



TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
Hanoi University of Science and Technology

HỆ THỐNG MÁY TÍNH

Computer Systems

Nguyễn Kim Khánh

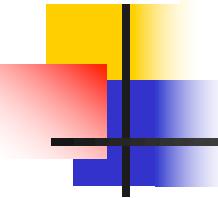
Bộ môn Kỹ thuật máy tính

Viện Công nghệ thông tin và Truyền thông

Department of Computer Engineering (DCE)

School of Information and Communication Technology (SoICT)

Version: CS-HEDSPI2019



Contact Information

- Address: 502-B1
- Mobile: 091-358-5533
- e-mail: khanhnk@soict.hust.edu.vn
khanh.nguyenkim@hust.edu.vn

Mục tiêu

- Hai học phần liên thông:
 - Kiến trúc máy tính (Computer Architecture)
 - Hệ thống máy tính (Computer Systems)
- Sinh viên được trang bị các kiến thức về kiến trúc tập lệnh và tổ chức của máy tính
- Sau khi học xong cả hai học phần, sinh viên có khả năng:
 - Tìm hiểu kiến trúc tập lệnh của các bộ xử lý cụ thể
 - Lập trình hợp ngữ
 - Đánh giá hiệu năng máy tính
 - Khai thác và quản trị hiệu quả các hệ thống máy tính
 - Phân tích và thiết kế các thành phần của máy tính

Mục tiêu của từng học phần

- **Kiến trúc máy tính**
 - Kiến trúc tập lệnh
 - Chương trình nguồn được dịch ra thành mã máy như thế nào ?
 - Phần cứng thực hiện chương trình mã máy như thế nào ?
- **Hệ thống máy tính**
 - Đánh giá hiệu năng hệ thống máy tính
 - Tổ chức các thành phần của hệ thống máy tính
 - Các kiến trúc máy tính song song

Tài liệu học tập

- *Bài giảng Hệ thống máy tính*
<ftp://dce.soict.hust.edu.vn/khanhnk/IT4272/>
- **Sách giáo trình:**
[1] David A. Patterson, John L. Hennessy
Computer Organization and Design – 2012, Revised 4th edition
- **Sách tham khảo:**
[2] William Stallings
Computer Organization and Architecture – 2013, 9th edition
- [3] David Money Harris, Sarah L. Harris
Digital Design and Computer Architecture – 2013, 2nd edition
- [4] Andrew S. Tanenbaum
Structured Computer Organization – 2013, 6th edition

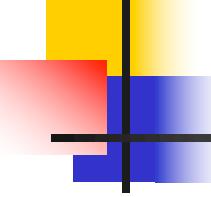
Nội dung học phần

Chương 1. Tổng quan hệ thống máy tính

Chương 2. Bộ nhớ máy tính

Chương 3. Hệ thống vào-ra

Chương 4. Các kiến trúc song song



Hệ thống máy tính

Chương 1

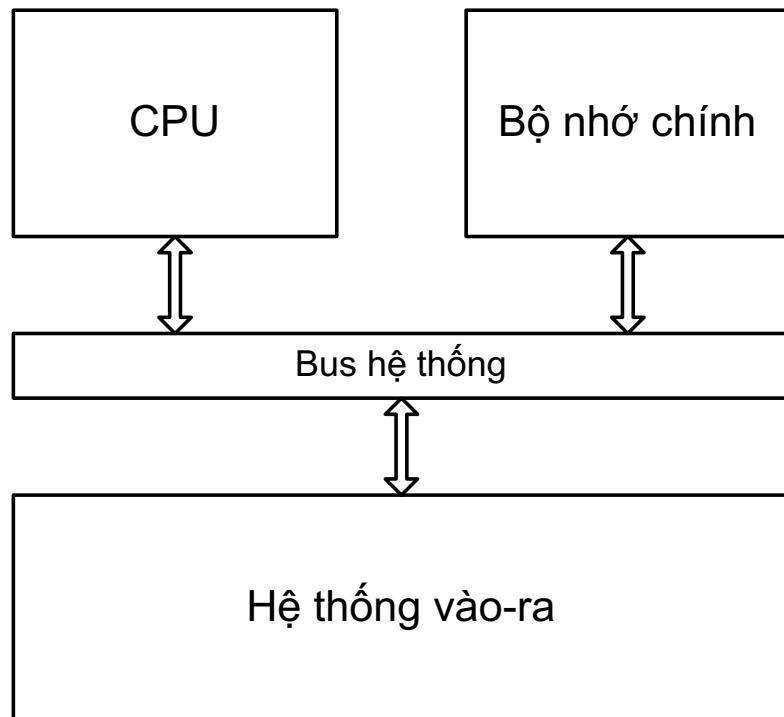
TỔNG QUAN HỆ THỐNG MÁY TÍNH

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Nội dung của chương 1

- 1.1. Các thành phần cơ bản của máy tính
- 1.2. Hoạt động cơ bản của máy tính
- 1.3. Bus máy tính
- 1.4. Hiệu năng máy tính

1.1. Các thành phần cơ bản của máy tính

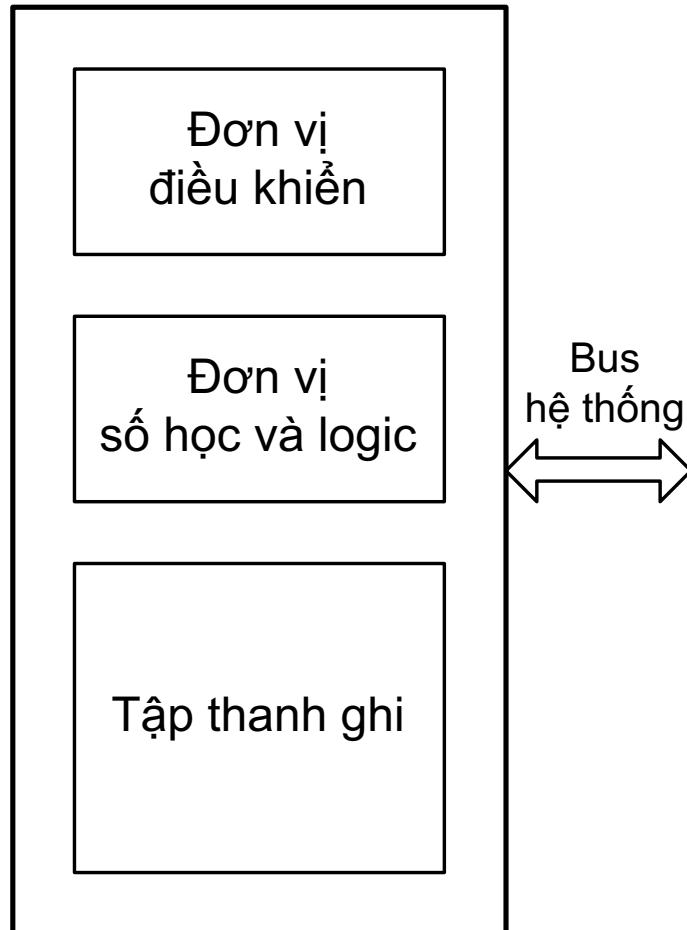


- **Bộ xử lý trung tâm (Central Processing Unit – CPU)**
 - Điều khiển hoạt động của máy tính và xử lý dữ liệu
- **Bộ nhớ chính (Main Memory)**
 - Chứa các chương trình đang thực hiện
- **Hệ thống vào-ra (Input/Output)**
 - Trao đổi thông tin giữa máy tính với bên ngoài
- **Bus hệ thống (System bus)**
 - Kết nối và vận chuyển thông tin

1. Bộ xử lý trung tâm (CPU)

- Chức năng:
 - điều khiển hoạt động của máy tính
 - xử lý dữ liệu
- Nguyên tắc hoạt động cơ bản:
 - CPU hoạt động theo chương trình nằm trong bộ nhớ chính.
- Là thành phần nhanh nhất trong hệ thống

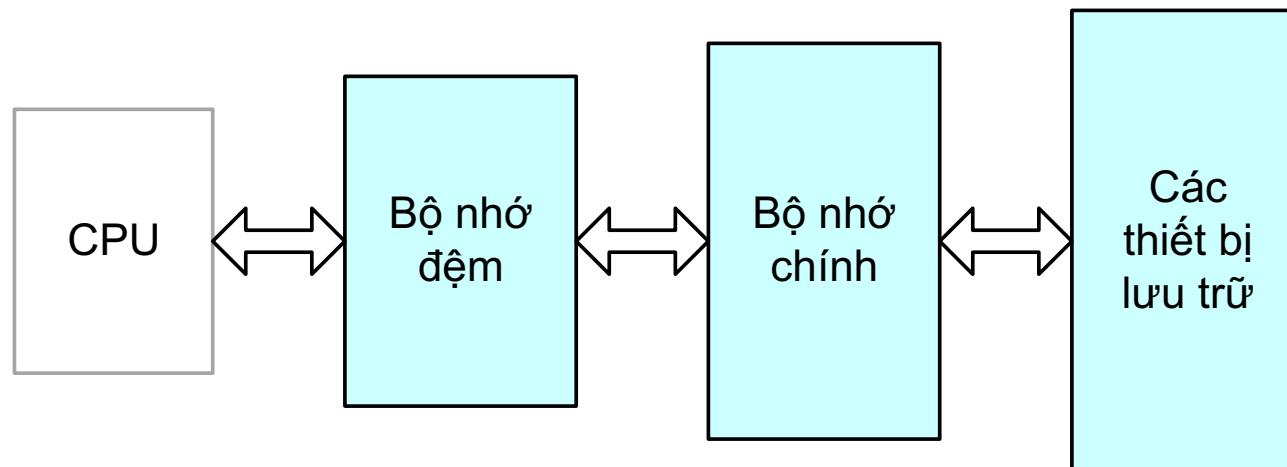
Các thành phần cơ bản của CPU



- **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 phép toán logic
- **Tập thanh ghi**
 - *Register File (RF)*
 - Gồm các thanh ghi chứa các thông tin phục vụ cho hoạt động của CPU

2. Bộ nhớ máy tính

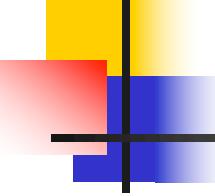
- Chức năng: nhớ chương trình và dữ liệu (dưới dạng nhị phân)
- Các thao tác cơ bản với bộ nhớ:
 - Thao tác ghi (Write)
 - Thao tác đọc (Read)
- Các thành phần chính:
 - Bộ nhớ chính (Main memory)
 - Bộ nhớ đệm (Cache memory)
 - Thiết bị lưu trữ (Storage Devices)



Bộ nhớ chính (Main memory)

- Tồn tại trên mọi máy tính
- Chứa các lệnh và dữ liệu của chương trình đang được thực hiện
- Sử dụng bộ nhớ bán dẫn
- Tổ chức thành các ngăn nhớ được đánh địa chỉ (thường đánh địa chỉ cho từng byte nhớ)
- Nội dung của ngăn nhớ có thể thay đổi, song địa chỉ vật lý của ngăn nhớ luôn cố định
- CPU muốn đọc/ghi ngăn nhớ cần phải biết địa chỉ ngăn nhớ đó

Nội dung	Địa chỉ
0100 1101	00...0000
0101 0101	00...0001
1010 1111	00...0010
0000 1110	00...0011
0111 0100	00...0100
1011 0010	00...0101
0010 1000	00...0110
1110 1111	00...0111
.	.
.	.
.	.
0110 0010	11...1110
0010 0001	11...1111



Bộ nhớ đệm (Cache memory)

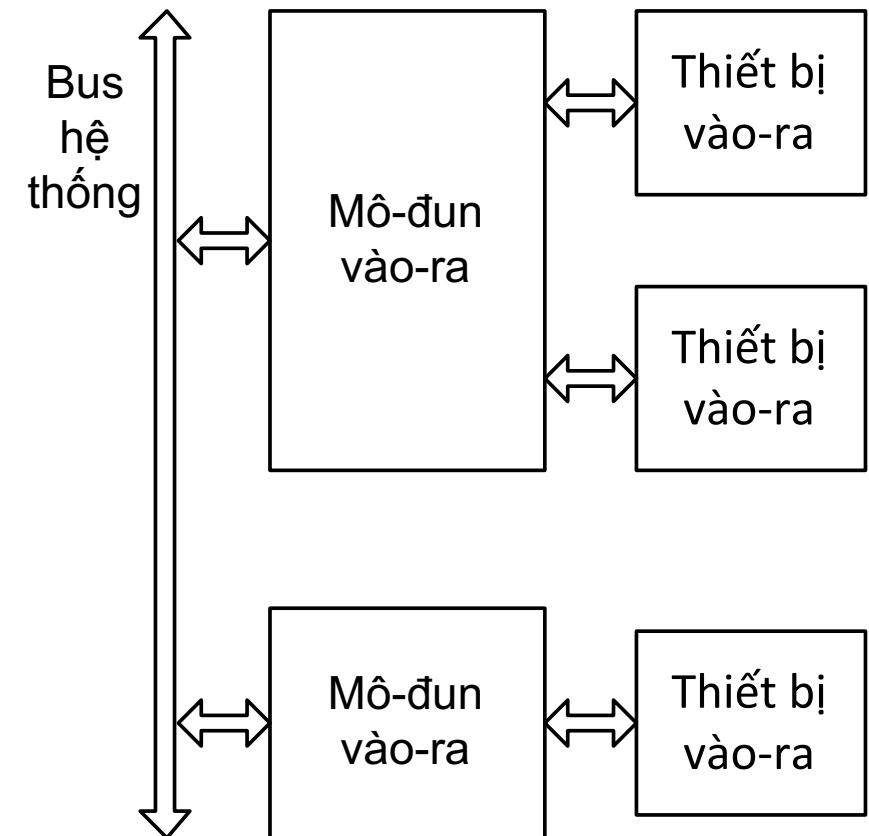
- Bộ nhớ có tốc độ nhanh được đặt đệm giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ
- Dung lượng nhỏ hơn bộ nhớ chính
- Sử dụng bộ nhớ bán dẫn tốc độ nhanh
- Cache thường được chia thành một số mức (L1, L2, L3)
- Cache thường được tích hợp trên cùng chip bộ xử lý
- Cache có thể có hoặc không

Thiết bị lưu trữ (Storage Devices)

- Còn được gọi là bộ nhớ ngoài
- Chức năng và đặc điểm
 - Lưu giữ tài nguyên phần mềm của máy tính
 - Được kết nối với hệ thống dưới dạng các thiết bị vào-ra
 - Dung lượng lớn
 - Tốc độ chậm
- Các loại thiết bị lưu trữ
 - Bộ nhớ từ: Ổ đĩa cứng HDD
 - Bộ nhớ bán dẫn: Ổ thẻ rắn SSD, Ổ nhớ flash, thẻ nhớ
 - Bộ nhớ quang: CD, DVD

3. Hệ thống vào-ra

- Chức năng: Trao đổi thông tin giữa máy tính với thế giới bên ngoài
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị vào-ra (IO devices)
 - Các mô-đun vào-ra (IO modules)



Các thiết bị vào-ra

- Còn được gọi là thiết bị ngoại vi (Peripherals)
- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Các loại thiết bị vào-ra:
 - Thiết bị vào (Input Devices)
 - Thiết bị ra (Output Devices)
 - Thiết bị lưu trữ (Storage Devices)
 - Thiết bị truyền thông (Communication Devices)

Mô-đun vào-ra

- Chức năng: nối ghép các thiết bị vào-ra với máy tính
- Mỗi mô-đun vào-ra có một hoặc một vài cổng vào-ra (I/O Port)
- Mỗi cổng vào-ra được đánh một địa chỉ xác định
- Các thiết bị vào-ra được kết nối và trao đổi dữ liệu với máy tính thông qua các cổng vào-ra
- CPU muốn trao đổi dữ liệu với thiết bị vào-ra, cần phải biết địa chỉ của cổng vào-ra tương ứng

1.2. Hoạt động cơ bản của máy tính

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

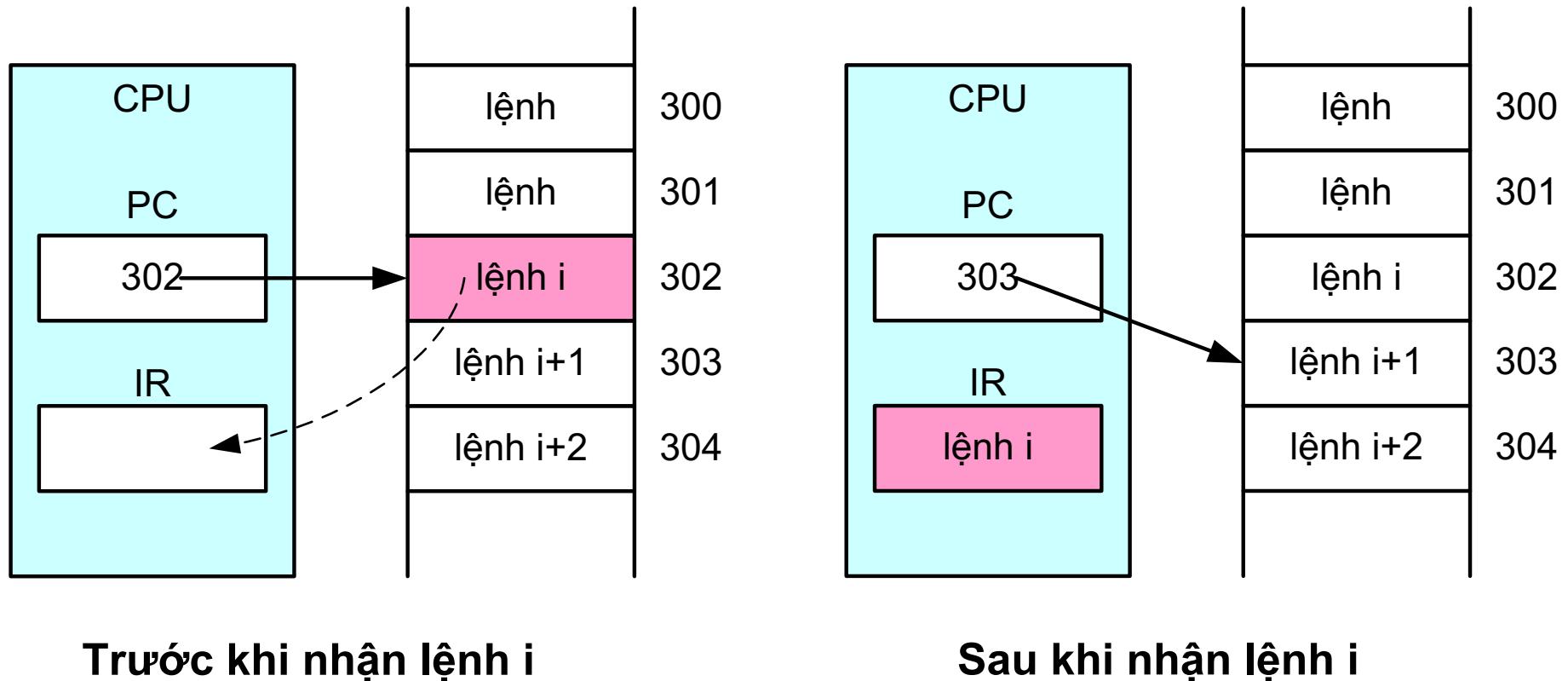
1. Thực hiện chương trình

- Là hoạt động cơ bản của máy tính
- Máy tính lặp đi lặp lại chu trình lệnh gồm hai bước:
 - Nhận lệnh
 - Thực hiện lệnh
- Hoạt động thực hiện chương trình bị dừng nếu:
 - Thực hiện lệnh bị lỗi
 - Gặp lệnh dừng
 - Tắt máy

Nhận lệnh

- Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính
- **Bộ đếm chương trình PC** (Program Counter) là thanh ghi của CPU dùng để giữ địa chỉ của lệnh sẽ được nhận vào
- CPU phát ra địa chỉ từ bộ đếm chương trình PC tìm ra ngăn nhớ chứa lệnh
- Lệnh được đọc từ bộ nhớ đưa vào thanh ghi lệnh IR (Instruction Register)
- Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trả đến lệnh kế tiếp.

Minh họa quá trình nhận lệnh



Thực hiện lệnh

- Bộ xử lý giải mã lệnh đã được nhận và phát tín hiệu điều khiển thực hiện thao tác mà lệnh yêu cầu
- Các kiểu thao tác cơ bản của lệnh:
 - Trao đổi dữ liệu giữa CPU với bộ nhớ chính hoặc CPU với mô-đun vào-ra
 - Thực hiện các phép toán số học hoặc phép toán logic với các dữ liệu
 - Chuyển điều khiển trong chương trình: rẽ nhánh hoặc nhảy đến vị trí khác

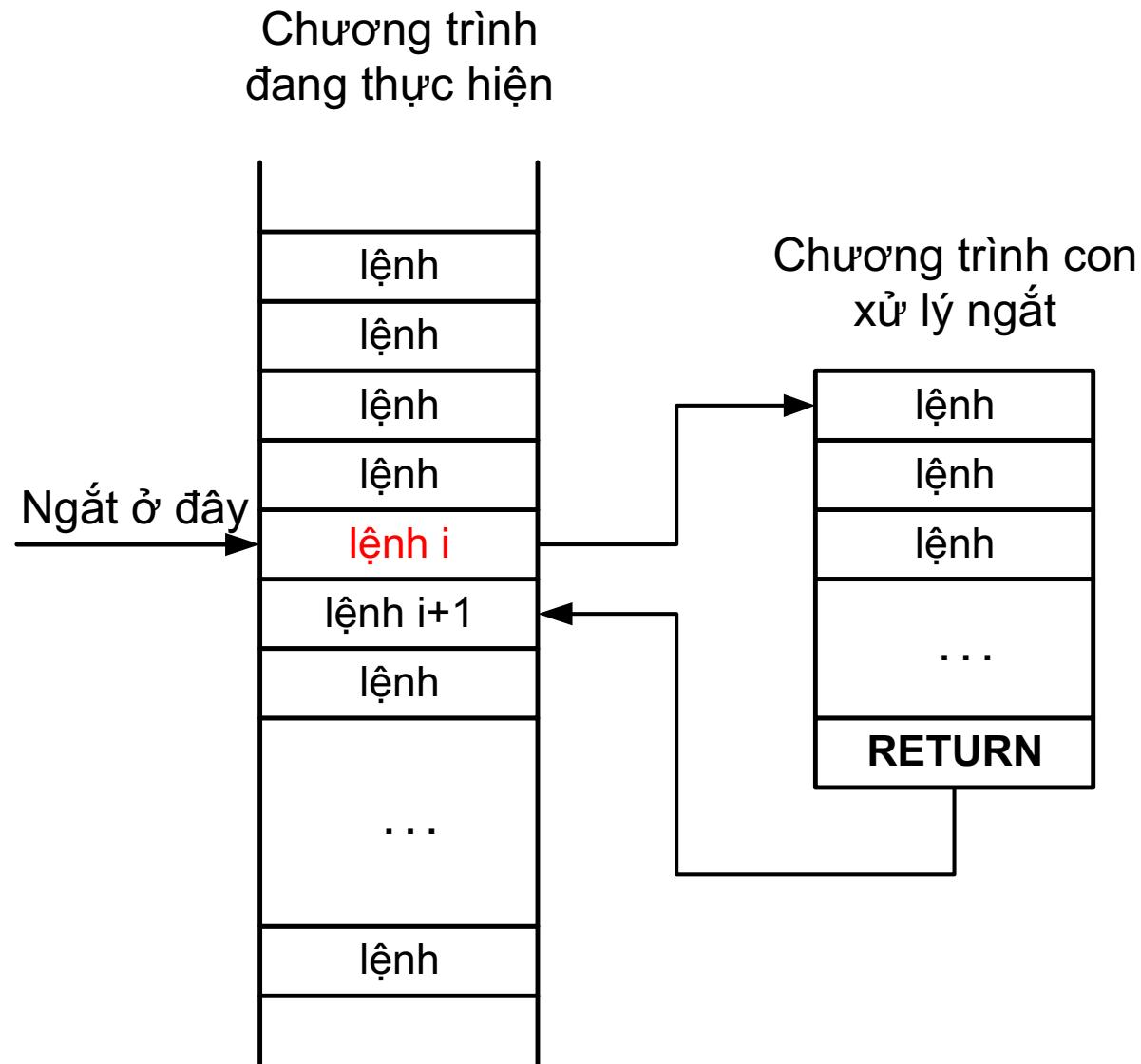
2. Ngắt (Interrupt)

- **Khái niệm chung về ngắt:** Ngắt là cơ chế cho phép CPU tạm dừng chương trình đang thực hiện để chuyển sang thực hiện một chương trình con có sẵn trong bộ nhớ.
 - **Chương trình con xử lý ngắt (Interrupt handlers)**
- **Các loại ngắt:**
 - **Biệt lệ (exception):** gây ra do lỗi khi thực hiện chương trình (VD: tràn số, mã lệnh sai, ...)
 - **Ngắt từ bên ngoài (external interrupt):** do thiết bị vào-ra (qua mô-đun vào-ra) gửi tín hiệu ngắt đến CPU để yêu cầu trao đổi dữ liệu

Hoạt động với ngắt từ bên ngoài

- Sau khi hoàn thành mỗi một lệnh, bộ xử lý kiểm tra tín hiệu ngắt
 - Nếu không có ngắt, bộ xử lý nhận lệnh tiếp theo của chương trình hiện tại
 - Nếu có tín hiệu ngắt:
 - Tạm dừng (suspend) chương trình đang thực hiện
 - Cắt ngũ cảnh (các thông tin liên quan đến chương trình bị ngắt)
 - Thiết lập bộ đếm chương trình PC trả đến chương trình con xử lý ngắt tương ứng
 - Chuyển sang thực hiện chương trình con xử lý ngắt
 - Khôi phục ngũ cảnh và trở về tiếp tục thực hiện chương trình đang bị tạm dừng

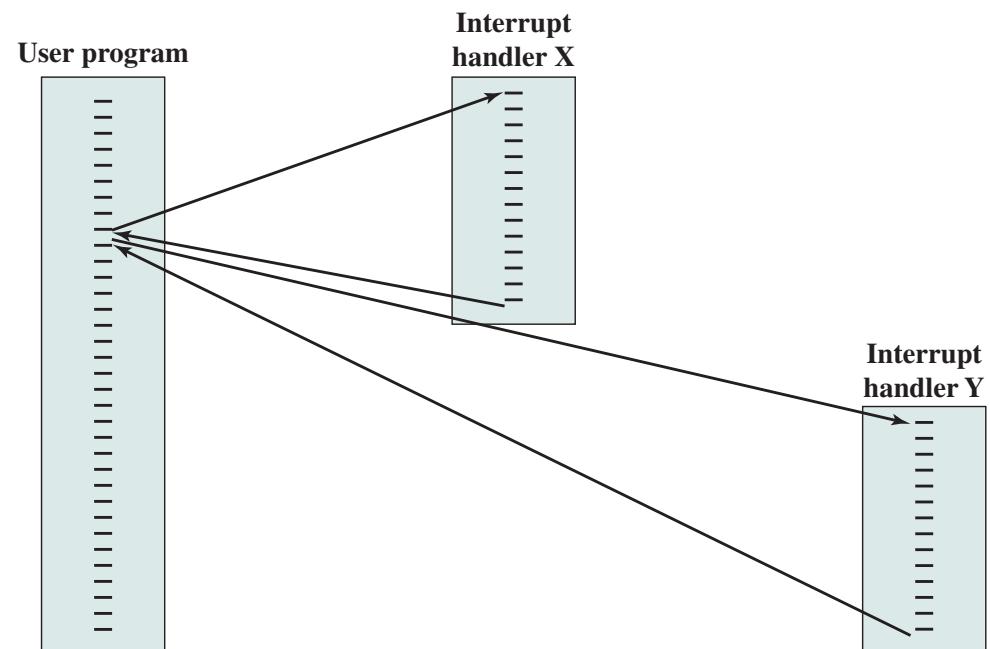
Hoạt động ngắt (tiếp)



Xử lý với nhiều tín hiệu yêu cầu ngắt

Xử lý ngắt tuần tự

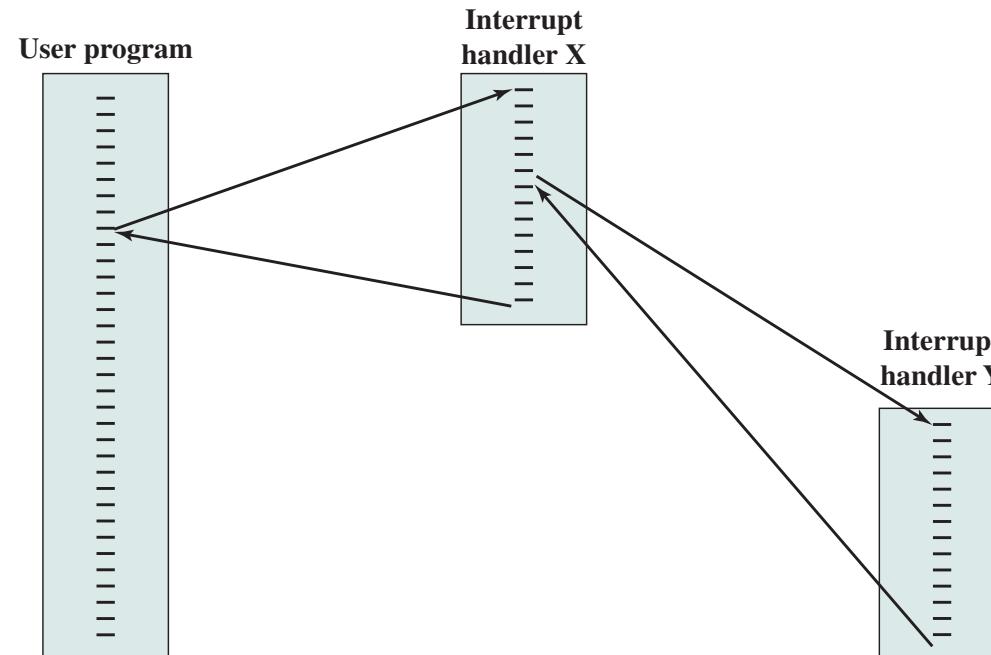
- Khi một ngắt đang được thực hiện, các ngắt khác bị cấm (disabled interrupt)
- Bộ xử lý sẽ bỏ qua các yêu cầu ngắt tiếp theo
- Các yêu cầu ngắt tiếp theo vẫn đang đợi và được kiểm tra sau khi ngắt hiện tại được xử lý xong
- Các ngắt được thực hiện tuần tự



Xử lý với nhiều tín hiệu yêu cầu ngắt (tiếp)

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 có mức ưu tiên cao hơn
- Xảy ra ngắt lồng nhau



3. Hoạt động vào-ra

- Hoạt động vào-ra: là hoạt động trao đổi dữ liệu giữa mô-đun vào-ra với bên trong máy tính.
- Các kiểu hoạt động vào-ra:
 - CPU trao đổi dữ liệu với mô-đun vào-ra bởi lệnh vào-ra trong chương trình
 - CPU trao quyền điều khiển cho phép mô-đun vào-ra trao đổi dữ liệu trực tiếp với bộ nhớ chính (DMA - Direct Memory Access).

1.3. Bus máy tính

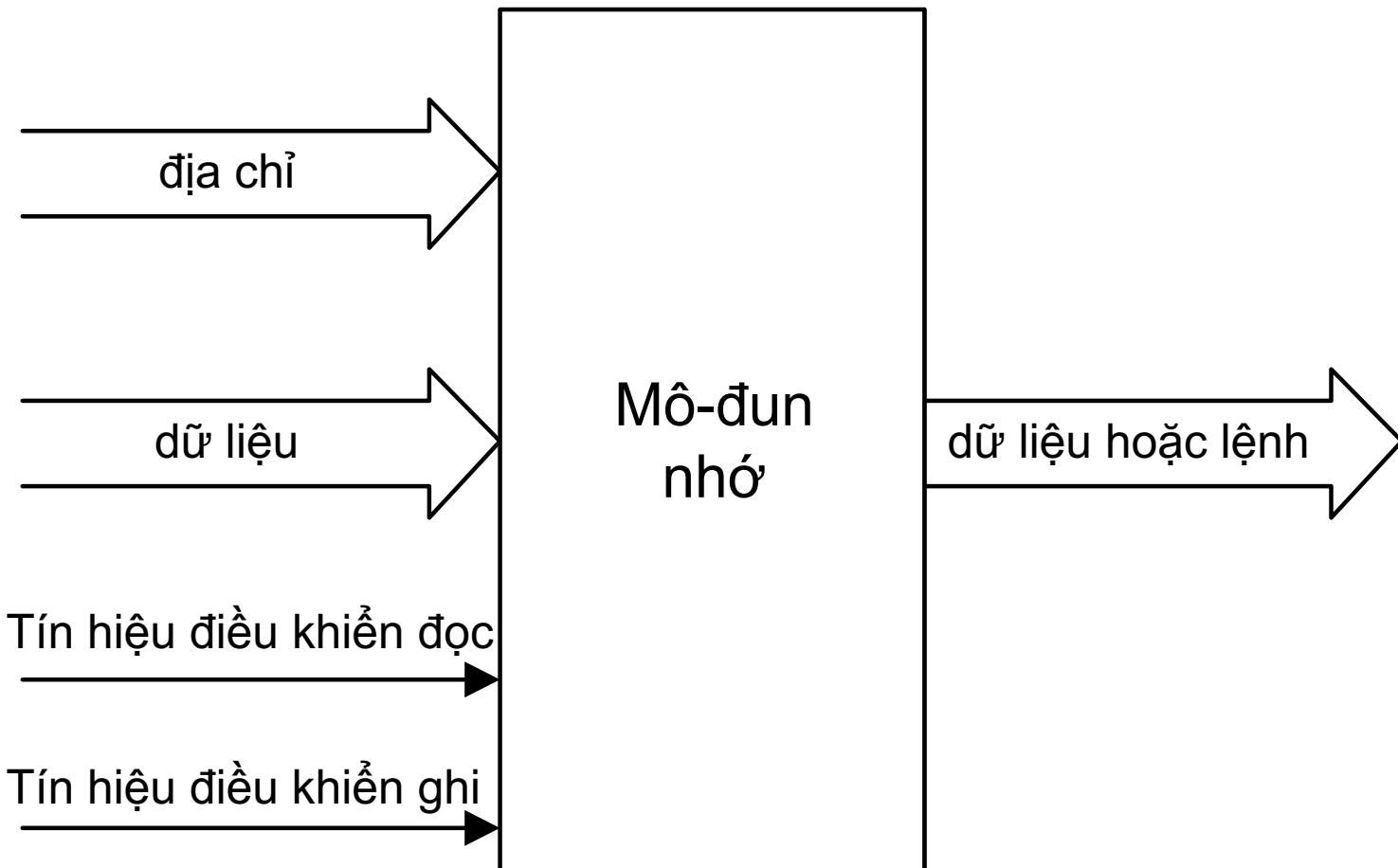
1. Luồng thông tin trong máy tính

- Các mô-đun trong máy tính:

- CPU
- Mô-đun nhớ
- Mô-đun vào-ra

→ cần được kết nối với nhau

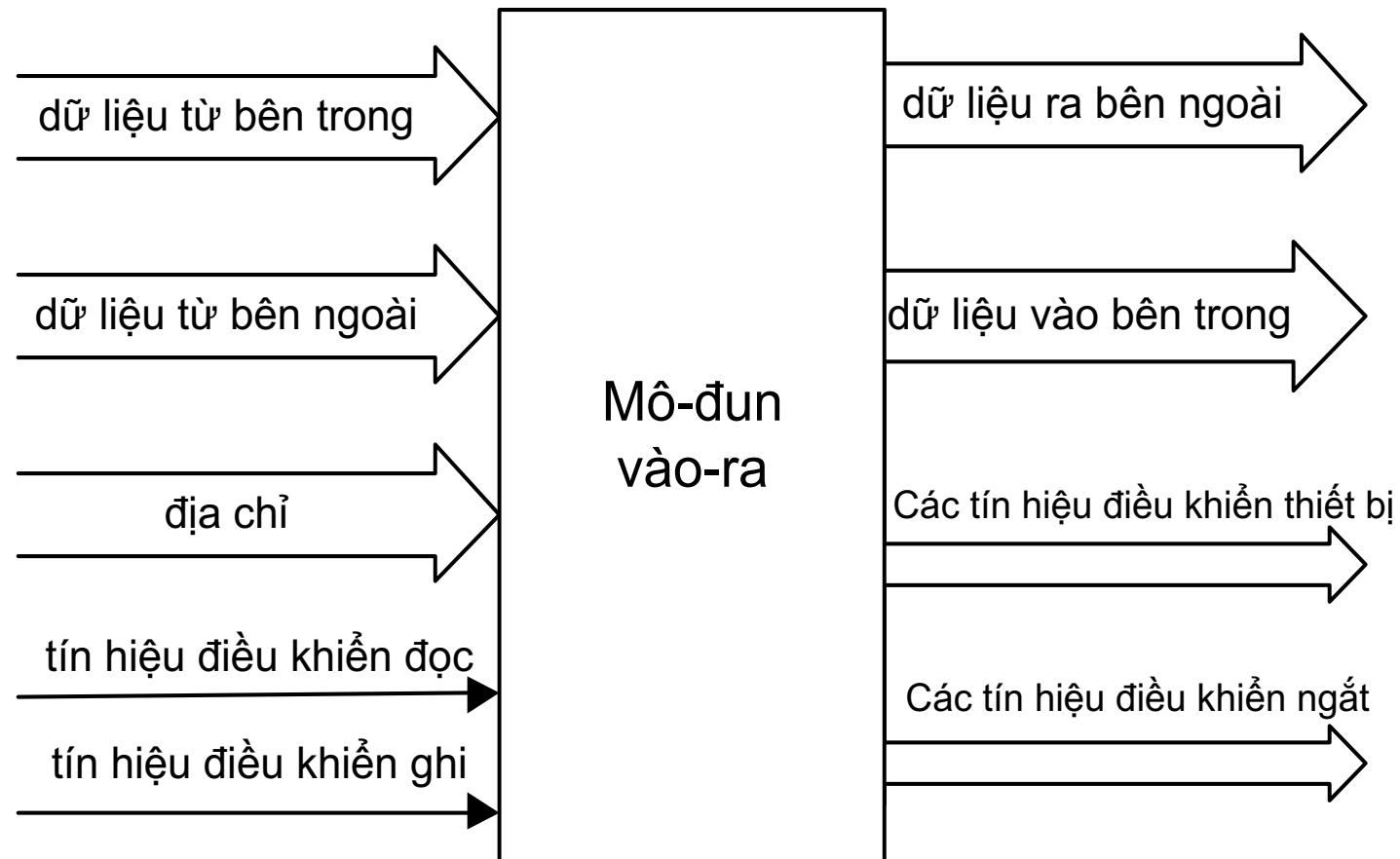
Kết nối mô-đun nhớ



Kết nối mô-đun nhớ (tiếp)

