

1. TỔNG QUAN MÁY TÍNH

NỘI DUNG BÀI GIẢNG

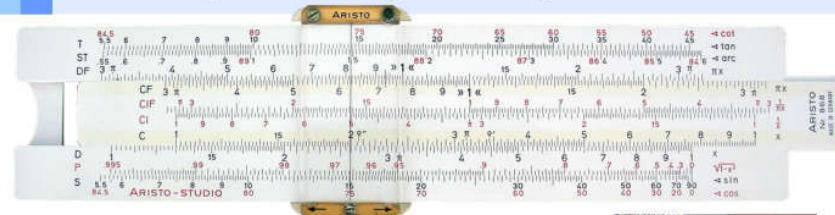
- ❖ Quá trình phát triển và một số nét đặc trưng của các thế hệ máy tính
- ❖ Định luật Moore
- ❖ Một số thành phần cơ bản của máy tính cá nhân ngày nay
- ❖ Giải thích các khái niệm wafer, chip, chipset
- ❖ Mô hình abstraction layers

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

2

Quá trình phát triển của máy tính

Thế hệ 0 - Non-digital computers



Polish analog
computer
[AKAT-1](#)

1959



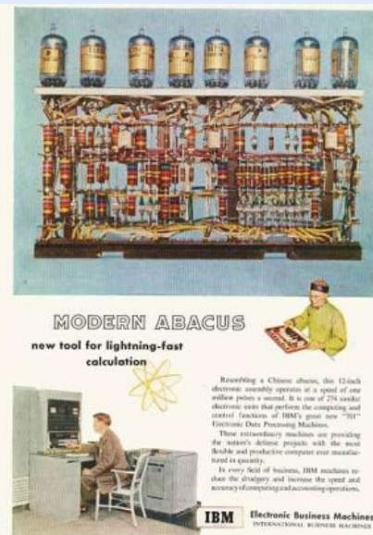
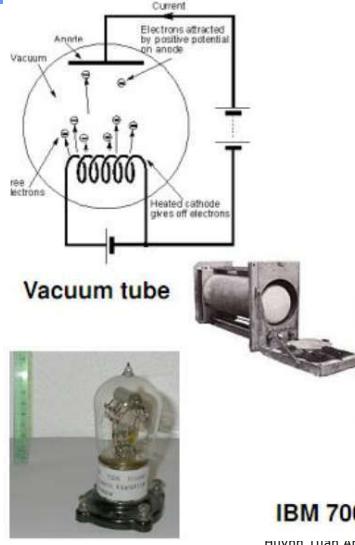
https://en.wikipedia.org/wiki/Analog_computer

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

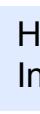
4



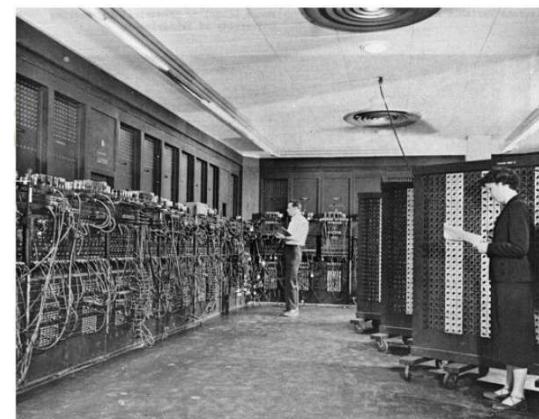
Thế hệ 1 – Vacuum tube (1940 – 1956)



5



Hệ thống ENIAC (Electronic Numerical Integrator and Computer)



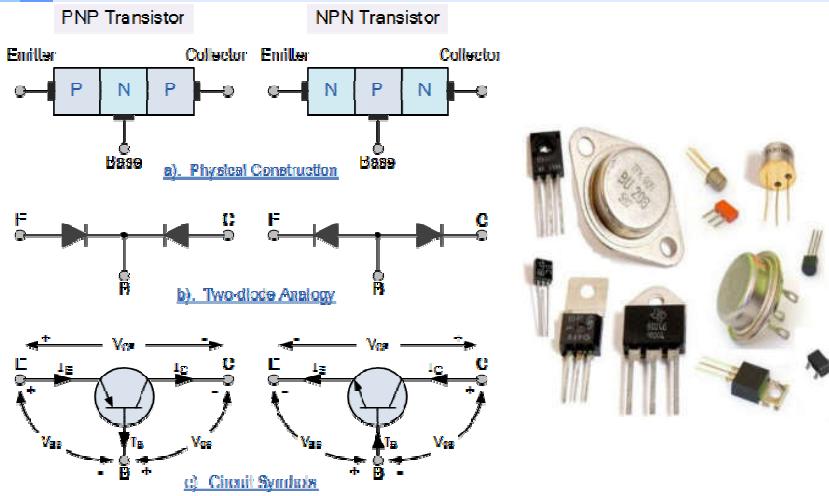
Detail of the back of a panel of ENIAC, showing vacuum tubes

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

6



Thế hệ 2 – Transistor (1956 – 1963)



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

7



Thế hệ 2 - Transistor

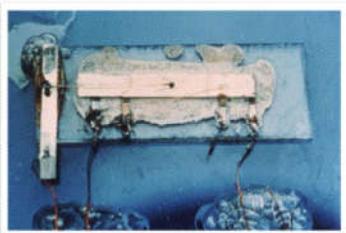


IBM 7094 System

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

8

Thế hệ 3 – IC (Integrated Circuit) (1963 – 1971)



The First Integrated Circuit

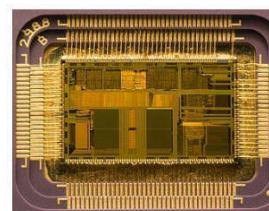


IBM 360 System (1964)

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

9

Thế hệ thứ 4 – Microprocessor (1971 – nay)



Bộ vi xử lý Intel 80486DX2



Intel 4004 with 2300 transistors inside



XT computer with Intel 8086 chip

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

10

Ngày nay – CPU đa nhân, siêu máy tính



[IBM Blue Gene/P supercomputer](#)
(12/10/2007)
164,000 processor cores using
normal data center air conditioning,
grouped in 40 racks/cabinets
connected by a high-speed 3-D
torus network

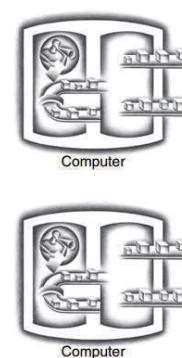


2016-2017

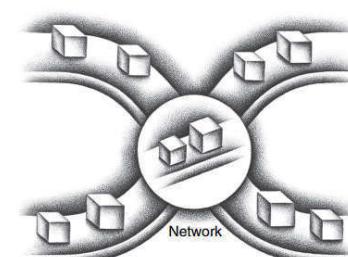
Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

11

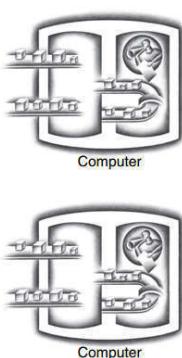
Thế hệ 5 – Parallel processing?



Computer



Network



Computer

Multiprocessor or Cluster Organization

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

12

Tổng kết

Thế hệ	Khoảng thời gian	Công nghệ
1	1940 – 1956	Vacuum tubes
2	1956 – 1963	Transistors
3	1964 – 1971	Integrated Circuits
4	1971 – nay	Microprocessors
5	Tương lai	Parallel Processing

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

13

ĐỊNH LUẬT MORE

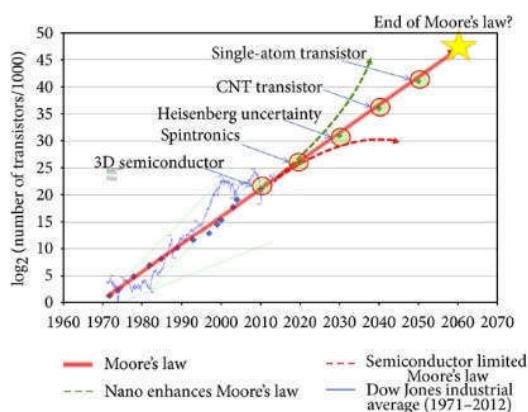
Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

14

Định luật More

Gordon Moore - một trong những sáng lập viên của tập đoàn sản xuất chip máy tính nổi tiếng Intel (1965)

"Số lượng transistor trên mỗi đơn vị inch vuông sẽ tăng lên gấp đôi sau mỗi năm." Năm 2000 định luật được sửa đổi và công nhận là sau mỗi chu kỳ 18-24 tháng



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

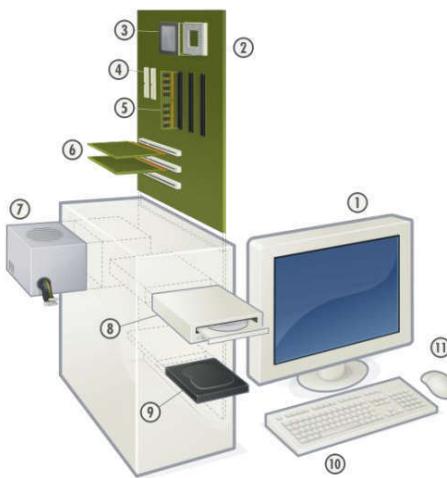
15

MỘT SỐ THÀNH PHẦN CƠ BẢN CỦA MÁY TÍNH

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

16

Một số TP cơ bản

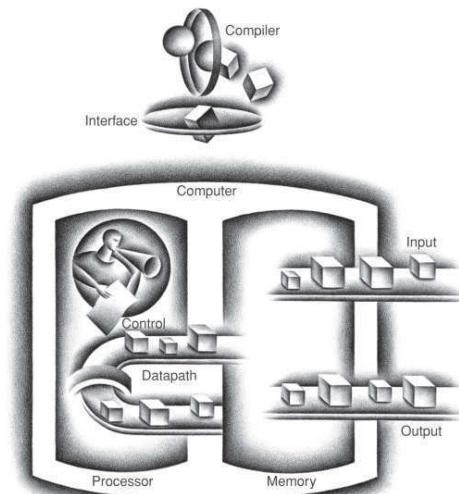


- 1: Màn hình
- 2: Mainboard
- 3: CPU
- 4: Chân cắm dây nối HDD
- 5: RAM
- 6: Chân cắm mở rộng PCI / PCI Express
- 7: Nguồn điện
- 8: Ó quang CD / DVD
- 9: Ó đĩa cứng
- 10: Bàn phím
- 11: Chuột

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

17

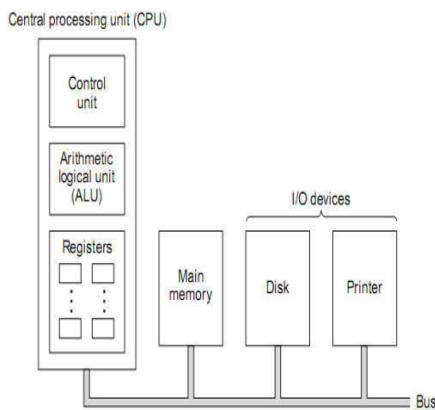
5 thành phần cơ bản



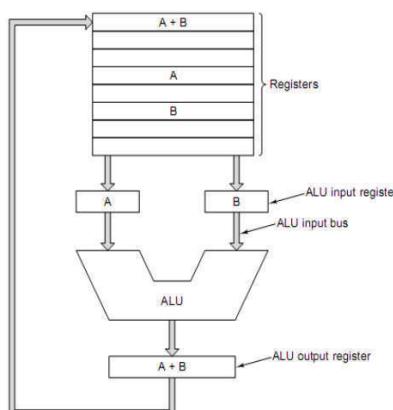
Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

18

Control – data path



The organization of a simple computer with one CPU and two I/O devices.

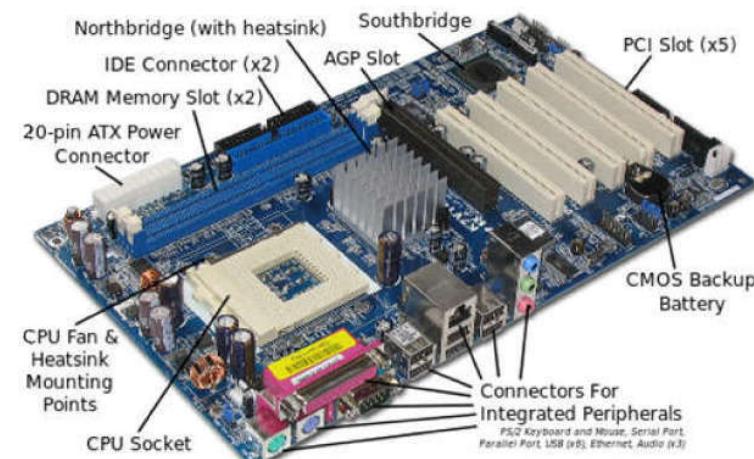


The data path of a typical Von Neumann machine

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

19

Mainboard

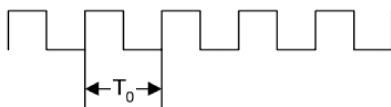


Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

20

Hiệu năng

- ❖ Execution time (thời gian thực hiện): t
- ❖ Hiệu năng $P = 1/t$
- ❖ Xung nhịp của CPU:
 - Hoạt động của CPU được điều khiển bởi xung nhịp có tần số xác định
 - Chu kỳ xung nhịp (T_0): Thời gian diễn ra một chu kỳ
 - Tần số xung nhịp (f_0 – clock rate): Số chu kỳ trong một giây



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

21

Hiệu năng

- **Số chu kỳ = Số lệnh x Số chu kỳ trên một lệnh**
(Thực hiện chương trình) $n = IC \times CPI$
n - số chu kỳ, IC - số lệnh (Instruction Count), CPI - số chu kỳ trên một lệnh (Cycles per Instruction)
 - **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

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

22

Ví dụ

- Máy tính A: $T_A = 250\text{ps}$, $CPI_A = 2.0$
- Máy tính B: $T_B = 500\text{ps}$, $CPI_B = 1.2$
- Cùng kiến trúc tập lệnh (ISA)
- Máy nào nhanh hơn và nhanh hơn bao nhiêu ?

$$\begin{aligned} t_A &= IC \times CPI_A \times T_A \\ &= IC \times 2.0 \times 250\text{ps} = IC \times 500\text{ps} \end{aligned}$$

$$\begin{aligned} t_B &= IC \times CPI_B \times T_B \\ &= IC \times 1.2 \times 500\text{ps} = IC \times 600\text{ps} \end{aligned}$$

$$\frac{t_B}{t_A} = \frac{IC \times 600\text{ps}}{IC \times 500\text{ps}} = 1.2$$

Vậy:
A nhanh hơn B 1.2 lần

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

23

Ví dụ 2

- 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	2	1	2
IC trong dãy lệnh 2	4	1	1

- Dãy lệnh 1: $IC = 5$
 - Số chu kỳ
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
 - $CPI_{TB} = 10/5 = 2.0$
- Dãy lệnh 2: $IC = 6$
 - Số chu kỳ
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$
 $= 9$
 - $CPI_{TB} = 9/6 = 1.5$

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

24

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 CPI của bộ xử lý với clock rate = 1Ghz và 400MIPS

Giải 1:

- 400 MPIS \rightarrow 1s thực hiện được $400 \times 10^6 = 4 \times 10^8$ lệnh
- Clock rate = 1GHz \rightarrow 1s có 10^9 chu kỳ
 - $\rightarrow 4 \times 10^8$ lệnh được thực hiện trong 10^9 chu kỳ
 - \rightarrow CPI = $10^9 / (4 \times 10^8) = 2.5$

Giải 2:

- Theo công thức CPI = $f_0 / (\text{MIPS} \times 10^6)$
 $= 10^9 / (400 \times 10^6) = 2.5$

Ví dụ 2

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

Giải 1:

- Clock rate = 2GHz \rightarrow 1s có 2×10^9 chu kỳ
- CPI = 4 \rightarrow 1 lệnh thực hiện trong 4 chu kỳ
 - \rightarrow Số triệu lệnh thực hiện trong 1s

$$2 \times 10^9 / (4 \times 10^6) = 500 \text{ MIPS}$$

Giải 2:

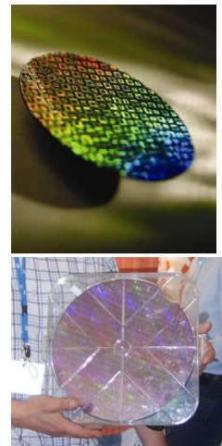
- Theo công thức MIPS = $f_0 / (\text{CPI} \times 10^6)$
 $= 2 \times 10^9 / (4 \times 10^6) = 500$

Một số khái niệm cơ bản



Wafer

- Wafer (Đế chip): Tấm silicon mỏng đã được cấy vật liệu khác nhau để tạo ra những vi mạch
- Có kích thước trung bình từ 25,4mm (1 inch) – 200mm (7.9 inch).
- Intel, TSMC hay Samsung đã nâng kích thước của wafer lên 300mm (12 inch), thậm chí lên 450mm (18 inch)
- Kích thước wafer được tăng lên đã làm giá thành của một vi mạch trở nên rất rẻ.



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

29



Chip

- ❖ Chip: Có thể hiểu là mạch tích hợp (Integrated Circuit) gắn trên đế chip (wafer) nhằm xử lý các công việc trên máy tính
- ❖ Chip có kích thước rất nhỏ nhưng có thể chứa hàng chục triệu transistor, số lượng transistor càng lớn thì tốc độ truyền và xử lý tín hiệu càng nhanh
- ❖ Hiện nay có các loại chip xử lý: 4, 8, 16, 32, 64 bit

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

30

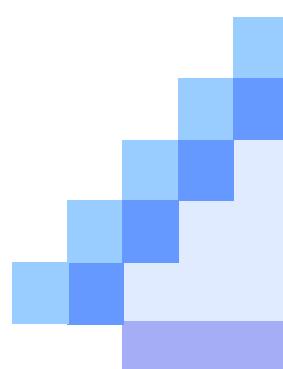


Chipset

- ❖ Chipset là tập hợp nhiều chip gắn kết lại với nhau trên cùng 1 đế chip (wafer) để xử lý nhiều công việc trên máy tính
- ❖ Một số chipset thông dụng:
 - CPU: Đơn vị xử lý trung tâm
 - GPU: Đơn vị xử lý đồ họa trên máy
 - RAM: Bộ nhớ truy cập tức thời chuyên phục vụ cho CPU
 - **Bán cầu bắc** (tích hợp trên mainboard): Hỗ trợ truyền thông tin cho CPU, RAM, nằm sát CPU (Hệ thống Mainboard AMD không có chipset này vì được tích hợp ngay trên CPU)
 - **Bán cầu nam** (tích hợp trên mainboard): Quản lý thiết bị ngoại vi như HDD, Mouse, Keyboard...Nằm cuối mainboard

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

31

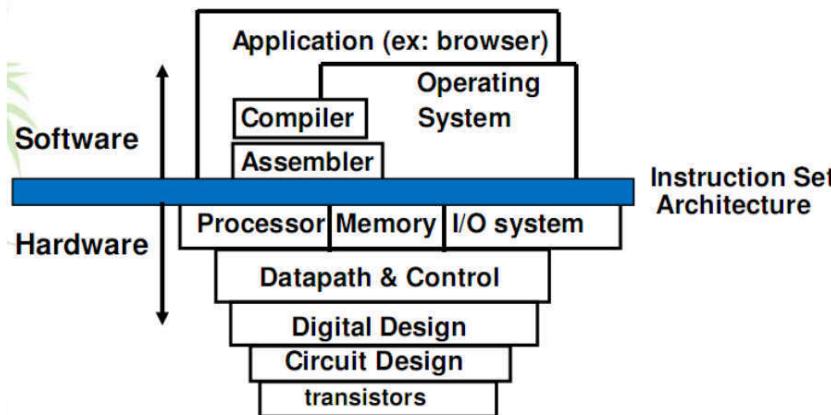


Abstraction layer

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

32

Abstraction layers



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

33

Bài tập

1. Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ?
2. Đặc trưng cơ bản của các máy tính thế hệ thứ nhất?
3. Đặc trưng cơ bản của các máy tính thế hệ thứ hai?
4. Đặc trưng cơ bản của các máy tính thế hệ thứ ba?
5. Đặc trưng cơ bản của các máy tính thế hệ thứ tư?
6. Khái niệm chip, chipset, vi xử lý 8bit là gì?

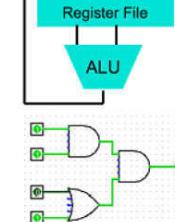
Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

35

temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw \$t0, 0(\$2)
lw \$t1, 4(\$2)
sw \$t1, 0(\$2)
sw \$t0, 4(\$2)

0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111



Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

34

Bài tập

7. Bốn chương trình đánh giá hiệu năng được thực hiện trên 3 máy tính với kết quả như sau (thời gian thực hiện bằng giây với 100 000 000 lệnh được thực hiện ở mỗi chương trình)

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Program 3	500	1000	50
Program 4	100	800	100

Tính MIPS đối với mỗi computer cho mỗi chương trình?

Huỳnh Tuấn Anh - Khoa CNTT ĐHNT

36



Bài tập

8. Xét 3 vi xử lý khác nhau P1, P2, P3 có cùng một kiến trúc tập lệnh. Clock rate và CPI của P1, P2, P3 lần lượt là: (3GHz, 1.5); (2.5GHz, 1.0), (4GHz, 2.2)

- Vi xử lý nào xử lý chương trình nhanh hơn và nhanh hơn các vi xử lý còn lại bao nhiêu lần.
- Vi xử lý nào có hiệu năng cao nhất tính theo MIPS
- Nếu mỗi vi xử lý thực hiện một chương trình nào đó trong 10 giây. Tính số lệnh và số chu kỳ trong mỗi trường hợp



2. SỐ HỌC MÁY TÍNH

Huỳnh Tuấn Anh
Khoa CNTT – ĐHNT
Phone: 094 5505 449
Email: anhht@ntu.edu.vn

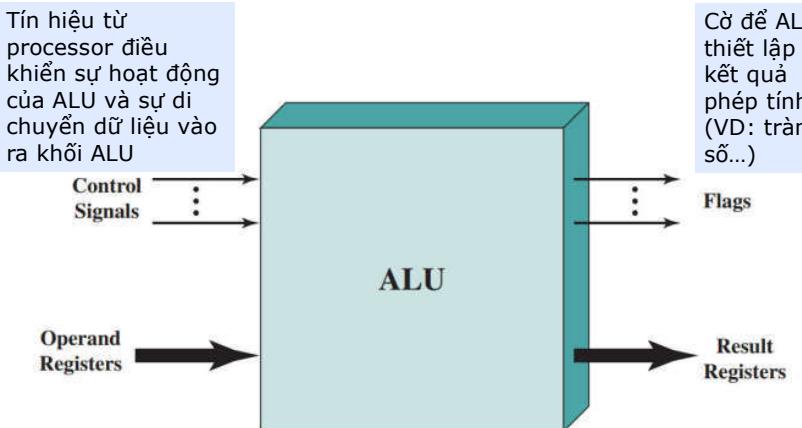
Nội dung

- ❖ Các hệ đếm
- ❖ Biểu diễn số nguyên không dấu
- ❖ Biểu diễn số nguyên không dấu
- ❖ Phép cộng và phép trừ số nguyên
- ❖ Phép nhân và phép chia số nguyên
- ❖ Số phẩy dấu động

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

2

ALU – Khối xử lý số học và logic



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

3

CÁC HỆ ĐẾM



Các hệ đếm cơ bản

- ❖ Hệ thập phân (Decimal System) → con người sử dụng
 - Dùng 10 ký số: 0, 1, 2, ..., 9 để biểu diễn
- ❖ Hệ nhị phân (Binary System) → máy tính sử dụng
 - Dùng 2 ký số: 0, 1 để biểu diễn
- ❖ Hệ mươi sáu (Hexadecimal System) → dùng để viết gọn cho số nhị phân
 - Dùng 16 ký số 0, 1, 2,..., 9, A, B, C, D, E, F để biểu diễn

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

5



Hệ thập phân

- ❖ Cơ số 10
- ❖ Dùng 10 ký số: 0, 1, 2,9
- ❖ Dùng n chữ số thập phân có thể biểu diễn được 10^n giá trị khác nhau
 - $00\dots000 = 0$ $99\dots999 = 10^n-1$
- ❖ Dạng tổng quát của số thập phân:

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m} h$$
 - Giá trị của A:
$$a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + \dots + a_{-m} 10^{-m}$$

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

6



Ví dụ số thập phân

- ❖ $472.38 = 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 8 \times 10^{-2}$
- ❖ Các chữ số phần nguyên
 - $472 : 10 = 47$ dư 2 ↑
 - $47 : 10 = 4$ dư 7
 - $4 : 10 = 0$ dư 4
- ❖ Các chữ số của phần lẻ:
 - $0.38 \times 10 = 3.8$ Phân nguyên 3 ↓
 - $0.8 \times 10 = 8.0$ Phân nguyên 8

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

7



Hệ nhị phân

- ❖ Cơ số 2
- ❖ Dùng 2 chữ số nhị phân: 0, 1 để biểu diễn.
- ❖ Ký số nhị phân gọi là **bit** (**binary digit**)
- ❖ bit là đơn vị đo thông tin nhỏ nhất
- ❖ Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau
 - $00\dots000_2 = 0$
 - $11\dots111_2 = 2^n-1$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

8



Bit, byte, Nibbles

- Bits

10010110
most significant bit least significant bit

- Bytes & Nibbles

byte
10010110
nibble

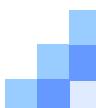
- Bytes

CEBF9AD7
most significant byte least significant byte



Lũy thừa 2

- $2^{10} = 1 \text{ kilo}$ $\approx 1000 \text{ (1024)}$
- $2^{20} = 1 \text{ mega}$ $\approx 1 \text{ triệu (1,048,576)}$
- $2^{30} = 1 \text{ giga}$ $\approx 1 \text{ tỷ (1,073,741,824)}$
- $2^{40} = 1 \text{ tera}$ $\approx 1000 \text{ tỷ}$
- $2^{50} = 1 \text{ peta}$ $\approx 1 \text{ triệu tỷ}$



Dạng tổng quát của một số nhị phân

- ❖ Số nhị phân A:

- $A = a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m}$ h
- Giá trị của a được tính như sau

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

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

- ❖ Ví dụ:

- $1101001,1011_{(2)} = ?_{10}$



Chuyển đổi số thập phân sang nhị phân

- ❖ Phần nguyên

- Phương pháp 1: Chia liên tiếp cho 2 và lấy phần dư
- Phân tích thành tổng các lũy thừa 2^i

- ❖ Phần thập phân

- Lấy phần thập phân nhân liên tiếp cho 2

Phương pháp chia liên tiếp cho 2

❖ Lấy số cơ số 10 chia cho 2

- Số dư đưa vào kết quả
- Số nguyên đem chia tiếp cho 2
- Quá trình lặp lại cho đến khi số nguyên = 0

❖ Ví dụ A=123

- $123 : 2 = 61$ dư 1
- $61 : 2 = 30$ dư 1
- $30 : 2 = 15$ dư 0
- $15 : 2 = 7$ dư 1
- $7 : 2 = 3$ dư 1
- $3 : 2 = 1$ dư 1
- $1 : 2 = 0$ dư 1

Kết quả: $123_{10} = 1111011_2$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

13

Phân tích thành tổng các 2^i

❖ Ví dụ 1: chuyển đổi $105_{(10)}$

$$105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

$$\text{Kết quả: } 105_{(10)} = 0110\ 1001_{(2)}$$

❖ Ví dụ 2: $17000_{(10)} = 16384 + 512 + 64 + 32 + 8$
 $= 2^{14} + 2^9 + 2^6 + 2^5 + 2^3$

$$17000_{(10)} = 0100\ 0010\ 0110\ 1000_{(2)}$$

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

14

Chuyển đổi phần thập phân sang hệ 2

❖ Ví dụ 1: $0.6875_{(10)} = 0,1011_2$

- $0.6875 \times 2 = 1,375$ phần nguyên 1
- $0.375 \times 2 = 0.75$ phần nguyên 0
- $0.75 \times 2 = 1.5$ phần nguyên 1
- $0.5 \times 5 = 1.0$ phần nguyên 1

❖ Ví dụ 2: $0.81_{10} \cong 11001_2$

- $0.81 \times 2 = 1.62$ phần nguyên 1
- $0.62 \times 2 = 1.24$ phần nguyên 1
- $0.24 \times 2 = 0.48$ phần nguyên 0
- $0.48 \times 2 = 0.96$ phần nguyên 0
- $0.96 \times 2 = 1.92$ phần nguyên 1
-

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

15

Hệ mười sáu

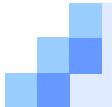
❖ Cơ số 16

❖ 16 chữ số: 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F

❖ Dùng để viết gọn cho số nhị phân: cứ một nhóm 4-bit sẽ được thay bằng một chữ số Hexa

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

16



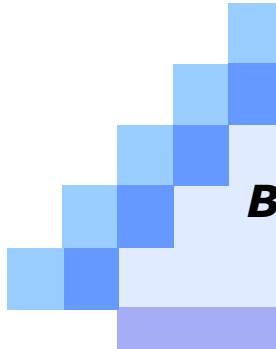
Quan hệ giữa số nhị phân và số Hexa

4-bit	Chữ số Hexa
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

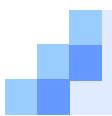
Ví dụ chuyển đổi số nhị phân → số Hexa:

- $1011\ 0011_2 = B3_{16}$
- $0000\ 0000_2 = 00_{16}$
- $0010\ 1101\ 1001\ 1010_2 = 2D9A_{16}$
- $1111\ 1111\ 1111\ 1111_2 = FFFF_{16}$

17



Biểu diễn số nguyên không dấu



Biểu diễn Số nguyên không dấu (Unsigned Integer)

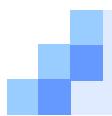
❖ Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên không dấu A:

$$a_{n-1}a_{n-2}\dots a_1a_0$$

❖ Giá trị của A được tính như sau

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

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



Ví dụ

❖ Biểu diễn các số nguyên không dấu sau đây bằng số nhị phân 8 bit

- $A=41; B=150$

- ❖ $A = 41 = 2^5 + 2^3 + 2^0 = 0010\ 1001_2$

- ❖ $B = 128 + 16 + 4 + 2 = 2^7 + 2^4 + 2^2 + 2^1 = 1001\ 0110_2$

N=8 bit

- ❖ Biểu diễn được các giá trị từ 0 đến 255

0000 0000 = 0

0000 0001 = 1

0000 0010 = 2

0000 0011 = 3

.....

1111 1111 = 255

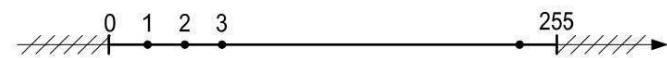
Chú ý:

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

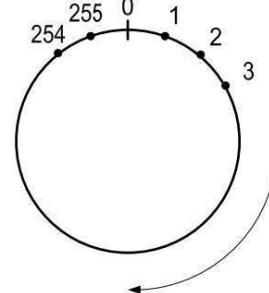
Vậy: $255 + 1 = 0$?
→ do tràn nhớ ra
ngoài

Trục số học với n=8 bit

Trục số học:



Trục số học máy tính:



n=16; n=32; n=64

- ❖ n= 16 bit: dài biểu diễn từ 0 đến $65535 (2^{16} - 1)$

- 0000 0000 0000 0000 = 0
-
- 0000 0000 1111 1111 = 255
- 0000 0001 0000 0000 = 256
-
- 1111 1111 1111 1111 = 65535

- ❖ n= 32 bit: dài biểu diễn từ 0 đến $2^{32} - 1$

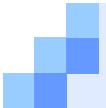
- ❖ n= 64 bit: dài biểu diễn từ 0 đến $2^{64} - 1$

Biểu diễn số nguyên có dấu



Số bù chín và Số bù mươi

- ❖ Cho một số thập phân A được biểu diễn bằng n chữ số thập phân, ta có:
 - Số bù 9 của A = $(10^n - 1) - A$
 - Số bù 10 của A = $10^n - A$
- ❖ Số bù mươi của A = (Số bù chín của A) + 1



Ví dụ

- ❖ Ví dụ: với n=4, cho A = 3265

- Số bù chín của A:

$$\begin{array}{r} 9999 \\ - 3265 \\ \hline 6734 \end{array} \quad (10^4 - 1)$$

- Số bù mươi của A:

$$\begin{array}{r} 10000 \\ - 3265 \\ \hline 6735 \end{array} \quad (10^4)$$



Số bù 1 và số bù 2

- ❖ Cho một số nhị phân A được biểu diễn bằng n bit, ta có:
 - Số bù 1 của A = $(2^n - 1) - A$
 - Số bù 2 của A = $2^n - A$
- ❖ Số bù hai của A = (Số bù một của A) + 1



Ví dụ

- ❖ Ví dụ: với n = 8 bit, cho A = 0010 0101

- Số bù một của A được tính như sau:

$$\begin{array}{r} 1111\ 1111 \\ - 0010\ 0101 \\ \hline 1101\ 1010 \end{array} \quad (2^8 - 1)$$

→ đảo các bit của A

- Số bù hai của A được tính như sau:

$$\begin{array}{r} 1\ 0000\ 0000 \\ - 0010\ 0101 \\ \hline 1101\ 1011 \end{array} \quad (2^8)$$

→ thực hiện khó khăn



Qui tắc tìm số bù 1 và số bù 2

- ❖ Số bù một của A = đảo giá trị các bit của A
- ❖ (Số bù hai của A) = (Số bù một của A) + 1

▪ Ví dụ:

▪ Cho A =	0010 0101
▪ Số bù một =	1101 1010
	+ 1
▪ Số bù hai =	1101 1011

▪ Nhận xét:

A =	0010 0101
Số bù hai =	+ 1101 1011
	1 0000 0000 = 0

(bỏ qua bit nhớ ra ngoài)

→ Số bù hai của A = -A



Biểu diễn số nguyên có dấu bằng mã bù 2

- ❖ Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên có dấu A:

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

- ❖ Với A là số dương: bit $a_{n-1}=0$, các bit còn lại biểu diễn độ lớn như số không dấu

- ❖ Với A là số âm: được biểu diễn bằng số bù hai của số dương tương ứng, vì vậy bit $a_{n-1}=1$



Biểu diễn số dương

- ❖ Dạng tổng quát của số dương A:
$$0a_{n-2}\dots a_2a_1a_0$$
- ❖ Giá trị của số dương A: $A = \sum_{i=0}^{n-2} a_i 2^i$
- ❖ Dải biểu diễn cho số dương: 0 đến $2^{n-1}-1$



Biểu diễn số âm

- ❖ Dạng tổng quát của số âm A:
$$1a_{n-2}\dots a_2a_1a_0$$
- ❖ Giá trị của số âm A: $A = -2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$
- ❖ Dải biểu diễn cho số âm: -1 đến -2^{n-1}



Biểu diễn tổng quát cho số nguyên có dấu

- Dạng tổng quát của số nguyên A:

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

- Giá trị của A được xác định như sau:

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

- Dải biểu diễn: từ $-(2n-1)$ đến $+(2n-1-1)$



Các ví dụ

- Biểu diễn các số nguyên có dấu sau đây bằng 8-bit:

$$A = 58 \quad B = -80$$

Giải:

$$A = +58 = 0011\ 1010$$

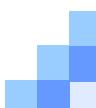
$$B = -80$$

$$\text{Ta có: } +80 = 0101\ 0000$$

$$\begin{array}{r} \text{Số bù một} \\ = 1010\ 1111 \end{array}$$

$$\begin{array}{r} \text{Số bù hai} \\ = \hline 1011\ 0000 \end{array}$$

$$\text{Vậy: } B = -80 = 1011\ 0000$$



Các ví dụ

- Xác định các giá trị nguyên có dấu của các số nhị phân sau đây:

▪ P = 0110 0010 Q = 1101 1011

- Giải:

