Kết quả của phép tính

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

- Một phép nhảy

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

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

=Thâm nhập bộ nhớ trong hoặc nhảy lần cuối

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

=Tham khảo đến bộ nhớ:

MBR ← M[MAR] hoặc M[MAR] ← MBR

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

=Nhảy:

If (điều kiện), PC ← ngả ra ALU

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

═Lưu trữ kết quả

Rd ← Ngã ra ALU hoặc Rd ← MBR

Lưu trữ kết quả trong thanh ghi đích.

5. Kỹ thuật ống dẫn lệnh

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

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

Hình III.4 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 III.5: 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 III.5, 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.

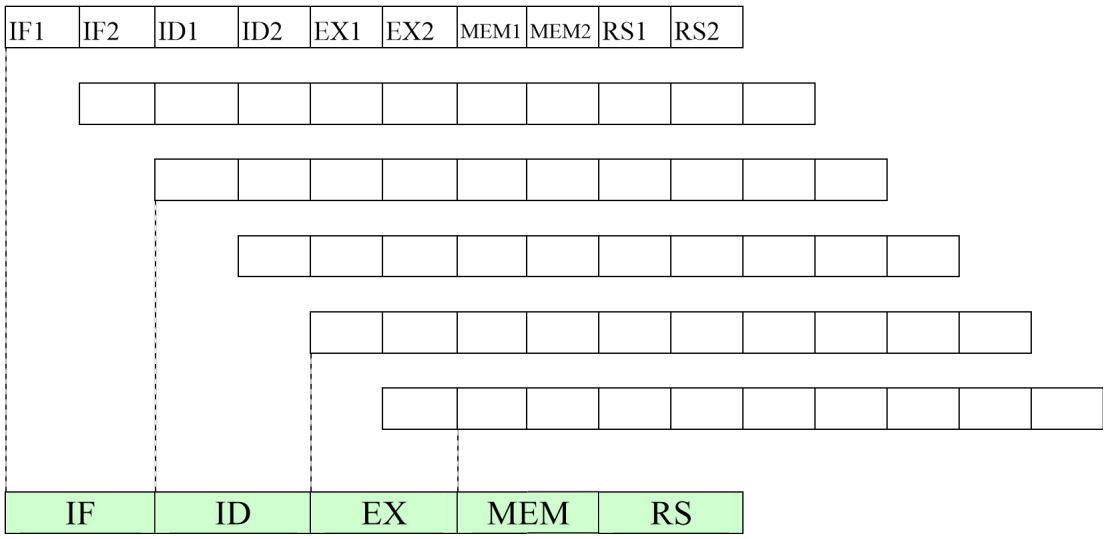
═══Cần phải có nhiều thanh ghi lệnh để lưu giữ lệnh mà chúng ta phải xem xét cho mỗi giai đoạn thi hành lệnh.

═══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

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 III.6 trình bày thí dụ về siêu ống dẫn bậc 2, có so sánh với siêu ống dẫn đơn giản. Ta thấy trong một chu kỳ Tc, máy dùng kỹ thuật siêu ống dẫn

làm 2 lệnh thay vì làm1 lệnh trong máy dùng kỹ thuật ống dẫn bình thường. Trong máy tính siêu ống dẫn, tốc độ thực hiện lệnh tương đương với việc thực hiện một lệnh trong khoảng thời gian Tc/n. Các bất lợi của siêu ống dẫn là thời gian thực hiện một giai đoạn con ngắn Tc/n và việc trì hoãn trong thi hành lệnh nhảy lớn. Trong ví dụ ở hình III.6, nếu lệnh thứ i là một lệnh nhảy tương đối thì lệnh này được giải mã trong giai đoạn ID, địa chỉ nhảy đến được tính vào giai đoạn EX, lệnh phải được nhảy tới là lệnh thứ i+4, vậy có trì trệ 3 lệnh thay vì 1 lệnh trong kỹ thuật ống dẫn bình thường.



i

i+1

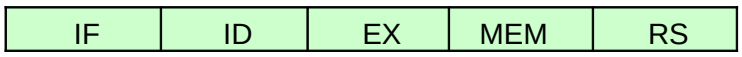
i+2

i+3

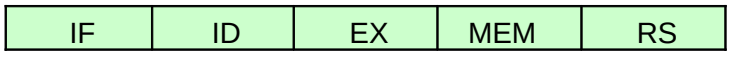
i+4

i+5

i



i+1



i+2

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

7. Các chướng ngại của ống dẫn lệnh

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

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

a. Khó khăn do cấu trúc:

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

b. Khó khăn do số liệu:

Lấy ví dụ trường hợp các lệnh liên tiếp sau:

Lệnh 1: **ADD R1, R2, R3**

Lệnh 2: **SUB R4, R1, R5**

Lệnh 3: **AND R6, R1, R7**

Lệnh 4: **OR R8, R1, R9**

Hình III.5 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.

IF	ID	EX	MEM	RS

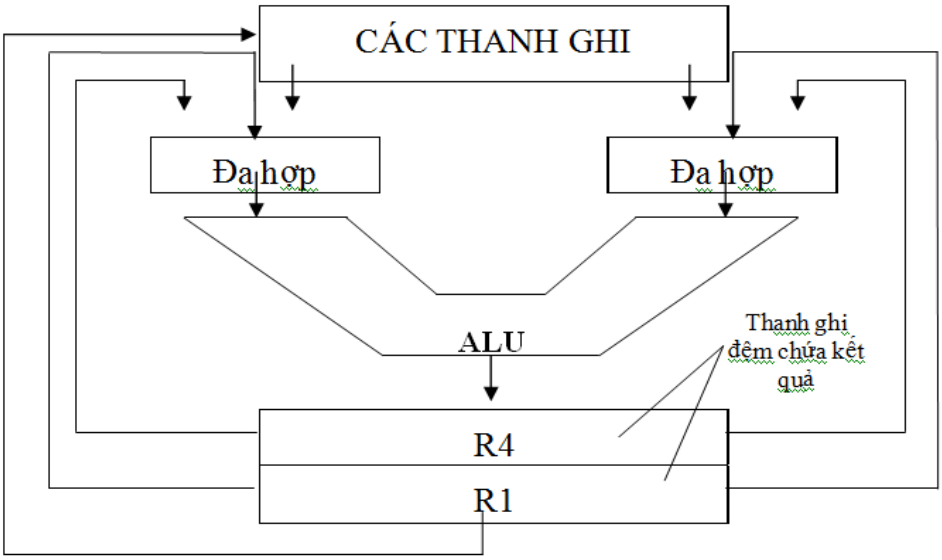
1- ADD R1, R2, R3

2- SUB R4, R1, R5 AND R6, R1, R4

3- OR R8, R1, R9

Hình III.7: 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 III.7.



Hình III.7: ALU với bộ phận phần cứng đưa kết quả tính toán trở lại ngã vào

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

c. Khó khăn do điều khiển:

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

Trong trường hợp đơn giản nhất, tác vụ nhảy không thể biết trước giai đoạn giải mã (xem hình III.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

= 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 liệu

=Các loại ngắt:

+ Ngắt do lỗi khi thực hiện chương, ví dụ : tràn số, chia cho 0

+ ngắt do lỗi phần cứng, ví dụ lỗi bộ nhớ RAM

+ngắt do mo- đun vao/ra phát tín hiệu ngắt đến CPU yêu cầu trao đổi dữ liệu.

Chương IV: Bộ nhớ

1. Phân loại bộ nhớ

Bộ nhớ chứa chương trình, nghĩa là chứa lệnh và số liệu. Người ta phân biệt các loại bộ nhớ: Bộ nhớ trong (RAM-Bộ nhớ vào ra ngẫu nhiên), được chế tạo bằng chất bán dẫn; bộ nhớ chỉ đọc (ROM) cũng là loại bộ

nhớ chỉ đọc và bộ nhớ ngoài bao gồm: đĩa cứng, đĩa mềm, băng từ, trống từ, các loại đĩa quang, các loại thẻ nhớ,...

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

<u>Hình IV.1:</u> Vận hành của bộ nhớ RAM

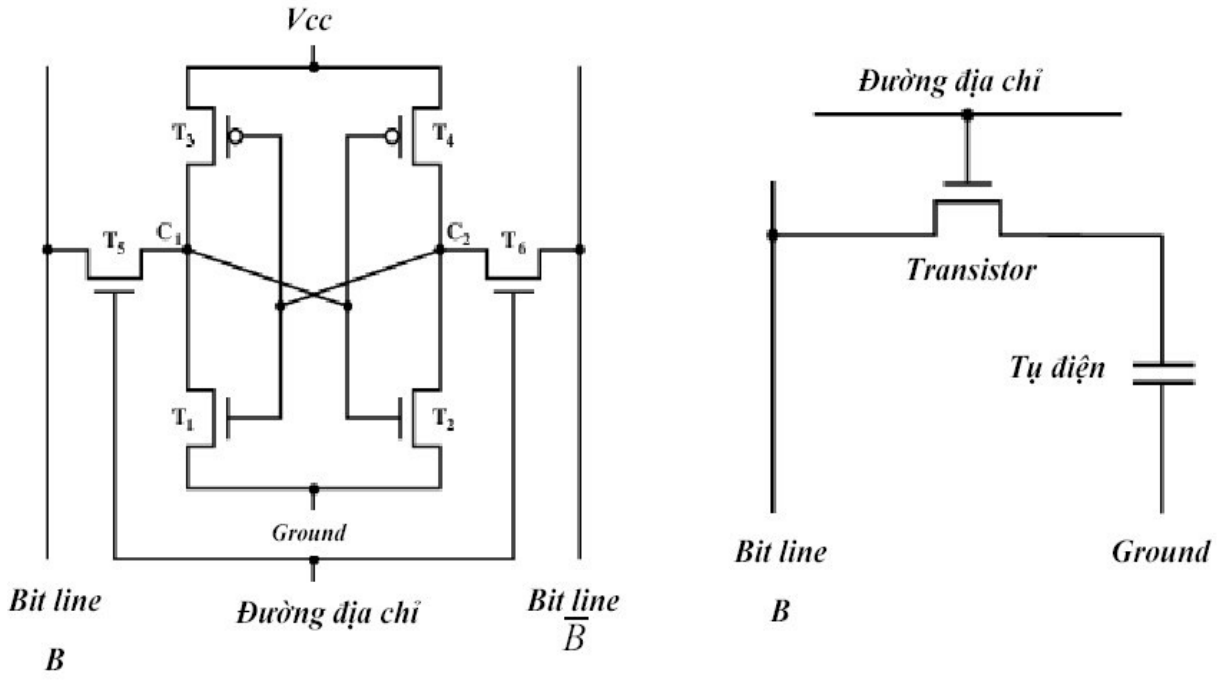
(Wi, Wj, R/W là các tín hiệu điều khiển)

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 IV.2: SRAM và DRAM

SDRAM (Synchronous DRAM – DRAM đồng bộ), một dạng DRAM đồng bộ bus bộ nhớ. Tốc độ SDRAM đạt từ 66-133MHz (thời gian thâm nhập bộ nhớ từ 75ns150ns).

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

RDRAM (Rambus RAM) là một loại DRAM được thiết kế với kỹ thuật hoàn toàn mới so với kỹ thuật SDRAM. RDRAM hoạt động đồng bộ theo một hệ thống lặp và truyền dữ liệu theo một hướng. Một kênh bộ nhớ RDRAM có thể hỗ trợ đến 32 chip DRAM. Mỗi chip được ghép nối tuần tự trên một module gọi là RIMM (Rambus Inline Memory Module) nhưng việc truyền dữ liệu giữa các mạch điều khiển và từng chíp riêng biệt chứ không truyền giữa các chip với nhau. Bus bộ nhớ RDRAM là đường dẫn liên tục đi qua các chip và module trên bus, mỗi module có các chân vào và ra trên các đầu đối diện. Do đó, nếu các khe cắm không chứa RIMM sẽ phải gắn một module liên tục để đảm bảo đường truyền được nối liền.Tốc độ RDRAM đạt từ 400-800MHz

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

PROM (Programable ROM): Chế tạo bằng các mối nối (cầu chì - có thể làm đứt bằng điện). Chương trình nằm trong PROM có thể được viết vào bởi người sử dụng bằng thiết bị đặc biệt và không thể xóa được.

