Msc. Võ Văn Chín
ThS. Nguyễn Hồng Vân
KS Phạm Hữu Tài

Giáo trình

KIẾN TRÚC MÁY TÍNH

Được biên soạn trong khuôn khổ dự án ASVIET002CNTT
"Tăng cường hiệu quả đào tạo và năng lực tự đào tạo của sinh viên khoa Công nghệ Thông tin - Đại học Cần thơ"

Đại học Cần Thơ - 12/2003

MỤC LỤC

*****

GIỚI THIỆU TỔNG QUAN

GIÁO TRÌNH KIẾN TRÚC MÁY TÍNH

MỤC ĐÍCH

Giáo trình này nhằm trang bị cho người đọc các nội dung chủ yếu sau:

- Lịch sử phát triển của máy tính, các thế hệ máy tính và cách phân loại máy tính. Cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự.
- Giới thiệu các thành phần cơ bản của một hệ thống máy tính, khái niệm về kiến trúc máy tính, tập lệnh. Các kiểu kiến trúc máy tính: mô tả kiến trúc, các kiểu định vị.
- Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý. Mô tả diễn tiến thi hành một lệnh mã máy và một số kỹ thuật xử lý thông tin: ống dẫn, siêu ống dẫn, siêu vô hướng, máy tính có lệnh thật dài, máy tính véc-tơ, xử lý song song và kiến trúc IA-64.
- Giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính.
- Giới thiệu một số thiết bị lưu trữ ngoài như: đĩa từ, đĩa quang, thẻ nhớ, băng từ. Hệ thống kết nối cơ bản các bộ phận bên trong máy tính. Cách giao tiếp giữa các ngoại vi và bộ xử lý.
- Phương pháp an toàn dữ liệu trên thiết bị lưu trữ ngoài.

YÊU CẦU

Sau khi học xong môn học này, người học được trang bị các kiến thức về:

- *S*inh viên được trang bị kiến thức về lịch sử phát triển của máy tính, các thế hệ máy tính và cách phân loại máy tính. Nắm vững các khái niệm cơ bản liên quan đến các hệ thống số được dùng trong máy tính. Thành thạo các thao tác biến đổi số giữa các hệ thống số.
- Sinh viên có kiến thức về các thành phần cơ bản của một hệ thống máy tính, khái niệm về kiến trúc máy tính, tập lệnh. Nắm vững các kiến thức về các kiểu kiến trúc máy tính, các kiểu định vị được dùng trong kiến trúc, loại và chiều dài của toán hạng, tác vụ mà máy tính có thể thực hiện. Phân biệt được hai loại kiến trúc: CISC (Complex Instruction Set Computer), RISC (Reduced Instruction Set Computer). Các kiến thức cơ bản về kiến trúc RISC, tổng quát tập lệnh của các kiến trúc máy tính.
- Sinh viên phải nắm vững cấu trúc của bộ xử lý trung tâm và diễn tiến thi hành một lệnh mã máy, vì đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật xử lý thông tin trong máy tính.
- Sinh viên phải hiểu được các cấp bộ nhớ và cách thức vận hành của các loại bộ nhớ được giới thiệu để có thể đánh giá được hiệu năng hoạt động của các loại bộ nhớ.
- Sinh viên phải nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ phận bên trong máy tính, cách giao tiếp giữa các ngoại vi và bộ xử lý. Biết được cấu tạo và các vận hành của các loại thiết bị lưu trữ ngoài và phương pháp an toàn dữ liệu trên đĩa cứng.

NỘI DUNG

- ***Chương I:*** **ĐẠI CƯƠNG**

Lịch sử phát triển của máy tính, thông tin và sự mã hoá thông tin.

- ***Chương II:*** **KIẾN TRÚC PHẦN MỀM BỘ XỬ LÝ**

Giới thiệu các thành phần cơ bản của một hệ thống máy tính, kiến trúc máy tính, tập lệnh và các kiểu định vị cơ bản. Khái niệm về kiến trúc RISC và CISC, ngôn ngữ cấp cao và ngôn ngữ máy.

- ***Chương III:*** **TỔ CHỨC BỘ XỬ LÝ**

Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý. Một số kỹ thuật xử lý thông tin.

- ***Chương IV:*** **CÁC CẤP BỘ NHỚ**

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

- ***Chương V:*** **NHẬP - XUẤT**

Thiết bị ngoại vi: các thành phần và hệ thống liên kết. Phương pháp an toàn dữ liệu trên thiết bị lưu trữ ngoài

KIẾN THỨC TIÊN QUYẾT

- KỸ THUẬT SỐ (TH 313)

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.

4. Principles of Computer Architecture, *Miles Murdocca and Vincent Heuring* (internet- http://iiusaedu.com).

5. Computer Organization and Design: The Hardware/Software Interface*, Patterson and Hennessy*, Second Edition (internet-http://engronline.ee.memphis.edu).

PHƯƠNG PHÁP HỌC TẬP

Do giáo trình chỉ mang tính chất giới thiệu tổng quát nên người đọc cần đọc thêm các tài liệu giới thiệu về kiến trúc cụ thể của các bộ xử lý. Người đọc cần tìm hiểu thêm các hình ảnh và ví dụ minh hoạ trong các tài liệu liên quan để thấy được sâu hơn vấn đề được đặt ra.

***Chương I*: ĐẠI CƯƠNG**

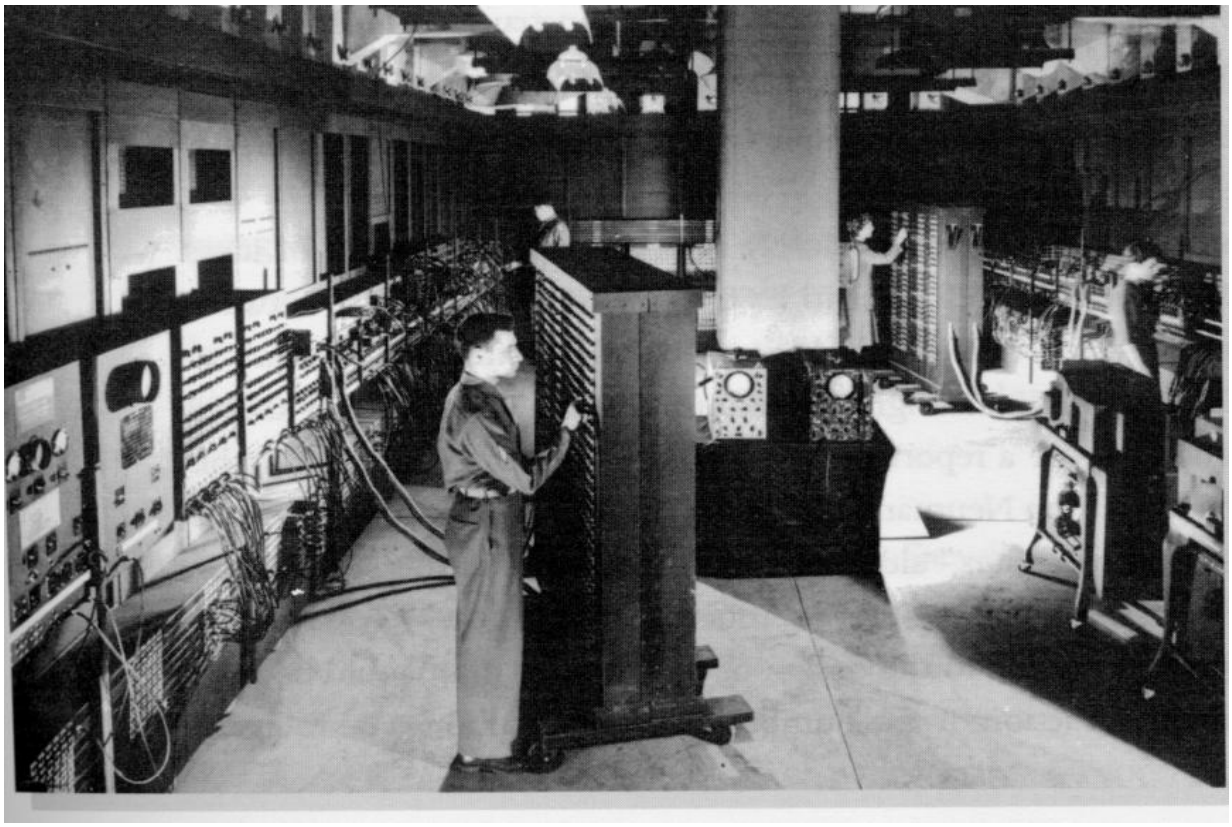
Mục đích: *Giới thiệu lịch sử phát triển của máy tính, các thế hệ máy tính và cách phân loại máy tính. Giới thiệu các cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự.*

Yêu cầu: *Sinh viên được trang bị kiến thức về lịch sử phát triển của máy tính, các thế hệ máy tính và cách phân loại máy tính. Nắm vững các khái niệm cơ bản liên quan đến các hệ thống số được dùng trong máy tính. Thành thạo các thao tác biến đổi số giữa các hệ thống số.*

I.1 CÁC THẾ HỆ MÁY TÍNH

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

a. Thế hệ đầu tiên (1946-1957)



Hình 1.1*: *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-????)

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 1.1*: Các thế hệ máy tính**

I.2 PHÂN LOẠI MÁY TÍNH

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

***a. Các siêu máy tính** (Super Computer):* là các máy tính đắt tiền nhất và tính năng kỹ thuật cao nhất. Giá bán một siêu máy tính từ vài triệu USD. Các siêu máy tính thường là các máy tính vectơ hay các máy tính dùng kỹ thuật vô hướng và được thiết kế để tính toán khoa học, mô phỏng các hiện tượng. Các siêu máy tính được thiết kế với kỹ

thuật xử lý song song với rất nhiều bộ xử lý (hàng ngàn đến hàng trăm ngàn bộ xử lý trong một siêu máy tính).

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

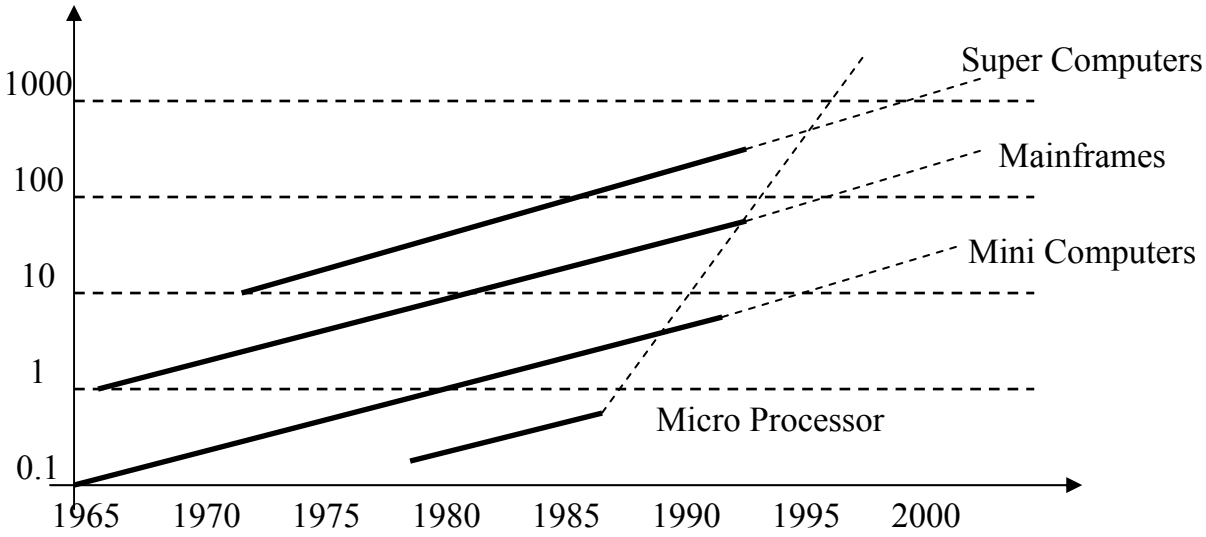
c. Máy tính mini *(Minicomputer)* là loại máy cở trung, giá một máy tính mini có thể từ vài chục USD đến vài trăm ngàn USD.

d. Máy vi tính *(Microcomputer)* là loại máy tính dùng bộ vi xử lý, giá một máy vi tính có thể từ vài trăm USD đến vài ngàn USD.

I.3 THÀNH QUẢ CỦA MÁY TÍNH

QUI LUẬT MOORE VỀ SỰ PHÁT TRIỂN CỦA MÁY TÍNH

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



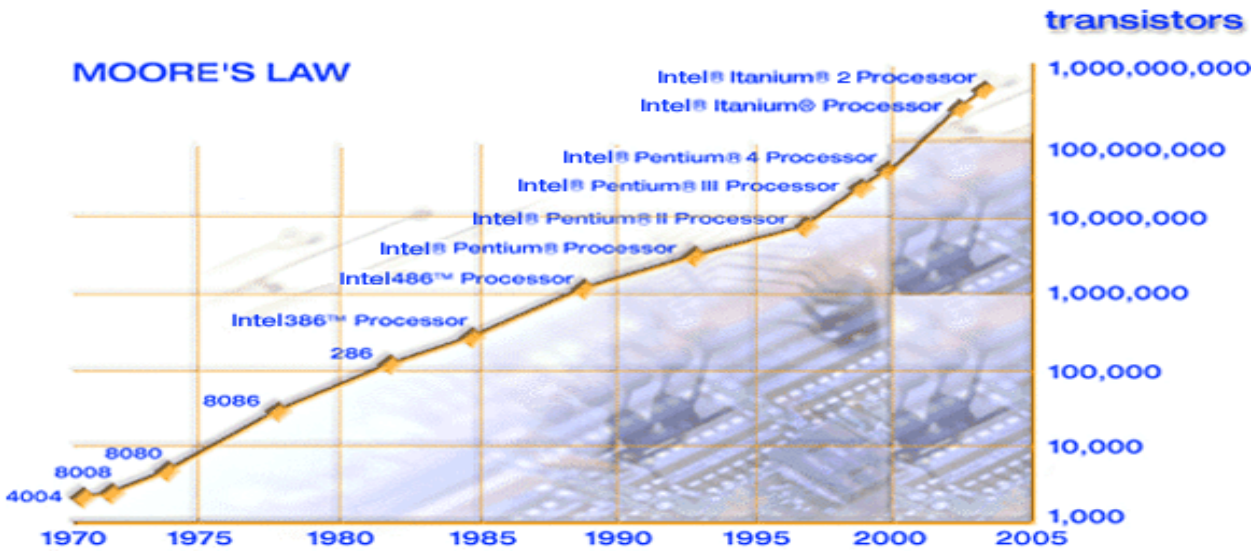
Hình 1.2: Đánh giá thành quả của máy tính

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

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

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

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



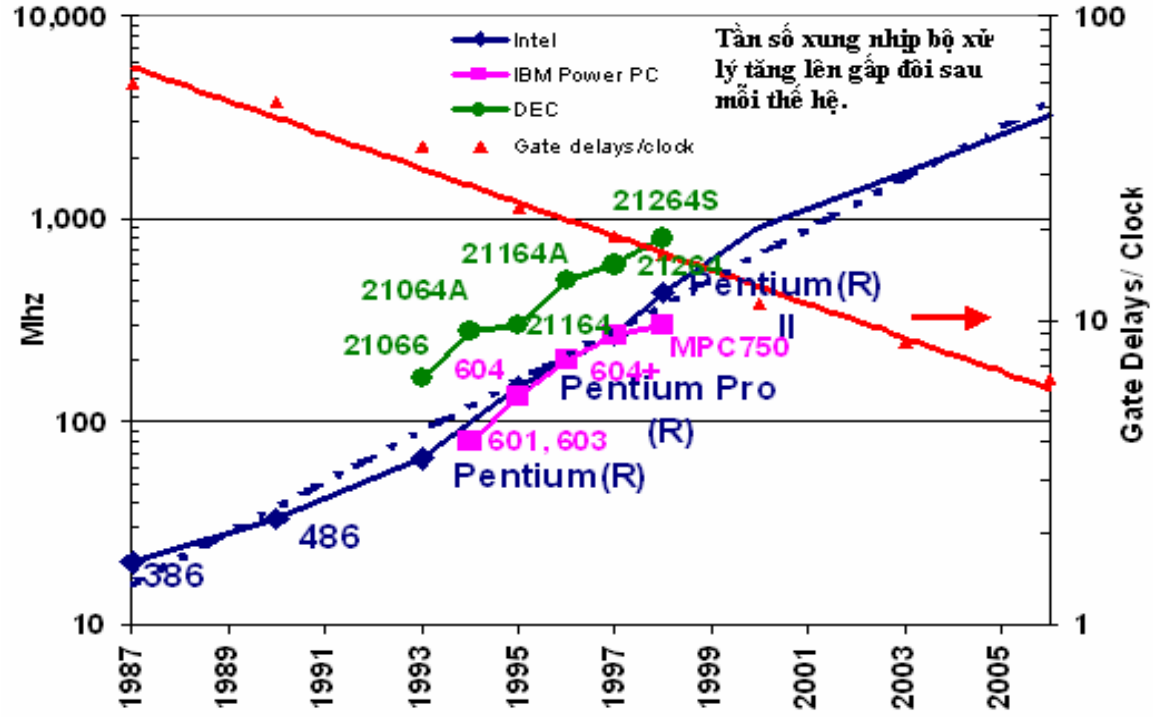
Bộ xử lý Intel	*Năm SX*	*Số lượng transistor tích hợp*
4004	**1971**	**2,250**
8008	**1972**	**2.500**
8080	**1974**	**5,000**
8086	**1978**	**29,000**
286	**1982**	**120,000**
Intel386™ processor	**1985**	**275,000**
Intel486™ processor	**1989**	**1,180,000**
Intel® Pentium® processor	**1993**	**3,100,000**
Intel® Pentium® II processor	**1997**	**7.500.000**
Intel® Pentium® III processor	**1999**	**24.000.000**
Intel® Pentium® 4 processor	**2000**	**42,000,000**
Intel® Itanium® processor	**2002**	**220.000.000**
Intel® Itanium® 2 processor	**2003**	**410,000,000**

Hình I.3 và Bảng I.2: Sự phát triển của bộ xử lý Intel dựa vào số lượng transistor trong một mạch tích hợp theo qui luật Moore

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

Kết quả của quy luật Moore là:

- Chi phí cho máy tính sẽ giảm.
- Giảm kích thước các linh kiện, máy tính sẽ giảm kích thước
- Hệ thống kết nối bên trong mạch ngắn: tăng độ tin cậy, tăng tốc độ .
- Tiết kiệm năng lượng cung cấp, toả nhiệt thấp.
- Các IC thay thế cho các linh kiện rời.



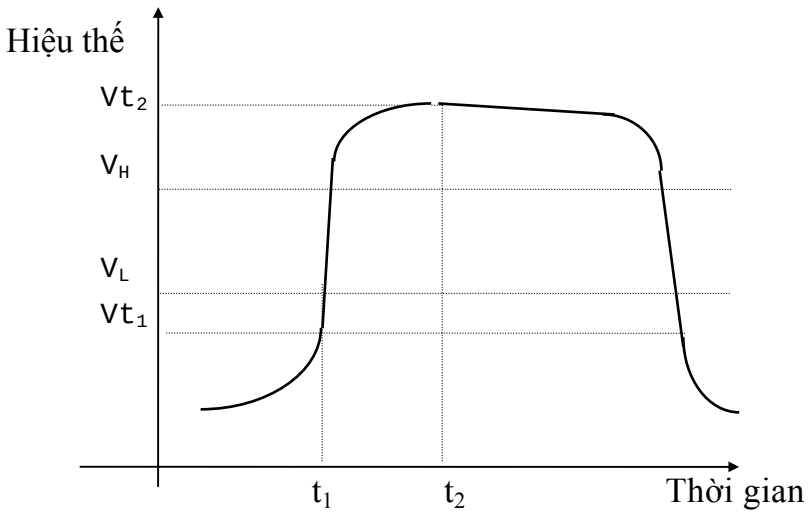
Hình I.4: Xung nhịp các bộ xử lý MOS

Một số khái niệm liên quan:

- Mật độ tích hợp là số linh kiện tích hợp trên một diện tích bề mặt tấm silicon cho sẵn, cho biết số nhiệm vụ và mạch có thực hiện.
- Tần số xung nhịp bộ xử lý cho biết tần số thực hiện các nhiệm vụ.
- Tốc độ xử lý của máy tính trong một giây (hay công suất tính toán của mỗi mạch): được tính bằng tích của mật độ tích hợp và tần số xung nhịp. Công suất này cũng tăng theo hàm mũ đối với thời gian.

I.4- THÔNG TIN VÀ SỰ MÃ HOÁ THÔNG TIN

I.4.1 - Khái niệm thông tin



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

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

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

I.4.2 - Lượng thông tin và sự mã hoá thông tin

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

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

Trong đó: I: là lượng thông tin tính bằng bit
N: là số trạng thái có thể có

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

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

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

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

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 I.3: Tám trạng thái khác nhau ứng với 3 số nhị phân

I.4.3 - Biểu diễn các số:

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

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

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

Trong đó:

V_k: Số cần biểu diễn giá trị

m: số thứ tự của chữ số phần lẻ

(phần lẻ của số có m chữ số được đánh số thứ tự từ -1 đến -m)

n-1: số thứ tự của chữ số phần nguyên

(phần nguyên của số có n chữ số được đánh số thứ tự từ 0 đến n-1)

b_i: giá trị của chữ số thứ i

k: hệ số (k=10: hệ thập phân; k=2: hệ nhị phân;...).

Ví dụ: biểu diễn số 541.25_{10}

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

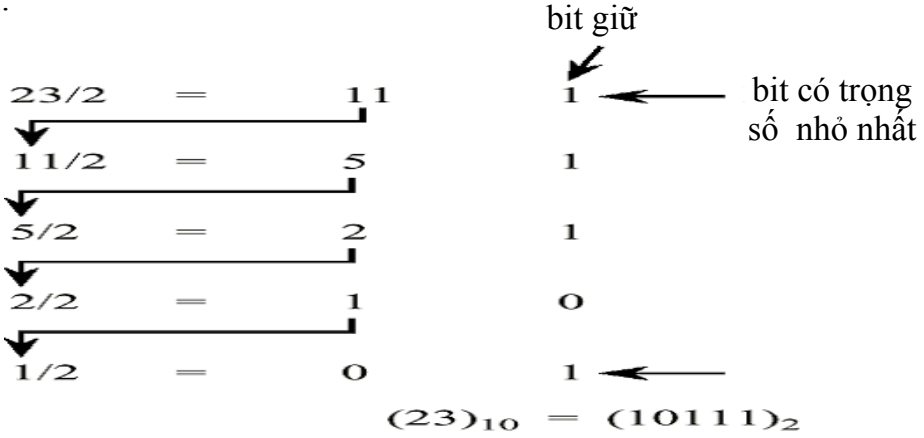
$= (500)_{10} + (40)_{10} + (1)_{10} + (2/10)_{10} + (5/100)_{10}$

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

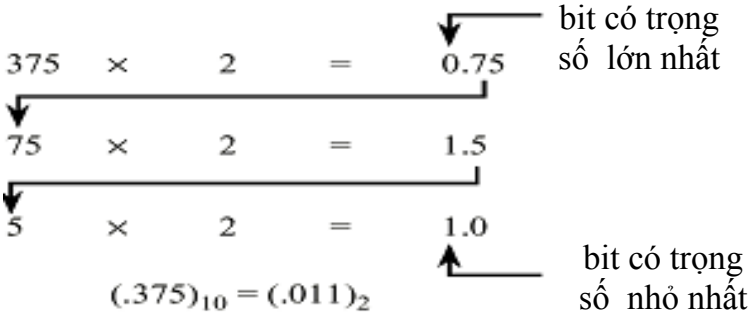
Để biến đổi một số hệ thập phân sang nhị phân, ta có hai phương thức biến đổi:

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

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



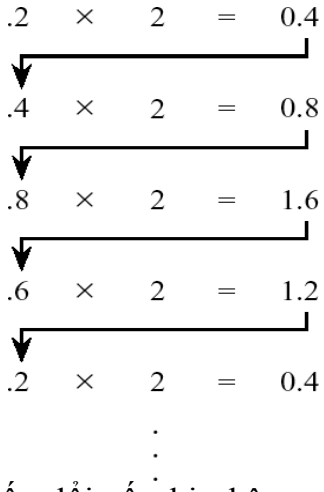
- Phương thức nhân để biến đổi phần lẻ của số thập phân sang nhị phân



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

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

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:

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

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

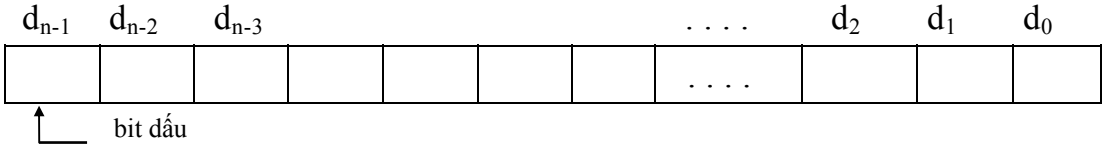
$$N = \sum_{i=0}^{n-1} d_i 2^i$$

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

I.4.4 Số nguyên có dấu

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

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



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

a) Cách biểu diễn bằng trị tuyệt đối và dấu

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

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

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 10011001_2$

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

b) Cách biểu diễn hằng số bù 1

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

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

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

c) Cách biểu diễn bằng số bù 2

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

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

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

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

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

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 I.4: Số 4 bit có dấu theo cách biểu diễn số âm bằng số bù 2

d) Cách biểu diễn bằng số thừa K

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

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

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

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

I.4.5 - 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:

$254 * 10^0$; $25.4 * 10^1$; $2.54 * 10^2$; $0.254 * 10^3$; $0.0254 * 10^4$; …

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

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

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:

$$\pm 1.fff...f \times 2^{\pm E}$$

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	.....	f_{22}	f_{23}

Hình I.7: 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	.....	f_{51}	f_{52}

Hình I.8: 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 (bit)	**23**	**≥ 31**	**52**	**≥63**

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).

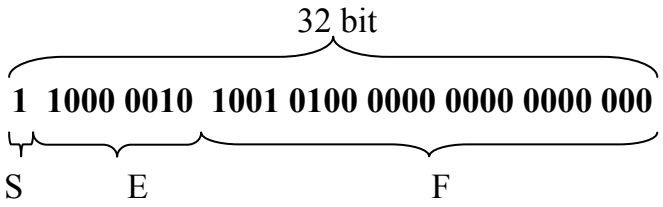
Ví dụ các bước biến đổi số thập phân -12.625_{10} sang số chấm động chuẩn IEEE 754 chính xác đơn (32 bit):

- 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 ($1000\ 0010_2$).



I.4.6 - Biểu diễn các số thập phân

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

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

Bảng I.5: Số thập phân mã 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

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

Cách tính toán trên tương đương với cách sau:

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

I.4.7 - 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	IFS
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

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

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

Bảng mã UNICODE

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG I

*****

1. Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ?
2. Đặc trưng cơ bản của các máy tính thế hệ thứ nhất?
3. Đặc trưng cơ bản của các máy tính thế hệ thứ hai?
4. Đặc trưng cơ bản của các máy tính thế hệ thứ ba?
5. Đặc trưng cơ bản của các máy tính thế hệ thứ tư?
6. Khuynh hướng phát triển của máy tính điện tử ngày nay là gì?
7. Việc phân loại máy tính dựa vào tiêu chuẩn nào?
8. Khái niệm thông tin trong máy tính được hiểu như thế nào?
9. Lượng thông tin là gì ?
10. Sự hiểu biết về một trạng thái trong 4096 trạng thái có thể có ứng với lượng thông tin là bao nhiêu?
11. Điểm chung nhất trong các cách biểu diễn một số nguyên n bit có dấu là gì?
12. Số nhị phân 8 bit $(11001100)_2$, số này tương ứng với số nguyên thập phân có dấu là bao nhiêu nếu số đang được biểu diễn trong cách biểu diễn:
    a. Dấu và trị tuyệt đối.
    b. Số bù 1.
    c. Số bù 2.
13. Đổi các số sau đây:
    a. $(011011)_2$ ra số thập phân.
    b. $(-2005)_{10}$ ra số nhị phân 16 bits.
    c. $(55.875)_{10}$ ra số nhị phân.
14. Biểu diễn số thực (31.75)10 dưới dạng số có dấu chấm động chính xác đơn 32 bit.

