



CẤU TRÚC MÁY TÍNH

Computer Architecture

Hoàng Văn Hiệp

Bộ môn Kỹ thuật máy tính, Khoa CNTT, ĐHBK Hà Nội

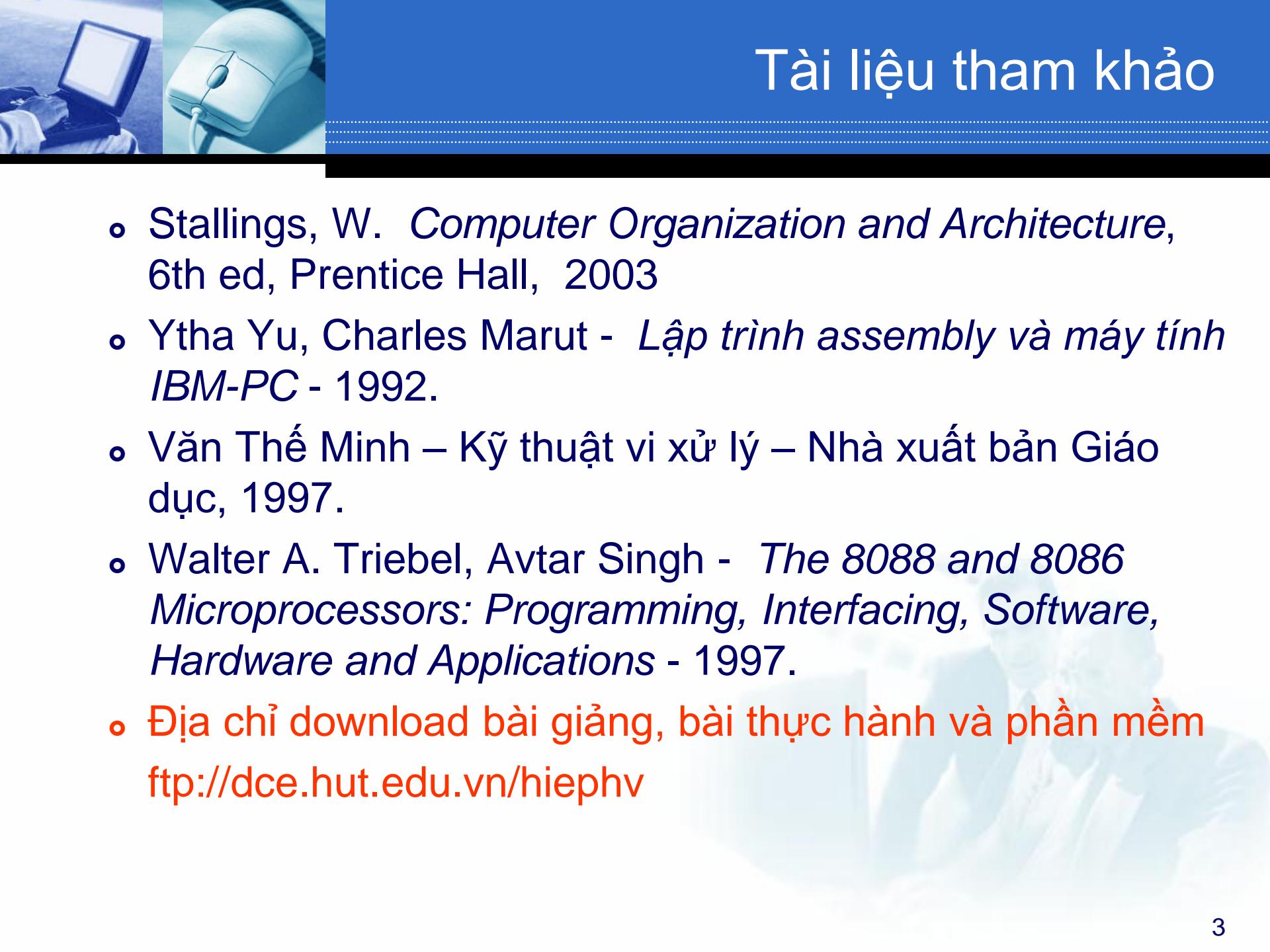
Mob. 0916093209

Email. hiephv@soict.hut.edu.vn



Chú ý về bản quyền

- Toàn bộ slide môn học Cấu trúc máy tính được xây dựng dựa trên slide của Thầy Nguyễn Kim Khánh và Thầy Nguyễn Phú Bình, bộ môn Kỹ thuật máy tính, khoa Công nghệ thông tin, Đại học Bách Khoa Hà Nội.
- Yêu cầu người học không phô biến, chỉnh sửa nội dung của slide này nếu chưa được sự cho phép của tác giả.
- **XIN CẢM ƠN!**



Tài liệu tham khảo

- Stallings, W. *Computer Organization and Architecture*, 6th ed, Prentice Hall, 2003
- Ytha Yu, Charles Marut - *Lập trình assembly và máy tính IBM-PC* - 1992.
- Văn Thế Minh – Kỹ thuật vi xử lý – Nhà xuất bản Giáo dục, 1997.
- Walter A. Triebel, Avtar Singh - *The 8088 and 8086 Microprocessors: Programming, Interfacing, Software, Hardware and Applications* - 1997.
- Địa chỉ download bài giảng, bài thực hành và phần mềm <ftp://dce.hut.edu.vn/hiephv>



Nội dung môn học

- Chương 1: Giới thiệu chung
- Chương 2: Biểu diễn dữ liệu và số học máy tính
- Chương 3: Hệ thống máy tính
- Chương 4: Họ máy tính IBM-PC
- Chương 5: Lập trình hợp ngữ trên PC

Chương 1

Giới thiệu chung

- 1. Máy tính và phân loại máy tính**
 - 2. Sự tiến hóa của máy tính**
- 



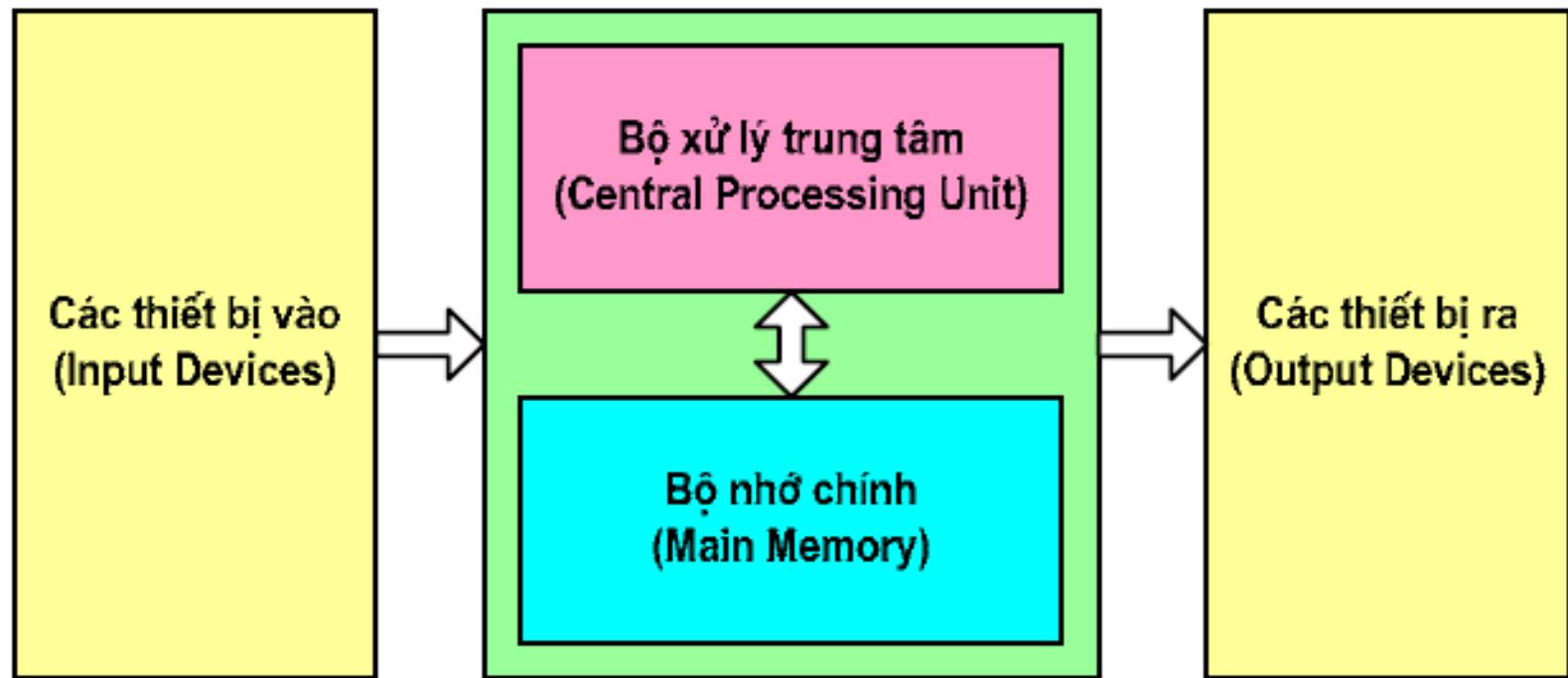
Máy tính và phân loại máy tính

Định nghĩa máy tính:

- Thiết bị điện tử thực hiện các công việc sau:
 - Nhận thông tin vào
 - Xử lý thông tin theo chương trình được nhớ sẵn bên trong
 - Đưa thông tin ra
- ⇒ **Máy tính hoạt động theo chương trình.**



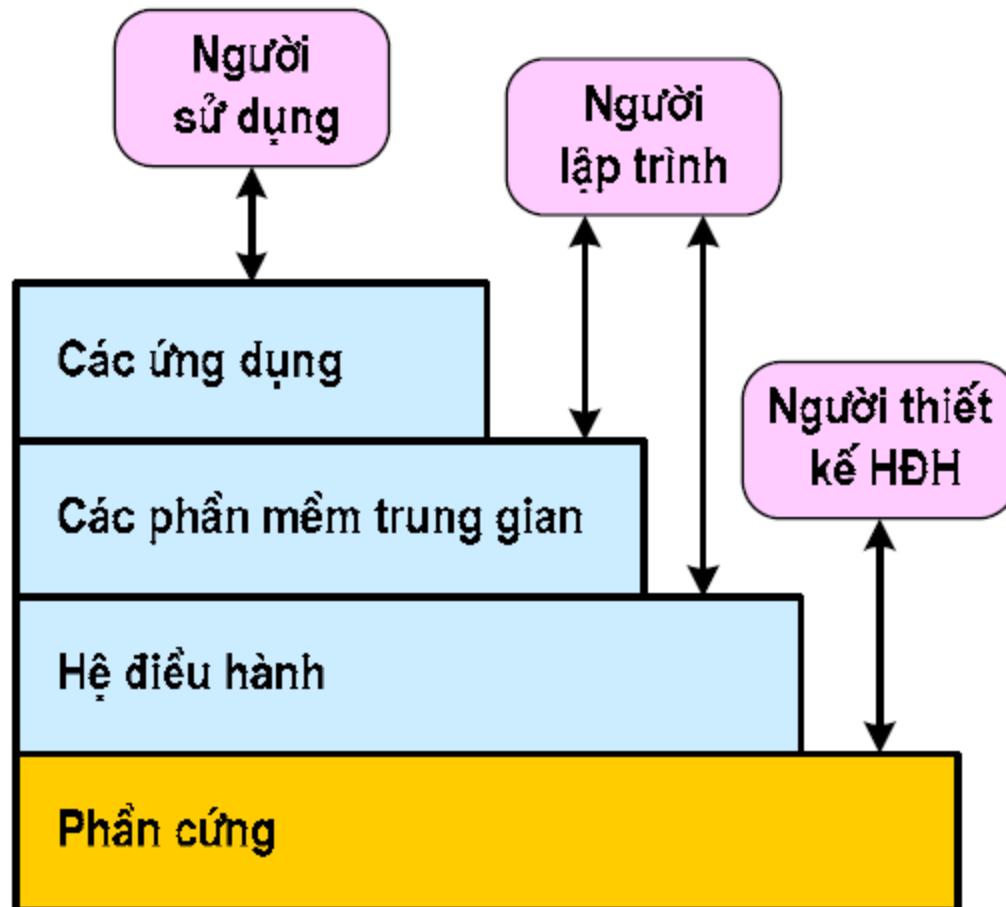
Máy tính và phân loại máy tính



Mô hình máy tính cơ bản



Máy tính và phân loại máy tính



Mô hình phân lớp của máy tính



Phân loại máy tính

Phân loại truyền thống:

- Máy vi tính (Microcomputer)
- Máy tính nhỏ (Minicomputer)
- Máy tính lớn (Mainframe Computer)
- Siêu máy tính (Supercomputer)



Phân loại máy tính

Phân loại hiện đại:

- Máy tính cá nhân (Personal Computer)
- Máy chủ (Server)
- Máy tính nhúng (Embedded Computer)





Máy tính cá nhân

- Là loại máy tính phổ biến nhất đối với người dùng thông thường.
- Thiết kế theo hướng tối ưu cả về giá thành và hiệu năng
- Một số loại:
 - Máy tính để bàn (Desktop)
 - Máy tính xách tay (Notebook)
 - Máy trạm làm việc (Workstation)
- Giá thành: từ vài trăm đến vài nghìn USD



Máy tính cá nhân



Desktop



Notebook



Workstation

Máy chủ (Server)

- Thực chất là máy phục vụ
- Dùng trong mạng máy tính theo mô hình Client/Server
- Tốc độ và hiệu năng tính toán cao
- Dung lượng bộ nhớ lớn
- Độ tin cậy cao
- Giá thành: từ hàng chục nghìn đến hàng triệu USD.



Máy Server



Máy tính nhúng (Embedded Computer)

- Được đặt trong thiết bị khác (bao gồm cả phần cứng và các kết cấu cơ khí) để điều khiển thiết bị đó làm việc
- Được thiết kế chuyên dụng
 - Ví dụ:
 - Điện thoại di động
 - Bộ điều khiển trong máy giặt, điều hòa nhiệt độ
 - Một số thiết bị mạng: Switch, Router, ...
- Giá thành: từ vài USD đến hàng trăm ngàn USD



Máy tính nhúng

Mobile phone



Camera



Washing machine



Play Station II

Kiến trúc tập lệnh
(Instruction Set Architecture – ISA)

Tổ chức máy tính
(Computer Organization)

Kiến trúc máy tính



Kiến trúc tập lệnh

- Nghiên cứu cấu trúc và hoạt động của máy tính theo cách nhìn của người lập trình.
- Kiến trúc tập lệnh của máy tính bao gồm
 - Tập lệnh: tập hợp các chuỗi số nhị phân mã hóa cho các thao tác mà máy tính có thể thực hiện được.
 - Kiểu dữ liệu: các kiểu dữ liệu mà máy tính có thể xử lý.
 - Chế độ địa chỉ

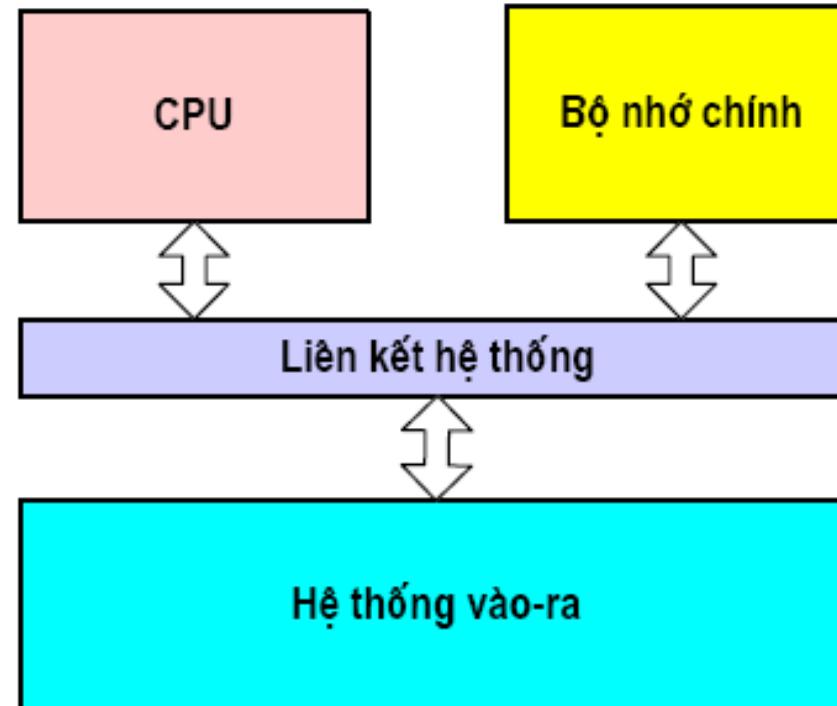


Tổ chức máy tính

- Nghiên cứu cấu trúc phần cứng của máy tính.
- Các thành phần cơ bản của máy tính
 - Bộ xử lý trung tâm (Central Processing Unit): điều khiển hoạt động của máy tính và xử lý dữ liệu.
 - Bộ nhớ chính (Main Memory): chứa các chương trình và dữ liệu đang được sử dụng.
 - Hệ thống vào ra (Input/Output System): trao đổi thông tin giữa máy tính và bên ngoài.
 - Liên kết hệ thống (System Interconnection): kết nối và vận chuyển thông tin giữa các thành phần với nhau



Tổ chức máy tính



Cấu trúc cơ bản của máy tính



Nội dung chương 1

- 1. Máy tính và phân loại máy tính**
- 2. Sự tiến hóa của máy tính**

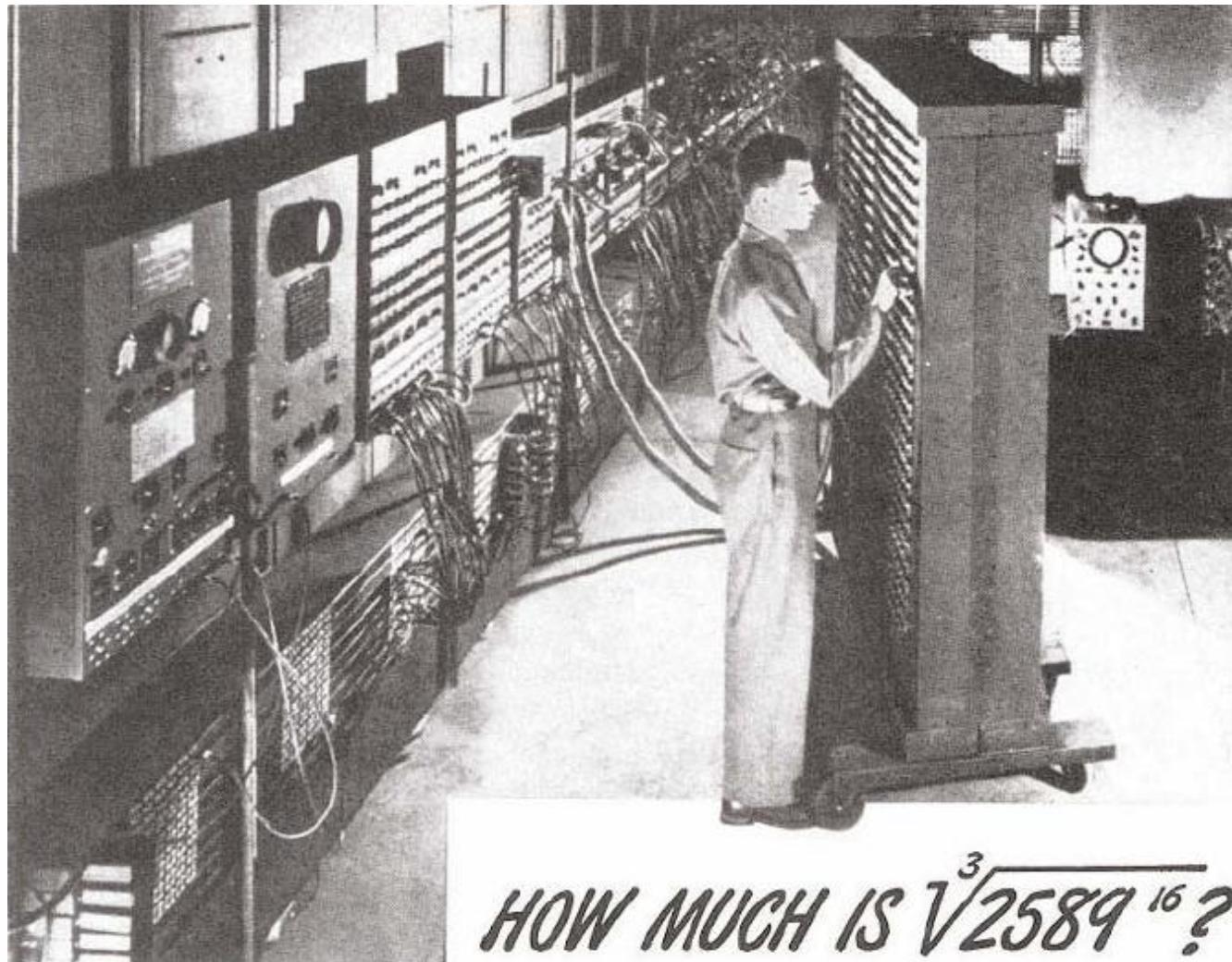


Các thế hệ máy tính

- Thế hệ 1: Máy tính dùng đèn điện tử chân không (1946 - 1955)
- Thế hệ 2: Máy tính dùng transistor (1956 - 1965)
- Thế hệ 3: Máy tính dùng mạch tích hợp (1966 - 1980)
- Thế hệ 4: Máy tính dùng mạch tích hợp VLSI (1981 - nay)



Máy tính dùng đèn chân không

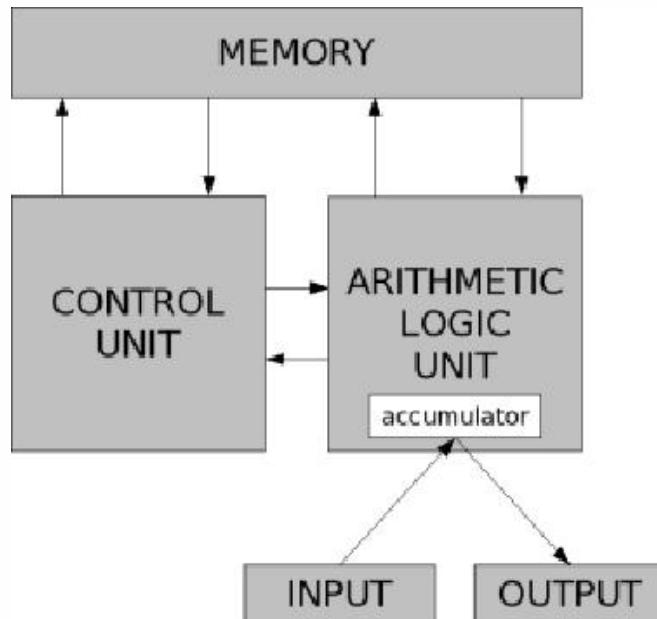


HOW MUCH IS $\sqrt[3]{2589}^{16}$?

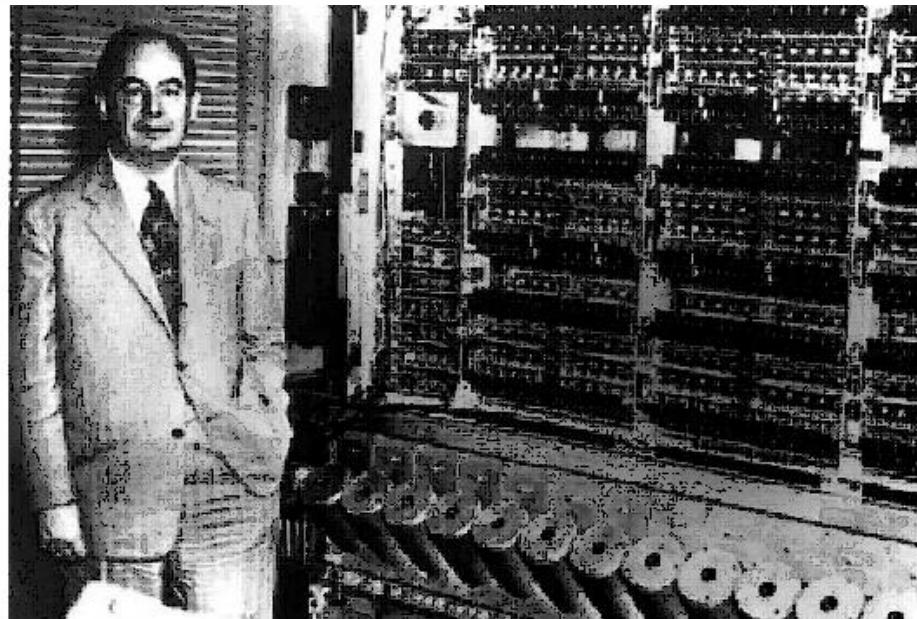


Kiến trúc Von Neumann

Dựa trên ý tưởng **chương trình được lưu trữ** (*stored-program concept*)



Kiến trúc Von Neumann



Máy tính IAS (1945-1952) do Von Neumann thiết kế



Máy tính dùng transistor



Máy PDP-1 và CDC 6600



Máy tính dùng mạch tích hợp

- Mạch tích hợp (Integrated Circuit – IC) hay còn gọi là vi mạch, là các chip bán dẫn trong đó chứa các transistor và các linh kiện khác.
- So với thế hệ trước, các máy tính thế hệ này:
 - Nhỏ gọn hơn
 - Nhanh hơn
 - Tiêu thụ ít năng lượng hơn
 - Rẻ tiền hơn



Siêu máy tính CRAY-1





Máy tính dùng mạch tích hợp VLSI

▪ Các công nghệ mạch tích hợp:

- SSI (Small scale integration) – từ 1965
 - Tích hợp tới 100 transistor trên một chip
- MSI (Medium scale integration) – cho đến 1971
 - Tích hợp từ 100 đến 3,000 transistor trên một chip
- LSI (Large scale integration) – từ 1971 đến 1977
 - Tích hợp từ 3,000 đến 100,000 transistor trên một chip
- VLSI (Very large scale integration) – từ 1978 đến nay
 - Tích hợp từ 100,000 đến 100,000,000 transistor trên một chip
- ULSI (Ultra large scale integration)
 - Có hơn 100,000,000 transistor trên một chip



Máy tính dùng mạch tích hợp VLSI

- **Các sản phẩm của công nghệ VLSI:**
 - **Bộ vi xử lý (Microprocessor):** CPU được chế tạo trên một chip.
 - **Các vi mạch điều khiển tổng hợp (Chipset):** các vi mạch thực hiện được nhiều chức năng điều khiển và nối ghép.
 - **Bộ nhớ bán dẫn,** gồm hai loại: ROM, RAM
 - **Các bộ vi điều khiển (Microcontroller):** máy tính chuyên dụng được chế tạo trên một chip.



Máy tính dùng mạch tích hợp VLSI





3. Sự tiến hóa của máy tính

- Thế hệ 1: Máy tính dùng đèn điện tử chân không (1946 - 1955)
- Thế hệ 2: Máy tính dùng transistor (1956 - 1965)
- Thế hệ 3: Máy tính dùng mạch tích hợp IC (1966 - 1980)
- Thế hệ 4: Máy tính dùng mạch tích hợp VLSI (1981 - nay)
- Thế hệ 5: Máy tính dùng ULSI,



Thế hệ 1: Máy tính dùng đèn điện tử chân không

■ Đặc điểm

- Xây dựng trên cơ sở các đèn điện tử chân không
- Kích thước, trọng lượng và công suất tiêu thụ rất lớn nhưng có tốc độ rất chậm



Ví dụ máy tính ENIAC

- Máy tính điện tử đầu tiên
- Dự án của bộ quốc phòng Mỹ
- Bắt đầu năm 1943, kết thúc năm 1946
- Đặc điểm
 - Nặng 30 tấn,
 - 18.000 đèn điện tử
 - 1500 role,
 - Công suất tiêu thụ 140KW
 - Tốc độ: 5000 phép cộng mỗi giây

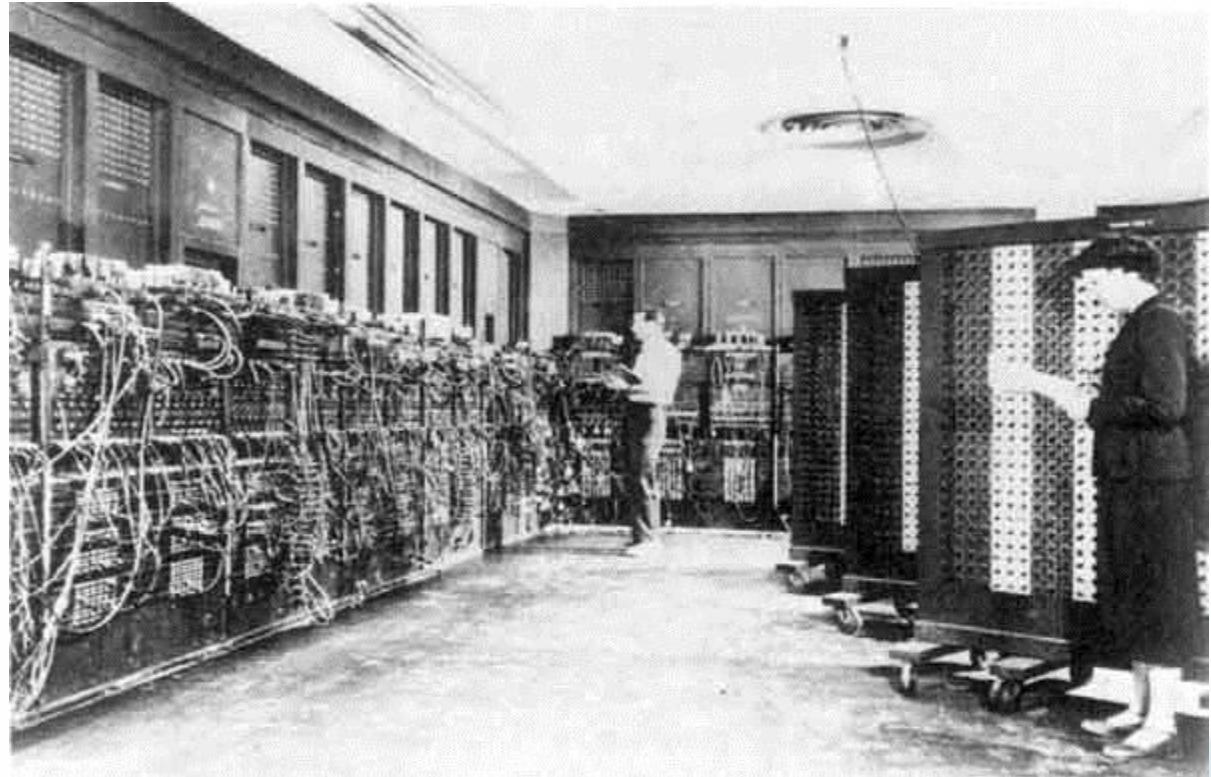


Ví dụ máy tính ENIAC (tiếp)

- Bộ nhớ chỉ lưu trữ dữ liệu
- Lập trình bằng cách thiết lập các chuyển mạch và các cáp nối



Ví dụ máy tính Eniac



- Dài 10m, rộng 3m, cao 3m

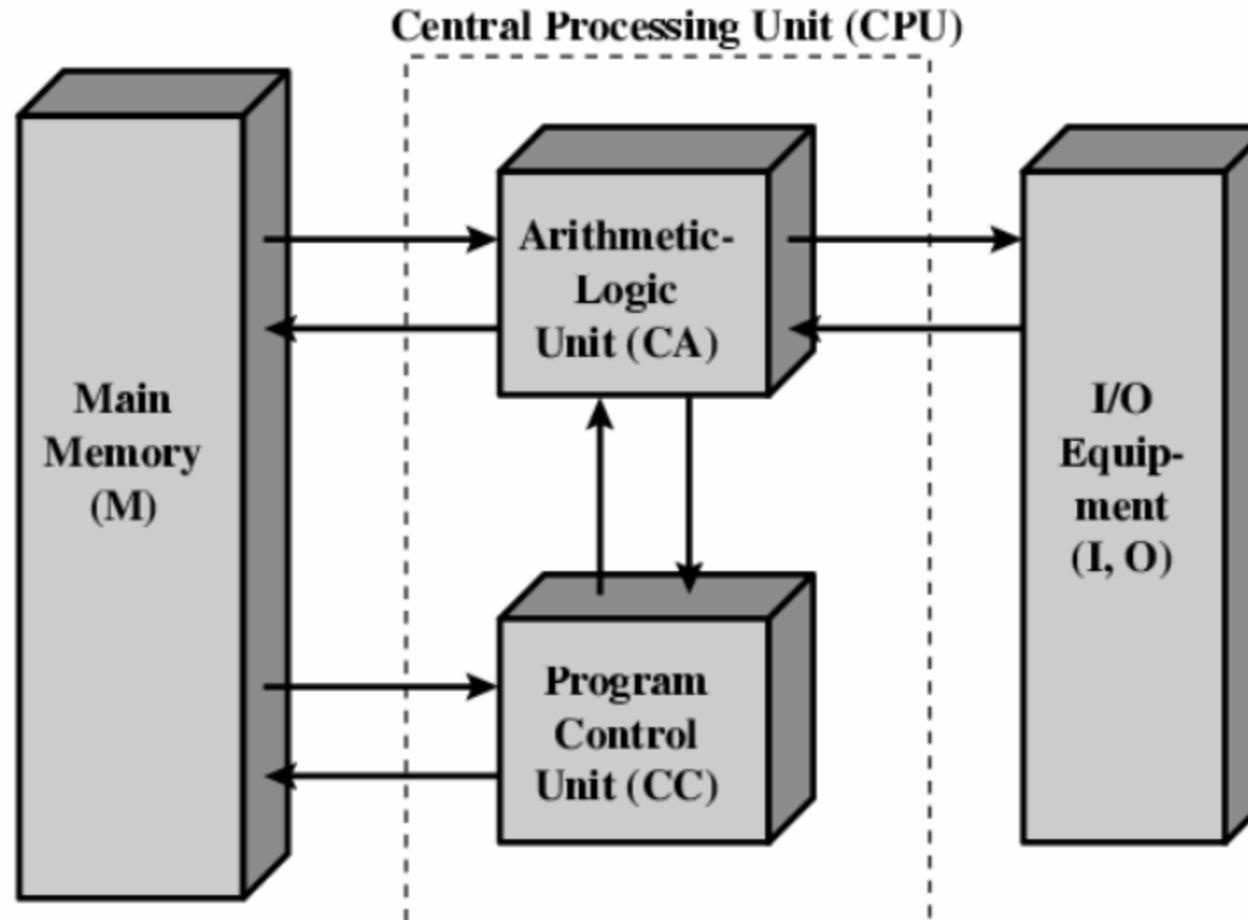


Kiến trúc Von-Neumann

- Khái niệm *nhớ chương trình* (stored program) được đưa ra (1947)
- **Đặc trưng cơ bản:**
 - Dữ liệu và các lệnh (chương trình) được chứa trong một bộ nhớ đọc ghi.
 - Bộ nhớ được đánh địa chỉ theo từng ngăn nhớ, không phụ thuộc vào nội dung của nó.
 - Máy tính thực hiện lệnh một cách tuần tự.



Kiến trúc Von neumann





Kiến trúc Von neumann

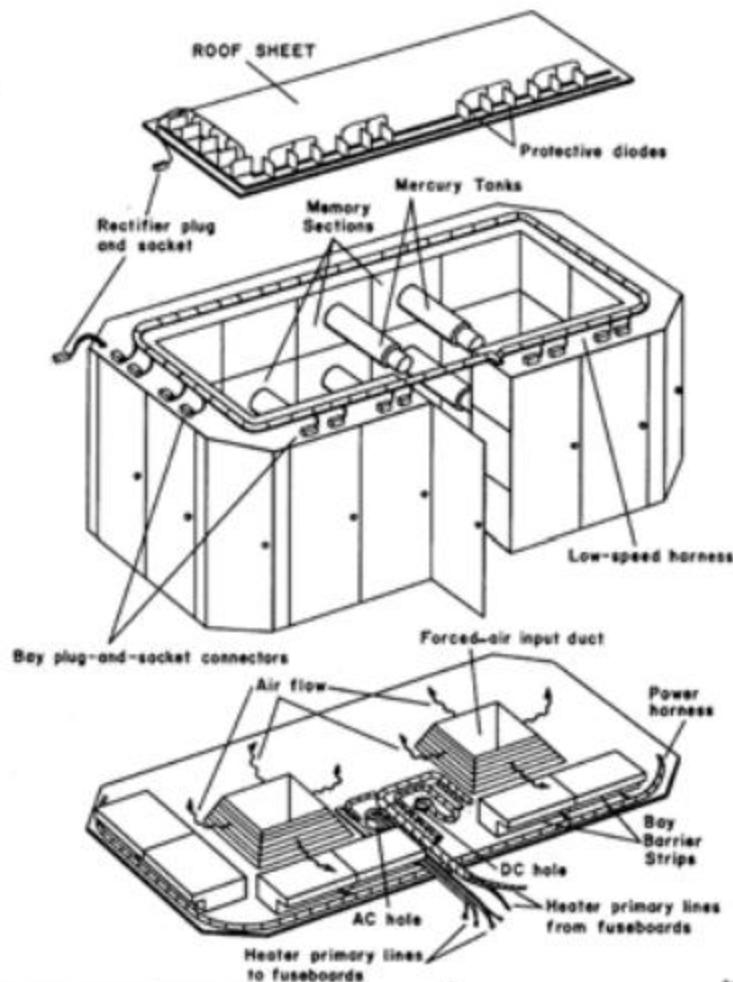
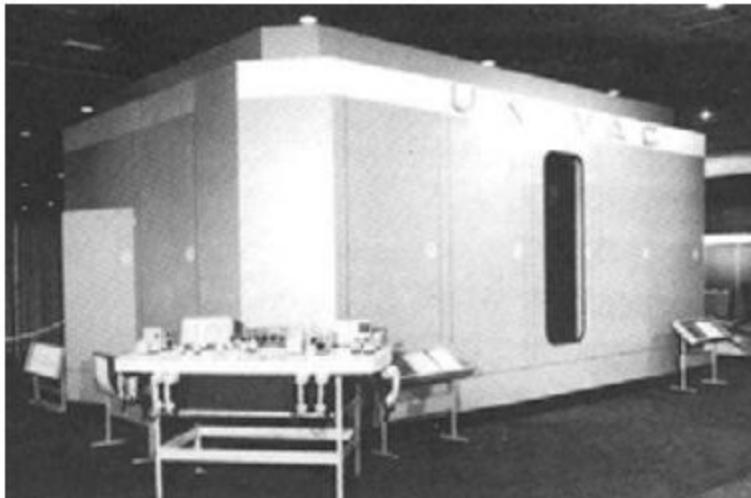
- Trên cơ sở kiến trúc này các máy tính thương mại ra đời
- 1947: UNIVAC 1 (Universal automatic computer)
- 1950s: UNIVAC 2



UNIVAC 1



UNIVAC I



- IBM: International Business Machine
- 1953: ra đời máy tính IBM 701
- 1955: IBM 702
-



IBM 701





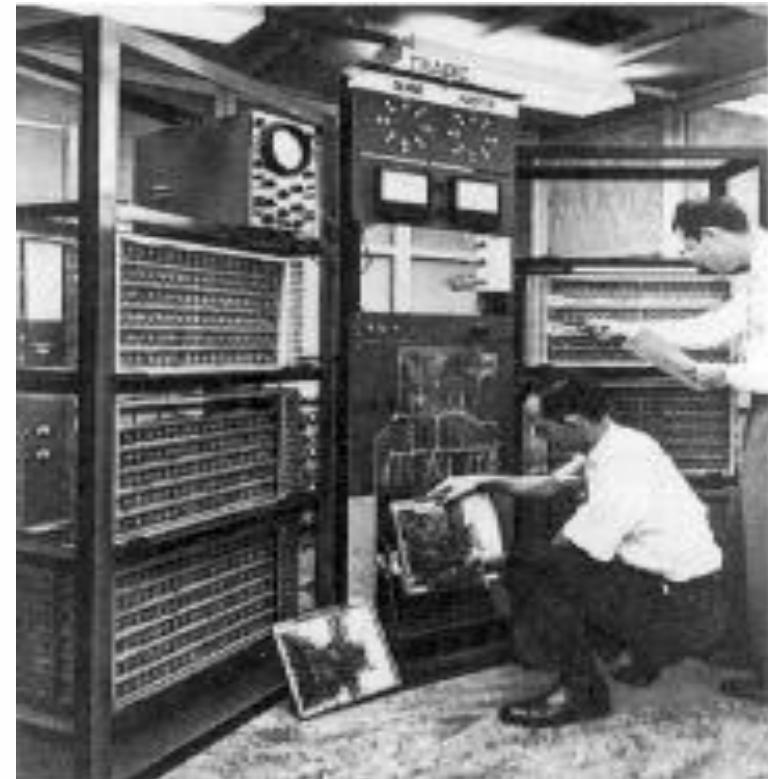
Thế hệ 2: Máy tính dùng transistor

- Trên cơ sở *phát minh ra transistor* ở Bell Labs vào năm 1948.
- Transistor có kích thước nhỏ hơn nhiều, tốc độ nhanh hơn và tiêu thụ năng lượng ít hơn nhiều → thay thế bóng đèn điện tử
- Máy tính thế hệ transistor có khả năng thực hiện hàng trăm nghìn phép tính cộng trong một giây
- Các *ngôn ngữ lập trình bậc cao* ra đời



Máy tính TRADIC

- Máy tính đầu tiên sử dụng hoàn toàn bóng bán dẫn:
 - 8000 transistors
 - Nhanh hơn
 - Nhỏ hơn
 - Rẻ hơn.





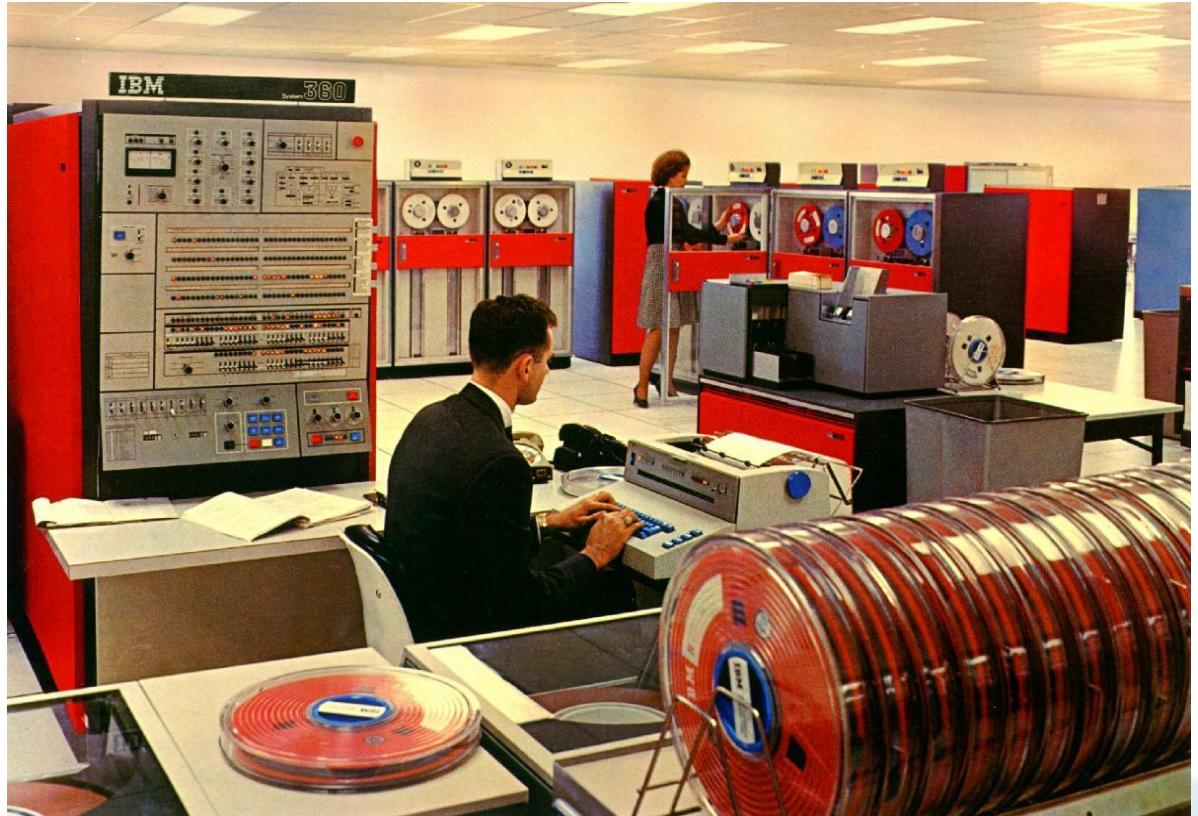
Thế hệ 3: Máy tính dùng mạch tích hợp (1966 - 1980)

- Dựa trên công nghệ **mạch tích hợp**, còn gọi là vi mạch (Integrated Circuit - IC)
- Các vi mạch cỡ SSI/MSI/LSI (small scale, Medium scale, ...)
- Xuất hiện các *siêu máy tính* như CRAY-1, VAX
- Các *bộ vi xử lý* - CPU được chế tạo trên một chip - cũng ra đời và bắt đầu phát triển



Máy tính thế hệ 3

- Hàng tỷ phép toán/s





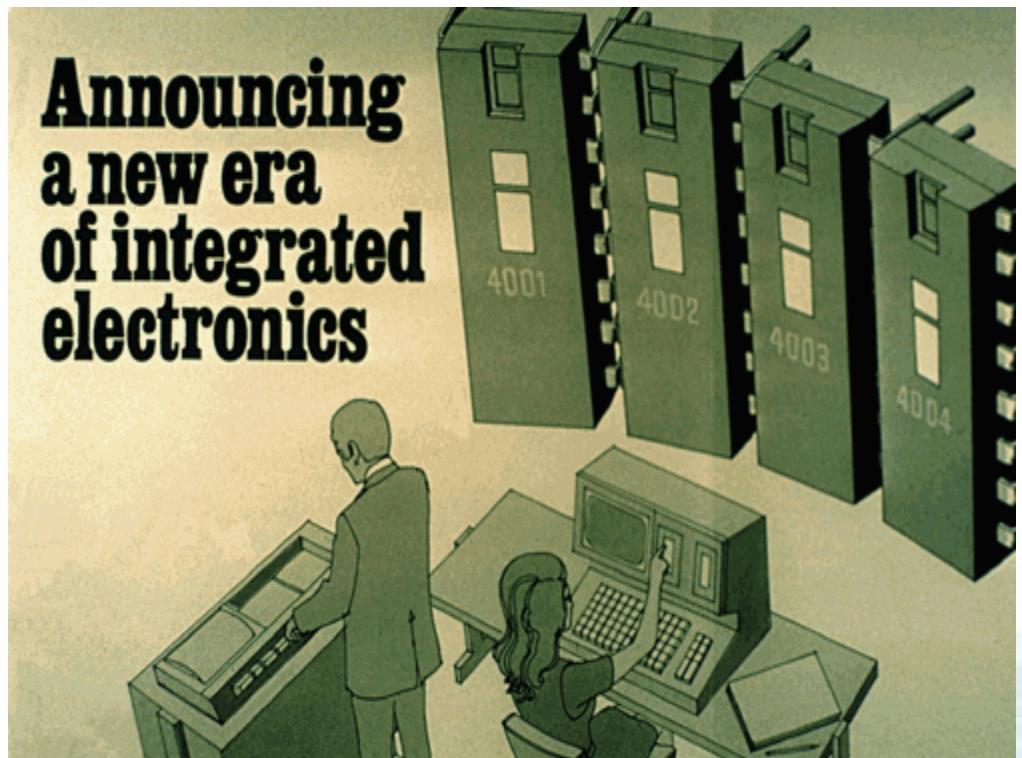
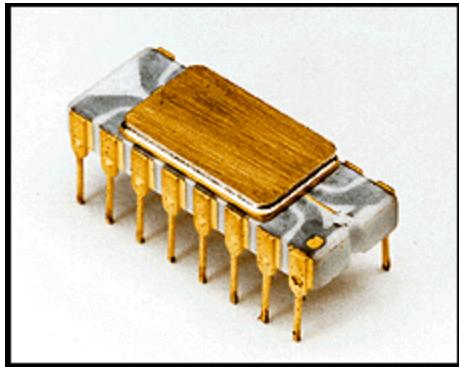
Thế hệ 4: Máy tính dùng mạch tích hợp VLSI (1981 - nay)

- Phát triển trên cơ sở các mạch tích hợp có mật độ tích hợp siêu lớn VLSI.
- Các sản phẩm của công nghệ VLSI:
 - Bộ vi xử lý (Microprocessor): CPU được chế tạo trên một chip
 - Các vi mạch điều khiển tổng hợp (Chipset): các vi mạch thực hiện được nhiều chức năng điều khiển và nối ghép.
 - Bộ nhớ bán dẫn (Semiconductor Memory): ROM, RAM
 - Các bộ vi điều khiển (Microcontroller): máy tính chuyên dụng được chế tạo trên một chip



Thế hệ thứ tư – Vi xử lý (Microprocessor)

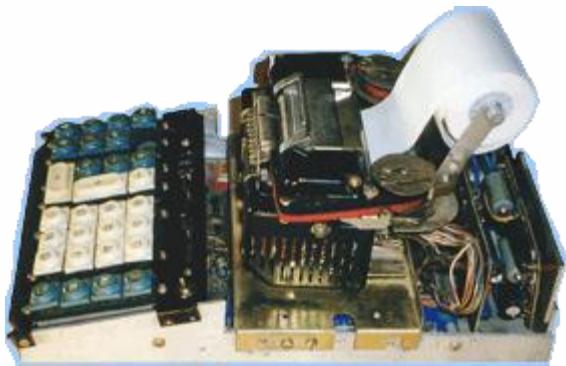
- Microprocessor = Central Processing Unit (CPU) thiết kế trong 1 chip đơn
- 1971 : Intel 4004
- tần số 108KHz , chứa 2300 transistors





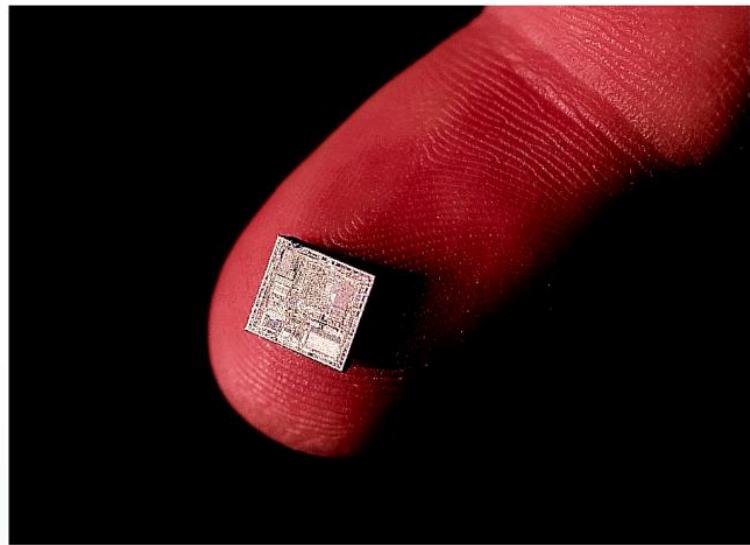
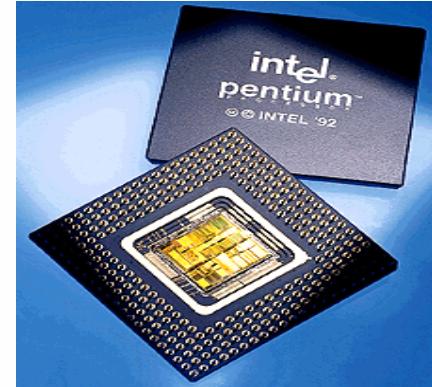
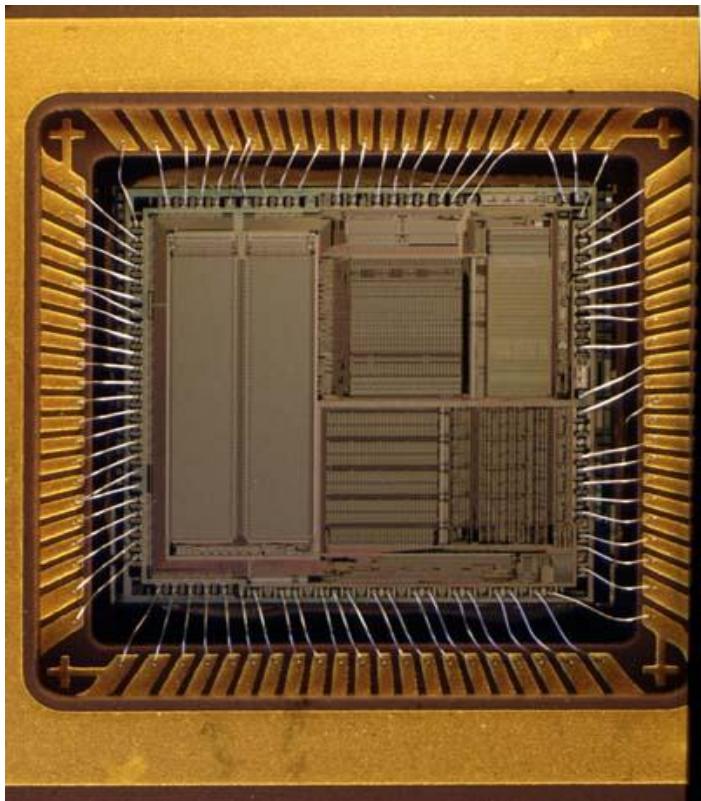
Thế hệ thứ tư – Vi xử lý (Microprocessor)

- Intel Corp. sử dụng chip Intel 4004 trong các máy tính (calculator)





Thế hệ thứ tư – Vi xử lý (Microprocessor)





Giai đoạn 1976 - 1981

Commodore
PET 2001



Tandy TRS-80

Kaypro



Osbourne

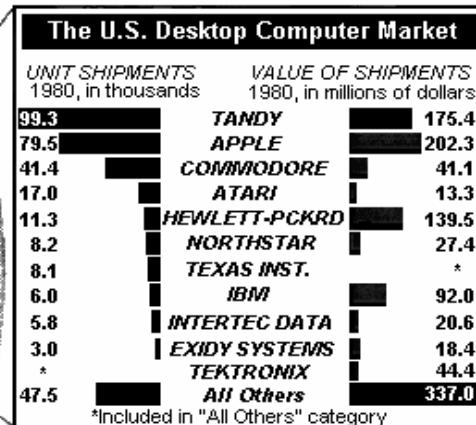
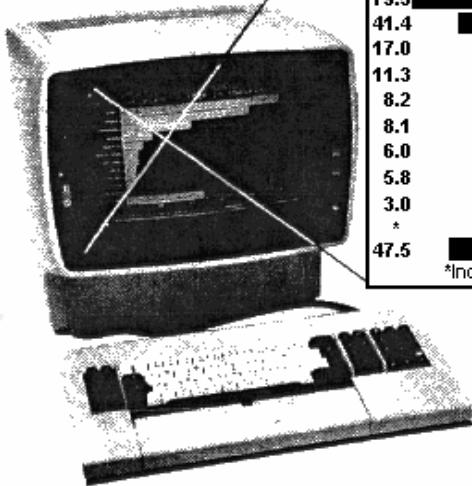


1981 – IBM PC



The New York Times Big I.B.M.'s Little Computer

Its Desk-Top
Model Brings
A New Image



Source: International Data Corporation

Graphic by Bill Aller



Thế hệ máy tính cá nhân mới với kiến trúc mở IBM



1984 – Apple Macintosh

If you can point, you can use a Macintosh.

Introducing Macintosh.

In the olden days, before 1984, not very many people used computers — for a very good reason:

Not very many people knew how.

And not very many people wanted to learn.

After all, in those days it meant listening to your stomach growl in computer seminars. Falling asleep over computer manuals. And staying awake nights to memorize commands so complicated you'd have to be a computer to understand them.

Then, on a particularly bright day in California, some particularly bright engineers had a brilliant idea: since computers are so smart, wouldn't it make sense to teach computers about people, instead of teaching people about computers?

So it was that those very engineers worked long days and late nights — and a few legal holidays — teaching tiny silicon chips all about people. How they make mistakes and change their minds. How they label their file folders and save old phone numbers. How they labor for their livelihoods. And doodle in their spare time.

For the first time in recorded computer history, hardware engineers actually talked to software engineers in a moderate tone of voice. And both became united by a common goal: to build the most powerful, most transportable, most flexible, most versatile computer not-very-much-money could buy.

And when the engineers were finally finished, they introduced us to a personal computer so personable it can practically shake hands.

And so easy to use, most people already know how. They didn't call it the QZ190, or the Zipchip 5000. They called it Macintosh™.



1990 – nay: Personal Computers

- Tốc độ vi xử lý tăng nhanh:
 - CPU 1 lõi,
 - CPU đa lõi
- Kiến trúc ít thay đổi



- Gordon Moore: Người đồng sáng lập intel
- Số lượng transistor trên chip sẽ tăng gấp đôi sau 18 tháng
- Giá thành chip thì hầu như không đổi
- Mật độ cao hơn
- Tốc độ nhanh hơn
- Điện năng tiêu thụ ít hơn
- Tăng độ tin cậy



Sự phát triển của Intel

- 4004:
 - Bộ vi xử lý đầu tiên
 - 4 bít
- 8080:
 - Bộ xử lý đa năng đầu tiên
 - 8 bit
- 8086:
 - 5Mhz – tích hợp 29,000 transistor
 - Bus dữ liệu ngoài: 16 bit
 - 8088: giống với 8080, bus dữ liệu ngoài 8 bít



Sự phát triển của Intel

- 80286:
 - Đánh địa chỉ bộ nhớ được 16Mbyte
- 80386:
 - 32 bit
 - Hỗ trợ đa nhiệm
- 80486
 - Tăng cường bộ nhớ cache
 - Hỗ trợ pipe line
 - Có bộ đồng xử lý toán trên chip



Sự phát triển của intel

- Pentium

- Siêu vô hướng (super scalar)
- Bus dữ liệu 64 bit
- Đa lệnh được thực hiện song song

- Pentium Pro

- Tăng cường chức năng vô hướng
- Dự đoán rẽ nhánh
- Phân tích luồng dữ liệu
- Suy đoán động



Sự phát triển của Intel

- Pentium II
 - Xử lý đồ họa, video, audio
- Pentium III
 - Thêm các lệnh xử lý dấu chấm động cho đồ họa 3D
- Pentium IV
 - Tăng cường xử lý dấu chấm động và multimedia
- Dual core:
 - 2 bộ xử lý trên 1 chip
- Core 2 dual: Kiến trúc 64 bit
- Core 2 quad: 4 bộ xử lý trên chip: tích hợp 820,000,000 transistor



Máy tính thế hệ 5

- Các máy tính thông minh, có khả năng “tư duy” như bộ óc con người



Xu hướng ngày nay



- Nhanh hơn
- Nhỏ hơn
- Rẻ hơn
- Dễ sử dụng hơn

Chương 2

BIỂU DIỄN DỮ LIỆU & SỐ HỌC MÁY TÍNH



2.1. Các hệ đếm cơ bản

2.2. Mã hóa và lưu trữ dữ liệu trong máy tính

2.3. Biểu diễn số nguyên

2.4. Các phép toán số học với số nguyên

2.5. Biểu diễn số thực

2.6. Biểu diễn kí tự



Các hệ đếm cơ bản

- Về mặt toán học, ta có thể biểu diễn số theo hệ đếm cơ số bất kì.
- Khi nghiên cứu về máy tính, ta chỉ quan tâm đến các hệ đếm sau đây:
 - Hệ thập phân (Decimal System) → con người sử dụng
 - Hệ nhị phân (Binary System) → máy tính sử dụng
 - Hệ mươi sáu (Hexadecimal System) → dùng để viết gọn cho số nhị phân

- Sử dụng 10 chữ số: 0,1,2,3,4,5,6,7,8,9 để biểu diễn số
- Dùng n chữ số thập phân có thể biểu diễn được 10^n giá trị khác nhau:
 - $00\dots000 = 0$
 -
 - $99\dots999 = 10^n - 1$
- Giả sử một số A được biểu diễn dưới dạng:
 - $A = a_n a_{n-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}$
- → Giá trị của A được hiểu như sau:

$$A = a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \dots + a_{-m} 10^{-m}$$

$$A = \sum_{i=-m}^n a_i 10^i$$

- Số thập phân 472.38 có giá trị được hiểu như sau:

$$472.38 = 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 8 \times 10^{-2}$$



Mở rộng cho hệ cơ số r ($r > 1$)

- Sử dụng r chữ số có giá trị riêng từ 0 đến $r-1$ để biểu diễn số
- Giả sử có số A được biểu diễn bằng các chữ số của hệ đếm theo cơ số r như sau:
 - $A = a_n a_{n-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}$
- Giá trị của A là:

$$A = a_n r^n + a_{n-1} r^{n-1} + \dots + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} + \dots + a_{-m} r^{-m}$$

$$A = \sum_{i=-m}^n a_i r^i$$

- Một chuỗi n chữ số của hệ đếm cơ số r sẽ biểu diễn được r^n giá trị khác nhau.



Hệ nhị phân

- Sử dụng 2 chữ số: 0,1
- Chữ số nhị phân gọi là **bit** (**binary digit**)
- Bit là đơn vị thông tin nhỏ nhất
- Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau:
 - $00\dots000 = 0$
 - ...
 - $11\dots111 = 2^n - 1$
- Giả sử có số A được biểu diễn theo hệ nhị phân như sau:
$$A = a_n a_{n-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}$$
- Với ai là các chữ số nhị phân, khi đó giá trị của A là:

$$A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + a_{-2} 2^{-2} + \dots + a_{-m} 2^{-m}$$

$$A = \sum_{i=-m}^n a_i 2^i$$

- Số nhị phân 1101001.1011 có giá trị được xác định như sau:

$$\begin{aligned}1101001.1011_{(2)} &= 2^6 + 2^5 + 2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} \\&= 64 + 32 + 8 + 1 + 0.5 + 0.125 + 0.0625 = 105.6875_{(10)}\end{aligned}$$



Đổi số thập phân sang nhị phân

- Thực hiện chuyển đổi phần nguyên và phần lẻ riêng.
- Chuyển đổi phần nguyên:**
 - Cách 1: chia dần số đó cho 2, xác định các phần dư, rồi viết các số dư theo chiều ngược lại.

▪ Ví dụ: chuyển đổi $105_{(10)}$ sang hệ nhị phân ta làm như sau:

$$\begin{array}{rcl} 105 : 2 & = 52 & \text{dư} \quad 1 \\ 52 : 2 & = 26 & \text{dư} \quad 0 \\ 26 : 2 & = 13 & \text{dư} \quad 0 \\ 13 : 2 & = 6 & \text{dư} \quad 1 \\ 6 : 2 & = 3 & \text{dư} \quad 0 \\ 3 : 2 & = 1 & \text{dư} \quad 1 \\ 1 : 2 & = 0 & \text{dư} \quad 1 \end{array}$$

Như vậy, ta có: $105_{(10)} = 1101001_{(2)}$



Đổi số thập phân sang nhị phân

- Chuyển đổi phần nguyên (tiếp):
 - Cách 2: phân tích số đó thành tổng các lũy thừa của 2, sau đó dựa vào các số mũ để xác định dạng biểu diễn nhị phân.
 - Ví dụ: $105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$
 $\rightarrow 105_{(10)} = 1101001_{(2)}$
- Chuyển đổi phần lẻ:
 - Nhân phần lẻ với 2 rồi lấy phần nguyên ... Sau đó viết các phần nguyên theo chiều thuận.
 - Ví dụ: chuyển đổi số $0.6875_{(10)}$ sang hệ nhị phân:

$$\begin{array}{rcl} 0.6875 \times 2 & = & 1.3750 \text{ phần nguyên } = 1 \\ 0.375 \times 2 & = & 0.750 \text{ phần nguyên } = 0 \\ 0.75 \times 2 & = & 1.50 \text{ phần nguyên } = 1 \\ 0.5 \times 2 & = & 1.0 \text{ phần nguyên } = 1 \end{array}$$

Kết quả là: $0.6875_{(10)} = 0.1011_{(2)}$



3. Hệ mười sáu (Hexa)

- Sử dụng 16 chữ số, kí hiệu như sau:
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Dùng để viết gọn cho số nhị phân.

Hệ thập phân	Hệ nhị phân	Hệ mười sáu
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F



Một số ví dụ

- Nhị phân → Hexa:
- Hexa → Nhị phân:
- Thập phân → Hexa:

$$11\ 1011\ 1110\ 0110_{(2)} = 3BE6_{(16)}$$

$$3E8_{(16)} = 11\ 1110\ 1000_{(2)}$$

$$14988 \rightarrow ?$$

$$14988 : 16 = 936 \quad \text{dư} \quad 12 \text{ tức là C}$$

$$936 : 16 = 58 \quad \text{dư} \quad 8$$

$$58 : 16 = 3 \quad \text{dư} \quad 10 \text{ tức là A}$$

$$3 : 16 = 0 \quad \text{dư} \quad 3$$

Như vậy, ta có: $14988_{(10)} = 3A8C_{(16)}$

- Hexa → Thập phân: $3A8C \rightarrow ?$

$$3A8C_{(16)} = 3 \times 16^3 + 10 \times 16^2 + 8 \times 16^1 + 12 \times 16^0$$

$$= 12288 + 2560 + 128 + 12 = 14988_{(10)}$$



Cộng trừ số Hexa

$$\begin{array}{r} + \\ \hline 8A9B \\ 37CD \\ \hline C268 \end{array}$$

$$\begin{array}{r} - \\ \hline B46E \\ 1AC9 \\ \hline 99A5 \end{array}$$

$$\begin{array}{r} + \\ \hline B7E5 \\ 2AF9 \\ \hline E2DE \end{array}$$

$$\begin{array}{r} - \\ \hline FA9D \\ 2BC5 \\ \hline CED8 \end{array}$$

$$\begin{array}{r} + \\ \hline B800 \\ 0FFF \\ \hline \end{array}$$

$$\begin{array}{r} - \\ \hline 8E9A \\ 3FE2 \\ \hline \end{array}$$

$$\begin{array}{r} + \\ \hline 1234 \\ ABCD \\ \hline \end{array}$$

$$\begin{array}{r} + \\ \hline CFFF \\ 1FFF \\ \hline \end{array}$$

$$\begin{array}{r} - \\ \hline A78D \\ 45FB \\ \hline \end{array}$$

$$\begin{array}{r} + \\ \hline 879D \\ 5DF8 \\ \hline \end{array}$$



Nội dung chương 2

- 2.1. Các hệ đếm cơ bản
- 2.2. Mã hóa và lưu trữ dữ liệu trong máy tính
- 2.3. Biểu diễn số nguyên
- 2.4. Các phép toán số học với số nguyên
- 2.5. Biểu diễn số thực
- 2.6. Biểu diễn kí tự



Mã hóa và lưu trữ dữ liệu

1. Nguyên tắc chung về mã hóa dữ liệu
2. Lưu trữ thông tin trong bộ nhớ chính



1. Nguyên tắc chung về mã hóa dữ liệu

- Mọi dữ liệu đưa vào máy tính đều phải được mã hóa thành số nhị phân.
- Các loại dữ liệu :
 - Dữ liệu nhân tạo: do con người quy ước
 - Dữ liệu tự nhiên: tồn tại khách quan với con người





Nguyên tắc mã hóa dữ liệu

- **Mã hóa dữ liệu nhân tạo:**

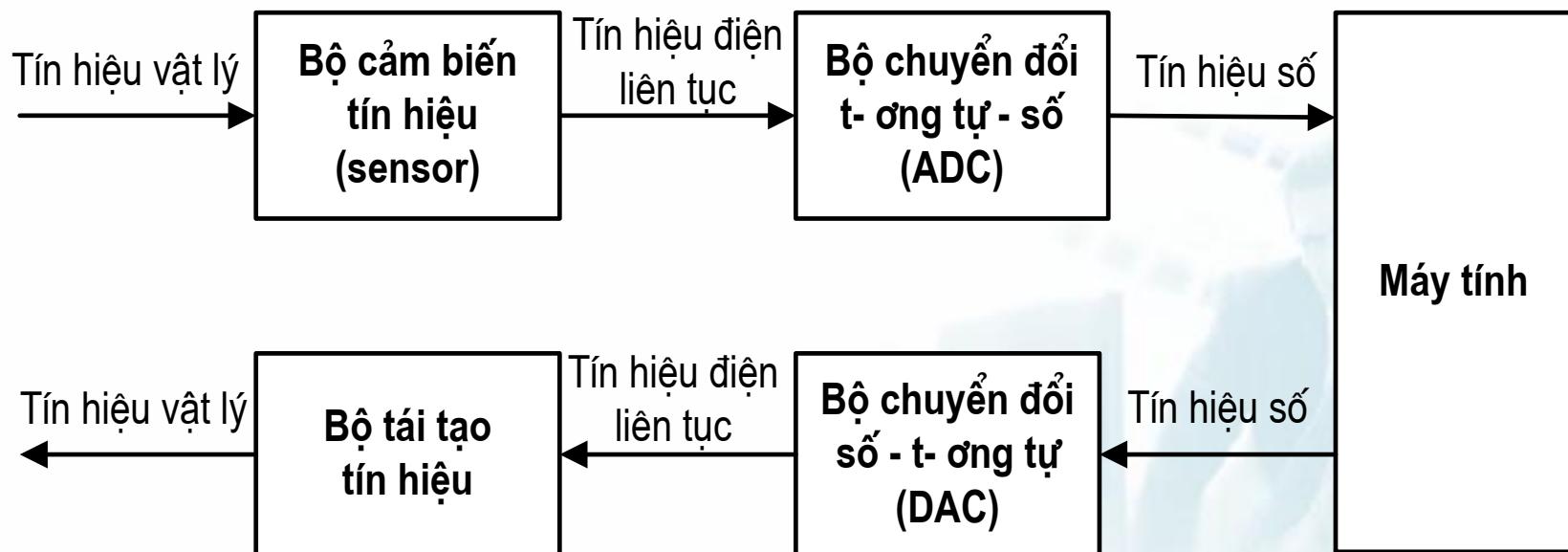
- Dữ liệu số nguyên: mã hóa theo chuẩn qui ước
- Dữ liệu số thực: mã hóa bằng số dấu chấm động
- Dữ liệu ký tự: mã hóa theo bộ mã ký tự



Nguyên tắc mã hóa dữ liệu (tiếp)

■ Mã hóa dữ liệu tự nhiên:

- Phổ biến là các tín hiệu vật lý như âm thanh, hình ảnh, ...
- Các dữ liệu tự nhiên cần phải được số hóa (digitalized) trước khi đưa vào trong máy tính.
- Sơ đồ mã hóa và tái tạo tín hiệu vật lý:





Độ dài từ dữ liệu

- Độ dài từ dữ liệu:

- Là số bit được sử dụng để mã hóa loại dữ liệu tương ứng
- Trong thực tế, độ dài từ dữ liệu thường là bội số của 8 bit, ví dụ: 8, 16, 32, 64 bit

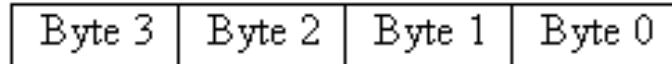


2. Lưu trữ thông tin trong bộ nhớ chính

- Bộ nhớ chính thường được tổ chức theo Byte
- Độ dài từ dữ liệu có thể chiếm 1 hoặc nhiều Byte
- Cần phải biết thứ tự lưu trữ các byte trong bộ nhớ chính:
 - Lưu trữ kiểu đầu nhỏ (Little-endian)
 - Lưu trữ kiểu đầu to (Big-endian)
- Little-endian: Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ nhỏ hơn.
- Big-endian: Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ lớn hơn.



Ví dụ



Dữ liệu thông tin

.....	
Byte 0	i
Byte 1	i+1
Byte 2	i+2
Byte 3	i+3
.....	

*Little-endian
(Intel)*

.....	
Byte 3	i
Byte 2	i+1
Byte 1	i+2
Byte 0	i+3
.....	

*Big-endian
(Motorola, RISC)*

- Intel 80x86, Pentium: Little-endian
- Motorola 680x0, các bộ xử lý RISC: Big-endian
- Power PC, Itanium: hỗ trợ cả hai (Bi-endian)

- Dữ liệu 16 bit có giá trị là 5B9D được lưu trữ vào bộ nhớ chính tổ chức theo kiểu Little-endian bắt đầu từ byte nhớ có địa chỉ là 1234. Hãy xác định nội dung các byte nhớ chứa lưu trữ dữ liệu đó dưới dạng nhị phân.