▪ P = 0110 0010 = $64+32+2 = +98$
 ▪ Q = 1101 1011 = $-128+64+16+8+2+1 = -37$



Với n=8 bit

- Biểu diễn được các giá trị từ -128 đến +127

$$0000\ 0000 = 0$$

$$0000\ 0001 = +1$$

$$0000\ 0010 = +2$$

$$0000\ 0011 = +3$$

Chú ý:

$$+127 + 1 = -128$$

$$-128 - 1 = +127$$

→ do tràn xảy ra

$$\dots$$

$$0111\ 1111 = +127$$

$$1000\ 0000 = -128$$

$$1000\ 0001 = -127$$

$$\dots$$

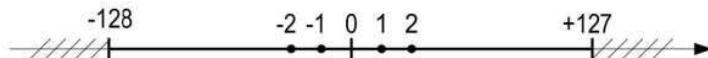
$$1111\ 1110 = -2$$

$$1111\ 1111 = -1$$

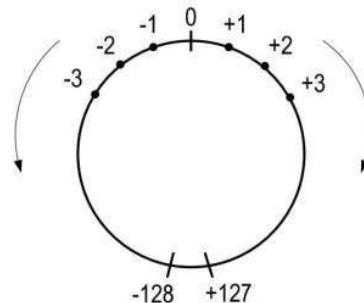


Trục số học với số nguyên có dấu 8 bit

- Trục số học:



- Trục số học máy tính:



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

37



Với n = 16 bit, 32 bit, 64 bit

- Với n=16 bit: biểu diễn từ -32768 đến +32767

- 0000 0000 0000 0000 = 0
- 0000 0000 0000 0001 = +1
- ...
- 0111 1111 1111 1111 = +32767
- 1000 0000 0000 0000 = -32768
- ...
- 1111 1111 1111 1111 = -1

- Với n=32 bit: biểu diễn từ -2^{31} đến $2^{31}-1$

- Với n=64 bit: biểu diễn từ -2^{63} đến $2^{63}-1$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

38



Chuyển đổi từ 8 bit thành 16 bit

- Đối với số dương:

- +19 = 0001 0011
- +19 = 0000 0000 0001 0011

→ Thêm 8 bit 0 vào bên trái

- Đối với số âm

- 19 = 1110 1101
- 19 = 1111 1111 1110 1101

→ Thêm 8 bit 1 vào bên trái

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

39



Phép cộng và phép trừ số nguyên

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

40

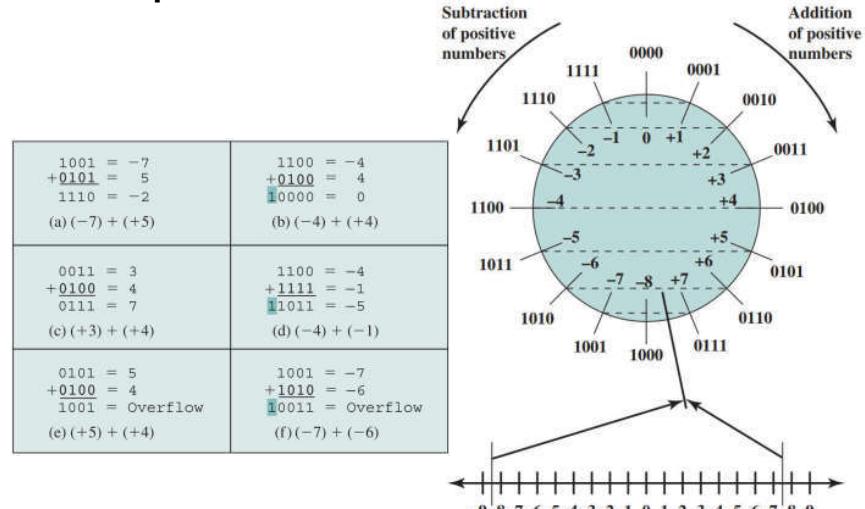
Phép cộng số nguyên

$\begin{array}{r} 1001 = -7 \\ +0101 = 5 \\ 1110 = -2 \\ \hline (a) (-7) + (+5) \end{array}$	$\begin{array}{r} 1100 = -4 \\ +0100 = 4 \\ 10000 = 0 \\ \hline (b) (-4) + (+4) \end{array}$
$\begin{array}{r} 0011 = 3 \\ +0100 = 4 \\ 0111 = 7 \\ \hline (c) (+3) + (+4) \end{array}$	$\begin{array}{r} 1100 = -4 \\ +1111 = -1 \\ 11011 = -5 \\ \hline (d) (-4) + (-1) \end{array}$
$\begin{array}{r} 0101 = 5 \\ +0100 = 4 \\ 1001 = \text{Overflow} \\ \hline (e) (+5) + (+4) \end{array}$	$\begin{array}{r} 1001 = -7 \\ +1010 = -6 \\ 10011 = \text{Overflow} \\ \hline (f) (-7) + (-6) \end{array}$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

41

Minh họa cộng hai số nguyên 4 bit trên trục số



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

42

Phép cộng hai số nguyên

- Qui tắc tràn số: Tràn số chỉ có thể xảy ra khi và chỉ khi cộng hai số nguyên cùng dấu và kết quả của phép cộng bị đảo dấu so với các toán hạng

$\begin{array}{r} 1001 = -7 \\ +0101 = 5 \\ 1110 = -2 \\ \hline (a) (-7) + (+5) \end{array}$	$\begin{array}{r} 1100 = -4 \\ +0100 = 4 \\ 10000 = 0 \\ \hline (b) (-4) + (+4) \end{array}$
$\begin{array}{r} 0011 = 3 \\ +0100 = 4 \\ 0111 = 7 \\ \hline (c) (+3) + (+4) \end{array}$	$\begin{array}{r} 1100 = -4 \\ +1111 = -1 \\ 11011 = -5 \\ \hline (d) (-4) + (-1) \end{array}$
$\begin{array}{r} 0101 = 5 \\ +0100 = 4 \\ 1001 = \text{Overflow} \\ \hline (e) (+5) + (+4) \end{array}$	$\begin{array}{r} 1001 = -7 \\ +1010 = -6 \\ 10011 = \text{Overflow} \\ \hline (f) (-7) + (-6) \end{array}$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

43

Trừ hai số nguyên

$\begin{array}{r} 0010 = 2 \\ +1001 = -7 \\ 1011 = -5 \\ \hline (a) M = 2 = 0010 \\ S = 7 = 0111 \\ -S = 1001 \end{array}$	$\begin{array}{r} 0101 = 5 \\ +1110 = -2 \\ 10011 = 3 \\ \hline (b) M = 5 = 0101 \\ S = 2 = 0010 \\ -S = 1110 \end{array}$
$\begin{array}{r} 1011 = -5 \\ +1110 = -2 \\ 11001 = -7 \\ \hline (c) M = -5 = 1011 \\ S = 2 = 0010 \\ -S = 1110 \end{array}$	$\begin{array}{r} 0101 = 5 \\ +0010 = 2 \\ 0111 = 7 \\ \hline (d) M = 5 = 0101 \\ S = -2 = 1110 \\ -S = 0010 \end{array}$
$\begin{array}{r} 0111 = 7 \\ +0111 = 7 \\ 1110 = \text{Overflow} \\ \hline (e) M = 7 = 0111 \\ S = -7 = 1001 \\ -S = 0111 \end{array}$	$\begin{array}{r} 1010 = -6 \\ +1100 = -4 \\ 10110 = \text{Overflow} \\ \hline (f) M = -6 = 1010 \\ S = 4 = 0100 \\ -S = 1100 \end{array}$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

44

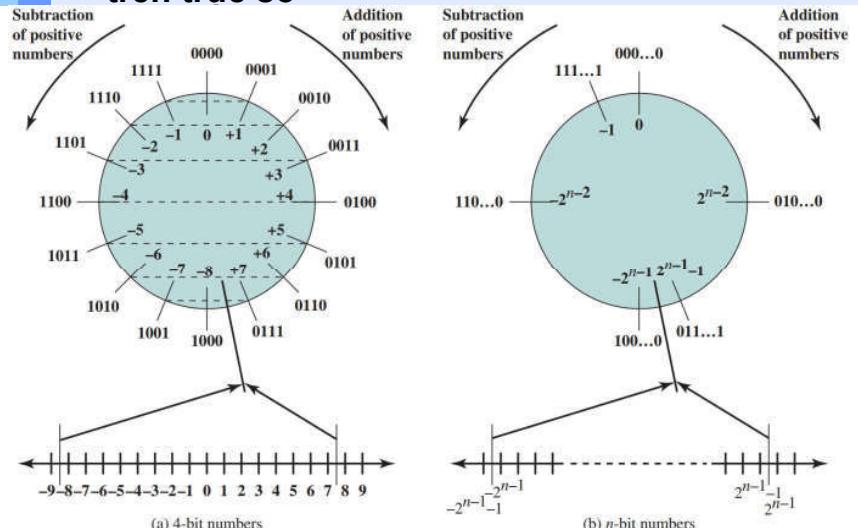
Trừ hai số nguyên

- Để trừ hai số nhị phân, ta lấy số bị trừ cộng với số bù 2 của số trừ
- Qui tắc tràn số: Tràn số chỉ có thể xảy ra khi và chỉ khi trừ hai số nguyên khác dấu và dấu của hiệu số ngược với dấu của số bị trừ

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

45

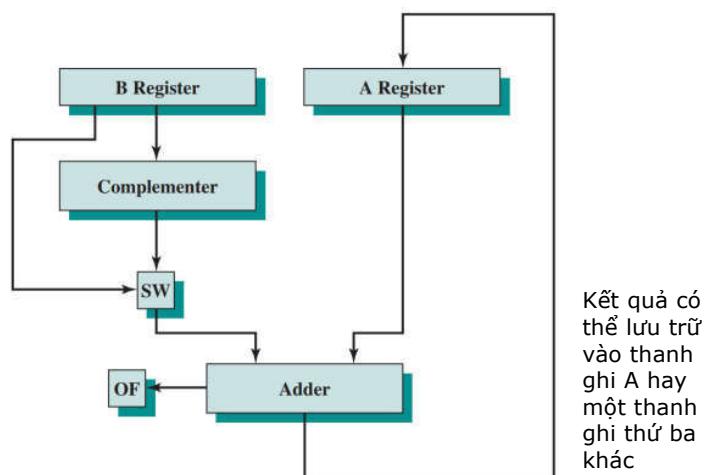
Minh họa việc cộng và trừ số nhị phân trên trục số



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

46

Sơ đồ khối phần cứng cho cộng và trừ



OF = Overflow bit

SW = Switch (select addition or subtraction)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

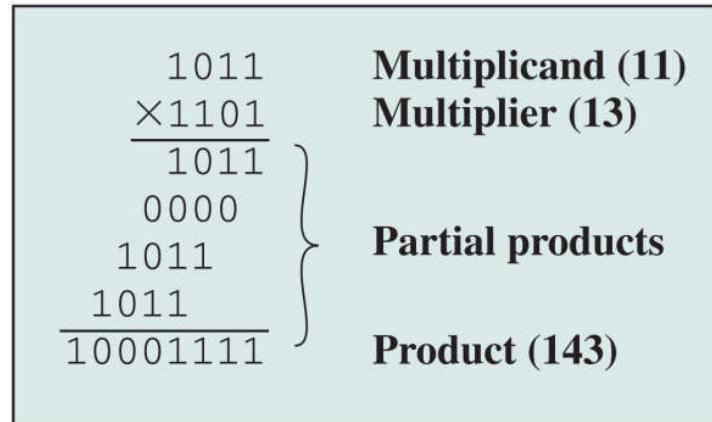
47

Phép nhân và phép chia số nguyên

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

48

Nhân hai số nguyên không dấu



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

49

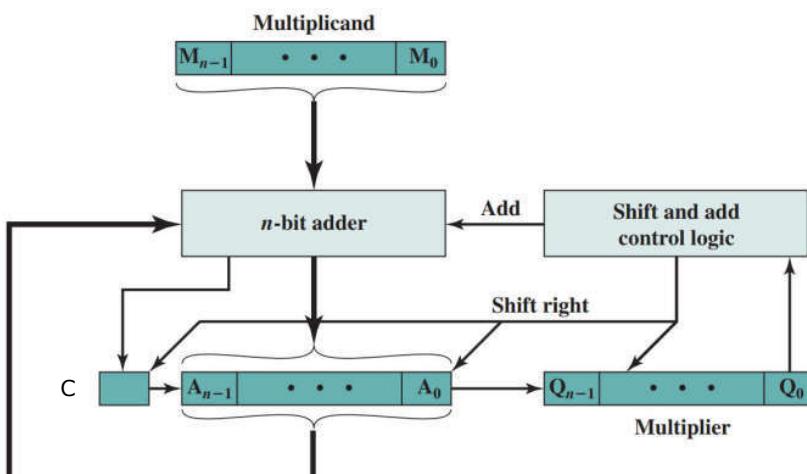
Nhân hai số nguyên không dấu

- ❖ Các tích riêng phần được xác định như sau:
 - Nếu bit của số nhân bằng 0 → tích riêng phần bằng 0.
 - Nếu bit của số nhân bằng 1 → Tích riêng phần bằng số bị nhân.
 - Tích riêng phần tiếp theo được dịch trái một bit so với tích riêng phần trước đó
- ❖ Tích bằng tổng các tích riêng phần
- ❖ Nhân hai số nguyên n-bit → tích có độ dài 2n-bit (không bao giờ bị tràn)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

50

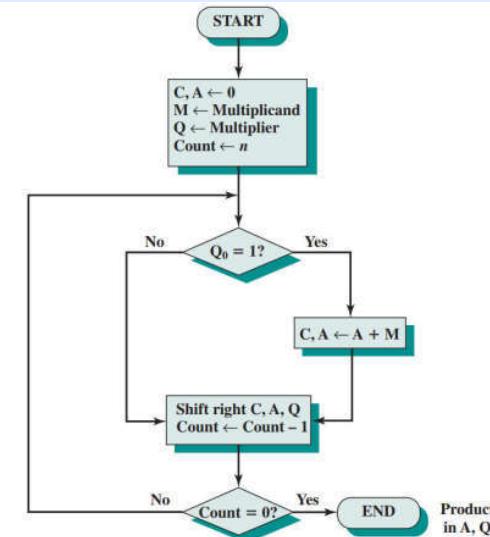
Bộ nhân số nguyên không dấu



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

51

Lưu đồ nhân số nguyên không dấu n-bit



52

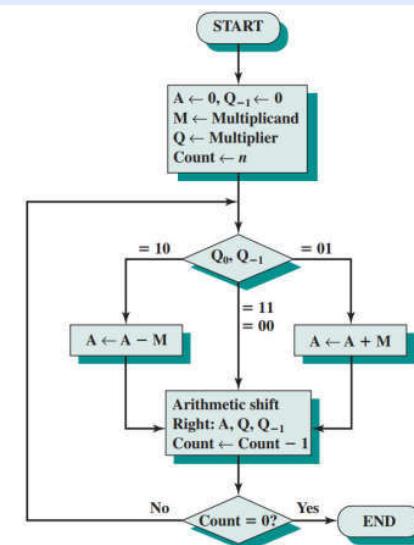
Ví dụ nhân hai số nguyên 4-bit

C	A	Q	M	
0	0000	1101	1011	Initial values
0	1011	1101	1011	Add } First
0	0101	1110	1011	Shift } cycle
0	0010	1111	1011	Shift } Second
0	1101	1111	1011	Add } Third
0	0110	1111	1011	Shift } cycle
1	0001	1111	1011	Add } Fourth
0	1000	1111	1011	Shift } cycle

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

53

Nhân hai số nguyên bù hai – giải thuật Booth



54

Ví dụ

$\begin{array}{r} 0111 \\ \times 0011 \\ \hline 11111001 \end{array}$	$\begin{array}{r} 0111 \\ \times 1101 \\ \hline 11111001 \end{array}$
(a) $(7) \times (3) = (21)$	(b) $(7) \times (-3) = (-21)$

$$(a) (7) \times (3) = (21)$$

$$(b) (7) \times (-3) = (-21)$$

$\begin{array}{r} 1001 \\ \times 0011 \\ \hline 00000111 \end{array}$	$\begin{array}{r} 1001 \\ \times 1101 \\ \hline 00000111 \end{array}$
(c) $(-7) \times (3) = (-21)$	(d) $(-7) \times (-3) = (21)$

$$(c) (-7) \times (3) = (-21)$$

$$(d) (-7) \times (-3) = (21)$$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

55

Ví dụ giải thuật Booth

$$\diamond M = 7 = 0111_2 \quad Q = 3 = 0011_2$$

A	Q	Q_{-1}	M	
0000	0011	0	0111	Initial values
1001	0011	0	0111	A ← A - M } First
1100	1001	1	0111	Shift } cycle
1110	0100	1	0111	Shift } Second
0101	0100	1	0111	A ← A + M } Third
0010	1010	0	0111	Shift } cycle
0001	0101	0	0111	Shift } Fourth
				cycle

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

56

Chia số nguyên không dấu

❖ Ví dụ

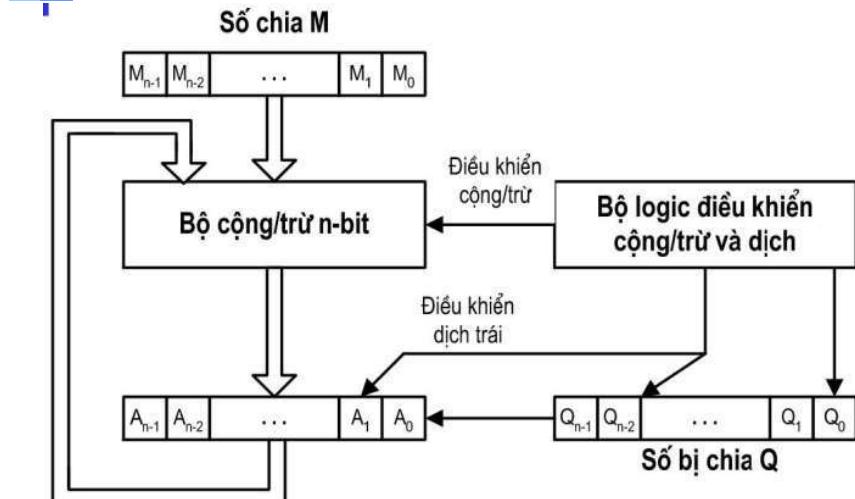
Số bị chia	10010011	\	1011	00001101	Số chia	Thương
	<u>1011</u>					
	001110					
	<u>1011</u>					
	001111					
	<u>1011</u>					
	100					

Phần dư

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

57

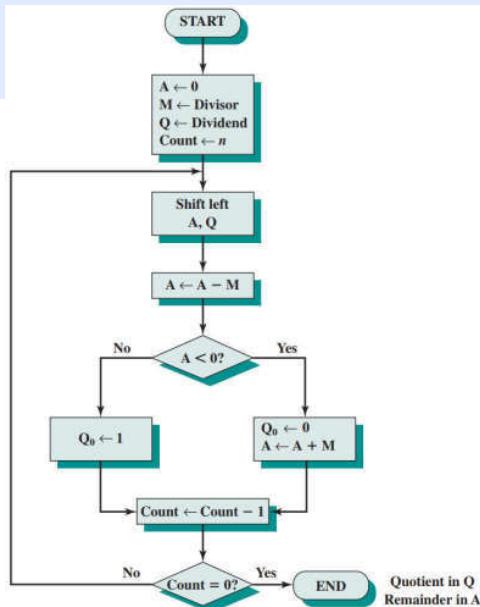
Bộ chia số nguyên không dấu



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

58

Lưu đồ



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

59

Số phẩy dấu động

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

60

Đặt vấn đề

❖ Biểu diễn số 123.375_{10} sang hệ nhị phân?

❖ **Ý tưởng đơn giản:** Biểu diễn phần nguyên và phần thập phân riêng lẻ

- Với phần nguyên: Dùng 8 bit ($[0_{10}, 255_{10}]$)
 $123_{10} = 64 + 32 + 16 + 8 + 2 + 1 = 0111\ 1011_2$
- Với phần thập phân: Tương tự dùng 8 bit
 $0.375 = 0.25 + 0.125 = 2^{-2} + 2^{-3} = 0110\ 0000_2$
 $123.375_{10} = 0111\ 1011.0110\ 0000_2$

❖ Tổng quát công thức khai triển của số thập phân hệ nhị phân:

$$x_{n-1}x_{n-2}\dots x_0.x_{-1}x_{-2}\dots x_{-m} = x_{n-1}.2^{n-1} + x_{n-2}.2^{n-2} \dots + x_0.2^0 + x_{-1}.2^{-1} + x_{-2}.2^{-2} + \dots + x_{-m}.2^{-m}$$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

61

Nguyên tắc chung

❖ Floating Point Number → biểu diễn cho số thực

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

$$X = M * R^E$$

- M là phần định trị (Mantissa),
- R là cơ số (Radix),
- E là phần mũ (Exponent).

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

63

Đặt vấn đề

❖ Với 8 bit

- Phần nguyên lớn nhất có thể biểu diễn: 255
- Phần thập phân nhỏ nhất có thể biểu diễn $2^{-8} \sim 10^{-3} = 0.001$

❖ Biểu diễn số nhỏ như $0.0001 (10^{-4})$ hay $0.000001 (10^{-5})$?

- Giải pháp: Tăng số bit cho phần thập phân
- Với 16 bit cho phần thập phân: min = $2^{-16} \sim 10^{-5}$

➔ Sử dụng Fixed-point chỉ có thể biểu diễn những số có giá trị nguyên không quá lớn và giá trị thập phân không quá nhỏ

➔ Khắc phục: Sử dụng Floating Point Number

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

62

Chuẩn IEEE 754

❖ Cơ số R=2

❖ Các dạng

- 32-bit
- 64-bit
- 128-bit

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

64



Dạng 32-bit



- ❖ S là bit dấu
 - S = 0 → số dương
 - S = 1 → số âm
- ❖ e (8 bit) là mã excess-127 của phần mũ E:
 - $e = E + 127 \rightarrow E = e - 127$
 - giá trị 127 gọi là độ lệch (bias)
- ❖ m (23 bit) là phần lẻ (sau phần nguyên) của phần định trị M:
 - $M = 1.m$
- ❖ Công thức xác định giá trị số thực

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



Ví dụ 1

- ❖ Xác định giá trị số thực được biểu diễn bằng 32 bit như sau:
- ❖ **1100 0001 0101 0110 0000 0000 0000 0000**
 - S = 1 → số âm
 - $e = 100\ 0001\ 0 = 130 \rightarrow E = 130 - 127 = 3$
 - Vậy $X = -1.10101100 \cdot 2^3 = -1101.011 = -13.375$
- ❖ **0011 1111 1000 0000 0000 0000 0000 0000** = ?
 - = +1.0



Ví dụ 2

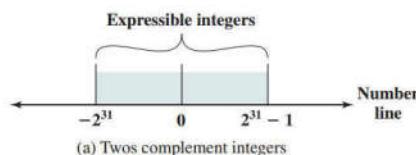
- ❖ Biểu diễn số thực $X = 83.75$ về dạng số dấu phẩy động IEEE754 32-bit
- ❖ $X = 83.75_{(10)} = 1010011.11_{(2)} = 1.01001111 \times 2^6$
- ❖ Ta có:
 - S = 0 vì X là số dương
 - $E = e - 127 = 6 \rightarrow e = 133_{10} = 1000\ 0101_2$
 - Vậy:
 $X = 0100\ 0010\ 1010\ 0111\ 1000\ 0000\ 0000$



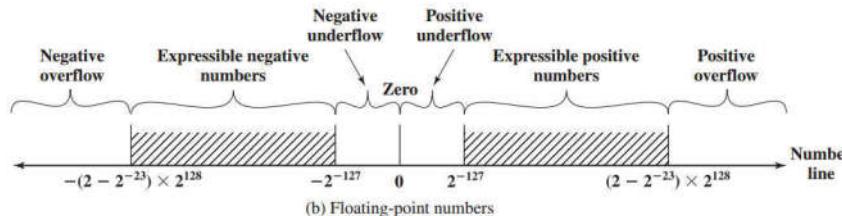
Ví dụ 3

- ❖ Biểu diễn số thực $X = -0,2$ về dạng số dấu phẩy động IEEE754 32-bit
- ❖ Giải
 - $X = -0,2_{10} = -0.00110011...0011..._{(2)} = -1.100110011..0011... \times 2^{-3}$
 - Ta có
 - S = 1 vì X là số âm
 - $E = e - 127 = -3 \rightarrow e = 124_{10} = 0111\ 1100_2$
 - Vậy:
 $X = 1011\ 1110\ 0100\ 1100\ 1100\ 1100\ 1100\ 1100$

Overflow / Underflow



(a) Two's complement integers

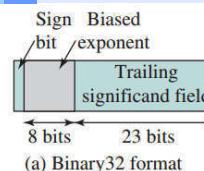


(b) Floating-point numbers

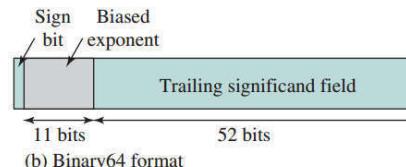
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

69

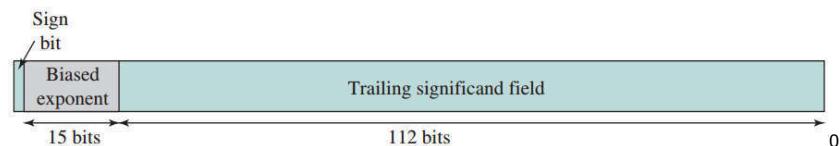
IEEE 754 Format



(a) Binary32 format



(b) Binary64 format



112 bits

0

IEEE 754 Format

Parameter	Format		
	Binary32	Binary64	Binary128
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	-126	-1022	-16382
Approx normal number range (base 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2^{23}	2^{52}	2^{112}
Number of values	1.98×2^{31}	1.99×2^{63}	1.99×2^{128}
Smallest positive normal number	2^{-126}	2^{-1022}	2^{-16362}
Largest positive normal number	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$
Smallest subnormal magnitude	2^{-149}	2^{-1074}	2^{-16494}

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

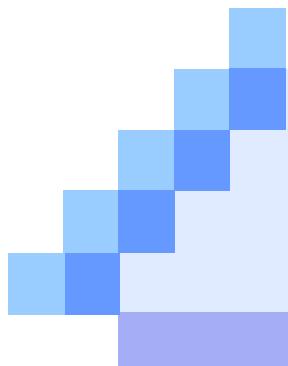
71

Bài tập

- Giả sử 185 và 122 là những số nguyên hệ thập phân không dấu 8-bit. Tính $185 - 122$, $185 + 122$
- $5E4D_{16} = ?_2$
- Biểu diễn các số nguyên có dấu 16-bit: 158, -210
- X= 54.45 về dạng số dấu phẩy động IEEE754 32-bit
- Giá trị thập phân của: các số IEEE754 32 bit
 - 1 10000011 11000000000000000000000000000000
 - 0 01111110 10100000000000000000000000000000
 - 0 10000000 00000000000000000000000000000000

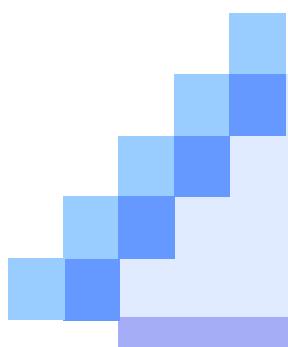
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

72



3. MẠCH LOGIC

Huỳnh Tuấn Anh
Khoa CNTT, ĐHNT
Phone: 094 5505 449
Email: anhht@ntu.edu.vn



Đại số Boole



Nội dung

- ❖ Đại số BOOLE
- ❖ Các công Logic
- ❖ Mạch tổ hợp
- ❖ Mạch dãy
- ❖ Bài tập

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

2



Đại số Boole

- ❖ Đại số Boole sử dụng các biến logic và phép toán logic
- ❖ Biến logic có thể nhận giá trị 1 (TRUE) hoặc 0 (FALSE)
- ❖ Phép toán logic cơ bản là AND, OR và NOT với ký hiệu như sau:
 - A AND B = A·B
 - A OR B = A + B
 - NOT A = \bar{A}
- ❖ Thứ tự ưu tiên NOT > AND > OR

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

4

Các phép toán logic..

❖ Các phép toán NAND, NOR, XOR:

- A NAND B = $\overline{A \cdot B}$
- A NOR B = $\overline{A + B}$
- A XOR B = $A \cdot \overline{B} + \overline{A} \cdot B$

Phép toán đại số Boole

P	Q	NOT P (\bar{P})	P AND Q ($P \cdot Q$)	P OR Q ($P + Q$)
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

P	Q	P NAND Q ($\overline{P \cdot Q}$)	P NOR Q ($\overline{P + Q}$)	P XOR Q ($P \oplus Q$)
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

Các đồng nhất thức của đại số Boole

Basic Postulates	
$A \cdot B = B \cdot A$	$A + B = B + A$
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
$1 \cdot A = A$	$0 + A = A$
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
Other Identities	
$0 \cdot A = 0$	$1 + A = 1$
$A \cdot A = A$	$A + A = A$
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Các cổng logic

Gates

- ❖ Thực hiện các hàm logic
 - NOT, AND, OR, NAND, NOR, etc.
- ❖ Cổng logic một đầu vào:
 - NOT
- ❖ Cổng logic hai đầu vào
 - AND, OR, XOR, NAND, NOR, XNOR
- ❖ Cổng logic nhiều đầu vào

Gates

Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \cdot B$ or $F = AB$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{A}$ or $F = A'$	<table border="1"> <thead> <tr> <th>A</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	

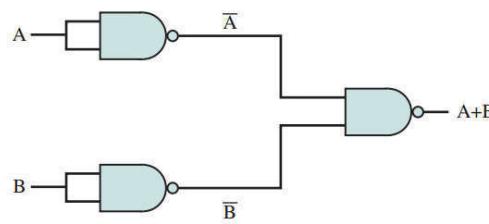
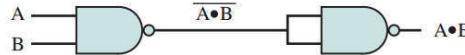
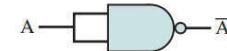
Gates

Name	Graphical Symbol	Algebraic Function	Truth Table															
NAND		$F = \overline{AB}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$F = A \oplus B$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Tập đầy đủ

- ❖ Là tập các cổng có thể thực hiện được bất kỳ hàm logic nào từ các cổng của tập đó.
- ❖ Một số ví dụ về tập đầy đủ:
 - {AND, OR, NOT}
 - {AND, NOT}
 - {OR, NOT}
 - {NAND}
 - {NOR}

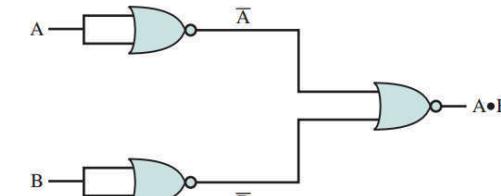
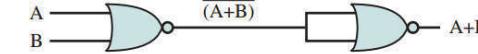
Sử dụng cổng NAND



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

13

Sử dụng cổng NOR



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

14

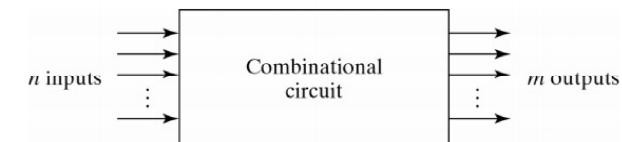
Mạch tổ hợp

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

15

Mạch tổ hợp

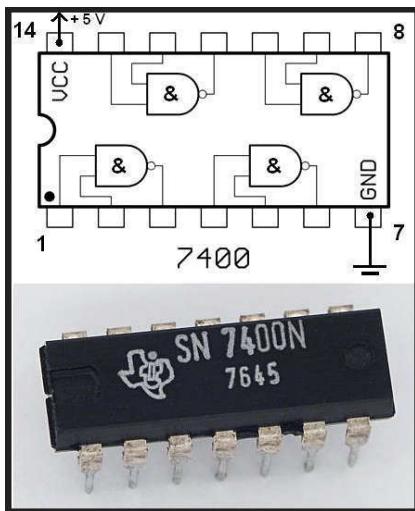
- ❖ Gồm n ngõ vào (input); m ngõ ra (output)
 - Mỗi ngõ ra là 1 hàm luận lý của các ngõ vào
- ❖ Mạch tổ hợp không mang tính ghi nhớ: Ngõ ra chỉ phụ thuộc vào Ngõ vào hiện tại, không xét những giá trị trong quá khứ



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

16

Ví dụ mạch tổ hợp



17

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

Mô tả mạch tổ hợp

- ❖ Bảng chân trị
 - n input, m output
 - 2^n hàng, $(n+m)$ cột
- ❖ Dạng sơ đồ
- ❖ Bảng công thức (hàm luận lý)

18

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

Thiết kế mạch logic

- ❖ Thường tiến hành qua ba bước

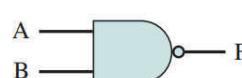
- Lập bảng chân trị

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

- Viết hàm luận lý

- $F = \overline{A} \cdot \overline{B}$

- Vẽ sơ đồ mạch và thử nghiệm



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

19

SOP – Sum of Products

- ❖ Giả sử đã có bảng chân trị cho mạch n đầu vào x_1, x_2, \dots, x_n và 1 đầu ra f .