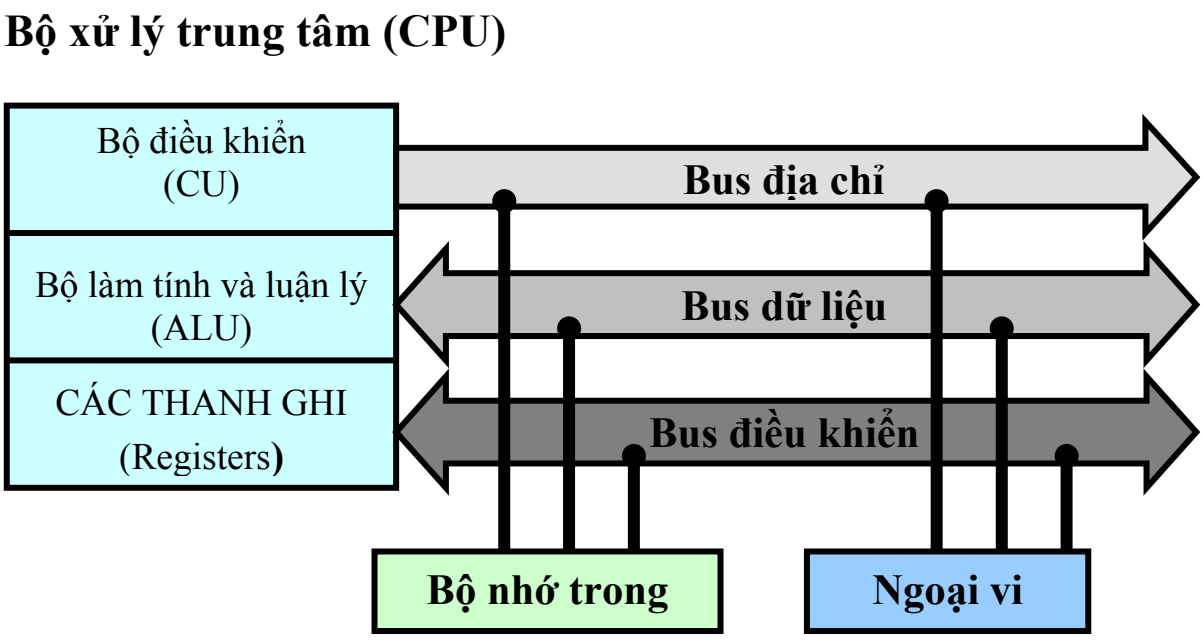
Chương II: KIẾN TRÚC PHẦN MỀM BỘ XỬ LÝ

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

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

II.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.



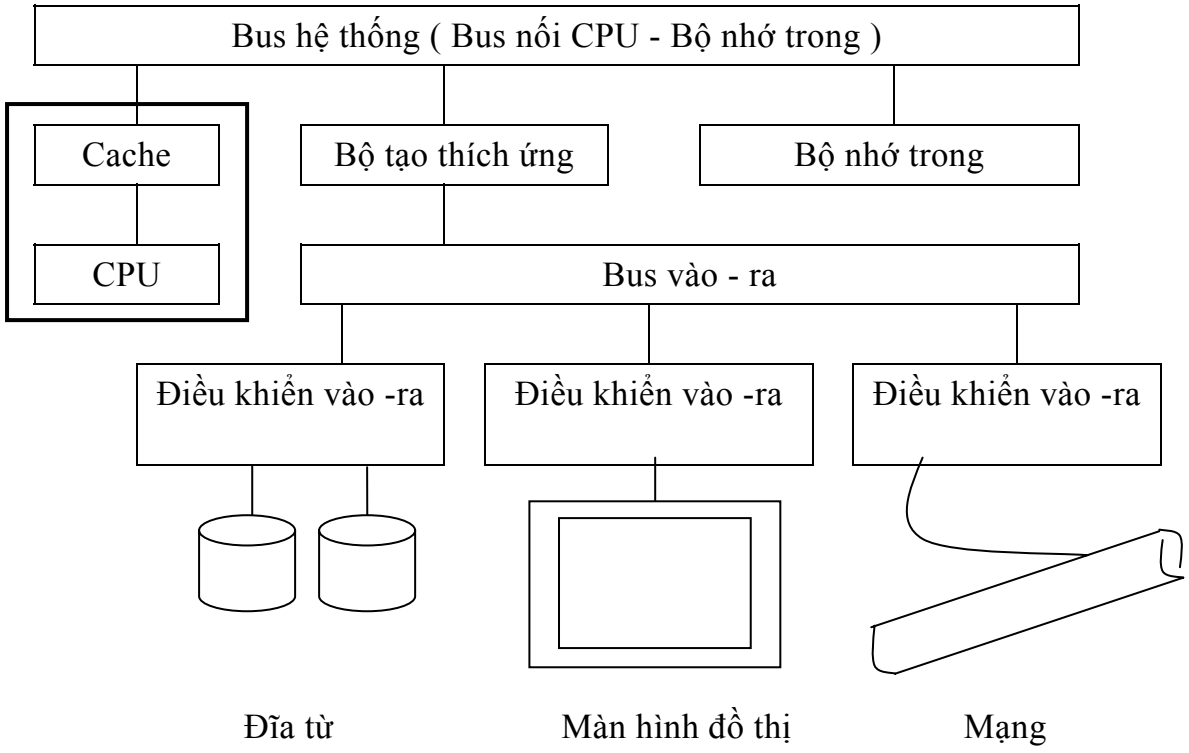
Hình II.1*: Cấu trúc của một hệ máy tính đơn giản*

Một chương trình sẽ được sao chép từ đĩa cứng vào bộ nhớ trong cùng với các thông tin cần thiết cho chương trình hoạt động, các thông tin này được nạp vào bộ nhớ

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

Thành phần cơ bản của một máy tính bao gồm :

- ***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 lý (ALU: Arithmetic And Logic Unit) và các thanh ghi. Nó có nhiệm vụ làm các phép toán trên số liệu. Phần điều khiển có nhiệm vụ đảm bảo thi hành các lệnh một cách tuần tự và tác động các mạch chức năng để thi hành các lệnh.
- ***Bộ phận vào - ra:*** đây là bộ phận xuất nhập thông tin, bộ phận này thực hiện sự giao tiếp giữa máy tính và người dùng hay giữa các máy tính trong hệ thống mạng (đối với các máy tính được kết nối thành một hệ thống mạng). Các bộ phận xuất nhập thường gặp là: bộ lưu trữ ngoài, màn hình, máy in, bàn phím, chuột, máy quét ảnh, các giao diện mạng cục bộ hay mạng diện rộng...Bộ tạo thích ứng là một vi mạch tổng hợp (chipset) kết nối giữa các hệ thống bus có các tốc độ dữ liệu khác nhau.



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

II.2 - ĐỊNH NGHĨA 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.

II.3 - CÁC KIỂU THI HÀNH MỘT LỆNH

Như đã mô tả, một lệnh mã máy bao gồm một mã tác vụ và các toán hạng.

Ví dụ: lệnh mã máy 01101001010101010000001101100101

Việc chọn số toán hạng cho một lệnh mã máy là một vấn đề then chốt vì phải có một sự cân đối giữa tốc độ tính toán và số các mạch tính toán phải dùng. Tuỳ theo tần số sử dụng các phép như trên mà các nhà thiết kế máy tính quyết định số lượng các mạch chức năng cần thiết cho việc tính toán. Thông thường số toán hạng thay đổi từ 0 tới 3.

Ví dụ: lệnh Y := A + B + C + D có thể được hiện bằng một lệnh mã máy nếu ta có 3 mạch cộng, hoặc được thực hiện bằng 3 lệnh mã máy nếu chúng ta chỉ có một mạch cộng, nếu việc tính toán trên xảy ra ít, người ta chỉ cần thiết kế một mạch cộng thay vì phải tốn chi phí lắp đặt 3 mạch cộng. Tuy nhiên, với một mạch cộng thì thời gian tính toán của hệ thống sẽ chậm hơn với hệ thống có ba mạch cộng.

Vị trí của toán hạng cũng được xem xét. Bảng II.1 chọn một vài nhà sản xuất máy tính và 3 kiểu cơ bản của vị trí các toán hạng đối với những lệnh tính toán trong ALU là: ở ngăn xếp, trên thanh ghi tích luỹ, và trên các thanh ghi đa dụng. Những kiến trúc phần mềm này được gọi là *kiến trúc ngăn xếp*, *kiến trúc thanh ghi tích luỹ* và *kiến trúc thanh ghi đa dụng*.

Vị trí các toán hạng	Thí dụ	Toán hạng cho lệnh tính toán trong ALU	Vị trí đặt kết quả	Cách thức thâm nhập vào toán hạng
Ngăn xếp	B 5500 HP 3000/70	0	Ngăn xếp	Lệnh Push, Pop
Thanh ghi tích luỹ	PDP 8 Motorola 6809	1	Thanh ghi tích luỹ	Lệnh nạp vào hoặc lấy ra từ thanh ghi tích luỹ (load, store)
Thanh ghi đa dụng	IBM 360 DEC, VAX	2 hoặc 3	Thanh ghi hoặc bộ nhớ	Lệnh nạp vào hoặc lấy ra từ thanh ghi hoặc bộ nhớ

***Bảng II.1** : Ví dụ về cách chọn lựa vị trí các toán hạng*

Một vài nhà sản xuất máy tính tuân thủ chặt chẽ các kiểu chọn vị trí toán hạng nêu trên, nhưng phần nhiều các bộ xử lý dùng kiểu hỗn tạp. Ví dụ, mạch xử lý 8086 của Intel dùng cùng một lúc kiểu "thanh ghi đa dụng" và kiểu "thanh ghi tích luỹ".

Ví dụ minh hoạ chuỗi lệnh phải dùng để thực hiện phép tính C := A + B trong 3 kiểu kiến trúc phần mềm.

Kiến trúc ngăn xếp	**Kiến trúc thanh ghi tích luỹ**	**Kiến trúc thanh ghi đa dụng**
Push A Push B ***ADD*** Pop C	Load A ***ADD B*** Store C	Load R1, A ***ADD R1, B*** Store R1, C

***Bảng II.2:** Chuỗi lệnh dùng thực hiện phép tính C := A + B*
(giả sử A, B, C đều nằm trong bộ nhớ trong)

Hiện tại các nhà sản xuất máy tính có khuynh hướng dùng kiến trúc phần mềm thanh ghi đa dụng vì việc thâm nhập các thanh ghi đa dụng nhanh hơn thâm nhập bộ nhớ trong, và vì các chương trình dịch dùng các thanh ghi đa dụng có hiệu quả hơn.

Loại kiến trúc	**Lợi điểm**	**Bất lợi**
Ngăn xếp *(Stack)*	- Lệnh ngắn - Ít mã máy - Làm tối thiểu trạng thái bên trong của máy tính - Dễ dàng tạo ra một bộ biên dịch đơn giản cho kiến trúc ngăn xếp	- Thâm nhập ngăn xếp không ngẫu nhiên. - Mã không hiệu quả - Khó dùng trong xử lý song song và ống dẫn - Khó tạo ra một bộ biên dịch tối ưu
Thanh ghi tích luỹ *(Accumulator Register)*	- Lệnh ngắn - Làm tối thiểu trạng thái bên trong của máy tính (yêu cầu ít mạch chức năng). - Thiết kế dễ dàng	- Lưu giữ ở thanh ghi tích luỹ là tạm thời. - Nghẽn ở thanh ghi tích luỹ - Khó dùng trong xử lý song song và ống dẫn - Trao đổi nhiều với bộ nhớ.

Thanh ghi đa dụng *(General Register)*	- Tốc độ xử lý nhanh, định vị đơn giản. - Ít thâm nhập bộ nhớ. - Kiểu rất tổng quát để tạo các mã hữu hiệu	- Lệnh dài - Số lượng thanh ghi bị giới hạn

***Bảng II.3:* Điểm lợi và bất lợi của 3 kiểu kiến trúc phần mềm**

II.4 - KIỂU KIẾN TRÚC THANH GHI ĐA DỤNG

Do hiện nay kiểu kiến trúc thanh ghi đa dụng chiếm vị trí hàng đầu nên trong các phần sau, ta chỉ đề cập đến kiểu kiến trúc này.

Đối với một lệnh tính toán hoặc logic điển hình (lệnh ALU), có 2 điểm cần nêu lên.

Trước tiên, một lệnh ALU phải có 2 hoặc 3 toán hạng. Nếu trong lệnh có 3 toán hạng thì một trong các toán hạng chứa kết quả phép tính trên hai toán hạng kia (Ví dụ: add A, B, C). Nếu trong lệnh có 2 toán hạng thì một trong hai toán hạng phải vừa là toán hạng nguồn, vừa là toán hạng đích (Ví dụ: add A, B).

Thứ hai, số lượng toán hạng bộ nhớ có trong lệnh. Số toán hạng bộ nhớ có thể thay đổi từ 0 tới 3.

Trong nhiều cách tổ hợp có thể có các loại toán hạng của một lệnh ALU, các máy tính hiện nay chọn một trong 3 kiểu sau : thanh ghi-thanh ghi (kiểu này còn được gọi *nạp - lưu trữ*), thanh ghi - bộ nhớ và bộ nhớ - bộ nhớ.

Kiểu thanh ghi - thanh ghi được nhiều nhà chế tạo máy tính lưu ý với các lý do: việc tạo các mã máy đơn giản, chiều dài mã máy cố định và số chu kỳ xung nhịp cần thiết cho việc thực hiện lệnh là cố định, ít thâm nhập bộ nhớ. Tuy nhiên, kiểu kiến trúc này cũng có một vài hạn chế của nó như: số lượng thanh ghi bị giới hạn, việc các thanh ghi có cùng độ dài dẫn đến không hiệu quả trong các lệnh xử lý chuỗi cũng như các lệnh có cấu trúc. Việc lưu và phục hồi các trạng thái khi có các lời gọi thủ tục hay chuyển đổi ngữ cảnh.

II.5 - TẬP 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 vòng. 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 vòng 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 vòng.

II.5.1 - 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]

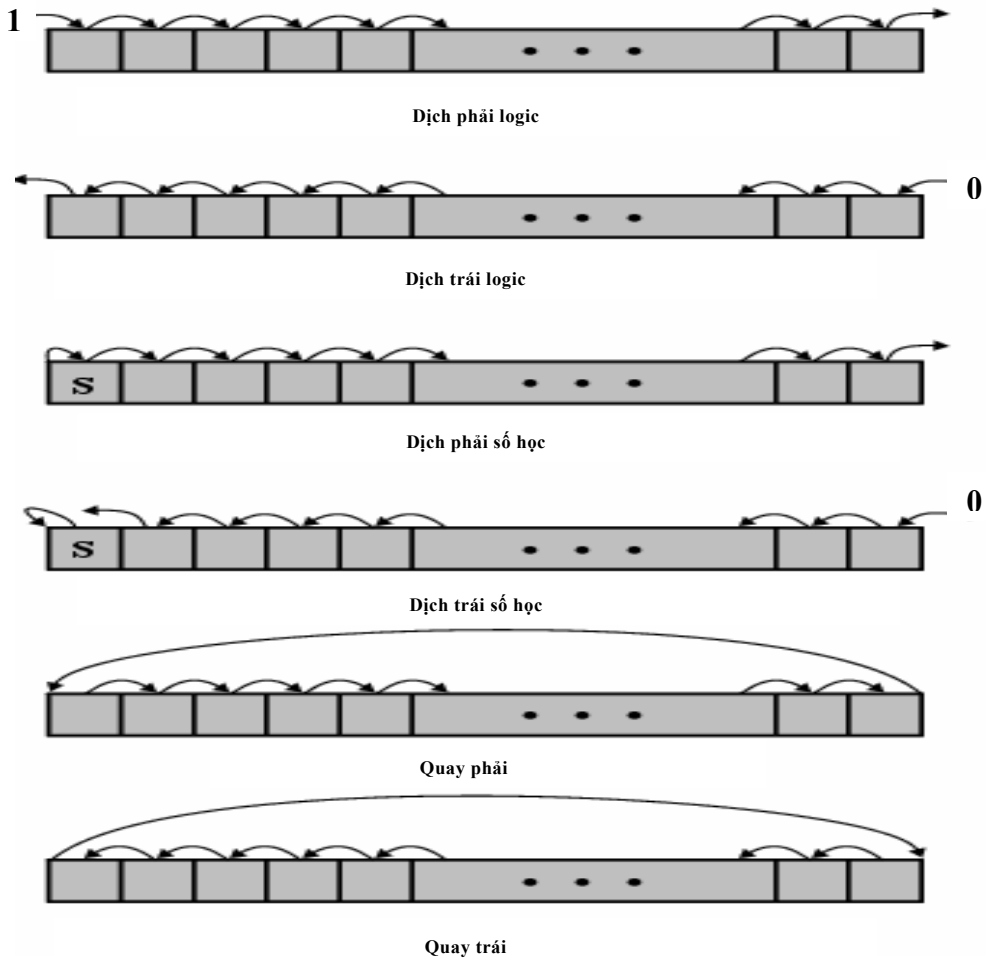
Địa chỉ được tính tuỳ theo kiểu định vị được dùng.

- ***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)



Hình II.7: Minh hoạ lệnh dịch chuyển và quay vòng

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

II.5.2 - Lệnh có điều kiện

Lệnh có điều kiện có dạng :

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

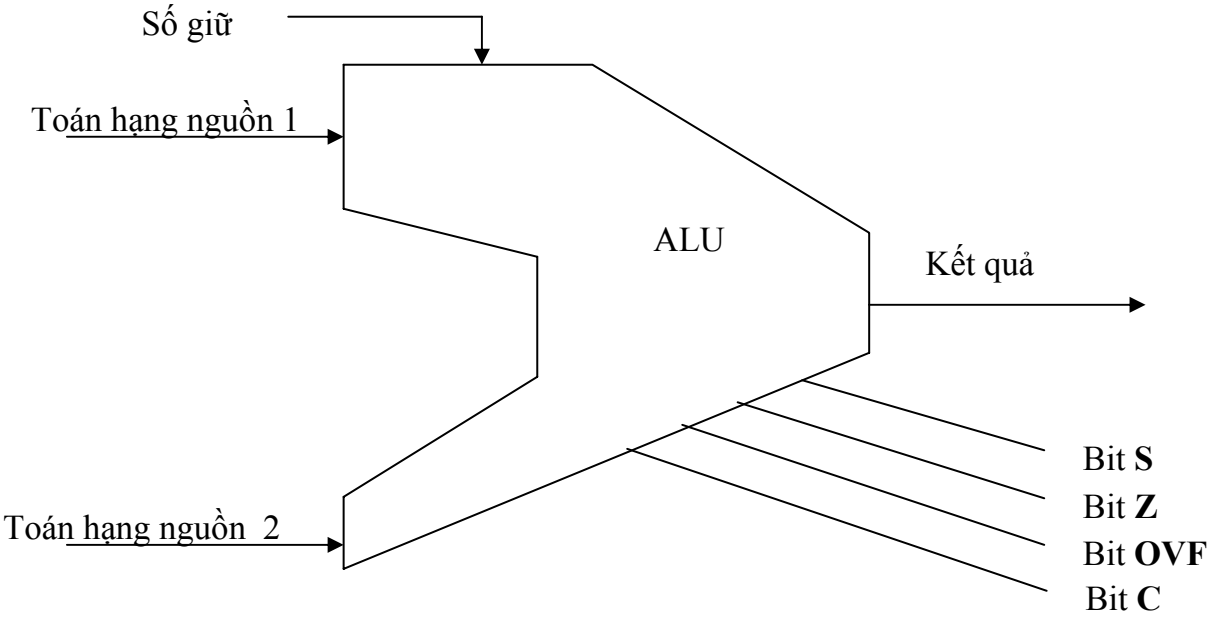
(IF <condition> THEN <instructions1> ELSE <instructions2>)

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

a) Ghi nhớ điều kiện .

Bộ làm tính ALU cung cấp kết quả ở ngã ra tuỳ theo các ngã vào và phép tính cần làm. Nó cũng cho một số thông tin khác về kết quả dưới dạng các bit trạng thái. Các bit này là những đại lượng logic ***ĐÚNG*** hoặc ***SAI*** (hình 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 ngã ra là 1 .... Các bit trên thường được gọi là bit mã 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 trạng thái vào một thanh ghi đặc biệt gọi là ***thanh ghi trạng thái***. Vấn đề lưu giữ nội dung thanh ghi này được giải quyết bằng nhiều cách. Trong kiến trúc SPARC, chỉ có một số giới hạn lệnh được phép thay đổi thanh ghi trạng thái ví dụ như lệnh ADDCC, SUBCC (các lệnh này thực hiện các phép tính cộng ADD và phép tính trừ SUB và còn làm thay đổi thanh ghi trạng thái). Trong kiến trúc PowerPC, thanh ghi trạng thái được phân thành 8 trường, mỗi trường 4 bit, vậy là thanh ghi đã phân thành 8 thanh ghi trạng thái con.

b) Nhảy vòng

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

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

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

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

Nếu R1 > R2 thì chuỗi lệnh được thi hành là 1, 2, 5, 6 được thi hành,
nếu không thì chuỗi lệnh 1, 2, 3, 4, 6 được thi hành.
Chuỗi các lệnh trên , trong đó có 2 lệnh nhảy, thực hiện công việc sau đây :
Nếu R1 > R2 thì R3 = R1 nếu không R3 = R2

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

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

II.5.3 - 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.

II.5.4 - 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
```

II.5.5 - Các thủ tục

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

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

Lệnh gọi thủ tục là một lệnh loại JMPL Ri, lệnh này làm các tác vụ :