- 2.1. Các hệ đếm cơ bản
- 2.2. Mã hóa và lưu trữ dữ liệu trong máy tính
- 2.3. Biểu diễn số nguyên**
- 2.4. Các phép toán số học với số nguyên
- 2.5. Biểu diễn số thực
- 2.6. Biểu diễn kí tự



Biểu diễn số nguyên

1. Số nguyên không dấu
2. Số nguyên có dấu
3. Biểu diễn số nguyên theo mã BCD



1. Số nguyên không dấu

- Dạng tổng quát: giả sử dùng n bit để biểu diễn cho một số nguyên không dấu A:

$$a_{n-1}a_{n-2}\dots a_3a_2a_1a_0$$

- Giá trị của A được tính như sau:

$$A = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \dots + a_12^1 + a_02^0$$

$$A = \sum_{i=0}^{n-1} a_i 2^i$$

- Dải biểu diễn của A: từ 0 đến $2^n - 1$

- Ví dụ 1. Biểu diễn các số nguyên không dấu sau đây bằng 8 bit:

$$A = 45$$

$$B = 156$$

Giải:

$$A = 45 = 32 + 8 + 4 + 1 = 2^5 + 2^3 + 2^2 + 2^0$$

$$\rightarrow A = 0010\ 1101$$

$$B = 156 = 128 + 16 + 8 + 4 = 2^7 + 2^4 + 2^3 + 2^2$$

$$\rightarrow B = 1001\ 1100$$



Các ví dụ (tiếp)

- Ví dụ 2. Cho các số nguyên không dấu X, Y được biểu diễn bằng 8 bit như sau:

$$X = 0010\ 1011$$

$$Y = 1001\ 0110$$

Giải:

$$\begin{aligned}X &= 0010\ 1011 = 2^5 + 2^3 + 2^1 + 2^0 \\&= 32 + 8 + 2 + 1 = 43\end{aligned}$$

$$\begin{aligned}Y &= 1001\ 0110 = 2^7 + 2^4 + 2^2 + 2^1 \\&= 128 + 16 + 4 + 2 = 150\end{aligned}$$



Trường hợp cụ thể: với $n = 8$ bit

- Dải biểu diễn là $[0, 255]$

$$0000\ 0000 = 0$$

$$0000\ 0001 = 1$$

$$0000\ 0010 = 2$$

$$0000\ 0011 = 3$$

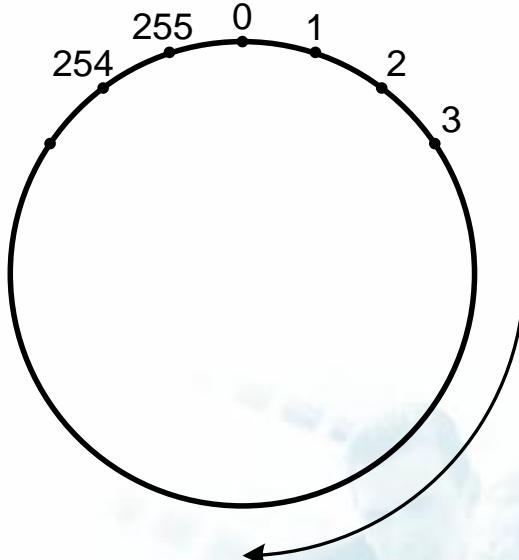
.....

$$1111\ 1111 = 255$$

- Trục số học:



- Trục số học máy tính:



- Kiểu dữ liệu tương ứng trong Turbo C là kiểu unsigned char.
- Ví dụ:

unsigned char a;

a = 255;

a = a + 1;

printf("%d",a); //Kết quả sai là 0

$$\begin{array}{r} 1111\ 1111 \\ + 0000\ 0001 \\ \hline 1\ 0000\ 0000 \end{array}$$

KQ sai: $255 + 1 = 0$?
(do phép cộng bị nhór ra
ngoài)



Với n = 16 bit, 32 bit, 64 bit

- n = 16 bit:

- Dải biểu diễn là [0, 65535]
- Kiểu dữ liệu tương ứng trong Turbo C là kiểu unsigned int
- Ví dụ:

```
unsigned int a;
```

```
a = 0xffff;
```

```
a = a + 1;
```

```
printf("%d",a);
```

- n = 32 bit:

- Dải biểu diễn là [0, $2^{32}-1$]

- n = 64 bit:

- Dải biểu diễn là [0, $2^{64}-1$]



2. Số nguyên có dấu

a. Khái niệm về số bù

- **Số bù chín và số bù mười (hệ thập phân):**
 - Giả sử có một số nguyên thập phân A được biểu diễn bởi n chữ số thập phân. Khi đó ta có:
 - Số bù chín của A = $(10^n - 1) - A$
 - Số bù mươi của A = $10^n - A$
 - NX: Số bù mươi = Số bù chín + 1
 - Ví dụ:
 - Xét n = 4 chữ số, A = 2874
 - Số bù chín của A = $(10^4 - 1) - 2874 = 7125$
 - Số bù mươi của A = $10^4 - 2874 = 7126$



Khái niệm về số bù

- Số bù một và số bù hai (hệ nhị phân):
 - Giả sử có một số nguyên nhị phân A được biểu diễn bởi n bit. Khi đó ta có:
 - Số bù một của A = $(2^n - 1) - A$
 - Số bù hai của A = $2^n - A$
 - NX: Số bù hai = Số bù một + 1
 - Ví dụ:
 - Xét n = 4 bit, A = 0110
 - Số bù một của A = $(2^4 - 1) - 0110 = 1001$
 - Số bù hai của A = $2^4 - 0110 = 1010$

- Có thể tìm số bù một của A bằng cách đảo tất cả các bit của A
- Số bù hai của A = Số bù một của A + 1

Ví dụ:

cho A = 0110 0101

Số bù một của A = 1001 1010
+ 1

Số bù hai của A = 1001 1011

Nhận xét

$$A = 0110\ 0101$$

$$\text{Số bù hai của } A \quad + = 1001\ 1011$$

$$1\ 0000\ 0000 = 0 \text{ (bỏ qua bit nhớ ra ngoài)}$$

-> Số bù hai của A = -A



Biểu diễn số nguyên có dấu

b. Biểu diễn số nguyên có dấu bằng số bù hai

- Dùng n bit biểu diễn số nguyên có dấu A:

$$a_{n-1}a_{n-2}\dots a_2a_1a_0$$

- Với số dương:

- Bit $a_{n-1} = 0$

- Các bit còn lại biểu diễn độ lớn của số dương đó

- Dạng tổng quát của số dương: $0a_{n-2}\dots a_2a_1a_0$

- Giá trị của số dương:

$$A = \sum_{i=0}^{n-2} a_i 2^i$$

- Dải biểu diễn của số dương: $[0, 2^{n-1}-1]$



Biểu diễn số nguyên có dấu (tiếp)

- Với số âm:

- Được biểu diễn bằng số bù hai của số dương tương ứng
 - \rightarrow Bit $a_{n-1} = 1$

- Dạng tổng quát của số âm: $1a_{n-2}...a_2a_1a_0$
 - Giá trị của số âm:

$$A = -2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

- Dải biểu diễn của số âm: $[-2^{n-1}, -1]$
- Dải biểu diễn của số nguyên có dấu n bit là $[-2^{n-1}, 2^{n-1}-1]$



Biểu diễn số nguyên có dấu (tiếp)

- Dạng tổng quát của số nguyên có dấu A:

$$a_{n-1}a_{n-2}\dots a_2a_1a_0$$

- Giá trị của A được xác định như sau:

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

- Dải biểu diễn: $[-2^{n-1}, 2^{n-1}-1]$

- Ví dụ 1. Biểu diễn các số nguyên có dấu sau đây bằng 8 bit
 $A = +50$ $B = -70$

Giải:

$$A = +50 = 32 + 16 + 2 = 2^5 + 2^4 + 2^1$$
$$\rightarrow A = 0011\ 0010$$

$$B = -70$$

Ta có: $+70 = 64 + 4 + 2 = 2^6 + 2^2 + 2^1$

$$+70 = 0100\ 0110$$

$$\text{Số bù } 1 = 1011\ 1001$$

$$\begin{array}{r} + \\ \hline \end{array}$$
$$\text{Số bù } 2 = \overline{1011\ 1010}$$

$$\rightarrow B = 1011\ 1010$$

- Ví dụ 2. Xác định giá trị của các số nguyên có dấu 8 bit sau đây:

$$A = 0101\ 0110$$

$$B = 1101\ 0010$$

Giải:

$$A = 2^6 + 2^4 + 2^2 + 2^1 = 64 + 16 + 4 + 2 = +86$$

$$B = -2^7 + 2^6 + 2^4 + 2^1 = -128 + 64 + 16 + 2 = -46$$



Trường hợp cụ thể: với $n = 8$ bit

- Dải biểu diễn là $[-128, +127]$
- Trục số học máy tính:

$$0000\ 0000 = 0$$

$$0000\ 0001 = +1$$

$$0000\ 0010 = +2$$

.....

$$0111\ 1111 = +127$$

$$1000\ 0000 = -128$$

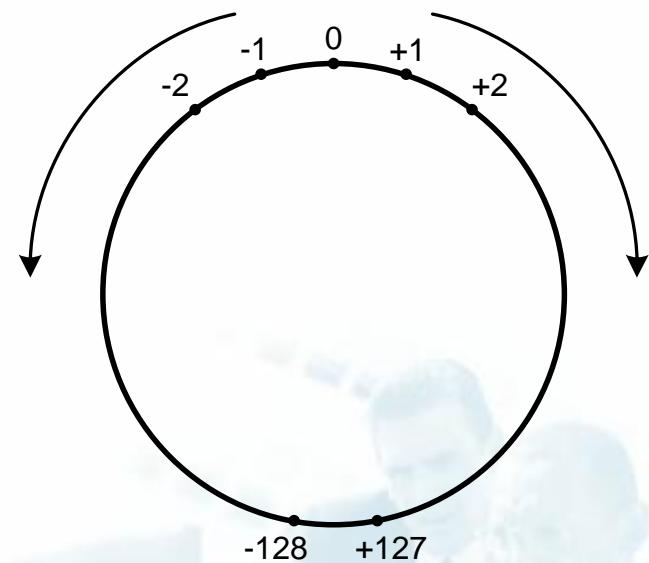
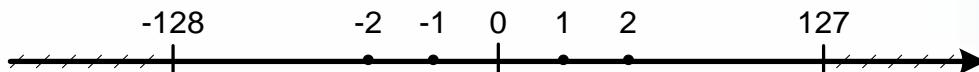
$$1000\ 0001 = -127$$

.....

$$1111\ 1110 = -2$$

$$1111\ 1111 = -1$$

- Trục số học:





Với n = 8 bit (tiếp)

- Kiểu dữ liệu tương ứng trong Turbo C là kiểu char.
- Ví dụ:

char a;

a = 127;

a = a + 1;

printf("%d",a); //Kết quả sai là -128

0111 1111

+ 0000 0001

—————

1000 0000

KQ sai: $127 + 1 = -128$?
(do phép cộng bị tràn số
học)



Với $n = 16$ bit, 32 bit, 64 bit

- $n = 16$ bit:
 - Dải biểu diễn là $[-32768, +32767]$
 - Kiểu dữ liệu tương ứng trong Turbo C là kiểu int
- $n = 32$ bit:
 - Dải biểu diễn là $[-2^{31}, 2^{31}-1]$
 - Kiểu dữ liệu tương ứng trong Turbo C là kiểu long int
- $n = 64$ bit:
 - Dải biểu diễn là $[-2^{63}, 2^{63}-1]$



Chuyển từ 8 bit sang 16 bit

- Với số dương:

$+35 = \underline{0010\ 0011}$ (8 bit)

$+35 = \underline{0000\ 0000}\ 0010\ 0011$ (16 bit)

→ Thêm 8 bit 0 vào bên trái

- Với số âm:

$-79 = \underline{1011\ 0001}$ (8 bit)

$-79 = \underline{1111\ 1111}\ 1011\ 0001$ (16 bit)

→ Thêm 8 bit 1 vào bên trái

- Kết luận: mở rộng sang bên trái 8 bit bằng bit dấu



3. Biểu diễn số nguyên theo mã BCD

- BCD – Binary Coded Decimal (Mã hóa số nguyên thập phân bằng nhị phân)
- Dùng 4 bit để mã hóa cho các chữ số thập phân từ 0 đến 9

$0 \rightarrow 0000$

$5 \rightarrow 0101$

$1 \rightarrow 0001$

$6 \rightarrow 0110$

$2 \rightarrow 0010$

$7 \rightarrow 0111$

$3 \rightarrow 0011$

$8 \rightarrow 1000$

$4 \rightarrow 0100$

$9 \rightarrow 1001$

- Có 6 tổ hợp không sử dụng:

$1010, 1011, 1100, 1101, 1110, 1111$



Ví dụ về số BCD

- $35 \rightarrow 0011\ 0101_{BCD}$
- $79 \rightarrow 0111\ 1001_{BCD}$
- $2281 \rightarrow 0010\ 0010\ 1000\ 0001_{BCD}$
- $1304 \rightarrow 0001\ 0011\ 0000\ 0100_{BCD}$



Phép cộng số BCD

- $$\begin{array}{r}
 35 \rightarrow 0011\ 0101_{BCD} \\
 + \underline{24} \rightarrow + \underline{0010\ 0100}_{BCD} \\
 59 \leftarrow 0101\ 1001_{BCD}
 \end{array}$$

Kết quả đúng (không phải hiệu chỉnh)
 - $$\begin{array}{r}
 89 \rightarrow 1000\ 1001_{BCD} \\
 + \underline{52} \rightarrow + \underline{0101\ 0010}_{BCD} \\
 141 \quad 1101\ 1011 \rightarrow \text{kết quả sai} \\
 \quad \quad + \underline{0110\ 0110} \quad \leftarrow \text{hiệu chỉnh} \\
 0001\ 0100\ 0001_{BCD} \rightarrow \text{kết quả đúng}
 \end{array}$$

1 4 1
 - Hiệu chỉnh: cộng thêm 6 ở những hàng chẵn



Các kiểu lưu trữ số BCD

- BCD dạng nén (Packed BCD): Hai số BCD được lưu trữ trong 1 Byte.
 - Ví dụ số 52 được lưu trữ như sau:
- BCD dạng không nén (Unpacked BCD): Mỗi số BCD được lưu trữ trong 4 bit thấp của mỗi Byte.
 - Ví dụ số 52 được lưu trữ như sau:

0101	0010
------	------

	0101		0010
--	------	--	------



Nội dung chương 2

- 2.1. Các hệ đếm cơ bản
- 2.2. Mã hóa và lưu trữ dữ liệu trong máy tính
- 2.3. Biểu diễn số nguyên
- 2.4. Các phép toán số học với số nguyên**
- 2.5. Biểu diễn số thực
- 2.6. Biểu diễn kí tự



Các phép toán số học với số nguyên

1. Bộ cộng
2. Cộng số nguyên không dấu
3. Cộng/trừ số nguyên có dấu
4. Nhân số nguyên
5. Chia số nguyên



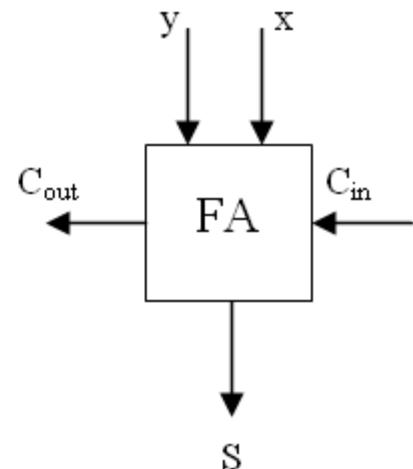


1. Bộ cộng

- Bộ cộng 1 bit toàn phần (Full Adder)

Bảng thật:

x	y	C _{in}	S	C _{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1



$$S = x \oplus y \oplus C_{in}$$

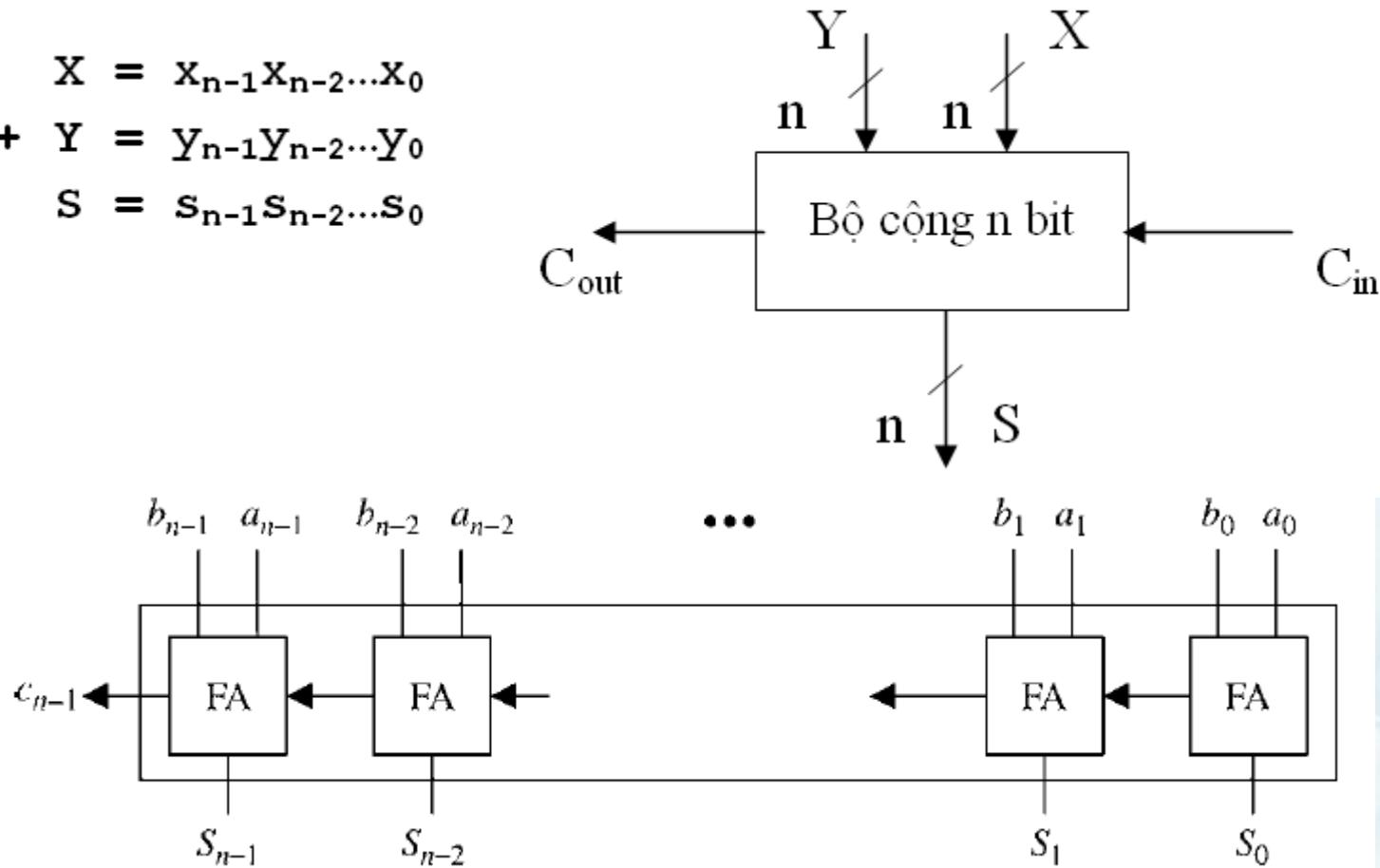
$$C_{out} = x.y + x.C_{in} + y.C_{in}$$

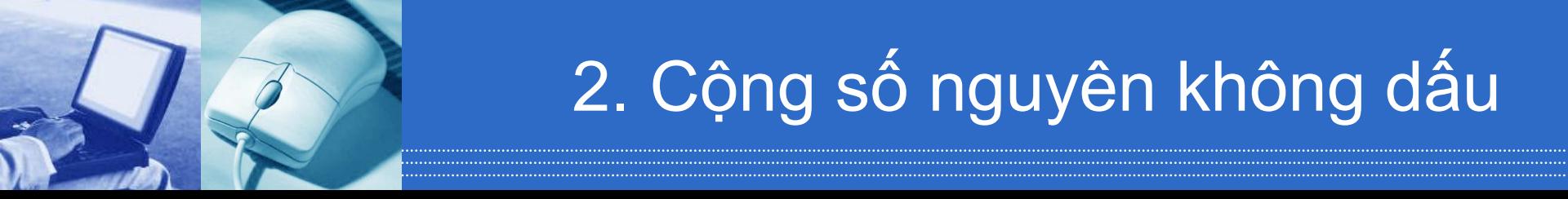


Bộ cộng (tiếp)

■ Bộ cộng n bit

$$\begin{aligned} X &= x_{n-1}x_{n-2}\dots x_0 \\ + \quad Y &= y_{n-1}y_{n-2}\dots y_0 \\ S &= s_{n-1}s_{n-2}\dots s_0 \end{aligned}$$





2. Cộng số nguyên không dấu

- Nguyên tắc: Sử dụng bộ cộng n bit để cộng 2 số nguyên không dấu n bit, kết quả nhận được cũng là n bit.
 - Nếu không có nhó ra khỏi bit cao nhất ($C_{out}=0$) thì kết quả nhận được là đúng.
 - Nếu có nhó ra khỏi bit cao nhất ($C_{out}=1$) thì kết quả nhận được là sai, khi đó đã xảy ra hiện tượng nhó ra ngoài.
- Hiện tượng *nhó ra ngoài* (Carry-out) xảy ra khi tổng của 2 số nguyên không dấu n bit $> 2^n - 1$



VD cộng số nguyên không dấu 8 bit

- Trường hợp không xảy ra carry-out:

$$X = 1001\ 0110 = 150$$

$$\underline{Y = 0001\ 0011 = 19}$$

$$S = 1010\ 1001 = 169$$

$$C_{out} = 0$$

- Trường hợp có xảy ra carry-out:

$$X = 1100\ 0101 = 197$$

$$\underline{Y = 0100\ 0110 = 70}$$

$$S = 0000\ 1011 \neq 267$$

$$C_{out} = 1 \rightarrow \text{carry-out}$$

$$(KQ sai = 2^3 + 2^1 + 2^0 = 11)$$

```
unsigned char x, y, s;  
x = 197;  
y = 70;  
s = x + y;  
printf("%d",s);
```



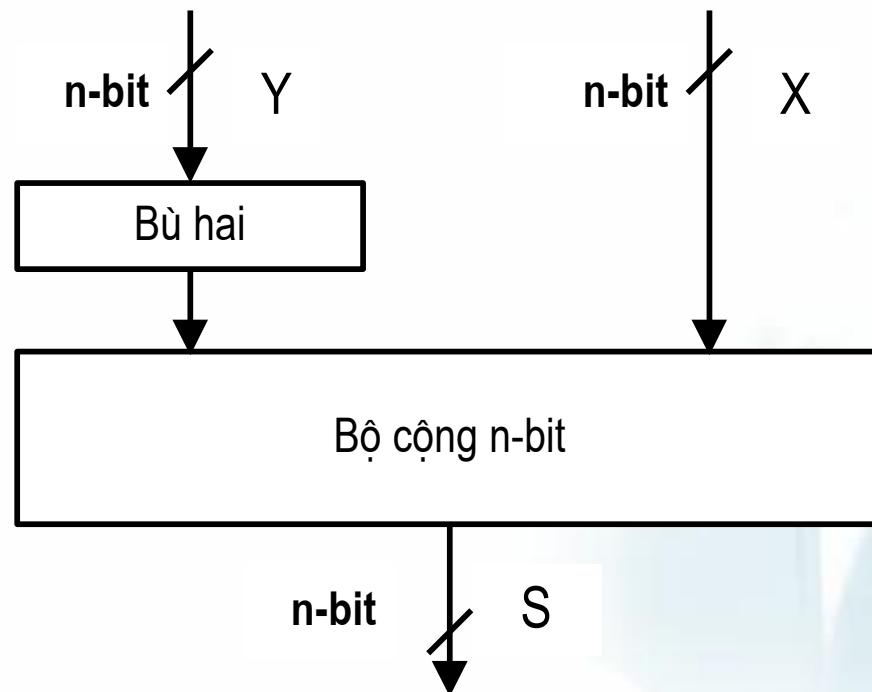
3. Cộng/trừ số nguyên có dấu

- Khi cộng hai số nguyên có dấu n bit, ta không quan tâm đến bit C_{out} và kết quả nhận được cũng là n bit.
 - Cộng hai số khác dấu: kết quả luôn đúng
 - Cộng hai số cùng dấu:
 - Nếu tổng nhận được cùng dấu với 2 số hạng thì kết quả là đúng
 - Nếu tổng nhận được khác dấu với 2 số hạng thì đã xảy ra hiện tượng *tràn số học* (Overflow) và kết quả nhận được là sai
 - Tràn số học xảy ra khi tổng thực sự của hai số nằm ngoài dải biểu diễn của số nguyên có dấu n bit:
$$[-2^{n-1}, 2^{n-1}-1]$$



Phép trừ số nguyên có dấu

- Nguyên tắc thực hiện phép trừ:
 - Ta có: $X - Y = X + (-Y)$
 - Cách thực hiện: lấy X cộng với số bù 2 của Y





Ví dụ cộng 2 số nguyên có dấu (không tràn)

$$\begin{array}{r} X = 0100 \ 0110 = +70 \\ + Y = 0010 \ 1010 = +42 \\ \hline S = 0111 \ 0000 = +112 \end{array}$$

$$\begin{array}{r} X = 0110 \ 0001 = +97 \\ + Y = 1100 \ 1100 = -52 \\ \hline S = 0010 \ 1101 = +45 \\ C_{out} = 1 \rightarrow bỏ qua \end{array}$$

$$\begin{array}{r} X = 1010 \ 0110 = -90 \\ + Y = 0010 \ 0100 = +36 \\ \hline S = 1100 \ 1010 = -54 \end{array}$$

$$\begin{array}{r} X = 1011 \ 0110 = -74 \\ + Y = 1110 \ 0010 = -30 \\ \hline S = 1001 \ 1000 = -104 \\ C_{out} = 1 \rightarrow bỏ qua \end{array}$$



Ví dụ cộng 2 số nguyên có dấu (Overflow)

$$\begin{array}{r} X = 0100\ 1011 = +75 \\ + Y = 0101\ 0001 = +81 \\ \hline S = 1001\ 1100 \neq +156 \\ (S = -2^7 + 2^4 + 2^3 + 2^2 = -100) \end{array}$$

$$\begin{array}{r} X = 1001\ 1000 = -104 \\ + Y = 1011\ 0110 = -74 \\ \hline S = 0100\ 1110 \neq -178 \\ C_{out} = 1 \rightarrow bỏ qua \\ (S = 2^6 + 2^3 + 2^2 + 2^1 = 78) \end{array}$$



4. Nhân số nguyên

- a. Nhân số nguyên không dấu
- b. Nhân số nguyên có dấu





a. Nhân số nguyên không dấu

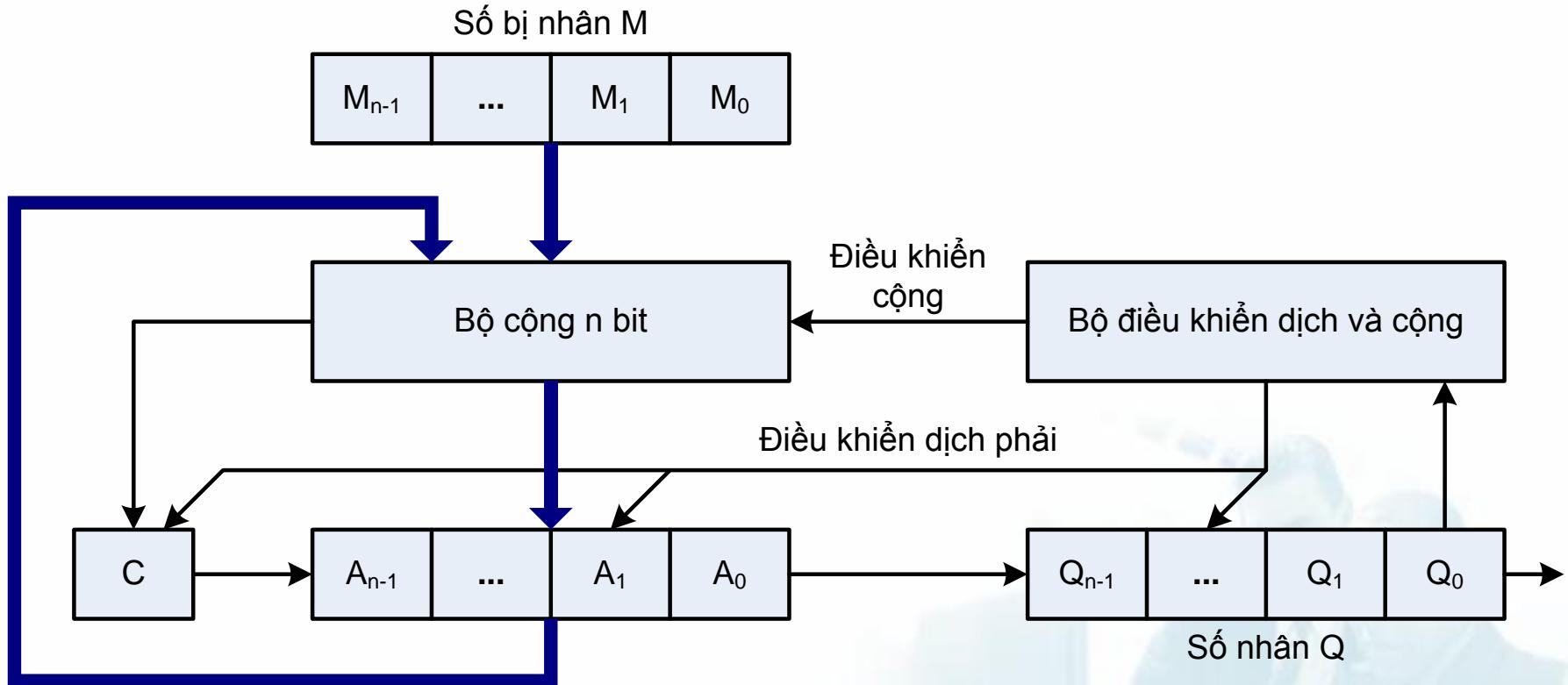
$$\begin{array}{r} 1011 & \text{số bị nhân (11)} \\ \times 1101 & \text{số nhân (13)} \\ \hline 1011 \\ 0000 \\ 1011 \\ 1011 \\ \hline 10001111 & \text{Tích (143)} \end{array}$$

} Các tích riêng phần

- Các tích riêng phần được xác định như sau:
 - Nếu bit của số nhân = 0 → tích riêng phần = 0
 - Nếu bit của số nhân = 1 → tích riêng phần = số bị nhân
 - Tích riêng phần tiếp theo được dịch trái 1 bit so với tích riêng phần trước đó
- Tích = tổng các tích riêng phần
- Nhân 2 số nguyên n bit, tích có độ dài 2n bit → không tràn

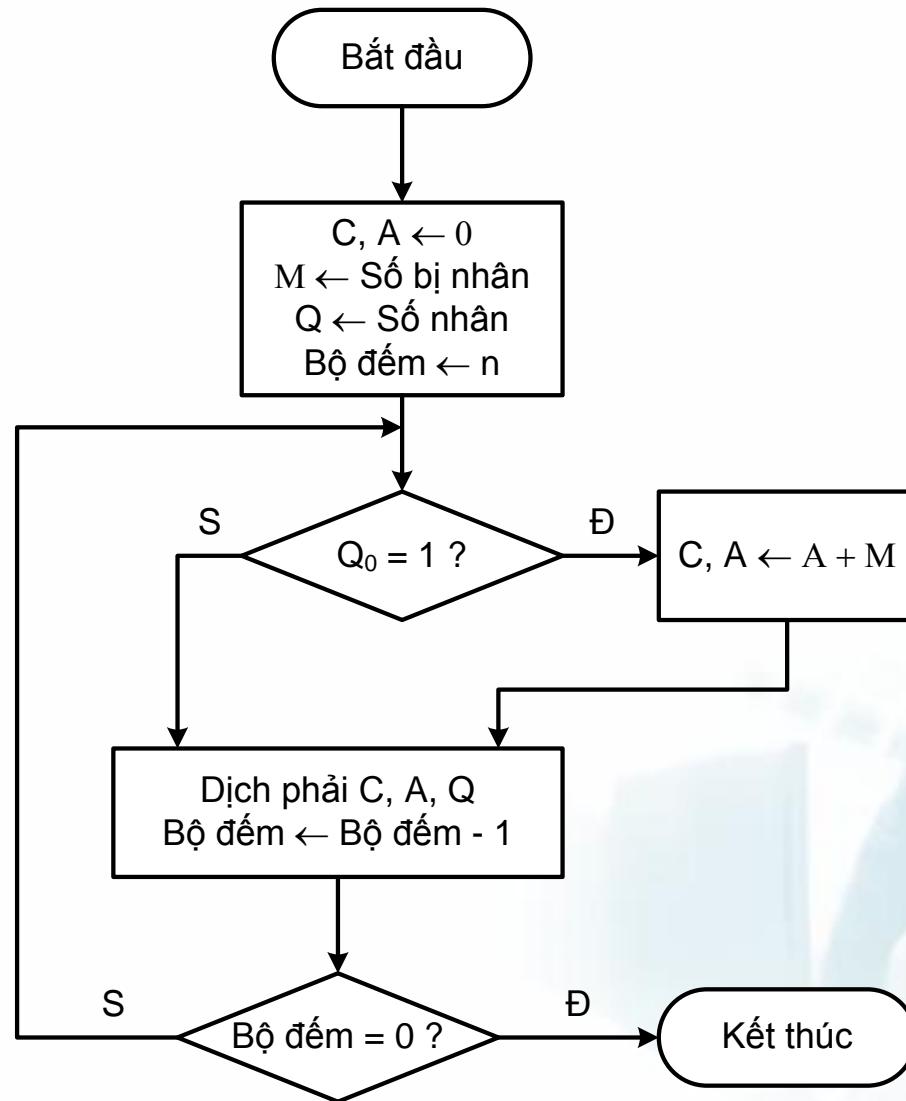


Bộ nhân số nguyên không dấu





Lưu đồ thực hiện

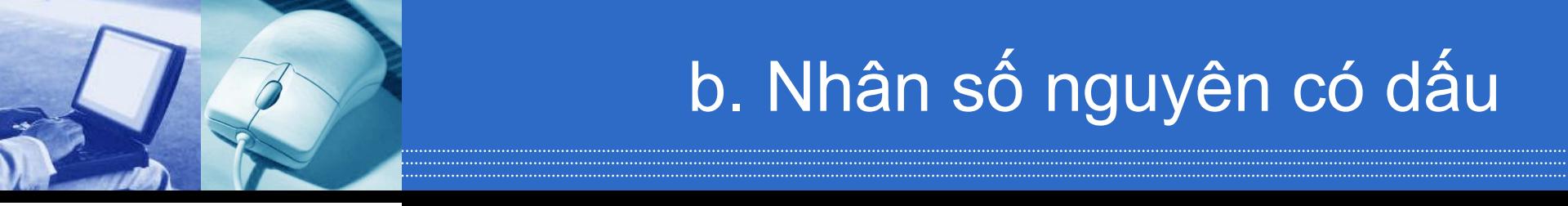




Ví dụ nhân số nguyên không dấu

- $M = 1011$ (11 - Số bị nhân)
- $Q = 1101$ (13 - Số nhân)
- $= 1000\ 1111$ (143 - Tích)

C	A	Q	
0	0000	1101	Các giá trị khởi đầu
	+ <u>1011</u>		
0	1011	1101	$A \leftarrow A + M$
0	0101	1110	Dịch phải
0	0010	1111	Dịch phải
	+ <u>1011</u>		
0	1101	1111	$A \leftarrow A + M$
0	0110	1111	Dịch phải
	+ <u>1011</u>		
1	0001	1111	$A \leftarrow A + M$
0	1000	1111	Dịch phải



b. Nhân số nguyên có dấu

- Sử dụng thuật giải nhân không dấu:
 - Bước 1: Chuyển đổi số nhân và số bị nhân thành số dương tương ứng.
 - Bước 2: Nhân 2 số bằng thuật giải nhân số nguyên không dấu → được tích 2 số dương.
 - Bước 3: Hiệu chỉnh dấu của tích:
 - Nếu 2 thừa số ban đầu cùng dấu thì tích nhận được ở bước 2 là kết quả cần tính.
 - Nếu 2 thừa số ban đầu khác dấu nhau thì kết quả là số bù 2 của tích nhận được ở bước 2.



Nhân số nguyên có dấu

- Sử dụng thuật giải Booth:

- Với số nhân dương:

- Ta có: $2^i + 2^{i-1} + \dots + 2^j = 2^{i+1} - 2^j$ (với $i \geq j$)

- VD: $M * \underline{01110010} = M * (2^7 - 2^4 + 2^2 - 2^1)$

- Quy tắc: duyệt từ trái sang phải:

- Nếu gặp 10 thì trừ A đi M rồi dịch phải

- Nếu gặp 01 thì cộng A với M rồi dịch phải

- Nếu gặp 00 hay 11 thì chỉ dịch phải

- Với số nhân âm:

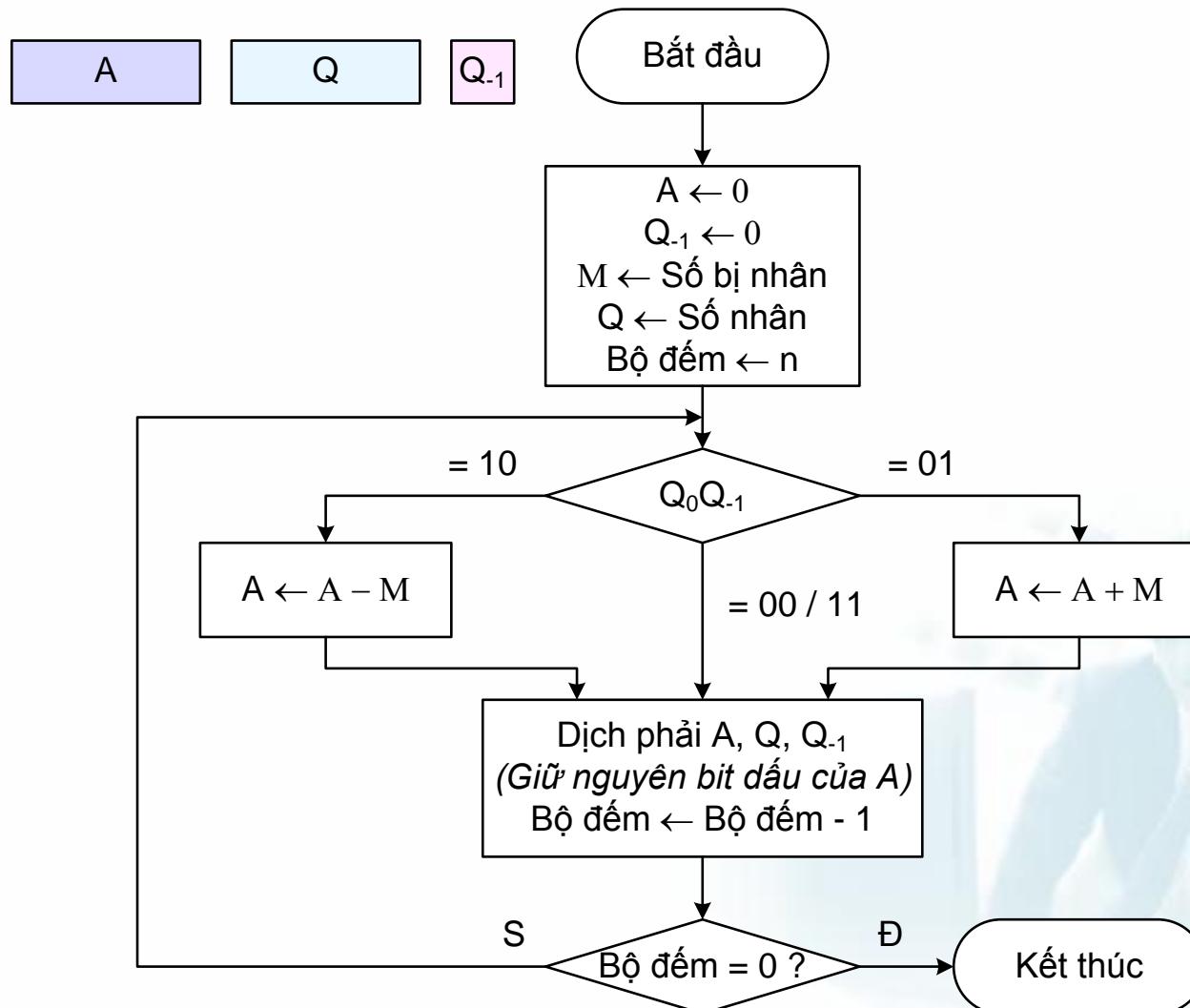
- Ta có:

$$\begin{aligned}11\dots\underline{10}a_{k-1}a_{k-2}\dots a_0 &= -2^{n-1} + 2^{n-2} + \dots + 2^{k+1} + a_{k-1}2^{k-1} + \dots + a_02^0 \\&= -2^{n-1} + 2^{n-1} - 2^{k+1} + a_{k-1}2^{k-1} + \dots + a_02^0\end{aligned}$$

- -2^{k+1} ứng với bit 10 nên vẫn đảm bảo quy tắc ở TH trên



Lưu đồ thực hiện thuật toán Booth





Ví dụ về thuật toán Booth

Ví dụ 1:

$$n = 4 \text{ bit}, M = +7, Q = +3$$

$$M = 0111, Q = 0011, -M = 1001$$

A	Q	Q ₋₁	
0000	001 1	0	; khởi tạo
+1001			
1001	0011	0	; A \leftarrow A - M
1100	100 1	1	; dịch phải
1110	0100	1	; dịch phải
+0111			
10101	0100	1	; A \leftarrow A + M
0010	101 0	0	; dịch phải
0001	0101	0	; dịch phải

Ví dụ 2:

$$n = 4 \text{ bit}, M = +7, Q = -3$$

$$M = 0111, Q = 1101, -M = 1001$$

A	Q	Q ₋₁	
0000	110 1	0	; khởi tạo
+1001			
1001	1101	0	; A \leftarrow A - M
1100	111 0	1	; dịch phải
+0111			
10011	1110	1	; A \leftarrow A + M
0001	111 1	0	; dịch phải
+1001			
1010	1111	0	; A \leftarrow A - M
1101	011 1	1	; dịch phải
1110	1011	1	; dịch phải



5. Chia số nguyên

- a. Chia số nguyên không dấu
- b. Chia số nguyên có dấu





a. Chia số nguyên không dấu

- Ví dụ:

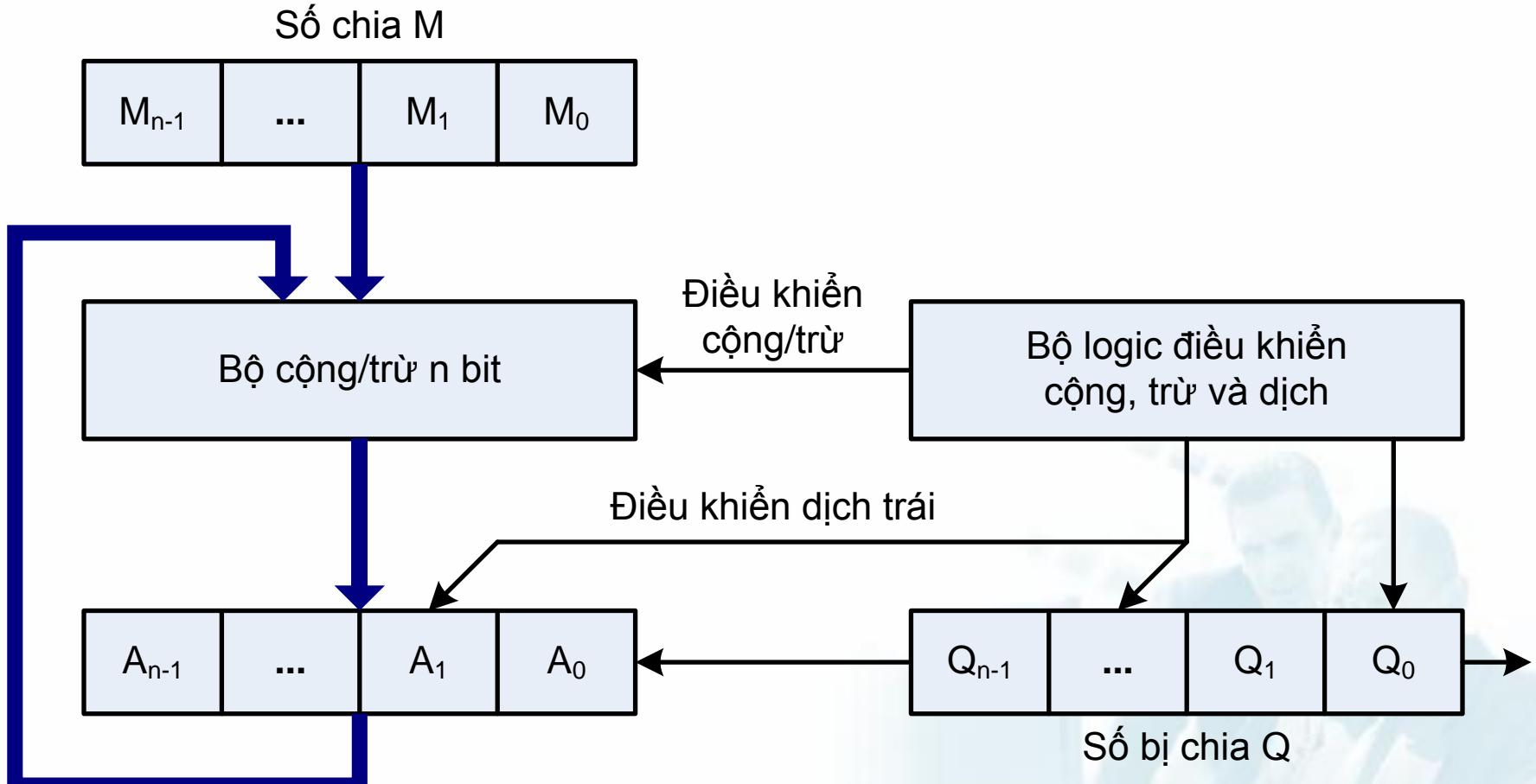
$$\begin{array}{r} 10010011 \\ \underline{-} 1011 \\ \hline 1110 \end{array} \quad \begin{array}{r} 1011 \\ \hline 00001101 \end{array} \quad \begin{array}{l} \text{Số chia} \\ \text{Thương} \end{array}$$

$$\begin{array}{r} 1011 \\ \hline 1111 \\ \hline 1011 \\ \hline 100 \end{array}$$

Số dư

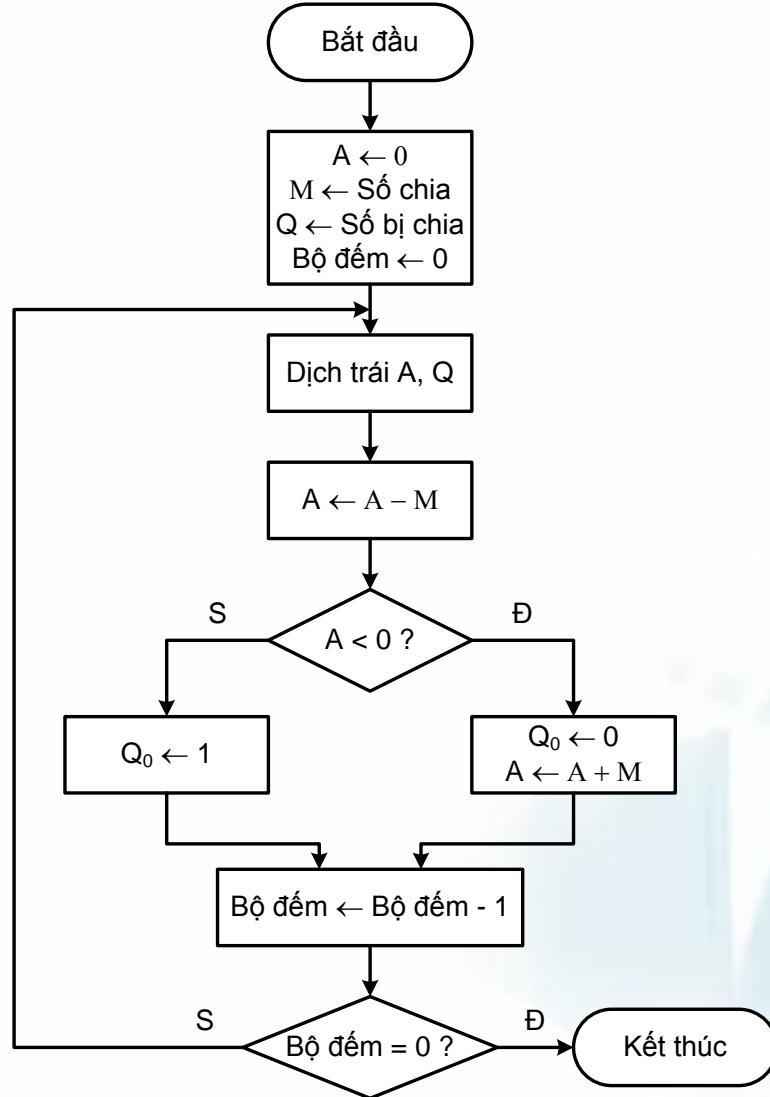


Bộ chia số nguyên không dấu





Lưu đồ thực hiện





b. Chia số nguyên có dấu

- Bước 1: Chuyển đổi số chia và số bị chia thành số dương tương ứng
- Bước 2: Sử dụng thuật giải chia số nguyên không dấu để chia 2 số dương, kết quả nhận được là thương Q và phần dư R đều dương
- Bước 3: Hiệu chỉnh dấu kết quả theo quy tắc sau:

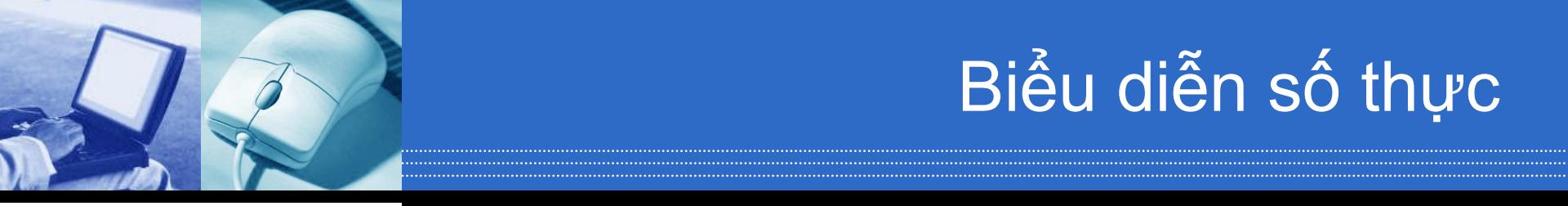
Số bị chia	Số chia	Thương	Phần dư
+	+	giữ nguyên	giữ nguyên
+	-	đảo dấu	giữ nguyên
-	+	đảo dấu	đảo dấu
-	-	giữ nguyên	đảo dấu



Nội dung chương 2

- 2.1. Các hệ đếm cơ bản
- 2.2. Mã hóa và lưu trữ dữ liệu trong máy tính
- 2.3. Biểu diễn số nguyên
- 2.4. Các phép toán số học với số nguyên
- 2.5. Biểu diễn số thực**
- 2.6. Biểu diễn kí tự

1. Khái niệm về số dấu chấm tĩnh
2. Khái niệm về số dấu chấm động
3. Chuẩn IEEE 754/85



Biểu diễn số thực

- Quy ước: "dấu chấm" (point) được hiểu là kí hiệu ngăn cách giữa phần nguyên và phần lẻ của 1 số thực.
- Có 2 cách biểu diễn số thực trong máy tính:
 - Số dấu chấm tĩnh (fixed-point number):
 - Dấu chấm là cố định (số bit dành cho phần nguyên và phần lẻ là cố định)
 - Dùng trong các bộ vi xử lý hay vi điều khiển thế hệ cũ.
 - Số dấu chấm động (floating-point number):
 - Dấu chấm không cố định
 - Dùng trong các bộ vi xử lý hiện nay, có độ chính xác cao hơn.



1. Khái niệm về số dấu chấm tĩnh

- Số bit dành cho phần nguyên và số bit phần lẻ là cố định.
- Giả sử rằng:
 - $U(a,b)$ là tập các số dấu chấm tĩnh **không dấu** có a bit trước dấu chấm và b bit sau dấu chấm.
 - $A(a,b)$ là tập các số dấu chấm tĩnh **có dấu** có a bit (không kể bit dấu) trước dấu chấm và b bit sau dấu chấm.



Số dấu chấm tinh không dấu

- Khoảng xác định của số dấu chấm tinh không dấu: $[0, 2^a - 2^{-b}]$
- Ví dụ:
 - Dùng 8 bit để mã hóa cho kiểu số dấu chấm tinh, trong đó có 2 bit dành cho phần lẻ. Khoảng xác định của kiểu dữ liệu này là: $0 \leq R \leq 2^6 - 2^{-2} = 63.75$
 - VD: giá trị của $101011.11 = 10101111 \times 2^{-2} = 43.75$



Số dấu chấm tĩnh có dấu

- Khoảng xác định của số dấu chấm tĩnh có dấu:
[- 2^a , $2^a - 2^{-b}$]
- Ví dụ:
 - Dùng 8 bit để biểu diễn số chấm tĩnh có dấu với $a=5$, $b=2$
 - Ta được tập các số chấm tĩnh thuộc $A(5,2)$ nằm trong khoảng:
[- 2^5 , $2^5 - 2^{-2}$] hay [-32, 31.75]



Đặc điểm của số dấu chấm tinh

- Các phép toán thực hiện nhanh.
- Độ chính xác khi thực hiện các phép toán không cao, đặc biệt là với phép tính nhân.
- Ví dụ:
 - Khi thực hiện phép nhân ta cần phải có thêm một số lượng bit nhất định để biểu diễn kết quả.
 - Đối với số không dấu:
$$U(a_1, b_1) \times U(a_2, b_2) = U(a_1 + a_2, b_1 + b_2)$$
 - Đối với số có dấu:
$$A(a_1, b_1) \times A(a_2, b_2) = A(a_1 + a_2 + 1, b_1 + b_2)$$



2. Khái niệm về số dấu chấm động

- Floating Point Number → biểu diễn cho số thực
- Một số thực X được biểu diễn theo kiểu số dấu chấm động như sau:

$$X = M * R^E$$

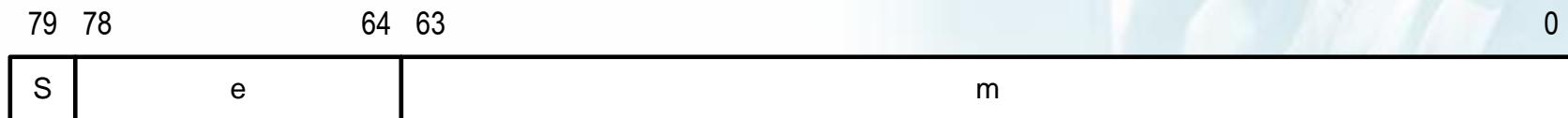
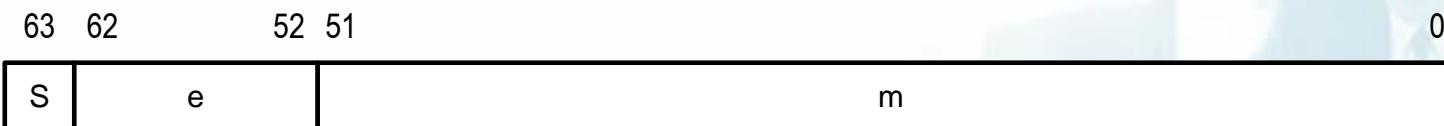
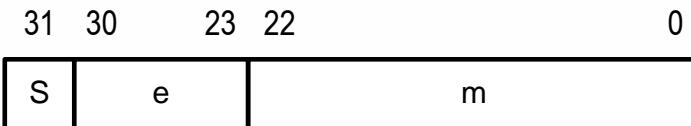
Trong đó:

- M là phần định trị (Mantissa)
- R là cơ số (Radix)
- E là phần mũ (Exponent)
- Với R cố định thì để lưu trữ X ta chỉ cần lưu trữ M và E (dưới dạng số nguyên)



3. Chuẩn IEEE 754/85

- Là chuẩn mã hóa số dấu chấm động
- Cơ số $R = 2$
- Có các dạng cơ bản:
 - Dạng có độ chính xác đơn, 32-bit
 - Dạng có độ chính xác kép, 64-bit
 - Dạng có độ chính xác kép mở rộng, 80-bit
- Khuôn dạng mã hóa:





Khuôn dạng mã hóa

- S là bit dấu, S=0 đó là số dương, S=1 đó là số âm.
- e là mã lệch (excess) của phần mũ E, tức là: $E = e - b$
Trong đó b là độ lệch (bias):
 - Dạng 32-bit : b = 127, hay $E = e - 127$
 - Dạng 64-bit : b = 1023, hay $E = e - 1023$
 - Dạng 80-bit : b = 16383, hay $E = e - 16383$
- m là các bit phần lẻ của phần định trị M, phần định trị được ngầm định như sau: $M = 1.m$
- Công thức xác định giá trị của số thực tương ứng là:

$$X = (-1)^S \times 1.m \times 2^{e-b}$$



Ví dụ về số dấu chấm động

- Ví dụ 1: Có một số thực X có dạng biểu diễn nhị phân theo chuẩn IEEE 754 dạng 32 bit như sau:

1100 0001 0101 0110 0000 0000 0000 0000

Xác định giá trị thập phân của số thực đó.

- Giải:

- S = 1 → X là số âm
- e = 1000 0010 = 130
- m = 10101100...00
- Vậy $X = (-1)^1 \times 1.10101100...00 \times 2^{130-127}$
 $= -1.101011 \times 2^3 = -1101.011 = -13.375$



Ví dụ về số dấu chấm động (tiếp)

- Ví dụ 2: Xác định giá trị thập phân của số thực X có dạng biểu diễn theo chuẩn IEEE 754 dạng 32 bit như sau:
0011 1111 1000 0000 0000 0000 0000 0000
- Giải:



Ví dụ về số dấu chấm động (tiếp)

- Ví dụ 3: Biểu diễn số thực $X = 9.6875$ về dạng số dấu chấm động theo chuẩn IEEE 754 dạng 32 bit
- Giải:

$$X = 9.6875_{(10)} = 1001.1011_{(2)} = 1.0011011 \times 2^3$$

Ta có:

- $S = 0$ vì đây là số dương
- $E = e - 127$ nên $e = 127 + 3 = 130_{(10)} = 1000\ 0010_{(2)}$
- $m = 001101100...00$ (23 bit)

Vậy:

$$X = 0100\ 0001\ 0001\ 1011\ 0000\ 0000\ 0000$$

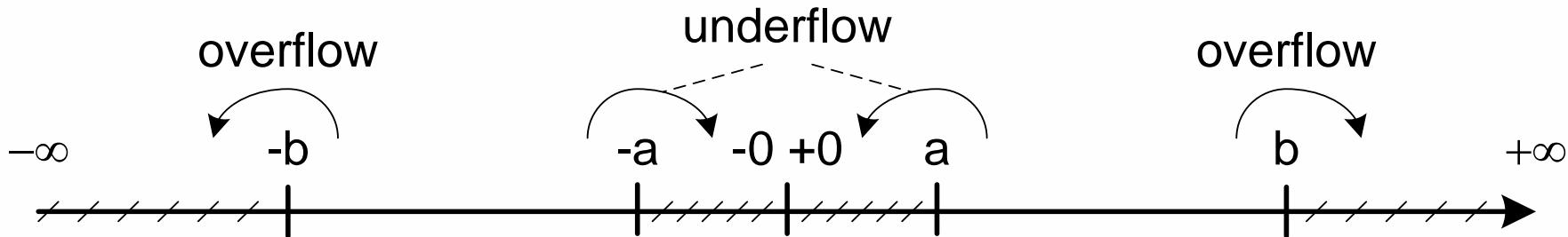


Các quy ước đặc biệt

- Nếu tất cả các bit của e đều bằng 0, các bit của m đều bằng 0, thì $X = \pm 0$
- Nếu tất cả các bit của e đều bằng 1, các bit của m đều bằng 0, thì $X = \pm \infty$
- Nếu tất cả các bit của e đều bằng 1, m có ít nhất một bit bằng 1, thì X không phải là số (not a number - NaN)



Trục số biểu diễn



- Dạng 32 bit: $a = 2^{-127} \approx 10^{-38}$ $b = 2^{+127} \approx 10^{+38}$
- Dạng 64 bit: $a = 2^{-1023} \approx 10^{-308}$ $b = 2^{+1023} \approx 10^{+308}$
- Dạng 80 bit: $a = 2^{-16383} \approx 10^{-4932}$ $b = 2^{+16383} \approx 10^{+4932}$



Thực hiện các phép toán

- $X_1 = M_1 * R^{E_1}$
- $X_2 = M_2 * R^{E_2}$
- Ta có
 - $X_1 \pm X_2 = (M_1 * R^{E_1-E_2} \pm M_2) * R^{E_2}$, với $E_2 \geq E_1$
 - $X_1 * X_2 = (M_1 * M_2) * R^{E_1+E_2}$
 - $X_1 / X_2 = (M_1 / M_2) * R^{E_1-E_2}$





Các khả năng tràn số

- Tràn trên số mũ (Exponent Overflow): mũ dương vượt ra khỏi giá trị cực đại của số mũ dương có thể.
- Tràn dưới số mũ (Exponent Underflow): mũ âm vượt ra khỏi giá trị cực đại của số mũ âm có thể.
- Tràn trên phần định trị (Mantissa Overflow): cộng hai phần định trị có cùng dấu, kết quả bị nhớ ra ngoài bit cao nhất.
- Tràn dưới phần định trị (Mantissa Underflow): Khi hiệu chỉnh phần định trị, các số bị mất ở bên phải phần định trị.



Phép cộng và phép trừ

- Kiểm tra các số hạng có bằng 0 hay không
 - Nếu có thì gán kết quả dựa trên số còn lại.
- Hiệu chỉnh phần định trị
 - Sao cho 2 số có phần mũ giống nhau: tăng số mũ nhỏ và dịch phải phần định trị tương ứng (dịch phải để hạn chế sai số nếu có).
 - VD: $1.01 * 2^3 + 1.11 = 1.01 * 2^3 + 0.00111 * 2^3$
- Cộng hoặc trừ phần định trị
 - Nếu tràn thì dịch phải và tăng số mũ, nếu bị tràn số mũ thì báo lỗi tràn số.
- Chuẩn hóa kết quả
 - Dịch trái phần định trị để bit trái nhất (bit MSB) khác 0.
 - Tương ứng với việc giảm số mũ nên có thể dẫn đến hiện tượng tràn dưới số mũ.



Nội dung chương 2

- 2.1. Các hệ đếm cơ bản
 - 2.2. Mã hóa và lưu trữ dữ liệu trong máy tính
 - 2.3. Biểu diễn số nguyên
 - 2.4. Các phép toán số học với số nguyên
 - 2.5. Biểu diễn số thực
 - 2.6. Biểu diễn kí tự**
- 



Biểu diễn kí tự trong máy tính

1. Bộ mã ASCII (American Standard Code for Information Interchange)
2. Bộ mã Unicode



1. Bộ mã ASCII

- Do ANSI (American National Standard Institute) thiết kế
- Là bộ mã 8 bit → mã hóa được cho $2^8 = 256$ kí tự, có mã từ $00_{16} \div FF_{16}$, bao gồm:
 - 128 kí tự chuẩn có mã từ $00_{16} \div 7F_{16}$
 - 128 kí tự mở rộng có mã từ $80_{16} \div FF_{16}$

HEXA	0	1	2	3	4	5	6	7
0	<NUL> 0	<DLE> 16	<space> 32	0 48	@ 64	P 80	` 96	p 112
1	<SOH> 1	<DC1> 17	! 33	1 49	A 65	Q 81	a 97	q 113
2	<STX> 2	<DC2> 18	" 34	2 50	B 66	R 82	b 98	r 114
3	<ETX> 3	<DC3> 19	# 35	3 51	C 67	S 83	c 99	s 115
4	<EOT> 4	<DC4> 20	\$ 36	4 52	D 68	T 84	d 100	t 116
5	<ENQ> 5	<NAK> 21	% 37	5 53	E 69	U 85	e 101	u 117
6	<ACK> 6	<SYN> 22	& 38	6 54	F 70	V 86	f 102	v 118
7	<BEL> 7	<ETB> 23	' 39	7 55	G 71	W 87	g 103	w 119
8	<BS> 8	<CAN> 24	(40	8 56	H 72	X 88	h 104	x 120
9	<HT> 9	 25) 41	9 57	I 73	Y 89	i 105	y 121
A	<LF> 10	<SUB> 26	*	:	J 74	Z 90	j 106	z 122
B	<VT> 11	<ESC> 27	+	;	K 75	[91	k 107	{ 123
C	<FF> 12	<FS> 28	,	< 60	L 76	\ 92	l 108	 124
D	<CR> 13	<GS> 29	- 45	= 61	M 77] 93	m 109	{ 125
E	<SO> 14	<RS> 30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	<SI> 15	<US> 31	/. 47	? 63	O 79	- 95	o 111	 127



a. Các kí tự chuẩn

- 95 kí tự hiển thị được: có mã từ $20_{16} \div 7E_{16}$
 - 26 chữ cái hoa Latin 'A' ÷ 'Z' có mã từ $41_{16} \div 5A_{16}$
 - 26 chữ cái thường Latin 'a' ÷ 'z' có mã từ $61_{16} \div 7A_{16}$
 - 10 chữ số thập phân '0' ÷ '9' có mã từ $30_{16} \div 39_{16}$
 - Các dấu câu: . , ? ! : ; ...
 - Các dấu phép toán: + - * / ...
 - Một số kí tự thông dụng: #, \$, &, @, ...
 - Dấu cách (mã là 20_{16})
- 33 mã điều khiển: mã từ $00_{16} \div 1F_{16}$ và $7F_{16}$ dùng để mã hóa cho các chức năng điều khiển



Điều khiển định dạng

BS	Backspace - Lùi lại một vị trí: Ký tự điều khiển con trỏ lùi lại một vị trí.
HT	Horizontal Tab - Tab ngang: Ký tự điều khiển con trỏ dịch tiếp một khoảng đã định trước.
LF	Line Feed - Xuống một dòng: Ký tự điều khiển con trỏ chuyển xuống dòng dưới.
VT	Vertical Tab - Tab đứng: Ký tự điều khiển con trỏ chuyển qua một số dòng đã định trước.
FF	Form Feed - Đẩy sang đầu trang: Ký tự điều khiển con trỏ di chuyển xuống đầu trang tiếp theo.
CR	Carriage Return - Về đầu dòng: Ký tự điều khiển con trỏ di chuyển về đầu dòng hiện hành.



Điều khiển truyền số liệu

SOH	Start of Heading - Bắt đầu tiêu đề: Ký tự đánh dấu bắt đầu phần thông tin tiêu đề.
STX	Start of Text - Bắt đầu văn bản: Ký tự đánh dấu bắt đầu khối dữ liệu văn bản và cũng chính là để kết thúc phần thông tin tiêu đề.
ETX	End of Text - Kết thúc văn bản: Ký tự đánh dấu kết thúc khối dữ liệu văn bản đã được bắt đầu bằng STX.
EOT	End of Transmission - Kết thúc truyền: Chỉ ra cho bên thu biết kết thúc truyền.
ENQ	Enquiry - Hỏi: Tín hiệu yêu cầu đáp ứng từ một máy ở xa.
ACK	Acknowledge - Báo nhận: Ký tự được phát ra từ phía thu báo cho phía phát biết rằng dữ liệu đã được nhận thành công.
NAK	Negative Acknowledge - Báo phủ nhận: Ký tự được phát ra từ phía thu báo cho phía phát biết rằng việc nhận dữ liệu không thành công.
SYN	Synchronous / Idle - Đồng bộ hóa: Được sử dụng bởi hệ thống truyền đồng bộ để đồng bộ hóa quá trình truyền dữ liệu.
ETB	End of Transmission Block - Kết thúc khối truyền: Chỉ ra kết thúc khối dữ liệu được truyền.



Điều khiển phân cách thông tin

FS	File Separator - Ký hiệu phân cách tập tin: Đánh dấu ranh giới giữa các tập tin.
GS	Group Separator - Ký hiệu phân cách nhóm: Đánh dấu ranh giới giữa các nhóm tin (tập hợp các bản ghi).
RS	Record Separator - Ký hiệu phân cách bản ghi: Đánh dấu ranh giới giữa các bản ghi.
US	Unit Separator - Ký hiệu phân cách đơn vị: Đánh dấu ranh giới giữa các phần của bản ghi.



Các ký tự điều khiển khác

NUL	Null - Ký tự rỗng: Được sử dụng để điền khoảng trống khi không có dữ liệu.
BEL	Bell - Chuông: Được sử dụng phát ra tiếng bíp khi cần gọi sự chú ý của con người.
SO	Shift Out - Dịch ra: Chỉ ra rằng các mã tiếp theo sẽ nằm ngoài tập ký tự chuẩn cho đến khi gặp ký tự SI.
SI	Shift In - Dịch vào: Chỉ ra rằng các mã tiếp theo sẽ nằm trong tập ký tự chuẩn.
DLE	Data Link Escape - Thoát liên kết dữ liệu: Ký tự sẽ thay đổi ý nghĩa của một hoặc nhiều ký tự liên tiếp sau đó.
DC1 ÷ DC4	Device Control - Điều khiển thiết bị : Các ký tự dùng để điều khiển các thiết bị phụ trợ.
CAN	Cancel - Hủy bỏ: Chỉ ra rằng một số ký tự nằm trước nó cần phải bỏ qua.
EM	End of Medium - Kết thúc phương tiện: Chỉ ra ký tự ngay trước nó là ký tự cuối cùng có tác dụng với phương tiện vật lý.
SUB	Substitute - Thay thế: Được thay thế cho ký tự nào được xác định là bị lỗi.
ESC	Escape - Thoát: Ký tự được dùng để cung cấp các mã mở rộng bằng cách kết hợp với ký tự sau đó.
DEL	Delete - Xóa: Dùng để xóa các ký tự không mong muốn.



b. Các ký tự mở rộng

- Được định nghĩa bởi:
 - Nhà chế tạo máy tính
 - Người phát triển phần mềm
- Ví dụ:
 - Bộ mã ký tự mở rộng của IBM: được dùng trên máy tính IBM-PC.
 - Bộ mã ký tự mở rộng của Apple: được dùng trên máy tính Macintosh.
 - Các nhà phát triển phần mềm tiếng Việt cũng đã thay đổi phần này để mã hóa cho các ký tự riêng của chữ Việt, ví dụ như bộ mã TCVN 5712.



2. Bộ mã Unicode

- Do các hãng máy tính hàng đầu thiết kế
- Là bộ mã 16-bit
- Được thiết kế cho đa ngôn ngữ, trong đó có tiếng Việt



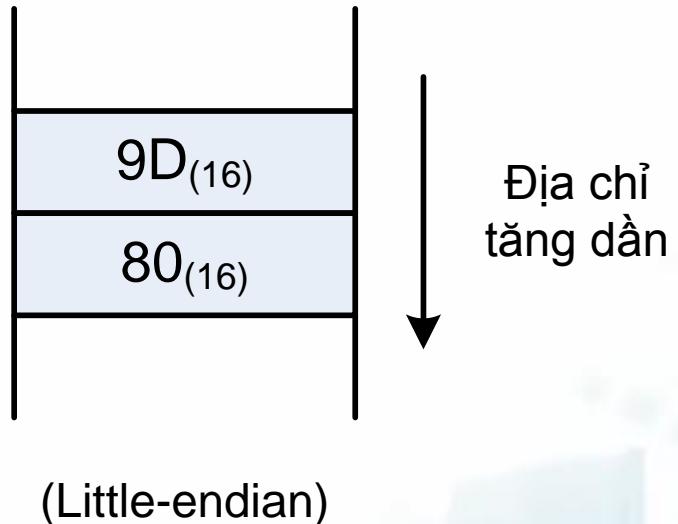
- Giả sử có các biến nhớ dưới đây chứa các số nguyên có dấu 8-bit với nội dung biểu diễn theo hệ 16 như sau:

$$P = 3A \quad Q = 7C \quad R = DE \quad S = FF$$

Hãy xác định giá trị của các biến nhớ đó dưới dạng số thập phân.

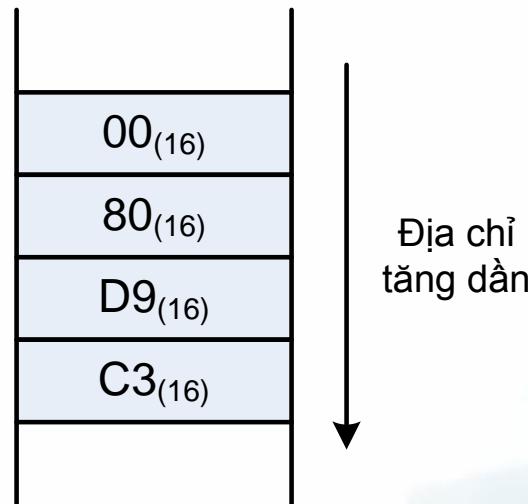
- Giả sử có X thuộc kiểu số nguyên có dấu 16-bit, nó được gán giá trị dưới dạng thập phân bằng -1234. Hãy cho biết nội dung của các byte nhớ chứa biến đó dưới dạng Hexa, biết rằng bộ nhớ lưu trữ theo kiểu đầu nhỏ (*little-endian*).

- Giả sử có biến P chứa số nguyên có dấu 16 bit. Nội dung của biến P được cho trong bộ nhớ như sau:



Hãy xác định giá trị của biến P dưới dạng thập phân.

- Giả sử có một biến số thực X được biểu diễn bằng số dấu chấm động theo chuẩn IEEE 754 dạng 32 bit, nó chiếm 4 byte trong bộ nhớ với nội dung được chỉ ra ở hình vẽ sau.



Biết rằng bộ nhớ tổ chức theo kiểu *đầu nhỏ (little-endian)*, hãy xác định giá trị thập phân của số thực đó.

- Giả sử có biến X thuộc kiểu số dấu chấm động theo chuẩn IEEE 754 dạng 32 bit. Nó được gán giá trị dưới dạng thập phân bằng **-124.125** và lưu trữ vào bộ nhớ bắt đầu từ byte nhớ có địa chỉ là 200. Hãy cho biết nội dung của các byte nhớ chứa biến đó dưới dạng Hexa, biết rằng bộ nhớ lưu trữ theo kiểu *đầu nhỏ (little-endian)*.



Kiến trúc máy tính

Chương 3

HỆ THỐNG MÁY TÍNH

- 3.1. Cấu trúc và hoạt động cơ bản của máy tính
- 3.2. Bộ xử lý trung tâm
- 3.3. Bộ nhớ máy tính
- 3.4. Hệ thống vào ra
- 3.5. Giới thiệu hệ điều hành



Cấu trúc và hoạt động cơ bản của máy tính

- Cấu trúc cơ bản của máy tính
- Liên kết hệ thống
- Hoạt động cơ bản của máy tính
- Cấu trúc một máy tính cá nhân điển hình





3.1.1 Cấu trúc cơ bản của máy tính

- Bộ xử lý trung tâm (Central Processing Unit)
- Bộ nhớ (Memory)
- Hệ thống vào-ra (Input-Output System)
- Liên kết hệ thống (System Interconnection)



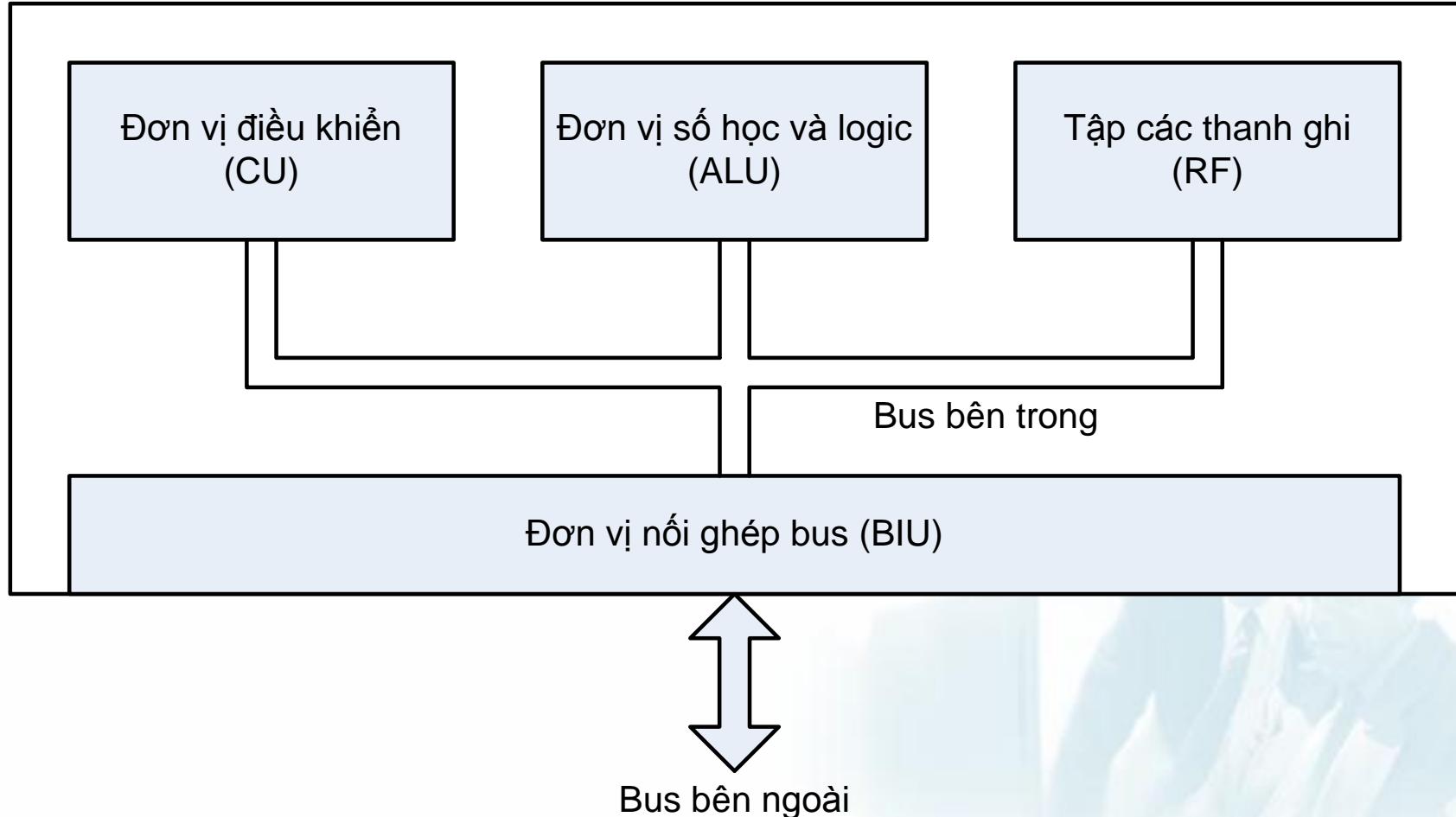


Bộ xử lý trung tâm (CPU)

- Chức năng:
 - Điều khiển hoạt động của toàn bộ hệ thống máy tính
 - Xử lý dữ liệu
- Nguyên tắc hoạt động cơ bản: CPU hoạt động theo chương trình nằm trong bộ nhớ chính, bằng cách:
 - Nhận lần lượt từng lệnh từ bộ nhớ chính,
 - Sau đó tiến hành giải mã lệnh và phát các tín hiệu điều khiển thực thi lệnh.
 - Trong quá trình thực thi lệnh, CPU có thể trao đổi dữ liệu với bộ nhớ chính hay hệ thống vào-ra.