- Địa chỉ đưa đến để xác định ngăn nhớ
- Dữ liệu được đưa đến khi ghi
- Dữ liệu hoặc lệnh được đưa ra khi đọc
 - Bộ nhớ không phân biệt lệnh và dữ liệu
- Nhận các tín hiệu điều khiển:
 - Điều khiển đọc (Read)
 - Điều khiển ghi (Write)

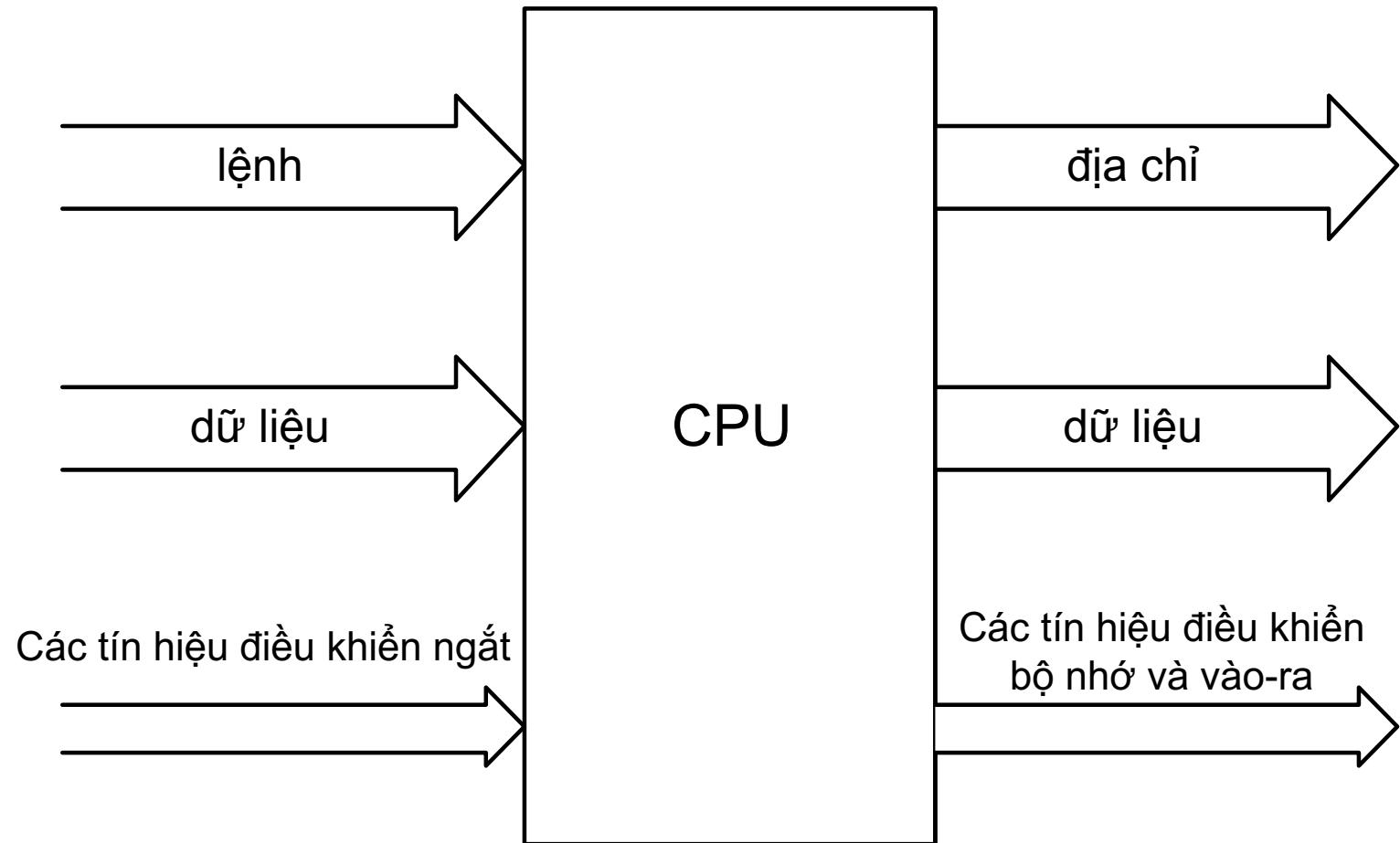
Kết nối mô-đun vào-ra



Kết nối mô-đun vào-ra (tiếp)

- Địa chỉ đưa đến để xác định cổng vào-ra
- Ra dữ liệu (Output)
 - Nhận dữ liệu từ bên trong (CPU hoặc bộ nhớ chính)
 - Đưa dữ liệu ra thiết bị vào-ra
- Vào dữ liệu (Input)
 - Nhận dữ liệu từ thiết bị vào-ra
 - Đưa dữ liệu vào bên trong (CPU hoặc bộ nhớ chính)
- Nhận các tín hiệu điều khiển từ CPU
- Phát các tín hiệu điều khiển đến thiết bị vào-ra
- Phát các tín hiệu ngắn đến CPU

Kết nối CPU



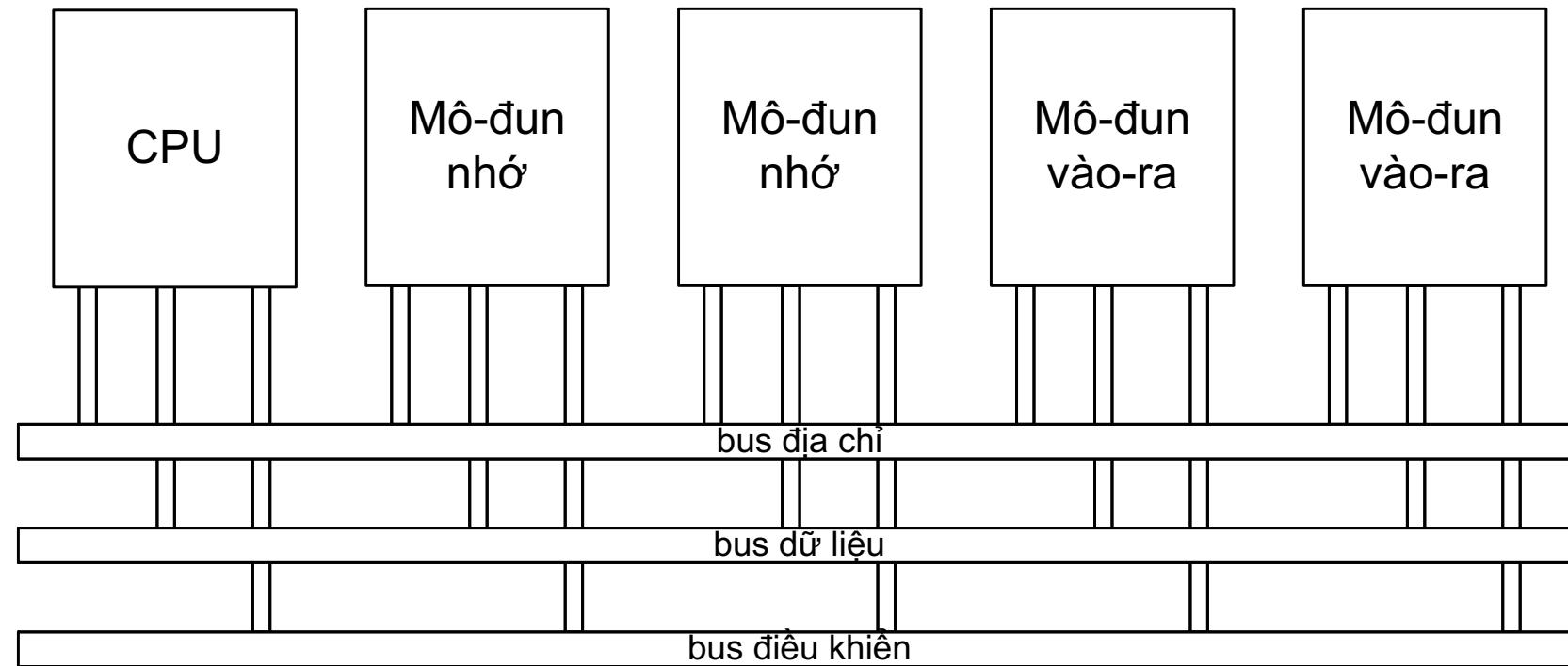
Kết nối CPU (tiếp)

- Phát địa chỉ đến các mô-đun nhớ hay các mô-đun vào-ra
- Đọc lệnh từ bộ nhớ
- Đọc dữ liệu từ bộ nhớ hoặc mô-đun vào-ra
- Đưa dữ liệu ra (sau khi xử lý) đến bộ nhớ hoặc mô-đun vào-ra
- Phát tín hiệu điều khiển đến các mô-đun nhớ và các mô-đun vào-ra
- Nhận các tín hiệu ngắt

2. Cấu trúc bus cơ bản

- **Bus:** tập hợp các đường kết nối để vận chuyển thông tin giữa các mô-đun của máy tính với nhau.
- **Các bus chức năng:**
 - Bus địa chỉ (Address bus)
 - Bus dữ liệu (Data bus)
 - Bus điều khiển (Control bus)
- **Độ rộng bus:** là số đường dây của bus có thể truyền các bit thông tin đồng thời (chỉ dùng cho bus địa chỉ và bus dữ liệu)

Sơ đồ cấu trúc bus cơ bản

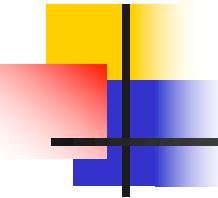


Bus địa chỉ

- Chức năng: vận chuyển địa chỉ để xác định vị trí ngăn nhớ hay cổng vào-ra
- Độ rộng bus địa chỉ:
 - N bit: $A_{N-1}, A_{N-2}, \dots, A_2, A_1, A_0$
→ Số lượng địa chỉ tối đa được sử dụng là: 2^N địa chỉ (gọi là không gian địa chỉ)
 - Địa chỉ nhỏ nhất: 00 ... 000₍₂₎
 - Địa chỉ lớn nhất: 11 ... 111₍₂₎
- Ví dụ:
 - Máy tính sử dụng bus địa chỉ 32-bit ($A_{31}-A_0$), bộ nhớ chính được đánh địa chỉ cho từng byte
→ Có khả năng đánh địa chỉ cho 2^{32} bytes nhớ = 4GiB

Bus dữ liệu

- Chức năng:
 - vận chuyển lệnh từ bộ nhớ đến CPU
 - vận chuyển dữ liệu giữa các thành phần của máy tính với nhau
- Độ rộng bus dữ liệu: số bit được truyền đồng thời
 - M bit: $D_{M-1}, D_{M-2}, \dots, D_2, D_1, D_0$
 - M thường là 8, 16, 32, 64 bit
- Ví dụ:
 - Máy tính có bus dữ liệu kết nối CPU với bộ nhớ là 64-bit
→ Có thể trao đổi 8 byte nhớ ở một thời điểm



Bus điều khiển

- Chức năng: vận chuyển các tín hiệu điều khiển
- Các loại tín hiệu điều khiển:
 - Các tín hiệu điều khiển đọc/ghi
 - Các tín hiệu điều khiển ngắn
 - Các tín hiệu điều khiển bus

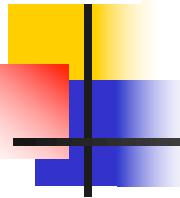
Một số tín hiệu điều khiển hiển thị

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

Một số tín hiệu điều khiển hiển thị (tiếp)

■ 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 để trao đổi vào-ra.
- *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.



Một số tín hiệu điều khiển hiển thị hình (tiếp)

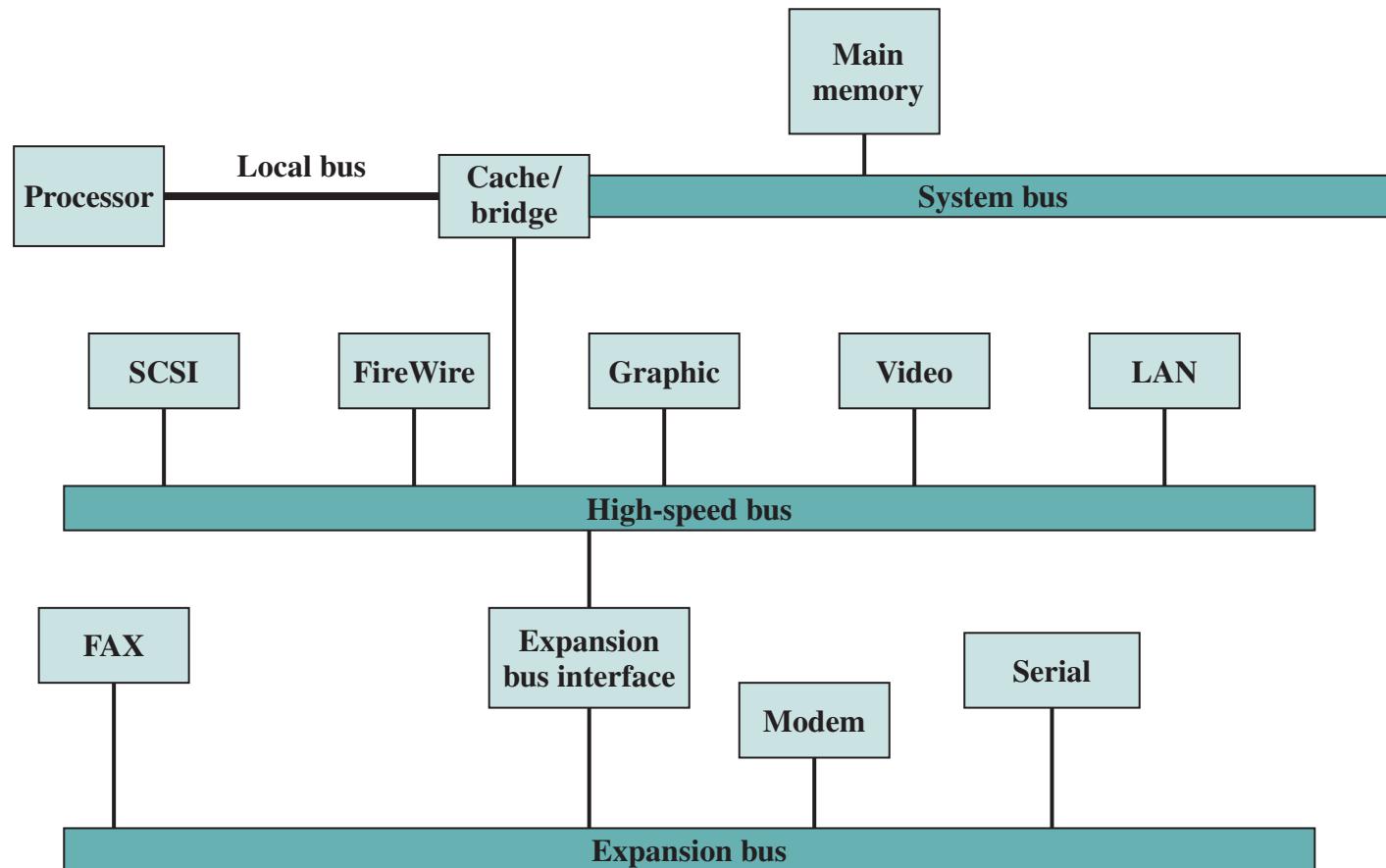
- Các tín hiệu điều khiển bus:

- *Bus Request (BRQ)* : Tín hiệu từ mô-đun vào-ra gửi đến yêu cầu CPU chuyển nhượng quyền sử dụng bus.
- *Bus Grant (BGT)*: Tín hiệu phát ra từ CPU chấp nhận chuyển nhượng quyền sử dụng bus cho mô-đun vào-ra.
- *Lock/ Unlock*: Tín hiệu *cấm/cho-phép* xin chuyển nhượng bus.

3. Phân cấp bus

- Đơn bus: Tất cả các mô-đun kết nối vào bus chung
 - Bus chỉ phục vụ được một yêu cầu trao đổi dữ liệu tại một thời điểm → độ trễ lớn
 - Bus phải có tốc độ bằng tốc độ bus của mô-đun nhanh nhất trong hệ thống
- Đa bus: Phân cấp thành nhiều bus cho các mô-đun khác nhau và có tốc độ khác nhau
 - Bus của bộ xử lý
 - Bus của RAM
 - Các bus vào-ra

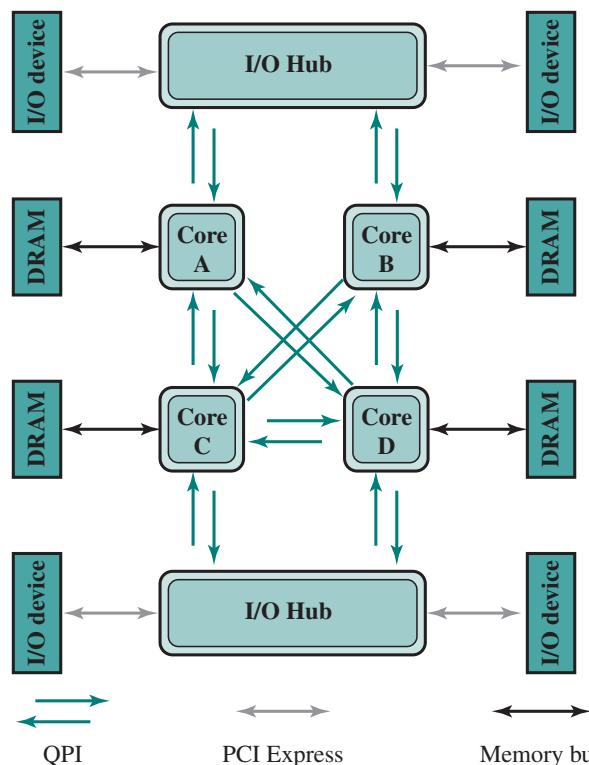
Phân cấp bus



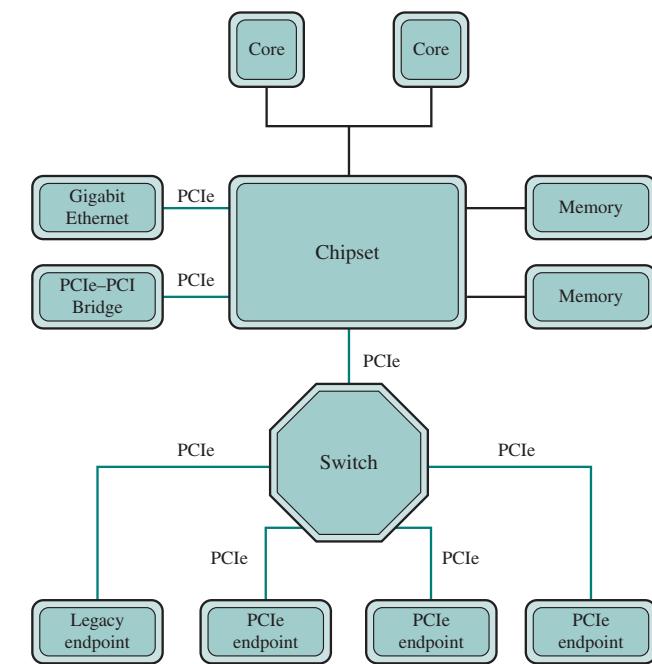
4. Kết nối điểm-điểm

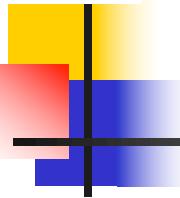
- Point-to-point connection
- Khắc phục nhược điểm của bus dùng chung (shared bus)

Kết nối QPI



Kết nối PCIe

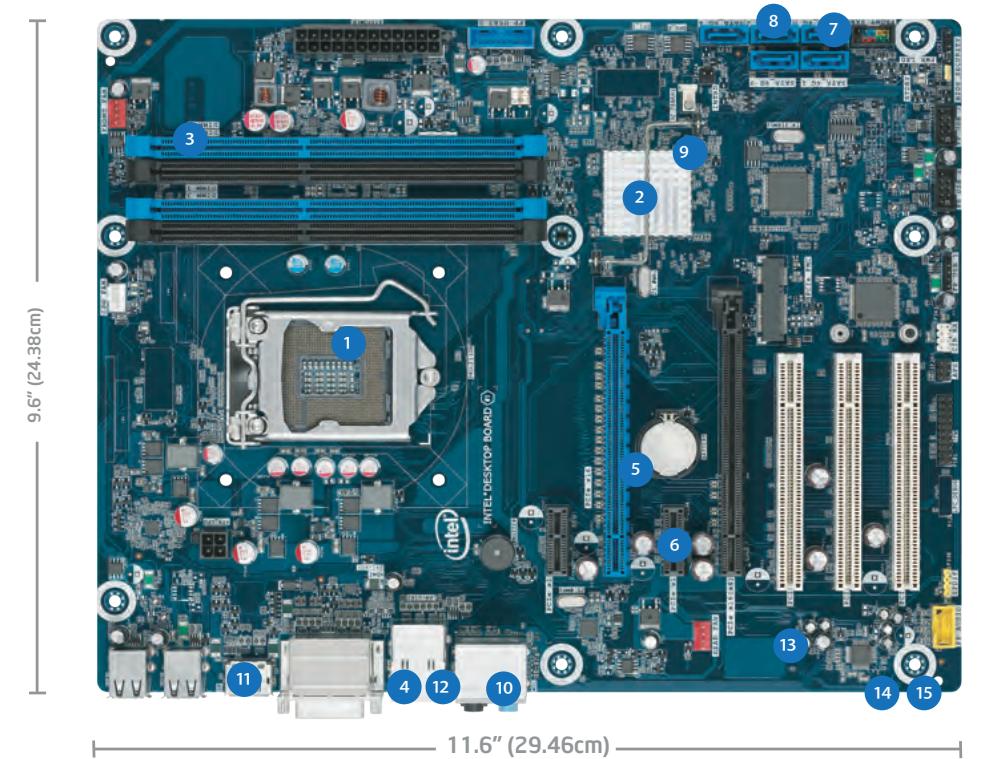
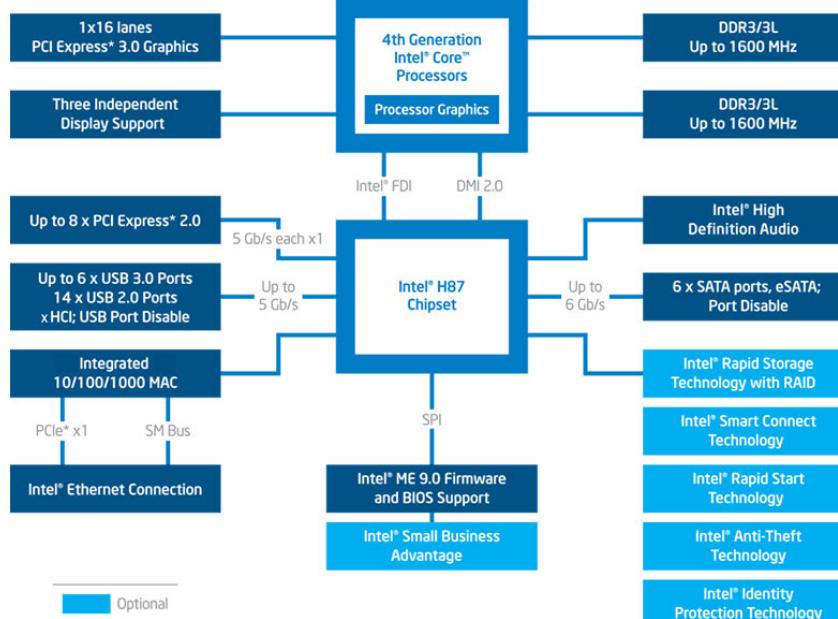




Một số bus điển hình trong máy tính

- QPI (Quick Path Interconnect)
- PCI bus (Peripheral Component Interconnect):
bus vào-ra đa năng
- PCIe: (PCI express) kết nối điểm-điểm đa năng
tốc độ cao
- SATA (Serial Advanced Technology Attachment):
Bus kết nối với ổ đĩa cứng hoặc ổ đĩa CD/DVD
- USB (Universal Serial Bus): Bus nối tiếp đa năng

Ví dụ bus trong máy tính Intel



HDMI®
HIGH-DEFINITION MULTIMEDIA INTERFACE

1.4. Hiệu năng máy tính

- Định nghĩa hiệu năng P (Performance):

Hiệu năng = 1/(thời gian thực hiện)

hay là: **$P = 1/t$**

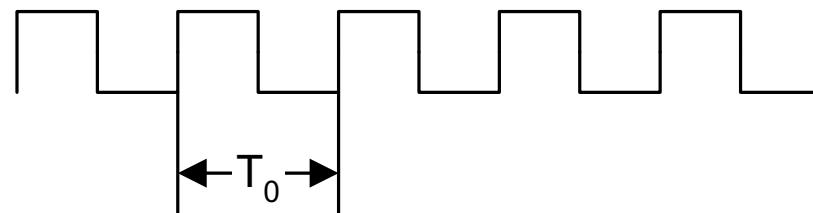
“Máy tính A nhanh hơn máy B k lần”

$$P_A / P_B = t_B / t_A = k$$

- Ví dụ: Thời gian chạy chương trình:
 - 10s trên máy A, 15s trên máy B
 - $t_B / t_A = 15s / 10s = 1.5$
 - Vậy máy A nhanh hơn máy B 1.5 lần

Tốc độ xung nhịp của CPU

- Về mặt thời gian, CPU hoạt động theo một xung nhịp (clock) có tốc độ xác định



- **Chu kỳ xung nhịp T_0** (Clock period): thời gian của một chu kỳ
- **Tốc độ xung nhịp f_0** (Clock rate) hay là **Tần số xung nhịp**: số chu kỳ trong 1s
 - $f_0 = 1/T_0$
- VD: Bộ xử lý có $f_0 = 4\text{GHz} = 4 \times 10^9\text{Hz}$
$$T_0 = 1/(4 \times 10^9) = 0.25 \times 10^{-9}\text{s} = 0.25\text{ns}$$

Thời gian thực hiện của CPU

- Để đơn giản, ta xét thời gian CPU thực hiện chương trình (CPU time):

Thời gian thực hiện của CPU =

Số chu kỳ xung nhịp x Thời gian một chu kỳ

$$t_{CPU} = n \times T_0 = \frac{n}{f_0}$$

n: số chu kỳ xung nhịp

- Hiệu năng được tăng lên bằng cách:
 - Giảm số chu kỳ xung nhịp n
 - Tăng tốc độ xung nhịp f_0

Ví dụ

- Hai máy tính A và B cùng chạy một chương trình
- Máy tính A:
 - Tốc độ xung nhịp của CPU: $f_A = 2\text{GHz}$
 - Thời gian CPU thực hiện chương trình: $t_A = 10\text{s}$
- Máy tính B:
 - Thời gian CPU thực hiện chương trình: $t_B = 6\text{s}$
 - Số chu kỳ xung nhịp khi chạy chương trình trên máy B (n_B) nhiều hơn 1.2 lần số chu kỳ xung nhịp khi chạy chương trình trên máy A (n_A)
- Hãy xác định tốc độ xung nhịp cần thiết cho máy B (f_B)?

Ví dụ (tiếp)

Ta có: $t = \frac{n}{f}$

Số chu kỳ xung nhịp khi chạy chương trình trên máy A:

$$n_A = t_A \times f_A = 10s \times 2GHz = 20 \times 10^9$$

Số chu kỳ xung nhịp khi chạy chương trình trên máy B:

$$n_B = 1.2 \times n_A = 24 \times 10^9$$

Tốc độ xung nhịp cần thiết cho máy B:

$$f_B = \frac{n_B}{t_B} = \frac{24 \times 10^9}{6} = 4 \times 10^9 Hz = 4GHz$$

Số lệnh và số chu kỳ trên một lệnh

Số chu kỳ xung nhịp của chương trình:

Số chu kỳ = Số lệnh của chương trình x Số chu kỳ trên một lệnh

$$n = IC \times CPI$$

- n - số chu kỳ xung nhịp
- IC - số lệnh của chương trình (Instruction Count)
- CPI - số chu kỳ trên một lệnh (Cycles per Instruction)

Vậy thời gian thực hiện của CPU:

$$t_{CPU} = IC \times CPI \times T_0 = \frac{IC \times CPI}{f_0}$$

Trong trường hợp các lệnh khác nhau có CPI khác nhau, cần tính CPI trung bình

Ví dụ

- Hai máy tính A và B có cùng kiến trúc tập lệnh
- Máy tính A có:
 - Chu kỳ xung nhịp: $T_A = 250\text{ps}$
 - Số chu kỳ/ lệnh trung bình: $CPI_A = 2.0$
- Máy tính B:
 - Chu kỳ xung nhịp: $T_B = 500\text{ps}$
 - Số chu kỳ/ lệnh trung bình: $CPI_B = 1.2$
- Hãy xác định máy nào nhanh hơn và nhanh hơn bao nhiêu ?

Ví dụ (tiếp)

Ta có: $t_{CPU} = IC \times CPI_{TB} \times T_0$

Hai máy cùng kiến trúc tập lệnh, vì vậy số lệnh của cùng một chương trình trên hai máy là bằng nhau:

$$IC_A = IC_B = IC$$

Thời gian thực hiện chương trình đó trên máy A và máy B:

$$t_A = IC_A \times CPI_A \times T_A = IC \times 2.0 \times 250 ps = IC \times 500 ps$$

$$t_B = IC_B \times CPI_B \times T_B = IC \times 1.2 \times 500 ps = IC \times 600 ps$$

Từ đó ta có: $\frac{t_B}{t_A} = \frac{IC \times 600 ps}{IC \times 500 ps} = 1.2$

Kết luận: máy A nhanh hơn máy B 1.2 lần

CPI trung bình

- Nếu loại lệnh khác nhau có số chu kỳ khác nhau, ta có tổng số chu kỳ:

$$n = \sum_{i=1}^K (CPI_i \times IC_i)$$

- CPI trung bình:

$$CPI_{TB} = \frac{n}{IC} = \frac{1}{IC} \sum_{i=1}^K (CPI_i \times IC_i)$$

Ví dụ

- Cho bảng chỉ ra các dãy lệnh sử dụng các lệnh thuộc các loại A, B, C. Tính CPI trung bình?

Loại lệnh	A	B	C
CPI theo loại lệnh	1	2	3
IC trong dãy lệnh 1	20	10	20
IC trong dãy lệnh 2	40	10	10

Ví dụ

- Cho bảng chỉ ra các dãy lệnh sử dụng các lệnh thuộc các loại A, B, C. Tính CPI trung bình?