EPROM (Erasable Programable ROM): Chế tạo bằng nguyên tắt phân cực tĩnh điện. Chương trình nằm trong ROM có thể được viết vào (bằng điện) và có thể xóa (bằng tia cực tím - trung hòa tĩnh điện) để viết lại bởi người sử dụng.

EEPROM (Eletrically Erasable Programable ROM): Chế tạo bằng công nghệ bán dẫn. Chương trình nằm trong ROM có thể được viết vào và có thể xóa (bằng điện) để viết lại bởi người sử dụng.

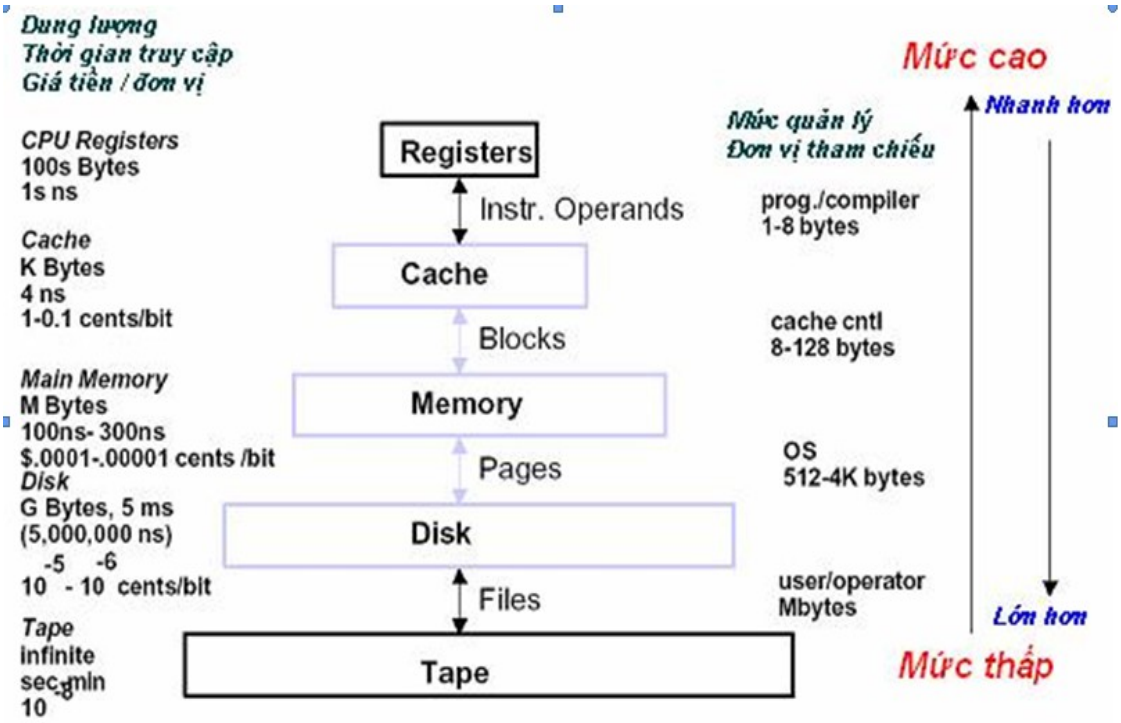
Kiểu bộ nhớ	**Loại**	**Cơ chế xoá**	**Cơ chế ghi**	**Tính bay hơi**
RAM	đọc/ghi	bằng điện, mức byte	bằng điện	Có
ROM	chỉ đọc	Không thể xoá	Mặt nạ	Không
Programmable ROM (PROM)			bằng điện	
Erasable PROM	hầu hết	Tia cực tím, mức		

	chỉ đọc	chip
Electrically Erasable PROM (EEPROM)		bằng điện, mức byte
Flash Memory		bằng điện, mức khối

2. Các loại bộ nhớ bắn dẫn

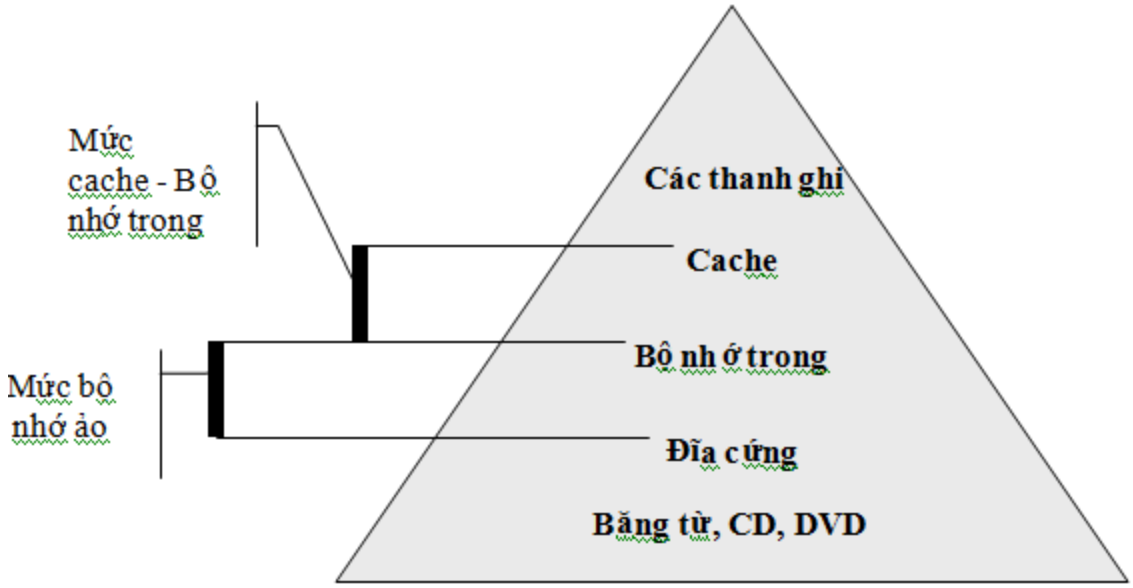
3. Hệ thống nhớ phân cấp

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 IV.3)



Hình IV.3 : Các cấp bộ nhớ

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



Hình IV.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 th´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 th´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 mính, ánh xạ một phần địa chỉ các ô nhớ của mính 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 lư và kiểm tra các địa chỉ ánh xạ.

4. Kết nối bộ nhớ với bộ xử lý

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. 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.

Các bus thường có hai loại: bus hệ thống nối bộ xử lý với bộ nhớ (system bus, Front Side Bus-FSB) và bus nối ngoại vi (bus vào/ra – I/O bus) (hình V.4). 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. 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 .

5. Các tổ chức cache

Như đã biết ,do giá thành rẻ ,DRAM được dùng làm bộ nhớ chính nhưng chúng lại có tốc độ truy cập chậm hơn so với SRAM . Để tăng hệ suất sử dụng DRAM có nhiều phương pháp đã được áp dụng như chế độ trang kể trên .Mục này đề cập đến phương pháp sử dụng bộ nhớ Cache là một cách tăng hiệu suất rất phổ biến hiện nay .

Cache là một lượng SRAM nhỏ (có tốc độ truy cập nhanh) được đưa vào làm việc cùng CPU và bộ nhớ chính là các DRAM có tốc độ truy cập chậm hơn) nhằm làm tăng hiệu suất của hệ thống nhớ .

Cache chứa các từ dữ liệu vừa được CPU truy xuất tại bộ nhớ chính gần đây nhất .Khi CPU truy suất các dữ liệu tiếp theo ,trước tiên nó sẽ đưa địa chỉ của các dữ liệu đó tới bộ điều khiển Cache ..Nếu xác định rằng dữ liệu có địa chỉ đó đã được sao lưu vào Cache ,gọi là trúng Cache (Cache hit) ,thì CPU sẽ truy suất ngay dữ liệu này với tốc độ nhanh của SRAM .Ngược lại khi thấy rằng không có địa chỉ cần truy cập trong Cache ,gọi là trật Cache (Cache miss),CPU sẽ truy cập bộ nhớ chính với tốc độ bình thường của DRAM .

Hiệu suất của Cache phụ thuộc vào tỷ số Cache hit trên Cache miss .Hiệu suất này được quyết định bởi tính cục bộ của vùng quy chiếu bộ nhớ .Tính cục bộ này có được là do các chương trình thường có những vùng lặp tương đối nhỏ nằm trên các địa chỉ nhớ liên tục với nhau .Có 2 loại :Cục bộ thời gian và cục bộ không gian .Tính cục bộ thời gian thể hiện ở chỗ :các chương trình chạy trong các vòng lặp ,các lệnh giống nhau phải được lấy ra từ bộ nhớ trên một cơ sở thường xuyên và liên tục ,nghĩa là các chương trình đó có khuynh hướng sử dụng lại hầu hết thông tin đã sử dụng gần đây .Thông tin càng cũ ,càng ít có khả năng sử dụng lại .Tính cục bộ không gian thể hiện ở chỗ :các chương trình và dữ liệu liên qua có khuynh hướng nằm trong các vùng nhớ liên tục nhau . Điều đó có nghĩa là chúng cần các mã lệnh hay dữ liệu nằm sát hay kề cận với những vị trí đã sử dụng trước đó .Hơn nữa ,các truy xuất bộ nhớ tạo ra trong khoảng thời gian ngắn bất kỳ có khuynh hướng chỉ sử dụng một phần nhỏ của bộ nhớ chính .

Tóm lại ,nếu một từ dữ liệu được truy xuất từ bộ nhớ chính với tốc độ chậm mà được sao chép vào Cache ,thì trong thời gian kế theo ,nếu từ này lại cần sử dụng thì có thể truy suất nó ngay từ Cache với tốc độ nhanh hơn .Hiệu suất mà hệ thống Cache đem lại phụ thuộc vào mức độ thường xuyên cao của các tác vụ truy suất bộ nhớ dựa vào các nguyên lý cục bộ kể trên .

Với các hệ thống Cache hiện nay .Tỷ số Cache hit / Cache miss thường đạt tới 90 % .Cache được tổ chức thành các hàng Cache (Cache line) ,mỗi hàng có thể nhận thông tin từ bộ nhớ chínhđược lưu giữ (đệm) lại trong một hoạt động đọc /viết .Kích thước của một hàng tuỳ thuộc vào dung lượng dữ liểu trong CPU hoặc dung lượng cache cấp 1 là cache được tích hợp ngay trong bộ xử lý .Thí dụ ,trong máy vi tính ,80386 ,mỗi hàng Cache rộng 32 bit (4 byte)

,với 80486 là 128 bit (16 byte) và Pentium là 256bit (32 byte) .Mỗi hàng Cache luôn được nạp đầy dữ liệu trong 1 lần bộ xử lý truy cập bộ nhớ ,không phụ thuộc vào số byte đọc/ viêt của nó trong lần truy cập đó . .Kích thước bộ nhớ cache càng lớn càng làm tăng hiệu suất Cache do lượng thông tin được lưu trữ trong mỗi lần xảy ra cache miss càng lớn hơn .VÌ các lệnh thường gồm các địa chỉ liên tục với nhau nên kích thước hàng càng lớn ,càng có một tỷ lệ cache hit cao .

Có 3 phương pháp viết cache :

-Write –Throgh :Khi xảy ra Cache hit ,bộ điều khiển cache cập nhật nội dung của bộ nhớ cache , đồng thời cũng chuyển hoạt động ghi của CPU tới bộ nhớ chính để đảm bảo tính nhất quán giữa cache và bộ nhớ chính .Thiết kế này đơn giản nhưng hiệu súât thấp vì phải truy cập bộ nhớ chính có tốc độ chậm

- Posted –Write –Throgh :là phương pháp mà hầu hết các hệ thống cache hay sử dụng .Nếu xảy ra cache hit ,bộ điều khiển cache sẽ viết từ mới tới cache với tốc độ nhanh và báo ngay cho bộ xử lý rằng việc viết bộ nhớ đã xong .Nếu xảy ra cache miss nó cũng báo như vậy cho CPU .Toàn bộ tác vụ được lưu trong 1 vùng đêm để rồi sau đó bộ điều khiển cache sẽ viết các từ dữ liệu đó vào bộ nhớ chính .Quá trình viêt này trong suốt (transperent)đối với bộ xử lý trừ khi bộ nhớ chính vẫn còn tham gia vào hoạt động viêt trước đó .Như vậy ,các bộ phận khác sẽ không được sử dụng bus hệ thống cho đến khi quá trình Write –Through được hoàn tất .

