MÔN NHẬP MÔN ĐIỆN TOÁN

Đối tượng : SV đại học chính quy khoa **Khoa học & Kỹ thuật Máy tính**

Nội dung chính gồm 7 chương :

1. Khái niệm cơ bản.
2. Phần cứng máy tính.
3. Hệ điều hành và mạng máy tính.
4. Ngôn ngữ lập trình.
5. Cơ sở dữ liệu.
6. Phần mềm ứng dụng.
7. Các vấn đề tổ chức & xã hội.

Tài liệu tham khảo :

- Computing, 3rd ed., Geoffrey Knott & Nick Waites, 2000.
- Tập slide bài giảng & thực hành của môn học này.

MÔN NHẬP MÔN ĐIỆN TOÁN

Chương 1

KHÁI NIỆM CƠ BẢN

1.1 Định nghĩa sơ khởi về máy tính số
1.2 Lịch sử phát triển máy tính số
1.3 Hệ thống số đếm
1.4 Biểu diễn dữ liệu
1.5 Luận lý máy tính

1.1 Định nghĩa sơ khởi về máy tính số

- Con người thông minh hơn các động vật khác nhiều, trong cuộc sống, họ đã chế tạo ngày càng nhiều công cụ, thiết bị để hỗ trợ mình trong hoạt động. Các công cụ, thiết bị do con người chế tạo ngày càng tinh vi, phức tạp và thực hiện nhiều công việc hơn trước đây. Mỗi công cụ, thiết bị thường chỉ thực hiện được 1 vài công việc cụ thể nào đó. Thí dụ, cây chổi để quét, radio để bắt và nghe đài audio...
- Máy tính số (digital computer) cũng là 1 thiết bị, nhưng thay vì chỉ thực hiện 1 số chức năng cụ thể, sát với nhu cầu đời thường của con người, nó có thể thực hiện 1 số hữu hạn các chức năng cơ bản (tập lệnh), mỗi lệnh rất sơ khai chưa giải quyết trực tiếp được nhu cầu đời thường nào của con người. Cơ chế thực hiện các lệnh là tự động, bắt đầu từ lệnh được chỉ định nào đó rồi tuần tự từng lệnh kế tiếp cho đến lệnh cuối cùng. Danh sách các lệnh được thực hiện này được gọi là chương trình.

Định nghĩa sơ khởi về máy tính số (tt)

- Các lệnh mà máy hiểu và thực hiện được được gọi là lệnh máy. Ta dùng ngôn ngữ để miêu tả các lệnh. Ngôn ngữ lập trình cấu thành từ 2 yếu tố chính yếu : cú pháp và ngữ nghĩa. Cú pháp qui định trật tự kết hợp các phần tử để cấu thành 1 lệnh (câu), còn ngữ nghĩa cho biết ý nghĩa của lệnh đó.
- Bất kỳ công việc (bài toán) ngoài đời nào cũng có thể được chia thành trình tự nhiều công việc nhỏ hơn. Trình tự các công việc nhỏ này được gọi là giải thuật giải quyết công việc ngoài đời. Mỗi công việc nhỏ hơn cũng có thể được chia nhỏ hơn nữa nếu nó còn phức tạp,... $\Rightarrow$ công việc ngoài đời có thể được miêu tả bằng 1 trình tự các lệnh máy (chương trình ngôn ngữ máy).

Định nghĩa sơ khởi về máy tính số (tt)

- Vấn đề mấu chốt của việc dùng máy tính giải quyết công việc ngoài đời là lập trình (được hiểu nôm na là qui trình xác định trình tự đúng các lệnh máy để thực hiện công việc). Cho đến nay, lập trình là công việc của con người (với sự trợ giúp ngày càng nhiều của máy tính).
- Với công nghệ phần cứng hiện nay, ta chỉ có thể chế tạo các máy tính mà tập lệnh máy rất sơ khai, mỗi lệnh máy chỉ có thể thực hiện 1 công việc rất nhỏ và đơn giản ⇒ công việc ngoài đời thường tương đương với trình tự rất lớn (hàng triệu) các lệnh máy ⇒ Lập trình bằng ngôn ngữ máy rất phức tạp, tốn nhiều thời gian, công sức, kết quả rất khó bảo trì, phát triển.
- Ta muốn có máy luận lý với tập lệnh (được đặc tả bởi ngôn ngữ lập trình) cao cấp và gần gủi hơn với con người. Ta thường hiện thực máy này bằng 1 máy vật lý + 1 chương trình dịch. Có 2 loại chương trình dịch : trình biên dịch (compiler) và trình thông dịch (interpreter).

Định nghĩa sơ khởi về máy tính số (tt)

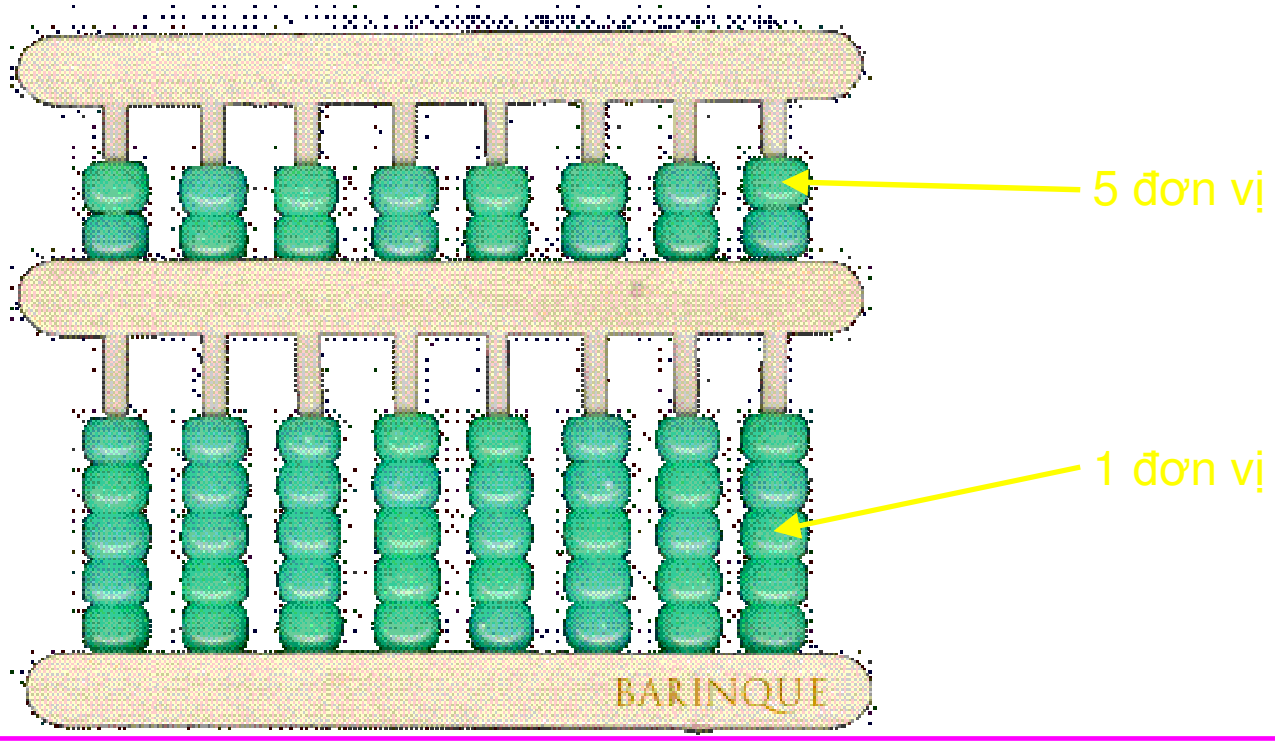
- Gọi ngôn ngữ máy vật lý là N_0. Trình biên dịch ngôn ngữ N_1 sang ngôn ngữ N_0 sẽ nhận đầu vào là chương trình được viết bằng ngôn ngữ N_1, phân tích từng lệnh N_1 rồi chuyển thành danh sách các lệnh ngôn ngữ N_0 có chức năng tương đương. Để viết chương trình dịch từ ngôn ngữ N_1 sang N_0 dễ dàng, độ phức tạp của từng lệnh ngôn ngữ N_1 không quá cao so với từng lệnh ngôn ngữ N_0.
- Sau khi có máy luận lý hiểu được ngôn ngữ luận lý N_1, ta có thể định nghĩa và hiện thực máy luận lý N_2 theo cách trên và tiếp tục đến khi ta có 1 máy luận lý hiểu được ngôn ngữ N_m rất gần gũi với con người, dễ dàng miêu tả giải thuật của bài toán cần giải quyết...
- Nhưng qui trình trên chưa có điểm dừng, với yêu cầu ngày càng cao và kiến thức ngày càng nhiều, người ta tiếp tục định nghĩa những ngôn ngữ mới với tập lệnh ngày càng gần gũi hơn với con người để miêu tả giải thuật càng dễ dàng, gọn nhẹ và trong sáng hơn.

Định nghĩa sơ khởi về máy tính số (tt)

- Ngôn ngữ máy vật lý là loại ngôn ngữ thấp nhất mà người lập trình bình thường có thể dùng được. Các lệnh và tham số của lệnh được miêu tả bởi các số binary (hay hexadecimal - sẽ được miêu tả chi tiết trong chương 2). Đây là loại ngôn ngữ mà máy vật lý có thể hiểu trực tiếp, nhưng con người thì gặp nhiều khó khăn trong việc viết và bảo trì chương trình ở cấp này.
- Ngôn ngữ assembly rất gần với ngôn ngữ máy, những lệnh cơ bản nhất của ngôn ngữ assembly tương ứng với lệnh máy nhưng được biểu diễn dưới dạng gợi nhớ. Ngoài ra, người ta tăng cường thêm khái niệm "lệnh macro" để nâng sức mạnh miêu tả giải thuật.
- Ngôn ngữ cấp cao theo trường phái lập trình cấu trúc như Pascal, C,... Tập lệnh của ngôn ngữ này khá mạnh và gần với tư duy của người bình thường.
- Ngôn ngữ hướng đối tượng như C++, Visual Basic, Java, C#,... cải tiến phương pháp cấu trúc chương trình sao cho trong sáng, ổn định, dễ phát triển và thay thế linh kiện.

1.2 Lịch sử phát triển máy tính số

- Máy tính xuất hiện từ rất lâu theo nhu cầu buôn bán và trao đổi tiền tệ.
- Bàn tính tay abacus là dạng sơ khai của máy tính.



Các thế hệ máy tính số

Blaise Pascal (Pháp-1642)

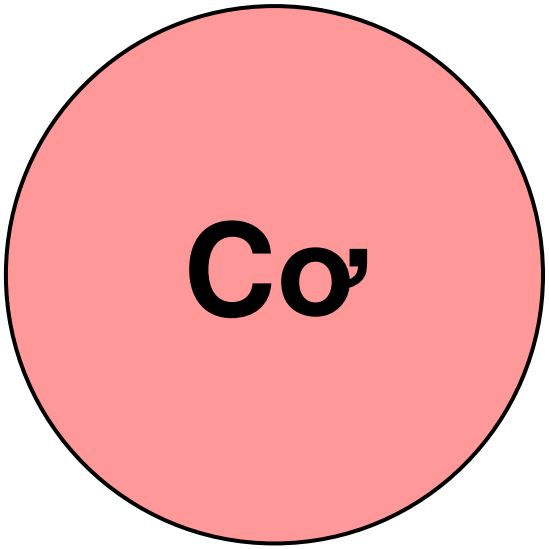
Charles Babbage (Anh-1830)

ENIAC (1946)
18.000 bóng đèn
1500 rờ le
30 tấn
140 KW

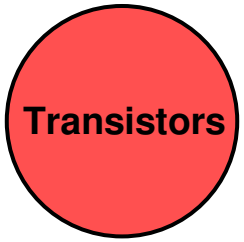
Intel 8080 (1974)
được xem như CPU đầu tiên được tích hợp trên 1 chip

IBM 360 (1965)

Von Neumann (1945)



PDP-1 (1961)



80x86 (1978)

(1642 - 1945)

(1945 - 1955)

(1955 - 1965)

(1965 - 1980)

(1980 - ????)

Herman Hollerith lập IBM (International Business Machine) ở Mỹ - 1890

Bộ nhớ dây trễ, tĩnh điện. Giấy, phiếu đục lỗ. Băng từ

Bộ nhớ xuyến từ. Băng từ, trống từ, đĩa từ.

1.3 Hệ thống số đếm

Hệ thống số (number system) là công cụ để biểu thị đại lượng. Một hệ thống số gồm 3 thành phần chính :

1. **cơ số** : số lượng ký số (ký hiệu để nhận dạng các số cơ bản).
2. **qui luật kết hợp các ký số** để miêu tả 1 đại lượng nào đó.
3. **các phép tính cơ bản** trên các số.

Trong 3 thành phần trên, chỉ có thành phần 1 là khác nhau giữa các hệ thống số, còn 2 thành phần 2 và 3 thì giống nhau giữa các hệ thống số.

Thí dụ :
- hệ thống số thập phân (**hệ thập phân**) dùng 10 ký số : 0,1,2,3,4,5,6,7,8,9.
- **hệ nhị phân** dùng 2 ký số : 0,1.
- hệ bát phân dùng 8 ký số : 0,1,2,3,4,5,6,7.
- **hệ thập lục phân** dùng 16 ký số : 0 đến 9,A,B,C,D,E,F.

Hệ thống số đếm - Cơ số

- Trước khi có máy tính, con người dùng hệ số đếm thập phân (10).

Thập phân (decimal)

Ký số

0 1 2 3 4
5 6 7 8 9

Quy tắc đếm

0 → 1 → 2 → . . . → 9 →

10 → 11 → 12 → . . . → 19 →

20 → 21 → 22 → . . . → 29 → . . . → 90 → 91 → 92 → . . . → 99 →

100 → 101 → . . . → 109 → . . . → 990 → 991 → . . . → 999 →

1000 → 1001 → 1002 → . . . → 1009 →

→ . . .

Hệ thống số đếm - Cơ số

- Sau khi máy tính số ra đời, các hệ số mới hình thành.

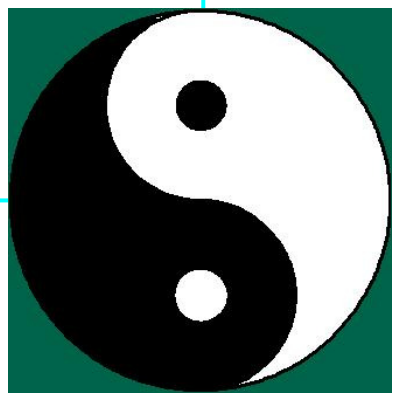
Hệ nhị phân (Binary)

Ký số

0 1

Quy tắc đếm

0 → 1 →
10 → 11 →
100 → 101 → 110 → 111 →
1000 → 1001 → . . . → 1110 → 1111 →
10000 → 10001 →
→ . . .



Hệ thống số đếm - Cơ số

- Số ở hệ nhị phân dài, khó nhớ, nhưng phần cứng máy tính chỉ hiểu trực tiếp hệ nhị phân.
- Con người dùng số hệ bát phân (8) và thập lục phân (16) thay cho hệ nhị phân (dạng tốc ký hay rút gọn của hệ nhị phân).

Hệ bát phân (Octal)

Ký số

0 1 2 3
4 5 6 7

Quy tắc đếm

0 → 1 → 2 → . . . → 7 →
10 → 11 → 12 → . . . → 17 →
20 → 21 → 22 → . . . → 77 →
100 → 101 → 102 → . . . → 107 → . . . → 777 →
1000 → 1001 → 1002 → . . . → 1007 →
→ . . .

Hệ thống số đếm - Cơ số

- Một ký số hệ 8 bằng 3 ký số hệ 2 ($2^3 = 8$).
- Một ký số hệ 16 bằng 4 ký số hệ 2 ($2^4 = 16$).

Hệ thập lục phân (hexadecimal)

Ký số

0 1 2 3 4 5 6 7
8 9 A B C D E F

Quy tắc đếm

0 → 1 → 2 → . . . → 9 → A → B → . . . → F →
10 → 11 → 12 → . . . → 19 → 1A → . . . → 1F → 20 → . . . → 9F →
A0 → A1 → A2 → . . . → AF → . . . → F0 → F1 → F2 → . . . → FF →
100 → 101 → 102 → . . . → 10F → . . . → FFF →
1000 → 1001 → 1002 → . . . → 100F →
→ . . .

Hệ thống số đếm - Qui luật miêu tả lượng

Biểu diễn của lượng Q trong hệ thống số B (B>1) là :

$d_n d_{n-1} ... d_1 d_0 d_{-1} ... d_{-m} \Leftrightarrow$

$Q = d_n * B^n + d_{n-1} * B^{n-1} + ... + d_0 * B^0 + d_{-1} * B^{-1} + ... + d_{-m} * B^{-m}$

trong đó mỗi d_i là 1 ký số trong hệ thống B.

Trong thực tế lập trình bằng ngôn ngữ cấp cao, ta thường dùng hệ thống số thập phân để miêu tả dữ liệu số của chương trình (vì đã quen). Chỉ trong 1 số trường hợp đặc biệt, ta mới dùng hệ thống số nhị phân (hay thập lục phân) để miêu tả 1 vài giá trị nguyên, trong trường hợp này, qui luật biểu diễn của lượng nguyên Q trong hệ thống số B sẽ đơn giản là :

$d_n d_{n-1} ... d_1 d_0 \Leftrightarrow$

$Q = d_n * B^n + d_{n-1} * B^{n-1} + ... + d_1 * B^1 + d_0 * B^0$

trong đó mỗi d_i là 1 ký số trong hệ thống B.