Loại lệnh	A	B	C
CPI theo loại lệnh	1	2	3
IC trong dãy lệnh 1	20	10	20
IC trong dãy lệnh 2	40	10	10

- Dãy lệnh 1: Số lệnh = 50
 - Số chu kỳ =
 $= 1 \times 20 + 2 \times 10 + 3 \times 20 = 100$
 - $\text{CPI}_{\text{TB}} = 100/50 = 2.0$
- Dãy lệnh 2: Số lệnh = 60
 - Số chu kỳ =
 $= 1 \times 40 + 2 \times 10 + 3 \times 10 = 90$
 - $\text{CPI}_{\text{TB}} = 90/60 = 1.5$

Tóm tắt về Hiệu năng

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

*Thời gian CPU = Số lệnh của chương trình x Số chu kỳ/lệnh
x Số giây của một chu kỳ*

$$t_{CPU} = IC \times CPI \times T_0 = \frac{IC \times CPI}{f_0}$$

■ Hiệu năng phụ thuộc vào:

- Thuật giải
- Ngôn ngữ lập trình
- Chương trình dịch
- Kiến trúc tập lệnh
- Phần cứng

MIPS như là thước đo hiệu năng

- MIPS: Millions of Instructions Per Second
(Số triệu lệnh trên 1 giây)

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

$$\text{MIPS} = \frac{f_0}{\text{CPI} \times 10^6}$$

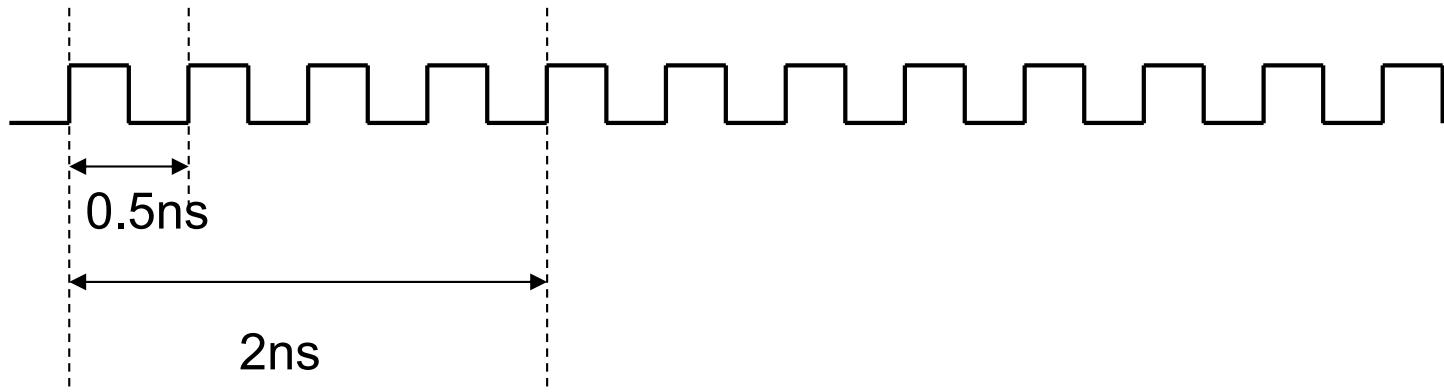
$$\text{CPI} = \frac{f_0}{\text{MIPS} \times 10^6}$$

Ví dụ

Tính MIPS của bộ xử lý với:
clock rate = 2GHz và CPI = 4

Ví dụ

Tính MIPS của bộ xử lý với:
clock rate = 2GHz và CPI = 4



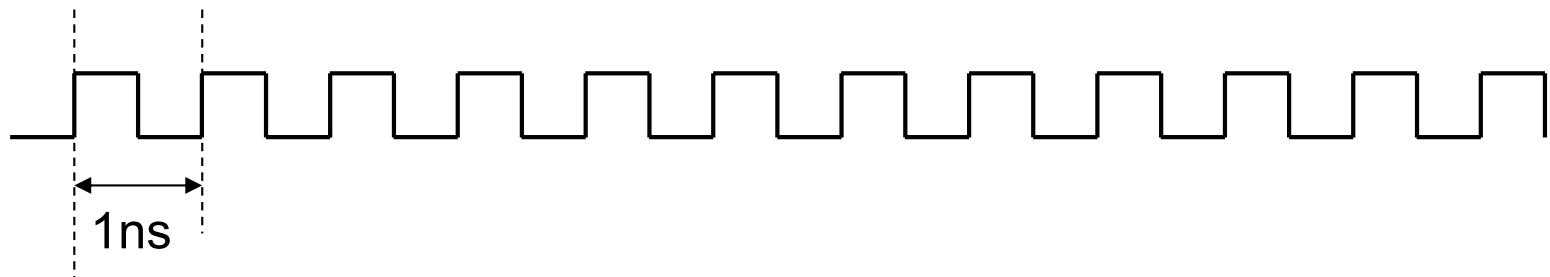
- Chu kỳ $T_0 = 1/(2 \times 10^9) = 0.5\text{ns}$
- CPI = 4 \rightarrow thời gian thực hiện 1 lệnh = $4 \times 0.5\text{ns} = 2\text{ns}$
- Số lệnh thực hiện trong 1s = $(10^9\text{ns})/(2\text{ns}) = 5 \times 10^8$ lệnh
- Vậy bộ xử lý thực hiện được 500 MIPS

Ví dụ

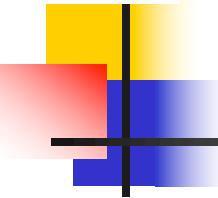
Tính CPI của bộ xử lý với:
clock rate = 1GHz và 400 MIPS

Ví dụ

Tính CPI của bộ xử lý với:
clock rate = 1GHz và 400 MIPS



- Chu kỳ $T_0 = 1/10^9 = 1\text{ns}$
- Số lệnh thực hiện trong 1 s là $400\text{MIPS} = 4 \times 10^8$ lệnh
- Thời gian thực hiện 1 lệnh $= 1/(4 \times 10^8)\text{s} = 2.5\text{ns}$
- Vậy ta có: CPI = 2.5



MFLOPS

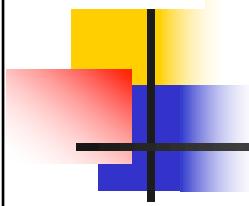
- Sử dụng cho các hệ thống tính toán lớn
- Millions of Floating Point Operations per Second
- Số triệu phép toán số dấu phẩy động trên một giây

$$\text{MFLOPS} = \frac{\text{Executed floating point operations}}{\text{Execution time} \times 10^6}$$

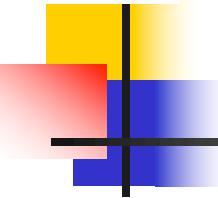
GFLOPS (10^9)

TFLOPS (10^{12})

PFLOPS (10^{15})



Hết chương 1

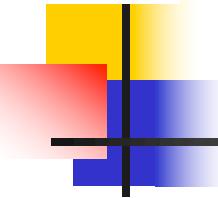


Hệ thống máy tính

Chương 2

BỘ NHỚ MÁY TÍNH

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội



Nội dung học phần

Chương 1. Tổng quan hệ thống máy tính

Chương 2. Bộ nhớ máy tính

Chương 3. Hệ thống vào-ra

Chương 4. Các kiến trúc song song

Nội dung của chương 2

- 2.1. Tổng quan hệ thống nhớ
- 2.2. Bộ nhớ chính
- 2.3. Bộ nhớ đệm (cache)
- 2.4. Bộ nhớ ngoài
- 2.5. Bộ nhớ ảo

2.1. Tổng quan hệ thống nhớ

1. Các đặc trưng của bộ nhớ

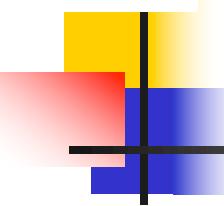
- Vị trí
 - Bên trong CPU:
 - tập thanh ghi
 - Bộ nhớ trong:
 - bộ nhớ chính
 - bộ nhớ đệm (cache)
 - Bộ nhớ ngoài:
 - các thiết bị lưu trữ
- Dung lượng
 - Độ dài từ nhớ (tính bằng bit)
 - Số lượng từ nhớ

Các đặc trưng của bộ nhớ (tiếp)

- Đơn vị truyỀn
 - Từ nhớ
 - Khối nhớ
- Phương pháp truy nhẬP
 - Truy nhẬP tuần tự (bĂng tÙ)
 - Truy nhẬP trực tiếp (cÁc loại đĩa)
 - Truy nhẬP ngẫu nhiên (bộ nhớ bán dẫn)
 - Truy nhẬP liên kết (cache)

Các đặc trưng của bộ nhớ (tiếp)

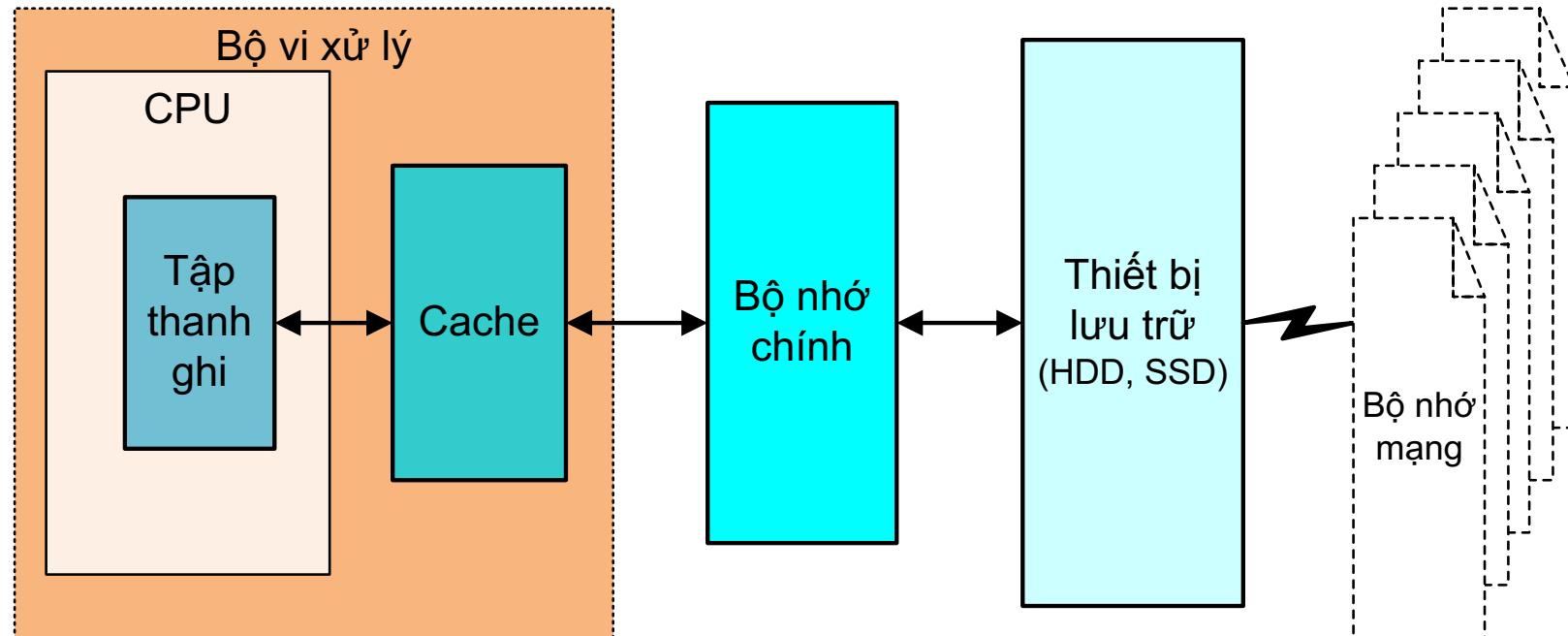
- Hiệu năng (performance)
 - Thời gian truy nhập
 - Chu kỳ nhớ
 - Tốc độ truyền
- Kiểu vật lý
 - Bộ nhớ bán dẫn
 - Bộ nhớ từ
 - Bộ nhớ quang



Các đặc trưng của bộ nhớ (tiếp)

- Các đặc tính vật lý
 - Khả biến / Không khả biến
(volatile / nonvolatile)
 - Xoá được / không xoá được
- Tổ chức

2. Phân cấp bộ nhớ



Từ trái sang phải:

- dung lượng tăng dần
- tốc độ giảm dần
- giá thành cùng dung lượng giảm dần

Công nghệ bộ nhớ

Công nghệ bộ nhớ	Thời gian truy nhập	Giá thành/GiB (2012)
SRAM	0,5 – 2,5 ns	\$500 – \$1000
DRAM	50 – 70 ns	\$10 – \$20
Flash memory	5.000 – 50.000 ns	\$0,75 – \$1
HDD	5 – 20 ms	\$0,05 – \$0,1

- Bộ nhớ lý tưởng
 - Thời gian truy nhập như SRAM
 - Dung lượng và giá thành như ổ đĩa cứng

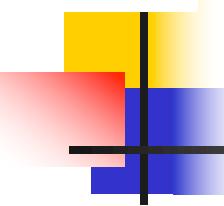
Nguyên lý cục bộ hoá tham chiếu bộ nhớ

- 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

2.2. Bộ nhớ chính

1. Bộ nhớ bán dẫn

Kiểu bộ nhớ	Tiêu chuẩn	Khả năng xoá	Cơ chế ghi	Tính khả biến
Read Only Memory (ROM)	Bộ nhớ chỉ đọc	Không xoá được	Mặt nạ	
Programmable ROM (PROM)				
Erasable PROM (EPROM)	Bộ nhớ hầu như chỉ đọc	bằng tia cực tím, cả chip		
Electrically Erasable PROM (EEPROM)		bằng điện, mức từng byte	Bằng điện	Không khả biến
Flash memory		bằng điện, từng khối		
Random Access Memory (RAM)	Bộ nhớ đọc-ghi	bằng điện, mức từng byte	Bằng điện	Khả biến



ROM (Read Only Memory)

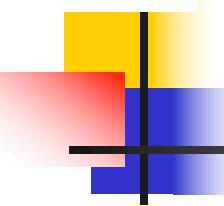
- Bộ nhớ không khả biến
- Lưu trữ các thông tin sau:
 - Thư viện các chương trình con
 - Các chương trình điều khiển hệ thống (BIOS)
 - Các bảng chức năng
 - Vi chương trình

Các kiểu ROM

- ROM mặt nạ:
 - thông tin được ghi khi sản xuất
- PROM (Programmable ROM)
 - Cần thiết bị chuyên dụng để ghi
 - Chỉ ghi được một lần
- EPROM (Erasable PROM)
 - Cần thiết bị chuyên dụng để ghi
 - Xóa được bằng tia tử ngoại
 - Ghi lại được nhiều lần
- EEPROM (Electrically Erasable PROM)
 - Có thể ghi theo từng byte
 - Xóa bằng điện

Bộ nhớ Flash

- Ghi theo khối
- Xóa bằng điện
- Dung lượng lớn

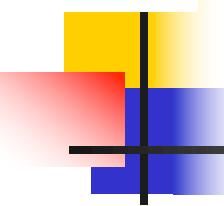


RAM (Random Access Memory)

- Bộ nhớ đọc-ghi (Read/Write Memory)
- Khả biến
- Lưu trữ thông tin tạm thời
- Có hai loại: SRAM và DRAM
(Static and Dynamic)

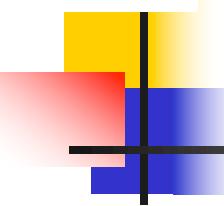
SRAM (Static) – RAM tĩnh

- Các bit được lưu trữ bằng các Flip-Flop
→ thông tin ổn định
- Cấu trúc phức tạp
- Dung lượng chip nhỏ
- Tốc độ nhanh
- Đắt tiền
- Dùng làm bộ nhớ cache



DRAM (Dynamic) – RAM động

- Các bit được lưu trữ trên tụ điện
→ cần phải có mạch làm tươi
- Cấu trúc đơn giản
- Dung lượng lớn
- Tốc độ chậm hơn
- Rẻ tiền hơn
- Dùng làm bộ nhớ chính

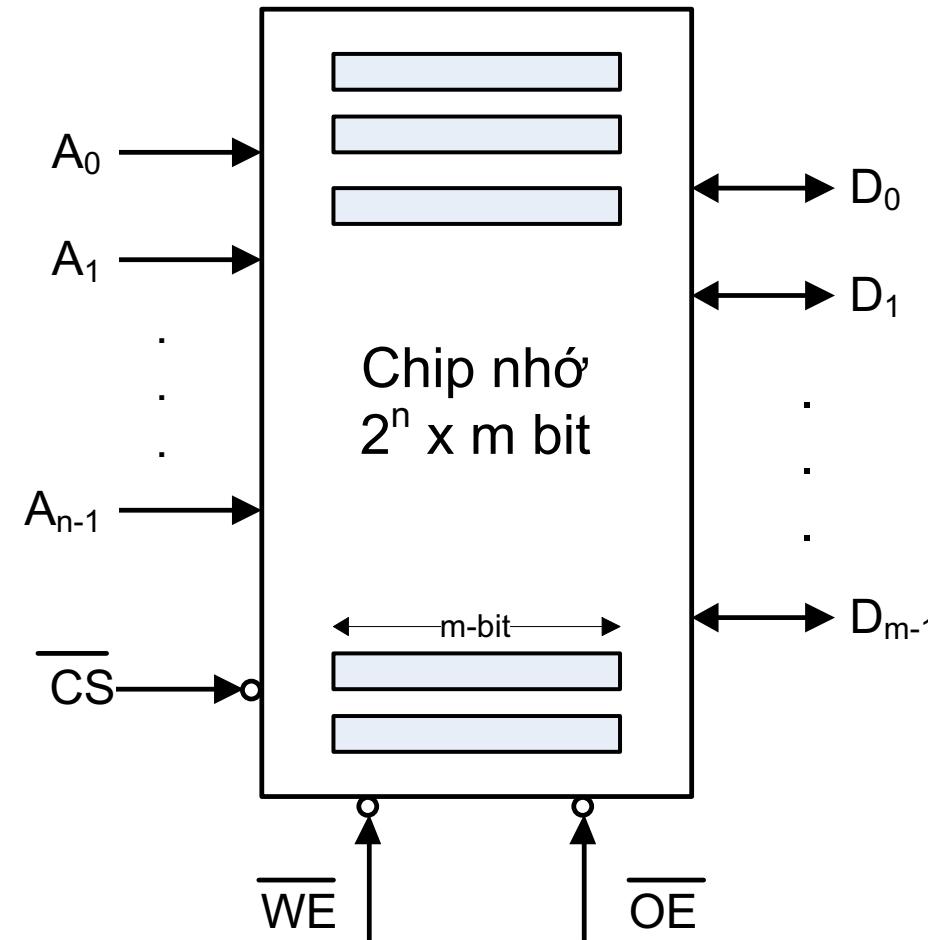


Một số DRAM tiên tiến thông dụng

- Cải tiến để tăng tốc độ
- Synchronous DRAM (SDRAM): làm việc được đồng bộ bởi xung clock
- DDR-SDRAM (Double Data Rate SDRAM)
- DDR3, DDR4

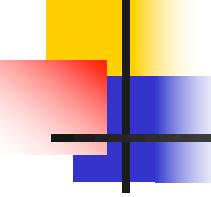
Tổ chức của chip nhớ

Sơ đồ cơ bản của chip nhớ



Các tín hiệu của chip nhớ

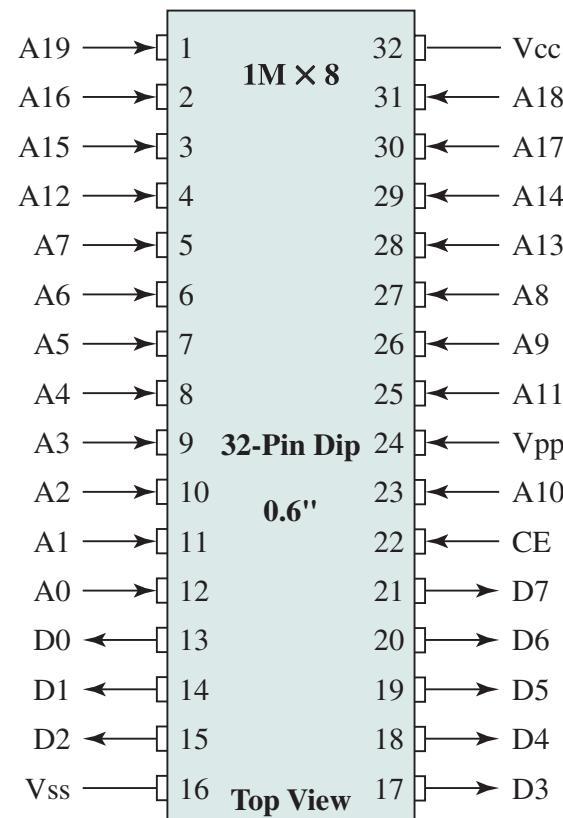
- Các đường địa chỉ: $A_{n-1} \div A_0 \rightarrow$ có 2^n từ nhớ
- Các đường dữ liệu: $D_{m-1} \div D_0 \rightarrow$ độ dài từ nhớ = m bit
- Dung lượng chip nhớ = $2^n \times m$ bit
- Các đường điều khiển:
 - Tín hiệu chọn chip CS (Chip Select)
 - Tín hiệu điều khiển đọc OE (Output Enable)
 - Tín hiệu điều khiển ghi WE (Write Enable)(Các tín hiệu điều khiển thường tích cực với mức 0)



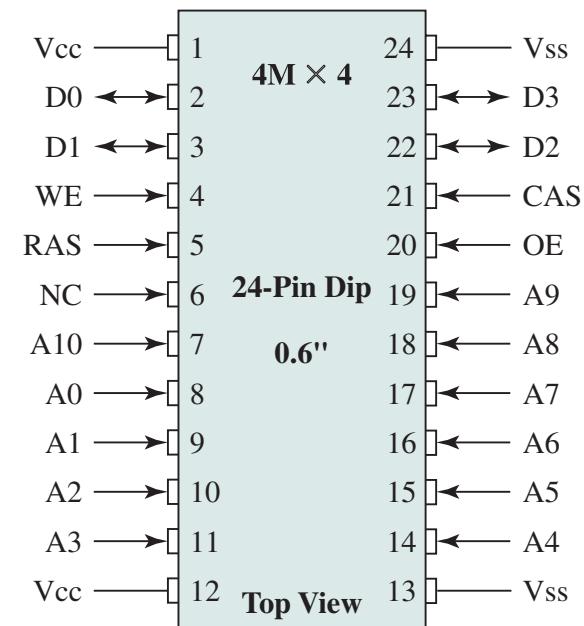
Tổ chức của DRAM

- Dùng n đường địa chỉ dòn kênh → cho phép truyền 2^n bit địa chỉ
- Tín hiệu chọn địa chỉ hàng RAS (Row Address Select)
- Tín hiệu chọn địa chỉ cột CAS (Column Address Select)
- Dung lượng của DRAM= $2^{2n} \times m$ bit

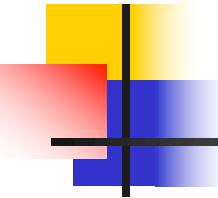
Ví dụ chip nhớ



(a) 8-Mbit EPROM



(b) 16-Mbit DRAM



Thiết kế mô-đun nhớ bán dẫn

- Dung lượng chip nhớ $2^n \times m$ bit
- Cần thiết kế để tăng dung lượng:
 - Thiết kế tăng độ dài từ nhớ
 - Thiết kế tăng số lượng từ nhớ
 - Thiết kế kết hợp

Tăng độ dài từ nhớ

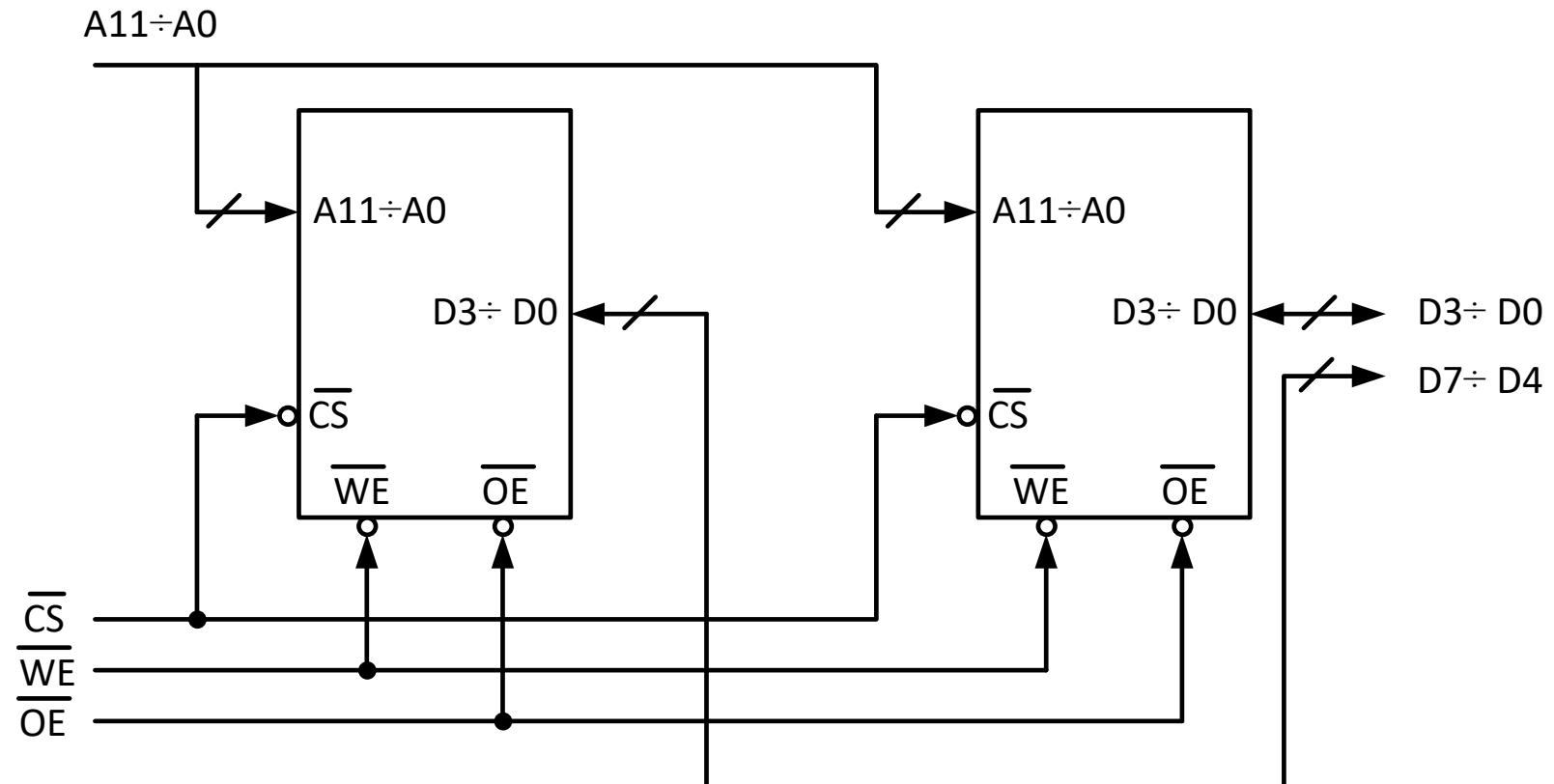
VD1:

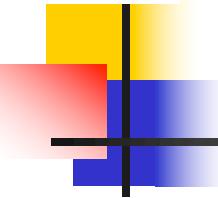
- Cho chip nhớ SRAM $4K \times 4$ bit
- Thiết kế mô-đun nhớ $4K \times 8$ bit

Giải:

- Dung lượng chip nhớ = $2^{12} \times 4$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 4 chân dữ liệu
- mô-đun nhớ cần có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu

Sơ đồ ví dụ tăng độ dài từ nhớ





Bài toán tăng độ dài từ nhớ tổng quát

- Cho chip nhớ $2^n \times m$ bit
- Thiết kế mô-đun nhớ $2^n \times (k.m)$ bit
- Dùng k chip nhớ

Tăng số lượng từ nhớ

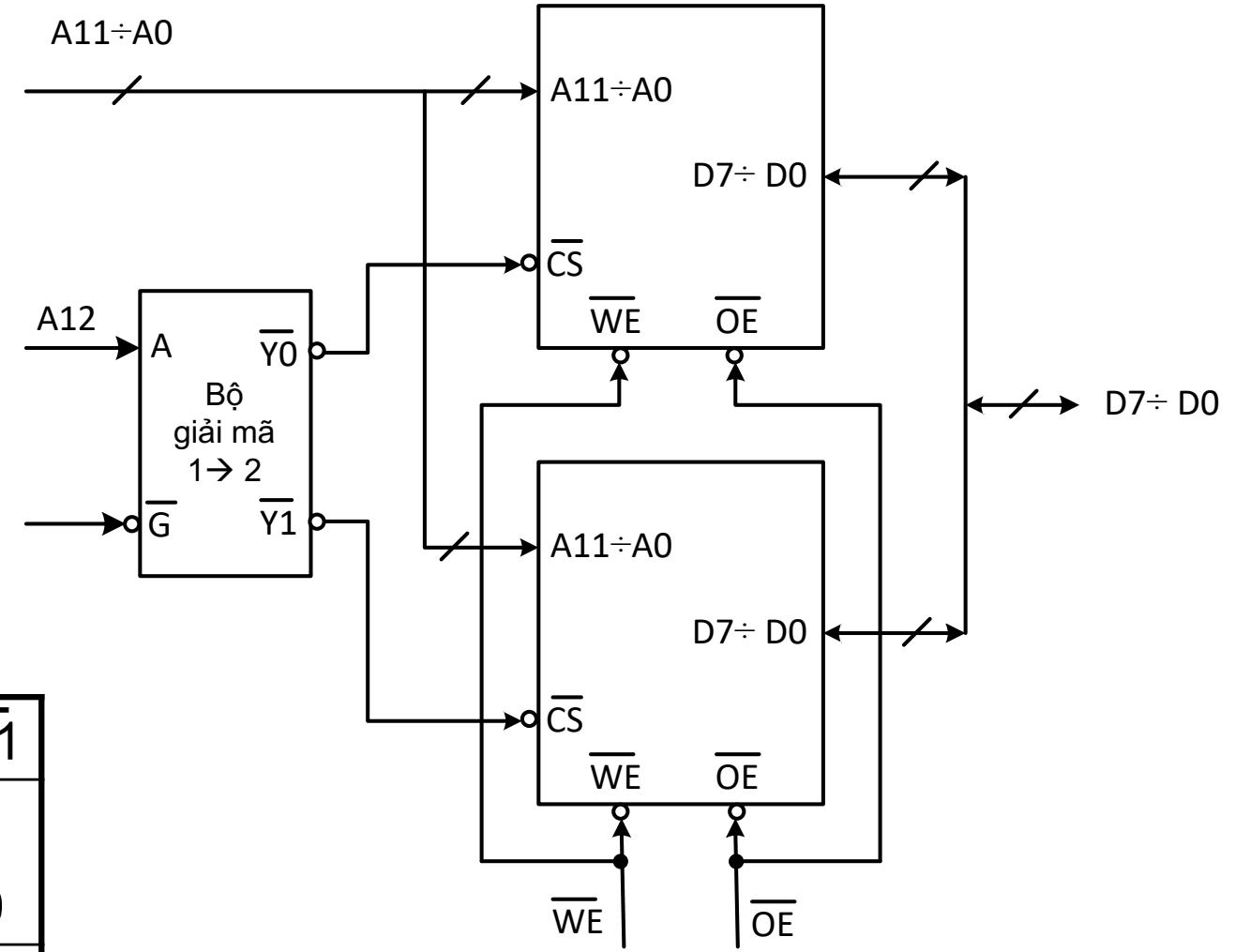
VD2:

- Cho chip nhớ SRAM $4K \times 8$ bit
- Thiết kế mô-đun nhớ $8K \times 8$ bit

Giải:

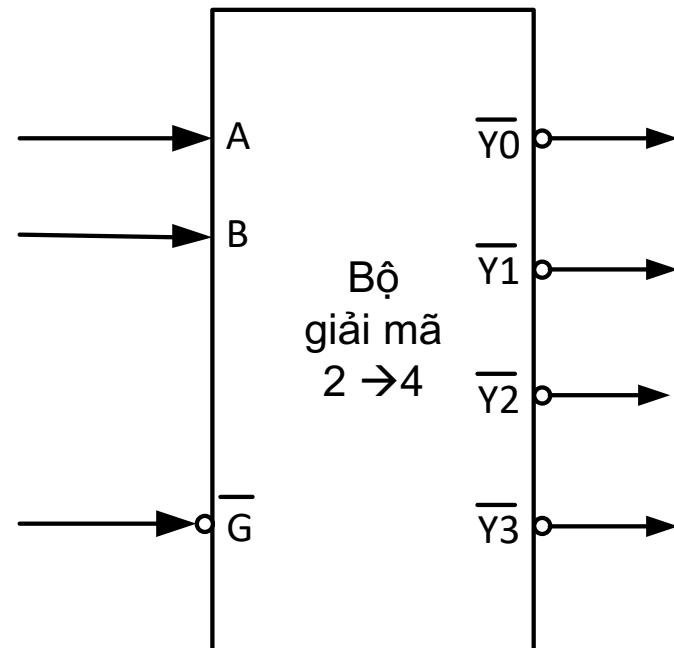
- Dung lượng chip nhớ = $2^{12} \times 8$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu
- Dung lượng mô-đun nhớ = $2^{13} \times 8$ bit
 - 13 chân địa chỉ
 - 8 chân dữ liệu

Sơ đồ ví dụ tăng số lượng từ nhớ

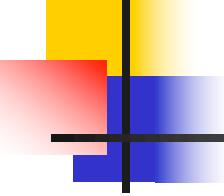


\bar{G}	A	\bar{Y}_0	\bar{Y}_1
0	0	0	1
0	1	1	0
1	X	1	1

Bộ giải mã 2→4



\bar{G}	B	A	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1



Thiết kế kết hợp

Ví dụ 3:

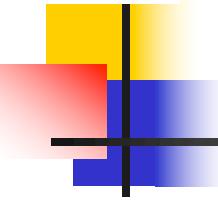
- Cho chip nhớ SRAM $4K \times 4$ bit
- Thiết kế mô-đun nhớ $8K \times 8$ bit

Phương pháp thực hiện:

- Kết hợp ví dụ 1 và ví dụ 2
- Tự vẽ sơ đồ thiết kế

2. Các đặc trưng cơ bản của bộ nhớ chính

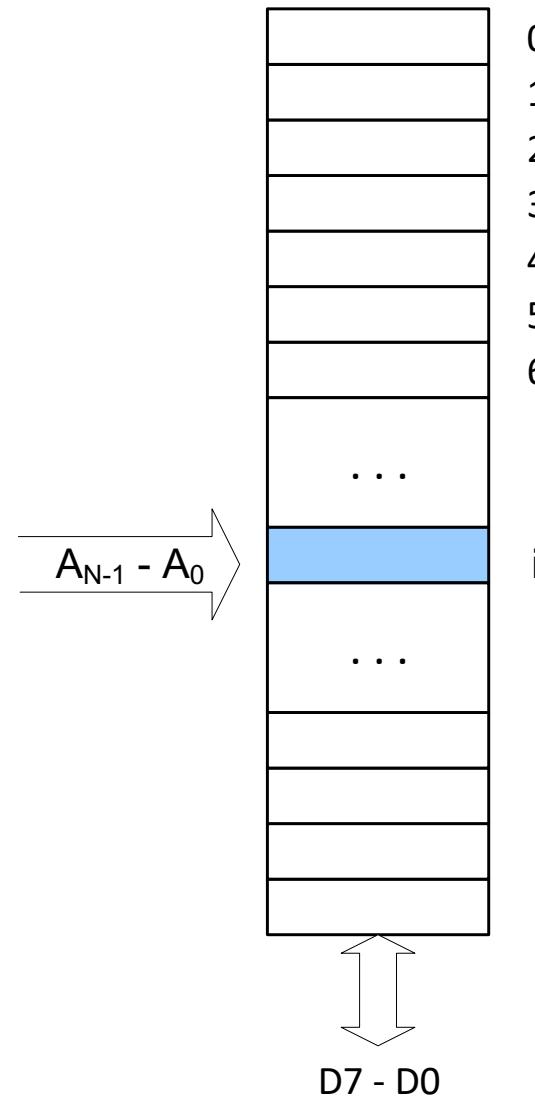
- Chứa các chương trình đang thực hiện và các dữ liệu đang được sử dụng
- Tồn tại trên mọi hệ thống máy tính
- Bao gồm các ngăn nhớ được đánh địa chỉ trực tiếp bởi CPU
- Dung lượng của bộ nhớ chính nhỏ hơn không gian địa chỉ bộ nhớ mà CPU quản lý.
- Việc quản lý logic bộ nhớ chính tùy thuộc vào hệ điều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



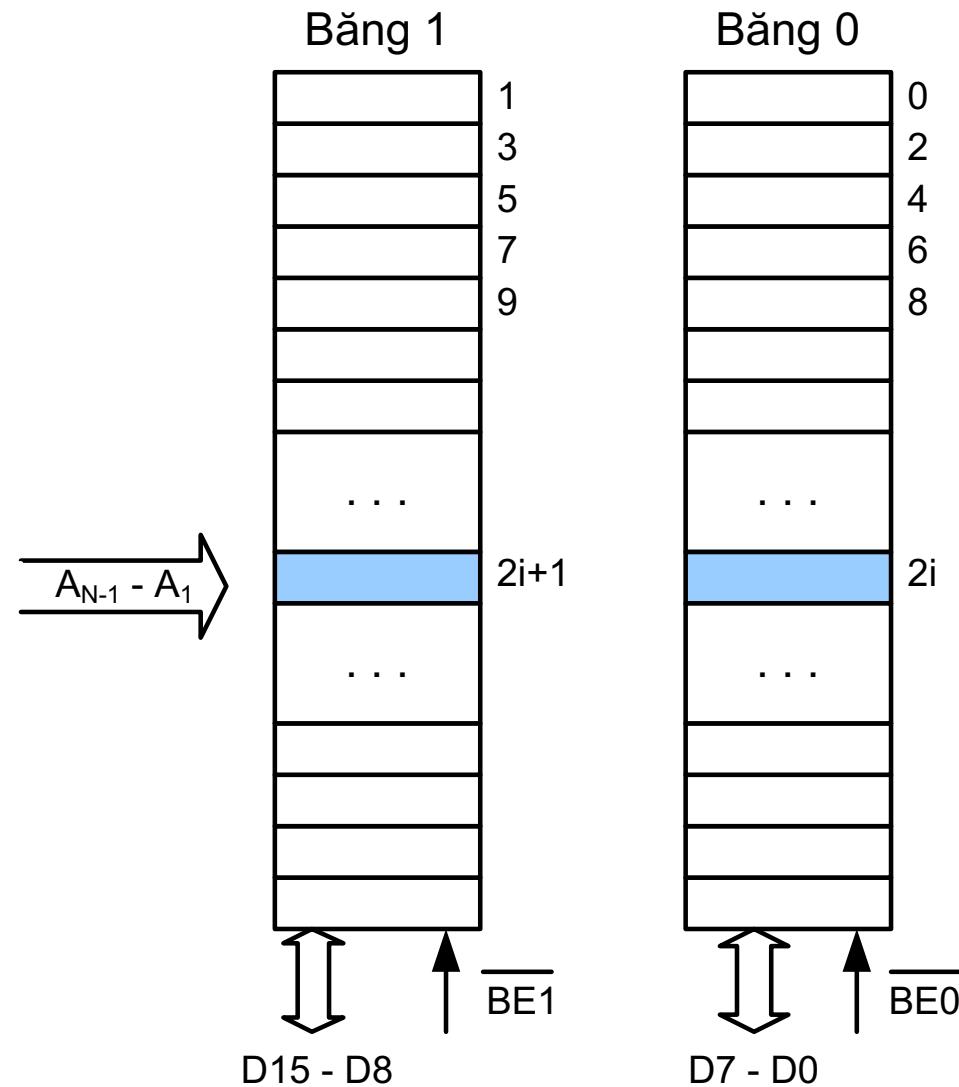
Tổ chức bộ nhớ đan xen (interleaved memory)

- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ: $m = 8, 16, 32, 64, 128 \dots$ bit
- Các ngăn nhớ được tổ chức theo byte
→ tổ chức bộ nhớ vật lý khác nhau