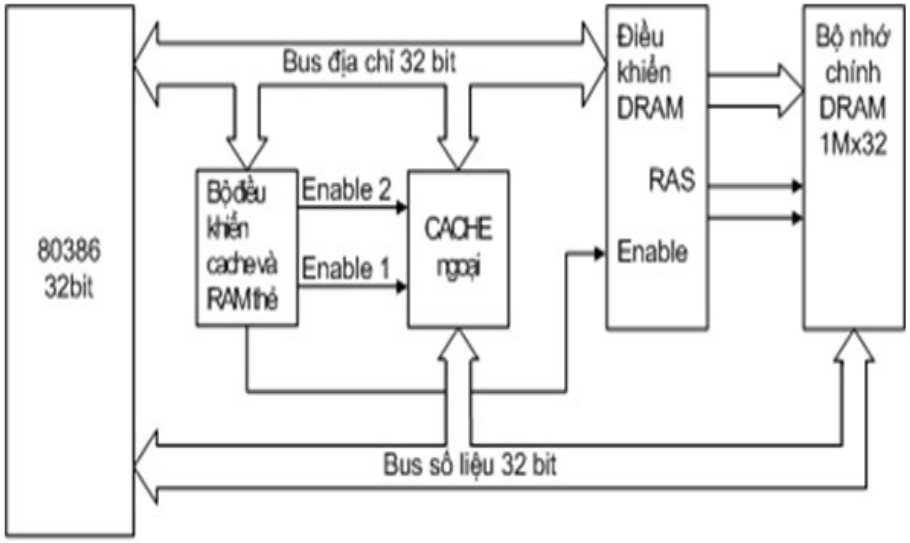
-Write-back :Bộ nhớ sẽ chỉ được cập nhật khi cần thiết .Bộ nhớ chính sẽ được ghi khi một truy suất đọc được thực hiện bởi một đơn vị làm chủ bus khác đối với một địa chỉ bộ nhớ mà hiện chứa dữ liệu là cũ (so với dữ liệu tương ứng trong bộ nhớ cache),hay khi một hàng cache chứa thông tin được cập nhật sắp sửa bị ghi đè lên vì thông tin này đã cũ .

Mỗi hàng cache được nhận diện bằng 1 thẻ bài (tag) , đó là nơi cất giữ địa chỉ này của giữ liệu cũng như thông tin tình trạng bổ sung .Trong cache cấp 2 là cache được ghêp trên bảng mạch chính ,các thể này nằm trong một bộ nhớ riêng biệt tại bộ điều khiển cache .Bộ điều khiển này có tốc độ truy cập còn nhanh hơn bộ nhớ cache .Trong khi truy tìm một địa chỉ trong bộ nhớ cache , địa chỉ này không chỉ được đọc ra từ thẻ mà còn được so sánh với địa chỉ của truy xuất cụ thể bằng cách sử dụng một bộ so sánh .Tất nhiên việc này làm mất thêm một khoảng thời gian nhưng được bù lại bởi tốc độ truy cập nhanh của cache .Bộ điều khiển cache nằm ngay trong CPU (với cache cấp 1) hoặc là một vi điều khiển lắp trên băn mạch chính (với cache cấp 2) .Nó có chức năng điều khiển liên lạc với CPU cũng như các thông tin đến và đi được lưu trữ trong hàng cache .

Hinh.IV.5. là một thí dụ sơ đồ tổ chức bộ nhớ chính trong vi xử lý 80386 khi sử dụng cache .

Để theo dõi vùng nhớ chính nào đang hiện diện trong RAM cache ,,bộ điều khiển cache sử dụng thư mục cache là một bộ nhớ lắp ráp trong vi mạch điều khiển cache .Mỗi vùng trong cache được đại diện bởi một điểm vào trong

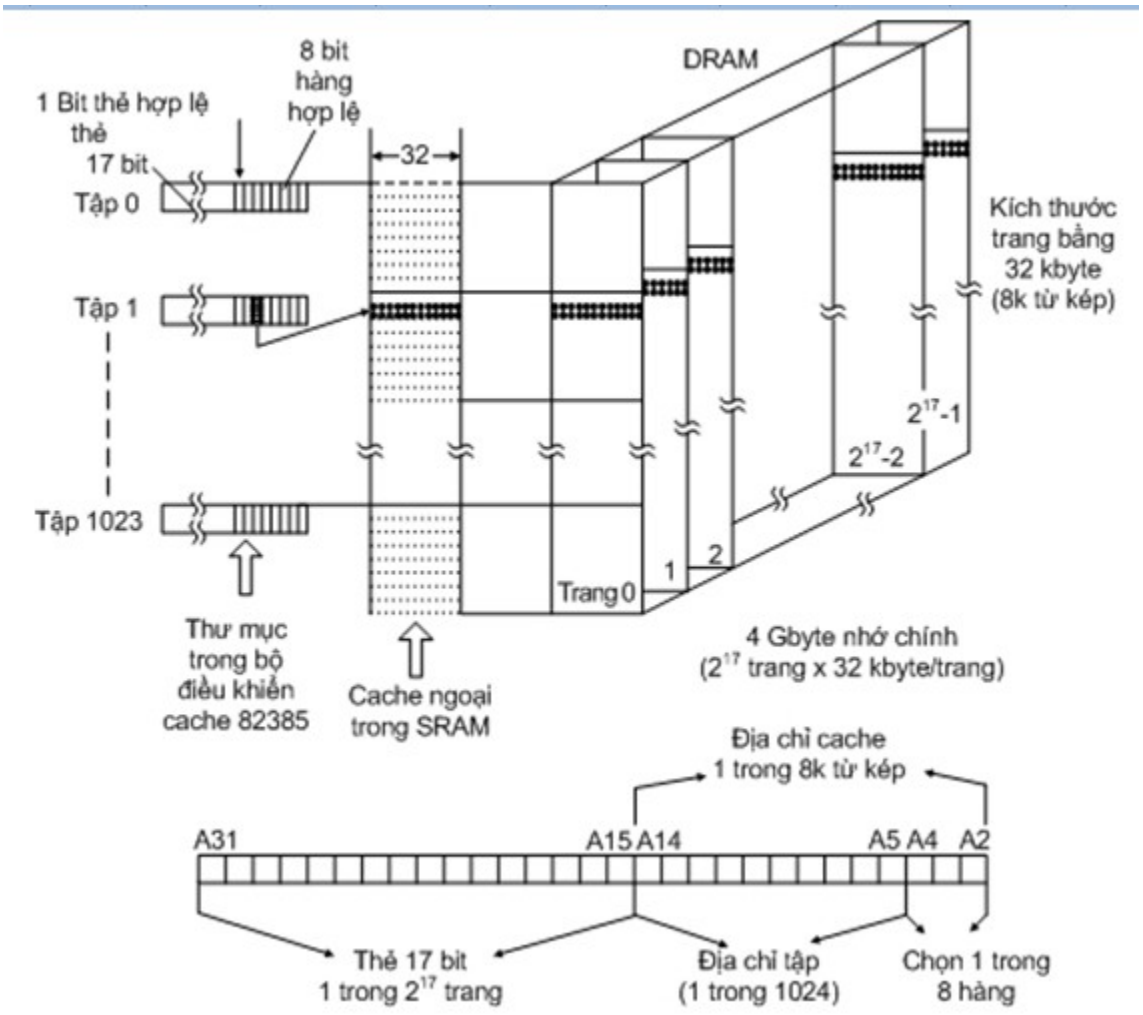
thư mục . Định dạng chính xác cho điểm vào thư mục phụ thuộc vào các sơ đồ cụ thể .Có 3 sơ đồ cơ bản cho cách tổ chức cache như sau .



Hình IV.5: Sơ đồ tổ chức bộ nhớ chính

a. **Cache ánh xạ trực tiếp (dirrect mapped cache)**

Có thể dùng sơ đồ Hinh IV.6 làm một thí dụ cho bộ nhớ cache 32kB nhớ ($=2^{15}$ byte) lắp cho vi xử lý 80386 . Đây là vi xử lý có bus địa chỉ rộng 32 bit nên có thể quản lý được 2^{32} =4GB nhớ .Bộ điều khiển cache coi bộ nhớ chính này được chia thành các trang nhớ lôgic ,mỗi trang có kích thước đúng bằng bộ nhớ cache là 32kB.vậy sẽ có $2^{32} : 2^{15} = 2^{17}$ trang nhớ .Vì bus dữ liệu của vi xử lý rộng 32 bit nên có thể cho phép truy xuất 4 byte một lần từ bộ nhớ .Do vậy một nhóm 4 byte sẽ được gán cho một hàng cache .Cứ 8 hàng cache này lại được gán thành một tập và toàn bộ nhớ cache sẽ có 1024 tập .

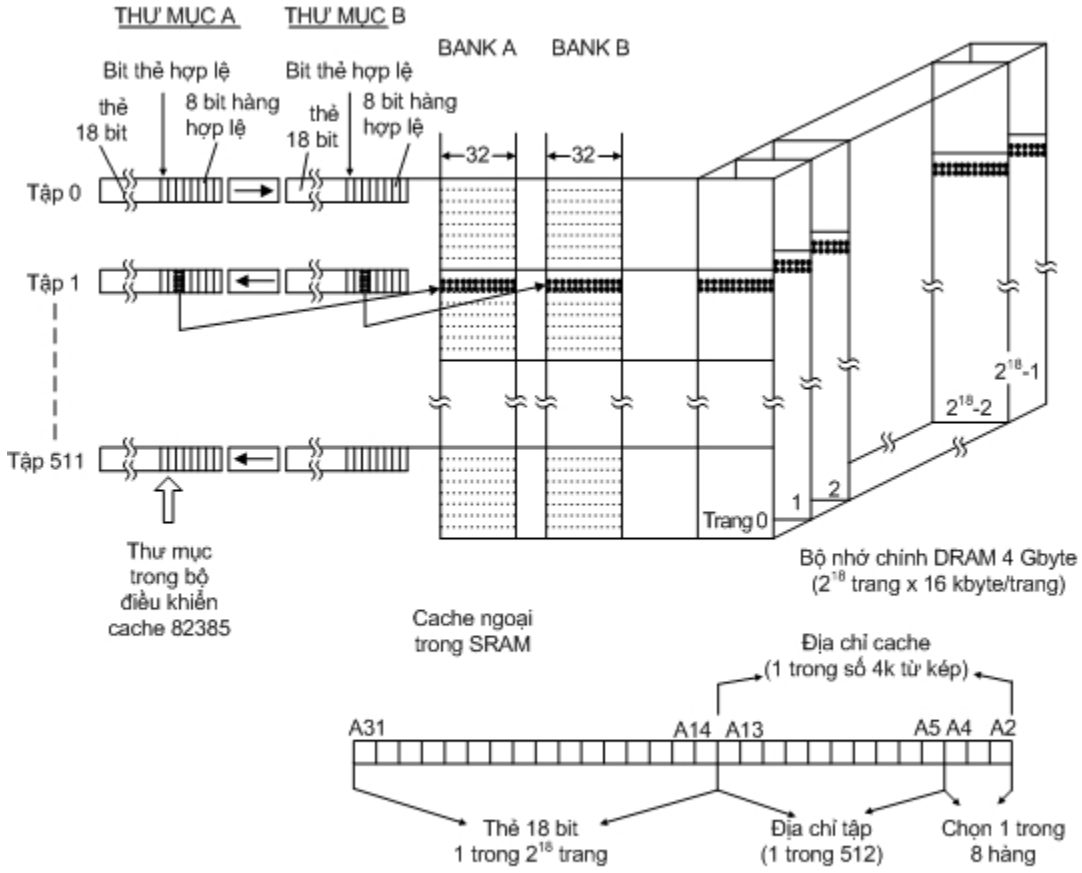


Việc ánh xạ trực tiếp ở đây có nghĩa là một hàng dọc đánh số cho một trang nhớ ở bộ nhớ chính sẽ luôn được copy tới một hàng tương ứng trong cache .Thư mục bên trái hình được dùng để theo dõi hàng nào trong bộ nhớ chính hiện đã được lưu vào bộ nhớ cache .Thư mục chứa các điểm vào 26 bit gồm 17 bit thẻ ,1 bit thẻ hợp lệ và một bit hàng hợp lệ .17 bit cao là thẻ bài (tag) .Thẻ dùng để nhận dạng trang nhớ trong bộ nhớ chính mà một hàng hoặc một tập các hàng trong cache được copy vào .Mỗi điểm vào thư mục cũng có 1 bit