Cấu trúc cơ bản của CPU





Các thành phần cơ bản của CPU

- **Đơn vị điều khiển (Control Unit - CU)**: điều khiển hoạt động của máy tính theo chương trình đã định sẵn.
- **Đơn vị số học và logic (Arithmetic and Logic Unit - ALU)**: thực hiện các phép toán số học và các phép toán logic trên các dữ liệu cụ thể.
- **Tập thanh ghi (Register File - RF)**: lưu giữ các thông tin tạm thời phục vụ cho hoạt động của CPU.
- **Bus bên trong (Internal Bus)**: kết nối các thành phần bên trong CPU với nhau.
- **Đơn vị nối ghép bus (Bus Interface Unit - BIU)** kết nối và trao đổi thông tin với nhau giữa bus bên trong (*internal bus*) với bus bên ngoài (*external bus*).



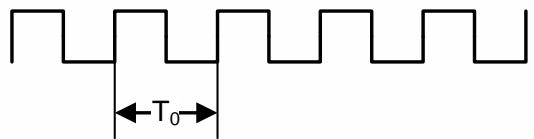
Tốc độ của bộ xử lý

- Tốc độ của bộ xử lý:
 - Số lệnh được thực hiện trong 1 giây
 - MIPS (Millions of Instructions per Second)
 - Khó đánh giá chính xác
- Tần số xung nhịp của bộ xử lý:
 - Bộ xử lý hoạt động theo một xung nhịp (Clock) có tần số xác định
 - Tốc độ của bộ xử lý được đánh giá gián tiếp thông qua tần số của xung nhịp



Tốc độ của bộ xử lý (tiếp)

- Dạng xung nhịp:



- T_0 : chu kỳ xung nhịp
- Mỗi thao tác của bộ xử lý mất một số nguyên lần chu kỳ T_0
⇒ T_0 càng nhỏ thì bộ xử lý chạy càng nhanh
- Tần số xung nhịp: $f_0 = 1/T_0$ gọi là tần số làm việc của CPU
- VD: Máy tính dùng bộ xử lý Pentium IV 2GHz
Ta có: $f_0 = 2\text{GHz} = 2 \times 10^9\text{Hz}$
 $\rightarrow T_0 = 1/f_0 = 1 / (2 \times 10^9) = 0,5\text{ ns}$

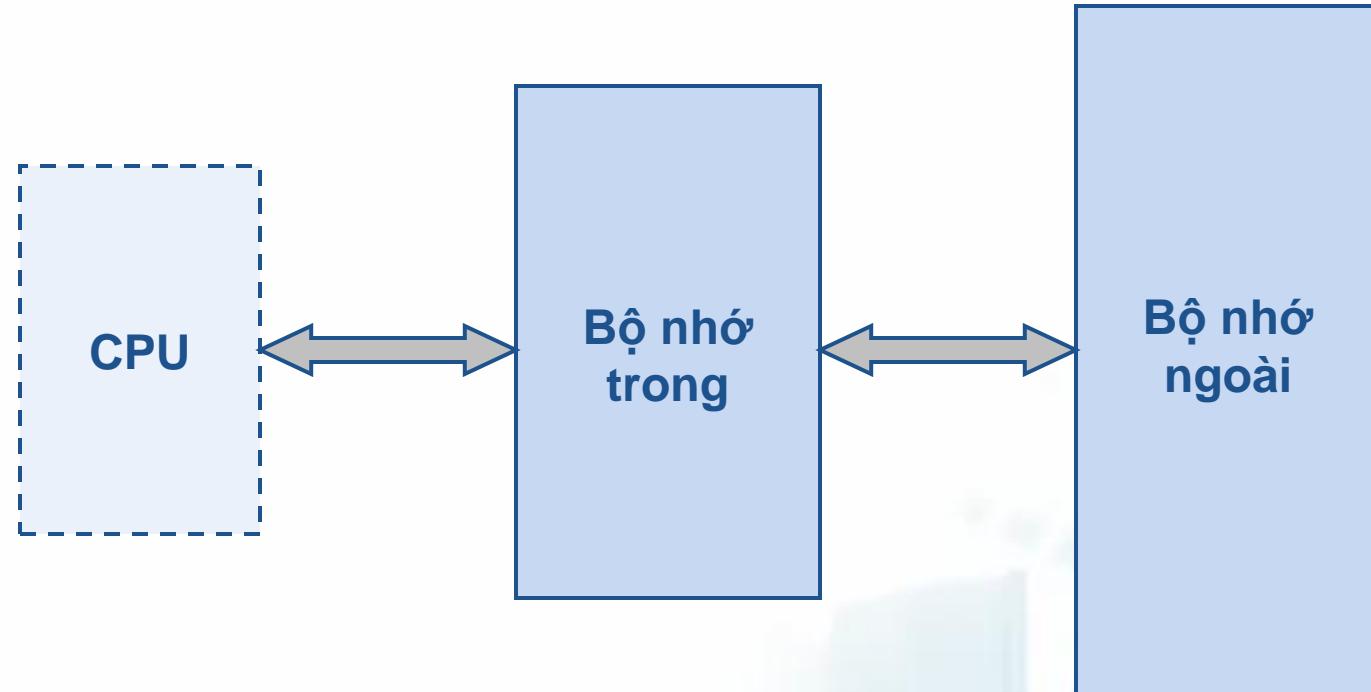


Bộ nhớ máy tính

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



Các thành phần bộ nhớ máy tính



- Chức năng và đặc điểm:
 - Chứa các thông tin mà CPU có thể trao đổi trực tiếp
 - Tốc độ rất nhanh
 - Dung lượng không lớn
 - Sử dụng bộ nhớ bán dẫn: ROM và RAM
- Các loại bộ nhớ trong:
 - Bộ nhớ chính
 - Bộ nhớ cache (bộ nhớ đệm nhanh)



Bộ nhớ chính (Main Memory)

- Là thành phần nhớ tồn tại trên mọi hệ thống máy tính
- Chứa các chương trình và dữ liệu đang được CPU sử dụng
- Tổ chức thành các ngăn nhớ được đánh địa chỉ
- Ngăn nhớ thường được tổ chức theo Byte
- Nội dung của ngăn nhớ có thể thay đổi, song địa chỉ vật lý của ngăn nhớ luôn cố định
- Thông thường, bộ nhớ chính bao gồm 2 phần:
 - Bộ nhớ RAM
 - Bộ nhớ ROM

Nội dung	Địa chỉ
00101011	0000
11010101	0001
00001010	0010
01011000	0011
11111011	0100
00001000	0101
11101010	0110
00000000	0111
10011101	1000
00101010	1001
11101011	1010
00000010	1011
00101011	1100
00101011	1101
11111111	1110
10101010	1111



Bộ nhớ đệm nhanh (Cache memory)

- Là thành phần nhớ tốc độ nhanh được đặt đệm giữa CPU và bộ nhớ chính nhằm tăng tốc độ truy cập bộ nhớ của CPU.
- Tốc độ của cache nhanh hơn bộ nhớ chính nhưng dung lượng nhỏ hơn.
- Cache thường được chia ra thành một số mức: cache L1, cache L2, ...
- Hiện nay cache được tích hợp trên các chip vi xử lý.
- Cache có thể có hoặc không.

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

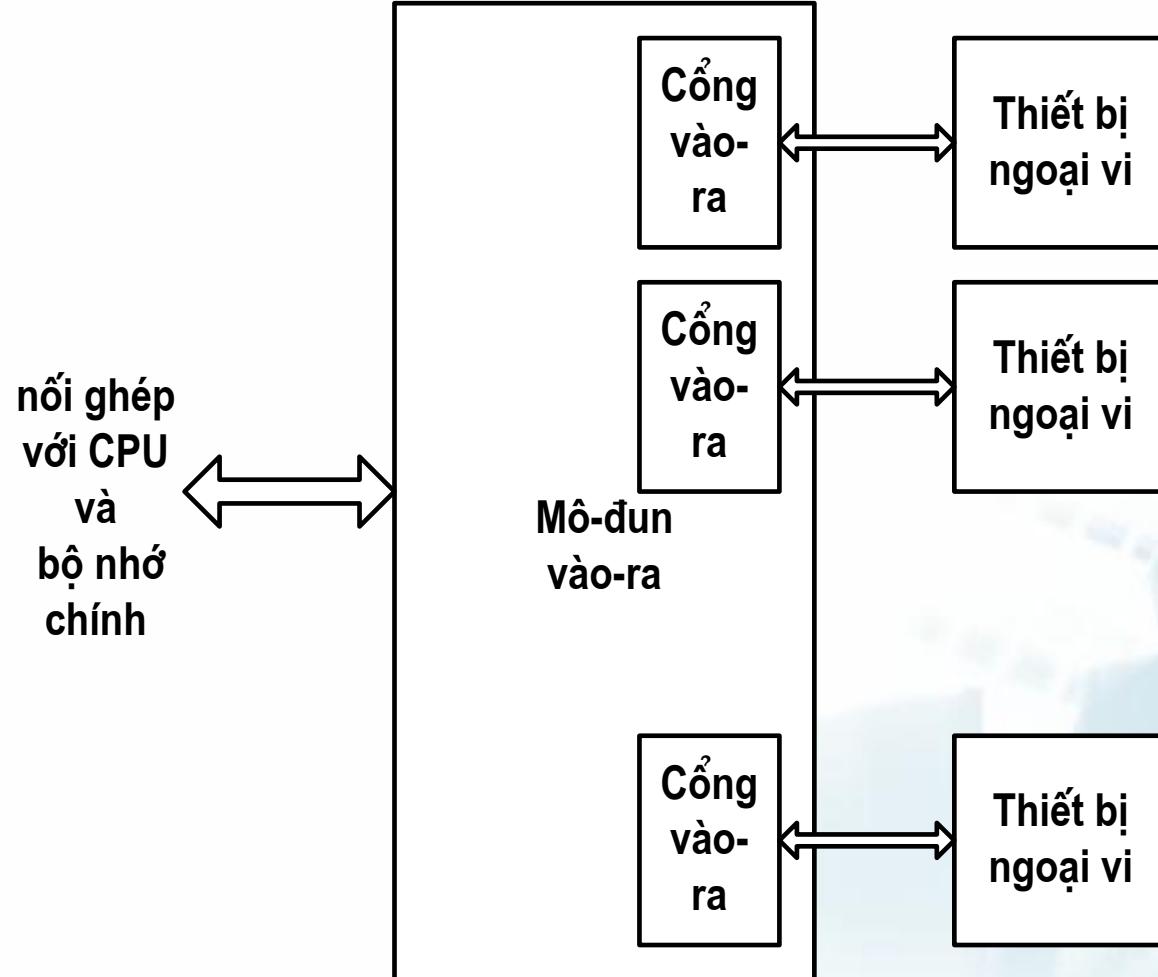


Hệ thống vào-ra (Input-Output)

- Chức năng: Trao đổi thông tin giữa máy tính với thế giới bên ngoài.
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị ngoại vi (Peripheral Devices)
 - Các mô-đun nối ghép vào-ra (IO Modules)



Cấu trúc cơ bản của hệ thống vào-ra





Các thiết bị ngoại vi

- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Các loại thiết bị ngoại vi cơ bản:
 - Thiết bị vào: bàn phím, chuột, máy quét ...
 - Thiết bị ra: màn hình, máy in ...
 - Thiết bị nhớ: các ổ đĩa ...
 - Thiết bị truyền thông: modem ...



Mô-đun vào-ra

- Chức năng: nối ghép thiết bị ngoại vi với máy tính
- Khái niệm cổng vào-ra:
 - Trong mỗi mô-đun vào-ra có một hoặc một vài cổng vào-ra (I/O Port).
 - Mỗi cổng vào-ra cũng được đánh một địa chỉ xác định.
 - Thiết bị ngoại vi được kết nối và trao đổi dữ liệu với bên trong máy tính thông qua các cổng vào-ra.

- Luồng thông tin trong máy tính
- Cấu trúc bus cơ bản
- Phân cấp bus trong máy tính



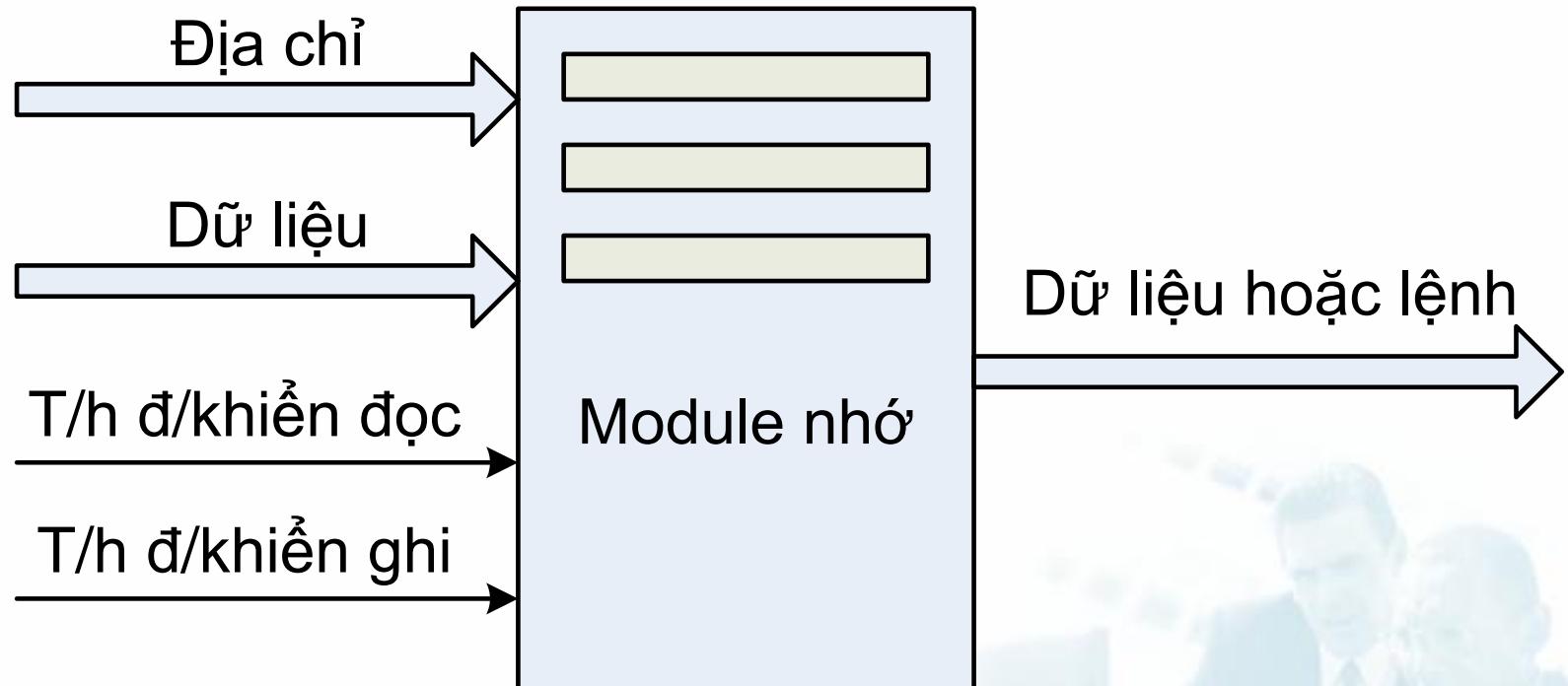
Luồng thông tin trong máy tính

- Các mô-đun trong máy tính:
 - CPU
 - Mô-đun nhớ
 - Mô-đun vào-ra
- ⇒ cần được kết nối với nhau



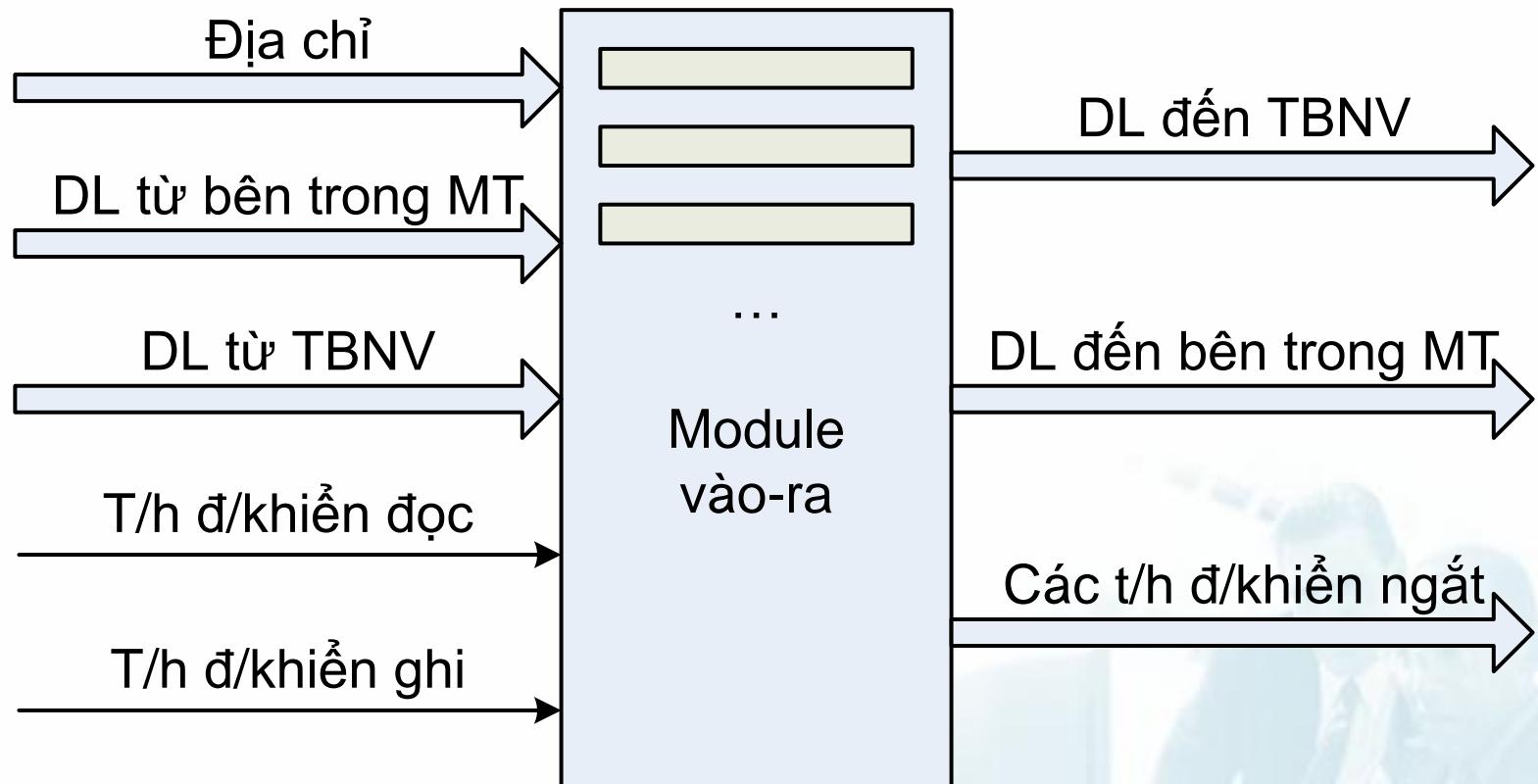


Kết nối mô-đun nhớ



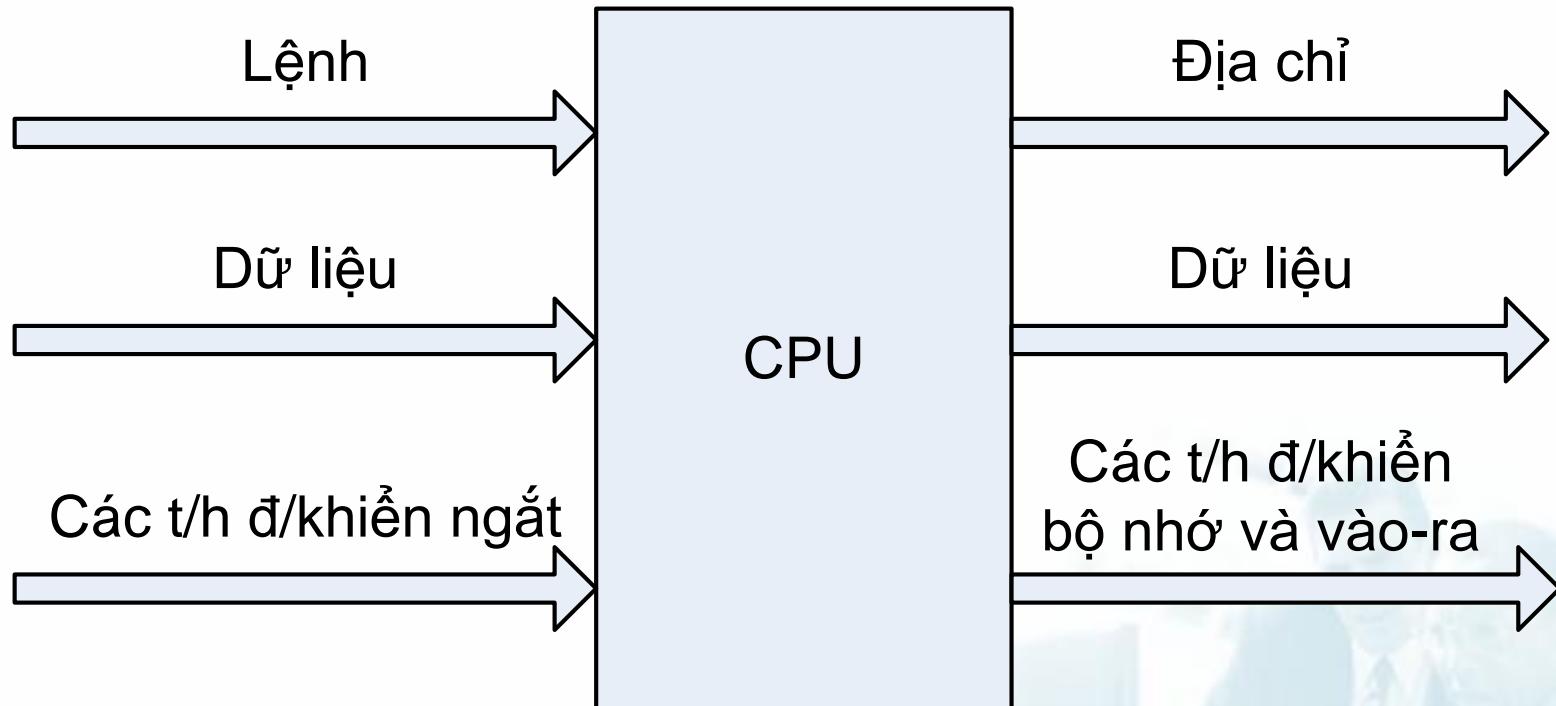


Kết nối mô-đun vào-ra





Kết nối CPU



- Có 4 loại thông tin:
 - Địa chỉ
 - Dữ liệu
 - Lệnh
 - Thông tin điều khiển



Cấu trúc bus cơ bản

- Khái niệm chung về bus:

- Bus: tập hợp các đường kết nối dùng để vận chuyển thông tin giữa các thành phần của máy tính với nhau.
- Độ rộng bus: là số đường dây của bus có thể truyền thông tin đồng thời. Tính bằng bit.
- Phân loại cấu trúc bus:
 - Cấu trúc đơn bus
 - Cấu trúc đa bus

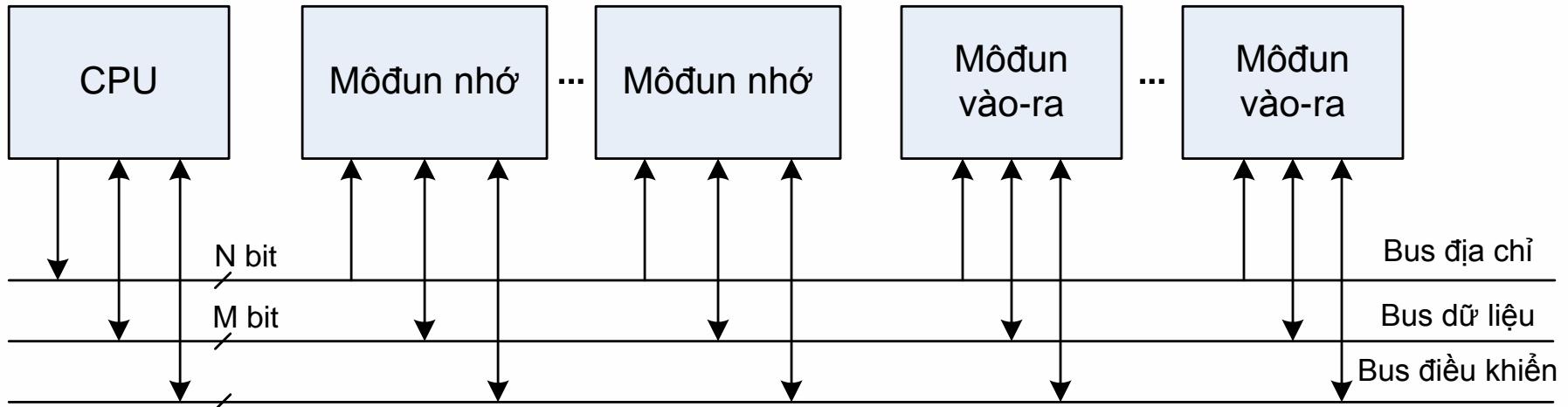


Bus đồng bộ và bus không đồng bộ

- Bus đồng bộ:
 - Có đường tín hiệu Clock
 - Các sự kiện xảy ra trên bus được xác định bởi xung nhịp Clock.
- Bus không đồng bộ:
 - Không có đường tín hiệu Clock
 - Một sự kiện trên bus kết thúc sẽ kích hoạt sự kiện tiếp theo.



Cấu trúc đơn bus





Bus địa chỉ (Address bus)

- Chức năng: vận chuyển địa chỉ từ CPU đến mô-đun nhớ hay mô-đun vào-ra để xác định ngăn nhớ hay cổng vào-ra mà CPU cần trao đổi thông tin.
- Độ rộng bus địa chỉ: xác định dung lượng bộ nhớ cực đại của hệ thống.
- Nếu độ rộng bus địa chỉ là N bit (gồm N đường dây $A_{N-1}, A_{N-2}, \dots, A_2, A_1, A_0$) thì:
 - có khả năng vận chuyển được N bit địa chỉ đồng thời
 - có khả năng đánh địa chỉ tối đa được 2^N ngăn nhớ = 2^N Byte → gọi là không gian địa chỉ bộ nhớ.

- Độ rộng bus địa chỉ của một số bộ xử lý của Intel
 - 8088/8086 : $N = 20$ bit $\rightarrow KGDCBN = 2^{20}$ Byte = 1 MB
 - 80286 : $N = 24$ bit $\rightarrow KGDCBN = 2^{24}$ Byte = 16 MB
 - 80386, 80486, Pentium : $N = 32$ bit $\rightarrow KGDCBN = 2^{32}$ Byte = 4 GB
 - Pentium II, III, 4 : $N = 36$ bit $\rightarrow KGDCBN = 2^{36}$ Byte = 64 GB



Bus dữ liệu (Data bus)

- Chức năng:
 - Vận chuyển lệnh từ bộ nhớ đến CPU
 - Vận chuyển dữ liệu giữa CPU, các mô-đun nhớ và mô-đun vào-ra với nhau
- Độ rộng bus dữ liệu: Xác định số bit dữ liệu có thể được trao đổi đồng thời.
 - Nếu độ rộng bus dữ liệu là M bit (gồm M đường dây $D_{M-1}, D_{M-2}, \dots, D_2, D_1, D_0$) thì nghĩa là đường bus dữ liệu đó có thể vận chuyển đồng thời được M bit dữ liệu
 - M thường là 8, 16, 32, 64 bit

- Độ rộng bus dữ liệu của một số bộ xử lý của Intel:
 - 8088 : $M = 8$ bit
 - 8086, 80286 : $M = 16$ bit
 - 80386, 80486 : $M = 32$ bit
 - Các bộ xử lý Pentium : $M = 64$ bit



Bus điều khiển (Control bus)

- Chức năng: vận chuyển các tín hiệu điều khiển
- Các loại tín hiệu điều khiển:
 - Các tín hiệu điều khiển phát ra từ CPU để điều khiển mô-đun nhớ hay mô-đun vào-ra
 - Các tín hiệu yêu cầu từ mô-đun nhớ hay mô-đun vào-ra gửi đến CPU

- Các tín hiệu phát ra từ CPU để điều khiển đọc/ghi:
 - *Memory Read (MEMR)*: điều khiển đọc dữ liệu từ một ngăn nhớ có địa chỉ xác định lên bus dữ liệu.
 - *Memory Write (MEMW)*: điều khiển ghi dữ liệu có sẵn trên bus dữ liệu đến một ngăn nhớ có địa chỉ xác định.
 - *I/O Read (IOR)*: điều khiển đọc dữ liệu từ một cổng vào-ra có địa chỉ xác định lên bus dữ liệu.
 - *I/O Write (IOW)*: điều khiển ghi dữ liệu có sẵn trên bus dữ liệu ra một cổng có địa chỉ xác định.

- Các tín hiệu điều khiển ngắt:

- *Interrupt Request (INTR)*: Tín hiệu từ bộ điều khiển vào-ra gửi đến yêu cầu ngắt CPU để trao đổi vào-ra. Tín hiệu INTR có thể bị che.
- *Interrupt Acknowledge (INTA)*: Tín hiệu phát ra từ CPU báo cho bộ điều khiển vào-ra biết CPU chấp nhận ngắt.
- *Non Maskable Interrupt (NMI)*: tín hiệu ngắt không che được gửi đến ngắt CPU.
- *Reset*: Tín hiệu từ bên ngoài gửi đến CPU và các thành phần khác để khởi động lại máy tính.

- Các tín hiệu điều khiển bus:

- *Bus Request (BRQ) / Hold*: Tín hiệu từ bộ điều khiển vào-ra chuyên dụng gửi đến yêu cầu CPU chuyển nhượng quyền sử dụng bus.
- *Bus Grant (BGT) / Hold Acknowledge*: Tín hiệu phát ra từ CPU chấp nhận chuyển nhượng quyền sử dụng bus.
- *Lock*: Tín hiệu khóa không cho xin chuyển nhượng bus.
- *Unlock*: Tín hiệu mở khóa cho xin chuyển nhượng bus.



Đặc điểm của cấu trúc đơn bus

- Tất cả các thành phần cùng nối vào một đường bus chung
- Tại một thời điểm, bus chỉ phục vụ được một yêu cầu trao đổi dữ liệu
- Bus phải có tốc độ bằng tốc độ của thành phần nhanh nhất trong hệ thống
- Bus phụ thuộc vào cấu trúc bus của bộ xử lý → các mô-đun nhớ và các mô-đun vào-ra cũng phụ thuộc vào bộ xử lý cụ thể.
→ Cần phải thiết kế bus phân cấp hay cấu trúc đa bus



Phân cấp bus trong máy tính

- Phân cấp thành nhiều bus khác nhau cho các thành phần:
 - Bus của bộ xử lý
 - Bus của bộ nhớ chính
 - Các bus vào-ra
- Phân cấp bus khác nhau về tốc độ
- Các bus nối ghép với mô-đun nhớ và mô-đun vào-ra không phụ thuộc vào bộ xử lý cụ thể.

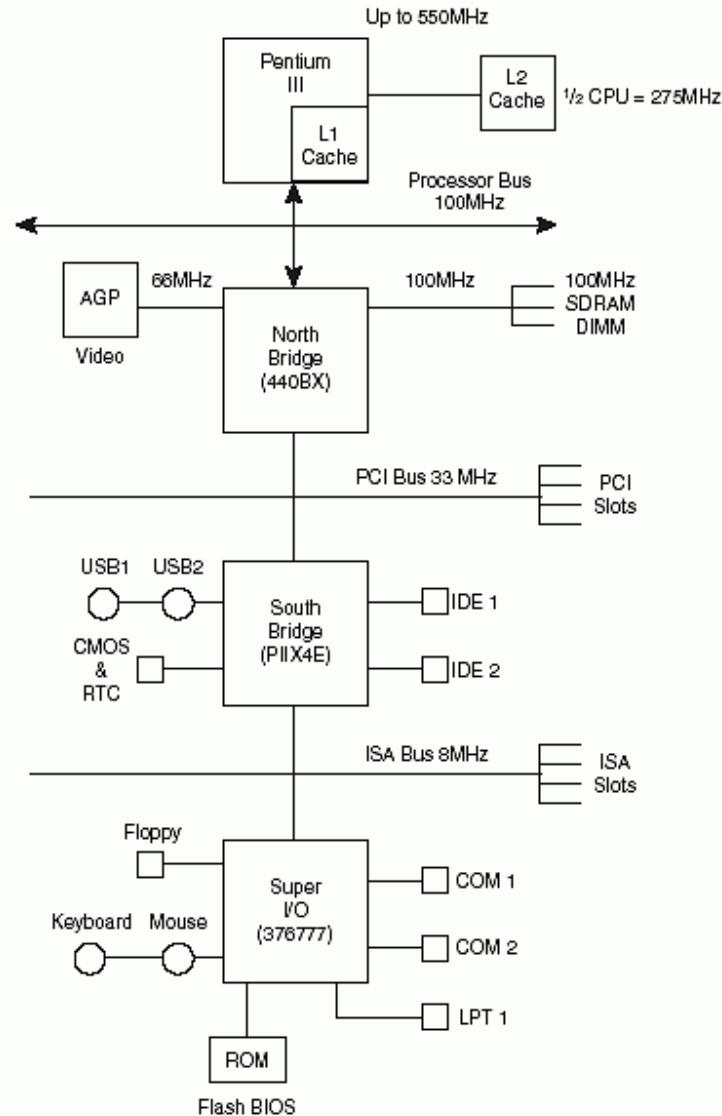


Các bus điển hình trong máy PC

- Bus của bộ xử lý (Front Side Bus – FSB): có tốc độ nhanh nhất
- Bus của bộ nhớ chính (nối ghép với các mô-đun nhớ RAM)
- AGP bus (Accelerated Graphic Port) – cổng tăng tốc đồ họa: nối ghép với card màn hình
- PCI bus (Peripheral Component Interconnect): nối ghép với các TBNV có tốc độ trao đổi dữ liệu nhanh.
- USB (Universal Serial Bus): bus nối tiếp đa năng
- IDE (Integrated Drive Electronics): bus kết nối với ổ đĩa cứng hoặc ổ đĩa quang (CD, DVD, ...)



VD 1: Hệ thống Pentium III

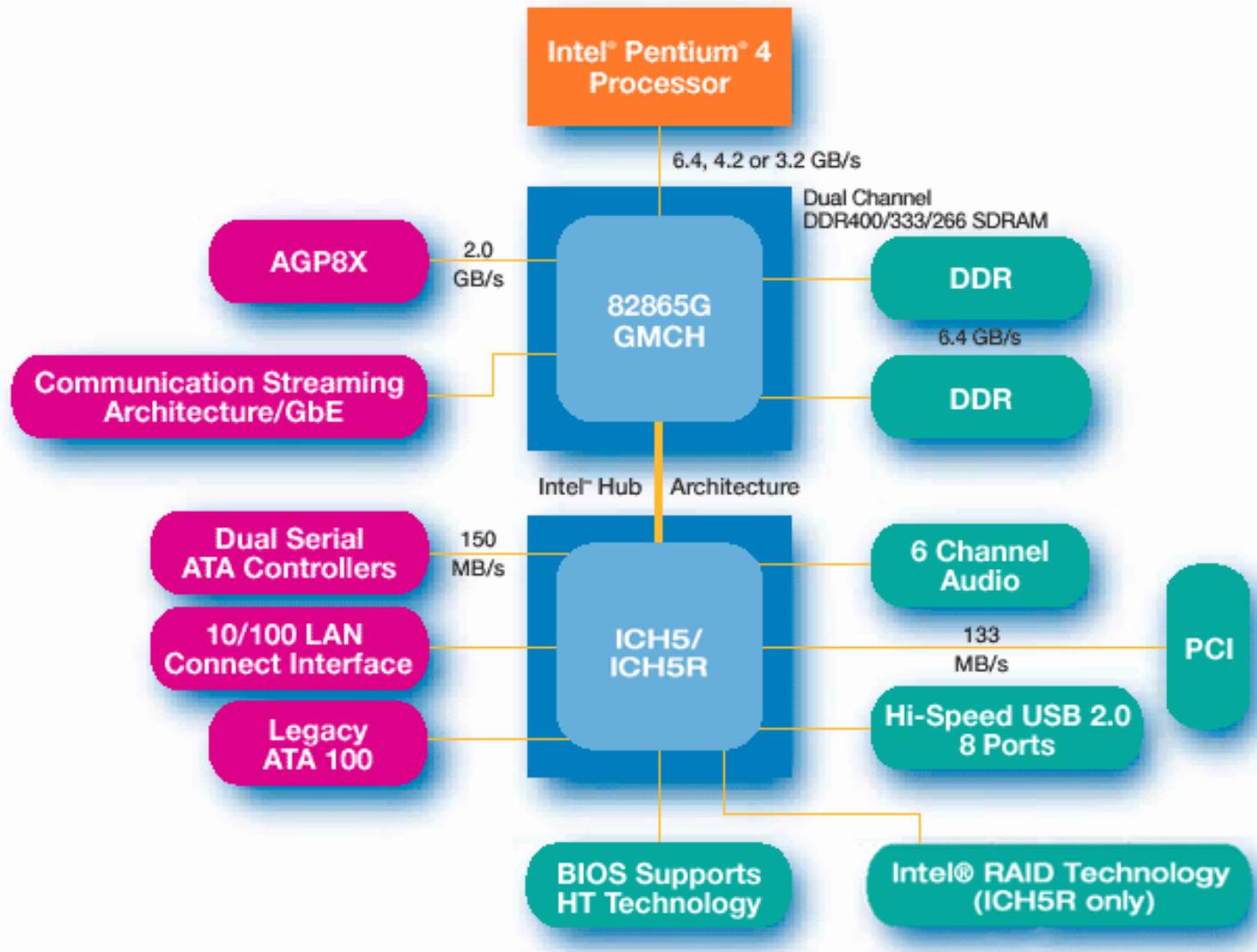




VD 2: Pentium 4 dùng chipset 865PE

- Bộ chipset Intel 865PE gồm có hai con chip 82865PE (MCH) và 82801ER (ICH5-R):
 - Chip Intel 82865PE MCH hỗ trợ CPU Pentium 4 sử dụng công nghệ siêu phân luồng (Hyper Threading – HT), bộ nhớ DDR400, mode Dual Channel (riêng chip 82865G (MCH) của bộ chipset i865G thì tích hợp cả nhân xử lý đồ họa Intel Extreme Graphics 2).
 - Chip Intel 82801ER (ICH5-R) tích hợp đủ các bộ điều khiển (controller) các thiết bị I/O như Ultra ATA 100, Serial ATA- RAID-0, USB 2.0, âm thanh AC'97 có 6 kênh, LAN, EHCI, ASF, ...

VD 2 (tiếp)





3.1.2. Hoạt động cơ bản của máy tính

- Thực hiện chương trình
- Xử lý ngắt
- Hoạt động vào ra





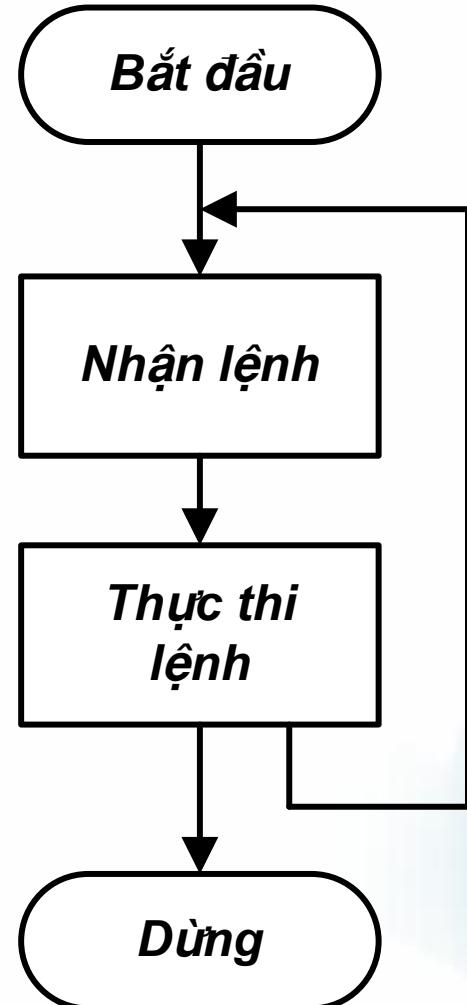
Thực hiện chương trình

- **Nguyên tắc hoạt động:**

- Chương trình đang được thực hiện phải nằm trong bộ nhớ chính của máy tính.
- Thực hiện chương trình là lặp đi lặp lại chu trình lệnh gồm hai bước:
 - Nhận lệnh
 - Thực thi lệnh
- Thực hiện chương trình bị dừng nếu bị lỗi nghiêm trọng khi thực thi lệnh hoặc gặp lệnh dừng chương trình



Chu trình lệnh



- Bắt đầu mỗi chu trình lệnh, CPU sẽ nhận lệnh từ bộ nhớ chính đưa vào bên trong CPU.
- Bên trong CPU có 2 thanh ghi liên quan trực tiếp đến quá trình nhận lệnh:
 - Thanh ghi bộ đếm chương trình (Program Counter - PC): chứa địa chỉ của lệnh sẽ được nhận vào.
 - Thanh ghi lệnh (Instruction Register - IR): lệnh được nhận từ bộ nhớ chính sẽ được nạp vào IR.

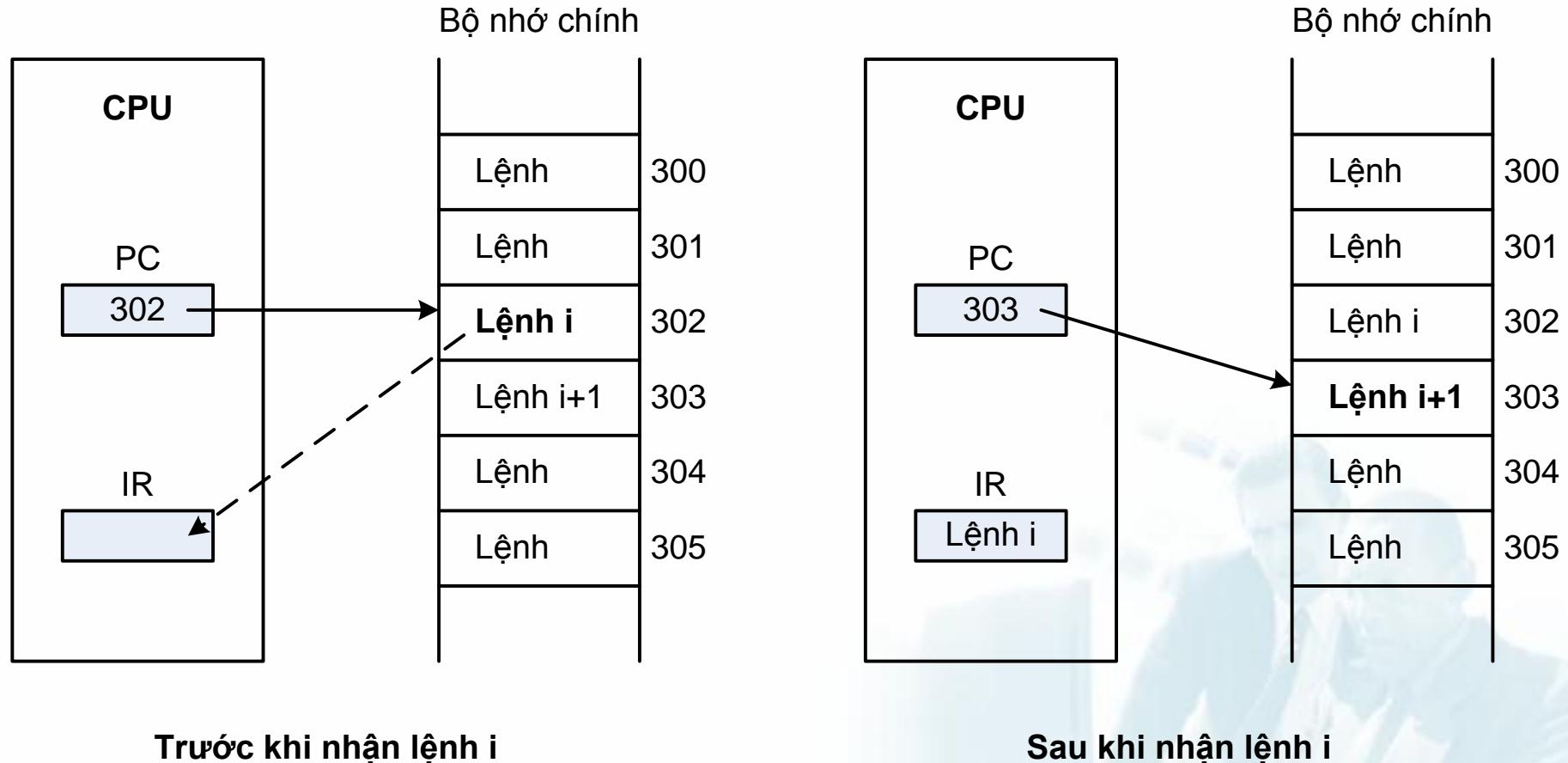


Nhận lệnh (tiếp)

- Hoạt động nhận lệnh diễn ra như sau:
 - CPU phát địa chỉ của lệnh cần nhận từ PC đến bộ nhớ chính
 - CPU phát tín hiệu điều khiển đọc bộ nhớ chính (MEMR - Memory Read)
 - Lệnh từ bộ nhớ chính được chuyển vào IR
 - Nội dung của PC tự động tăng để trả sang lệnh kế tiếp nằm ngay sau lệnh vừa được nhận.



Minh họa quá trình nhận lệnh



- Lệnh nằm ở IR sẽ được chuyển sang đơn vị điều khiển (Control Unit). Đơn vị điều khiển sẽ tiến hành giải mã lệnh và phát các tín hiệu điều khiển thực thi thao tác mà lệnh yêu cầu.
- Các kiểu thao tác của lệnh:
 - Trao đổi dữ liệu giữa CPU và bộ nhớ chính
 - Trao đổi dữ liệu giữa CPU và mô-đun vào-ra
 - Xử lý dữ liệu: thực hiện các phép toán số học hoặc phép toán logic với các dữ liệu
 - Điều khiển rẽ nhánh
 - Kết hợp các thao tác trên



Hoạt động ngắt

- Khái niệm chung về ngắt (Interrupt): Ngắt là cơ chế cho phép CPU tạm dừng chương trình đang thực hiện để chuyển sang thực hiện một chương trình khác, gọi là *chương trình con phục vụ ngắt*.
- Các loại ngắt:
 - Ngắt do lỗi khi thực hiện chương trình, ví dụ: tràn số, chia cho 0
 - Ngắt do lỗi phần cứng, ví dụ: lỗi bộ nhớ RAM
 - Ngắt do tín hiệu yêu cầu từ mô-đun vào-ra gửi đến CPU yêu cầu trao đổi dữ liệu

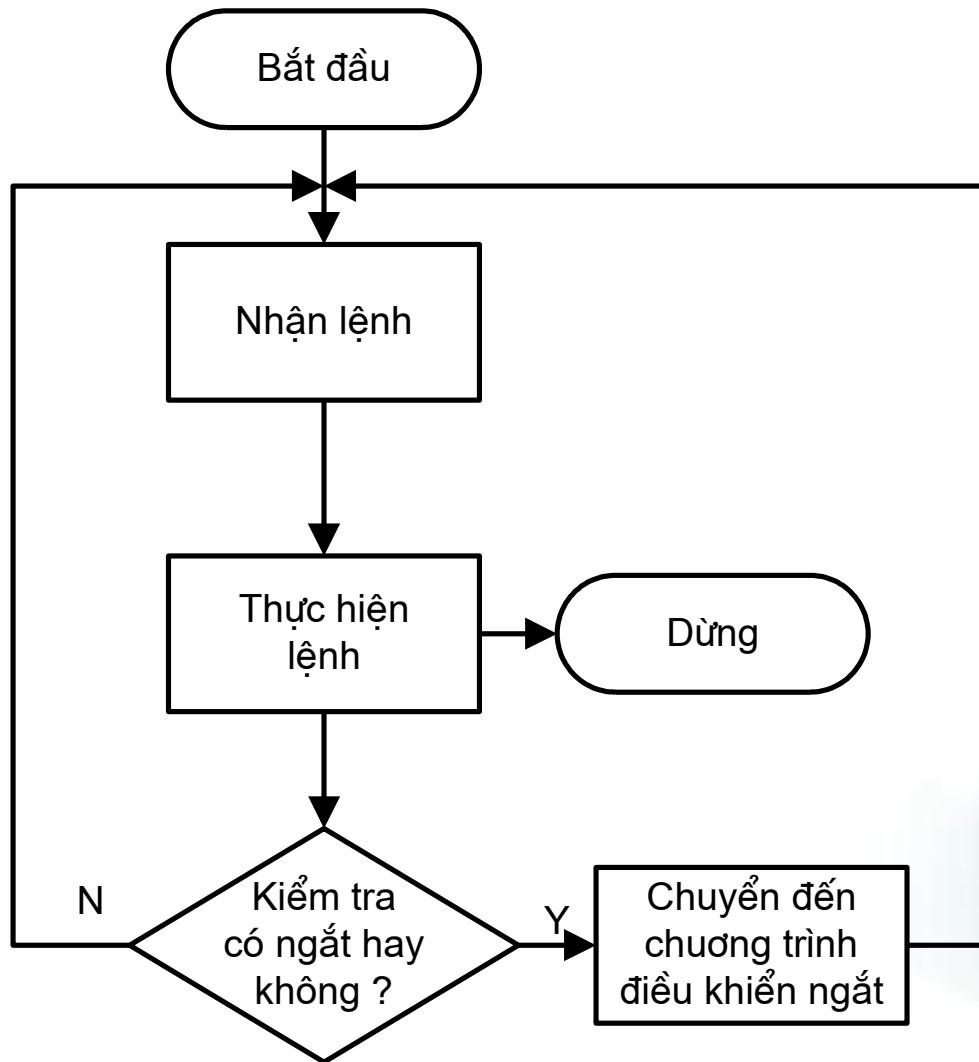


Chu trình xử lý ngắn

- Được thêm vào cuối chu trình lệnh
- Sau khi hoàn thành một lệnh, CPU kiểm tra xem có yêu cầu ngắn gửi đến hay không
 - Nếu không có tín hiệu yêu cầu ngắn thì CPU nhận lệnh kế tiếp
 - Nếu có yêu cầu ngắn và ngắn đó được chấp nhận thì:
 - CPU cất ngũ cảnh hiện tại của chương trình đang thực hiện (các thông tin liên quan đến chương trình bị ngắn)
 - CPU chuyển sang thực hiện chương trình con phục vụ ngắn tương ứng
 - Kết thúc chương trình con đó, CPU khôi phục lại ngũ cảnh và trở về tiếp tục thực hiện chương trình đang tạm dừng

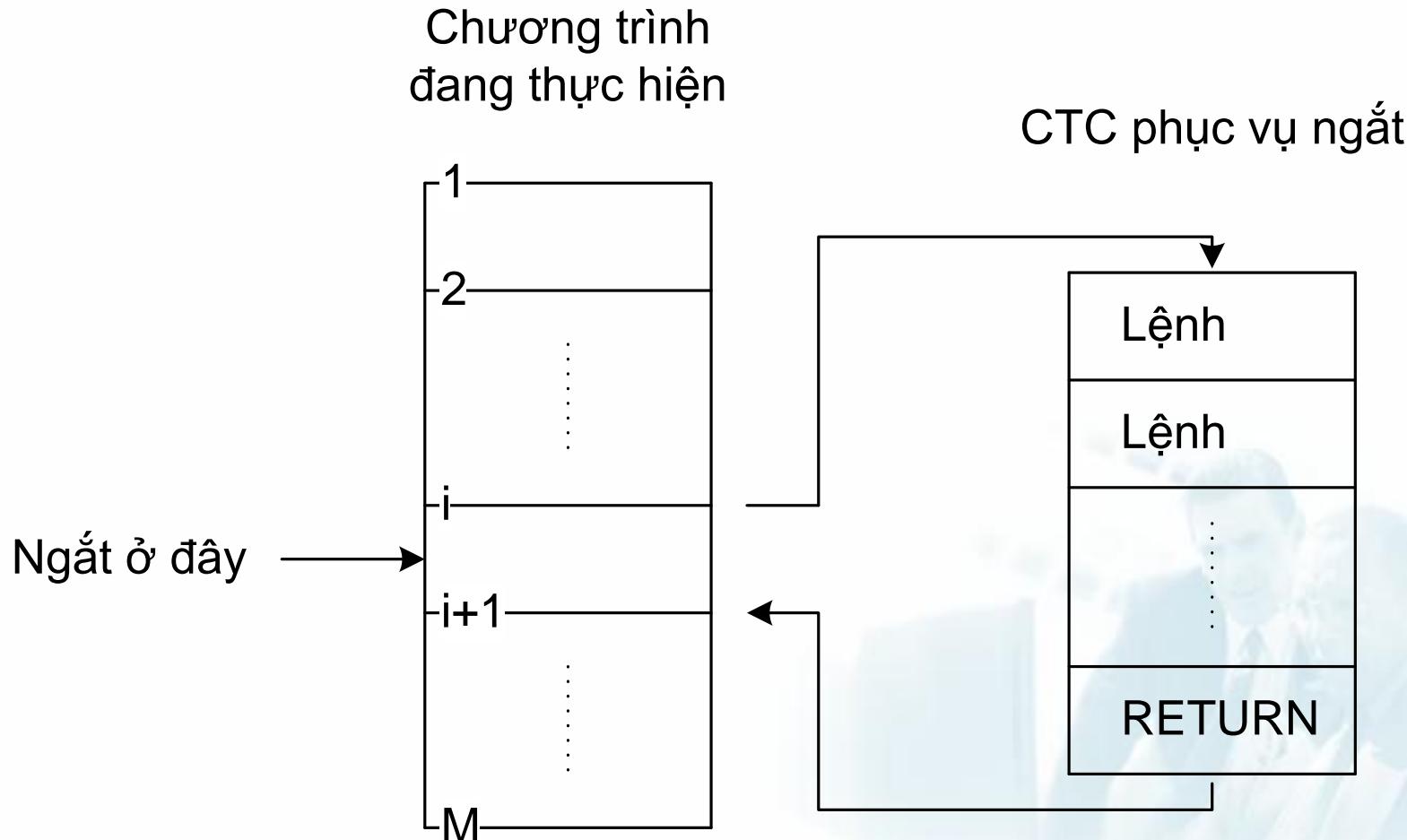


Hoạt động ngắt (tiếp)





Hoạt động ngắt (tiếp)





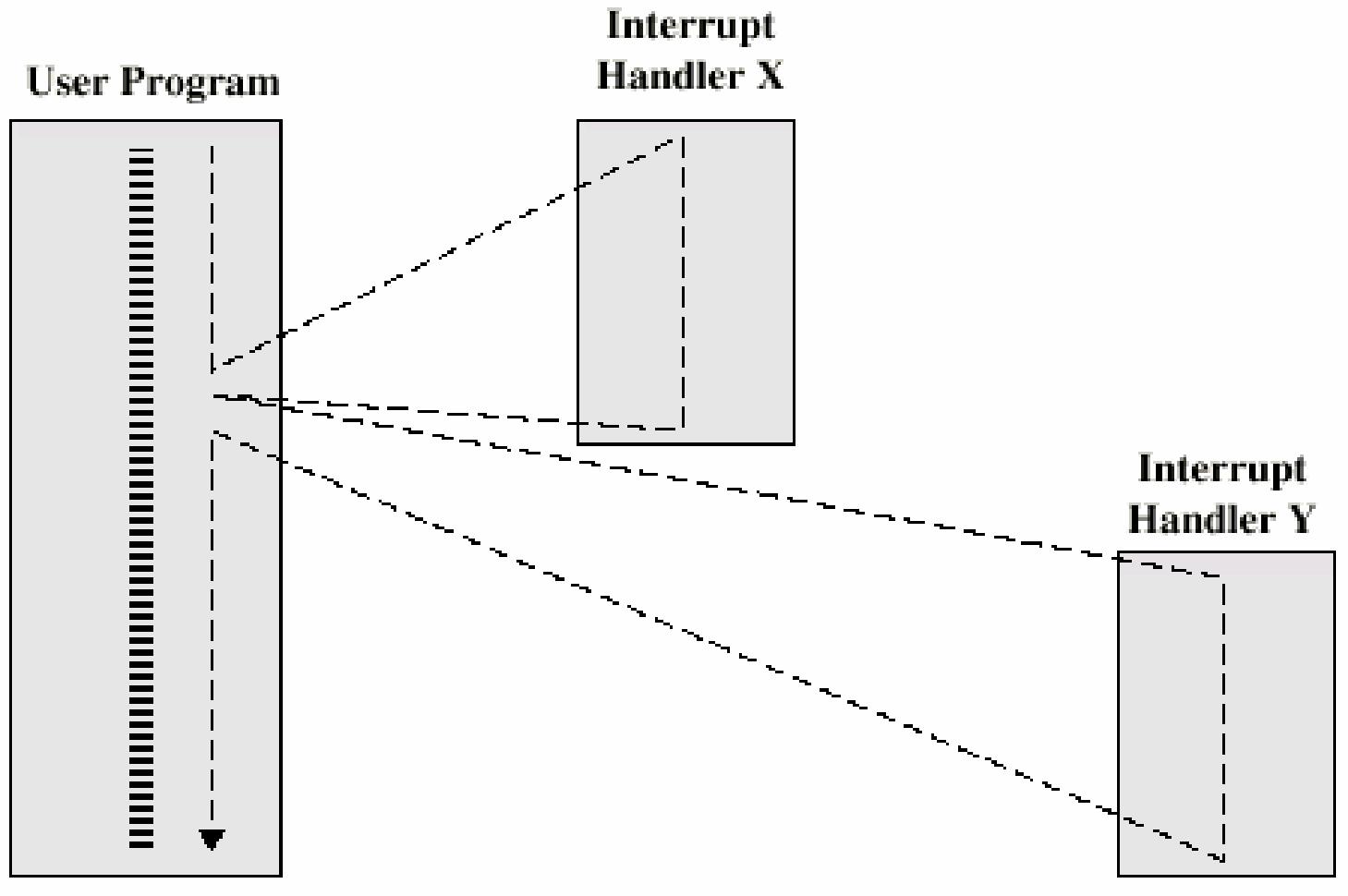
Xử lý với nhiều tín hiệu yêu cầu ngắt

- Xử lý ngắt tuần tự:

- Khi một ngắt đang được thực hiện, các ngắt khác sẽ bị cấm
- Bộ xử lý sẽ bỏ qua các ngắt tiếp theo trong khi đang xử lý một ngắt
- Các ngắt vẫn đang đợi và được kiểm tra sau khi ngắt đầu tiên được xử lý xong
- Các ngắt được thực hiện tuần tự



Xử lý ngắt tuần tự (tiếp)





Xử lý với nhiều tín hiệu yêu cầu ngắt

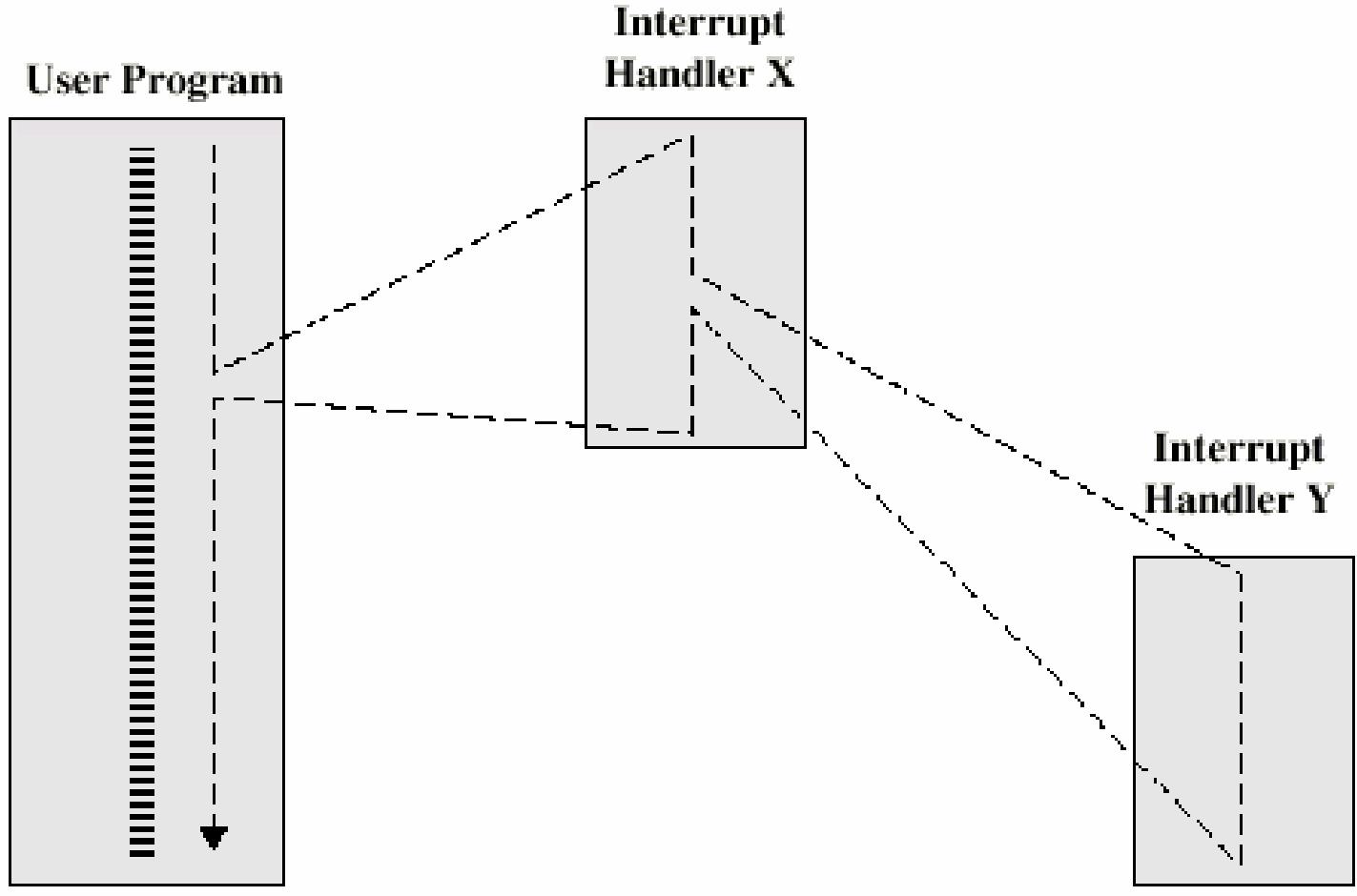
- Xử lý ngắt ưu tiên:

- Các ngắt được định nghĩa mức ưu tiên khác nhau
- Ngắt có mức ưu tiên thấp hơn có thể bị ngắt bởi ngắt ưu tiên cao hơn ⇒ ngắt xảy ra lồng nhau





Xử lý ngắt ưu tiên





Hoạt động vào-ra

- Là hoạt động trao đổi dữ liệu giữa thiết bị ngoại vi với bên trong máy tính.
- Các kiểu hoạt động vào-ra:
 - CPU trao đổi dữ liệu với mô-đun vào-ra
 - Mô-đun vào-ra trao đổi dữ liệu trực tiếp với bộ nhớ chính



3.1.3. Cấu trúc của MTCN điển hình

- Sơ đồ khối
- Các linh kiện trên bản mạch chính
- Các thiết bị ngoại vi cơ bản





Sơ đồ khối



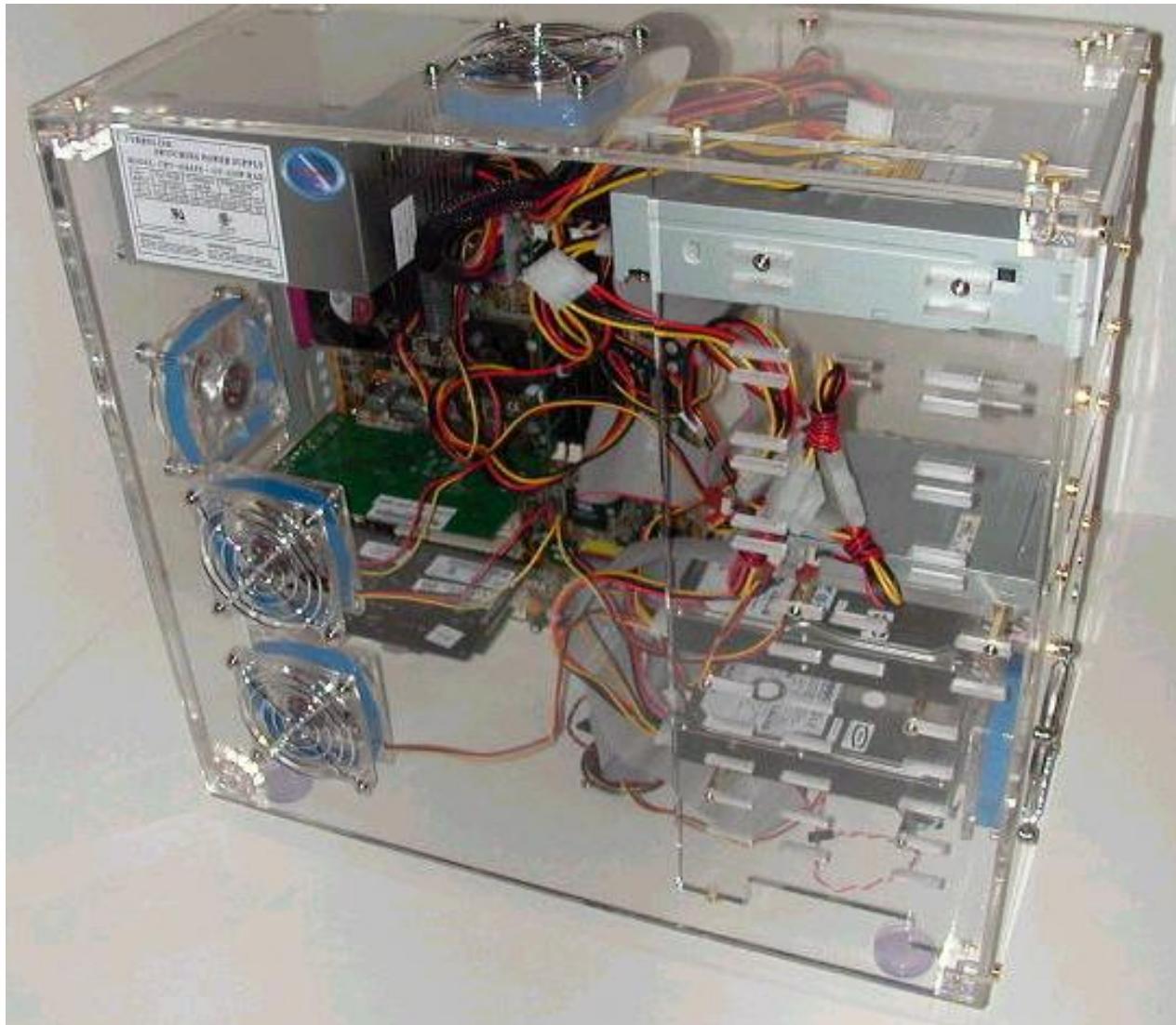


Sơ đồ khối (tiếp)

- **Hộp máy tính (Case):**
 - **Bản mạch chính (Mainboard):**
 - Bộ vi xử lý
 - Bộ nhớ hệ thống: chip nhớ ROM và các module nhớ RAM
 - Các vi mạch điều khiển tổng hợp (chipset)
 - Các khe cắm mở rộng
 - Các kênh truyền tín hiệu (bus)
 - Các loại ổ đĩa: ổ đĩa cứng, ổ đĩa mềm, ổ đĩa quang, ...
 - Các cổng vào-ra
 - Bộ nguồn và quạt
- **Các thiết bị ngoại vi (Peripheral Devices):**
 - Màn hình (monitor), bàn phím (keyboard), chuột (mouse), loa (speaker), máy in (printer), máy quét ảnh (scanner), modem, ...



Hộp máy tính (Case)

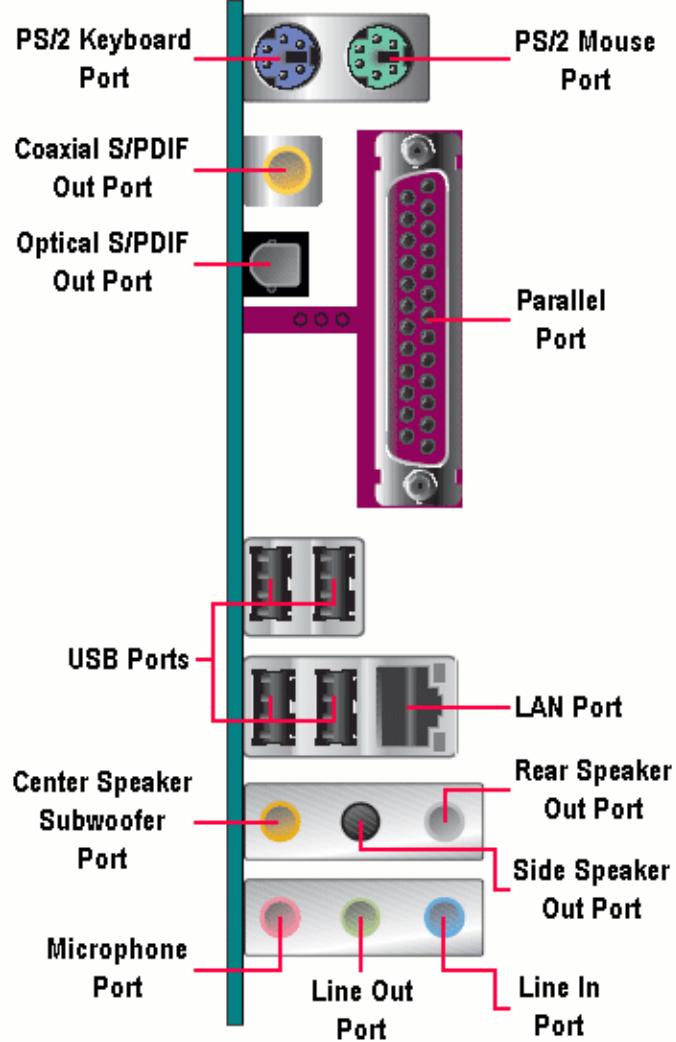




Các loại ổ đĩa



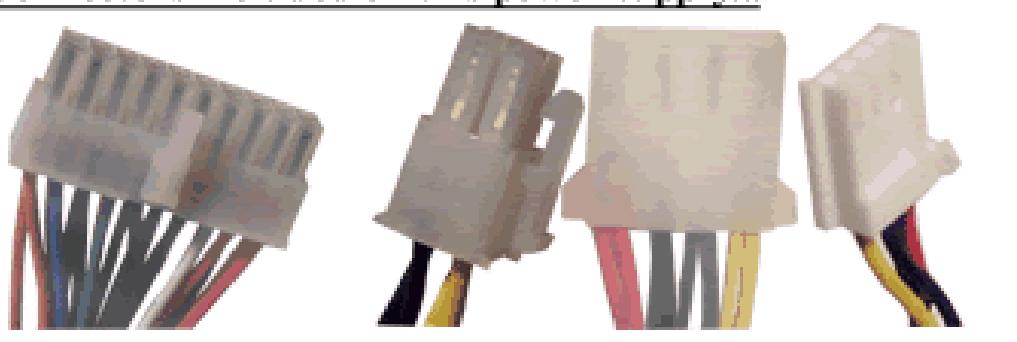
Các cổng vào-ra



Bộ nguồn và quạt



Connectors included on this power supply...