- ❖ Ta dễ dàng thiết lập công thức (hàm) logic theo cách sau:

- Ứng với mỗi hàng của bảng chân trị có đầu ra = 1 ta tạo thành 1 tích có dạng $u_1 \cdot u_2 \dots u_n$ với:

- $u_i = x_i$ nếu $x_i = 1$

- $u_i = \overline{x}_i$ nếu $x_i = 0$

- Cộng các tích tìm được lại thành tổng ta có được công thức của hàm f

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

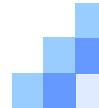
20



Ví dụ

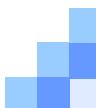
x_1	x_2	x_3	f
0	0	0	0
0	0	1	1 → $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3$
0	1	0	1 → $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$
0	1	1	0
1	0	0	0
1	0	1	1 → $x_1 \cdot \bar{x}_2 \cdot x_3$
1	1	0	0
1	1	1	0

$f = \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 x_3$



POS – Product Of Sum

- ❖ Trường hợp số hàng có giá trị đầu ra bằng 1 nhiều hơn bằng 0, ta có thể đặt $g = \bar{f}$
 - Viết công thức dạng SOP cho g
 - Lấy $f = \bar{g}$ để có công thức dạng POS (tích các tổng) của f



Ví dụ

x	y	z	f	g
0	0	0	1	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$$g = \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$

$$f = \bar{g} = (x + \bar{y} + z)(\bar{x} + y + z)$$



Đơn giản hóa hàm logic

- ❖ Sau khi viết được hàm logic, ta có thể vẽ sơ đồ của mạch tổ hợp từ những cộng luận lý cơ bản
 - Ví dụ: $f = xy + xz$
- ❖ Tuy nhiên ta có thể viết lại hàm logic sao cho sơ đồ mạch sử dụng ít công hơn
 - Ví dụ: $f = xy + xz = x(y + z)$
- ❖ Cách đơn giản hóa hàm tổng quát? Một số cách phổ biến:
 - Dùng đại số Boole
 - Dùng bản đồ Karnaugh

PP sử dụng đại số Boole

- ❖ Dùng các phép biến đổi đại số Boole để lược giản hàm logic
- ❖ Khuyết điểm:
 - Không có cách làm tổng quát cho mọi bài toán
 - Không có cách làm tổng quát cho mọi bài toán
- ❖ Ví dụ: Đơn giản hóa các hàm sau:
 - $F(x,y,z) = xyz + \bar{x}yz + x\bar{y}z + xy\bar{z}$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

25

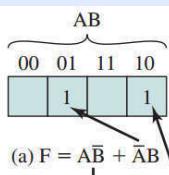
PP Bản đồ Karnaugh

- ❖ Karnaugh là một cách tiện lợi để đơn giản hóa hàm Boolean với số biến nhỏ (tối đa 4 biến)
 - n biến sử dụng mảng 2^n hình vuông
 - Mỗi hình vuông tương ứng với một tích duy nhất trong tổng các tích

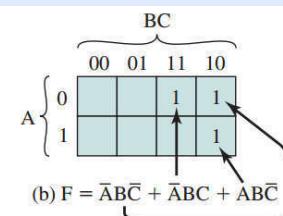
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

26

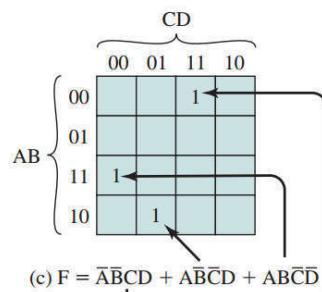
Ví dụ



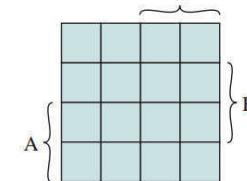
$$(a) F = A\bar{B} + \bar{A}B$$



$$(b) F = \bar{A}\bar{B}\bar{C} + \bar{A}BC + ABC$$



$$(c) F = \bar{A}\bar{B}CD + A\bar{B}CD + AB\bar{C}D$$



(d) Simplified labeling of map

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

27

Đơn giản hàm theo dạng SOP

- ❖ Hàm logic F biểu diễn bằng chân trị được đưa vào bản đồ bằng các trị 1 tương ứng
- ❖ Các ô liền kề có giá trị 1 được gom thành nhóm sao cho mỗi nhóm sau khi gom có tổng số ô là luỹ thừa của 2 (2, 4, 8,...)
 - Hai ô liền kề nếu bộ trị của chúng chỉ khác nhau ở một giá trị của một biến
- ❖ Các nhóm có thể dùng chung ô có giá trị 1 để tạo thành nhóm lớn hơn. Có gắng tạo những nhóm lớn nhất có thể
- ❖ Các nhóm có thể dùng chung ô có giá trị 1 để tạo thành nhóm lớn hơn. Có gắng tạo những nhóm lớn nhất có thể.
- ❖ Mọi nhóm biểu diễn 1 số hạng nhân (Product), Cộng (Sum - OR) các số hạng này ta sẽ được biểu thức tối giản của hàm logic F

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

28

B1. Lập bảng chân trị

❖ Gọi các ngõ vào là x, y, z - ngõ ra là f

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

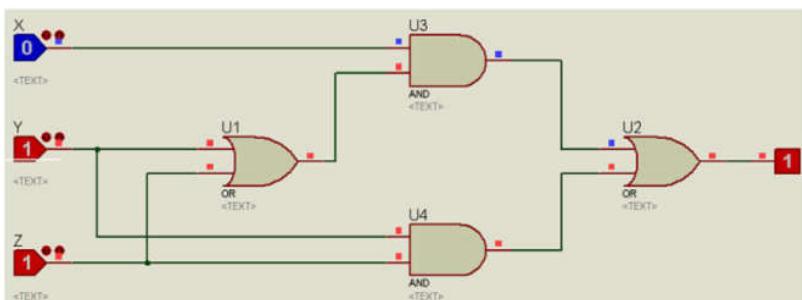
Bước 2. Viết hàm logic

❖ $F(x,y,z) = xyz + \bar{x}yz + x\bar{y}z + xy\bar{z}$

❖ Đơn giản hàm F:

❖ $F = xy + yz + zx = x(y + z) + yz$

B3. Vẽ sơ đồ mạch và thử nghiệm



Một số mạch tổ hợp cơ bản

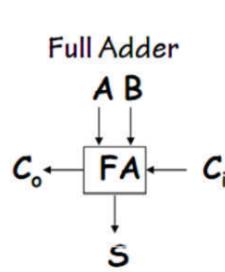
❖ Mạch toàn cộng (Full adder)

❖ Bộ dòn kênh (Multiplexer-MUX)

❖ Bộ giải mã (Decoder)

Mạch toàn cộng (Full Adder)

- Mạch tổ hợp thực hiện phép cộng số học 3 bit ở 3 ngõ vào
- Gồm 3 ngõ vào (A, B : bit cần cộng – C_i : bit nhớ) và 2 ngõ ra
(kết quả có thể từ 0 đến 3 với giá trị 2 và 3 cần 2 bit biểu diễn
– S : ngõ tổng, C_o : ngõ nhớ)

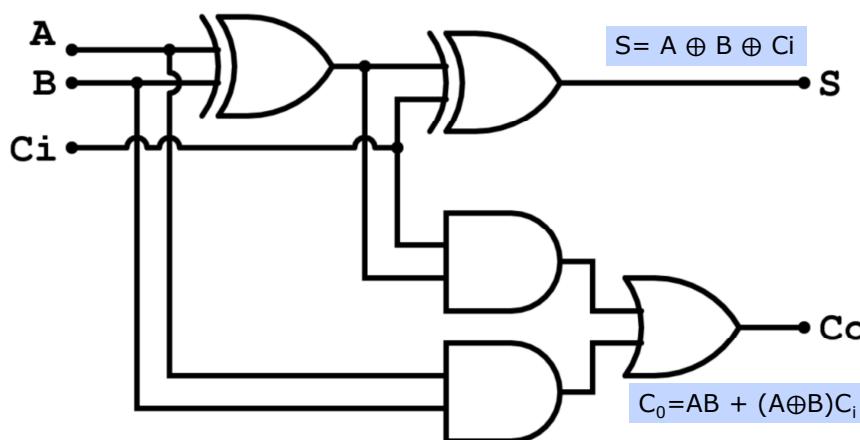


A	B	C_i	S	C_o
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

37

Sơ đồ mạch Full Adder



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

39

B2. Viết hàm logic

- $S = \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i$
- $C_o = AB\bar{C}_i + \bar{A}BC_i + A\bar{B}C_i + ABC_i$

		AB			
		00	01	11	10
C_i	0	1			
	1			1	

		AB			
		00	01	11	10
C_0	0		1		
	1			1	1

$$S = \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i \\ = A \oplus B \oplus C_i$$

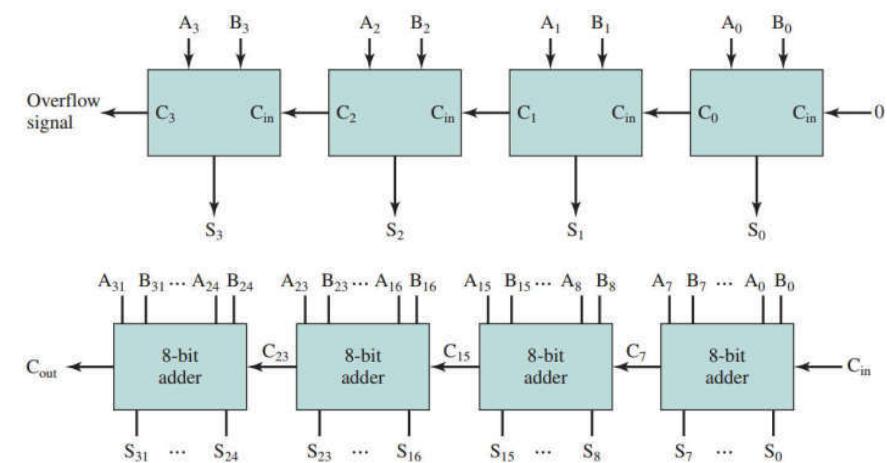
Chú ý: $x \oplus y = \bar{x}y + x\bar{y}$
 $\bar{x} \oplus y = xy + x\bar{y}$

$$C_o = AB\bar{C}_i + \bar{A}BC_i + A\bar{B}C_i + ABC_i \\ = \bar{A}BC_i + A\bar{B}C_i + AB \\ = AB + (\bar{A}B + A\bar{B})C_i \\ = AB + (A \oplus B)C_i$$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

38

Mạch cộng 4-bit, 32-bit

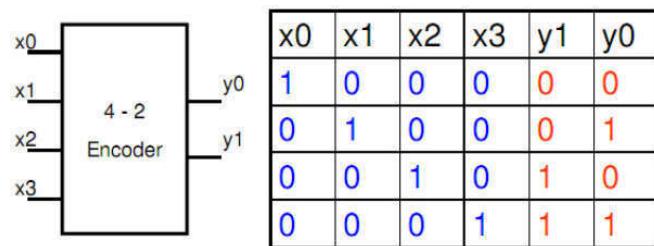


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

40

Mạch mã hóa nhị phân (Binary Encoder)

- ❖ Có 2^n (hoặc ít hơn) ngõ vào, n ngõ ra
- ❖ Quy định chỉ có duy nhất một ngõ vào mang giá trị = 1 tại một thời điểm
- ❖ Nếu ngõ vào = 1 đó là ngõ thứ k thì các ngõ ra tạo thành số nhị phân có giá trị = k



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

41

Sơ đồ mạch 4-2 Binary Encoder

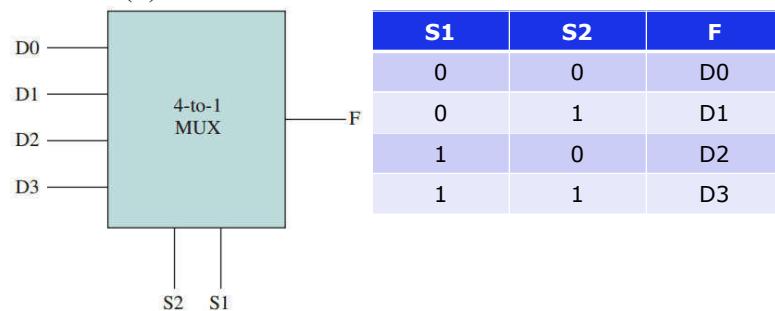
- ❖ Ngõ vào X0, X1, X2, X3
- ❖ Ngõ ra: Y0, Y1
- ❖ Hàm chức năng:
 - $Y_0 = \overline{X_0} \cdot X_1 \cdot \overline{X_2} \cdot \overline{X_3} + \overline{X_0} \cdot \overline{X_1} \cdot \overline{X_2} \cdot X_3$
 - $Y_1 = \overline{X_0} \cdot \overline{X_1} \cdot X_2 \cdot \overline{X_3} + \overline{X_0} \cdot \overline{X_1} \cdot \overline{X_2} \cdot X_3$

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

42

Bộ dòn kênh (Multiplexer-MUX)

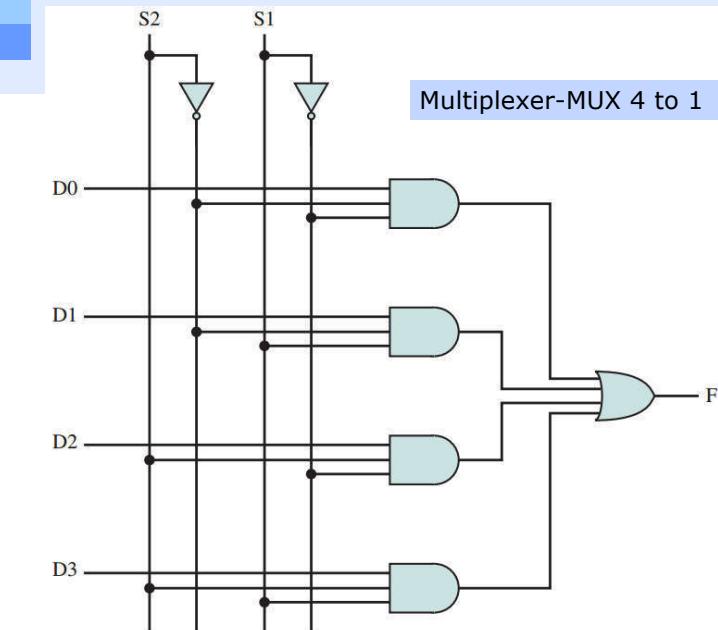
- ❖ 2^n đầu vào dữ liệu
- ❖ n đầu vào chọn
- ❖ 1 đầu ra
- ❖ Đầu vào chọn (S) xác định đầu vào nào (D) sẽ được nối với đầu ra (F).



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

43

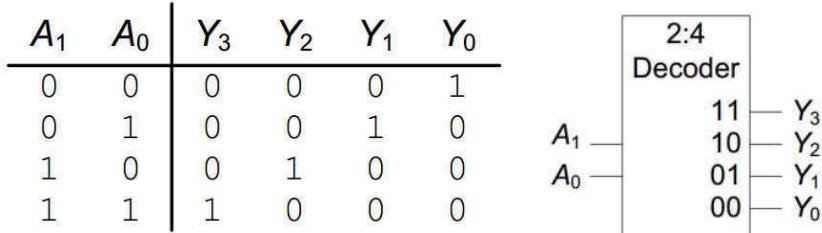
Multiplexer-MUX 4 to 1



44

Bộ giải mã (Decoder)

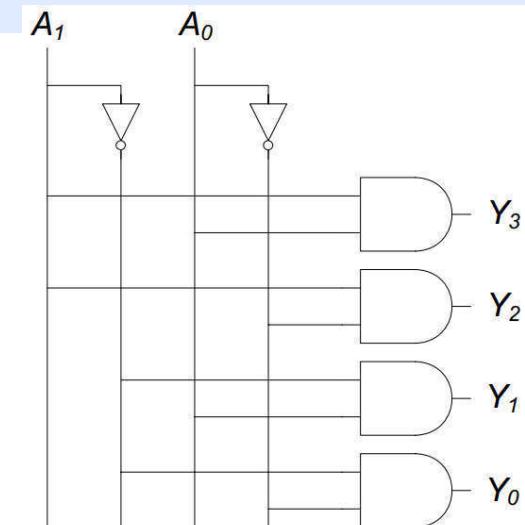
- N đầu vào, 2^N đầu ra
- Chỉ có một đầu ra tích cực (được chọn) tương ứng với một tổ hợp của N đầu vào.



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

45

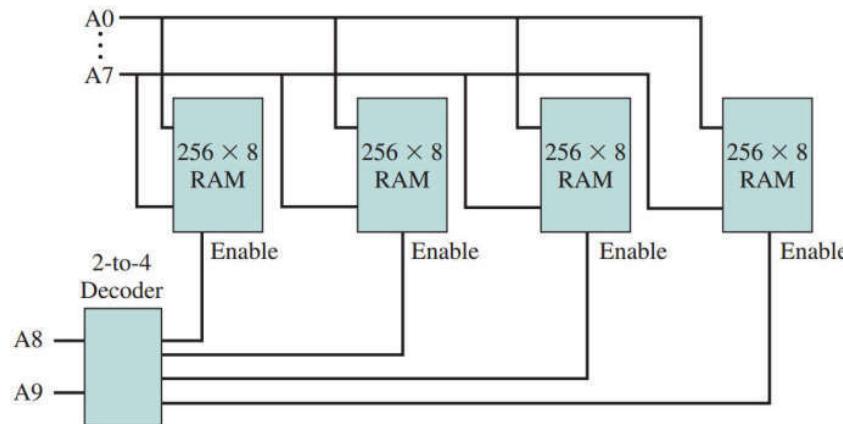
Bộ giải mã 2 to 4



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

46

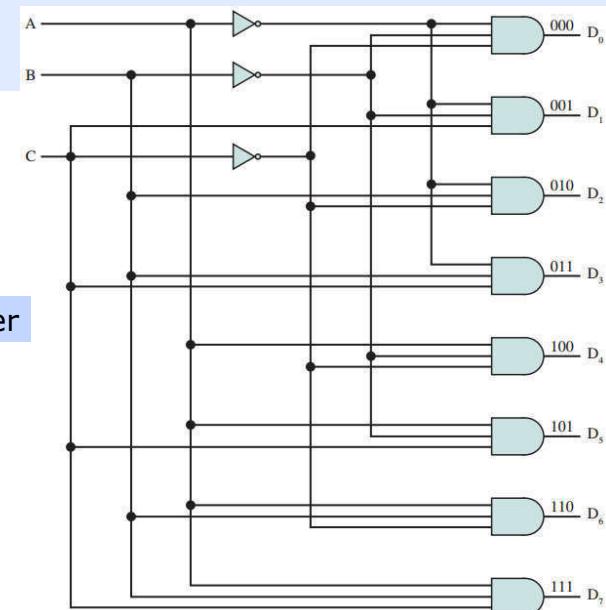
Giải mã địa chỉ (Address Decoding)



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

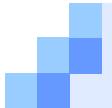
47

3 to 8 Decoder



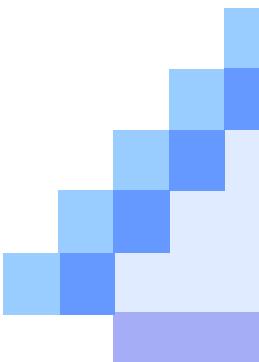
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

48

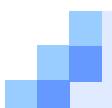


Bài tập: Thiết kế mạch ALU

- ❖ $F = (5X + 2Y) \% 4$
- ❖ Input: X (2 bit), Y (2 bit)
- ❖ Output: F (2 bit)

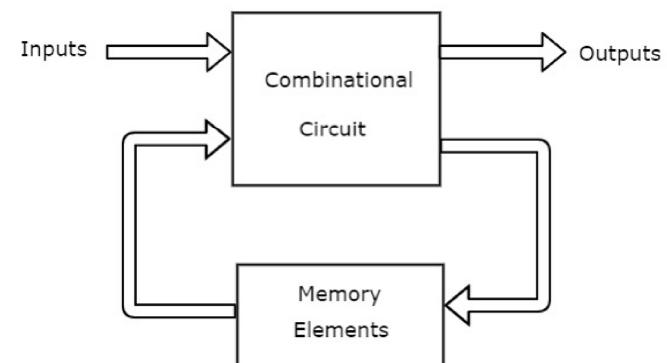


MẠCH DÃY



Mạch dãy

- ❖ Mạch dãy là mạch logic trong đó đầu ra phụ thuộc giá trị đầu vào ở thời điểm hiện tại và quá khứ
- ❖ Là mạch có nhớ, được thực hiện bằng phần tử nhớ (Latch, Flip-Flop) và có thể kết hợp với các công nghệ logic cơ bản
- ❖ Mạch tuần tự có khả năng “ghi nhớ” các trạng thái trong quá khứ
- ❖ Mạch dãy bao gồm:
 - Mạch tổ hợp
 - Mạch hồi tiếp

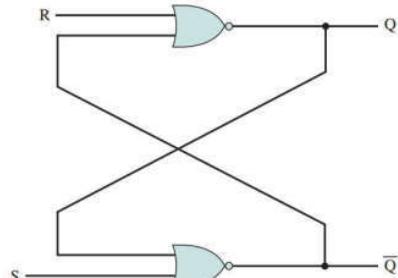




Flip-Flops

S-R Latch

- Có hai cổng
 - S: Set
 - R: Reset
- Có thể thực hiện chức năng của bộ nhớ 1-bit
 - Q: Giá trị của bit nhớ
 - Đầu vào S, R điều khiển việc ghi giá trị 0 và 1 tương ứng vào bộ nhớ



(b) Simplified Characteristic Table

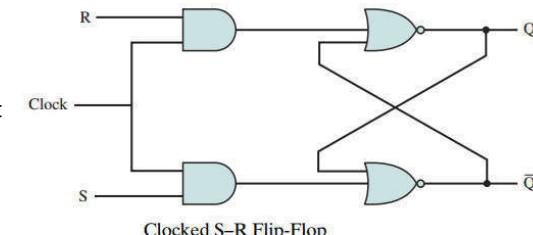
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

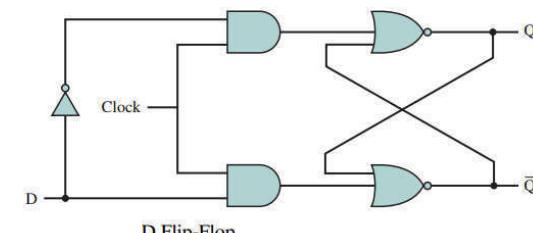
53

CLOCKED S-R FLIP-FLOP

Giá trị đầu ra Q chỉ có thể thay đổi khi xung clock (clock pulse) xuất hiện



Clocked S-R Flip-Flop



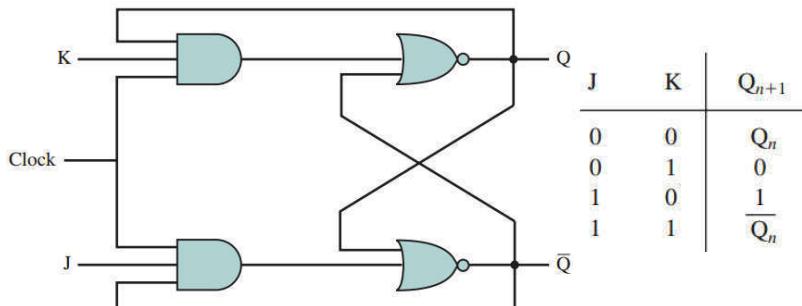
D Flip-Flop

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

54



J-K Flip-Flop



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	\bar{Q}_n

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

55

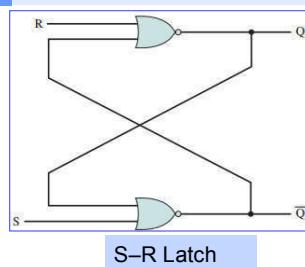
Các Flip-Flop cơ bản

Name	Graphical Symbol	Truth Table															
S-R		<table border="1"> <thead> <tr> <th>S</th> <th>R</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>-</td> </tr> </tbody> </table>	S	R	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	-
S	R	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	-															
J-K		<table border="1"> <thead> <tr> <th>J</th> <th>K</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>\bar{Q}_n</td> </tr> </tbody> </table>	J	K	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	\bar{Q}_n
J	K	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	\bar{Q}_n															
D		<table border="1"> <thead> <tr> <th>D</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D	Q_{n+1}	0	0	1	1									
D	Q_{n+1}																
0	0																
1	1																

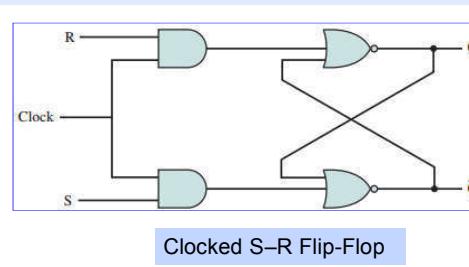
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

56

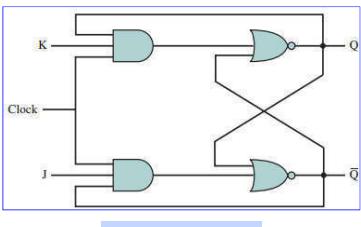
R-S Latch và các Flip-Flop



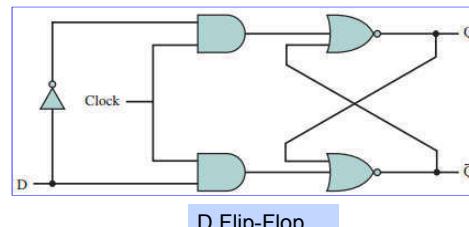
S-R Latch



Clocked S-R Flip-Flop



J-K Flip-Flop

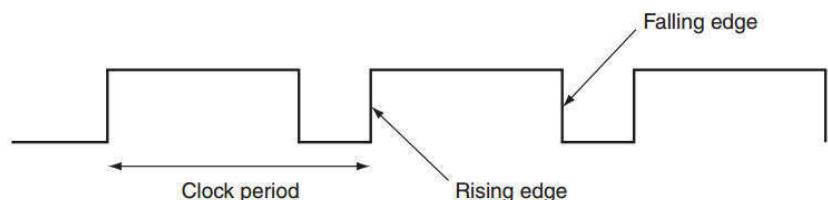


D Flip-Flop

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

57

Clock



edge-triggered clocking: Là một hệ thống xung mà tất cả sự thay đổi trạng thái xảy ra ở cạnh xung (clock edge)

- rising edge-triggered flip-flops
- falling edge-triggered flip-flops

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

58

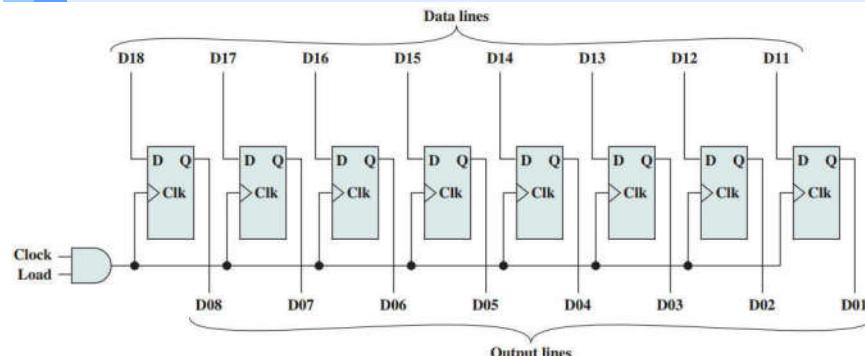
Registers

- ❖ Là các thành phần cơ bản của CPU
- ❖ Là các mạch số được sử dụng bên trong CPU để lưu trữ một hay nhiều bit dữ liệu
- ❖ 2 dạng thanh ghi cơ bản:
 - thanh ghi song song (parallel register)
 - thanh ghi dịch (shift register)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

59

PARALLEL REGISTERS



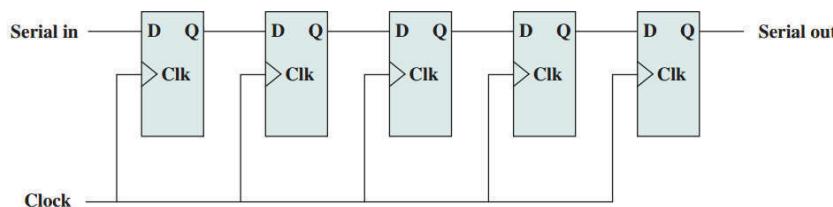
- Thanh ghi 8-bit song song
- Bao gồm một tập các bộ nhớ 1-bit có thể đọc ghi đồng thời
- Mỗi phần tử nhớ là một D flip-flops
- Load: Tín hiệu điều khiển ghi vào thanh ghi các tín hiệu từ Data lines, D11→D18

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

60



SHIFT REGISTER



- ❖ 5-bit shift register
- ❖ Mỗi phần tử là một Clocked D flip-flops
- ❖ Dữ liệu được đưa vào flip-flops bên trái cùng
- ❖ Ứng với mỗi Clock dữ liệu được dịch chuyển sang bên phải một vị trí và bit ở ngoài cùng bên phải được đưa ra ngoài

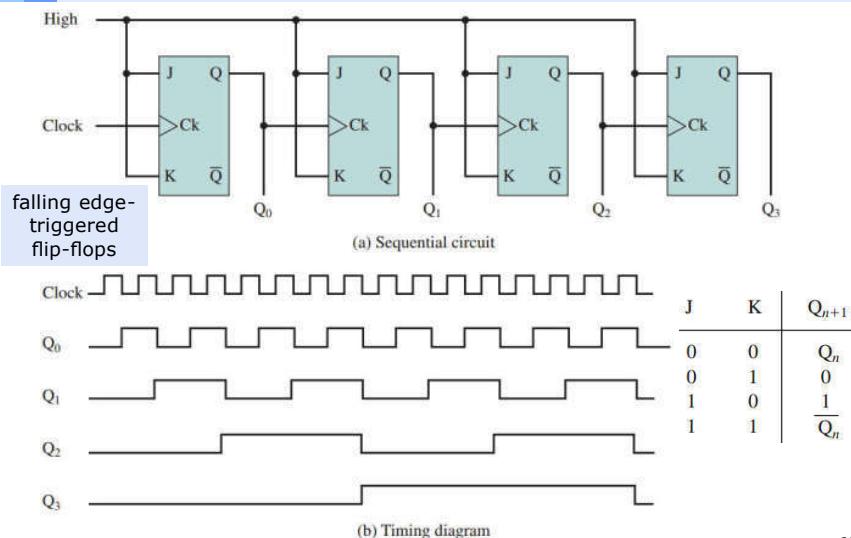


Counter

- ❖ Bộ đếm là một thanh ghi mà giá trị của nó có thể tăng thêm 1 chia lấy phần dư cho khả năng đếm tối đa của thanh ghi.
 - Giá trị tiếp theo của giá trị cực đại là 0.
 - Than ghi gồm n flip-flops có thể đếm được tối đa $2^n - 1$
- ❖ Ví dụ về bộ đếm trong CPU là thanh ghi Program Counter



4 bit Counter



Bài tập

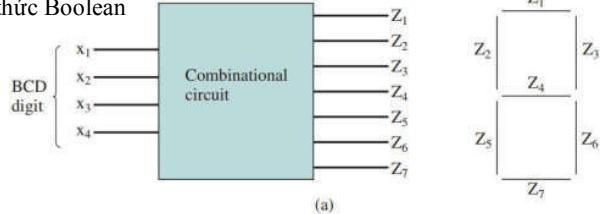
1. Xây dựng bảng chân trị cho các biểu thức Boolean:
 - a. $ABC + \overline{A} \overline{B} \overline{C}$
 - b. $ABC + A \overline{B} \overline{C} + \overline{A} \overline{B} \overline{C}$
 - c. $A(B\overline{C} + \overline{B}C)$
 - d. $(A + B)(A + C)(\overline{A} + \overline{B})$
2. Xây dựng toán tử XOR từ các toán tử Boolean cơ bản AND, OR và NOT
3. Cho cổng NOR và cổng NOT, vẽ mạch logic thực hiện chức năng AND ba ngõ vào.
4. Thiết kế mạch dòn kênh (Multiplexer) 8 – to – 1



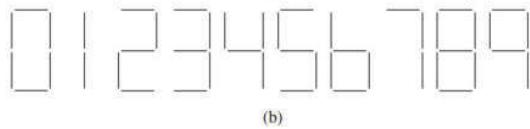
Bài tập

5. Thiết kế mạch tổ hợp hiển thị các chữ số từ 0 → 9

- Trình bày bảng chân trị.
- Biểu diễn bảng chân trị bằng dạng SOP
- Biểu diễn bảng chân trị bằng dạng POS
- Rút gọn biểu thức Boolean



(a)



(b)



Bài tập

1. Thiết kế mạch giải mã 5×32 bằng cách sử dụng 4 mạch giải mã 3×8 (với cổng vào kích hoạt – enable input) và một cổng giải mã 2×4 .

4. HỆ THỐNG MÁY TÍNH

Huỳnh Tuấn Anh
Khoa CNTT, ĐHNT
Phone: 094 5505 449
Email: anhht@ntu.edu.vn

Các thành phần cơ bản của máy tính

Nội dung

- ❖ Các thành phần cơ bản của máy tính
- ❖ Hoạt động cơ bản của máy tính
- ❖ Bus máy tính

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

2

Các thành phần cơ bản của máy tính

- ❖ Bộ xử lý trung tâm (CPU)
- ❖ Bộ nhớ (Memory)
- ❖ Vào ra (Input/Output)
- ❖ Bus liên kết hệ thống (System Interconnection Bus)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

4

1. Bộ xử lý trung tâm (CPU)

❖ Chức năng:

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

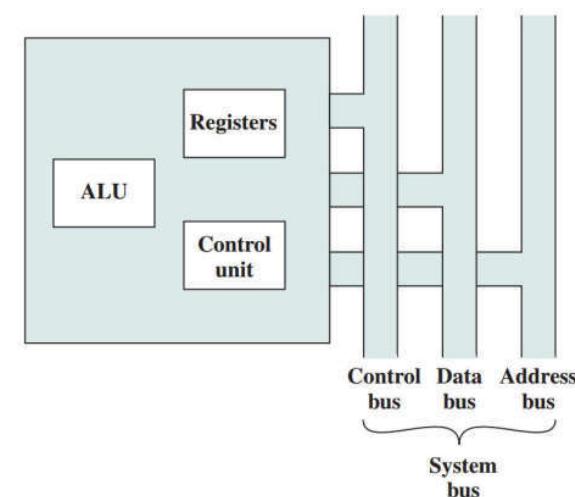
❖ Nguyên tắc hoạt động cơ bản:

- CPU hoạt động theo chương trình nằm trong bộ nhớ chính

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

5

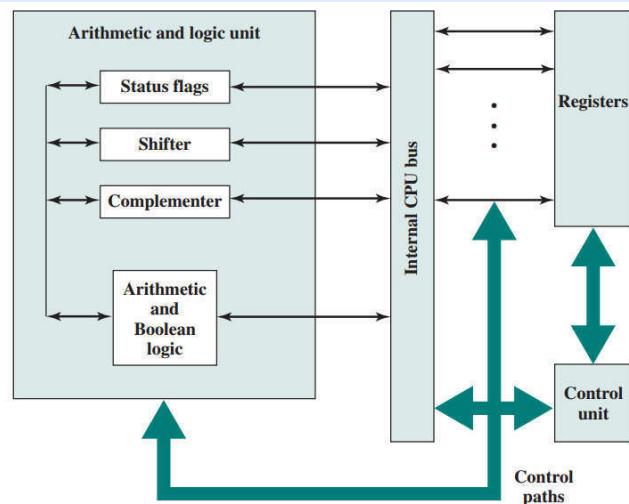
Cấu trúc cơ bản của CPU



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

6

ALU



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

7

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):** lưu giữ các thông tin tạm thời phục vụ cho hoạt động của CPU.
- ❖ **Đơn vị nối ghép bus (Bus Interface Unit - BIU):** kết nối và trao đổi thông tin giữa bus bên trong (*internal bus*) và bus bên ngoài (*external bus*).

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

8

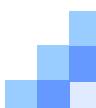
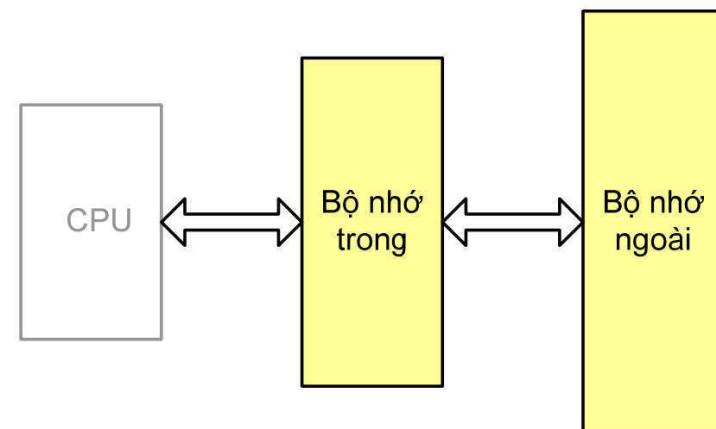


2. Bộ nhớ máy tính

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



Các thành phần chính của bộ nhớ máy tính



Bộ nhớ trong

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



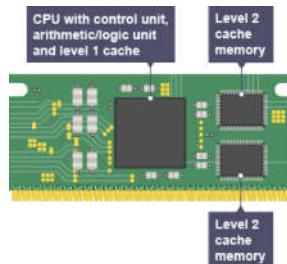
Bộ nhớ chính (Main Memory)

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

Nội dung	Địa chỉ
1011 0010	0000
1110 0010	0001
0001 1111	0010
1010 1011	0011
0000 1000	0100
1111 1111	0101
0011 1100	0110
1000 1111	0111
1111 0001	1000
0011 1101	1001
1000 1111	1010
0011 0011	1011
1100 1101	1100
0101 1010	1101
1000 1101	1110
1111 0000	1111

Bộ nhớ cache

- ❖ 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
- ❖ Tốc độ nhanh hơn
- ❖ Cache thường được chia thành một số mức
- ❖ Cache có thể được tích hợp trên cùng chip bộ xử lý.
- ❖ Cache có thể có hoặc không



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

13

Bộ nhớ ngoài (External Memory)

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

14

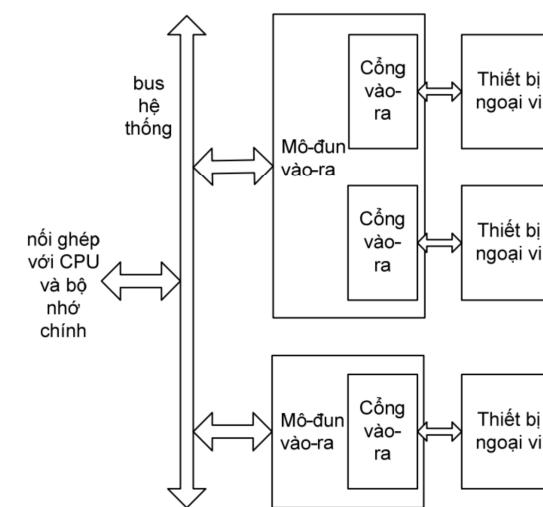
3. Modul vào ra (Input – Output)

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

15

Cấu trúc cơ bản của modul vào-ra



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

16



Các thiết bị ngoại vi (Peripheral devices)

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



Module vào ra

- ❖ Chức năng: nối ghép các thiết bị ngoại vi với máy tính
- ❖ Mỗi module 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 ngoại vi được kết nối và trao đổi dữ liệu với máy tính thông qua các cổng vào-ra.