thẻ hợp lệ (tag valid bit) và 8 bit hàng hợp lệ (line valid bit) ,mỗi bit cho một hàng trong tập .Khi bộ xử lý đưa ra một địa chỉ 32 bit để truy xuất 1 từ từ bộ nhớ như h.3.11,các bit địa chỉ từ A15 đến A31 dùng để tìm một trong số 2^{17} trang trong bộ nhớ chính ,các bit từ A5 đến A14 đại diện cho tập chứa một hàng mong muốn và các bit từ A2 –A4 dùng để nhận dạng số hàng trong tập chứa từ dữ liệu cần truy nhập .Bộ điều khiển cache trước tiên dùng các bit địa chỉ từ A5 –A14 để chọn điểm vào thư mục cho tập chứa hàng định vị .Sau đó nó so sánh nó so sánh với 17 bit địa chỉ từ bộ xử lý đưa ra với thẻ 17 bit được lưu trữ trong điểm vào thư mục .Nếu tương đương nó tiếp tục kiêmtra bit thẻ hợp lệ .Nếu bit hàng được đặt nó kiểm tra bit hàng hợp lệ cho hàng được định vị bởi các bit từ A2 đến A4 .Nếu tất cả thẻ và hàng hợp lệ thì hàng dữ liệu đó đã ở trong cache và có nghĩa là xảy ra cache hit .Trong trường hợp này ,bộ điều khiển sẽ đưa các bit địachỉ từ A2-A14 đến bộ nhớ cache và cho phép và bộ nhớ cache xuất từ cần truy nhập lên bus số liệu .Nếu 17 bit cao của địa chỉ 80386 không giống như thẻ trong thư mục (bit thẻ không hợp lệ) hoặc bit hàng cho hàng được đụnh vị không hợp lệ thì hoạt động đọc là cache miss .Trong trường hợp này ,bộ điều khiên cache sẽ gửi một địa chỉ toàn bộ từ vi xử lý tới bộ điều khiển DRAM của bộ nhớ chính .Bộ điều khiển DRAM sẽ điều khiển bộ nhớ chính xuất radữ liệu trong hàng được định vị lên bus dữ liệu .Khi hàng dữ liệu này xuất hiện ,bộ điều khiển cache sẽ kích hoạt (enable) bộ nhớ cache ,nhằm làm cho hàng này cũng có thể được viết vào cache cũng như toàn bộ xử lý .Nó cũng đồng thời cập nhật để chỉ thị rằng hàng này bây giờ đã ở trong cache .Nếu hàng này hoặc một phần bất kỳ của nó được cần đến lần nữa ,nó có thể được đọc trực tiếp từ cache .

b. **Cache liên hợp hai hàng**

Một khó khăn với tổ chức kể trên là khi chương trình sử dụng cùng hàng được đánh số như nhau từ 2 trang nhớ trong cùng một lúc ,nó sẽ phải tráo đổi 2 hàng giữa bộ nhớ chính và cache . Để tránh điều này phải dùng sơ đồ tổ chức cache liên hợp 2 hàng (2-way set associative cache system)như hình 3.12. Ở đây dùng 2 bộ nhớ cache và hai thư mục cache riêng biệt nhằm cho cùng các hàng từ các trang khác nhau có thể được ghi vào cùng một lúc .



Hình 3.12. Tổ chức cache liên hợp tập 2 hàng.

Mỗi bộ nhớ cache có kích thước bằng một nửa so với cache ánh xạ trực tiếp .Như vậy bộ điều khiển cache coi bộ nhớ chính gồm 262 144 = 2^{34}

trang ,mỗi trang gồm 4096 =2^{12} hàng . Để nhận diện một trong số các trang này ,thẻ trogn mỗi điểm vào của thư mục cache chứa 18 bit .Mỗi điểm vào thư mục trong hệ thống này cũng có 1 bit thẻ hợp lệ ,8 bit hàng hợp lệ và 1 bit LRU (Least Recently Used bit).Xét quá trình đọc dữ liệu .Khi vi xử lý xuất ra một địa chỉ ,bộ điều khiển cache dùng các bit địa chỉ từ A13-A15 để chọn điểm vào thích hợp trong mỗi thư mục cache .Sau đó nó sẽ so sánh 18 bit cao của địa chỉ từ bộ xử lý với thẻ trong mỗi điểm vào thư mục được chọn .Nếu một trong những thẻ này phù hợp ,bộ điều khiển sẽ kiểm tra bit thẻ hợp lệ trong điểm vào thư mục đó .Nó cũng kiểm tra bit hàng hợp lệ cho hàng được hiện diện bởi các bit địa chỉ từ A2 –A4 .Néu các bit này được đặt ,bộ điều khiển cache sẽ xuất ra các bit địa chỉ từ A2-A13 tới cache liên quan tới thư mục đó và cho phép cache xuất ra từ dữ liệu mong muốn trên bus dữ liệu .Nếu từ dữ liệu được tìm thấy trong cache A thì bit LRU trong điểm vào thư mục được đặt để chỉ thị rằng cache A vừa mới được sử dụng .Nếu từ dữ liệu được tìm được trong cache B thì bit LRU được đặt để chỉ thị rằng cache B vừa mới được sử dụng .Cơ chế này được dùng để xác định cache nào sẽ được sử dụng để giữ một hàng mới được đọc vào từ bộ nhớ chính .Khi một hoạt động xảy ra cache miss ,bộ điều khiển cache sẽ gửi địa chỉ và các tín hiệu điều khiển mong muốn tới bộ nhớ chính để đọc một hàng đang chứa từ dữ liệu mong muốn .Khi hàng này xuất hiện trên bus dữ liệu ,,bộ điều khiển cache sẽ viết nó tới cache được sử dụng lâu nhất và cập nhật điểm vào thư mục tương ứng .Nếu bộ điều khiển cache phát hiện rằng thẻ cho quá trình đọc chính xác nhưng một bit hàng hợp lệ không hợp lệ ,nó sẽ đọc hàng từ bộ nhớ chính và viết nó vào bộ nhớ cache mà thư mục của nó chứa thẻ . ĐIều này đảm bảo

rằng các hàng liền kề từ một trang trong bộ nhớ chính được kết thúc trong cùng một cache

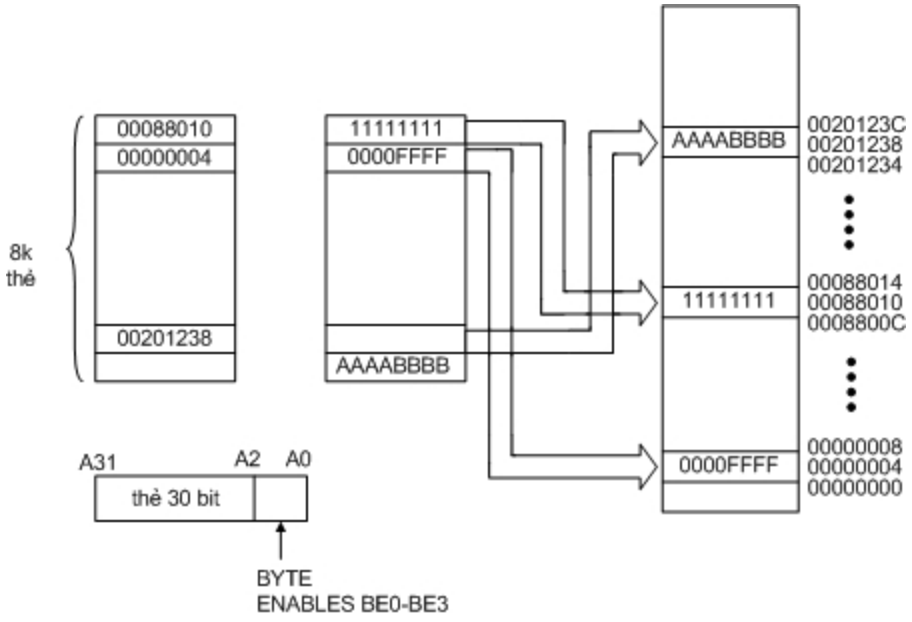
Đối với hoạt động viết ,tổ chức cache liên hợp tập 2 đường này sử dụng thiết kế kiểu posted –write –through nói trên .Bộ điều khiển cache luôn viết từ dữ liệu ra tới bộ nhớ chính và nếu từ này hiện diện trong cache thì nó cũng sẽ cập nhật từ trong cache .

Tổ chức cache liên hợp 4 đường chỉ đơn giản bằng sự mở rộng của tổ chức liên hợp tập 2 đường .Bộ nhớ cache được chia thành 4 khối có kích thước bằng nhau và bộ nhớ chính được xem như được chia thành các trang có kích thước bằng với kích thước của mỗi khối .Tổ chức này cho phép một khă năng linh động hơn loại cache liên hợp tập 2 đường .

c. **Cache liên hợp hoàn toàn (Fully associative cache sytem)**

H.3.13. chỉ ra sơ đồ khối của hệ thống này . Ở đây ,một khối 4 byte hoặc 1 hàng từ bộ nhớ có thể được viêt vào bất kỳ vùng nào trong cache .Hệ thống có một bus địa chỉ 32 bit như vậy có thể định vị được 4Gbyte nhớ hay một Gyga các hàng 4 byte .Vì 1Gbyte bằng 2^{30} byte nên cần một thẻ bài dài 30 bit để nhận diện mỗi khối hoặc hàng được lưu trữ trong cache .Mỗi điểm vào trong thư muc như vậy phải có 30 bit cho thẻ và các bit dùng để theo dõi hàng được dùng gần đây như thế nào .Cache liên hợp hoàn toàn có ưu điểm là nó có thể giữ các hàng được đánh số như nhau trong 1 vào trang khác nhau cùng một thời gian .Nhược điểm của nó là 30 bit cao của mỗi địa chỉ nhớ được gửi ra bởi CPU phải được so sánh với tất cả các thẻ trong thư mục để xem liệu hàng

dữ liệu đó có hiện diện trong cache hay không . Điều này làm mất thêm nhiều thời gian .Cũng vậy khi cache loại này chứa đầy dữ liệu thì một vài thuật giải phải được sử dụng để xác định xem hàng nào sẽ bị viết đè lên khi một hàng mới phải cần được viết vào cache từ bộ nhớ chính .Thuật giải thông dụng nhất là thay thế hàng già nhất bằng hàng mới .



Hình 3.13. Tổ chức cache liên hợp hoàn toàn.

Chương V: Thiết bị nhớ ngoài

1. Các thiết bị nhớ trên vật liệu từ

1.1. Đĩa 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.

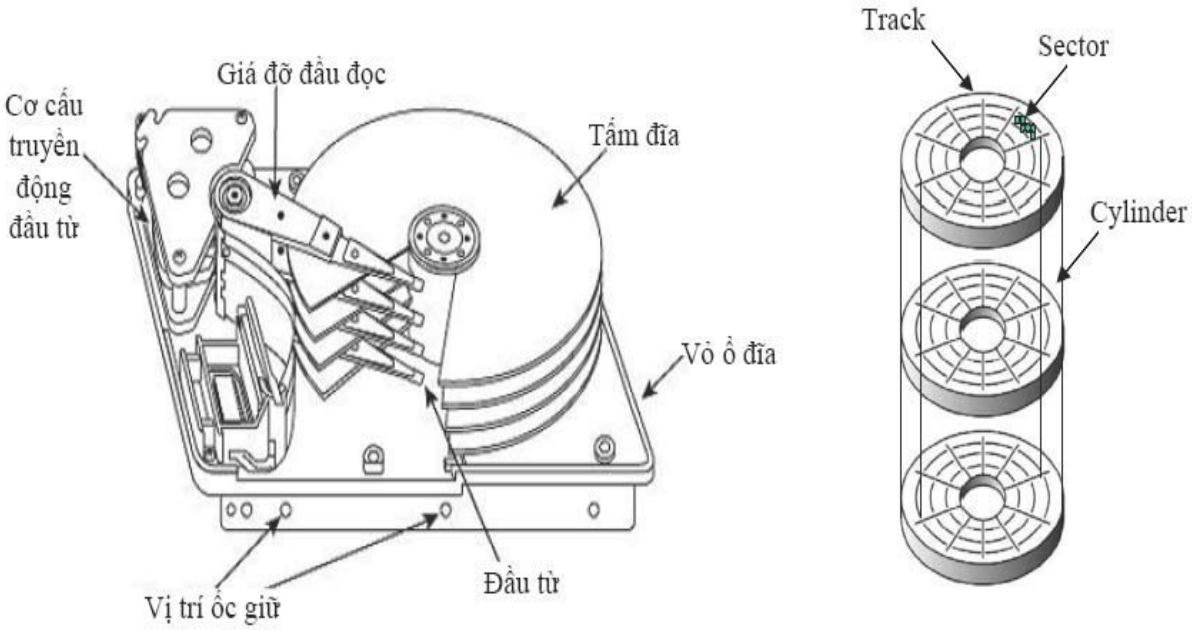
═══Lưu trữ dài hạn các tập tin.