ATX 2.03X1

P4 ATX 12VX1

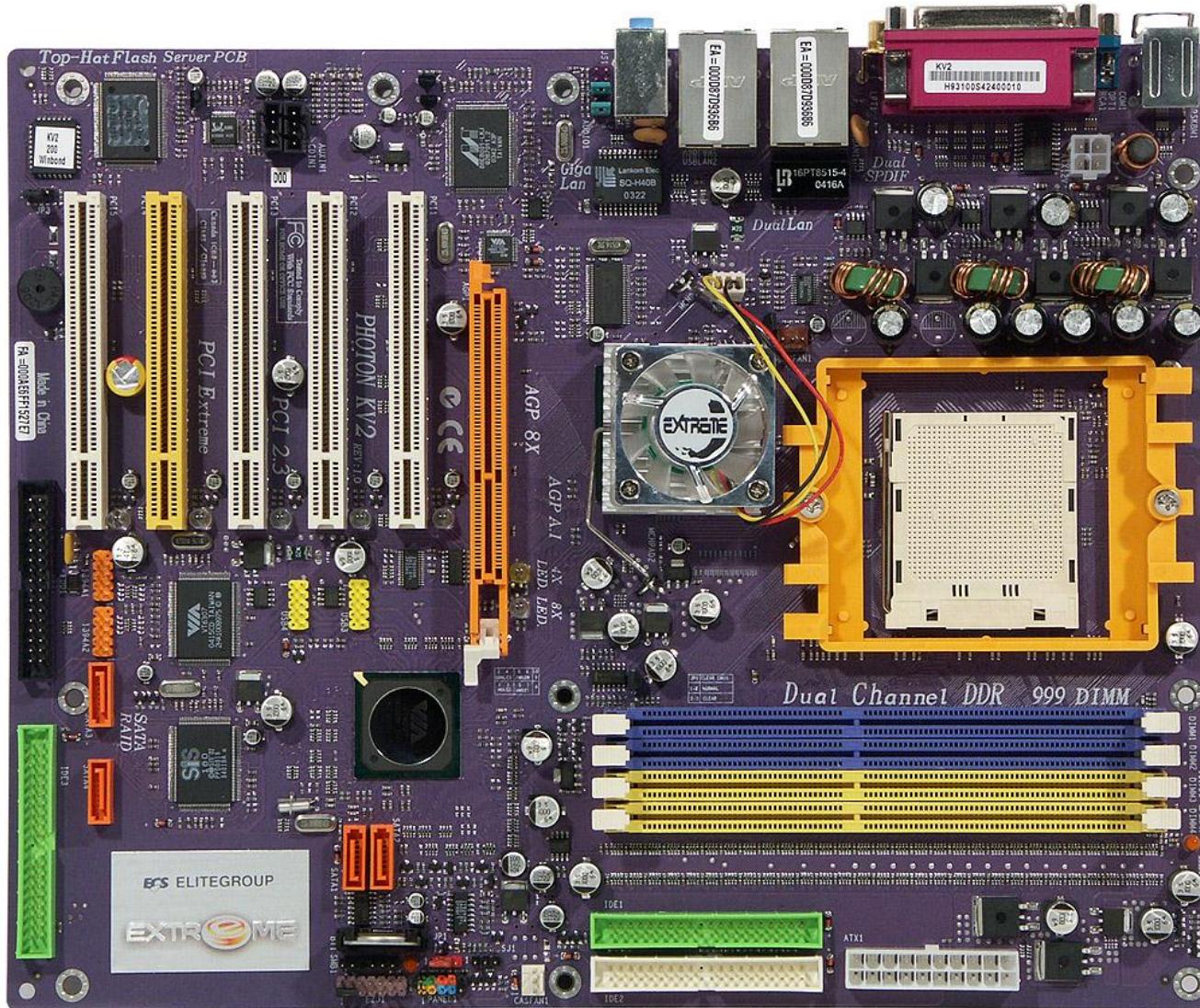
IDE 4 PINX4

FLOPPY 4 PINX1





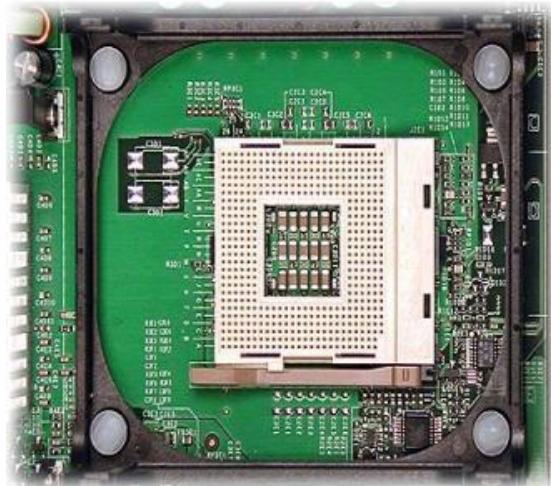
Các linh kiện trên bản mạch chính



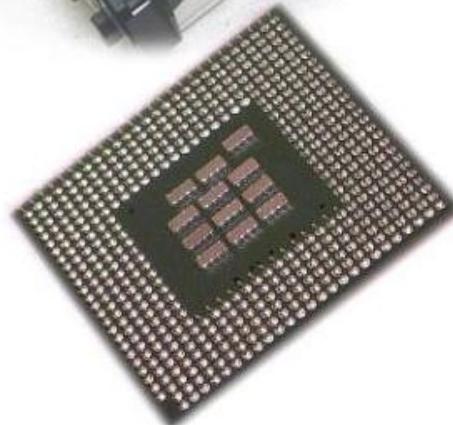
Bộ vi xử lý



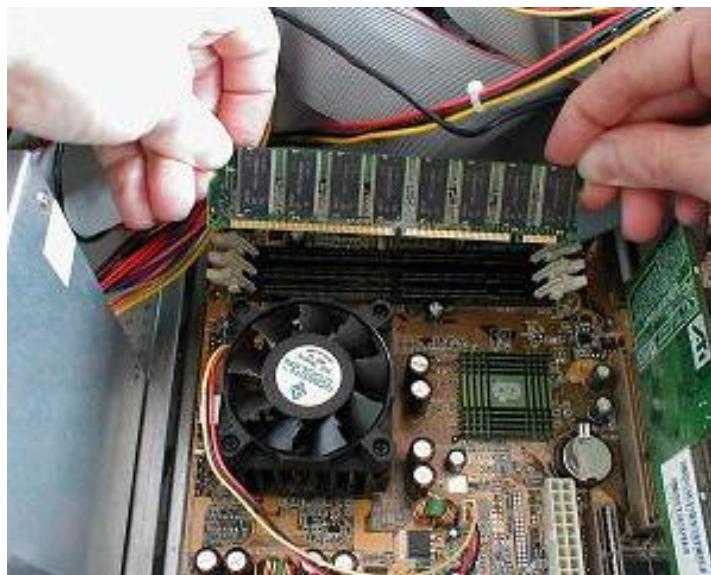
Pentum4/1.5GHz



Socket 478

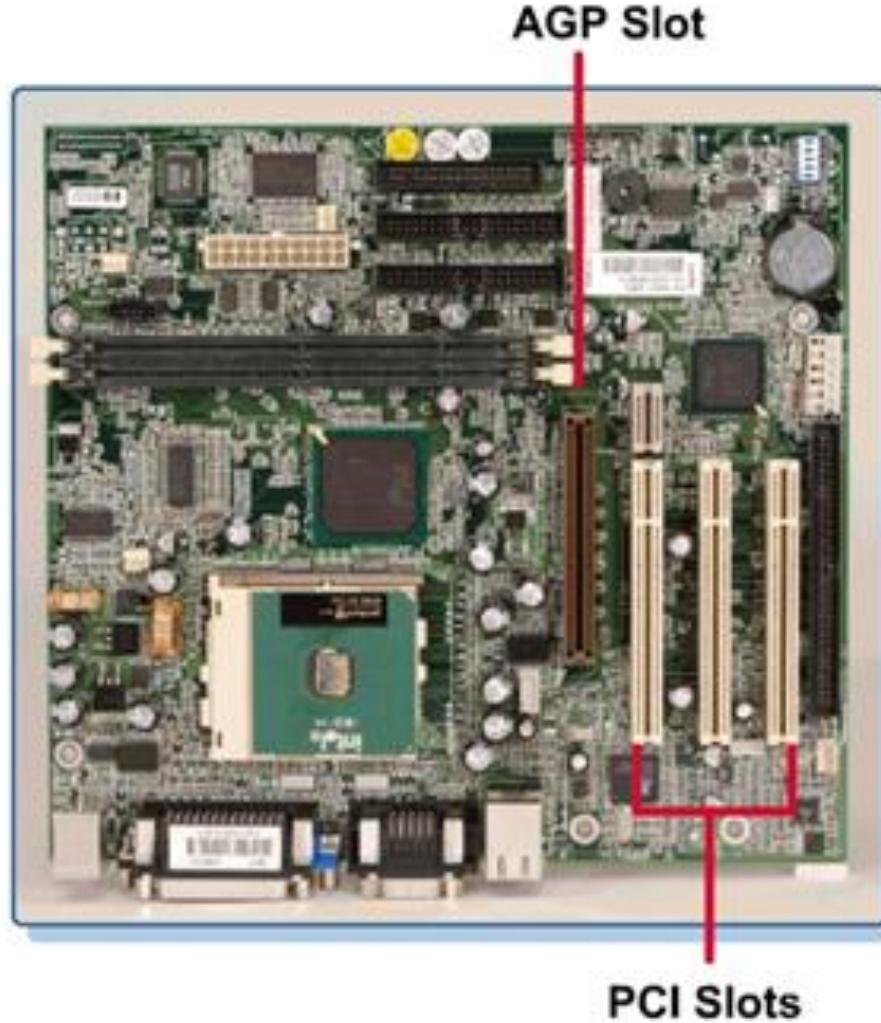


Bộ nhớ hệ thống





Các khe cắm mở rộng





Các thiết bị ngoại vi cơ bản





Các thiết bị ngoại vi (tiếp)





Các thiết bị ngoại vi (tiếp)





Các thiết bị ngoại vi (tiếp)



Còn tiếp !



3.2. Bộ xử lý trung tâm

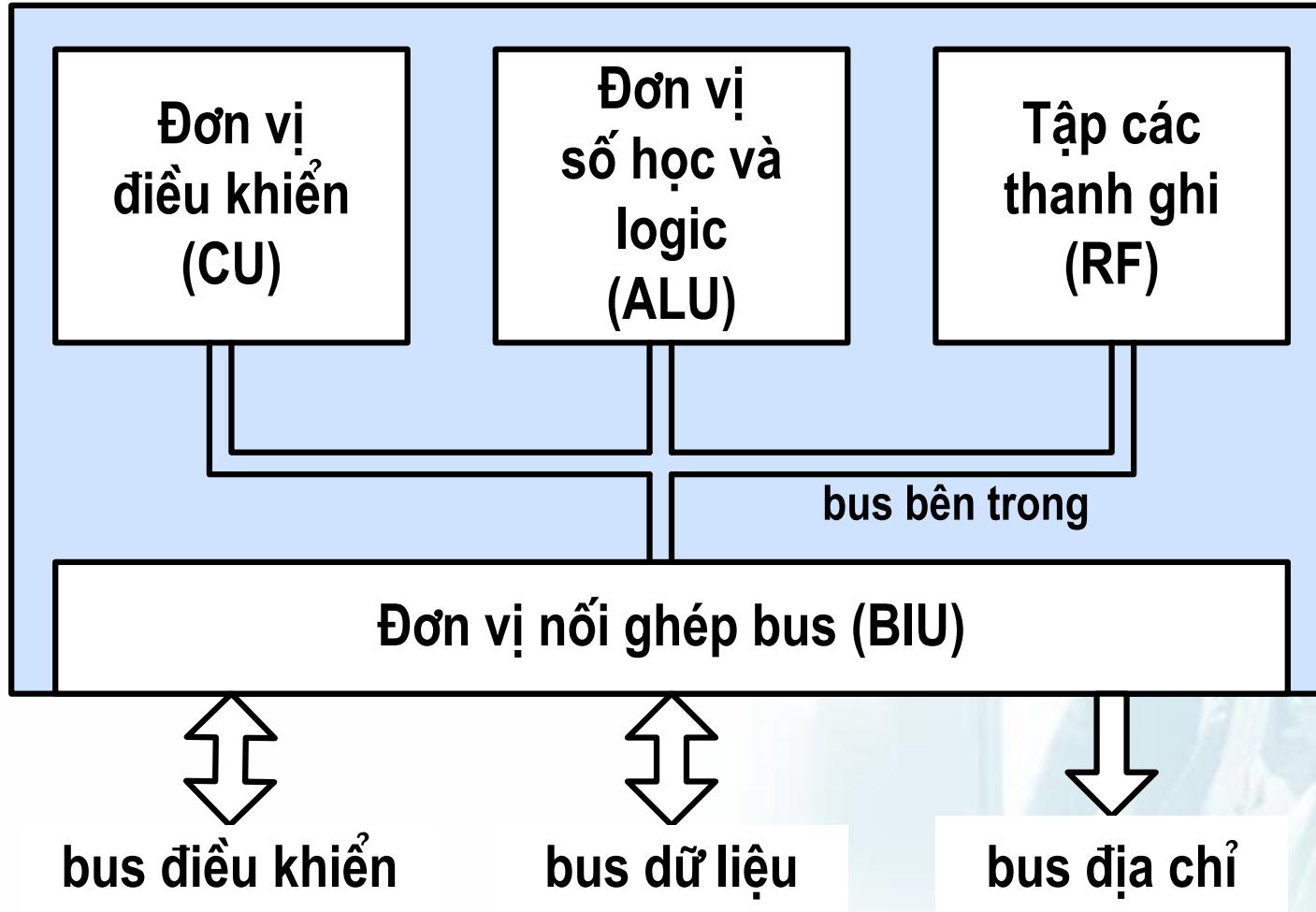
3.2.1. Cấu trúc cơ bản của CPU

3.2.2. Tập lệnh

3.2.3. Hoạt động của CPU



3.2.1. Cấu trúc cơ bản của CPU





1. Đơn vị điều khiển (CU)

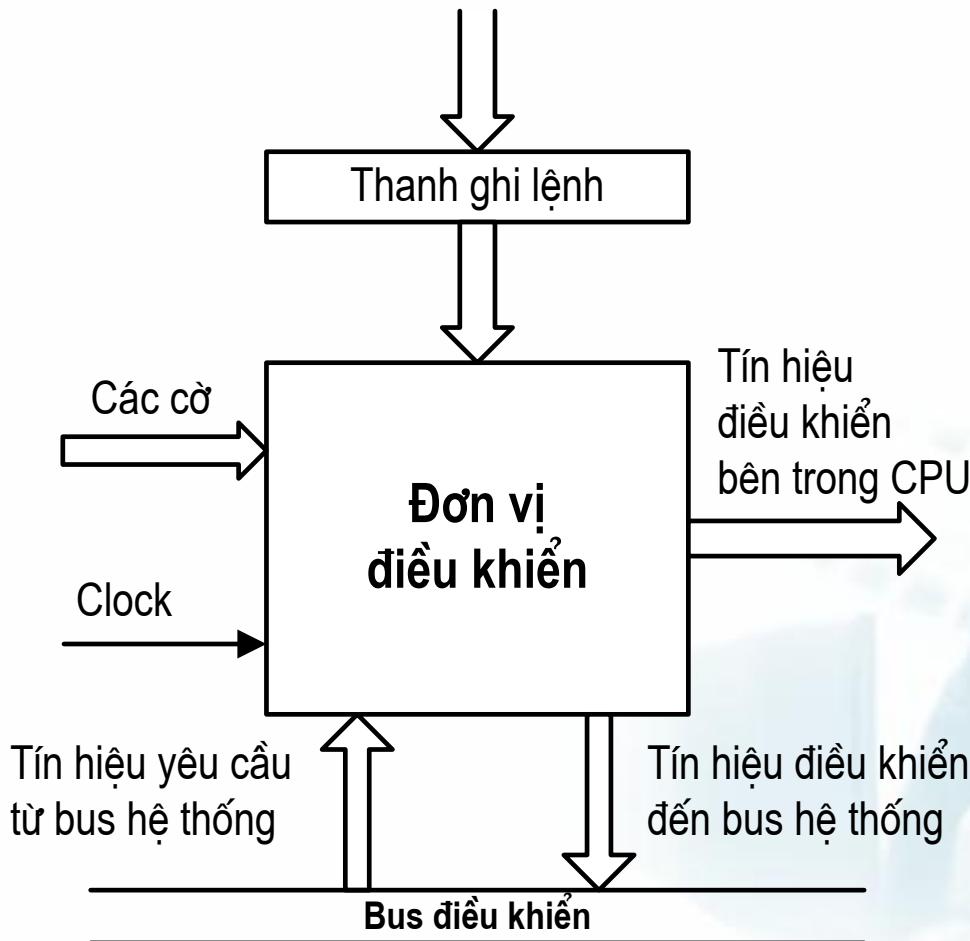
- Chức năng:

- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh và tăng nội dung của PC để trả sang lệnh kế tiếp.
- Giải mã lệnh nằm trong thanh ghi lệnh để xác định thao tác cần thực hiện và phát ra tín hiệu điều khiển thực hiện lệnh đó.
- Nhận tín hiệu yêu cầu từ bus hệ thống và đáp ứng với các yêu cầu đó.



Đơn vị điều khiển (tiếp)

- Mô hình kết nối của đơn vị điều khiển:





Đơn vị điều khiển (tiếp)

- Các tín hiệu đưa đến đơn vị điều khiển:
 - Mã lệnh từ thanh ghi lệnh đưa đến để giải mã
 - Các cờ từ thanh ghi cờ cho biết trạng thái của CPU
 - Xung clock từ bộ tạo xung bên ngoài cung cấp cho đơn vị điều khiển làm việc
 - Các tín hiệu yêu cầu từ bus điều khiển



Đơn vị điều khiển (tiếp)

- Các tín hiệu phát ra từ đơn vị điều khiển:
 - Các tín hiệu điều khiển bên trong CPU:
 - Điều khiển các thanh ghi
 - Điều khiển hoạt động của ALU
 - Các tín hiệu điều khiển bên ngoài CPU:
 - Điều khiển bộ nhớ chính
 - Điều khiển các module vào-ra



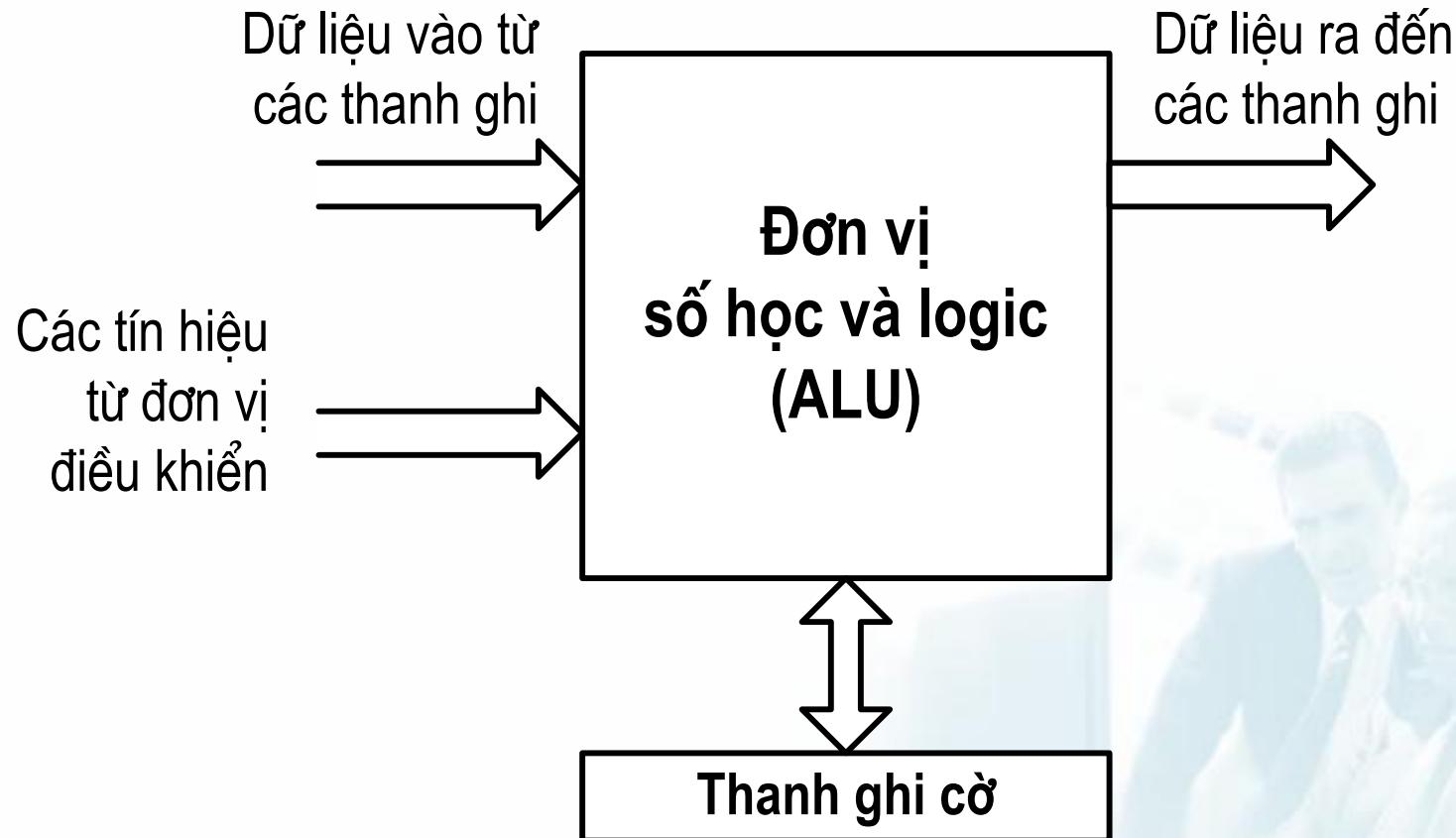
2. Đơn vị số học và logic (ALU)

- Chức năng: Thực hiện các phép toán số học và các phép toán logic.
 - Số học: cộng, trừ, nhân, chia, tăng, giảm, đảo dấu, ...
 - Logic: AND, OR, XOR, NOT, các phép dịch và quay bit



Đơn vị số học và logic (tiếp)

- Mô hình kết nối của ALU:





3. Tập thanh ghi (RF)

- a. Chức năng và phân loại
- b. Một số thanh ghi điển hình



a. Chức năng và phân loại

- Chức năng:

- Là tập hợp các thanh ghi nằm trong CPU
- Chứa các thông tin tạm thời phục vụ cho hoạt động hiện tại của CPU.





Phân loại tập thanh ghi

- Phân loại theo khả năng can thiệp của người lập trình:
 - Các thanh ghi không lập trình được: người lập trình không can thiệp được
 - Các thanh ghi lập trình được: người lập trình can thiệp được
- Phân loại theo chức năng:
 - Thanh ghi địa chỉ: quản lý địa chỉ của ngăn nhớ hay cổng vào-ra
 - Thanh ghi dữ liệu: chứa các dữ liệu tạm thời hoặc kết quả trung gian phục vụ cho việc xử lý dữ liệu của CPU
 - Thanh ghi điều khiển và trạng thái: chứa các thông tin điều khiển và trạng thái của CPU
 - Thanh ghi lệnh: chứa lệnh đang được thực hiện
 - Thanh ghi đa năng: có thể chứa địa chỉ hoặc dữ liệu



b. Một số thanh ghi điện hình

- Các thanh ghi địa chỉ
 - Bộ đếm chương trình (Program Counter – PC)
 - Con trỏ dữ liệu (Data Pointer – DP)
 - Con trỏ ngăn xếp (Stack Pointer – SP)
 - Thanh ghi cơ sở và thanh ghi chỉ số (Base Register & Index Register)
- Các thanh ghi dữ liệu
- Thanh ghi trạng thái





Các vùng nhớ cơ bản của CT

- Chương trình đang thực hiện phải nằm trong bộ nhớ chính và nó chiếm 3 vùng nhớ cơ bản sau:
 - Vùng nhớ lệnh (Code): chứa các lệnh của chương trình.
 - Vùng dữ liệu (Data): chứa dữ liệu của chương trình. Thực chất đây là nơi cấp phát các ngăn nhớ cho các biến nhớ.
 - Vùng ngăn xếp (Stack): là vùng nhớ có cấu trúc LIFO (Last In First Out) dùng để cất giữ thông tin và sau đó có thể khôi phục lại. Thường dùng cho việc thực hiện các chương trình con.



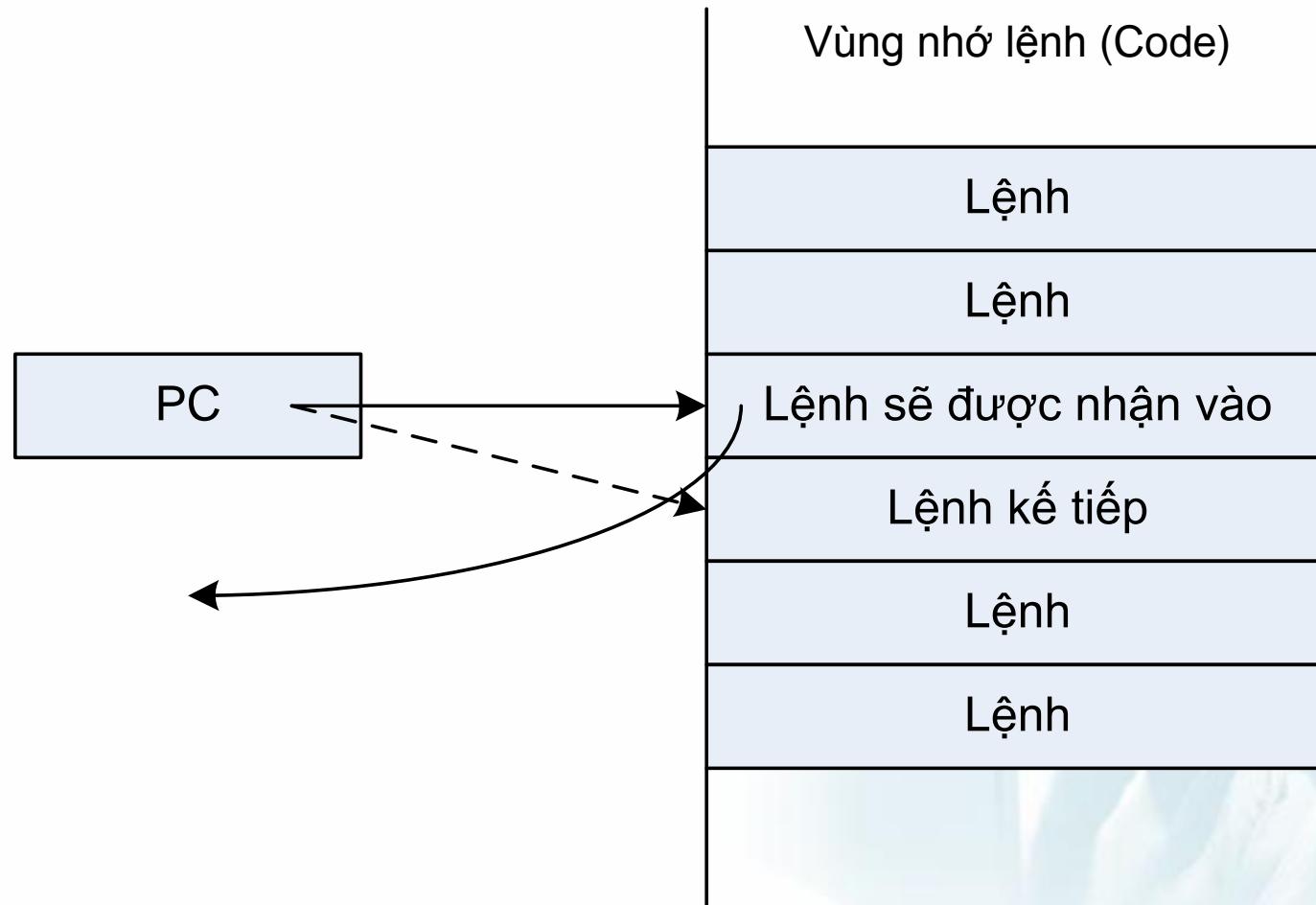
Bộ đếm chương trình (PC)

- Còn gọi là con trỏ lệnh (Instruction Pointer - IP)
- Là thanh ghi chứa địa chỉ của lệnh tiếp theo sẽ được nhận vào.
- Sau khi một lệnh được nhận vào thì nội dung của PC tự động tăng để trỏ sang lệnh kế tiếp nằm ngay sau lệnh vừa được nhận.





Minh họa hoạt động của PC





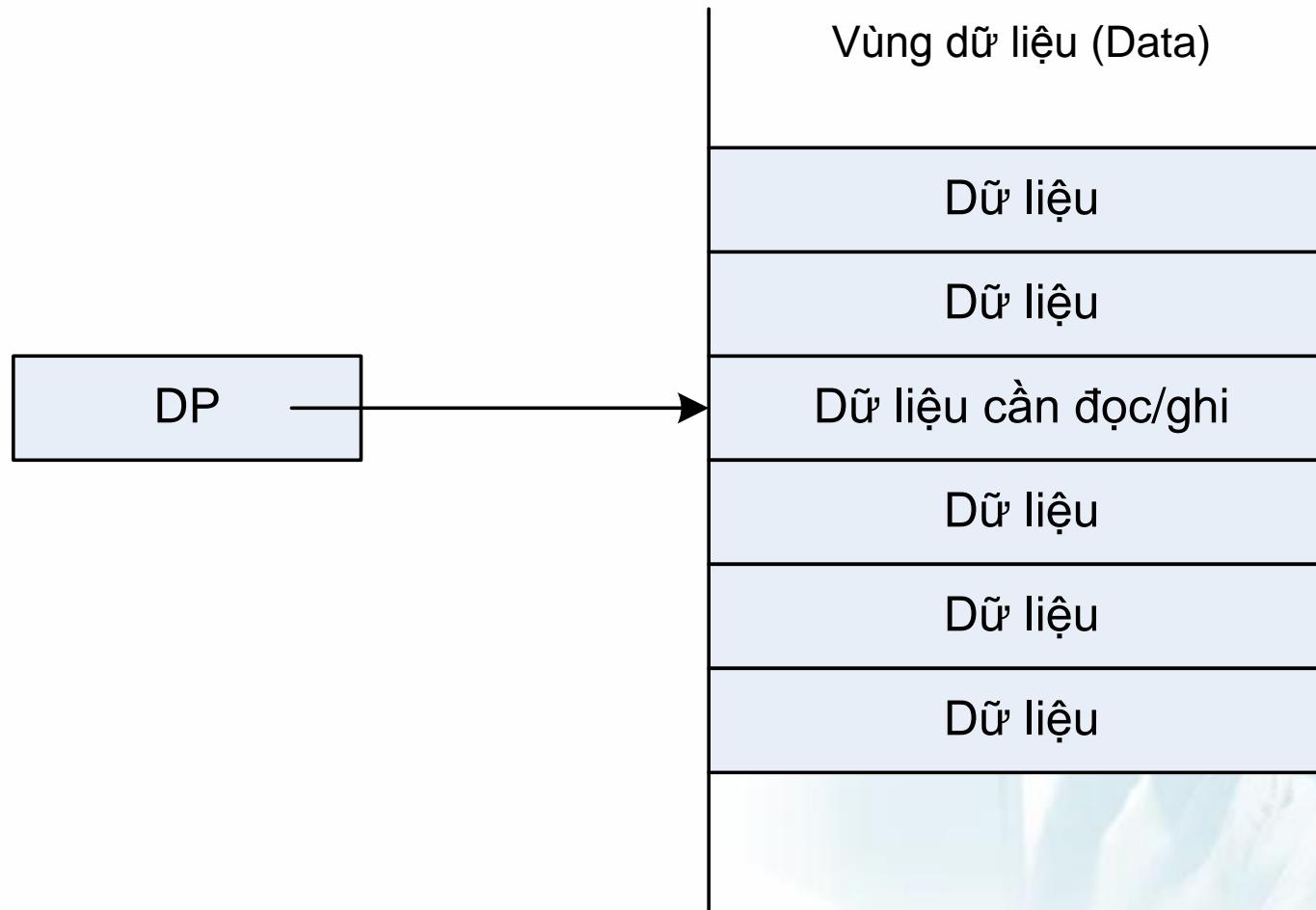
Thanh ghi con trỏ dữ liệu (DP)

- Chứa địa chỉ của ngăn nhớ dữ liệu mà CPU muốn truy cập.
- Thường có một số thanh ghi con trỏ dữ liệu.





Minh họa hoạt động của DP



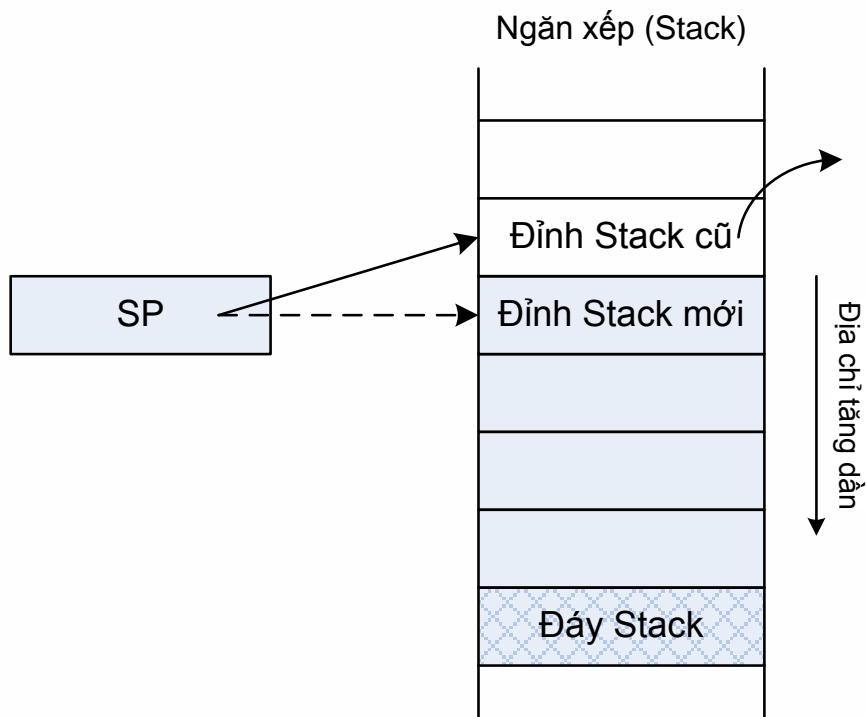


Con trỏ ngăn xếp (SP)

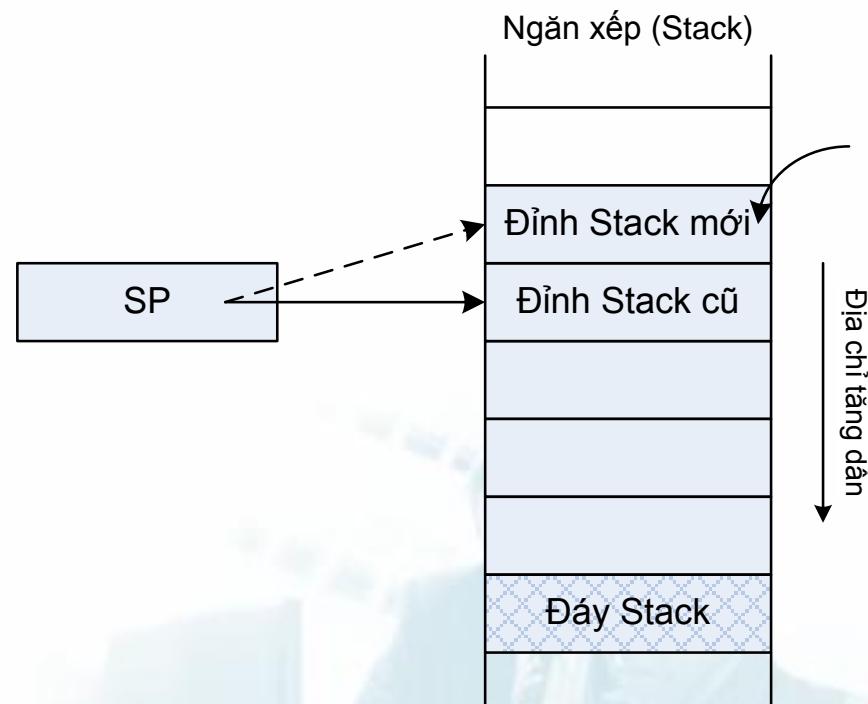
- Chứa địa chỉ của ngăn nhớ đỉnh ngăn xếp (ngăn xếp có chiều từ đáy lên đỉnh ngược với chiều tăng của địa chỉ)
- Khi cất thêm một thông tin vào ngăn xếp:
 - Nội dung của SP tự động giảm
 - Thông tin được cất vào bắt đầu từ ngăn nhớ trỏ bởi SP
- Khi lấy một thông tin ra khỏi ngăn xếp:
 - Thông tin được lấy ra bắt đầu từ ngăn nhớ trỏ bởi SP
 - Nội dung của SP tự động tăng
- Khi ngăn xếp rỗng: SP trỏ vào đáy ngăn xếp



Minh họa hoạt động của SP



Khi lấy 1 thông tin ra khỏi ngăn xếp, SP tự động tăng



Khi cất 1 thông tin vào ngăn xếp, SP tự động giảm

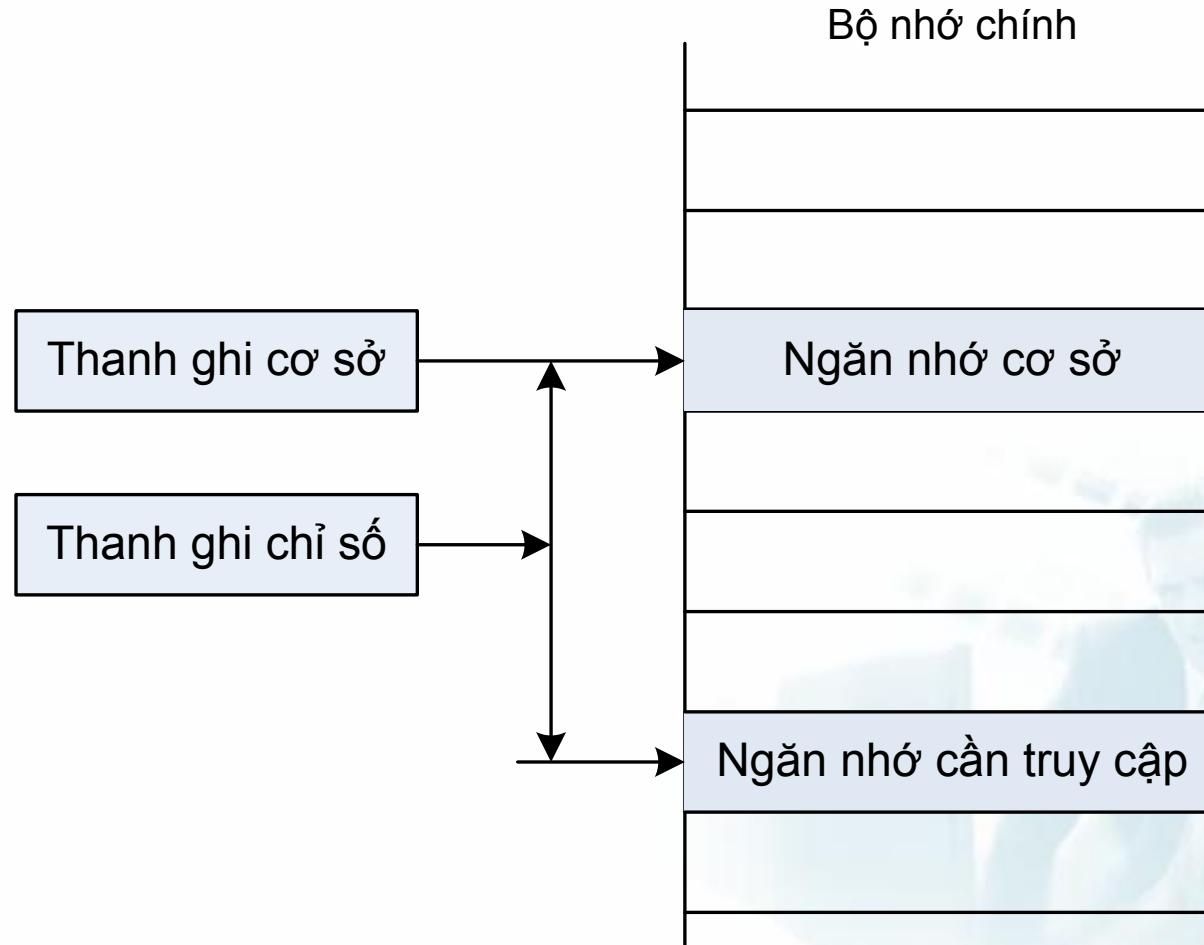


Thanh ghi cơ sở và thanh ghi chỉ số

- Thanh ghi cơ sở: chứa địa chỉ của ngăn nhớ cơ sở (địa chỉ cơ sở).
- Thanh ghi chỉ số: chứa độ lệch địa chỉ giữa ngăn nhớ mà CPU cần truy cập so với ngăn nhớ cơ sở (chỉ số).
- Địa chỉ của ngăn nhớ cần truy cập = địa chỉ cơ sở + chỉ số



Minh họa thanh ghi cơ sở và chỉ số





Các thanh ghi dữ liệu

- Chứa các dữ liệu tạm thời hoặc các kết quả trung gian phục vụ cho việc xử lý dữ liệu của CPU
- Cần có nhiều thanh ghi dữ liệu
- Các thanh ghi số nguyên: 8, 16, 32, 64 bit
- Các thanh ghi số dấu chấm động



Thanh ghi trạng thái

- Còn gọi là thanh ghi cờ (Flag Register)
- Chứa các thông tin trạng thái của CPU
 - Các cờ phép toán: biểu thị trạng thái của kết quả phép toán
 - Các cờ điều khiển: điều khiển chế độ làm việc của CPU



Ví dụ cờ phép toán

- Cờ Zero (ZF - cờ rỗng): được thiết lập lên 1 khi kết quả của phép toán vừa thực hiện xong bằng 0.
- Cờ Sign (SF - cờ dấu): được thiết lập lên 1 khi kết quả của phép toán vừa thực hiện nhỏ hơn 0, hay nói cách khác, cờ Sign nhận giá trị bằng bit dấu của kết quả.
- Cờ Carry (CF - cờ nhớ): được thiết lập lên 1 nếu phép toán xảy ra hiện tượng carry-out.
- Cờ Overflow (OF - cờ tràn): được thiết lập lên 1 nếu phép toán xảy ra hiện tượng overflow.



Ví dụ cờ điều khiển

- Cờ Interrupt (IF - cờ cho phép ngắt):
 - Nếu IF = 1 thì CPU ở trạng thái cho phép ngắt với tín hiệu yêu cầu ngắt từ bên ngoài gửi tới.
 - Nếu IF = 0 thì CPU ở trạng thái cấm ngắt với tín hiệu yêu cầu ngắt từ bên ngoài.

- Giả sử có các biến nhớ a, b, c, d, e, f thuộc kiểu số nguyên có dấu 8 bit. Các biến a, b được gán giá trị như sau:

$$a:=-58 \quad b:=72$$

Hãy biểu diễn các phép tính sau đây dưới dạng số nhị phân và cho biết kết quả dạng thập phân cùng với giá trị của các cờ ZF, SF, CF, OF tương ứng.

$$c:=a-b \quad d:=a+b \quad e:=b-a \quad f:=-a-b$$

3.2.1. Cấu trúc cơ bản của CPU

3.2.2. Tập lệnh

3.2.3. Hoạt động của CPU



3.2.2. Tập lệnh

1. Giới thiệu chung về tập lệnh
2. Các kiểu thao tác điển hình
3. Các phương pháp địa chỉ hóa toán hạng



1. Giới thiệu chung về tập lệnh

- Mỗi bộ xử lý có một tập lệnh xác định (mang tính kế thừa trong cùng một dòng họ).
- Tập lệnh thường có hàng chục đến hàng trăm lệnh.
- Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.
- Các lệnh được mô tả bằng các kí hiệu gợi nhớ → các lệnh hợp ngữ.



Khuôn dạng của một lệnh máy

Mã thao tác

Tham chiếu toán hạng

- Mã thao tác (Operation Code - Opcode): mã hóa cho thao tác mà CPU phải thực hiện.
- Tham chiếu toán hạng: mã hóa cho toán hạng hoặc nơi chứa toán hạng mà thao tác sẽ tác động.
 - Toán hạng nguồn (Source Operand): dữ liệu vào của thao tác (CPU sẽ đọc)
 - Toán hạng đích (Destination Operand): dữ liệu ra của thao tác (CPU sẽ ghi)



Số lượng toán hạng trong lệnh

- Ba toán hạng:

- 2 toán hạng nguồn, 1 toán hạng đích
- VD: $c = a + b$
- Từ lệnh dài vì phải mã hóa địa chỉ cho cả 3 toán hạng
- Thường được sử dụng trên các bộ xử lý tiên tiến

- Hai toán hạng:

- 1 toán hạng là toán hạng nguồn, toán hạng còn lại vừa là nguồn vừa là đích.
- VD: $a = a + b$
- Giá trị cũ của 1 toán hạng nguồn sẽ bị ghi đè bằng KQ
- Rút gọn độ dài từ lệnh
- Thường được dùng phổ biến



Số lượng toán hạng trong lệnh (tiếp)

■ Một toán hạng:

- Chỉ có 1 toán hạng được chỉ ra trong lệnh
- Toán hạng còn lại được ngầm định, thường là thanh ghi (VD: thanh chứa – Accumulator)
- Thường được sử dụng trên các bộ xử lý thế hệ cũ

■ Không có toán hạng:

- Các toán hạng đều được ngầm định
- Sử dụng Stack
- VD: lệnh $c = a + b$

push a

push b

add

pop c

- Không thông dụng





2. Các kiểu thao tác điển hình

- Chuyển dữ liệu
- Xử lý số học với số nguyên
- Xử lý logic
- Điều khiển vào-ra
- Chuyển điều khiển (rẽ nhánh)
- Điều khiển hệ thống



Các lệnh chuyển dữ liệu

MOVE	Copy dữ liệu từ nguồn đến đích
LOAD	Copy dữ liệu từ bộ nhớ đến bộ xử lý
STORE	Copy dữ liệu từ bộ xử lý đến bộ nhớ
EXCHANGE	Tráo đổi nội dung của nguồn và đích
CLEAR	Chuyển các bit 0 vào toán hạng đích
SET	Chuyển các bit 1 vào toán hạng đích
PUSH	Copy dữ liệu từ nguồn đến đindh ngăn xếp
POP	Copy dữ liệu từ đindh ngăn xếp đến đích



Các lệnh số học

ADD	Tính tổng hai toán hạng
SUBTRACT	Tính hiệu hai toán hạng
MULTIPLY	Tính tích hai toán hạng
DIVIDE	Tính thương hai toán hạng
ABSOLUTE	Thay toán hạng bằng trị tuyệt đối của nó
NEGATE	Đổi dấu toán hạng (lấy bù 2)
INCREMENT	Cộng 1 vào toán hạng
DECREMENT	Trừ toán hạng đi 1
COMPARE	So sánh hai toán hạng để lập cờ



Các lệnh logic

AND	Thực hiện phép AND hai toán hạng
OR	Thực hiện phép OR hai toán hạng
XOR	Thực hiện phép XOR hai toán hạng
NOT	Đảo bit của toán hạng (lấy bù 1)
TEST	Thực hiện phép AND hai toán hạng để lập cờ
SHIFT	Dịch trái (phải) toán hạng
ROTATE	Quay trái (phải) toán hạng
CONVERT	Chuyển đổi dữ liệu từ dạng này sang dạng khác



VD các lệnh AND, OR, XOR, NOT

- Giả sử có hai thanh ghi chứa dữ liệu như sau:

$$(R1) = 1010\ 1010$$

$$(R2) = 0000\ 1111$$

- Khi đó ta có:

$$(R1) \text{ AND } (R2) = 0000\ 1010$$

- Phép toán AND có thể được dùng để xoá một số bit và giữ nguyên các bit còn lại của toán hạng.

$$(R1) \text{ OR } (R2) = 1010\ 1111$$

- Phép toán OR có thể được dùng để thiết lập một số bit và giữ nguyên các bit còn lại của toán hạng.

$$(R1) \text{ XOR } (R2) = 1010\ 0101$$

- Phép toán XOR có thể được dùng để đảo một số bit và giữ nguyên các bit còn lại của toán hạng.

$$\text{NOT } (R1) = 0101\ 0101$$

- Phép toán NOT dùng để đảo tất cả các bit của toán hạng.



Các lệnh SHIFT và ROTATE

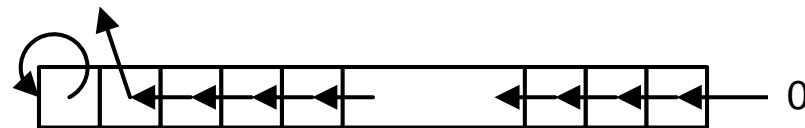
Dịch trái logic



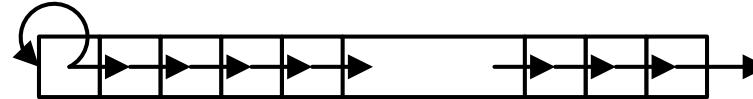
Dịch phải logic



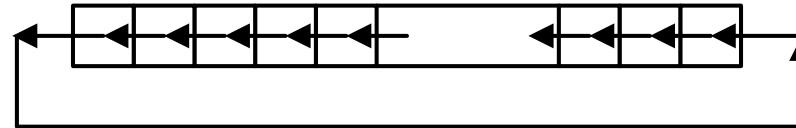
Dịch trái số học



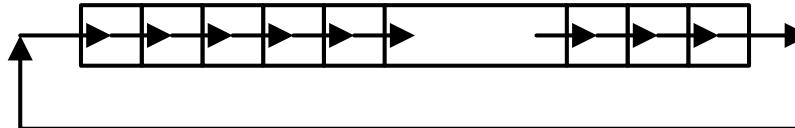
Dịch phải số học



Quay trái logic



Quay phải logic





Các lệnh vào-ra chuyên dụng

IN

Copy dữ liệu từ một cổng xác định đến đích

OUT

Copy dữ liệu từ nguồn đến một cổng xác định



Các lệnh chuyển điều khiển

JUMP (BRANCH)

Nhảy (rẽ nhánh) không điều kiện; nạp vào PC một địa chỉ xác định

JUMP CONDITIONAL

Kiểm tra điều kiện xác định, hoặc nạp vào PC một địa chỉ xác định hoặc không làm gì cả

CALL

Cắt nội dung PC vào ngăn xếp, nạp vào PC địa chỉ xác định để nhảy đến thực hiện chương trình con

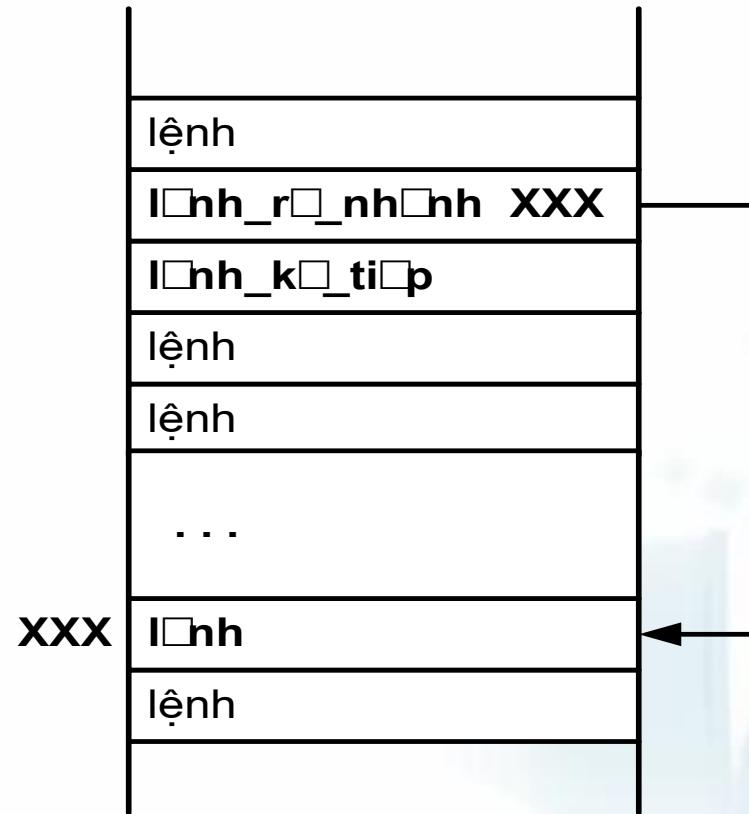
RETURN

Khôi phục nội dung PC từ đỉnh ngăn xếp để trở về chương trình chính



Lệnh rẽ nhánh không điều kiện

- Chuyển tới thực hiện lệnh ở vị trí có địa chỉ là XXX:
PC ← XXX



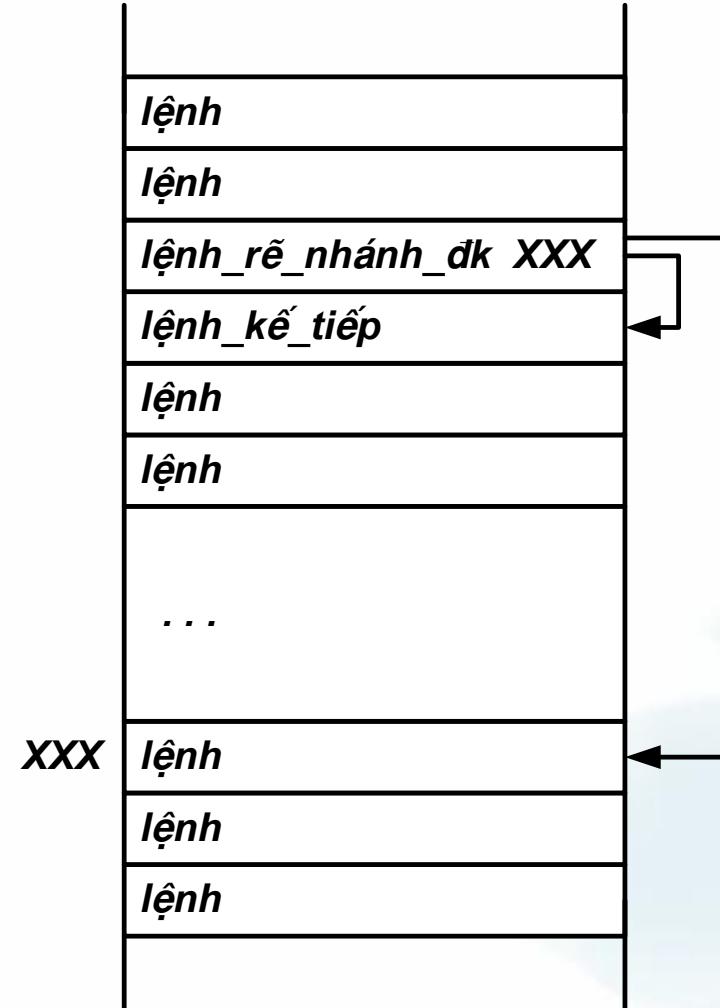


Lệnh rẽ nhánh có điều kiện

- Kiểm tra điều kiện trong lệnh:
 - Nếu điều kiện đúng → chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX
PC ← XXX
 - Nếu điều kiện sai → chuyển sang thực hiện **lệnh_kế_tiếp**
- Điều kiện thường được kiểm tra thông qua các cờ.
- Có nhiều lệnh rẽ nhánh có điều kiện.



Minh họa lệnh rẽ nhánh có điều kiện



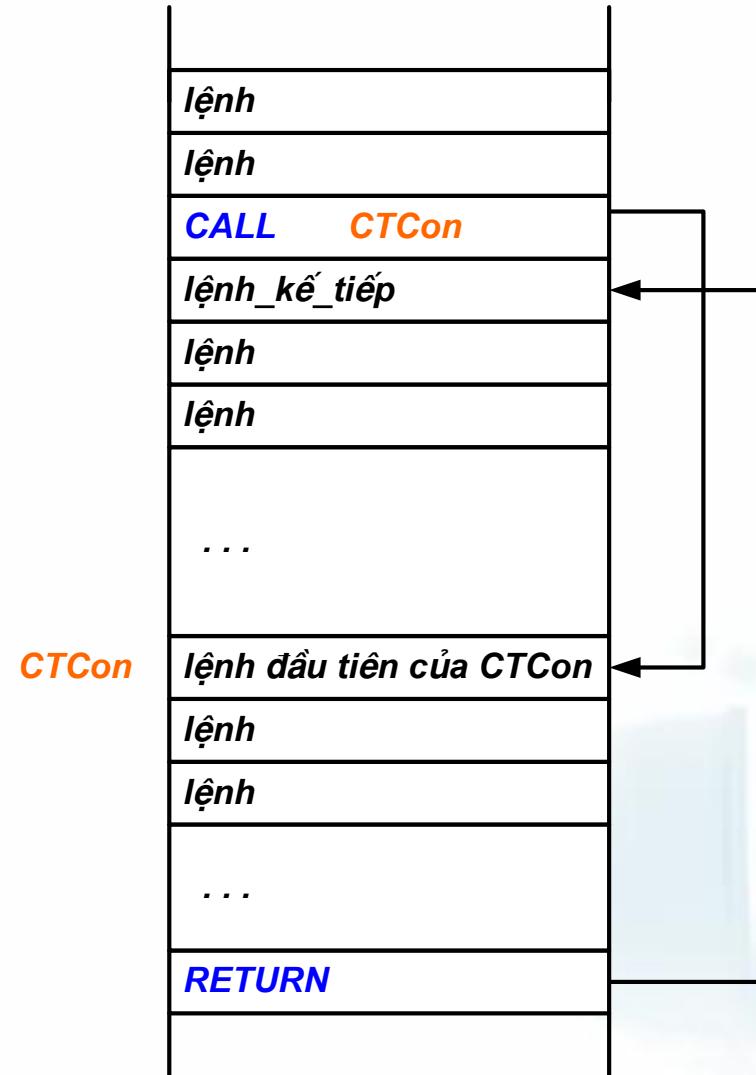


Lệnh CALL và RETURN

- Lệnh gọi chương trình con: lệnh CALL
 - Cắt nội dung PC (chứa địa chỉ của lệnh _kế _tiếp) vào Stack
 - Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con được gọi
 - Bộ xử lý chuyển sang thực hiện chương trình con tương ứng
- Lệnh trả về từ chương trình con: lệnh RETURN
 - Lấy địa chỉ của lệnh _kế _tiếp được cắt ở Stack nạp trả lại cho PC
 - Bộ xử lý được điều khiển quay trở về thực hiện tiếp lệnh nằm sau lệnh CALL



Minh họa lệnh CALL và RETURN





Các lệnh điều khiển hệ thống

HALT	Dừng thực hiện chương trình
WAIT	Dừng thực hiện chương trình, lặp kiểm tra điều kiện cho đến khi thoả mãn thì tiếp tục thực hiện
NO OPERATION (NOP)	Không thực hiện gì cả
LOCK	Cấm không cho xin chuyển nhượng bus
UNLOCK	Cho phép xin chuyển nhượng bus



3. Các phương pháp địa chỉ hóa toán hạng

- Phương pháp địa chỉ hóa toán hạng là cách thức chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động.
- Toán hạng có thể là:
 - Hằng số → cần cho biết giá trị của hằng số đó
 - Nội dung của một thanh ghi bên trong CPU → cần cho biết tên của thanh ghi
 - Nội dung của một ngăn nhớ → cần cho biết địa chỉ ngăn nhớ
 - Nội dung của một cổng vào-ra → cần cho biết địa chỉ của cổng vào-ra



Các chế độ địa chỉ thông dụng

- Chế độ địa chỉ tức thì
- Chế độ địa chỉ thanh ghi
- Chế độ địa chỉ trực tiếp
- Chế độ địa chỉ gián tiếp qua thanh ghi
- Chế độ địa chỉ dịch chuyển



Chế độ địa chỉ tức thì



- Immediate Addressing Mode
- Toán hạng là một hằng số ở ngay trong lệnh
- Ví dụ:

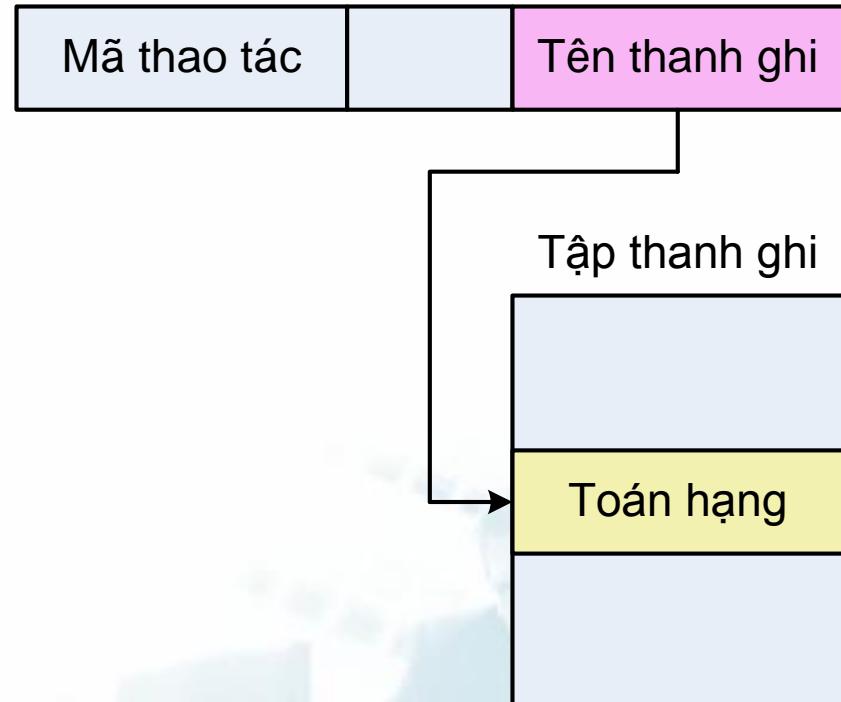
ADD AX, 5 ; AX \leftarrow AX + 5

- Truy nhập toán hạng rất nhanh



Chế độ địa chỉ thanh ghi

- Register Addressing Mode
- Toán hạng là nội dung của một thanh ghi mà tên thanh ghi được cho biết ở trong lệnh.
- Ví dụ:
MOV AX, BX ; AX ← BX
- Tốc độ truy cập nhanh hơn so với những lệnh có truy cập đến bộ nhớ.



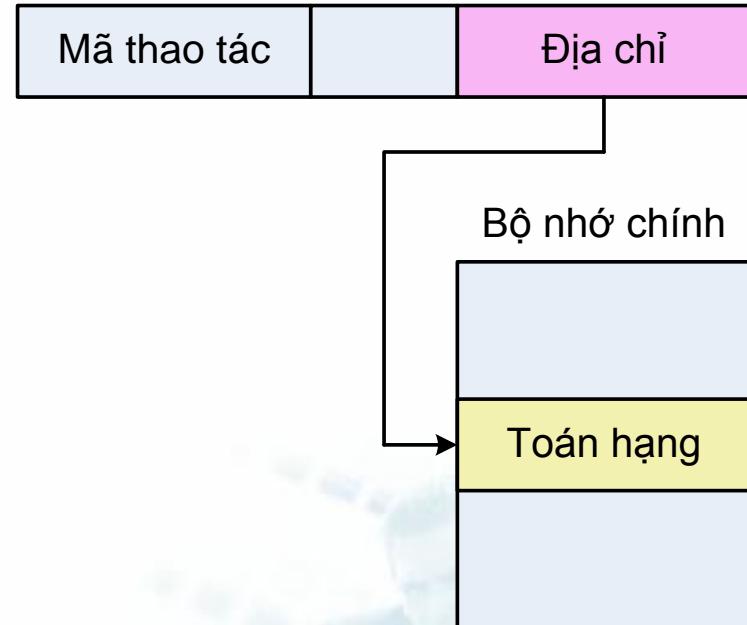


Chế độ địa chỉ trực tiếp

- Direct Addressing Mode
- Toán hạng là nội dung của một ngăn nhớ mà địa chỉ ngăn nhớ được cho trực tiếp ở trong lệnh.
- Ví dụ:

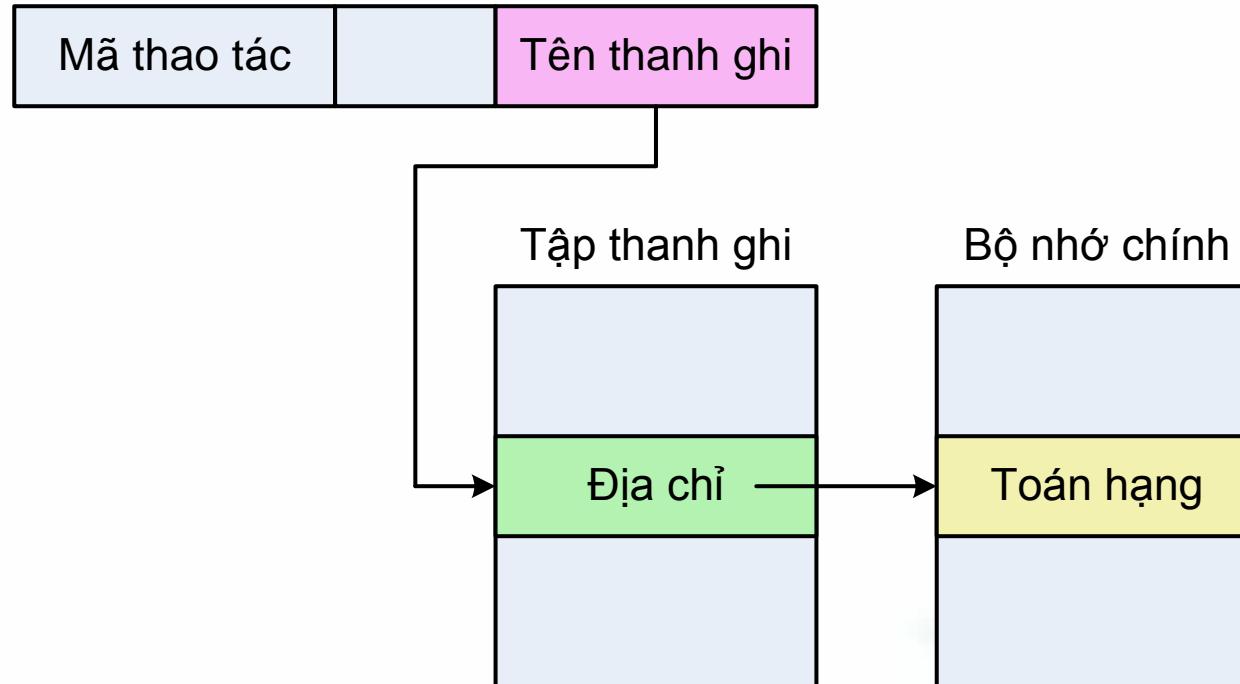
MOV AL, [1000]

; AL ← nội dung byte nhớ có
địa chỉ là 1000





Chế độ địa chỉ gián tiếp qua thanh ghi



- Register Indirect Addressing Mode
- Ví dụ: **MOV AL, [BX]** ; $AL \leftarrow$ nội dung của byte nhớ có địa chỉ bằng giá trị của thanh ghi BX

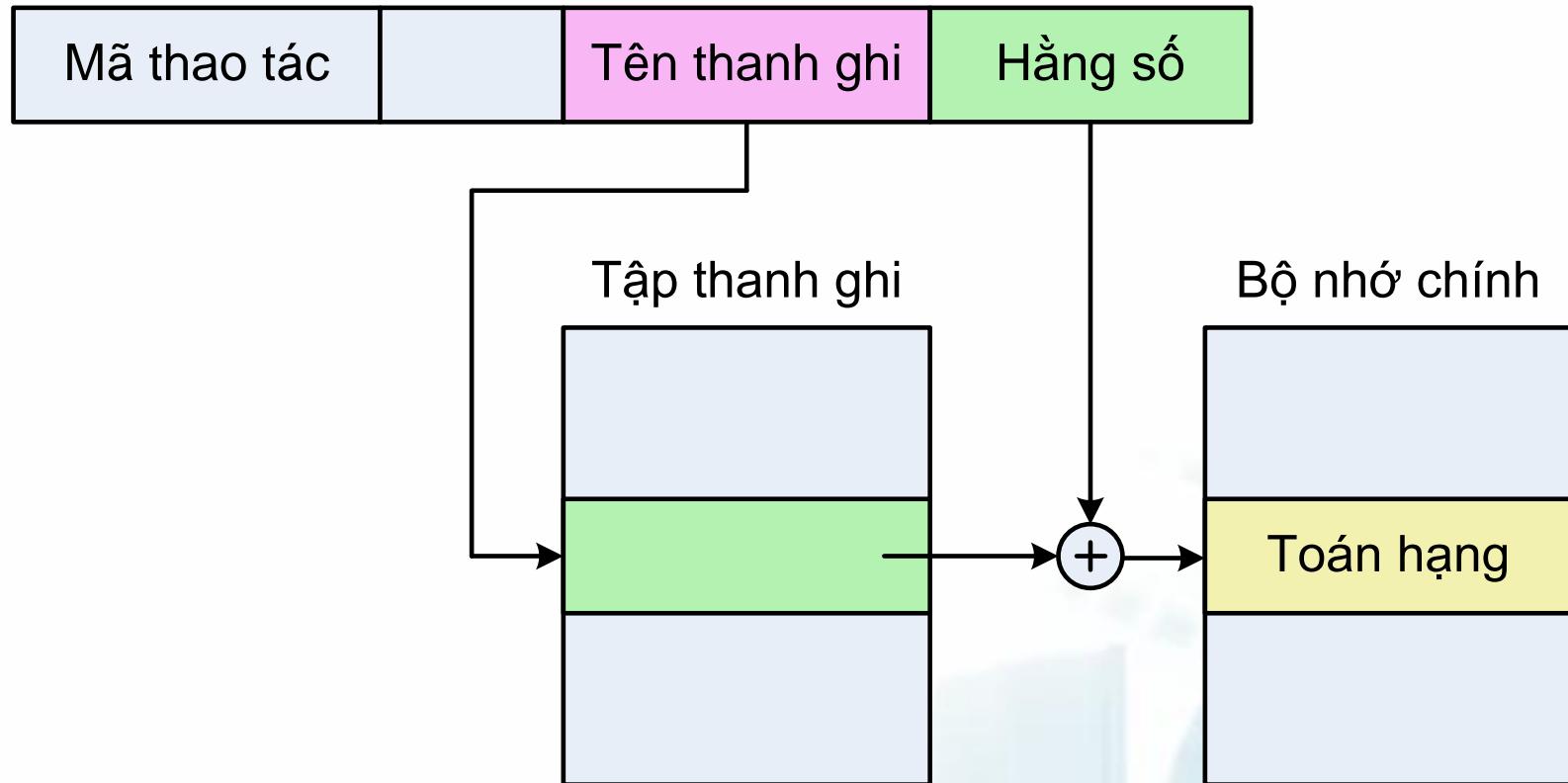


Chế độ địa chỉ dịch chuyển

- Displacement Addressing Mode
- Trường địa chỉ chứa 2 thành phần:
 - Tên thanh ghi
 - Hằng số
- Địa chỉ của toán hạng = nội dung thanh ghi + hằng số
- Thanh ghi có thể được ngầm định



Minh họa chế độ địa chỉ dịch chuyển





Các dạng chế độ địa chỉ dịch chuyển

- Địa chỉ hóa tương đối với PC:
 - Thanh ghi là PC
 - VD: các lệnh chuyển điều khiển
- Định địa chỉ cơ sở:
 - Thanh ghi là thanh ghi cơ sở (chứa địa chỉ cơ sở)
 - Hằng số là chỉ số
- Định địa chỉ chỉ số:
 - Thanh ghi là thanh ghi chỉ số (chứa chỉ số)
 - Hằng số là địa chỉ cơ sở



3.2. Bộ xử lý trung tâm

3.2.1. Cấu trúc cơ bản của CPU

3.2.2. Tập lệnh

3.3.3. Hoạt động của CPU



3.2.3. Hoạt động của CPU

1. Chu trình lệnh
2. Đường ống lệnh



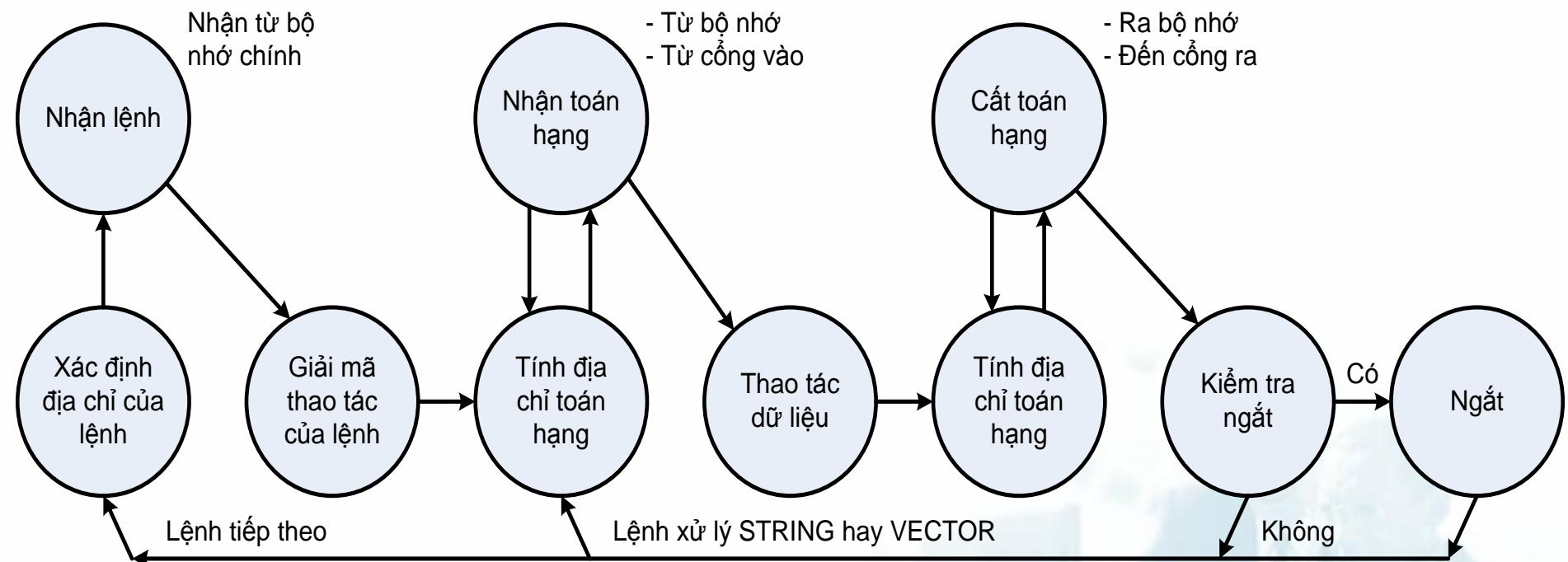


1. Chu trình lệnh

- Bao gồm các công đoạn chính sau đây:
 - Nhận lệnh
 - Giải mã lệnh
 - Nhận toán hạng
 - Thực hiện lệnh
 - Cắt toán hạng
 - Ngắt



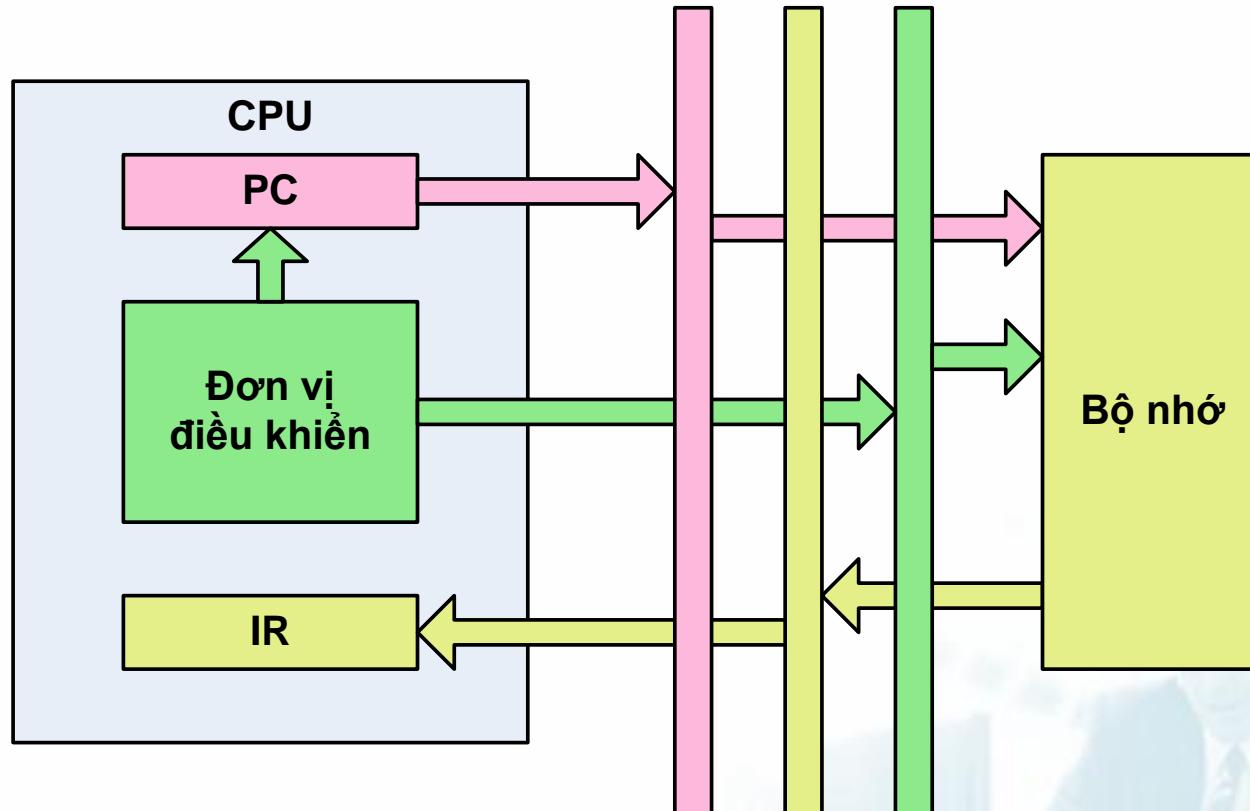
Giản đồ trạng thái chu trình lệnh



- CPU đưa địa chỉ của lệnh cần nhận từ thanh ghi bộ đếm chương trình PC ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc bộ nhớ
- Lệnh từ bộ nhớ được đặt lên bus dữ liệu và được CPU copy vào trong thanh ghi lệnh IR
- CPU tăng nội dung của PC để trả sang lệnh kế tiếp



Minh họa quá trình nhận lệnh



PC : Bộ đếm chương trình
IR : Thanh ghi lệnh

Bus địa chỉ
Bus dữ liệu
Bus điều khiển

- Lệnh từ thanh ghi lệnh IR được đưa đến đơn vị điều khiển
- Đơn vị điều khiển tiến hành giải mã lệnh để xác định thao tác cần phải thực hiện



Nhận toán hạng

- CPU đưa địa chỉ của toán hạng ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc
- Toán hạng được chuyển vào trong CPU

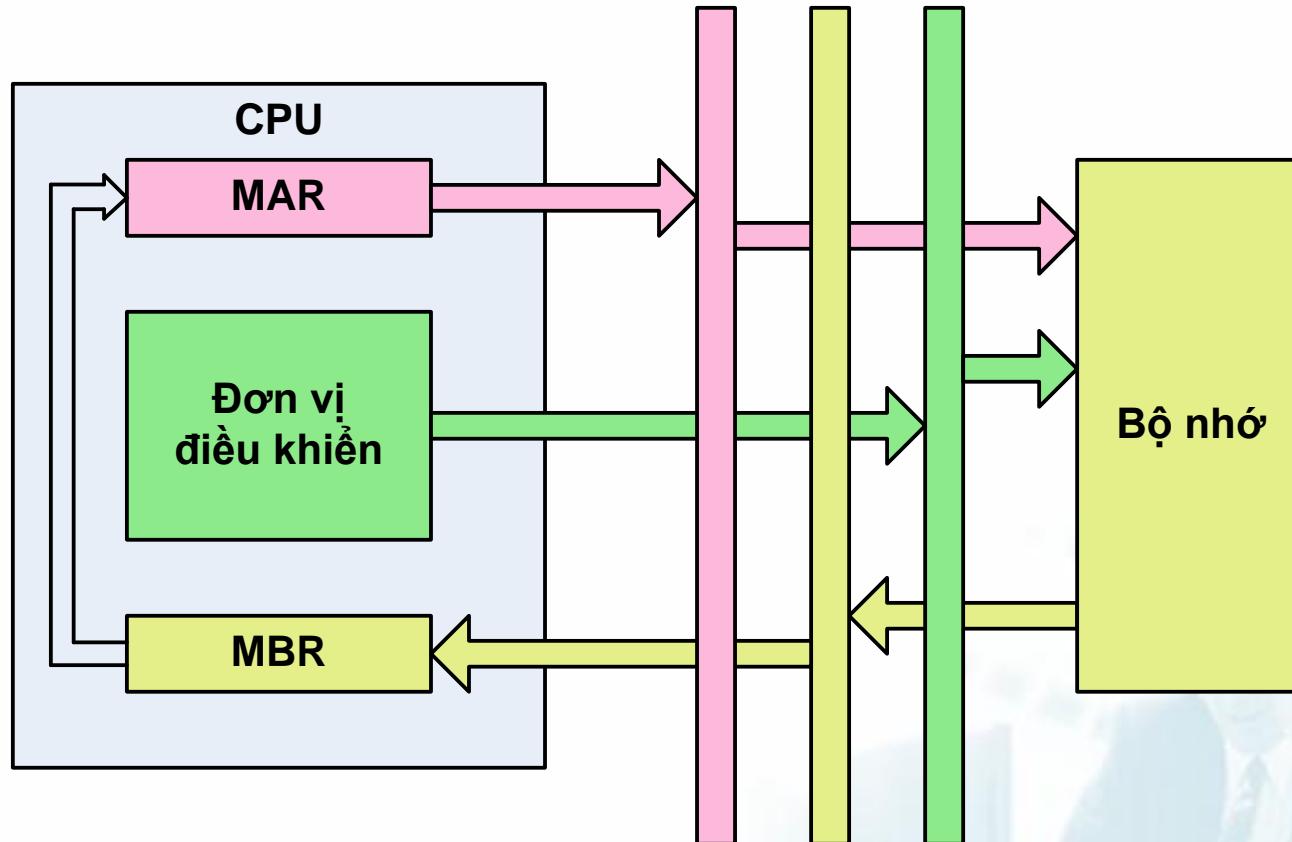


Nhận toán hàng gián tiếp

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



Minh họa nhận toán hạng gián tiếp



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

MBR (Memory Buffer Register):
Thanh ghi đệm bộ nhớ

Bus
địa
chỉ

Bus
dữ
liệu

Bus
điều
khiển

- Có nhiều dạng thao tác tùy thuộc vào lệnh
- Có thể là:
 - Đọc/ghi bộ nhớ
 - Vào-ra dữ liệu
 - Chuyển dữ liệu giữa các thanh ghi
 - Thực hiện phép toán số học hoặc logic
 - Chuyển điều khiển (rẽ nhánh)
 - ...