Ví dụ số nguyên

1011_2

$1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 8+0+2+1 = 11_{10}$

173_8

$1\times8^2 + 7\times8^1 + 3\times8^0 = 64+56+3 = 123_{10}$

$A4B5_{16}$

$A\times16^3 + 4\times16^2 + B\times16^1 + 5\times16^0$

$10\times4096 + 4\times256 + 11\times16 + 5\times1 = 40960+1024+176+5$

$= 42165$

Ví dụ số lẻ

1011.01_2

$1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$

$1\times8 + 0\times4 + 1\times2 + 1\times1 + 0\times0.5 + 1\times0.25 = 11.25_{10}$

10.4_8

$1\times8^1 + 0\times8^0 + 4\times8^{-1}$

$1\times8 + 0\times1 + 4\times0.125 = 8.5_{10}$

Các phương pháp chuyển miêu tả số

Để chuyển 1 miêu tả số từ hệ thống số này sang hệ thống số khác, ta cần dùng 1 phương pháp chuyển thích hợp. Có 4 phương pháp sau tương ứng với từng yêu cầu chuyển tương ứng :

1. chuyển từ hệ thống số khác về thập phân.
2. chuyển từ nhị phân về thập lục phân (hay bát phân).
3. chuyển từ thập lục phân (hay bát phân) về nhị phân.
4. chuyển từ hệ thống số thập phân về hệ thống số khác.

Chuyển từ hệ thống nhị phân về thập lục phân

Để chuyển 1 miêu tả số từ hệ thống số khác (nhị phân, thập lục phân hay bát phân) sang hệ thập phân, ta dùng công thức tính Q.

Thí dụ :

1. $1A2_H = 1*16^2+10*16^1+2*16^0 = 256+160+2 = 418_D$

2. $642_O = 6*8^2+4*8^1+2*8^0 = 384+32+2 = 418_D$

3. $110100010_B = 2^8 + 2^7+2^5+2^1 = 256+128+32+2 = 418_D$

Chuyển từ hệ thống nhị phân về thập lục phân

Lưu ý rằng có 1 mối quan hệ mật thiết giữa hệ nhị phân và thập lục phân (hay bát phân), đó là 4 ký số nhị phân tương đương với 1 ký số thập lục phân (hay 3 ký số nhị phân tương đương với 1 ký số bát phân) theo bảng tương đương.

Dec	Hex	Oct	Binary
0	0	00	0000
1	1	01	0001
2	2	02	0010
3	3	03	0011
4	4	04	0100
5	5	05	0101
6	6	06	0110
7	7	07	0111

Dec	Hex	Oct	Binary
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

Chuyển từ hệ thống nhị phân về thập lục phân

Để đổi 1 số nhị phân về thập lục phân (hay bát phân), ta đi từ phải sang trái và chia thành từng nhóm 4 ký số nhị phân (hay 3 ký số nhị phân), sau đó đổi từng nhóm 4 ký số (hay 3 ký số) thành 1 ký số thập lục phân tương đương (hay 1 ký số bát phên tương đương).

Thí dụ :

1. $\overleftarrow{110100010}_B = 0001.1010.0010 = 1A2_H$
2. $\overleftarrow{110100010}_B = 110.100.010 = 642_O$

Chuyển từ hệ thống thập lục phân về nhị phân

Để đổi 1 số thập lục phân (hay bát phân) về nhị phân, ta đổi từng ký số thập lục phân (hay bát phân) thành từng nhóm 4 ký số nhị phân (hay 3 ký số nhị phân).

Thí dụ :

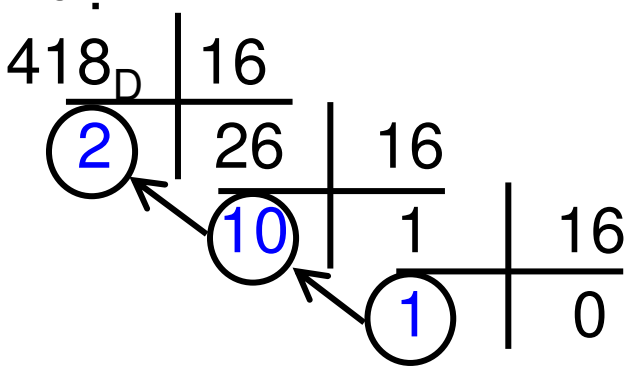
1. $1A2_H = 0001.1010.0010 = 110100010_B$

2. $642_O = 110.100.010 = 110100010_B$

Chuyển từ hệ thống thập phân về hệ thống khác

Để đổi 1 số thập phân về hệ thống số khác, ta hãy chia số cần đổi cho cơ số đích để có được thương và dư số, ta lặp lại hoạt động chia thương số cho cơ số đích để có được thương và dư số mới, cứ thế lặp mãi cho đến khi thương số = 0 thì dừng lại. Ghép các dư số theo chiều ngược chiều lặp để tạo ra kết quả (đó là sự miêu tả số tương đương nhưng ở hệ thống số khác.

Thí dụ :



Kết quả là $418_D = 1A2_H$

Chuyển từ hệ thống thập lục phân về bát phân

Để đổi 1 số thập lục phân về bát phân (hay ngược lại), ta nên chuyển tuần tự từ thập lục phân về nhị phân, rồi từ nhị phân về bát phân.

Hệ thống số đếm - Các phép tính

Các phép tính cơ bản trong 1 hệ thống số là :

1. phép cộng (+).
2. phép trừ (-).
3. phép chia (/).
4. phép nhân (*).
5. phép dịch trái n ký số (<< n).
6. phép dịch phải n ký số (>> n).

Ngoài ra do đặc điểm của hệ nhị phân, hệ này còn cung cấp 1 số phép tính sau (các phép tính luận lý) :

1. phép OR bit (|).
2. phép AND bit (&).
3. phép XOR bit (^).
4. ....

Thí dụ về phép cộng, trừ, nhân

Thí dụ về các phép tính cơ bản (các giá trị đều được biểu diễn bằng hệ nhị phân :

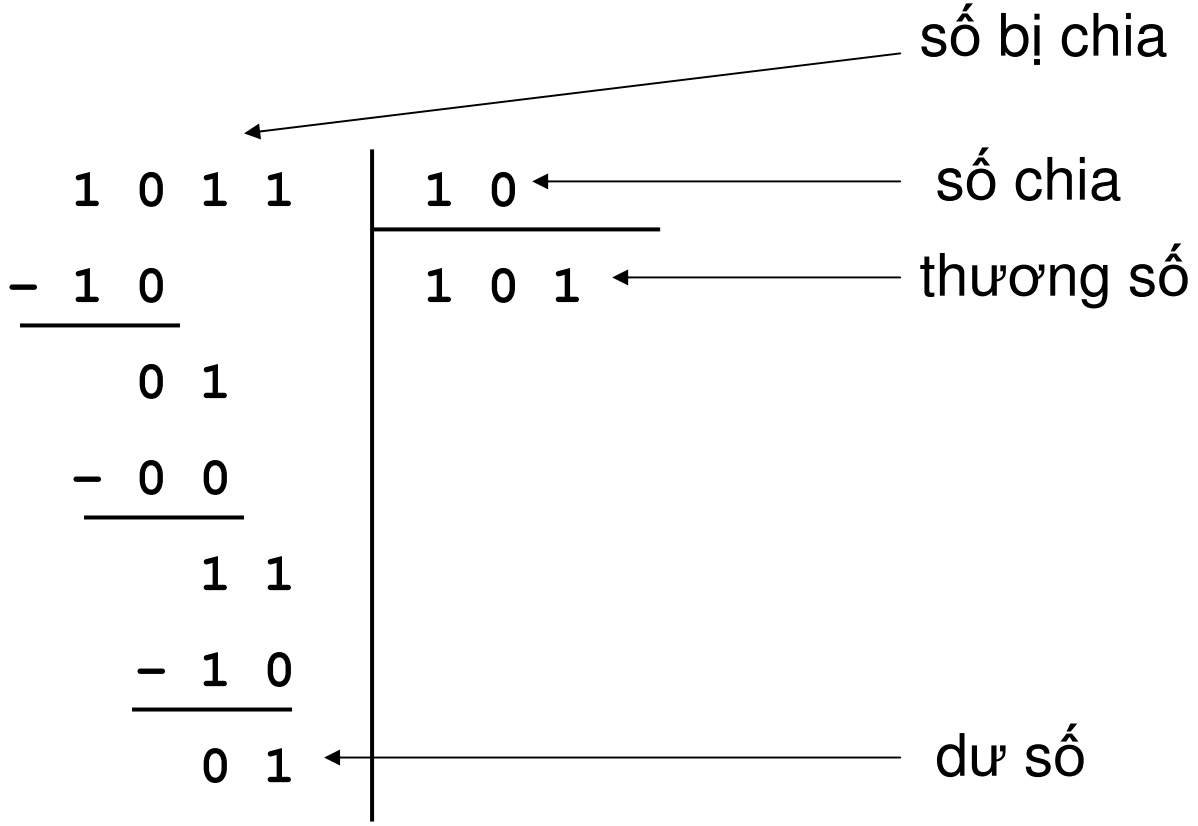
```
  0 1 1 0
+ 0 0 1 1
---------
  1 0 0 1
```

```
  1 0 0 1
- 0 0 1 1
---------
  0 1 1 0
```

```
    1 0 0 1
  * 0 1 0 1
-----------
    1 0 0 1
  0 0 0 0
1 0 0 1
-------------
1 0 1 1 0 1
```

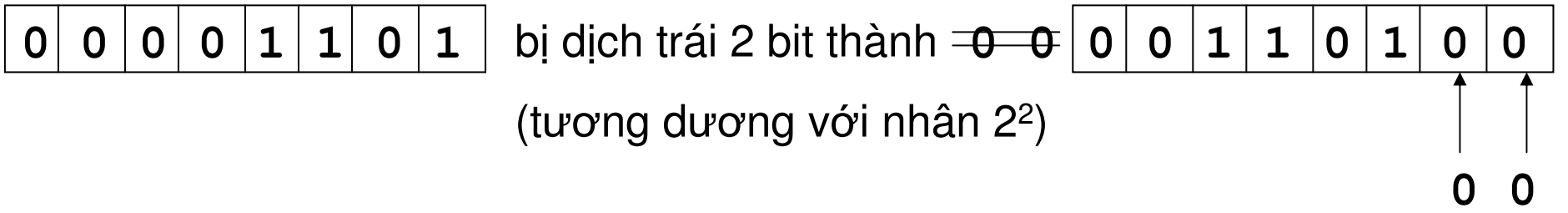
Thí dụ về phép chia

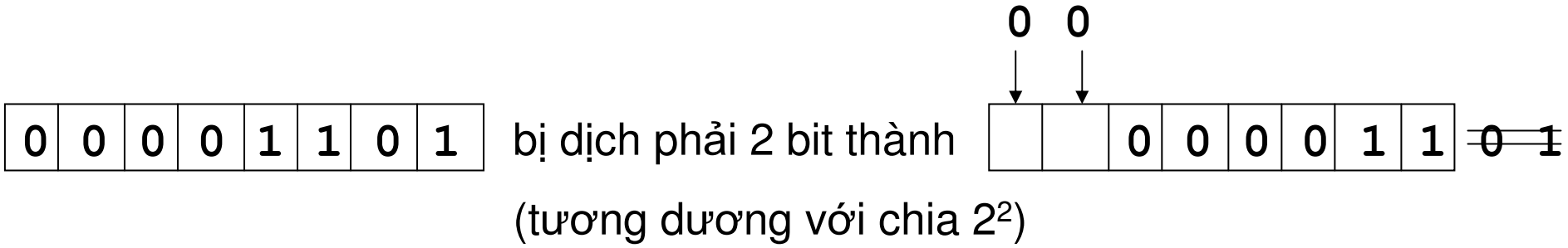
Thí dụ về các phép tính cơ bản (các giá trị đều được biểu diễn bằng hệ nhị phân) :



Thí dụ về phép dịch ký số

Thí dụ về các phép tính dịch ký số (các giá trị đều được biểu diễn bằng hệ nhị phân) :





Các phép tính của đại số Boole

Đại số Boole nghiên cứu các phép toán thực hiện trên các biến chỉ có 2 giá trị 0 và 1, tương ứng với hai thái cực luận lý "sai" và "đúng" (hay "không" và "có") của đời thường. Các phép toán này gồm :

x	y	**not** x	x **and** y	x **nand** y	x **or** y	x **nor** y	x **xor** y
0	0	1	0	1	0	1	0
0	1		0	1	1	0	1
1	0	0	0	1	1	0	1
1	1		1	0	1	0	0

Biểu thức Boole là 1 biểu thức toán hoc cấu thành từ các phép toán Boole trên các toán hạng là các biến chỉ chứa 2 trị 0 và 1.

Hàm Boole

Một hàm Boole theo n biến boole (hàm n ngôi) là 1 biểu thức boole cấu thành từ các phép toán Boole trên các biến boole.

Thay vì miêu tả hàm boole bằng biểu thức boole, ta có thể miêu tả hàm boole bằng bảng thực trị. Bảng thực trị của hàm boole n biến có 2^n hàng, mỗi hàng miêu tả 1 tổ hợp trị cụ thể của các biến và giá trị cụ thể của hàm tương ứng với tổ hợp trị này (xem slide ngay trước).

Như vậy 1 hàm boole n biến được miêu tả như 1 chuỗi 2^n bit $\Rightarrow$ có chính xác 2^{2^n} hàm boole n ngôi khác nhau. Cụ thể có :

$2^{2^1} = 4$ hàm boole 1 ngôi khac nhau

$2^{2^2} = 2^4 = 16$ hàm boole 2 ngôi khac nhau

$2^{2^3} = 2^8 = 256$ hàm boole 3 ngôi khac nhau

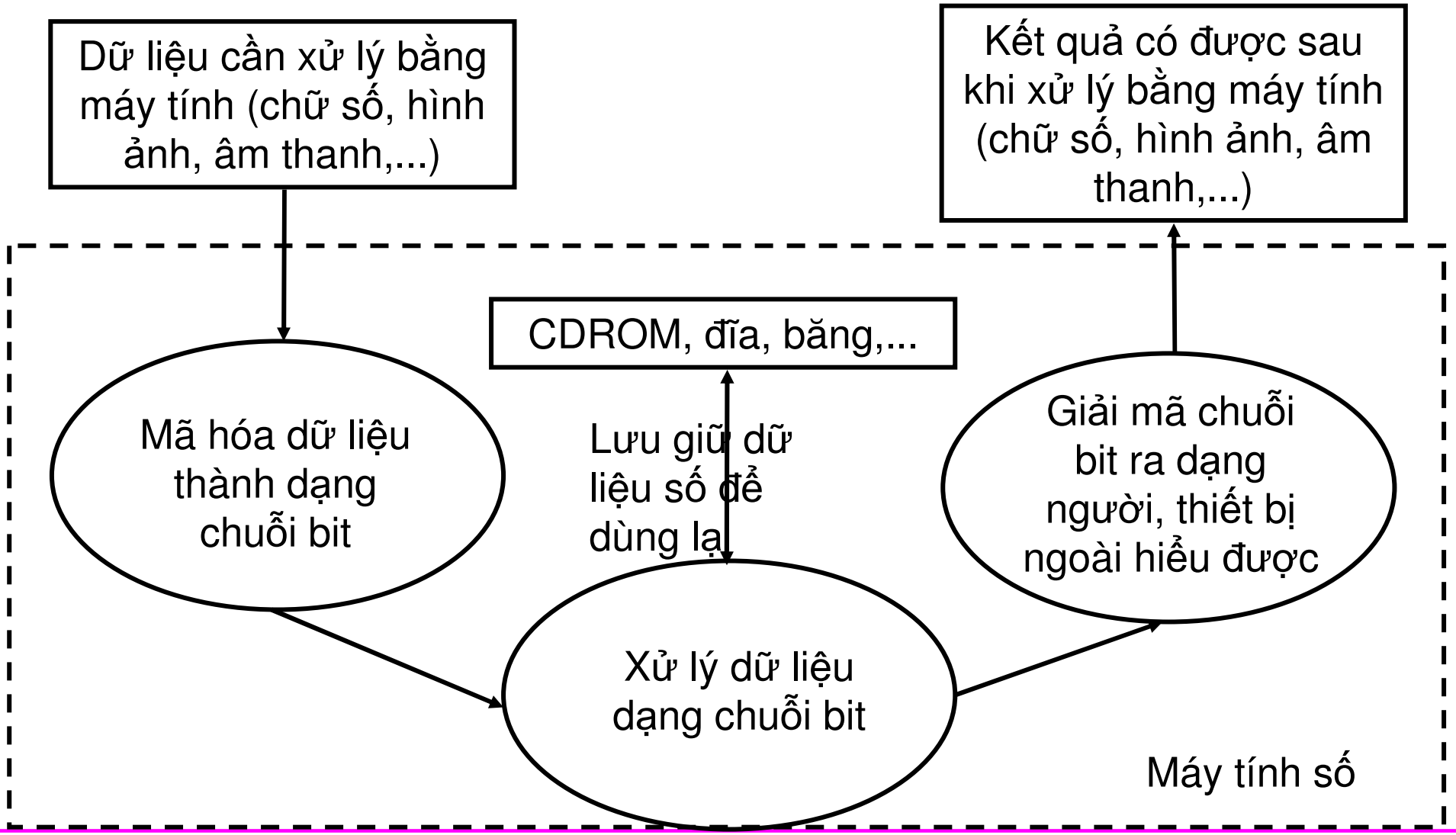
1.4 Biểu diễn dữ liệu

Máy tính dùng trực tiếp hệ nhị phân, các đơn vị biểu diễn thông tin thường dùng là :