═══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.

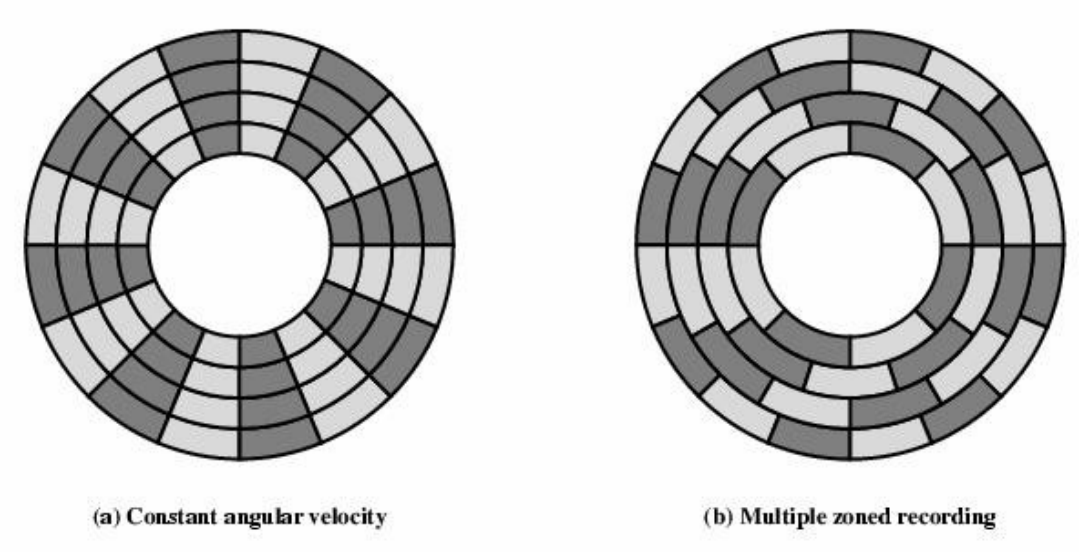
Một đĩa cứng chứa nhiều lớp đĩa (từ 1 đến 4) quay quanh một trục khoảng 3.60015.000 vòng mỗi phút. Các lớp đĩa này được làm bằng kim loại với hai mặt được phủ một chất từ tính (hình V.1). Đường kính của đĩa thay đổi từ 1,3 inch đến 8 inch. Mỗi mặt của một lớp đĩa được chia thành nhiều đường tròn đồng trục gọi là rãnh. Thông thường mỗi mặt của một lớp đĩa có từ 10.000 đến gần 30.000 rãnh. Mỗi rãnh được chia thành nhiều cung (sector) dùng chứa thông tin. Một rãnh có thể chứa từ 64 đến 800 cung. Cung là đơn vị nhỏ nhất mà máy tính có thể đọc hoặc viết (thông thường khoảng 512 bytes). Chuỗi thông tin ghi trên mỗi cung gồm có: số thứ tự của cung, một khoảng trống, số liệu của cung đó bao gồm cả các mã sửa lỗi, một khoảng trống, số thứ tự của cung tiếp theo.

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



Hình V.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.



Hình V.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 V.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

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-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,

đĩa CD-ROM được dùng để chứa các phần mềm và các chương trình điều khiển thiết bị.

=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**

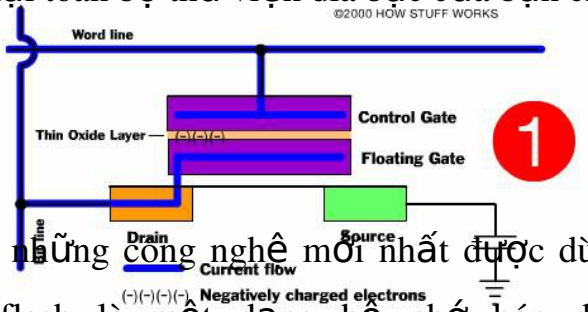
= DVD-R (DVD-Recordable): Giống như đĩa DVD-ROM, người dùng có thể ghi dữ liệu lên đĩa một lần và đọc được nhiều lần. Đĩa này chỉ có thể ghi được trên một mặt đĩa, dung lượng ghi trên mỗi mặt tối đa là 4.7 GB.

= 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.

2.3. **Blu-ray**

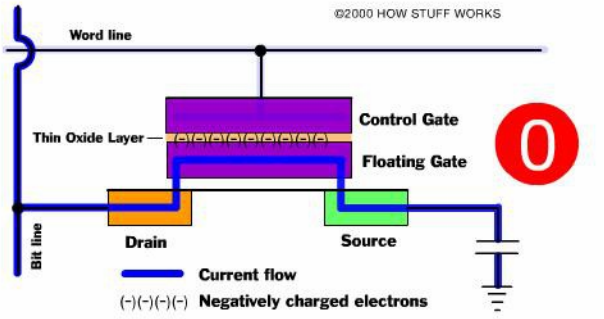
Blu-ray là một định dạng đĩa quang được sử dụng để lưu video độ sắc nét cao cũng như game. Những đĩa này là DVD thế hệ mới cung cấp chất lượng hình ảnh Full HD và vì chúng có hình thức giống DVD chuẩn, các đĩa này rất dễ sử dụng. Có lẽ quan trọng nhất là, các trình đơn giống nhau ở cả hai định

dạng, và đầu đọc Blu-ray có thể phát lại toàn bộ thư viện đĩa bạc của bạn cho dù chúng là CD hay DVD.



3. Các loại thẻ nhớ

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



4. An toàn dữ liệu trong lưu trữ

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).

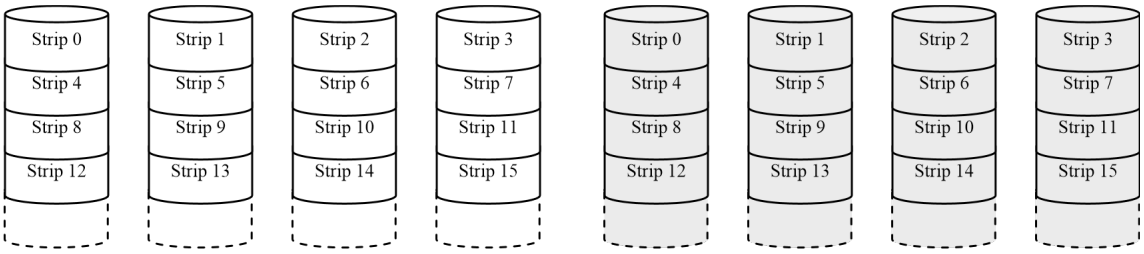
═══Trong mảng đĩa có lưu các thông tin kiểm tra lỗi dữ liệu; do đó, dữ liệu có thể được phục hồi nếu có một đĩa trong mảng đĩa bị hư hỏng . Tuỳ theo kỹ thuật thiết lập, RAID có thể có các mức sau:

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 V.8: 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 V.9: 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 V.10: 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).

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 V.12: 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 V.13: 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

Chương VI : Các loại bus

1. **Định nghĩa bus, bus hệ thống**

Trong kiến trúc máy tính, bus là một hệ thống phụ chuyển dữ liệu giữa các thành phần bên trong máy tính, hoặc giữa các máy tính với nhau.

Các bus máy tính đầu tiên theo nghĩa đen là các dây điện song song với đa kết nối, nhưng thuật ngữ này bây giờ được sử dụng cho bất cứ sắp xếp vật lý cung cấp cùng một chức năng như các bus điện tử song song.

Các bus máy tính hiện đại có thể dùng cả thông tin liên lạc song song và các kết nối chuỗi bit, và có thể được đi dây trong một multidrop (dòng điện song song) hoặc chuỗi Daisy (kỹ thuật điện tử) có cấu trúc liên kết, hoặc kết nối với các hub chuyển mạch, như USB.

Bus là các đường truyền. Thông tin sẽ được chuyển qua lại giữa các thành phần linh kiện thông qua mạng lưới gọi là các Bus.

Bus hệ thống (Bus system) sẽ kết nối tất cả các thành phần lại với nhau

2. **Bus đồng bộ và không đồng bộ**

a. **Bus đồng bộ**

Bus đồng bộ có một đường dây điều khiển bởi một bô dao động thạch anh, tín hiệu trên đường dây này có dạng sóng vuông, với tần số thường nằm trong khoảng 5MHz - 50 MHz. Mọi hoạt động bus xảy ra trong một số nguyên lần chu kỳ này và được gọi là chu kỳ bus.

Giản đồ thời gian của một bus đồng bộ với tần số đồng hồ là 4MHz, như vậy chu kỳ bus là 250nS.

T1 bắt đầu bằng sườn lên của tin hiệu đồng bộ, trong một phần thời gian của T1, MPU đặt địa chỉ của byte cần đọc lên bus địa chỉ. Sau khi tín hiệu địa chỉ được thiết lập giá trị mới, MPU đặt các tín hiệu và tích cực. Tín hiệu (memory request, truy cập bộ nhớ) chứ không phải thiết bị I/O; còn tín hiệu (Read) chọn Read.

- T2 là thời gian cần thiết để bộ nhớ giải mã địa chỉ và đưa dữ liệu lên bus dữ liệu.

- T3 tại sườn xung xuống của T3, MPU nhận dữ liệu trên bus dữ liệu, chứa vào thanh ghi bên trong MPU và chốt dữ liệu. Sau đó MPU đảo các tín hiệu và .
Như vậy đã kết thúc một thao tác đọc, tại chu kỳ máy tiếp theo MPU có thể thực hiện một thao tác khác.

- TAD : theo giản đồ thời gian, TAD 110ns, đây là thông số do nhà sản xuất đảm bảo, MPU sẽ đưa ra tín hiệu địa chỉ không chậm hươn 110ns tính từ thời điểm giữa sườn lên của T1.

- TDS : Giá trị nhỏ nhất là 50ns, thông số này cho phép dữ liệu được đưa ra ổn định trên bus dữ liệu ít nhất là 50ns trước thời điểm giữa sườn xuống

của T3. Yêu cầu về thời gian này đảm bảo cho MPU đọc dữ liệu liệu tin cậy. Khoảng thời gian bắt buộc đối với TAD và TDS cũng nói lên rằng, trong trường hợp xấu nhất, bộ nhớ chỉ có 250 + 250+ 125 - 110 - 50 = 465ns tính từ thời điểm có tín hiệu địa chỉ cho tới khi nó đưa dữ liệu ra bus địa chỉ. Nếu bộ nhớ không đáp ứng đủ nhanh, nó cần phải phát tín hiệu xin chờ trước sườn xuống của T2. Thao tác này đưa thêm vào một trạng thái chờ (wait state), khi bộ nhớ đã đưa ra dữ liệu ổn định, nó sẽ đảo tín hiệu thành WAIT.

- TML: Đảm bảo rằng tín hiệu địa chỉ sẽ được thiết lập trước tín hiệu ít nhất là 60ns. Khoảng thời gian này là quan trọng nếu tín hiệu điều khiển sự tạo ra tín hiệu chọn chip CS, bởi vì một số chip nhớ đòi hỏi phải nhận được tín hiệu địa chỉ trước tín hiệu chọn chip. Như vậy không thể chọn chip nhớ với thời gian thiết lập là 75ns.

- TM, TRL: Các giá trị bắt buộc đối với 2 đại lượng này có ý nghĩa là cả hai tín hiệu và sẽ là tích cực trong khoảng thời gian 85ns tính từ thừi điểm xuống của xung đồng hồ T1. Trong trường hợp xấu nhất, chip nhớ chỉ có 250 + 250 - 85- 50 = 365ns sau khi hai tín hiệu trên là tích cực để đưa dữ liệu ra bus. Sự bắt buộc về thời gian này bổ sung thêm sự bắt buộc thời gian với tín hiệu đồng hồ.