```
R31 := PC     ; để địa chỉ trở về trong R31
PC := Ri      ; nhảy tới địa chỉ của thủ tục nằm trong thanh ghi Ri
```

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

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

Giải pháp 1: có nhiều thanh ghi để lưu trữ địa chỉ trở về

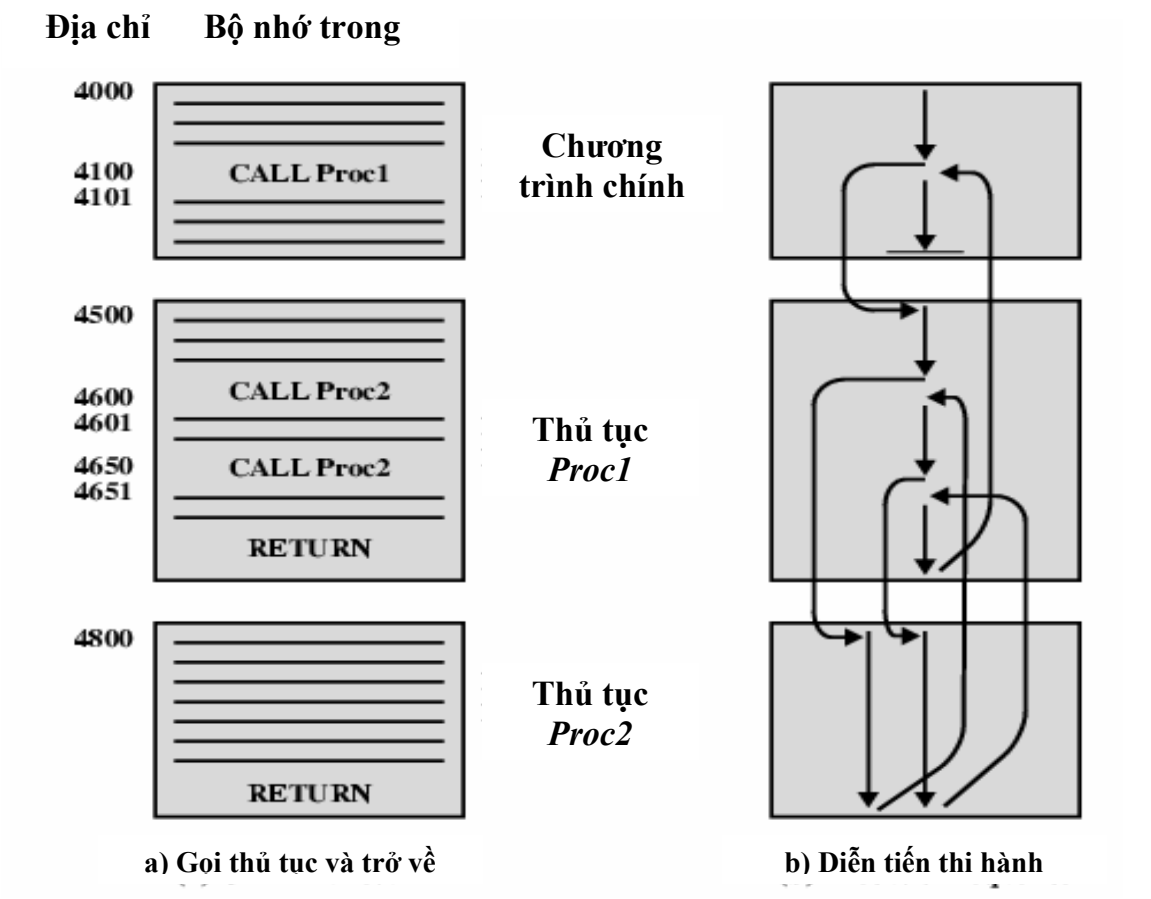
Giải pháp 2: lưu giữ địa chỉ trở về ở ngăn xếp.

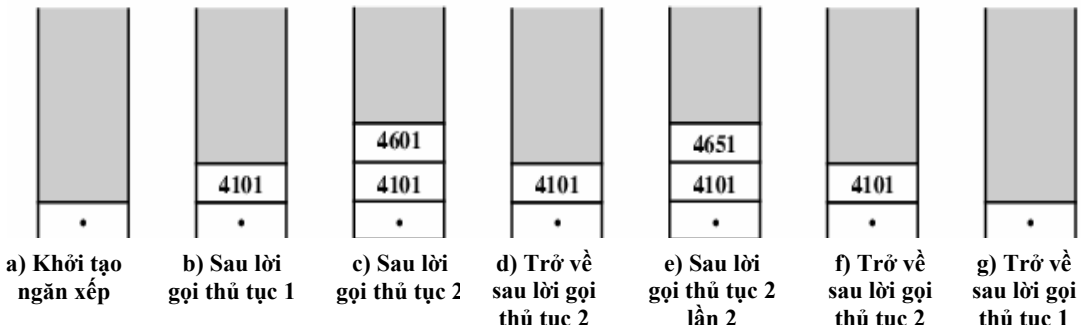
Việc gọi thủ tục có thể được thực hiện bằng chuỗi lệnh sau đây :

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

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

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





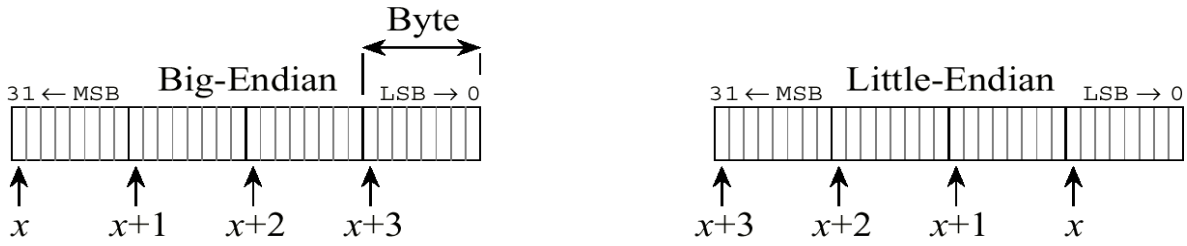
***Hình II.9*: Gọi thủ tục và trở về khi thực hiện xong thủ tục**

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

II.6 - CÁC KIỂU ĐỊNH VỊ

Kiểu định vị định nghĩa cách thức thâm nhập các toán hạng. Một vài kiểu xác định cách thâm nhập toán hạng bộ nhớ, nghĩa là cách tính địa chỉ của toán hạng, các kiểu khác xác định các toán hạng nằm trong các thanh ghi.

Chú ý rằng, trong các kiểu định vị, ta cần lưu ý khi chuyển đổi dữ liệu nhị phân giữa hai kiểu định địa chỉ liên quan đến ô nhớ, vì mỗi từ máy tính gồm bốn byte, mỗi ô nhớ chứa một byte. Như vậy, một từ máy tính được lưu trong bốn ô nhớ liên tiếp trong bộ nhớ trong, có nhiều cách xác một từ máy tính, trong đó, hai cách tiêu biểu nhất là:



Địa chỉ từ là x cho cả hai minh hoạ

***Hình II.3*: Minh hoạ hai cách sắp xếp địa chỉ trong bộ nhớ**

- *Định vị kiểu Big-Endian*: byte thấp nhất được đặt trong ô nhớ có địa chỉ cao nhất (IBM, Motorolla, Sun, HP).
- *Định vị kiểu Little-Endian:* byte thấp nhất được đặt trong ô nhớ có địa chỉ thấp nhất (Intel, DEC)

Kiểu định vị	Ví dụ về lệnh	Giải thích
Thanh ghi	Add R3, R4	R3 ← R3 + R4
Tức thì	Add R4, #3	R4 ← R4 + 3
Trực tiếp	Add R1, (1001)	R1 ← R1 + M [1001]
Gián tiếp (thanh ghi)	ADD R4, (R1)	R4 ← R4 + M [R1]
Gián tiếp (bộ nhớ)	Add R1, @ (R3)	R1 ← R1 + M[M [R3]]
Gián tiếp (thanh ghi + Độ dời)	Add R4, 100(R1)	R4 ← R4 + M[R1 + 100]
Gián tiếp (thanh ghi + thanh ghi)	Add R3, (R1 + R2)	R3 ← R3 + M[R1 + R2]
Gián tiếp (t/g nền + t/g chỉ số + độ dời)	Add R1, 100(R2)[R3]	R1←R1+M[100+R2+ d * R3]
Tự tăng	Add R1, (R2)+	R1 ← R1 + M[R2] R2 ← R2 +d
Tự giảm	Add R1, -(R2)	R2 ← R2 - d R1 ← R1 + M[R2]

Bảng II.4 : ***Kiểu định vị của một bộ xử lý có kiến trúc phần mềm kiểu thanh ghi đa dụng.***

R1, R2, R3, R4 : các thanh ghi

R4 ← R3 + R4 : Cộng các thanh ghi R3 và R4 rồi để kết quả và R4

M[R1] : R1 chứa địa chỉ bộ nhớ mà toán hạng được lưu trữ

M[1001] : toán hạng được lưu trữ ở địa chỉ 1001

d : số byte số liệu cần thâm nhập (d = 4 cho từ máy tính, d = 8 cho từ đôi máy tính).

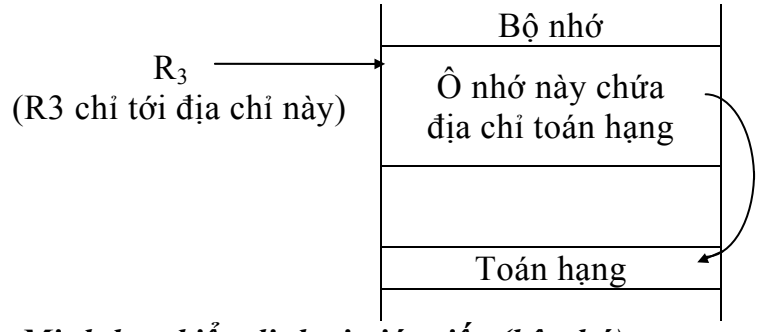
Trong kiểu định vị thanh ghi, các toán hạng đều được chứa trong các thanh ghi.

Trong kiểu định vị tức thì, toán hạng được chứa trong lệnh.

Trong kiểu định vị trực tiếp, địa chỉ của toán hạng được chứa trong lệnh.

Trong kiểu định vị gián tiếp (thanh ghi), địa chỉ toán hạng được chứa trong thanh ghi.

Trong kiểu định vị gián tiếp (bộ nhớ), thanh ghi R3 chứa địa chỉ của địa chỉ của toán hạng như trong hình II.4



Hình II.4: Minh hoạ kiểu định vị gián tiếp (bộ nhớ)

II.7 - KIỂU CỦA TOÁN HẠNG VÀ CHIỀU DÀI CỦA TOÁN HẠNG

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

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

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

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

II.8 - TÁC VỤ MÀ LỆNH THỰC HIỆN

Bảng II.5 cho các loại tác vụ mà một máy tính có thể thực hiện. Trên tất cả máy tính ta đều thấy 3 loại đầu tiên (tính toán số học và luận lý, di chuyển số liệu, chuyển điều khiển). Tuỳ theo kiến trúc của mỗi máy tính, người ta có thể thấy 0 hoặc vài loại tác vụ trong số 5 tác vụ còn lại (hệ thống, tính toán với số có dấu chấm động, tính toán với số thập phân, tính toán trên chuỗi ký tự).

Loại tác vụ	Thí dụ
Tính toán số học và luận lý	Phép tính số nguyên và phép tính luận lý: cộng, trừ, AND, OR
Di chuyển số liệu	Nạp số liệu, lưu giữ số liệu

Chuyển điều khiển	Lệnh nhảy, lệnh vòng lặp, gọi chương trình con và trở về, ngắt quãng
Hệ thống	Gọi hệ điều hành, quản lý bộ nhớ ảo
Tính số có dấu chấm động	Các phép tính trên số có dấu chấm động: cộng, nhân
Tính số thập phân	Các phép tính trên số thập phân: cộng, nhân, đổi từ thập phân sang ký tự
Tính toán trên chuỗi ký tự	Chuyển, so sánh, tìm kiếm chuỗi ký tự
Đồ hoạ và đa phương tiện	Nén và giải nén dữ liệu hình ảnh đồ hoạ (3D) và dữ liệu đa phương tiện (hình ảnh động và âm thanh)

Bảng II.5: Các tác vụ mà lệnh có thể thực hiện

II.9 - KIẾN TRÚC RISC (REDUCED INSTRUCTION SET COMPUTER)

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

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

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

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

Bảng II.6: 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.7 diễn tả ba mẫu máy RISC đầu tiên: mẫu máy của IBM (IBM 801) của Berkeley (RISC1 của Patterson) và của Stanford (MIPS của Hennessy). Ta nhận thấy cả ba máy đó đều có bộ điều khiển bằng mạch điện (không có ô nhớ vi chương trình), có chiều dài các lệnh cố định (32 bits), có một kiểu thi hành lệnh (kiểu thanh ghi - thanh ghi) và chỉ có một số ít lệnh.

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.7 : 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.

II.10 - KIỂU ĐỊNH VỊ TRONG CÁC BỘ XỬ LÝ RISC

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

II.10.1 - Kiểu định vị thanh ghi

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

MIPS	Op code 6	Nguồn 1 5	Nguồn 2 5	Đích 5	Dịch chuyển 5	Hàm 6						
SPARC	Op code 2	Đích 5	Op code 6	Nguồn 1 5	0 1	Khoảng trống khác 8	Nguồn 2 5					
Power PC	Op code 6	Đích 5	Nguồn 1 5	Nguồn 2 5	Op code mở rộng 10	0 1						
ALPHA	Op code 6	Nguồn 1 5	Nguồn 2 5	3	0 1	Op code mở rộng 7	Đích 5					

***Hình II.5** : Dạng lệnh trong kiểu định vị thanh ghi - thanh ghi cho vài CPU RISC*

II.10.2 - Kiểu định vị tức thì

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

MIPS	Op code 6	Thanh ghi nguồn 5	Thanh ghi đích 5	Số có dấu (toán hạng tức thì) 16		
SPARC	Op code 2	Thanh ghi đích 5	Op code 6	Thanh ghi nguồn 5	1 1	Toán hạng tức thì có dấu 13
ALPHA	Op code 6	Thanh ghi nguồn 5	Toán hạng tức thì > 0 8	1 1	Op code mở rộng 7	Thanh ghi đích 5
Power PC	Op code 6	Thanh ghi đích 5	Thanh ghi nguồn 5	Toán hạng tức thì có dấu 16		

***Hình II.6** : Dạng lệnh trong kiểu định vị thanh ghi - tức thì cho vài CPU RISC*

II.10.3 - Kiểu định vị trực tiếp

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

MIPS	Op code 6	Thanh ghi địa chỉ 5	Thanh ghi số liệu 5	Độ dời có dấu 16		
SPARC	Op code 2	Thanh ghi số liệu 5	Op code 6	Thanh ghi địa chỉ 5	1 1	Độ dời có dấu 13
ALPHA	Op code 6	Thanh ghi số liệu 5	Thanh ghi địa chỉ 5	Độ dời có dấu 16		
Power PC	Op code 6	Thanh ghi số liệu 5	Thanh ghi địa chỉ 5	Độ dời có dấu 16		

***Hình II.7** : Dạng lệnh thâm nhập bộ nhớ trong của vài kiến trúc RISC*

II.10.4 - Kiểu định vị gián tiếp bằng thanh ghi + độ dời

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

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

II.10.5 - Kiểu định vị tự tăng

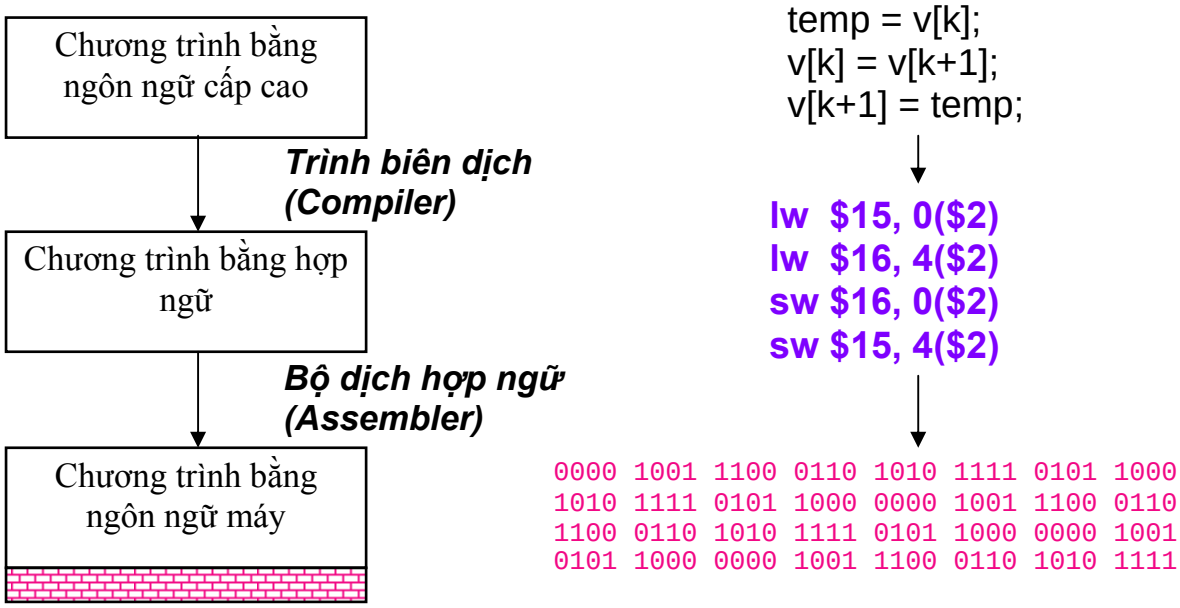
Một vài bộ xử lý RISC, ví dụ bộ xử lý PowerPC, dùng kiểu định vị này.

II.11 - NGÔN NGỮ CẤP CAO VÀ NGÔN NGỮ MÁY

Trong chi phí cho một hệ thống tin học, bao gồm giá tiền của máy tính, giá tiền các phần mềm hệ thống và các phần mềm ứng dụng, thì chi phí cho triển khai phần mềm luôn lớn hơn chi phí mua phần cứng. Vì thế các nhà tin học đã triển khai từ lâu các ngôn ngữ gọi là ngôn ngữ cấp cao. Ngôn ngữ cấp cao dùng các lệnh có cấu trúc gần với ngôn ngữ thông thường hơn ngôn ngữ máy. Các ngôn ngữ cấp cao nổi tiếng là: FORTRAN cho tính toán khoa học, COBOL cho quản lý, LISP và PROLOG dùng trong trí tuệ nhân tạo, PASCAL, C, ADA ... Điểm chính của các ngôn ngữ này là sự cô động và sự độc lập đối với mọi bộ xử lý. Sự độc lập đối với mọi máy tính có nghĩa là có thể được thi hành trên mọi kiến trúc phần mềm của bộ xử lý, với điều kiện là phải có chương trình dịch để dịch chương trình viết bằng ngôn ngữ cấp cao thành chương trình mã máy của máy tính đang sử dụng.

Ở đây, chúng ta không quan tâm đến các đặc tính của ngôn ngữ cấp cao mà chỉ quan tâm đến quan hệ của nó đối với ngôn ngữ máy. Thậy vậy, muốn cho một chương trình ngôn ngữ máy được thực hiện một cách hữu hiệu thì chương trình dịch phải dịch hữu hiệu các lệnh của ngôn ngữ cấp cao thành lệnh mã máy. Muốn thế thì kiến trúc phần mềm của bộ xử lý rất quan trọng đối với chương trình dịch.

Quá trình chuyển đổi từ ngôn ngữ cấp cao sang ngôn ngữ máy: một bộ biên dịch (Compiler) chuyển đổi ngôn ngữ cấp cao (độc lập với kiến trúc phần mềm) sang dạng hợp ngữ (phụ thuộc kiến trúc phần mềm). Một chương trình dịch hợp ngữ (Assembler) chuyển đổi một chương trình viết bằng hợp ngữ (Assembly Language) sang ngôn ngữ máy để máy tính có thể thực hiện được chương trình đó .



***Hình II.10**: Mô tả quá trình chuyển đổi từ ngôn ngữ cấp cao sang ngôn ngữ máy*

Trước đây, kỹ thuật chế tạo các bộ xử lý còn kém, việc quyết định một kiến trúc phần mềm nào đó cho một bộ xử lý nhằm giúp ích cho lập trình bằng hợp ngữ. Người ta đã cố gắng tách kiến trúc phần mềm của bộ xử lý ra khỏi việc thực hiện các chương trình dịch hữu hiệu. Nhưng dần dần, với sự tiến bộ trong công nghệ chế tạo máy tính, người ta bắt đầu nghĩ tới thiết kế các kiến trúc phần mềm làm giảm nhẹ các công việc của chương trình dịch của những ngôn ngữ cấp cao. Trong những năm

1970, người ta đã cố gắng giảm bớt chi phí phát triển phần mềm bằng cách thiết kế các kiến trúc bộ xử lý có những chức năng mà những bộ xử lý trước đó phải dùng một phần mềm để thực hiện. Do vậy các kiến trúc phần mềm mạnh như kiến trúc phần mềm của máy VAX, đã được thực hiện. Máy VAX có nhiều kiểu định vị và một tập lệnh phong phú có thể sử dụng nhiều kiểu dữ liệu. Tuy nhiên, vào đầu những năm 1980, với sự tiến bộ của công nghệ viết các chương trình dịch, người ta đã xem xét lại các kiến trúc phần mềm phức tạp và có chuyển hướng chế tạo các kiến trúc phần mềm đơn giản và hữu hiệu. Chính vì vậy mà các máy tính dùng bộ xử lý kiểu RISC (Reduced Instruction Set Computer) đã ra đời. Với những tiến bộ không ngừng của công nghệ chế tạo máy tính, của công nghệ viết chương trình dịch và của công nghệ lập trình, người ta đang tiến tới chế tạo các kiến trúc phần mềm hấp dẫn hơn trong tương lai.

*****

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG II

*****

1. Các thành phần của một hệ máy tính đơn giản
2. Nhiệm vụ của mỗi bus trong hệ thống bus của một hệ máy tính đơn giản? Tại sao trong thực tế cần có một hệ thống bus vào ra?
3. Mô tả các kiểu thi hành lệnh của một máy tính. Tại sao kiểu thi hành lệnh thanh ghi – thanh ghi được dùng nhiều hiện tại?
4. Mô tả mỗi kiểu định vị trong các kiểu định vị mà một CPU có thể có. Cho CPU RISC, các kiểu định vị nào thường được dùng nhất?
5. Sự khác biệt giữa CPU RISC và CPU CISC?
6. Trong CPU Power PC, giả sử mã tác vụ của lệnh ADD là 011010. Viết lệnh mã máy tương ứng với ADD R1, R19, #-15673

***Chương III*: TỔ CHỨC BỘ XỬ LÝ**

Mục đích: *Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý: đường đi của dữ liệu, bộ điều khiển tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra. Mô tả diễn tiến thi hành một lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật ống dẫn, siêu ống dẫn, siêu vô hướng,...Một số kỹ thuật xử lý thông tin: ống dẫn, siêu ống dẫn, siêu vô hướng, máy tính có lệnh thật dài, máy tính véc-tơ, xử lý song song và kiến trúc IA-64*

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

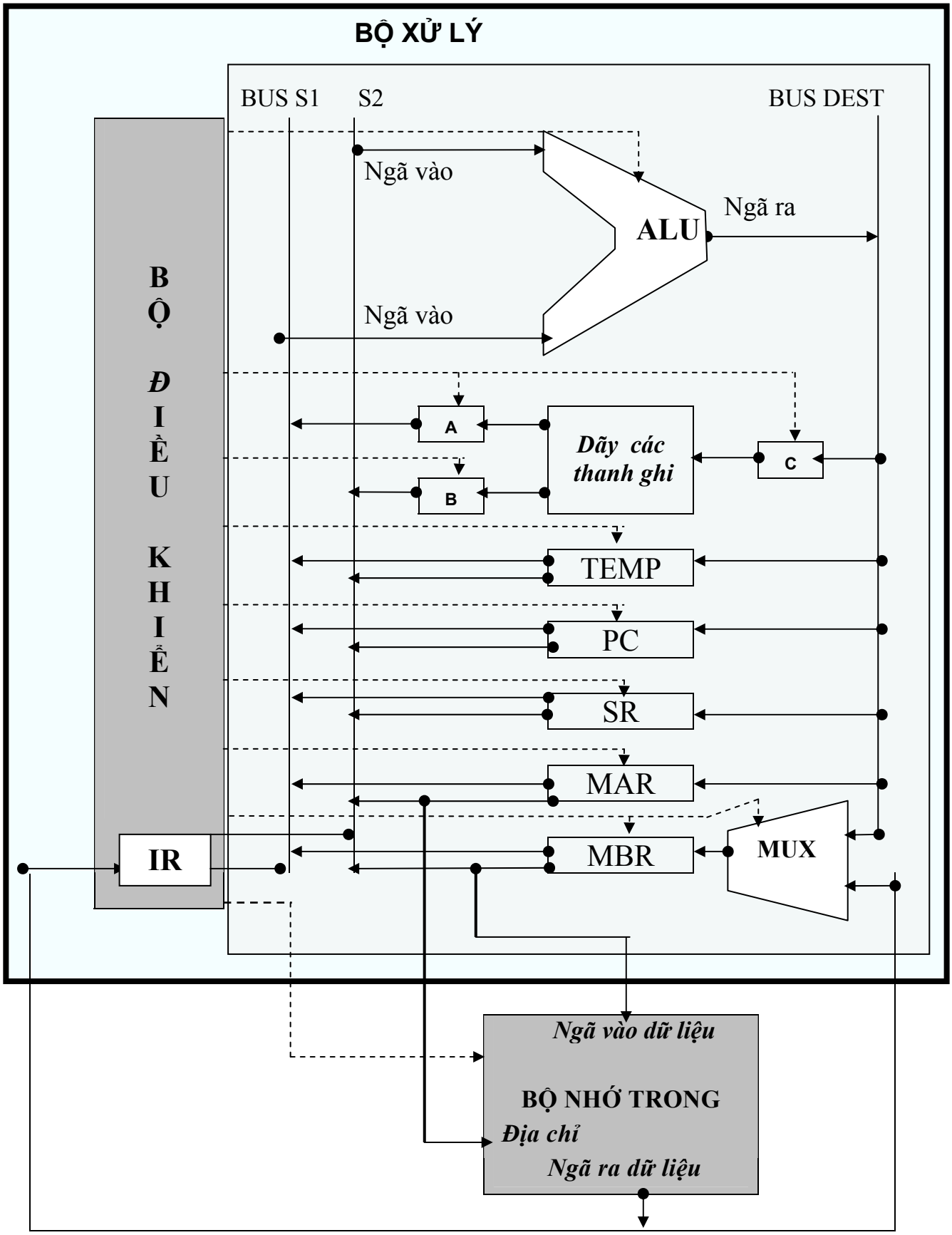
Bộ xử lý được chia chủ yếu thành hai bộ phận: Phần điều khiển và phần đường đi của dữ liệu (data path) như được vẽ trong hình III.1.

III.1. ĐƯỜNG ĐI CỦA 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.1**: Tổ chức của một xử lý điển hình*
(Các đường không liên tục là các đường điều khiển)

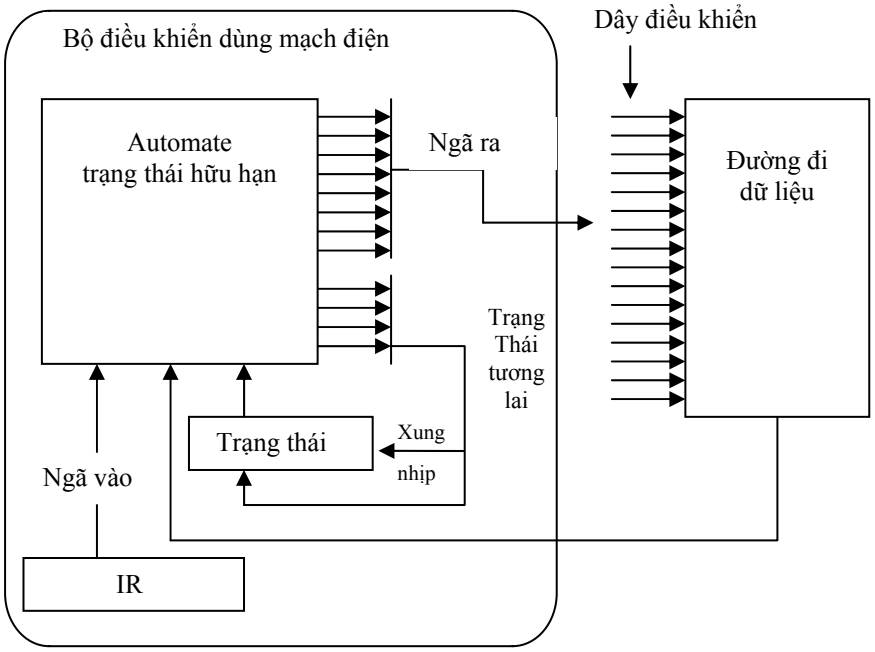
III.2. BỘ ĐIỀU KHIỂN

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).

III.2.1. Bộ điều khiển mạch điện tử

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



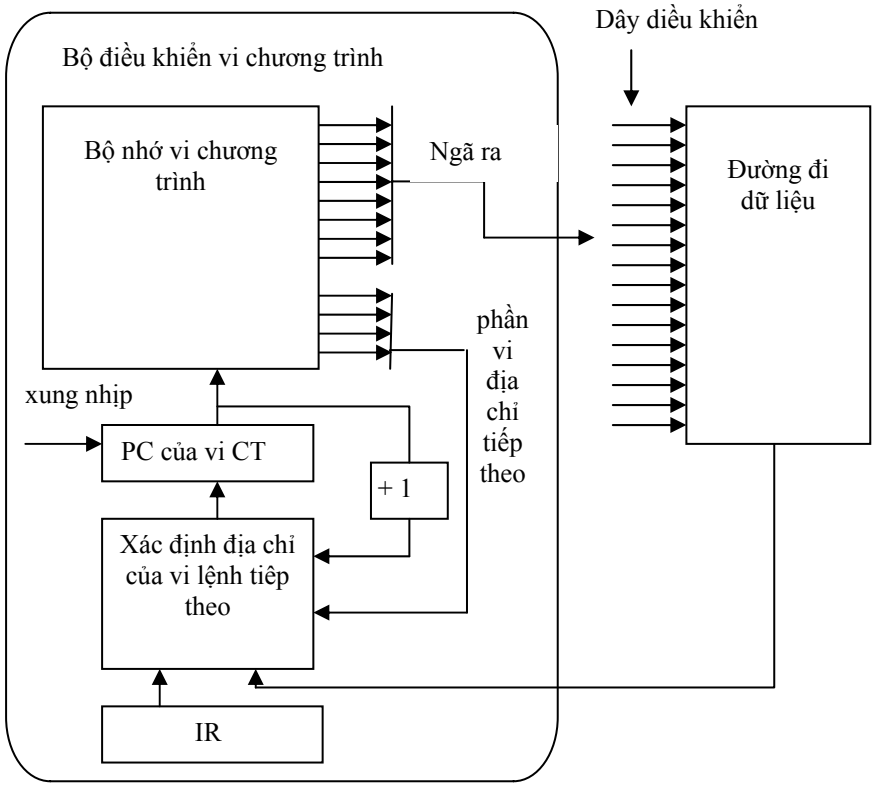
***Hình III.2:** Nguyên tắc vận hành của bộ điều khiển dùng mạch điện*

Hình III.2 cho thấy nguyên tắc của một bộ điều khiển bằng mạch điện. Các đường điều khiển của phần đường đi số liệu là các ngã ra của một hoặc nhiều Automate trạng thái hữu hạn. Các ngã vào của Automate gồm có thanh ghi lệnh, thanh ghi này chứa lệnh phải thi hành và những thông tin từ bộ đường đi số liệu. Ứng với

cấu hình các đường vào và trạng thái hiện tại, Automate sẽ cho trạng thái tương lai và các đường ra tương ứng với trạng thái hiện tại. Automate được cài đặt dưới dạng là một hay nhiều mạch mảng logic lập trình được (PLA: Programmable Logic Array) hoặc các mạch logic ngẫu nhiên.