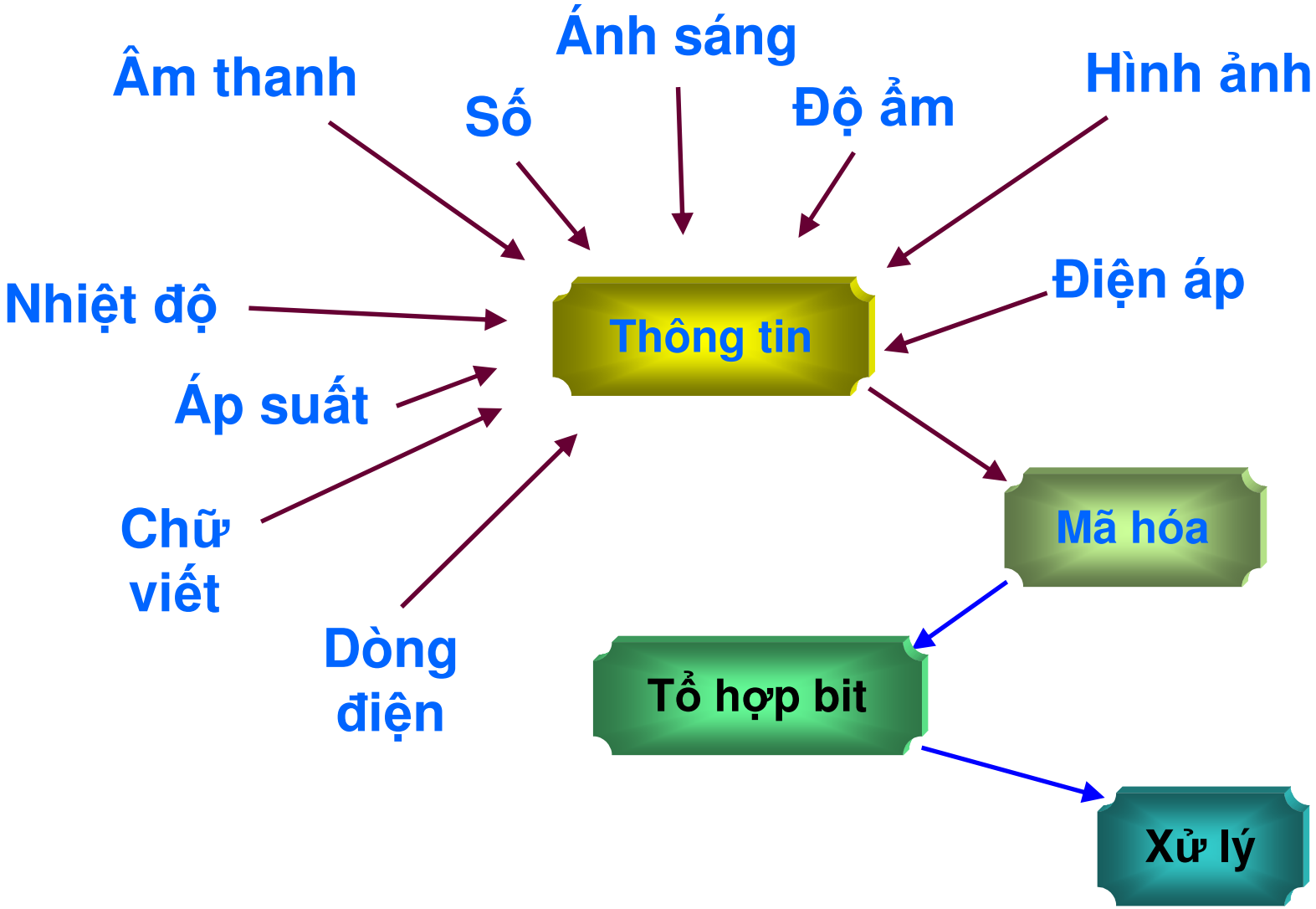
1. **bit :** miêu tả 2 giá trị khác nhau (đúng/sai, 0/1,..)
2. **byte** : 8bit, có thể miêu tả được $2^8 = 256$ giá trị khác nhau.
3. **word :** 2 byte, có thể miêu tả được $2^{16} = 65536$ giá trị khác nhau.
4. **double word** : 4 byte, có thể miêu tả được $2^{32} =$ 4.294.967.296 giá trị khác nhau.
5. **KB (kilo byte)** = $2^{10} = 1024$ byte.
6. **MB (mega byte)** = $2^{20} = 1024$KB = 1.048.576 byte.
7. **GB (giga byte)** = $2^{30} = 1024$MB = 1.073.741.824 byte.
8. **TB (tetra byte)** = $2^{40} = 1024$GB = 1.099.511.627.776 byte.

Thí dụ, RAM của máy bạn là 512MB, đĩa cứng là 320GB.

Qui trình tổng quát để giải quyết bài toán bằng máy tính số



Mã hóa thông tin đầu vào



Biểu diễn số

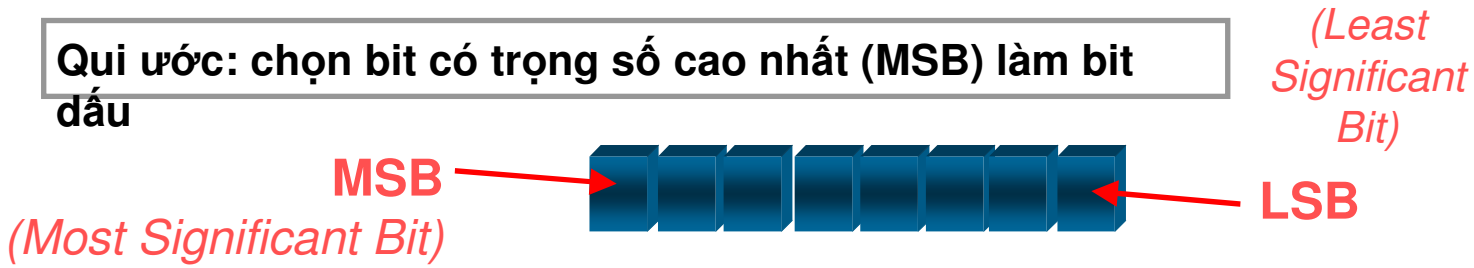
Số không dấu

Số n bit có giá trị : 0 ÷ (2^n – 1)

Số 8 bit có giá trị : 0 ÷ 255
Số 16 bit có giá trị : 0 ÷ 65 535
Số 32 bit có giá trị : 0 ÷ 4 294 967 295

Số có dấu

Qui ước: chọn bit có trọng số cao nhất (MSB) làm bit dấu



Số 8 bit có dấu có giá trị : -128 ÷ +127
Số 16 bit có dấu có giá trị: -32768 ÷ +32767

Biểu diễn số nguyên có dấu trong máy

- Phần dương có 32768 số từ số 0 tới 32767, được miêu tả theo công thức Q.
- Phần âm có 32768 số từ -32768 tới -1, được miêu tả ở dạng số bù 2 như sau :
- Số bù 1 của 1 số n bit là n bit mà mỗi bit là ngược với bit gốc (0 → 1 và 1 → 0)
- Số bù 2 của 1 số n bit là số bù 1 của số đó rồi tăng lên 1 đơn vị.

Sự biểu diễn	giá trị
00000000 00000000	0
00000000 00000001	1
....	
01111111 11111111	32767
10000000 00000000	-32768
10000000 00000001	-32767
....	
11111111 11111111	-1

Biểu diễn số nguyên có dấu trong máy

Vì mỗi ô nhớ máy tính chỉ chứa được 1 byte, do đó ta phải dùng nhiều ô liên tiếp (2 hay 4) để chứa số nguyên. Có 2 cách chứa các byte của số nguyên (hay dữ liệu khác) vào các ô nhớ : BE & LE.

Cách BE (Big Endian) chứa byte trọng số cao nhất vào ô nhớ địa chỉ thấp trước, sau đó lần lượt đến các byte còn lại. Cách LE (Little Endian) chứa byte trong số nhỏ nhất vào ô nhớ địa chỉ thấp trước, sau đó lần lượt đến các byte còn lại.

CPU Intel & HĐH Windows sử dụng cách LE để chứa số nguyên vào bộ nhớ (Integer và Long).

Biểu diễn số nguyên trong VB - Thí dụ

- Số 15 được miêu tả dưới dạng nhị phân 16 bit như sau :

  0000 0000 0000 1111

- Do đó, nếu dùng kiểu Integer để lưu số 15, ta dùng 16 bit như trên hay viết ngắn gọn là $000F_H$. Nếu lưu vào bộ nhớ dạng LE thì ô nhớ có địa chỉ thấp (i) chứa byte $0F_H$, và ô nhớ kế (i+1) chứa byte 00. Nếu dùng kiểu Long để lưu số 15, ta dùng 4 byte $0000000F_H$ và lưu vào bộ nhớ dạng LE tốn 4 ô nhớ với giá trị lần lượt từ địa chỉ thấp đến cao là $0F_H$, 00, 00, 00.
- Số bù 1 của 15 là 1111 1111 1111 0000, số bù 2 của 15 là

  1111 1111 1111 0001

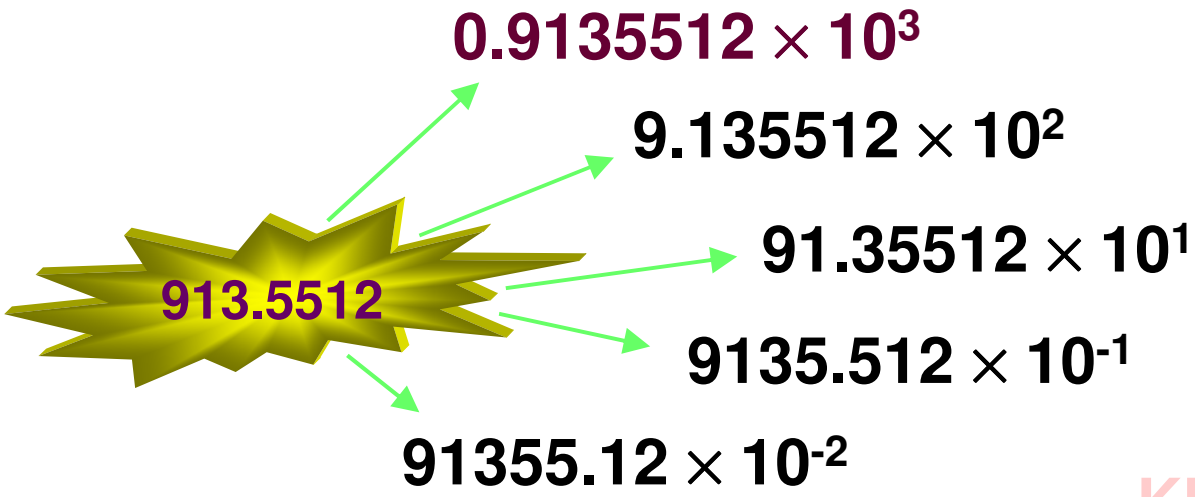
- Như vậy -15 được lưu vào máy dạng Integer là 2 byte có giá trị $FFF1_H$. Nếu lưu vào ô nhớ dạng LE thì ô nhớ có địa chỉ thấp (i) chứa byte $F1_H$, và ô nhớ kế (i+1) chứa byte FF_H.

Số thực - số chấm động

Trong khoa học, ta có thể miêu tả số thực theo dạng $\pm m*B^e$, m gọi là định trị, B là cơ số và e là số mũ. Như vậy 1 số thực cụ thể có thể được miêu tả bởi rất nhiều miêu tả khác nhau, trong đó miêu tả có $0.1 \le m < 1$ được gọi là miêu tả chuẩn tắc của số thực. Đây là miêu tả mà máy tính sẽ dùng.

$$\pm m \times B^{\pm e}$$

- **m (mantissa) quyết định độ chính xác**
- **B (base)**
- **e (exponent) quyết định độ lớn/nhỏ**



.Khó xử lý
.Cần chuẩn hóa

Số chấm động theo chuẩn IEEE 754

Trước khi lưu vào máy tính, số thực được đổi về dạng miêu tả nhị phân dưới dạng ±1.m*2e (m là chuỗi bit nhị phân miêu tả phần lẻ).
VB lưu số thực theo chuẩn IEEE 754, dùng 1 trong 2 dạng lưu :

- Chính xác đơn (Single) : VB dùng 4 byte - 4 ô nhớ (32 bit) để lưu số thực theo dạng thức cụ thể sau :

trong đó bit S = 1 (âm), =0 (dương).
M = m & E = 127 + e

- Chính xác kép (Double) : VB dùng 8 byte - 8 ô nhớ (64 bit) để lưu số thực theo dạng thức cụ thể sau :

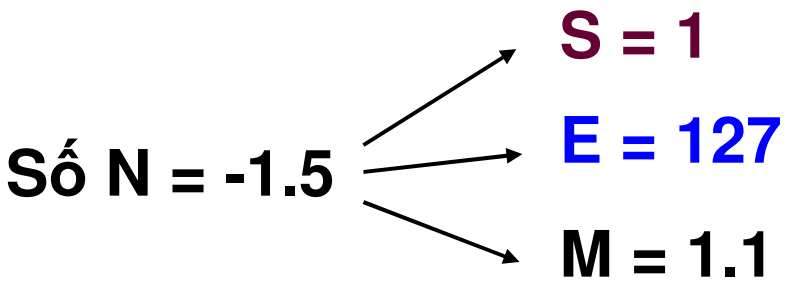
trong đó bit S = 1 (âm), =0 (dương); M = m & E = 1023 + e

Số chấm động - Ví dụ

Thí dụ giá trị -1.5 được miêu tả dạng nhị phân là $-1.1*2^0$.

- Do đó nếu dùng kiểu Single chứa số thực -1.5, ta tốn 4 byte (32 bit) với các thành phần S = 1, M = 10...0 (22 bit 0), E = 127. Kết quả, giá trị của 4 byte miêu tả số -1.5 như sau : BF C0 00 00
- Tương tự, nếu dùng kiểu Double chứa số thực -1.5, ta tốn 8 byte (64 bit) với các thành phần S = 1, M = 10...0 (51 bit 0), E = 1023. Kết quả, giá trị của 8 byte miêu tả số -1.5 như sau : BF F8 00 00 00 00 00 00.
- VB dùng cách chứa LE, do đó giá trị -1.5 được lưu vào bộ nhớ theo kiểu Single sẽ chiếm 4 byte theo giá trị lần lượt từ địa chỉ thấp đến cao là 00 00 C0 BF. Tương tự nếu miêu tả -1.5 vào bộ nhớ theo kiểu Double thì sẽ cần 8 ô nhớ với giá trị lần lượt từ địa chỉ thấp đến cao là 00 00 00 00 00 00 F8 BF.

Ví dụ số chấm động



$$(-1)^1 2^{127-127}(1.1)$$

N = 10111111100000000000000000000

Giới hạn

$$(-1)^0 2^{-127}(1.0) \div 2^{+127}(2-2^{-23})$$

nghĩa là

1.401298E-45 to 3.402823E38

Biểu diễn chuỗi ký tự

Chuỗi ký tự là danh sách nhiều ký tự, mỗi ký tự được miêu tả trong máy bởi n bit nhớ :

- mã ASCII dùng 7 bit (dùng luôn 1 byte nhưng bỏ bit 8) để miêu tả 1 ký tự $\Rightarrow$ tập ký tự mà mã ASCII miêu tả được là 128.
- mã ISO8859-1 dùng 8 bit (1byte) để miêu tả 1 ký tự $\Rightarrow$ tập ký tự mà mã ISO8859-1 miêu tả được là 256.
- mã Unicode trên Windows dùng 16 bit (2 byte) để miêu tả 1 ký tự $\Rightarrow$ tập ký tự mà mã Unicode trên Windows miêu tả được là 65536.
- ...

Hiện có nhiều loại mã tiếng Việt khác nhau, đa số dùng mã ISO8859-1 rồi qui định lại cách hiển thị 1 số ký tự thành ký tự Việt. Riêng Unicode là bộ mã thống nhất toàn cầu, trong đó có đủ các ký tự Việt.