- TMH, TRH: Hai đại lượng này cho biết cần có bao nhiêu thời gian để các tín hiệu và sẽ được đảo sau khi dữ liệu đã được MPU đọc vào.
- TDH: Cho biết bộ nhớ cần phải lưu dữ liệu bao lâu trên bus sau khi tín hiệu đã đảo.

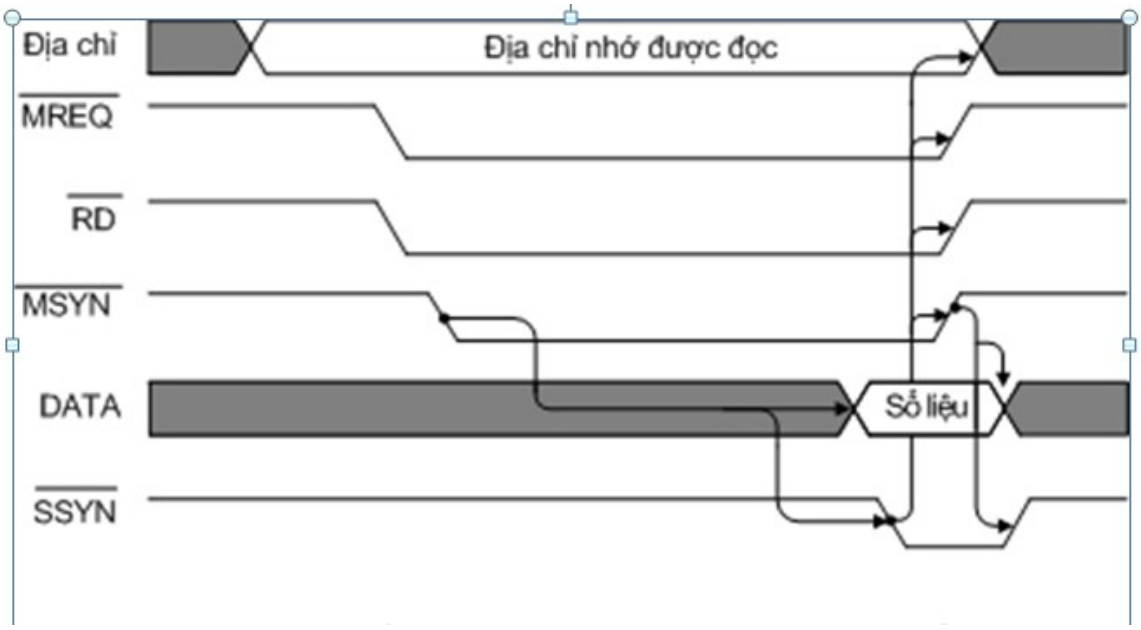
Block Transfer, truyền tải khối dữ liệu. Ngoài các chu kỳ đọc/ ghi, một số bus đồng bộ còn hỗ trợ truyền dữ liệu theo khối. Khi một thao tác đọc/ ghi

bắt đầu, bus master báo cho slave biết có bao nhiêu byte cần truyền đi, sau đó slave sẽ liên tục đưa ra mỗi chu kỳ một byte, cho đến khi đủ số byte được thông báo. Như vậy, khi đọc dữ liệu theo khối, n byte dữ liệu cần n+2 chu kỳ, thay cho 3n chu kỳ như trước.Cách khác làm cho bus truyền dữ liệu nhanh hơn là làm cho các chu kỳ ngắn lại. Trong ví dụ trên, mỗi byte được truyền đi trong 750ns, vậy bus có dải thông là 1.33MBs. Nếu xung đồng hồ là 8MHz, thời gian một chu kỳ chỉ còn một nửa, giải thông sẽ là 2.67MBs.Tuy vậy việc giảm chu kỳ bus dẫn đến các khó khăn về mặt kỹ thuật, các bit tín hiệu truyền trên các đường dây khác nhau trong bus không phải luôn có cùng vận tốc, dẫn đến một hiệu ứng, gọi là bus skew.

Khi nghiên cứu về bus cần phải quan tâm đến vấn đề tín hiệu tích cực nên là mức thấp hay mức cao. Điều này tuỳ thuộc vào người thiết kế bus xác định mức nào là thuận lợi hơn.

Ký hiệu	Tham số	Min	Max
T_{AD}	Thời gian trễ của tín hiệu địa chỉ		110
T_{ML}	Thời gian địa chỉ ổn định trước tín hiệu $\overline{MREQ}$	60	
T_{M}	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{RL}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{DS}	Thời gian thiết lập dữ liệu trước sườn xuống của tín hiệu đồng hồ T3	50	
T_{MH}	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{RH}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{DH}	Thời gian lưu trữ dữ liệu từ lúc đảo tín hiệu $\overline{RD}$	0	

Bảng 5.1. Giá trị của một số thông số thời gian



Hinh 5.2: Giãn đồ thời gian đọc dữ liệu trên bus đồng bộ

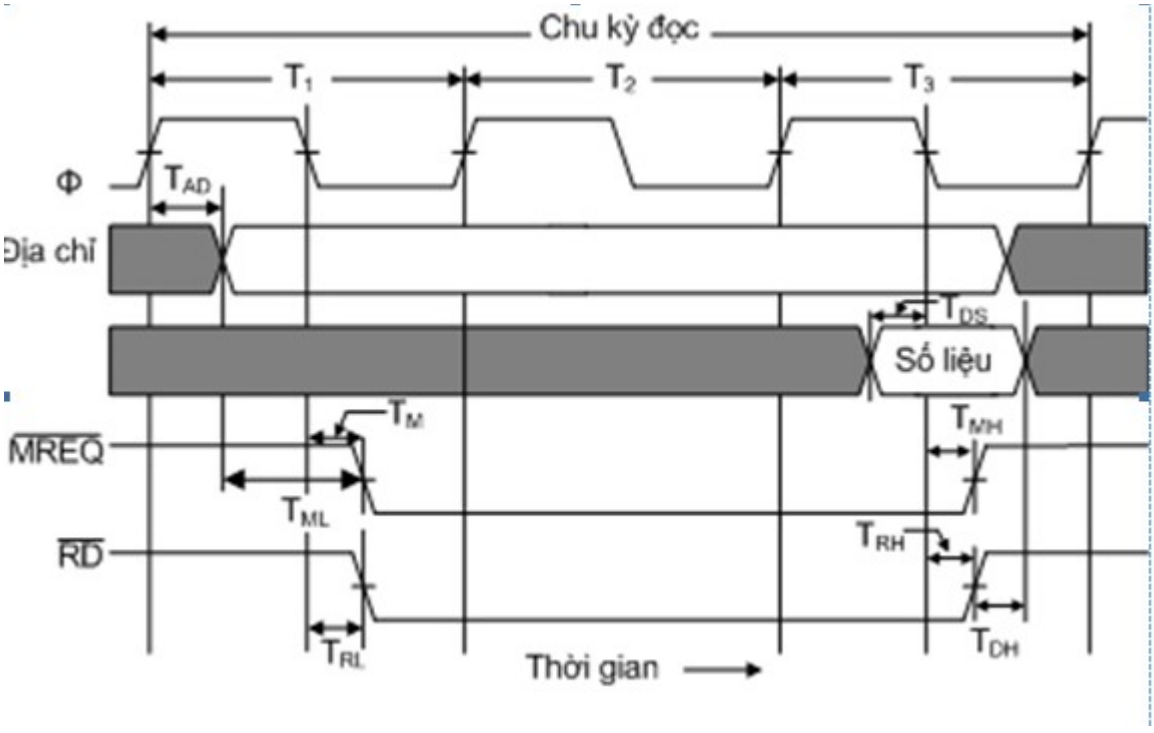
b. **Bus không đồng bộ**

Bus không đồng bộ không sử dụng một xung đồng hồ định nhịp. Chu kỳ của nó có thể kéo dài tuỳ ý và có thể khác nhau đối với các cặp thiết bị trao đổi tin khác nhau.

Làm việc với bus đồng bộ dễ dàng hơn do nó được định thời một cách gián đoạn, tuy vậy chính đặc điểm này cũng dẫn đến nhược điểm. Thứ nhất là: mọi công việc được tiến hành trong những khoảng thời gian là bội số nhịp đồng hồ bus, nếu một thao tác nào đó của CPU hay bộ nhớ có thể hoàn thành trong 3,2 chu kỳ thì nó sẽ phải kéo dài thành 4 chu kỳ. Điều hạn chế lớn nữa là đã chọn chu kỳ bus và đã xây dựng bộ nhớ, I/O Card cho bus này thì khó có thể tận dụng được được những tiến bộ của công nghệ. Chẳng hạn sau khi đã xây dựng bus với sự định thời như trên, công nghệ mới đưa ra các chip CPU và chip nhớ có thời gian chu kỳ là 100ns (thay cho 250ns như cũ), chúng vẫn cứ phải chạy với tốc độ thấp như các CPU và chip nhớ loại cũ, bởi vì nghi thức bus đòi hỏi chip nhớ phải đưa ra dữ liệu và ổn định dữ liệu ngay trước thời điểm ứng với sườn xuống của T3. Nếu có nhiều thiết bị khác nhau nối với một bus, trong đó có một số thiết bị có thể hoạt động nhanh hơn các thiết bị khác thì cần phải đặt bus hoạt động phù hợp với thiết bị chậm nhất.

Bus không đồng bộ ra đời nhằm khắc phục các nhược điểm của bus đồng bộ. Hình 5.3 minh hoạ sự hoạt động của bus không đồng bộ, trong đó master yêu cầu đọc bộ nhớ.

Trước hết master cần phát ra địa chỉ nhớ mà nó muốn truy cập, sau đó phát tín hiệu tích cực để báo rằng nó muốn truy cập bộ nhớ chứ không phải cổng I/O. Tín hiệu này là cần thiết vì bộ nhớ và các cổng I/O đều có thể dùng chung một miền địa chỉ. Tiếp theo master phải phát tín hiệu tích cực để bên slave biết rằng master sẽ thực hiện thao tác đọc chứ không phải là thao tác ghi. Các tín hiệu và được đưa ra sau tín hiệu định địa chỉ bao lâu tuỳ thuộc vào tốc độ của master. Sau khi hai tín hiệu này đã ổn định, master sẽ phát tín hiệu đặc biệt, là (Master SYNchronization) ở mức tích cực để báo cho slave biết rằng các tín hiệu cần thiết đã sẵn sàng trên bus, slave có thể nhận lấy. Khi slave nhận các tín hiệu này, nó sẽ thực hiện công việc với tốc độ nhanh nhất có thể được (nhanh chóng đưa dữ liệu của ô nhớ yêu cầu lên bus dữ liệu). Khi hoàn thành, slave sẽ phát tín hiệu (Slave SYNchronization) tích cực.



Hinh: Hoạt động của bus không đồng bộ

3. **Hệ thống bus phân cấp**

Phân cấp bus cho các thành phần:

a. Bus của bộ xử lý

b. Bus của bộ nhớ chính

c. Các bus vào-ra

Phân cấp bus khác nhau về tốc độ

Bus bộ nhớ chính và các bus vào-ra không phụ thuộc vào bộ xử lý cụ thể.

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

Bus IBM PC

Đây là ví dụ điển hình về một loại bus được sử dụng trong các hệ thống thương mại, nó được sử dụng rộng rãi trong các hệ thống vi xử lý dựa trên chip 8088. Hầu hết họ PC, kể cả các máy tương thích đều sử dụng bus này. Chính bus IBM PC tạo nên cơ sở cho bus IBM PC/AT và nhiều loại bus khác. Bus này có 62 đường dây, trong đó có 20 đường địa chỉ, 8 đường số liệu và các đường tín hiệu khác. ....

Về mặt vật lý, bus IBM PC được thiết kế ngay trên bo mạch chính và có khoảng gần chục đầu nối dạng khe cắm (slot) để cắm các card mở rộng, trên mỗi khe cắm có 62 chân được chia thành hai hàng.

Bus IBM PC/AT

Bus IBM PC/AT là bước phát triển tiếp theo của thế hệ bus IBM PC nhằm phát huy được những khả năng hơn hẳn của bộ VXL 80286 so với 8088 trước nó. Với bus địa chỉ 24 dây, có khả năng đánh địa chỉ cho 2^{24} = 16MB bộ nhớ và có bus dữ liệu 16 bit.

Với giải pháp mở rộng PC bus, bổ sung thêm vào các khe cắm cũ một đoạn khe cắm ngắn, trên đó có 36 dây tín hiệu, tăng thêm cho bus địa chỉ 4 dây, bus dữ liệu 8 dây, các đường yêu cầu ngắt, kênh DMA, ... Nhờ vậy các card mở rộng trước đây vẫn dùng cho IBM PC có thể dùng cho IBM PC/AT.