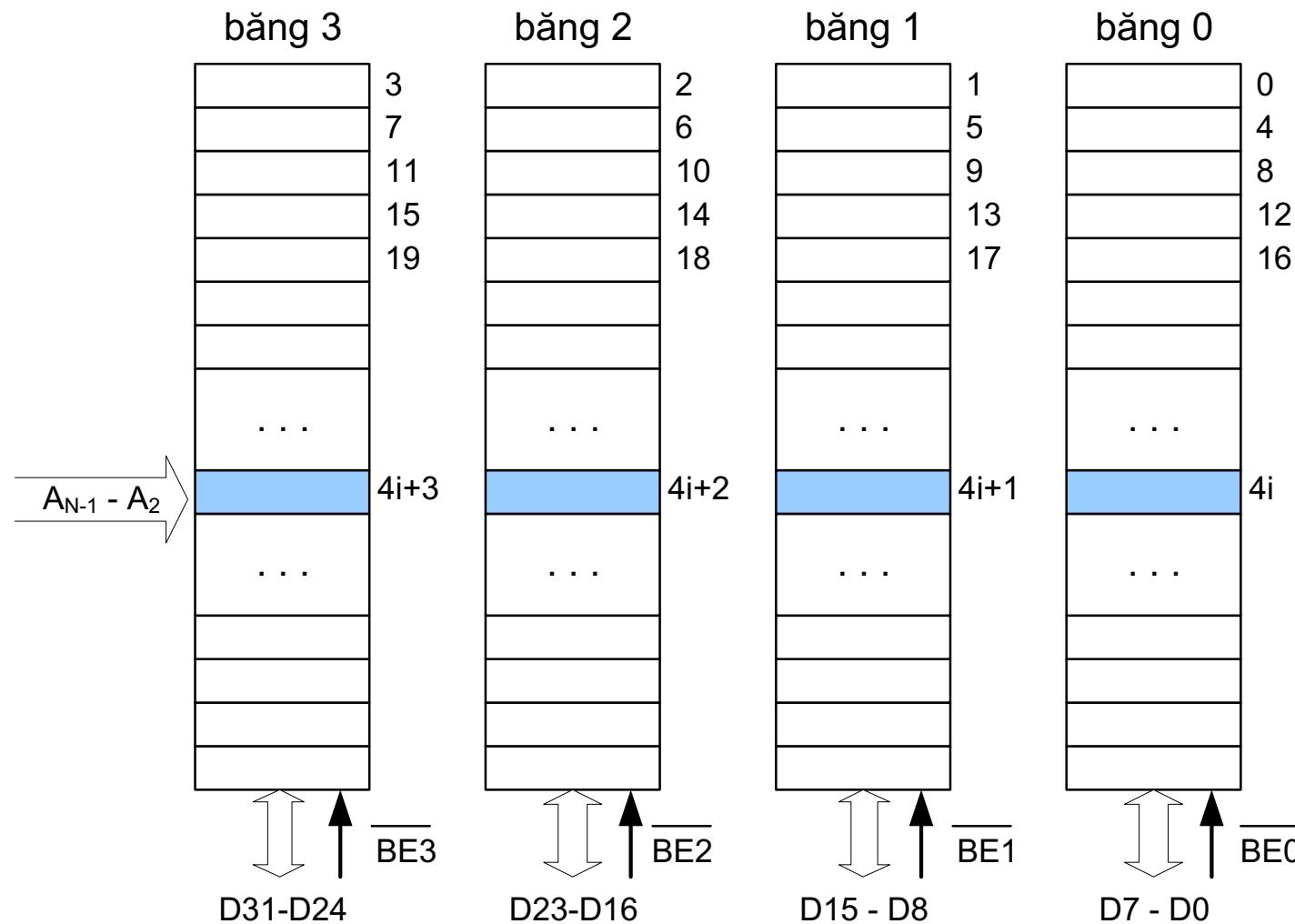
m=8bit → một băng nhớ tuyến tính



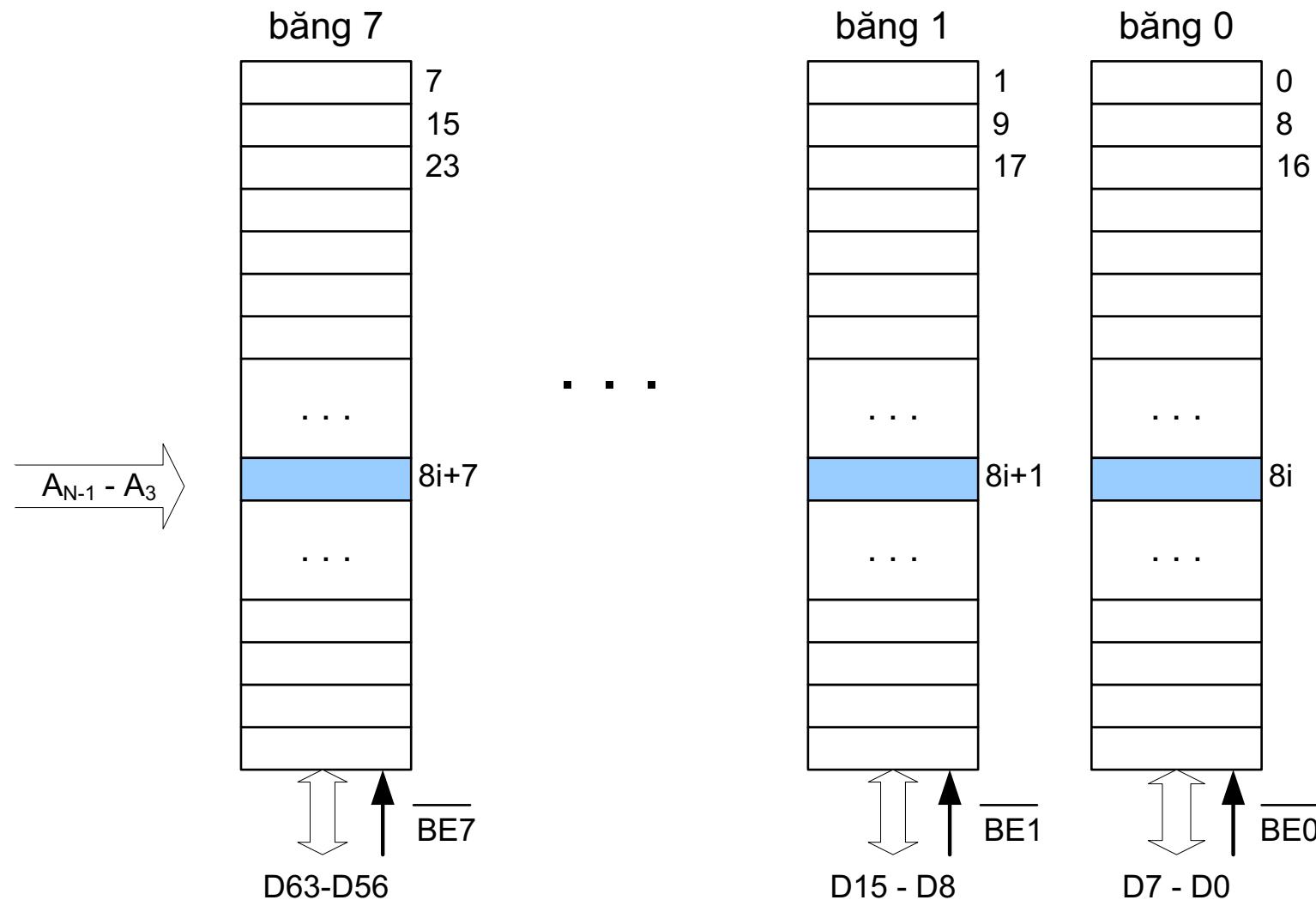
$m = 16\text{bit} \rightarrow \text{hai bảng nhớ đan xen}$



$m = 32\text{bit} \rightarrow$ bốn băng nhớ đan xen



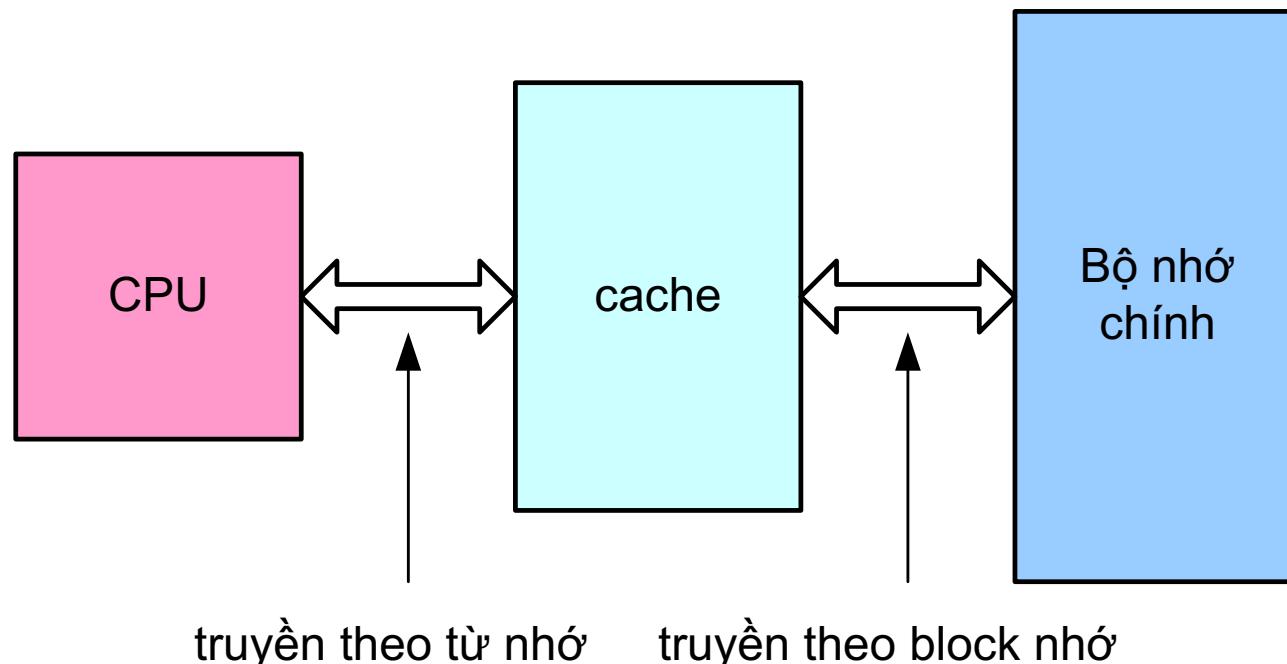
$m = 64\text{bit} \rightarrow \text{tám bảng nhớ đan xen}$

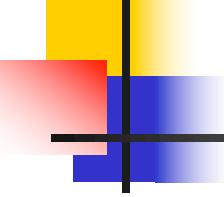


2.3. Bộ nhớ đệm (cache memory)

1. Nguyên tắc chung của cache

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

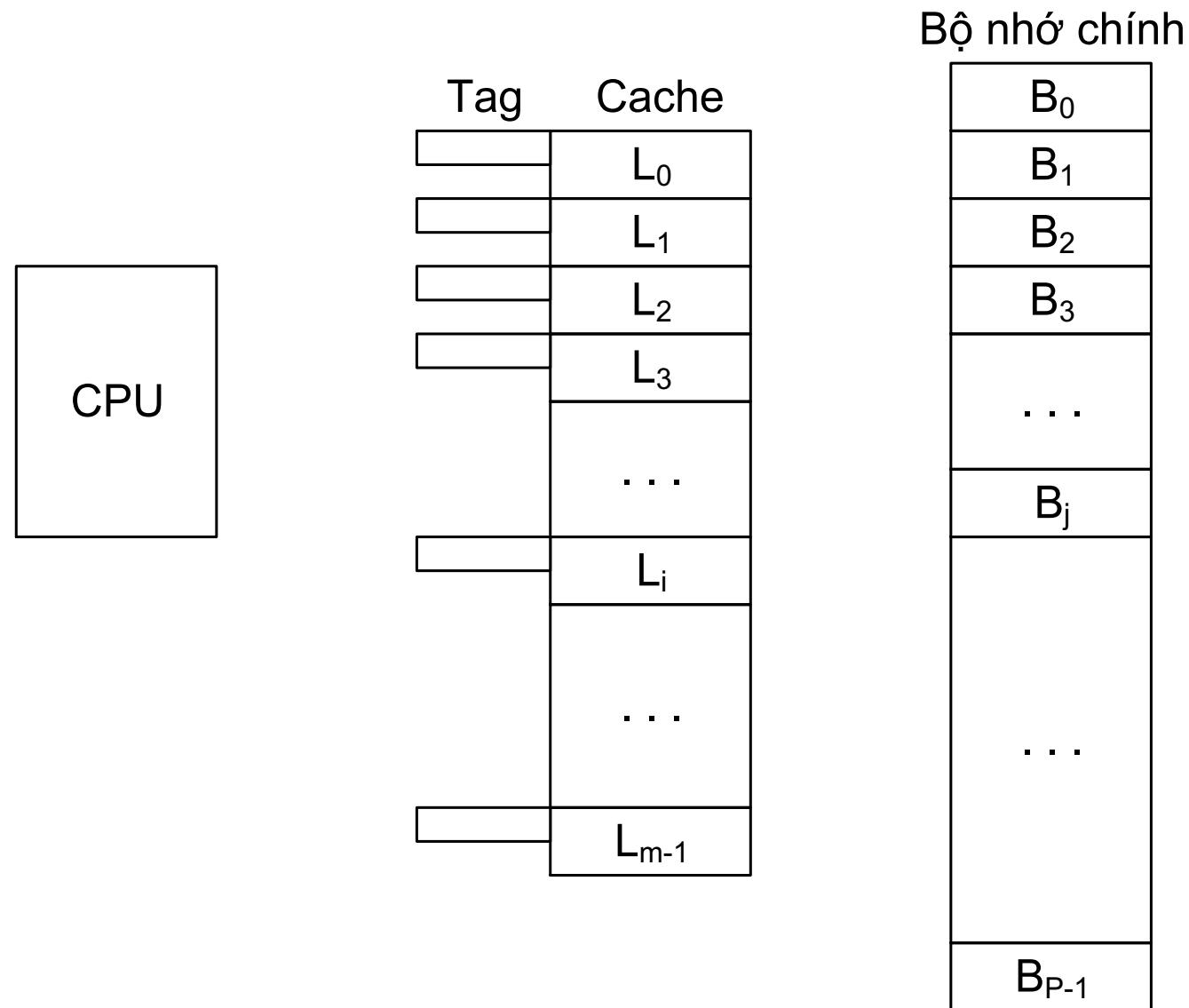




Ví dụ về thao tác của cache

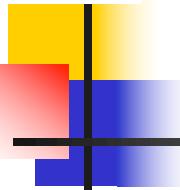
- CPU yêu cầu nội dung của ngăn nhớ
- CPU kiểm tra trên cache với dữ liệu này
- Nếu có, CPU nhận dữ liệu từ cache (nhanh)
- Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache
- Tiếp đó chuyển dữ liệu từ cache vào CPU

Cấu trúc chung của cache / bộ nhớ chính



Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- Bộ nhớ chính có 2^N byte nhớ
- Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
 - Bộ nhớ chính: $B_0, B_1, B_2, \dots, B_{p-1}$ (p Blocks)
 - Bộ nhớ cache: $L_0, L_1, L_2, \dots, L_{m-1}$ (m Lines)
 - Kích thước của Block (Line) = 8,16,32,64,128 byte
- Mỗi Line trong cache có một thẻ nhớ (Tag) được gắn vào



Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- 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
- Nội dung Tag được cập nhật mỗi khi Block từ bộ nhớ chính nạp vào 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

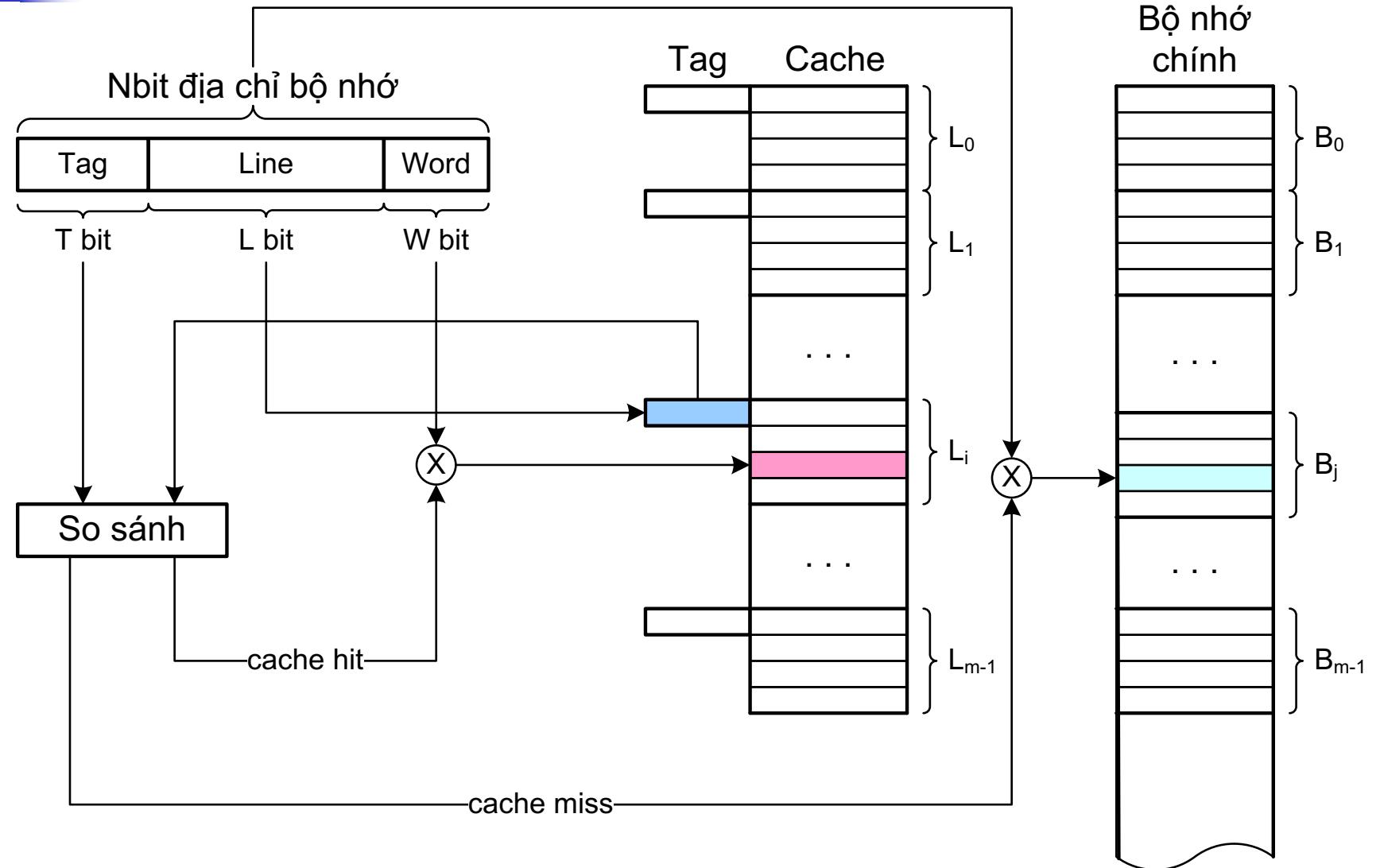
(Chính là các phương pháp tổ chức bộ nhớ cache)

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

Ánh xạ trực tiếp

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

Ánh xạ trực tiếp (tiếp)



Ánh xạ trực tiếp (tiếp)

- Địa chỉ N bit của bộ nhớ chính chia thành 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*:
$$2^L = \text{số } Line \text{ trong } cache = m$$
 - Trường *Tag* gồm T bit:
$$T = N - (W+L)$$

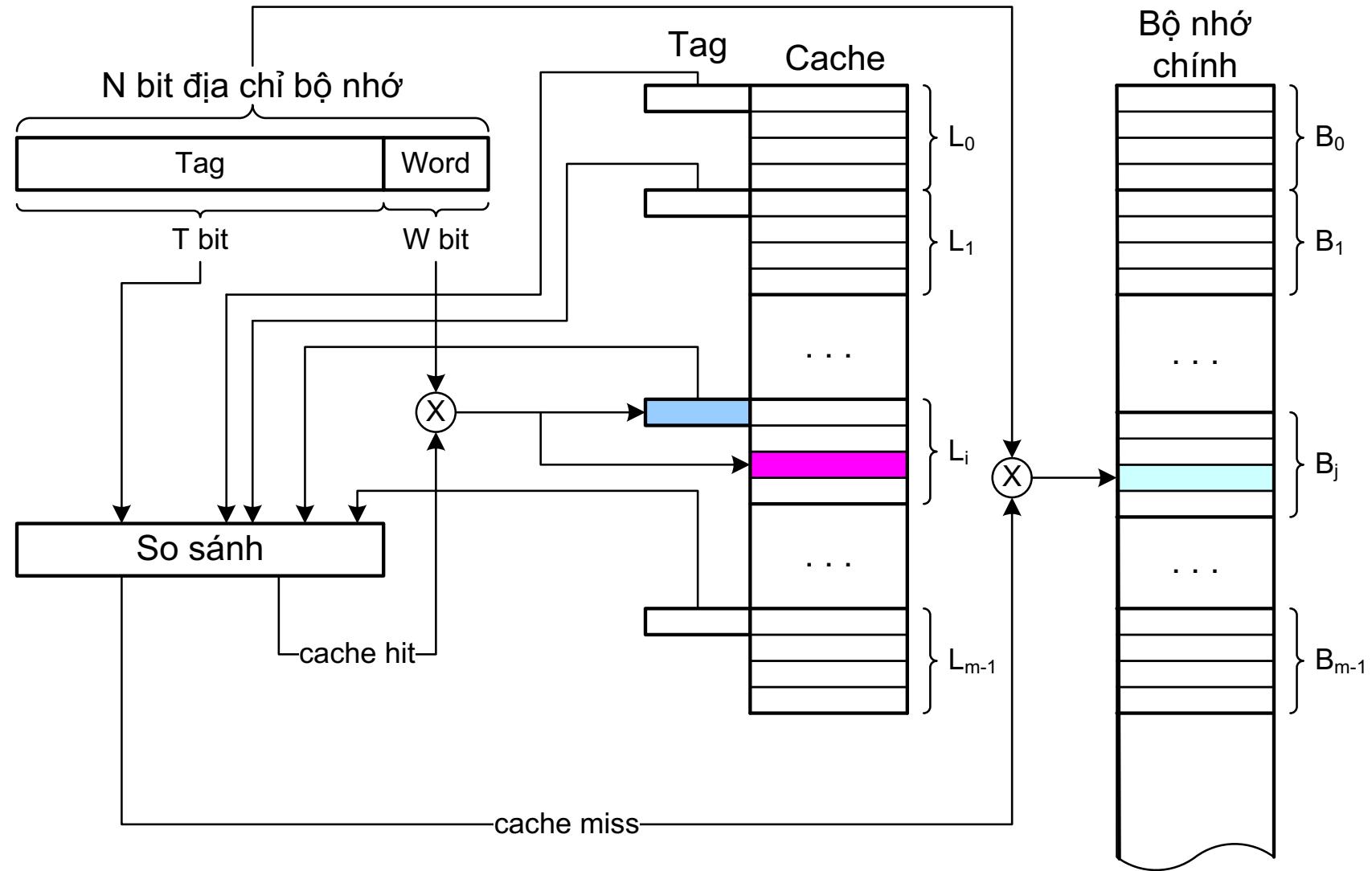
Ánh xạ trực tiếp (tiếp)

- Mỗi thẻ nhớ (Tag) của một Line chứa được T bit
- Khi Block từ bộ nhớ chính được nạp vào Line của cache thì Tag ở đó được cập nhật giá trị là T bit địa chỉ cao của Block đó
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ N bit cụ thể
 - Nhờ vào giá trị L bit của trường Line sẽ tìm ra Line tương ứng
 - Đọc nội dung Tag ở Line đó (T bit), rồi so sánh với T bit bên trái của địa chỉ vừa phát ra
 - Giống nhau: cache hit
 - Khác nhau: cache miss
- Ưu điểm: Bộ so sánh đơn giản
- Nhược điểm: Xác suất *cache hit* thấp

Ánh xạ liên kết toàn phần

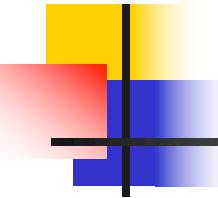
- Mỗi *Block* có thể nạp vào bất kỳ *Line* nào của *cache*
- Địa chỉ của bộ nhớ chính chia thành hai trường:
 - Trường Word
 - Trường *Tag* dùng để xác định *Block* của bộ nhớ chính
- Tag xác định *Block* đang nằm ở *Line* đó

Ánh xạ liên kết toàn phần (tiếp)



Ánh xạ liên kết toàn phần (tiếp)

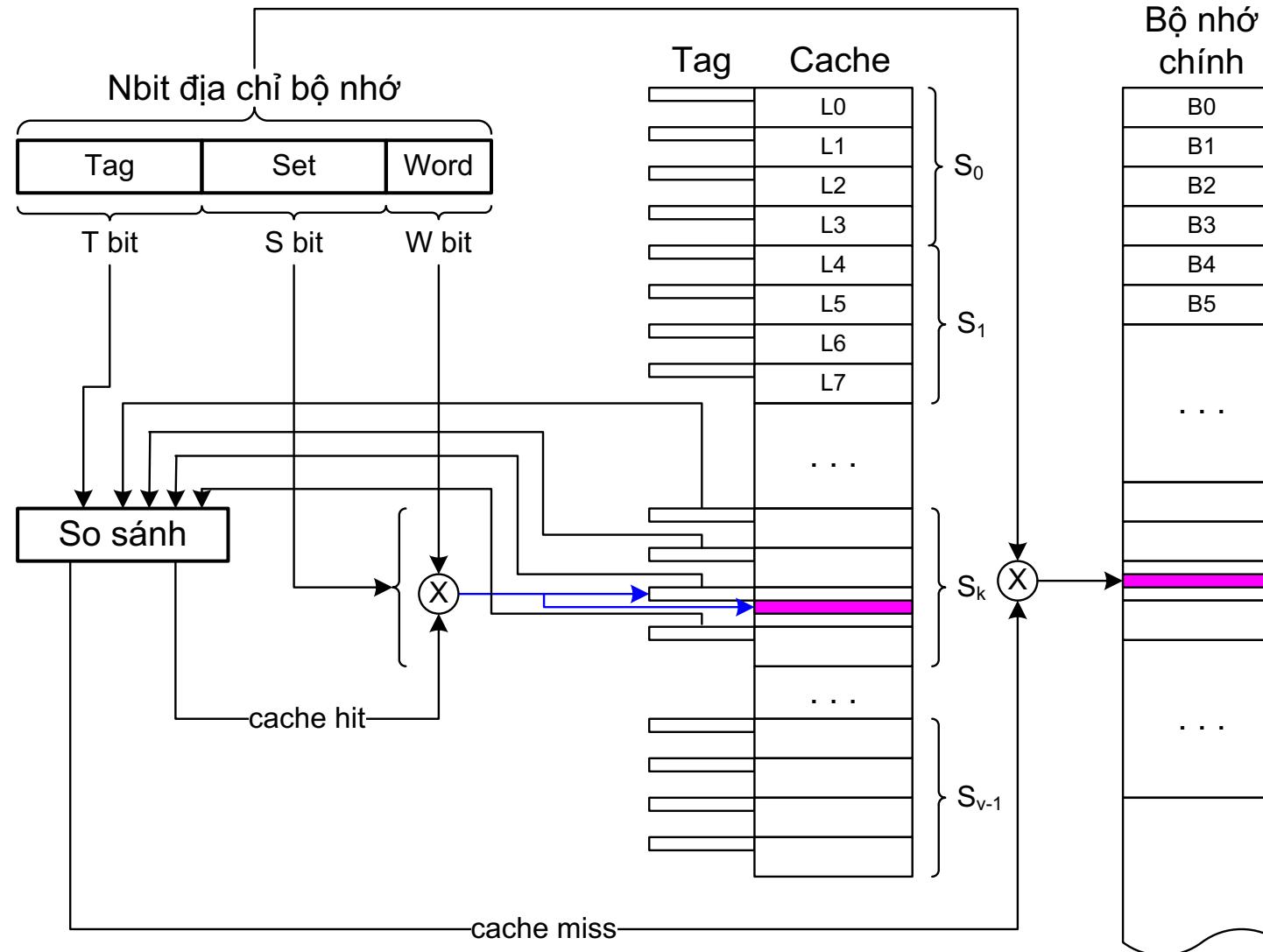
- Mỗi thẻ nhớ (Tag) của một Line chứa được T bit
- Khi Block từ bộ nhớ chính được nạp vào Line của cache thì Tag ở đó được cập nhật giá trị là T bit địa chỉ cao của Block đó
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ N bit cụ thể
 - So sánh T bit bên trái của địa chỉ vừa phát ra với lần lượt nội dung của các Tag trong cache
 - Nếu gặp giá trị bằng nhau: cache hit xảy ra ở Line đó
 - Nếu không có giá trị nào bằng: cache miss
- **Ưu điểm:** Xác suất *cache hit* cao
- **Nhược điểm:**
 - So sánh đồng thời với tất cả các Tag → mất nhiều thời gian
 - Bộ so sánh phức tạp
- Ít sử dụng



Ánh xạ liên kết tập hợp

- Dung hòa cho hai phương pháp trên
- 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:
 - $B_0 \rightarrow S_0$
 - $B_1 \rightarrow S_1$
 - $B_2 \rightarrow S_2$
 -

Ánh xạ liên kết tập hợp (tiếp)



Ánh xạ liên kết tập hợp (tiếp)

- Kích thước *Block* = 2^W Word
- Trường *Set* có S bit dùng để xác định một trong số các Set trong cache. 2^S = Số *Set* trong cache
- Trường *Tag* có T bit: $T = N - (W+S)$
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ N bit cụ thể
 - Nhờ vào giá trị S bit của trường Set sẽ tìm ra Set tương ứng
 - So sánh T bit bên trái của địa chỉ vừa phát ra với lần lượt nội dung của các Tag trong Set đó
 - Nếu gặp giá trị bằng nhau: cache hit xảy ra ở Line tương ứng
 - Nếu không có giá trị nào bằng: cache miss
- Tổng quát cho cả hai phương pháp trên
- Thông dụng với: 2,4,8,16 Lines/Set

Ví dụ về ánh xạ địa chỉ

- Giả sử máy tính đánh địa chỉ cho từng byte
- Không gian địa chỉ bộ nhớ chính = 4GiB
- Dung lượng bộ nhớ cache là 256KiB
- Kích thước *Line (Block)* = 32byte.
- Xác định số bit của các trường địa chỉ cho ba trường hợp tổ chức:
 - Ánh xạ trực tiếp
 - Ánh xạ liên kết toàn phần
 - Ánh xạ liên kết tập hợp 4 đường

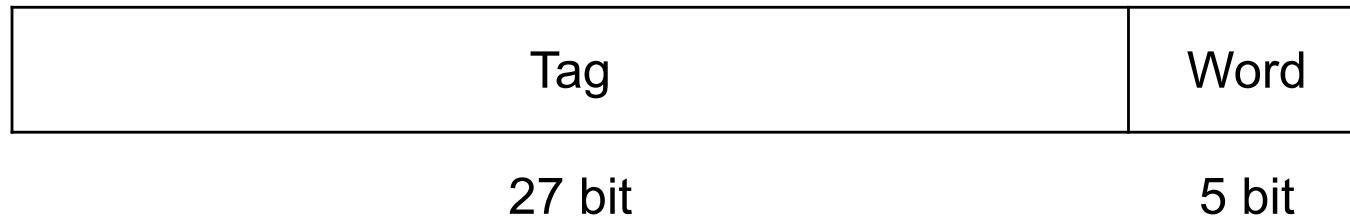
Với ánh xạ trực tiếp

- Bộ nhớ chính = $4\text{GiB} = 2^{32}$ byte \rightarrow Số bit địa chỉ của bộ nhớ chính là: $N = 32$ bit
- Cache = $256\text{ KiB} = 2^{18}$ byte
- Kích thước *Line* = 32 byte = 2^5 byte \rightarrow số bit địa chỉ của trường Word là: $W = 5$ bit
- Số *Line* trong cache = $2^{18}/ 2^5 = 2^{13}$ *Line* \rightarrow số bit địa chỉ trường Line là: $L = 13$ bit
- Số bit địa chỉ của trường Tag là:
 $T = 32 - (13 + 5) = 14$ bit

Tag	Line	Word
14 bit	13 bit	5 bit

Với ánh xạ liên kết toàn phần

- Bộ nhớ chính = $4\text{GiB} = 2^{32}$ byte \rightarrow số bit địa chỉ của bộ nhớ chính là: $N = 32$ bit
- Kích thước *Line* = 32 byte = 2^5 byte \rightarrow số bit địa chỉ của trường Word là: $W = 5$ bit
- Số bit địa chỉ của trường Tag là:
 $T = 32 - 5 = 27$ bit



Với ánh xạ liên kết tập hợp 4 đường

- Bộ nhớ chính = $4\text{GiB} = 2^{32}$ byte \rightarrow số bit địa chỉ của bộ nhớ chính là: $N = 32$ bit
- Kích thước *Line* = 32 byte = 2^5 byte \rightarrow số bit địa chỉ của trường Word là: $W = 5$ bit
- Số *Line* trong *cache* = $2^{18}/ 2^5 = 2^{13}$ *Line*
- Một *Set* có 4 *Line* = 2^2 *Line*
 \rightarrow số *Set* trong *cache* = $2^{13}/ 2^2 = 2^{11}$ *Set*
 \rightarrow số bit địa chỉ của trường Set là: $S = 11$ bit
- Số bit địa chỉ của trường *Tag* là:
 $T = 32 - (11 + 5) = 16$ bit

Tag	Set	Word
16 bit	11 bit	5 bit

3. Thay thế block trong cache

Với ánh xạ trực tiếp:

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

Thay thế block trong cache (tiếp)

Với ánh xạ liên kết: cần có thuật giải thay thế:

- **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**

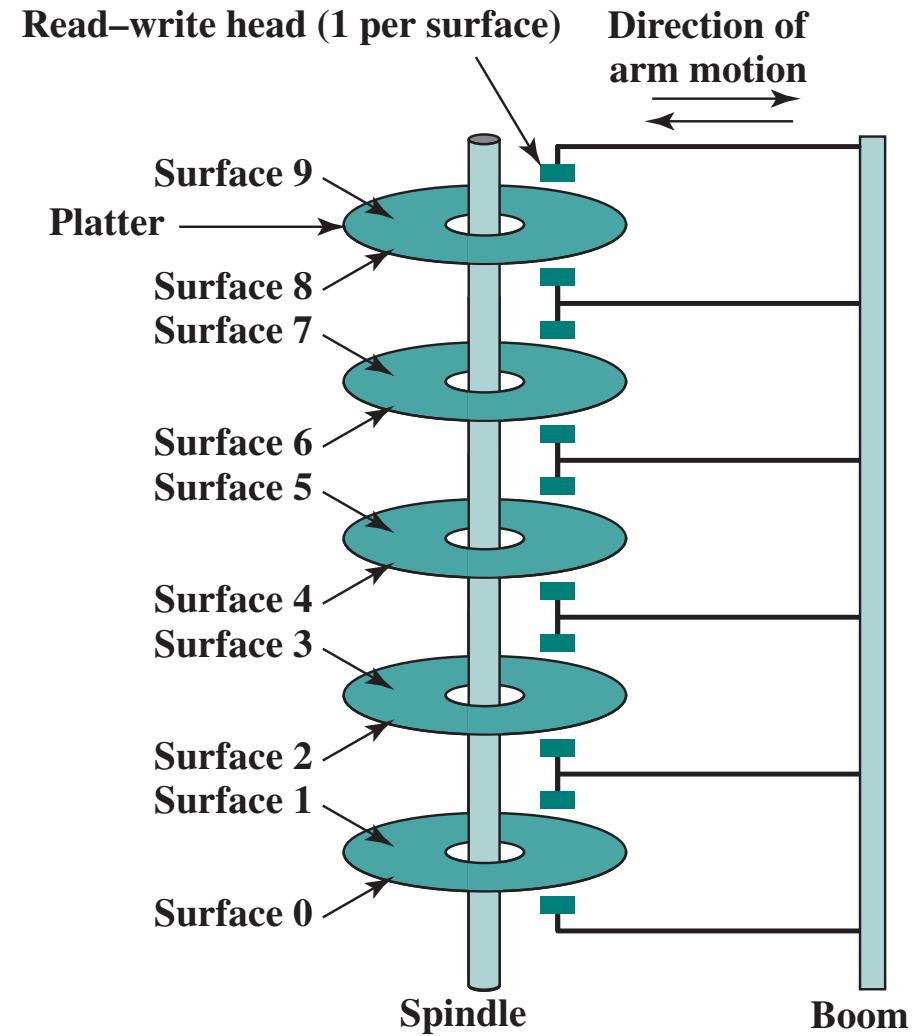
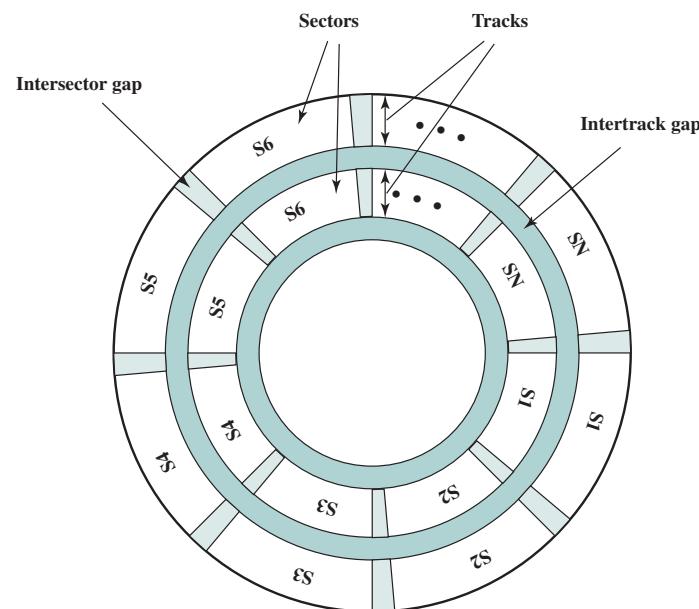
4. Phương pháp ghi dữ liệu khi cache hit

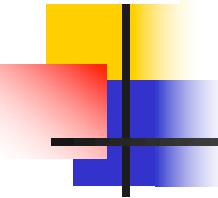
- Ghi xuyên qua (Write-through):
 - ghi cả cache và cả bộ nhớ chính
 - tốc độ chậm
- Ghi trả sau (Write-back):
 - chỉ ghi ra cache
 - tốc độ nhanh
 - khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính

2.4. Bộ nhớ ngoài

- Tồn tại dưới dạng các thiết bị lưu trữ
- Các kiểu bộ nhớ ngoài
 - Băng từ: ít sử dụng
 - Đĩa từ: Ổ đĩa cứng HDD (Hard Disk Drive)
 - Đĩa quang: CD, DVD
 - Bộ nhớ Flash:
 - Ổ nhớ thẻ rắn SSD (Solid State Drive)
 - USB flash
 - Thẻ nhớ

Ổ đĩa cứng (HDD –Hard Disk Drive)





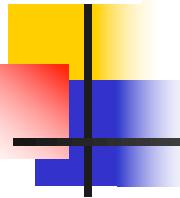
HDD

- Dung lượng lớn
- Tốc độ đọc/ghi chậm
- Tốn năng lượng
- Dễ bị lỗi cơ học
- Rẻ tiền

Ổ SSD (Solid State Drive)

- Bộ nhớ bán dẫn flash
- Không khả biến
- Tốc độ nhanh
- Tiêu thụ năng lượng ít
- Gồm nhiều chip nhớ flash và cho phép truy cập song song
- Ít bị lỗi
- Đắt tiền