Bảng mã ASCII 7 bit

Mã ASCII dùng các giá trị (mã) từ 0 - 127 để miêu tả các ký tự :

- mã từ 0 - 31 là các mã điều khiển như CR=13 (Carriage Return), LF=10 (Line Feed), ESC=27 (Escape)...
- mã 32 miêu tả ký tự trống, 33 miêu tả ký tự !,... theo bảng sau :

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Bảng mã ISO8859-1 (8 bit)

Mã ISO8859-1 dùng các giá trị (mã) từ 0 - 255 để miêu tả các ký tự (128 mã ký tự đầu qui định giống như mã ASCII) :

- mã từ 0 - 31 là các mã điều khiển như CR=13 (Carriage Return), LF=10 (Line Feed), ESC=27 (Escape)...
- mã 32 miêu tả ký tự trống, 33 miêu tả ký tự !,... theo bảng sau :

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	Ž		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ		
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Bảng mã tiếng Việt ĐHBK 1 byte

Mã ĐHBK 1 byte có được bằng cách hiệu chỉnh bảng mã ISO8859-1 :

- mã từ 0 - 31 là các mã điều khiển như CR=13 (Carriage Return), LF=10 (Line Feed), ESC=27 (Escape)...
- mã 32 miêu tả ký tự trống, 33 miêu tả ký tự !,... theo bảng sau :

!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?		
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
Á	,	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	Ị	‘	’	“	”	•	–	—	˜	™	š	›	œ	Ạ	Ỹ					
Ấ	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Một phần mã tiếng Việt Unicode

Mã Unicode Windows dùng 2 byte để miêu tả 1 ký tự :

- 256 mã đầu từ 0 - 255 giống y như mã ISO8859-1.
- mã từ 256 trở đi chứa các ký tự của hầu hết các ngôn ngữ trên thế giới (quá khứ, hiện tại và tương lai).
- thí dụ sau là 1 phần mã tiếng Việt trong mã Unicode :

mã $1ea0_H$ biểu diễn ký tự Ạ

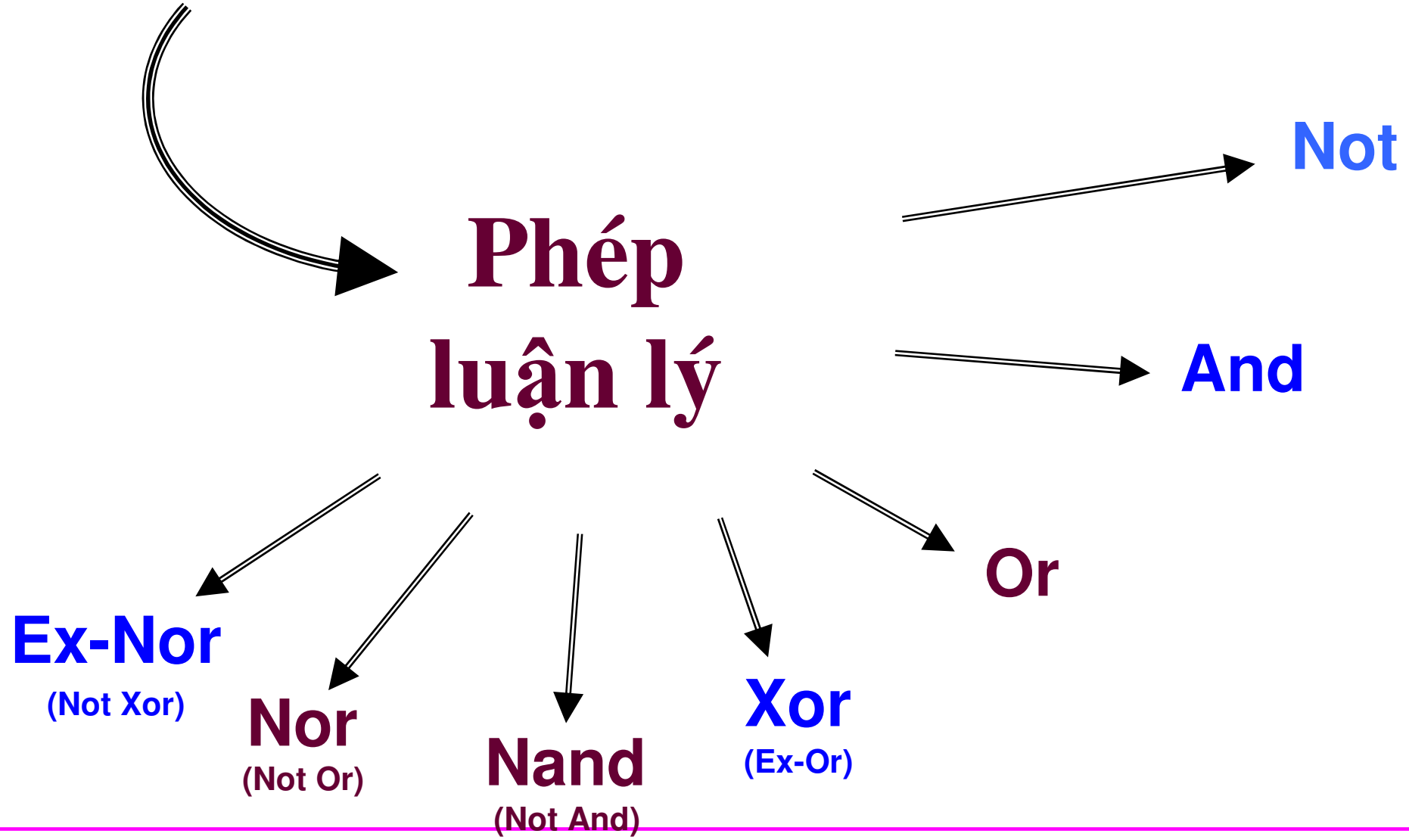
mã $1ef9_H$ biểu diễn ký tự ỹ

Ạ	ạ	Ả	ả	Ấ	ấ	Ầ	ầ	Ẩ	ẩ	Ẫ	ẫ	Ậ	ậ	Ắ	ắ	Ằ	ằ	Ẳ	ẳ	Ẵ	ẵ	Ặ	ặ	Ẹ	ẹ	Ẻ	ẻ	Ẽ	ẽ	Ế	ế
Ề	ề	Ể	ể	Ễ	ễ	Ệ	ệ	Ỉ	ỉ	Ị	ị	Ọ	ọ	Ỏ	ỏ	Ố	ố	Ồ	ồ	Ổ	ổ	Ỗ	ỗ	Ộ	ộ	Ớ	ớ	Ờ	ờ	Ở	ở
Ỡ	ỡ	Ợ	ợ	Ụ	ụ	Ủ	ủ	Ứ	ứ	Ừ	ừ	Ử	ử	Ữ	ữ	Ự	ự	Ỳ	ỳ	Ỵ	ỵ	Ỷ	ỷ	Ỹ	ỹ						

1.4 Luận lý máy tính

- Luận lý máy tính dựa trên nền tảng một nhánh của luận lý toán học được gọi là đại số Boole (George Boole).
- Biến luận lý (boolean variable) có hai giá trị, thường được biểu diễn bằng 1 và 0 (bit).
- Về mặt hiện thực, biến luận lý thể hiện trạng thái điện áp trên dây dẫn tín hiệu (1 = 5V; 0 = 0V).

Các phép toán trên đại số Boole



Phép Not

Ký hiệu dấu gạch ngang trên đầu

Bảng sự thật

x	$\overline{x}$
0	1
1	0

$x = 1011 \Rightarrow \overline{x} = 0100$

$\Rightarrow \overline{\overline{x}} = 1011 = x$

Phép And

Ký hiệu dấu chấm
như phép nhân

Bảng sự thật

x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

Nhận xét

y . 0 = 0

y . 1 = y

Phép Or

Ký hiệu dấu cộng như phép cộng

Bảng sự thật

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Nhận xét

y + 0 = y

y + 1 = 1

Ví dụ phép luận lý

Tính hàm $f(x,y) = x \cdot \overline{y} + \overline{x} \cdot y$

x	y	$\overline{x}$	$\overline{y}$	$x \cdot \overline{y}$	$\overline{x} \cdot y$	f(x,y)
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

Phép Xor (Ex-Or)

Ký hiệu dấu cộng trong vòng tròn
như phép modulo

Bảng sự thật

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Nhận xét

$y \oplus 0 = y$

$y \oplus 1 = \overline{y}$

Bảng tóm tắt

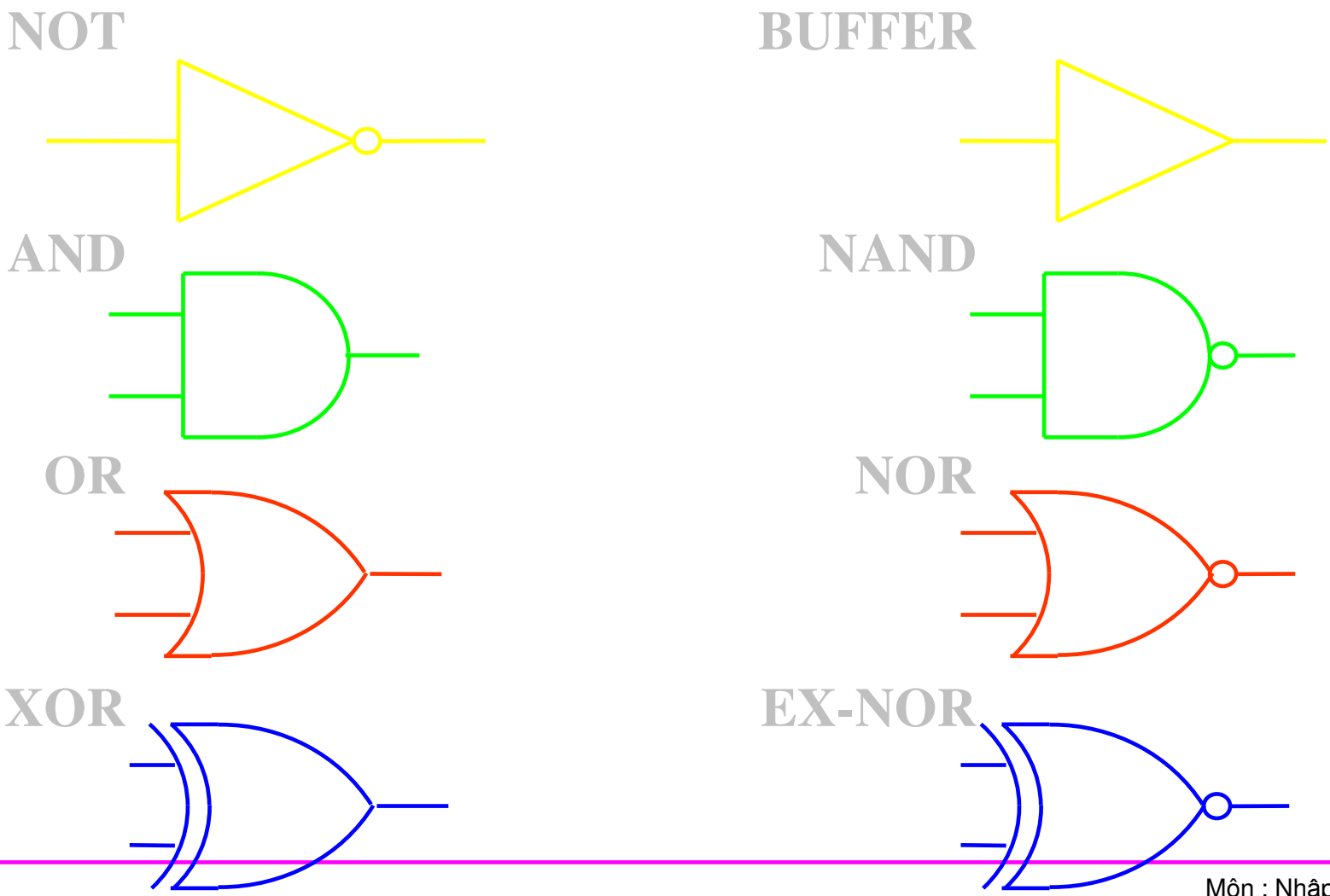
Bảng sự thật		NOT	AND	OR	XOR
x	y	not y	x and y	x or y	x xor y
0	0	1	0	0	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0

y and 0 = 0
y and 1 = y

y or 0 = y
y or 1 = 1

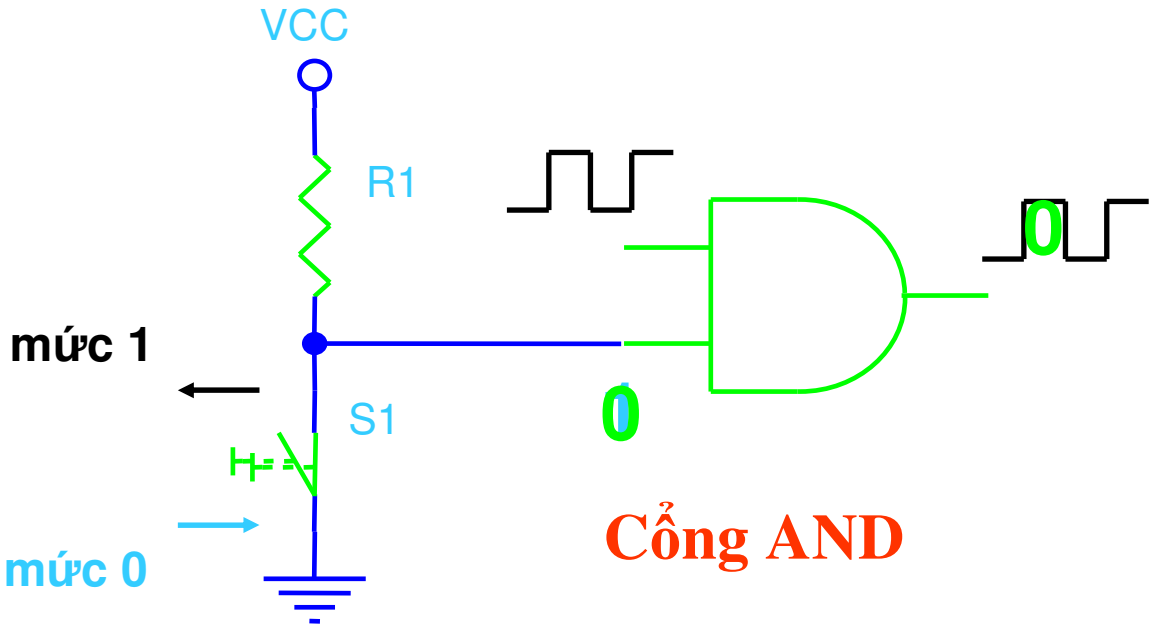
y xor 0 = y
y xor 1 = not y

Cổng luận lý



Chức năng đóng mở

mức luận lý 1 = 5V
mức luận lý 0 = 0V



Cổng AND

y and 1 = y

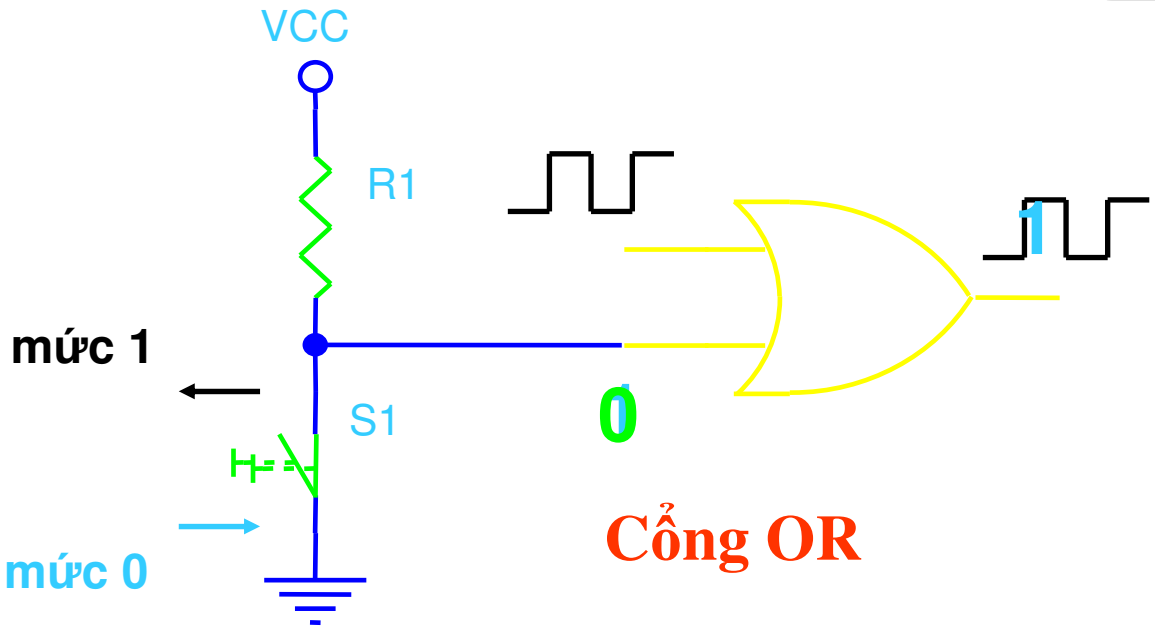
1 = mở

y and 0 = 0

0 = đóng

Chức năng đóng mở (tt.)

mức luận lý 1 = 5V
mức luận lý 0 = 0V



y or 1 = 1

1 = đóng

y or 0 = y

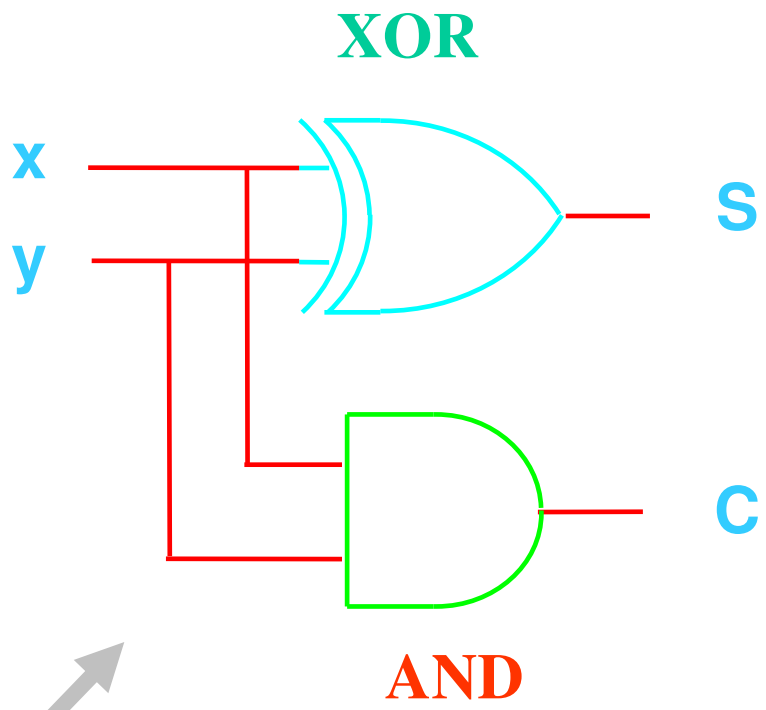
0 = mở

Ứng dụng đơn giản của cổng luận lý

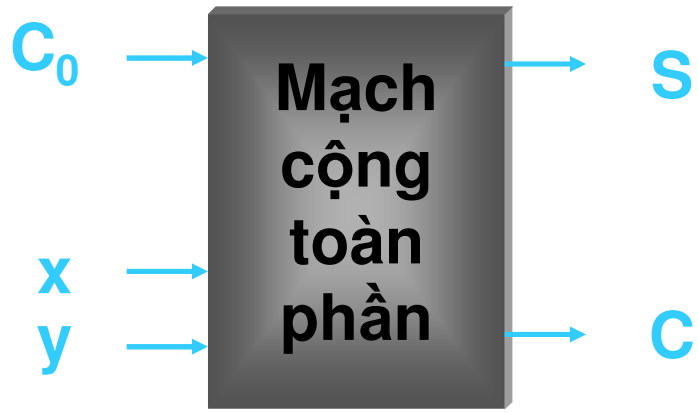
- Mạch cộng bán phần thực hiện phép cộng trên hai bit, cho ra kết quả là bit tổng S và bit nhớ C.
- Mạch cộng toàn phần cũng tương tự mạch cộng bán phần nhưng đầu vào có cộng thêm bit nhớ C_0.
- Mạch cộng toàn phần có thể được thiết kế dựa vào mạch cộng bán phần.