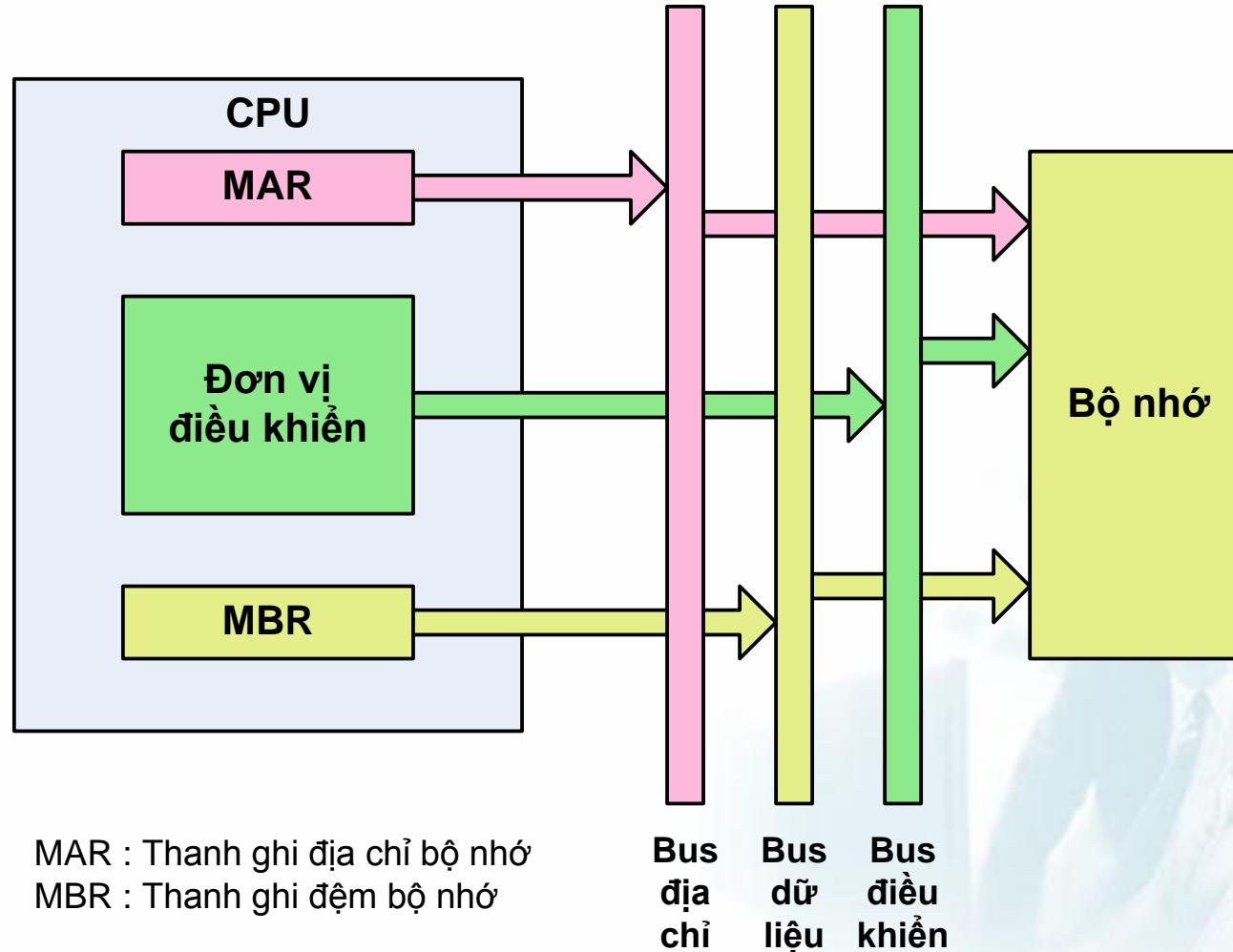


Ghi toán hạng

- CPU đưa địa chỉ ra bus địa chỉ
- CPU đưa dữ liệu cần ghi ra bus dữ liệu
- CPU phát tín hiệu điều khiển ghi
- Dữ liệu trên bus dữ liệu được copy đến vị trí xác định



Minh họa quá trình ghi toán hạng



- CPU lưu lại giá trị hiện tại của PC (là địa chỉ trả về sau khi hoàn thành ngắt) – thường lưu vào Stack:
 - CPU đưa nội dung của PC ra bus dữ liệu
 - CPU đưa địa chỉ (thường được xác định từ con trỏ ngăn xếp SP) ra bus địa chỉ
 - CPU phát tín hiệu điều khiển ghi bộ nhớ
 - Địa chỉ trả về (nội dung của PC) trên bus dữ liệu được lưu vào ngăn nhớ tương ứng ở ngăn xếp
- CPU nạp vào PC địa chỉ lệnh đầu tiên của chương trình con phục vụ ngắt tương ứng:
 - CPU xác định địa chỉ của vector ngắt tương ứng
 - CPU phát địa chỉ này ra bus địa chỉ
 - CPU phát tín hiệu điều khiển đọc bộ nhớ
 - Giá trị của vector ngắt (địa chỉ lệnh đầu tiên của CTC phục vụ ngắt) được chuyển ra bus dữ liệu
 - Giá trị này được nạp vào trong PC



2. Đường ống lệnh

- Nguyên tắc của Pipeline: chia chu trình lệnh thành các công đoạn và cho phép thực hiện gối lén nhau theo kiểu dây chuyền.
- Giả sử chu trình lệnh gồm 6 công đoạn với thời gian thực hiện nhau nhau (T):
 - Nhận lệnh (Fetch Instruction – FI)
 - Giải mã lệnh (Decode Instruction – DI)
 - Tính đ/chỉ toán hạng (Calculate Operand Address – CO)
 - Nhận toán hạng (Fetch Operands – FO)
 - Thực hiện lệnh (Execute Instruction – EI)
 - Ghi toán hạng (Write Operands – WO)



Biểu đồ thời gian của đường ống lệnh

	1	2	3	4	5	6	7	8	9	10	11	12
Lệnh 1	FI	DI	CO	FO	EI	WO						
Lệnh 2		FI	DI	CO	FO	EI	WO					
Lệnh 3			FI	DI	CO	FO	EI	WO				
Lệnh 4				FI	DI	CO	FO	EI	WO			
Lệnh 5					FI	DI	CO	FO	EI	WO		
Lệnh 6						FI	DI	CO	FO	EI	WO	



Các xung đột của đường ống lệnh

- Xung đột cấu trúc: do nhiều công đoạn dùng chung một tài nguyên
- Xung đột dữ liệu: lệnh sau sử dụng kết quả của lệnh trước
- Xung đột điều khiển: do rẽ nhánh gây ra





3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.1. Tổng quan hệ thống nhớ

1. Các đặc trưng của hệ thống nhớ
2. Phân cấp hệ thống nhớ của máy tính



1. Các đặc trưng của hệ thống nhớ

- Vị trí:
 - Bên trong CPU: tập thanh ghi
 - Bộ nhớ trong: bộ nhớ chính và cache
 - Bộ nhớ ngoài: các thiết bị nhớ
- Dung lượng:
 - Độ dài từ nhớ (tính bằng bit)
 - Số lượng từ nhớ
- Đơn vị truyền:
 - Theo từng từ nhớ
 - Theo từng khối (block) nhớ
- Phương pháp truy cập:
 - Truy cập tuần tự (băng từ)
 - Truy cập trực tiếp (các loại đĩa)
 - Truy cập ngẫu nhiên (bộ nhớ bán dẫn)
 - Truy cập liên kết (cache)

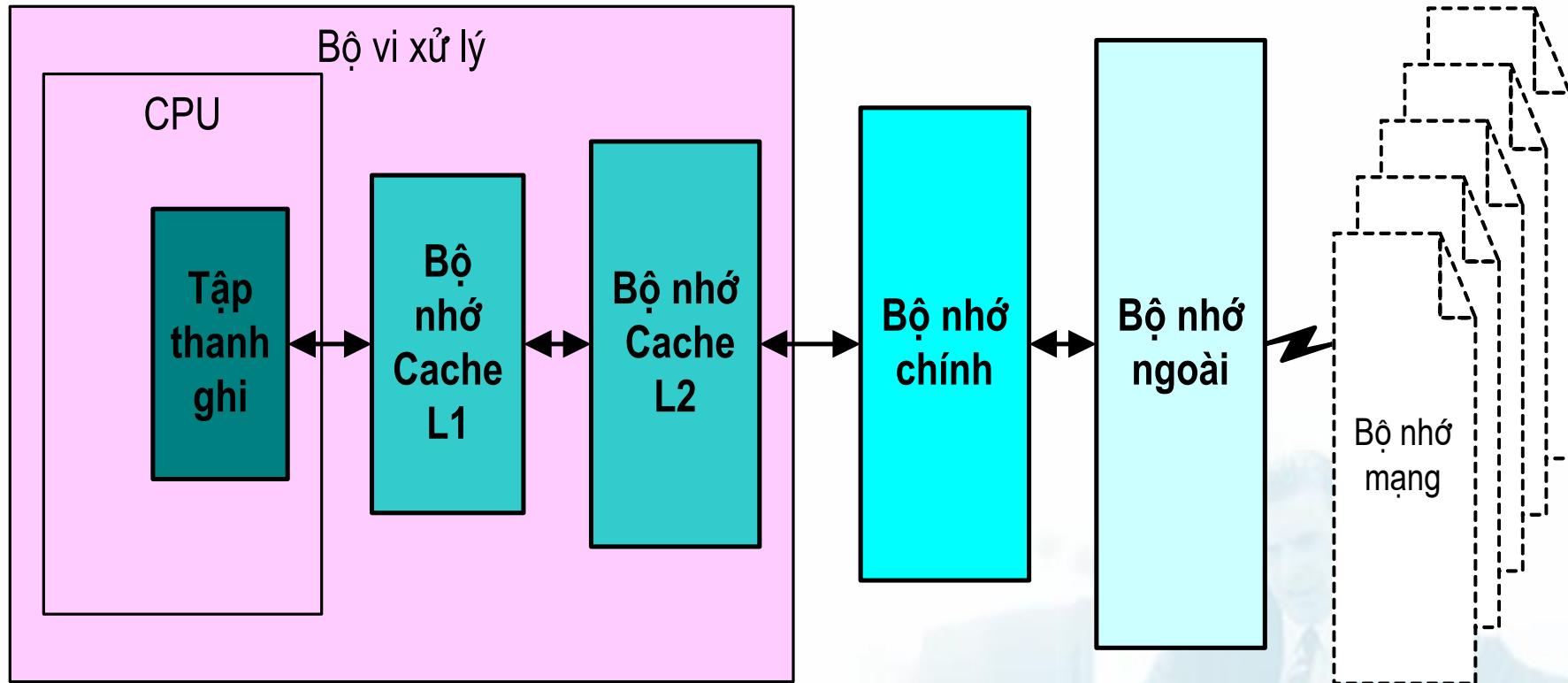


Các đặc trưng của hệ thống nhớ

- **Hiệu năng:**
 - Thời gian truy cập
 - Chu kỳ nhớ
 - Tốc độ truyền
- **Kiểu vật lý:**
 - Bộ nhớ bán dẫn
 - Bộ nhớ từ
 - Bộ nhớ quang
- **Các đặc tính vật lý:**
 - Khả biến (mất điện thì mất thông tin) / Không khả biến
 - Xóa được / Không xóa được
- **Tổ chức**



2. Phân cấp hệ thống nhớ của MT



Dung lượng ↑, tốc độ ↓, tần suất CPU truy cập ↓, giá thành / bit thông tin ↓, ...



Hệ thống nhớ của máy tính (tiếp)

- **Tập thanh ghi (Registers):**
 - Là thành phần nhớ nằm trong CPU, được coi là mức nhớ đầu tiên
 - Chứa các thông tin phục vụ cho hoạt động ở thời điểm hiện tại của CPU
- **Bộ nhớ đệm nhanh (Cache):**
 - Bộ nhớ có tốc độ nhanh được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ truy cập bộ nhớ của CPU.
 - Thường được chia thành một vài mức (L1, L2)
- **Bộ nhớ chính (Main Memory):**
 - Chứa các chương trình và dữ liệu đang được sử dụng.
- **Bộ nhớ ngoài (External Memory):**
 - Chứa các tài nguyên phần mềm của máy tính.



3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.2. Bộ nhớ bán dẫn

1. Phân loại
2. Mô hình cơ bản của chip nhớ



1. Phân loại

- Gồm 2 loại chính: ROM và RAM
- ROM (Read Only Memory): bộ nhớ chỉ đọc
- Đặc điểm:
 - Bộ nhớ chủ yếu dùng để đọc thông tin
 - Bộ nhớ không khả biến
 - Chứa các chương trình và dữ liệu cố định với hệ thống

■ Các loại bộ nhớ ROM:

- Maskable ROM (ROM mặt nạ): thông tin được ghi khi chế tạo
- PROM (Programmable ROM):
 - Khi chế tạo chưa có thông tin
 - Cho phép ghi thông tin được 1 lần bằng thiết bị chuyên dụng
- EPROM (Erasable PROM):
 - Cho phép xóa bằng tia cực tím
 - Ghi lại bằng thiết bị nạp EEPROM
- EEPROM (Electrically Erasable PROM):
 - Có thể xóa bằng tín hiệu điện và ghi lại thông tin ngay trong mạch làm việc (không cần thiết bị ghi riêng)
 - Có thể xóa và ghi lại ở mức từng Byte
 - Dung lượng nhỏ
- Flash Memory: giống EEPROM nhưng:
 - Đọc/ghi theo từng block
 - Tốc độ rất nhanh
 - Dung lượng lớn



RAM (Random Access Memory)

- RAM (Random Access Memory): bộ nhớ truy cập ngẫu nhiên
- Đặc điểm:
 - Là bộ nhớ đọc/ghi (Read/Write Memory – RWM)
 - Bộ nhớ khả biến
 - Chứa các thông tin tạm thời



■ Các loại bộ nhớ RAM:

- SRAM (Static): RAM tĩnh

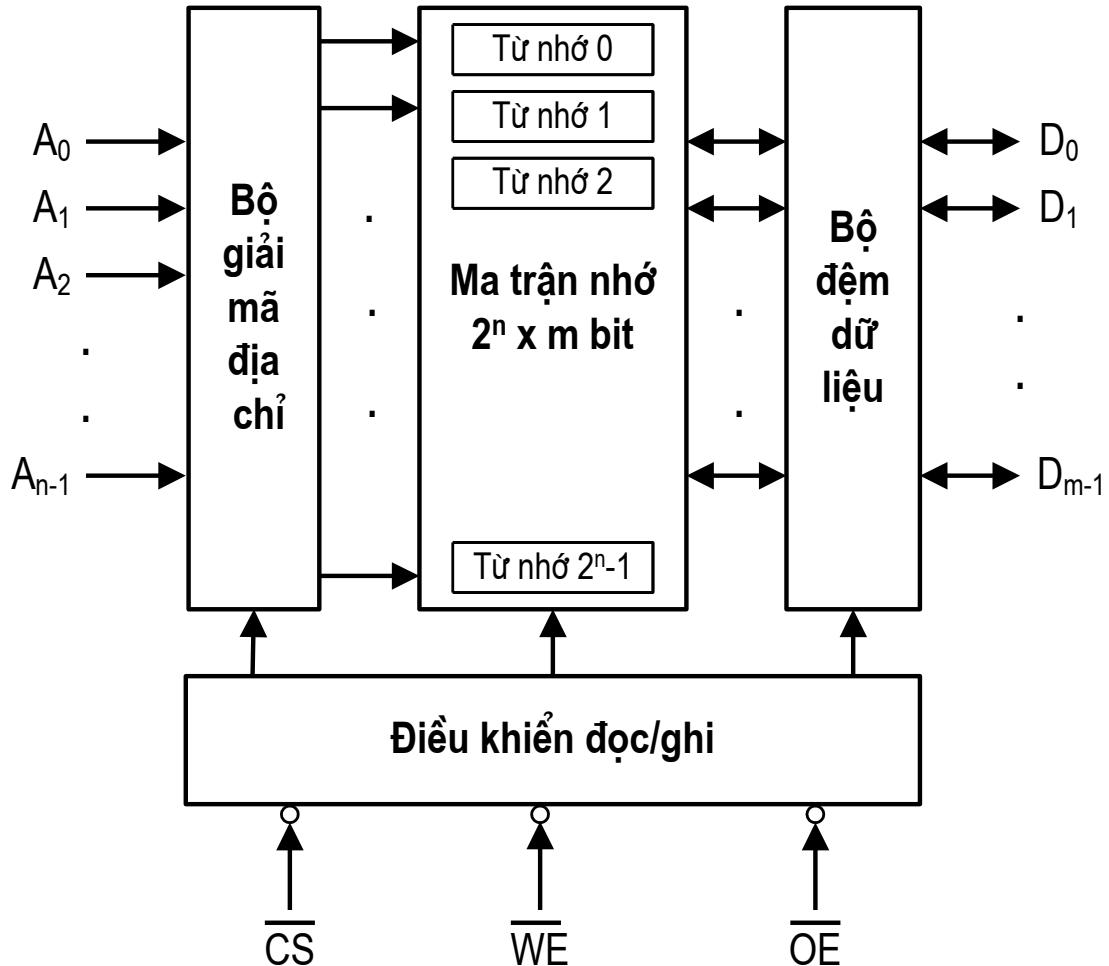
- Mỗi phần tử nhớ là một mạch lật 2 trạng thái ổn định → thông tin trên SRAM ổn định
- Tốc độ nhanh
- Dung lượng chip nhớ nhỏ
- Giá thành đắt
- Thường dùng làm bộ nhớ Cache

- DRAM (Dynamic): RAM động

- Mỗi phần tử nhớ là một tụ điện rất nhỏ → cứ sau một khoảng thời gian thì điện tích trên tụ điện sẽ bị mất, cho nên thông tin trên DRAM không ổn định → khắc phục bằng mạch làm tươi (refresh) DRAM
- Tốc độ chậm (do mất thời gian làm tươi DRAM)
- Dung lượng chip nhớ lớn
- Giá thành rẻ
- Thường dùng làm bộ nhớ chính



2. Mô hình cơ bản của chip nhớ





Mô hình cơ bản của chip nhớ (tiếp)

- Có n chân địa chỉ ($A_{n-1} \div A_0$) : vận chuyển vào chip nhớ được n bit địa chỉ đồng thời → trong chip nhớ có 2^n từ nhớ.
- Có m chân dữ liệu: ($D_{m-1} \div D_0$) : cho phép vận chuyển đồng thời được m bit dữ liệu → độ dài từ nhớ là m bit.
→ Dung lượng của chip nhớ là: $2^n \times m$ bit
- Các chân tín hiệu điều khiển:
 - CS (Chip Select): tín hiệu điều khiển chọn chip nhớ làm việc
 - OE (Output Enable): tín hiệu điều khiển đọc dữ liệu của 1 từ nhớ đã được xác định.
 - WE (Write Enable): tín hiệu điều khiển ghi dữ liệu vào 1 từ nhớ đã được xác định.



Hoạt động của chip nhớ

■ Hoạt động đọc:

- Các bit địa chỉ được đưa đến các chân địa chỉ.
- Tín hiệu điều khiển chọn chip nhớ làm việc được đưa đến \overline{CS}
- Tín hiệu điều khiển đọc đưa đến \overline{OE}
- Dữ liệu từ ngăn nhớ tương ứng với địa chỉ đã có sẽ được đưa ra các chân dữ liệu.



Hoạt động của chip nhớ (tiếp)

■ Hoạt động ghi:

- Các bit địa chỉ được đưa đến các chân địa chỉ
- Dữ liệu cần ghi được đưa đến các chân dữ liệu
- Tín hiệu điều khiển chọn chip được đưa đến \overline{CS}
- Tín hiệu điều khiển ghi được đưa đến \overline{WE}
- Dữ liệu từ các chân dữ liệu sẽ được ghi vào ngăn nhớ tương ứng.



3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.3. Bộ nhớ chính

1. Các đặc trưng của bộ nhớ chính
2. Tổ chức bộ nhớ đan xen



1. Các đặc trưng của bộ nhớ chính

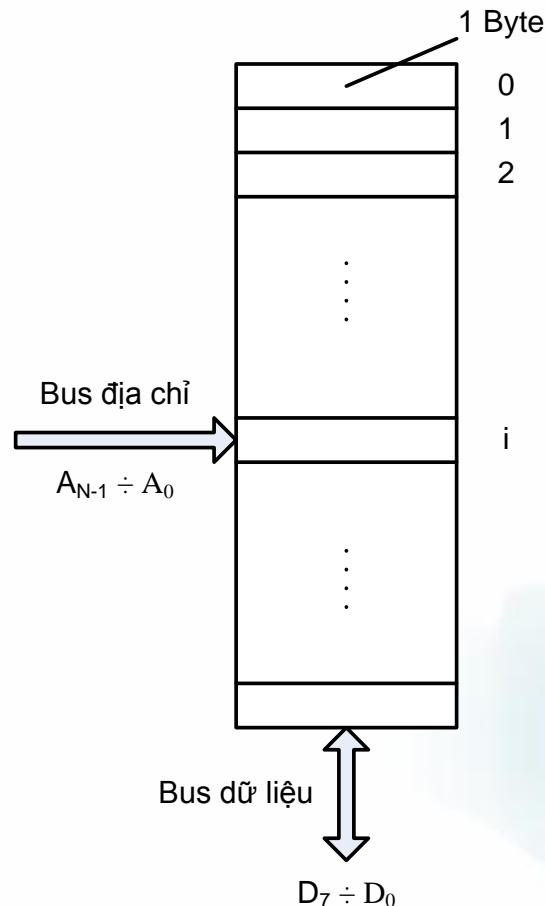
- Là thành phần nhớ tồn tại trên mọi hệ thống máy tính
- Chứa các chương trình đang được thực hiện và các dữ liệu đang được sử dụng
- Bao gồm các ngăn nhớ được đánh địa chỉ trực tiếp bởi CPU
- Dung lượng vật lý của bộ nhớ chính \leq không gian địa chỉ bộ nhớ mà CPU quản lý
- Việc quản lý logic bộ nhớ chính tùy thuộc vào hệ điều hành



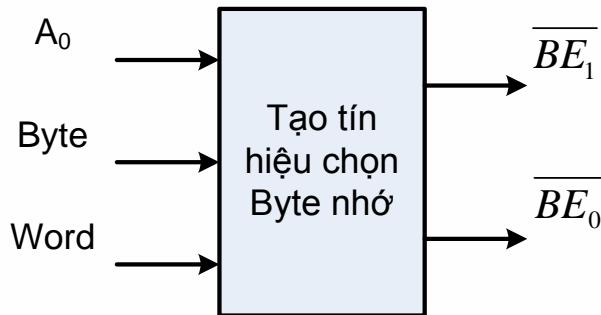
2. Tổ chức bộ nhớ đan xen

- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ chính $M = 8, 16, 32, 64, 128 \dots$ bit
- Các ngăn nhớ được tổ chức theo từng Byte nhớ
→ Tổ chức bộ nhớ chính khác nhau

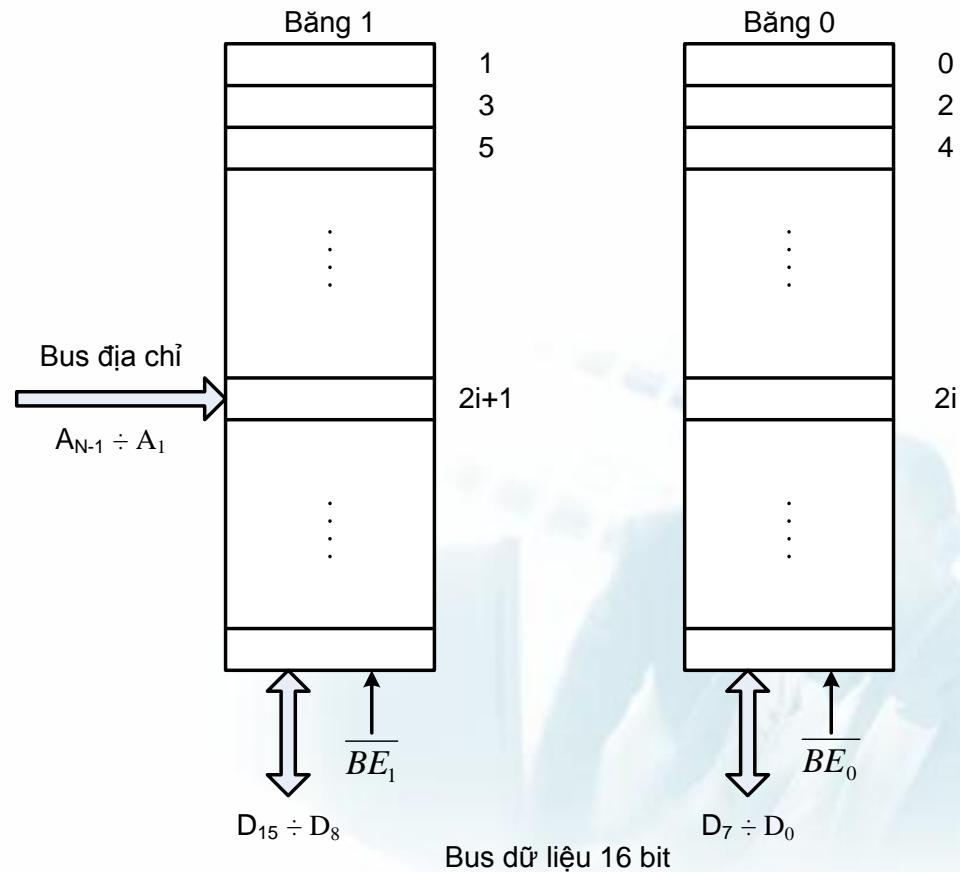
- VD: Intel 8088
- BN chính là 1 băng (bank) nhớ tuyến tính



- VD: Intel 8086 ÷ 80286
- Bộ nhớ chính gồm 2 băng (bank) nhớ đan xen



\overline{BE}_1	\overline{BE}_0	Ý nghĩa
0	0	Chọn cả 2 Byte
0	1	Chọn Byte cao
1	0	Chọn Byte thấp
1	1	Không chọn





Các trường hợp khác

- Với $M = 32$ bit (80386, 80486): bộ nhớ chính gồm 4 băng nhớ đan xen
- Với $M = 64$ bit (các bộ xử lý Pentium): bộ nhớ chính gồm 8 băng nhớ đan xen



3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.4. Bộ nhớ cache

1. Nguyên tắc chung của cache
 2. Các phương pháp ánh xạ
 3. Thuật giải thay thế
 4. Phương pháp ghi dữ liệu khi cache hit
 5. Cache trên các bộ xử lý Intel
- 



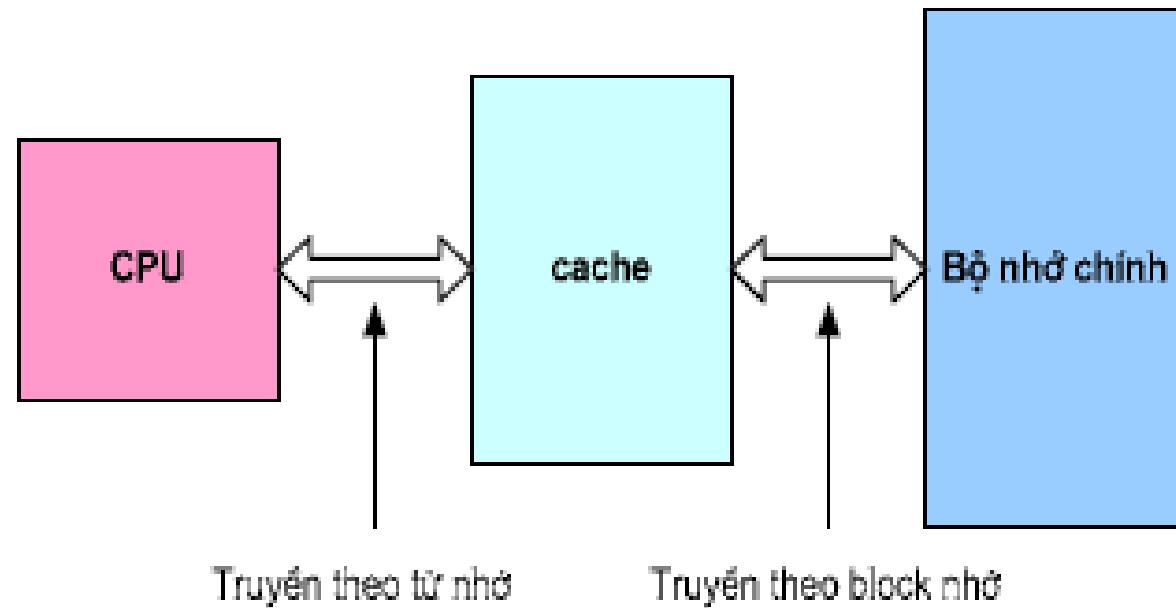
1. Nguyên tắc chung của cache

- Nguyên lý cục bộ hoá tham chiếu bộ nhớ: Trong một khoảng thời gian đủ nhỏ CPU thường chỉ tham chiếu các thông tin trong một khối nhớ cục bộ
- Ví dụ:
 - Cấu trúc chương trình tuần tự
 - Vòng lặp có thân nhỏ
 - Cấu trúc dữ liệu mảng



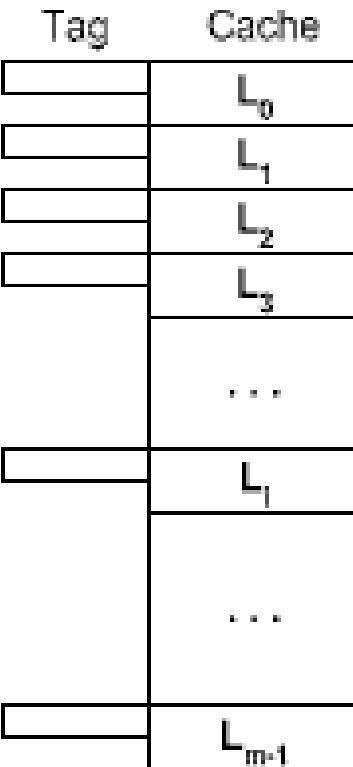
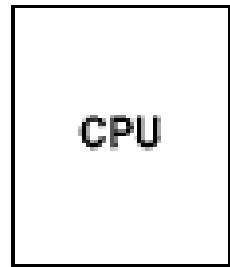
1. Nguyên tắc chung của cache (tiếp)

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

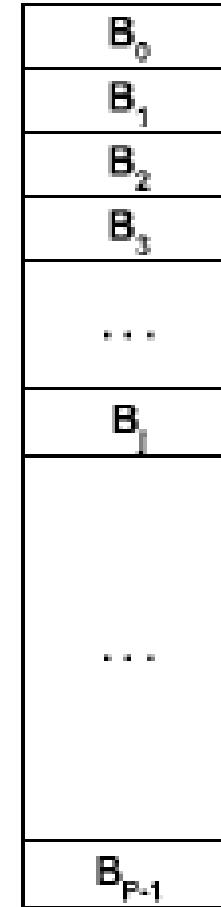




Cấu trúc chung của cache / Bộ nhớ chính



Bộ nhớ chính





Cấu trúc chung của cache / Bộ nhớ chính

- Bộ nhớ chính có 2^N byte nhớ
- Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
 - Bộ nhớ chính: B0, B1, B2, ... , B_{p-1} (p Blocks)
 - Bộ nhớ cache: L0, L1, L2, ... , L_{m-1} (m Lines)
 - Kích thước của Block = 8,16,32,64,128 byte



Cấu trúc chung của cache / Bộ nhớ chính

- Một số Block của bộ nhớ chính được nạp vào các Line của cache.
- Nội dung Tag (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó.
- Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:
 - Từ nhớ đó có trong cache (cache hit)
 - Từ nhớ đó không có trong cache (cache miss).

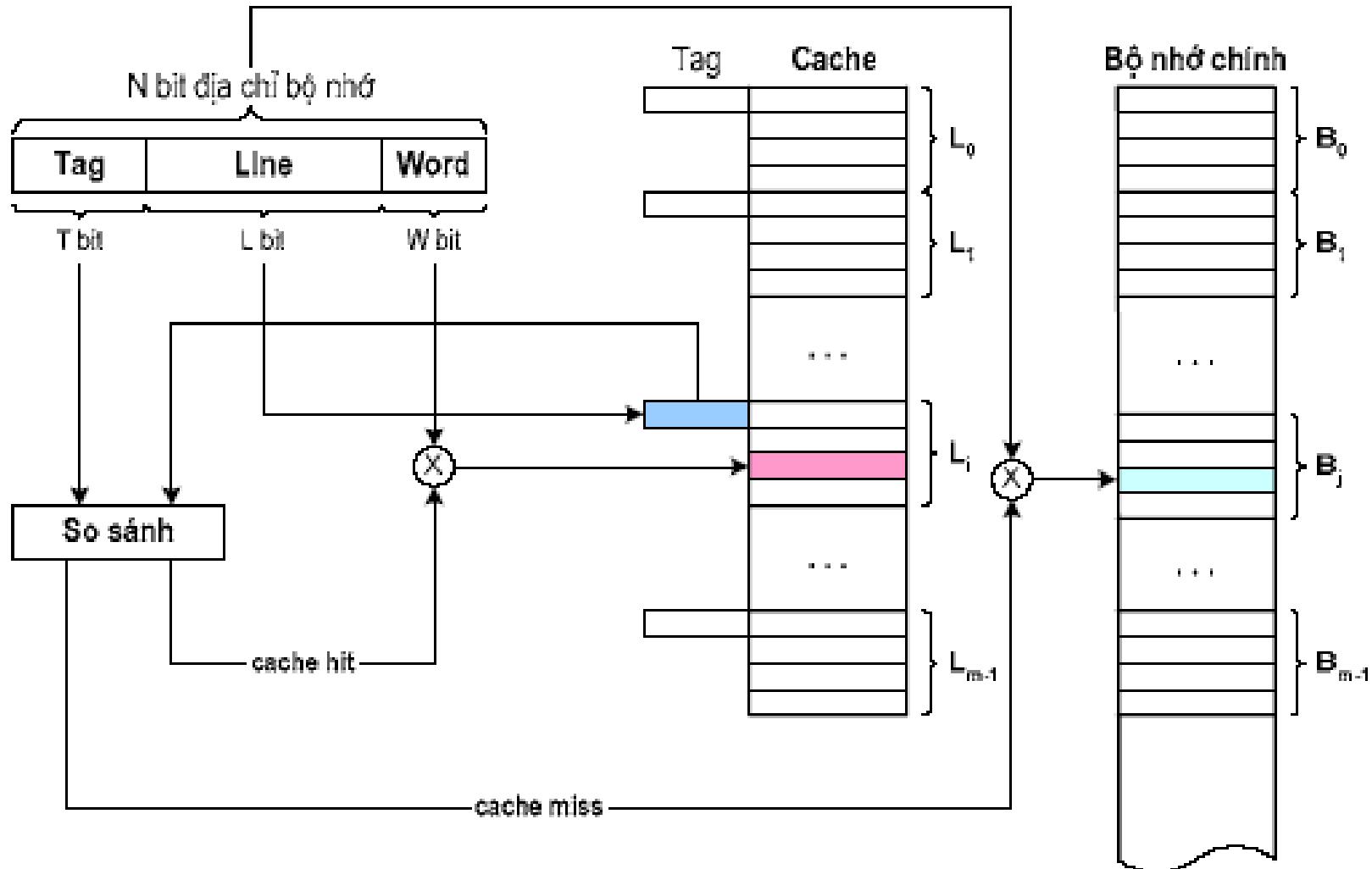


2. Các phương pháp ánh xạ

- Ánh xạ trực tiếp
(Direct mapping)
- Ánh xạ liên kết toàn phần
(Fully associative mapping)
- Ánh xạ liên kết tập hợp
(Set associative mapping)

- Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:
 - $B_0 \square L_0$
 - $B_1 \square L_1$
 -
 - $B_{m-1} \square L_{m-1}$
 - $B_m \square L_0$
 - $B_{m+1} \square L_1$
 -
- Tổng quát
 - B_j chỉ có thể nạp vào $L_{j \bmod m}$
 - m là số Line của cache.

Minh họa ánh xạ trực tiếp





Đặc điểm của ánh xạ trực tiếp

- Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:
 - Trường Word gồm W bit xác định một từ nhớ trong *Block* hay *Line*:
$$2^W = \text{kích thước của } Block \text{ hay } Line$$
 - Trường *Line* gồm L bit xác định một trong số các *Line* trong *cache*:
$$2^L = \text{số Line trong cache} = m$$
 - Trường *Tag* gồm T bit:
$$T = N - (W+L)$$
- Bộ so sánh đơn giản
- Xác suất *cache hit* thấp



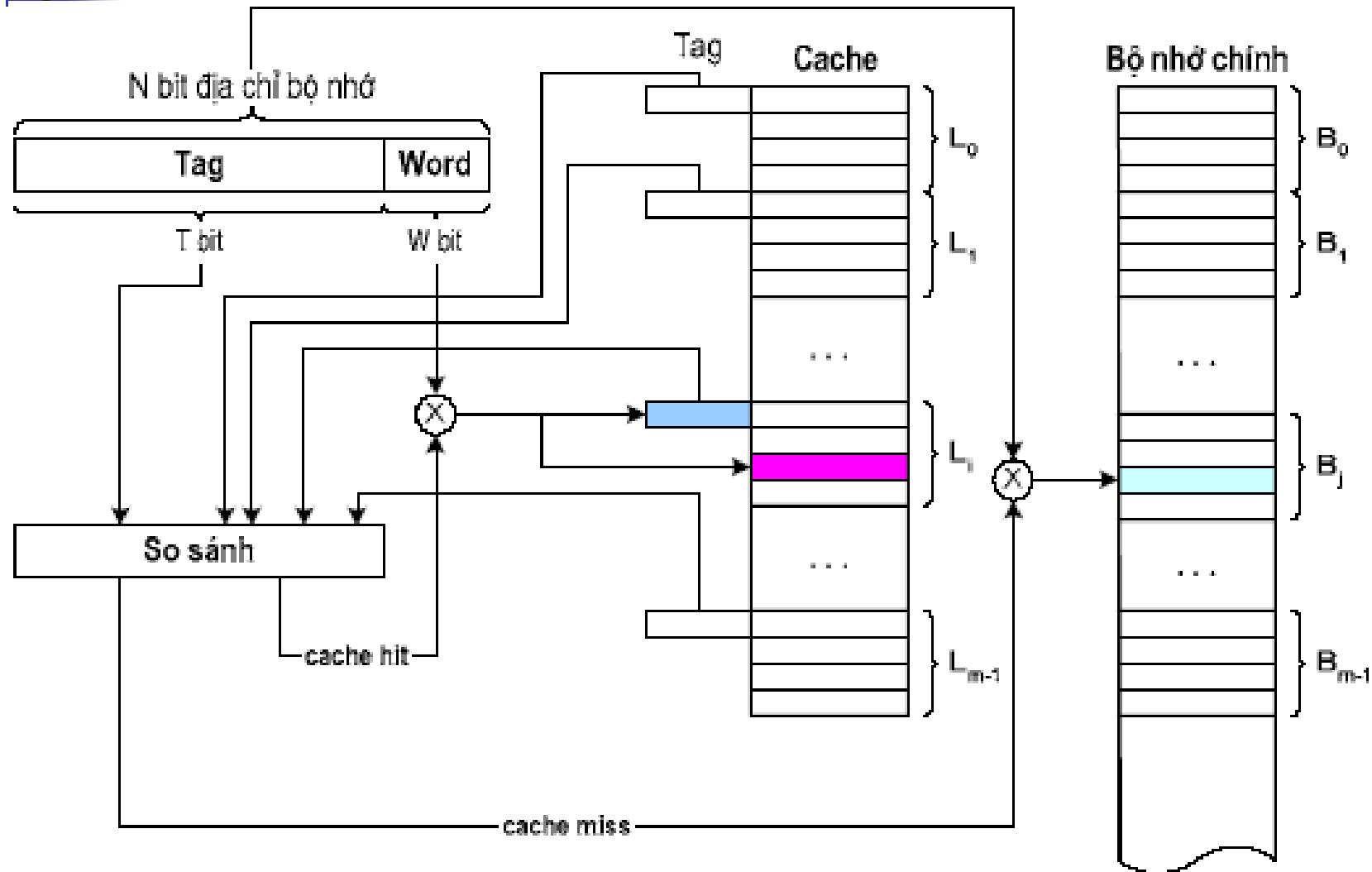


Ánh xạ liên kết toàn phần

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



Minh họa ánh xạ liên kết toàn phần





Đặc điểm của ánh xạ liên kết toàn phần

- So sánh đồng thời với tất cả các Tag mất nhiều thời gian
- Xác suất *cache hit* cao.
- Bộ so sánh phức tạp.





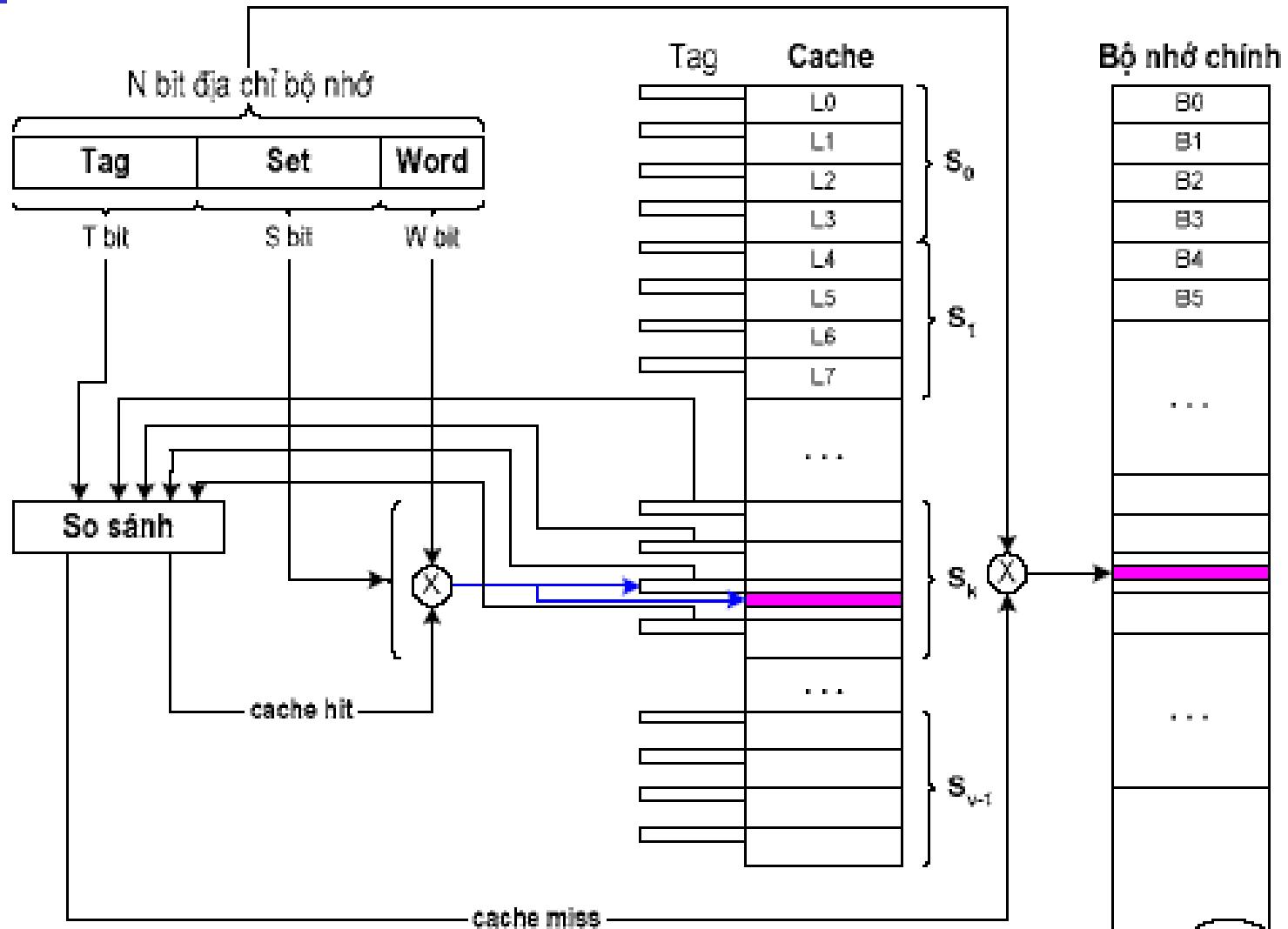
Ánh xạ liên kết tập hợp

- Cache được chia thành các Tập (Set)
- Mỗi một Set chứa một số Line
- Ví dụ:
 - 4 Line/Set 4-way associative mapping
 - Ánh xạ theo nguyên tắc sau:
 - B0 S0
 - B1 S1
 - B2 S2
 -





Minh họa ánh xạ liên kết tập hợp





Đặc điểm của ánh xạ liên kết tập hợp

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



Thuật giải thay thế (Ánh xạ trực tiếp)

- Không phải lựa chọn
- Mỗi Block chỉ ánh xạ vào một Line xác định
- Thay thế Block ở Line đó





Thuật giải thay thế (Ánh xạ liên kết)

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

- Ghi xuyên qua (Write-through):
 - ghi cả cache và cả bộ nhớ chính
 - tốc độ chậm
- Ghi trả sau (Write-back):
 - chỉ ghi ra cache
 - tốc độ nhanh
 - khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính



5. Cache trên các bộ xử lý intel

- 80486: 8KB cache L1 trên chip
- Pentium: có hai cache L1 trên chip
 - Cache lệnh = 8KB
 - Cache dữ liệu = 8KB
- Pentium 4: hai mức cache L1 và L2 trên chip
 - Cache L1:
 - mỗi cache 8KB
 - Kích thước Line = 64 byte
 - ánh xạ liên kết tập hợp 4 đường
 - cache L2
 - 256KB
 - Kích thước Line = 128 byte
 - ánh xạ liên kết tập hợp 8 đường





3.3.5. Bộ nhớ ngoài

1. Đĩa từ
2. Đĩa quang
3. Flash disk
4. Các chuẩn nối ghép ổ đĩa
5. RAID

- Các đặc tính của đĩa từ:

- Đầu từ cố định hay di động
- Đĩa cố định hay thay đổi
- Một mặt hay hai mặt
- Một đĩa hay nhiều đĩa
- Cơ chế đầu từ:
 - Tiếp xúc
 - Không tiếp xúc

- Gồm 2 loại phổ biến:

- Đĩa mềm
- Đĩa cứng

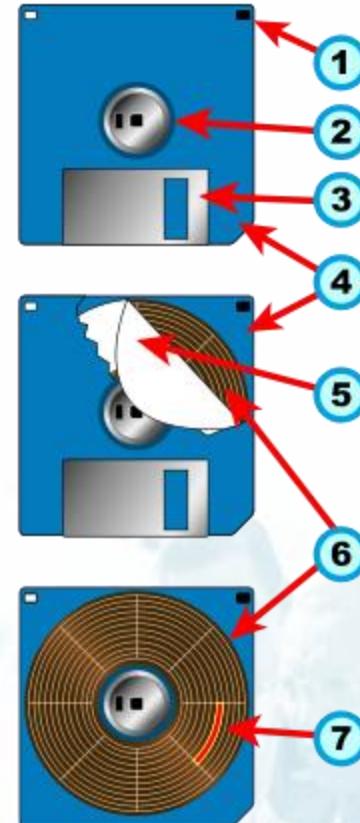




a. Đĩa mềm

- 8", 5.25", 3.5"
- Dung lượng nhỏ ($\leq 1.44\text{MB}$)
- Tốc độ chậm
- Thông dụng
- Rẻ tiền
- Tương lai có thể không dùng nữa

Đĩa mềm (tiếp)





b. Đĩa cứng

- Một hoặc nhiều đĩa
- Thông dụng
- Dung lượng tăng nhanh
- Tốc độ đọc/ghi nhanh
- Tương đối rẻ tiền



Đĩa cứng (tiếp)





2. Đĩa quang

- Các loại chính:

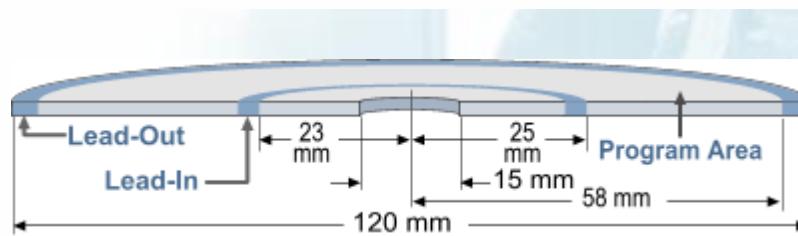
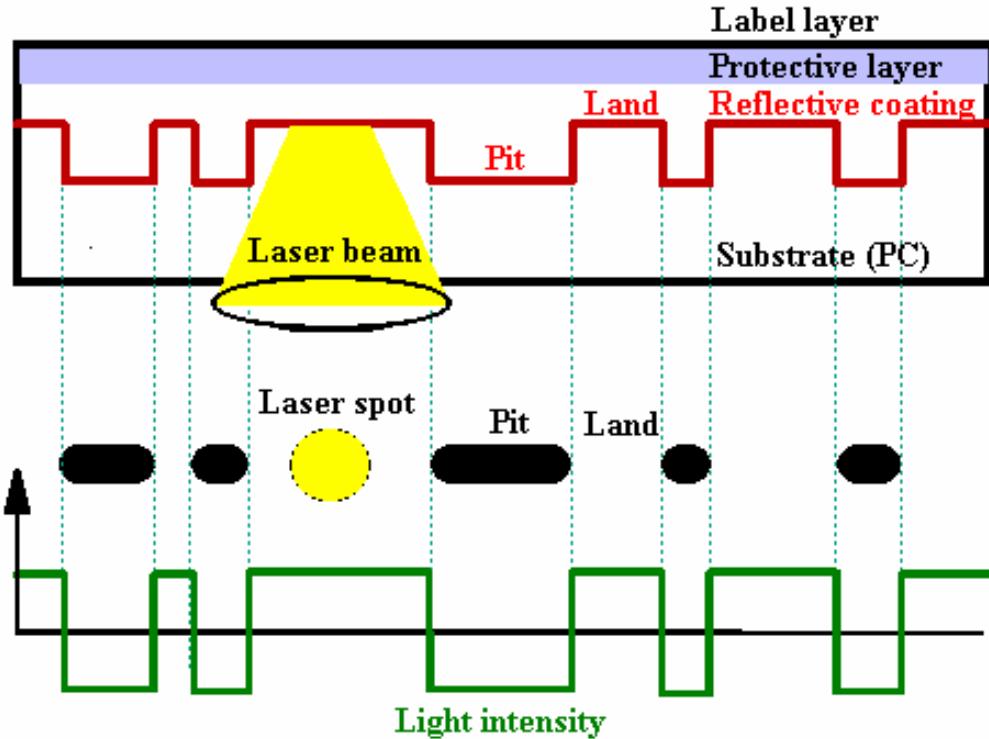
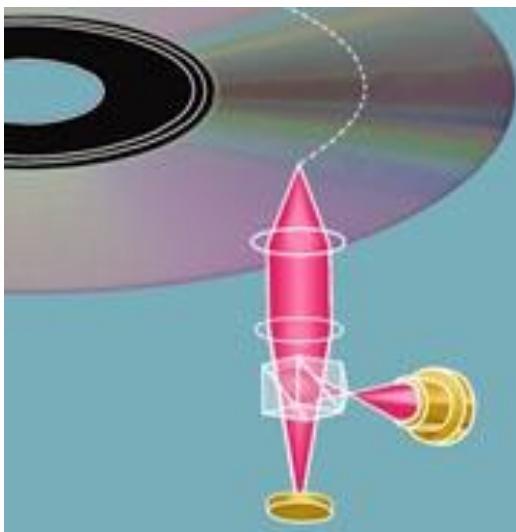
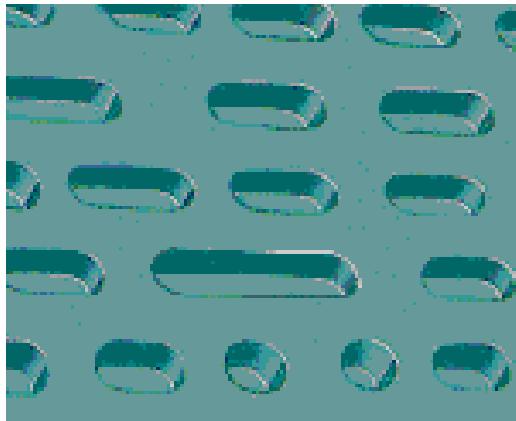
- CD-ROM (Compact Disk Read Only Memory)
- CD-R (Recordable CD)
- CD-RW (Rewriteable CD)
- DVD (Digital Video Disk)



a. CD-ROM

- Thông tin được ghi ngay khi sản xuất đĩa.
- Dữ liệu tồn tại dưới dạng các mặt phẳng (land) và các lỗ (pit).
 - Bit 1 tương ứng với sự thay đổi từ mặt phẳng thành lỗ hay ngược lại;
 - còn những lỗ hay mặt phẳng kéo dài (không có sự thay đổi) tương ứng với bit 0.
- Tốc độ đọc cơ sở của một ổ đĩa CD-ROM ban đầu là 150KB/s (tốc độ 1X).
- Các ổ đĩa hiện nay có tốc độ đọc là bội số của tốc độ cơ sở này (ví dụ 48X, 52X,...)

CD-ROM (tiếp)





b. CD-R

- Khi sản xuất ra, các đĩa này đều là đĩa trắng (chưa có thông tin). Sau đó có thể ghi dữ liệu lên đĩa này nhưng chỉ ghi được một lần nhờ ổ ghi CD-R riêng.
- CD-R có cấu trúc và hoạt động tương tự như CD-ROM.
 - Cấu tạo gồm nhiều lớp, trong đó lớp chứa dữ liệu là một lớp màu polymer hữu cơ.
 - Khi bị tia laser đốt cháy, lớp màu này chuyển sang màu đen và đóng vai trò như các lỗ (pit) của CD-ROM.
- Các đĩa CD-R sau khi ghi có thể được đọc từ ổ CD-ROM hoặc từ ổ CD-R. Các đĩa CD-R còn được gọi là WORM (write one read multiple).

- CD-RW có cấu trúc và hoạt động tương tự như CD-R. Trong đó lớp chứa dữ liệu là một lớp kim loại.
- Nguyên tắc ghi dữ liệu dựa trên sự thay đổi trạng thái của lớp kim loại:
 - trạng thái tinh thể (phản xạ ánh sáng - mặt phẳng)
 - và trạng thái vô định hình (không phản xạ ánh sáng - vùng lỗ trong CD-ROM hay màu bị đốt đen trong CD-R).
- Quá trình thay đổi trạng thái này có thể thay đổi bất kì tùy theo công suất laser nên đĩa CD-RW có thể được ghi rồi xóa đi ghi lại nhiều lần.

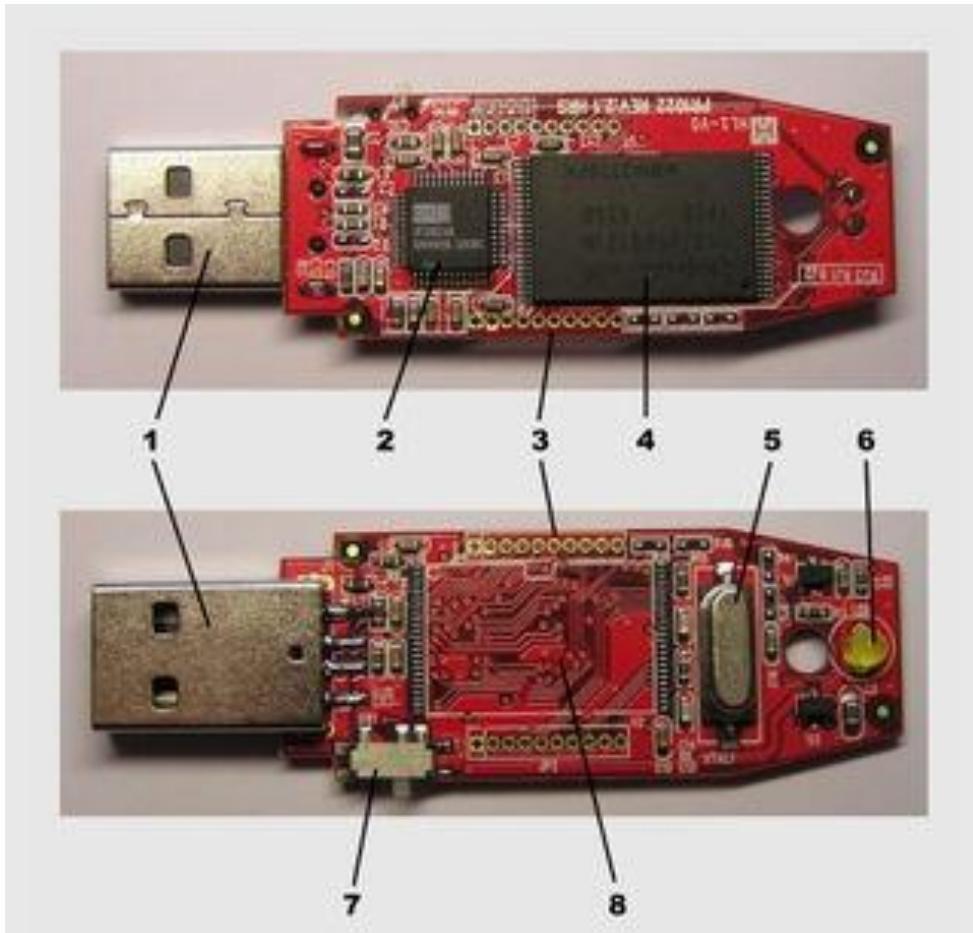
- Đây là loại đĩa quang có dung lượng lớn và có tốc độ nhanh hơn so với các đĩa quang trên.
- Đĩa DVD có thể lưu trữ thông tin trên hai mặt, mỗi mặt có thể có đến 2 lớp dữ liệu.
- Các đĩa DVD hiện nay thường có dung lượng là 4.7GB/mặt hoặc 9.4GB/mặt.
- Tốc độ truy nhập cơ bản của ổ đĩa DVD là 1.321 MByte/s.
- DVD cũng có nhiều loại



3. Flash disk

- Thực chất là bộ nhớ bán dẫn tốc độ cao (flash memory)
- Thường được kết nối với máy tính thông qua giao tiếp USB
- Dung lượng tăng nhanh
- Thuận tiện, giá thành hợp lý

Flash disk (tiếp)



The internal components of a typical flash drive

1	USB connector
2	USB mass storage controller device
3	Test points
4	Flash memory chip
5	Crystal oscillator
6	LED
7	Write-protect switch
8	Unpopulated space for second flash memory chip



4. Các chuẩn nối ghép ổ đĩa

- Giao diện IDE-ATA (Integrated Drive Electronics – AT Attachment):

- Được IBM thiết kế để nối trực tiếp ổ cứng kèm mạch điều khiển với Bus của máy tính AT gọi là giao diện ATA. Sau đó giao diện này được kết hợp với ổ đĩa và bộ điều khiển trong các ổ đĩa → giao diện IDE/ATA.
- Giao diện IDE (mạch điện tử tích hợp trong ổ đĩa) chỉ bắt cứ ổ đĩa nào có tích hợp bộ điều khiển đĩa gắn bên trong.
 - Cáp IDE chuẩn: gồm 40 dây, tín hiệu truyền song song trên cả dây chẵn và dây lẻ nên độ dài của cáp bị hạn chế ở 46 cm.
 - Giới hạn dung lượng đĩa tối đa là 504MB và có tốc độ tương đối chậm.
- EIDE (Enhanced IDE - IDE được nâng cao):
 - Gia tăng dung lượng ổ đĩa lên tới hơn 8GB
 - Tăng tốc độ truyền tải dữ liệu lên hơn hai lần khả năng của IDE
 - Tăng gấp đôi số lượng ổ đĩa mà một máy PC có thể có



Các chuẩn nối ghép ổ đĩa (tiếp)

- Giao diện ATA gồm nhiều phiên bản:

- ATA-1 (1986-1994)
- ATA-2 (1996)
- ATA-3 (1997)
- ATA-4 (1998, còn gọi là Ultra-ATA/33 MHz)
- ATA-5 và ATA-6 (từ 1999 đến nay, còn gọi là Ultra-ATA/66/100/133 MHz). Cáp cho các chuẩn này được thiết kế gồm 80 dây để truyền dữ liệu tốc độ cao (các dây nối đất và dây tín hiệu xen kẽ nhau nhằm mục đích khử nhiễu)



Các chuẩn nối ghép ổ đĩa (tiếp)

■ Giao diện Serial ATA:

- Do một số cty lớn đưa ra vào năm 1999
- Giao tiếp Serial Advanced Technology Attachment (Serial ATA) cho ổ cứng và thiết bị ATA Packet Interface (ATAPI)
- So với Parallel ATA, Serial ATA dùng điện áp thấp, đầu chân cắm nhỏ gọn và ít dây hơn.
- Serial ATA tương thích hoàn toàn với phần mềm trước đây dành cho thiết bị Parallel ATA và ATAPI.
- Thế hệ sản phẩm Serial ATA đầu tiên xuất hiện trên thị trường vào giữa 2002, đạt tốc độ 150MBps. Trong tương lai, các phiên bản kế tiếp có thể đạt băng thông 300MBps và 600MBps.



Các chuẩn nối ghép ổ đĩa (tiếp)

- Giao diện SCSI (Small Computer System Interface):
 - Dùng để kết nối nhiều loại thiết bị có tốc độ trao đổi dữ liệu cao trong một máy tính, thường được dùng trong các máy chủ.
 - Một bus SCSI hỗ trợ tối đa 7 hoặc 15 thiết bị
 - Có nhiều chuẩn SCSI:
 - SCSI-1 (1986): truyền dữ liệu trên bus song song 8 bit, tốc độ 5 MB/s, dùng cáp 50 dây.
 - SCSI-2 (1994): truyền dữ liệu trên bus song song 16 bit, tốc độ 10 MB/s, dùng cáp 50 dây mật độ cao.
 - SCSI-3: được thiết kế cho các máy tính đời mới hiện nay, gồm 2 phiên bản: Ultra 2 SCSI (tốc độ truyền tới 40 MB/s) và Ultra 3 SCSI (tốc độ truyền tới 80 MB/s hoặc 160 MB/s)

- Redundant Array of Independent Disks
- Là tập hợp các ổ đĩa cứng vật lý mà hệ điều hành coi như là một ổ đĩa logic duy nhất
- Khi ghi lên hệ thống RAID, các tệp dữ liệu được phân mảnh và lưu trữ phân tán trên các ổ cứng vật lý
- Có khả năng tạo ra và lưu trữ thông tin dự thửa để đảm bảo khôi phục lại thông tin trong trường hợp ổ đĩa bị hỏng
- Có 7 loại phổ biến: RAID 0 ÷ 6



3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.6. Bộ nhớ ảo

- Khái niệm bộ nhớ ảo: là bộ nhớ bao gồm bộ nhớ chính và bộ nhớ ngoài mà được CPU coi như là một bộ nhớ duy nhất.
- Các kỹ thuật thực hiện bộ nhớ ảo:
 - Kỹ thuật phân trang: Chia không gian địa chỉ bộ nhớ thành các trang nhớ có kích thước bằng nhau và nằm liền kề nhau
Thông dụng: kích thước trang = 4KBytes
 - Kỹ thuật phân đoạn: Chia không gian nhớ thành các đoạn nhớ có kích thước thay đổi, các đoạn nhớ có thể gối lên nhau.



3.3. Bộ nhớ máy tính

3.3.1. Tổng quan hệ thống nhớ

3.3.2. Bộ nhớ bán dẫn

3.3.3. Bộ nhớ chính

3.3.4. Bộ nhớ cache

3.3.5. Bộ nhớ ngoài

3.3.6. Bộ nhớ ảo

3.3.7. Bộ nhớ trên máy tính cá nhân



3.3.7. Bộ nhớ trên máy tính cá nhân

1. Bộ nhớ Cache
2. RAM
3. ROM BIOS
4. CMOS RAM
5. Video RAM
6. Các loại bộ nhớ ngoài



1. Bộ nhớ Cache

- Thường được chia thành nhiều mức:
 - Cache L1:
 - Cache lệnh
 - Cache dữ liệu
 - Cache L2: 128, 256, 512 KB, 1 MB ...
- Được tích hợp trên các chip vi xử lý

- Sử dụng DRAM, thường được coi là bộ nhớ chính.
- Các loại bộ nhớ RAM:
 - FPM (Fast Page Mode) DRAM
 - EDO (Extended Data Out) DRAM
 - SDRAM (Synchronous DRAM)
 - DDR SDRAM (Double Data Rate SDRAM)
 - RDRAM (Rambus DRAM)
- Các loại module nhớ RAM:
 - Máy tính Desktop:
 - SIMM (Single Inline Memory Module)
 - DIMM (Dual Inline Memory Module)
 - RIMM (Rambus Inline Memory Module)
 - Máy tính Laptop:
 - SODIMM (Small Outline Dual Inline Memory Module)
 - MicroDIMM (Micro Dual Inline Memory Module)



a. Các loại bộ nhớ RAM

- FPM DRAM (Fast Page Mode DRAM)

- Khi truy cập bộ nhớ: địa chỉ hàng không đổi, chỉ thay đổi địa chỉ cột.
- Chế độ truy cập burst mode (từ 486) cho phép sau khi thiết lập các địa chỉ hàng, cột cho 1 lần truy cập, CPU có thể truy cập thêm 3 địa chỉ tiếp mà không có trạng thái chờ.
- Chế độ burst mode của DRAM chuẩn được mô tả dưới dạng các thông số x-y-y-y.
- VD: FPM DRAM 60ns có thông số định thời của chế độ burst mode là 5-3-3-3. Với bus hệ thống 66MHz thì mất $5 \times 15 = 75$ ns cho lần truy cập đầu và $3 \times 15 = 45$ ns cho mỗi lần trong số 3 lần truy cập tiếp theo (nhanh hơn 5-5-5-5).



Các loại bộ nhớ RAM (tiếp)

■ EDO DRAM (Extended Data Out DRAM)

- Sử dụng chủ yếu từ 1995 – 1997.
- Là dạng cải tiến của FPM DRAM: các bộ điều khiển dữ liệu ra không bị tắt khi bộ điều khiển bộ nhớ xóa địa chỉ cột cho chu kỳ tiếp theo => cho phép chu kỳ tiếp theo gói lên chu kỳ trước (tiết kiệm khoảng 10ns cho 1 chu kỳ).
- Giá thành ngang với FPM nhưng hiệu năng cao hơn.
- VD: burst mode của EDO là 5-2-2-2 (cần 11 chu kỳ cho 4 lần truyền) so với 5-3-3-3 của FPM (truyền 4 lần trong 14 chu kỳ).



Các loại bộ nhớ RAM (tiếp)

- SDRAM (Synchronous DRAM):

- Sử dụng từ 1997, chủ yếu cho các máy tính PII, PIII.
- Chạy đồng bộ với bus bộ nhớ (66, 100, 133 MHz).
- Thời gian xác định địa chỉ vẫn như cũ nhưng tổng thời gian nhanh hơn so với FPM và EDO DRAM.
- VD: SDRAM : 5-1-1-1 (cần 8 chu kỳ cho 4 lần truyền), nhanh hơn 11 và 14 chu kỳ của EDO và FPM.



Các loại bộ nhớ RAM (tiếp)

- DDR SDRAM (Double Data Rate SDRAM):
 - Xuất hiện từ năm 2000.
 - Là dạng cải tiến của SDRAM, cho phép truyền dữ liệu 2 lần ở cả sườn dương và sườn âm của 1 chu kỳ.
- DDR2 SDRAM:
 - Xuất hiện từ năm 2004.
 - Là dạng cải tiến của DDR SDRAM: sử dụng cặp dây tín hiệu vi sai cho phép truyền nhanh và ít nhiễu hơn.
 - Sử dụng điện áp thấp hơn DDR SDRAM (1.8V so với 2.5V).



Các loại module nhớ DDR SDRAM

Module Standard	Module Format	Chip Type	Clock Speed (MHz)	Cycles per Clock	Bus Speed (MT/s)	Bus Width (Bytes)	Transfer Rate (MBps)
PC1600	DDR DIMM	DDR200	100	2	200	8	1,600
PC2100	DDR DIMM	DDR266	133	2	266	8	2,133
PC2400	DDR DIMM	DDR300	150	2	300	8	2,400
PC2700	DDR DIMM	DDR333	166	2	333	8	2,667
PC3000	DDR DIMM	DDR366	183	2	366	8	2,933
PC3200	DDR DIMM	DDR400	200	2	400	8	3,200
PC3500	DDR DIMM	DDR433	216	2	433	8	3,466
PC3700	DDR DIMM	DDR466	233	2	466	8	3,733
PC4000	DDR DIMM	DDR500	250	2	500	8	4,000
PC4200	DDR DIMM	DDR533	266	2	533	8	4,266



Các loại module nhớ DDR2 SDRAM

Module Standard	Module Format	Chip Type	Clock Speed (MHz)	Cycles per Clock	Bus Speed (MT/s)	Bus Width (Bytes)	Transfer Rate (MBps)
PC2-3200	DDR2 DIMM	DDR2-400	200	2	400	8	3,200
PC2-4200	DDR2 DIMM	DDR2-533	266	2	533	8	4,266
PC2-5300	DDR2 DIMM	DDR2-667	333	2	667	8	5,333
PC2-6000	DDR2 DIMM	DDR2-750	375	2	750	8	6,000
PC2-6400	DDR2 DIMM	DDR2-800	400	2	800	8	6,400
PC2-7200	DDR2 DIMM	DDR2-900	450	2	900	8	7,200
PC2-8000	DDR2 DIMM	DDR2-1000	500	2	1000	8	8,000



Các loại bộ nhớ RAM (tiếp)

- RDRAM (Rambus DRAM):

- Là loại RAM tốc độ cao, được sản xuất theo công nghệ của hãng Rambus.
- Xuất hiện chủ yếu từ 1999 đến 2002 (sau 2001 Intel không còn hỗ trợ công nghệ này).