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

III.2.2. Bộ điều khiển vi chương trình:



***Hình III.3*: Nguyên tắc vận hành của bộ điều khiển vi chương trình**

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

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

III.3. DIỄN TIẾN THI HÀNH LỆNH MÃ MÁY

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

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

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

1. 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.

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

A ← Rs1
B ← Rs2
PC ← PC + 4

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

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

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

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

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

3. 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.

4. 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.

5. Lưu trữ kết quả

Rd ← Ngã ra ALU hoặc Rd ← MBR

Lưu trữ kết quả trong thanh ghi đích.

III.4. NGẮT QUÃNG (INTERRUPT)

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

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

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

- Ngoại vi đòi hỏi nhập hoặc xuất số liệu.
- Người lập trình muốn dùng dịch vụ của hệ điều hành.

- Cho một chương trình chạy từng lệnh.
- Làm điểm dừng của một chương trình.
- Báo tràn số liệu trong tính toán số học.
- Trang bộ nhớ thực sự không có trong bộ nhớ.
- Báo vi phạm vùng cấm của bộ nhớ.
- Báo dùng một lệnh không có trong tập lệnh.
- Báo phần cứng máy tính bị hư.
- Báo điện bị cắt.

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

Để đơn giản việc thiết kế, một vài bộ xử lý chỉ chấp nhận ngắt sau khi thực hiện xong lệnh đang chạy. Khi một ngắt xảy ra, bộ xử lý thi hành các bước sau đây:

1. *Thực hiện xong lệnh đang làm.*
2. *Lưu trữ trạng thái hiện tại.*
3. *Nhảy đến chương trình phục vụ ngắt*
4. *Khi chương trình phục vụ chấm dứt, bộ xử lý khôi phục lại trạng thái cũ của nó và tiếp tục thực hiện chương trình mà nó đang thực hiện khi bị ngắt.*

III.5. KỸ THUẬT ỐNG DẪN (PIPELINE)

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

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

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

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

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

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

III.6. KHÓ KHĂN TRONG KỸ THUẬT ỐNG DẪN

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

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

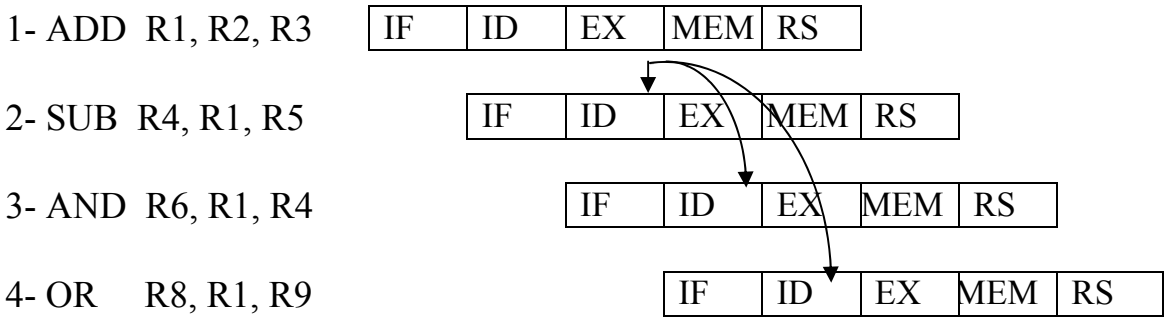
a. Khó khăn do cấu trúc:

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

b. Khó khăn do số liệu:

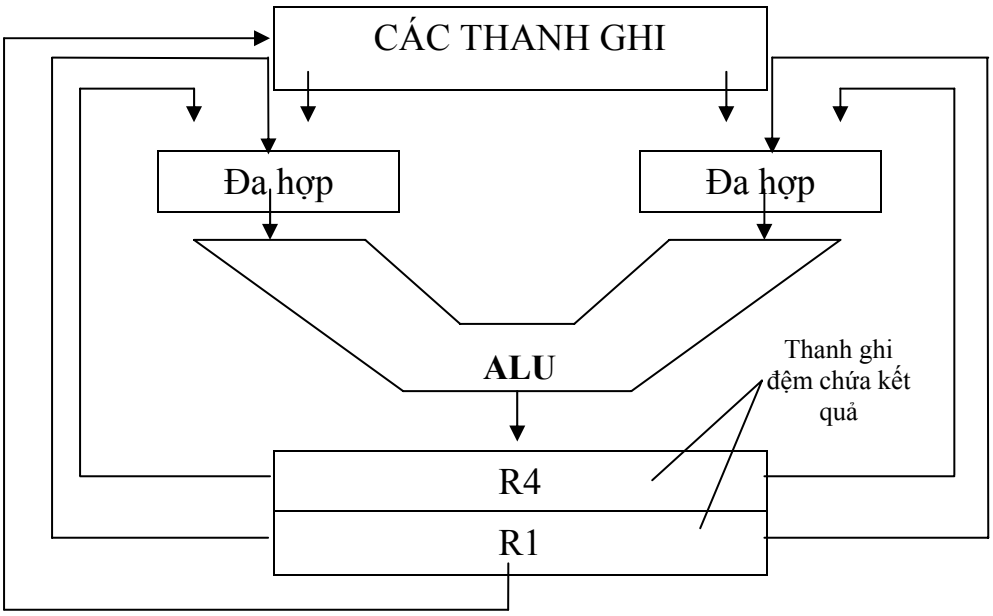
Lấy ví dụ trường hợp các lệnh liên tiếp sau:
Lệnh 1: ***ADD R1, R2, R3***
Lệnh 2: ***SUB R4, R1, R5***
Lệnh 3: ***AND R6, R1, R7***
Lệnh 4: ***OR R8, R1, R9***

Hình 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.



Hình III.5: 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.6.



Hình III.6: 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.

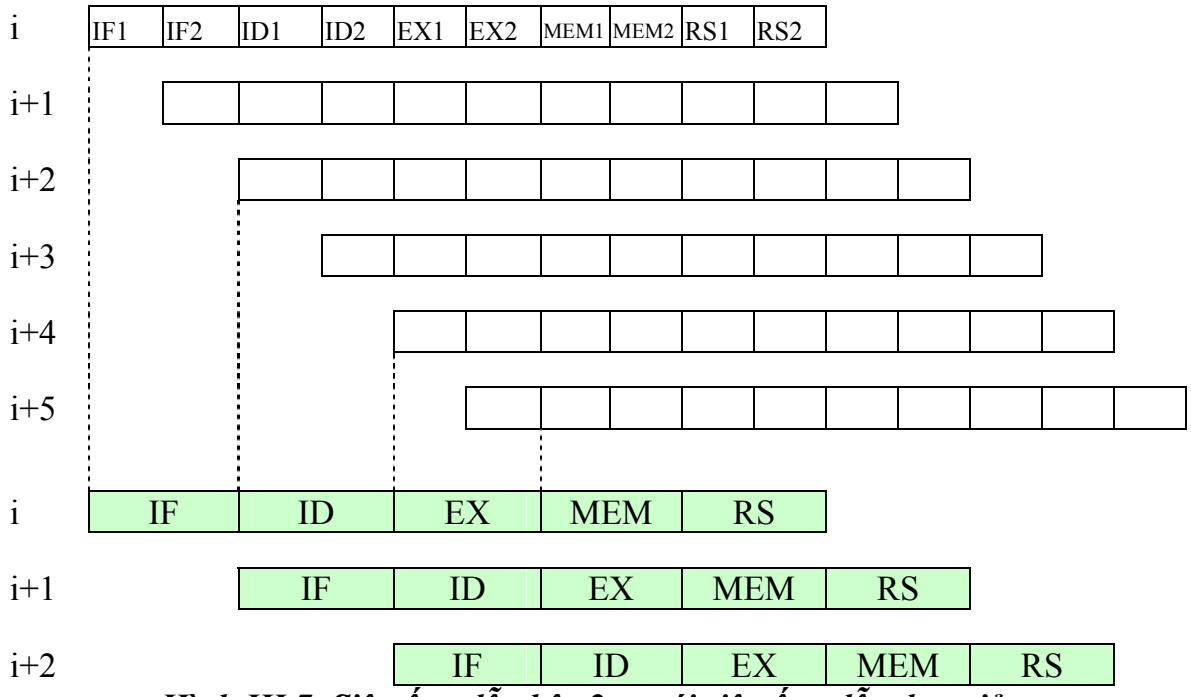
Trong trường hợp nhảy có điều kiện, việc nhảy có thể được thực hiện hay không thực hiện. Lệnh hữu ích đặt sau lệnh nhảy không làm sai lệch chương trình dù điều kiện nhảy đúng hay sai.

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.

III.7. SIÊU ỐNG DẪN

Máy tính có kỹ thuật siêu ống dẫn bậc n bằng cách chia các giai đoạn của kỹ thuật ống dẫn đơn giản, mỗi giai đoạn được thực hiện trong khoản thời gian Tc, thành n giai đoạn con thực hiện trong khoản thời gian Tc/n. Độ hữu hiệu của kỹ thuật này tương đương với việc thi hành n lệnh trong mỗi chu kỳ Tc. Hình III.7 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.7, nếu lệnh thứ i là một lệnh nhảy tương đối thì lệnh này được giải

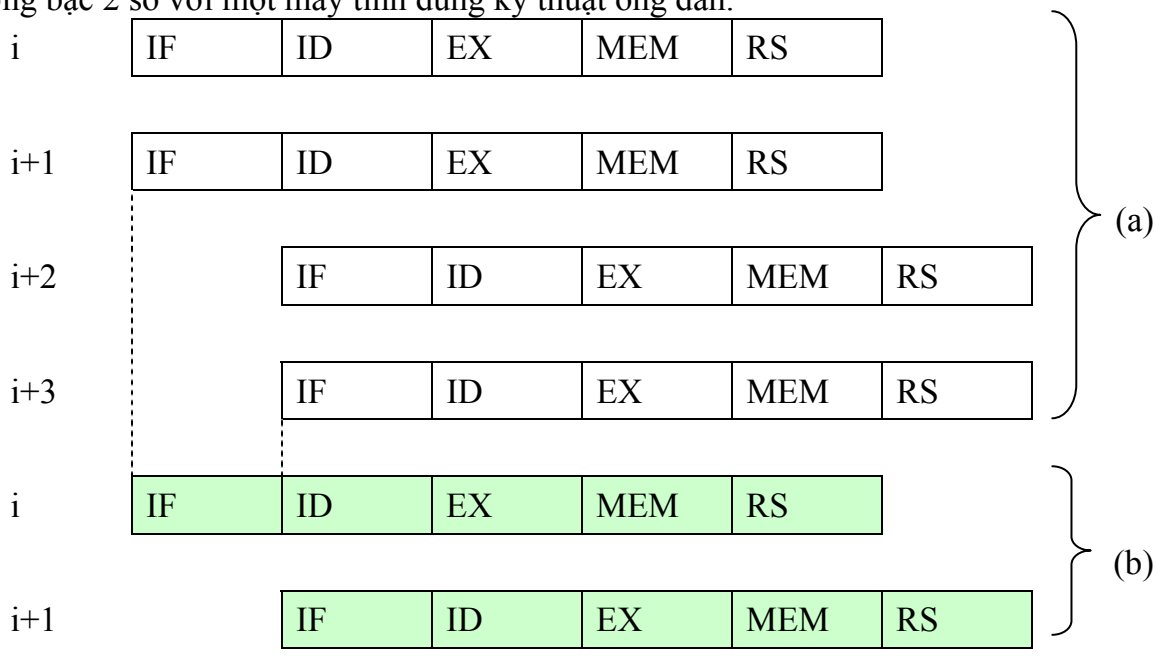
mã trong giai đoạn ID, địa chỉ nhảy đến được tính vào giai đoạn EX, lệnh phải được nhảy tới là lệnh thứ i+4, vậy có trì trệ 3 lệnh thay vì 1 lệnh trong kỹ thuật ống dẫn bình thường.



***Hình III.7:** 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.*

III.8. SIÊU VÔ HƯỚNG (SUPERSCALAR)

Máy tính siêu vô hướng bậc n có thể thực hiện đồng thời n lệnh trong một chu kỳ xung nhịp Tc. Hình III.8 trình bày một ví dụ về sự vận hành của một máy tính siêu vô hướng bậc 2 so với một máy tính dùng kỹ thuật ống dẫn.



***Hình III.8:** Siêu vô hướng (a) so với kỹ thuật ống dẫn (b).*

Trong một máy tính siêu vô hướng phần cứng phải quản lý việc đọc và thi hành đồng thời nhiều lệnh. Vậy nó phải có khả năng quản lý các quan hệ giữa số liệu với nhau. Cũng cần phải chọn các lệnh có khả năng được thi hành cùng một lúc. Những bộ xử lý đầu tiên đưa ra thị trường dùng kỹ thuật này là các bộ xử lý Intel i860 và IBM RS/6000. Các bộ xử lý này có khả năng thực hiện song song nhiều tác vụ trên số nguyên và trên số lẻ.

Năm 1992, người ta thấy xuất hiện các bộ xử lý có nhiều bộ thực hiện tác vụ độc lập với nhau (nhiều ALU, bộ tính toán số lẻ, nạp dữ liệu, lưu dữ liệu, nhảy), có thể thực hiện song song nhiều lệnh (lệnh tính số nguyên, số lẻ, lệnh bộ nhớ, lệnh nhảy...). Số lệnh có thể được thi hành song song càng nhiều thì phần cứng thực hiện việc này càng phức tạp.

III.9. MÁY TÍNH CÓ LỆNH THẬT DÀI VLIW (VERY LONG INSTRUCTION WORD)

Máy tính siêu vô hướng có thể thực hiện 2 hoặc 3 lệnh trong mỗi chu kỳ xung nhịp. Do kỹ thuật ống dẫn đòi hỏi các lệnh phải phụ thuộc vào nhau nên rất khó thực hiện nhiều lệnh trong một chu kỳ. Như vậy, thay vì cố thực hiện nhiều lệnh trong một chu kỳ, người ta tìm cách đưa vào nhiều lệnh trong một từ lệnh dài. Một lệnh VLIW có thể chứa hai tác vụ tính toán số nguyên, hai tác vụ tính toán số lẻ, hai tác vụ thâm nhập bộ nhớ và một lệnh nhảy. Một lệnh như vậy được chia thành nhiều trường, mỗi trường có thể có từ 16 đến 24 bít và chiều dài của lệnh VLIW là từ 112 đến 168 bít. Có nhiều kỹ thuật tạo ra một lệnh VLIW trong đó tất cả các trường đều được dùng. Giá thành và độ phức tạp của một máy tính có lệnh thật dài tăng lên rất nhiều nếu người ta tăng số trường trong một lệnh VLIW.

III.10. MÁY TÍNH VECTƠ

Một máy tính vectơ bao gồm một bộ tính toán vô hướng bình thường dùng kỹ thuật ống dẫn và một bộ làm tính vectơ. Bộ tính toán vô hướng, giống như bộ xử lý dùng kỹ thuật ống dẫn, thực hiện các phép tính vô hướng, còn bộ làm tính vectơ thực hiện các phép tính vectơ. Đa số các máy tính vectơ cho phép làm các phép tính trên vectơ số nguyên, vectơ số lẻ và vectơ số logic (số Boolean).

Có 2 kiểu kiến trúc máy tính vectơ: kiểu *vectơ ô nhớ - ô nhớ* và *kiểu thanh ghi vectơ*. Trong máy tính loại vectơ bộ nhớ - bộ nhớ, các phép tính vectơ được thực hiện trong bộ nhớ. Kiến trúc kiểu thanh ghi vectơ được thực hiện trong các siêu máy tính CRAY - 1, CRAY - 2, X - MP, Y - MP, trong các siêu máy tính của Nhật NEC SX/2, Fujitsu VP200 và Hitachi S820. Các máy này có một bộ nhiều thanh ghi vectơ và những tác vụ vectơ được thực hiện trên các thanh ghi này ngoại trừ các tác vụ nạp dữ liệu và lưu dữ liệu. Máy CRAY-2 (1995) có 8 thanh ghi vectơ, mỗi thanh ghi có thể chứa 64 vectơ, mỗi vectơ có chiều dài 64 bít.

III.11. MÁY TÍNH SONG SONG

Trong các máy tính siêu ống dẫn, siêu vô hướng, máy tính vectơ, máy tính VLIW, người ta đã dùng tính thực hiện song song các lệnh ở các mức độ khác nhau để làm tăng hiệu quả của chúng. Giới hạn về khả năng tính toán của loại máy trên cùng

với sự phát triển của công nghệ máy tính khiến người ta nghĩ tới giải pháp song song theo đó người ta tăng cường hiệu quả của máy tính bằng cách tăng số lượng bộ xử lý.

Các máy tính có thể sắp xếp vào 4 loại sau:

1- **SISD** (Single Instructions Stream, Single Data Stream): Máy tính một dòng lệnh, một dòng số liệu.

2- **SIMD** (Single Instructions Stream, Multiple Data Stream): Máy tính một dòng lệnh, nhiều dòng số liệu.

3- **MISD** (Multiple Instructions Stream, Single Data Stream):Máy tính nhiều dòng lệnh, một dòng số liệu.

4- **MIMD** (Multiple Instruction Stream, Multiple Data Stream): Máy tính nhiều dòng lệnh, nhiều dòng số liệu.

Kiểu phân loại này đơn giản, dễ hiểu, vẫn còn hiệu lực đến hôm nay, mặc dù có những máy tính dùng kiến trúc hỗn tạp.

Các máy tính SISD tương ứng với các máy một bộ xử lý mà chúng ta đã nghiên cứu.

Các máy MISD kiểu máy tính này không sản xuất thương mại.

Các máy SIMD có một số lớn các bộ xử lý giống nhau, cùng thực hiện một lệnh giống nhau để xử lý nhiều dòng dữ liệu khác nhau. Mỗi bộ xử lý có bộ nhớ dữ liệu riêng, nhưng chỉ có một bộ nhớ lệnh và một bộ xử lý điều khiển, bộ này đọc và thi hành các lệnh. Máy CONNECTION MACHINE 2 (65536 bộ xử lý 1 bít) của công ty Thinking Machine Inc, là một ví dụ điển hình của SIMD. Tính song song dùng trong các máy SIMD là tính song song của các dữ liệu. Nó chỉ có hiệu quả nếu cấu trúc các dữ liệu dễ dàng thích ứng với cấu trúc vật lý của các bộ xử lý thành viên. Các bộ xử lý véc-tơ và mảng thuộc loại máy tính này

Các máy MIMD có kiến trúc song song, những năm gần đây, các máy MIMD nổi lên và được xem như một kiến trúc đương nhiên phải chọn cho các máy nhiều bộ xử lý dùng trong các ứng dụng thông thường, một tập hợp các bộ xử lý thực hiện một chuỗi các lệnh khác nhau trên các tập hợp dữ liệu khác nhau. Các máy MIMD hiện tại có thể được xếp vào ba loại hệ thống sẽ được giới thiệu trong phần tiếp theo của chương trình là: *SMP (Symmetric Multiprocesors), Cluster* và *NUMA (Nonunifrom Memory Access)*

a). Một hệ thống SMP bao gồm nhiều bộ xử lý giống nhau được lắp đặt bên trong một máy tính, các bộ xử lý này kết nối với nhau bởi một hệ thống bus bên trong hay một vài sự sắp xếp chuyển mạch thích hợp. Vấn đề lớn nhất trong hệ thống SMP là sự kết hợp các hệ thống cache riêng lẻ. Vì mỗi bộ xử lý trong SMP có một cache riêng của nó, do đó, một khối dữ liệu trong bộ nhớ trong có thể tồn tại trong một hay nhiều cache khác nhau. Nếu một khối dữ liệu trong một cache của một bộ xử lý nào đó bị thay đổi sẽ dẫn đến dữ liệu trong cache của các bộ xử lý còn lại và trong bộ nhớ trong không đồng nhất. Các giao thức cache kết hợp được thiết kế để giải quyết vấn đề này.

b). Trong hệ thống cluster, các máy tính độc lập được kết nối với nhau thông qua một hệ thống kết nối tốc độ cao (mạng tốc độ cao Fast Ethernet hay Gigabit) và hoạt động như một máy tính thống nhất. Mỗi máy trong hệ thống được xem như là một phần của cluster, được gọi là một nút (node). Hệ thống cluster có các ưu điểm:

- Tốc độ cao: Có thể tạo ra một hệ thống cluster có khả năng xử lý mạnh hơn bất cứ một máy tính đơn lẻ nào. Mỗi cluster có thể bao gồm hàng tá máy tính, mỗi máy có nhiều bộ xử lý.

- Khả năng mở rộng cao: có thể nâng cấp, mở rộng một cluster đã được cấu hình và hoạt động ổn định.

- Độ tin cậy cao: Hệ thống vẫn hoạt động ổn định khi có một nút (node) trong hệ thống bị hư hỏng. Trong nhiều hệ thống, khả năng chịu lỗi (fault tolerance) được xử lý tự động bằng phần mềm.

- Chi phí đầu tư thấp: hệ thống cluster có khả năng mạnh hơn một máy tính đơn lẻ mạnh nhất với chi phí thấp hơn.

c). Một hệ thống NUMA (Nonunifrom Memory Access) là hệ thống đa xử lý được giới thiệu trong thời gian gần đây, đây là hệ thống với bộ nhớ chia sẻ, thời gian truy cập các vùng nhớ dành riêng cho các bộ xử lý thì khác nhau. Điều này khác với kiểu quản lý bộ nhớ trong hệ thống SMP (bộ nhớ dùng chung, thời gian truy cập các vùng nhớ khác nhau trong hệ thống cho các bộ xử lý là như nhau). Hệ thống này có những thuận lợi và bất lợi như sau:

Thuận lợi:

- Thực hiện hiệu quả hơn so với hệ thống SMP trong các xử lý song song.
- Không thay đổi phần mềm chính.
- Bộ nhớ có khả năng bị nghẽn nếu có nhiều truy cập đồng thời, nhưng điều này có thể được khắc phục bằng cách:

+ Cache L1&L2 được thiết kế để giảm tối thiểu tất cả các thâm nhập bộ nhớ.

+ Cần các phần mềm cục bộ được quản lý tốt để việc các ứng dụng hoạt động hiệu quả.

+ Quản trị bộ nhớ ảo sẽ chuyển các trang tới các nút cần dùng.

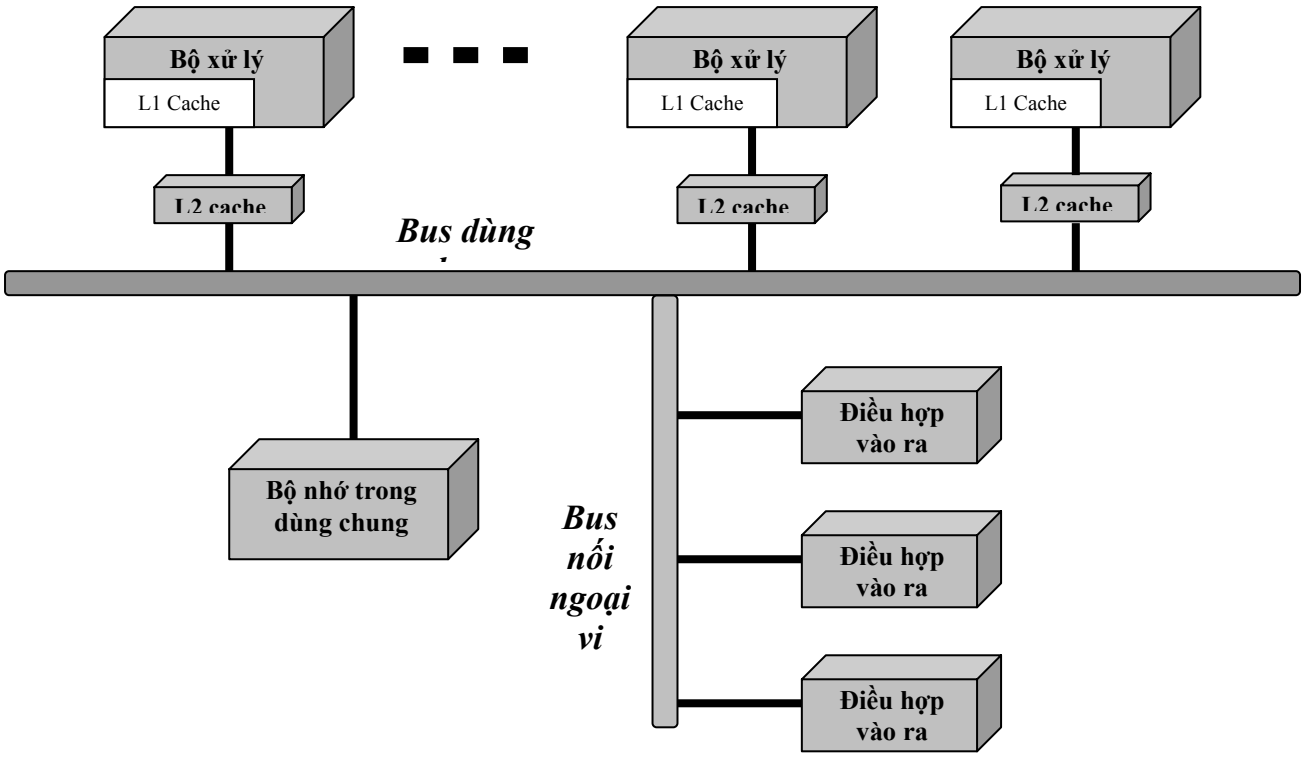
Bất lợi:

- Hệ thống hoạt động không trong suốt như SMP: việc cấp phát các trang, các quá trình có thể được thay đổi bởi các phần mềm hệ thống nếu cần.
- Hệ thống phức tạp.

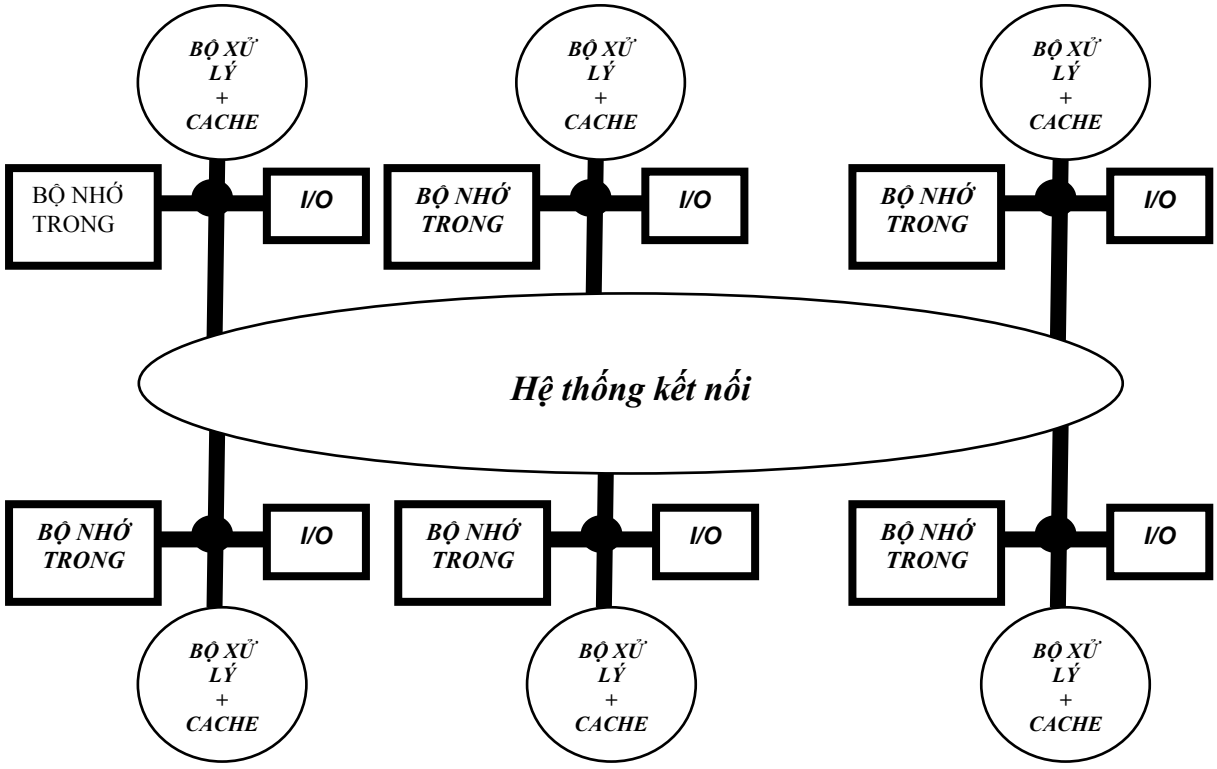
Liên quan đến bộ nhớ trong các máy tính song song, chúng ta có thể chia thành hai nhóm máy:

- Nhóm máy thứ nhất, mà ta gọi là máy có kiến trúc bộ nhớ chia sẻ, có một bộ nhớ trung tâm duy nhất được phân chia cho các bộ xử lý và một hệ thống bus chia sẻ để nối các bộ xử lý và bộ nhớ. Vì chỉ có một bộ nhớ trong nên hệ thống bộ nhớ không đủ khả năng đáp ứng nhu cầu thâm nhập bộ nhớ của một số lớn các bộ xử lý. Kiểu kiến trúc bộ nhớ chia sẻ được dùng trong hệ thống SMP.