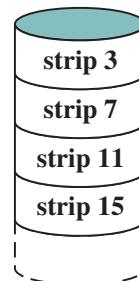
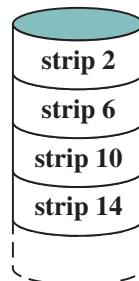
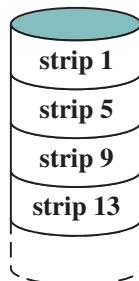
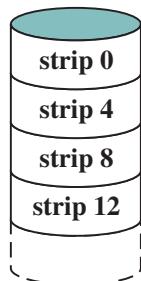
Đĩa quang

- CD (Compact Disc)
 - Dung lượng thông dụng 650MB
- DVD
 - Digital Video Disc hoặc Digital Versatile Disk
 - Ghi một hoặc hai mặt
 - Một hoặc hai lớp trên một mặt
 - Thông dụng: 4,7GB/lớp

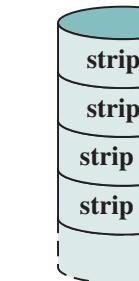
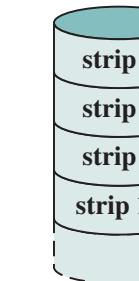
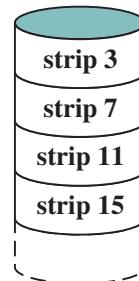
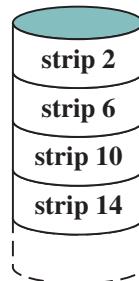
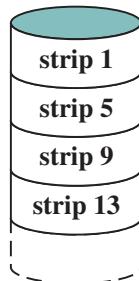
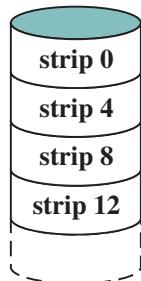
Hệ thống lưu trữ dung lượng lớn: RAID

- Redundant Array of Inexpensive Disks
- (Redundant Array of Independent Disks)
- Tập các ổ đĩa cứng vật lý được OS coi như một ổ logic duy nhất → dung lượng lớn
- Dữ liệu được lưu trữ phân tán trên các ổ đĩa vật lý → truy cập song song (nhanh)
- Lưu trữ thêm thông tin dư thừa, cho phép khôi phục lại thông tin trong trường hợp đĩa bị hỏng → an toàn thông tin
- 7 loại phổ biến (RAID 0 – 6)

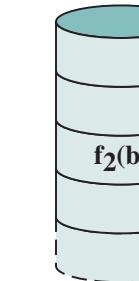
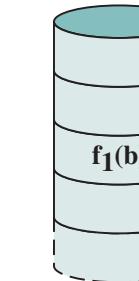
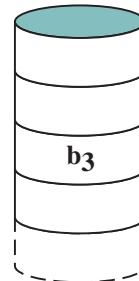
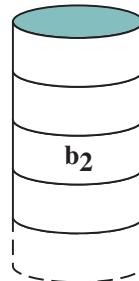
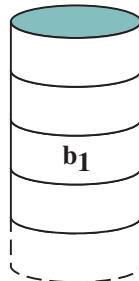
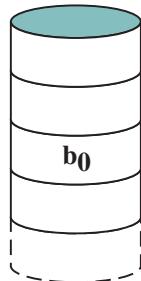
RAID 0, 1, 2



(a) RAID 0 (Nonredundant)

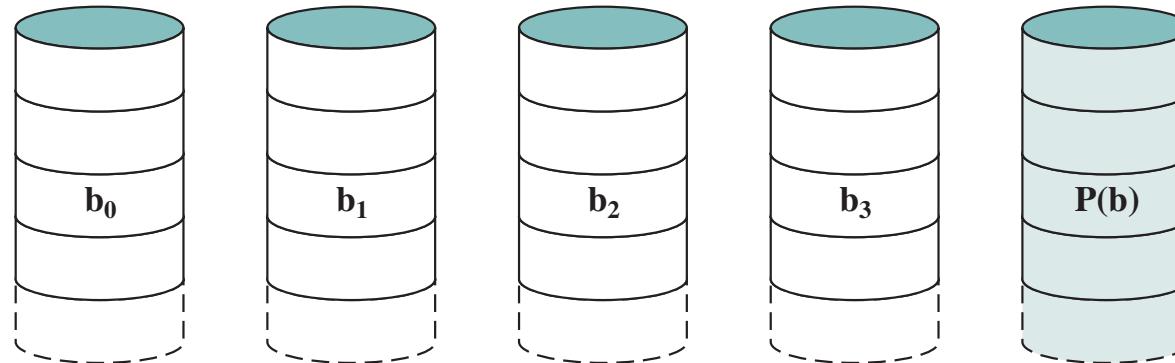


(b) RAID 1 (Mirrored)

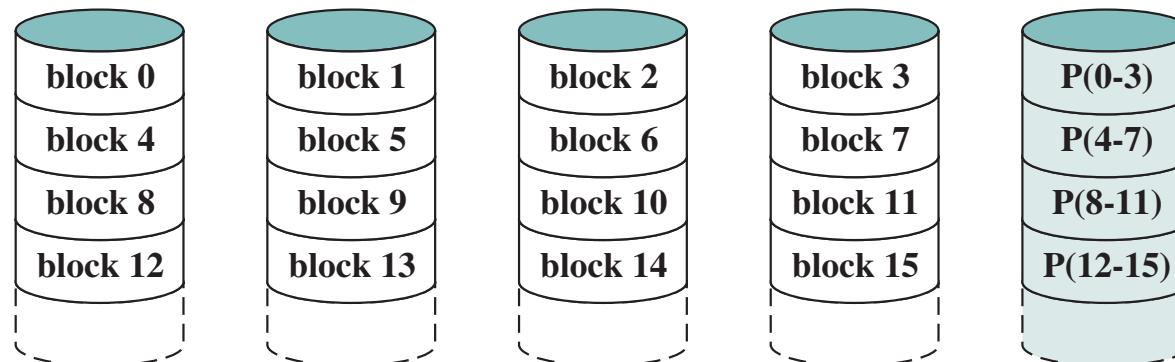


(c) RAID 2 (Redundancy through Hamming code)

RAID 3 & 4

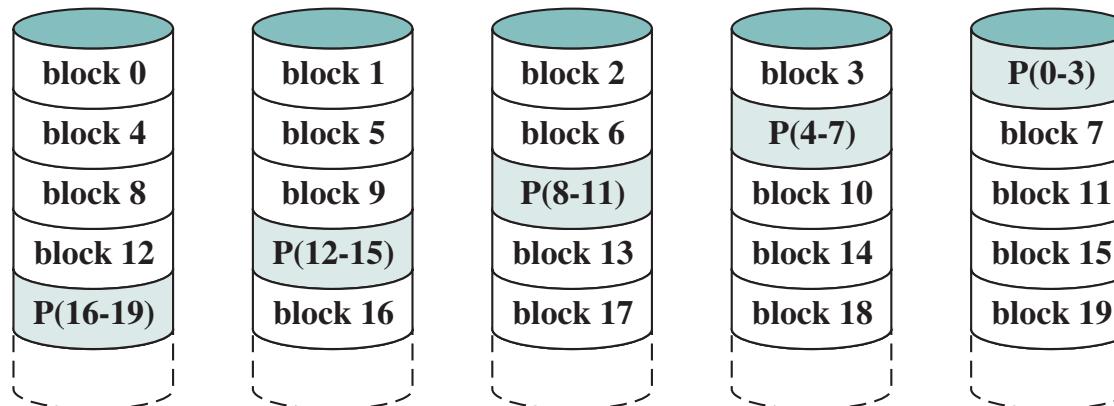


(d) RAID 3 (Bit-interleaved parity)

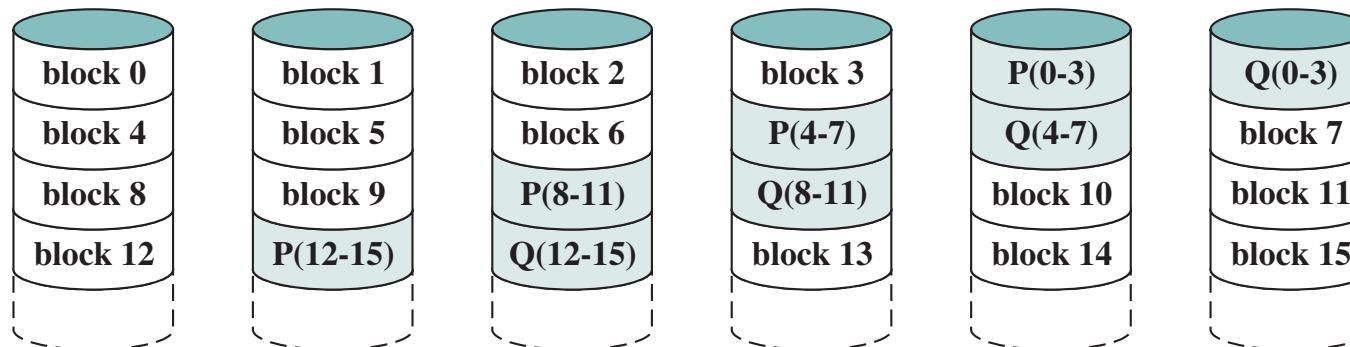


(e) RAID 4 (Block-level parity)

RAID 5 & 6

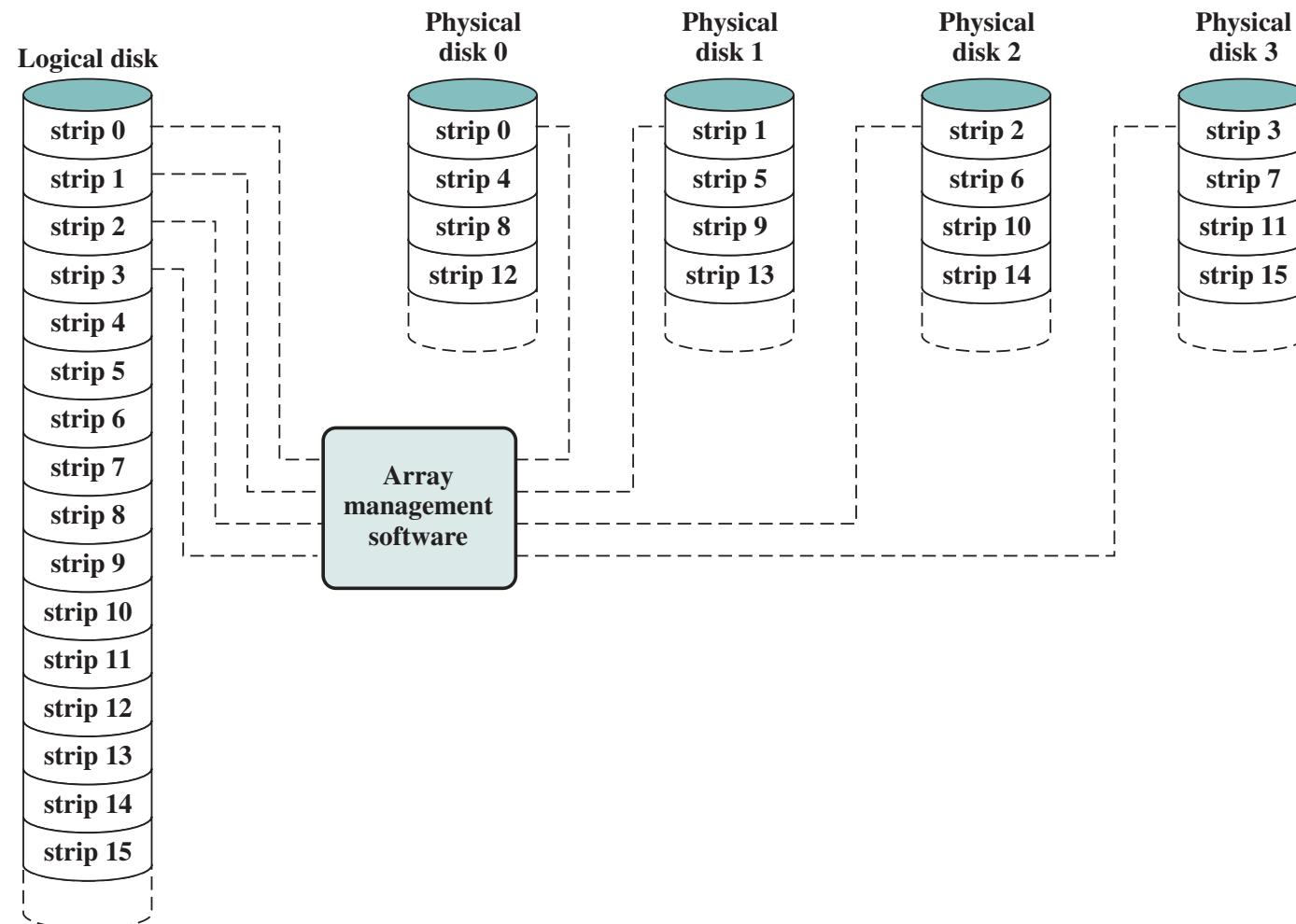


(f) RAID 5 (Block-level distributed parity)



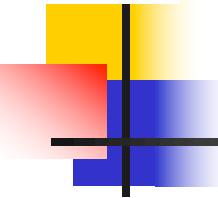
(g) RAID 6 (Dual redundancy)

Ánh xạ dữ liệu của RAID 0



2.5. Bộ nhớ ảo (Virtual Memory)

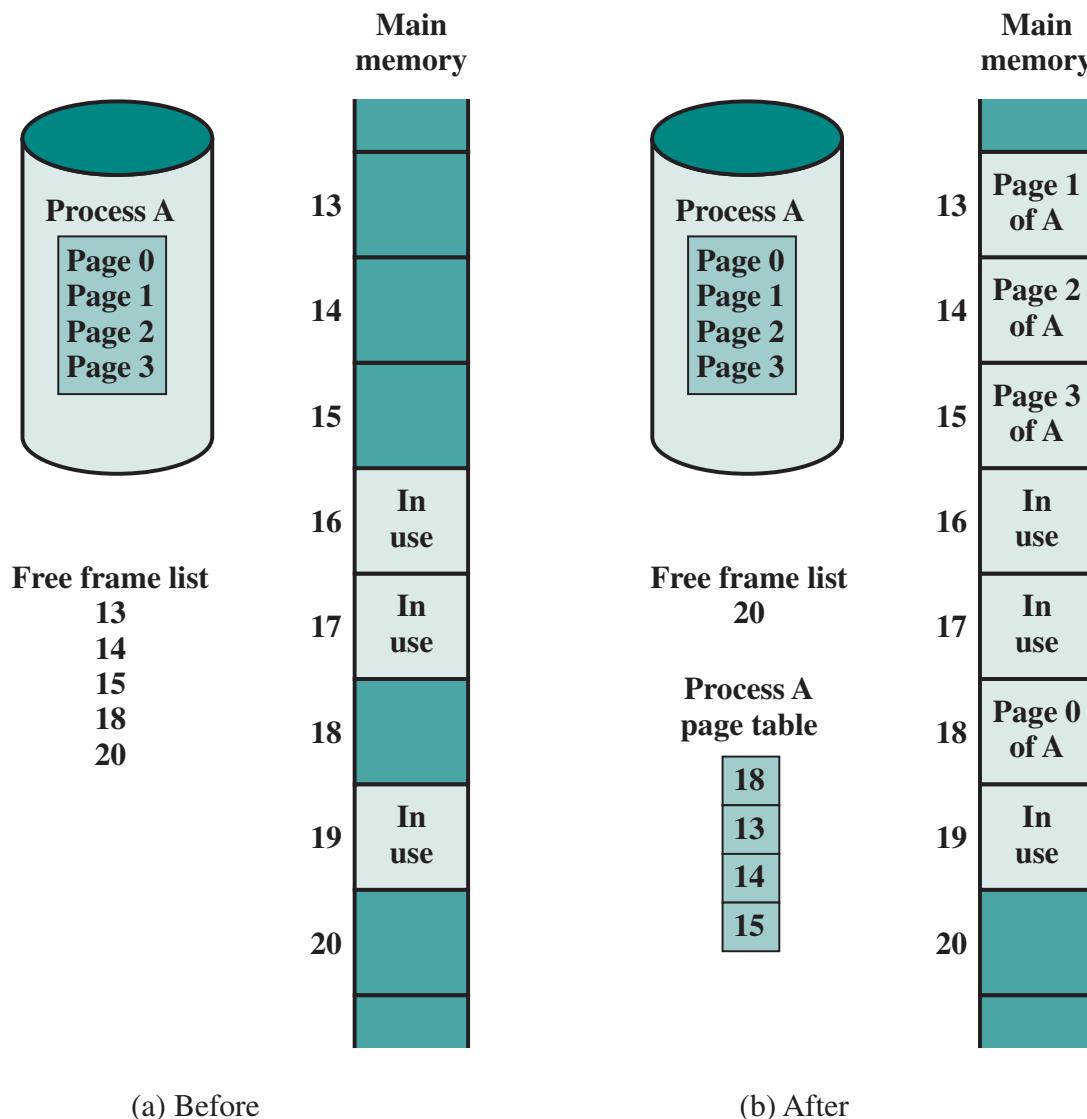
- Khái niệm bộ nhớ ảo: gồm bộ nhớ chính và bộ nhớ ngoài mà được CPU coi như là một bộ nhớ duy nhất (bộ nhớ chính).
- Các kỹ thuật thực hiện bộ nhớ ảo:
 - Kỹ thuật phân trang: Chia không gian địa chỉ bộ nhớ thành các trang nhớ có kích thước bằng nhau và nằm liền kề nhau
Thông dụng: kích thước trang =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



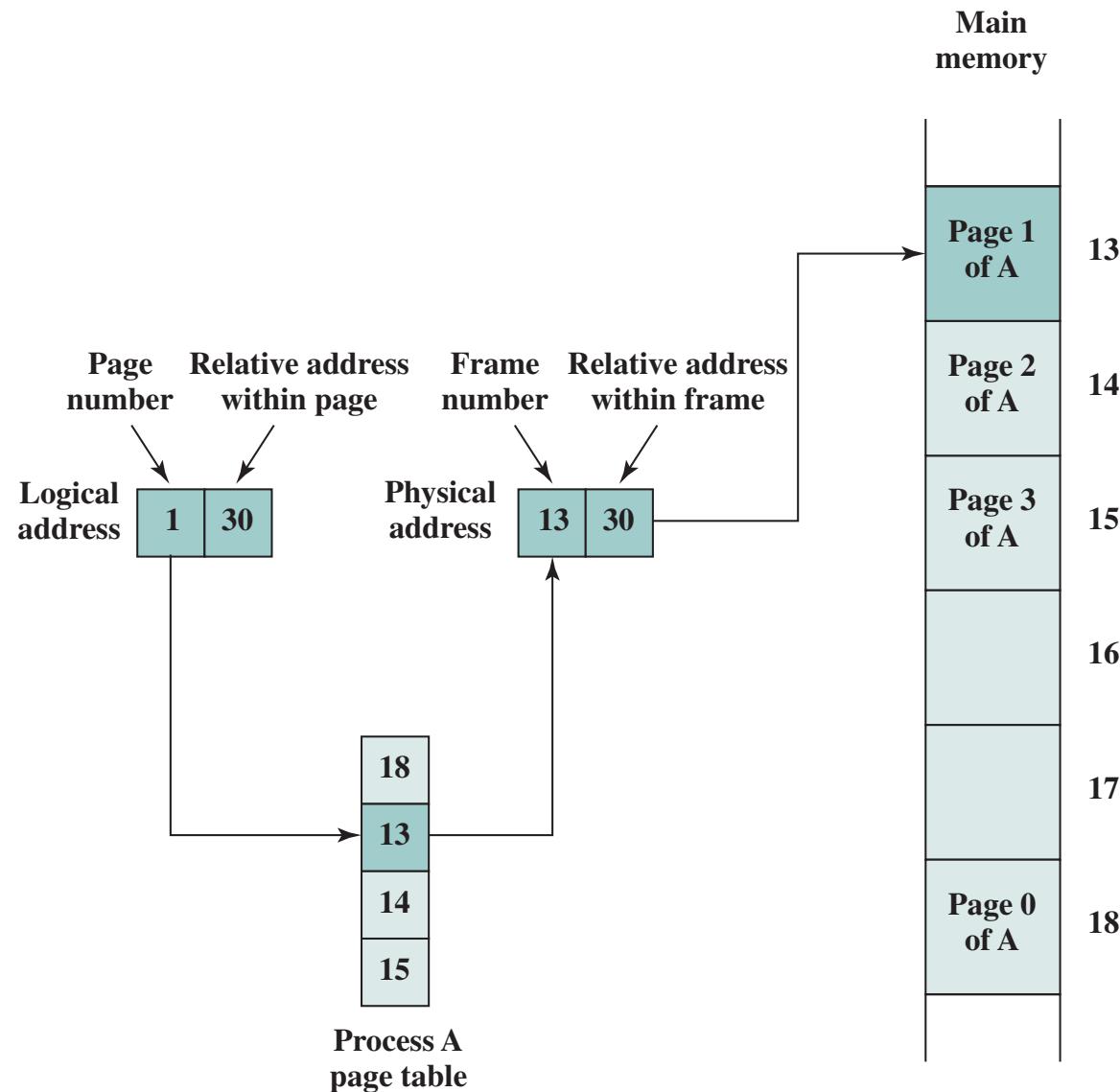
Phân trang

- Phân chia bộ nhớ thành các phần có kích thước bằng nhau gọi là các khung trang
- Chia chương trình (tiến trình) thành các trang
- Cấp phát số hiệu khung trang yêu cầu cho tiến trình
- OS duy trì danh sách các khung trang nhớ trống
- Tiến trình không yêu cầu các khung trang liên tiếp
- Sử dụng bảng trang để quản lý

Cấp phát các khung trang



Địa chỉ logic và địa chỉ vật lý của phân trang



Nguyên tắc làm việc của bộ nhớ ảo phân trang

- Phân trang theo yêu cầu
 - Không yêu cầu tất cả các trang của tiến trình nằm trong bộ nhớ
 - Chỉ nạp vào bộ nhớ những trang được yêu cầu
- Lỗi trang
 - Trang được yêu cầu không có trong bộ nhớ
 - HĐH cần hoán đổi trang yêu cầu vào
 - Có thể cần hoán đổi một trang nào đó ra để lấy chỗ
 - Cần chọn trang để đưa ra

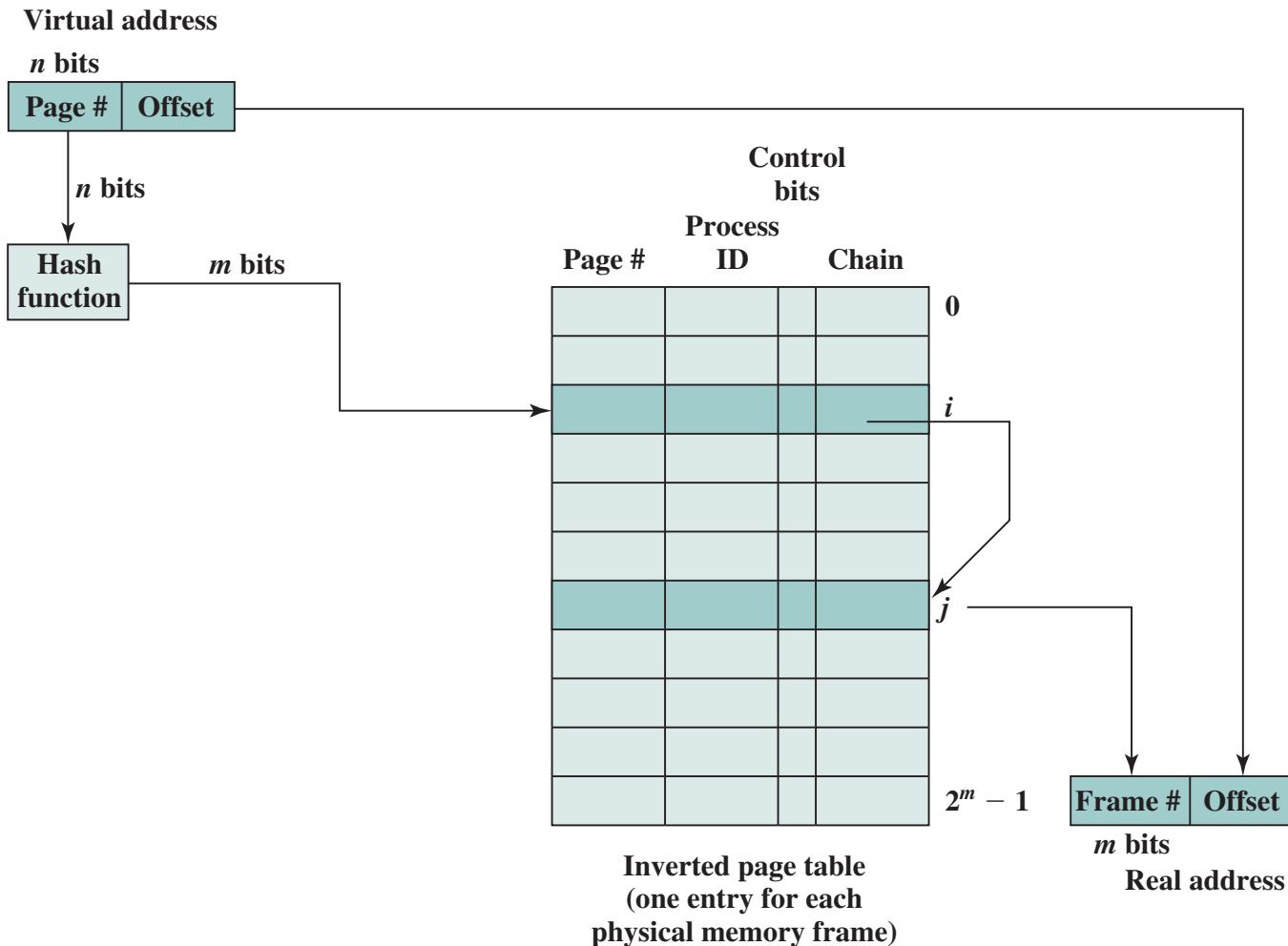
Thất bại

- Quá nhiều tiến trình trong bộ nhớ quá nhỏ
- OS tiêu tốn toàn bộ thời gian cho việc hoán đổi
- Có ít hoặc không có công việc nào được thực hiện
- Đĩa luôn luôn sáng
- Giải pháp:
 - Thuật toán thay trang
 - Giảm bớt số tiến trình đang chạy
 - Thêm bộ nhớ

Lợi ích

- Không cần toàn bộ tiến trình nằm trong bộ nhớ để chạy
- Có thể hoán đổi trang được yêu cầu
- Như vậy có thể chạy những tiến trình lớn hơn tổng bộ nhớ sẵn dùng
- Bộ nhớ chính được gọi là bộ nhớ thực
- Người dùng cảm giác bộ nhớ lớn hơn bộ nhớ thực

Cấu trúc bảng trang

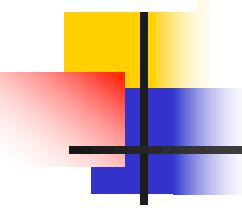


Bộ nhớ trên máy tính PC

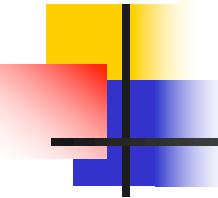
- Bộ nhớ cache: tích hợp trên chip vi xử lý:
 - L1: cache lệnh và cache dữ liệu
 - L2, L3
- Bộ nhớ chính: Tồn tại dưới dạng các mô-đun nhớ RAM

Bộ nhớ trên PC (tiếp)

- ROM BIOS chứa các chương trình sau:
 - Chương trình POST (Power On Self Test)
 - Chương trình CMOS Setup
 - Chương trình Bootstrap loader
 - Các trình điều khiển vào-ra cơ bản (BIOS)
- CMOS RAM:
 - Chứa thông tin cấu hình hệ thống
 - Đồng hồ hệ thống
 - Có pin nuôi riêng
- Video RAM: quản lý thông tin của màn hình
- Các loại bộ nhớ ngoài



Hết chương 2

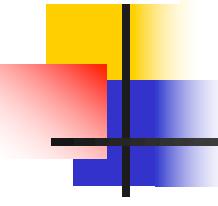


Hệ thống máy tính

Chương 3

HỆ THỐNG VÀO-RA

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội



Nội dung học phần

Chương 1. Tổng quan hệ thống máy tính

Chương 2. Bộ nhớ máy tính

Chương 3. Hệ thống vào-ra

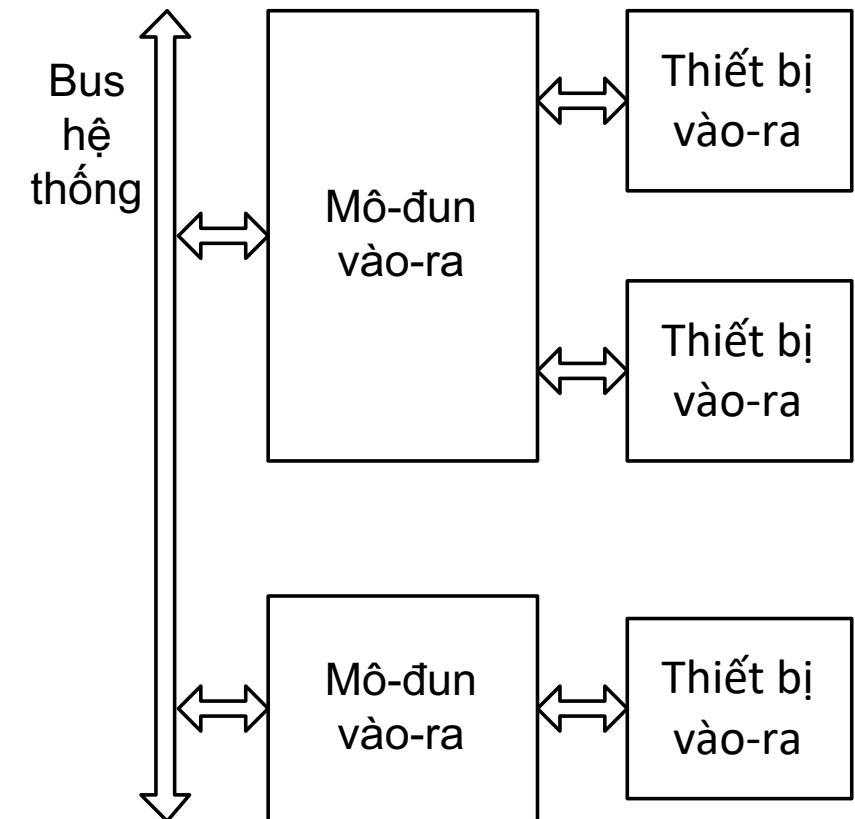
Chương 4. Các kiến trúc song song

Nội dung của chương 3

- 3.1. Tổng quan về hệ thống vào-ra
- 3.2. Các phương pháp điều khiển vào-ra
- 3.3. Nối ghép thiết bị vào-ra

3.1. Tổng quan về hệ thống vào-ra

- Chức năng: Trao đổi thông tin giữa máy tính với bên ngoài
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị vào-ra
 - Các mô-đun vào-ra



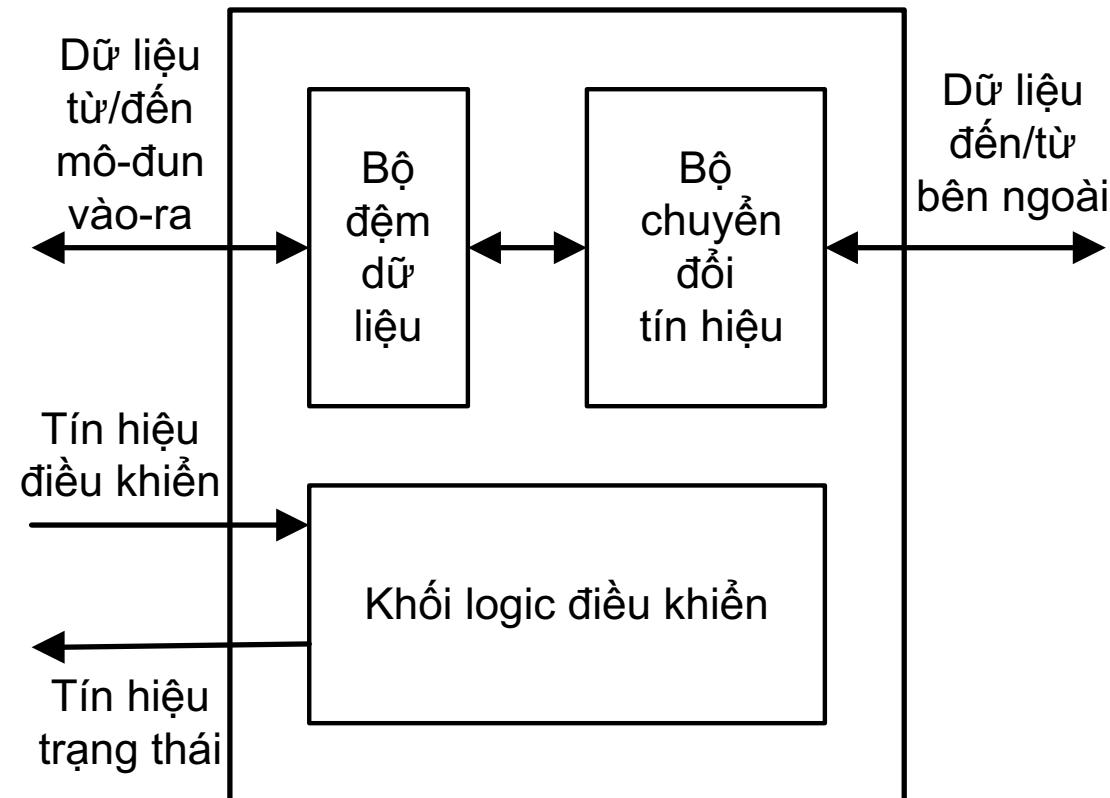
Đặc điểm của hệ thống vào-ra

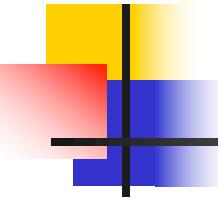
- Tồn tại đa dạng các thiết bị vào-ra khác nhau về:
 - Nguyên tắc hoạt động
 - Tốc độ
 - Khuôn dạng dữ liệu
- Tất cả các thiết bị vào-ra đều chậm hơn CPU và RAM
→ Cần có các mô-đun vào-ra để nối ghép các thiết bị với CPU và bộ nhớ chính

Thiết bị vào-ra

- Còn gọi là thiết bị ngoại vi (Peripherals)
- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Phân loại:
 - Thiết bị vào (Input Devices)
 - Thiết bị ra (Output Devices)
 - Thiết bị lưu trữ (Storage Devices)
 - Thiết bị truyền thông (Communication Devices)
- Giao tiếp:
 - Người - máy
 - Máy - máy

Cấu trúc chung của thiết bị vào-ra

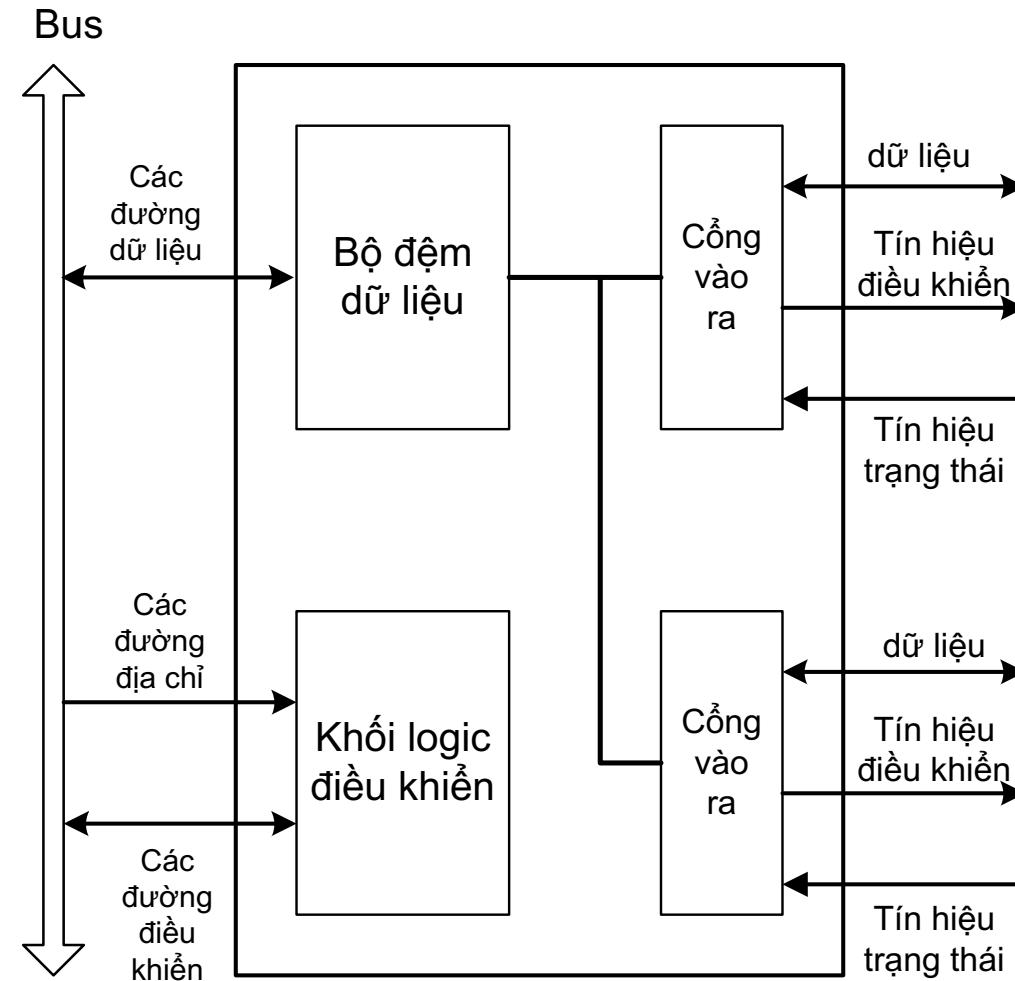




Mô-đun vào-ra

- Chức năng:
 - Điều khiển và định thời
 - Trao đổi thông tin với CPU hoặc bộ nhớ chính
 - Trao đổi thông tin với thiết bị vào-ra
 - Đệm giữa bên trong máy tính với thiết bị vào-ra
 - Phát hiện lỗi của thiết bị vào-ra