Các loại module nhớ RDRAM

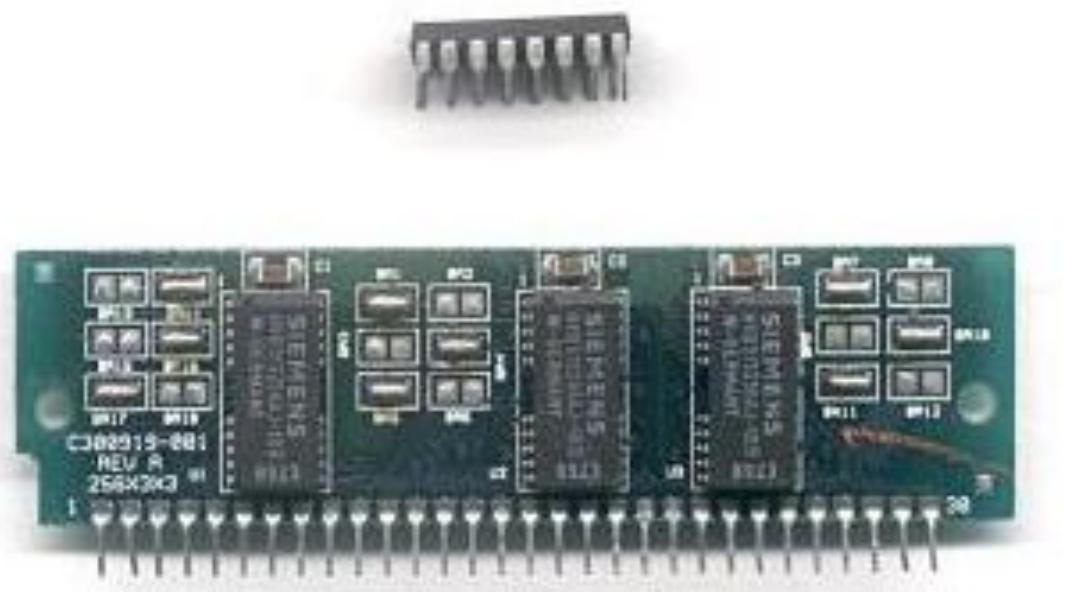
Module Standard	Module Format	Chip Type	Clock Speed (MHz)	Cycles per Clock	Bus Speed (MT/s)	Bus Width (Bytes)	Transfer Rate (Mbps)
RIMM1200	RIMM-16	PC600	300	2	600	2	1,200
RIMM1400	RIMM-16	PC700	350	2	700	2	1,400
RIMM1600	RIMM-16	PC800	400	2	800	2	1,600
RIMM2100	RIMM-16	PC1066	533	2	1,066	2	2,133
RIMM2400	RIMM-16	PC1200	600	2	1,200	2	2,400
RIMM3200	RIMM-32	PC800	400	2	800	4	3,200
RIMM4200	RIMM-32	PC1066	533	2	1,066	4	4,266
RIMM4800	RIMM-32	PC1200	600	2	1,200	4	4,800



b. Các loại module nhớ RAM

- Các module RAM thế hệ cũ:
 - DIP (Dual Inline Package)
 - SIPP (Single Inline Pin Package)
- Máy tính Desktop:
 - SIMM (Single Inline Memory Module)
 - DIMM (Dual Inline Memory Module)
 - RIMM (Rambus Inline Memory Module)
- Máy tính Laptop:
 - SODIMM (Small Outline Dual Inline Memory Module)
 - MicroDIMM (Micro Dual Inline Memory Module)

- Thường là dạng đóng gói của các module nhớ FPM DRAM.
- Dùng trong các máy tính tương đương với hệ 80286 trở về trước.



- Module nhớ đơn hàng chân, gồm 2 loại chính:
 - SIMM 32 chân (8 bit dữ liệu + 1 bit parity) : FPM DRAM
 - SIMM 72 chân (32 bit dữ liệu + 4 bit parity tùy chọn) : EDO DRAM



- Module nhớ hai hàng chân, gồm 3 loại chính:
 - DIMM 168 chân: SDRAM
 - DIMM 184 chân: DDR SDRAM
 - DIMM 240 chân: DDR2 SDRAM
- Độ rộng đường dữ liệu: 64 bit (non-ECC/parity)
hoặc 72 bit (parity/ECC).



Minh họa các module nhớ DIMM



- Module nhớ 2 hàng chân (184 chân), là dạng đóng gói của loại bộ nhớ RDRAM.



- Thường dùng trong các máy laptop, notebook, printer, router, ...
- Gồm 4 loại chính:
 - SODIMM 72 chân, 32 bit dữ liệu, FPM/EDO
 - SODIMM 144 chân, 64 bit dữ liệu, FPM/EDO
 - SODIMM 144 chân, 64 bit dữ liệu, SDRAM
 - SODIMM 200 chân, 64 bit dữ liệu, DDR/DDR2 SDRAM



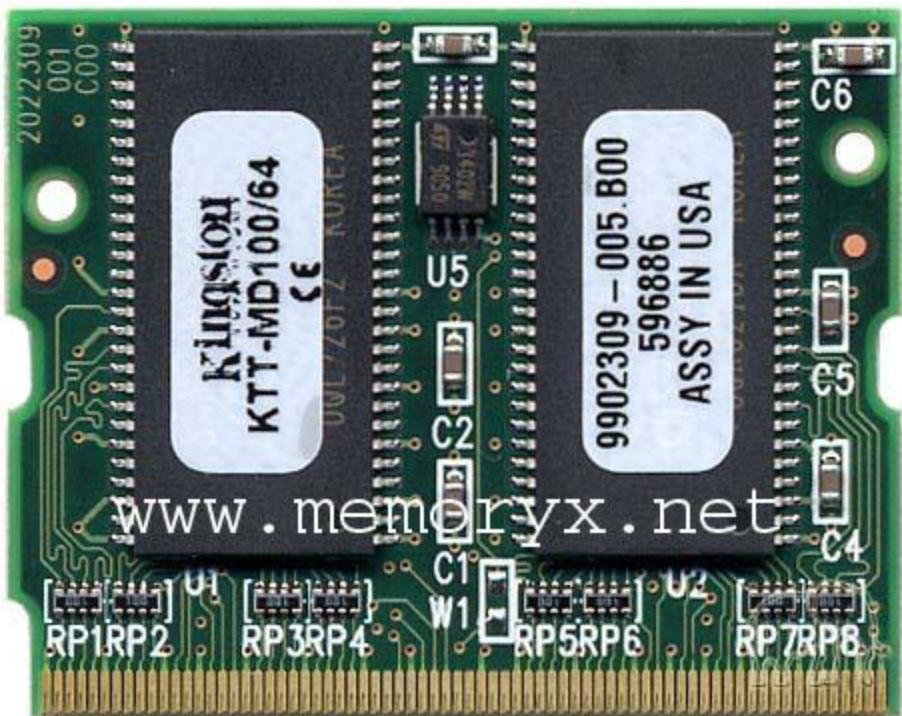
Các module nhớ SODIMM



- Thường dùng trong các máy notebook cỡ nhỏ, PDA, palmtop, ...
- Gồm 2 loại chính:
 - MicroDIMM 144 chân, 64 bit dữ liệu, SDRAM
 - MicroDIMM 172 chân, 64 bit dữ liệu, DDR SDRAM



Các module nhớ MicroDIMM





3. ROM BIOS

- BIOS: Basic Input Output System. Chứa các chương trình:
- Chương trình POST (Power On Self Test): tự kiểm tra khi bật nguồn. Mọi lỗi thông báo ở đây đều là lỗi về phần cứng.
- Chương trình CMOS Setup:
 - Cho phép người sử dụng có thể thiết lập các thông số cấu hình và thời gian của hệ thống.
 - Các thông tin sau khi thiết lập sẽ được cất vào bộ nhớ CMOS RAM.
- Chương trình Bootstrap Loader: tìm và nạp Boot Record của đĩa khởi động vào một địa chỉ xác định ở trong RAM và trao quyền điều khiển cho đoạn mã đó.
- Các chương trình điều khiển vào-ra cơ bản: tập hợp các chương trình con phục vụ vào-ra.



4. CMOS RAM

- Là một vùng nhớ có dung lượng nhỏ, được chế tạo bằng công nghệ CMOS, có một nguồn pin nuôi riêng, dùng để chứa các thông tin cấu hình và thời gian của hệ thống.



5. Video RAM

- Vùng nhớ có tốc độ nhanh, dung lượng lớn, dùng để quản lý các thông tin hiển thị trên màn hình.



6. Các loại bộ nhớ ngoài

- Đĩa mềm
- Ổ đĩa cứng
- Các loại đĩa quang
- Flash disk



3.4. Hệ thống vào ra

3.4.1. Tổng quan về hệ thống vào-ra

3.4.2. Các phương pháp điều khiển vào-ra

3.4.3. Nối ghép với thiết bị ngoại vi

3.4.4. Các cổng vào-ra thông dụng trên PC



3.4.1. Tổng quan về hệ thống vào-ra

1. Giới thiệu chung
2. Các thiết bị ngoại vi
3. Module nối ghép vào-ra
4. Các phương pháp địa chỉ hóa cổng vào-ra



1. Giới thiệu chung

- Chức năng: trao đổi thông tin giữa máy tính và hệ thống bên ngoài.
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị ngoại vi
 - Các module nối ghép vào-ra

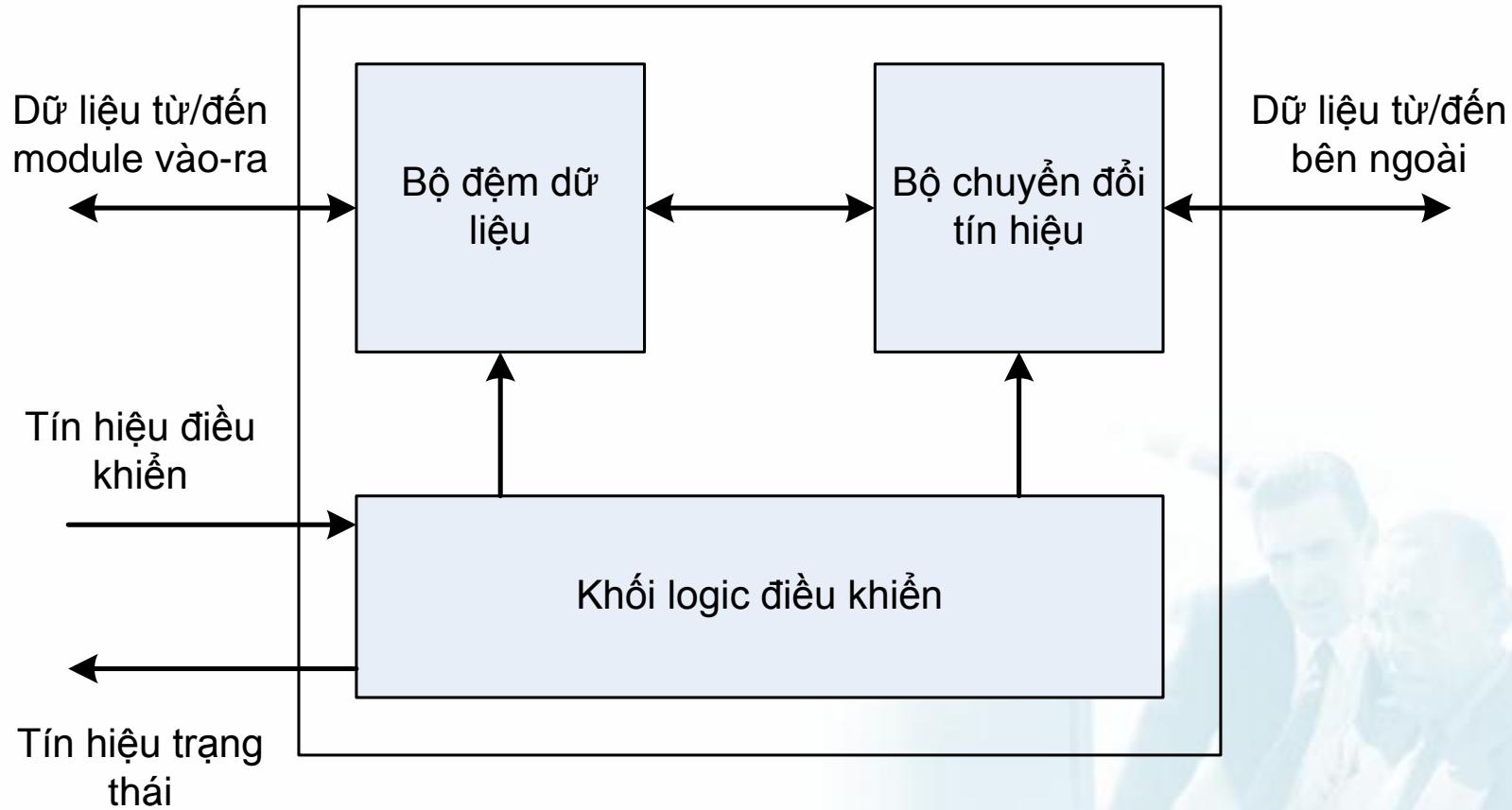


2. Các thiết bị ngoại vi

- Chức năng: Chuyển đổi thông tin từ một dạng vật lý nào đó về dạng dữ liệu phù hợp với máy tính hoặc ngược lại.
- Phân loại:
 - Các thiết bị thu nhận dữ liệu: như bàn phím, chuột, máy quét ảnh, ...
 - Các thiết bị hiển thị dữ liệu: màn hình, máy in, ...
 - Các thiết bị lưu trữ: ổ đĩa mềm, ổ đĩa cứng, ổ đĩa quang CD, DVD, ...
 - Các thiết bị truyền thông: modem, card mạng, ...



Cấu trúc chung của TBNV





Các thành phần chính của TBNV

- Bộ chuyển đổi tín hiệu: chuyển đổi dữ liệu giữa bên ngoài và bên trong máy tính.
- Bộ đệm dữ liệu: đệm dữ liệu khi truyền giữa module vào-ra và thiết bị ngoại vi.
- Khối logic điều khiển: điều khiển hoạt động của thiết bị ngoại vi đáp ứng theo yêu cầu từ module vào-ra.



3. Module vào-ra

- **Đặc điểm của vào-ra:**

- Các thiết bị ngoại vi rất đa dạng, khác nhau về:
 - Nguyên tắc hoạt động
 - Tốc độ
 - Khuôn dạng dữ liệu
- Tất cả các thiết bị ngoại vi đều chậm hơn CPU và RAM
→ Cần có các module vào-ra để nối ghép các thiết bị ngoại vi với CPU và bộ nhớ chính.



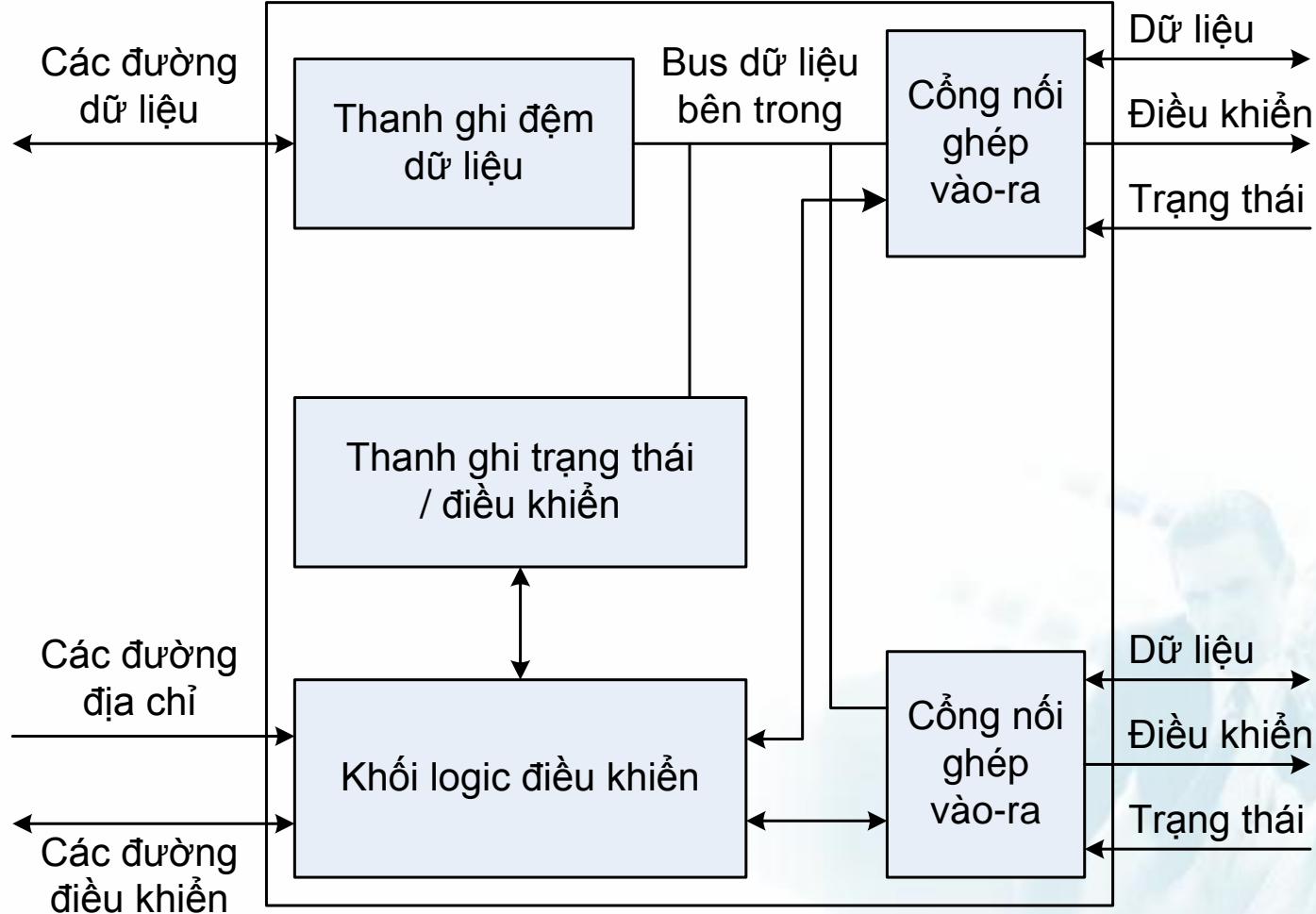
Chức năng của module vào-ra

- Chức năng:

- Điều khiển và định thời
- Trao đổi thông tin với CPU
- Trao đổi thông tin với thiết bị ngoại vi
- Đệm giữa bên trong máy tính với thiết bị ngoại vi
- Phát hiện lỗi của thiết bị ngoại vi



Cấu trúc chung của module vào-ra





Các thành phần của module vào-ra

- Thanh ghi đếm dữ liệu: đếm dữ liệu trong quá trình trao đổi.
- Các cổng vào-ra (I/O Port): kết nối với thiết bị ngoại vi, mỗi cổng có một địa chỉ xác định.
- Khối logic điều khiển: điều khiển module vào-ra.
- Thanh ghi trạng thái / điều khiển: lưu giữ thông tin trạng thái / điều khiển cho các cổng vào-ra.



4. Địa chỉ hóa cổng vào-ra

- Các thiết bị ngoại vi được nối ghép và trao đổi dữ liệu thông qua các cổng vào-ra.
- Mỗi cổng vào-ra phải có 1 địa chỉ xác định → cần phải có các phương pháp địa chỉ hóa cho cổng vào-ra.



a. KGĐC bộ nhớ và KGĐC vào-ra

- Mỗi CPU đều có khả năng quản lý được một không gian địa chỉ bộ nhớ xác định.
 - KGĐC bộ nhớ = 2^N byte (N là số bit địa chỉ mà CPU có khả năng phát ra)
- Một số CPU có khả năng quản lý thêm 1 không gian địa chỉ vào ra riêng biệt với không gian địa chỉ bộ nhớ.
 - KGĐC vào-ra = 2^{N_1} byte
 - (N_1 : số bit địa chỉ dùng để quản lý không gian địa chỉ vào-ra, $2^{N_1} \ll 2^N$)
- Trong trường hợp CPU quản lý được cả 2 KGĐC thì:
 - CPU phải có tín hiệu để phân biệt không gian địa chỉ bộ nhớ và không gian địa chỉ vào-ra.
 - CPU phải có các lệnh vào-ra chuyên dụng.

- BXL 68030 của Motorola chỉ quản lý 1 KGĐC bộ nhớ là 2^{32} byte.
- BXL Pentium của Intel có khả năng quản lý 2 KGĐC:
 - KGĐC bộ nhớ = 2^{32} byte = 4GB
 - KGĐC vào-ra = 2^{16} byte = 64KB
- Pentium có:
 - Tín hiệu điều khiển phân biệt truy nhập không gian địa chỉ: IO/M
 - Có 2 lệnh vào-ra chuyên dụng: IN và OUT



b. Các pp địa chỉ hóa cổng vào-ra

- Vào ra riêng biệt (Isolated I/O):
 - Cổng vào-ra được địa chỉ hóa theo không gian địa chỉ vào-ra riêng biệt.
 - Để trao đổi dữ liệu với cổng, trong chương trình sử dụng các lệnh vào-ra chuyên dụng.
- Vào ra theo bản đồ bộ nhớ (Memory-mapped IO):
 - Cổng vào-ra được địa chỉ hóa theo không gian địa chỉ bộ nhớ.
 - Để trao đổi dữ liệu với cổng, trong chương trình sử dụng các lệnh trao đổi dữ liệu với bộ nhớ.



3.4. Hệ thống vào ra

3.4.1. Tổng quan về hệ thống vào-ra

3.4.2. Các phương pháp điều khiển vào-ra

3.4.3. Nối ghép với thiết bị ngoại vi

3.4.4. Các cổng vào-ra thông dụng trên PC



3.4.2. Các pp điều khiển vào-ra

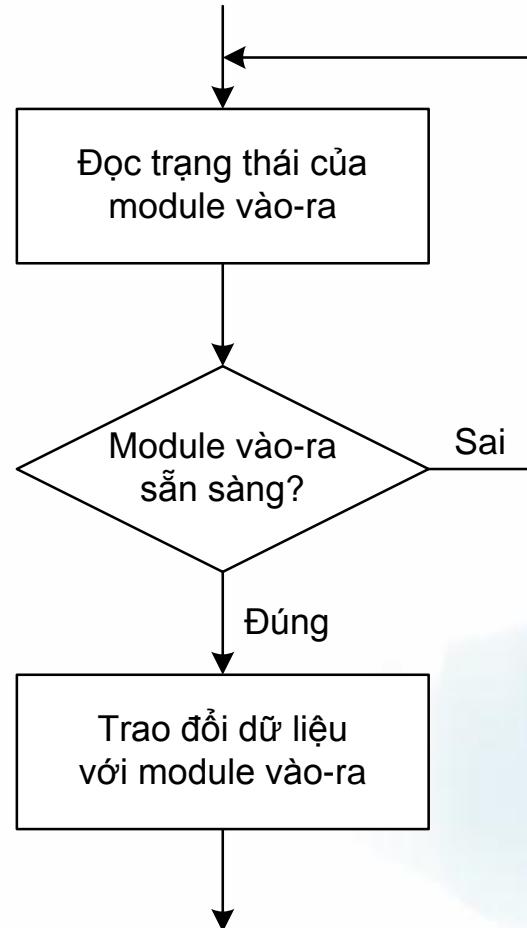
1. Vào-ra bằng chương trình
2. Vào-ra điều khiển bằng ngắt
3. Truy cập trực tiếp bộ nhớ - DMA
4. Bộ xử lý vào-ra



1. Vào-ra bằng chương trình

- Nguyên tắc chung:
 - Trong chương trình người lập trình chủ động viết các lệnh vào-ra.
 - Khi thực hiện các lệnh vào-ra đó, CPU trực tiếp điều khiển việc trao đổi dữ liệu với cổng vào-ra.

Lưu đồ thực hiện



- CPU yêu cầu thao tác vào-ra.
- Module vào-ra thực hiện thao tác.
- Module vào-ra thiết lập các bit trạng thái.
- CPU kiểm tra các bit trạng thái:
 - Nếu chưa sẵn sàng thì quay lại tiếp tục kiểm tra.
 - Nếu đã sẵn sàng thì chuyển sang trao đổi dữ liệu với module vào-ra.

- Vào-ra do ý muốn của người lập trình
- CPU trực tiếp điều khiển vào-ra
- CPU phải đợi module vào-ra sẵn sàng → tiêu tốn thời gian của CPU



2. Vào-ra điều khiển bằng ngắt

- Nguyên tắc chung:
 - CPU không phải đợi trạng thái sẵn sàng của module vào-ra.
 - CPU đang thực hiện một chương trình nào đó, nếu module vào-ra sẵn sàng thì nó phát tín hiệu yêu cầu ngắt gửi đến CPU.
 - Nếu yêu cầu ngắt được chấp nhận thì CPU thực hiện chương trình con vào-ra tương ứng để trao đổi dữ liệu.
 - Kết thúc chương trình con đó, CPU quay trở lại tiếp tục thực hiện chương trình đang bị ngắt.

- Hoạt động vào dữ liệu – nhìn từ phía module vào-ra:
 - Module vào-ra nhận tín hiệu điều khiển đọc từ CPU.
 - Module vào-ra nhận dữ liệu từ thiết bị ngoại vi, trong khi đó CPU làm việc khác.
 - Khi đã có dữ liệu, module vào-ra phát tín hiệu ngắt CPU.
 - CPU yêu cầu dữ liệu.
 - Module vào-ra chuyển dữ liệu đến CPU.

- Hoạt động vào dữ liệu – nhìn từ phía CPU:
 - CPU phát tín hiệu điều khiển đọc.
 - CPU làm việc khác.
 - Cuối mỗi chu trình lệnh, CPU kiểm tra tín hiệu ngắt.
 - Nếu bị ngắt, CPU:
 - Cắt ngũ cảnh hiện tại của chương trình.
 - Thực hiện chương trình con phục vụ ngắt để vào dữ liệu.
 - Sau khi hoàn thành chương trình con đó, CPU khôi phục ngũ cảnh và trở về tiếp tục thực hiện chương trình đang tạm dừng.



Các vấn đề nảy sinh khi thiết kế

- Làm thế nào để xác định được module vào-ra nào phát tín hiệu yêu cầu ngắt.
- Khi có nhiều yêu cầu ngắt cùng gửi đến, CPU sẽ xử lý như thế nào.





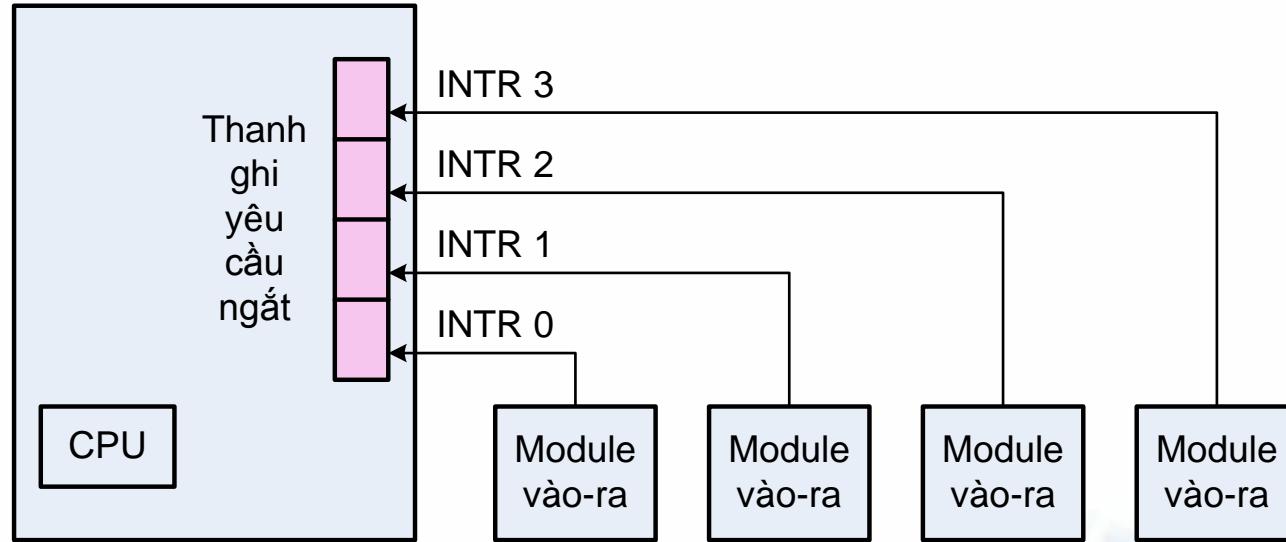
Các phương pháp nối ghép ngắn

- Sử dụng nhiều đường yêu cầu ngắn
- Kiểm tra vòng bằng phần mềm (Software Poll)
- Kiểm tra vòng bằng phần cứng (Daisy Chain of Hardware Poll)
- Sử dụng bộ điều khiển ngắn (PIC)





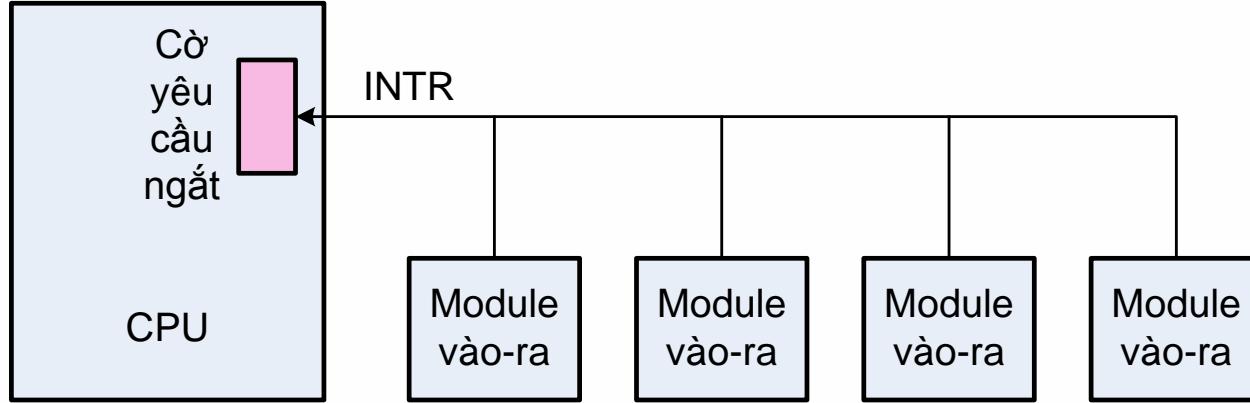
Sử dụng nhiều đường yêu cầu ngắt



- Mỗi module vào-ra được nối với 1 đường yêu cầu ngắt
- CPU phải có nhiều đường tín hiệu yêu cầu ngắt
- Hạn chế số lượng module vào-ra
- Các đường yêu cầu ngắt được quy định mức ưu tiên



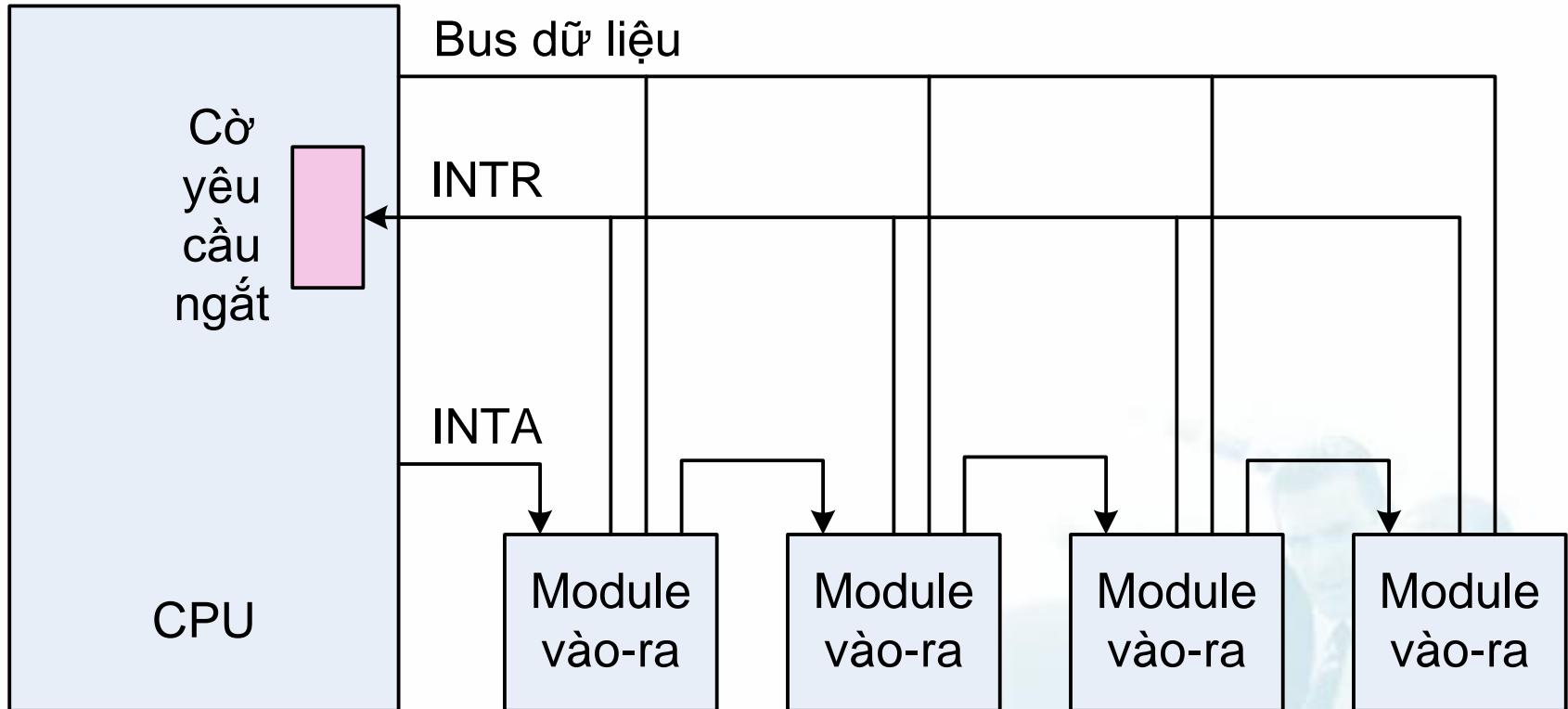
Kiểm tra vòng bằng phần mềm



- CPU thực hiện phần mềm hỏi lần lượt từng module vào-ra
- Tốc độ chậm
- Thứ tự các module vào-ra được hỏi vòng chính là thứ tự ưu tiên



Kiểm tra vòng bằng phần cứng



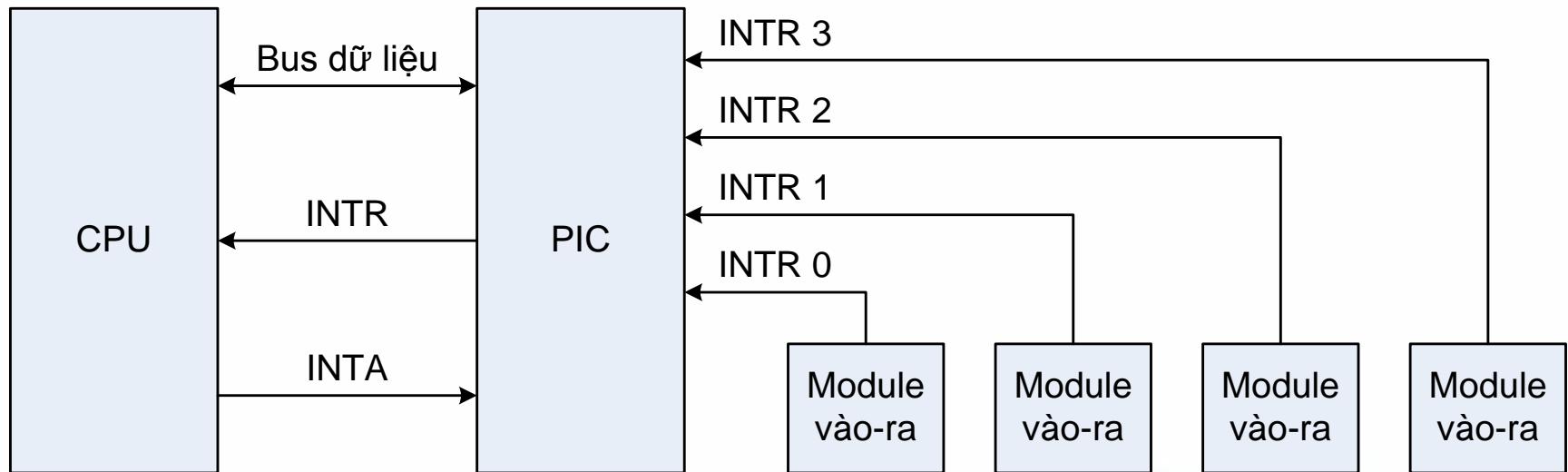


Kiểm tra vòng bằng phần cứng (tiếp)

- CPU phát tín hiệu chấp nhận ngắn (INTA) đến module vào-ra đầu tiên.
- Nếu module vào-ra đó không gây ra ngắn thì nó gửi tín hiệu đến module kế tiếp cho đến khi xác định được module gây ngắn.
- Module vào-ra gây ngắn sẽ đặt vector ngắn lên bus dữ liệu.
- CPU sử dụng vector ngắn để xác định nơi chứa chương trình con phục vụ ngắn.
- Thứ tự các module vào-ra kết nối trong chuỗi xác định thứ tự ưu tiên.



Bộ điều khiển ngắt lập trình được



- PIC: Programmable Interrupt Controller.
- PIC có nhiều đường yêu cầu ngắt có quy định mức ưu tiên.
- PIC chọn một yêu cầu ngắt không bị cấm có mức ưu tiên cao nhất gửi đến CPU.



Tổ chức ngắt của 80x86

- Tổ chức kiểu vector ngắt.
- Mỗi ngắt được đặc trưng bằng số hiệu ngắt N ($00 \div FF$).
- Bảng vector ngắt:
 $256 \times 4 = 1024$ byte
 $00000 \div 003FF$
- Gọi CTC phục vụ ngắt bằng lệnh: INT N

Địa chỉ	Bộ nhớ	
00000	IP của INT 0	Vector 0 : Chia cho 0
00002	CS của INT 0	
00004	IP của INT 1	
00006	CS của INT 1	Vector 1 : Chạy từng lệnh
00008	IP của INT 0	
0000A	CS của INT 0	
0000C	IP của INT 1	Vector 2 : Ngắt không che đưọc (NMI)
0000E	CS của INT 1	
00010	IP của INT 0	
00012	CS của INT 0	Vector 3 : Điểm dừng
00014	IP của INT 1	
00016	CS của INT 1	
		Vector 4 : Tràn số học
00018	IP của INT 0	Vector 5
0001A	CS của INT 0	
0001C	IP của INT 1	
0001E	CS của INT 1	Được định nghĩa trước hoặc dành riêng
00020	IP của INT 0	
00022	CS của INT 0	
00024	IP của INT 1	Vector 31
00026	CS của INT 1	
00028	IP của INT 0	
0002A	CS của INT 0	Vector 32
0002C	IP của INT 1	
0002E	CS của INT 1	
00030	IP của INT 0	Vector 254
00032	CS của INT 0	
00034	IP của INT 1	
00036	CS của INT 1	Vector 255
00038	IP của INT 0	
0003A	CS của INT 0	Dành cho người lập trình
0003C	IP của INT 1	
0003E	CS của INT 1	



Đặc điểm của vào-ra bằng ngắt

- Có sự kết hợp giữa phần cứng và phần mềm:
 - Phần cứng: gây ngắt CPU.
 - Phần mềm: trao đổi dữ liệu.
- CPU trực tiếp điều khiển vào-ra.
- CPU không phải đợi module vào-ra → hiệu suất sử dụng CPU tốt hơn.

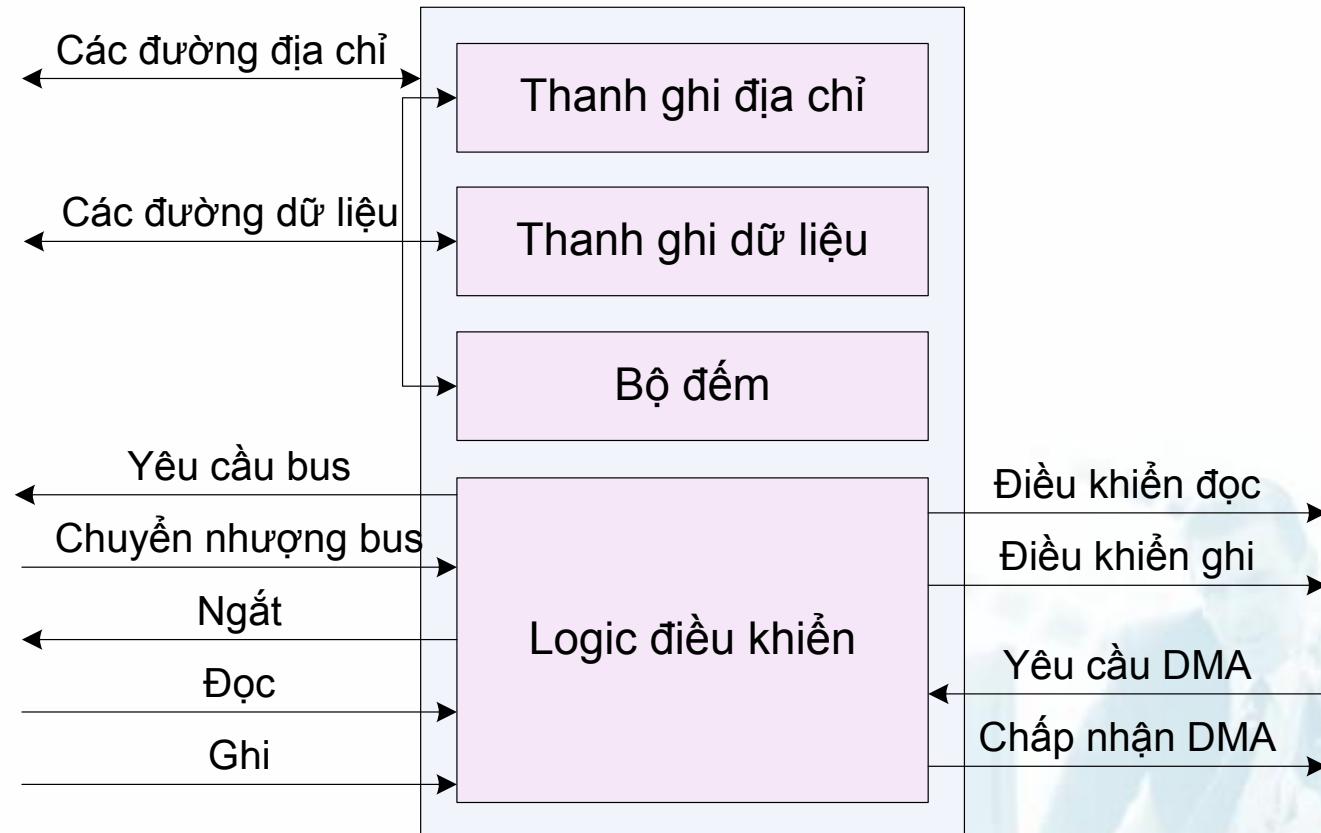




3. Truy cập trực tiếp bộ nhớ

- DMA (Direct Memory Access)
- Các phương pháp vào-ra bằng chương trình và vào-ra điều khiển bằng ngắt do CPU trực tiếp điều khiển:
 - Chiếm thời gian của CPU
 - Tốc độ trao đổi dữ liệu bị hạn chế vì phải chuyển qua CPU
- Để khắc phục → dùng DMA:
 - Thêm module phần cứng là DMAC (Direct Memory Access Controller)
 - DMAC điều khiển trao đổi dữ liệu giữa module vào-ra với bộ nhớ chính.

Cấu trúc của DMA





Các thành phần của DMAc

- Thanh ghi dữ liệu: chứa dữ liệu cần trao đổi
- Thanh ghi địa chỉ: chứa địa chỉ ngăn nhớ dữ liệu
- Bộ đếm dữ liệu: chứa số từ dữ liệu cần trao đổi
- Logic điều khiển: điều khiển hoạt động của DMAc



Hoạt động của DMA

- CPU gửi cho DMA các thông tin:
 - Chiều trao đổi dữ liệu: vào hay ra dữ liệu
 - Địa chỉ thiết bị vào-ra (cổng vào-ra tương ứng)
 - Địa chỉ đầu của mảng nhớ dữ liệu → nạp vào thanh ghi địa chỉ
 - Số từ dữ liệu cần truyền → nạp vào bộ đếm dữ liệu
- CPU làm việc khác
- DMA điều khiển trao đổi dữ liệu
- Sau khi truyền được 1 từ dữ liệu:
 - Nội dung thanh ghi địa chỉ tăng
 - Nội dung bộ đếm dữ liệu giảm
- Khi bộ đếm dữ liệu = 0, DMA gửi yêu cầu ngắt đến CPU để báo hiệu đã kết thúc DMA



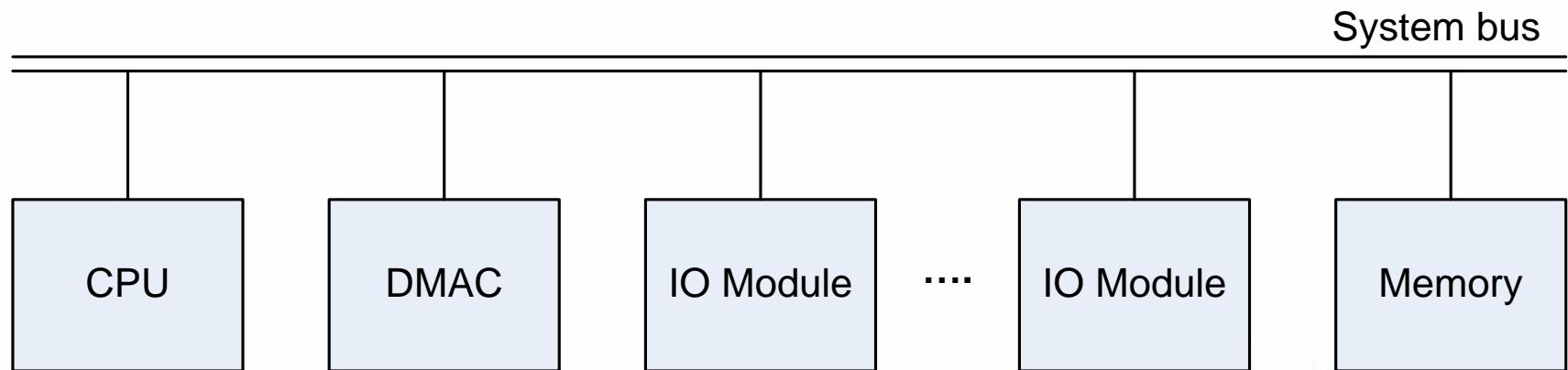
Các kiểu DMA

- DMA truyền theo khối (Block Transfer DMA): CPU trao quyền sử dụng bus cho DMAC trong một khoảng thời gian đủ lớn để DMAC thực hiện trao đổi xong cả khối dữ liệu.
- DMA xen kẽ chu kỳ máy với CPU (Cycle Stealing DMA): DMAC và CPU thay nhau sử dụng bus trong từng chu kỳ máy.
- DMA trong suốt (Transparent DMA): Trong quá trình hoạt động, không phải chu kỳ nào CPU cũng sử dụng bus hệ thống, DMAC sẽ phát hiện xem những chu kỳ CPU không dùng bus để chiếm dụng bus trong chu kỳ đó và điều khiển trao đổi 1 từ dữ liệu → không làm ảnh hưởng đến CPU.



Các cấu hình thiết kế DMA

■ Cấu hình 1:

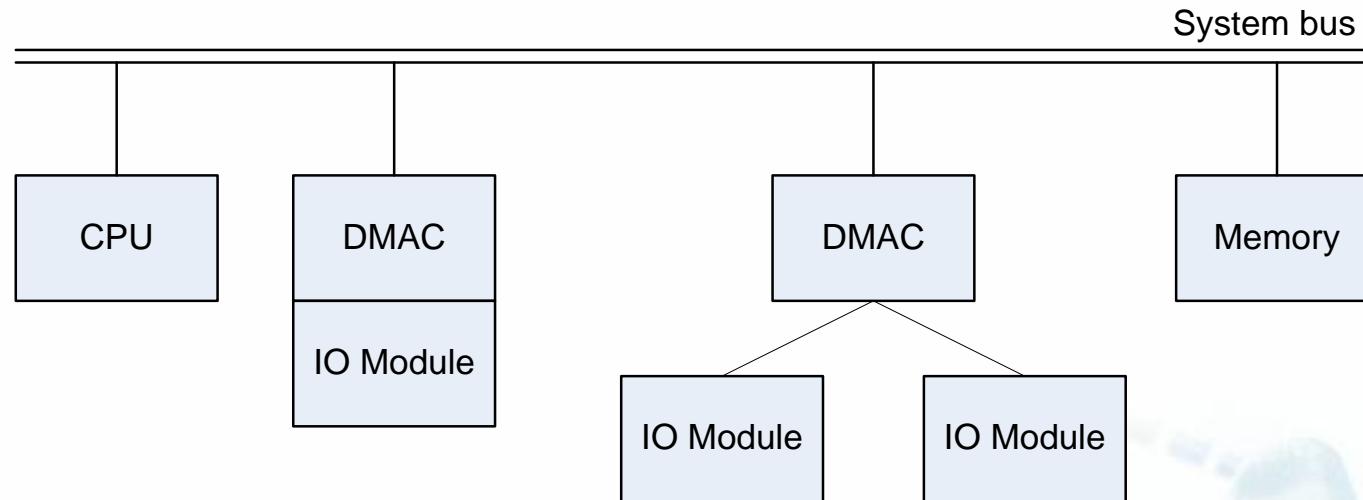


- Mỗi lần truyền, DMAC sử dụng bus 2 lần:
 - Giữa DMAC với module vào-ra
 - Giữa DMAC với bộ nhớ



Các cấu hình thiết kế DMA (tiếp)

■ Cấu hình 2:

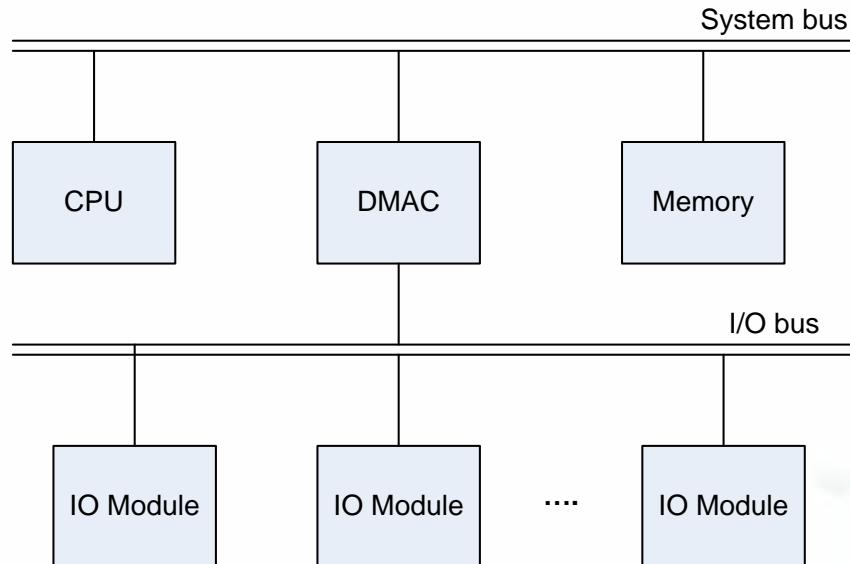


- DMAc điều khiển một hoặc một vài module vào-ra
- Mỗi lần truyền, DMAc sử dụng bus 1 lần:
 - Giữa DMAc với bộ nhớ



Các cấu hình thiết kế DMA (tiếp)

■ Cấu hình 3:



- Bus vào-ra tách rời hỗ trợ tất cả các thiết bị cho phép DMA
- Mỗi lần truyền, DMAC sử dụng bus 1 lần:
 - Giữa DMAC với bộ nhớ



Đặc điểm của DMA

- CPU không tham gia vào quá trình trao đổi dữ liệu
- DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính với module vào-ra hoàn toàn bằng phần cứng → tốc độ nhanh.
- Thích hợp với các yêu cầu trao đổi dữ liệu kích thước lớn.



4. Bộ xử lý vào-ra

- Việc điều khiển vào-ra được thực hiện bởi một bộ xử lý vào-ra chuyên dụng.
- Bộ xử lý vào-ra hoạt động theo chương trình của riêng nó.
- Chương trình của bộ xử lý vào-ra có thể nằm trong bộ nhớ chính hoặc nằm trong một bộ nhớ riêng.
- Hoạt động theo kiến trúc đa xử lý.



3.4. Hệ thống vào ra

3.4.1. Tổng quan về hệ thống vào-ra

3.4.2. Các phương pháp điều khiển vào-ra

3.4.3. Nối ghép với thiết bị ngoại vi

3.4.4. Các cổng vào-ra thông dụng trên PC



3.4.3. Nối ghép với thiết bị ngoại vi

1. Các kiểu nối ghép
2. Các cấu hình nối ghép





1. Các kiểu nối ghép

- Nối ghép song song:
 - Truyền nhiều bit song song
 - Tốc độ nhanh
 - Cần nhiều đường truyền





Các kiểu nối ghép (tiếp)

- Nối ghép nối tiếp:
 - Truyền lần lượt từng bit
 - Cần có bộ truyền đổi qua lại giữa dữ liệu song song và nối tiếp
 - Tốc độ chậm hơn
 - Cần ít đường dây → truyền được xa hơn





2. Các cấu hình nối ghép

- Cấu hình điểm tới điểm (Point to Point): thông qua một cổng vào-ra cho phép nối ghép với một thiết bị ngoại vi.
 - Nối ghép bàn phím
 - Nối ghép chuột
 - Nối ghép ổ đĩa mềm
 - ...
- Cấu hình điểm tới đa điểm (Point to Multipoint): thông qua một cổng vào-ra cho phép nối ghép với nhiều thiết bị ngoại vi.
 - Chuẩn nối ghép SCSI: cho phép nối ghép tới 7 hoặc 15 thiết bị
 - Cổng USB: nối ghép tới 127 thiết bị
 - Cổng IEEE 1394: nối ghép tới 63 thiết bị



3.4. Hệ thống vào ra

3.4.1. Tổng quan về hệ thống vào-ra

3.4.2. Các phương pháp điều khiển vào-ra

3.4.3. Nối ghép với thiết bị ngoại vi

3.4.4. Các cổng vào-ra thông dụng trên PC

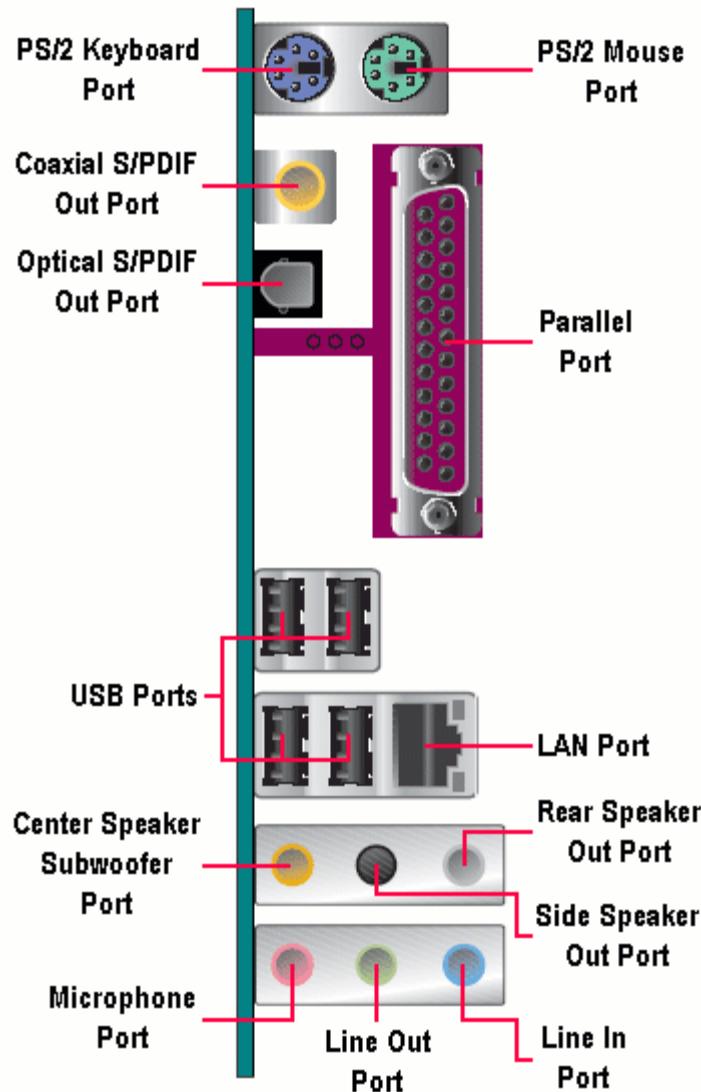
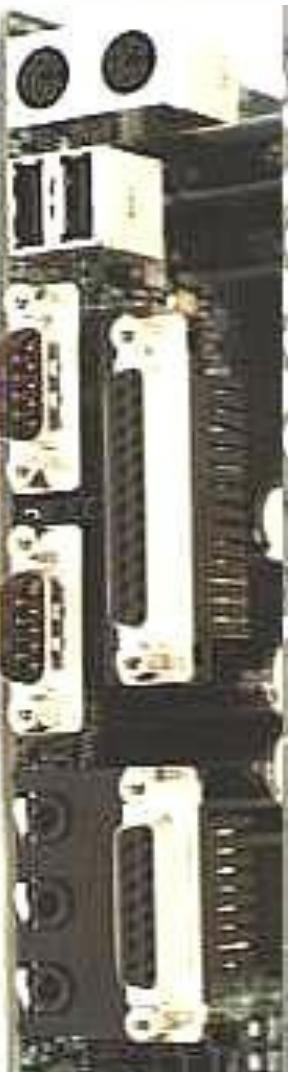


3.4.4. Các cổng vào-ra thông dụng

- Các cổng PS/2: nối ghép bàn phím và chuột
- Các cổng nối ghép màn hình
- Cổng LPT (Line Printer): thường nối ghép với máy in, là cổng song song (Parallel Port)
- Cổng COM (Communication): thường nối ghép với MODEM, là cổng nối tiếp (Serial Port)
- Cổng USB (Universal Serial Bus): cổng nối tiếp đa năng
- ...



Các cổng vào-ra thông dụng (tiếp)





Chương 4 Bộ vi xử lý Intel 8088

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trả lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ

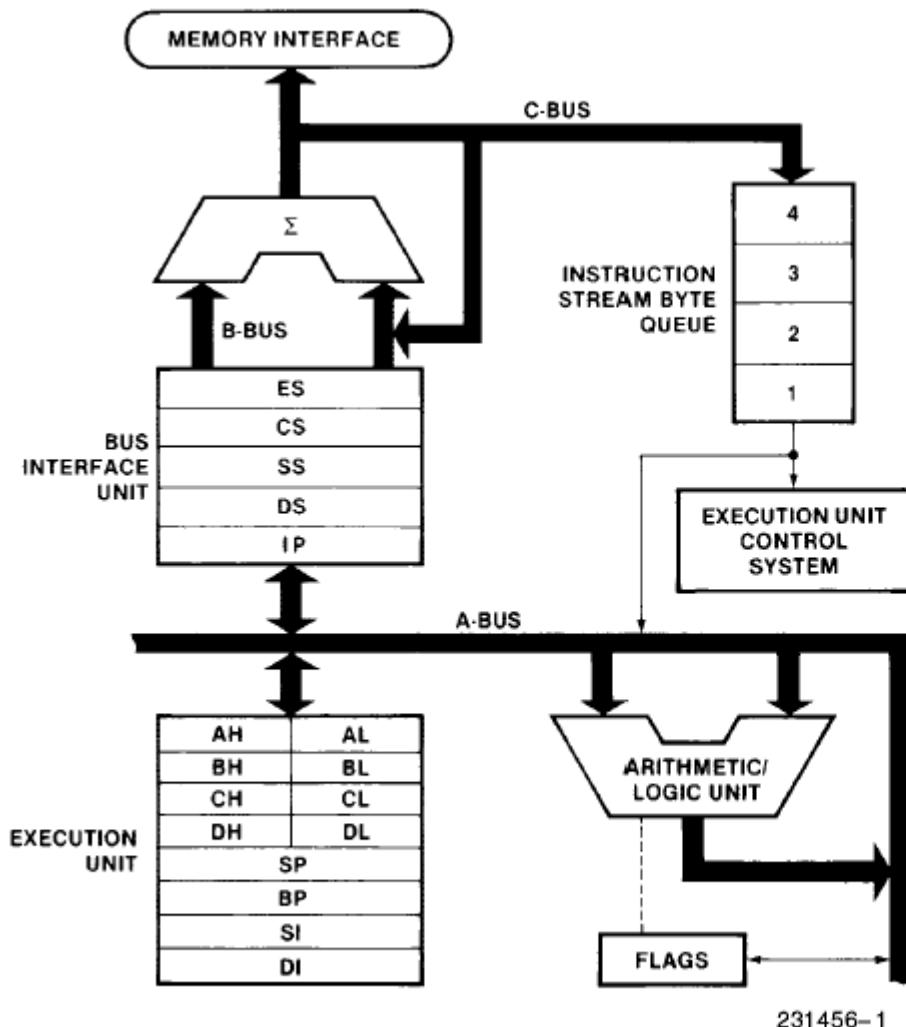


Bộ vi xử lý 8088/8086

- Hai BXL 8088 và 8086 có cấu tạo tương tự nhau, điểm khác nhau cơ bản là:
 - 8088: Bus dữ liệu ngoài là 8 bit
 - 8086: Bus dữ liệu ngoài là 16 bit
- Hệ thống máy tính dùng 8088 chậm hơn 8086 nhưng có giá thành rẻ hơn (do dùng bus dữ liệu ngoài 8 bit nên giảm được khá nhiều chip ghép nối và bổ trợ).
- Hãng IBM đã sử dụng 8088 để thiết kế máy IBM-PC (1981).



4.1. Cấu trúc bên trong của 8088



	MIN MODE	MAX MODE
GND	1	40 V _{CC}
A14	2	39 A15
A13	3	38 A16/S3
A12	4	37 A17/S4
A11	5	36 A18/S5
A10	6	35 A19/S6
A9	7	34 SS0 (HIGH)
A8	8	33 MN/MX
AD7	9	32 RD
AD6	10	8088 CPU 31 HOLD (RQ/GTO)
AD5	11	30 HLDA (RQ/GT1)
AD4	12	29 WR (LOCK)
AD3	13	28 IO/M (S2)
AD2	14	27 DT/R (S1)
AD1	15	26 DEN (S0)
AD0	16	25 ALE (QS0)
NMI	17	24 INTA (QS1)
INTR	18	23 TEST
CLK	19	22 READY
GND	20	21 RESET

Figure 2. 8088 Pin Configuration

231456-2



Cấu trúc bên trong của 8088

- Gồm 2 phần:
 - Đơn vị nối ghép bus (Bus Interface Unit – BIU)
 - Đơn vị thực hiện (Execution Unit – EU)
- Hai phần này có thể hoạt động đồng thời: trong khi EU đang thực hiện lệnh trước thì BIU đã tìm và nhận lệnh tiếp theo từ bộ nhớ chính.



Bus Interface Unit - BIU

- Bao gồm:
 - Các thanh ghi đoạn
 - Con trỏ lệnh
 - Mạch tạo địa chỉ và điều khiển bus
 - Hàng đợi lệnh (8088: 4 Byte, 8086: 6 Byte)
- Nhiệm vụ:
 - Tạo và phát địa chỉ
 - Nhận lệnh từ bộ nhớ
 - Trao đổi dữ liệu với bộ nhớ chính và cổng vào-ra
 - Phát tín hiệu điều khiển bộ nhớ và mạch vào-ra
 - Nhận các tín hiệu yêu cầu từ bên ngoài

- Gồm:
 - Các thanh ghi chung
 - Các thanh ghi đệm
 - Đơn vị số học và logic (ALU)
 - Khối giải mã lệnh
- Nhiệm vụ:
 - Giải mã lệnh
 - Thực hiện lệnh



Nội dung chương 4

4.1. Cấu trúc bên trong của 8088

4.2. Mô hình lập trình của 8088

4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ

4.4. Đoạn lệnh và thanh ghi con trả lệnh

4.5. Stack và các thanh ghi BP, SP

4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX

4.7. Các thanh ghi AX, BX, CX, DX

4.8. Thanh ghi cờ

4.9. Tập lệnh và các chế độ địa chỉ



4.2. Mô hình lập trình của 8088

- Là mô hình mà người lập trình có thể can thiệp được.
- Bao gồm:
 - Tập thanh ghi
 - Không gian nhớ
 - Không gian vào-ra
 - Các kiểu dữ liệu



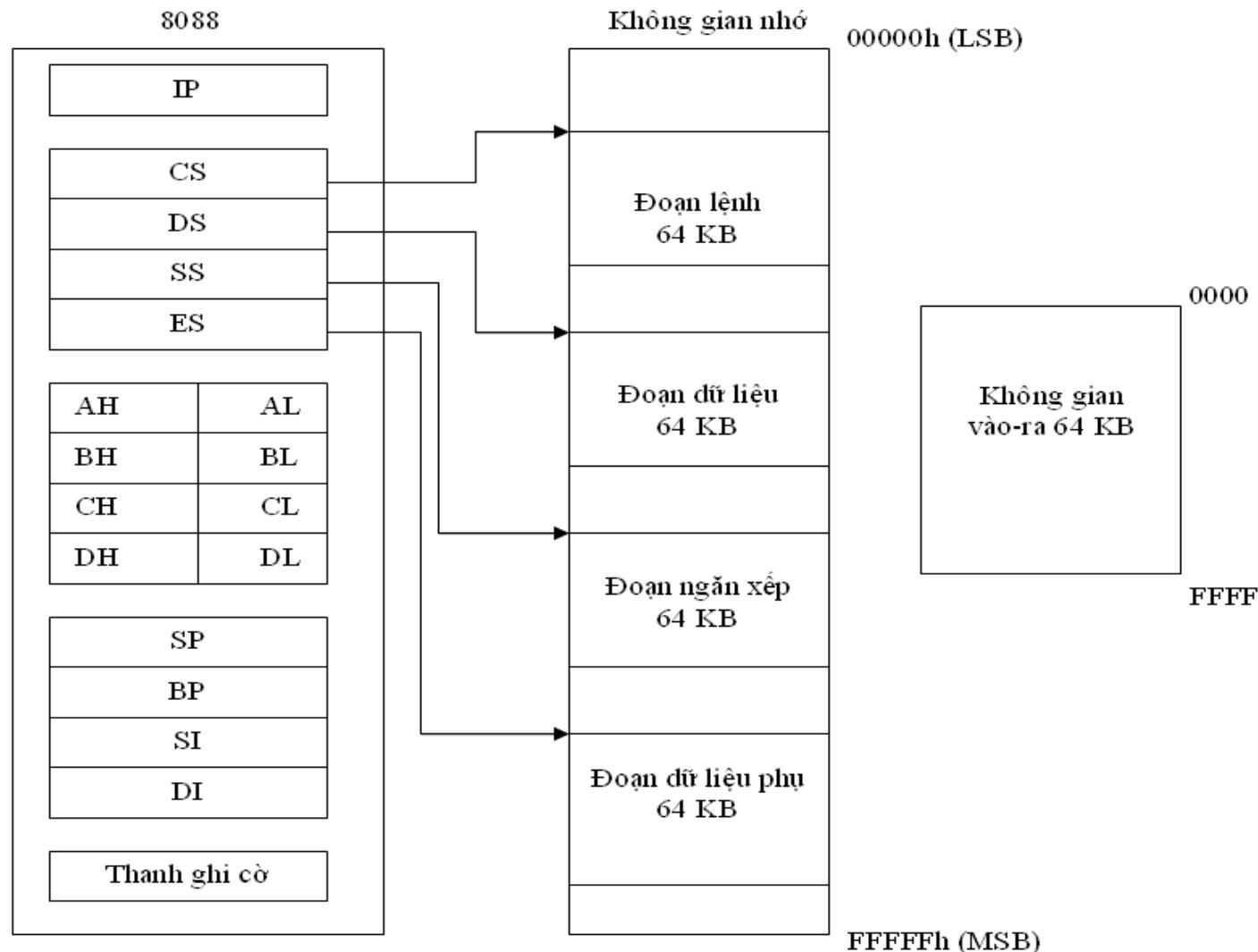


Tập thanh ghi

- 4 thanh ghi đoạn:
 - CS (Code Segment): thanh ghi đoạn lệnh
 - DS (Data Segment): thanh ghi đoạn dữ liệu
 - SS (Stack Segment): thanh ghi đoạn ngắn xếp
 - ES (Extra Segment): thanh ghi đoạn dữ liệu phụ
- 3 thanh ghi con trỏ:
 - IP (Instruction Pointer): thanh ghi con trỏ lệnh
 - SP (Stack Pointer): con trỏ ngắn xếp
 - BP (Base Pointer): thanh ghi con trỏ cơ sở
- 4 thanh ghi dữ liệu:
 - AX (Accumulator): thanh chứa - thanh ghi tích lũy
 - BX (Base): thanh ghi cơ sở
 - CX (Count): thanh ghi đếm
 - DX (Data): thanh ghi dữ liệu

Mỗi thanh ghi này đều có thể được chia ra thành 2 nửa có khả năng sử dụng độc lập.
- 2 thanh ghi chỉ số:
 - SI (Source Index): thanh ghi chỉ số nguồn
 - DI (Destination Index): thanh ghi chỉ số đích
- Thanh ghi cờ

Tập thanh ghi (tiếp)



- 8088 có bus địa chỉ 20 bit \Rightarrow KGĐCBN = 2^{20} Byte = 1MB.
- 8088 có khả năng truy nhập bộ nhớ theo:
 - Từng byte
 - Từng word: truy nhập theo 2 byte có địa chỉ liên tiếp
- 8088 lưu trữ thông tin trong bộ nhớ chính theo kiểu đầu nhỏ (Little-endian)

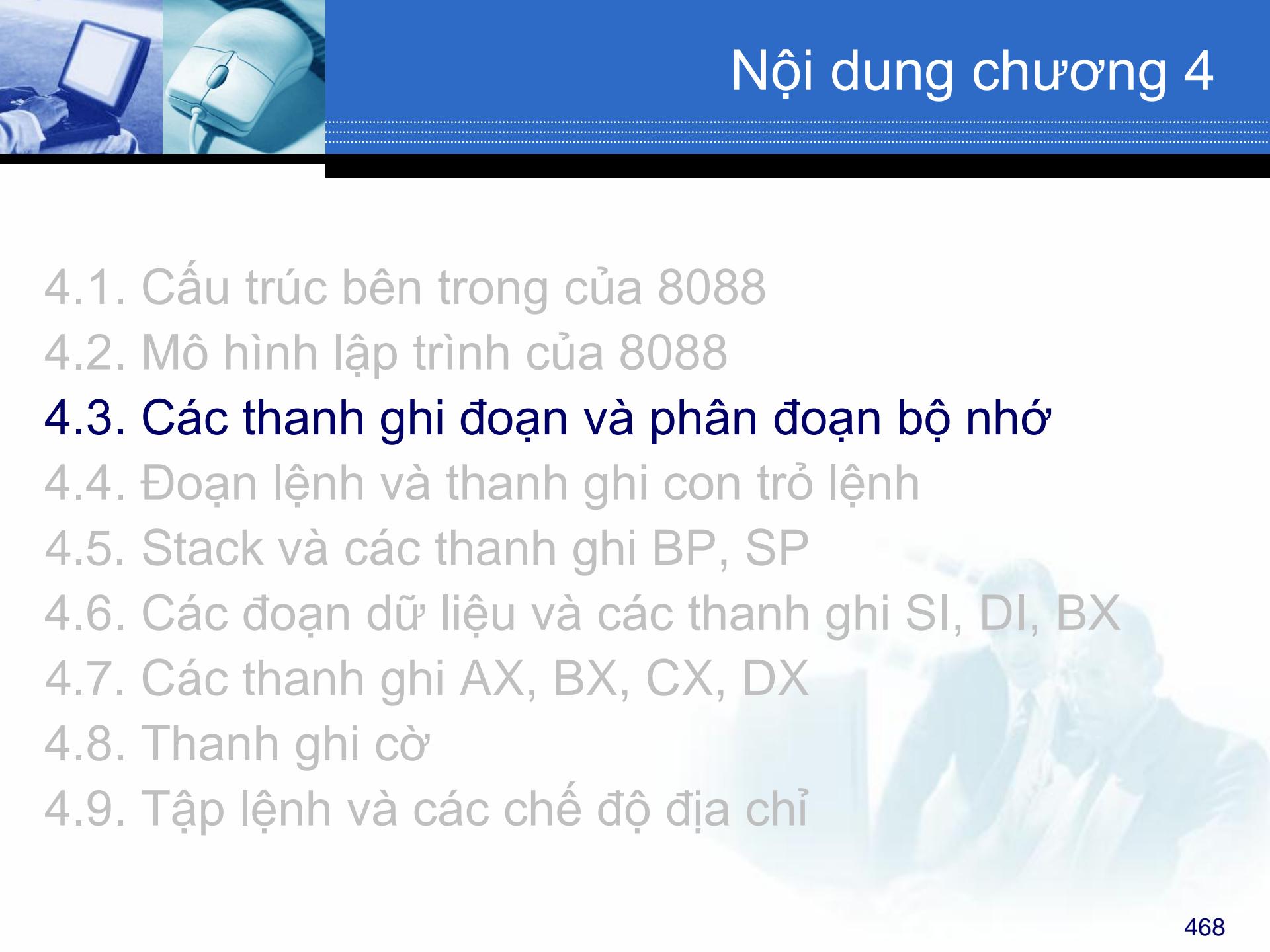




Không gian vào-ra

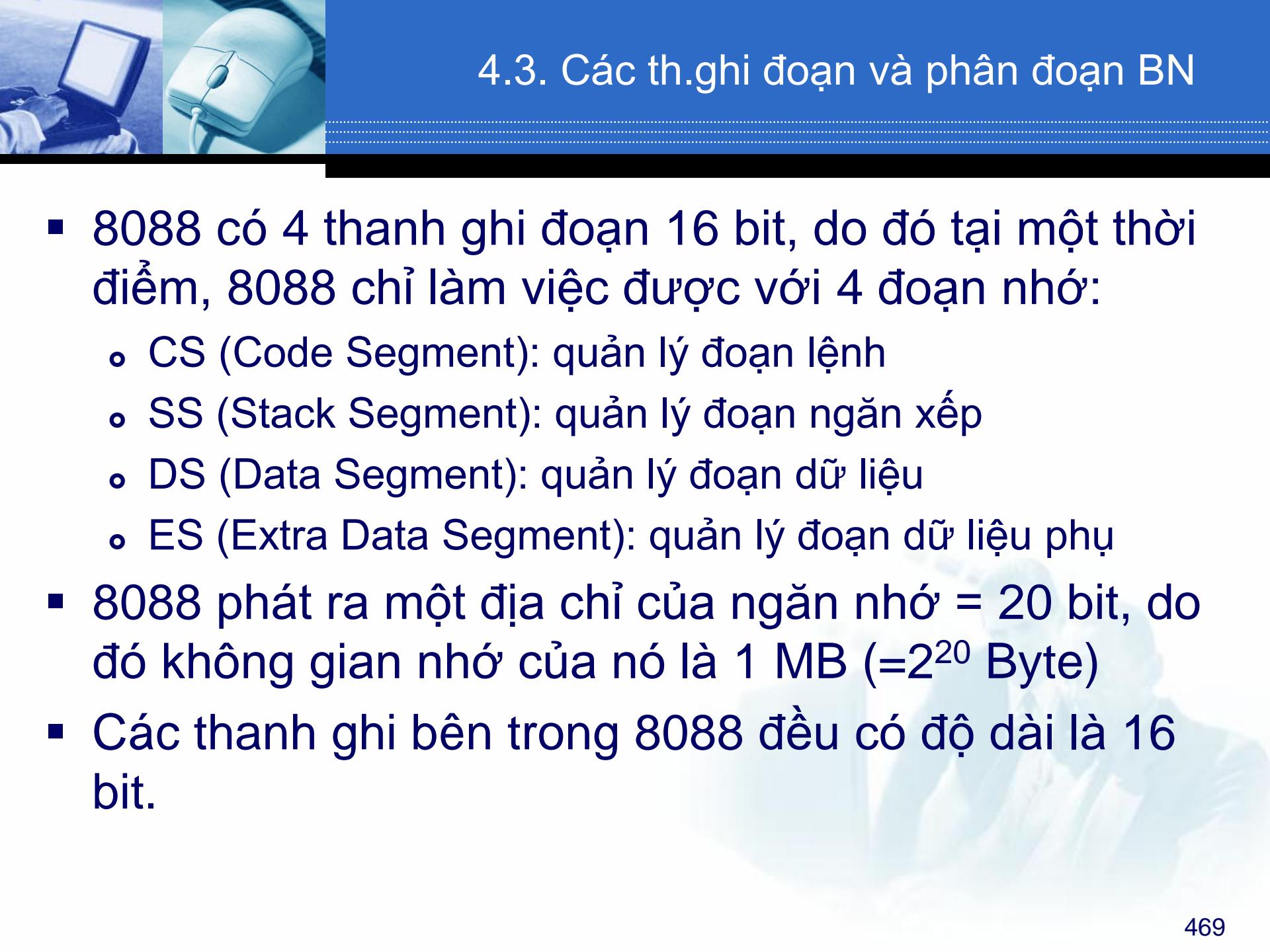
- 8088 có khả năng quản lý không gian vào-ra 64 KB = 2^{16} Byte, do đó sẽ phải phát ra 16 bit địa chỉ để tìm cổng vào-ra tương ứng trên các chân địa chỉ từ A₀ đến A₁₅.
- Trong trường hợp phát ra 8 bit địa chỉ từ A₀ đến A₇ để xác định một cổng vào-ra thì sẽ quản lý được 256 cổng vào-ra.