Mạch cộng bán phần

x	y	S XOR	C AND
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Mạch cộng toàn phần



$S = x + y + C_0$

$S = (x + y) + C_0$

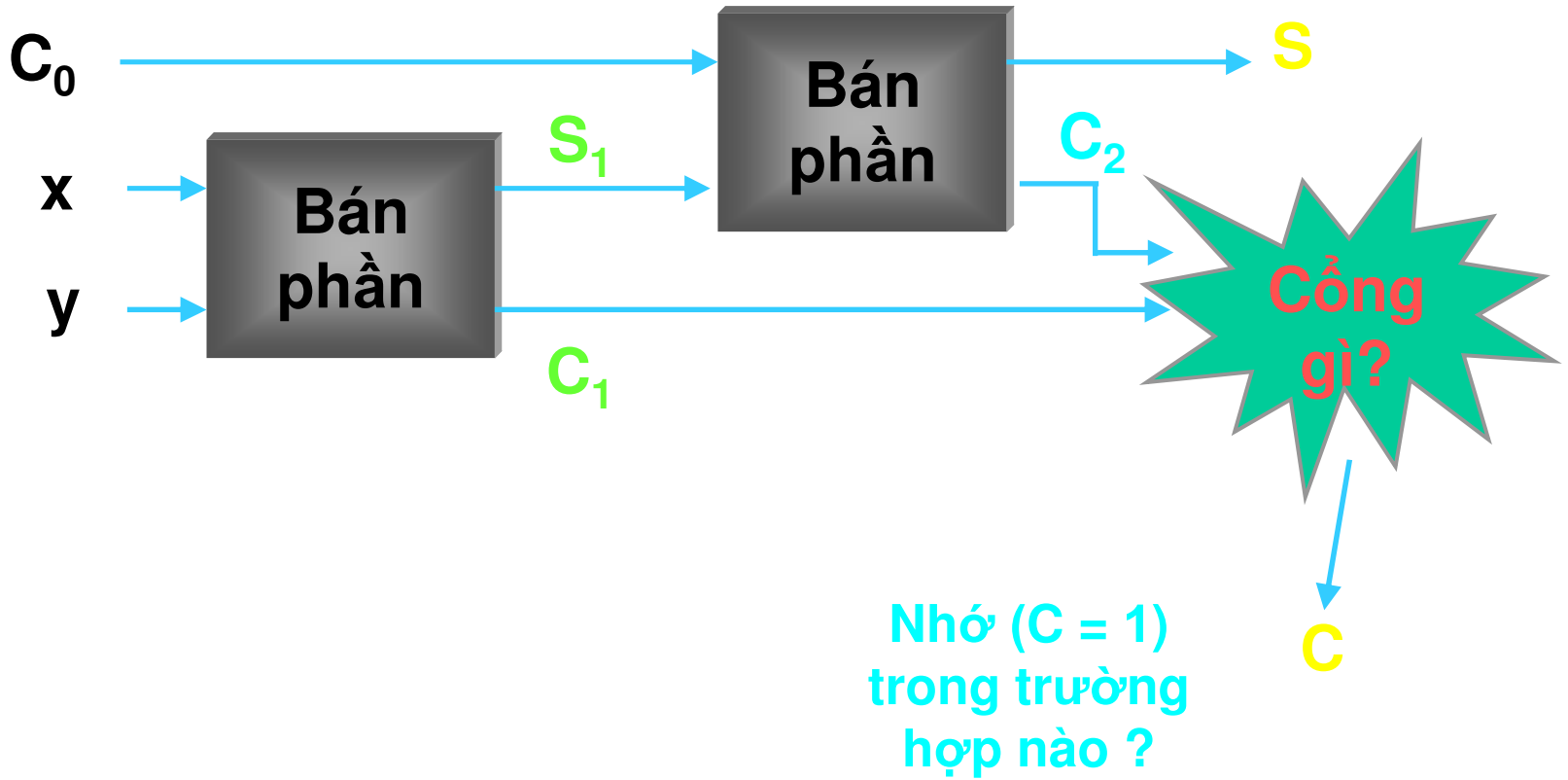
Tính: $S_1 = x + y$

Tính: $S_2 = S_1 + C_0$

Cần bộ cộng bán phần 1

Cần bộ cộng bán phần 2

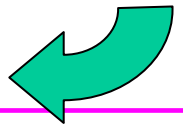
Mạch cộng toàn phần (tt.)



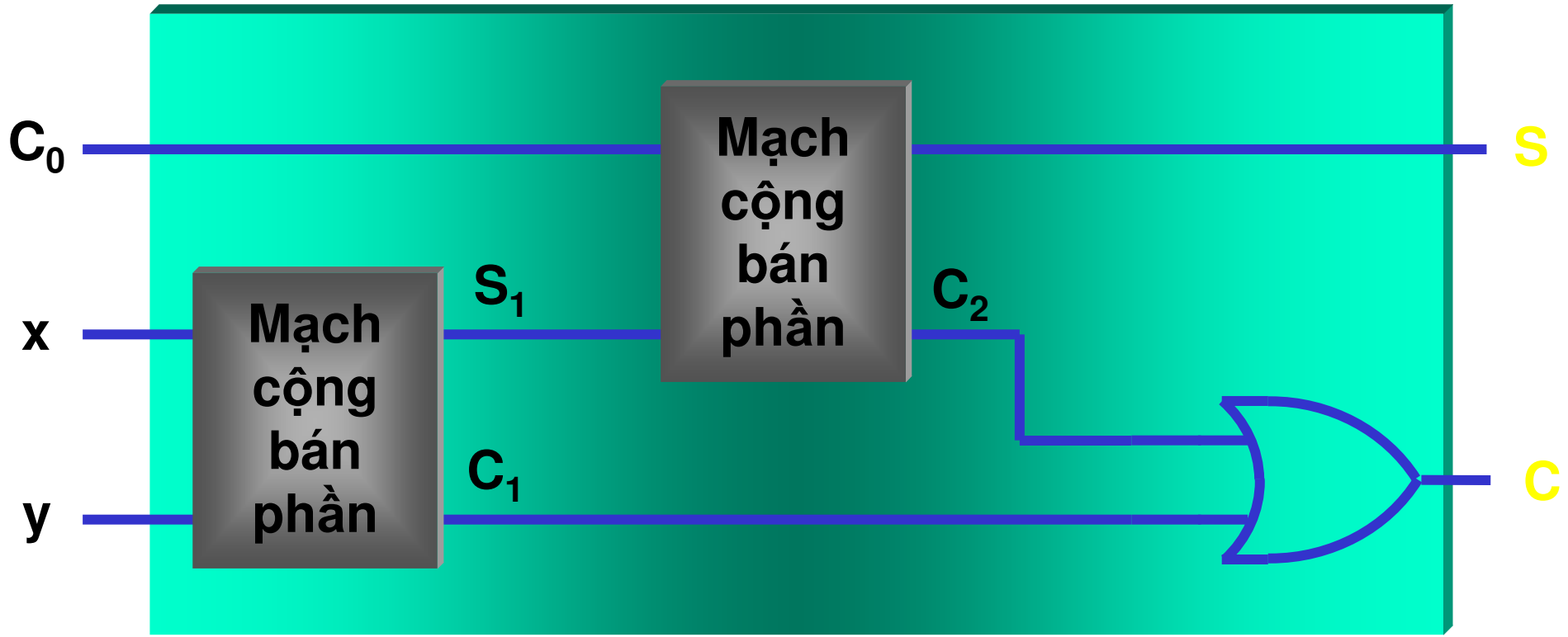
Mạch cộng toàn phần (tt.)

C_0	x	y	S	C	C_0	S_1	C_1	C_2	C
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0
0	1	0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1	0	1
1	0	0	1	0	1	0	0	0	0
1	0	1	0	1	1	1	0	1	1
1	1	0	0	1	1	1	0	1	1
1	1	1	1	1	1	0	1	0	1

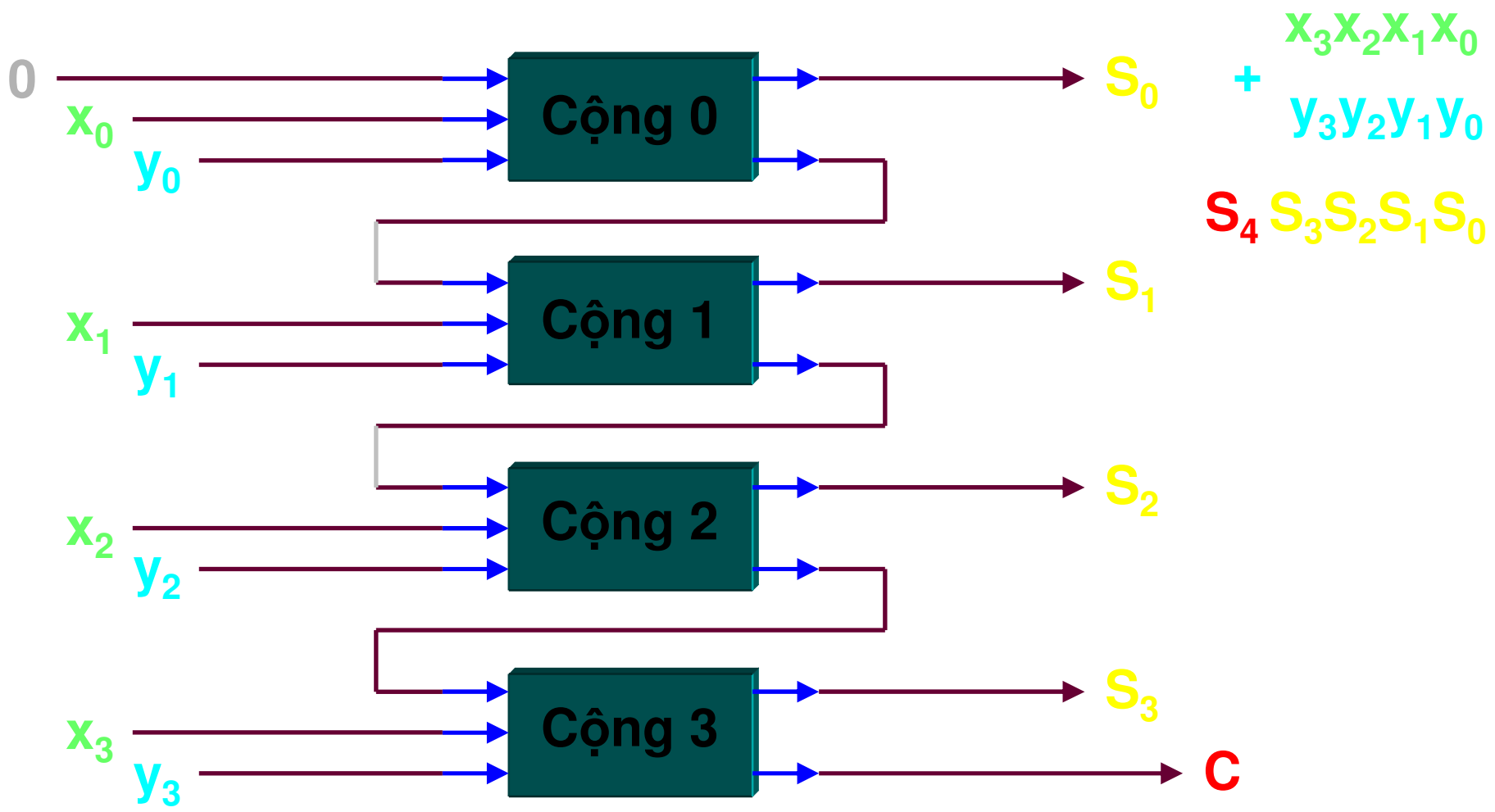
$C = 1$ khi $C_1 = 1$ hoặc $C_2 = 1$



Mạch cộng toàn phần (tt.)



Mạch cộng nhiều bit



Chương 2

PHẦN CỨNG

2.1 Hệ thống máy tính
2.2 Kiến trúc máy tính
2.3 Thiết bị xuất nhập

2.1 Hệ thống máy tính

- Hệ thống máy tính có các khối chức năng luận lý sau :
  - Khối nhập (input).
  - Bộ nhớ chính (memory).
  - Đơn vị xử lý trung tâm CPU (Central processing unit).
  - Khối xuất (output).
  - Bộ nhớ phụ (storage).
  - Thiết bị ngoại vi (peripherals).

Khối nhập - Input

- Giữ vai trò nhận dữ liệu cho máy tính.
- Có nhiệm vụ chuyển đổi các thông tin từ thế giới ngoài thành dữ liệu mà máy tính có thể xử lý.
- Có rất nhiều thiết bị có thể làm việc này nhưng bàn phím (keyboard) là thiết bị được dùng phổ biến nhất.

Bộ nhớ chính - Main memory

- Còn gọi là bộ nhớ RAM và ROM.
- Có 2 chức năng chính :
  - Chứa tạm chương trình đang được sử dụng để xử lý thông tin.
  - Chứa tạm dữ liệu.
- Dữ liệu dùng trong máy tính có 3 loại :
  - Dữ liệu ban đầu nhận từ khối nhập.
  - Dữ liệu trung gian đang được xử lý.
  - Kết quả cuối cùng chờ đưa ra khối xuất.

Đơn vị xử lý trung tâm - CPU

- Thường còn gọi là bộ xử lý (processor), vi xử lý (micro-processor).
- CPU có nhiệm vụ thi hành lệnh của chương trình và xử lý các dữ liệu trong chương trình.
- Trong CPU có 2 phần chính :
  - Đơn vị số học luận lý ALU (Arithmetic / logic unit).
  - Đơn vị điều khiển (control unit).
- ALU dùng để tính toán các phép số học (cộng, trừ, nhân, chia) và các phép luận lý (not, and, or, xor).
- Đơn vị điều khiển chi phối toàn bộ hoạt động của máy tính bằng cách lấy lệnh từ bộ nhớ, giải mã lệnh và thực hiện lệnh đó.

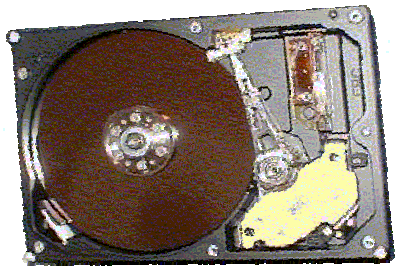
Khối xuất - Output

- Ngược lại với khối nhập, khối xuất chuyển dữ liệu mà máy xử lý (số nhị phân) ra thành dạng thông tin mà con người có thể chấp nhận.
- Hai thiết bị thông dụng dùng trong khối này là màn hình và máy in.

- Đôi khi các thông tin mà máy tính đưa ra cần được xử lý tiếp sau này nên còn phải được lưu trên bộ nhớ phụ (chủ yếu là trên đĩa từ).