Nhóm máy thứ hai bao gồm các máy có bộ nhớ phân tán vật lý. Mỗi máy của nhóm này gồm có các nút, mỗi nút chứa một bộ xử lý, bộ nhớ, một vài ngã vào ra và một giao diện với hệ thống kết nối giữa các nút (hình III.10).



***Hình III.9:** Máy tính song song với bộ nhớ dùng chung, hệ thống bus dùng chung*



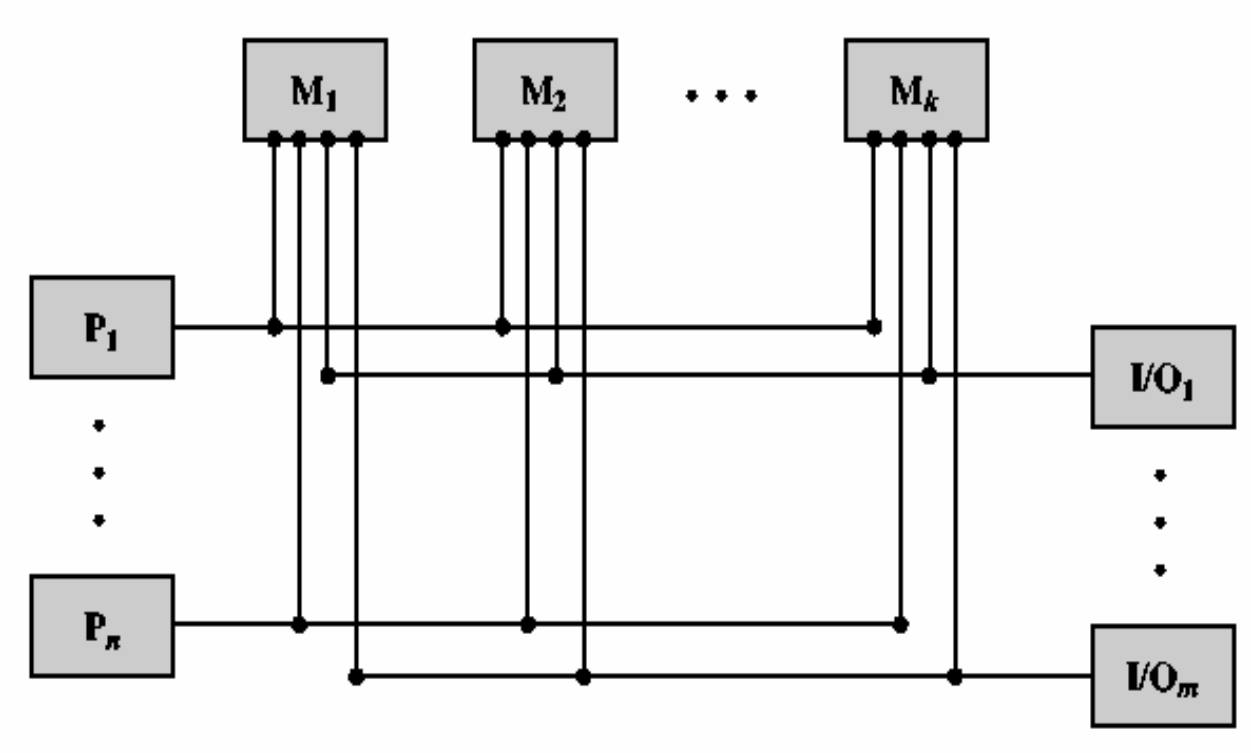
***Hình III.10:** Cấu trúc nền của một bộ nhớ phân tán*

Việc phân tán bộ nhớ cho các nút có hai điểm lợi. *Trước hết*, đây là một cách phân tán việc thâm nhập bộ nhớ. *Thứ hai*, cách này làm giảm thời gian chờ đợi lúc thâm nhập bộ nhớ cục bộ. Các lợi điểm trên làm cho kiến trúc có bộ nhớ phân tán được dùng cho các máy đa xử lý có một số ít bộ xử lý. Điểm bất lợi chính của kiến trúc máy tính này là việc trao đổi dữ liệu giữa các bộ xử lý trở nên phức tạp hơn và mất nhiều thời gian hơn vì các bộ xử lý không cùng chia sẻ một bộ nhớ trong chung. Cách thực hiện việc trao đổi thông tin giữa bộ xử lý và bộ nhớ trong, và kiến trúc logic của bộ nhớ phân tán là một tính chất đặc thù của các máy tính với bộ nhớ phân tán.

Có 2 phương pháp được dùng để truyền dữ liệu giữa các bộ xử lý.

i). *Phương pháp thứ nhất* là các bộ nhớ được phân chia một cách vật lý có thể được thâm nhập với một định vị chia sẻ một cách logic, nghĩa là nếu một bộ xử lý bất kỳ có quyền truy xuất, thì nó có thể truy xuất bất kỳ ô nhớ nào. Trong phương pháp này các máy được gọi có kiến trúc bộ nhớ chia sẻ phân tán (DSM: Distributed Sharing Memory). Từ bộ nhớ chia sẻ cho biết không gian định vị bị chia sẻ. Nghĩa là cùng một địa chỉ vật lý cho 2 bộ xử lý tường ứng với cùng một ô nhớ.

ii). *Phương pháp thứ hai,* không gian định vị bao gồm nhiều không gian định vị nhỏ không giao nhau và có thể được một bộ xử lý thâm nhập. Trong phương pháp này, một địa chỉ vật lý gắn với 2 máy khác nhau thì tương ứng với 2 ô nhớ khác nhau trong 2 bộ nhớ khác nhau. Mỗi mô-đun bộ *xử lý-bộ nhớ* thì cơ bản là một máy tính riêng biệt và các máy này được gọi là đa máy tính. Các máy này có thể gồm nhiều máy tính hoàn toàn riêng biệt và được nối vào nhau thành một mạng cục bộ.



***Hình III.11*: Tổ chức kết nối của máy tính song song có bộ nhớ phân tán**

Kiến trúc song song phát triển mạnh trong thời gian gần đây do các lý do:

- Việc dùng xử lý song song đặc biệt trong lãnh vực tính toán khoa học và công nghệ. Trong các lãnh vực này người ta luôn cần đến máy tính có tính năng cao hơn.

- Người ta đã chấp nhận rằng một trong những cách hiệu quả nhất để chế tạo máy tính có tính năng cao hơn các máy đơn xử lý là chế tạo các máy tính đa xử lý.

- Máy tính đa xử lý rất hiệu quả khi dùng cho đa chương trình. Đa chương trình được dùng chủ yếu cho các máy tính lớn và cho các máy phục vụ lớn.

Các ví dụ về các siêu máy tính dùng kỹ thuật xử lý song song:

- Máy điện toán Blue Gene/L của IBM đang được đặt tại Phòng thí nghiệm Lawrence Livermore, và đứng đầu trong số 500 siêu máy tính mạnh nhất thế giới. Siêu máy tính Blue Gene/L sẽ được sử dụng cho các công việc "phi truyền thống", chủ yếu là giả lập và mô phỏng các quá trình sinh học và nguyên tử. Máy điện toán Blue Gene/L đã đạt tốc độ hơn 70 teraflop (nghìn tỷ phép tính/giây). Kết quả này có thể sẽ đưa cỗ máy lên vị trí dẫn đầu trong danh sách các siêu máy tính nhanh nhất thế giới, được công bố ngày 8/11/2004. Theo đó, siêu máy tính do IBM lắp ráp đã đạt tốc độ 70,72 teraflop trong các cuộc thử nghiệm hồi tháng 10/2004. IBM nghiên cứu và phát triển Blue Gene với mục đích thử nghiệm nhằm tạo ra các hệ thống cực mạnh nhưng chiếm ít không gian và tiêu thụ ít năng lượng. IBM dự kiến, sẽ lắp đặt cho phòng thí nghiệm quốc gia Lawrence Livermore một siêu máy tính có tốc độ nhanh gấp 4 lần so với kỷ lục vừa đạt được. Khi đó, thiết bị sẽ được ứng dụng vào nhiều nghiên cứu khoa học. Hệ thống mới bao gồm 16,384 giao điểm điện toán kết nối *32.768 bộ xử lý*.
- Thông tin mới nhất (02/2005) cho biết: siêu máy tính IBM Blue Gene/L vừa thiết lập kỷ lục mới đó là có khả năng xử lý 135,5 nghìn tỷ phép tính/giây (135,3 teraflop), vượt xa kỷ lục 70,72 teraflop do chính siêu máy tính này lập nên. Số bộ xử lý (BXL) của Blue Gene/L vừa được các nhà khoa học tăng lên gấp đôi (64.000 BXL) nhằm tăng cường khả năng tính toán cho siêu máy tính này. Cũng cần phải nhắc lại rằng thiết kế hoàn thiện của siêu máy tính Blue Gene/L, dự kiến sẽ hoàn tất vào khoảng tháng 6 tới, sẽ bao gồm 130.000 BXL với tốc độ tính toán được kỳ vọng vào khoảng 360 teraflop.

Blue Gene là tên gọi chung cho dự án nghiên cứu siêu máy tính được IBM khởi động từ năm 2000, với mục đích ban đầu là thiết kế một "cỗ máy" có khả năng xử lý 1 teraflop. Trong khi đó, siêu máy tính Blue Gene/L là một trong nhiều sản phẩm chủ lực của IBM nhằm cạnh tranh với các hãng đối thủ Silicon Graphics và NEC.

- Hãng điện tử khổng lồ NEC phát hành một supercomputer dạng vector, máy SX-8 mới ra đời có tốc độ xử lý cực đại lên tới 65 teraflop (65 nghìn tỷ phép tính dấu phẩy động/giây) và khả năng hoạt động ổn định ở mức xấp xỉ 90% của tốc độ 58,5% teraflop. Máy SX-8 có kiến trúc khác hẳn Blue Gene/L của IBM. Nó dùng kiến trúc vector nên đem đến độ ổn định khi hoạt động cao hơn nhiều so với dạng máy tính vô hướng (scalar) như của IBM
- Một hệ thống tại trung tâm nghiên cứu của Cơ quan hàng không vũ trụ Mỹ (NASA) tại California cũng đạt được tốc độ 42,7 teraflop. Với tên gọi Columbia, siêu máy tính này sẽ được sử dụng để nghiên cứu khí tượng và thiết kế máy bay. Hệ thống trị giá 50 triệu USD (thời điểm tháng 10/2004) này sử dụng phần mềm Linux và đã được SGI ký hợp đồng bán cho Cơ quan hàng không vũ trụ Mỹ NASA. Nó có thể thực hiện

42,7 nghìn tỷ phép tính/giây (42,7 teraflop). Tuy nhiên, tốc độ đó chưa phải là tất cả những gì nổi bật của siêu máy tính này: hệ thống mới chỉ khai thác có 4/5 công suất của *10.240 bộ xử lý Intel Itanium 2* trong toàn bộ cỗ máy đặt ở trung tâm nghiên cứu của NASA ở California (Mỹ). Siêu máy tính này không giống với hầu hết các siêu máy tính hiện nay thường được tạo nên theo kiểu cluster, với sự tham gia của nhiều cỗ máy giá rẻ. Columbia được thiết lập từ 20 máy tính mà mỗi chiếc có 512 bộ xử lý, kết nối bằng công nghệ mạng cao tốc và đều chạy một hệ điều hành độc lập. Cách xây dựng này rất hữu ích cho những công việc như giả lập các yếu tố khí động lực cho tàu không gian. Một ứng dụng khác của siêu máy tính Columbia là việc dự báo bão. Phần mềm cho tác vụ này đang được thiết kế và hứa hẹn khả năng dự báo chính xác đường đi của bão sớm 5 ngày. Toàn bộ máy Columbia chiếm dụng một diện tích bằng khoảng 3 sân bóng rổ.

III.12 KIẾN TRÚC IA-64

Kiến trúc IA-64 là một kiến trúc mới được giới thiệu trong những năm gần đây. Kiến trúc này là sản phẩm của sự kết hợp nghiên cứu giữa hai công ty máy tính hàng đầu thế giới là Intel, HP (Hewlett Packard) và một số trường đại học. Kiến trúc mới dựa trên sự phát triển của công nghệ mạch tích hợp và kỹ thuật xử lý song song. Kiến trúc IA-64 giới thiệu một sự khởi đầu mới quan trọng của kỹ thuật siêu vô hướng - kỹ thuật xử lý lệnh song song (EPIC: Expicitly Parallel Intruction Computing) - kỹ thuật ảnh hưởng nhiều đến sự phát triển của bộ xử lý hiện nay. Sản phẩm đầu tiên thuộc kiến trúc này là bộ xử lý ***Itanium***.

a) Đặc trưng của kiến trúc IA-64:

– Cơ chế xử lý song song là song song các lệnh mã máy (EPIC) thay vì các bộ xử lý song song như hệ thống đa bộ xử lý.

– Các lệnh dài hay rất dài (LIW hay VLIW).

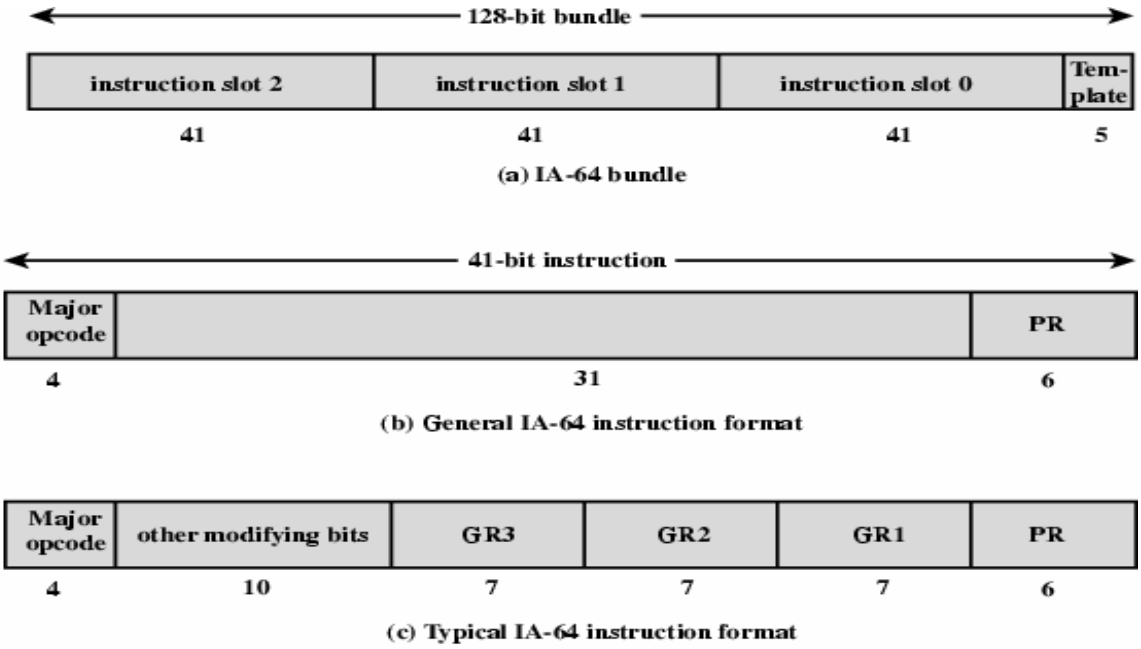
– Các lệnh rẽ nhánh xác định (thay vì đoán các lệnh rẽ nhánh như các kiến trúc trước).

– Nạp trước các lệnh (theo sự suy đoán).

Các đặc trưng của tổ chức của bộ xử lý theo kiến trúc IA-64:

- Có nhiều thanh ghi: số lượng thanh ghi các bộ xử lý kiến trúc IA-64 là 256 thanh ghi. Trong đó, 128 thanh ghi tổng quát (GR) 64 bit cho các tính toán số nguyên, luận lý; 128 thanh ghi 82 bit (FR) cho các phép tính dấu chấm động và dữ liệu đồ hoạ; ngoài ra, còn có 64 thanh ghi thuộc tính (PR)1 bit để chỉ ra các thuộc tính lệnh đang thi hành.
- Nhiều bộ thi hành lệnh: hiện nay, một máy tính có thể có tám hay nhiều hơn các bộ thi hành lệnh song song. Các bộ thi hành lệnh này được chia thành bốn kiểu:
  + Kiểu I (I-Unit): dùng xử lý các lệnh tính toán số nguyên, dịch, luận lý, so sánh, đa phương tiện.
  + Kiểu M (M-Unit): Nạp và lưu trữ giữa thanh ghi và bộ nhớ thêm vào một vài tác vụ ALU.
  + Kiểu B (B-Unit): Thực hiện các lệnh rẽ nhánh.
  + Kiểu F (F-Unit): Các lệnh tính toán số dấu chấm động

b) Định dạng lệnh trong kiến trúc IA-64



PR: Predicate register
GR: General hay Floating-point

Hình III.12: Định dạng lệnh trong kiến trúc IA-64

Kiến trúc IA-64 định nghĩa một *gói* (buldle) 128 bit chứa ba lệnh (mỗi lệnh dài 41 bit) và một trường mẫu (template field) 5 bit. Bộ xử lý có thể lấy một hay nhiều *gói lệnh* thi hành cùng lúc. Trường mẫu (template field) này chứa các thông tin chỉ ra các lệnh có thể thực hiện song song (Bảng III.1). Các lệnh trong một bó có thể là các lệnh độc lập nhau. Bộ biên dịch sẽ sắp xếp lại các lệnh trong các gói lệnh kề nhau theo một thứ tự để các lệnh có thể được thực hiện song song

Hình III.12a chỉ ra định dạng lệnh trong kiến trúc IA-64. Hình III.12b mô tả dạng tổng quát của một lệnh trong gói lệnh. Trong một lệnh, mã lệnh chỉ có 4 bit chỉ ra 16 khả năng có thể để thi thi hành một lệnh và 6 bit chỉ ra thanh ghi thuộc tính được dùng với lệnh. Tuy nhiên, các mã tác vụ này còn tuỳ thuộc vào vị trí của lệnh bên trong gói lệnh, vì vậy khả năng thi hành của lệnh nhiều hơn số mã tác vụ được chỉ ra. Hình III.12c mô tả chi tiết các trường trong một lệnh (41 bit)

Trong bảng III.1 , các kiểu L-Unit, X-Unit là các kiểu mở rộng, có thể thực hiện lệnh bởi I-Unit hay B-Unit.

Template	***Slot 0***	***Slot 1***	***Slot 2***
00	M-Unit	I-Unit	I-Unit
01	M-Unit	I-Unit	I-Unit
02	M-Unit	I-Unit	I-Unit
03	M-Unit	I-Unit	I-Unit
04	M-Unit	L-Unit	X-Unit
05	M-Unit	L-Unit	X-Unit
08	M-Unit	M-Unit	I-Unit
09	M-Unit	M-Unit	I-Unit
0A	M-Unit	M-Unit	I-Unit

0B	M-Unit	M-Unit	I-Unit
0C	M-Unit	F-Unit	I-Unit
0D	M-Unit	F-Unit	I-Unit
0E	M-Unit	M-Unit	F-Unit
0F	M-Unit	M-Unit	F-Unit
10	M-Unit	I-Unit	B-Unit
11	M-Unit	I-Unit	B-Unit
12	M-Unit	B-Unit	B-Unit
13	M-Unit	B-Unit	B-Unit
16	B-Unit	B-Unit	B-Unit
17	B-Unit	B-Unit	B-Unit
18	M-Unit	M-Unit	B-Unit
19	M-Unit	M-Unit	B-Unit
1C	M-Unit	F-Unit	B-Unit
1D	M-Unit	F-Unit	B-Unit

***Bảng III.1*: Bảng mã hoá tập hợp các ánh xạ trong trường mẫu.**

*****

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG III

*****

1. Các thành phần và nhiệm vụ của đường đi dữ liệu?
2. Mô tả đường đi dữ liệu ứng với các lệnh sau:
    i. ADD R1,R2,R3
    ii. SUB R1, R2, (R3)
    iii. ADD R1, R5, #100
    iv. JMP R1 (Nhảy đến ô nhớ mà R1 trỏ tới)
    v. BRA +5 (Nhảy bỏ 5 lệnh)
3. Thế nào là ngắt quãng? Các giai đoạn thực hiện ngắt quãng của CPU.
4. Vẽ hình để mô tả kỹ thuật ống dẫn. Kỹ thuật ống dẫn làm tăng tốc độ CPU lên bao nhiêu lần (theo lý thuyết)? Tại sao trên thực tế sự gia tăng này lại ít hơn?
5. Các điều kiện mà một CPU cần phải có để tối ưu hoá kỹ thuật ống dẫn. Giải thích từng điều kiện.
6. Các khó khăn trong kỹ thuật ống dẫn và cách giải quyết khó khăn này.
7. Thế nào là máy tính vectơ? Các kiểu của kiến trúc vectơ?
8. Cho ví dụ về máy tính một dòng lệnh, nhiều dòng số liệu (SIMD)
9. Các máy tính song song nhiều dòng lệnh, nhiều dòng số liệu (MIMD) dùng nhiều bộ xử lý, được phân thành 2 loại tuỳ theo tổ chức bộ nhớ của chúng là: máy tính đa xử lý có bộ nhớ tập trung chia sẻ và máy tính đa xử lý có bộ nhớ phân tán. Phân tích ưu - khuyết điểm của hai loại máy tính này.
10. Các loại hệ thống MIMD.
11. Các đặc trưng của kiến trúc IA-64? Định dạng lệnh trong kiến trúc IA-64?

Chương IV: CÁC CẤP BỘ NHỚ

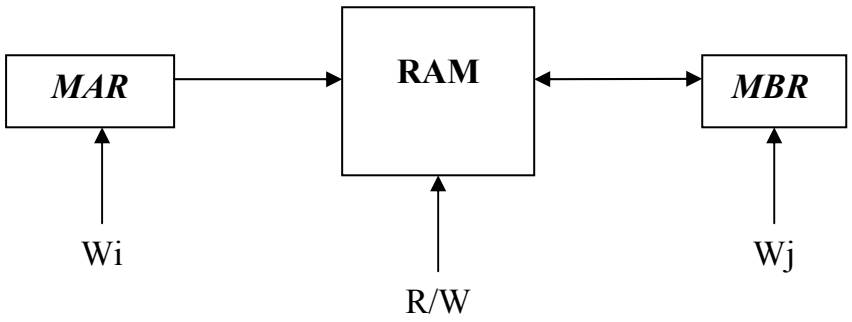
Mục đích: *Chương này giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache: nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên lý vận hành của bộ nhớ ảo.*

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

IV.1. CÁC 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 đưa ra địa chỉ ô nhớ đến lúc đọc được nội dung ô nhớ đó) và chu kỳ bộ nhớ (thời gian giữa hai lần liên tiếp thâm nhập bộ nhớ).



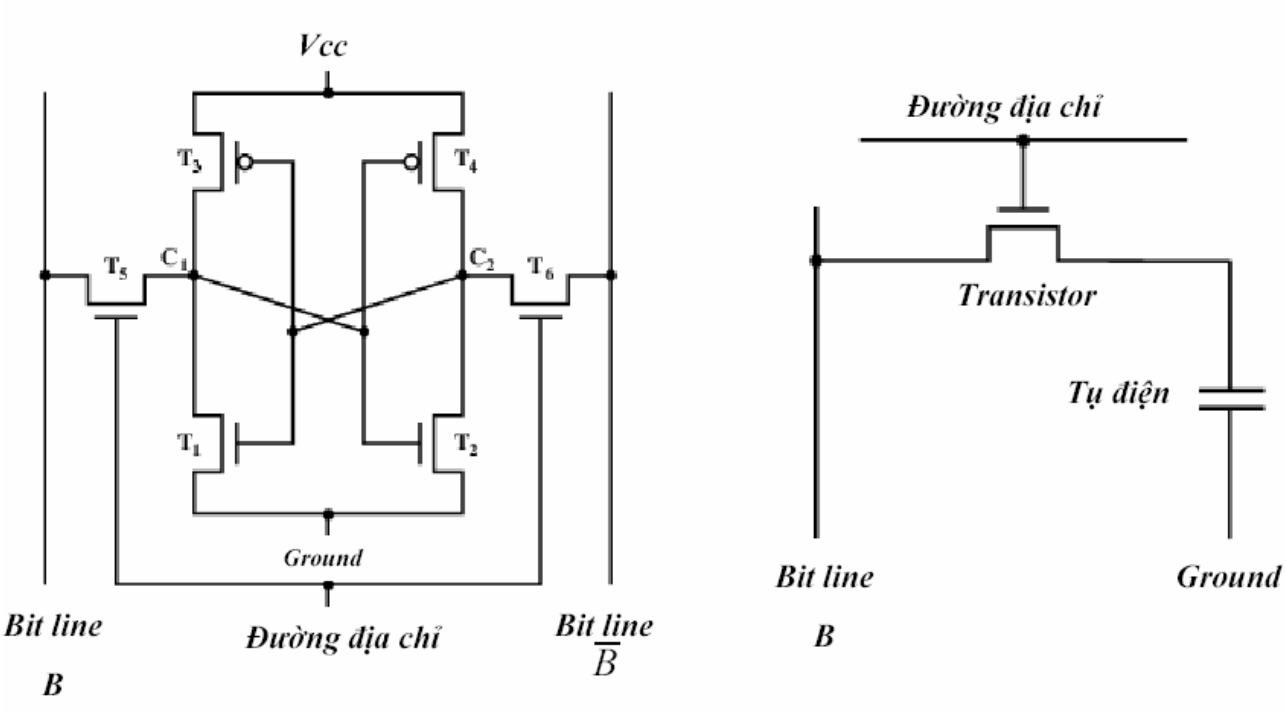
Hình IV.1: ***Vận hành của bộ nhớ RAM***
(Wi, Wj, R/W là các tín hiệu điều khiển)

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 75ns-150ns).

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

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

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

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.

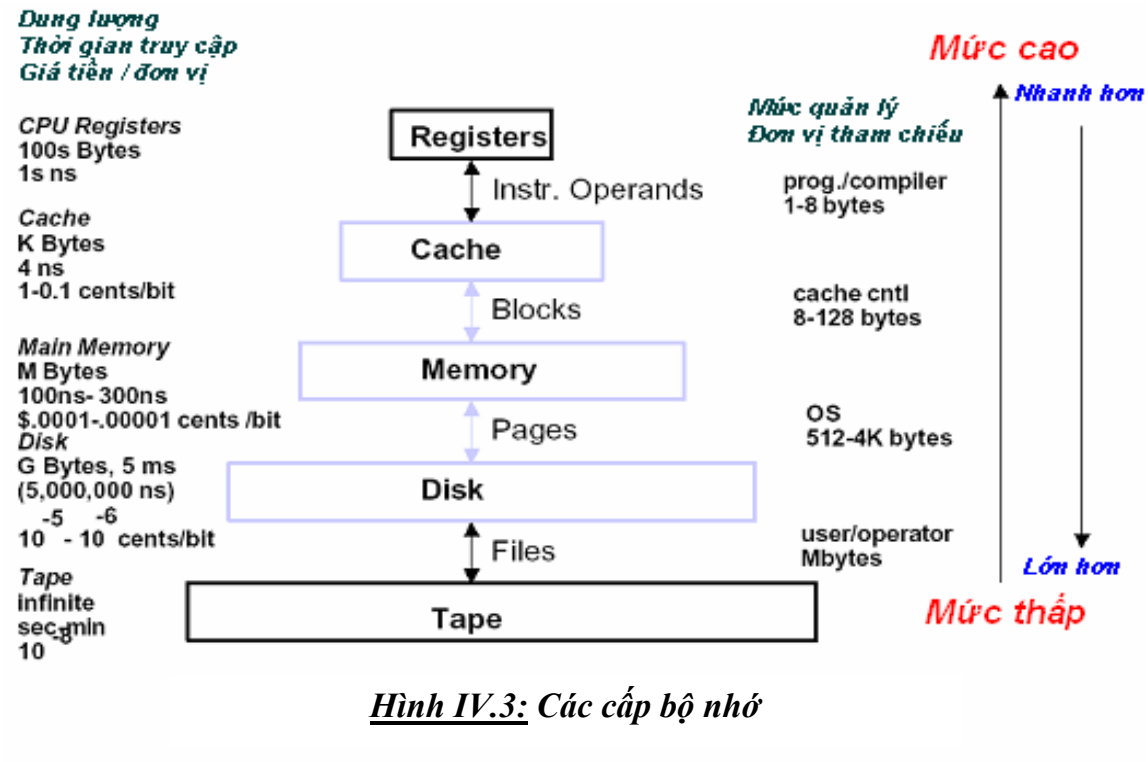
EEPROM (Eletrically Erasable Programable ROM): Chế tạo bằng công nghệ bán dẫn. Chương trình nằm trong ROM có thể được viết vào và có thể xóa (bằng điện) để viết lại bởi người sử dụng.

<table>
<tr><th>Kiểu bộ nhớ</th><th>Loại</th><th>Cơ chế xoá</th><th>Cơ chế ghi</th><th>Tính bay hơi</th></tr>