- Kiểu dữ liệu số nguyên, gồm 2 loại:
 - Không dấu:
 - 8 bit (1 byte), biểu diễn các số từ 0 đến 255
 - 16 bit (2 byte), biểu diễn các số từ 0 đến 65535
 - Có dấu:
 - 8 bit (1 byte), biểu diễn các số từ -128 đến 127
 - 16 bit (2 byte), biểu diễn các số từ -32768 đến 32767
- Kiểu dữ liệu số BCD, gồm 2 dạng: dạng nén và dạng không nén.
- Mã ASCII: tổ chức theo từng byte, theo mã 8 bit.



Nội dung chương 4

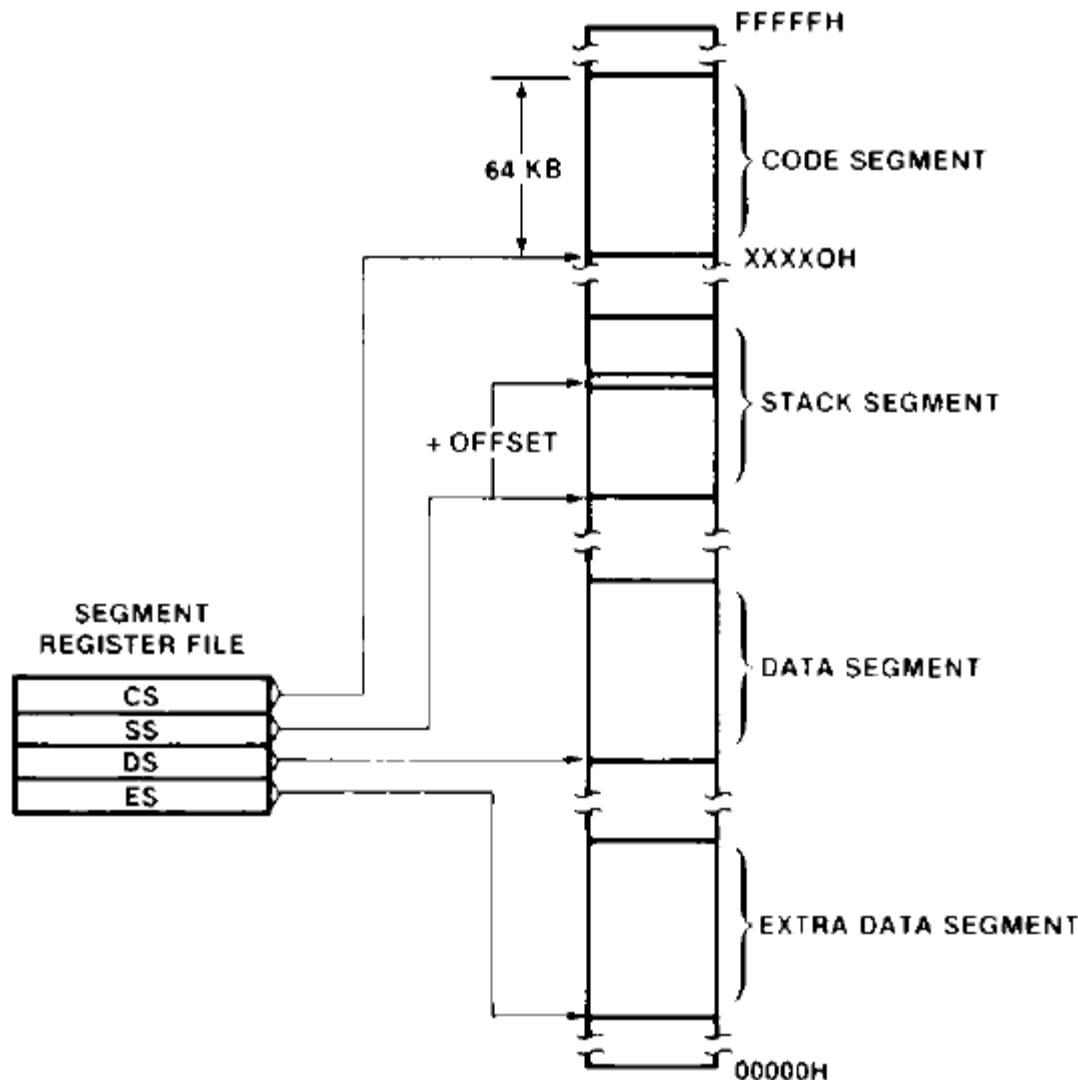
- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ**
- 4.4. Đoạn lệnh và thanh ghi con trả lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ



4.3. Các th.ghi đoạn và phân đoạn BN

- 8088 có 4 thanh ghi đoạn 16 bit, do đó tại một thời điểm, 8088 chỉ làm việc được với 4 đoạn nhớ:
 - CS (Code Segment): quản lý đoạn lệnh
 - SS (Stack Segment): quản lý đoạn ngăn xếp
 - DS (Data Segment): quản lý đoạn dữ liệu
 - ES (Extra Data Segment): quản lý đoạn dữ liệu phụ
- 8088 phát ra một địa chỉ của ngăn nhớ = 20 bit, do đó không gian nhớ của nó là 1 MB ($=2^{20}$ Byte)
- Các thanh ghi bên trong 8088 đều có độ dài là 16 bit.

Các thanh ghi đoạn





Phân đoạn bộ nhớ

- Intel chia không gian nhớ của 8088 thành các đoạn nhớ (segment) có dung lượng $64\text{ KB} = 2^{16}\text{ Byte}$
- Địa chỉ đầu của mỗi đoạn nhớ chia hết cho 16, do đó địa chỉ đầu của một đoạn nào đó sẽ có dạng: xxxx0h (x là chữ số Hexa bất kỳ).
- Để quản lý địa chỉ đầu của một đoạn nhớ chỉ cần lưu trữ 4 số Hexa (16 bit cao), đây gọi là địa chỉ đoạn.
 - VD: nếu địa chỉ đoạn là 1234h thì địa chỉ vật lý của đầu đoạn nhớ đó là 12340h



Phân đoạn bộ nhớ (tiếp)

- Giả sử có một đoạn nhớ xác định (dung lượng tối đa = 64 KB), để xác định 1 byte nhớ cụ thể trong đoạn đó, cần biết khoảng cách (offset – độ lệch) giữa byte nhớ đó so với ngăn nhớ đầu đoạn.
- Địa chỉ logic có dạng:
 Địa chỉ đoạn (16 bit): offset (16 bit)
- Địa chỉ vật lý (20 bit) = địa chỉ đoạn * 10h + offset
- Ví dụ:
 Có địa chỉ logic 1234h:0076h
 ⇒ địa chỉ vật lý = 1234h * 10h + 0076h = 123B6h
- Người lập trình chỉ lập trình với địa chỉ logic, còn việc chuyển sang địa chỉ vật lý là do bộ vi xử lý thực hiện.



Phân đoạn bộ nhớ (tiếp)

- Địa chỉ đoạn: xxxxh
Địa chỉ vật lý đầu đoạn: xxxx0h
Địa chỉ vật lý cuối đoạn: xxxx0h + FFFFh
- Địa chỉ đoạn do các thanh ghi đoạn quản lý.
- Địa chỉ offset do các thanh ghi IP, BX, BP, SP, SI, DI quản lý.
- Với một địa chỉ vật lý, có thể tìm ra nhiều địa chỉ logic khác nhau.
- Ví dụ:
 $00070h = 0000h:0070h = 0001h:0060h = 0002h:0050h \dots$



Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trả lệnh**
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ



4.4. Đoạn lệnh và thanh ghi con trỏ lệnh

- Thanh ghi CS sẽ xác định đoạn lệnh.
- Đoạn lệnh dùng để chứa lệnh của chương trình. Bộ vi xử lý sẽ nhận lần lượt từng lệnh ở đây để giải mã và thực hiện.
- Thanh ghi IP (con trỏ lệnh) chứa địa chỉ offset của lệnh tiếp theo sẽ được nhận vào.
⇒ CS:IP chứa địa chỉ logic của lệnh tiếp theo sẽ được nhận vào.



Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trỏ lệnh
- 4.5. Stack và các thanh ghi BP, SP**
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ



4.5. Stack và các thanh ghi SP, BP

- Stack (ngăn xếp): vùng nhớ tổ chức theo cơ chế LIFO, dùng để cất giữ thông tin và có thể khôi phục lại.
- Chiều từ đáy lên đỉnh của ngăn xếp ngược với chiều tăng của địa chỉ.
- Đoạn Stack được quản lý nhờ thanh ghi SS.
- Thông tin được trao đổi với Stack theo word (16 bit).
- SP chứa địa chỉ offset của ngăn nhớ đỉnh Stack
 - Nếu cất thêm một thông tin vào Stack thì nội dung của SP giảm đi 2
 - Nếu lấy ra một thông tin của Stack thì nội dung của SP tăng lên 2
 - Nếu Stack rỗng thì SP trở vào đáy Stack
 - SS:SP chứa địa chỉ logic của ngăn nhớ đỉnh Stack
- BP là thanh ghi chứa địa chỉ offset của một ngăn nhớ nào đó trong Stack \Rightarrow địa chỉ logic của ngăn nhớ đó là SS:BP



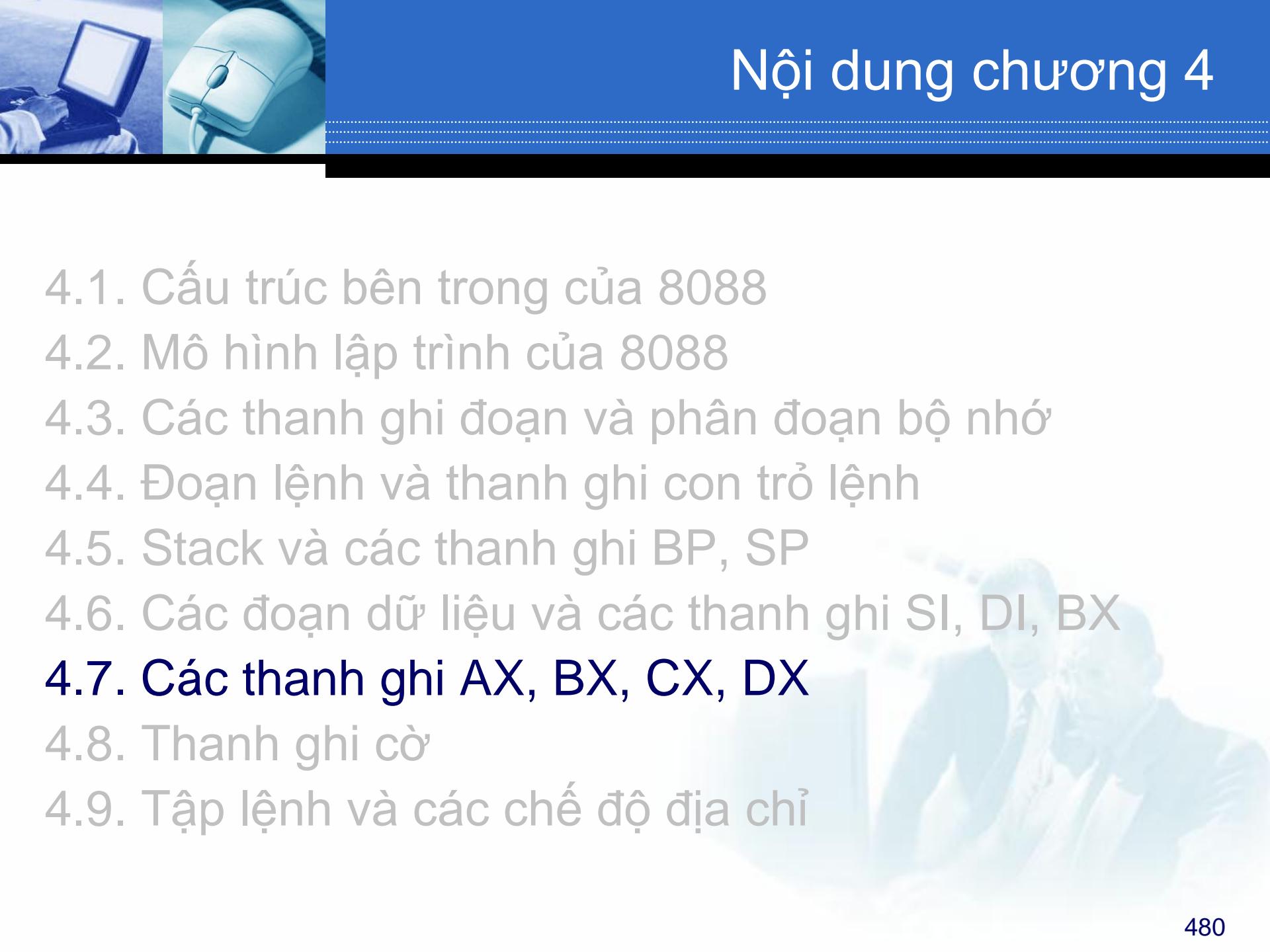
Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trỏ lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX**
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ



4.6. Các đoạn dl và các th.ghi SI, DI, BX

- DS: quản lý một đoạn dữ liệu 64 KB
- ES: quản lý một đoạn dữ liệu phụ 64 KB
- Offset sẽ được xác định bởi nội dung của các thanh ghi SI, DI, BX.
- Sự khác nhau giữa chương trình kiểu EXE và COM:
 - Trong chương trình EXE: CS, DS và SS quản lý 3 đoạn nhớ khác nhau. Nghĩa là : $CS \neq SS \neq DS$.
 - Trong chương trình COM: CS, DS và SS có thể quản lý chung một đoạn nhớ (không lớn hơn 64 KB). Nghĩa là $CS = DS = SS$.



Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trỏ lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX**
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ



4.7. Các thanh ghi AX, BX, CX, DX

- AX, BX, CX, DX là các thanh ghi 16 bit
- AH, AL, BH, BL, CH, CL, DH, DL là các thanh ghi 8 bit
- Chức năng chung: chứa dữ liệu tạm thời
- Chức năng riêng:
 - AX: . Dùng cho lệnh nhân chia theo word
. Dùng cho vào ra theo word
 - AL: . Dùng cho lệnh nhân chia theo byte
. Dùng cho vào ra theo byte
. Dùng cho các lệnh số học với số BCD
 - AH: . Dùng cho các lệnh nhân chia theo byte
 - BX: . Dùng để chứa địa chỉ cơ sở
 - CX: . Dùng để chứa số lần lặp của lệnh LOOP và các lệnh xử lý xâu ký tự
 - CL: . Dùng để chứa số lần dịch của lệnh dịch, lệnh quay
 - DX: . Dùng cho lệnh nhân chia theo word
. Dùng chứa địa chỉ cổng vào ra



Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trỏ lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ**
- 4.9. Tập lệnh và các chế độ địa chỉ



4.8. Thanh ghi cờ



- Bao gồm:
 - Các cờ phép toán: biểu thị trạng thái của kết quả phép toán.
 - Các cờ điều khiển: đặt chế độ làm việc cho bộ vi xử lý.



Các cờ phép toán

- Cờ ZF (Zero - cờ không/cờ rỗng): Được thiết lập (= 1) nếu kết quả phép toán bằng 0 và ngược lại sẽ bị xóa (=0) nếu kết quả phép toán khác 0.
- Cờ SF (Sign - cờ dấu): Được thiết lập nếu kết quả phép toán nhỏ hơn 0 và bị xoá nếu kết quả phép toán lớn hơn hoặc bằng 0.
- Cờ CF (Carry - cờ nhớ): Nếu phép cộng có nhớ ra khỏi bit cao nhất hay phép toán trừ có mượn ra khỏi bit cao nhất thì CF được thiết lập (báo tràn với số nguyên không dấu).
- Cờ OF (Overflow - cờ tràn): Nếu phép toán xảy ra overflow thì OF được thiết lập (báo tràn với số nguyên có dấu).
- Cờ PF (Parity - cờ kiểm tra chẵn lẻ): Nếu tổng số bit 1 của kết quả là chẵn thì cờ PF được thiết lập.
- Cờ AF (Auxiliary - cờ nhớ phụ): Nếu phép cộng có nhớ từ bit 3 sang bit 4 hoặc phép trừ có mượn từ bit 4 sang bit 3 thì cờ AF được thiết lập.



Các cờ điều khiển

- Cờ TF (Trap - cờ bẫy):
 - Nếu TF = 1 thì bộ vi xử lý hoạt động theo chế độ thực hiện từng lệnh (chế độ gõ rối chương trình).
- Cờ IF (Interrupt - cờ ngắt):
 - Nếu IF = 1 thì bộ vi xử lý cho phép ngắt với yêu cầu ngắt đưa đến chân tín hiệu INTR (Interrupt Request) của bộ vi xử lý.
 - Nếu IF = 0 thì cấm ngắt.
- Cờ DF (Director - cờ hướng): chỉ hướng xử lý xâu ký tự.
 - Nếu DF = 0, xử lý từ trái sang phải.
 - Nếu DF = 1, xử lý từ phải sang trái.



Nội dung chương 4

- 4.1. Cấu trúc bên trong của 8088
- 4.2. Mô hình lập trình của 8088
- 4.3. Các thanh ghi đoạn và phân đoạn bộ nhớ
- 4.4. Đoạn lệnh và thanh ghi con trả lệnh
- 4.5. Stack và các thanh ghi BP, SP
- 4.6. Các đoạn dữ liệu và các thanh ghi SI, DI, BX
- 4.7. Các thanh ghi AX, BX, CX, DX
- 4.8. Thanh ghi cờ
- 4.9. Tập lệnh và các chế độ địa chỉ**



4.9. Tập lệnh và các chế độ địa chỉ

▪ **Tập lệnh:**

Tập lệnh của 8088 có khoảng 120 lệnh, chia thành các nhóm như sau:

- Các lệnh chuyển dữ liệu (copy)
- Các lệnh số học
- Các lệnh logic, dịch, quay
- Các lệnh xử lý xâu ký tự (string)
- Các lệnh điều khiển hệ thống
- Các lệnh chuyển điều khiển (rẽ nhánh)
- Các lệnh xử lý đặc biệt
- Các lệnh vào-ra trực tiếp



Các lệnh chuyển dữ liệu

MOV = Move:

Register/Memory to/from Register

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 0 0 0 1 0 d w

mod reg r/m

Immediate to Register/Memory

1 1 0 0 0 1 1 w

mod 0 0 0 r/m

data

data if w = 1

Immediate to Register

1 0 1 1 w reg

data

data if w = 1

Memory to Accumulator

1 0 1 0 0 0 0 w

addr-low

addr-high

Accumulator to Memory

1 0 1 0 0 0 1 w

addr-low

addr-high

Register/Memory to Segment Register

1 0 0 0 1 1 1 0

mod 0 reg r/m

Segment Register to Register/Memory

1 0 0 0 1 1 0 0

mod 0 reg r/m

PUSH = Push:

Register/Memory

1 1 1 1 1 1 1 1

mod 1 1 0 r/m

Register

0 1 0 1 0 reg

Segment Register

0 0 0 reg 1 1 0

POP = Pop:

Register/Memory

1 0 0 0 1 1 1 1

mod 0 0 0 r/m

Register

0 1 0 1 1 reg

Segment Register

0 0 0 reg 1 1 1



Các lệnh chuyển dữ liệu (tiếp)

XCHG = Exchange:

Register/Memory with Register

1000011 w	mod reg r/m
-----------	-------------

Register with Accumulator

10010 reg

IN = Input from:

Fixed Port

1110010 w	port
-----------	------

Variable Port

1110110 w

OUT = Output to:

Fixed Port

1110011 w	port
-----------	------

Variable Port

1110111 w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101	mod reg r/m
----------	-------------

LDS = Load Pointer to DS

11000101	mod reg r/m
----------	-------------

LES = Load Pointer to ES

11000100	mod reg r/m
----------	-------------

LAHF = Load AH with Flags

10011111

SAHF = Store AH into Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101



Các lệnh số học

ARITHMETIC

ADD = Add:

Reg./Memory with Register to Either

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

0 0 0 0 0 0 d w	mod reg r/m		
1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s: w = 01
0 0 0 0 0 1 0 w	data	data if w = 1	

ADC = Add with Carry:

Reg./Memory with Register to Either

0 0 0 1 0 0 d w

mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 s w

mod 0 1 0 r/m

data if s: w = 01

Immediate to Accumulator

0 0 0 1 0 1 0 w

data

data if w = 1

INC = Increment:

Register/Memory

1 1 1 1 1 1 1 w

mod 0 0 0 r/m

Register

0 1 0 0 0 reg

AAA = ASCII Adjust for Add

0 0 1 1 0 1 1 1

BAA = Decimal Adjust for Add

0 0 1 0 0 1 1 1



Các lệnh số học (tiếp)

SUB = Subtract:

Reg./Memory and Register to Either

001010 dw	mod reg r/m		
100000 sw	mod 101 r/m	data	data if s w = 01
0010110 w	data	data if w = 1	

SSB = Subtract with Borrow

Reg./Memory and Register to Either

000110 dw	mod reg r/m		
100000 sw	mod 011 r/m	data	data if s w = 01
000111 w	data	data if w = 1	

DEC = Decrement:

Register/memory

1111111 w	mod 001 r/m	
01001 reg		
1111011 w	mod 011 r/m	

Register

NEG = Change sign



Các lệnh số học (tiếp)

CMP = Compare:

Register/Memory and Register

Immediate with Register/Memory

Immediate with Accumulator

AAS = ASCII Adjust for Subtract

DAS = Decimal Adjust for Subtract

MUL = Multiply (Unsigned)

IMUL = Integer Multiply (Signed)

AAM = ASCII Adjust for Multiply

DIV = Divide (Unsigned)

IDIV = Integer Divide (Signed)

AAD = ASCII Adjust for Divide

CBW = Convert Byte to Word

CWD = Convert Word to Double Word

001110 dw	mod reg r/m		
100000 sw	mod 111 r/m	data	data if s w = 01
0011110 w	data	data if w = 1	
00111111			
00101111			
1111011 w	mod 100 r/m		
1111011 w	mod 101 r/m		
11010100	00001010		
1111011 w	mod 110 r/m		
1111011 w	mod 111 r/m		
11010101	00001010		
10011000			
10011001			



Các lệnh logic

LOGIC

NOT = Invert

SHL/SAL = Shift Logical/Arithmetic Left

SHR = Shift Logical Right

SAR = Shift Arithmetic Right

ROL = Rotate Left

ROR = Rotate Right

RCL = Rotate Through Carry Flag Left

RCR = Rotate Through Carry Right

7 6 5 4 3 2 1 0

1 1 1 1 0 1 1 w

7 6 5 4 3 2 1 0

mod 0 1 0 r/m

1 1 0 1 0 0 v w

mod 1 0 0 r/m

1 1 0 1 0 0 v w

mod 1 0 1 r/m

1 1 0 1 0 0 v w

mod 1 1 1 r/m

1 1 0 1 0 0 v w

mod 0 0 0 r/m

1 1 0 1 0 0 v w

mod 0 0 1 r/m

1 1 0 1 0 0 v w

mod 0 1 0 r/m

1 1 0 1 0 0 v w

mod 0 1 1 r/m



Các lệnh logic (tiếp)

AND = And:

Reg./Memory and Register to Either

001000 dw	mod reg r/m		
1000000 w	mod 100 r/m	data	data if w = 1
0010010 w	data	data if w = 1	

TEST = And Function to Flags, No Result:

Register/Memory and Register

1000010 w	mod reg r/m		
1111011 w	mod 000 r/m	data	data if w = 1
1010100 w	data	data if w = 1	

OR = Or:

Reg./Memory and Register to Either

000010 dw	mod reg r/m		
1000000 w	mod 001 r/m	data	data if w = 1
0000110 w	data	data if w = 1	

XOR = Exclusive or:

Reg./Memory and Register to Either

001100 dw	mod reg r/m		
1000000 w	mod 110 r/m	data	data if w = 1
0011010 w	data	data if w = 1	



Các lệnh xử lý chuỗi

STRING MANIPULATION

REP = Repeat

MOVS = Move Byte/Word

CMPS = Compare Byte/Word

SCAS = Scan Byte/Word

LODS = Load Byte/Wd to AL/AX

STOS = Stor Byte/Wd from AL/A

1 1 1 1 0 0 1 z

1 0 1 0 0 1 0 w

1 0 1 0 0 1 1 w

1 0 1 0 1 1 1 w

1 0 1 0 1 1 0 w

1 0 1 0 1 0 1 w



Các lệnh chuyển điều khiển

CONTROL TRANSFER

CALL = Call:

Direct within Segment

Indirect within Segment

Direct Intersegment

Indirect Intersegment

1 1 1 0 1 0 0	disp-low	disp-high
1 1 1 1 1 1 1	mod 0 1 0 r/m	
1 0 0 1 1 0 1 0	offset-low	offset-high
	seg-low	seg-high
1 1 1 1 1 1 1 1	mod 0 1 1 r/m	



Các lệnh chuyển điều khiển (tiếp)

JMP = Unconditional Jump:

Direct within Segment

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 1 1 0 1 0 0 1

disp-low

disp-high

Direct within Segment-Short

1 1 1 0 1 0 1 1

disp

Indirect within Segment

1 1 1 1 1 1 1 1

mod 1 0 0 r/m

Direct Intersegment

1 1 1 0 1 0 1 0

offset-low

offset-high

seg-low

seg-high

Indirect Intersegment

1 1 1 1 1 1 1 1

mod 1 0 1 r/m

RET = Return from CALL:

Within Segment

1 1 0 0 0 0 1 1

Within Seg Adding Immed to SP

1 1 0 0 0 0 1 0

data-low

data-high

Intersegment

1 1 0 0 1 0 1 1

Intersegment Adding Immediate to SP

1 1 0 0 1 0 1 0

data-low

data-high



Các lệnh chuyển điều khiển (tiếp)

JE/JZ = Jump on Equal/Zero

JL/JNGE = Jump on Less/Not Greater or Equal

JLE/JNG = Jump on Less or Equal/Not Greater

JB/JNAE = Jump on Below/Not Above or Equal

JBE/JNA = Jump on Below or Equal/Not Above

JP/JPE = Jump on Parity/Parity Even

JO = Jump on Overflow

JS = Jump on Sign

JNE/JNZ = Jump on Not Equal/Not Zero

JNL/JGE = Jump on Not Less/Greater or Equal

JNLE/JG = Jump on Not Less or Equal/Greater

JNB/JAE = Jump on Not Below/Above or Equal

JNBE/JA = Jump on Not Below or Equal/Above

JNP/JPO = Jump on Not Par/Par Odd

0 1 1 1 0 1 0 0	disp
0 1 1 1 1 1 0 0	disp
0 1 1 1 1 1 1 0	disp
0 1 1 1 0 0 1 0	disp
0 1 1 1 0 1 1 0	disp
0 1 1 1 1 0 1 0	disp
0 1 1 1 0 0 0 0	disp
0 1 1 1 1 0 0 0	disp
0 1 1 1 0 1 0 1	disp
0 1 1 1 1 1 0 1	disp
0 1 1 1 1 1 1 1	disp
0 1 1 1 0 0 1 1	disp
0 1 1 1 0 1 1 1	disp
0 1 1 1 1 0 1 1	disp



Các lệnh chuyển điều khiển (tiếp)

JNO = Jump on Not Overflow

JNS = Jump on Not Sign

LOOP = Loop CX Times

LOOPZ/LOOPE = Loop While Zero/Equal

LOOPNZ/LOOPNE = Loop While Not
Zero/Equal

JCXZ = Jump on CX Zero

INT = Interrupt

Type Specified

Type 3

INTO = Interrupt on Overflow

IRET = Interrupt Return

0 1 1 1 0 0 0 1	disp
0 1 1 1 1 0 0 1	disp
1 1 1 0 0 0 1 0	disp
1 1 1 0 0 0 0 1	disp
1 1 1 0 0 0 0 0	disp
1 1 1 0 0 0 1 1	disp

1 1 0 0 1 1 0 1	type
1 1 0 0 1 1 0 0	
1 1 0 0 1 1 1 0	
1 1 0 0 1 1 1 1	



Các lệnh điều khiển hệ thống

PROCESSOR CONTROL

CLC = Clear Carry

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 1 1 1 1 0 0 0

CMC = Complement Carry

1 1 1 1 0 1 0 1

STC = Set Carry

1 1 1 1 1 0 0 1

CLD = Clear Direction

1 1 1 1 1 1 0 0

STD = Set Direction

1 1 1 1 1 1 0 1

CLI = Clear Interrupt

1 1 1 1 1 0 1 0

STI = Set Interrupt

1 1 1 1 1 0 1 1

HLT = Halt

1 1 1 1 0 1 0 0

WAIT = Wait

1 0 0 1 1 0 1 1

ESC = Escape (to External Device)

1 1 0 1 1 x x x

mod x x x r/m

LOCK = Bus Lock Prefix

1 1 1 1 0 0 0 0

Chú thích



NOTES:

AL = 8-bit accumulator

AX = 16-bit accumulator

CX = Count register

DS = Data segment

ES = Extra segment

Above/below refers to unsigned value

Greater = more positive;

Less = less positive (more negative) signed values

if d = 1 then "to" reg; if d = 0 then "from" reg

if w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field

if mod = 00 then DISP = 0*, disp-low and disp-high are absent

if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent

if mod = 10 then DISP = disp-high; disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP

if r/m = 001 then EA = (BX) + (DI) + DISP

if r/m = 010 then EA = (BP) + (SI) + DISP

if r/m = 011 then EA = (BP) + (DI) + DISP

if r/m = 100 then EA = (SI) + DISP

if r/m = 101 then EA = (DI) + DISP

if r/m = 110 then EA = (BP) + DISP*

if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high; disp-low.

if s w = 01 then 16 bits of immediate data form the operand

if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand

if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care

z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)



Tập lệnh và các chế độ địa chỉ (tiếp)

■ Các chế độ địa chỉ (Addressing modes):

- Chế độ địa chỉ là cách xác định toán hạng của lệnh
- Toán hạng gồm: toán hạng nguồn và toán hạng đích
 - Họ Intel x86: nếu trong lệnh có 2 toán hạng thì toán hạng đích được viết ở bên trái.
- Toán hạng có thể là:
 - Hằng số (được cho ngay trong lệnh)
 - Nội dung của thanh ghi (trong lệnh cần cho biết tên thanh ghi)
 - Nội dung của ngăn nhớ
 - Nội dung của cổng vào-ra





Các chế độ địa chỉ

- Chế độ địa chỉ tức thì:

- Toán hạng là một giá trị hằng số nằm ngay trong lệnh.
 - Ví dụ:

MOV CX, 5 ; nạp giá trị 5 vào thanh ghi CX

MOV 5, CX ; chú ý: không tồn tại lệnh này !!!

- Chế độ địa chỉ thanh ghi:

- Toán hạng là nội dung của 1 thanh ghi mà tên thanh ghi được cho biết ở trong lệnh.
 - Ví dụ:

MOV AX, BX ; chuyển nội dung của BX vào AX



Các chế độ địa chỉ (tiếp)

- Chế độ địa chỉ trực tiếp:

- Toán hạng là nội dung của ngăn nhớ mà địa chỉ của ngăn nhớ đó được cho ở trong lệnh.

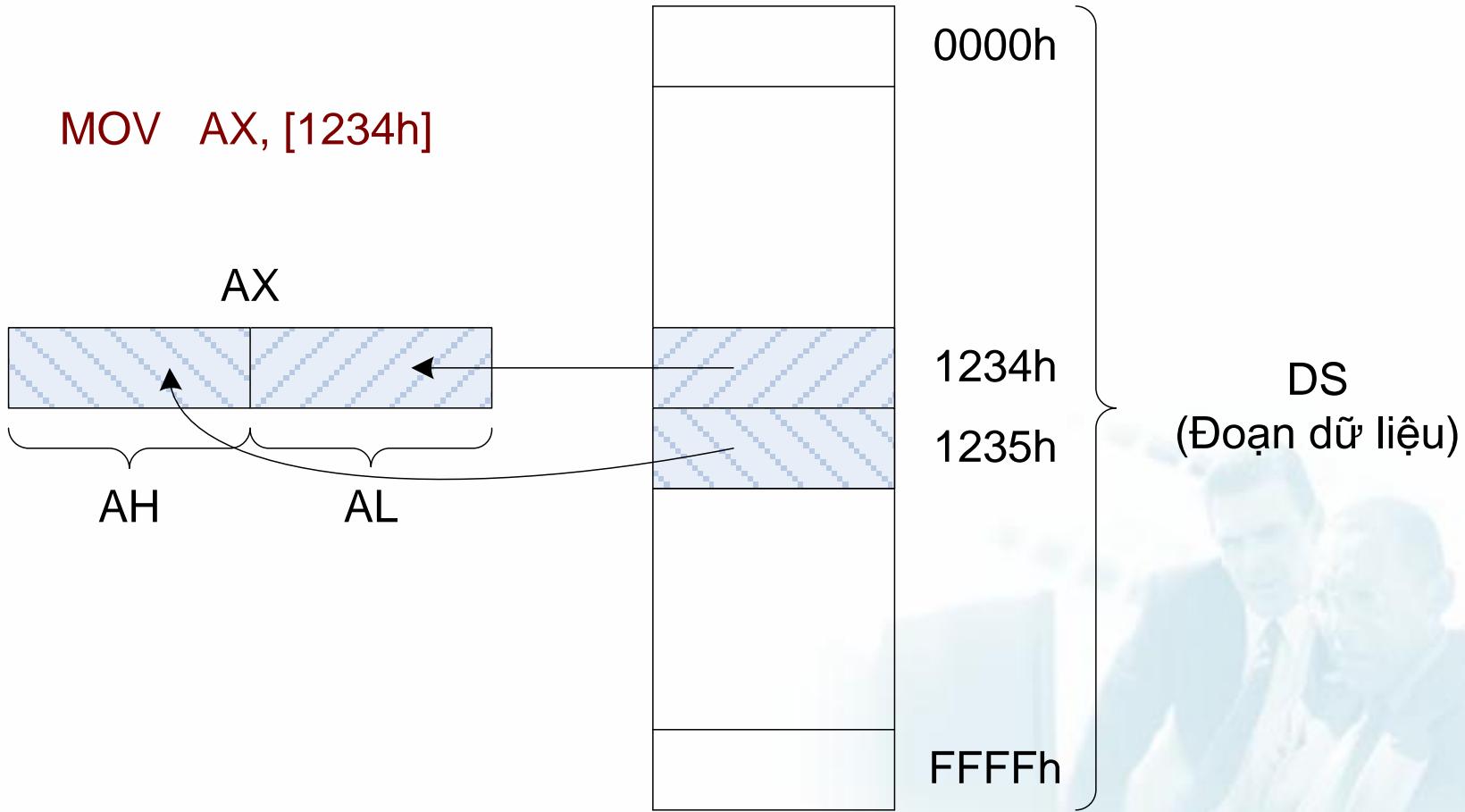
- Ví dụ:

MOV AX, [1234h] ; khác với **MOV AX, ES:[1234h]**

- 1234h là địa chỉ offset của ngăn nhớ
 - Nếu không chỉ định địa chỉ đoạn thì ngầm định thanh ghi đoạn tương ứng là DS
 - Lệnh này chuyển 1 word nằm trong bộ nhớ bắt đầu từ địa chỉ DS:1234h vào thanh ghi AX



Minh họa chế độ địa chỉ trực tiếp





Các chế độ địa chỉ (tiếp)

- Chế độ địa chỉ gián tiếp qua thanh ghi:
 - Toán hạng là nội dung của ngăn nhớ có địa chỉ offset nằm trong 1 trong các thanh ghi sau: BX, SI, DI.
 - Chú ý: nếu không chỉ định địa chỉ đoạn thì ngầm định thanh ghi đoạn tương ứng là DS.
 - Ví dụ:

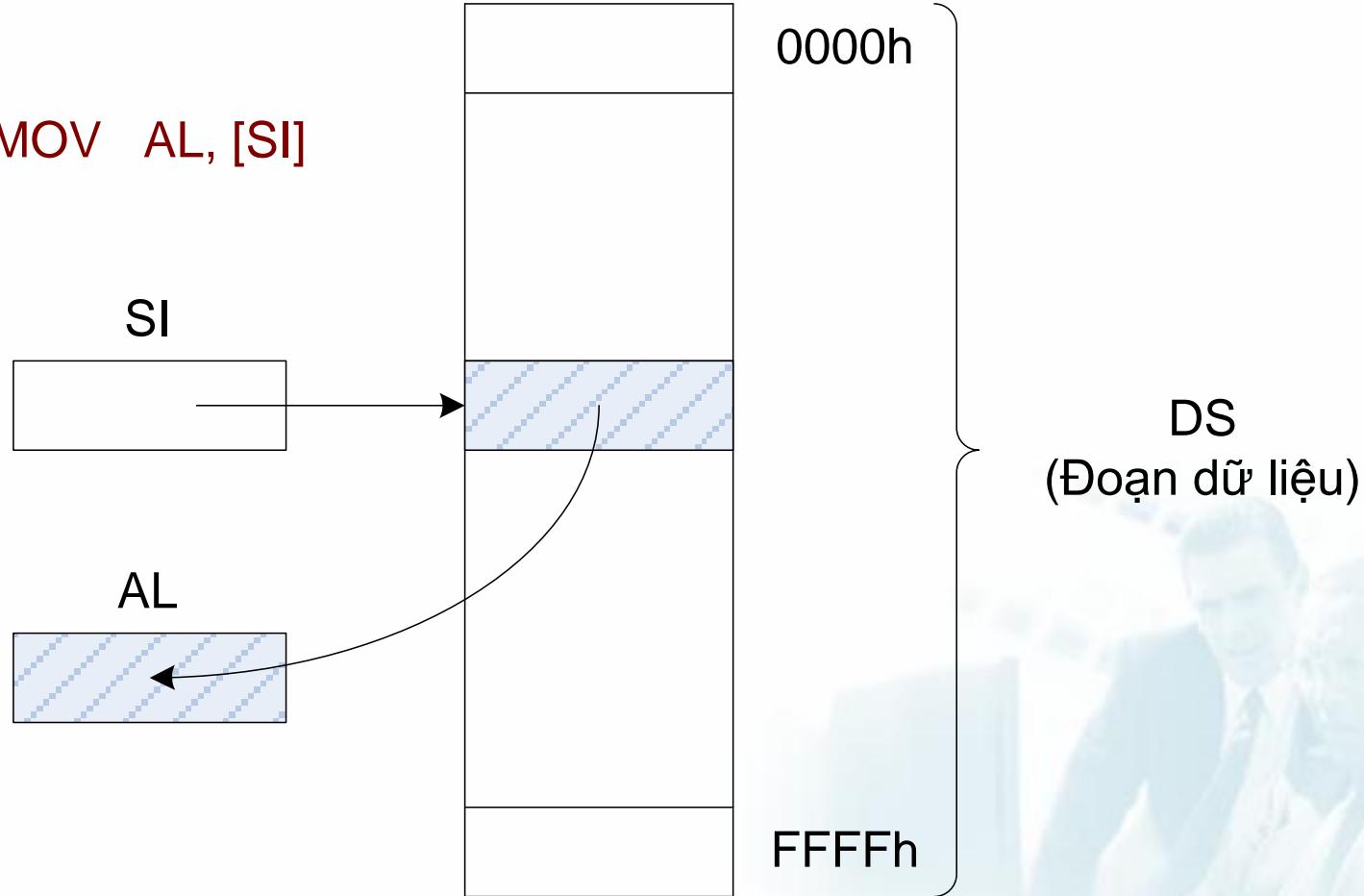
MOV AL, [SI] ; tương đương MOV AL, DS:[SI]

Lệnh này chuyển 1 byte nhớ ở địa chỉ DS:SI vào thanh ghi AL



Minh họa

MOV AL, [SI]





Các chế độ địa chỉ (tiếp)

- Chế độ địa chỉ cơ sở:

- Toán hạng là nội dung của ngăn nhớ có địa chỉ offset bằng tổng nội dung của một thanh ghi cơ sở (BX hoặc BP) + hằng số.
- Nếu không chỉ định thanh ghi đoạn thì ngầm định thanh ghi đoạn đó là:
 - DS nếu thanh ghi cơ sở là BX
 - SS nếu thanh ghi cơ sở là BP

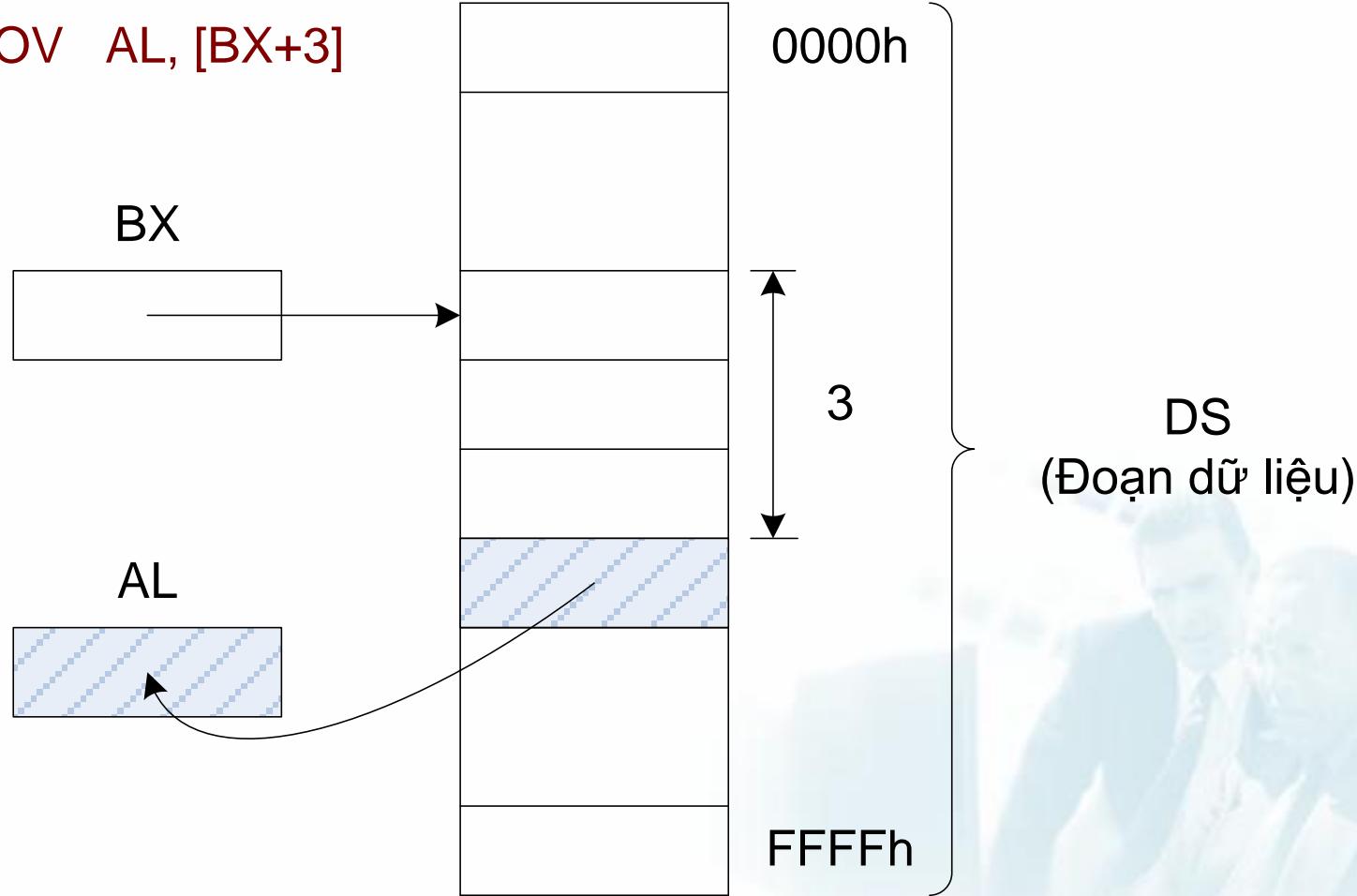
- Ví dụ:

MOV AL, [BX+3] ; tương đương **MOV AL, [BX]+3**
; tương đương **MOV AL, 3[BX]**



Minh họa

MOV AL, [BX+3]





Các chế độ địa chỉ (tiếp)

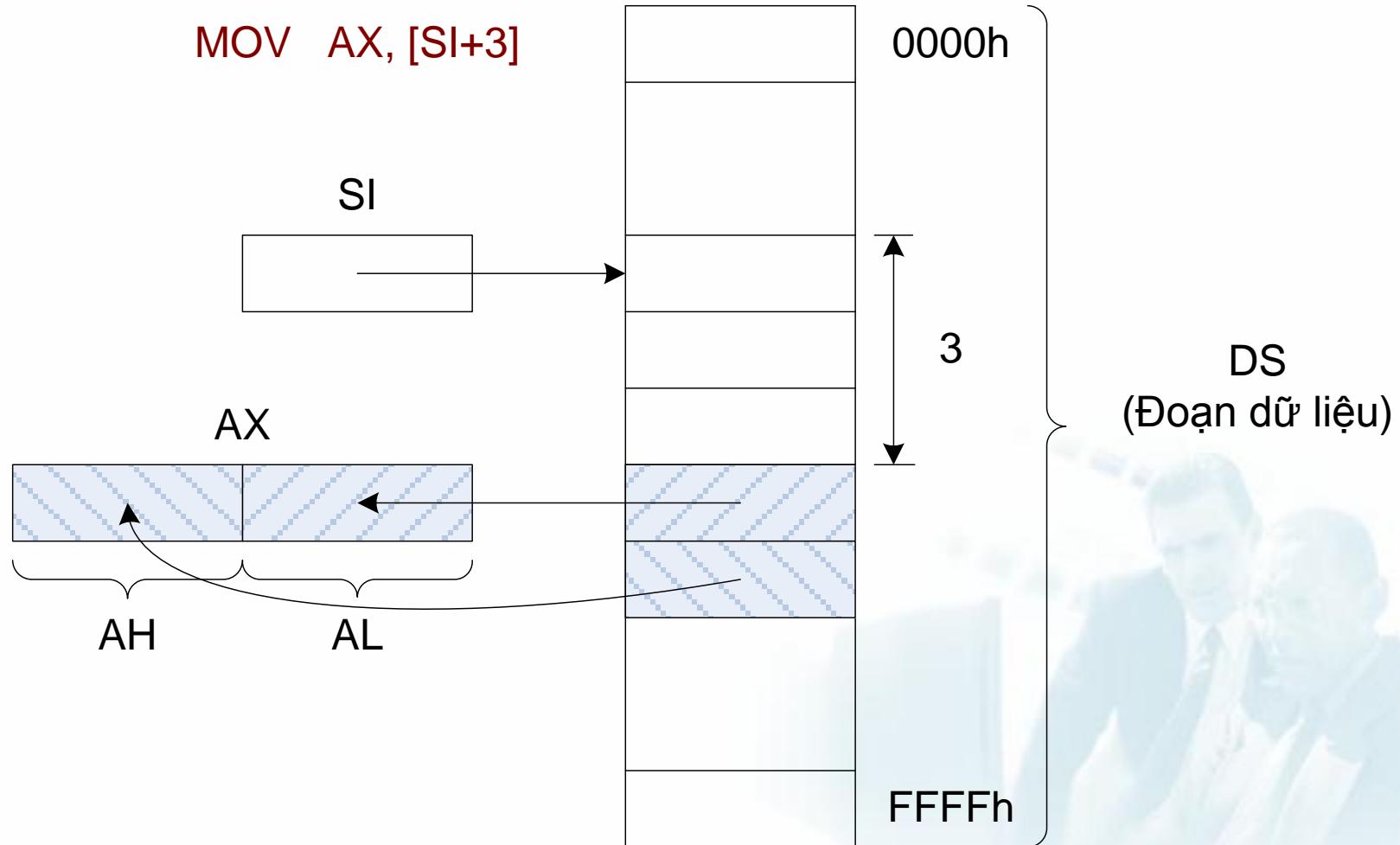
■ Chế độ địa chỉ chỉ số:

- Toán hạng là nội dung của ngăn nhớ có địa chỉ offset bằng tổng nội dung của một thanh ghi chỉ số (SI hoặc DI) + hằng số.
- Nếu không chỉ định thanh ghi đoạn thì ngầm định thanh ghi đoạn đó là DS.
- Ví dụ:

MOV AX, [SI+3] ; tương đương MOV AX, [SI]+3
; tương đương MOV AX, 3+[SI]
; tương đương MOV AX, 3[SI]



Minh họa





Các chế độ địa chỉ (tiếp)

- Chế độ địa chỉ chỉ số cơ sở:

- Toán hạng là ngăn nhớ có địa chỉ offset bằng tổng của nội dung một thanh ghi cơ sở (BX, BP) với một thanh ghi chỉ số (SI, DI) và một hằng số.
- Ví dụ:

MOV AL, [BX][SI]+4 ; \Leftrightarrow MOV AL, [BX+SI+4]



Tổng kết chế độ địa chỉ

Chế độ địa chỉ	Toán hạng	Thanh ghi đoạn ngầm định
Thanh ghi	Reg	-
Tức thì	Data	-
Trực tiếp	[Offset]	DS
Gián tiếp	[DX] [SI] [DI] [BX]	DS
Tương đối cơ sở	[BX] + disp [BP] + disp	DS SS
Tương đối chỉ số	[SI hoặc DI] + disp	DS
Tương đối chỉ số cơ sở	[BX][SI hoặc DI] + disp [BP][SI hoặc DI] + disp	DS SS



Các cặp thanh ghi đoạn:lệch ngầm định

Thanh ghi đoạn	CS	<u>DS</u>	<u>ES</u>	SS
Thanh ghi lệch	IP	BX, <u>SI</u> , DI	<u>DI</u>	SP, BP

Ghi chú: các cặp DS:SI và ES:DI dùng với các lệnh thao tác chuỗi



HẾT CHƯƠNG 4



Chương 5

LẬP TRÌNH HỢP NGỮ VỚI 8088

5.1. Mở đầu về lập trình hợp ngũ

5.2. Các cấu trúc lập trình với hợp ngũ

5.3. Các lệnh logic, lệnh dịch và lệnh quay

5.4. Ngăn xếp và thủ tục

5.5. Các lệnh nhân, chia

5.6. Các lệnh thao tác chuỗi

5.7. Một số ví dụ



5.1. Mở đầu về lập trình hợp ngữ

1. Các loại ngôn ngữ lập trình
2. Cú pháp của hợp ngữ
3. Dữ liệu của chương trình
4. Khai báo biến
5. Khai báo hằng
6. Một số lệnh cơ bản
7. Cấu trúc chương trình
8. Chương trình EXE và COM
9. Vào-ra đơn giản
10. Các ví dụ
11. Dịch và chạy chương trình



1. Các loại ngôn ngữ lập trình

- **Ngôn ngữ máy:**
 - Chỉ được biểu diễn bằng số nhị phân.
 - Bộ vi xử lý chỉ hiểu được các chương trình mã máy.
 - Con người rất khó khăn để tạo lập hay đọc hiểu chương trình ngôn ngữ máy.
- **Hợp ngữ (Assembly Language):**
 - Là ngôn ngữ lập trình bậc thấp (gần ngôn ngữ máy nhất).
 - Được xây dựng trên cơ sở ký hiệu tập lệnh của bộ vi xử lý tương ứng.
 - Phụ thuộc hoàn toàn vào bộ vi xử lý cụ thể.
- **Ngôn ngữ lập trình bậc cao:**
 - Gần với ngôn ngữ tự nhiên hơn.
 - Được xây dựng độc lập với cấu trúc của máy tính.

- **Ưu điểm:**

- Can thiệp sâu vào cấu trúc hệ thống.
- Hiểu sâu hơn về hệ thống.
- Chương trình mã máy tương ứng sẽ ngắn hơn, thường nhanh hơn và tốn ít bộ nhớ hơn.

- **Nhược điểm:**

- Khó học vì gần với mã máy.
- Chương trình nguồn dài, không thích hợp để xây dựng những chương trình lớn.

⇒ Kết hợp ngôn ngữ lập trình bậc cao với hợp ngữ.



Chương trình dịch hợp ngữ

- Được gọi là ASSEMBLER
- Một số chương trình dịch hợp ngữ cho IBM-PC:
 - MASM – Microsoft Macro Assembler:
 - Các tệp: MASM.EXE, LINK.EXE, EXE2BIN.EXE ...
 - TASM – Turbo Assembler:
 - Các tệp: TASM.EXE, TLINK.EXE ...



Các bước lập trình

- Bước 1: Phát biểu bài toán
- Bước 2: Xây dựng thuật giải
- Bước 3: Viết mã chương trình
- Bước 4: Dịch và sửa lỗi cú pháp
- Bước 5: Chạy thử và hiệu chỉnh chương trình



Các cấu trúc lập trình cơ bản

- Cấu trúc tuần tự
- Cấu trúc rẽ nhánh
- Cấu trúc lặp





2. Cú pháp của hợp ngữ

- Chương trình hợp ngữ gồm các dòng lệnh, mỗi lệnh viết trên một dòng, mỗi dòng có thể là:
 - Lệnh của bộ vi xử lý (instruction)
 - Chỉ dẫn của chương trình dịch ASSEMBLER
- Các lệnh hợp ngữ không phân biệt chữ hoa, chữ thường.
- Khi dịch thành mã máy thì chỉ có các lệnh của bộ vi xử lý mới được dịch.
- Cấu trúc của một dòng lệnh :

Tên (Name)	Thao tác Operation	Toán hạng Operand	Chú thích Comment)
-----------------	-----------------------	----------------------	------------------------

- Giữa các trường phải có ít nhất một dấu cách (hoặc TAB)
- Ví dụ:

MAIN	PROC	
BAT_DAU:	MOV CX, 50	; khai tao bo dem



Ý nghĩa các trường trong lệnh

■ Trường tên:

- Sử dụng cho: nhãn lệnh, tên thủ tục, tên biến
- Quy ước đặt tên: dài từ 1 đến 31 ký tự, cho phép sử dụng:
 - Chữ cái (không phân biệt chữ hoa và chữ thường)
 - Chữ số (không được dùng làm ký tự đầu tiên)
 - Các ký tự khác: ?, @, \$, %, . (dấu . chỉ được dùng khi nó là ký tự đầu tiên).



Ý nghĩa các trường trong lệnh (tiếp)

- Trường thao tác:

- Nếu là lệnh của vi xử lý thì đó chính là mã lệnh (MOV, CALL, ADD,...).
- Nếu là chỉ dẫn thì đó là lệnh giả của chương trình dịch (Pseudo-op).



Ý nghĩa các trường trong lệnh (tiếp)

- Trường toán hạng:

- Đối với lệnh thì toán hạng xác định dữ liệu bị tác động bởi mã lệnh.
- Một lệnh có thể có 0, 1, 2 toán hạng.
- Ví dụ:
 - MOV CX,5 ; 2 toán hạng
 - INC AX ; 1 toán hạng
 - NOP ; 0 toán hạng
- Đối với lệnh giả thì toán hạng cho thêm thông tin cho lệnh giả đó.

- Trường chú thích:

- Bắt đầu bằng dấu ";" theo sau đó là lời giải thích.



3. Dữ liệu của chương trình

- Hợp ngữ cho phép biểu diễn dưới dạng:
 - Số nhị phân: 1011b, 1011B, ...
 - Số thập phân: 35, 35d, 35D, ...
 - Số Hexa: 4Ah, 0ABCDh, 0FFFFH, ...
 - Kí tự: "A", 'HELLO', "Bach Khoa", ...
- Tất cả các kiểu dữ liệu trên sau đó đều được trình dịch Assembler dịch ra mã nhị phân.
- Mỗi kí tự được dịch thành mã ASCII tương ứng
 - Chương trình không phân biệt 'A' với 41h hay 65



Các chỉ thị giả định số liệu

Chỉ thị giả	Biểu diễn
DB	Định nghĩa byte
DW	Định nghĩa word (2 byte)
DD	Định nghĩa double word (4 byte)
DQ	Định nghĩa quadword (8 byte liên tiếp)
DT	Định nghĩa tenbyte (10 byte liên tiếp)



4. Khai báo biến

- Biến Byte:

- Khai báo:

Ten_bien	DB	Gia_tri_khoi_dau
----------	----	------------------

Ten_bien	DB	?
----------	----	---

- Ví dụ:

Age	DB	25	; Khởi tạo giá trị ban đầu Age = 25
-----	----	----	-------------------------------------

Alpha	DB	?	; Ban đầu Alpha không xác định
-------	----	---	--------------------------------

- Khoảng xác định của biến Byte:

- Số không dấu: [0, 255]

- Số có dấu: [-128, 127]



Khai báo biến (tiếp)

■ Biến Word:

- Khai báo:

Ten_bien DW Gia_tri_khoi_dau

Ten_bien DW ?

- Ví dụ:

Test DW -5 ; -5 = 1111111111111011b

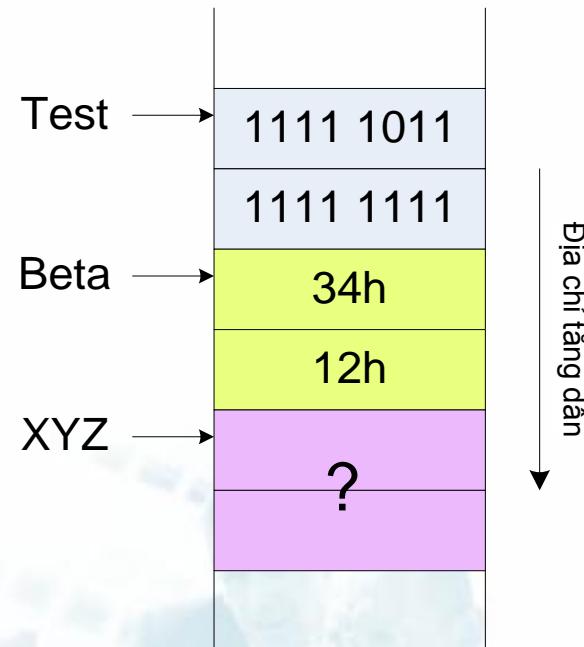
Beta DW 1234h ; 1234h = 0001001000110100b

XYZ DW ?

- Khoảng xác định của biến Word:

- Số không dấu: [0, 65535]

- Số có dấu: [-32768, 32767]





Khai báo biến (tiếp)

■ Biến mảng:

- Mảng Byte:

MangB DB 10h, 20h, 30h, 40h

Buffer DB 100 dup (?)

- Mảng Word:

MangW DW -12, 127, 0A48Bh

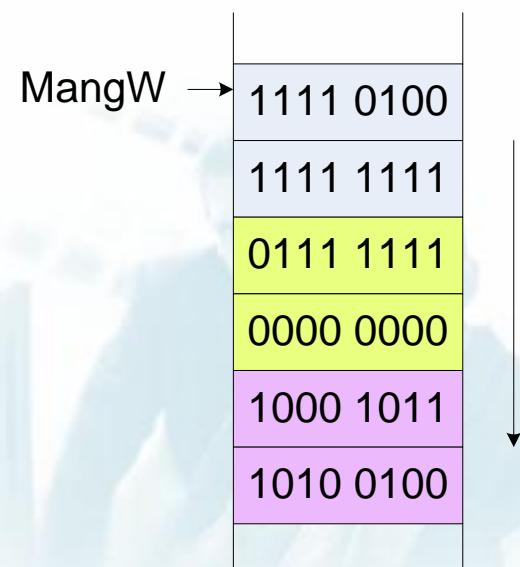
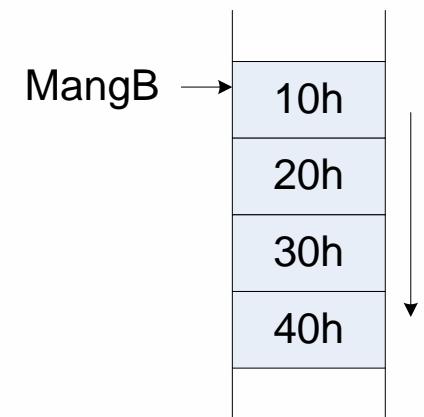
- Mảng kí tự:

- Thực chất là mảng Byte

- Ví dụ: 2 cách viết sau là tương đương

M DB 'ABC'

M DB 41h, 42h, 43h





5. Khai báo hằng

- Cú pháp:

Ten_hang	EQU Gia_tri
----------	-------------

- Ví dụ:

TenTruong	EQU 'BACH KHOA'
-----------	-----------------

CR	EQU 13
----	--------

LF	EQU 10
----	--------

...

ThongBao	DB 'DAI HOC', CR, LF, TenTruong
----------	---------------------------------

DoDaiChuoi	EQU \$ - offset ThongBao
------------	--------------------------

- Hằng không được cấp phát ngăn nhớ



6. Một số lệnh cơ bản

- **Lệnh MOV (Move):** **MOV** **đích, nguồn**
 - Copy dữ liệu từ toán hạng nguồn sang toán hạng đích
 - Kích thước của 2 toán hạng phải giống nhau

Ví dụ:

MOV AX, BX
MOV AL, 'A'
MOV BH, 120

; MOV DS, 0A000h ; SAI
MOV AX, 0A000h
MOV DS, AX

; MOV Bien_2, Bien_1; SAI
MOV AL, Bien_1
MOV Bien_2, AL

Nguồn \ Đích	Thanh ghi chung	Thanh ghi đoạn	Ngăn nhớ
Thanh ghi chung	Có	Có	Có
Thanh ghi đoạn	Có	Có	Có
Ngăn nhớ	Có	Có	Không
Hằng	Có	Không	Có



Một số lệnh cơ bản (tiếp)

- Lệnh XCHG (Exchange): XCHG đích, nguồn
 - Hoán đổi nội dung 2 toán hạng cho nhau
 - Kích thước của 2 toán hạng phải giống nhau

Ví dụ:

XCHG AX, BX

XCHG AH, Byte_1

XCHG Word_1, BX

; XCHG Word_1, Word_2 ; SAI

MOV AX, Word_1

MOV BX, Word_2

MOV Word_1, BX

MOV Word_2, AX

Nguồn	Dích	Thanh ghi chung	Ngăn nhớ
Thanh ghi chung	Có	Có	
Ngăn nhớ	Có		Không



Các lệnh ADD và SUB

- Cú pháp:

ADD đích, nguồn ; đích \leftarrow đích + nguồn

SUB đích, nguồn ; đích \leftarrow đích - nguồn

Ví dụ:

MOV AX, 50

MOV BX, 30

ADD BX, 10 ; BX = 40

SUB AX, BX ; AX = 10

; ADD Byte_1, Byte_2 ; SAI

MOV AL, Byte_1

ADD AL, Byte_2

MOV Byte_1, AL

Nguồn \ Đích	Thanh ghi chung	Ngăn nhớ
Thanh ghi chung	Có	Có
Ngăn nhớ	Có	Không
Hằng	Có	Có



Các lệnh INC, DEC và NEG

- Cú pháp:

INC đích ; đích \leftarrow đích + 1

DEC đích ; đích \leftarrow đích – 1

NEG đích ; đích \leftarrow - đích (lấy bù 2 của đích)

Toán hạng đích là thanh ghi hoặc ngăn nhớ

- Ví dụ:

MOV AX, 20 ; AX = 20

INC AX ; AX = 21 = 0000000000010101b

NEG AX ; AX = 1111111111101011b

DEC AX ; AX = FFEAh

Giả sử A và B là các biến kiểu Word, hãy thực hiện các phép gán sau đây bằng hợp ngữ:

1. $A := B$
2. $A := 10 - A;$
3. $A := B - A * 2;$



7. Cấu trúc chương trình

- Chương trình mã máy khi được thực thi sẽ chiếm 3 vùng nhớ cơ bản trong bộ nhớ chính:
 - Vùng nhớ lệnh (Code)
 - Vùng dữ liệu (Data)
 - Vùng ngăn xếp (Stack)
- Chương trình hợp ngũ cũng được tổ chức tương tự như vậy.
- Mã lệnh, dữ liệu và ngăn xếp được cấu trúc như các đoạn chương trình.



Các chế độ bộ nhớ

- Kích thước của đoạn mã và dữ liệu trong chương trình được chỉ định bằng cách chỉ ra chế độ bộ nhớ nhờ chỉ thị biên dịch **.MODEL**
- Cú pháp:
.Model Kieu_bo_nho
- Chế độ bộ nhớ thường dùng khi lập trình hợp ngữ là **SMALL**.



Các chế độ bộ nhớ (tiếp)

Kiểu	Mô tả
TINY	Mã lệnh và dữ liệu gói gọn trong một đoạn
SMALL	Mã lệnh trong một đoạn Dữ liệu trong một đoạn
MEDIUM	Mã lệnh chiếm nhiều hơn một đoạn Dữ liệu trong một đoạn
COMPACT	Mã lệnh trong một đoạn Dữ liệu chiếm nhiều hơn một đoạn
LARGE	Mã lệnh chiếm nhiều hơn một đoạn Dữ liệu chiếm nhiều hơn một đoạn Không có mảng nào lớn hơn 64 KB
HUGE	Mã lệnh chiếm nhiều hơn một đoạn Dữ liệu chiếm nhiều hơn một đoạn Các mảng có thể lớn hơn 64 KB



Đoạn dữ liệu (Data Segment)

- Đoạn dữ liệu chứa tất cả các khai báo biến.
- Các khai báo hằng cũng thường để ở đây.
- Để khai báo đoạn dữ liệu ta dùng chỉ thị **.DATA**
- Ví dụ:

.Data

Bien_1 db 10

Bien_2 dw 0FEDCh

TBao db 'Xin chao ban', '\$'

Nam equ 2006



Đoạn ngắn xếp (Stack Segment)

- Cú pháp:

.STACK Kich_thuoc

- Kich_thuoc: là số Byte của Stack (nếu không chỉ định Kich_thuoc thì ngầm định là 1KB)
- Ví dụ:

.Stack 100h





Đoạn mã lệnh (Code Segment)

- Đoạn mã lệnh được khai báo với chỉ thị **.CODE**
- Bên trong đoạn mã, các dòng lệnh được tổ chức dưới dạng 1 chương trình chính và các chương trình con (nếu cần).
- Ví dụ:

.Code

Main Proc

; các lệnh của CT chính

Main EndP





Cấu trúc chương trình thông dụng

.Model Small

.Stack 100h

.Data

; khai báo biến, hằng ở đây

.Code

Main Proc

; các lệnh của chương trình chính ở đây

Main EndP

; các chương trình con khác ở đây

End Main



8. Chương trình EXE và COM

- Có 2 loại chương trình mã máy có thể thực thi được trong DOS, đó là chương trình .EXE và .COM
 - Chương trình EXE:
 - Ở đầu file chương trình có 1 vùng thông tin gọi là Header
 - Khi thực thi, CS, DS và SS trở đến 3 phân đoạn khác nhau
 - Chương trình COM:
 - File chương trình có kích thước nhỏ gọn (< 64KB), chứa cả mã lệnh và dữ liệu
 - Khi thực thi, CS, DS và SS trở đến cùng 1 phân đoạn
- DOS sẽ chọn 1 địa chỉ phân đoạn gọi là PSP (Program Segment Prefix) làm địa chỉ cơ sở để tải chương trình.
- PSP thường có kích thước là 256 Byte (=100h), chứa các thông tin liên quan đến chương trình được thực thi.



Thực thi chương trình EXE

- Nội dung file EXE được tải vào bộ nhớ bắt đầu từ địa chỉ PSP:0100h.
- Sau đó các địa chỉ phân đoạn được tái định vị nhờ các thông tin được đọc từ Header nằm ở đầu file EXE.
- Sau khi chương trình EXE lấy quyền điều khiển:
 - DS và ES trở đến PSP (chứ không phải đoạn dữ liệu)
=> Trong chương trình hợp ngữ ta cần thay đổi DS (và ES) để trở đến đúng đoạn dữ liệu.
 - CS, IP, SS và SP được đặt theo những giá trị chỉ ra trong EXE Header.



Khung chương trình EXE

```
.Model      Small
.Stack      100h
.Data
; khai báo biến và hằng ở đây
.Code
Main Proc
    mov ax, @Data
    mov ds, ax        ; khởi tạo DS trở đến đoạn Data
;    mov es, ax        ; bỏ dấu ; để khởi tạo ES = DS
;
;    thân chương trình

    mov ah, 4Ch        ; hàm thoát về DOS
    int 21h
Main EndP
End Main
```



Thực thi chương trình COM

- Nội dung file COM được tải vào bộ nhớ bắt đầu từ địa chỉ PSP:0100h.
- Sau khi file .COM được nạp vào bộ nhớ:
 - CS, DS, ES và SS được đặt cùng giá trị với PSP
 - SP trở đến cuối đoạn PSP (thường thì SP = 0FFEh)
 - IP được đặt là 100h



Khung chương trình COM

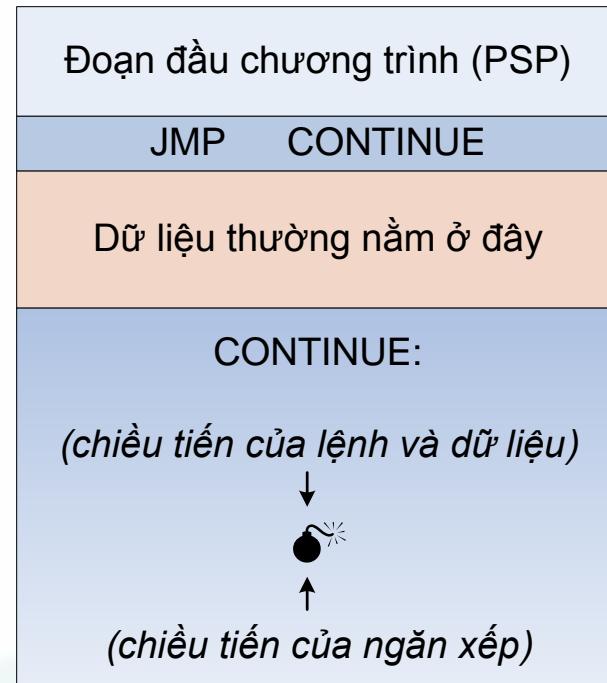
```
.Model      Tiny
.Code
    Org     100h
Start:
    jmp    Continue
;   khai báo dữ liệu ở đây
Continue:
Main Proc
;   thân chương trình
    int    20h    ; về DOS
Main EndP
End    Start
```

Offset

0000h

0100h

FFFFh





9. Vào-ra đơn giản

- CPU có thể trao đổi dữ liệu với các thiết bị ngoại qua các cổng vào-ra nhờ các lệnh IN và OUT.
- Cách vào-ra đơn giản hơn là dùng các dịch vụ ngắt có sẵn của BIOS hoặc DOS.
- Ta thường cần thực hiện các thao tác trao đổi dữ liệu với bàn phím và màn hình ⇒ dùng hàm DOS.
- Lệnh INT (Interrupt): **INT N**
 - Là lệnh gọi CTC phục vụ ngắt số hiệu N (N từ 0 ÷ 255)
 - Dịch vụ ngắt số 21h chứa nhiều hàm tiện ích của DOS.



Lệnh nạp địa chỉ hiệu dụng

- Lệnh LEA (Load Effective Address):

LEA thanh_ghi_chung, ngan_nho

- Lấy địa chỉ offset của ngăn nhớ nạp vào thanh ghi
- Ví dụ:

LEA DX, Thong_Bao

MOV DX, offset Thong_Bao ; lệnh cùng chức năng



Một số hàm vào-ra của DOS

- Khi gọi dịch vụ ngắt của DOS bằng lệnh Int 21h thì AH chứa số hiệu dịch vụ hàm.
- Hàm 01h (chờ người sử dụng vào 1 phím)
 - Vào:
 - AH = 01h
 - Ra:
 - AL = mã ASCII nếu 1 phím kí tự được nhấn
= 0 nếu 1 phím điều khiển hay chức năng được nhấn
 - Ví dụ:

```
MOV AH, 1
INT 21h
```



Một số hàm vào-ra của DOS (tiếp)

- Hàm 02h (hiện 1 kí tự hay điều khiển)
 - Vào:
 - AH = 02h
 - DL = mã ASCII của kí tự hiển thị hay điều khiển
 - Ra:
 - AL = mã ASCII của kí tự hiển thị hay điều khiển
 - Ví dụ:
 - MOV AH, 2
 - MOV DL, 'A' ; viết ra kí tự 'A'
 - INT 21h
 - MOV DL, 10 ; điều khiển con trỏ xuống dòng
 - INT 21h
 - MOV DL, 13 ; điều khiển con trỏ về đầu dòng
 - INT 21h



Một số hàm vào-ra của DOS (tiếp)

- Hàm 09h (hiện 1 chuỗi kí tự)

- Vào:

- AH = 09h
 - DS:DX = địa chỉ của chuỗi kí tự có kí tự kết thúc là '\$'

- Ra: không

- Ví dụ:

ThongBao DB 'Chao cac ban\$'

; giả sử DS = địa chỉ đoạn của ThongBao

MOV AH, 9

LEA DX, ThongBao ; hoặc MOV DX, OFFSET ThongBao

INT 21h



10. Các ví dụ

- Ví dụ 1: Chương trình "Hello World" bằng hợp ngữ.
- Ví dụ 2: Lập trình thực hiện các công việc sau:
 - Hiển thị thông báo : 'Hãy gõ vào một chữ cái thường: '
 - Vào chữ cái thường
 - Xuống dòng, về đầu dòng
 - Hiển thị thông báo : 'Chữ cái hoa tương ứng là: '
 - Hiển thị chữ cái hoa tương ứng
 - Thoát về DOS.