Hoạt động cơ bản của máy tính

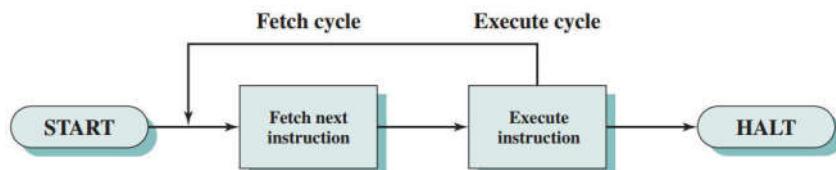


Hoạt động cơ bản của máy tính

- ❖ Thực hiện chương trình
- ❖ Hoạt động ngắn
- ❖ Hoạt động vào ra

1. Thực hiện chương trình

- ❖ Là hoạt động cơ bản của máy tính
- ❖ Máy tính lặp đi lặp lại hai bước:
 - Nhận lệnh
 - Thực hiện lệnh
- ❖ Thực hiện chương trình bị dừng nếu việc thực hiện lệnh bị lỗi hoặc gặp lệnh dừng.



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

21

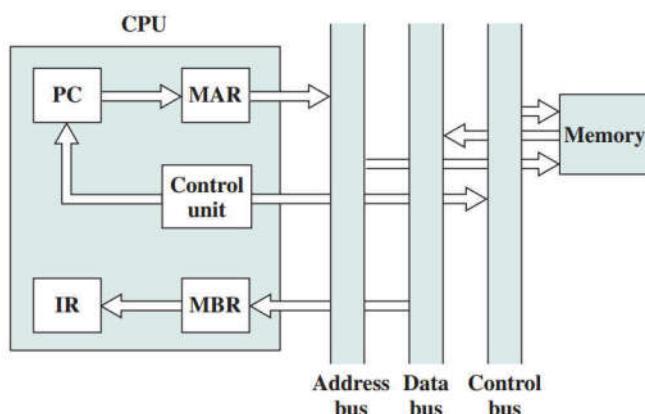
Nhận lệnh

- ❖ Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính.
- ❖ Bộ đếm chương trình PC (Program Counter) của CPU giữ địa chỉ của lệnh sẽ được nhận.
- ❖ CPU nhận lệnh từ ngăn nhớ được trỏ bởi PC.
- ❖ Lệnh được nạp vào thanh ghi lệnh IR (Instruction Register).
- ❖ Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

22

Luồng dữ liệu, chu trình nhận lệnh



MBR = Memory buffer register

MAR = Memory address register

IR = Instruction register

PC = Program counter

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

23

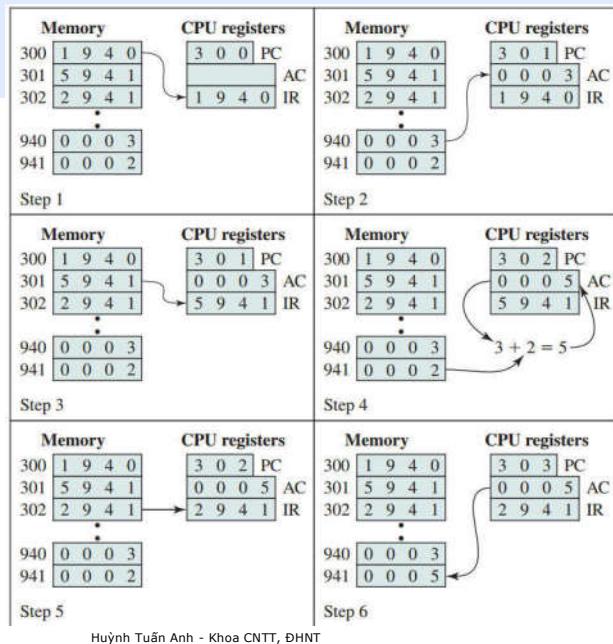
Thực hiện lệnh

- ❖ 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 của lệnh:
 - Trao đổi dữ liệu giữa CPU và bộ nhớ chính
 - Trao đổi dữ liệu giữa CPU và module vào-ra
 - Xử lý dữ liệu: thực hiện các phép toán số học hoặc phép toán logic với các dữ liệu.
 - Điều khiển rẽ nhánh
 - Kết hợp các thao tác trên.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

24

Ví dụ về thực hiện chương trình (Nội dung trong bộ nhớ và thanh ghi được biểu diễn ở hệ 16)



25

2. Hoạt động 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 khác, gọi là *chương trình con phục vụ ngắt*.

❖ Các loại ngắt:

- Ngắt do lỗi khi thực hiện chương trình, ví dụ: tràn số, chia cho 0.
- Ngắt do lỗi phần cứng, ví dụ lỗi bộ nhớ RAM
- Ngắt do môđun vào-rơ phát tín hiệu ngắt đến CPU yêu cầu trao đổi dữ liệu.
- Ngắt do bộ định thời trong chế độ đa chương trình

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

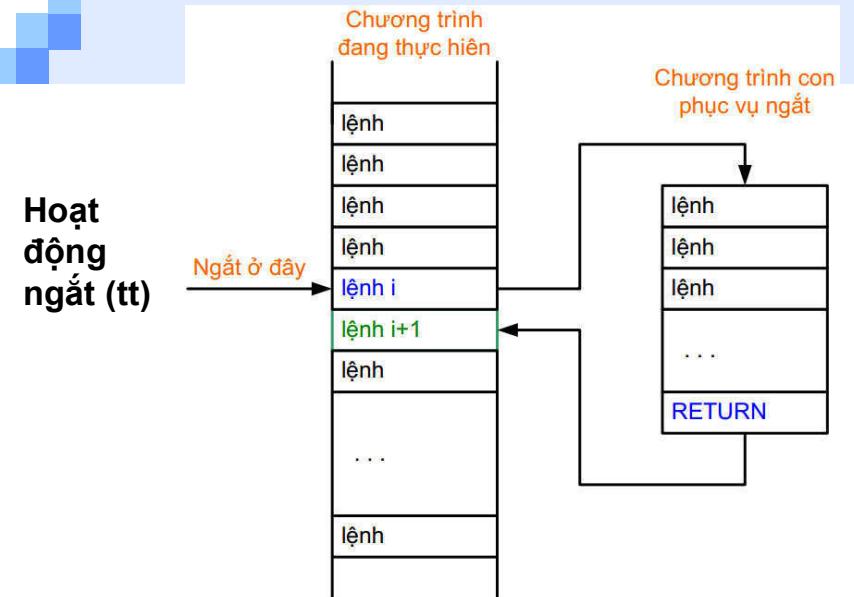
26

Hoạt động ngắt (tiếp)

- ❖ 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 chương trình đang thực hiện
 - Cắt ngắt cảnh (các thông tin liên quan đến chương trình bị ngắt)
 - Thiết lập PC trả đến chương trình con phục vụ ngắt
 - Chuyển sang thực hiện chương trình con phục vụ ngắt
 - Cuối chương trình con phục vụ ngắt, khôi phục ngắt cảnh và tiếp tục chương trình đang bị tạm dừng

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

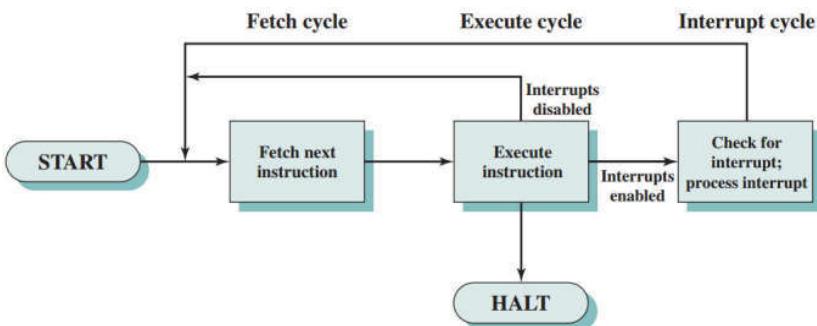
27



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

28

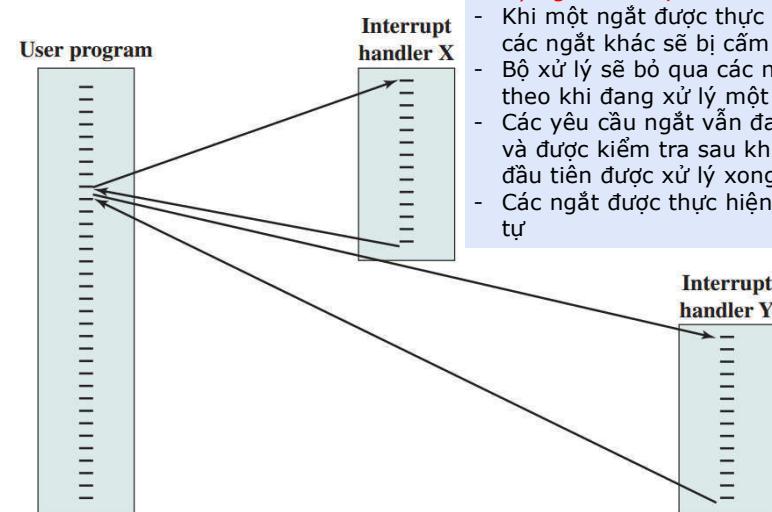
Chu trình lệnh với các ngắt



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

29

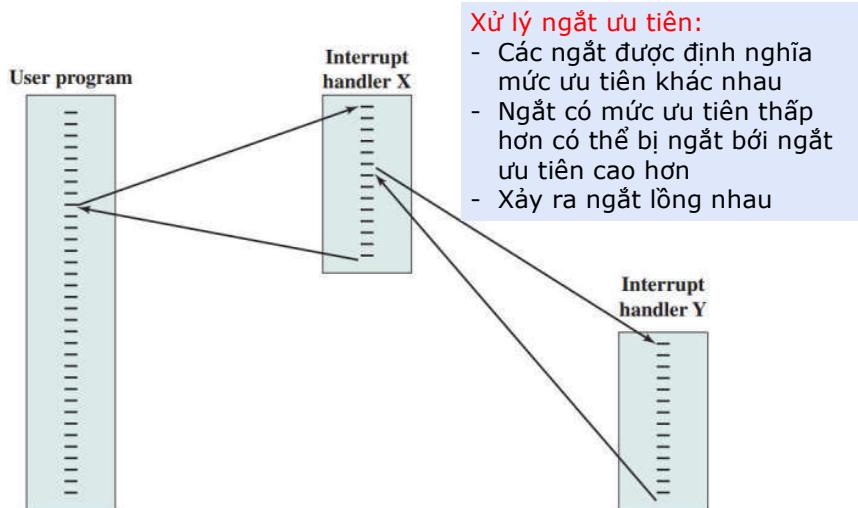
Xử lý với nhiều tín hiệu yêu cầu ngắt



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

30

Xử lý với nhiều tín hiệu yêu cầu ngắt



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

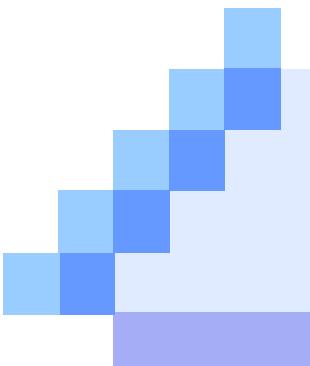
31

3. Hoạt động vào ra

- ❖ Hoạt động vào-ra: là hoạt động trao đổi dữ liệu giữa module vào-ra với bên trong máy tính.
- ❖ Các kiểu hoạt động vào-ra:
 - CPU trao đổi dữ liệu với module vào-ra
 - Module vào-ra trao đổi dữ liệu trực tiếp với bộ nhớ chính (DMA- Direct Memory Access)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

32



Bus máy tính

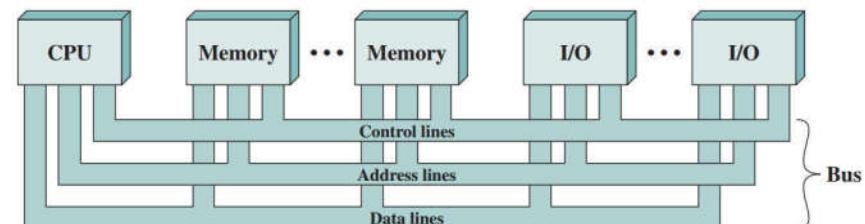


1. Luồng thông tin trong máy tính

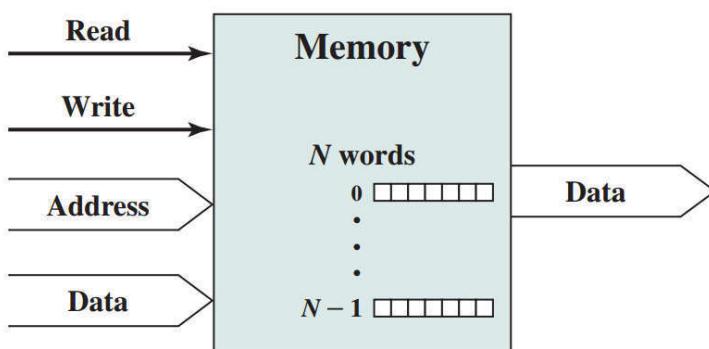
❖ Các modul trong máy tính:

- CPU
- Module nhớ
- Module vào-ra

→ Cần được kết nối với nhau

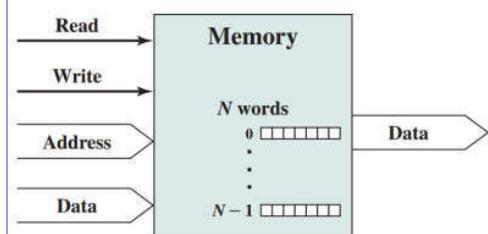


Kết nối modul nhớ



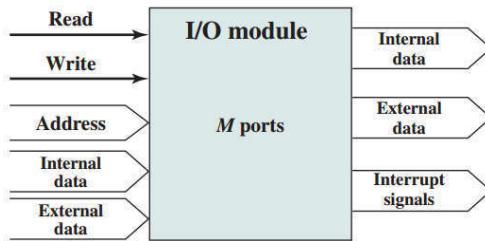
Kết nối modul nhớ

- ❖ 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 (lưu ý: 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 module vào ra

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

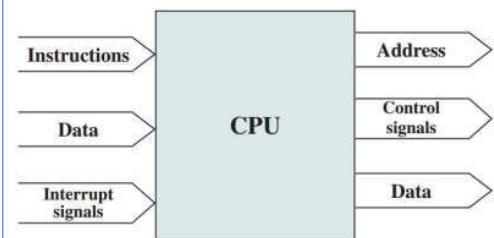


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

37

Kết nối CPU

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



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

38

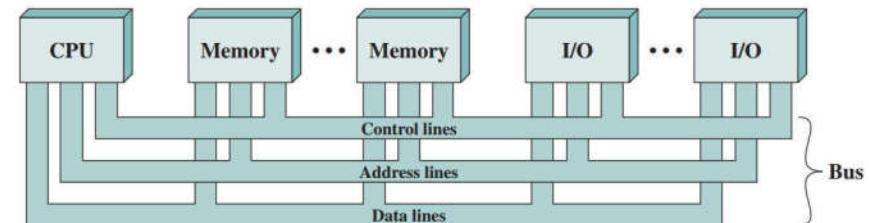
Cấu trúc Bus cơ bản

- ❖ Bus: tập hợp các đường kết nối dùng để vận chuyển thông tin giữa các module của máy tính với nhau
- ❖ Các bus chức năng:
 - Bus địa chỉ
 - Bus dữ liệu
 - Bus điều khiển
- ❖ 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)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

39

Sơ đồ cấu trúc bus cơ bản



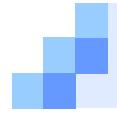
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

40



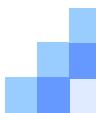
Bus địa chỉ

- ❖ **Chức năng:** vận chuyển địa chỉ để xác định ngăn nhớ hay cổng vào-ra
- ❖ **Độ rộng bus địa chỉ:** cho biết số lượng ngăn nhớ tối đa được đánh địa chỉ.
 - N bit: $A_{N-1}, A_{N-2}, \dots, A_2, A_1, A_0$
 - có thể đánh địa chỉ tối đa cho 2^N ngăn nhớ (không gian địa chỉ bộ nhớ)
- ❖ **Ví dụ:**
 - Bộ xử lý Pentium có bus địa chỉ 32 bit
 - có khả năng đánh địa chỉ cho 2^{32} bytes nhớ (4GBytes) (ngăn nhớ tổ chức theo byte)



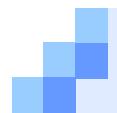
Bus dữ liệu

- **Chức năng:**
 - vận chuyển lệnh từ bộ nhớ đến CPU
 - vận chuyển dữ liệu giữa CPU, bộ nhớ, bộ đun vào-ra với nhau
- **Độ rộng bus dữ liệu:** Xác định số bit dữ liệu có thể được trao đổi đồng thời.
 - M bit: $D_{M-1}, D_{M-2}, \dots, D_2, D_1, D_0$
 - M thường là 8, 16, 32, 64, 128 bit.
- **Ví dụ:** Các bộ xử lý Pentium có bus dữ liệu 64 bit



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 ngắt
 - Các tín hiệu điều khiển bus



Một số tín hiệu điều khiển điển hình

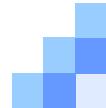
- ❖ Các tín hiệu (phát ra từ CPU) điều khiển đọc-ghi:
 - *Memory Read (MEMR)*: điều khiển đọc dữ liệu từ một ngăn nhớ có địa chỉ xác định lên bus dữ liệu.
 - *Memory Write (MEMW)*: điều khiển ghi dữ liệu có sẵn trên bus dữ liệu đến một ngăn nhớ có địa chỉ xác định.
 - *I/O Read (IOR)*: điều khiển đọc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



Một số tín hiệu điều khiển diễn hình (tiếp)

❖ Các tín hiệu điều khiển ngắn:

- *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 diễn hình (tiếp)

❖ Các tín hiệu điều khiển bus:

- *Bus Request (BRQ)* hay là *Hold*: Tín hiệu từ module điều khiển vào-ra gửi đến yêu cầu CPU chuyển nhượng quyền sử dụng bus
- *Bus Grant (BGT)* hay là *Hold Acknowledge (HLDA)*: Tín hiệu phát ra từ CPU chấp nhận chuyển nhượng quyền sử dụng bus.
- *Lock/ Unlock*: Tín hiệu *cấp/cho-phép* xin chuyển nhượng bus



Đặc điểm của cấu trúc đơn bus

- ❖ Bus hệ thống chỉ phục vụ được một yêu cầu trao đổi dữ liệu tại một thời điểm
- ❖ Bus hệ thống phải có tốc độ bằng tốc độ bus của module nhanh nhất trong hệ thống.
- ❖ Bus hệ thống phụ thuộc vào cấu trúc bus (các tín hiệu) của bộ xử lý → các module nhớ và các module vào-ra cũng phụ thuộc vào bộ xử lý.
- ❖ Khắc phục: phân cấp bus → cấu trúc đa bus



3. Phân cấp bus trong máy tính

- ❖ Tổ chức thành nhiều bus trong hệ thống máy tính
 - Cho các thành phần khác nhau:
 - Bus của bộ xử lý
 - Bus của bộ nhớ chính
 - Các bus vào-ra
 - Các bus khác nhau về tốc độ
- ❖ Bus bộ nhớ chính và các bus vào-ra không phụ thuộc vào bộ xử lý cụ thể.

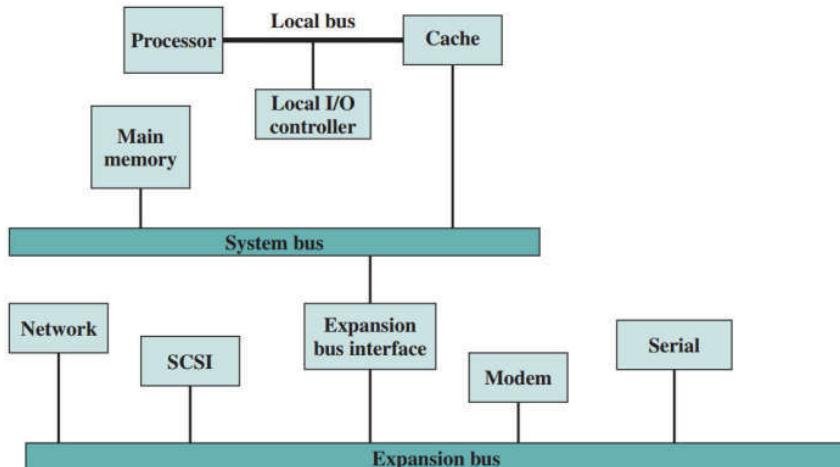
Một số bus điển hình trong máy để bàn

- ❖ Bus của bộ xử lý: có tốc độ nhanh nhất
- ❖ Bus của bộ nhớ chính (nối ghép với các module RAM)
- ❖ PCI Express bus (Peripheral Component Interconnect): nối ghép với các thiết bị ngoại vi có tốc độ trao đổi dữ liệu nhanh.
- ❖ 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

49

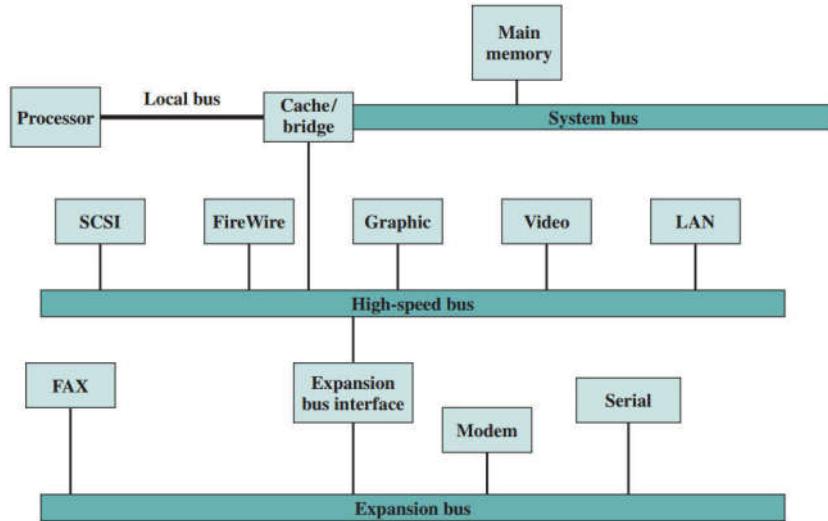
Cấu trúc bus điển hình



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

50

Cấu trúc bus hiệu năng cao



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

51

Bài tập

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

52

5. KIẾN TRÚC TẬP LỆNH

NỘI DUNG

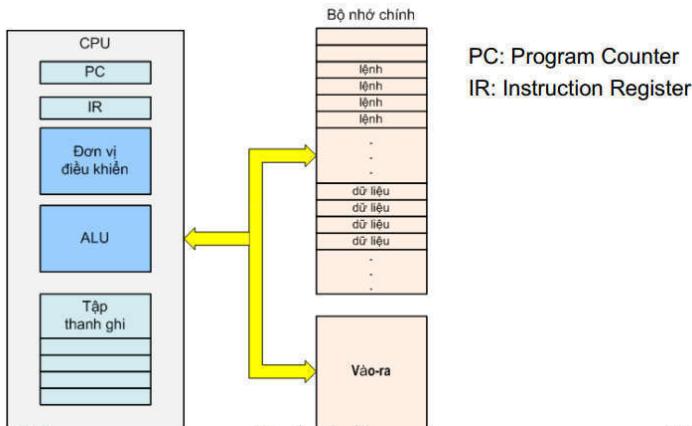
- ❖ Giới thiệu chung về kiến trúc tập lệnh
- ❖ Kiến trúc tập lệnh MIPS

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

2

1. Giới thiệu chung về kiến trúc tập lệnh

- ❖ Mô hình lập trình của máy tính



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

3

Tập thanh ghi

- ❖ Chứa các thông tin (dữ liệu, địa chỉ, trạng thái) cho hoạt động điều khiển và xử lý dữ liệu của CPU ở thời điểm hiện tại
- ❖ Được coi là mức đầu tiên của hệ thống nhớ
- ❖ Số lượng thanh ghi nhiều → tăng hiệu năng của CPU
- ❖ Có hai loại thanh ghi:
 - Thanh ghi lập trình được
 - Thanh ghi không lập trình được

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

4

Một số thanh ghi điển hình

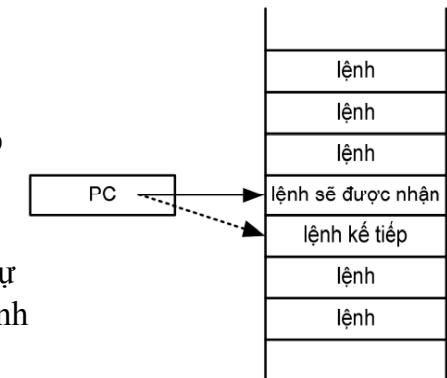
- ❖ Bộ đếm chương trình PC (Program Counter)
- ❖ Con trỏ dữ liệu DP (Data Pointer)
- ❖ Con trỏ ngăn xếp SP (Stack Pointer)
- ❖ Thanh ghi cơ sở và Thanh ghi chỉ số (Base Register & Index Register)
- ❖ Các thanh ghi dữ liệu
- ❖ Thanh ghi trạng thái

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

5

Bộ đếm chương trình

- ❖ Còn được gọi là con trỏ lệnh IP (Instruction Pointer)
- ❖ Giữ địa chỉ của lệnh tiếp theo sẽ được nhận vào
- ❖ Sau khi một lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp.
- ❖ PC tăng bao nhiêu?

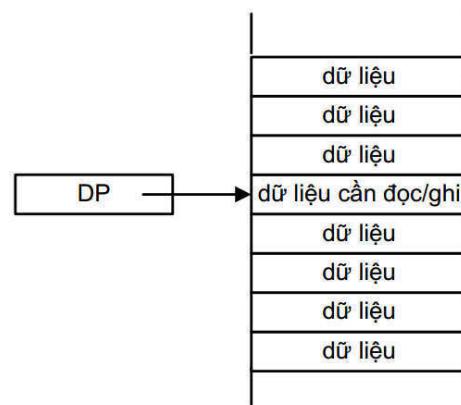


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

6

Thanh ghi con trỏ dữ liệu

- ❖ Chứa địa chỉ của ngăn nhớ dữ liệu mà CPU muốn truy nhập



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

7

Ngăn xếp (Stack)

- ❖ Ngăn xếp là vùng nhớ có cấu trúc LIFO (Last In - First Out → vào sau – ra trước)
- ❖ Ngăn xếp thường dùng để phục vụ cho chương trình con
- ❖ Đây ngăn xếp là một ngăn nhớ xác định
- ❖ Định ngăn xếp là thông tin nằm ở vị trí trên cùng trong ngăn xếp
- ❖ Định ngăn xếp có thể bị thay đổi

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

8

Con trỏ ngăn xếp

- ❖ Chứa địa chỉ của ngăn nhớ đindh ngăn xếp
- ❖ Khi cất thông tin vào một ngăn xếp
 - Nội dung của SP giảm
 - Thông tin được cất vào ngăn nhớ được trỏ bởi SP
- ❖ Khi lấy một thông tin ra khỏi ngăn xếp
 - Thông tin được đọc từ ngăn xếp được trỏ bởi SP
 - Nội dung của SP tang
- ❖ Khi ngăn xếp rỗng, SP trả vào đây

Các thanh ghi dữ liệu

- ❖ Chứa các dữ liệu tạm thời hoặc các kết quả trung gian
- ❖ Cần có nhiều thanh ghi dữ liệu
- ❖ Các thanh ghi số nguyên: 8, 16, 32, 64 bit
- ❖ Các thanh ghi số dấu phẩy động

Thanh ghi cơ sở và thanh ghi chỉ số

- ❖ Để truy nhập một ngăn nhớ có thể sử dụng hai tham số:
 - Địa chỉ cơ sở (base address)
 - Phần dịch chuyển địa chỉ (offset)
 - Địa chỉ của ngăn nhớ cần truy nhập = địa chỉ cơ sở + offset
- ❖ Có thể sử dụng các thanh ghi để quản lý các tham số này
 - Thanh ghi cơ sở: chứa địa chỉ cơ sở
 - Thanh ghi chỉ số: chứa phần dịch chuyển địa chỉ



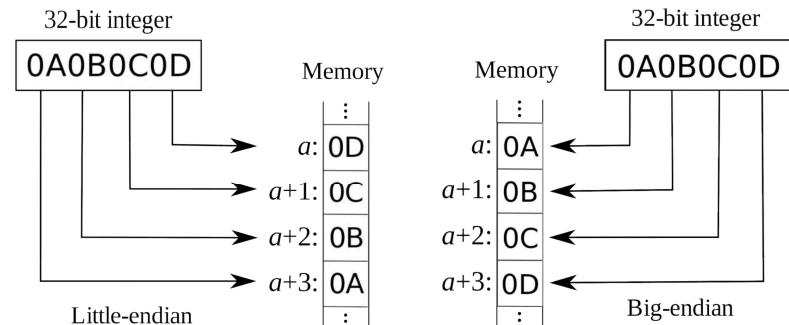
Thanh ghi trạng thái (Status Register)

- ❖ Được sử dụng trên một số kiến trúc cụ thể
- ❖ Còn gọi là thanh ghi cờ (Flag Register)
- ❖ Chứa các thông tin trạng thái của CPU
 - Các cờ phép toán: báo hiệu trạng thái của kết quả phép toán
 - Các cờ điều khiển: biểu thị trạng thái điều khiển của CPU

2. Thứ tự lưu trữ các byte trong bộ nhớ chính

- ❖ Bộ nhớ chính thường đánh địa chỉ theo byte
- ❖ Hai cách lưu trữ thông tin nhiều byte:
 - Đầu nhỏ (*Little-endian*): Byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ lớn.
 - Đầu to (*Big-endian*): Byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ lớn.

Ví dụ lưu trữ dữ liệu 32 bit



Lưu trữ của các bộ vi xử lý điển hình

- ❖ Intel x86: little-endian
- ❖ Motorola 680x0, MIPS, SunSPARC: big-endian
- ❖ Power PC, Itanium: bi-endian

3. Giới thiệu chung về tập lệnh

- ❖ Mỗi bộ xử lý có một tập lệnh xác định
- ❖ Tập lệnh thường có hàng chục đến hàng trăm lệnh
- ❖ Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định
- ❖ Các lệnh được mô tả bằng các ký hiệu gọi nhở dạng text → chính là các lệnh của hợp ngữ (assembly language)



Các thành phần của lệnh máy

Mã thao tác	Địa chỉ của các toán hạng
-------------	---------------------------

- ❖ Mã thao tác (operation code → opcode): mã hóa cho thao tác mà bộ xử lý phải thực hiện
- ❖ Địa chỉ toán hạng: chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động
- ❖ Toán hạng nguồn (source operand): dữ liệu vào của thao tác
- ❖ Toán hạng đích (destination operand): dữ liệu ra của thao tác



Các kiểu thao tác thông dụng của tập lệnh

- ❖ Các lệnh chuyển dữ liệu
- ❖ Các lệnh xử lý số học
- ❖ Các lệnh xử lý logic
- ❖ Các lệnh chuyển điều khiển (rẽ nhánh, nhảy)



Địa chỉ của toán hạng

- ❖ Toán hạng của lệnh có thể là:
 - Một giá trị cụ thể nằm ngay trong lệnh
 - Nội dung của thanh ghi
 - Nội dung của ngăn nhớ hoặc cổng vào-ra
- ❖ Phương pháp định địa chỉ (addressing modes) là cách thức địa chỉ hóa trong trường địa chỉ của lệnh để xác định nơi chứa toán hạng



Các phương pháp định địa chỉ thông dụng

- ❖ Định địa chỉ tức thì
- ❖ Định địa chỉ trực tiếp
- ❖ Định địa chỉ gián tiếp
- ❖ Định địa chỉ thanh ghi
- ❖ Định địa chỉ gián tiếp qua thanh ghi
- ❖ Định địa chỉ dịch chuyển

Địa chỉ tức thì



- ❖ Toán hạng là hằng số nằm ngay trong lệnh
 - Chỉ có thể là toán hạng nguồn
 - Ví dụ:

ADD R1, 5 ; $R1 \leftarrow R1 + 5$

- Không tham chiếu bộ nhớ
- Truy nhập toán hạng rất nhanh
- Dài giá trị của toán hạng bị hạn chế

Địa chỉ thanh ghi

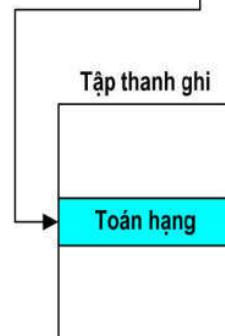
- ❖ Toán hạng *nằm trong thanh ghi* có tên được chỉ ra trong lệnh

- ❖ Ví dụ:

ADD R1, R2 ; $R1 \leftarrow R1 + R2$

- ❖ Số lượng thanh ghi ít → Trường địa chỉ toán hạng chỉ cần ít bit

- ❖ Không tham chiếu bộ nhớ
- ❖ Truy nhập toán hạng nhanh
- ❖ Tăng số lượng thanh ghi → hiệu quả hơn



Địa chỉ trực tiếp

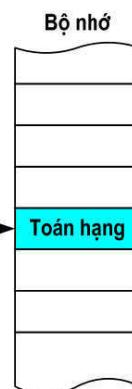


- ❖ Toán hạng là ngăn nhớ có địa chỉ được cho trực tiếp trong lệnh

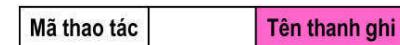
- ❖ Ví dụ:

ADD R1, A ; $R1 \leftarrow R1 + (A)$

- Cộng nội dung thanh ghi R1 với nội dung của ngăn nhớ có địa chỉ là A
- Tìm toán hạng trong bộ nhớ ở địa chỉ A
- CPU tham chiếu bộ nhớ một lần để truy nhập dữ liệu



Định địa chỉ gián tiếp qua thanh ghi

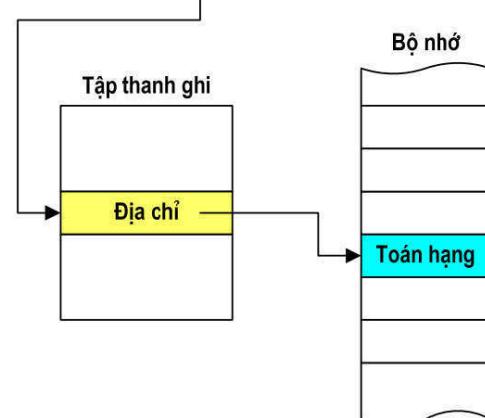


- ❖ Toán hạng *nằm ở ngăn nhớ có địa chỉ ở thanh ghi*

- ❖ Trường địa chỉ toán hạng cho biết thanh ghi đó

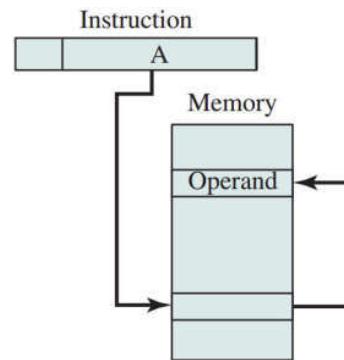
- ❖ Thanh ghi có thể là ngầm định

- ❖ Vùng nhớ có thể được tham chiếu là lớn (2^n), (với n là độ dài của thanh ghi)



Địa chỉ gián tiếp

- ❖ Địa chỉ trực tiếp: Chiều dài của trường địa chỉ thường nhỏ hơn chiều dài của từ \rightarrow giới hạn không gian địa chỉ có thể truy cập.