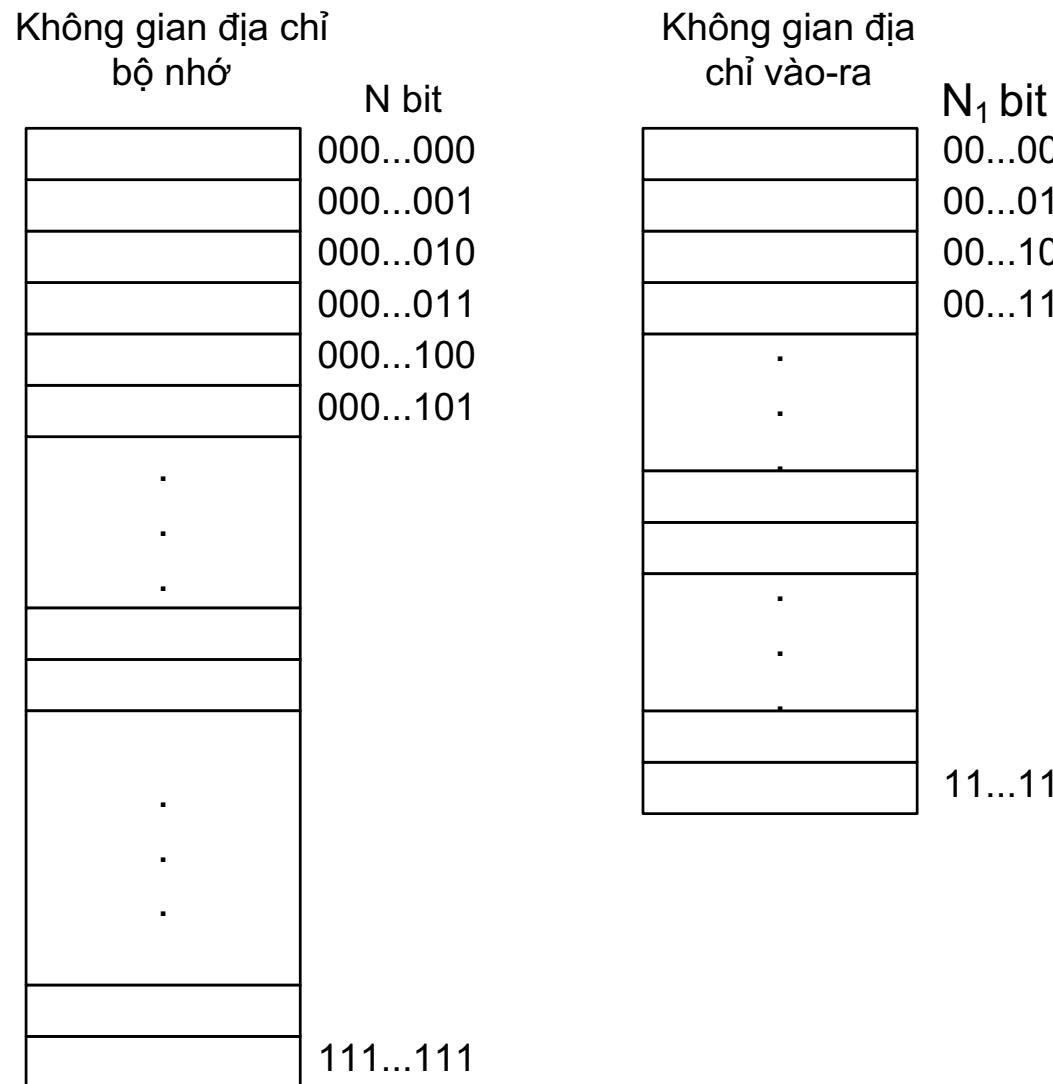
Cấu trúc của mô-đun vào-ra

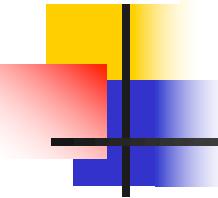


4. Địa chỉ hóa cổng vào-ra (IO addressing)

- Hầu hết các bộ xử lý chỉ có một không gian địa chỉ chung cho cả các ngăn nhớ và các cổng vào-ra
 - Các bộ xử lý 680x0 của Motorola
 - Các bộ xử lý theo kiến trúc RISC: MIPS, ARM, ...
- Một số bộ xử lý có hai không gian địa chỉ tách biệt:
 - Không gian địa chỉ bộ nhớ
 - Không gian địa chỉ vào-ra
 - Ví dụ: Intel x86

Không gian địa chỉ tách biệt





Các phương pháp địa chỉ hóa cổng vào-ra

- Vào-ra theo bản đồ bộ nhớ
(Memory mapped IO)
- Vào-ra riêng biệt
(Isolated IO hay IO mapped IO)

Vào-ra theo bản đồ bộ nhớ

- Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ bộ nhớ
- CPU coi cổng vào-ra như ngăn nhớ
- Lập trình trao đổi dữ liệu với cổng vào-ra bằng các lệnh truy nhập dữ liệu bộ nhớ
- Có thể thực hiện trên mọi hệ thống
- Ví dụ: Bộ xử lý MIPS
 - 32-bit địa chỉ cho một không gian địa chỉ chung cho cả các ngăn nhớ và các cổng vào-ra
 - Các cổng vào-ra được gắn các địa chỉ thuộc vùng địa chỉ dự trữ
 - Vào/ra dữ liệu: sử dụng lệnh load/store

Ví dụ lập trình vào-ra cho MIPS

- Ví dụ: Có hai cổng vào-ra được gán địa chỉ:
 - Cổng 1: 0xFFFFFFFF4
 - Cổng 2: 0xFFFFFFFF8
- Ghi giá trị 0x41 ra cổng 1

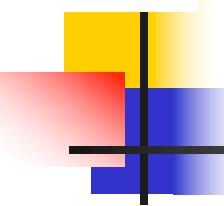
```
addi $t0, $0, 0x41      # đưa giá trị 0x41  
sw    $t0, 0xFFFF4($0)   # ra cổng 1
```

Chú ý: giá trị 16-bit 0xFFFF4 được sign-extended thành 32-bit 0xFFFFFFFF4

- Đọc dữ liệu từ cổng 2 đưa vào \$t3
- ```
lw $t3, 0xFFFF8($0) # đọc dữ liệu cổng 2 đưa vào $t3
```

# Vào-ra riêng biệt (Isolated IO)

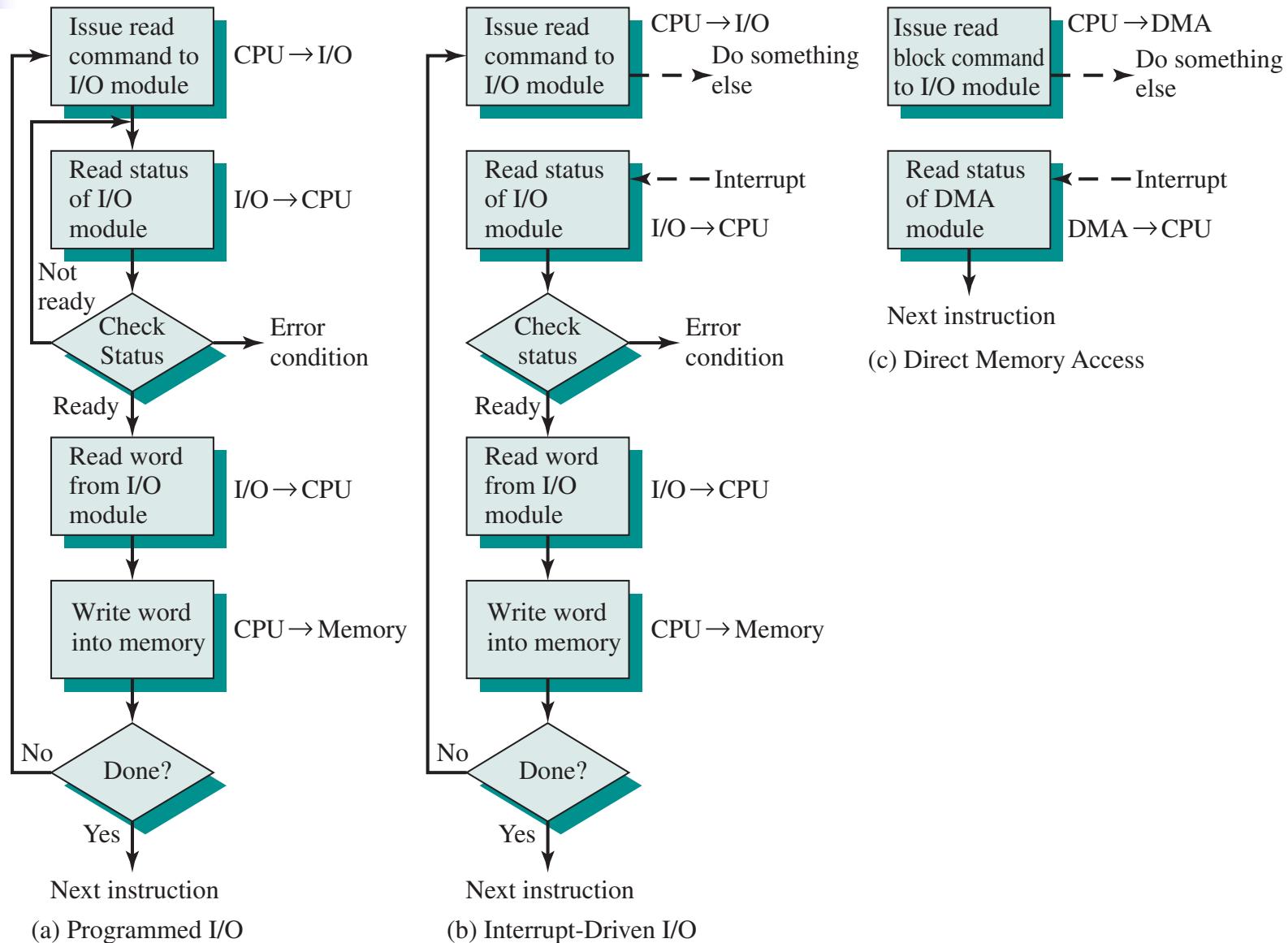
- Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ vào-ra riêng
- Lập trình trao đổi dữ liệu với cổng vào-ra bằng các lệnh vào-ra chuyên dụng
- Ví dụ: Intel x86
  - Dùng 8-bit hoặc 16-bit địa chỉ cho không gian địa chỉ vào-ra riêng
  - Có hai lệnh vào-ra chuyên dụng
    - Lệnh IN: nhận dữ liệu từ cổng vào
    - Lệnh OUT: đưa dữ liệu đến cổng ra



## 3.2. Các phương pháp điều khiển vào-ra

- Vào-ra bằng chương trình  
(Programmed IO)
- Vào-ra điều khiển bằng ngắt  
(Interrupt Driven IO)
- Truy nhập bộ nhớ trực tiếp - DMA  
(Direct Memory Access)

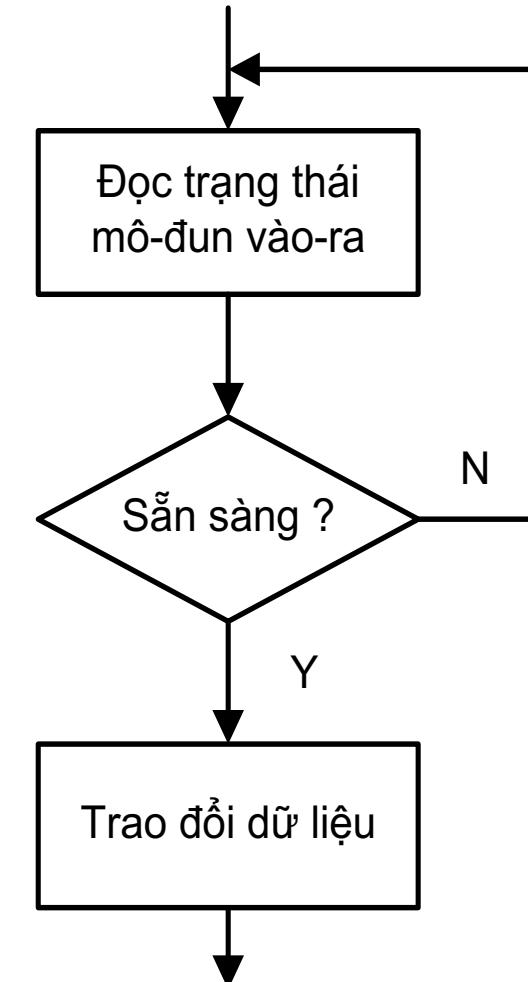
# Ba kỹ thuật thực hiện vào một khối dữ liệu



# 1. Vào-ra bằng chương trình

## ■ Nguyên tắc chung:

- CPU điều khiển trực tiếp vào-ra bằng chương trình → cần phải lập trình vào-ra để trao đổi dữ liệu giữa CPU với mô-đun vào-ra
- CPU nhanh hơn thiết bị vào-ra rất nhiều lần, vì vậy trước khi thực hiện lệnh vào-ra, chương trình cần đọc và kiểm tra trạng thái sẵn sàng của mô-đun vào-ra

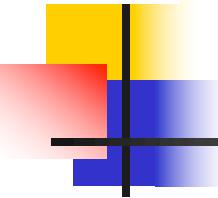


# Các tín hiệu điều khiển vào-ra

- Tín hiệu **điều khiển (Control)**: kích hoạt thiết bị vào-ra
- Tín hiệu **kiểm tra (Test)**: kiểm tra trạng thái của mô-đun vào-ra và thiết bị vào-ra
- Tín hiệu điều khiển **đọc (Read)**: yêu cầu mô-đun vào-ra nhận dữ liệu từ thiết bị vào-ra và đưa vào bộ đệm dữ liệu, rồi CPU nhận dữ liệu đó
- Tín hiệu điều khiển **ghi (Write)**: yêu cầu mô-đun vào-ra lấy dữ liệu trên bus dữ liệu đưa đến bộ đệm dữ liệu rồi chuyển ra thiết bị vào-ra

# Các lệnh vào-ra

- Với vào-ra theo bản đồ bộ nhớ: sử dụng các lệnh trao đổi dữ liệu với bộ nhớ để trao đổi dữ liệu với cổng vào-ra
- Với vào-ra riêng biệt: sử dụng các lệnh vào-ra chuyên dụng (IN, OUT)



## Đặc điểm

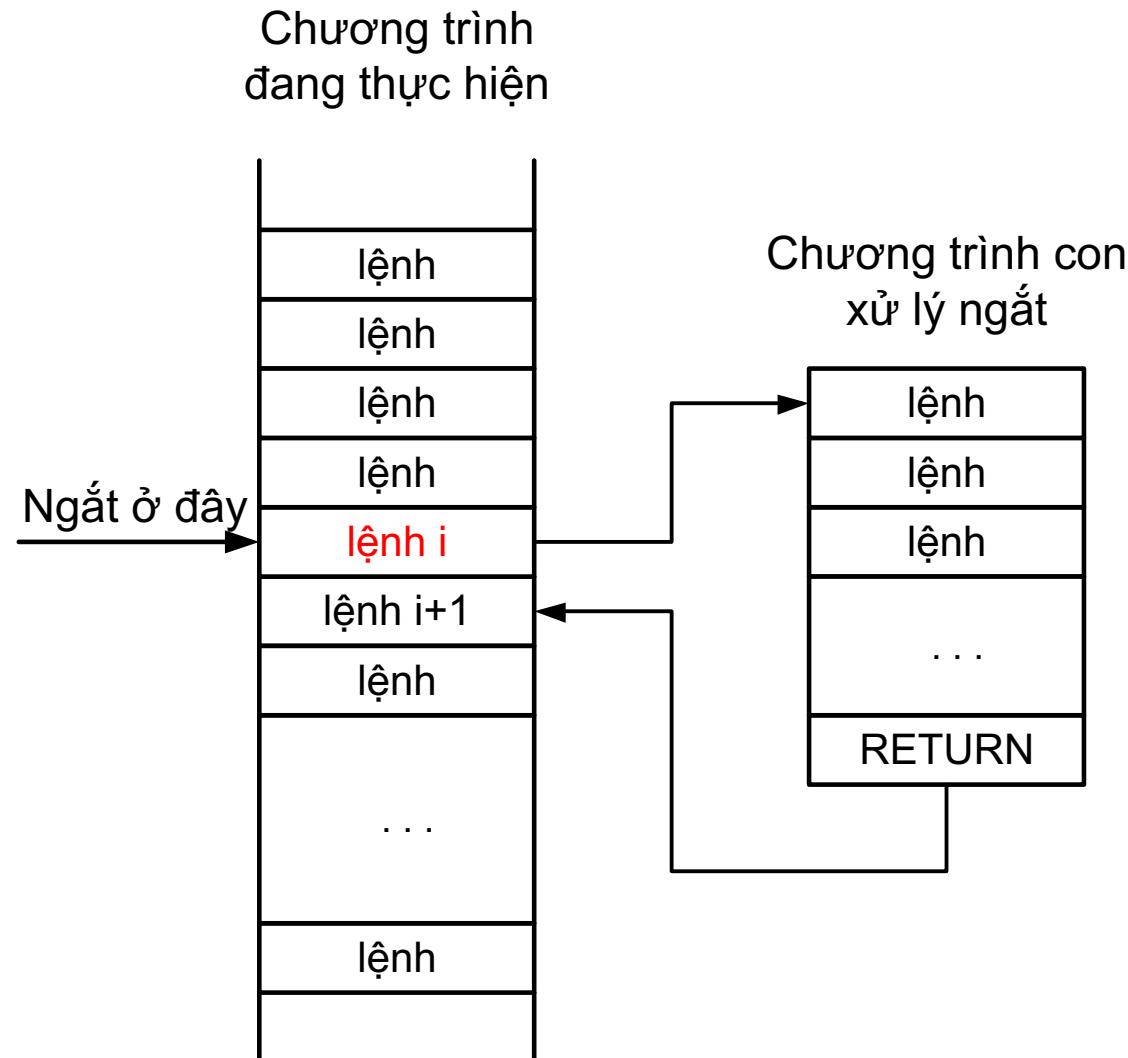
- Vào-ra do ý muốn của người lập trình
- CPU trực tiếp điều khiển trao đổi dữ liệu giữa CPU với mô-đun vào-ra
- CPU đợi mô-đun vào-ra → tiêu tốn nhiều thời gian của CPU

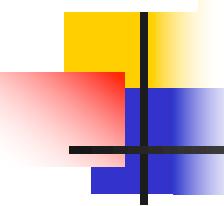
## 2. Vào-ra điều khiển bằng ngắt

### ■ Nguyên tắc chung:

- CPU không phải đợi trạng thái sẵn sàng của mô-đun vào-ra, CPU thực hiện một chương trình nào đó
- Khi mô-đun vào-ra sẵn sàng thì nó phát tín hiệu ngắt CPU
- CPU thực hiện chương trình con xử lý ngắt vào-ra tương ứng để trao đổi dữ liệu
- CPU trở lại tiếp tục thực hiện chương trình đang bị ngắt

# Chuyển điều khiển đến chương trình con ngắt



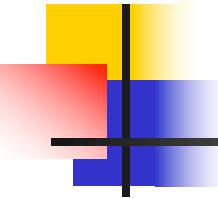


## Hoạt động vào dữ liệu: nhìn từ mô-đun vào-ra

- Mô-đun vào-ra nhận tín hiệu điều khiển đọc từ CPU
- Mô-đun vào-ra nhận dữ liệu từ thiết bị vào-ra, trong khi đó CPU làm việc khác
- Khi đã có dữ liệu → mô-đun vào-ra phát tín hiệu ngắt CPU
- CPU yêu cầu dữ liệu
- Mô-đun vào-ra chuyển dữ liệu đến CPU

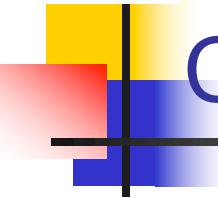
# Hoạt động vào dữ liệu: nhìn từ CPU

- Phát tín hiệu điều khiển **đọc**
- Làm việc khác
- Cuối mỗi chu trình lệnh, kiểm tra tín hiệu yêu cầu ngắt
- Nếu bị ngắt:
  - Cắt ngũ cảnh (nội dung các thanh ghi liên quan)
  - Thực hiện chương trình con xử lý ngắt để vào dữ liệu
  - Khôi phục ngũ cảnh của chương trình đang thực hiện



## Các vấn đề nảy sinh khi thiết kế

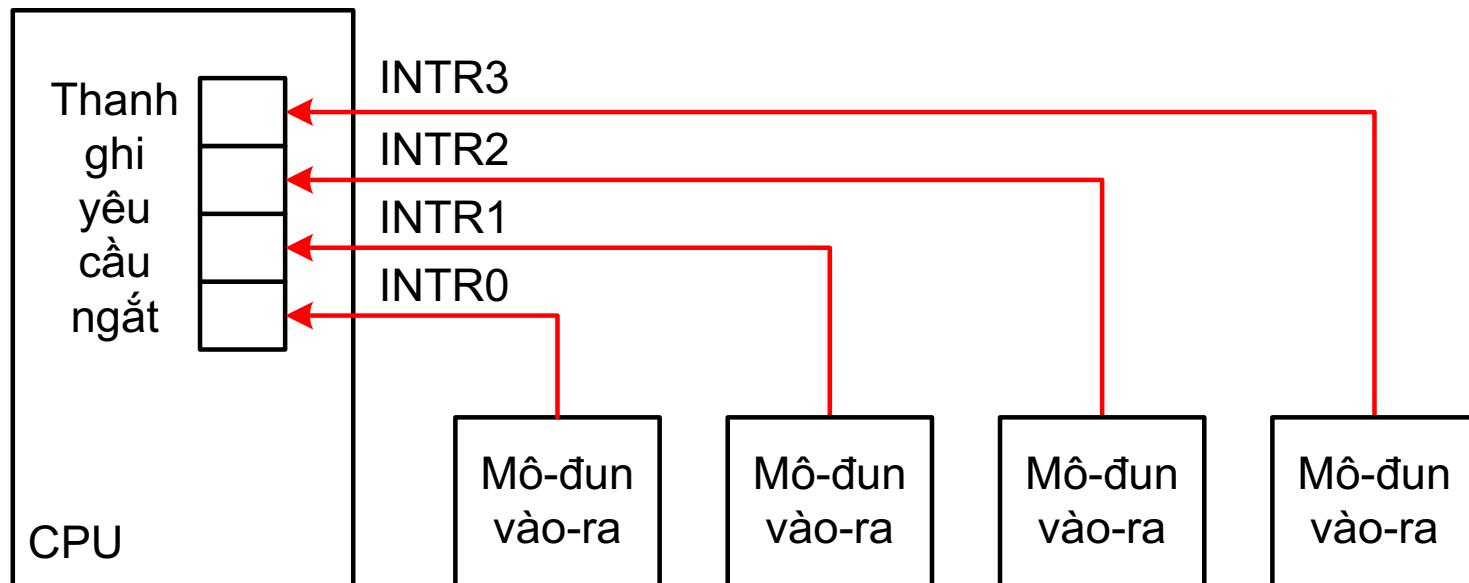
- Làm thế nào để xác định được mô-đun vào-ra nào phát tín hiệu ngắt ?
- CPU làm như thế nào khi có nhiều yêu cầu ngắt cùng xảy ra ?



# Các phương pháp nối ghép ngắn

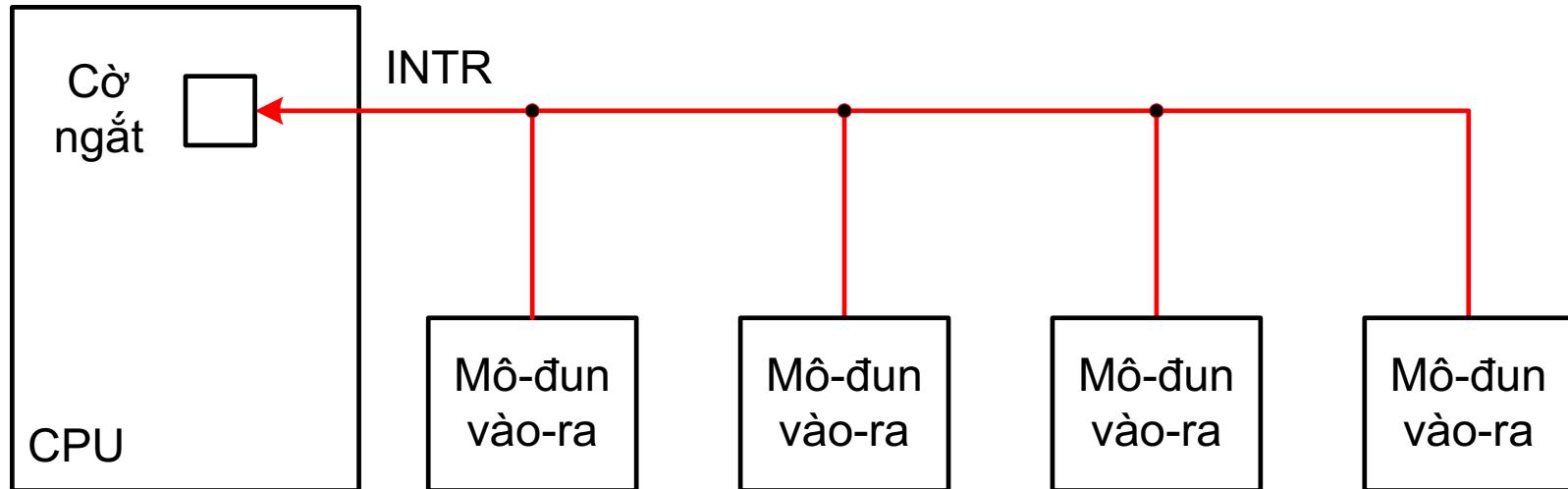
- Sử dụng nhiều đường yêu cầu ngắn
- Hỏi vòng bằng phần mềm (Software Poll)
- Hỏi vòng bằng phần cứng (Daisy Chain or Hardware Poll)
- Sử dụng bộ điều khiển ngắn (PIC)

# Nhiều đường yêu cầu ngắt



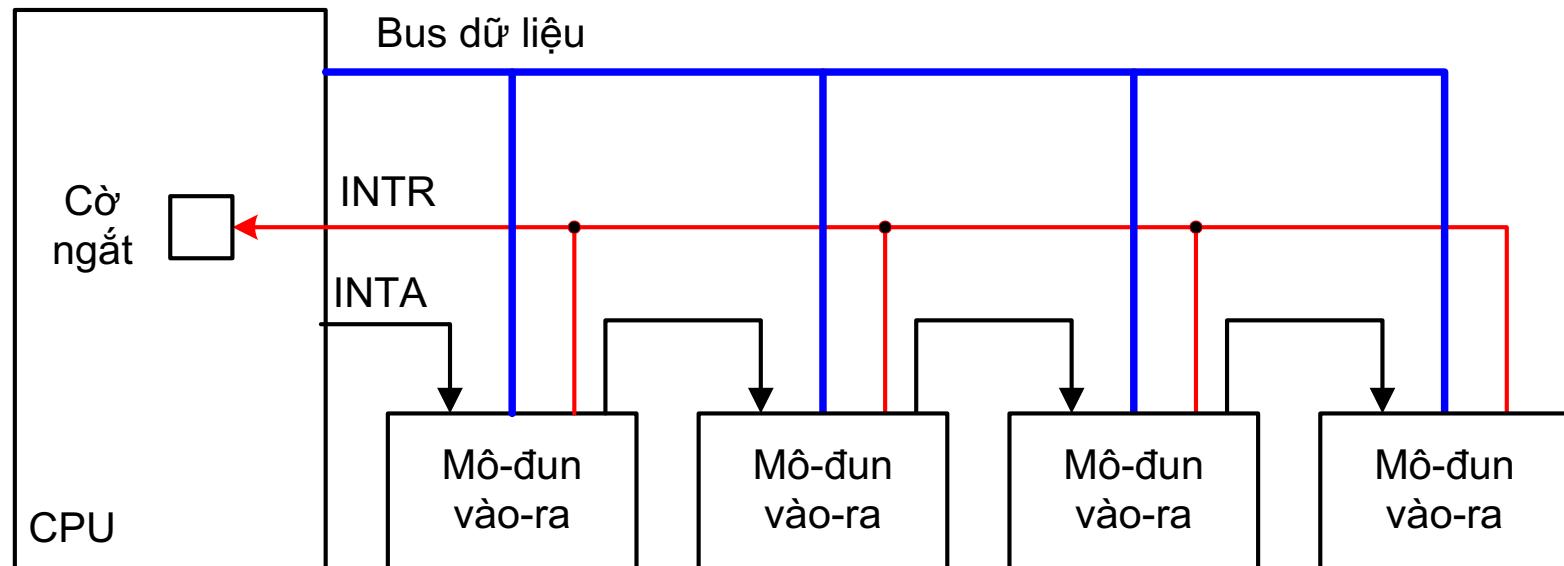
- Mỗi mô-đun vào-ra được nối với một đường yêu cầu ngắt
- CPU phải có nhiều đường tín hiệu yêu cầu ngắt
- Hạn chế số lượng mô-đun vào-ra
- Các đường ngắt được qui định mức ưu tiên

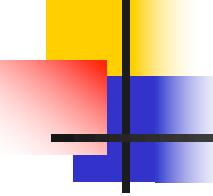
# Hỏi vòng bằng phần mềm



- CPU thực hiện phần mềm hỏi lắc lượt từng mô-đun vào-ra
- Chậm
- Thứ tự các mô-đun được hỏi vòng chính là thứ tự ưu tiên

# Hỏi vòng bằng phần cứng

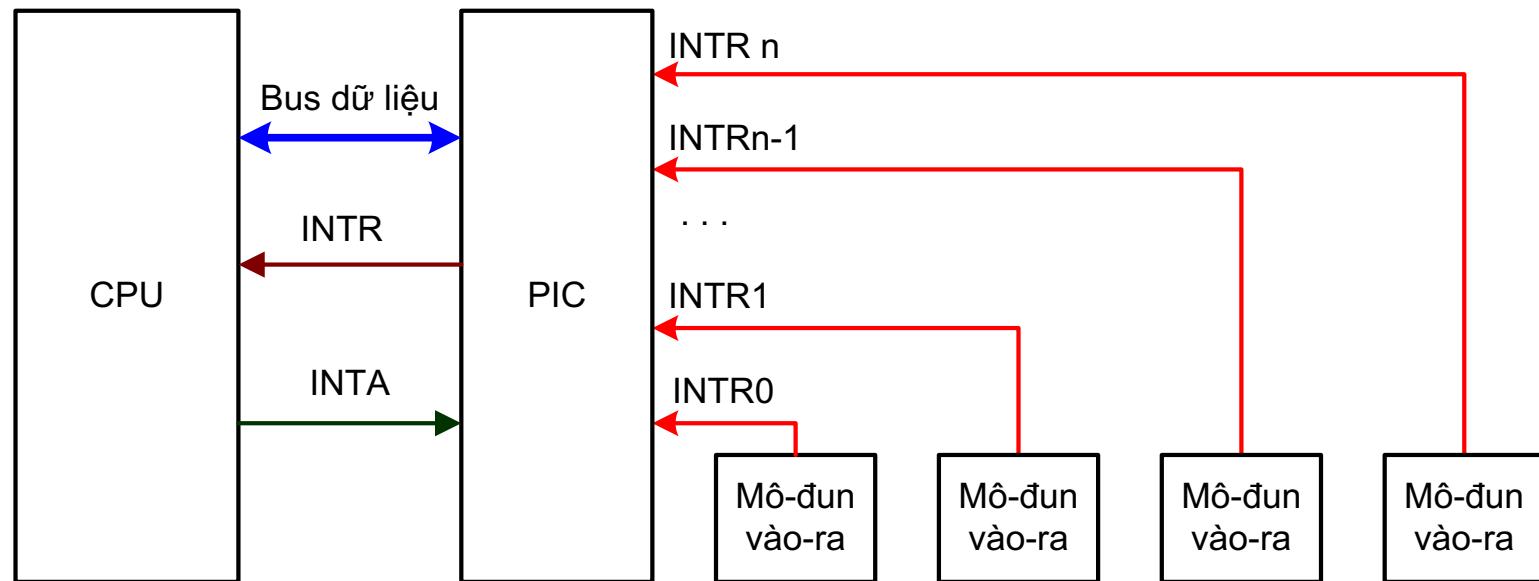




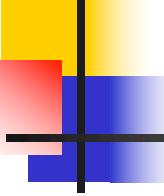
## Hỏi vòng bằng phần cứng (tiếp)

- CPU phát tín hiệu chấp nhận ngắt (INTA) đến mô-đun vào-ra đầu tiên
- Nếu mô-đun vào-ra đó không gây ra ngắt thì nó gửi tín hiệu đến mô-đun kế tiếp cho đến khi xác định được mô-đun gây ngắt
- Thứ tự các mô-đun 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 vào yêu cầu ngắt có qui định mức ưu tiên
- PIC chọn một yêu cầu ngắt không bị cấm có mức ưu tiên cao nhất gửi tới CPU



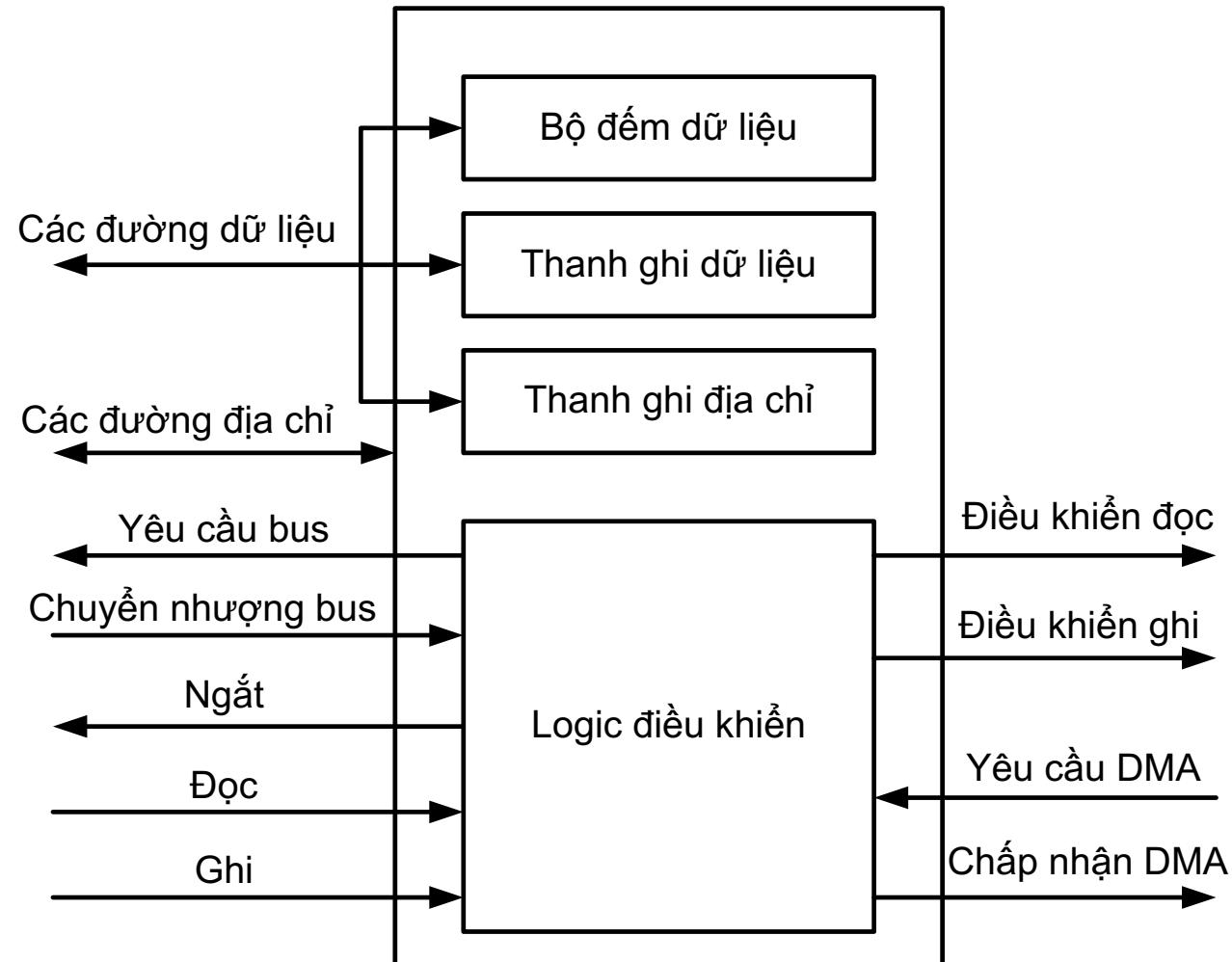
## Đặc điểm của vào-ra điều khiển bằng ngắt