<tr><td>RAM</td><td>đọc/ghi</td><td>bằng điện, mức byte</td><td>bằng điện</td><td>Có</td></tr>
<tr><td>ROM</td><td rowspan="2">chỉ đọc</td><td rowspan="2">Không thể xoá</td><td>Mặt nạ</td><td rowspan="5">Không</td></tr>
<tr><td>Programmable ROM (PROM)</td><td rowspan="4">bằng điện</td></tr>
<tr><td>Erasable PROM</td><td rowspan="3">hầu hết chỉ đọc</td><td>Tia cực tím, mức chip</td></tr>
<tr><td>Electrically Erasable PROM (EEPROM)</td><td>bằng điện, mức byte</td></tr>
<tr><td>Flash Memory</td><td>bằng điện, mức khối</td></tr>
</table>

***Bảng IV.1:** Các kiểu bộ nhớ bán dẫn*

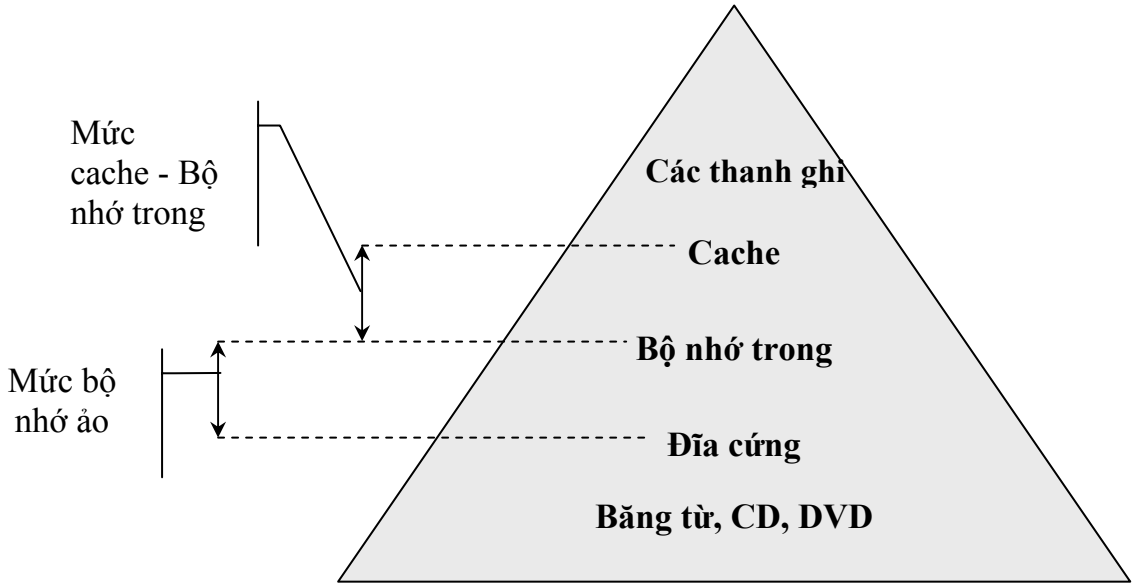
IV.2. CÁC CẤP BỘ NHỚ

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 thì nhanh nhưng đắt tiền hơn cấp có dung lượng cao hơn. Mục tiêu của việc thiết lập các cấp bộ nhớ là người dùng có một hệ thống bộ nhớ rẻ tiền như cấp bộ nhớ thấp nhất và gần nhanh như cấp bộ nhớ cao nhất. Các cấp bộ nhớ thường được lồng vào nhau. Mọi dữ liệu trong một cấp thì được gặp lại trong cấp thấp hơn và có thể tiếp tục gặp lại trong cấp thấp nhất.

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

IV.3. XÁC SUẤT TRUY CẬP DỮ LIỆU TRONG BỘ NHỚ TRONG

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

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

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

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

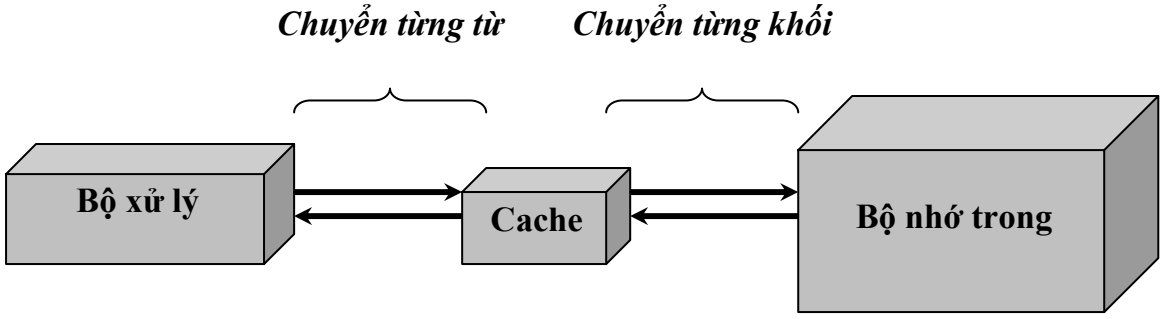
xử lý các trường hợp ngoại lệ. Còn số liệu luôn có cấu trúc và thông thường chỉ có một phần số liệu được thâm nhập nhiều nhất mà thôi.

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

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

IV.4. VẬN HÀNH CỦA CACHE

Mức cache -bộ nhớ trong trong bảng các cấp bộ nhớ có cơ cấu vận hành trong suốt đối với bộ xử lý. Với thao tác đọc bộ nhớ, bộ xử lý gởi một địa chỉ và nhận một dữ liệu từ bộ nhớ trong. Với thao tác ghi bộ nhớ, bộ xử lý viết một dữ liệu vào một ô nhớ với một địa chỉ được chỉ ra trong bộ nhớ. Để cho chương trình vận hành bình thường thì cache phải chứa một phần con của bộ nhớ trong để bộ xử lý có thể thâm nhập vào các lệnh hoặc dữ liệu thường dùng từ bộ nhớ cache. Do dung lượng của bộ nhớ cache nhỏ nên nó chỉ chứa một phần chương trình nằm trong bộ nhớ trong. Để đảm bảo sự đồng nhất giữa nội dung của cache và bộ nhớ trong thì cache và bộ nhớ trong phải có cùng cấu trúc. Việc chuyển dữ liệu giữa cache và bộ nhớ trong là việc tải lên hay ghi xuống các *khối dữ liệu*. Mỗi khối chứa nhiều từ bộ nhớ tuỳ thuộc vào cấu trúc bộ nhớ cache. Sự lựa chọn kích thước của khối rất quan trọng cho vận hành của cache có hiệu quả.



***Hình IV.5:** Trao đổi dữ liệu giữa các thành phần CPU-Cache-Bộ nhớ trong*

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

- **Thành công cache (cache hit):** bộ xử lý tìm gặp phần tử cần đọc (ghi) trong cache.
- **Thất bại cache (cache miss):** bộ xử lý không gặp phần tử cần đọc (ghi) trong cache.
- **Trừng phạt thất bại cache (cache penalty):** Thời gian cần thiết để xử lý một thất bại cache. Thời gian bao gồm thời gian thâm nhập bộ nhớ trong cộng với thời gian chuyển khối chứa từ cần đọc từ bộ nhớ trong đến cache. Thời gian này tuỳ thuộc vào kích thước của khối.

Để hiểu được cách vận hành của cache, ta lần lượt xem xét và trả lời bốn câu hỏi liên quan đến các tình huống khác nhau xảy ra trong bộ nhớ trong.

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

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

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

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

Trả lời câu hỏi 1: ***Phải để một khối bộ nhớ vào chỗ nào của cache (sắp xếp khối)?***

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

- *Kiểu tương ứng trực tiếp*: Nếu mỗi khối bộ nhớ chỉ có một vị trí đặt khối duy nhất trong cache được xác định theo công thức: **K= i mod n**

Trong đó:

K: vị trí khối đặt trong cache

i: số thứ tự của khối trong bộ nhớ trong

n: số khối của cache

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

Ví dụ:

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

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

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

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

Trong đó:

K: vị trí khối đặt trong cache

i: số thứ tự của khối trong bộ nhớ trong

s: số lượng tập hợp trong cache.

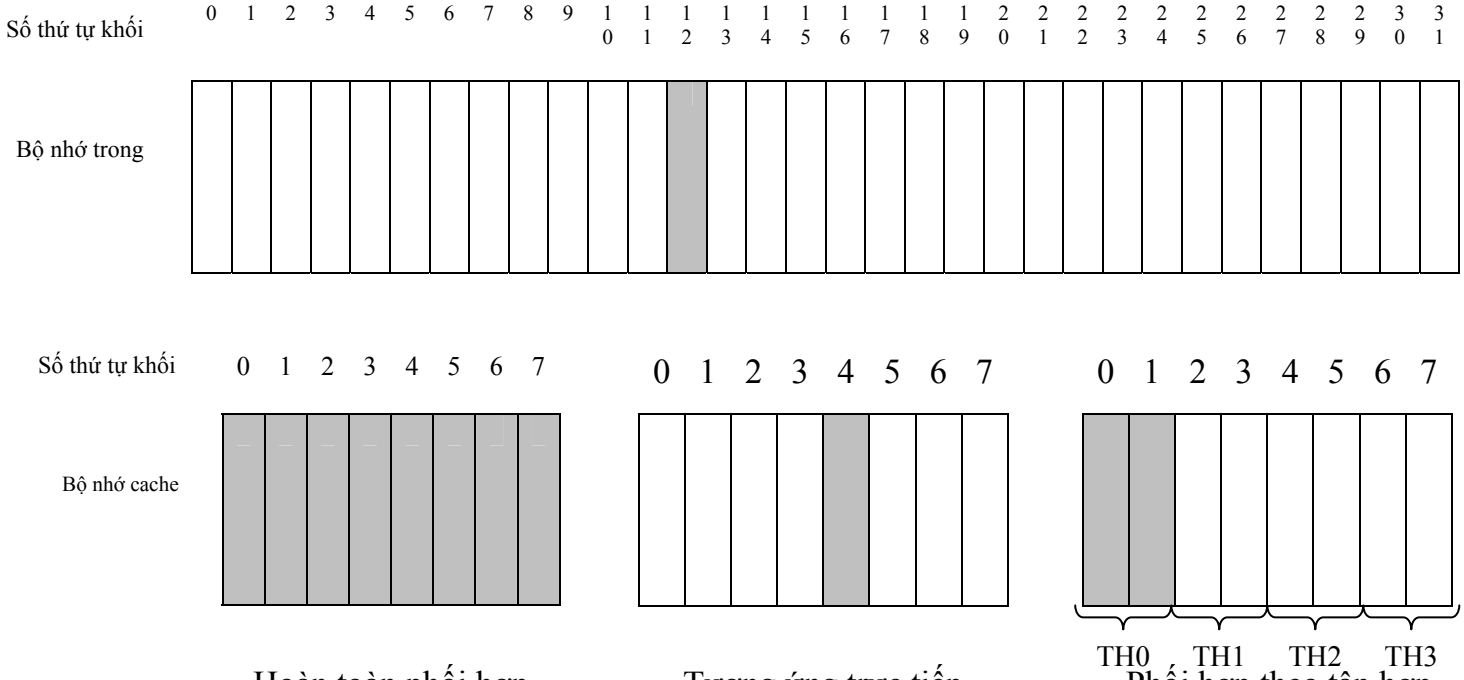
Trong cách đặt khối theo kiểu phối hợp theo tập hợp, nếu tập hợp có ***m*** khối, sự tương ứng giữa các khối trong bộ nhớ trong và các khối của cache được gọi là phối hợp theo tập hợp ***m*** khối.

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

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

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

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



Trả lời câu hỏi 2: ***Làm sao để tìm một khối khi nó hiện diện trong cache (nhận diện khối)?***

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

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

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

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

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

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

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

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

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

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

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

Chỉ số khối	Địa chỉ từ cần đọc trong khối

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

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

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

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

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

- Đối với kiểu tương ứng trực tiếp:

10 bit

Chỉ số khối bộ nhớ (5 bit)		Địa chỉ từ cần đọc trong khối (5 bit)
Nhãn (2 bit)	Chỉ số khối cache (3 bit)	

- Đối với kiểu hoàn toàn phối hợp:

10 bit

Chỉ số khối bộ nhớ (Nhãn) (5 bit)	Địa chỉ từ cần đọc trong khối (5 bit)

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

10 bit

Chỉ số khối bộ nhớ (5 bit)		Địa chỉ từ cần đọc trong khối (5 bit)
Nhãn (3 bit)	Chỉ số tập hợp (2 bit)	

Trả lời câu hỏi 3: ***Khối nào phải được thay thế trong trường hợp thất bại cache (thay thế khối)?***

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

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

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

Trả lời câu hỏi 4: ***Việc gì xảy ra khi ghi vào bộ nhớ (chiến thuật ghi)?***

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

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

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

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

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

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

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

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

- Mỗi bộ điều khiển cache sẽ theo dõi các thao tác ghi vào bộ nhớ từ các bộ phận khác. Nếu thao tác ghi vào phần bộ nhớ chia sẻ được ánh xạ vào cache của nó quản lý, bộ điều khiển cache sẽ vô hiệu hoá sự thâm nhập này. Chiến lược này phụ thuộc vào cách ghi đồng thời trên tất cả các bộ điều khiển cache.
- Một vi mạch được dùng để điều khiển việc cập nhật, một thao tác ghi vào bộ nhớ từ một cache nào đó sẽ được cập nhật trên các cache khác.
- Một vùng nhớ chia sẻ cho một hay nhiều bộ xử lý thì không được ánh xạ lên cache. Như vậy, tất cả các thâm nhập vào vùng nhớ chia sẻ này đều bị thất bại cache.

IV.5. HIỆU QUẢ CỦA CACHE

Thông thường người ta dùng thời gian thâm nhập trung bình bộ nhớ trong để đánh giá hiệu quả của cache.

Thời gian thâm nhập trung bình được cho bởi công thức:

$$\begin{pmatrix}\text{Thời gian thâm nhập}\\ \text{trung bình của hệ}\end{pmatrix} = \begin{pmatrix}\text{Thời gian thâm}\\ \text{nhập thành công}\end{pmatrix} + \begin{pmatrix}\text{Tỉ lệ}\\ \text{thất bại}\end{pmatrix} * \begin{pmatrix}\text{Trừng phạt}\\ \text{thất bại}\end{pmatrix}$$

Thời gian thâm nhập thành công là thời gian thâm nhập vào một thông tin trong một thành công cache. Tỉ số thất bại là tỉ số giữa số thất bại cache và tổng số thâm nhập cache. Thời gian thâm nhập thành công và trừng phạt thất bại được đo bằng đơn vị thời gian hoặc bằng chu kỳ xung nhịp (clock cycle).

Trong việc tìm kiếm thông tin trong cache phải chú ý làm giảm tỉ lệ thất bại mà các nguyên nhân chính là như sau:

- *Khởi động:* trong lần thâm nhập cache đầu tiên, không có thông tin cần tìm trong cache nên phải chuyển khối chứa thông tin đó vào cache.

- *Khả năng*: vì cache không thể chứa tất cả các khối cần thiết cho việc thi hành một chương trình nên gặp thất bại do cache thiếu khả năng, do đó một khối bị lấy ra khỏi cache rồi lại được đưa vào sau này.

- *Tranh chấp*: Nếu chiến thuật thay thế các khối là phối hợp theo tập hợp hay tương ứng trực tiếp, các thất bại do tranh chấp xảy ra vì một khối có thể bị đưa ra khỏi cache rồi được gọi vào sau đó nếu có nhiều khối phải được thay thế trong các tập hợp.

Ba nguyên nhân trên cho ta ý niệm về nguyên nhân thất bại, nhưng mô hình đơn giản trên có những hạn chế của nó. Mô hình này giúp ta thấy một số liệu trung bình nhưng chưa giải thích được từng thất bại một. Ví dụ, nếu tăng kích thước cache thì giảm thất bại do tranh chấp và thất bại do khả năng vì cache càng lớn thì nhiều khối có thể được đưa vào. Tuy nhiên, một thất bại có thể đi từ thất bại do khả năng đến thất bại do tranh chấp khi kích thước của cache thay đổi. Khi nêu ba nguyên nhân trên ta đã không lưu ý đến cách thức thay thế các khối. Cách thức này có thể dẫn đến những vận hành bất thường như là tỉ lệ thất bại cao lên khi độ phối hợp lớn lên.

IV.6. CACHE DUY NHẤT HAY CACHE RIÊNG LẺ

Cache duy nhất chứa đồng thời lệnh và dữ liệu.

Cache riêng lẻ phân biệt cache lệnh và cache dữ liệu.

Giải pháp sau có lợi là tránh các khó khăn do kiến trúc, khi thi hành các lệnh dùng kỹ thuật ống dẫn.

Với một cache duy nhất, sẽ có tranh chấp khi một lệnh muốn thâm nhập một số liệu trong cùng một chu kỳ của giai đoạn đọc một lệnh khác. Cache riêng lẻ còn giúp tối ưu hoá mỗi loại cache về mặt kích thước tổng quát, kích thước các khối và độ phối hợp các khối.

IV.7. CÁC MỨC CACHE

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

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

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

- Ngoài ra, trong một số hệ thống (PowerPC G4, IBM S/390 G4, Itanium của Intel) còn có tổ chức *cache mức ba* (L3 cache), đây là mức cache trung gian giữa cache L2 và một thẻ bộ nhớ.

Bộ xử lý	Kiểu	Năm phát hành	L1 Cache [a]	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 KB	-	-
PDP-11/70	Mini Computer	1975	1 KB	-	-
VAX 11/780	Mini Computer	1978	16 KB	-	-
IBM 3033	Mainframe	1978	64 KB	-	-
IBM 3090	Mainframe	1985	128 to 256 KB	-	-
Intel 80486	PC	1989	8 KB	-	-
Pentium	PC	1993	8 KB / 8 KB	256 to 512 KB	-
PowerPC 601	PC	1993	32 KB	-	-
PowerPC 620	PC	1996	32 KB / 32 KB	-	-
PowerPC G4	PC/Server	1999	32 KB / 32 KB	256KB to 1MB	2 MB
IBM S390/G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S390/G6	Mainframe	1999	256 KB	8 MB	-
Pentium 4	PC/Server	2000	8 KB / 8 KB	256 KB	-
IBM SP	High-End server/ Super Computer	2000	64 KB / 32 KB	8 MB	-
CRAY MTA [b]	Super Computer	2000	8 KB	2 MB	-
Itanium	PC/Server	2001	16 KB / 16 KB	96 KB	2 MB
SGI Origin 2001	High-End server	2001	32 KB / 32 KB	4 MB	-

[a] Hai giá trị cách nhau bởi dấu "/" chỉ giá trị cache lệnh và cache dữ liệu
[b] Cả hai giá trị đều là cache lệnh

Bảng IV.2: Kích thước cache của một số hệ thống

IV.8. BỘ NHỚ TRONG

Bộ nhớ trong thoả mãn các yêu cầu của cache và được dùng làm đệm vào ra vì bộ nhớ trong vừa là nơi chứa các thông tin từ ngoài đưa vào, vừa là nơi xuất ra các thông tin cho cache. Việc đo hiệu quả của bộ nhớ trong dựa vào thời gian thâm nhập và bề rộng dãi thông. Thông thường thời gian thâm nhập bộ nhớ trong là phần tử quan trọng cho cache trong lúc dãi thông bộ nhớ là phần chính cho các tác vụ xuất nhập. Với việc dùng phổ biến các cache ngoài, dãi thông của bộ nhớ trong cũng trở thành quan trọng cho cache.

Mặc dù cache cần bộ nhớ trong có thời gian thâm nhập nhỏ, nhưng thường thì dễ cải thiện dãi thông bộ nhớ nhờ nhiều cách tổ chức bộ nhớ mới, hơn là giảm thời gian thâm nhập cho cache. Cache thụ hưởng các tiến bộ về dãi thông bằng cách tăng kích thước của mỗi khối của cache mà không tăng đáng kể trừng phạt thất bại cache.

Người ta dùng các kỹ thuật sau đây để nới rộng dãi thông của bộ nhớ trong:

– *Nới rộng chiều dài ô nhớ trong*. Đây là kỹ thuật đơn giản để tăng giải thông bộ nhớ. Thông thường cache và bộ nhớ trong có chiều rộng ô nhớ là chiều rộng 1 từ vì bộ xử lý thâm nhập vào một từ ô nhớ. Nhân đôi, nhân bốn chiều rộng ô nhớ của cache và bộ nhớ trong làm lưu lượng thâm nhập bộ nhớ trong được nhân đôi hay nhân bốn. Vậy cũng phải chi tiêu thêm để nới rộng bus bộ nhớ (là bus nối bộ xử lý với bộ nhớ).

Một ví dụ bộ xử lý có chiều dài ô nhớ trong lớn là bộ xử lý ALPHA AXP 21064 (Hãng DEC). Cache ngoài, bộ nhớ trong và bus bộ nhớ đều có độ rộng là 256 bit.

– *Bộ nhớ đan chéo đơn giản:* các IC bộ nhớ có thể được tổ chức thành dãi để đọc hay viết nhiều từ cùng một lúc thay vì chỉ đọc một từ, độ rộng của bus và của cache không thay đổi. Khi gởi nhiều địa chỉ đến nhiều dãi thì ta đọc được nhiều từ cùng một lúc. Bộ nhớ đan chéo cũng cho phép ghi vào bộ nhớ nhiều từ cùng một lúc. Tổ chức bộ nhớ đan chéo đơn giản không rắc rối nhiều so với tổ chức bình thường của bộ nhớ trong vì các dãi có thể dùng chung các đường địa chỉ với bộ điều khiển ô nhớ, và như thế mỗi dãi có thể dùng phần số liệu của bus bộ nhớ. SDRAM và DDR SDRAM là các loại RAM dùng kỹ thuật này

– *Bộ nhớ đan chéo tổ chức thành dãi độc lập:* một tổ chức bộ nhớ đan chéo hiệu quả hơn, là cho phép nhiều thâm thập bộ nhớ và như thế cho phép các dãi làm việc độc lập với nhau. Mỗi dãi cần có các đường địa chỉ riêng biệt và đôi khi cần bus số liệu riêng biệt: Trong trường hợp này bộ xử lý có thể tiếp tục công việc của mình trong lúc chờ đợi số liệu (trường hợp thất bại cache). RDRAM là bộ nhớ loại này

– *Tránh xung đột giữa các dãi bộ nhớ*. Trong các máy tính đa xử lý và máy tính vectơ, hệ thống bộ nhớ được thiết kế nhằm cho phép nhiều yêu cầu thâm nhập độc lập nhau. Sự hiệu quả của hệ thống tuỳ thuộc vào tần số các trường hợp có yêu cầu độc lập thâm nhập vào các dãi khác nhau. Với sự đan chéo bình thường (hình IV.6), các thâm nhập tuần tự hoặc tất cả các thâm nhập vào các địa chỉ cách biệt nhau một số chẵn, thì vận hành tốt nhưng sẽ gặp rắc rối nếu sự cách biệt giữa các địa chỉ là một số lẻ. Một biện pháp mà các máy tính lớn dùng là làm giảm bớt các trường hợp xung đột tĩnh bằng cách tăng số lượng các dãi. Thí dụ, máy NEC SX/3 chia bộ nhớ trong ra 128 dãi.

Địa chỉ	Dãi 0	Địa chỉ	Dãi 1	Địa chỉ	Dãi 2	Địa chỉ	Dãi 3
0		1		2		3	
4		5		6		7	
8		9		10		11	
12		13		14		15	

Hình IV.6: Bộ nhớ đan chéo bậc 4.

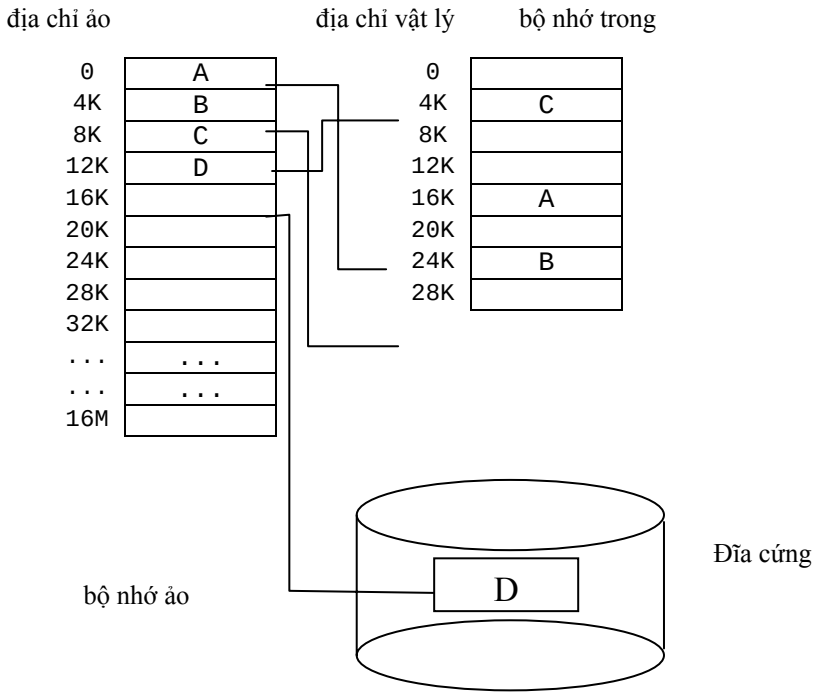
Dãi thứ i chứa tất cả các từ có địa chỉ thoả mãn công thức **(địa chỉ) mod 4 = i**

IV.9. BỘ NHỚ ẢO

Bộ nhớ ảo xác định một cơ chế vận chuyển tự động số liệu giữa bộ nhớ trong và bộ nhớ ngoài (đĩa từ).

Trước đây, khi độ dài của chương trình vượt quá giới hạn dung lượng bộ nhớ thì người lập trình phải phân chia chương trình của mình thành từng phần tự loại bỏ nhau (overlays) và phải tự quản lý việc trao đổi thông tin giữa bộ nhớ và đĩa từ. Bộ nhớ ảo làm nhẹ trách nhiệm của các nhà lập trình bằng cách làm cho việc trao đổi thông tin này được thực hiện một cách tự động.

Trong các bộ xử lý hiện đại, bộ nhớ ảo được dùng để cho phép thực hiện cùng lúc nhiều *tiến trình* (process), mỗi tiến trình có một không gian định vị riêng. Nếu tất cả các không gian định vị này đều thuộc không gian định vị bộ nhớ trong thì rất tốn kém. Bộ nhớ ảo bao gồm bộ nhớ trong và bộ nhớ ngoài được phân tích thành khối để có thể cung cấp cho mỗi chương trình một số khối cần thiết cho việc thực hiện chương trình đó. Hình IV.7 cho thấy một chương trình chứa trong bộ nhớ ảo gồm 4 khối, 3 trong 4 khối nằm ở bộ nhớ trong, khối thứ tư nằm trên đĩa.



***Hình IV.7.* Một chương trình gồm 4 trang A,B,C,D trong đó trang D nằm trong ổ đĩa**

Ngoài việc phân chia không gian bộ nhớ, cần bảo vệ và quản lý tự động các cấp bộ nhớ, bộ nhớ ảo đơn giản hoá việc nạp chương trình vào bộ nhớ để thi hành nhờ một cơ chế được gọi là *sự tái định địa chỉ* (address relocation). Cơ chế này cho phép một chương trình có thể được thi hành khi nó nằm ở bất cứ vị trí nào trong bộ nhớ.

Tham số	Cache	Bộ nhớ ảo
Chiều dài mỗi khối (trang)	16 - 128 byte	4096 - 65536 bytes
Thời gian thâm nhập thành công	1 - 2 xung nhịp	40 - 100 xung nhịp
Trừng phạt khi thất bại (Thời gian thâm nhập) (Di chuyển số liệu)	8 - 100 xung nhịp 6 - 60 xung 2 - 40 xung	700.000 - 6 triệu xung 500.000 - 4 triệu xung 200.000 - 2 triệu xung
Tỉ số thất bại	0,5% - 10%	0,00001% - 0,001%
Dung lượng	8 KB – 8MB	16 MB – 8GB

***Bảng IV.3*: Đại lượng điển hình cho bộ nhớ cache và bộ nhớ ảo. So với bộ nhớ cache thì các tham số của bộ nhớ ảo tăng từ 10 đến 100.000 lần**

Ngoài sự khác biệt định lượng mà ta thấy trong hình IV.9, có những khác biệt khác giữa bộ nhớ cache và bộ nhớ ảo là:

- Khi thất bại cache, sự thay thế một khối trong cache được điều khiển bằng phần cứng, trong khi sự thay thế trong bộ nhớ ảo là chủ yếu do hệ điều hành.

- Không gian định vị mà bộ xử lý quản lý là không gian định vị của bộ nhớ ảo, trong lúc đó thì dung lượng bộ nhớ cache không tuỳ thuộc vào không gian định vị bộ xử lý.

- Bộ nhớ ngoài còn được dùng để lưu trữ tập tin ngoài nhiệm vụ là hậu phương của bộ nhớ trong (trong các cấp bộ nhớ).

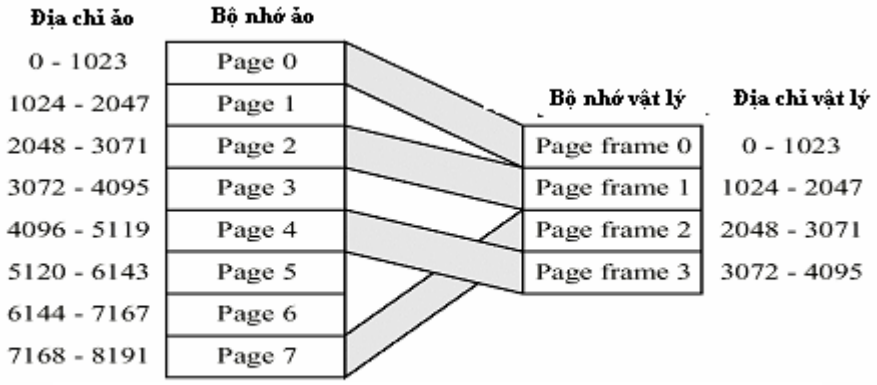
Bộ nhớ ảo cũng được thiết kế bằng nhiều kỹ thuật đặc thù cho chính nó.

Các hệ thống bộ nhớ ảo có thể được chia thành 2 loại: loại với khối có dung lượng cố định gọi là trang, và loại với khối có chiều dài thay đổi gọi là đoạn. Định vị trang xác định một địa chỉ trong trang, giống như định vị trong cache. Trong định vị đoạn cần 2 từ: một từ chứa số thứ tự đoạn và một từ chứa độ dời trong đoạn. Chương trình dịch gặp khó khăn nhiều hơn trong định vị đoạn.

Do việc thay thế các đoạn, ngày nay ít máy tính dùng định vị đoạn thuần tuý. Một vài máy dùng cách hỗn hợp gọi là đoạn trang. Trong đó mỗi đoạn chứa một số nguyên các trang.Bây giờ chúng ta trả lời 4 câu hỏi đặt ra trong các cấp bộ nhớ cho bộ nhớ ảo.

Câu hỏi 1: ***Một khối được đặt tại đâu trong bộ nhớ trong?***

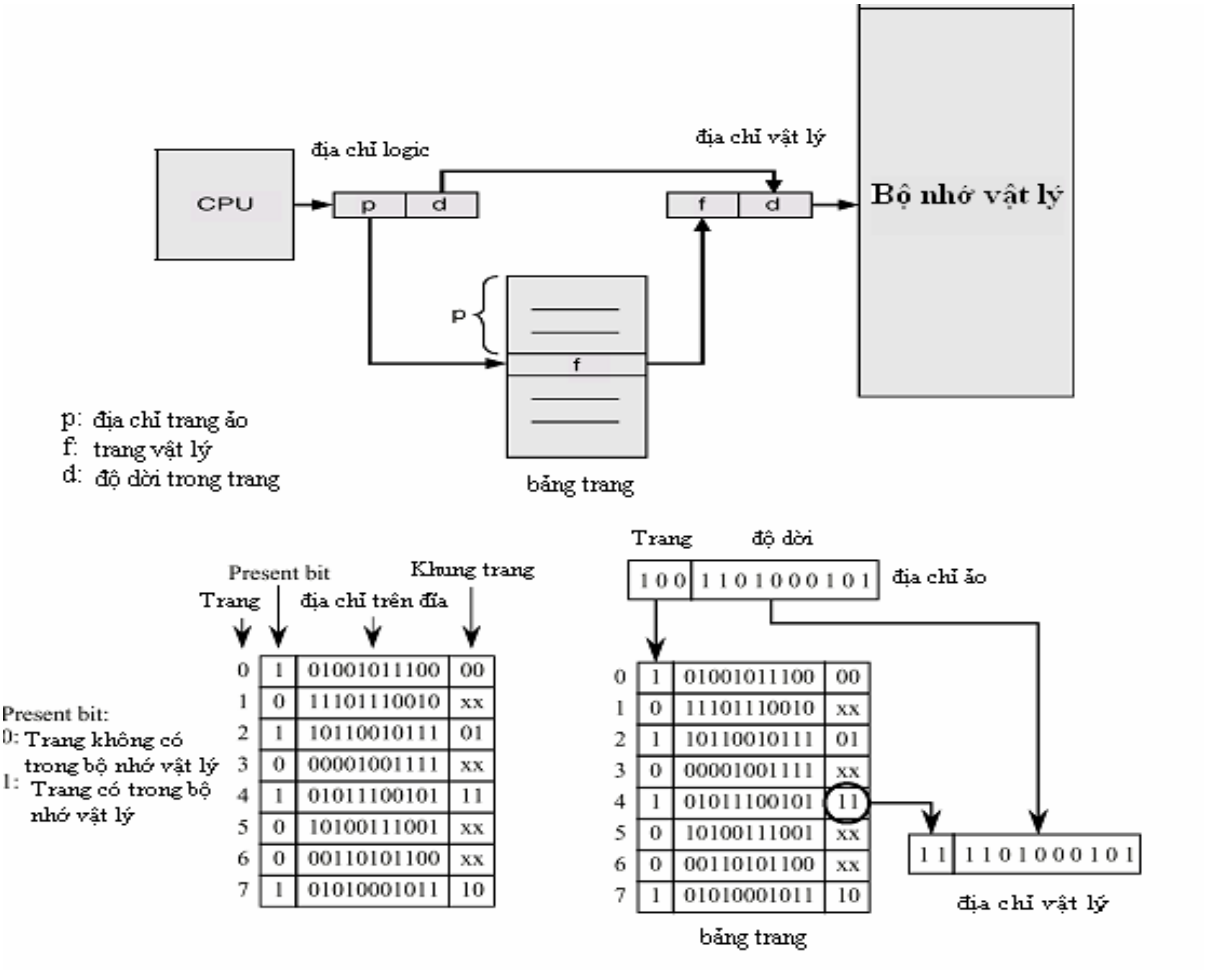
Việc trừng phạt bộ nhớ ảo khi có thất bại, tương ứng với việc phải thâm nhập vào ổ đĩa. Việc thâm nhập này rất chậm nên người ta chọn phương án hoàn toàn phối hợp trong đó các khối (trang) có thể nằm ở bất kỳ vị trí nào trong bộ nhớ trong. Cách này cho tỉ lệ thất bại thấp.



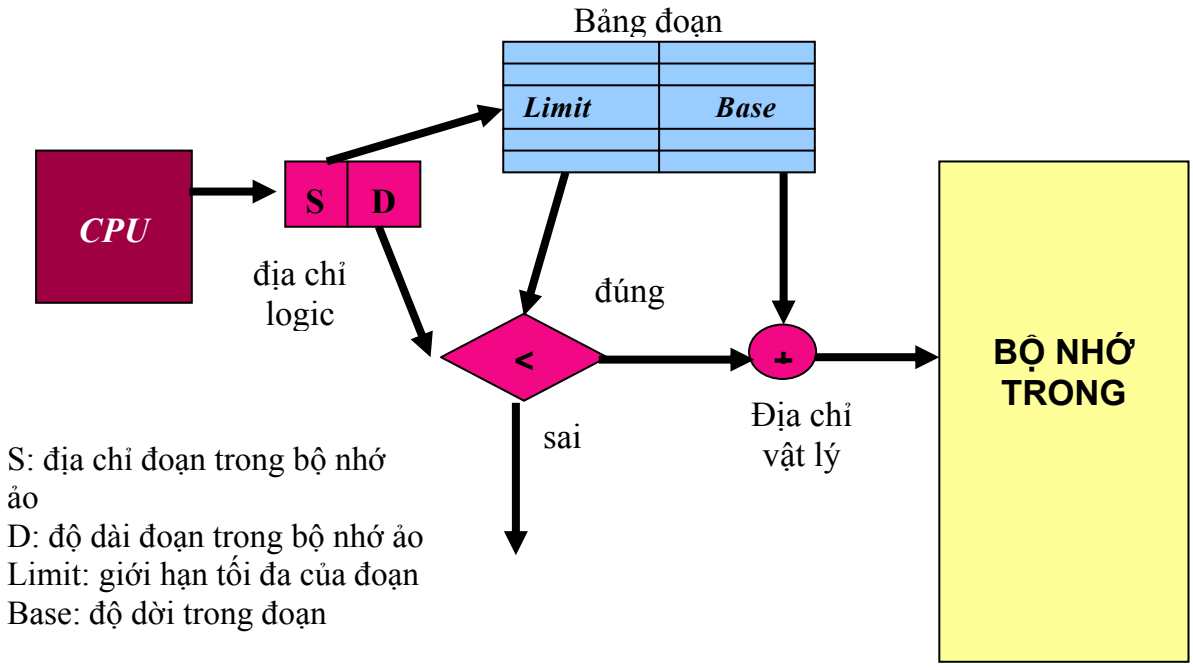
Hình IV.8: ***Ánh xạ các trang ảo vào bộ nhớ vật lý***

Câu hỏi 2: ***Làm thế nào để tìm một khối khi nó đang nằm trong bộ nhớ trong?***

Định vị trang và định vị đoạn đều dựa vào một cấu trúc dữ liệu trong đó số thứ tự trang hoặc số thứ tự đoạn được có chỉ số. Cho định vị trang, dựa vào bảng trang, địa chỉ trong bộ nhớ vật lý được xác lập cuối cùng là việc đặt kề nhau số thứ của trang vật lý với địa chỉ trong trang (hình IV.9). Cho định vị đoạn, dựa vào thông tin trên bảng đoạn, việc kiểm tra tính hợp lệ của địa chỉ được tiến hành. Địa chỉ vật cuối cùng được xác lập bằng cách cộng địa chỉ đoạn và địa chỉ trong đoạn (độ dời trong đoạn) (hình IV.10).



***Hình IV.9** : Minh hoạ sự ánh xạ địa chỉ giữa bộ nhớ ảo và bộ nhớ vật lý trong định vị trang*



***Hình IV.10** : Ánh xạ địa chỉ giữa bộ nhớ ảo và bộ nhớ vật lý trong cách định vị đoạn*

Câu hỏi 3: ***Khối nào phải được thay thế khi có thất bại trang?***

Hầu hết các hệ điều hành đều cố gắng thay thế khối ít dùng gần đây nhất (LRU: Least Recent Utilized) vì nghĩ rằng đây là khối ít cần nhất.

Câu hỏi 4: ***Việc gì xảy ra khi cần ghi số liệu?***

Chiến thuật ghi luôn là một sự ghi lại nghĩa là thông tin chỉ được viết vào trong khối của bộ nhớ trong. Khối có thay đổi thông tin, được chép vào đĩa từ nếu khối này bị thay thế.

IV.10. BẢO VỆ CÁC TIẾN TRÌNH BẰNG CÁCH DÙNG BỘ NHỚ ẢO

Sự xuất hiện của đa chương trình (multiprogram) trong đó máy tính chạy nhiều chương trình song song với nhau, dẫn tới các đòi hỏi mới về việc bảo vệ và phân chia giữa các chương trình.

Đa chương trình đưa đến khái niệm tiến trình (process): *một tiến trình gồm có một chương trình đang thực hiện và tất cả các thông tin cần thiết để tiếp tục thực hiện chương trình này*.

Trong đa chương trình, bộ xử lý và bộ nhớ trong được nhiều người sử dụng chia sẻ một cách qua lại (interactive), cùng một thời điểm, để tạo cảm giác rằng mỗi người dùng đang có một máy tính riêng. Và như thế, tại bất cứ lúc nào, phải có thể chuyển đổi từ một tiến trình này sang một tiến trình khác.

Một tiến trình phải vận hành đúng đắn, dù nó được thi hành liên tục từ đầu tới cuối, hay nó bị ngắt qua lại bởi các tiến trình khác. Trách nhiệm đảm bảo các tiến trình đều chạy đúng, được chia sẻ giữa nhà thiết kế máy tính và nhà thiết kế hệ điều hành. Nhà thiết kế máy tính phải đảm bảo bộ xử lý có thể lưu giữ trạng thái các tiến trình và phục hồi các trạng thái này, còn nhà thiết kế hệ điều hành phải đảm bảo các tiến trình không ảnh hưởng lên nhau. Hệ điều hành giải quyết vấn đề này bằng cách chia bộ nhớ trong cho các tiến trình và trạng thái của mỗi tiến trình này hiện diện trong phần bộ nhớ được chia cho nó. Điều này có nghĩa rằng các nhà thiết kế hệ điều hành phải được sự giúp sức của các nhà chế tạo máy tính để bảo vệ một tiến trình không bị ảnh hưởng bởi tiến trình khác.

Nhà thiết kế máy tính có thêm 3 trách nhiệm trong việc giúp các nhà thiết kế hệ điều hành bảo vệ các tiến trình là:

1. Cung cấp hai chế độ vận hành cho biết tiến trình đang thực hiện là tiến trình của người sử dụng hay tiến trình hệ thống (của người điều hành).

2. Cung cấp một tập hợp con trạng thái của bộ xử lý mà tiến trình người sử dụng có thể dùng nhưng không thể sửa đổi.

3. Cung cấp các cơ chế để có thể chuyển đổi từ chế độ người dùng sang chế độ người điều hành và ngược lại.

Chúng ta đã thấy, địa chỉ mà bộ xử lý đưa ra phải được biến đổi từ địa chỉ ảo sang địa chỉ vật lý. Điều này giúp phần cứng đi xa nữa trong việc bảo vệ các tiến trình. Cách đơn giản nhất làm việc này là cho phép tiến trình người sử dụng tác động lên các bit cho phép thâm nhập vào mỗi trang hay mỗi đoạn. Khi bộ xử lý phát ra tín hiệu đọc (hay viết) và tín hiệu người dùng (hay hệ thống) thì rất dễ dàng phát hiện các việc thâm nhập

trái phép bộ nhớ trước khi việc thâm nhập này gây hư hại. Các tiến trình được bảo vệ và có bảng trang riêng cho mình trỏ đến các trang tách rời nhau trong bộ nhớ.

*****

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG IV

*****

1. Sự khác nhau giữa SRAM và DRAM? Trong máy tính chúng được dùng ở đâu?
2. Mục tiêu của các cấp bộ nhớ?
3. Nêu hai nguyên tắc mà cache dựa vào đó để vận hành.
4. Cho một bộ nhớ cache tương ứng trực tiếp có 8 khối, mỗi khối có 16 byte. Bộ nhớ trong có 64 khối. Giả sử lúc khởi động máy, 8 khối đầu tiên của bộ nhớ trong được đưa lên cache.
   a. Viết bảng nhãn của các khối hiện đang nằm trong cache
   b. CPU lần lượt đưa các địa chỉ sau đây để đọc số liệu: O4AH, 27CH, 3F5H. Nếu thất bại thì cập nhật bãng nhãn.
   c. CPU dùng cách ghi lại. Khi thất bại cache, CPU dùng cách ghi có nạp. Mô tả công việc của bộ quản lý cache khi CPU đưa ra các từ sau đây để ghi vào bộ nhớ trong: 0C3H, 05AH, 1C5H.
5. Các nguyên nhân chính gây thất bại cache?
6. Các giải pháp đảm bảo tính đồng nhất dữ liệu trong hệ thống bộ đa xử lý có bộ nhớ chia sẻ dùng chung?
7. Các cách nới rộng dãy thông của bộ nhớ trong?
8. Tại sao phải dùng bộ nhớ ảo?
9. Sự khác biệt giữa cache và bộ nhớ ảo?

Chương V: NHẬP - XUẤT

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

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

V.1. DẪN NHẬP

Bộ xử lý của máy tính điện tử liên hệ với bên ngoài nhờ các bộ phận xuất nhập (I/O) mà ta còn gọi là ngoại vi.

Các ngoại vi thông dụng là:

- Màn hình, bàn phím, chuột, máy in, thẻ mạng... là những bộ phận giúp con người sử dụng máy tính dễ dàng.
- Các đĩa từ, băng từ, đĩa quang, các loại thẻ nhớ là những bộ phận lưu trữ thông tin trữ lượng lớn.

Tất cả các ngoại vi đều được nối vào bộ xử lý và bộ nhớ trong bằng một hệ thống dây nối phức tạp vì tính đa dạng của các ngoại vi.

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

V.2. ĐĨA TỪ

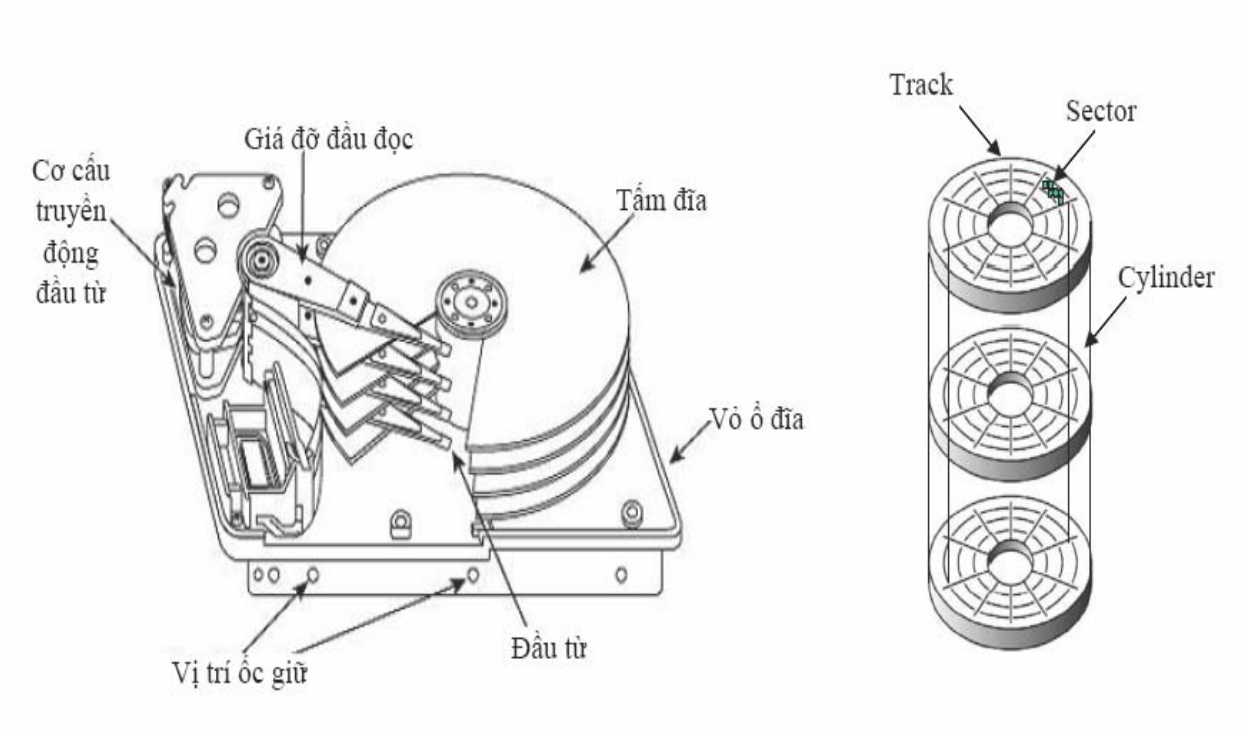
Dù rằng công nghệ mới không ngừng phát minh nhiều loại bộ phận lưu trữ một lượng thông tin lớn nhưng đĩa từ vẫn giữ vị trí quan trọng từ năm 1965. Đĩa từ có hai nhiệm vụ trong máy tính.

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

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

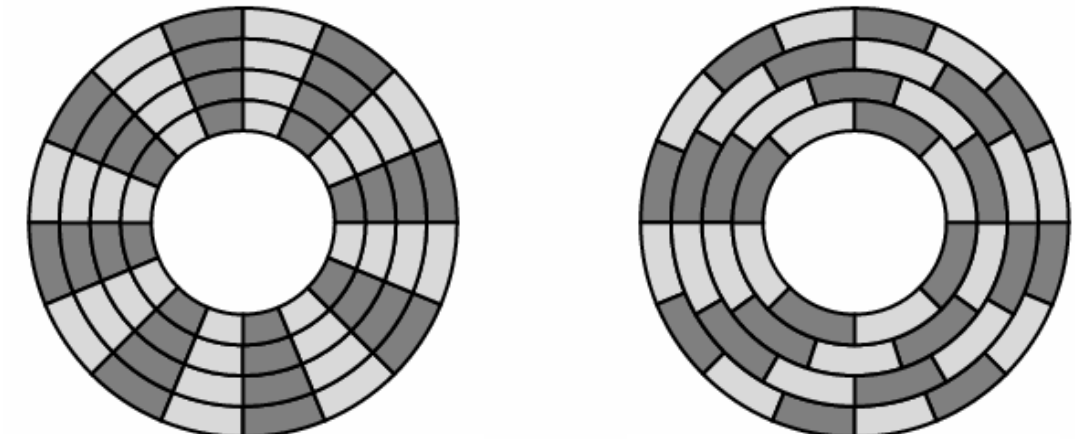
Một đĩa cứng chứa nhiều lớp đĩa (từ 1 đến 4) quay quanh một trục khoảng 3.600-15.000 vòng mỗi phút. Các lớp đĩa này được làm bằng kim loại với hai mặt được phủ một chất từ tính (hình V.1). Đường kính của đĩa thay đổi từ 1,3 inch đến 8 inch. Mỗi mặt của một lớp đĩa được chia thành nhiều đường tròn đồng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

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



Hình 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.



Mật độ ghi không đều ***Mật độ ghi đều***

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.

Bảng 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**