Địa chỉ gián tiếp:

- Địa chỉ của toán hạng được lưu trong một từ của bộ nhớ \rightarrow Mở rộng dài địa chỉ bộ nhớ có thể truy cập
- CPU tham chiếu bộ nhớ hai lần để truy nhập dữ liệu
 - Lần 1 dùng để lấy địa chỉ
 - Lần 2 dùng để truy nhập dữ liệu

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

25

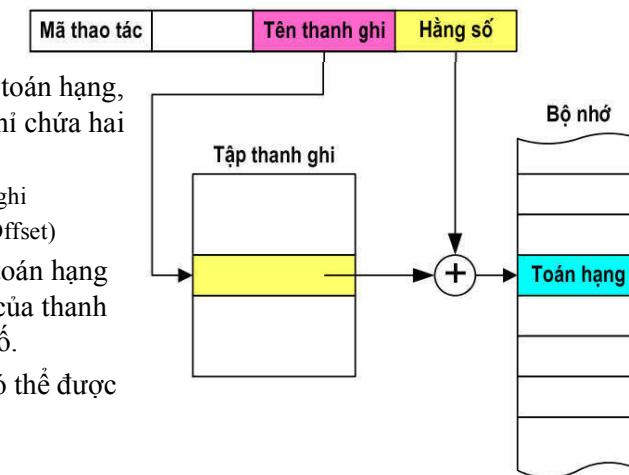
Địa chỉ dịch chuyển

- ❖ Để xác định toán hạng, trường địa chỉ chứa hai thành phần:

- Tên thanh ghi
- Hằng số (Offset)

- ❖ Địa chỉ của toán hạng = Nội dung của thanh ghi + hằng số.

- ❖ Thanh ghi có thể được ngầm định



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

26

Số lượng địa chỉ toán hạng trong lệnh

Ba địa chỉ toán hạng:

- 2 toán hạng nguồn, 1 toán hạng đích
- $c = a + b$
- Từ lệnh dài vì phải mã hoá địa chỉ cho cả ba toán hạng
- Được sử dụng trên các bộ xử lý tiên tiến

Hai địa chỉ toán hạng (phổ biến)

- Một toán hạng vừa là nguồn vừa là toán hạng đích; Toán hạng còn lại là toán hạng nguồn
- $a = a + b$
- Giá trị cũ của một toán hạng nguồn bị mất vì phải chia kết quả
- Rút gọn độ dài từ lệnh

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

27

Số lượng địa chỉ toán hạng trong lệnh (tt)

Một địa chỉ toán hạng:

- Một toán hạng được chỉ ra trong lệnh
- Một toán hạng là ngầm định \rightarrow thường là thanh ghi (thanh chứa –accumulator)
- Được sử dụng trên các máy ở các thế hệ trước



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

28



Số lượng địa chỉ toán hạng trong lệnh (tt)

❖ 0 địa chỉ toán hạng (Không thông dụng)

- Các toán hạng đều được ngầm định
- Sử dụng Stack
- Ví dụ:
 - push a
 - push b
 - add
 - pop c
$$(c = a + b)$$



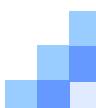
4. CISC và RISC

❖ CISC: Complex Instruction Set Computer:

- Máy tính với tập lệnh phức tạp
- Các bộ xử lý truyền thống: Intel x86, Motorola 680x0

❖ RISC: Reduced Instruction Set Computer:

- Máy tính với tập lệnh thu gọn
- SunSPARC, Power PC, MIPS, ARM ...
- RISC đối nghịch với CISC
- Kiến trúc tập lệnh tiên tiến



Các đặc trưng của RISC

- ❖ Số lượng lệnh ít
- ❖ Hầu hết các lệnh truy nhập toán hạng ở các thanh ghi
- ❖ Truy nhập bộ nhớ bằng các lệnh LOAD/STORE
- ❖ Thời gian thực hiện lệnh là một chu kỳ máy
- ❖ Các lệnh có độ dài cố định (32 bit)
- ❖ Số lượng dạng lệnh ít (≤ 4)
- ❖ CPU có tập thanh ghi lớn
- ❖ Có ít phương pháp định địa chỉ toán hạng (≤ 4)
- ❖ Hỗ trợ các thao tác của ngôn ngữ bậc cao



MIPS

- ❖ MIPS- Microprocessor without Interlocked Pipeline Stages
- ❖ Được phát triển ở đại học Stanford, sau đó được thương mại hóa bởi Công ty MIPS Technologies (www.mips.com)
- ❖ Kiến trúc RISC
- ❖ Chiếm thị phần lớn trong các sản phẩm nhúng
- ❖ Diễn hình cho nhiều kiến trúc tập lệnh hiện đại



MIPS - Thanh ghi

- ❖ Là đơn vị lưu trữ data duy nhất trong CPU
- ❖ Trong kiến trúc MIPS:
 - Có tổng cộng 32 thanh ghi đánh số từ \$0 → \$31
 - Càng ít càng dễ quản lý, tính toán càng nhanh
 - Có thể truy xuất thanh ghi qua tên của nó
 - Mỗi thanh ghi có kích thước cố định 32 bit
 - Bị giới hạn bởi khả năng tính toán của chip xử lý
 - Kích thước toán hạng trong các câu lệnh MIPS bị giới hạn ở 32 bit, nhóm 32 bit gọi là từ (word)
 - Trong kiến trúc MIPS không tồn tại khái niệm biến, thay vào đó là thanh ghi toán hạng



Bài tập

1. Định địa chỉ tức thì?
2. Định địa chỉ trực tiếp?
3. Định địa chỉ gián tiếp?
4. Định địa chỉ thanh ghi?
5. Định địa chỉ gián tiếp qua thanh ghi?
6. Định địa chỉ dịch chuyển?



Bài tập

2. Cho các giá trị bộ nhớ sau và một cơ chế định địa chỉ với một thanh ghi tích lũy (accumulator register). Các giá trị nào sẽ được load vào thanh ghi tích lũy tương ứng với các lệnh sau
 - Từ 20 chứa giá trị 40.
 - Từ 30 chứa giá trị 50.
 - Từ 40 chứa giá trị 60.
 - Từ 50 chứa giá trị 70.
 - a. LOAD IMMEDIATE 20
 - b. LOAD DIRECT 20
 - c. LOAD INDIRECT 20
 - d. LOAD IMMEDIATE 30
 - e. LOAD DIRECT 30
 - f. LOAD INDIRECT 30



Bài tập

3. Một trường địa chỉ trong một lệnh có giá trị thập phân là 14. Toán hạng tương ứng của nó nằm ở đâu trong các trường hợp sau:
 - a) Định địa chỉ tức thời?
 - b) Định địa chỉ trực tiếp
 - c) Định địa chỉ gián tiếp
 - d) Định địa chỉ thanh ghi
 - e) Định địa chỉ gián tiếp qua thanh ghi



Bài tập

- ❖ Giải sử một vi xử lý 16 bit và đoạn mã sau trong bộ nhớ chính, bắt đầu tại vị trí 200:

200	Load to AC	Mode
201	500	
202	Next instruction	

- ❖ Phần đầu của từ đầu tiên cho biết lệnh này load một giá trị vào thanh ghi tích lũy (AC). Trường Mode cho biết cách định địa chỉ, trong trường hợp thích hợp là thanh ghi nguồn R1 (có giá trị 400). Thanh ghi cơ sở có giá trị 100. Giá trị 500 tại vị trí 201 có thể là phần của tính toán địa chỉ. Giải sử vị trí 399 chứa giá trị 999, 400 chứa giá trị 1000..... Xác định địa chỉ thực và toán hạng được load vào ứng với các trường hợp sau của Mode:

- | | |
|--------------|------------------------|
| a. Trực tiếp | b. Tức thì |
| c. Gián tiếp | d. Dịch chuyển |
| e. Thanh ghi | f. Gián tiếp thanh ghi |

6. CẤU TRÚC BỘ XỬ LÝ

NỘI DUNG

- ❖ Tổ chức của bộ xử lý
- ❖ Thiết kế đơn vị điều khiển
- ❖ Kỹ thuật đường ống lệnh
- ❖ Ví dụ thiết kế bộ xử lý theo kiến trúc MIPS*
- ❖ Họ vi xử lý x86
- ❖ Vi xử lý ARM

Huỳnh Tuấn Anh - Đại học Nha Trang

2

Tổ chức của CPU

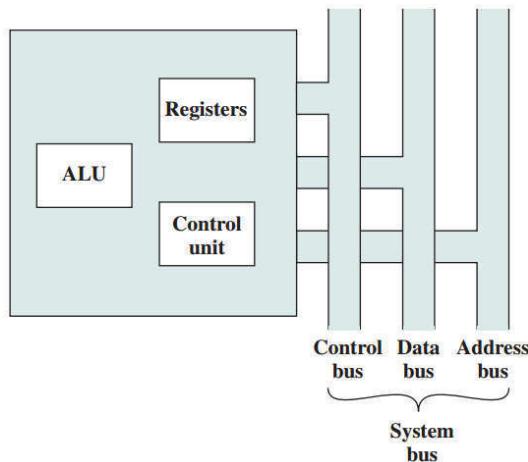
1. Cấu trúc cơ bản của CPU

- ❖ Nhiệm vụ của CPU:
 - Nhận lệnh (Fetch Instruction): CPU đọc lệnh từ bộ nhớ.
 - Giải mã lệnh (Decode Instruction): xác định thao tác mà lệnh yêu cầu.
 - Nhận dữ liệu (Fetch Data): nhận dữ liệu từ bộ nhớ hoặc các cổng vào-ra.
 - Xử lý dữ liệu (Process Data): thực hiện phép toán số học hay phép toán logic với các dữ liệu.
 - Ghi dữ liệu (Write Data): ghi dữ liệu ra bộ nhớ hay cổng vào-ra

Huỳnh Tuấn Anh - Đại học Nha Trang

4

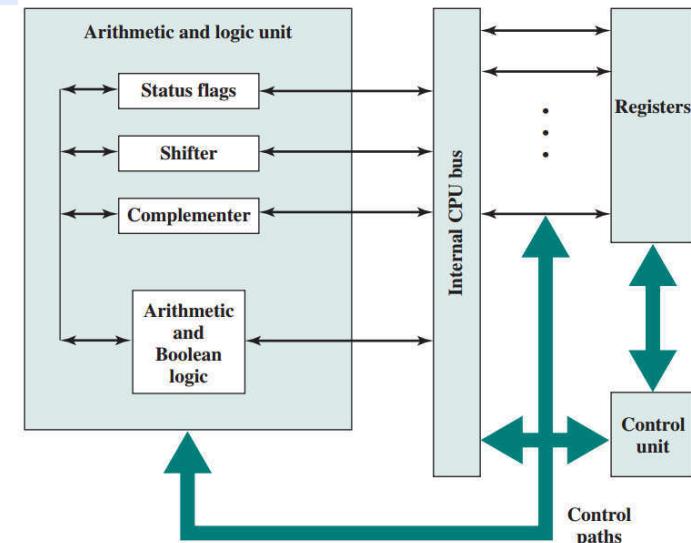
Sơ đồ cấu trúc cơ bản của CPU



Huỳnh Tuấn Anh - Đại học Nha Trang

5

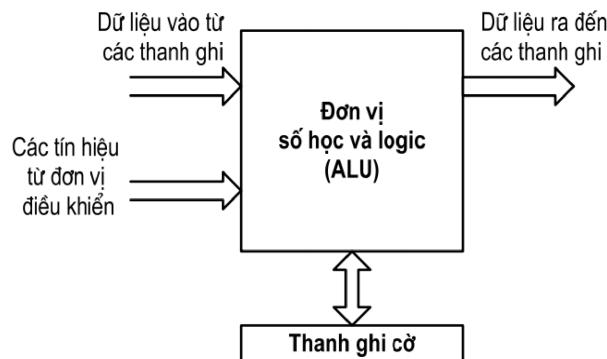
Cấu trúc bên trong của CPU



6

Đơn vị số học và logic - ALU

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



Huỳnh Tuấn Anh - Đại học Nha Trang

7

3. Đơn vị điều khiển

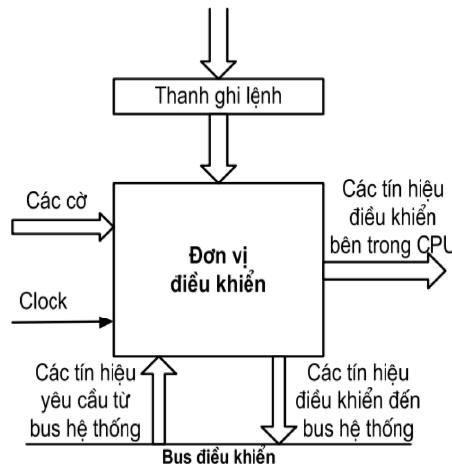
- Chức năng
 - Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh
 - Tăng nội dung của PC để trỏ sang lệnh kế tiếp.
 - Giải mã lệnh đã được nhận để xác định thao tác mà lệnh yêu cầu
 - Phát ra các tín hiệu điều khiển thực hiện lệnh
 - Nhận các tín hiệu yêu cầu từ bus hệ thống và đáp ứng với các yêu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

Huỳnh Tuấn Anh - Đại học Nha Trang

8



Mô hình kết nối đơn vị điều khiển



Huỳnh Tuấn Anh - Đại học Nha Trang

9



Các tín hiệu đưa đến đơn vị điều khiển

- ❖ Clock: tín hiệu nhịp từ mạch tạo dao động bên ngoài.
- ❖ Mã lệnh từ thanh ghi lệnh đưa đến để giải mã
- ❖ Các cờ từ thanh ghi cờ cho biết trạng thái của CPU
- ❖ Các tín hiệu yêu cầu từ bus điều khiển

Huỳnh Tuấn Anh - Đại học Nha Trang

10



Các tín hiệu phát ra từ đơn vị điều khiển

- ❖ Các tín hiệu điều khiển bên trong CPU:
 - Điều khiển các thanh ghi
 - Điều khiển ALU
- ❖ Các tín hiệu điều khiển bên ngoài CPU:
 - Điều khiển bộ nhớ
 - Điều khiển các mô-đun vào-ra

Huỳnh Tuấn Anh - Đại học Nha Trang

11



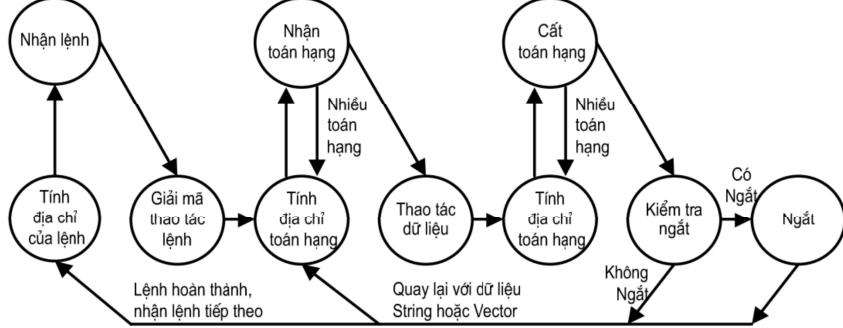
4. Hoạt động của chu trình lệnh

- ❖ Chu trình lệnh
 - Nhận lệnh
 - Giải mã lệnh
 - Nhận toán hạng
 - Thực hiện toán hạng
 - Cắt toán hạng
 - Ngắt

Huỳnh Tuấn Anh - Đại học Nha Trang

12

Giản đồ trạng thái chu trình lệnh

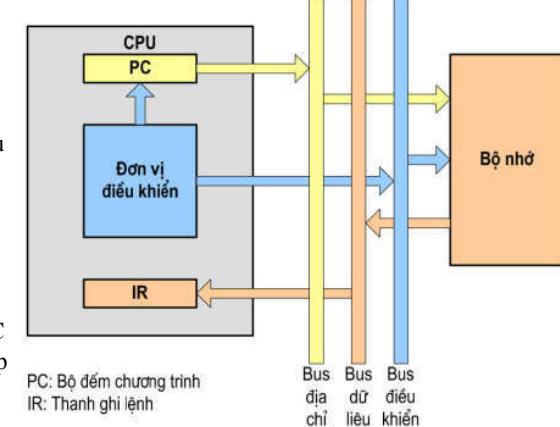


Huỳnh Tuấn Anh - Đại học Nha Trang

13

Nhận lệnh

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



Huỳnh Tuấn Anh - Đại học Nha Trang

14

Giải mã lệnh

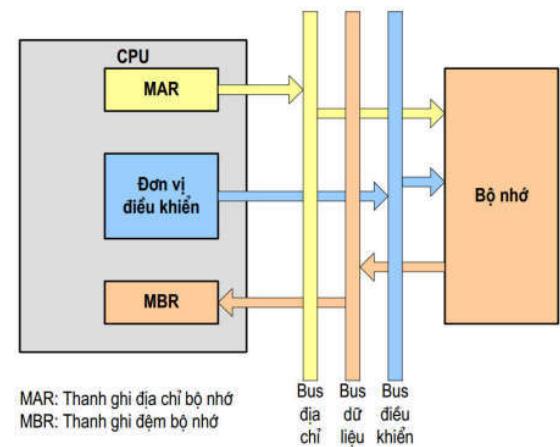
- Lệnh từ thanh ghi lệnh IR được đưa đến đơn vị điều khiển
- Đơn vị điều khiển tiến hành giải mã lệnh để xác định thao tác phải thực hiện
- Giải mã lệnh xảy ra bên trong CPU

Huỳnh Tuấn Anh - Đại học Nha Trang

15

Nhận dữ liệu từ bộ nhớ

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



Huỳnh Tuấn Anh - Đại học Nha Trang

16

Thực hiện lệnh

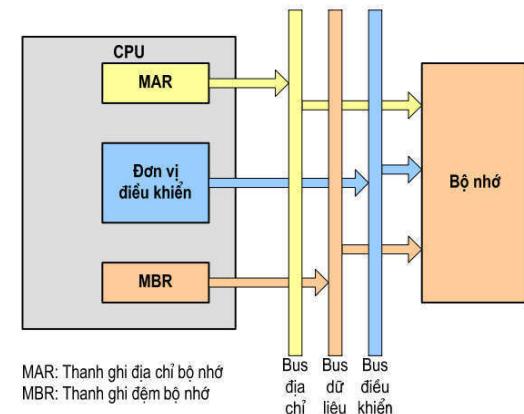
- ❖ Có nhiều dạng tùy thuộc vào lệnh
- ❖ Có thể là:
 - Đọc/Ghi bộ nhớ
 - Vào/Ra
 - Chuyển giữa các thanh ghi
 - Thao tác số học/logic
 - Chuyển điều khiển (rẽ nhánh)
 -

Huỳnh Tuấn Anh - Đại học Nha Trang

17

Ghi toán hạng

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



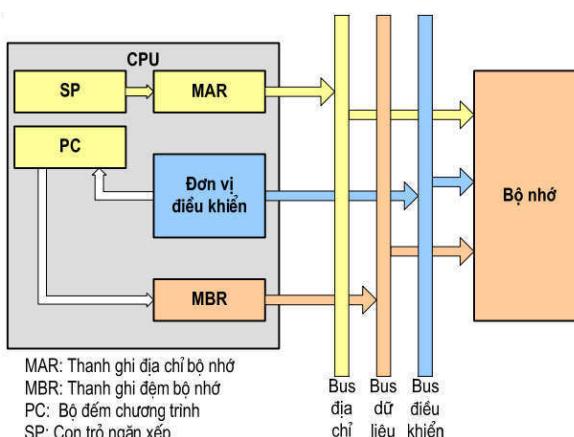
MAR: Thanh ghi địa chỉ bộ nhớ
MBR: Thanh ghi đệm bộ nhớ

Huỳnh Tuấn Anh - Đại học Nha Trang

18

Ngắt

- ❖ Nội dung của bộ đếm chương trình PC (địa chỉ trả về sau khi ngắt) được đưa ra bus dữ liệu
- ❖ CPU đưa địa chỉ (thường được lấy từ con trỏ ngăn xếp SP) ra bus địa chỉ
- ❖ CPU phát tín hiệu điều khiển ghi bộ nhớ



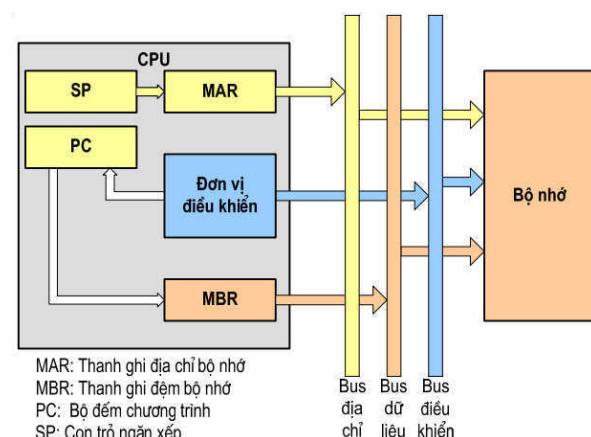
MAR: Thanh ghi địa chỉ bộ nhớ
MBR: Thanh ghi đệm bộ nhớ
PC: Bộ đếm chương trình
SP: Con trỏ ngăn xếp

Huỳnh Tuấn Anh - Đại học Nha Trang

19

Ngắt

- ❖ Địa chỉ trả về trên bus dữ liệu được ghi ra vị trí xác định (ở ngăn xếp)
- ❖ Địa chỉ lệnh đầu tiên của chương trình con điều khiển ngắt được nạp vào PC



MAR: Thanh ghi địa chỉ bộ nhớ
MBR: Thanh ghi đệm bộ nhớ
PC: Bộ đếm chương trình
SP: Con trỏ ngăn xếp

Huỳnh Tuấn Anh - Đại học Nha Trang

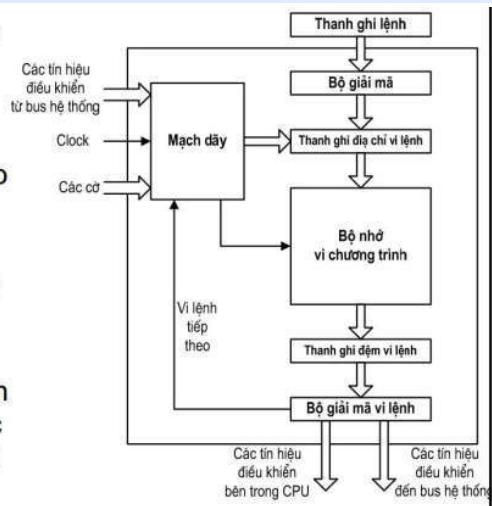
20

Thiết kế đơn vị điều khiển

- ❖ Đơn vị điều khiển vi chương trình (Microprogrammed Control Unit)
- ❖ Đơn vị điều khiển nối kết cứng (Hardwired Control Unit)

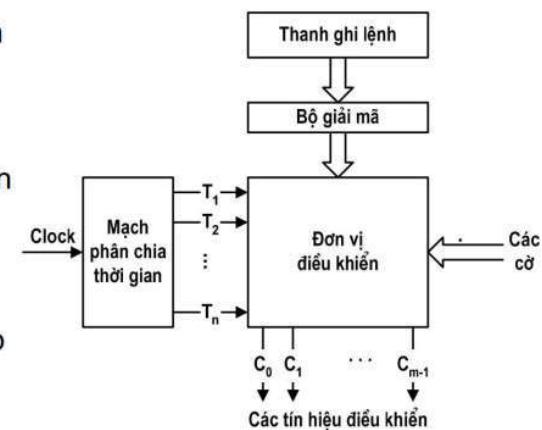
1. Đơn vị điều khiển vi chương trình

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



2. Đơn vị điều khiển nối kết cứng

- Sử dụng mạch cứng để giải mã và tạo các tín hiệu điều khiển thực hiện lệnh
- Tốc độ nhanh
- Đơn vị điều khiển phức tạp



Kỹ thuật đường ống lệnh

Huỳnh Tuấn Anh - Đại học Nha Trang

25

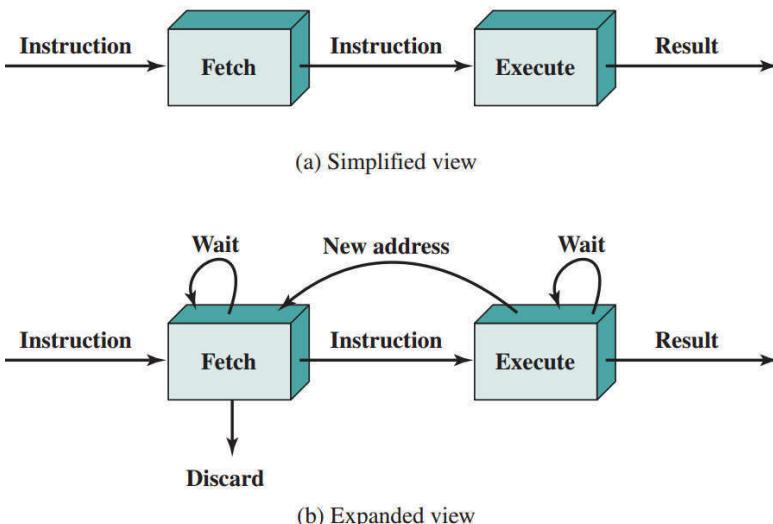
Instruction Pipelining

- ❖ Kỹ thuật đường ống lệnh (Instruction Pipelining): Chia chương trình thành các công đoạn và cho phép thực hiện gối lên nhau (như dây chuyền lắp ráp)
- ❖ Chặng hạn có 6 công đoạn
 - Nhận lệnh (Fetch Instruction - FI)
 - Giải mã lệnh (Decode Instruction - DI)
 - Tính địa chỉ toán hạng (Calculate Operand Address-CO)
 - Nhận toán hạng (Fetch Operands - FO)
 - Thực hiện lệnh (Execute Instruction - EI)
 - Ghi toán hạng (Write Operands - WO)

Huỳnh Tuấn Anh - Đại học Nha Trang

26

2 giai đoạn của đường ống lệnh



Huỳnh Tuấn Anh - Đại học Nha Trang

27

Biểu đồ thời gian của đường ống lệnh



Huỳnh Tuấn Anh - Đại học Nha Trang

28



Các Hazard (trở ngại) của đường ống lệnh

- ❖ Hazard: Tình huống ngăn cản bắt đầu của lệnh tiếp theo ở chu kỳ tiếp theo.
 - Hazard cấu trúc: do tài nguyên được yêu cầu đang bận.
 - Hazard dữ liệu: cần phải đợi để lệnh trước hoàn thành việc đọc/ghi dữ liệu
 - Hazard điều khiển: do rẽ nhánh gây ra

Huỳnh Tuấn Anh - Đại học Nha Trang

29



Hazard về cấu trúc

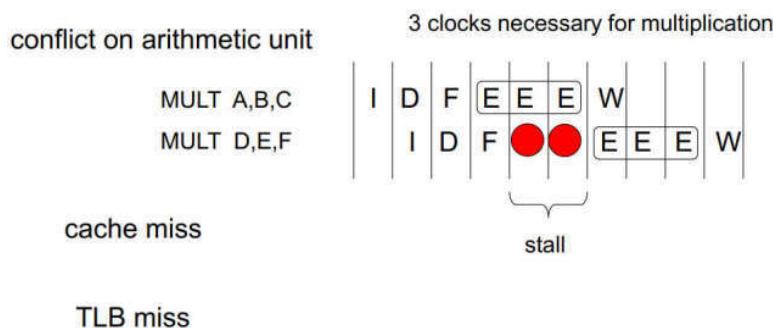
- ❖ Khắc phục:
 - Thêm tài nguyên để tránh xung đột
 - Làm trễ
 - Ví dụ
 - Bus dữ liệu: Truyền lệnh và dữ liệu
→ Bus lệnh riêng, bus dữ liệu riêng (cache lệnh và cache dữ liệu)

Huỳnh Tuấn Anh - Đại học Nha Trang

30



Ví dụ Hazard về cấu trúc



Huỳnh Tuấn Anh - Đại học Nha Trang

31



Hazard về dữ liệu

- ❖ Nguyên nhân: lệnh sau sử dụng dữ liệu kết quả của lệnh trước
- ❖ Các dạng:
 - RAW (Read After Write)
 - WAR (Write After Read)
 - WAW (Write After Write)

Huỳnh Tuấn Anh - Đại học Nha Trang

32

Sự phụ thuộc về dữ liệu

RAW	ADD A,B,C ADD E,A,D	Write-A must be earlier than Read-A
-----	------------------------	-------------------------------------

WAR	ADD A,B,C ADD B,D,E	Read-B must be earlier than Write-B
-----	------------------------	-------------------------------------

WAW	ADD A,B,C ADD A,D,E	First Write-A must be earlier Than second Write-A
-----	------------------------	---

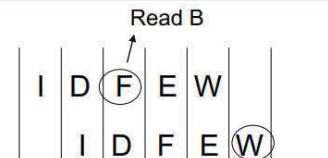
Huỳnh Tuấn Anh - Đại học Nha Trang

33

WAR and WAW

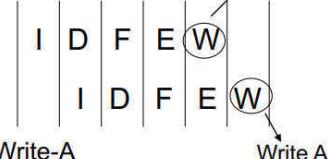
WAR	ADD A,B,C ADD B,D,E
-----	------------------------

Read-B is earlier than Write-B



WAW	ADD A,B,C ADD A,D,E
-----	------------------------

first Write-A is earlier than second Write-A



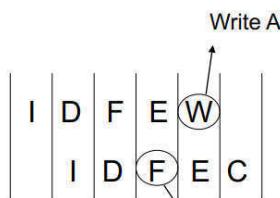
no conflict at in-order pipeline
conflict at out-of-order pipeline

Huỳnh Tuấn Anh - Đại học Nha Trang

34

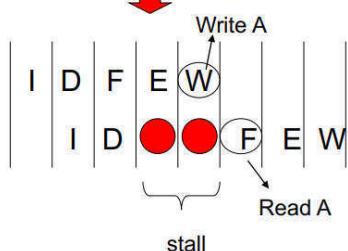
RAW

ADD A,B,C



Write-A must be earlier

Than Read-A



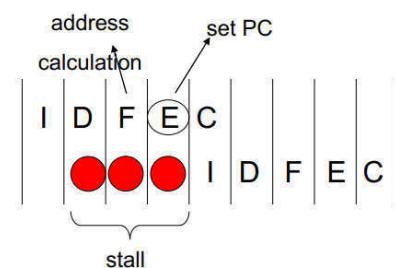
Huỳnh Tuấn Anh - Đại học Nha Trang

35

Hazard điều khiển

Wait for branch

BEQ A, B, Label
LOAD C, X
....

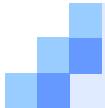


Label: LOAD C, Y

next instruction to a branch instruction cannot be fetched until branch condition defined and PC updated

Huỳnh Tuấn Anh - Đại học Nha Trang

36

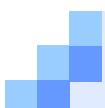
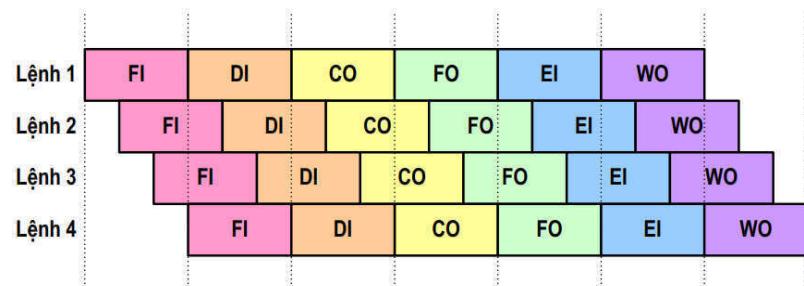


Các kiến trúc song song mức lệnh

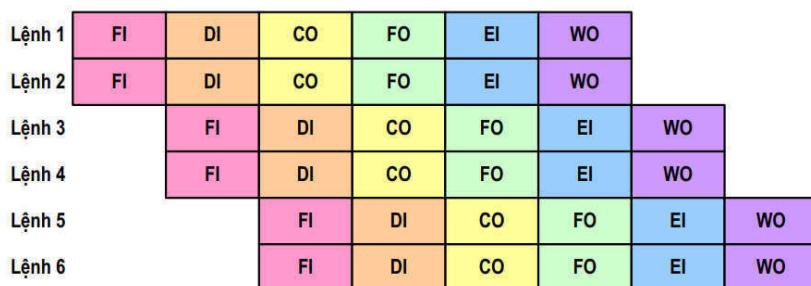
- ❖ Siêu đường ống (Superpipeline & Hyperpipeline)
- ❖ Siêu vô hướng (Superscalar)



Superpipeline



Superscalar



7. BỘ NHỚ

- ❖ Phân cấp bộ nhớ
- ❖ Bộ nhớ chính
- ❖ Bộ nhớ cache
- ❖ Bộ nhớ ngoài
- ❖ Bộ nhớ ảo

PHÂN CẤP BỘ NHỚ MÁY TÍNH

1. Các đặc trưng của bộ nhớ máy tính

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



Các đặc trưng của hệ thống nhớ (tiếp)

❖ Đơn vị truyền

- Từ nhớ
- Khối nhớ

❖ Phương pháp truy nhập

- Truy nhập tuần tự (băng từ)
- Truy nhập trực tiếp (các loại đĩa)
- Truy nhập ngẫu nhiên (bộ nhớ bán dẫn)
- Truy nhập liên kết (cache)



Các đặc trưng của hệ thống 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



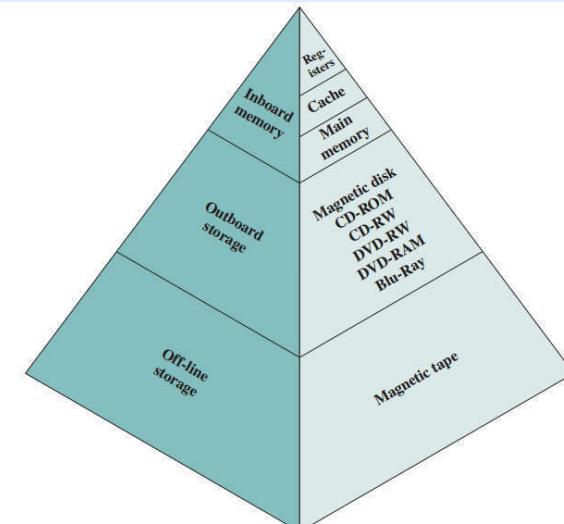
Các đặc trưng của hệ thống nhớ (tiếp)

❖ Các đặc tính vật lý

- Khả biến / Không khả biến (volatile / nonvolatile)
- Xoá được / không xoá được
- Tổ chức



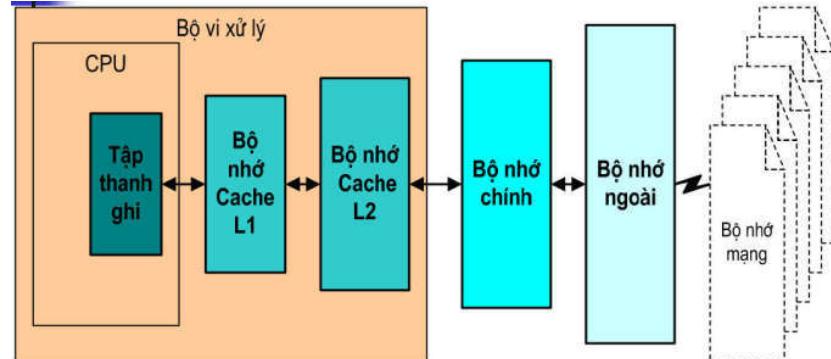
Phân cấp bộ nhớ



Công nghệ bộ nhớ

- ❖ Static RAM (SRAM)
 - 0.5ns – 2.5ns, \$2000 – \$5000 per GB
- ❖ Dynamic RAM (DRAM)
 - 50ns – 70ns, \$20 – \$75 per GB
 - SSD (Solid State Drive)
- ❖ Ổ đĩa từ
 - 5ms – 20ms, \$0.20 – \$2 per GB

Ví dụ hệ thống nhớ thông dụng



Từ trái sang phải:

- dung lượng tăng dần
- tốc độ giảm dần
- giá thành/1bit giảm dần

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

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)	Bộ nhớ chỉ đọc	bằng tia cực tím, cả chip		
Erasable PROM (EPROM)	Bộ nhớ hầu như chỉ đọc	bằng điện, mức từng byte	Băng điện	Không khả biến
Electrically Erasable PROM (EEPROM)	Bộ nhớ chỉ đọc	bằng điện, từng khối		
Flash memory	Bộ nhớ đọc-ghi	bằng điện, từng byte	Băng điện	Khả biến
Random Access Memory (RAM)	Bộ nhớ đọc-ghi	bằng điện, mức từng byte	Băng điện	Khả biến

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

13

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
 - Ví chương trình

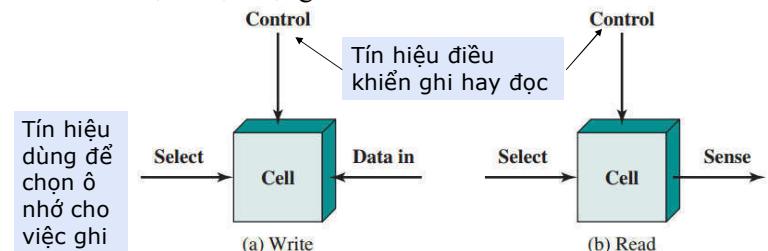


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

15

Cấu tạo bộ nhớ chính bán dẫn (SEMICONDUCTOR MAIN MEMORY)

- ❖ Thành phần cơ bản của bộ nhớ bán dẫn là cell nhớ. Ô nhớ có các đặc trưng sau:
 - Có hai trạng thái ổn định (biểu diễn giá trị nhị phân 0, 1)
 - Có khả năng ghi được (ít nhất một lần) để thiết lập trạng thái.
 - Có thể đọc được trạng thái của nó.