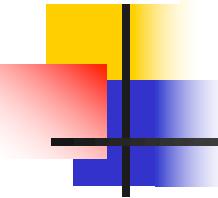
- Có sự kết hợp giữa phần cứng và phần mềm
  - Phần cứng: gây ngắt CPU
  - Phần mềm: trao đổi dữ liệu giữa CPU với mô-đun vào-ra
- CPU trực tiếp điều khiển vào-ra
- CPU không phải đợi mô-đun vào-ra, do đó hiệu quả sử dụng CPU tốt hơn

### 3. DMA (Direct Memory Access)

- Vào-ra bằng chương trình và bằng ngắt do CPU trực tiếp điều khiển:
  - Chiếm thời gian của CPU
- Để khắc phục dùng kỹ thuật DMA
  - Sử dụng mô-đun điều khiển vào-ra chuyên dụng, gọi là DMAC (Controller), điều khiển trao đổi dữ liệu giữa mô-đun vào-ra với bộ nhớ chính

# Sơ đồ cấu trúc của DMA





## Các thành phần của DMAC

- Thanh ghi dữ liệu: chứa dữ liệu 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 DMA

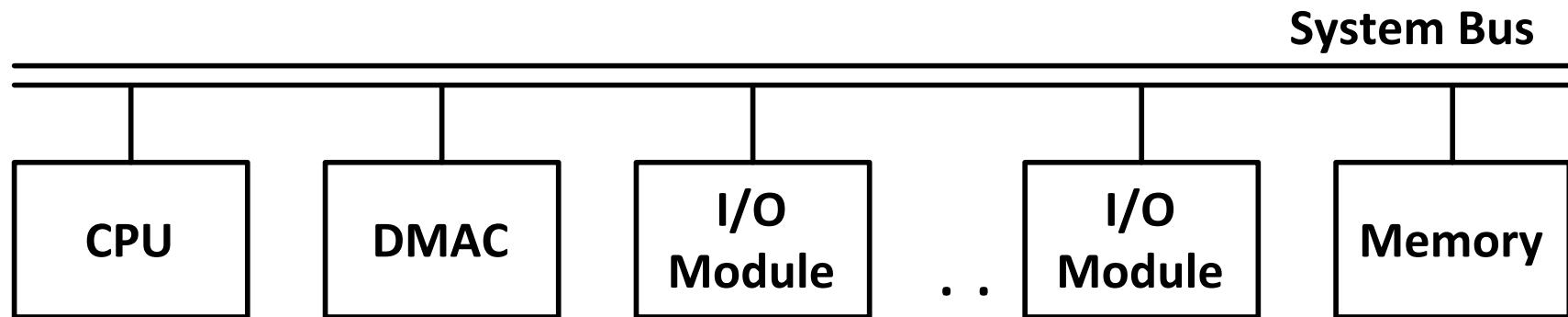
## CPU “nói” cho DMAC

- Vào hay Ra dữ liệu
- Địa chỉ thiết bị vào-ra (cổng vào-ra tương ứng)
- Địa chỉ đầu của mảng nhớ chứa dữ liệu → nạp vào thanh ghi địa chỉ
- Số từ dữ liệu cần truyền → nạp vào bộ đếm dữ liệu
- CPU làm việc khác
- DMAC điều khiển trao đổi dữ liệu
- Sau khi truyền được một từ dữ liệu thì:
  - nội dung thanh ghi địa chỉ tăng
  - nội dung bộ đếm dữ liệu giảm
- Khi bộ đếm dữ liệu = 0, DMAC gửi tín hiệu ngắt CPU để báo kết thúc DMA

# Các kiểu thực hiện DMA

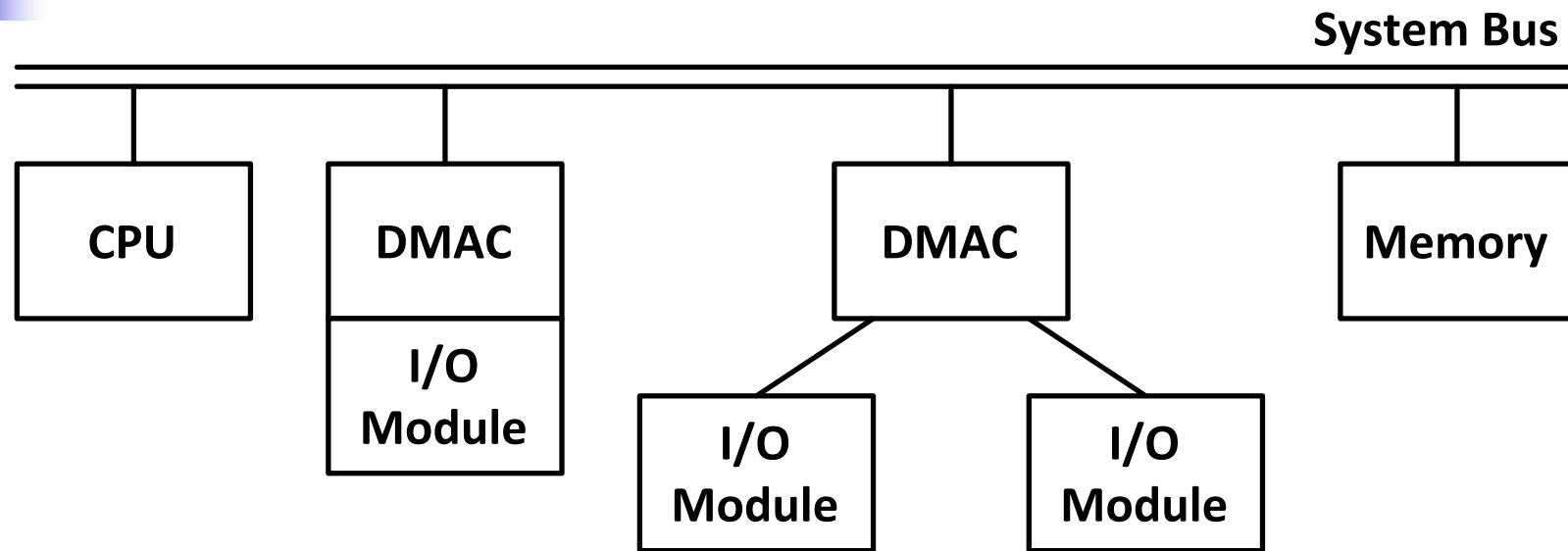
- DMA truyền theo khối (Block-transfer DMA): DMAC sử dụng bus để truyền xong cả khối dữ liệu
- DMA lấy chu kỳ (Cycle Stealing DMA): DMAC cưỡng bức CPU treo tạm thời từng chu kỳ bus, DMAC chiếm bus thực hiện truyền một từ dữ liệu.
- DMA trong suốt (Transparent DMA): DMAC nhận biết những chu kỳ nào CPU không sử dụng bus thì chiếm bus để trao đổi một từ dữ liệu.

# Cấu hình DMA (1)



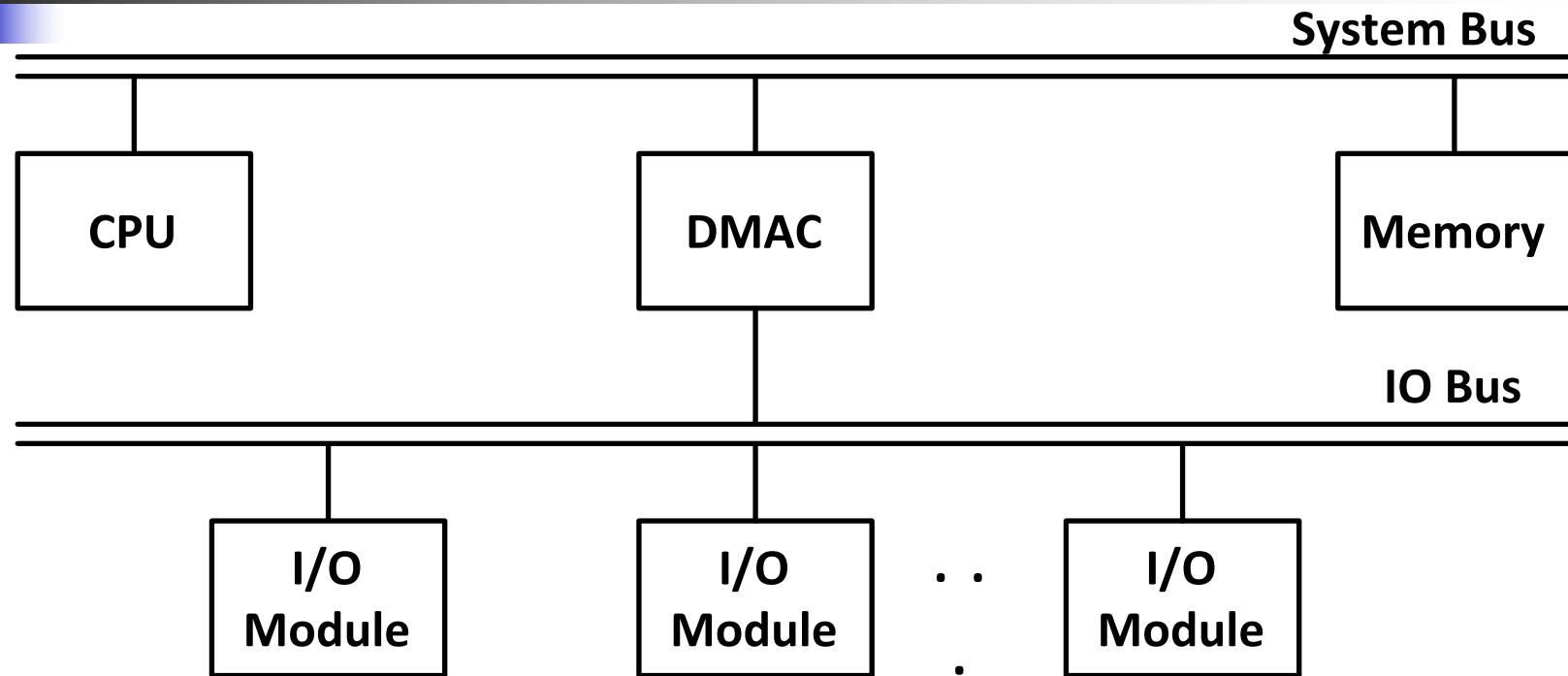
- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus hai lần
  - Giữa mô-đun vào-ra với DMAC
  - Giữa DMAC với bộ nhớ

# Cấu hình DMA (2)

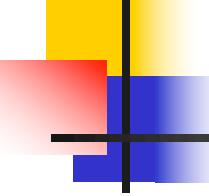


- DMAC điều khiển một hoặc vài mô-đun vào-ra
- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus một lần
  - Giữa DMAC với bộ nhớ

# Cấu hình DMA (3)



- Bus vào-ra tách rời hỗ trợ tất cả các thiết bị cho phép DMA
- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus một lần
  - Giữa DMAC với bộ nhớ



## Đặc điểm của DMA

- CPU không tham gia trong quá trình trao đổi dữ liệu
- DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính với mô-đun vào-ra (hoàn toàn bằng phần cứng) → tốc độ nhanh
- Phù hợp với các yêu cầu trao đổi mảng dữ liệu có kích thước lớn

## 4. Bộ xử lý vào-ra

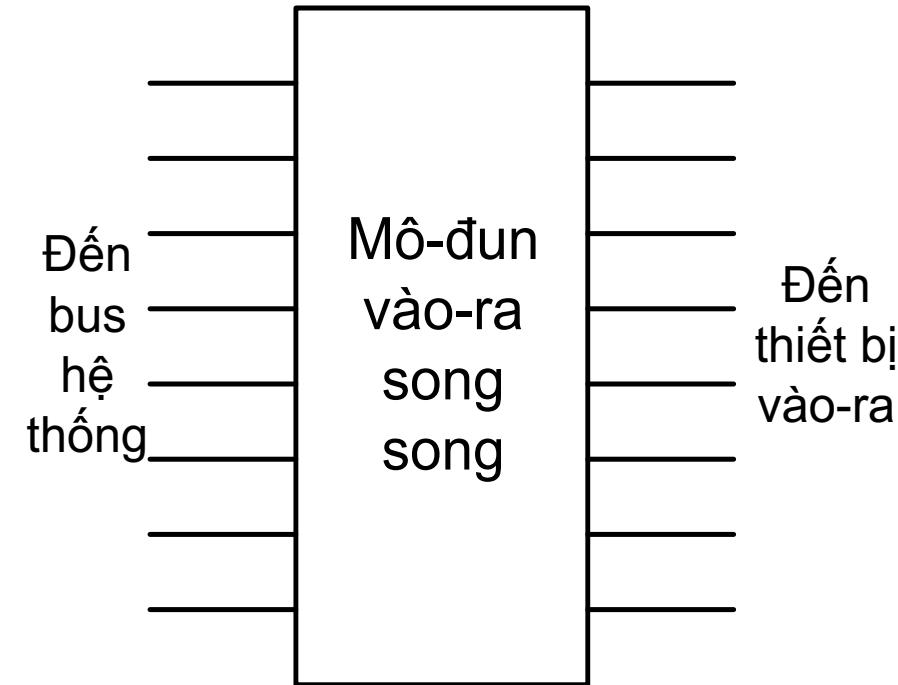
- Việc điều khiển vào-ra được thực hiện bởi một bộ xử lý vào-ra chuyên dụng
- Bộ xử lý vào-ra hoạt động theo chương trình của riêng nó
- Chương trình của bộ xử lý vào-ra có thể nằm trong bộ nhớ chính hoặc nằm trong một bộ nhớ riêng

### 3.3. Nối ghép thiết bị vào-ra

#### 1. Các kiểu nối ghép vào-ra

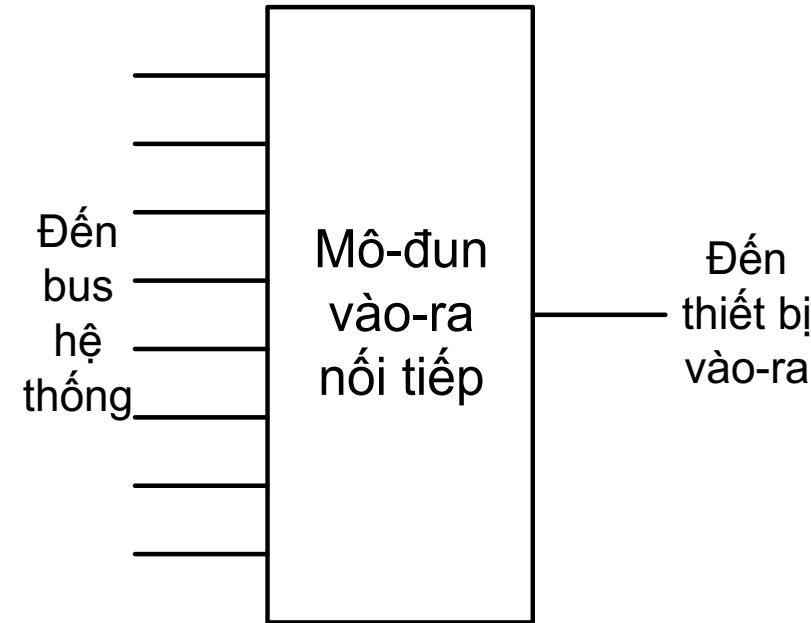
- Nối ghép song song
- Nối ghép nối tiếp

# Nối ghép song song



- Truyền nhiều bit song song
- Tốc độ nhanh
- Cần nhiều đường truyền dữ liệu

# Nối ghép nối tiếp

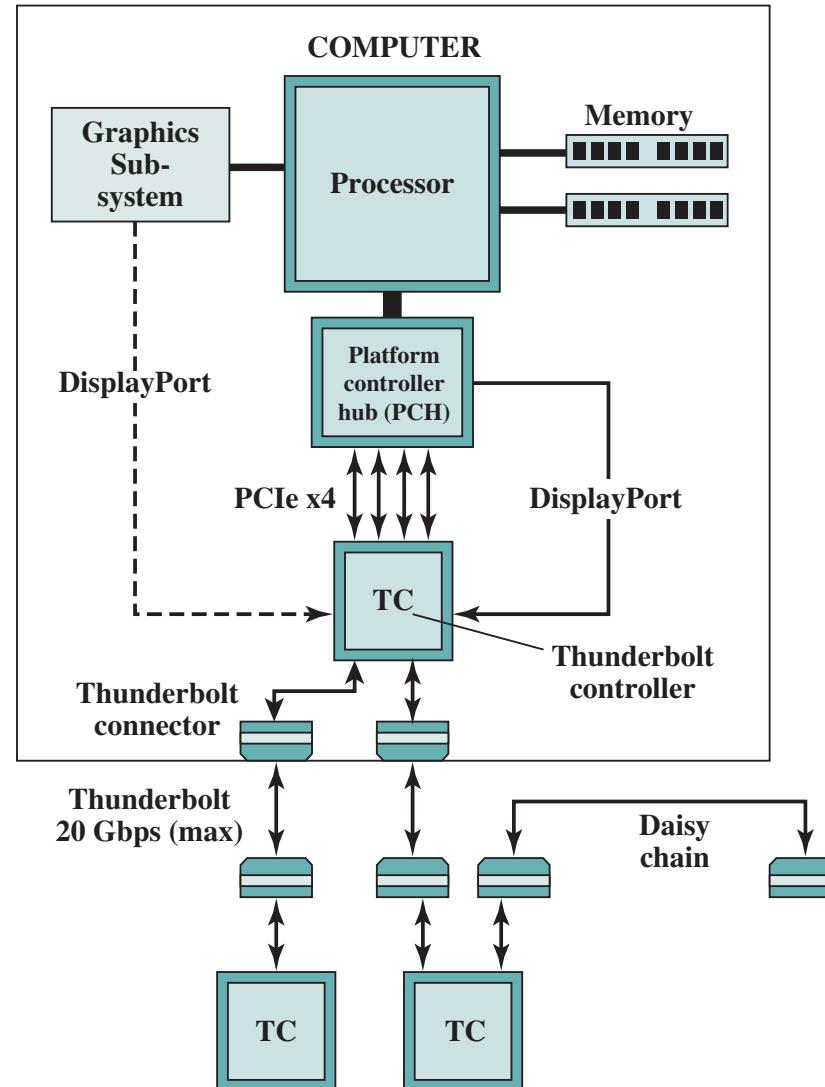


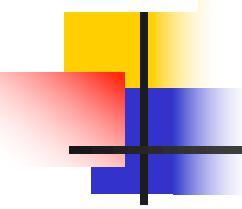
- Truyền lần lượt từng bit
- Cần có bộ chuyển đổi từ dữ liệu song song sang nối tiếp hoặc/và ngược lại
- Tốc độ chậm hơn
- Cần ít đường truyền dữ liệu

## 2. Các cấu hình nối ghép

- **Điểm tới điểm (Point to Point)**
  - Thông qua một cổng vào-ra nối ghép với một thiết bị
- **Điểm tới đa điểm (Point to Multipoint)**
  - Thông qua một cổng vào-ra cho phép nối ghép được với nhiều thiết bị
  - Ví dụ:
    - USB (Universal Serial Bus): 127 thiết bị
    - IEEE 1394 (FireWire): 63 thiết bị
    - Thunderbolt

# Thunderbolt





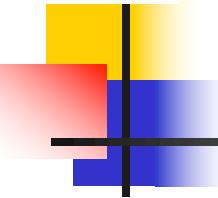
# Hết chương 3

## Chương 4

# CÁC KIẾN TRÚC SONG SONG

Nguyễn Kim Khánh

Trường Đại học Bách khoa Hà Nội



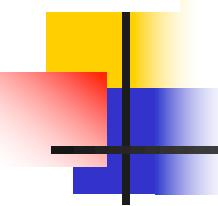
## Nội dung học phần

Chương 1. Tổng quan hệ thống máy tính

Chương 2. Bộ nhớ máy tính

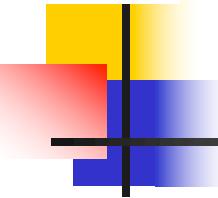
Chương 3. Hệ thống vào-ra

Chương 4. Các kiến trúc song song



# Nội dung của chương 4

- 4.1. Phân loại kiến trúc máy tính
- 4.2. Đa xử lý bộ nhớ dùng chung
- 4.3. Đa xử lý bộ nhớ phân tán
- 4.4. Bộ xử lý đồ họa đa dụng

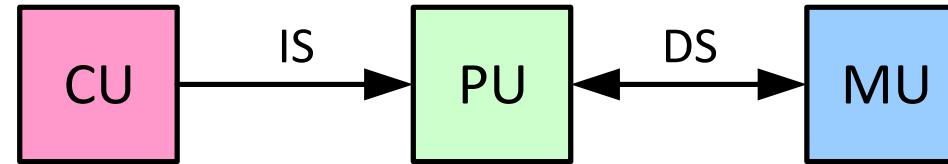


## 4.1. Phân loại kiến trúc máy tính

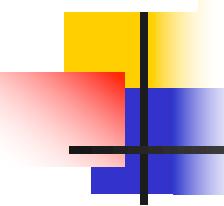
Phân loại kiến trúc máy tính (Michael Flynn -1966)

- SISD - Single Instruction Stream, Single Data Stream
- SIMD - Single Instruction Stream, Multiple Data Stream
- MISD - Multiple Instruction Stream, Single Data Stream
- MIMD - Multiple Instruction Stream, Multiple Data Stream

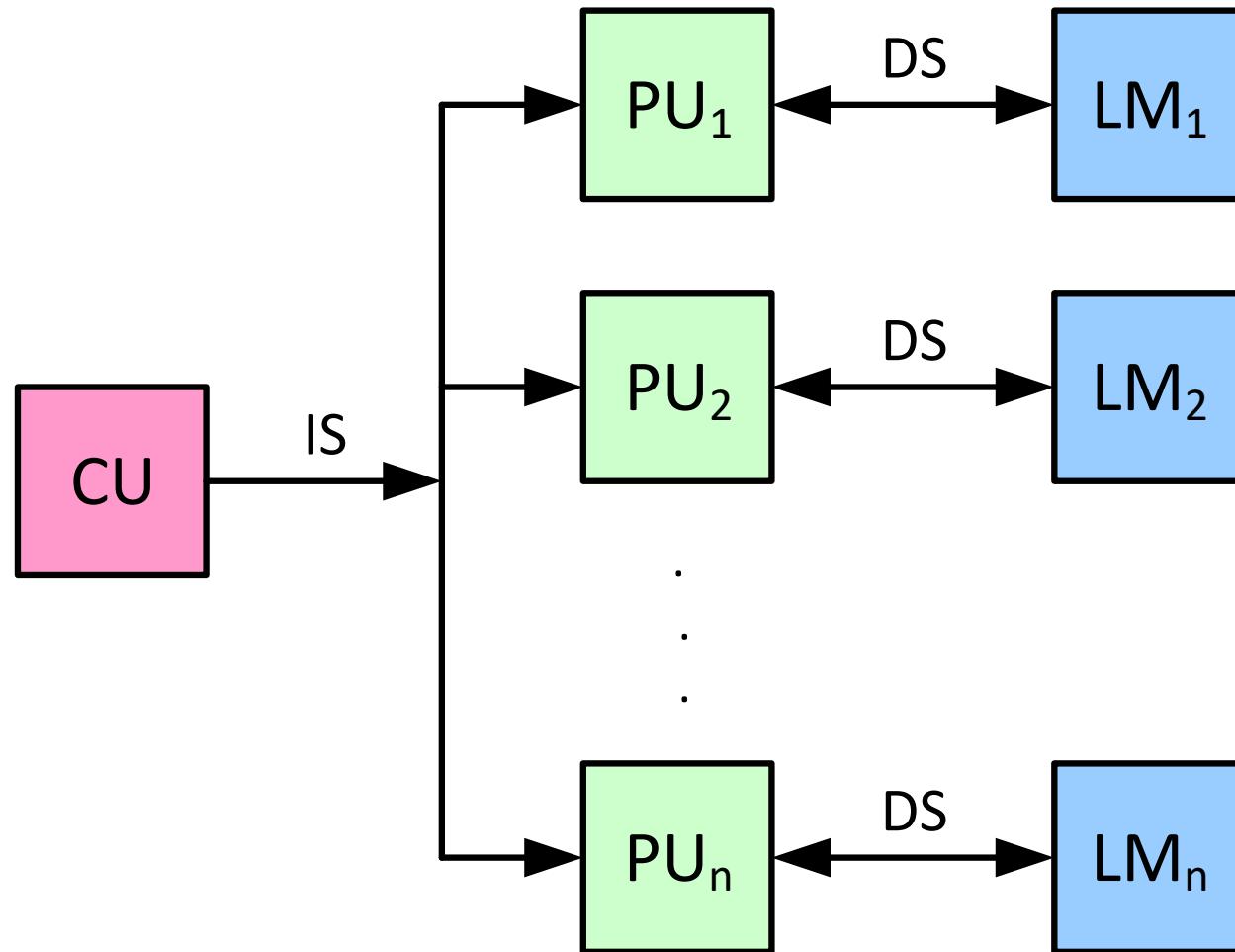
# SISD



- CU: Control Unit
- PU: Processing Unit
- MU: Memory Unit
- Một bộ xử lý
- Đơn dòng lệnh
- Dữ liệu được lưu trữ trong một bộ nhớ
- Chính là Kiến trúc von Neumann (tuần tự)



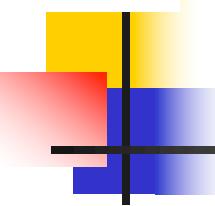
# SIMD



# SIMD (tiếp)

- Đơn dòng lệnh điều khiển đồng thời các đơn vị xử lý PUs
- Mỗi phần tử xử lý có một bộ nhớ dữ liệu riêng LM (local memory)
- Mỗi lệnh được thực hiện trên một tập các dữ liệu khác nhau
- Các mô hình SIMD
  - Vector Computer
  - Array processor

- Một luồng dữ liệu cùng được truyền đến một tập các bộ xử lý
- Mỗi bộ xử lý thực hiện một dãy lệnh khác nhau.
- Chưa tồn tại máy tính thực tế
- Có thể có trong tương lai

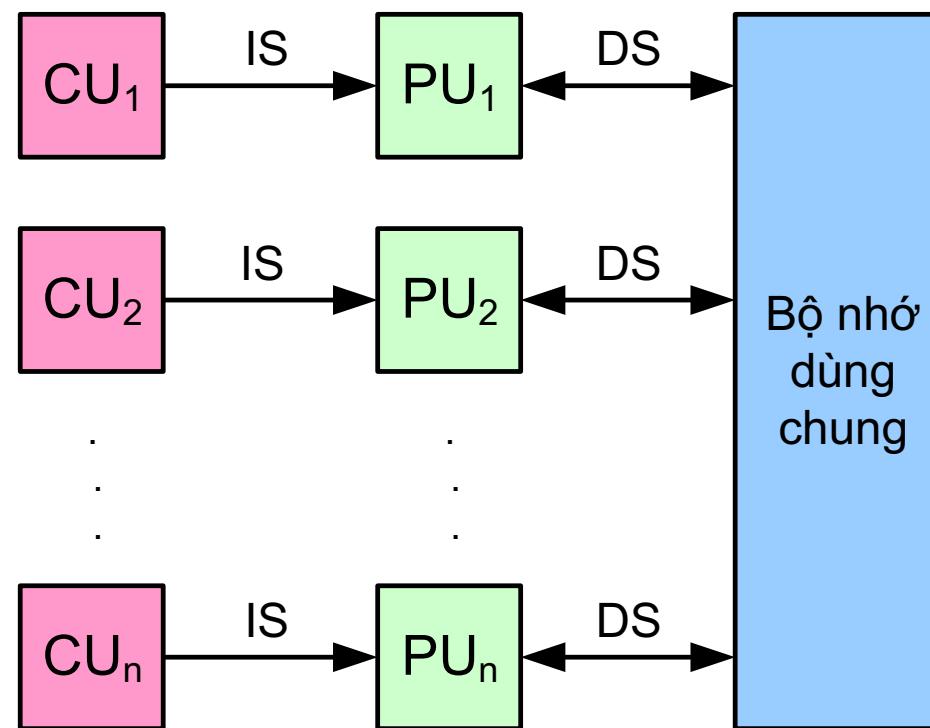


# MIMD

- Tập các bộ xử lý
- Các bộ xử lý đồng thời thực hiện các dãy lệnh khác nhau trên các dữ liệu khác nhau
- Các mô hình MIMD
  - Multiprocessors (Shared Memory)
  - Multicomputers (Distributed Memory)

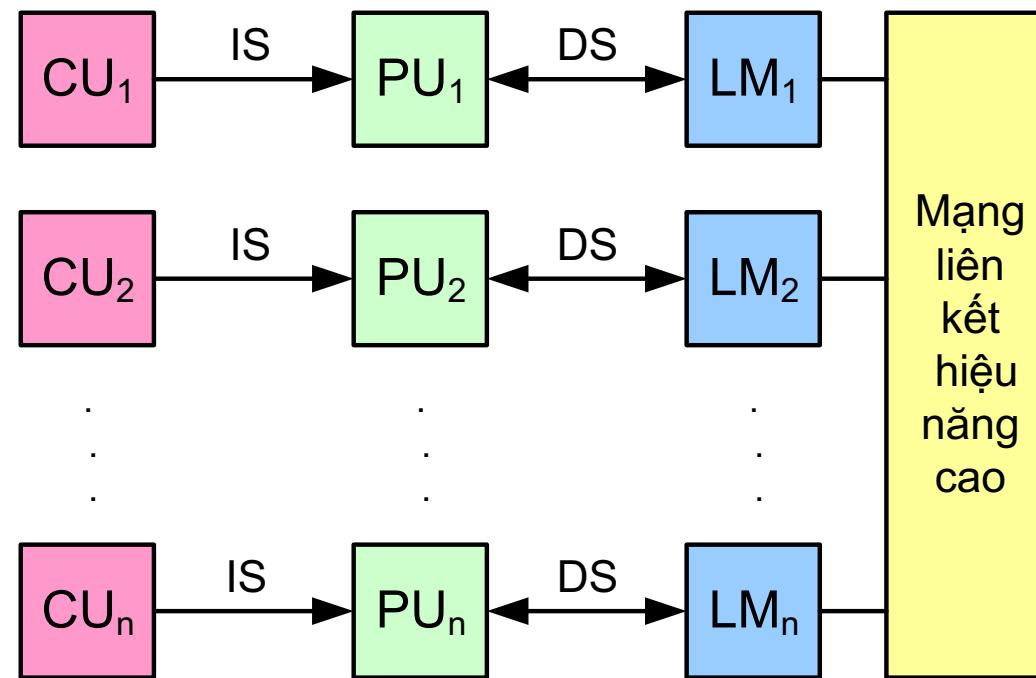
# MIMD - Shared Memory

Đa xử lý bộ nhớ dùng chung  
(shared memory multiprocessors)



# MIMD - Distributed Memory

Đa xử lý bộ nhớ phân tán  
(distributed memory multiprocessors or multiccomputers)



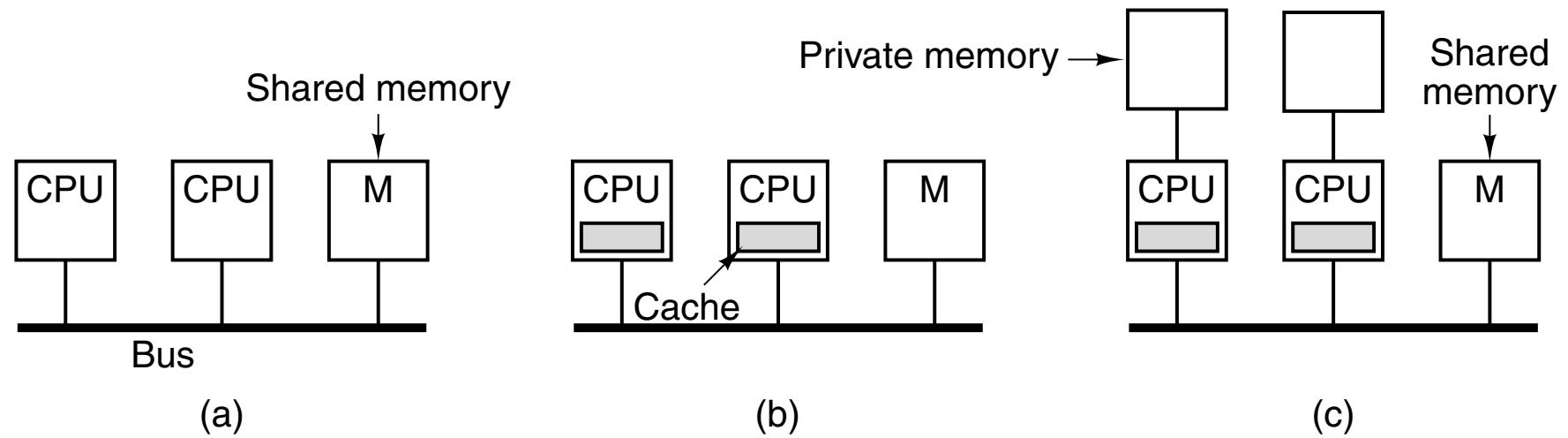
# Phân loại các kỹ thuật song song

- Song song mức lệnh
  - pipeline
  - superscalar
- Song song mức dữ liệu
  - SIMD
- Song song mức luồng
  - MIMD
- Song song mức yêu cầu
  - Cloud computing

## 4.2. Đa xử lý bộ nhớ dùng chung

- Hệ thống đa xử lý đối xứng (SMP – Symmetric Multiprocessors)
- Hệ thống đa xử lý không đối xứng (NUMA – Non-Uniform Memory Access)
- Bộ xử lý đa lõi (Multicore Processors)

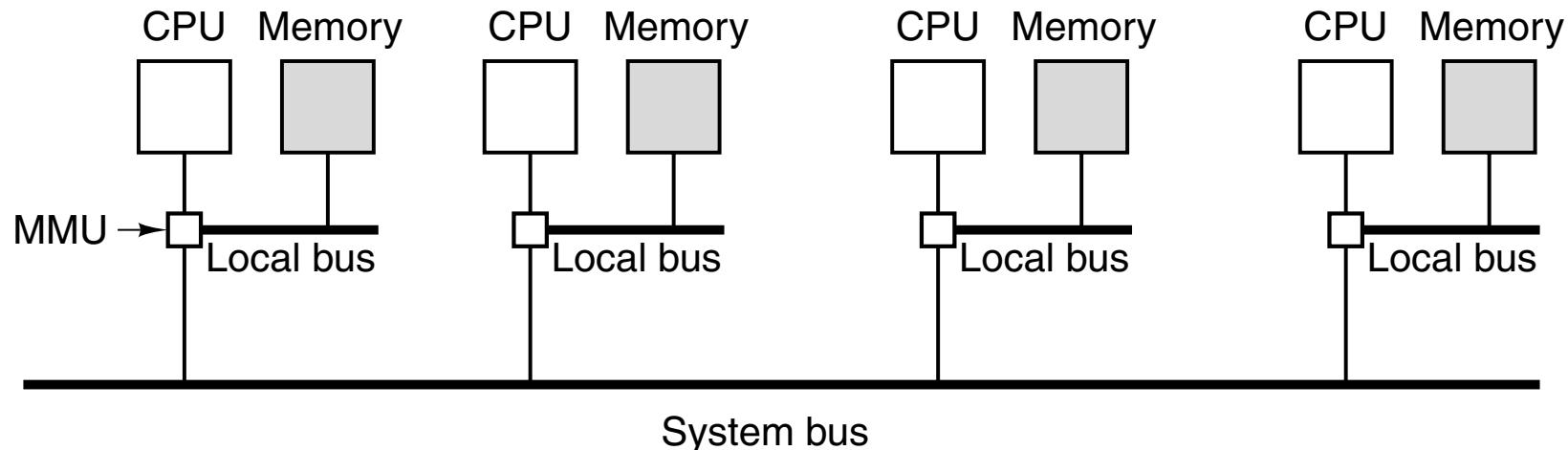
# SMP hay UMA (Uniform Memory Access)



## SMP (tiếp)

- Một máy tính có  $n \geq 2$  bộ xử lý giống nhau
- Các bộ xử lý dùng chung bộ nhớ và hệ thống vào-ra
- Thời gian truy cập bộ nhớ là bằng nhau với các bộ xử lý
- Các bộ xử lý có thể thực hiện chức năng giống nhau
- Hệ thống được điều khiển bởi một hệ điều hành phân tán
- Hiệu năng: Các công việc có thể thực hiện song song
- Khả năng chịu lỗi

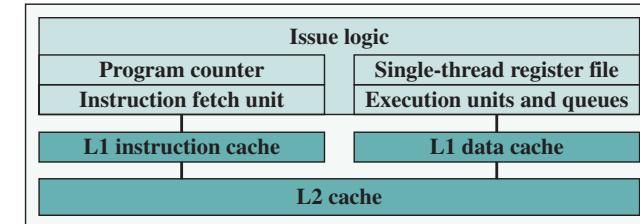
# NUMA (Non-Uniform Memory Access)