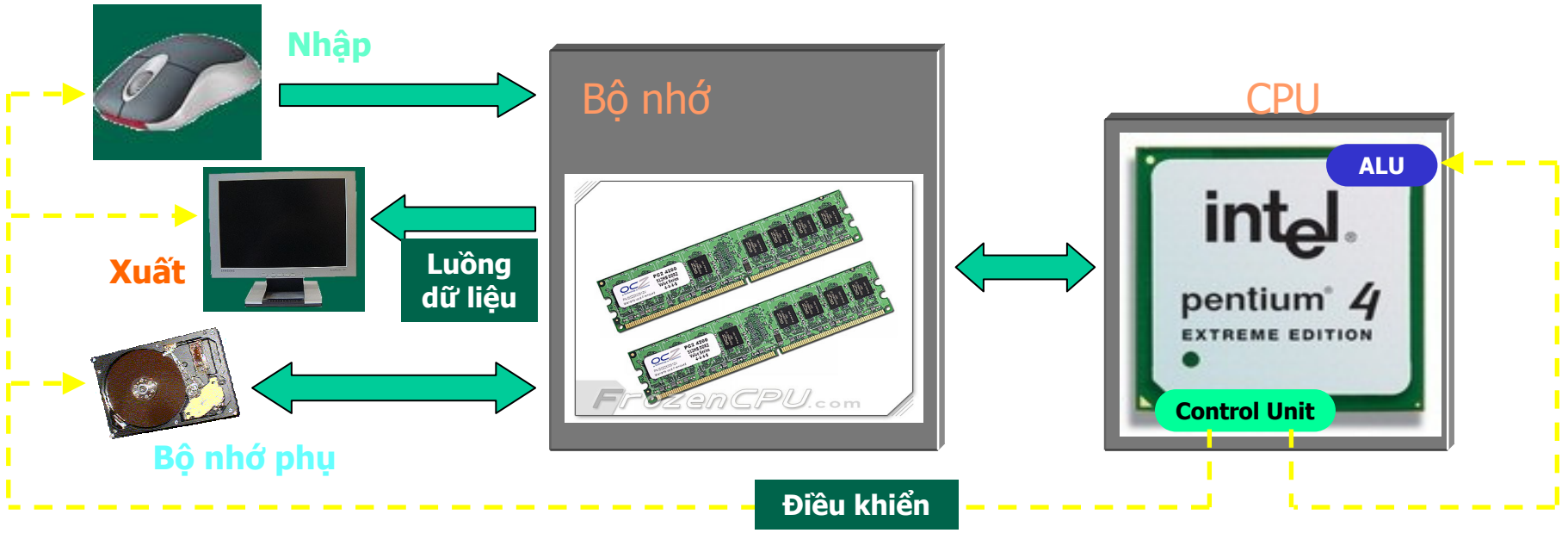
Bộ nhớ phụ - Storage

- Cung cấp cho máy tính chức năng lưu trữ, sắp xếp, phân loại thông tin theo dạng tập tin (file).
- Cần phân biệt hai khái niệm sau :
  - Bộ nhớ bốc hơi (memory volatility) : là bộ nhớ mà thông tin lưu giữ trong nó sẽ bị mất đi, hoặc là do tắt máy, hoặc là do thông tin khác ghi chồng lên. Chính vì vậy nên loại bộ nhớ này còn được gọi là RAM (Random Access Memory).
  - Dữ liệu có thể dùng lại (retrievable data) : ROM & bộ nhớ phụ có thể giữ chương trình hay dữ liệu lâu dài mà không bị bốc hơi. Điều đó cho phép ta có thể sử dụng lại các thông tin này nhiều lần.



Thiết bị ngoại vi - Peripherals

- Thiết bị ngoại vi là các thiết bị phụ trợ xung quanh CPU và bộ nhớ chính.
- Các thiết bị đáp ứng chức năng của các khối nhập, xuất và bộ nhớ phụ đều là thiết bị ngoại vi.

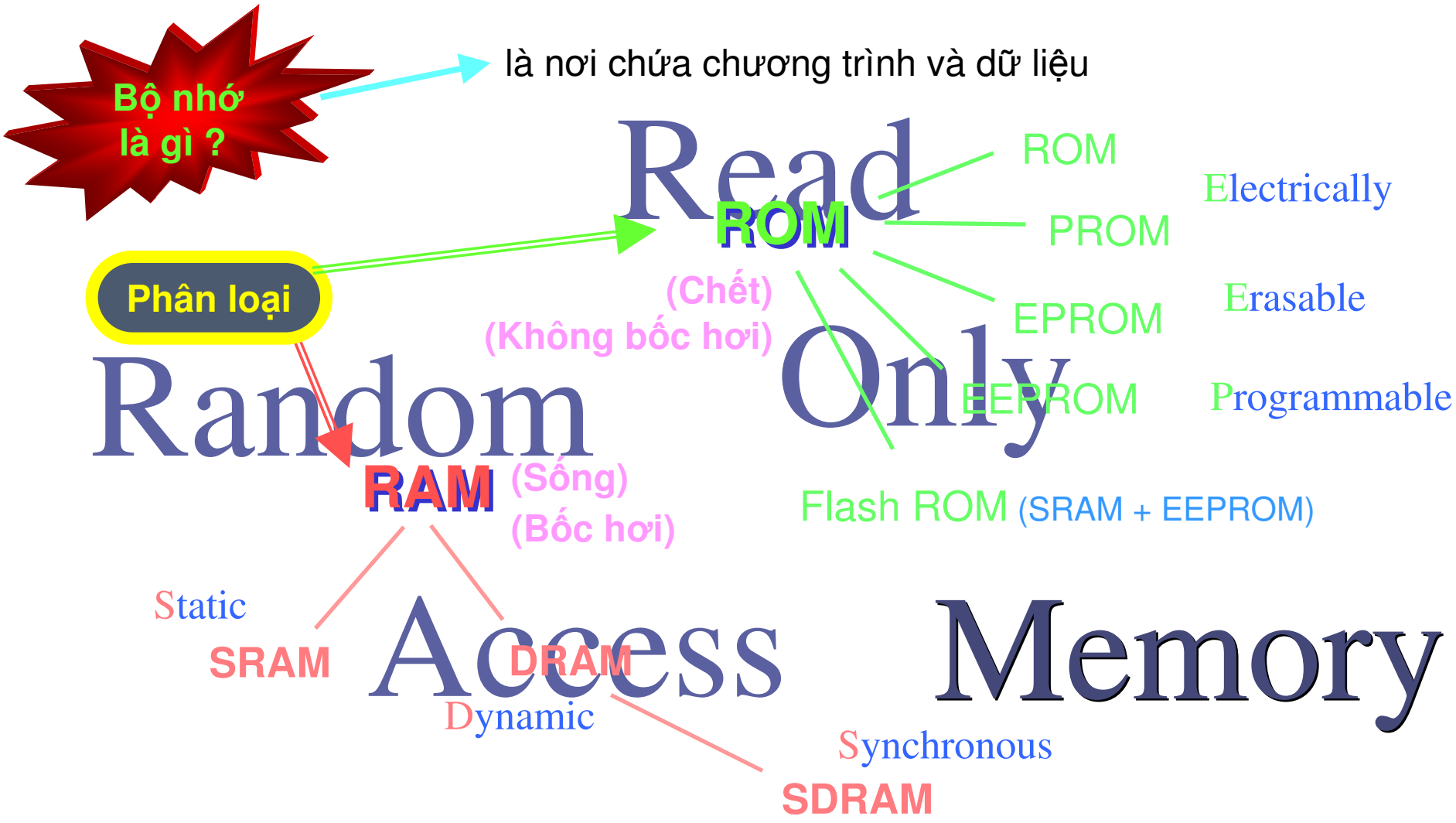


Cấu trúc luận lý của một máy tính

2.2 Kiến trúc máy tính

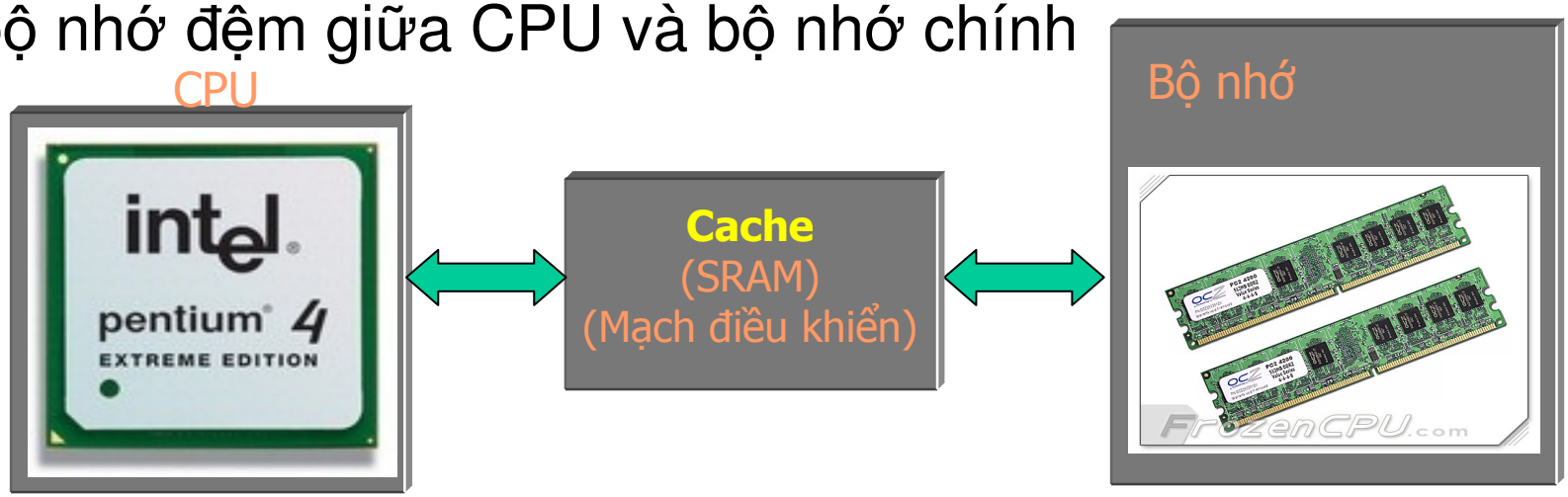
- Kiến trúc phần cứng máy tính ngày nay được biết đến như là một hệ thống gồm có :
  - Bộ nhớ (memory).
  - Bộ xử lý (processor).
  - Các tuyến (buses).

Bộ nhớ


Bộ nhớ là gì ?
là nơi chứa chương trình và dữ liệu
Phân loại
Read
ROM
(Chết)
(Không bốc hơi)
Only
ROM
PROM
EPROM
EEPROM
Electrically
Erasable
Programmable
Flash ROM (SRAM + EEPROM)
Random
Access
Memory
RAM
(Sống)
(Bốc hơi)
Static
SRAM
DRAM
Dynamic
Synchronous
SDRAM

Bộ nhớ đệm - Cache

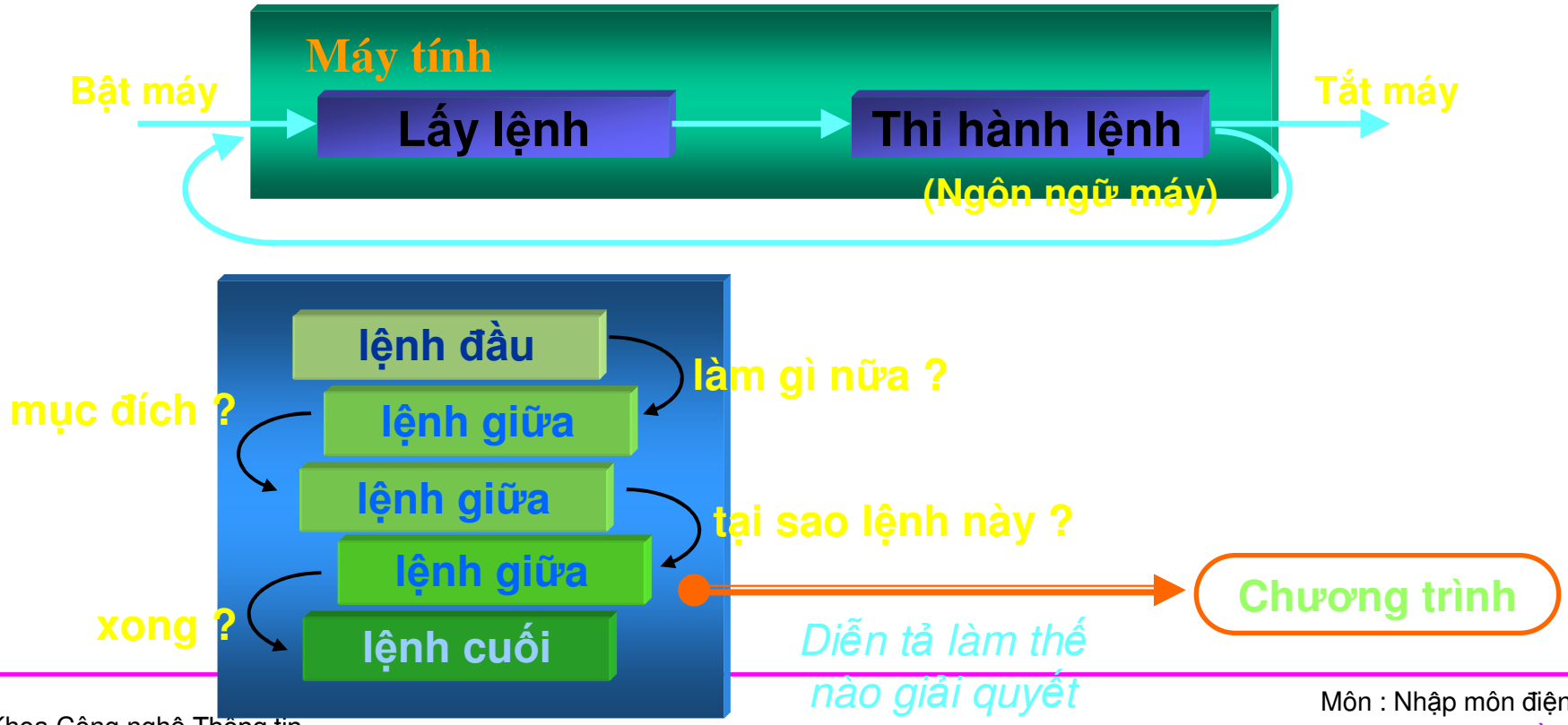
- Cache là bộ nhớ đệm giữa CPU và bộ nhớ chính



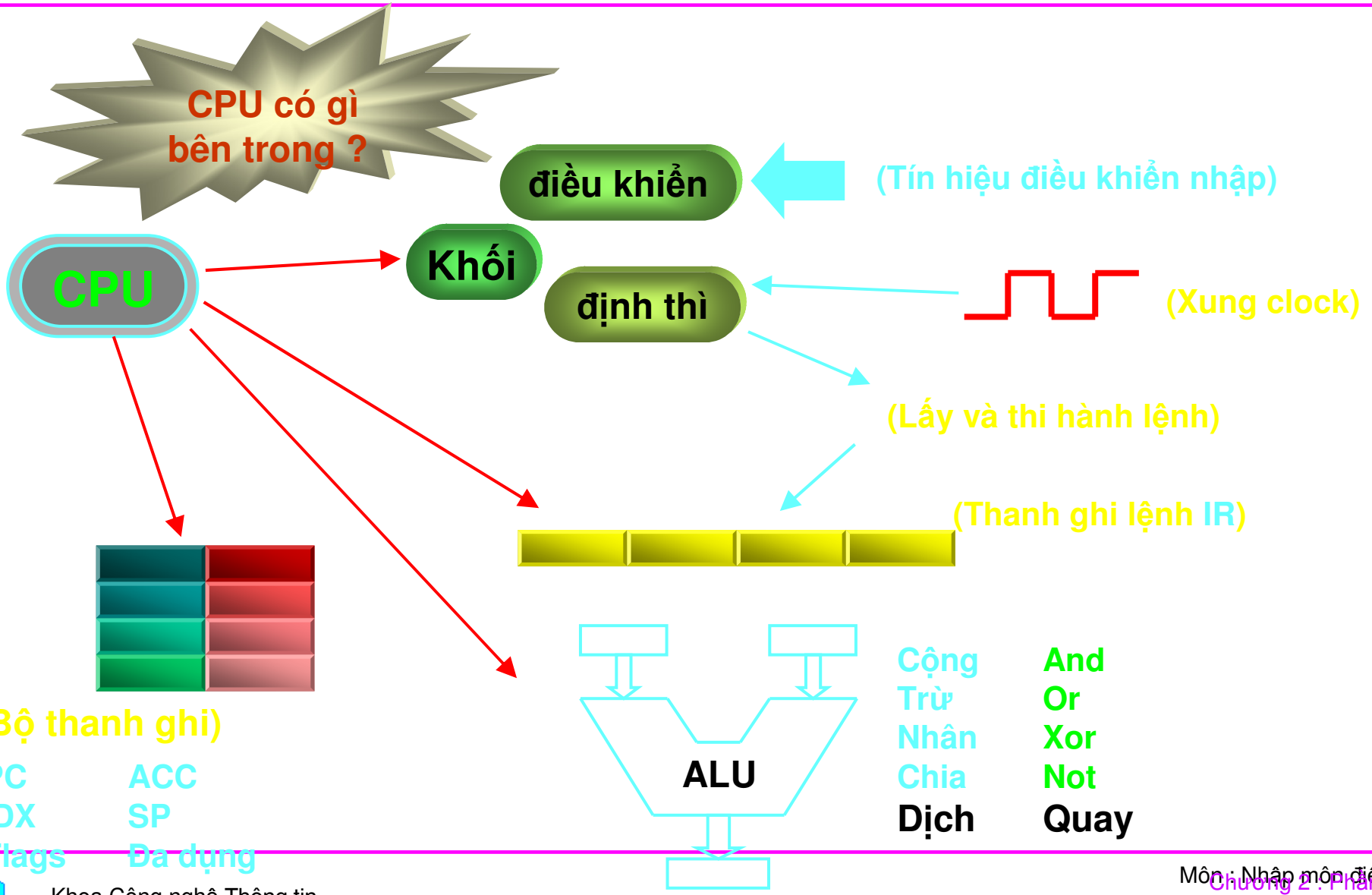
- Cache được chế tạo từ SRAM có tốc độ làm việc rất cao và có dung lượng nhỏ.
- Nhiệm vụ của cache là làm giảm thời gian đợi (wait-state) của CPU khi truy xuất bộ nhớ chính bằng cơ chế đọc trước các ô nhớ kế tiếp.
- Khái niệm "trúng cache".
- Các bộ xử lý hiện đại đều có cache bên trong.