Ví dụ 1

```
.Model Small
.Stack 100h
.Data
    TBao    db      'Hello World$'          ; kết thúc bằng '$'
.Code
Main  Proc
    mov     ax, @Data
    mov     ds, ax          ; DS trỏ đến đoạn Data
HiemTB:
    mov     ah, 9            ; hàm hiện chuỗi
    lea     dx, TBao        ; DS:DX ⇒ chuỗi TBao
    int     21h             ; gọi hàm
Thoat:
    mov     ah, 4Ch          ; hàm thoát về DOS
    int     21h
Main  EndP
End    Main
```



Ví dụ 2

```
.Model Small
.Stack 100
.Data
    TB1 db 'Hay go vao mot chu cai thuong: $'
    TB2 db 'Chu cai hoa tuong ung la: $'
.Code
Main Proc
    mov ax, @Data
    mov ds, ax      ; DS trỏ đến đoạn Data
    mov ah, 9       ; hàm hiển thị chuỗi
    lea dx, TB1     ; DS:DX ⇒ chuỗi TB1
    int 21h
    mov ah, 1       ; hàm nhập kí tự
    int 21h
    mov bl, al      ; lưu kí tự vào BL
```

```
    mov ah, 2      ; hiện kí tự
    mov dl, 10     ; LF
    int 21h
    mov dl, 13     ; CR
    int 21h
    mov ah, 9      ; hiện chuỗi
    lea dx, TB2     ; TB2
    int 21h
    mov ah, 2      ; hiện kí tự
    mov dl, bl
    sub dl, 20h    ; ⇒ chữ HOA
    int 21h
    mov ah, 4Ch     ; về DOS
    int 21h
Main EndP
End Main
```



11. Dịch và chạy chương trình

- Nếu dùng MASM:

- Viết chương trình nguồn, ghi ra file .ASM (chẳng hạn là BAITAP.ASM)
- Dịch mã nguồn: **MASM BAITAP.ASM**
 - Nếu không có lỗi thì ta có file BAITAP.OBJ
 - Nếu có lỗi thì xem thông báo lỗi và đến dòng xuất hiện lỗi để sửa
- Liên kết: **LINK BAITAP.OBJ**
 - Nếu không có lỗi thì ta có file BAITAP.EXE
- Nếu mã nguồn viết theo dạng .COM thì cần chuyển từ file .EXE sang dạng COM bằng lệnh:
EXE2BIN BAITAP.EXE BAITAP.COM
- Thực hiện file BAITAP.EXE (hoặc BAITAP.COM)



Dịch và chạy chương trình (tiếp)

- Nếu dùng TASM:

- Viết chương trình nguồn, ghi ra file .ASM (chẳng hạn là BAITAP.ASM)
- Dịch mã nguồn: **TASM BAITAP.ASM**
 - Nếu không có lỗi thì ta có file BAITAP.OBJ
 - Nếu có lỗi thì xem thông báo lỗi và đến dòng xuất hiện lỗi để sửa
- Liên kết:
 - File EXE: **TLINK BAITAP.OBJ /X**
 - File COM: **TLINK BAITAP.OBJ /T /X**
- Thực hiện file BAITAP.EXE (hoặc BAITAP.COM)



Nội dung file TEXE.BAT

```
@echo off
if not "A%1"=="A" goto ok
echo Syntax: TEXE Filename
goto End
:OK
if not exist %1.asm goto End
echo NPB's Assembly Compiler
echo Compiling : %1.asm
tasm %1.asm
if not exist %1.obj goto End
tlink %1.obj /x
if not ErrorLevel 0 goto End
del %1.obj
:End
```



Sử dụng file TEXE.BAT

- File **TEXE.BAT** chứa các lệnh gọi 2 file **TASM.EXE** và **TLINK.EXE** để dịch 1 chương trình **.ASM** ra dạng file **.EXE**
- Giả sử ta có file chương trình **BAITAP.ASM**
- Gõ lệnh sau: **TEXE BAITAP**
(Chú ý: không gõ "BAITAP.ASM")
- Nếu chương trình viết không có lỗi thì ta sẽ có file **BAITAP.EXE**

- Dữ liệu của 1 chương trình hợp ngữ được khai báo dưới dạng:

DATA SEGMENT

mem1 dw 500

mem2 dw -50

vec2 db 10, 20, -10, -20, -30, -40

DATA ENDS

- Hãy xác định nội dung của AX (Hexa) sau khi thực hiện đoạn lệnh sau:

mov bx, 1

mov ax, SEG vec2

mov es, ax

mov ax, es:[bx]

a) 01F4

b) 0A14

c) F4FF

d) 14F6

e) CE01



Bài tập 1 (tiếp)

	Offset
mem1 →	F4h 0
	01h 1
mem2 →	CEh 2
	FFh 3
vec2 →	0Ah 4
	14h 5
	F6h 6
	ECh 7
	E2h 8
	D8h 9

- Dữ liệu của 1 chương trình hợp ngữ được khai báo dưới dạng:

DATA SEGMENT

mem1 dw 500

mem2 dw -50

vec1 db 1, 2, 3, 4, 8, 7

vec2 db 10, 20, -10, -20, -30, -40

DATA ENDS

- Hãy xác định nội dung của CX (Hexa) sau khi thực hiện đoạn lệnh sau:

mov bx, OFFSET vec1

mov cx, 3[bx]

a) 0304

b) 0408

c) F3F4

d) 0203

e) 0804



Bài tập 2 (tiếp)

	Offset
mem1	F4h 0
	01h 1
mem2	CEh 2
	FFh 3
vec1	01h 4
	02h 5
	03h 6
	04h 7
	08h 8
	07h 9
vec2	14h A
	F6h B
	ECh C
	E2h D
	D8h E

- Lập trình thực hiện các công việc sau:
 - Hiển thị thông báo : 'Hãy gõ vào một chữ số: '
 - Vào một chữ số
 - Xuống dòng, về đầu dòng
 - Hiển thị thông báo : 'Chữ số bù 9 là: '
 - Hiển thị chữ số bù 9 tương ứng
 - Thoát về DOS.



Nội dung chương 5

5.1. Mở đầu về lập trình hợp ngữ

5.2. Các cấu trúc lập trình với hợp ngữ

5.3. Các lệnh logic, lệnh dịch và lệnh quay

5.4. Ngăn xếp và thủ tục

5.5. Các lệnh nhân, chia

5.6. Các lệnh thao tác chuỗi

5.7. Một số ví dụ



5.2. Các cấu trúc lập trình với hợp ngũ

1. Các lệnh liên quan
2. Cấu trúc điều kiện
3. Cấu trúc lặp





1. Các lệnh liên quan

- Các cấu trúc lập trình:
 - Tuần tự
 - Điều kiện
 - Lặp
- Các cấu trúc điều kiện và lặp trong hợp ngữ được tạo bởi phần lớn là các lệnh sau:
 - Lệnh so sánh CMP
 - Lệnh nhảy không điều kiện JMP
 - Các lệnh nhảy có điều kiện
 - Các câu lệnh lặp



Lệnh so sánh CMP

- Cú pháp: **CMP** **đích, gốc**
- Đích và gốc không đồng thời là ngăn nhớ, ngoài ra đích không được là hằng số.
- Lệnh CMP sẽ thực hiện trừ thử đích cho gốc (hơi giống lệnh SUB) nhưng không thay đổi giá trị của đích mà chỉ cập nhật thanh ghi cờ.
- Theo sau lệnh CMP thường là các lệnh nhảy có điều kiện.



Lệnh nhảy không điều kiện JMP

- Cú pháp: **JMP Target**
- Chuyển điều khiển không điều kiện đến Target
- Target có thể là nhãn lệnh, tên thanh ghi hoặc nội dung ngắn nhớ.
- Các dạng của lệnh nhảy:
 - Nhảy ngắn (short): $IP \leftarrow IP + (\text{giá trị có dấu 8 bit thay cho Target})$
 - **JMP SHORT NhanLenh**
 - Nhảy gần (near): $IP \leftarrow IP + (\text{giá trị có dấu 16 bit thay cho Target})$
 - **JMP NEAR NhanLenh**
 - Nhảy xa (far): $IP \leftarrow \text{Target_Ofs}; CS \leftarrow \text{Target_Seg}$
 - **NhanLenh LABEL FAR**
 - **JMP FAR NhanLenh**
 - Gián tiếp: $IP \leftarrow \text{thanh ghi / bộ nhớ} (\text{và } CS \leftarrow \text{thanh ghi / bộ nhớ})$
 - **JMP NEAR PTR BX** ; $IP \leftarrow BX$
 - **JMP WORD PTR [BX]** ; $IP \leftarrow [BX]$
 - **JMP DWORD PTR [BX]** ; $IP \leftarrow [BX] \text{ và } CS \leftarrow [BX+2]$



Các lệnh nhảy có điều kiện

- Có nhiều lệnh nhảy có điều kiện với cú pháp chung là:
JMP_{đk} NhanLenh
- Nhảy trong phạm vi khoảng cách từ -128 ÷ 127 byte.
- Các kí hiệu cần nhớ:
 - J : Jump (nhảy)
 - N : Not (không ...)
 - Z : cờ ZF; C : cờ CF; O : cờ OF; S : cờ SF; P : cờ PF
 - A : Above (lớn hơn – so sánh số không dấu)
 - B : Below (nhỏ hơn – so sánh số không dấu)
 - G : Greater (lớn hơn – so sánh số có dấu)
 - L : Less (nhỏ hơn – so sánh số có dấu)
 - E : Equal (bằng)



Nhảy hai bước

- Các lệnh nhảy có điều kiện có bước nhảy rất ngắn (khoảng cách từ -128 đến 127 byte)
- Muốn nhảy đến nhãn lệnh ở xa thì cần thực hiện qua 2 bước:
 - Bước 1: nhảy đến một nhãn lệnh trung gian ở gần đó.
 - Bước 2: từ nhãn lệnh trung gian này sử dụng lệnh JMP để nhảy đến nhãn lệnh ở xa.



Các lệnh nhảy so sánh số có dấu

Ký hiệu	Chức năng	Điều kiện nhảy
JG / JNLE	Nhảy nếu lớn hơn Nhảy nếu không nhỏ hơn hoặc bằng	ZF=0 và SF=OF
JGE / JNL	Nhảy nếu lớn hơn hoặc bằng Nhảy nếu không nhỏ hơn	SF=OF
JL / JNGE	Nhảy nếu nhỏ hơn Nhảy nếu không lớn hơn hoặc bằng	SF<>OF
JLE / JNG	Nhảy nếu nhỏ hơn hoặc bằng Nhảy nếu không lớn hơn	ZF=1 hoặc SF<>OF



Các lệnh nhảy so sánh số không dấu

Ký hiệu	Chức năng	Điều kiện nhảy
JA / JNBE	Nhảy nếu lớn hơn Nhảy nếu không nhỏ hơn hoặc bằng	ZF=0 và CF=0
JAE / JNB	Nhảy nếu lớn hơn hoặc bằng Nhảy nếu không nhỏ hơn	CF=0
JB / JNAE	Nhảy nếu nhỏ hơn Nhảy nếu không lớn hơn hoặc bằng	CF=1
JBE / JNA	Nhảy nếu nhỏ hơn hoặc bằng Nhảy nếu không lớn hơn	ZF=1 hoặc CF=1



Các lệnh nhảy điều kiện đơn

Ký hiệu	Chức năng	Điều kiện nhảy
JE / JZ	Nhảy nếu bằng Nhảy nếu bằng 0	ZF=1
JNE / JNZ	Nhảy nếu không bằng Nhảy nếu khác 0	ZF=0
JC	Nhảy nếu có nhớ	CF=1
JNC	Nhảy nếu không có nhớ	CF=0
JO	Nhảy nếu tràn	OF=1
JNO	Nhảy nếu không tràn	OF=0
JS	Nhảy nếu kết quả âm	SF=1
JNS	Nhảy nếu kết quả không	SF=0
JP / JPE	Nhảy nếu cờ chẵn	PF=1
JNP / JPO	Nhảy nếu cờ lẻ	PF=0
JCXZ	Nhảy nếu thanh ghi CX = 0	CX=0

■ Lệnh LOOP:

- Cú pháp: **LOOP NhanLenh**
- Lặp lại các lệnh từ NhanLenh đến hết lệnh LOOP cho đến khi CX = 0
- Sau mỗi lần lặp CX tự động giảm 1
- NhanLenh cách xa lệnh LOOP không quá -128 byte
- Thông thường CX được gán bằng số lần lặp trước khi vào vòng lặp.
- Ví dụ:

```
MOV AL, 0           ; gán AL = 0
MOV CX, 16          ; số lần lặp
LAP: INC AL         ; tăng AL thêm 1
LOOP LAP            ; lặp 16 lần, AL = 16
```



Các câu lệnh lặp (tiếp)

- Lệnh LOOPE / LOOPZ:

- Cú pháp: LOOPE NhanLenh
 LOOPZ NhanLenh
- Lặp lại các lệnh từ NhanLenh đến hết lệnh LOOPE / LOOPZ cho đến khi CX = 0 hoặc ZF = 0
- Phạm vi tác dụng và điều kiện CX: giống lệnh LOOP

- Lệnh LOOPNE / LOOPNZ:

- Cú pháp: LOOPNE NhanLenh
 LOOPNZ NhanLenh
- Lặp lại các lệnh từ NhanLenh đến hết lệnh LOOPNE / LOOPNZ cho đến khi CX = 0 hoặc ZF = 1
- Phạm vi tác dụng và điều kiện CX: giống lệnh LOOP

- Nhận các kí tự '0' từ bàn phím cho đến khi nhận đủ 20 lần hoặc kí tự nhập vào khác '0'.
- Mã lệnh:

MOV **AH, 1** ; hàm nhập kí tự

MOV **CX, 20** ; lặp tối đa 20 lần

DocKiTu:

INT **21h** ; nhận 1 kí tự

CMP **AL, '0'** ; so sánh với '0'

LOOPZ **DocKiTu** ; lặp lại DocKiTu

- Nhận các ký tự từ bàn phím cho đến khi nhận đủ 20 ký tự hoặc ký tự nhập vào là ENTER.
- Mã lệnh:

MOV **AH, 1** ; hàm nhập ký tự

MOV **CX, 20** ; lặp tối đa 20 lần

DocKiTu:

INT **21h** ; nhận 1 ký tự

CMP **AL, 13** ; so sánh với ENTER

LOOPNZ **DocKiTu** ; lặp lại DocKiTu



2. Cấu trúc điều kiện

- Các cấu trúc điều kiện thông dụng:

- IF <điều kiện> THEN <công việc>
- IF <điều kiện> THEN <công việc 1> ELSE <công việc 2>
- CASE <biểu thức> OF

<giá trị 1> : <công việc 1>

<giá trị 2> : <công việc 2>

....

<giá trị N> : <công việc N>

Else

<công việc N+1>

END



a. Cấu trúc IF ... THEN

- IF <điều kiện> THEN <công việc>

- Dạng lệnh:

CMP <???> ; suy ra từ <điều kiện>

JMP_{đksai} BoQua

<công việc> ; các lệnh thực hiện <công việc>

BoQua:

....



Ví dụ lệnh IF ... THEN

- Gán BX = giá trị tuyệt đối của AX
- Thuật giải:

BX := AX

If BX < 0 Then

 BX := -BX

EndIf

- Mã lệnh:

MOV BX, AX

CMP BX, 0

JNL BoQua

NEG BX

BoQua: ...



b. Cấu trúc IF ... THEN ... ELSE

- IF <điều kiện> THEN <công việc 1> ELSE <công việc 2>
 - Dạng lệnh:
 - CMP <???> ; suy ra từ <điều kiện>
 - JMP _{đksai} Viec2
 - <công việc 1> ; các lệnh thực hiện <công việc 1>
 - JMP TiepTuc
- Viec2:
- <công việc 2> ; các lệnh thực hiện <công việc 2>
- TiepTuc:
-



Ví dụ lệnh IF ... THEN ... ELSE

- AL và BL đang chứa mã ASCII của 2 ký tự. Hãy hiển thị ra màn hình ký tự có mã ASCII nhỏ hơn.

Thuật giải:

```
If AL <= BL Then  
    Hiển thị ký tự trong AL  
Else  
    Hiển thị ký tự trong BL  
EndIf
```

Mã lệnh chương trình:

MOV	AH, 2	; hàm hiển ký tự
CMP	AL, BL	; AL <= BL ?
JA	Viec2	; không đúng => tới Viec2
MOV	DL, AL	; chuyển ký tự trong AL vào DL
JMP	TiepTuc	; tới TiepTuc

Viec2:

MOV	DL, BL	; chuyển ký tự trong BL vào DL
-----	--------	--------------------------------

TiepTuc:

INT	21h	; hiện ký tự trong DL
-----	-----	-----------------------



c. Cấu trúc CASE ... OF

```
CMP      <so_sanh_1> ; suy ra từ <bíểu thức> = <giá trị 1>
JMP_dkdung    Viec_1
CMP      <so_sanh_2> ; suy ra từ <bíểu thức> = <giá trị 2>
JMP_dkdung    Viec_2
.....
CMP      <so_sanh_N> ; suy ra từ <bíểu thức> = <giá trị N>
JMP_dkdung    Viec_N
<cong_viec_N+1> ; trường hợp còn lại => thực hiện việc N+1
JMP TiepTuc

Viec_1:
<cong_viec_1> ; thực hiện việc 1
JMP TiepTuc

Viec_2:
<cong_viec_2> ; thực hiện việc 2
JMP TiepTuc

...
Viec_N:
<cong_viec_N> ; thực hiện việc N
TiepTuc: ...
```



Ví dụ lệnh CASE ... OF

- Kiểm tra nếu AX < 0 thì gán BX = -1, nếu AX = 0 thì gán BX = 0, còn nếu AX > 0 thì gán BX = 1

Thuật giải:

Case AX OF

<0: BX := -1

=0: BX := 0

Else

BX := 1

End

Mã lệnh chương trình:

CMP AX, 0 ; so sánh AX với 0

JL SoAm ; AX < 0 => tới SoAM

JE BangKhong ; AX = 0 => tới BangKhong

MOV BX, 1 ; TH còn lại, gán BX = 1

JMP TiepTuc ; kết thúc xử lý

SoAm:

MOV BX, -1 ; gán BX = -1

JMP TiepTuc ; kết thúc xử lý

BangKhong:

MOV BX, 0 ; gán BX = 0

TiepTuc:

...



d. Điều kiện chứa AND

- If <điều kiện 1> AND <điều kiện 2> Then <công việc>

- Dạng lệnh:

CMP

<so sánh 1> ; suy ra từ <điều kiện 1>

JMP_{đksai}

BoQua

CMP

<so sánh 2> ; suy ra từ <điều kiện 2>

JMP_{đksai}

BoQua

<công việc>

; thực hiện <công việc>

BoQua:

....

- Đọc 1 kí tự, nếu là chữ cái hoa thì hiển thị.

Thuật giải:

```
Đọc 1 kí tự (vào AL)
If (AL >= 'A') AND (AL <= 'Z') Then
    Hiển thị kí tự trong AL
End
```

Mã lệnh chương trình:

MOV	AH, 1	; hàm đọc kí tự
INT	21h	; AL chứa kí tự đọc được
CMP	AL, 'A'	; so sánh kí tự với 'A'
JB	BoQua	; kí tự < 'A' => BoQua
CMP	AL, 'Z'	; so sánh kí tự với 'Z'
JA	BoQua	; kí tự > 'Z' => BoQua
MOV	AH, 2	; hàm hiển kí tự
MOV	DL, AL	; DL chứa kí tự cần hiển
INT	21h	; hiển kí tự trong DL

BoQua:



e. Điều kiện chứa OR

- If <điều kiện 1> OR <điều kiện 2> Then <công việc>

- Dạng lệnh:

CMP

<so sánh 1> ; suy ra từ <điều kiện 1>

JMP_{đkđúng}

ThucHien

CMP

<so sánh 2> ; suy ra từ <điều kiện 2>

JMP_{đkđúng}

ThucHien

JMP

BoQua

ThucHien:

<công việc>

; thực hiện <công việc>

BoQua:

....

- Đọc 1 kí tự, nếu là 'y' hoặc 'Y' thì hiển thị lại, nếu không phải thì thoát chương trình.

Thuật giải:

Đọc 1 kí tự (vào AL)

If (AL = 'y') OR (AL = 'Y') Then

Hiển thị kí tự trong AL

Else

Thoát chương trình

End



Ví dụ (tiếp)

Mã lệnh chương trình:

MOV	AH, 1	; hàm đọc kí tự
INT	21h	; AL chứa kí tự đọc được
CMP	AL, 'y'	; so sánh kí tự với 'y'
JE	HienThi	; kí tự = 'y' => Hiển thị
CMP	AL, 'Y'	; so sánh kí tự với 'Y'
JE	HienThi	; kí tự = 'Y' => Hiển thị
JMP	Thoat	; các TH khác => Thoat

HienThi:

MOV	AH, 2	; hàm hiện kí tự
MOV	DL, AL	; DL chứa kí tự cần hiện
INT	21h	; hiện kí tự trong DL
JMP	TiepTuc	; tiếp tục

Thoat:

MOV	AH, 4Ch	; thoát khỏi chương trình
INT	21h	

TiepTuc:



3. Cấu trúc lặp

- Các cấu trúc lặp thông dụng:
 - FOR <số lần lặp> DO <công việc>
 - REPEAT <công việc> UNTIL <điều kiện>
 - WHILE <điều kiện> DO <công việc>



a. Lệnh lặp FOR

- FOR <số lần lặp> DO <công việc>
- Dạng lệnh:
 $<\text{Khởi tạo CX} = \text{số lần lặp}>$
; JCXZ BoQua ; nếu CX = 0 thì không lặp
VongLap:
 $<\text{công việc}>$
LOOP VongLap
BoQua:

- Hiển thị ra màn hình 80 dấu '*'
- Mã lệnh:

MOV	CX, 80	; số kí tự cần hiện
MOV	AH, 2	; hàm hiện kí tự
MOV	DL, '*'	; kí tự cần hiện là *

HienSao:

INT	21h	; hiện kí tự
LOOP	HienSao	; lặp lại 80 lần



b. Lệnh lặp REPEAT ... UNTIL

- REPEAT <công việc> UNTIL <điều kiện>
- Dạng lệnh:

VongLap:

<công việc> ; thân vòng lặp

CMP <so sánh> ; suy ra từ <điều kiện>

JMP_{đksai} VongLap

- Đọc vào các kí tự cho đến khi gặp ENTER
- Mã lệnh:

MOV AH, 1 ; hàm đọc kí tự

DocKiTu:

INT 21h ; đọc 1 kí tự vào AL

CMP AL, 13 ; kí tự là ENTER ?

JNE DocKiTu ; sai => lặp tiếp



Ví dụ 2 – REPEAT lồng với FOR

- Hiển thị ra 5 dòng, mỗi dòng gồm 50 dấu '*'
- Mã lệnh:

MOV BL, 5 ; số dòng cần hiện

HienDong:

MOV CX, 50 ; hiện 50 dấu * trên 1 dòng

MOV AH, 2 ; hàm hiện kí tự

MOV DL, '*' ; DL = kí tự cần hiện

VietSao:

INT 21h ; gọi hàm hiện kí tự trong DL

LOOP VietSao ; lặp lại cho đến khi đủ 50 '*'

MOV DL, 10 ; xuống dòng

INT 21h

MOV DL, 13 ; về đầu dòng

INT 21h

DEC BL ; giảm số dòng cần phải hiện tiếp

JNZ HienDong ; chưa hết 5 dòng => quay lại



c. Lệnh lặp WHILE ... DO

- WHILE <điều kiện> DO <công việc>
- Dạng lệnh:

VongLap:

CMP <so sánh> ; suy ra từ <điều kiện>

JMP_{đksai} DungLap ; <điều kiện> sai thì dừng

 ; thực hiện <công việc>

JMP VongLap ; lặp lại

DungLap:

- Nhập 1 dòng kết thúc bằng ENTER. Đếm số kí tự đã được nhập.
- Mã lệnh:

MOV DX, 0 ; khởi tạo bộ đếm = 0

MOV AH, 1 ; hàm đọc kí tự

DocKiTu:

INT 21h ; đọc 1 kí tự vào AL

CMP AL, 13 ; kí tự đó là ENTER ?

JE DungLap ; đúng => DungLap

INC DX ; sai => tăng bộ đếm lên 1

JMP DocKiTu ; lặp lại

DungLap:



Nội dung chương 5

- 5.1. Mở đầu về lập trình hợp ngũ
- 5.2. Các cấu trúc lập trình với hợp ngũ
- 5.3. Các lệnh logic, lệnh dịch và lệnh quay**
- 5.4. Ngăn xếp và thủ tục
- 5.5. Các lệnh nhân, chia
- 5.6. Các lệnh thao tác chuỗi
- 5.7. Một số ví dụ



5.3. Các lệnh logic, dịch và quay

1. Các lệnh logic
2. Các lệnh dịch
3. Các lệnh quay
4. Vào-ra số nhị phân và Hexa



1. Các lệnh logic

- Các phép toán logic:

a	b	a AND b	a OR b	a XOR b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

a	NOT a
0	1
1	0

- Các lệnh logic: AND, OR, XOR, NOT, TEST



Các lệnh AND, OR và XOR

■ Cú pháp:

- AND đích, nguồn ; đích \Leftarrow đích AND nguồn
- OR đích, nguồn ; đích \Leftarrow đích OR nguồn
- XOR đích, nguồn ; đích \Leftarrow đích XOR nguồn
- TEST đích, nguồn ; Phép AND nhưng không thay đổi đích

■ Chú ý:

- Toán hạng nguồn: hằng số, thanh ghi hay ngăn nhớ
- Toán hạng đích: thanh ghi hay ngăn nhớ
- Hai toán hạng không được đồng thời là ngăn nhớ

■ Ảnh hưởng tới các cờ:

- SF, ZF, PF phản ánh kết quả của lệnh
- AF không xác định
- CF = OF = 0

- VD 1: Đổi mã ASCII của 1 chữ số thành số tương ứng.
 - Giả sử AL chứa kí tự (chẳng hạn '5' – mã ASCII là 35h)
 - Cần chuyển AL về giá trị chữ số (là 5)
 - Thực hiện: **SUB AL, 30h** hoặc **AND AL, 0Fh**
- VD 2: Đổi chữ thường thành chữ hoa.
 - Giả sử DL chứa kí tự chữ thường, cần chuyển về chữ hoa.
 - Thực hiện: **SUB DL, 20h** hoặc **AND DL, 0DFh**
- VD 3: Xóa thanh ghi AX về 0.
 - Thực hiện: **XOR AX, AX**
- VD 4: Kiểm tra xem AX có bằng 0 hay không?
 - Thực hiện: **OR AX, AX** ; $AX = 0 \Leftrightarrow ZF = 1$

- Cú pháp: NOT đích
- Lệnh này không ảnh hưởng đến cờ

- Cú pháp: **TEST** đích, nguồn
- Thực hiện phép toán AND nhưng không thay đổi đích mà chỉ cập nhật các cò.
- Các cò bị tác động:
 - SF, ZF, PF phản ánh kết quả của lệnh
 - AF không xác định
 - CF = OF = 0
- Ví dụ: Kiểm tra tính chẵn lẻ của AL
 - AL chẵn \Leftrightarrow bit LSB của = 0
 - Thực hiện: TEST AL, 1 ; AL chẵn \Leftrightarrow ZF = 1



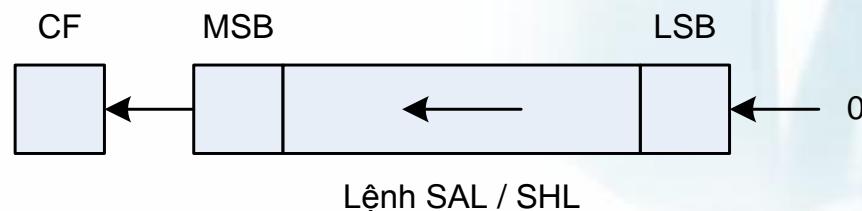
2. Các lệnh dịch

- Các lệnh dịch và quay có 2 dạng:
 - Dịch (hoặc quay) 1 vị trí:
Lệnh đích, 1
 - Dịch (hoặc quay) N vị trí:
Lệnh đích, CL ; với CL = N

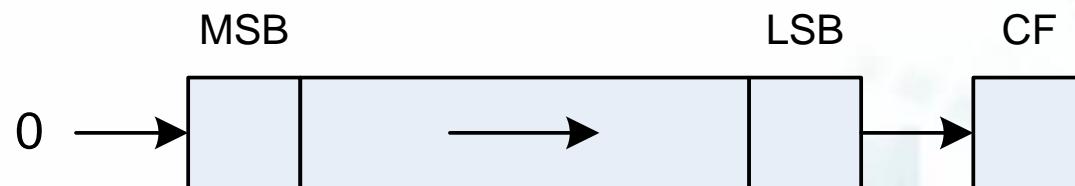


Các lệnh dịch trái

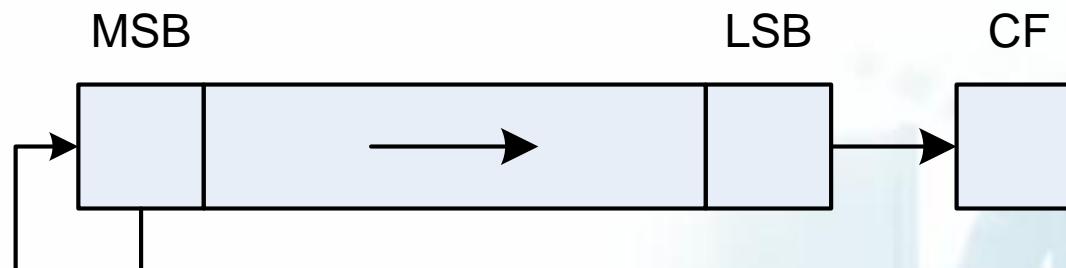
- Dịch trái số học (SAL – Shift Arithmetically Left) và dịch trái logic (SHL – Shift (Logically) Left):
 - SAL đítch, 1 hoặc SAL đítch, CL
 - SHL đítch, 1 hoặc SHL đítch, CL
- Lệnh SAL và SHL là tương đương
- Tác động vào các cờ:
 - SF, PF, ZF phản ánh kết quả
 - AF không xác định
 - CF chứa bit cuối cùng được dịch ra khỏi đítch
 - OF = 1 nếu kết quả bị thay đổi dấu trong phép dịch cuối cùng



- Dịch phải logic: SHR – Shift (Logically) Right
 - Cú pháp:
SHR đích, 1
SHR đích, CL
 - Các cờ bị tác động như là lệnh dịch trái
 - Minh họa:



- Dịch phải số học: SAR – Shift Arithmetically Right
 - Cú pháp:
SAR đích, 1
SAR đích, CL
 - Các cờ bị tác động như là lệnh SHR
 - Minh họa:





3. Các lệnh quay

- Các dạng lệnh quay:

- ROL : quay trái
- ROR : quay phải
- RCL : quay trái qua cờ nhớ
- RCR : quay phải qua cờ nhớ

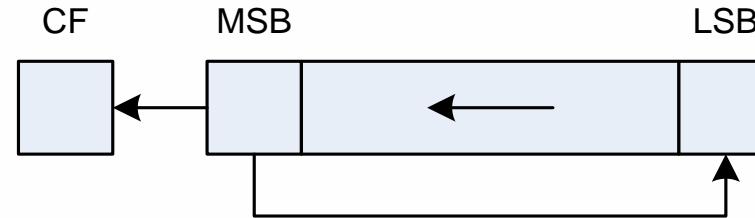
- Tác động vào các cờ:

- SF, PF, ZF phản ánh kết quả
- AF không xác định
- CF chứa bit cuối cùng bị dịch ra khỏi toán hạng
- OF = 1 nếu kết quả bị thay đổi dấu trong lần quay cuối cùng

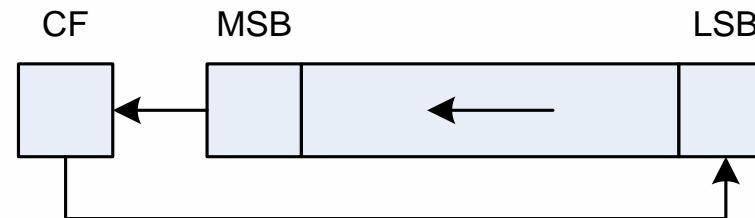


Minh họa các lệnh quay

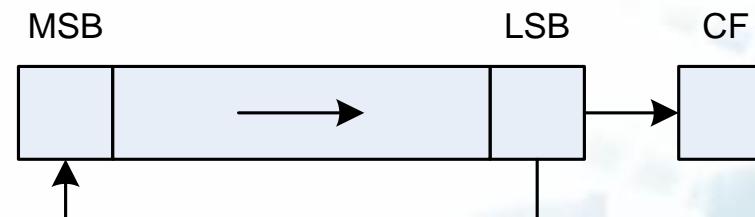
Lệnh ROL



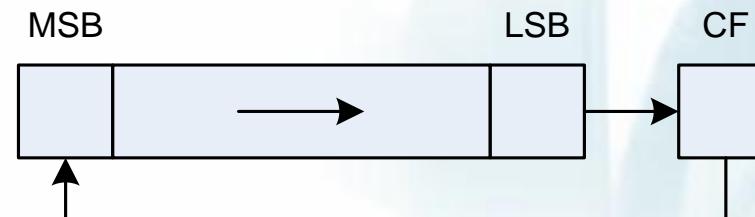
Lệnh RCL



Lệnh ROR



Lệnh RCR



- Xác định giá trị của AX và BX dưới dạng Hexa sau khi thực hiện đoạn chương trình sau:

MOV CX, 16

MOV AX, 5A6Bh

LAP :

ROL AX, 1

RCR BX, 1

LOOP LAP



4. Vào-ra số nhị phân và Hexa

- Các thao tác:
 - Nhập số nhị phân
 - In số nhị phân
 - Nhập số Hexa
 - In số Hexa



a. Nhập số nhị phân

- Đọc các bit nhị phân từ bàn phím (kết thúc nhập bằng ENTER), chuyển thành số nhị phân rồi lưu vào BX.
- Thuật giải:
 - Xóa BX (là thanh ghi chứa kết quả)
 - Nhập 1 kí tự ('0' hoặc '1')
 - WHILE kí tự <> Enter DO
 - Đổi kí tự ra giá trị nhị phân
 - Dịch trái BX
 - Chèn giá trị nhận được vào bit LSB của BX
 - Nhập kí tự
 - END WHILE



Đoạn lệnh nhập số nhị phân

XOR BX, BX
MOV AH, 1
INT 21h

; Xóa BX
; Hàm nhập ký tự
; Nhập ký tự

NhapKyTu:

CMP AL, 13
JE DungNhap
AND AL, 0Fh
SHL BX, 1
OR BL, AL
INT 21h
JMP NhapKyTu

; Là phím ENTER?
; Đúng ⇒ kết thúc nhập
; Sai ⇒ đổi ra giá trị nhị phân
; Dành chỗ cho bit mới tìm được
; Chèn bit này vào cuối BX
; Nhập tiếp ký tự khác
; Lặp lại

DungNhap:



b. In số nhị phân

- In giá trị ở BX ra màn hình dưới dạng số nhị phân.

- Thuật giải:

FOR 16 lần DO

Quay trái BX (bit MSB của BX được đưa ra CF)

IF CF = 1 THEN

Đưa ra '1'

ELSE

Đưa ra '0'

END IF

END FOR

- Có thể dùng lệnh ADC: ADC đích, nguồn
đích \Leftarrow đích + nguồn + CF

```
MOV CX, 16      ; số bit cần hiện  
MOV AH, 2       ; hàm hiện kí tự
```

Print:

```
ROL BX, 1        ; quay trái BX ⇒ CF = MSB  
MOV DL, 0        ; DL = 0  
ADC DL, 30h      ; DL ← 30h + CF  
INT 21h          ; in kí tự trong DL  
LOOP Print       ; lặp lại 16 lần
```



c. Nhập số Hexa

- Đọc các ký tự Hexa từ bàn phím (tối đa 4 chữ số, chỉ nhập các chữ số và các chữ cái hoa, kết thúc nhập bằng ENTER). Chuyển thành số Hexa tương ứng rồi lưu vào BX.
- Thuật giải:
 - Xóa BX (là thanh ghi chứa kết quả)
 - Nhập ký tự Hexa
 - WHILE ký tự <> Enter DO
 - Đổi ký tự ra nhị phân
 - Dịch trái BX 4 lần
 - Chèn giá trị mới vào 4 bit thấp nhất của BX
 - Nhập ký tự tiếp
 - END WHILE



Đoạn lệnh nhập số Hexa

```
XOR BX, BX          ; Xóa BX  
MOV CL, 4           ; Số lần dịch trái BX  
MOV AH, 1           ; Hàm nhập kí tự  
INT 21h             ; Nhập 1 kí tự ⇒ AL = mã ASCII
```

VongLap:

```
CMP AL, 13          ; Kí tự vừa nhập là Enter?  
JE KetThucLap       ; Đúng ⇒ Kết thúc  
CMP AL, '9'          ; So sánh với '9'  
JG ChuCai            ; Lớn hơn ⇒ là chữ cái hoa  
AND AL, 0Fh          ; Không lớn hơn ⇒ đổi chữ số ra nhị phân  
JMP ChenBit          ; Rồi chèn vào cuối BX
```

ChuCai:

```
SUB AL, 37h          ; Đổi chữ cái ra giá trị nhị phân
```

ChenBit:

```
SHL BX, CL           ; Dịch trái BX để dành chỗ cho c/s mới  
OR BL, AL            ; Chèn chữ số mới vào 4 bit thấp của BX  
INT 21h              ; Nhận tiếp kí tự từ bàn phím  
JMP VongLap          ; Lặp lại
```

KetThucLap:



d. In số Hexa

- Đưa giá trị Hexa 4 chữ số trong BX ra màn hình.
- Thuật giải:

FOR 4 lần DO

 Chuyển BH vào DL (giá trị cần in nằm trong BX)

 Dịch phải DL 4 vị trí

 IF DL < 10 THEN

 Đổi thành kí tự '0' ... '9'

 ELSE

 Đổi thành kí tự 'A' ... 'F'

END IF

 Đưa kí tự ra

 Quay trái BX 4 vị trí

END FOR

```
MOV CX, 4  
MOV AH, 2
```

InHexa:

```
MOV DL, BH  
SHR DL, 1  
SHR DL, 1  
SHR DL, 1  
SHR DL, 1  
CMP DL, 9  
JA ChuCai  
ADD DL, 30h  
JMP TiepTuc
```

ChuCai:

```
ADD DL, 37h
```

TiepTuc:

```
INT 21h  
ROL BX, 1  
ROL BX, 1  
ROL BX, 1  
ROL BX, 1  
LOOP InHexa
```



Nội dung chương 5

- 5.1. Mở đầu về lập trình hợp ngũ
- 5.2. Các cấu trúc lập trình với hợp ngũ
- 5.3. Các lệnh logic, lệnh dịch và lệnh quay
- 5.4. Ngăn xếp và thủ tục**
- 5.5. Các lệnh nhân, chia
- 5.6. Các lệnh thao tác chuỗi
- 5.7. Một số ví dụ



Ngăn xếp và thủ tục

1. Ngăn xếp
2. Thủ tục
3. Các ví dụ



1. Ngăn xếp

- Ngăn xếp (Stack):
 - Vùng nhớ tổ chức theo cấu trúc LIFO dùng để cất giữ thông tin.
 - Chiều của Stack từ đáy lên đỉnh ngược với chiều tăng của địa chỉ.
- Khai báo ngăn xếp: **.STACK kích_thuoc**
- Ví dụ: **.Stack 100h**
 - Khi chương trình được thực thi thì:
 - SS : chứa địa chỉ đoạn ngăn xếp
 - SP : chứa địa chỉ offset của đỉnh ngăn xếp. Ban đầu ngăn xếp rỗng nên 0100h cũng là địa chỉ của đáy ngăn xếp (=0100h).



Lệnh PUSH và PUSHF

- Lệnh **PUSH** dùng để cất 1 dữ liệu 16 bit vào trong ngăn xếp.
- Cú pháp: **PUSH** nguồn
 - nguồn là 1 thanh ghi 16 bit hoặc 1 từ nhớ (2 Byte)
- Các bước thực hiện:
 - $SP \leftarrow SP - 2$
 - Một bản sao của toán hạng nguồn được chuyển vào địa chỉ xác định bởi SS:SP (toán hạng nguồn không đổi)
- Lệnh **PUSHF** cất nội dung của thanh ghi cờ vào trong ngăn xếp.



Ví dụ về lệnh PUSH

- Xác định nội dung các Byte nhớ trong Stack.

.Stack 100h

...

Start: ; lệnh đầu tiên của chương trình

MOV AX, 1234h

MOV BX, 5678h

PUSH AX

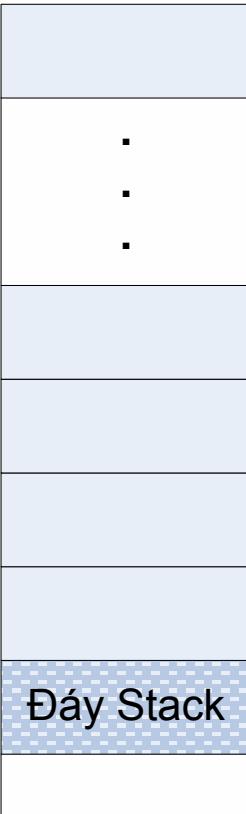
PUSH BX



Ví dụ (tiếp)

Offset

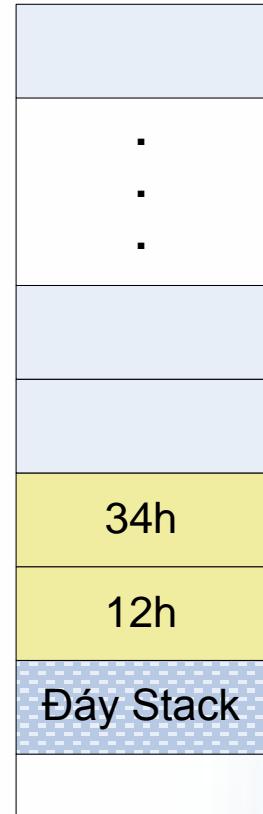
0000



Ban đầu

Offset

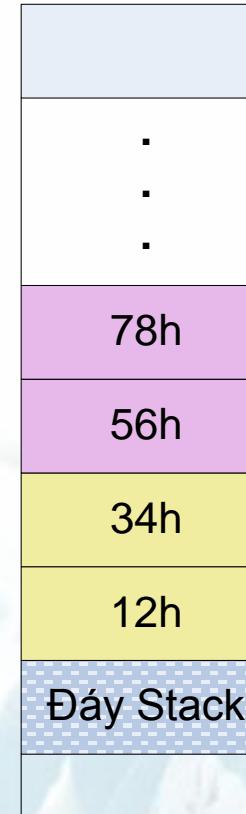
0000



Sau lệnh PUSH AX

Offset

0000



Sau lệnh PUSH BX



Lệnh POP và POPF

- Lệnh **POP** dùng để lấy ra 1 từ dữ liệu bắt đầu từ đỉnh ngăn xếp.
- Cú pháp: **POP** đích
 - đích là 1 thanh ghi 16 bit (trừ IP) hoặc 1 từ nhớ
- Các bước thực hiện:
 - Nội dung của từ nhớ ở địa chỉ xác định bởi SS:SP được chuyển tới toán hạng đích.
 - $SP \leftarrow SP + 2$
- Lệnh **POPF** đưa vào thanh ghi cờ nội dung của từ nhớ ở đỉnh ngăn xếp.



Ví dụ về lệnh POP

- Xác định nội dung các Byte nhớ trong Stack.

.Stack 100h

...

Start: ; lệnh đầu tiên của chương trình

MOV AX, 1234h

MOV BX, 5678h

PUSH AX

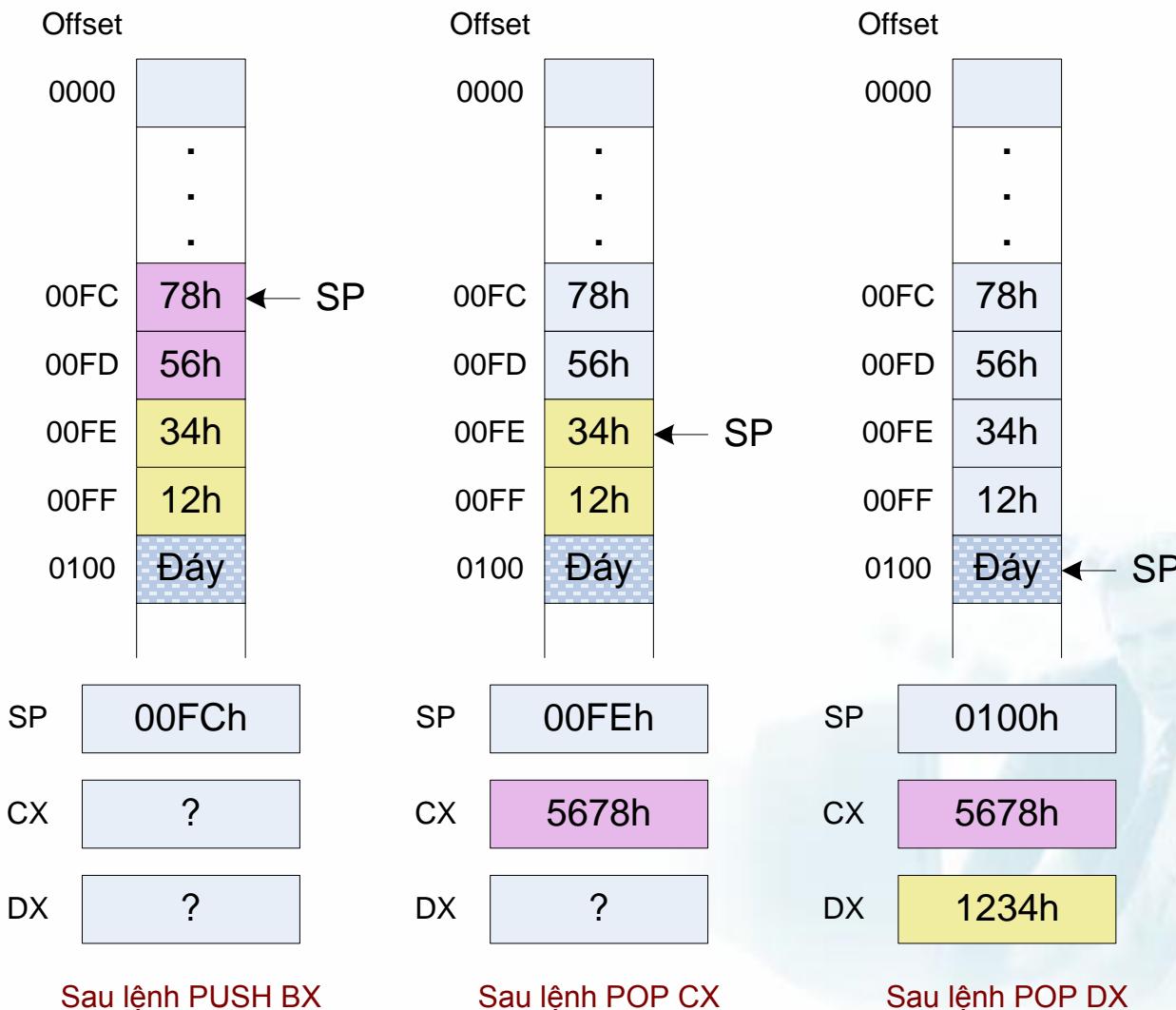
PUSH BX

POP CX

POP DX



Ví dụ (tiếp)





Một số lưu ý

- Các lệnh PUSH, PUSHF, POP và POPF không ảnh hưởng đến các cờ.
- Các lệnh trên chỉ thao tác với các WORD.
- Các lệnh sau là không hợp lệ:
 - PUSH AH ; thanh ghi 8 bit
 - POP DL ; thanh ghi 8 bit
 - PUSH 2 ; giá trị hằng số

- Dữ liệu của 1 chương trình hợp ngữ được khai báo dưới dạng:

DATA SEGMENT

mem1 dw 500

mem2 dw -50

vec1 db 1, 2, 3, 4, 8, 7

vec2 db 10, 20, -10, -20, -30, -40

DATA ENDS

- Hãy xác định nội dung của DX (Hexa) sau khi thực hiện đoạn lệnh sau:

push mem1

push mem2

mov bp, sp

mov dx, [bp]+2

a) FFCE

b) 0000

c) 01F4

d) FFFF



Bài tập 2 (tiếp)

Offset	
mem1	F4h
	01h
mem2	CEh
	FFh
vec1	01h
	02h
	03h
	04h
	08h
	07h
vec2	14h
	F6h
	ECh
	E2h
	D8h

Offset
0000
▪
▪
▪
i-4 CEh ← SP
i-3 FFh
i-2 F4h
i-1 01h
i Đáy



2. Thủ tục

- Ngoài thủ tục chính, ta có thể khai báo và sử dụng các thủ tục khác.
- Khai báo thủ tục:

Tên_thủ_tục PROC Kiểu_thủ_tục

 <Thân thủ tục>

 RET

Tên_thủ_tục ENDP

- Trong đó:

- Tên_thủ_tục: do người lập trình định nghĩa

- Kiểu_thủ_tục:

- NEAR : gọi thủ tục ở trong cùng 1 đoạn

- FAR : gọi thủ tục ở đoạn khác

- Là lệnh gọi chương trình con (thủ tục)
- Thông dụng: **CALL Tên_thủ_tục**
- Các bước thực hiện:
 - Thủ tục NEAR
 - $SP \leftarrow SP - 2$
 - Cắt nội dung của IP (địa chỉ quay về) vào Stack
 - Nạp địa chỉ của lệnh đầu tiên của chương trình con vào IP
 - Thủ tục FAR
 - $SP \leftarrow SP - 2$
 - Cắt nội dung của CS vào Stack
 - $SP \leftarrow SP - 2$
 - Cắt nội dung của IP vào Stack
 - Nạp vào CS và IP địa chỉ đầu của chương trình con

- Là lệnh trả về từ chương trình con
- Các bước thực hiện:
 - Trở về kiểu NEAR
 - IP \leftarrow word nhớ đỉnh Stack
 - SP \leftarrow SP + 2
 - Trở về kiểu FAR (RETF)
 - IP \leftarrow word nhớ đỉnh Stack
 - SP \leftarrow SP + 2
 - CS \leftarrow word nhớ tiếp
 - SP \leftarrow SP + 2



Truyền dữ liệu giữa các thủ tục

- Các thủ tục của hợp ngữ không có danh sách tham số đi kèm như các ngôn ngữ lập trình bậc cao.
- Người lập trình phải nghĩ ra cách truyền dữ liệu giữa các thủ tục.
- Các cách truyền dữ liệu thông dụng:
 - Truyền qua thanh ghi
 - Sử dụng biến toàn cục
 - Truyền địa chỉ của dữ liệu
 - Sử dụng ngăn xếp (thường dùng trong các NNLT bậc cao)



3. Các ví dụ

- VD1: Nhập 1 chuỗi kí tự kết thúc bởi ENTER. Hiện chuỗi kí tự viết theo thứ tự ngược lại ở dòng tiếp theo.
- VD2: Cài đặt các thủ tục viết số nhị phân và số Hexa ra màn hình.



Nội dung chương 5

- 5.1. Mở đầu về lập trình hợp ngũ
- 5.2. Các cấu trúc lập trình với hợp ngũ
- 5.3. Các lệnh logic, lệnh dịch và lệnh quay
- 5.4. Ngăn xếp và thủ tục
- 5.5. Các lệnh nhân, chia**
- 5.6. Các lệnh thao tác chuỗi
- 5.7. Một số ví dụ



5.5. Các lệnh nhân, chia

1. Các lệnh MUL và IMUL
2. Các lệnh DIV và IDIV
3. Vào-ra số thập phân



1. Các lệnh MUL và IMUL

- Có sự khác nhau giữa phép nhân các số không dấu với phép nhân các số có dấu.
- Lệnh nhân cho các số không dấu: **MUL** nguồn
- Lệnh nhân cho các số có dấu: **IMUL** nguồn
- Các lệnh trên làm việc với byte (cho KQ là 1 word) hoặc word (cho KQ là 1 double word)
- **nguồn** (thanh ghi / ngăn nhớ) được coi là số nhân, nếu nguồn là giá trị:
 - 8 bit: $AX \leftarrow AL \times \text{nguồn}$
 - Số bị nhân là số 8 bit chứa trong AL
 - Tích là số 16 bit chứa trong AX
 - 16 bit: $DXAX \leftarrow AX \times \text{nguồn}$
 - Số bị nhân là số 16 bit chứa trong AX
 - Tích là số 16 bit chứa trong DXAX





Ảnh hưởng đến các cờ

- SF, ZF, AF, PF : không xác định
- Sau lệnh MUL:
 - CF = OF = 0 nếu nửa cao của kết quả = 0
 - CF = OF = 1 trong các trường hợp còn lại
- Sau lệnh IMUL:
 - CF = OF = 0 nếu nửa cao của kết quả chỉ chứa các giá trị của dấu
 - CF = OF = 1 trong các trường hợp còn lại
- Nói cách khác, CF = OF = 1 nghĩa là kết quả quá lớn để chứa trong nửa thấp (AL hoặc AX) của tích.



2. Các lệnh DIV và IDIV

- Phép chia không dấu: DIV số_chia
- Phép chia có dấu: IDIV số_chia
- Chia số 16 bit (trong AX) cho số chia 8 bit hoặc chia số 32 bit (trong DXAX) cho số chia 16 bit.
- Thương và số dư có cùng kích thước với số chia.
 - Số chia 8 bit: AL chứa thương, AH chứa số dư
 - Số chia 16 bit: AX chứa thương, DX chứa số dư
- Số dư và số chia có cùng dấu.
- Nếu số chia = 0 hoặc thương nằm ngoài khoảng xác định thì BXL thực hiện INT 0 (lỗi chia cho 0).
- Các cờ không xác định sau phép chia.



Sự mở rộng dấu của số bị chia

- Trong phép chia cho Word, số bị chia được đặt trong DXAX ngay cả khi nó có thể chứa vừa trong AX. Khi đó DX phải được chuẩn bị như sau:
 - Với lệnh DIV, DX phải được xóa về 0.
 - Với lệnh IDIV, DX được lấp đầy bằng bit dấu của AX. Phép biến đổi này được thực hiện bởi lệnh **CWD**.
- Trong phép chia cho Byte, số bị chia được đặt trong AX ngay cả khi nó có thể chứa vừa trong AL. Khi đó AH phải được chuẩn bị như sau:
 - Với lệnh DIV, AH phải được xóa về 0.
 - Với lệnh IDIV, AH được lấp đầy bằng bit dấu của AL. Phép biến đổi này được thực hiện bởi lệnh **CBW**.



3. Vào-ra số thập phân

- Các thao tác:
 - In số thập phân
 - Nhập số thập phân



a. In số thập phân

- In số nguyên có dấu trong AX ra màn hình dưới dạng số thập phân.
- Thuật giải:

IF AX < 0 THEN

In ra dấu '-'

AX := số bù 2 của AX

END IF

Lấy dạng thập phân của từng chữ số trong AX

Đổi các chữ số này ra kí tự rồi in ra màn hình



In số thập phân (tiếp)

- Lấy dạng thập phân của từng chữ số trong AX:

Đếm := 0

REPEAT

 Chia số bị chia cho 10

 ; số bị chia ban đầu = AX

 Cắt số dư vào trong Stack

 Đếm := Đếm + 1

UNTIL Thương = 0

- Đổi các chữ số ra kí tự rồi in ra màn hình:

FOR Đếm lần DO

 Lấy từng chữ số từ Stack

 Đổi ra kí tự

 In kí tự đó ra màn hình

END FOR



b. Nhập số thập phân

- Thuật giải (đơn giản):

Tổng := 0

Đọc 1 kí tự ASCII

REPEAT

 Đổi kí tự ra giá trị thập phân

 Tổng := Tổng * 10 + giá trị nhận được

 Đọc kí tự

UNTIL kí tự vừa nhận = Enter



Nội dung chương 5

- 5.1. Mở đầu về lập trình hợp ngũ
- 5.2. Các cấu trúc lập trình với hợp ngũ
- 5.3. Các lệnh logic, lệnh dịch và lệnh quay
- 5.4. Ngăn xếp và thủ tục
- 5.5. Các lệnh nhân, chia
- 5.6. Các lệnh thao tác chuỗi**
- 5.7. Một số ví dụ



5.6. Các lệnh thao tác chuỗi

1. Cờ định hướng
2. Chuyển một chuỗi
3. Lưu kí tự vào chuỗi
4. Nạp kí tự của chuỗi
5. Tìm kí tự trong chuỗi
6. So sánh chuỗi
7. Tổng kết thao tác chuỗi





1. Cờ định hướng

- Cờ định hướng DF (Direction Flag) xác định hướng cho các thao tác chuỗi.
- Các thao tác chuỗi được thực hiện thông qua 2 thanh ghi chỉ số SI và DI.
- Nếu $DF = 0$ thì SI và DI được xử lý theo chiều tăng của địa chỉ bộ nhớ (từ trái qua phải trong chuỗi).
- Nếu $DF = 1$ thì SI và DI được xử lý theo chiều giảm của địa chỉ bộ nhớ (từ phải qua trái trong chuỗi).



Các lệnh CLD và STD

- Lệnh CLD (Clear Direction Flag): xóa cờ hướng
CLD ; xóa DF = 0
- Lệnh STD (Set Direction Flag): thiết lập cờ hướng
STD ; thiết lập DF = 1
- Các lệnh này không ảnh hưởng đến các cờ khác.



2. Chuyển một chuỗi

- Bài toán: giả sử có 2 chuỗi được định nghĩa như sau:

.DATA

STRING1 DB 'BACH KHOA'

STRING2 DB 9 DUP (?)

- Cần chuyển nội dung của chuỗi STRING1 (chuỗi nguồn) sang chuỗi STRING2 (chuỗi đích).

- **Lệnh: MOVSB (Move String Byte)**
 - Chuyển 1 phần tử 1 byte của chuỗi gốc (trỏ bởi DS:SI) sang 1 phần tử của chuỗi đích (trỏ bởi ES:DI).
 - Sau khi thực hiện:
 - SI và DI tăng thêm 1 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI và DI giảm đi 1 nếu cờ hướng DF = 1 (dùng lệnh STD)
- **Lệnh: MOVSW (Move String Word)**
 - Chuyển 1 phần tử 1 word (2 byte) của chuỗi gốc (trỏ bởi DS:SI) sang 1 phần tử của chuỗi đích (trỏ bởi ES:DI).
 - Sau khi thực hiện:
 - SI và DI tăng thêm 2 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI và DI giảm đi 2 nếu cờ hướng DF = 1 (dùng lệnh STD)



Các lệnh liên quan (tiếp)

- Để chuyển nhiều kí tự ta cần sử dụng các lệnh lặp.
- Lệnh: **REP <lệnh thao tác kí tự>**
 - Lặp lại lệnh viết sau đó cho đến khi CX = 0
 - Mỗi lần lặp CX giảm đi 1 \Rightarrow số lần lặp phải gán trước vào CX. Ví dụ:
MOV CX, 5
REP MOVSB ; chuyển 5 byte từ chuỗi nguồn đến chuỗi đích
- Lệnh: **REPE/REPZ <lệnh thao tác kí tự>**
 - Lặp lại lệnh viết sau đó cho đến khi CX = 0 hoặc ZF = 0
- Lệnh: **REPNE/REPNZ <lệnh thao tác kí tự>**
 - Lặp lại lệnh viết sau đó cho đến khi CX = 0 hoặc ZF = 1



Ví dụ

```
MOV AX, @DATA  
MOV DS, AX      ; khởi tạo DS  
MOV ES, AX      ; và ES đều trỏ đến đoạn dữ liệu DATA  
LEA SI, STRING1 ; SI trỏ đến chuỗi nguồn  
LEA DI, STRING2 ; DI trỏ đến chuỗi đích  
CLD            ; Xóa cờ hướng  
MOV CX, 9       ; Số byte cần chuyển  
REP MOVSB      ; Chuyển 9 byte từ STRING1 sang STRING2
```



Giải thích ví dụ

SI	
STRING1	'B' 'A' 'C' 'H' '' 'K' 'H' 'O' 'A'
Offset	0 1 2 3 4 5 6 7 8

DI	
STRING2	
Offset	9 10 11 12 13 14 15 16 17

Trước khi thực hiện các lệnh MOVSB

SI	
STRING1	'B' 'A' 'C' 'H' '' 'K' 'H' 'O' 'A'
Offset	0 1 2 3 4 5 6 7 8

DI	
STRING2	'B'
Offset	9 10 11 12 13 14 15 16 17

Sau khi thực hiện lệnh MOVSB thứ 1

SI	
STRING1	'B' 'A' 'C' 'H' '' 'K' 'H' 'O' 'A'
Offset	0 1 2 3 4 5 6 7 8

DI	
STRING2	'B' 'A'
Offset	9 10 11 12 13 14 15 16 17

Sau khi thực hiện lệnh MOVSB thứ 2

SI	
STRING1	'B' 'A' 'C' 'H' '' 'K' 'H' 'O' 'A'
Offset	0 1 2 3 4 5 6 7 8

SI	
STRING2	'B' 'A' 'C' 'H' '' 'K' 'H' 'O' 'A'
Offset	9 10 11 12 13 14 15 16 17

DI

Sau khi thực hiện lệnh MOVSB thứ 9



3. Lưu kí tự vào chuỗi

- **Lệnh: STOSB** (Store String Byte from AL)
 - Chuyển nội dung thanh ghi AL sang 1 phần tử (1 byte) được trả bởi ES:DI của chuỗi đích.
 - Sau khi thực hiện:
 - DI tăng thêm 1 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - DI giảm đi 1 nếu cờ hướng DF = 1 (dùng lệnh STD)
- **Lệnh: STOSW** (Store String Word from AX)
 - Chuyển nội dung thanh ghi AX sang 1 phần tử (2 byte) được trả bởi ES:DI của chuỗi đích.
 - Sau khi thực hiện:
 - DI tăng thêm 2 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - DI giảm đi 2 nếu cờ hướng DF = 1 (dùng lệnh STD)

- **Lưu 5 ký tự 'A' vào đầu chuỗi STRING2**

MOV	AX, @DATA	
MOV	ES, AX	; ES trỏ đến đoạn dữ liệu DATA
LEA	DI, STRING2	; ES:DI trỏ đến đầu chuỗi STRING2
CLD		; Xóa cờ hướng
MOV	CX, 5	; Số ký tự cần lưu
MOV	AL, 'A'	; Ký tự cần lưu vào chuỗi
REP	STOSB	; Lặp lưu 5 lần ký tự 'A' vào STRING2



Ví dụ 2

- Nhập các ký tự từ bàn phím rồi lưu vào chuỗi STRING cho đến khi nhập đủ 20 ký tự hoặc gặp phím ENTER.

MOV	AX, @DATA
MOV	ES, AX
LEA	DI, STRING
CLD	
MOV	CX, 20
XOR	BX, BX
MOV	AH, 1

; ES trỏ đến đoạn dữ liệu DATA
; ES:DI trỏ đến đầu chuỗi STRING
; Xóa cờ hướng
; Số ký tự tối đa được nhập từ bàn phím
; Khởi tạo số ký tự được nhập ban đầu = 0
; Hàm nhập ký tự từ bàn phím

DocKiTu:

INT	21h
CMP	AL, 13
JZ	DungNhap
STOSB	
INC	BX
LOOP	DocKiTu

; Nhập 1 ký tự \Rightarrow AL chứa mã ASCII của ký tự
; Là phím ENTER ?
; \Rightarrow Dừng nhập
; Nếu không phải thì lưu AL vào chuỗi STRING
; Tăng số đếm số ký tự được nhập
; Lặp lại (tối đa 20 lần)

DungNhap:



4. Nạp kí tự của chuỗi

- Lệnh: **LODSB** (Load String Byte into AL)
 - Chuyển 1 phần tử (1 byte) được trả bởi DS:SI của chuỗi nguồn vào thanh ghi AL.
 - Sau khi thực hiện:
 - SI tăng thêm 1 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI giảm đi 1 nếu cờ hướng DF = 1 (dùng lệnh STD)
- Lệnh: **LODSW** (Load String Word into AX)
 - Chuyển 1 phần tử (2 byte) được trả bởi DS:SI của chuỗi nguồn vào thanh ghi AX.
 - Sau khi thực hiện:
 - SI tăng thêm 2 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI giảm đi 2 nếu cờ hướng DF = 1 (dùng lệnh STD)



Ví dụ

- Giả sử STR1 và STR2 là các chuỗi có độ dài là 40 kí tự. Viết đoạn chương trình chuyển các kí tự chữ hoa từ STR1 sang STR2.

MOV	AX, @DATA	
MOV	DS, AX	; khởi tạo DS
MOV	ES, AX	; và ES đều trỏ đến đoạn dữ liệu DATA
LEA	SI, STR1	; SI trỏ đến chuỗi nguồn STR1
LEA	DI, STR2	; DI trỏ đến chuỗi đích STR2
CLD		; Xóa cờ hướng
MOV	CX, 40	; Số kí tự của STR1 cần xét (độ dài xâu STR1)
XOR	BX, BX	; Khởi tạo số kí tự thực sự được chuyển ban đầu = 0

VongLap:

LODSB		; Nạp 1 kí tự của STR1 vào AL
CMP	AL, 'A'	; Nếu AL < 'A' \Rightarrow không phải là chữ hoa
JB	LapTiep	; thì xét kí tự tiếp theo
CMP	AL, 'Z'	; Nếu AL > 'Z' \Rightarrow không phải là chữ hoa
JA	LapTiep	; thì xét kí tự tiếp theo
STOSB		; Nếu AL là chữ cái hoa thì cất vào chuỗi STR2
INC	BX	; Tăng số đếm số chữ cái hoa

LapTiep:

LOOP	VongLap	; Lặp lại, xét kí tự tiếp theo của STR1
------	---------	---



5. Tìm kí tự trong chuỗi

- Lệnh: **SCASB** (Scan String Byte)
 - Trù thử nội dung của AL cho 1 byte đích đang được trỏ bởi ES:DI, không thay đổi giá trị AL mà chỉ cập nhật cờ.
 - Sau khi thực hiện:
 - DI tăng thêm 1 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - DI giảm đi 1 nếu cờ hướng DF = 1 (dùng lệnh STD)
- Lệnh: **SCASW** (Scan String Word)
 - Trù thử nội dung của AX cho 1 word đích đang được trỏ bởi ES:DI, không thay đổi giá trị AX mà chỉ cập nhật cờ.
 - Sau khi thực hiện:
 - DI tăng thêm 2 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - DI giảm đi 2 nếu cờ hướng DF = 1 (dùng lệnh STD)

- Cho 1 chuỗi được khai báo như sau:

```
.DATA  
STRING1      DB      'ABC'
```

- Khảo sát đoạn chương trình sau:

```
MOV      AX, @DATA
```

```
MOV      ES, AX
```

```
CLD
```

```
LEA      DI, STRING1
```

```
MOV      AL, 'B'
```

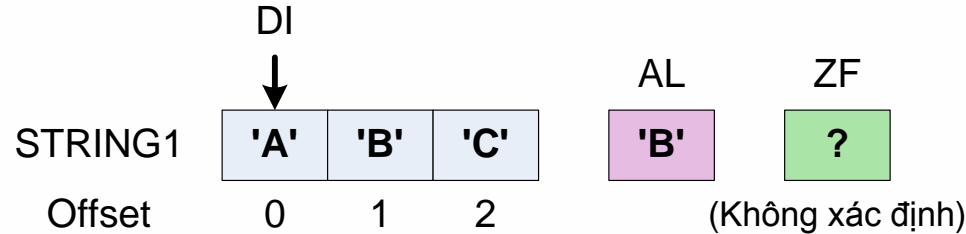
```
SCASB
```

```
SCASB
```

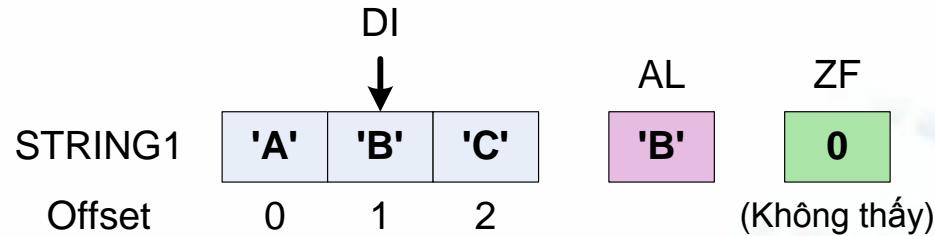


Ví dụ 1 (tiếp)

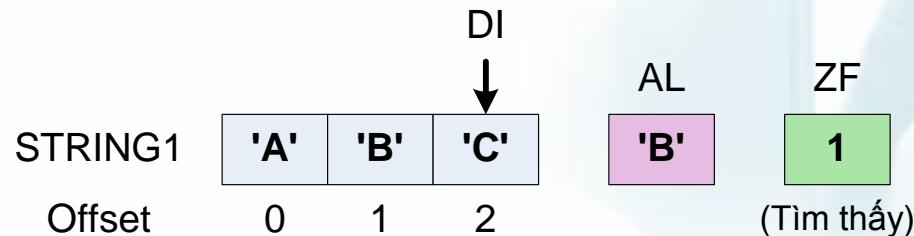
Trước khi thực hiện lệnh SCASB



Sau khi thực hiện lệnh SCASB thứ 1



Sau khi thực hiện lệnh SCASB thứ 2



- Tìm chữ cái 'A' đầu tiên trong chuỗi STRING2 có độ dài 40 kí tự.
- Đoạn chương trình:

```
MOV      AX, @DATA  
MOV      ES, AX          ; ES trỏ đến đoạn dữ liệu  
CLD  
LEA      DI, STRING2    ; ES:DI trỏ đến chuỗi đích STRING2  
MOV      CX, 40          ; Độ dài chuỗi STRING2  
MOV      AL, 'A'          ; AL chứa kí tự cần tìm  
REPNE   SCASB           ; Tìm cho đến khi thấy hoặc CX=0
```

- Ra khỏi đoạn chương trình:
 - Nếu ZF = 1 thì ES:[DI-1] là kí tự 'A' đầu tiên tìm thấy
 - Nếu ZF = 0 thì trong chuỗi STRING2 không chứa kí tự 'A'



6. So sánh chuỗi

- Lệnh: **CMPSB** (Compare String Byte)
 - Trừ thử 1 byte ở địa chỉ DS:SI cho 1 byte ở địa chỉ ES:DI, kết quả không được lưu lại mà chỉ cập nhật cờ.
 - Sau khi thực hiện:
 - SI, DI tăng thêm 1 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI, DI giảm đi 1 nếu cờ hướng DF = 1 (dùng lệnh STD)
- Lệnh: **CMPSW** (Compare String Word)
 - Trừ thử 1 word ở địa chỉ DS:SI cho 1 word ở địa chỉ ES:DI, kết quả không được lưu lại mà chỉ cập nhật cờ.
 - Sau khi thực hiện:
 - SI, DI tăng thêm 2 nếu cờ hướng DF = 0 (dùng lệnh CLD)
 - SI, DI giảm đi 2 nếu cờ hướng DF = 1 (dùng lệnh STD)

- Cho 2 chuỗi được khai báo như sau:

.DATA

STRING1 DB 'ABC'

STRING2 DB 'ACB'

- Khảo sát đoạn chương trình sau:

MOV AX, @DATA

MOV DS, AX

MOV ES, AX

CLD

LEA SI, STRING1

LEA DI, STRING2

CMPSB

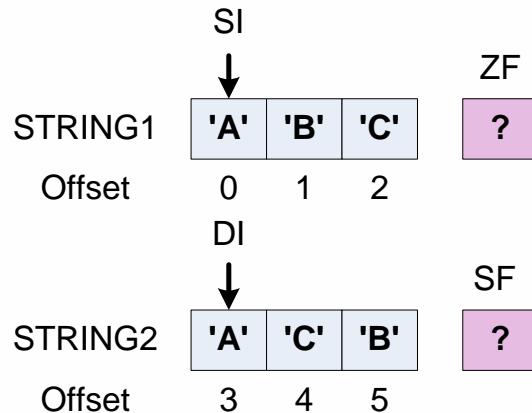
CMPSB

CMPSB

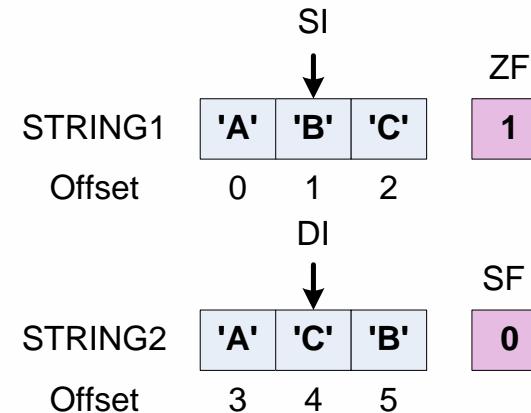



Ví dụ (tiếp)

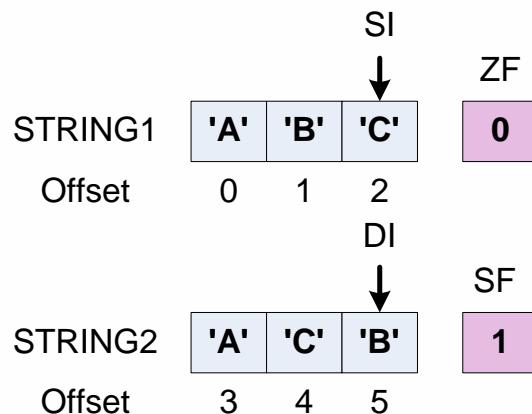
Trước lệnh CMPSB thứ 1



Sau lệnh CMPSB thứ 1



Sau lệnh CMPSB thứ 2



Sau lệnh CMPSB thứ 3





7. Tổng kết thao tác chuỗi

Lệnh	Toán hạng nguồn	Toán hạng đích	Dạng byte	Dạng word
Chuyển chuỗi	DS : SI	ES : DI	MOVSB	MOVSW
Lưu kí tự vào chuỗi	AL hay AX	ES : DI	STOSB	STOSW
Nạp kí tự của chuỗi	DS : SI	AL hay AX	LODSB	LODSW
Tìm kí tự trong chuỗi	AL hay AX	ES : DI	SCASB	SCASW
So sánh chuỗi <i>(Không lưu KQ)</i>	DS : SI	ES : DI	CMPSB	CMPSW



Nội dung chương 5

- 5.1. Mở đầu về lập trình hợp ngũ
- 5.2. Các cấu trúc lập trình với hợp ngũ
- 5.3. Các lệnh logic, lệnh dịch và lệnh quay
- 5.4. Ngăn xếp và thủ tục
- 5.5. Các lệnh nhân, chia
- 5.6. Các lệnh thao tác chuỗi
- 5.7. Một số ví dụ**



5.7. Một số ví dụ

- Bài tập 1:
 - Đọc 1 chuỗi kí tự từ bàn phím cho đến khi gặp phím ENTER.
 - Hiện các kí tự vừa nhập theo chiều ngược lại.
- Bài tập 2:
 - Nhập nội dung chuỗi STR1 có tối đa 40 kí tự từ bàn phím. Quá trình nhập kết thúc khi gặp phím ENTER.
 - Duyệt qua chuỗi STR1, chuyển các kí tự chữ hoa sang chuỗi STR2.
 - Hiển thị nội dung chuỗi STR2 ra màn hình.



Một số ví dụ (tiếp)

- Bài tập 3: Dùng vòng lặp hiển thị ra hình vẽ sau:

```
*  
**  
***  
****  
*****
```

- Bài tập 4: Dùng vòng lặp hiển thị ra hình vẽ sau:

```
*  
***  
*****  
*****  
*****
```