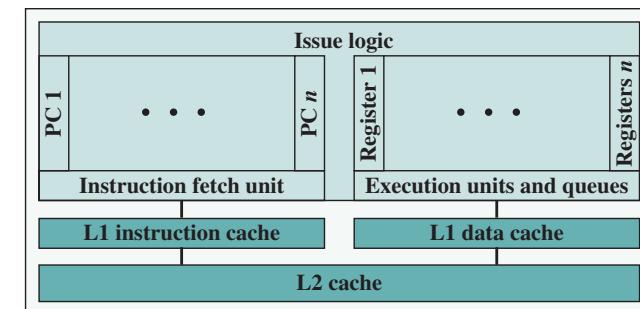
- Có một không gian địa chỉ chung cho tất cả CPU
- Mỗi CPU có thể truy cập từ xa sang bộ nhớ của CPU khác
- Truy nhập bộ nhớ từ xa chậm hơn truy nhập bộ nhớ cục bộ

# Bộ xử lý đa lõi (multicores)

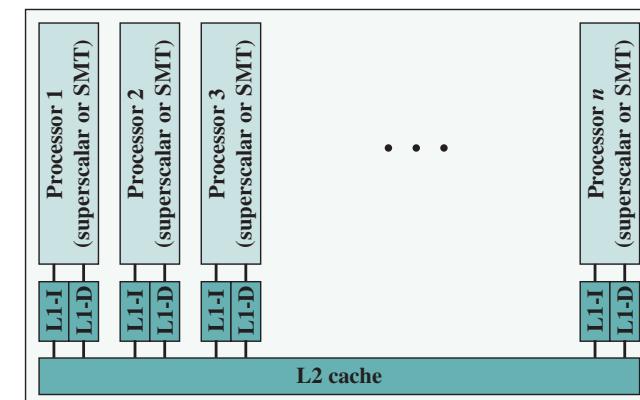
- Thay đổi của bộ xử lý:
  - Tuần tự
  - Pipeline
  - Siêu vô hướng
  - Đa luồng
  - Đa lõi: nhiều CPU trên một chip



(a) Superscalar

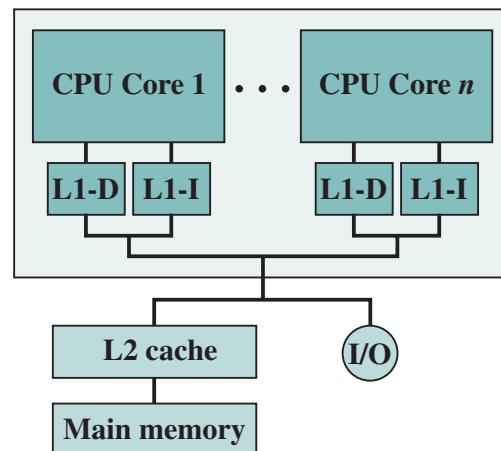


(b) Simultaneous multithreading

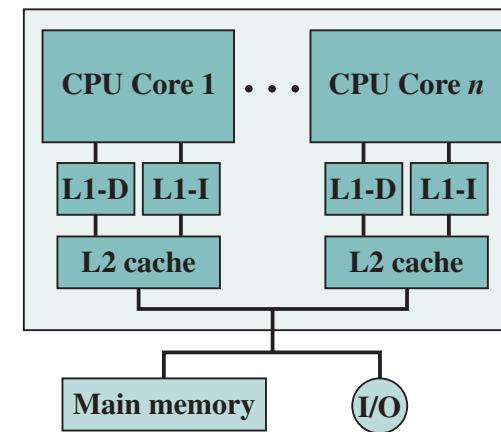


(c) Multicore

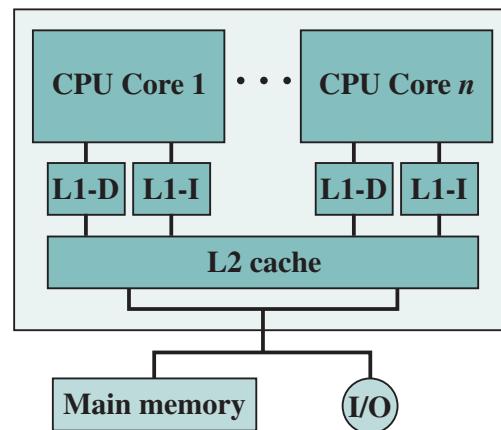
# Các dạng tổ chức bộ xử lý đa lõi



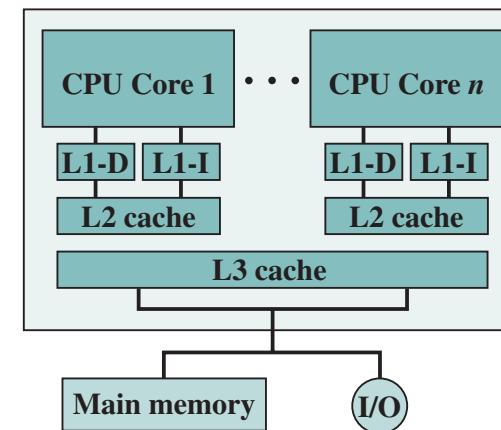
(a) Dedicated L1 cache



(b) Dedicated L2 cache



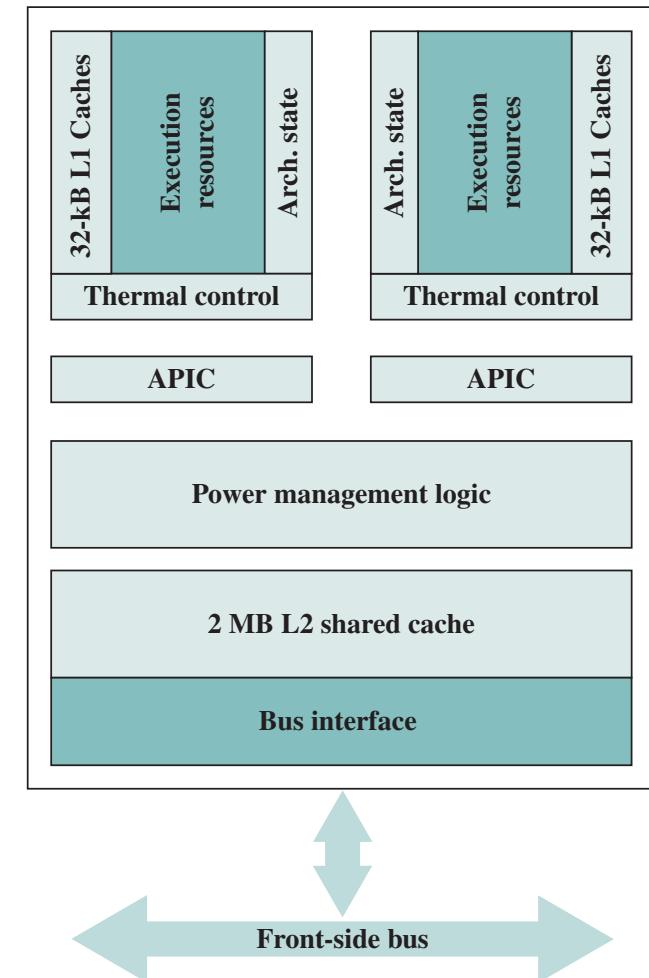
(c) Shared L2 cache



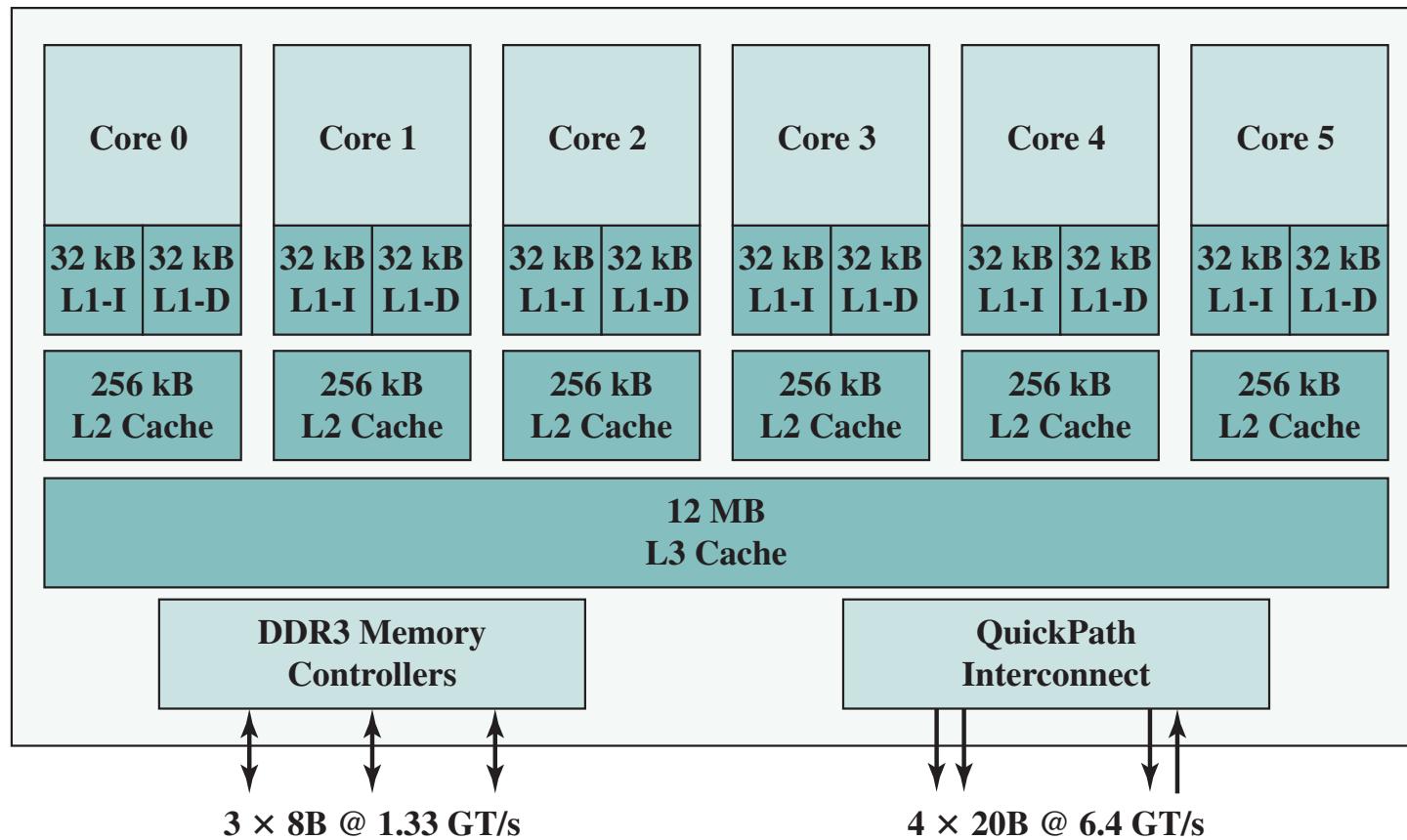
(d) Shared L3 cache

# Intel - Core Duo

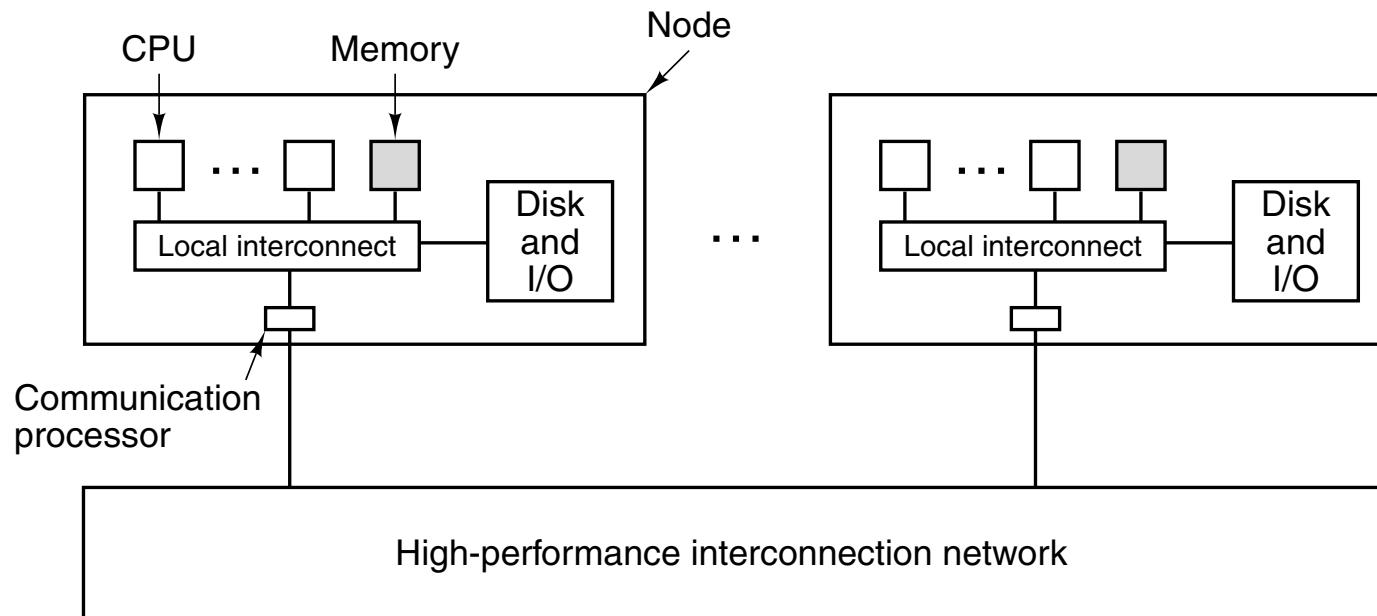
- 2006
- Two x86 superscalar, shared L2 cache
- Dedicated L1 cache per core
  - 32KiB instruction and 32KiB data
- 2MiB shared L2 cache



# Intel Core i7-990X

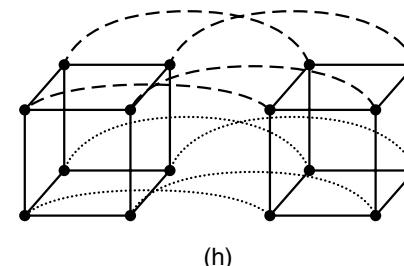
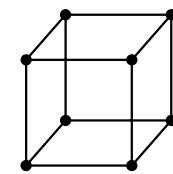
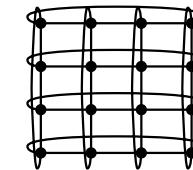
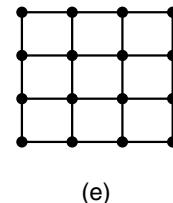
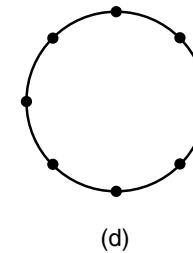
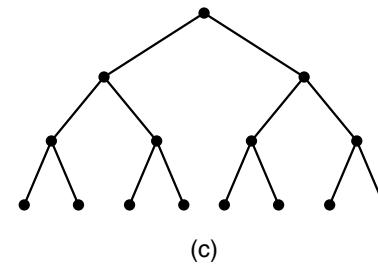
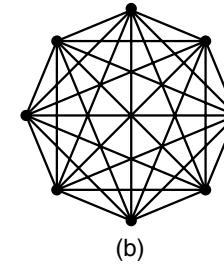
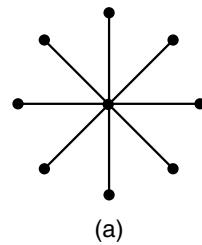


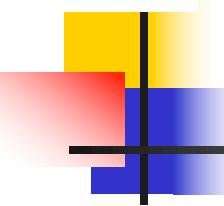
## 4.3. Đa xử lý bộ nhớ phân tán



- Máy tính qui mô lớn (Warehouse Scale Computers or Massively Parallel Processors – MPP)
- Máy tính cụm (clusters)

# Mạng liên kết

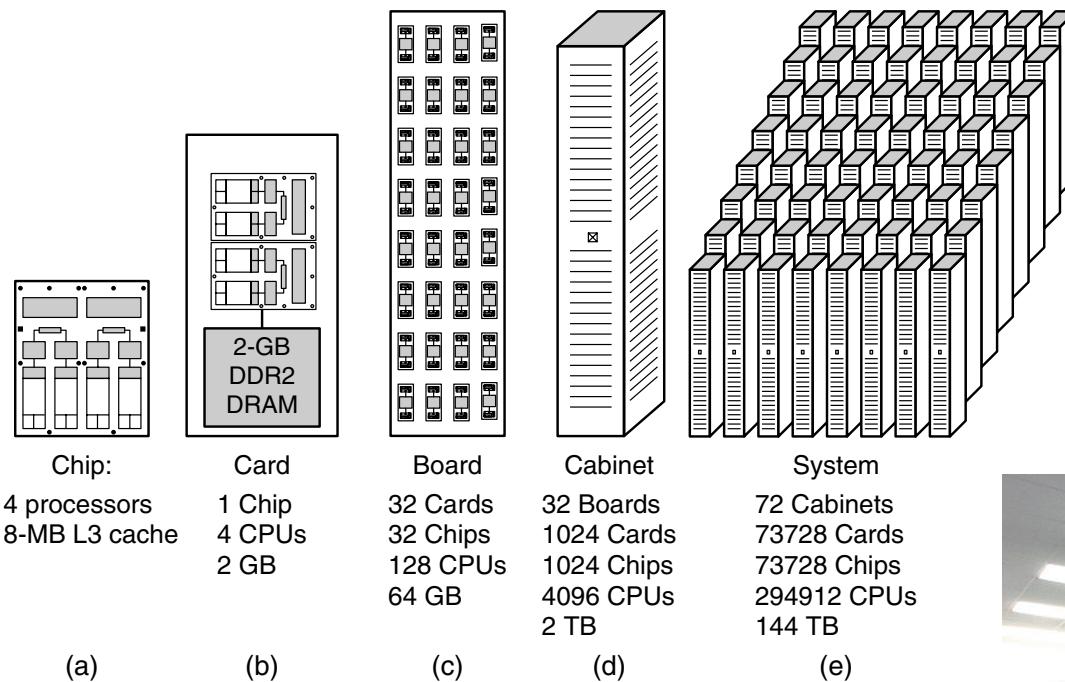


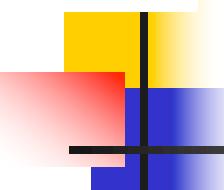


# Massively Parallel Processors

- Hệ thống quy mô lớn
- Đắt tiền: nhiều triệu USD
- Dùng cho tính toán khoa học và các bài toán có số phép toán và dữ liệu rất lớn
- Siêu máy tính

# IBM Blue Gene/P

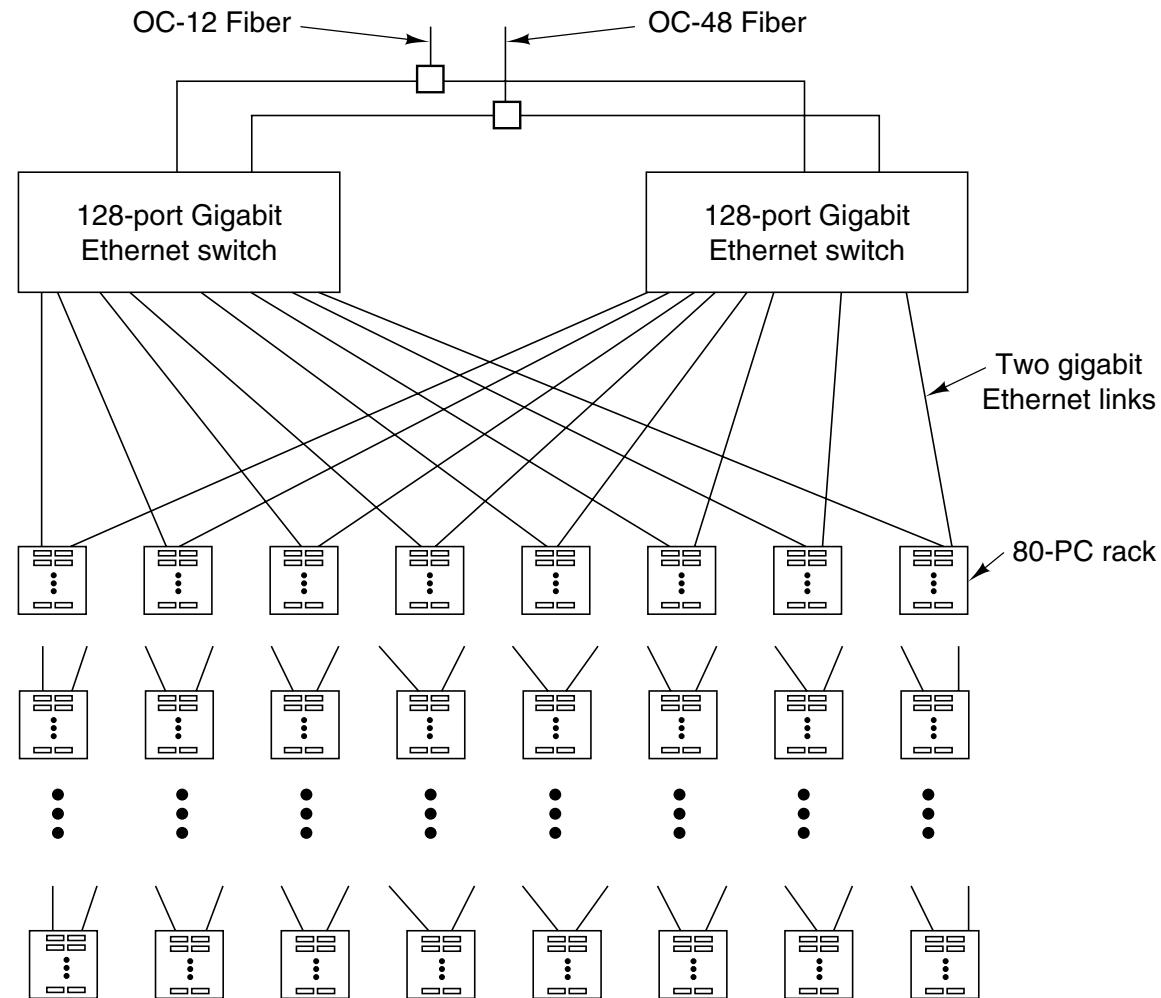




# Cluster

- Nhiều máy tính được kết nối với nhau bằng mạng liên kết tốc độ cao (~ Gbps)
- Mỗi máy tính có thể làm việc độc lập (PC hoặc SMP)
- Mỗi máy tính được gọi là một node
- Các máy tính có thể được quản lý làm việc song song theo nhóm (cluster)
- Toàn bộ hệ thống có thể coi như là một máy tính song song
- Tính sẵn sàng cao
- Khả năng chịu lỗi lớn

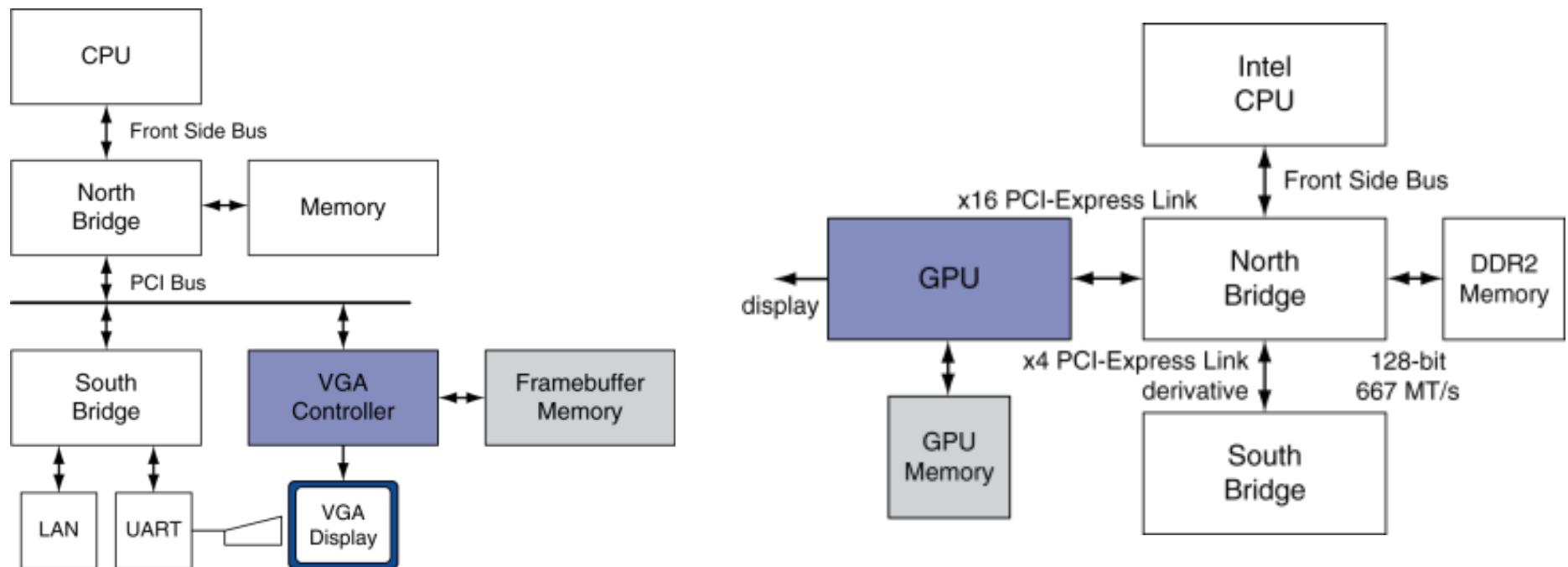
# PC Cluster của Google



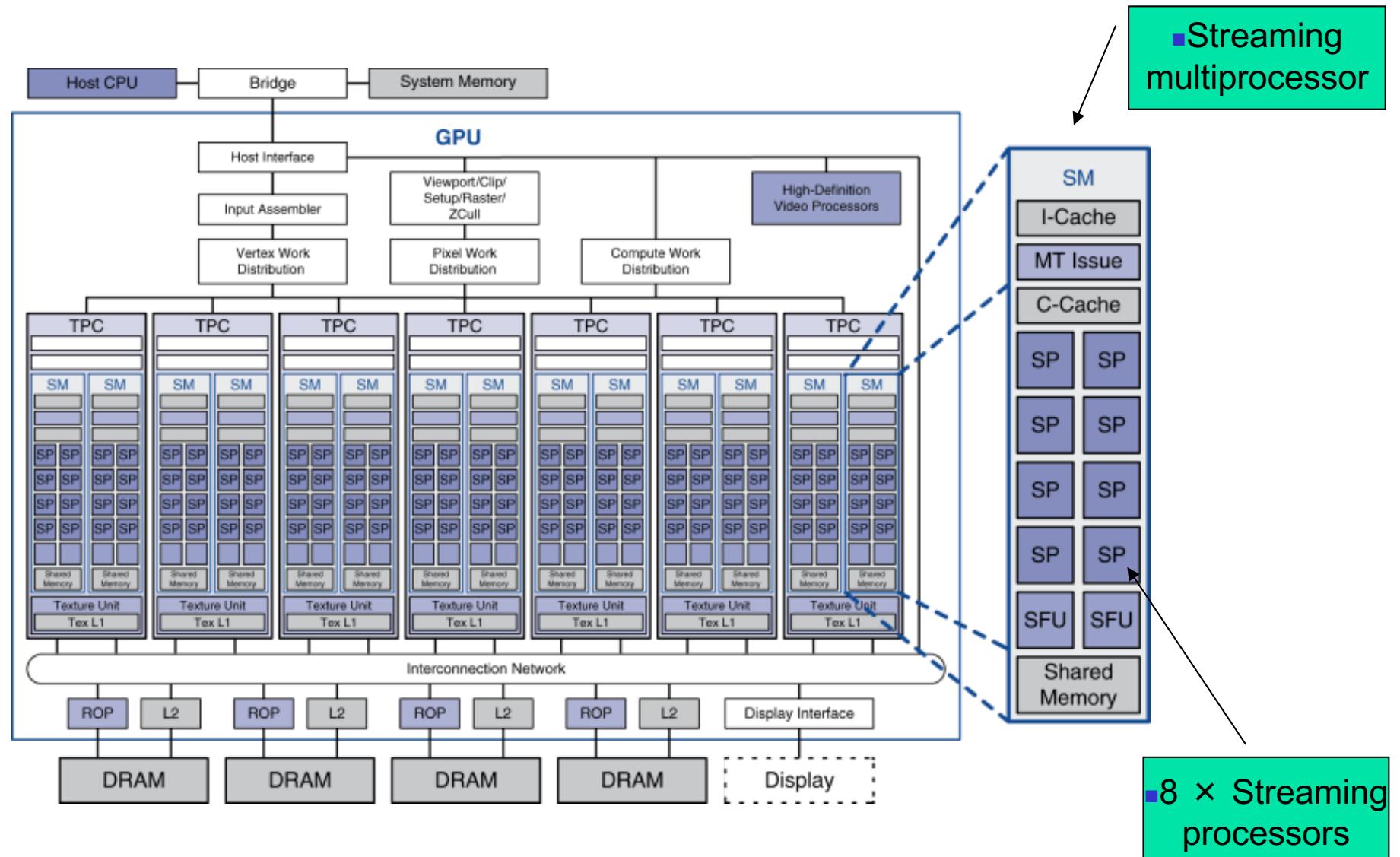
## 4.4. Bộ xử lý đồ họa đa dụng

- Kiến trúc SIMD
- Xuất phát từ bộ xử lý đồ họa GPU (Graphic Processing Unit) hỗ trợ xử lý đồ họa 2D và 3D: xử lý dữ liệu song song
- GPGPU – General purpose Graphic Processing Unit
- Hệ thống lai CPU/GPGPU
  - CPU là host: thực hiện theo tuần tự
  - GPGPU: tính toán song song

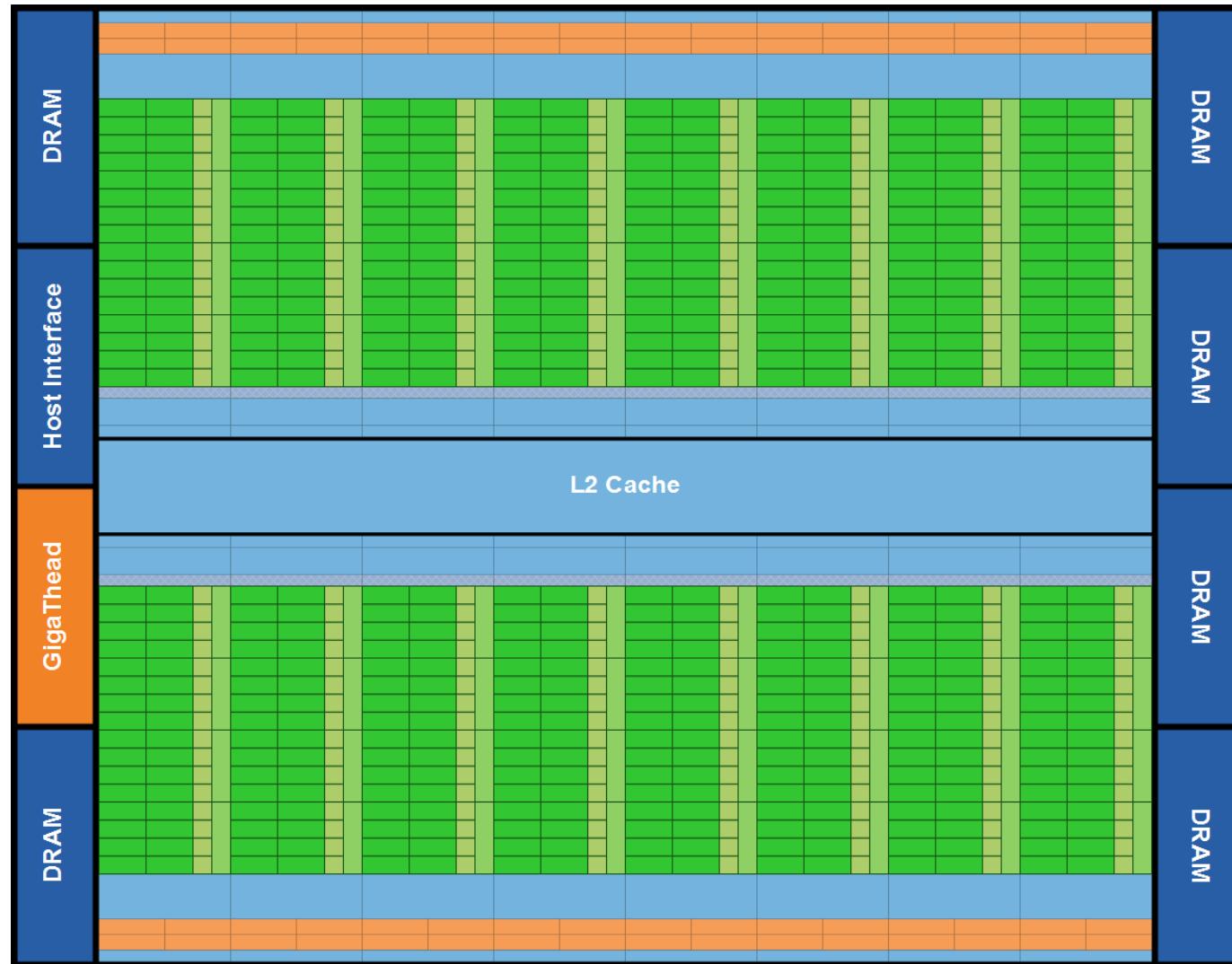
# Bộ xử lý đồ họa trong máy tính



# GPGPU: NVIDIA Tesla

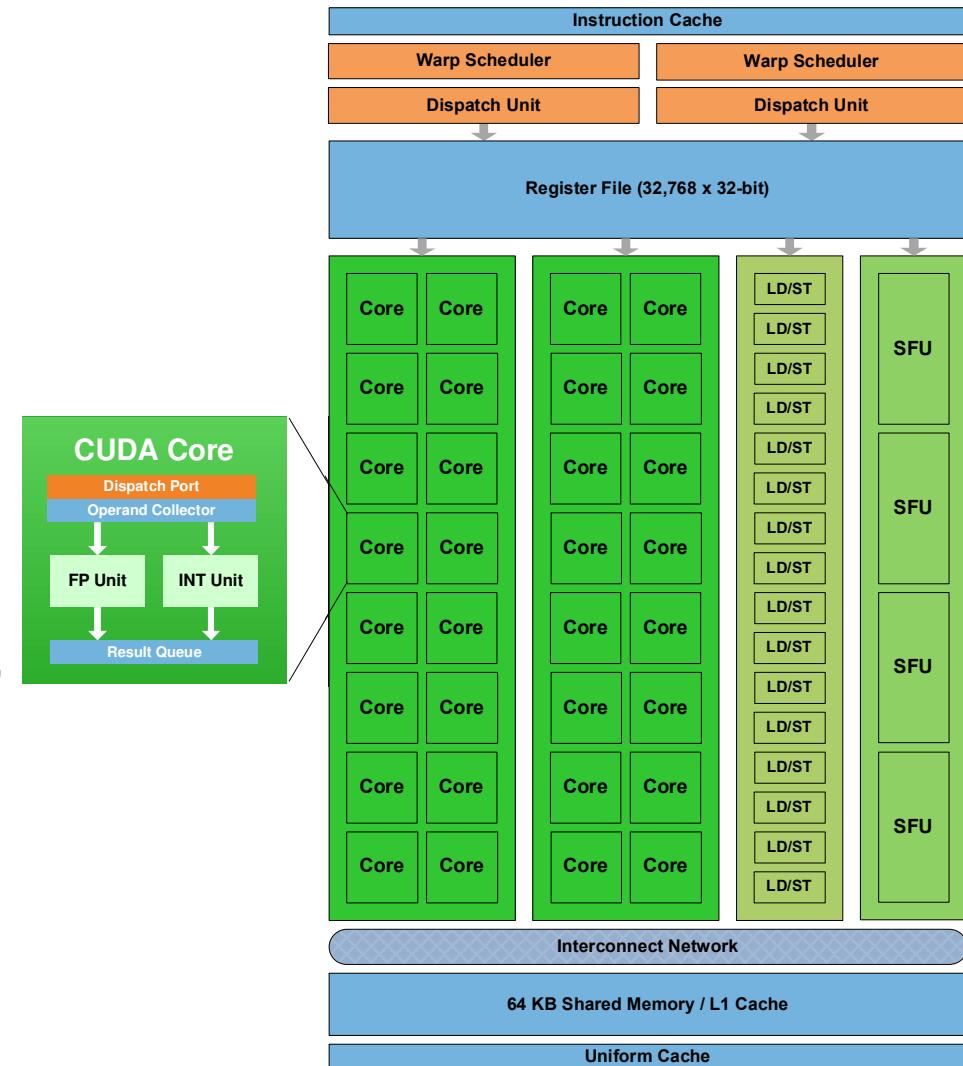


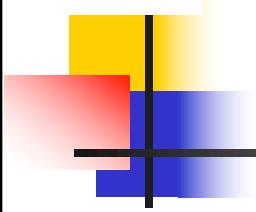
# GPGPU: NVIDIA Fermi



# NVIDIA Fermi

- Có 16 Streaming Multiprocessors (SM)
- Mỗi SM có 32 CUDA cores.
- Mỗi CUDA core (Compute Unified Device Architecture) có 01 FPU và 01 IU





Hết