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

14

Các kiểu ROM

- ❖ ROM mặt nạ:
 - thông tin được ghi khi sản xuất
 - rất đắt
- ❖ PROM (Programmable ROM)
 - Cần thiết bị chuyên dụng để ghi bằng chương trình → Chỉ được ghi một lần
- ❖ EPROM (Erasable PROM)
 - Cần thiết bị chuyên dụng để ghi bằng chương trình → Ghi được nhiều lần
 - Trước khi ghi lại, xóa bằng tia cực tím

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

16



Các kiểu ROM (tiếp)

❖ EEPROM (Electrically Erasable PROM)

- Có thể ghi theo từng byte
- Xóa bằng điện

❖ Flash memory (Bộ nhớ cực nhanh)

- Ghi theo khối
- Xóa bằng điện

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

17



RAM (Random Access Memory)

❖ Bộ nhớ đọc-ghi (Read/Write Memory)

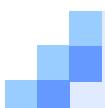
❖ Khả biến

❖ Lưu trữ thông tin tạm thời

❖ Có hai loại: SRAM và DRAM (Static and Dynamic)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

18



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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

19



DRAM (Dynamic) – RAM động

❖ Các bit được lưu trữ trên tụ điện

- Tụ có khuynh hướng phóng điện → cần phải có mạch làm tươi một cách định kỳ để duy trì dữ liệu lưu trữ

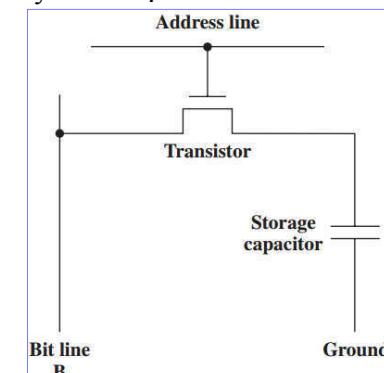
❖ Cấu trúc đơn giản

❖ Dung lượng lớn

❖ Tốc độ chậm hơn

❖ Rẻ tiền hơn

❖ Dùng làm bộ nhớ chính



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

20

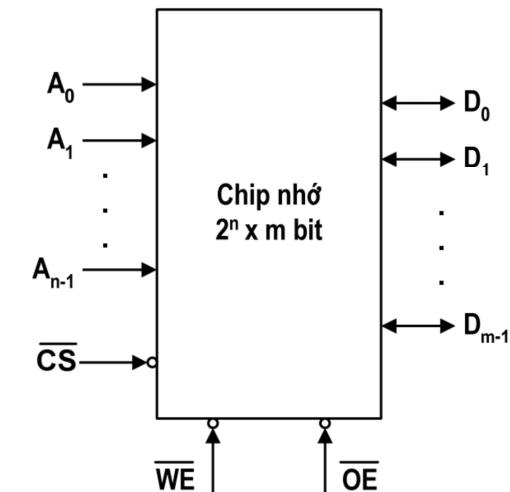
Các DRAM tiên tiến

- ❖ Enhanced DRAM
- ❖ Cache DRAM
- ❖ Synchronous DRAM (SDRAM): làm việc được đồng bộ bởi xung clock
- ❖ DDR-SDRAM (Double Data Rate SDRAM)
- ❖ Rambus DRAM (RDRAM)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

21

Tổ chức của chip nhớ



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

22

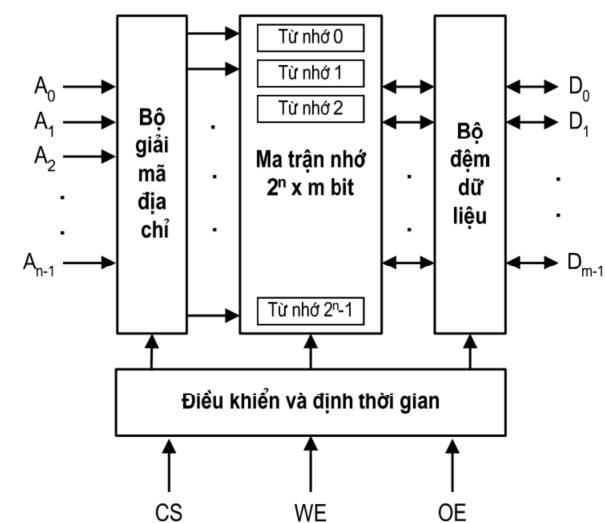
Các tín hiệu của chip nhớ

- ❖ Các đường địa chỉ: A_{n-1} ÷ A₀ → có 2ⁿ từ nhớ
 - ❖ Các đường dữ liệu: D_{m-1} ÷ D₀ → độ dài từ nhớ = m bit
 - ❖ Dung lượng chip nhớ = 2ⁿ x m bit
 - ❖ Các đường điều khiển
 - Tín hiệu chọn chip CS (Chip Select)
 - Tín hiệu điều khiển đọc OE (Output Enable)
 - Tín hiệu điều khiển ghi WE (Write Enable)
- (Các tín hiệu điều khiển thường tích cực với mức 0)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

23

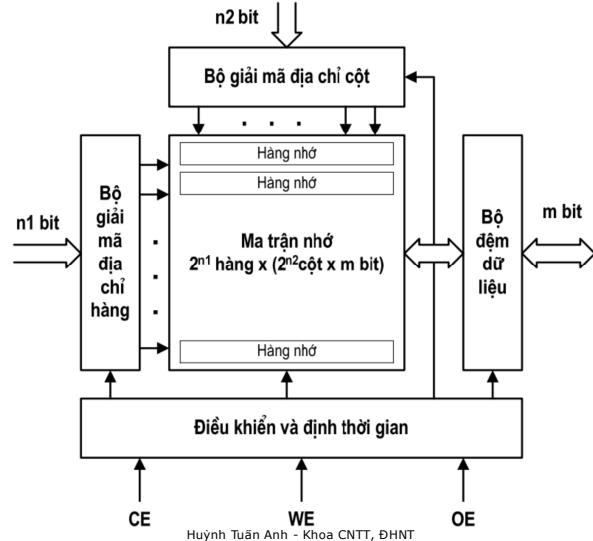
Tổ chức bộ nhớ một chiều



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

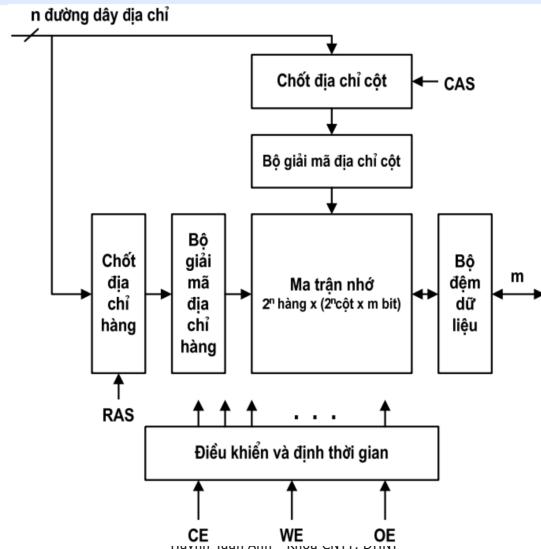
24

Tổ chức bộ nhớ hai chiều



25

Tổ chức của DRAM



27

Tổ chức bộ nhớ hai chiều

- ❖ Có n đường địa chỉ: $n = n_1 + n_2$
 - 2^{n_1} hàng,
 - mỗi hàng có 2^{n_2} từ nhớ
- ❖ Có m đường dữ liệu
 - mỗi từ nhớ có độ dài m -bit
- ❖ Dung lượng của chip nhớ:
 - $2^{n_1} \times (2^{n_2} \times m)]$ bit = $(2^{n_1+n_2} \times m)$ bit = $(2^n \times m)$ bit.
- ❖ Hoạt động giải mã địa chỉ:
 - Bước 1: bộ giải mã hàng chọn 1 trong 2^{n_1} hàng.
 - Bước 2: bộ giải mã cột chọn 1 trong 2^{n_2} từ nhớ (cột) của hàng đã được chọn.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

26

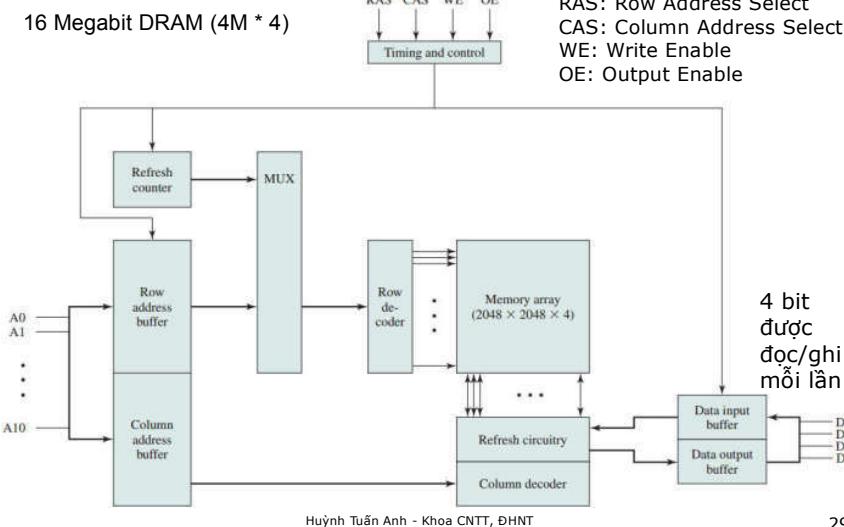
Tổ chức của DRAM

- ❖ Dùng n đường địa chỉ dồn 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 = $2^n \times 2^n \times m$ bit = $2^{2n} \times m$ bit

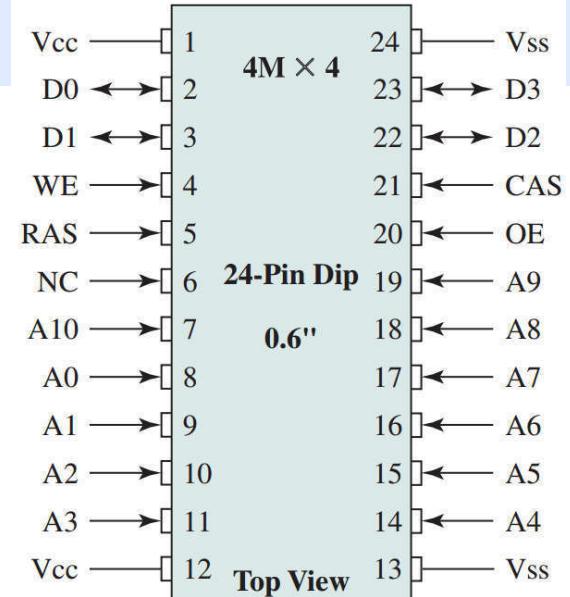
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

28

Tổ chức logic của chip nhớ



Đóng gói chip nhớ
16 Megabit DRAM
($4M \times 4$)



Thiết kế module nhớ bán dẫn

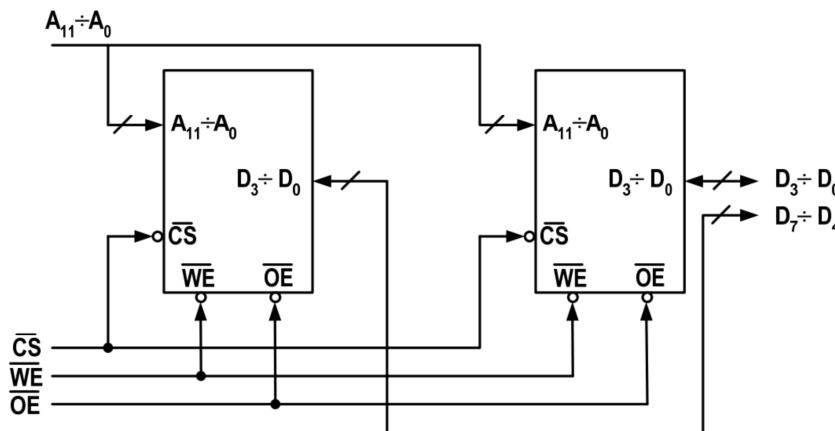
- ❖ Dung lượng chip nhớ $2^n \times m$ bit
- ❖ Cần thiết kế để tăng dung lượng:
 - Thiết kế tăng độ dài từ nhớ
 - Thiết kế tăng số lượng từ nhớ
 - Thiết kế kết hợp

Tăng độ dài từ nhớ

- ❖ Ví dụ 1:
 - 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



Ví dụ tăng độ dài từ nhớ



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

33

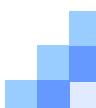


Bài toán tăng độ dài từ nhớ tổng quát

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

34



Tăng số lượng từ nhớ

VD2:

- Cho chip nhớ SRAM 4K x 8 bit
- Thiết kế mô-đun nhớ 8K x 8 bit

Giải

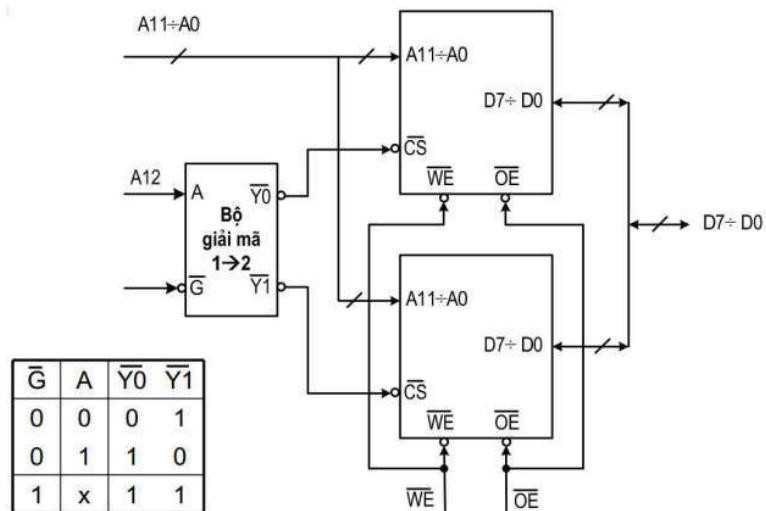
- Dung lượng chip nhớ = $2^{12} \times 8$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu
- Dung lượng mô-đun nhớ = $2^{13} \times 8$ bit
 - 13 chân địa chỉ
 - 8 chân dữ liệu

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

35



Tăng số lượng từ nhớ



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

36



Bài tập

1. Tăng số lượng từ gấp 4 lần:

- Cho chip nhớ SRAM 4K x 8 bit
- Thiết kế mô-đun nhớ 16K x 8 bit

2. Tăng số lượng từ gấp 8 lần

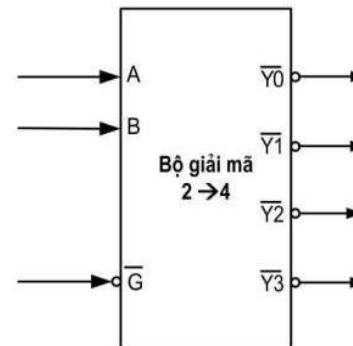
- Cho chip nhớ SRAM 4K x 8 bit
- Thiết kế mô-đun nhớ 32K x 8 bit

3. Thiết kế kết hợp

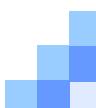
- Cho chip nhớ SRAM 4K x 4 bit
- Thiết kế mô-đun nhớ 8K x 8 bit



Bộ giải mã 2→4



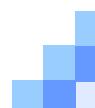
\bar{G}	B	A	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1



Bài tập

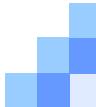
❖ Tăng số lượng từ nhớ gấp 3 lần

❖ Tăng số lượng từ nhớ gấp 5, 6, 7 lần



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 tuỳ thuộc vào hệ điều hành

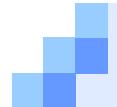


Tổ chức bộ nhớ đan xen (interleaved memory)

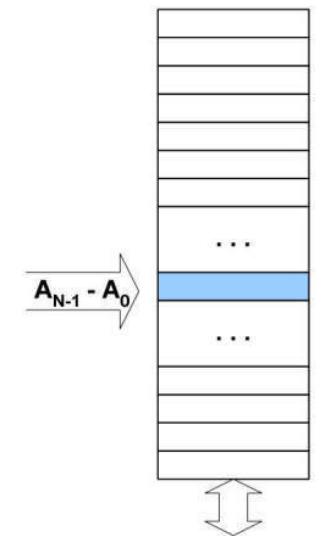
- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ: $m = 8, 16, 32, 64, 128 \dots$ bit
- Các ngăn nhớ được tổ chức theo byte
→ tổ chức bộ nhớ vật lý khác nhau

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

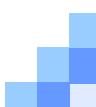
41



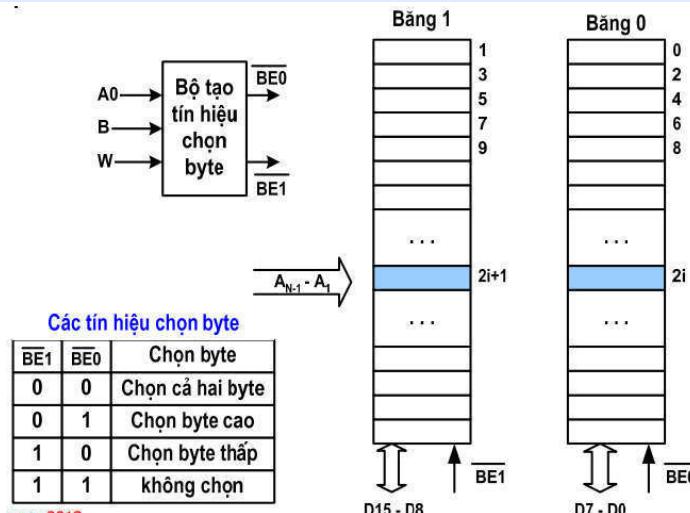
$m=8\text{bit} \rightarrow$ một băng nhớ tuyến tính



42

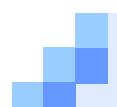


$m=16\text{bit} \rightarrow$ hai băng nhớ đan xen

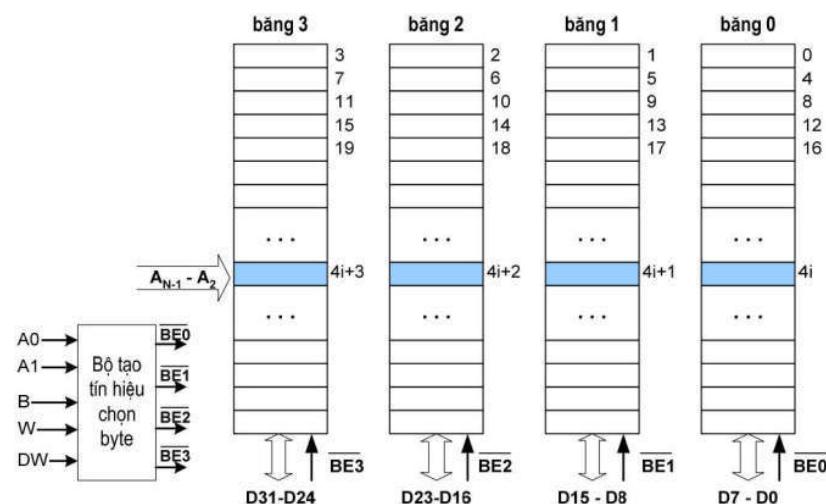


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

43



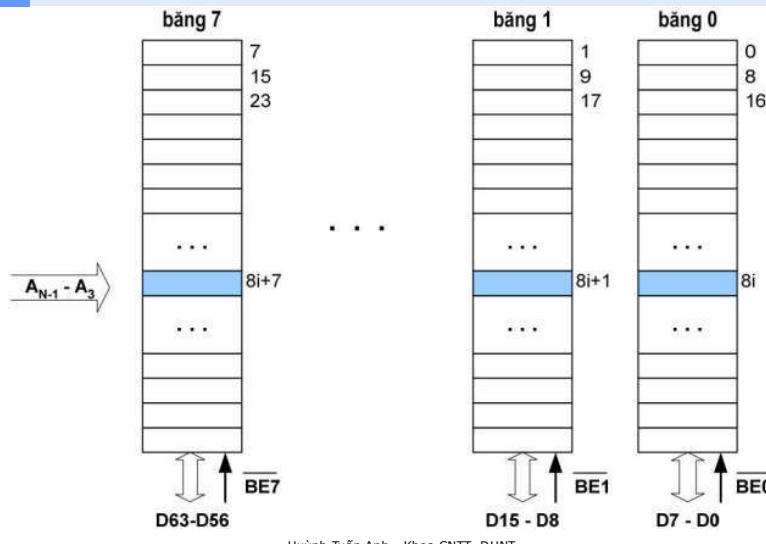
$m = 32\text{bit} \rightarrow$ 4 băng nhớ đan xen



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

44

m = 64 bit 8 băng nhớ đan xen

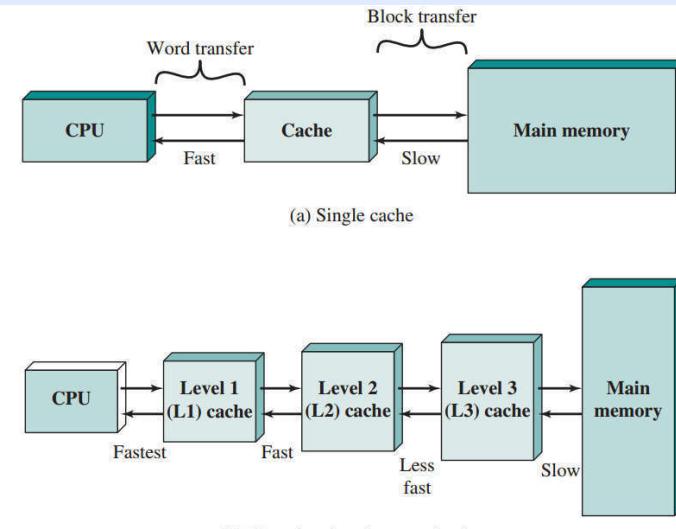


Bộ nhớ cache

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

Cache và bộ nhớ chính

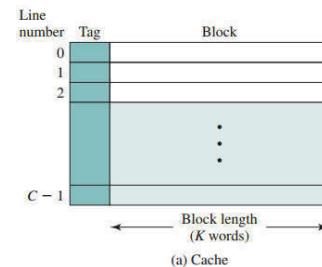




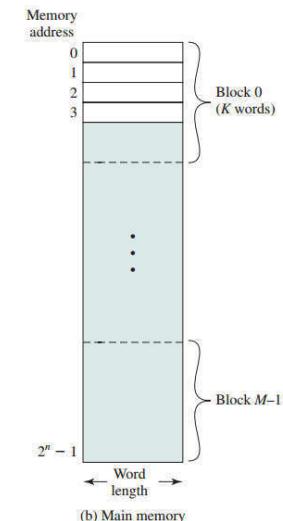
Ví dụ về thao tác của cache

- ❖ 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



(a) Cache



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

- ❖ Bộ nhớ chính có $2N$ 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 = 8, 16, 32, 64, 128 byte
- ❖

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
- ❖ 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

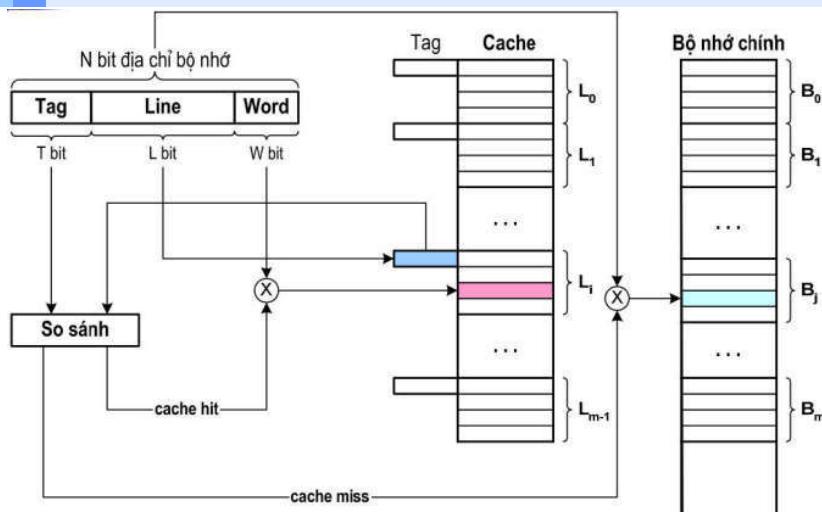
❖ Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:

- $B_0 \rightarrow L_0$
- $B_1 \rightarrow L_1$
- ...
- $B_{m-1} \rightarrow L_{m-1}$
- $B_m \rightarrow L_0$
- $B_{m+1} \rightarrow L_1$
- ...

❖ Tổng quát

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

Minh họa ánh xạ trực tiếp



Đặc điểm của ánh xạ trực tiếp

❖ Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:

- Trường Word gồm W bit xác định một từ nhớ trong Block hay Line:
 2^W = kích thước của Block hay Line
- Trường Line gồm L bit xác định một trong số các Line trong cache:
 2^L = số Line trong cache = m
- Trường Tag gồm T bit:
 $T = N - (W+L)$
- Bộ so sánh đơn giản
- Xác suất cache hit thấp

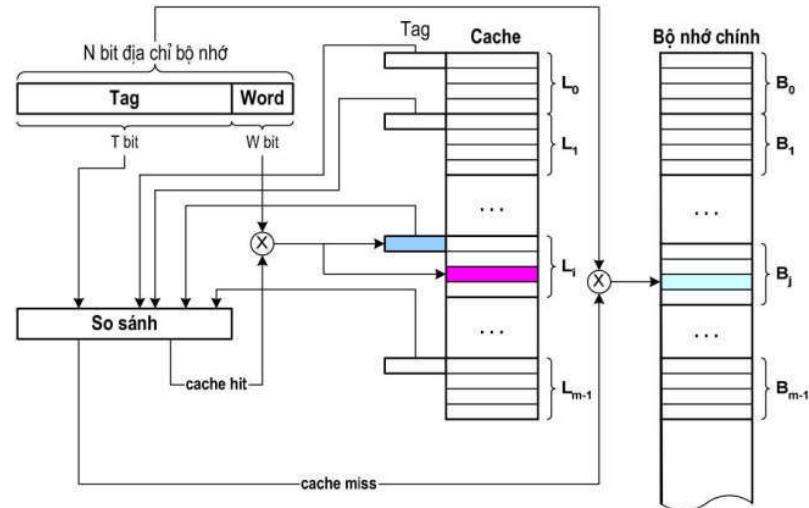
Ánh xạ liên kết toàn phần

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

57

Minh họa ánh xạ liên kết toàn phần



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

58

Đặc điểm của ánh xạ liên kết toàn phần

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

59

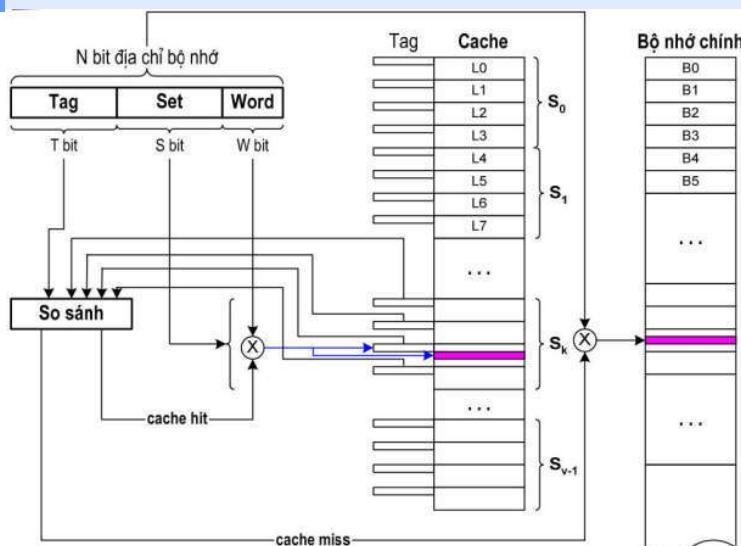
Ánh xạ liên kết tập hợp

- ❖ Cache được chia thành các Tập (Set)
- ❖ Mỗi một Set chứa một số *Line*
- ❖ Ví dụ:
 - 4 Line/Set \rightarrow 4-way associative mapping
- ❖ Ánh xạ theo nguyên tắc sau
 - $B_0 \rightarrow S_0$
 - $B_1 \rightarrow S_1$
 - $B_2 \rightarrow S_2$
 - ...

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

60

Minh họa ánh xạ liên kết tập hợp



61

Đặc điểm của ánh xạ liên kết tập hợp

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

62

Ví dụ về ánh xạ địa chỉ

- ❖ Không gian địa chỉ bộ nhớ chính = 4GB
- ❖ Dung lượng bộ nhớ cache là 256KB
- ❖ 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

63

Với ánh xạ trực tiếp

- ❖ Bộ nhớ chính = 4GB = 2^{32} byte $\rightarrow N = 32$ bit
- ❖ Cache = 256 KB = 2^{18} byte
- ❖ Line = 32 byte = 2^5 byte $\rightarrow W = 5$ bit
- ❖ Số Line trong cache = $2^{18} / 2^5 = 2^{13}$ Line
 $\rightarrow L = 13$ bit
- ❖ $T = 32 - (13 + 5) = 14$ bit

Tag	Line	Word
14 bit	13 bit	5 bit

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

64



Với ánh xạ liên kết toàn phần

- ❖ Bộ nhớ chính = $4\text{GB} = 2^{32}$ byte $\rightarrow N = 32$ bit
- ❖ $Line = 32$ byte $= 2^5$ byte $\rightarrow W = 5$ bit
- ❖ Số bit của trường Tag sẽ là: $T = 32 - 5 = 27$ bit

Tag	Word
27 bit	5 bit



Với ánh xạ liên kết tập hợp 4 đường

- ❖ Bộ nhớ chính = $4\text{GB} = 2^{32}$ byte $\rightarrow N = 32$ bit
- ❖ $Line = 32$ byte $= 2^5$ byte $\rightarrow W = 5$ 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
- $\rightarrow S = 11$ bit
- ❖ Số bit của trường Tag sẽ là: $T = 32 - (11 + 5) = 16$ bit

Tag	Set	Word
16 bit	11 bit	5 bit



Bài tập

- ❖ Giả thiết rằng máy tính có 128KB cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4-line. Cache có tất cả là 1024 Set từ S0 đến S1023. Địa chỉ bộ nhớ chính là 32-bit và đánh địa chỉ cho từng byte.
- a) Tính số bit cho các trường địa chỉ khi truy nhập cache ?
- b) Xác định byte nhớ có địa chỉ $003D02AF_{(16)}$ được ánh xạ vào Set nào của cache?



3. Thuật giải thay thế (1): Ánh xạ trực tiếp

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

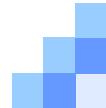


Thuật giải thay thế (2): Ánh xạ liên kết

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

69

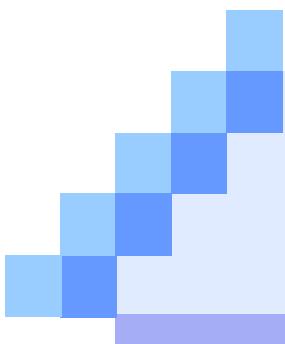


4. Phương pháp ghi dữ liệu khi cache hit

- ❖ 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

70



Bộ nhớ ngoài

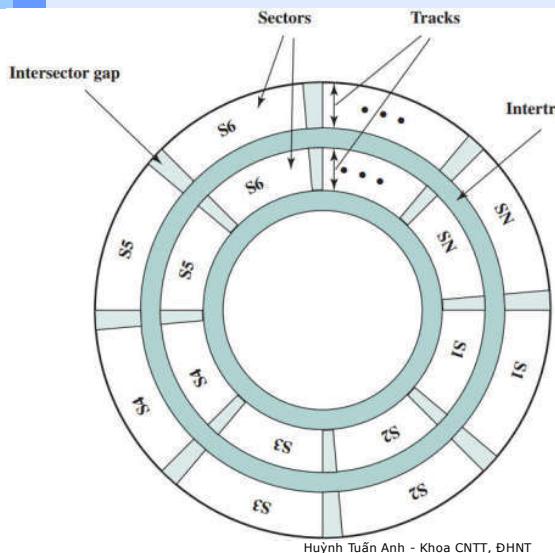


- ❖ Các kiểu bộ nhớ ngoài
 - Băng từ
 - Đĩa từ
 - Đĩa quang
 - SSD (Solid State Drives)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

72

Đĩa từ



Track: Những vòng tròn đồng tâm có cùng độ rộng

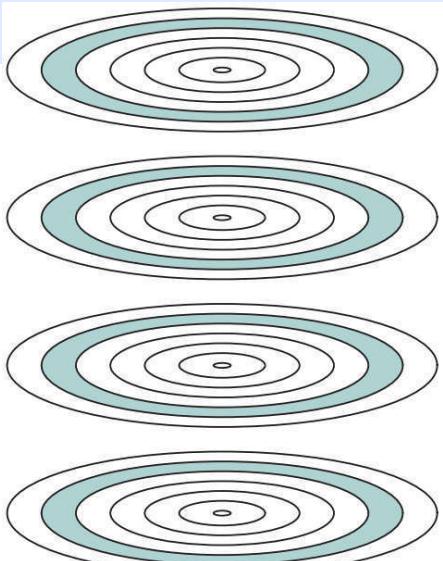
Sector: Mỗi track được chia thành các sector có độ dài biến đổi hay cố định. Thông thường chiều dài cố định được sử dụng với giá trị sector size = 512 byte

73

Cylinder

Cylinder: Tập hợp của tất cả các rãnh (track) ở cùng vị trí tương đối trên các đĩa.