Ngoài việc mở rộng bus, tần số tín hiệu đồng hồ bus cũng được tăng từ 4,77 MHz ở PC bus thành 8MHz, nhờ đó tốc độ truyền thông trên bus cũng tăng lên nhiều.

Năm 1991 tổ chức IEEE (Institute of Electrical and Electronic Engineers) đã đưa ra tiêu chuẩn quốc tế cho bus của máy AT, gọi là bus ISA (Industrial Standard Architecture)

Các nhà sản xuất PC khác đã đưa ra một chuẩn khác, đó là bus EISA (Extended ISA), về căn bản bus này là sự mở rộng bus PC/AT thành 32 bit, giữ nguyên tính tương thích với các máy tính và các card mở rộng đã có.

Ở thế hệ PS/2, thế hệ sau của IBM PC/AT một bus hoàn toàn mới được áp dụng, bus Micro chanel

Bus PCI

Vào đầu năm 1992, Intel đã thành lập nhóm công nghệ mới. Nhằm nghiên cứu cải thiện các đặc tính kỹ thuật và những hạn chế của các bus hiện có như: bus ISA, bus EISA.

PCI (Peripheral Component Interconnect, liên kết các thành phần ngoại vi). Định chuẩn bus PCI đã được đưa ra vào tháng 6 năm 1992 và được cập nhật vào tháng 4 năm 1993, đã thiết kế lại bus PC truyền thống bằng cách bổ sung thêm một bus khác vào giữa CPU và bus I/O.

Bus PCI thường được gọi là bus mezzanine vì nó bổ sung thêm một tầng khác vào cấu hình bus truyền thống. PCI bỏ qua bus I/O tiêu chuẩn, nó sử dụng bus hệ thống để tăng tốc độ đồng hồ bus lên và khai thác hết lợi thế của đường dẫn dữ liệu của CPU.

Thông tin được truyền qua bus PCI ở 33MHz và độ rộng dữ liệu đầy đủ của CPU. Khi bus ấy được sử dụng để nối với CPU 32 bit, dải thông là 132 MBit/s, được tính theo công thức: 33MHz*32bit/8 = 132MBit/s. Khi bus ấy được sử dụng với những hệ thống bổ sung 64 bit, dải thông tăng gấp đôi, nghĩa là tốc đọ truyền dữ liệu đạt tới 264MBs. Lý do chíng mà bus PCI đã tăng tốc độ nhanh hơn các bus khác là nó có thể hoạt động đồng thời với bus vi xử lý. CPU có thể được xử lý dữ liệu trong các cache ngoại trú, trong khi bus PCI phải truyền thông tin liên tục giữa các thành phần khác của hệ thống, đây là ưu điểm thiết kế chính của bus PCI.

Định chuẩn PCI có ba cấu hình, mỗi cấu hình được thiết kế cho một kiểu hệ thống riêng biệt với những quy định nguồn riêng. Định chuẩn 5V cho những hệ thống máy tính văn phòng, định chuẩn 3,3V cho các hệ thống máy tính xách tay và những định chuẩn chung cho những bo mẹ và các cạc hoạt động trong hai kiểu ấy.

===*Bus nối tiếp chung USB*

Bus USB (Universal Serial Bus) là một công nghệ bus mới đầy triển vọng, nhanh chóng phổ biến trong những thế máy tính ngày nay. Chủ yếu USB là cáp

cho phép nối lên tới 127 thiết bị bằng cách sử dụng chuỗi xích. Tuy nhiên nó truyền dữ liệu không nhanh bằng FireWire, ở tốc độ 12MBs nó có khả năng đáp ứng cho hầu hết các thiết bị ngoại vi. Định chuẩn USB được đưa ra vào năm 1996 do một hội đồng gồm những đại diện của các nhà sản xuất máy tính lớn như Compaq, Digital, IBM, NEC và Northen Telecom.

Một ưu điểm nổi bật của USB là những thiết bị ngoại vi tự nhận dạng, một đặc trưng hết sức thuận lợi cho việc cài đặt, xác lập các thiết bị ngoại vi. Đặc trưng này hoàn toàn tương thích với những công nghệ PnP và cung cấp tiêu chuẩn công nghệ cho kết nối tương lai. Hơn nữa, những thiết bị USB có khả năng cắm nóng

Chương VII: Ngôn ngữ Assembly

1. Tổng quan

Ngôn ngữ assembly (còn gọi là hợp ngữ) là một ngôn ngữ bậc thấp được dùng trong việc viết các chương trình máy tính. Ngôn ngữ assembly sử dụng các từ có tính gợi nhớ, các từ viết tắt để giúp ta dễ ghi nhớ các chỉ thị phức tạp và làm cho việc lập trình bằng assembly dễ dàng hơn. Mục đích của việc dùng các từ gợi nhớ là nhằm thay thế việc lập trình trực tiếp bằng ngôn ngữ máy được sử dụng trong các máy tính đầu tiên thường gặp nhiều lỗi và tốn thời gian. Một chương trình viết bằng ngôn ngữ assembly được dịch thành mã máy bằng một chương trình tiện ích được gọi là assembler (Một chương trình assembler khác với một trình biên dịch ở chỗ nó chuyển đổi mỗi lệnh của

chương trình assembly thành một lệnh Các chương trình viết bằng ngôn ngữ assembly liên quan rất chặt chẽ đến kiến trúc của máy tính. Điều này khác với ngôn ngữ lập trình bậc cao, ít phụ thuộc vào phần cứng.

Trước đây ngôn ngữ assembly được sử dụng khá nhiều nhưng ngày nay phạm vi sử dụng khá hẹp, chủ yếu trong việc thao tác trực tiếp với phần cứng hoặc hoặc làm các công việc không thường xuyên. Ngôn ngữ này thường được dùng cho trình điều khiển (tiếng Anh: driver), hệ nhúng bậc thấp (tiếng Anh: low-level embedded systems) và các hệ thời gian thực. Những ứng dụng này có ưu điểm là tốc độ xử lí các lệnh assembly nhanh.

a. **Cú pháp của hợp ngữ**

Chương trình hợp ngữ bao gồm các dòng lệnh.Một dòng lệnh là một lệnh

Mà trình biên dịch sẽ dịch ra mã máy,hay là một hướng dẫn biên dịch để chỉ dẫn cho trình biên dịch thực hiện một vài nhiệm vụ đặc biệt nào đó.Mỗi lệnh hay hướng dẫn biên dịch thường có 4 trường

Tên Toán tử Toán hạng ; Lời bình

Các trường phải được phân cách nhau bằng ít nhất một ký tự trống hay TAB. Cũng không bắt buộc phải sắp xếp các trường theo cột nhưng chúng nhất định phải xuất hiện theo đúng thứ tự nêu trên.

ví dụ : Láp : MOV AH, 1 ; Nhập một ký tự từ bàn phím .

Trong ví dụ này,trường tên là nhãn Láp, toán tử là MOV, toán hạng là AH

Và 1 lời bình là 'nhập một ký tự t ừ bàn phím'.

Ví dụ: về hướng dẫn biên dịch:

Cong2so PROC

Cong2so là tên và toán hạng là PROC. Dẫn hướng biên dịch này khai báo

một chương trình con có tên Cong2so

1.1.Trường tên

Trường tên được sử dụng làm nhãn lệnh ,các tên thủ tục và các tên biến.

Chương trình biên dịch sẽ chuyển các tên thành các địa chỉ bộ nhớ.Các tên có thể có chiều dài từ 1 đến 31 ký tự,có thể chứ a các chữ cái,chữ số và các ký tự đặc biệt (?, @ _ $%) .Không được phép chèn dấu trống vào giữa một tên.

Nếu sử dụng dấu chấm (.) thì nó phải đứng đầu tiên.Các tên được bắt đầu bằng một chữ cái.Chương trình biên dịch không phân biệt chữ hoa và chữ thường trong tên

Ví dụ các tên hợp lệ :

HANOI 1

@_VIDU

Lưu ý: các tên không hợp lệ :

- chúa một khoang trông

- bắt đầu bằng một chữ số

- dấu chấm không phải là ký tự đầu tiên

1.2 .Trường toán tử

Trong một lệnh,trường toán tử chứa mã lệnh dạng tượng trưng. Chương

Trình biên dịch sẽ chuyển mã lệnh dạng tượng trưng sang mã lệnh của ngôn ngữ máy.Tượng trưng của mã lệnh thường biểu thị chức năng của các thao tác.

Ví dụ như: MOV,ADD,SUB .

Trong một hướng dẫn biên dịch,trường toán tử chứa toán tử giả.Các toán tử giả sẽ không dịch ra mã máy mà đơn giản chúng chỉ báo cho trình biên dịch làm một việc gì đó.Chẳng hạn toán tử giả PROC đế tạo ra một thủ tục.

1.3 .Trường toán hạng

Đối với một lệnh,trường toán hạng xác định dữ liệu sẽ được các thao tác

Tác động lên.Một lệnh có thể không có,có 1 hoặc 2 toán hạng.

Vídụ:

NOP không toán hạng không làm gì cả

DEC BX một toán hạng,trừ 1 vào nội dung BX

MOD CX 10 hai toán hạng , khởi tạo CX =10

Trong một lệnh hai toán hạng,toán hạng đầu tiên gọi là toán hạng đích. Nó có thể là một thanh ghi hoặc một ô nhớ,là nơi chứa kết quả.Toán hạng thứ hai là toán hạng nguồn.Các lệnh thường không làm thay đổi toán hạng nguồn.

1.4 .Trường lời giải thích

Người lập chương trình thường sử dụng trường lời giải thích của một dòng lệnh để giải thích dòng lệnh đó làm cái gì.Mở đầu trường này là một dấu chấm phẩy (;) và trình biên dịch bỏ qua mọi cái được đánh dấu vào sau dấu chấm phẩy này.Hợp ngữ là ngôn ngữ bậc thấp cho nên ta hầu như không thể hiểu được một chương trình viết bằng hợp ngữ nếu không có lời bình.

Không nên viết những điều đã qua rõ rang như:

MOV CX, 0; chuyển 0 vào CX .

Thay vào đó,ta nên sử dụng các lời giải thích để đặt các chỉ thị vào trong ngữ cảnh của chương trình:

MOV CX, 0; khởi tạo vòng lặp CX=0

b. **Các biến**

Trong hợp ngữ các biến có vai trò going như trong các ngôn ngữ bậc cao . Mỗi biến có một kiểu dữ liệu và được chương trình gán cho một địa chỉ bộ nhớ.Các toán tử giả định nghĩa số liệu.Mỗi toán tử giả có thể được dung để thiết lập một hay nhiều dữ liệu của kiểu đã được đưa ra .

Trong phần này chúng ta sử dụng DB và DW để định nghĩa tạo nên một byte các biến kiểu byte và các biến kiểu word.

Các biến kiểu byte

Dẫn hướng định nghĩa một biến kiểu byte của trình biên dịch có dạng sau đây:

Tên DB giá_trị_khởi_tạo

Trong đó toán tử giả DB được hiểu là "định nghĩa byte"

Vídụ:

TONG DB 4

Với dẫn hướng này,Hợp ngữ sẽ gán tên TONG cho một byte nhớ và khởi tạo nó giá trị 4.Một dấu chấm hỏi (?) đặt ở vị trí của giá trị khởi tạo sẽ tạo nên một byte không được khởi tạo.

Vídụ:

TONG DB ?

Giới hạn thập phân của các giá trị khởi tạo nằm trong khoảng từ -128 đến 127 với kiểu có dấu và từ 0 đến 255 với kiểu không dấu.Các khoảng này vừa đúng giá trị của một byte.

Các biên kiêu word

Dẫn hướng định nghĩa một biến kiểu word của trình biên dịch có dạng sau đây:

Tên DW giá_trị_khởi_tạo

Toán tử giả DW có nghĩa là "định nghĩa word".

Vídụ:

BIEN1 DW?

BIEN2 DW- 8

BIEN3 DW16

Giống như với biến kiểu byte một dấu chấm hỏi ờ vị trí giá trị khởi tạo có nghĩa là word không được khởi tạo giá trị đầu.Giới hạn thập phân của giá trị khởi tạo được xácđịnh từ -32768 đến 32767 đối với kiểu có dấu và từ 0 đến 65535 đối với kiểu không dấu.