V.3. ĐĨA QUANG

Các thiết bị lưu trữ quang rất thích hợp cho việc phát hành các sản phẩm văn hoá, sao lưu dữ liệu trên các hệ thống máy tính hiện nay. Ra đời vào năm 1978, đây là sản phẩm của sự hợp tác nghiên cứu giữa hai công ty Sony và Philips trong công nghiệp giải trí. Từ năm 1980 đến nay, công nghiệp đĩa quang phát triển mạnh trong cả hai lĩnh vực giải trí và lưu trữ dữ liệu máy tính. Quá trình đọc thông tin dựa trên sự phản chiếu của các tia laser năng lượng thấp từ lớp lưu trữ dữ liệu. Bộ phận tiếp nhận ánh sáng sẽ nhận biết được những điểm mà tại đó tia laser bị phản xạ mạnh hay biến mất do các vết khắc (pit) trên bề mặt đĩa. Các tia phản xạ mạnh chỉ ra rằng tại điểm đó không có lỗ khắc và điểm này được gọi là điểm nền (land). Bộ nhận ánh sáng trong ổ đĩa thu nhận các tia phản xạ và khuếch tán được khúc xạ từ bề mặt đĩa. Khi các nguồn sáng được thu nhận, bộ vi xử lý sẽ dịch các mẫu sáng thành các bit dữ liệu hay âm thanh. Các lỗ trên CD sâu 0,12 micron và rộng 0,6 micron (1 micron bằng một phần ngàn mm). Các lỗ này được khắc theo một track hình xoắn ốc với khoảng cách 1,6 micron giữa các vòng, khoảng 16.000 track/inch. Các lỗ (pit) và nền (land) kéo dài khoản 0,9 đến 3,3 micron. Track bắt đầu từ phía trong và kết thúc ở phía ngoài theo một đường khép kín các rìa đĩa 5mm. Dữ liệu lưu trên CD thành từng khối, mỗi khối chứa 2.352 byte. Trong đó, 304 byte chứa các

thông tin về bit đồng bộ, bit nhận dạng (ID), mã sửa lỗi (ECC), mã phát hiện lỗi (EDC). Còn lại 2.048 byte chứa dữ liệu. Tốc độ đọc chuẩn của CD-ROM là 75 khối/s hay 153.600 byte/s hay 150KB/s (1X).

Dưới đây là một số loại đĩa quang thông dụng.

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

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

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

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

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

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

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

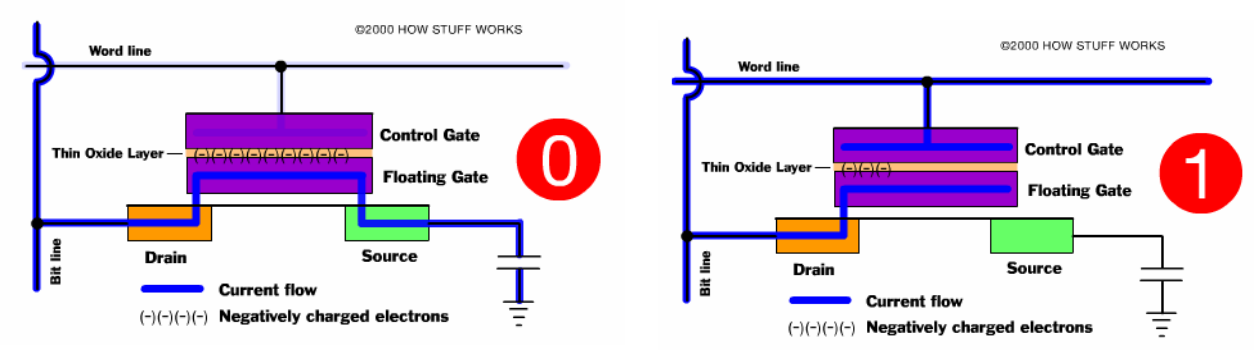
Đặc trưng	CDROM	DVDROM
Kích thước Pit	0.834 micron	0.4 micron
Khoảng cách rãnh	1.6 micron	0.74 micron
Số lớp dữ liệu trên đĩa	1 lớp	2 lớp
Số mặt đĩa	1 mặt	1 - 2 mặt
Dung lượng	640-700 MB	1.36 – 17 GB
Độ phân giải phim	VCD=320x200	720x640

***Bảng V.2:** So sánh một số thông số của hai loại đĩa CDROM và DVDROM*

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

V.4. 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 rộng rãi trong công nghệ lưu trữ và giải trí hiện nay.



***Hình V.3:** Minh hoạ hai trạng thái của một bit nhớ trong thẻ nhớ*

V.5. 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).

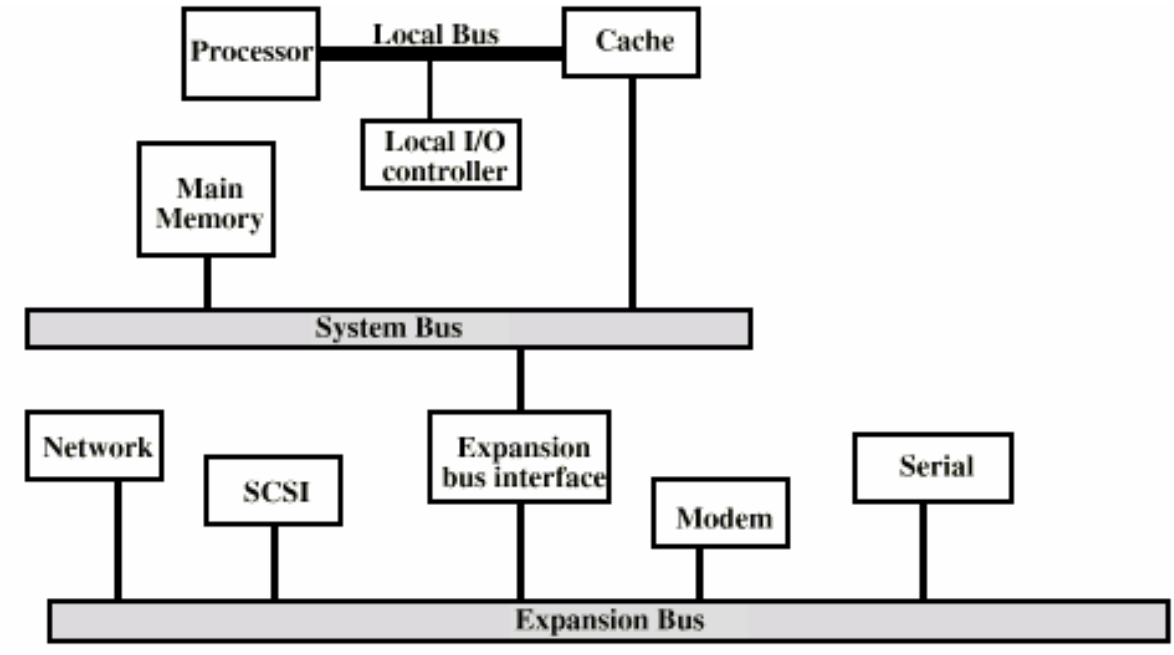
V.6. BUS NỐI NGOẠI VI VÀO BỘ XỬ LÝ VÀ BỘ NHỚ TRONG

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.

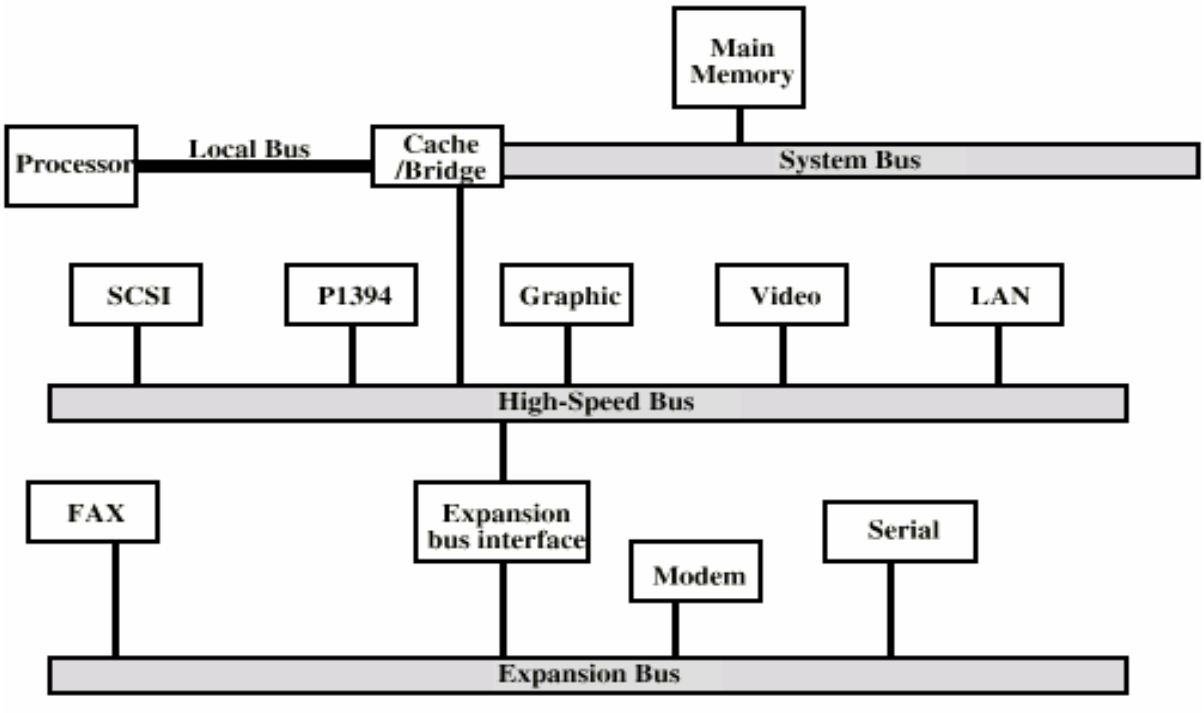
Một trong những lý do khiến cho việc thiết kế một hệ thống bus khó khăn là tốc độ tối đa của bus bị giới hạn bởi các yếu tố vật lý như chiều dài của bus và số bộ phận được mắc vào bus.

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 (xem hình V.6).

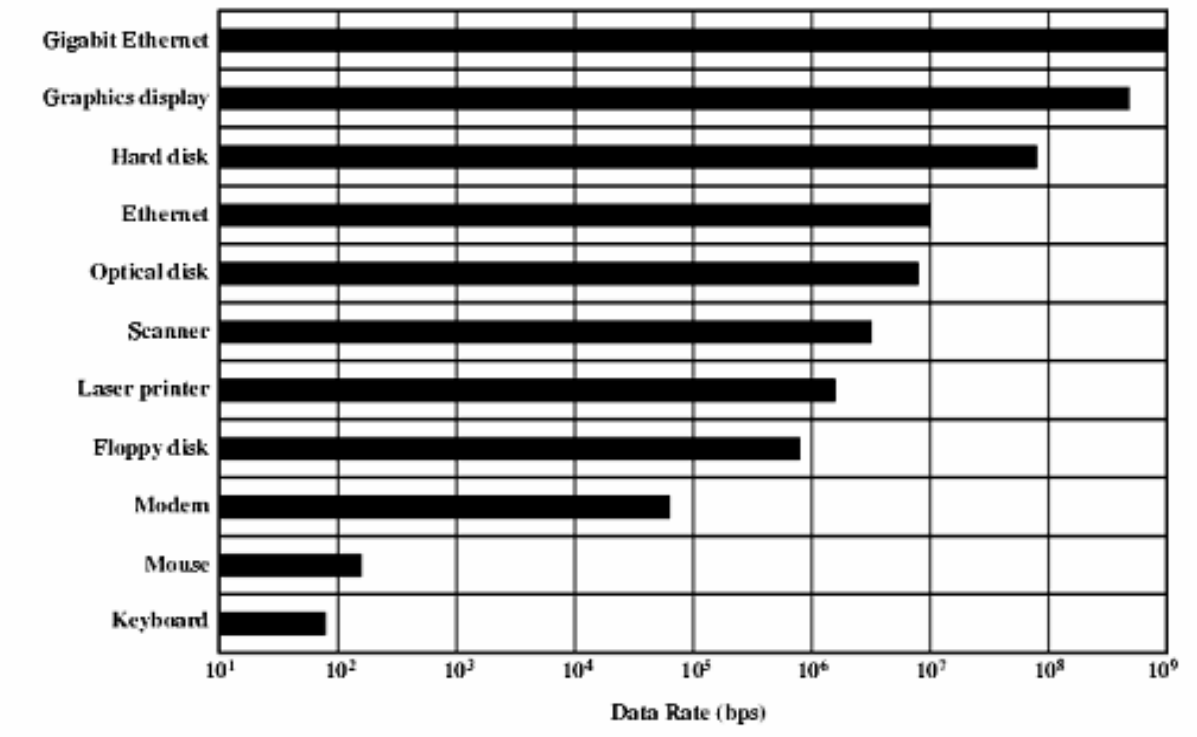


***Hình V.4:* Hệ thống bus trong một máy tính**

Hiện nay, trong một số hệ thống máy tính, bus nối ngoại vi được phân cấp thành hai hệ thống bus con. Trong đó, bus tốc độ cao (high-speed bus) hỗ trợ kết nối các thiết bị tốc độ cao như SCSI, LAN, Graphic, Video,…và hệ thống bus mở rộng (expansion bus) được thiết kế để kết nối với các ngoại vi yêu cầu tốc độ thấp như: modem, cổng nối tiếp, cổng song song,…Giữa hai hệ thống bus nối ngoại vi trong tổ chức hệ thống bus phân cấp là một giao diện đệm (hình V.5).



***Hình V.5:** Hệ thống bus phân cấp*



***Hình V.6:** Bảng biểu diễn tốc độ dữ liệu của các ngoại vi*

Ta có thể có nhiều lựa chọn trong việc thiết kế một bus, như trong bảng V.3.

Đặc tính của bus	Bus hệ thống	Bus nối ngoại vi
Độ rộng của bus	Đường dây địa chỉ và số liệu khác nhau	Đường địa chỉ và số liệu được đa hợp
Độ rộng bus số liệu	Càng rộng càng nhanh (ví dụ 64 bít)	Càng hẹp càng ít tốn kém (ví dụ 8 bít)
Số từ được chuyển	Chuyển nhiều từ	Chuyển đơn giản mỗi lần một từ
Chủ nhân của bus	Nhiều	Một
Chuyển từng gói	Có. Cần nhiều chủ nhân bus	Không. Kết nối một lần và chuyển hết thông tin
Xung nhịp	Đồng bộ	Bất đồng bộ

***Bảng V.3:** Các lựa chọn chính yếu cho một bus*

Trong bảng V.3 có khái niệm sau đây liên quan đến các *chủ nhân của bus* - các bộ phận có thể khởi động một tác vụ đọc hoặc viết trên bus. Ví dụ bộ xử lý luôn là một chủ nhân của bus. Một bus có nhiều chủ nhân khi nó có nhiều bộ xử lý, hoặc khi các ngoại vi có thể khởi động một tác vụ có dùng bus. Nếu có nhiều chủ nhân của bus thì phải có một cơ chế trọng tài để quyết định chủ nhân nào được quyền chiếm lĩnh bus. Một bus có nhiều chủ, có thể cấp một dãi thông rộng (bandwidth) bằng cách sử dụng các gói tin thay vì dùng bus cho từng tác vụ riêng lẻ. Kỹ thuật sử dụng gói tin được gọi là phân chia nhỏ tác vụ (dùng bus chuyển gói). Một tác vụ đọc được phân tích thành một tác vụ yêu cầu đọc (tác vụ này chứa địa chỉ cần đọc), và một tác vụ trả lời của bộ nhớ (chứa thông tin cần đọc). Mỗi tác vụ đều có một nhãn cho biết loại của tác vụ. Trong kỹ thuật phân chia nhỏ tác vụ, trong khi bộ nhớ đọc các thông tin ở địa chỉ đã xác định thì bus được dành cho các chủ khác.

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

Ngược lại, các bus vào/ra thuộc loại bus bất đồng bộ, các bus này không có xung nhịp đồng bộ trong hệ thống bus. Thay vào đó có các nghi thức bắt tay với các quy định riêng về thời gian, được dùng giữa các bộ phận phát và bộ phận thu của bus. Bus bất đồng bộ rất dễ thích ứng với nhiều ngoại vi và cho phép nối dài bus mà không phải lo ngại gì đến vấn đề đồng bộ. Bus bất đồng bộ cũng dễ thích ứng với những thay đổi công nghệ.

V.7. CÁC CHUẨN VỀ BUS

Số lượng và chủng loại các bộ phận vào/ra không cần định trước trong các hệ thống xử lý thông tin. Điều này giúp cho người sử dụng máy tính dùng bộ phận vào/ra nào đáp ứng được các yêu cầu của họ. Vào/ra là giao diện trên đó các bộ phận (thiết bị)

được kết nối vào hệ thống. Nó có thể xem như một bus nới rộng dùng để kết nối thêm ngoại vi vào máy tính. Các chuẩn làm cho việc nối kết các ngoại vi vào máy tính được dễ dàng; bởi vì, trong khi các nhà thiết kế-sản xuất máy tính và các nhà thiết kế-sản xuất ngoại vi có thể thuộc các công ty khác nhau. Sự tồn tại các chuẩn về bus là rất cần thiết. Như vậy, nếu nhà thiết kế máy tính và nhà thiết kế ngoại vi tôn trọng các chuẩn về bus này thì các ngoại vi có thể kết nối dễ dàng vào máy tính. Chuẩn của bus vào/ra là tài liệu quy định cách kết nối ngoại vi vào máy tính.

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

V.8. GIAO DIỆN GIỮA BỘ XỬ LÝ VỚI CÁC BỘ PHẬN VÀO RA

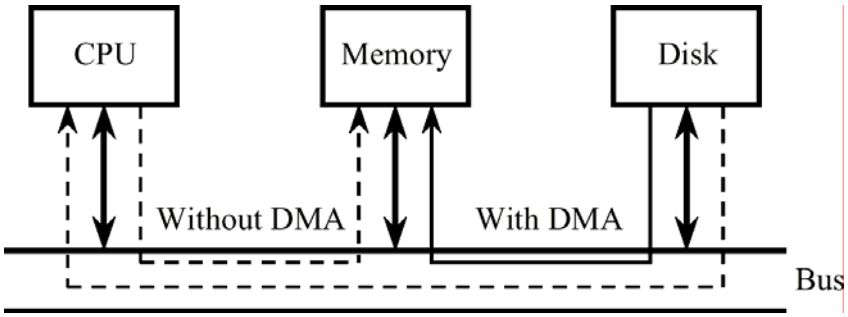
Bộ xử lý dùng 2 cách để liên lạc với các bộ phận vào ra:

Cách thứ nhất, thường được dùng: là cách dùng một vùng địa chỉ của bộ nhớ làm vùng địa chỉ của các ngoại vi. Khi đọc hay viết vào vùng địa chỉ này của bộ nhớ là liên hệ đến các ngoại vi.

Cách thứ hai, dùng mã lệnh riêng biệt cho vào/ra (tức là có các lệnh vào/ra riêng, không trùng với lệnh đọc hay viết vào ô nhớ). Trong trường hợp này, bộ xử lý gởi một tín hiệu điều khiển cho biết địa chỉ đang dùng là của một ngoại vi. Vi mạch Intel 8086 và máy IBM 370 là các ví dụ về bộ xử lý dùng lệnh vào/ra riêng biệt.

Dù dùng cách nào để định vị vào/ra thì mỗi bộ phận vào/ra đều có các thanh ghi để cung cấp thông tin về trạng thái và về điều khiển. Bộ phận vào/ra dùng bit trạng thái "*sẵn sàng*" để báo cho bộ xử lý nó sẵn sàng nhận số liệu. Định kỳ bộ xử lý xem xét bít này để biết bộ phận vào ra có sẵn sàng hay không. Phương pháp này là *phương pháp thăm dò* (polling). Và nhược điểm của phương pháp này là làm mất thời gian của bộ xử lý vì định kỳ phải thăm dò tính sẵn sàng của các thiết bị ngoại vi. Điều này đã được nhận thấy từ lâu và đã dẫn đến phát minh ra ngắt quãng (interrupt) để báo cho bộ xử lý biết lúc có một bộ phận vào/ra cần được phục vụ.

Việc dùng ngắt quãng làm cho bộ xử lý không mất thời gian thăm dò xem các ngoại vi có yêu cầu phục vụ hay không, nhưng bộ xử lý phải mất thời gian chuyển dữ liệu. Thông thường việc trao đổi số liệu giữa ngoại vi và CPU là theo khối số liệu, nên vi mạch thâm nhập trực tiếp bộ nhớ trong (DMA: Direct Memory Access) được dùng trong nhiều máy tính để chuyển một khối nhiều từ mà không có sự can thiệp của CPU.



Hình V.7. Sơ đồ hoạt động của hệ thống bus có vi mạch DMA

DMA là một vi mạch chức năng đặc biệt. Nó chuyển số liệu giữa ngoại vi và bộ nhớ trong, trong lúc đó CPU rãnh rỗi để làm công việc khác. Vậy DMA nằm ngoài CPU và tác động như là một chủ nhân của bus. Bộ xử lý khởi động các thanh ghi của DMA, các thanh ghi này chứa địa chỉ ô nhớ và số byte cần chuyển. DMA chủ động chuyển số liệu và khi chấm dứt thì trả quyền điều khiển cho bộ xử lý.

Vi mạch DMA càng thông minh thì công việc của CPU càng nhẹ đi. Nhiều vi mạch được gọi là bộ xử lý vào/ra (hay bộ điều khiển vào/ra) thực hiện công việc mình theo một chương trình cố định (chứa trong ROM), hay theo một chương trình mà hệ điều hành nạp vào bộ nhớ trong. Hệ điều hành thiết lập một hàng chờ đợi gồm các khối điều khiển các bộ phận vào/ ra. Các khối chứa các thông tin như là vị trí của số liệu (nguồn và đích) và số số liệu. Các bộ xử lý vào/ra lấy các thông tin này trong hàng chờ đợi, thực hiện các việc cần phải làm và gởi về CPU tín hiệu ngắt khi đã thực hiện xong công việc.

Một máy tính có bộ xử lý vào/ra được xem như một máy tính đa xử lý vì DMA giúp cho máy tính thực hiện cùng lúc nhiều quá trình. Tuy nhiên bộ xử lý vào/ra không tổng quát bằng các bộ xử lý vì chúng chỉ làm được một số việc nhất định. Hơn nữa bộ xử lý vào/ra không chế biến số liệu như các bộ xử lý thường làm. Nó chỉ di chuyển số liệu từ nơi này sang nơi khác.

V.9. MỘT SỐ BIỆN PHÁP AN TOÀN DỮ LIỆU TRONG VIỆC LƯU TRỮ THÔNG TIN TRONG ĐĨA TỪ

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

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

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

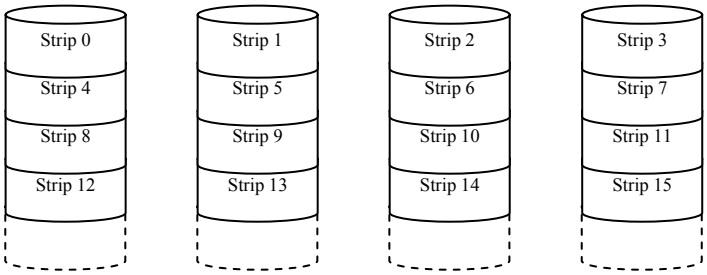
2. Với cơ chế đọc/ghi thông tin diễn ra trên nhiều đĩa (ghi đan chéo hay soi gương).

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

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

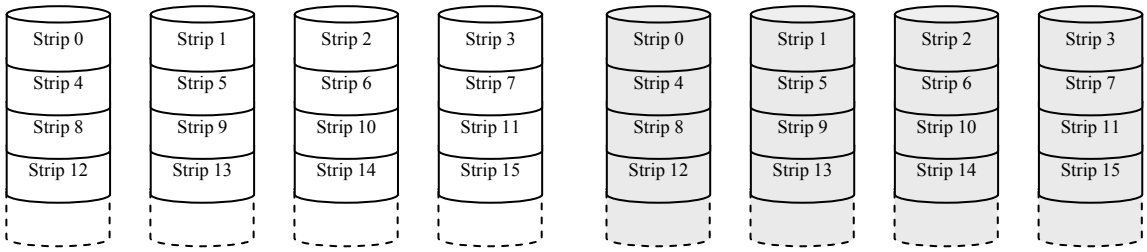
i). ***RAID 0***: Thực ra, kỹ thuật này không nằm trong số các kỹ thuật có cơ chế an toàn dữ liệu. Khi mảng được thiết lập theo RAID 0, ổ đĩa logic có được (mà hệ điều hành nhận biết) có dung dượng bằng tổng dung lượng của các ổ đĩa thành viên. Điều này giúp cho người dùng có thể có một ổ đĩa logic có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



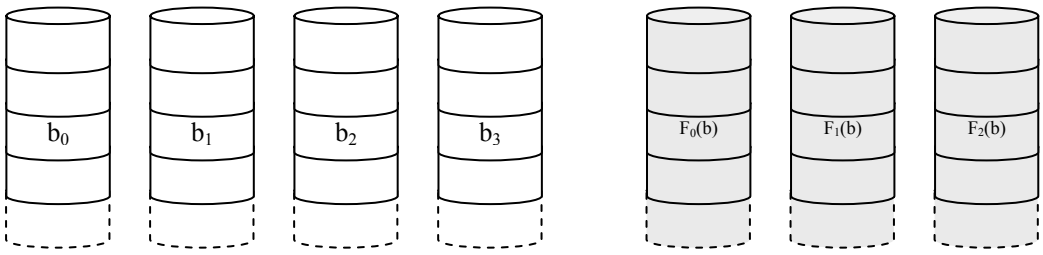
Hình V.8: RAID 0

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%).



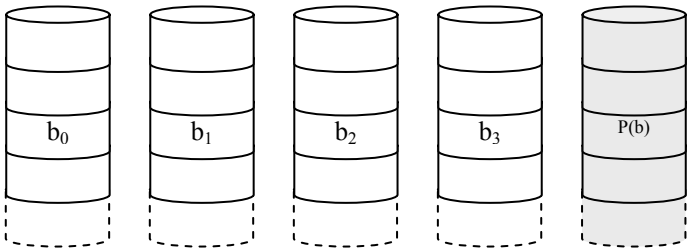
Hình V.9: RAID 1

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.



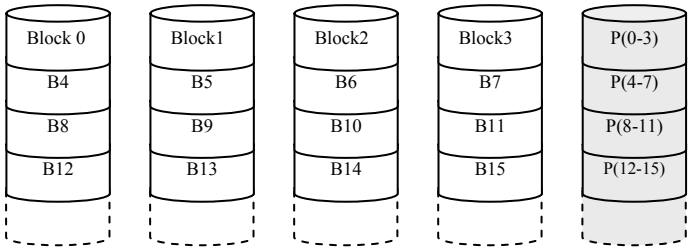
Hình V.10: RAID 2

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).



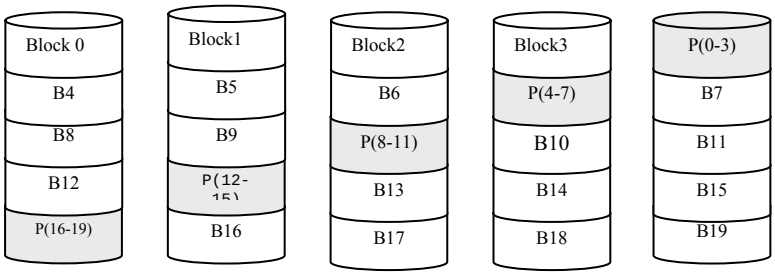
Hình V.11: RAID 3

iv) ***RAID 4:*** từ RAID 4 đến RAID 6 dùng kỹ thuật truy cập các đĩa trong mảng độc lập. Trong một mảng truy cập độc lập, mỗi đĩa thành viên được truy xuất độc lập, do đó mảng có thể đáp ứng được các yêu cầu song song của ngoại vi. Kỹ thuật này thích hợp với các ứng dụng yêu cầu nhiều ngoại vi là các ứng dụng yêu cầu tốc độ truyền dữ liệu cao. Trong RAID 4, một đĩa dùng để chứa các bit kiểm tra được tính toán từ dữ liệu được lưu trên các đĩa dữ liệu. Khuyết điểm lớn nhất của RAID 4 là bị nghẽn cổ chai tại đĩa kiểm tra khi có nhiều yêu cầu đồng thời từ các ngoại vi.



Hình V.12: RAID 4

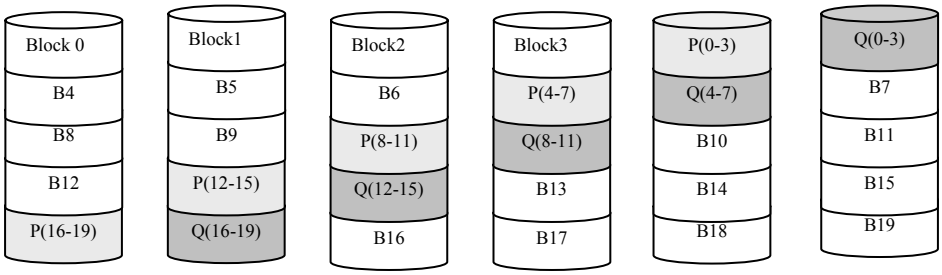
v). ***RAID 5:*** yêu cầu thiết lập giống như RAID 4, dữ liệu được ghi từng khối trên các đĩa thành viên, các bit chẵn lẻ được tính toán mức độ khối được ghi trải đều lên trên tất cả các ổ đĩa trong mảng. Tương tự RAID 4, khi một đĩa bất kỳ trong mảng bị hư hỏng, hệ thống vẫn hoạt động bình thường. Khi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



Hình V.14: RAID 6

*****

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG V

*****

1. Mô tả vận hành của ổ đĩa cứng. Cách lưu trữ thông tin trong ổ đĩa cứng
2. Mô tả các biện pháp an toàn trong việc lưu trữ thông tin trong đĩa cứng.
3. Nguyên tắc vận hành của đĩa quang. Ưu khuyết điểm của các loại đĩa quang.
4. Thông thường có bao nhiêu loại bus? Tại sao phải có các chuẩn cho các bus vào ra?
5. Thế nào là chủ nhân của bus? Khi bus có nhiều chủ nhân thì làm thế nào để giải quyết tranh chấp bus?
6. Giải thích việc nới rộng dãi thông bằng cách sử dụng các gói tin.
7. Sự khác biệt giữa bộ xử lý vào ra và bộ xử lý trung tâm của máy tính.