VD: Tất cả các **rãnh đậm** của hình bên thuộc về một **cylinder**
- Tất cả các **đầu đọc (head)** của đĩa cứng sẽ đọc cùng một **cylinder** tại cùng một thời điểm



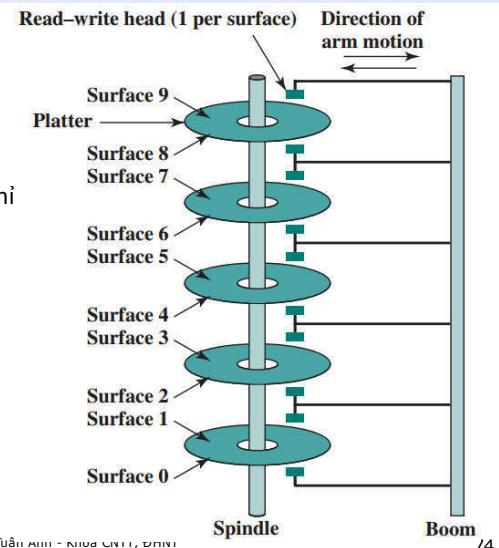
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

75

Đĩa cứng

Double sided: Lớp phủ từ tính được phủ cả hai mặt của mỗi đĩa

Single sided: Lớp phủ từ tính chỉ được ở một mặt của mỗi đĩa



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

/4

RAID

- ❖ Redundant Array of Inexpensive Disks
- ❖ Hệ thống nhớ dung lượng lớn

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

76

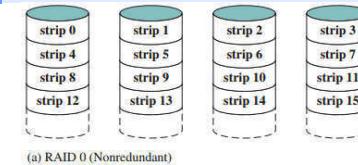
Đặc điểm của RAID

- ❖ Tập các 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)
- ❖ Có thể sử dụng dung lượng dư thừa để lưu trữ các thông tin kiểm tra chẵn lẻ, cho phép khôi phục lại thông tin trong trường hợp đĩa bị hỏng → an toàn thông tin.
- ❖ 7 loại phổ biến (RAID 0 – 6)

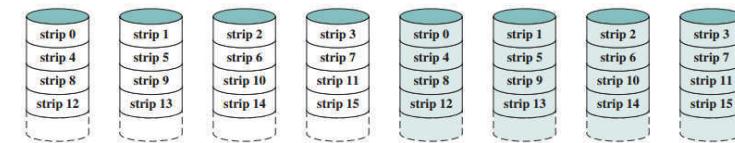
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

77

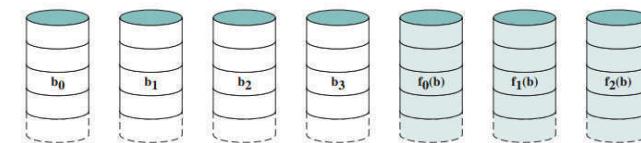
RAID 0, 1, 2



(a) RAID 0 (Nonredundant)



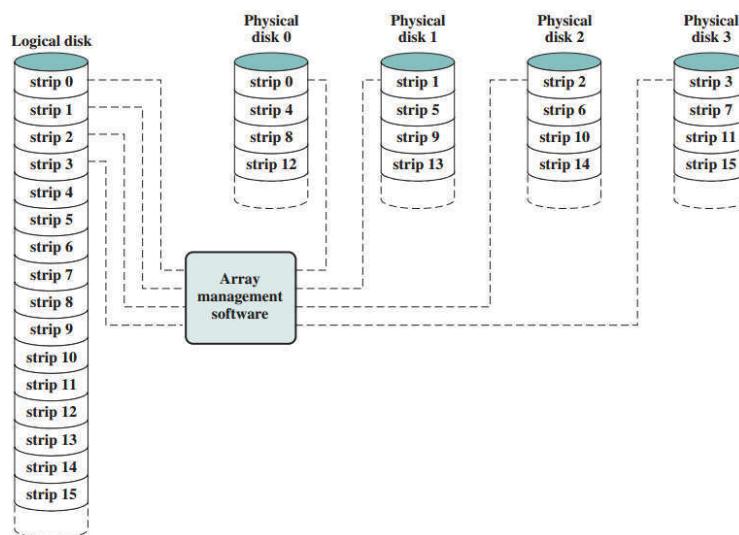
(b) RAID 1 (Mirrored)



(c) RAID 2 (Redundancy through Hamming code)

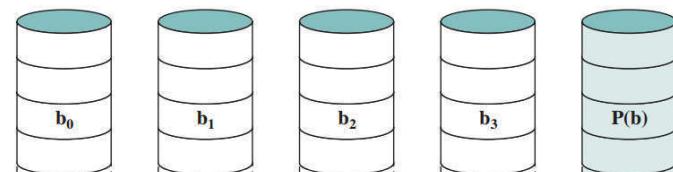
78

Ánh xạ dữ liệu đối với RAID 0

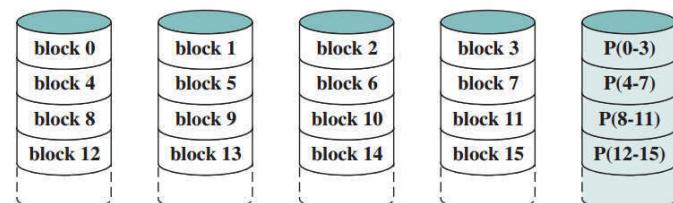


79

RAID 3 và 4



(d) RAID 3 (Bit-interleaved parity)

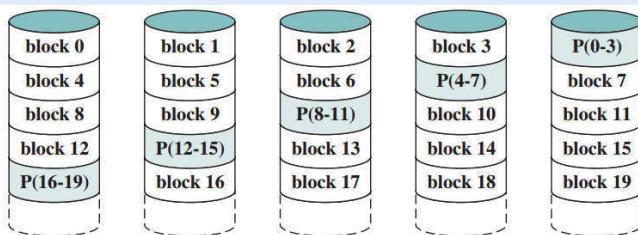


(e) RAID 4 (Block-level parity)

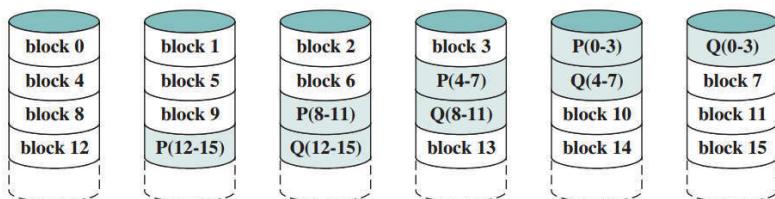
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

80

RAID 5, 6



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



(g) RAID 6 (Dual redundancy)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

81

Đĩa quang

- ❖ CD-ROM (Compact Disk ROM)
- ❖ CD-R (Recordable CD)
- ❖ CD-RW (Rewriteable CD)
- ❖ Dung lượng thông dụng 650MB
- ❖ Ổ đĩa CD:
 - Ổ CD-ROM
 - Ổ CD-Writer: Ghi một phiên hoặc ghi nhiều phiên
 - Ổ CD-RW
- ❖ Tốc độ đọc cơ sở 150KByte/s.
- ❖ Tốc độ bội, ví dụ: 48x, 52x,...

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

82

Đĩa quang (tt)

- ❖ DVD
 - Digital Video Disk: chỉ dùng trên ổ đĩa xem video
 - Digital Versatile Disk: ổ trên máy tính
 - Ghi một hoặc hai mặt
 - Một hoặc hai lớp trên một mặt
 - Thông dụng: 4,7GB/lớp

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

83

SSD (Solid State Drives)

- ❖ *solid state*: Ám chỉ đến các mạch điện + bán dẫn
- ❖ SSD: là thiết bị nhớ ché tạo từ các thành phần solid state dùng để thay thế HDD
 - Flash drive
 - SSD

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

84



Flash drive

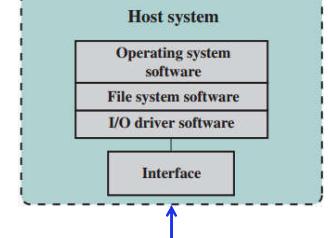
- ❖ Bộ nhớ bán dẫn (flash memory)
- ❖ Tốc độ nhanh
- ❖ Các dạng:
 - Ổ nhớ kết nối qua cổng USB
 - Thẻ nhớ
 - Ổ SSD (Solid State Drive): kết nối nhiều chip nhớ flash và cho phép truy cập song song



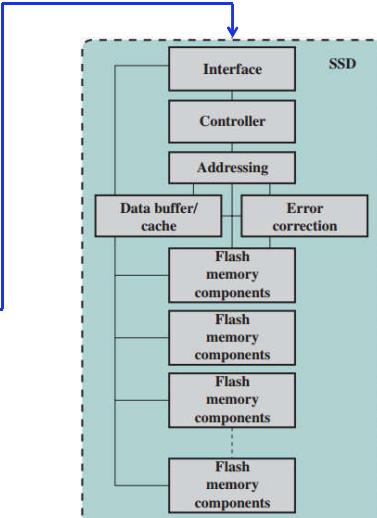
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

85

SSD



Solid State Drive Architecture



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

86



SSD

- ❖ **Controller:** Cung cấp giao diện mức thiết bị và thực hiện chương trình cơ sở (firmware)
- ❖ **Addressing:** Mạch logic lựa chọn chức năng trên các thành phần nhớ flash
- ❖ **Data buffer/cache:** Thành phần nhớ RAM tốc độ cao được sử dụng cho việc phối hợp tốc độ và tăng thông lượng dữ liệu
- ❖ **Error correction:** Mạch logic dùng trong việc dò lỗi và sửa lỗi
- ❖ **Flash memory components:** Các chip flash NAND riêng lẻ.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

87

Các vấn đề của SSD

- ❖ Hiệu năng của SSD có xu hướng giảm khi thiết bị được sử dụng. Vì:
 - file được lưu trữ trên đĩa theo một tập các trang, thường có chiều dài 4KB, các trang này không lưu trữ một cách liên tục trên đĩa.
 - Bộ nhớ flash thường được lưu trữ theo Block, kích thước thường là 512 KB ~ 128 trang.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

88



Hiệu năng của SSD giảm...

❖ Khi ghi một trang vào bộ nhớ flash:

- Toàn bộ block được đọc từ bộ nhớ flash và đưa vào Ram buffer. Sau đó trang tương ứng trong Ram được cập nhật.
- Trước khi block có thể được ghi vào bộ nhớ flash, một block trong bộ nhớ flash phải được xóa. Không thể xóa một trang trong bộ nhớ flash.
- Toàn bộ block từ bộ đệm được ghi ngược trở lại vào bộ nhớ flash.

❖ Theo thời gian sử dụng: bộ nhớ bị phân mảnh

- Lúc đầu: Các trang của file được lưu trữ liên tục → tốn ít block
- Lúc sau: Các trang của file được lưu trữ trên nhiều block --.
Việc đọc ghi các trang từ một block chậm

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

89



Vấn đề 2: Số lần ghi bị hạn chế

❖ Các flash cell mất khả năng ghi và lưu giữ các giá trị

- Thường sau 100 000 lần ghi

❖ Kỹ thuật kéo dài tuổi thọ SSD:

- Sử dụng cache để trì hoãn và nhóm các hoạt động ghi
- Sử dụng wear-leveling algorithms

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

90



Bộ nhớ ảo

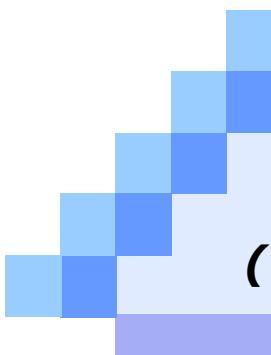
❖ Khái niệm bộ nhớ ảo: gồm bộ nhớ chính và bộ nhớ ngoài mà được CPU coi như là một bộ nhớ duy nhất (bộ nhớ chính)

❖ 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 = 4KBytes
- Kỹ thuật phân đoạn: Chia không gian nhớ thành các đoạn nhớ có kích thước thay đổi, các đoạn nhớ có thể gối lên nhau

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

92



BỘ NHỚ ẢO (VIRTUAL MEMORY)

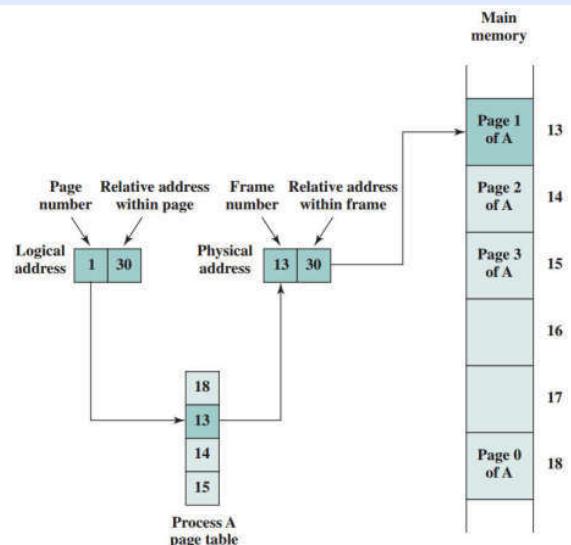
Phân trang

- ❖ 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
- ❖ HĐH 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

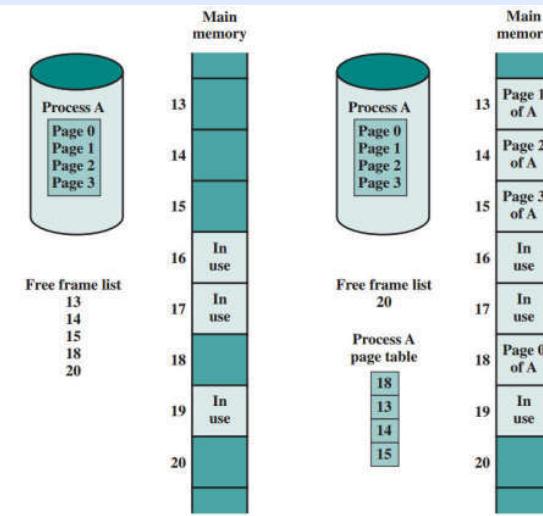
93

Địa chỉ logic và địa chỉ vật lý của phân trang



95

Cấp phát các khung trang



(a) Before

(b) After

94

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

96

Thất bại

- ❖ Quá nhiều tiến trình trong bộ nhớ quá nhỏ
- ❖ HĐH 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

97

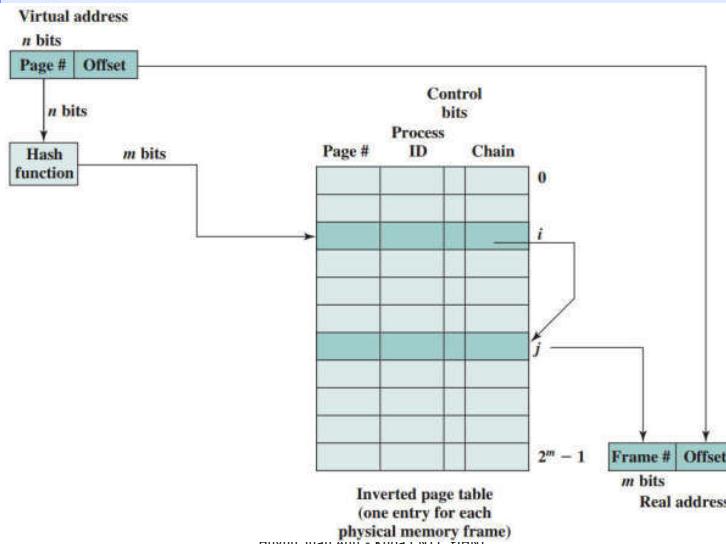
Lợi ích

- ❖ 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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

98

Cấu trúc bảng trang



99

Translation Lookaside Buffer

- ❖ Mỗi tham chiếu bộ nhớ ảo gây ra hai truy cập bộ nhớ vật lý
 - Tìm điểm vào của bảng trang
 - Tìm dữ liệu
- ❖ Sử dụng cache đặc biệt cho bảng trang
 - TLB

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

100



Minh họa: bộ nhớ trên PC hiện nay

❖ 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

- SIMM – Single Inline Memory Module
- DIMM – Dual Inline Memory Module



Hệ thống 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



Bài tập

1. Phân biệt SRAM và DRAM
2. Sự khác nhau giữa ánh xạ trực tiếp, ánh xạ liên kết toàn phần và ánh xạ liên kết tập hợp.
3. Một cache kiểu liên kết tập hợp có 64 line được chia thành các các tập hợp, mỗi tập hợp có 4 line. Bộ nhớ chính chứa 4K block mỗi block có 128 từ. Cho biết định dạng địa chỉ của bộ nhớ chính
4. Một cache liên kết tập hợp 2 đường có mỗi line chứa 16 byte, dung lượng cache 8 Kbyte. Bộ nhớ chính 64-Mbyte được định địa chỉ theo từng byte. Cho biết định dạng địa chỉ của bộ nhớ chính.



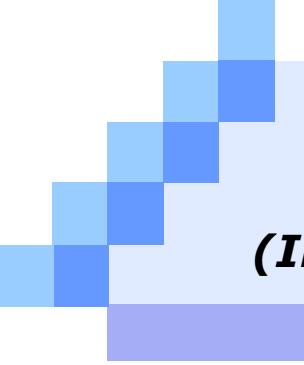
Bài tập

5. Không gian địa chỉ bộ nhớ chính = 8GB. Dung lượng bộ nhớ cache là 512KB. Kích thước Line (Block) = 64byte. Xác định số bit của các trường địa chỉ cho ba trường hợp tổ chức:
 1. Ánh xạ trực tiếp
 2. Ánh xạ liên kết toàn phần
 3. Ánh xạ liên kết tập hợp 4 đường
6. Một vi xử lý 32 bit có bộ nhớ cache 16 Kbyte theo kiểu liên kết tập hợp 4 đường. Giả sử kích thước mỗi line là 4 từ nhớ 32-bit.
 1. Cho biết định dạng địa chỉ của bộ nhớ chính
 2. Xác định địa chỉ TAG, SET của byte nhớ có địa chỉ ABCDE8F8



Bài tập

1. Nêu khái niệm track, cylinder, sector. Giá trị mặc định của mỗi sector là bao nhiêu.
2. Đặc điểm chung của các RAID
3. Phân biệt RAID 0 và RAID 1
4. Ưu và nhược điểm của SSD
5. Các thành phần chính của SSD
6. Các thành phần của bộ nhớ trên PC hiện nay



8. VÀO RA (INPUT – OUTPUT)



TỔNG QUAN VỀ VÀO-RA



Nội dung

- ❖ Tổng quan về vào-ra
- ❖ Các phương pháp điều khiển vào-ra
- ❖ Nối ghép thiết bị ngoại vi
- ❖ Các cổng vào-ra thông dụng trên PC

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

2



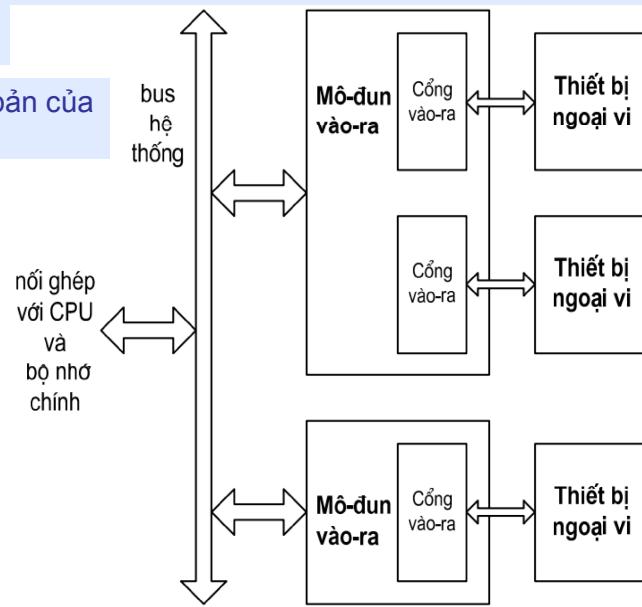
1. Giới thiệu chung

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

4

Cấu trúc cơ bản của vào-ra



5

Đặc điểm của vào-ra

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

6

2. CÁC THIẾT BỊ NGOẠI VI

- ❖ Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- ❖ Phân loại:
 - Thiết bị ngoại vi giao tiếp người-máy: Bàn phím, Màn hình, Máy in,...
 - Thiết bị ngoại vi giao tiếp máy-máy: gồm các thiết bị theo dõi và kiểm tra
 - Thiết bị ngoại vi truyền thông: Modem, Network Interface Card (NIC)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

7

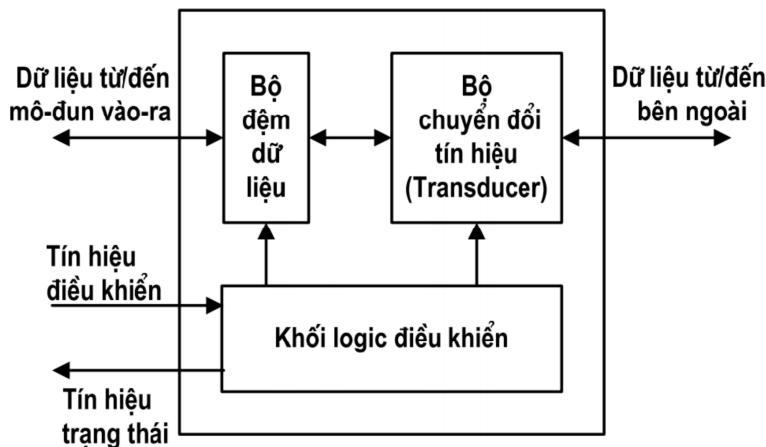
Các thiết bị ngoại vi

Input type	Prime examples	Other examples	Data rate (b/s)	Main uses
Symbol	Keyboard, keypad	Music note, OCR	10s	Ubiquitous
Position	Mouse, touchpad	Stick, wheel, glove	100s	Ubiquitous
Identity	Barcode reader	Badge, fingerprint	100s	Sales, security
Sensory	Touch, motion, light	Scent, brain signal	100s	Control, security
Audio	Microphone	Phone, radio, tape	1000s	Ubiquitous
Image	Scanner, camera	Graphic tablet	1000s-10 ⁶ s	Photos, publishing
Video	Camcorder, DVD	VCR, TV cable	1000s-10 ⁹ s	Entertainment
Output type	Prime examples	Other examples	Data rate (b/s)	Main uses
Symbol	LCD line segments	LED, status light	10s	Ubiquitous
Position	Stepper motor	Robotic motion	100s	Ubiquitous
Warning	Buzzer, bell, siren	Flashing light	A few	Safety, security
Sensory	Braille text	Scent, brain stimulus	100s	Personal assistance
Audio	Speaker, audiotape	Voice synthesizer	1000s	Ubiquitous
Image	Monitor, printer	Plotter, microfilm	1000s	Ubiquitous
Video	Monitor, TV screen	Film/video recorder	1000s-10 ⁹ s	Entertainment
Two-way I/O	Prime examples	Other examples	Data rate (b/s)	Main uses
Mass storage	Hard/floppy disk	CD, tape, archive	10 ⁶ s	Ubiquitous
Network	Modem, fax, LAN	Cable, DSL, ATM	1000s-10 ⁹ s	Ubiquitous

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

8

Cấu trúc chung của thiết bị ngoại vi

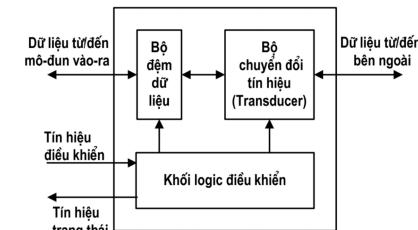


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

9

Các thành phần của thiết bị ngoại vi

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



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

10

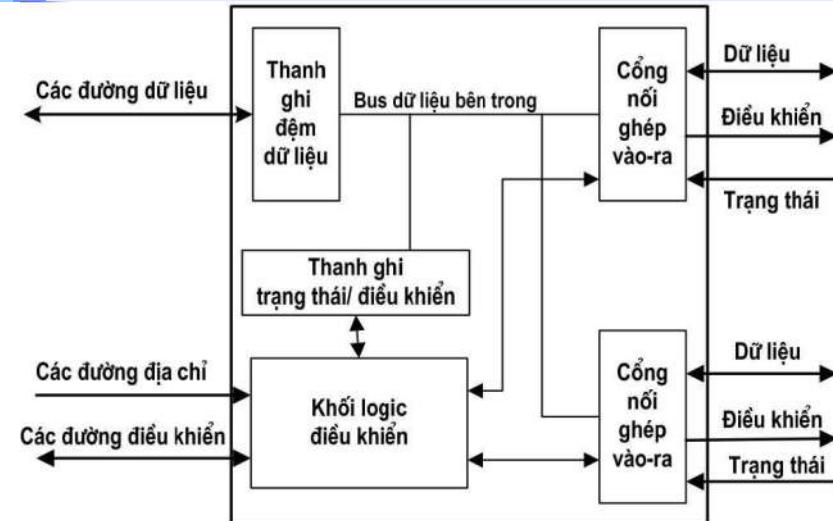
3. MODULE VÀO-RA

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

11

Cấu trúc chung của module vào ra



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

12

Các thành phần của module vào-ra

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

4. Địa chỉ hóa cổng vào-ra

Không gian địa chỉ bộ nhớ	N bit	Không gian địa chỉ vào-ra	N ₁ bit
000...000		00...00	00...00
000...001		00...01	00...01
000...010		00...10	00...10
000...011		-	00...11
000...100		-	
000...101		-	
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
111...111			11...11

Không gian địa chỉ của bộ xử lý (tiếp)

- Một số bộ xử lý chỉ quản lý duy nhất một không gian địa chỉ:
 - không gian địa chỉ bộ nhớ: 2^N địa chỉ
- Ví dụ:
 - Các bộ xử lý 680x0 (Motorola)
 - Các bộ xử lý theo kiến trúc RISC: MIPS, ...

Không gian địa chỉ của bộ xử lý (tiếp)

- Một số bộ xử lý quản lý hai không gian địa chỉ tách biệt:
 - Không gian địa chỉ bộ nhớ: 2^N địa chỉ
 - Không gian địa chỉ vào-ra: 2^{N_1} địa chỉ
 - Có tín hiệu điều khiển phân biệt truy nhập không gian địa chỉ
 - Tập lệnh có các lệnh vào-ra chuyên dụng
- Ví dụ: Pentium (Intel)
 - không gian địa chỉ bộ nhớ = 2^{32} byte = 4GB
 - không gian địa chỉ vào-ra = 2^{16} byte = 64KB
 - Tín hiệu điều khiển M/ \overline{IO}
 - Lệnh vào-ra chuyên dụng: IN, OUT



Các phương pháp địa chỉ hóa cổng vào-ra

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



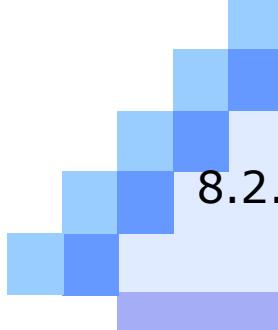
Vào-ra riêng biệt

- ❖ Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ vào-ra
- ❖ CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh vào-ra chuyên dụng (IN, OUT)
- ❖ Chỉ có thể thực hiện trên các hệ thống có quản lý không gian địa chỉ vào-ra riêng biệt



Vào-ra theo bản đồ bộ nhớ

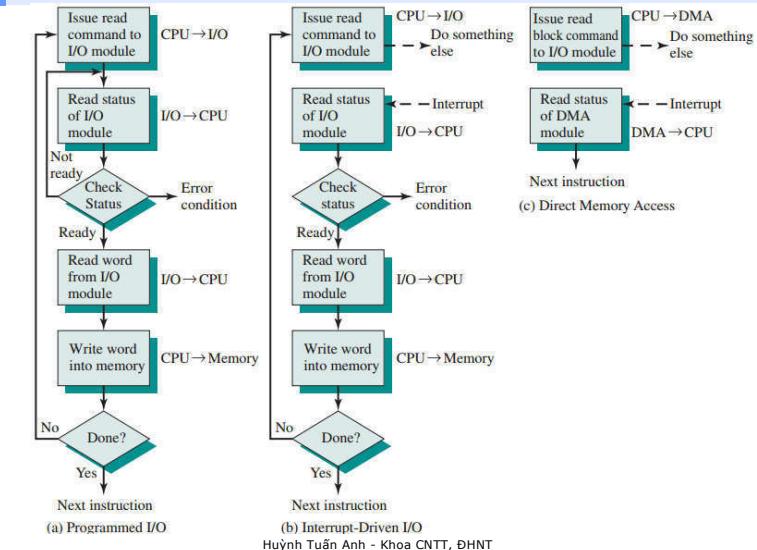
- ❖ Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ bộ nhớ
- ❖ Vào-ra giống như đọc/ghi bộ nhớ
- ❖ CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh truy nhập dữ liệu bộ nhớ
- ❖ Có thể thực hiện trên mọi hệ thống



8.2. 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

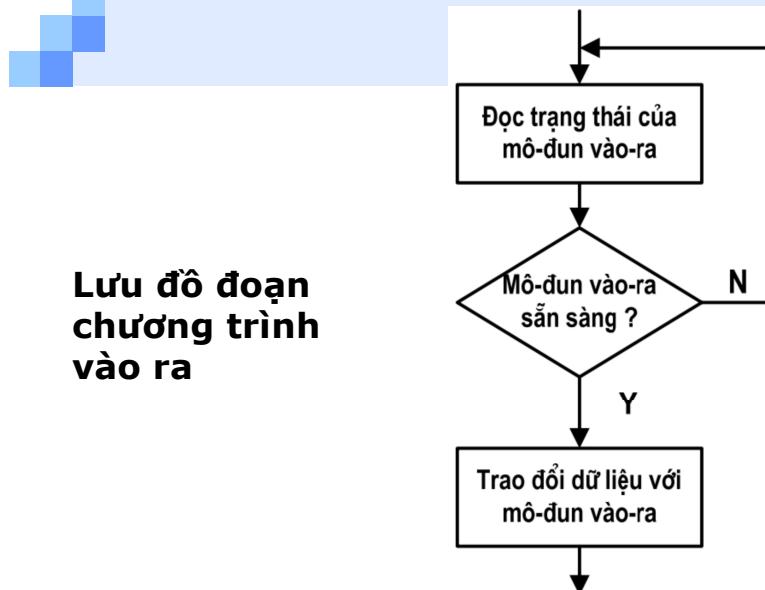
Các tín hiệu điều khiển vào ra

- ❖ Tín hiệu **điều khiển (Control)**: kích hoạt thiết bị ngoại vi
- ❖ Tín hiệu **kiểm tra (Test)**: kiểm tra trạng thái của mô-đun vào-ra và thiết bị ngoại vi
- ❖ Tín hiệu điều khiển **đọc (Read)**: yêu cầu mô-đun vào-ra nhận dữ liệu từ thiết bị ngoại vi và đưa vào thanh ghi 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 thanh ghi đệm dữ liệu rồi chuyển ra thiết bị ngoại vi

Các lệnh vào ra

- ❖ Với vào-ra riêng biệt: sử dụng các lệnh vào-ra chuyên dụng (IN, OUT).
- ❖ 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.

Lưu đồ đoạn chương trình vào ra



Hoạt động của vào-ra bằng chương trình

- ❖ CPU yêu cầu thao tác vào-ra
- ❖ Mô-đun vào-ra thực hiện thao tác
- ❖ Mô-đun vào-ra thiết lập các bit trạng thái
- ❖ CPU kiểm tra các bit trạng thái:
 - Nếu chưa sẵn sàng thì quay lại kiểm tra
 - Nếu sẵn sàng thì chuyển sang trao đổi dữ liệu với mô-đun vào-ra

Đặc điểm

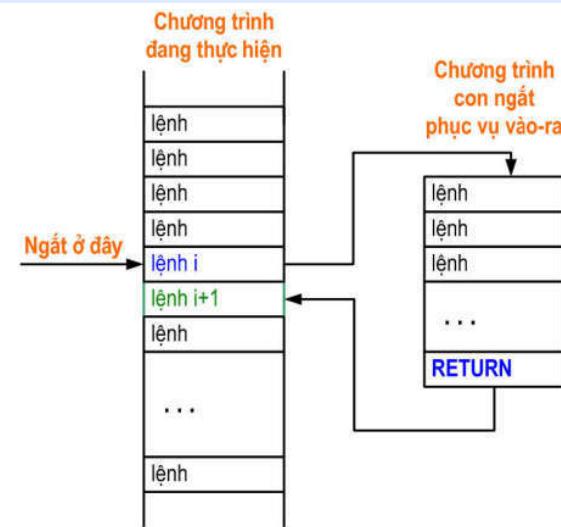
- ❖ Vào-ra do ý muốn của người lập trình
- ❖ CPU trực tiếp điều khiển vào-ra
- ❖ CPU đợi module vào-ra → tiêu tốn thời gian của CPU

2. Vào-ra điều khiển bằng ngắt

❖ Nguyên tắc chung:

- CPU không phải đợi trạng thái sẵn sang của module vào-ra, CPU thực hiện một chương trình nào đó
- Khi module vào-ra sẵn sàng thì nó phát tín hiệu ngắt CPU
- CPU thực hiện chương trình con 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 dữ liệu

Nhìn từ CPU

- ❖ Phát tín hiệu điều khiển **đọc**
- ❖ Làm việc khác
- ❖ Cuối mỗi chu trình lệnh, kiểm tra tín hiệu ngắt.
- ❖ **Nếu bị ngắt:**
 - Cắt ngũ cảnh (nội dung các thanh ghi)
 - Thực hiện chương trình

Nhìn từ module vào ra

- ❖ Nhận tín hiệu điều khiển đọc từ CPU.
- ❖ Nhận dữ liệu từ thiết bị ngoại vi, trong khi đó CPU làm việc khác.
- ❖ **Khi đã có dữ liệu → module vào ra phát tín hiệu ngắt CPU**
 - CPU yêu cầu dữ liệu
 - Module vào ra chuyển dữ liệu đến CPU

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 ?