.Các biến mảng

Trong ngôn ngữ hợp ngữ,mảng chỉ là một chuỗi các byte nhớ hay Word.

Vídụ để định nghĩa mảng 3 byte có tên MANG với các giá trị khởi tạo là 5h , 10h ,15 h chúng ta có thể viết:

MANG DB 5h,10h,15 h

Tên MANG được gán cho byte đầu tiên , MANG+1 cho byte thứ hai và MANG+2 cho byte thứ ba . Nếu như trình biên dịch gán địa chỉ offset 0400 h cho MANG thì bộ nhớ sẽ như sau :

Phần tử	Địa chỉ	Nội dung
MANG	0400h	5h
MANG+1	0401h	10h
MANG+2	0402h	15h

Các biến mảng word có thể được định nghĩa một cách tương tự.

Vídụ: MANG DW 100,72 ,48,54

Sẽ tạo nên một mảng có 4 phần tử với các giá trị khởi tạo là 100 ,72,48, 54.Từ đầu tiên được gán với tên MANG, từ tiếp theo gán với MANG+2, rồi đến MANG+4 v.v . Nếu mảng bắt đầu tại 07F0 h thì bộ nhớ sẽ như sau :

Phân tử	Địa chỉ	Nội dung
MANG	07F0h	100d
MANG +2	07F2h	72d
MANG +4	07F4h	48d
MANG +6	07F6h	54d

Khi chúng ta khởi đầu các phần tử của mảng với cùng

một giá trị ta ding toán tử DUP trong lệnh .

c. **Các hằng có tên**

Để tạo ra các mã lệnh Hợp ngữ dễ hiểu, người ta thường dùng các tên tượng trưng để biểu diễn các hằng số.

EQU (EQates :coi như bằng) .

Để gán tên cho hằng,chúng ta có thể sử dụng toán tử giả EQU.

Cúpháp:

Tên EQU h ằ ng_s ố

Vídụ:

LF EQU OA h

sẽ gán tên LF ch o OAh. là mã ASCII của ký tự xuống dòng. Tên LF có thể được dùng để thay cho OA h tại bất cứ đâu trong chương trình.Trình biên dịch sẽ dịch các lệnh:

MOV DL,OAH

và:

MOV DL,LF

ra cùng một chỉ thị máy.

2. **Cấu trúc chương trình**

Phần khai báo Segment đơn giản

.MODEL kiêu

.STACK độ lớn (tính theo byte)

.DATA

Khai báo biến

.CODE

Nhãn:

Mov AX,@DATA

Mov DS,AX

Than chương trình

…

…

lệnh trở về DOS

[**các chương trình con**] (nêu có)

END Nhãn

3. **Các lệnh điều khiển**

a. Lệnh điều khiển .STASK

Cú pháp:

.STACK kích thước ngăn xếp

Chức năng: Xác định kích thước ngăn xếp (tính theo Byte).Với lệnh này DOS sẽ xác lập địa chỉ đầu của ngăn xếp và giá trị đó được ghi vào thanh ghi đoạn SS.

b. Lệnh điều .CODE

Cú pháp:

.COD E

Chức năng: Đánh dấu điểm khởi đầu của vùng nhớ chứa mã lệnh.

c. Lệnh điểu khiển .DATA

Cúpháp: .DAT A

Ph ần khai báo và gán giá trị ban đầu của các biến nhớ

Chức năng: Đánh dấu điểm khởi đầu của vùng nhớ chứa số liệu.

d. Lệnh điều khiển .MODEL

Cú pháp:

.MODEL kiểu bộ nhớ (Tiny ,Small,Medium,Compact,Large, Huge)

Chức năng: Xác định mô hình bộ nhớ cho một Module Assembly sử dụng tập lệnh điều khiển Segment đơn giản.

•Tiny: Cả phần mã máy của chương trình (CODE) phần dữ liệu (DATA) cùng nằm trong một Segment 64KB. Cả CODE và DATA đều là NEAR.

• Small:Phần mã máy của chương trình (CODE) có thể lớn hơn 64KB phần dữ liệu (DATA) cùng nằm trong một Segment 64KB.Cả CODE và DATA đều là NEAR.

• Medium: Phần mã máy của chương trình (CODE) nằm trong một Segment 64 KB phần dữ liệu (DATA) cùng nằm trong một 64KB.CODE là

FAR phầnDATA là NEAR.

•Compact: Phẩn mã máy của chương trình (CODE) nằm trong một Segment 64KB phần dữ liệu (DATA) cùng nằm trong một vùng nhớ lớn hơn 64KB .CODE là NEAR và DATA là FAR.

•Large: Phẩn mã máy của chương trình (CODE) và phần dữ liệu (DATA) nằm trong một vùng nhớ lớn hơn 64KB.CODE và DATA là FAR. Một trường số liệu không vượt quá 64KB.

•Huge:Phần mã máy c ủa chương trình (CODE) và phần dữ liệu (DATA) nằm trong một vùng nhớ lớn hơn 64KB.CODE và DATA là FAR. Cho phép trường số liệu vượt quá 64KB.

Vídụ:

Hãy viết một xâu ký tự 'XI NCHAOCÁCBAN!' ra màn hình.

Cách giải:

Dùng chức năng hiện một xâu ký tự kết thúc bằng dấu $ ra màn hình. Chức năng thứ 9 của hàm ngắt int2lh của DOS cho phép chúng ta hiện một xâu ký tự kết thúc bằng $ ra màn hình nếu D S : DX chứa địa chi SEG:OFFSE T của biến xâu.Do vậy chương trình sẽ như sau :

```
.MODEL   small

.STACK    lOO h

.DATA

Thao    db 'XI NCHAOCÁCBAN!$'   ;Khai báo  biến xâu ký tự

.COD E

Program1:

Mov      AX,@DATA          ;Đưa phẩn địa chi SEGMENT của phân

Mov      DS,AX.            ;đoạn dữ liệu vào thanh ghi segment DS

Mov      DX,OFFSET Tbao    ;DX chứa phần địa chi OFFSET

Mov      AH,9              ;Gọi hàm-hiện xâu ký tự
```

```
Int   21h                    ;Hiện xâu ký tự Tbao ra màn hình
Mov   AH,4Ch                 ;Kết thúc chương trình trở về DOS
Int   21 h
END  Programl
```

4. **Ngăn xếp và các thủ tục**

Ngăn xếp (stack)

STACK : là một cấu trúc dữ liệu một chiều. Các phần tử cất vào và lấy ra theo phương thức LIFO (Last In First Out). Mỗi chương trình phải dành ra một khối bộ nhớ để làm stack bằng khai báo STACK. Ví dụ :
.STACK 100H ; Xin cấp phát 256 bytes làm stack

Là 1 phần của bộ nhớ, được tổ chức lưu trữ dữ liệu theo cơ chế vào sau ra trước (LIFO).

Trong lập trình có khi cần truy xuất đến các phần tử trong STACK nhưng không được thay đổi trật tự của STACK. Để thực hiện điều này ta dùng thêm thanh ghi con trỏ BP : trỏ BP về đỉnh Stack : MOV BP,SP thay đổi giá trị của BP để truy xuất đến các phần tử trong Stack : [BP+2]

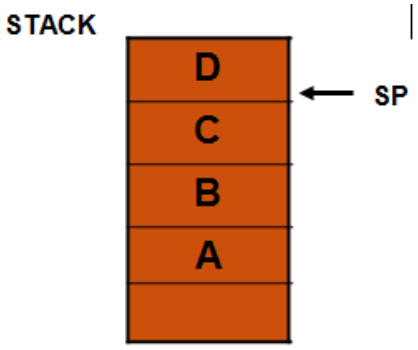
Phần tử được đưa vào STACK lần đầu tiên gọi là đáy STACK, phần tử cuối cùng được đưa vào STACK được gọi là đỉnh STACK.

Khi thêm một phần tử vào STACK ta thêm từ đỉnh, khi lấy một phần tử ra khỏi STACK ta cũng lấy ra từ đỉnh ➔ địa chỉ của ô nhớ đỉnh STCAK luôn luôn bị thay đổi.

SS dùng để lưu địa chỉ segemnt của đoạn bộ nhớ dùng làm STACK

SP để lưu địa chỉ của ô nhớ đỉnh STACK (trỏ tới đỉnh STACK)

Ví dụ:



A,B,C là các Word

```
MOV    BP,SP
MOV    AX,[BP]          ;AX =D
MOV    AX,[BP+2]        ;AX= C
MOV    AX,[BP+6]        ;AX=A
```

Để lưu 1 phần tử vào Stack ta dùng lệnh PUSH

Để lấy 1 phần tử ra từ Stack ta dùng lệnh POP

PUSH nguồn : đưa nguồn vào đỉnh STACK

PUSHF : cất nội dung thanh ghi cờ vào STACK

POP và POPF : dùng để lấy một phần tử ra khỏi STACK.

Cú pháp : POP đích : đưa nguồn vào đỉnh STACK

POPF : cất nội dung ở đỉnh STACK vào thanh ghi cờ

Chú ý : - Ở đây đích là một thanh ghi 16 bit (trừ thanh ghi IP) hay một từ nhớ .Các lệnh PUSH, PUSHF, POP và POPF không ảnh hưởng tới các cờ

Khai báo thủ tục

Cú pháp khai báo một thủ tục

Tên PROC type

;...cáclệnh trong thủ tục

RET

Tên ENDP

Tên: là tên của thủ tục do người viết định nghĩa

type: Toán hạng tuy chọn là:

- NEAR:Dòng lệnh gọi thủ tục ở cùng đoạn với thủ tục

- FAR: Dòng lệnh gọi thủ tục ở trong một đoạn khác

Các lệnh thủ tục CALL- RET

Lệnh CALL được dùng để gọi một thủ tục.

Cú pháp như sau :

CALL Tên

Trong đóTên là tên của thủ tục do người lập trình đặt.

Để trở về từ một thủ tục chúng ta dùng lệnh RET, mọi thủ tục ngoại trừ thủ t ục chính đều phải có lệnh RET ở cuối thủ tục.

Ví dụ về chương trình con

Trình bày 1 chương trình cộng 2 số gồm 2 phần tổ chức theo kiểu chương trình chính và chương trình con.Hai phần này truyền tham số với nhau thông qua ô nhớ dành cho biến (thanh ghi ngoài).

```
TITLE  CT :  ADD2SO

.MODEL  SMALL

.STACK  l00 h

.DATA

TBAO        DB          'HAY  VÀO  2 s o  NGUYÊN  $'

Soi  DB  ?

So 2  DB  7

Ton g  DB  7

.COD E

Program3:

MOV  AX,@Data            ;khởi tạo DS

MOV  DS,AX

MOV  AH,9                ;hàm hiện thông báo
```

```
LEA  DX,TBAO          ;nạp địa chỉ nội dung thông báo vào DX
INT  21 h             ;hiện thông báo
MOV  AH1              ;hàm đọc ký tự Soi
INT  21 h             ;đọc 1 ký tự
MOV  Sol,AL           ;cất mã của số 1
MOV  DL', '           ;dấu phẩy xen giữa
MOV  AH,2             ;hàm hiện ký tự ra màn hình
INT  21 h             ;hiện ký tự Số 1 ra màn hình
MOV  AH,1             ;hàm đọc kýt ự
INT   21 h            ;đọc ký t ự số 2
MOV  So2,AL           ;Cất mã c ủ a Số 2
CALL  CONG            ;gọi thủ tục cộng 2 số
MOV  AH,4Ch
INT  21 h             ;trở  về  DOS
CON G  Proc           ;chương trình con cộng hai số
MOV  AL,Sol           ;lấy mã Số 1
ADD  AL,So2           ;cộng với mã Số 2
ADD  Tong,AL          ;Đưa giá trị tổng vào  biến  Tong
```

```
RET                 ;lệnh trở về chương trình chính
CONG  Endp          ; kết thúc chương trình con
END  Program 3  ; kết thúc chương trình chính
```

TÀI LIỆU THAM KHẢO

1. Kiến trúc máy tính – Võ Văn Chín, Đại học Cần Thơ, 1997.

2. Computer Architecture: A Quantitative Approach, A. Patterson and J. Hennesy, Morgan Kaufmann Publishers, 2nd Edition, 1996.

3. Computer Otganization and Architecture: Designing for Performance, Sixth Edtion, William Stallings, Prentice Hall

MỤC LỤC