Bộ xử lý - Processor

- Bộ xử lý hay còn gọi là CPU là nguồn phát sinh mọi hoạt động của máy tính.
- Bộ xử lý điều khiển hoạt động của máy tính thông qua việc lấy và thi hành lệnh nằm trong bộ nhớ.



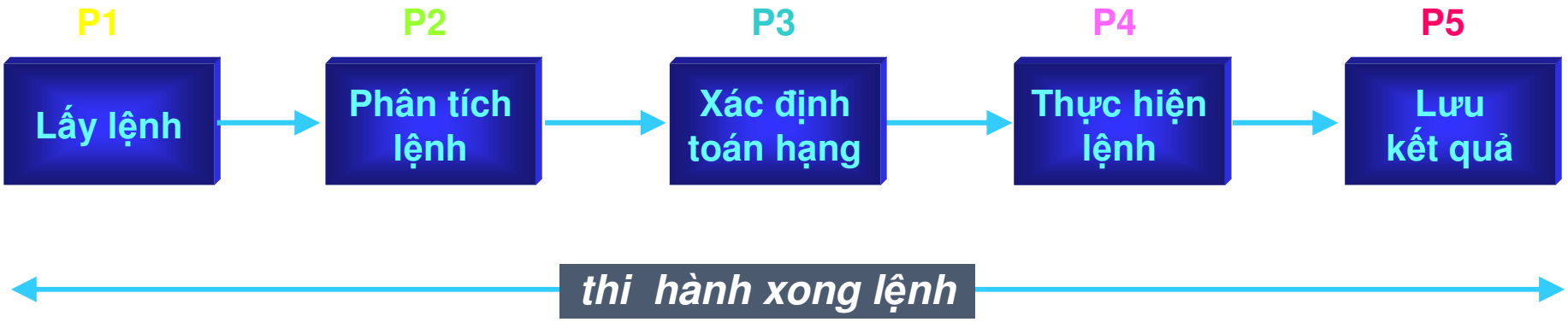
CPU


CPU có gì
bên trong ?
CPU
điều khiển
Khối
định thì
(Tín hiệu điều khiển nhập)
(Xung clock)
(Lấy và thi hành lệnh)
(Thanh ghi lệnh IR)
(Bộ thanh ghi)
PC
ACC
IDX
SP
Flags
Đa dụng
ALU
Cộng
Trừ
Nhân
Chia
Dịch
And
Or
Xor
Not
Quay

Kiến trúc bộ xử lý

- Kiến trúc CISC (Complex Instruction Set Computer)
  - Các lệnh của CPU có chiều dài khác nhau.
  - Thời gian thi hành lệnh cũng khác nhau.
- Kiến trúc RISC (Reduced Instruction Set Computer)
  - Các lệnh dài bằng nhau.
  - Thời gian thi hành các lệnh chỉ bằng 1 chu kỳ xung clock.
  - Cung cấp khả năng thi hành nhiều hoạt động cùng lúc (Super scalar execution).
  - Dùng cơ chế đường ống (Pipelining) để giảm thời gian thi hành.
  - Vấn đề đoán trước rẽ nhánh (Branche prediction).

Cơ chế đường ống - Pipelining



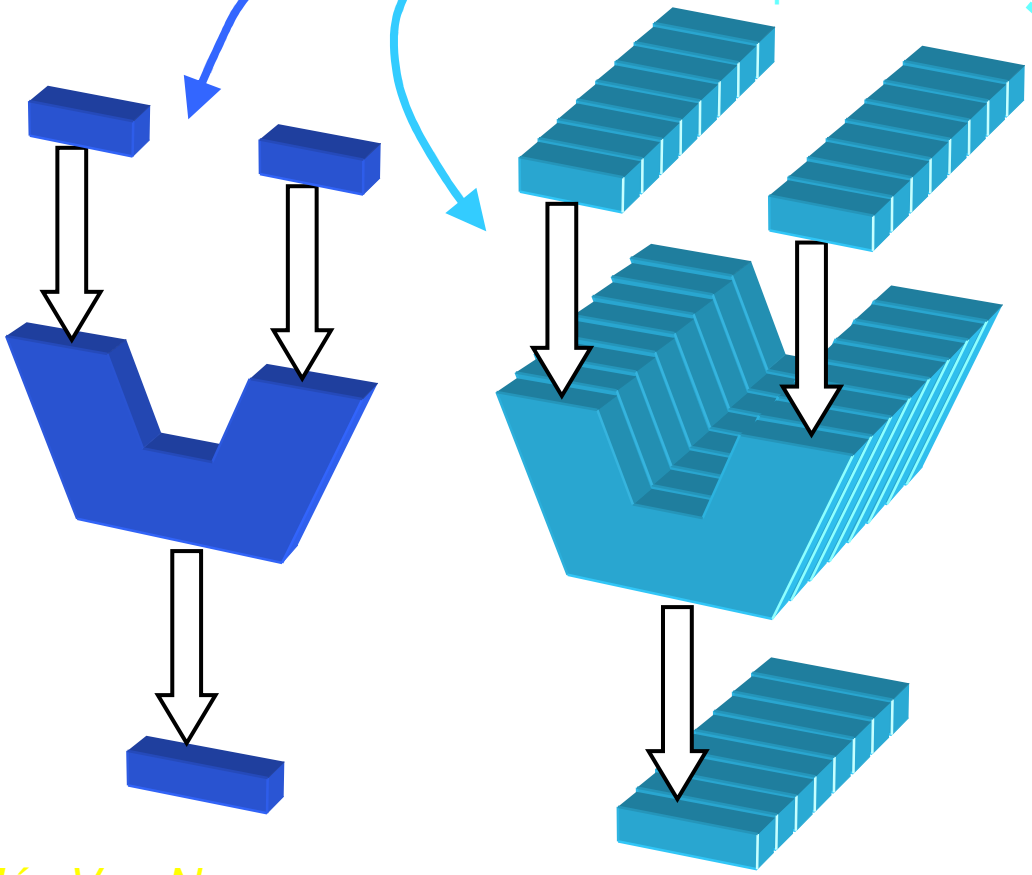
Máy tính song song

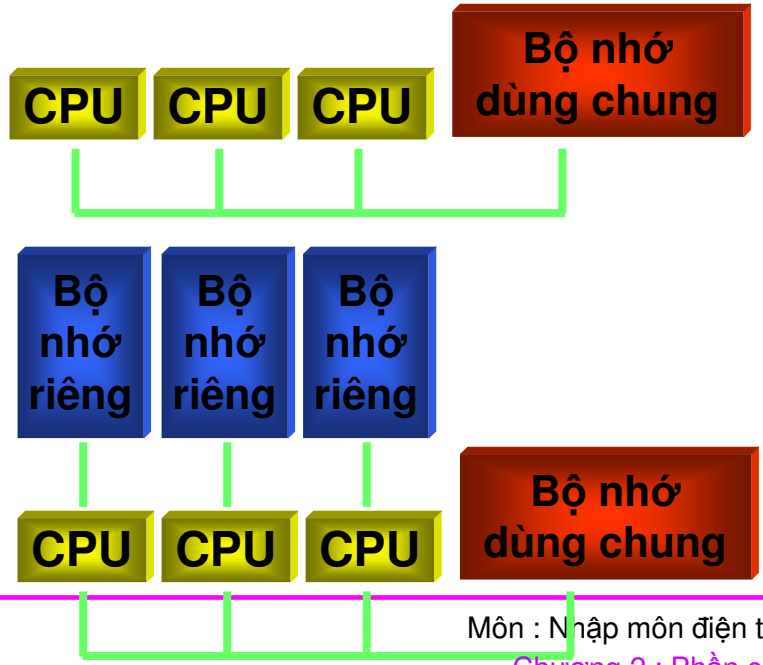
3 loại máy song song

SISD : single Instruction stream, single data stream

SIMD : single Instruction stream, multiple data stream

MIMD : multiple Instruction stream, multiple data stream





Máy Von Neumann

Máy Vector 8 ALU

Tuyến - Bus

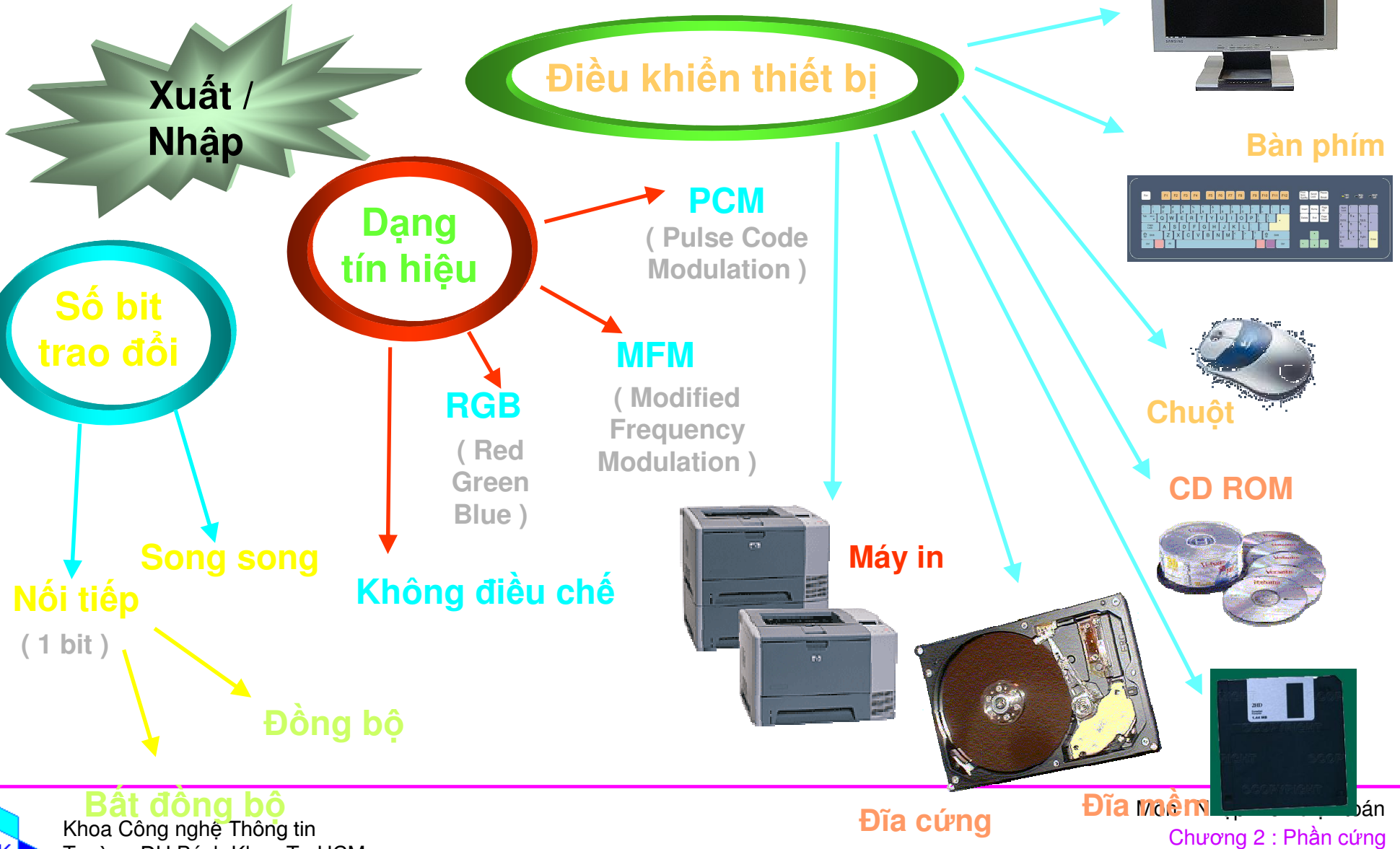
- Tuyến là một nhóm các dây dẫn song song mà mỗi đường có nhiệm vụ truyền tải 1 bit thông tin.
- Tuyến hệ thống là tuyến kết nối giữa CPU với các bộ phận mà nó muốn trao đổi thông tin mà cụ thể là bộ nhớ và khối xuất nhập (I/O).
- Trên một tuyến có thể truyền tải nhiều loại thông tin khác nhau.
- Một số tuyến có khả năng truyền thông tin theo cả 2 chiều. Tuy nhiên, trong từng thời điểm, luồng dữ liệu chỉ đi một chiều.
- Độ rộng của tuyến (số đường) xác định chiều dài của một từ (word) thông tin mà CPU trao đổi mỗi lần.

  Ví dụ : CPU dùng bus 16 bit để truyền dữ liệu 32 bit thì phải thực hiện 2 lần.

Kiến trúc tuyến

- Tuyến chuẩn (standard bus) :
  - MCA : micro channel architecture.
  - ISA : insdustry standard architecture.
  - IBM AT : advanced technology.
  - PS/2 : personal system 2.
  - EISA : extended insdustry standard architecture.
- Tuyến cục bộ (local bus) :
  - VESA : video electronics standard association.
  - PCI : Peripheral Component Interface.
  - AGP : Accelerated Graphics Port.

2.3 Thiết bị xuất nhập



Màn hình và card màn hình

Card màn hình

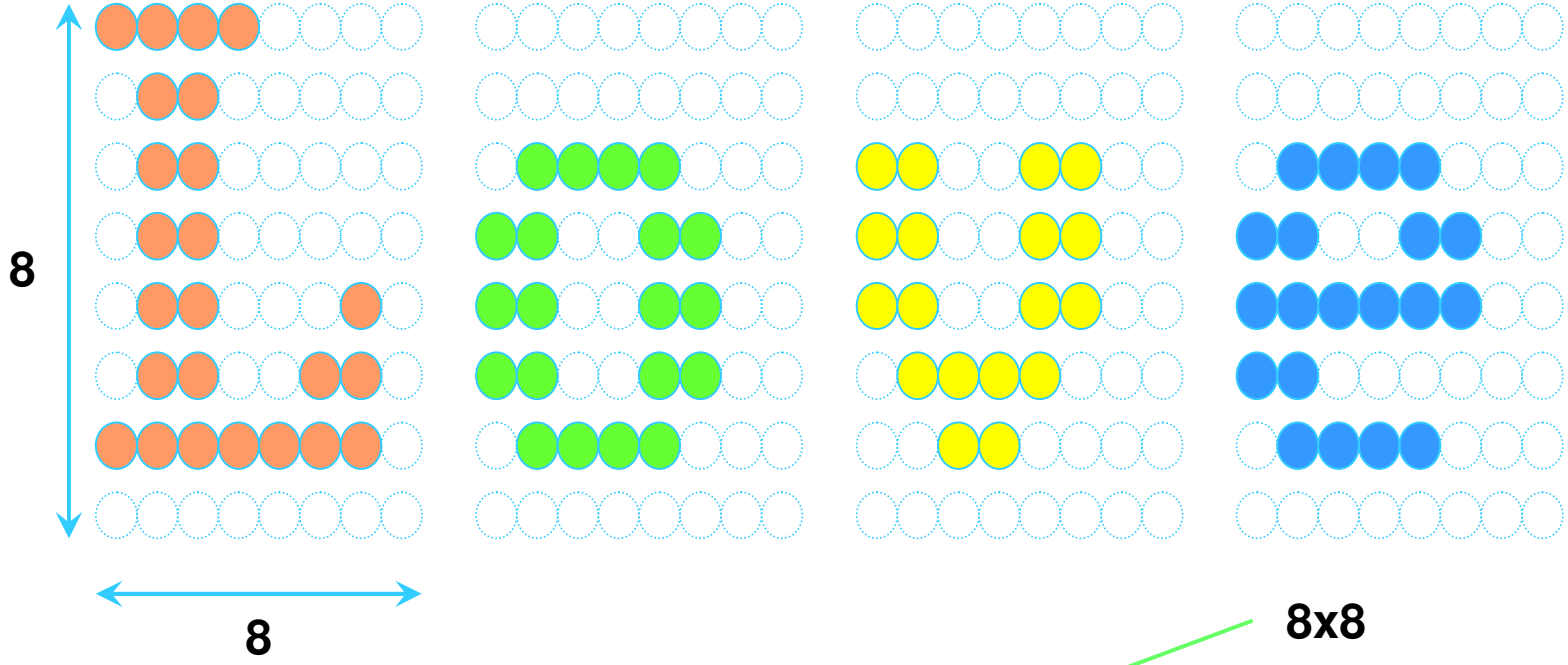
Màn hình LCD

Màn hình CRT

Hiển thị trong chế độ văn bản (text)