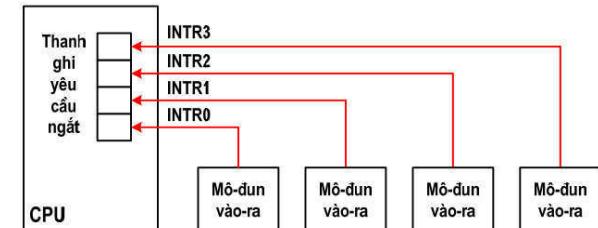
Các phương pháp nối ghép ngắt

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

33

Nhiều đường yêu cầu ngắt

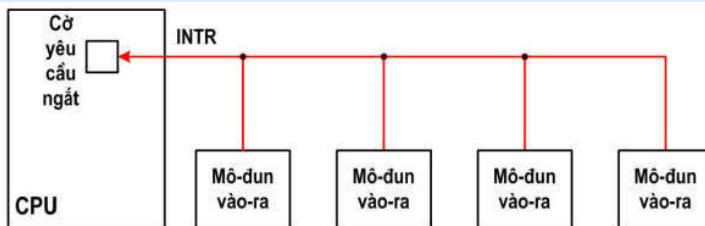


- ❖ Mỗi module vào-ra được nối với một đường yêu cầu ngắt
- ❖ CPU phải có nhiều đường tín hiệu yêu cầu ngắt
- ❖ Hạn chế số lượng module vào-ra
- ❖ Các đường ngắt được qui định mức ưu tiên

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

34

Hỏi vòng bằng phần mềm

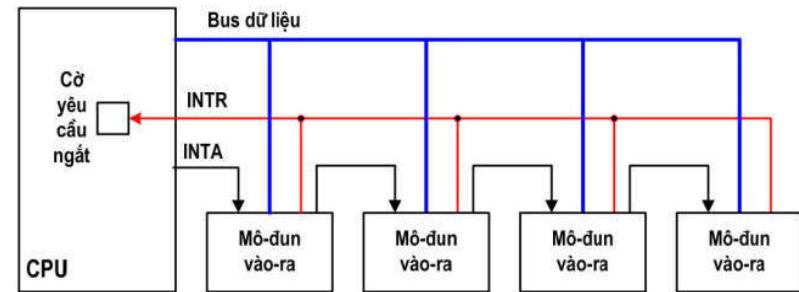


- ❖ CPU thực hiện phần mềm hỏi lần lượt từng module vào-ra
- ❖ Chậm
- ❖ Thứ tự các mô-đun được hỏi vòng chính là thứ tự ưu tiên

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

35

Hỏi vòng bằng phần cứng



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

36

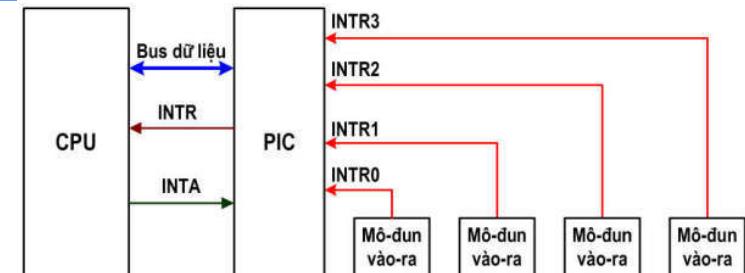
Kiểm tra vòng bằng phần cứng (tiếp)

- ❖ CPU phát tín hiệu chấp nhận ngắt (INTA) đến module vào-ra đầu tiên
- ❖ Nếu module vào-ra đó không gây ra ngắt thì nó gửi tín hiệu đến module kế tiếp cho đến khi xác định được module gây ngắt.
- ❖ Thứ tự các module vào-ra kết nối trong chuỗi xác định thứ tự ưu tiên

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
- ❖ CPU trực tiếp điều khiển vào-ra
- ❖ CPU không phải đợi mô-đun vào-ra → hiệu quả sử dụng CPU tốt hơn

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

Ngắt của 80x86

- ❖ Tổ chức kiểu vector ngắt
- ❖ Số hiệu ngắt: n (00-FF)
- ❖ Bảng vector ngắt: 256 x 4 byte = 1024bytes
00000 – 003FF
- ❖ Lệnh INT n

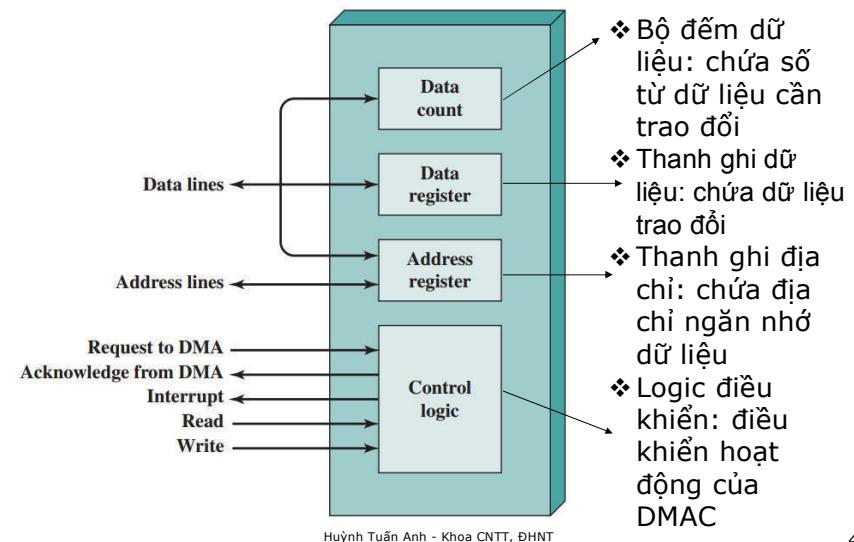
3. DMA (Direct Memory Access)

- ❖ Vào-ra bằng chương trình và bằng ngắt do CPU trực tiếp điều khiển:
 - Chiếm thời gian của CPU
 - Tốc độ truyền bị hạn chế vì phải chuyển qua CPU
- ❖ Để khắc phục dùng DMA
 - Thêm module phần cứng trên bus → DMAC (Controller)
 - DMAC điều khiển trao đổi dữ liệu giữa module vào-ra với bộ nhớ chính

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

41

Sơ đồ cấu trúc của DMAC



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

42

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

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

43

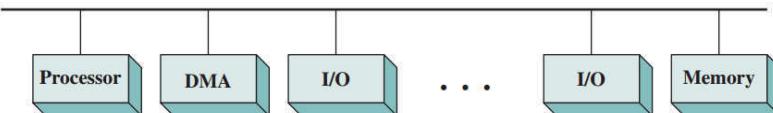
Các kiểu thực hiện DMA

- ❖ 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.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

44

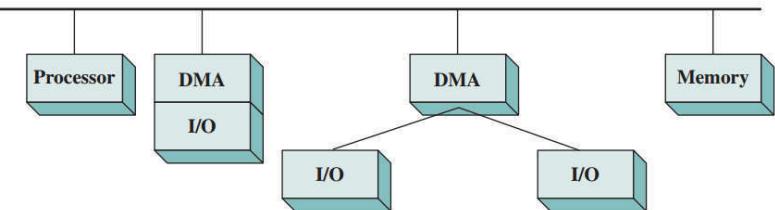
Cấu hình DMA (1): Bus chung, DMA tách rời



(a) Single-bus, detached DMA

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

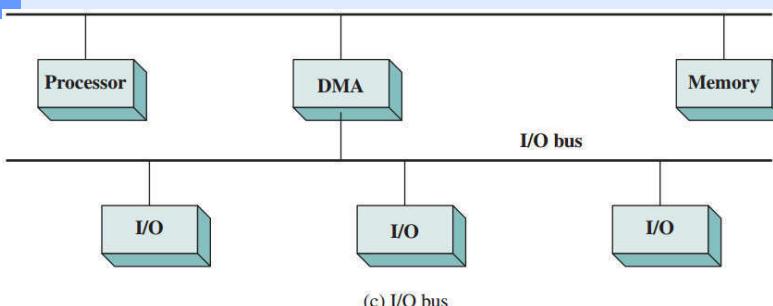
Cấu hình DMA 2: Bus chung, DMA-I/O tích hợp



(b) Single-bus, integrated DMA-I/O

- ❖ DMAC điều khiển một hoặc vài module 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 I/O tách rời

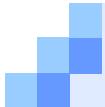


(c) I/O bus

- ❖ Bus vào-ra tách rời hỗ trợ tất cả các thiết bị cho phép DMA

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. Kênh vào-ra hay là bộ xử lý vào-ra

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

GHÉP NỐI CÁC THIẾT BỊ NGOẠI VI

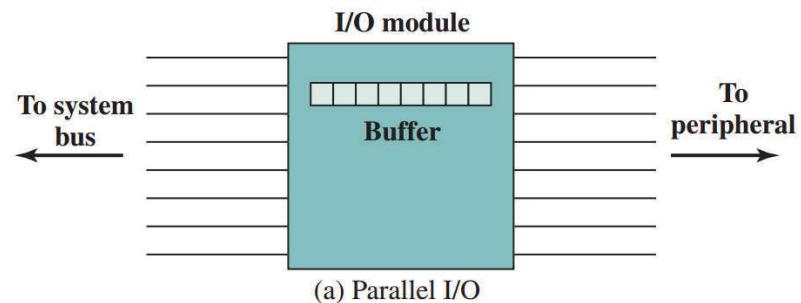


1. Các kiểu ghép nối vào ra

- ❖ Ghép nối song song
- ❖ Ghép nối nối tiếp



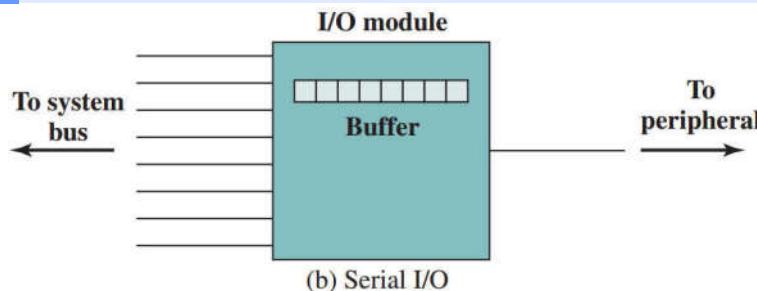
Ghép nối song song



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



Ghép nối 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 ngoại vi

❖ Điểm tới đa điểm (Point to Multipoint)

- Thông qua một cổng vào-ra cho phép nối ghép được với nhiều thiết bị ngoại vi
 - SCSI (Small Computer System Interface): 7 hoặc 15 thiết bị
 - USB (Universal Serial Bus): 127 thiết bị
 - IEEE 1394 (FireWire): 63 thiết bị

CÁC KIẾN TRÚC SONG SONG

- ❖ 9.1. Phân loại kiến trúc máy tính
- ❖ 9.2. Đa xử lý bộ nhớ dùng chung
- ❖ 9.3. Đa xử lý bộ nhớ phân tán
- ❖ 9.4. Bộ xử lý đa lõi
- ❖ 9.5. Bộ xử lý đồ họa*

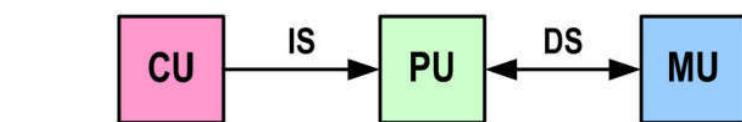
2

9.1. Phân loại kiến trúc máy tính

- ❖ Phân loại của 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

3

SISD - Single Instruction Stream, Single Data Stream

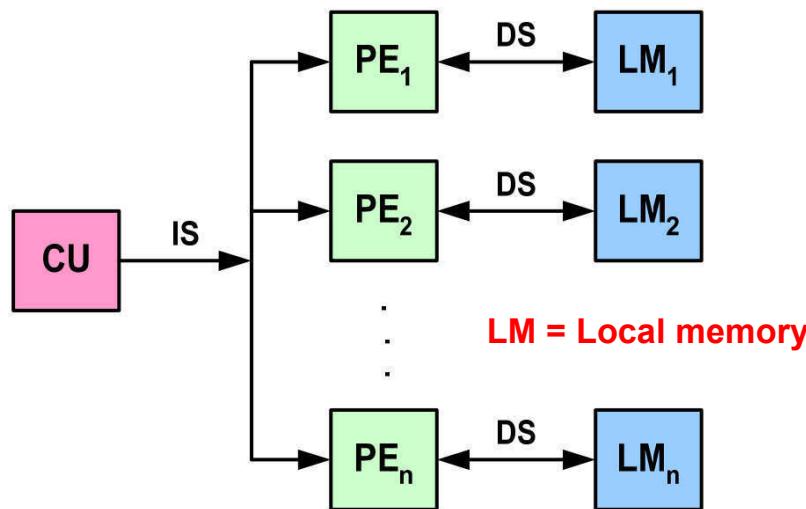


- ❖ CU: Control Unit
- ❖ PU: Processing Unit
- ❖ MU: Memory Unit
- ❖ Một bộ xử lý
- ❖ Dữ liệu được lưu trữ trong một bộ nhớ
- ❖ Chính là Kiến trúc von Neumann

4



SIMD - Single Instruction Stream, Multiple Data Stream



5



SIMD (tiếp)

- ❖ Đơn dòng lệnh điều khiển đồng thời các phần tử xử lý PE (processing elements)
- ❖ 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

6



MISD - Multiple Instruction Stream, Single Data Stream

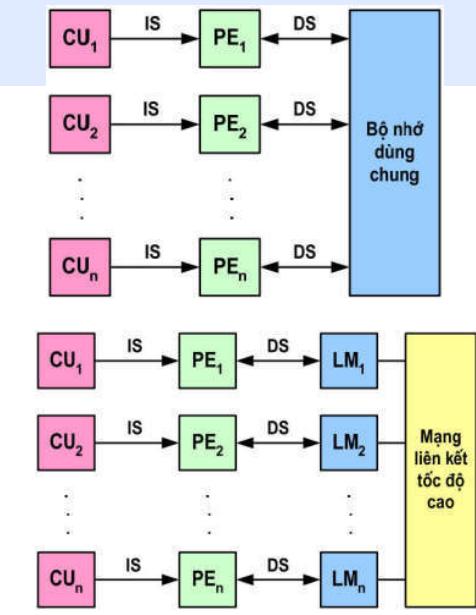
- ❖ Một luồng dữ liệu cùng được truyền đến một tập các bộ xử lý
- ❖ Mỗi bộ xử lý thực hiện một dãy lệnh khác nhau
- ❖ Chưa tồn tại máy tính thực tế
- ❖ Có thể có trong tương lai

7



MIMD

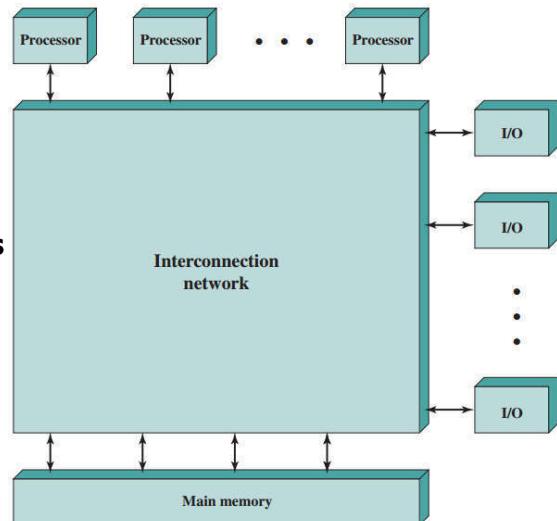
- ❖ Tập các bộ xử lý
- ❖ Các bộ xử lý đồng thời thực hiện các dãy lệnh khác nhau trên các dữ liệu khác nhau
- ❖ Các mô hình MIMD
 - Multiprocessors (Shared Memory)
 - Multicomputers (Distributed Memory)



8

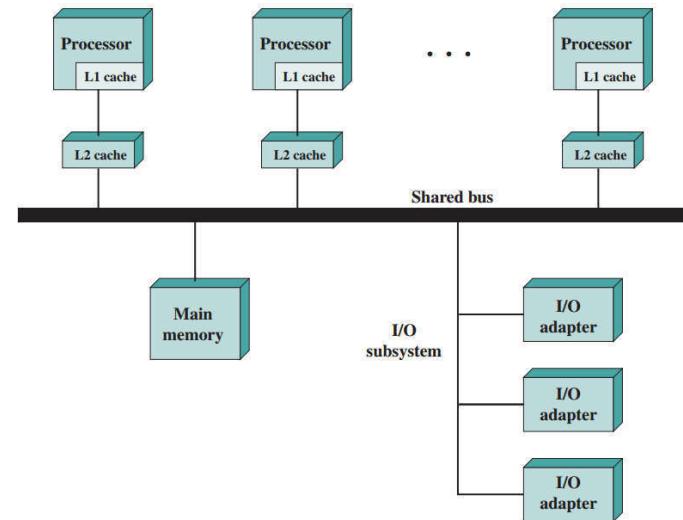
9.2. Đa xử lý bộ nhớ dùng chung

**SMP-
Symmetric
MultiProcessors**



9

Kiến trúc đơn giản nhất của đa xử lý bộ nhớ dùng chung: Time – shared bus



10

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
- ❖ Tất cả các bộ xử lý chia sẻ truy nhập vào-ra
- ❖ Các bộ xử lý có thể thực hiện chức năng giống nhau
- ❖ Hệ thống được điều khiển bởi một hệ điều hành phân tán

11

Ưu điểm của SMP

- ❖ Hiệu năng
 - Các công việc có thể thực hiện song song
- ❖ Tính sẵn dùng
 - Các bộ xử lý có thể thực hiện các chức năng giống nhau, vì vậy lỗi của một bộ xử lý sẽ không làm dừng hệ thống
- ❖ Khả năng mở rộng
 - Người sử dụng có thể tăng hiệu năng bằng cách thêm bộ xử lý

12

9.3. Đa xử lý bộ nhớ phân tán

❖ Clusters

- 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
- 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

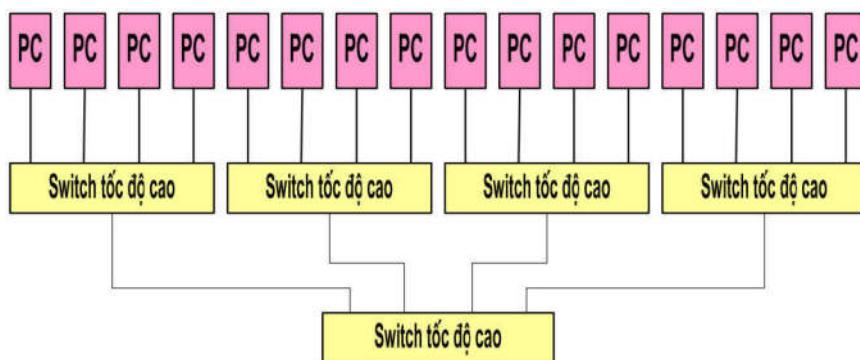
13

Cluster (tiếp)

- ❖ Dễ dàng xây dựng và mở rộng
- ❖ Tính sẵn sàng cao
- ❖ Khả năng chịu lỗi
- ❖ Giá thành rẻ với hiệu năng cao

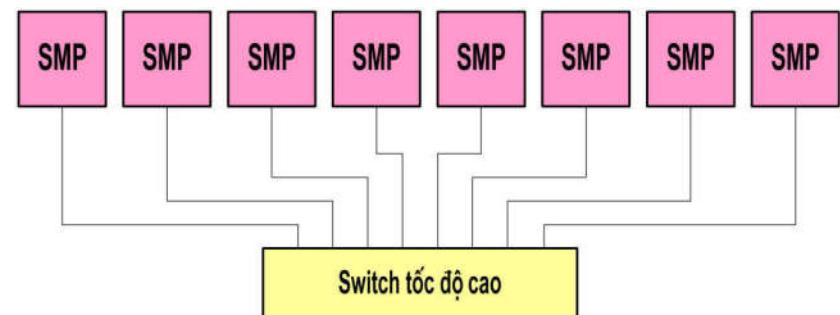
14

Cluster of PCs



15

Cluster of SMPs



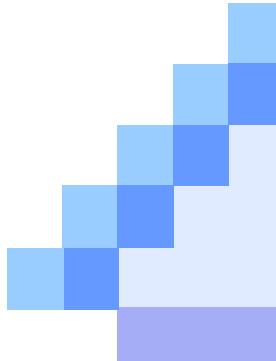
16



Ví dụ: Hệ thống máy chủ Google (12/ 2000)

- ❖ Cluster of PCs
- ❖ Hơn 6.000 bộ xử lý
- ❖ Hệ thống lưu trữ dùng RAID: có 12.000 đĩa cứng ~ 1petabyte (1triệu GB)
- ❖ 2 site ở Silicon Valley, 1site ở Virginia
- ❖ Mỗi site được kết nối với Internet qua OC48 (2488Mbps)

17



9.4. Bộ xử lý đa lõi (multicores)



Pipelining

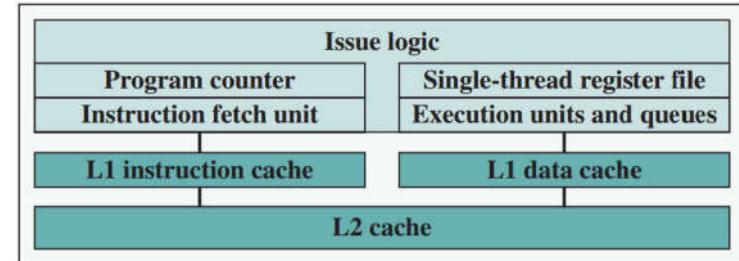
- ❖ Pipelining (Kỹ thuật ống dẫn): Đây là một kỹ thuật làm cho các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc.

Chuỗi lệnh	Chu kỳ xung nhịp									
	1	2	3	4	5	6	7	8	9	
Lệnh thứ i	IF	ID	EX	MEM	RS					
Lệnh thứ i+1		IF	ID	EX	MEM	RS				
Lệnh thứ i+2			IF	ID	EX	MEM	RS			
Lệnh thứ i+3				IF	ID	EX	MEM	RS		
Lệnh thứ i+4					IF	ID	EX	MEM	RS	

19



Siêu vô hướng:

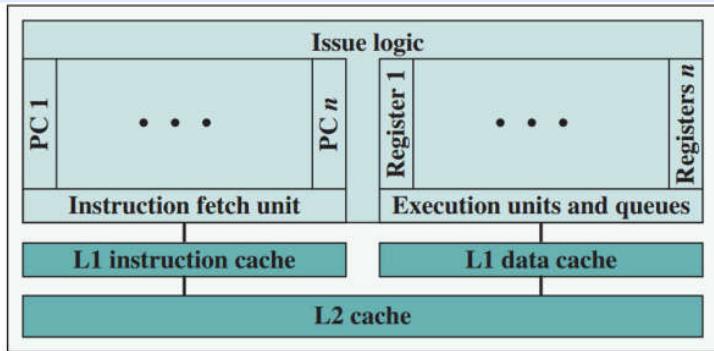


(a) Superscalar

- ❖ Nhiều đường ống dẫn được xây dựng bằng cách nhân rộng việc xử lý các tài nguyên
 - Cho phép xử lý song song các lệnh trong các đường ống song song.

20

Đa luồng - Simultaneous MultiThreading (SMT)

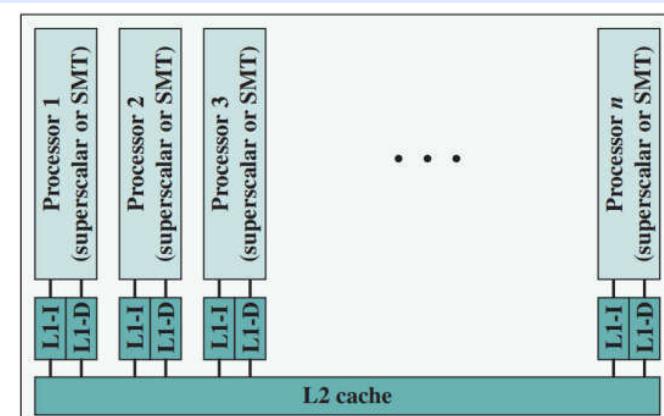


(b) Simultaneous multithreading

- Các dãy thanh ghi được nhân lên sao cho nhiều luồng có thể chia sẻ việc sử dụng các tài nguyên pipeline

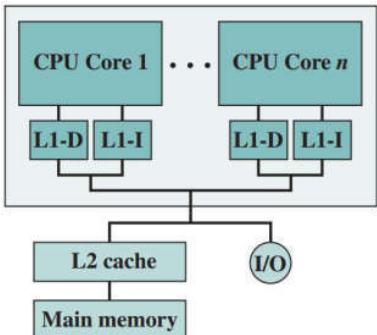
21

Bộ xử lý đa lõi



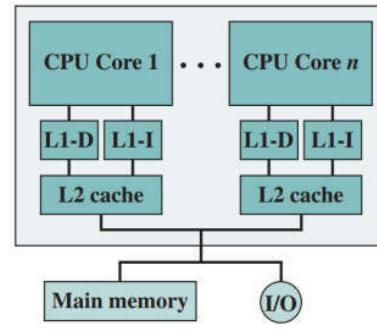
22

Các dạng tổ chức bộ xử lý đa lõi



(a) Dedicated L1 cache

- Mỗi nhân có một cache L1 riêng
- L1-D: Data cache
- L1-I: Instruction cache
- Ví dụ: ARM11 MPCore

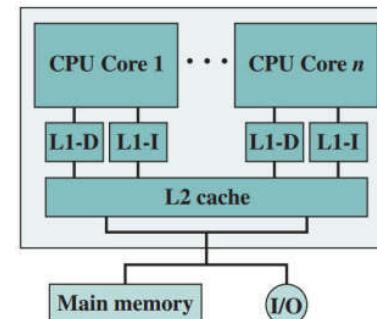


(b) Dedicated L2 cache

- Chip với cache không chia sẻ
- Cache L2 đặt bên trong chip
- Ví dụ: AMD Opteron

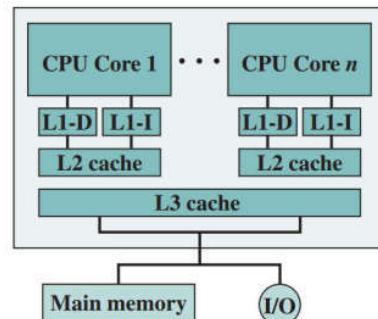
23

Các dạng tổ chức bộ xử lý đa lõi (tt)



(c) Shared L2 cache

- Sử dụng cache L2 chia sẻ giữa các nhân
- Ví dụ: Intel Core Duo



(d) Shared L3 cache

- Cache L1, L2 riêng cho từng nhân
- Cache L3 được chia sẻ giữa các nhân
- Ví dụ: Intel core i7

24

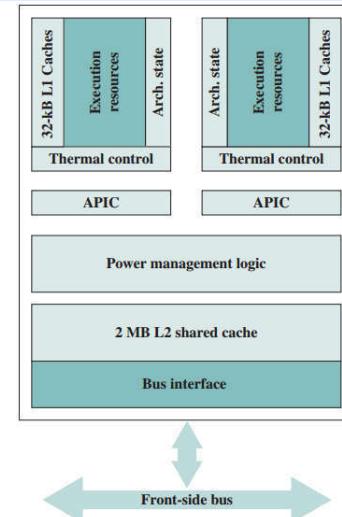
Ưu điểm của sử dụng cache chung

- ❖ Giảm thiểu miss rates:
 - Nếu một nhân truy cập địa chỉ nhớ nào đó trong bộ nhớ chính → nội dung của ô nhớ sẽ đưa vào cache chia sẻ.
 - Nếu một thread trên một nhân khác truy cập cùng một địa chỉ trong bộ nhớ chính thì khi đó nội dung của ô nhớ này đã sẵn sang trong cache chia sẻ
- ❖ Dữ liệu chia sẻ bởi các nhân sẽ không bị sao chép tại mức cache chia sẻ.
- ❖ Các nhân giao tiếp với nhau dễ dàng

25

Intel - Core Duo

- ❖ 2006
- ❖ Two x86 superscalar, shared L2 cache
- ❖ Dedicated L1 cache per core
 - 32KB instruction and 32KB data
- ❖ 2MB shared L2 cache



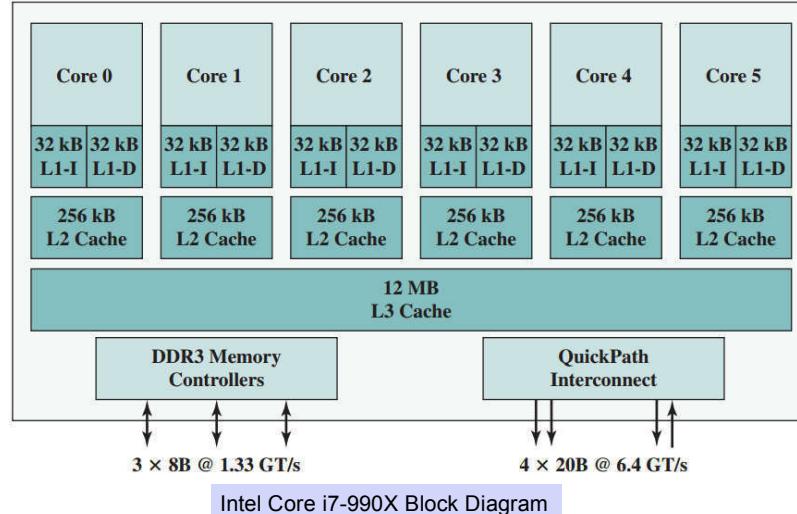
26

Intel x86 Multicore Organization - Core i7

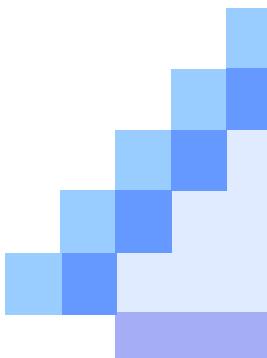
- ❖ November 2008
- ❖ Four x86 SMT processors
- ❖ Dedicated L2, shared L3 cache
- ❖ Speculative pre-fetch for caches
- ❖ On chip DDR3 memory controller
 - Three 8 byte channels (192 bits) giving 32GB/s
 - No front side bus
- ❖ QuickPath Interconnection
 - Cache coherent point-to-point link
 - High speed communications between processor chips
 - 6.4G transfers per second, 16 bits per transfer
 - Dedicated bi-directional pairs
 - Total bandwidth 25.6GB/s

27

Intel Core i7



28



BỘ XỬ LÝ ĐỒ HỌA (GPU)

History of GPUs

❖ Early video cards

- Frame buffer memory with address generation for video output.

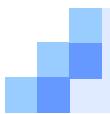
❖ 3D graphics processing

- Originally high-end computers (e.g., SGI)
- Moore's Law ⇒ lower cost, higher density
- 3D graphics cards for PCs and game consoles

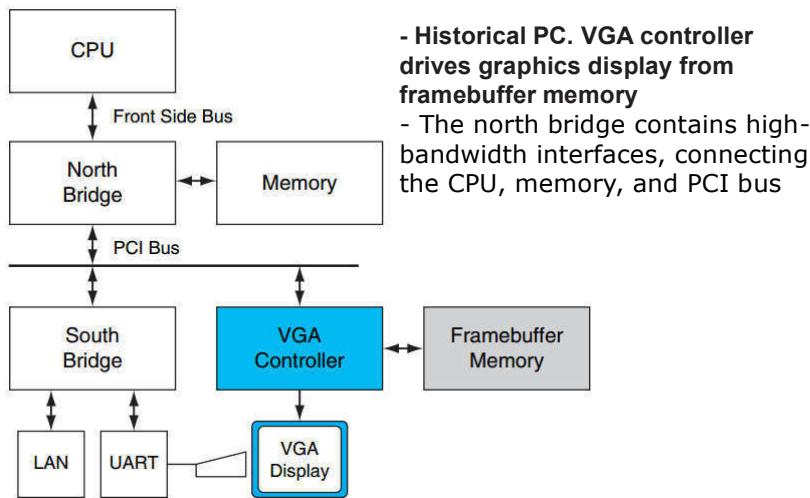
❖ 3D graphics cards for PCs and game consoles

- Processors oriented to 3D graphics tasks
- Vertex/pixel processing, shading, texture mapping, rasterization

30



Graphics in the System

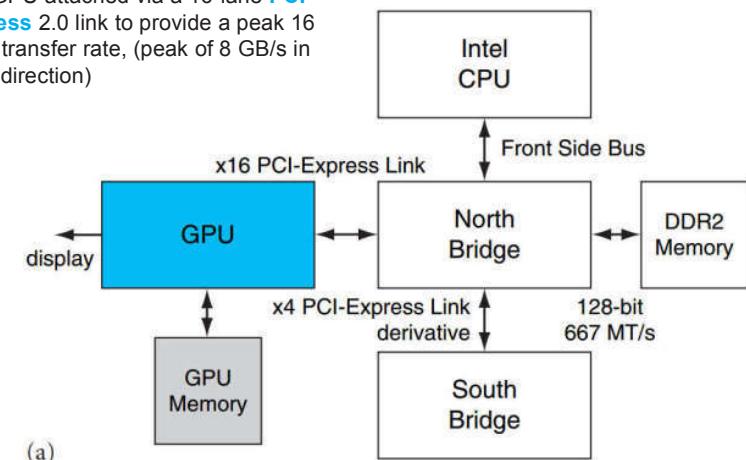


31



Graphics in the System (tt): Intel PC

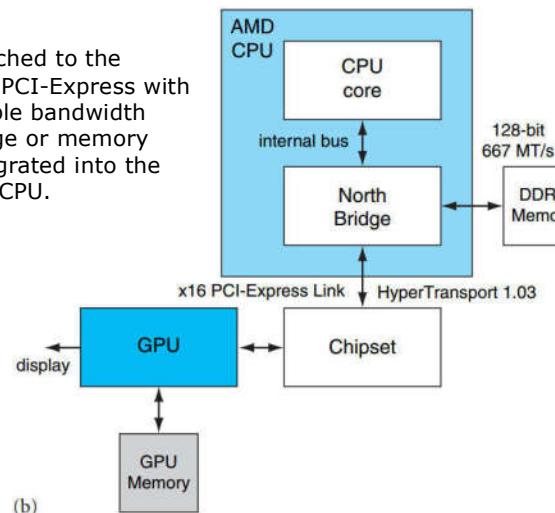
- The GPU attached via a 16-lane PCI-Express 2.0 link to provide a peak 16 GB/s transfer rate, (peak of 8 GB/s in each direction)



32

Graphics in the System (tt): AMD PC

- The GPU is attached to the chipset, also via PCI-Express with the same available bandwidth
- The north bridge or memory controller is integrated into the same die as the CPU.



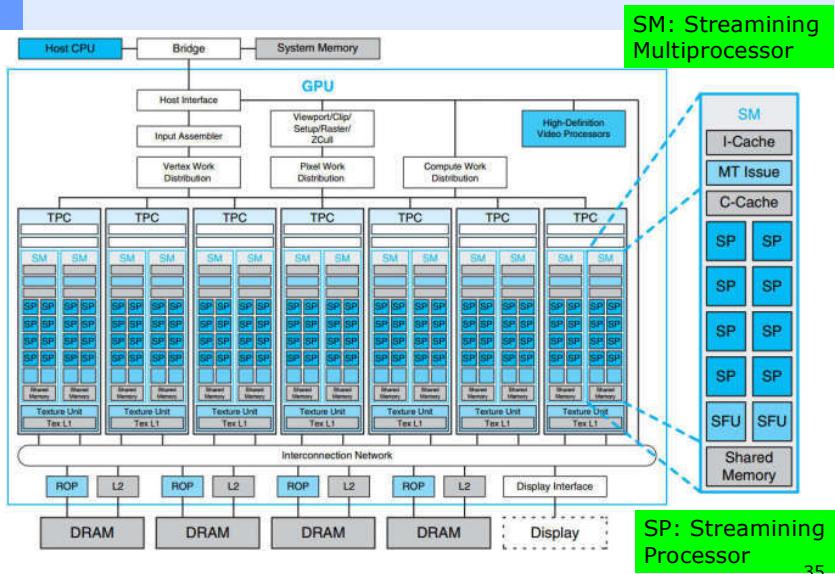
33

GPU Architectures

- ❖ Processing is highly data-parallel
 - GPUs are highly multithreaded
 - Use thread switching to hide memory latency
 - Less reliance on multi-level caches
 - Graphics memory is wide and high-bandwidth
- ❖ Trend toward general purpose GPUs
 - Heterogeneous CPU/GPU systems
 - CPU for sequential code, GPU for parallel code
- ❖ Programming languages/APIs
 - DirectX, OpenGL
 - C for Graphics (Cg), High Level Shader Language (HLSL)
 - Compute Unified Device Architecture (CUDA)

34

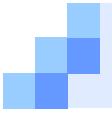
Ex: Tesla architecture - NVIDIA GeForce 8800



Tesla architecture - NVIDIA GeForce 8800

- ❖ Streaming Processors
 - Single-precision FP and integer units
 - Each SP is fine-grained multithreaded
- ❖ Warp: group of 32 threads
 - Executed in parallel, SIMD style
 - 8 SPs × 4 clock cycles
 - Hardware contexts for 24 warps
 - Registers, PCs, ...

36



Classifying GPUs

- ❖ Don't fit nicely into SIMD/MIMD model
 - Conditional execution in a thread allows an illusion of MIMD
 - But with performance degradation
 - Need to write general purpose code with care

	Static: Discovered at Compile Time	Dynamic: Discovered at Runtime
Instruction-Level Parallelism	VLIW	Superscalar
Data-Level Parallelism	SIMD or Vector	Tesla Multiprocessor