Chế độ văn bản

Ma trận điểm



Kích thước
- 8x8
- 14x8
- 16x8

Hiển thị trong chế độ đồ họa (graphics)

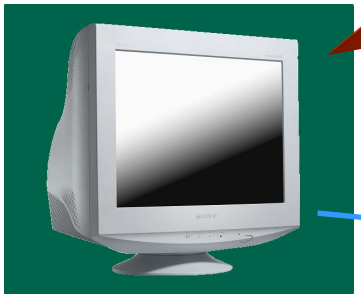
Card màn hình

cung cấp các chế độ màn hình

(độ phân giải)

Chế độ đồ họa

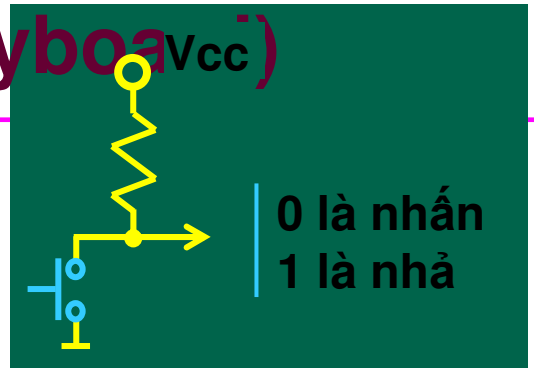
800 x 600 x 16bit = 960.000 byte ⇒ 1MB
1024 x 768 x 32bit = 3.145.728 byte ⇒ 4 MB



thể hiện các chế độ màn hình

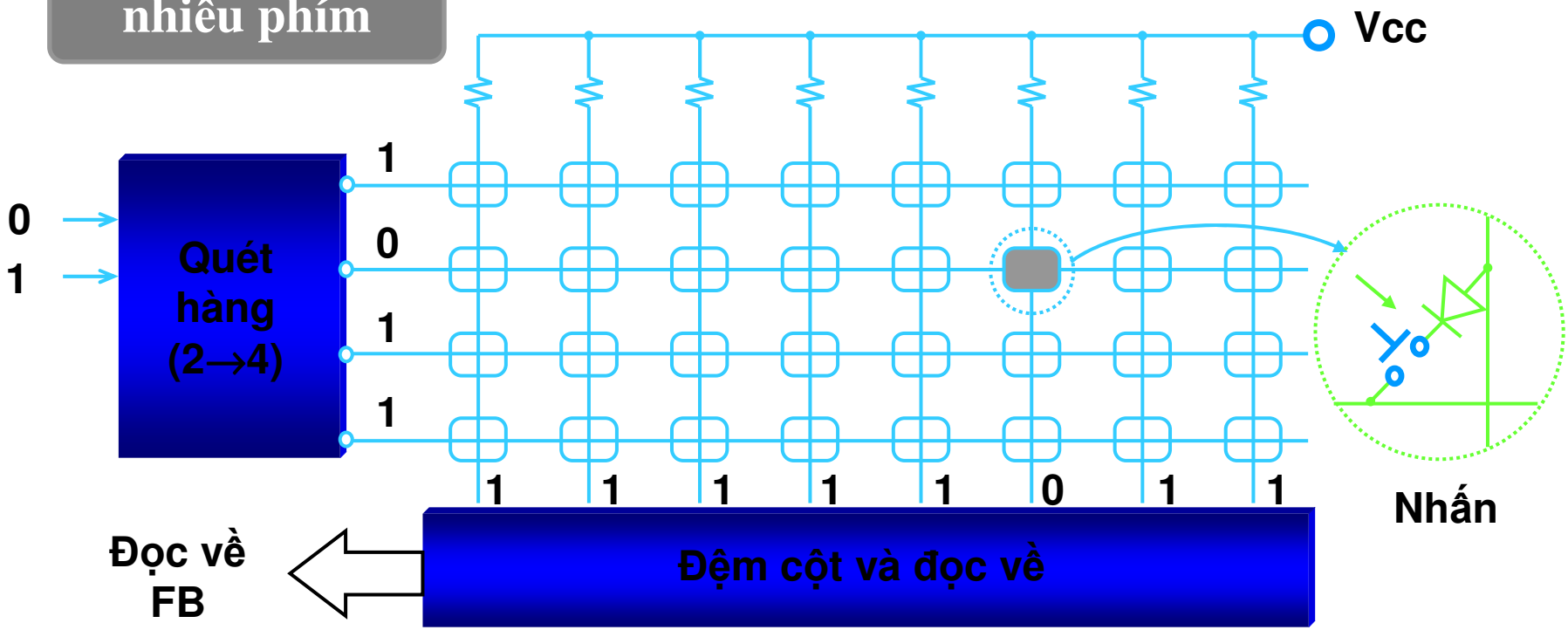
- kích thước điểm sáng:
  .31 mm, .29 mm, .22 mm
- tần số quét ngang (dòng)
  40 KHz, 70 KHz, 90 KHz
- tần số quét dọc (mành)
  50 Hz, 75 Hz, 100 Hz, ...

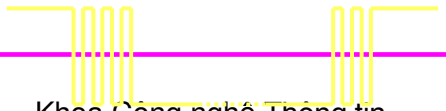
Tổ chức ma trận bàn phím (keyboard)



1 phím

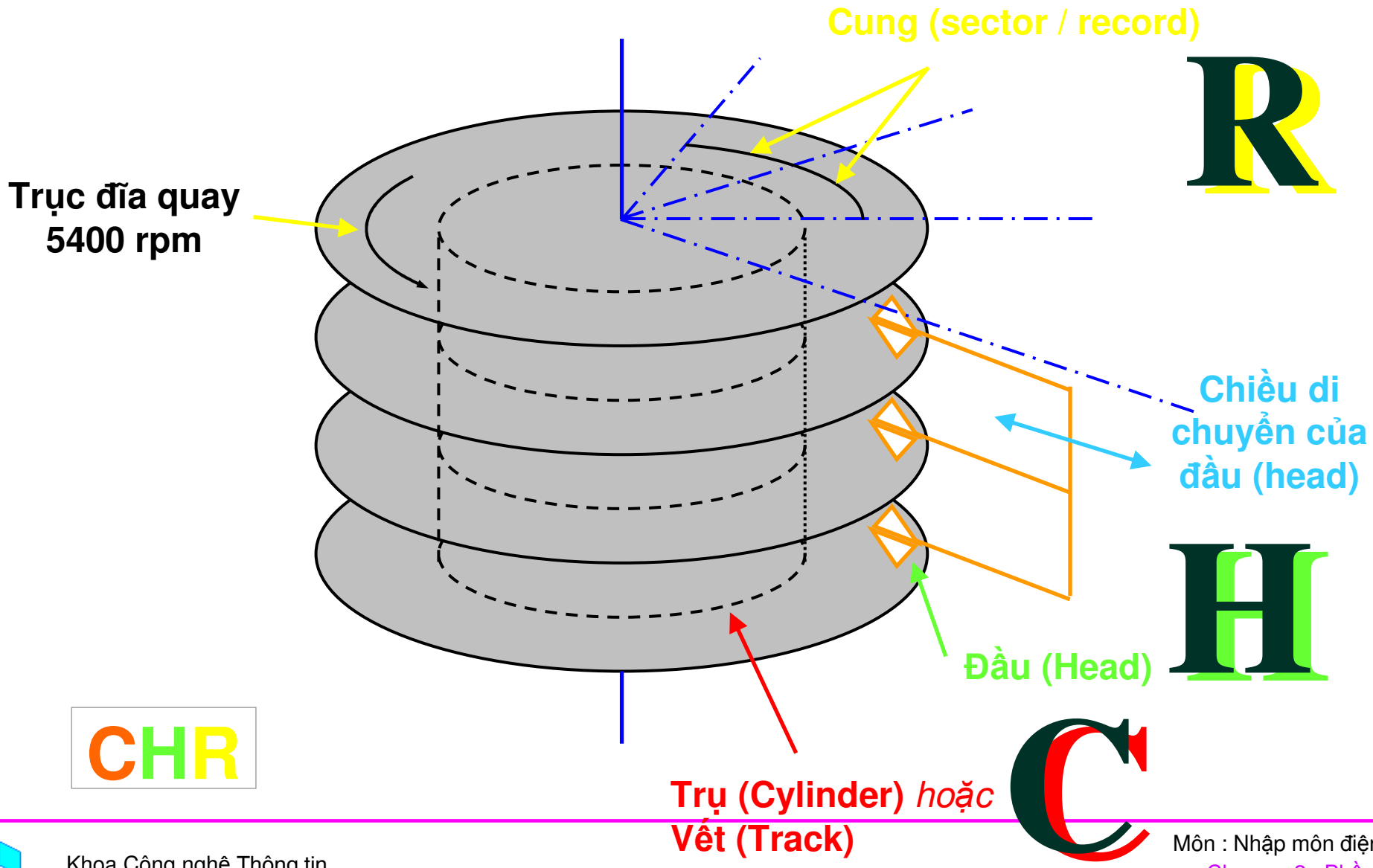
nhiều phím



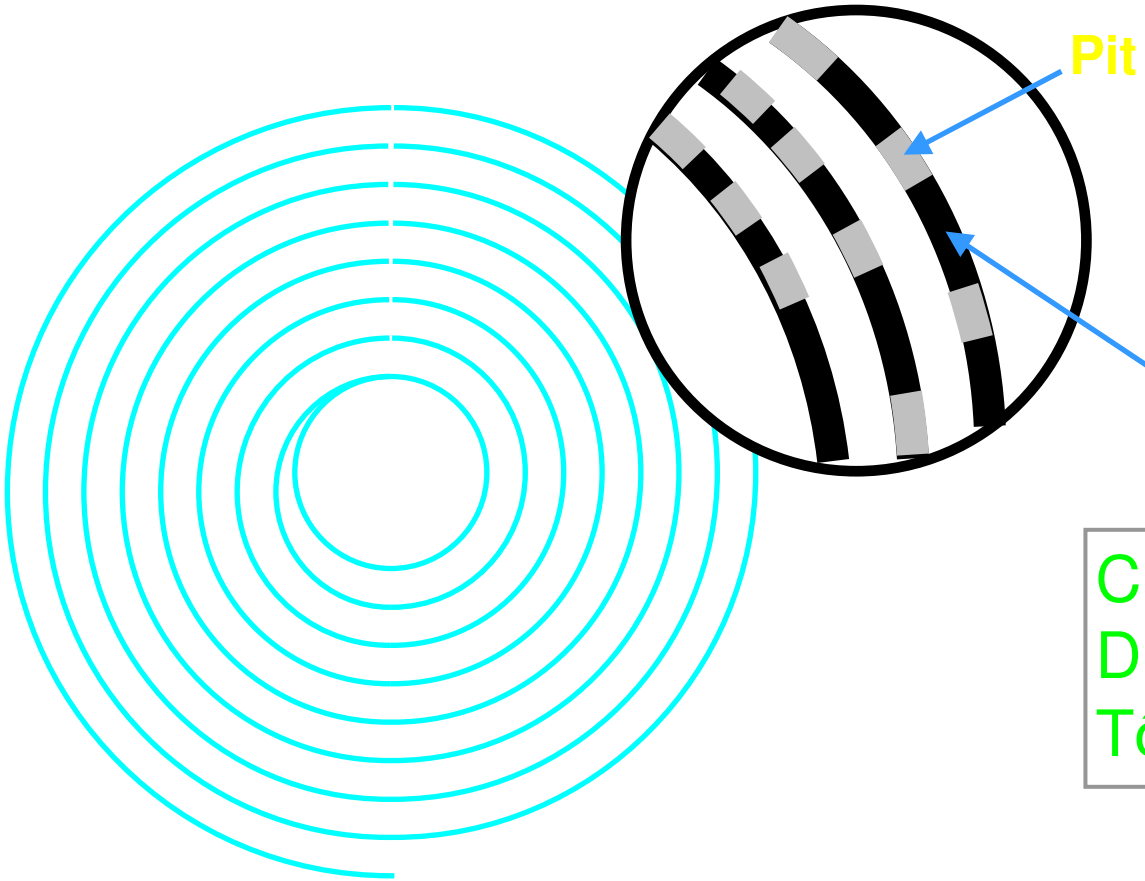


Hiện tượng rung phím (5 - 15 ms) → Chống rung → Cứng / Mềm

Tổ chức thông tin trên đĩa cứng (hard disk)



CDROM



Chứa 330.000 khối dữ liệu.
Dung lượng 650 MB / 74 min
Tốc độ x1 = 153.60 KByte/s

Thông tin ghi theo rãnh (track) hình xoắn ốc.
Dùng tia laser đục lổ 1 μm trên rãnh gọi là Pit.
Phần không bị đục lổ trên rãnh gọi là Land.

Máy in

Máy in kim

+ Máy rẻ tiền
+ Băng mực rẻ tiền
+ Lâu hết mực
+ In chậm

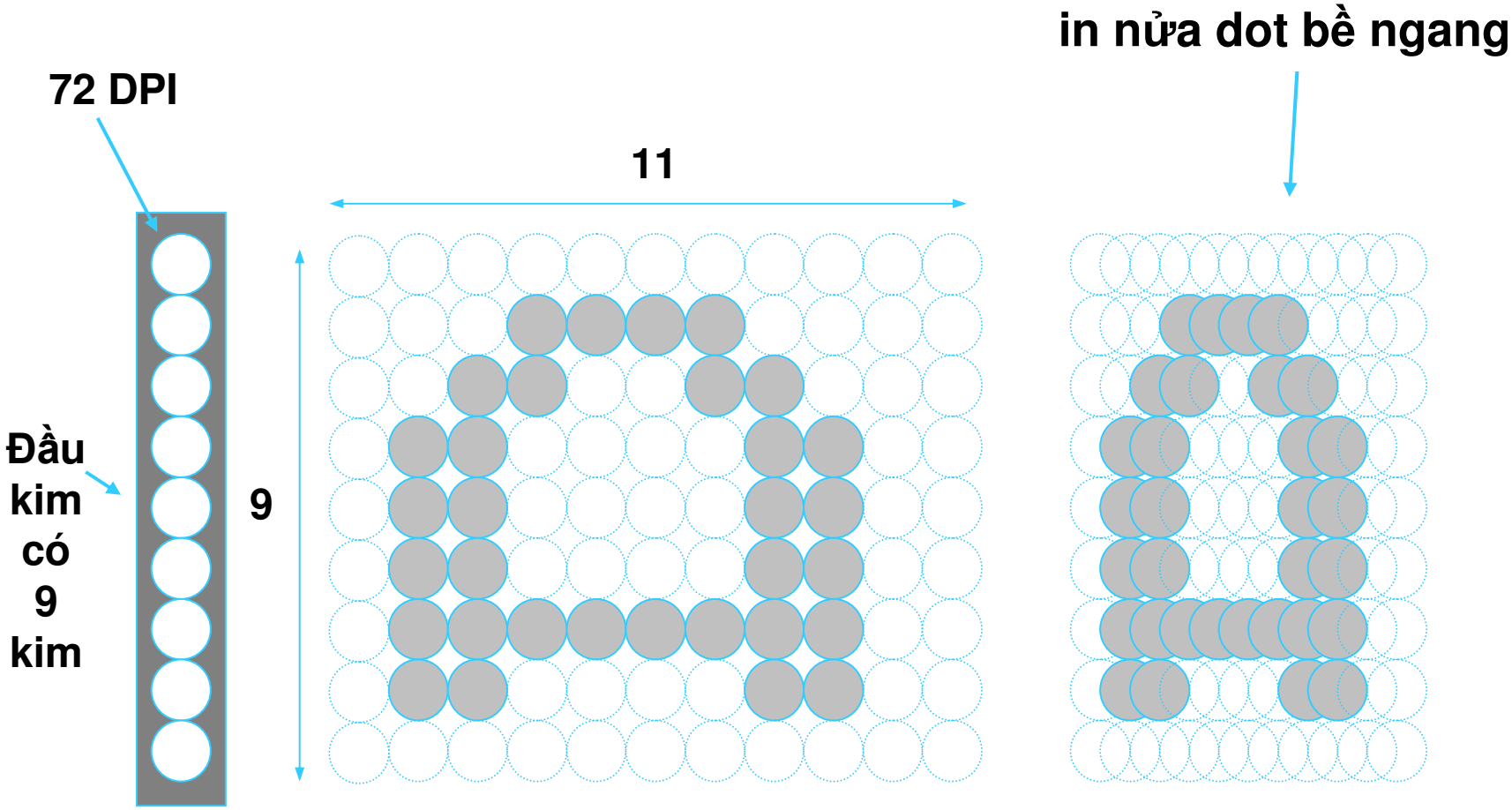
Máy in phun

+ Máy rẻ tiền
+ Mực lỏng, đắt tiền
+ Mau hết mực
+ In chậm

Máy in laser

+ Máy đắt tiền
+ Mực bột, đắt tiền
+ Lâu hết mực
+ In nhanh

Ma trận điểm trên máy in kim



MÔN NHẬP MÔN ĐIỆN TOÁN

Chương 3

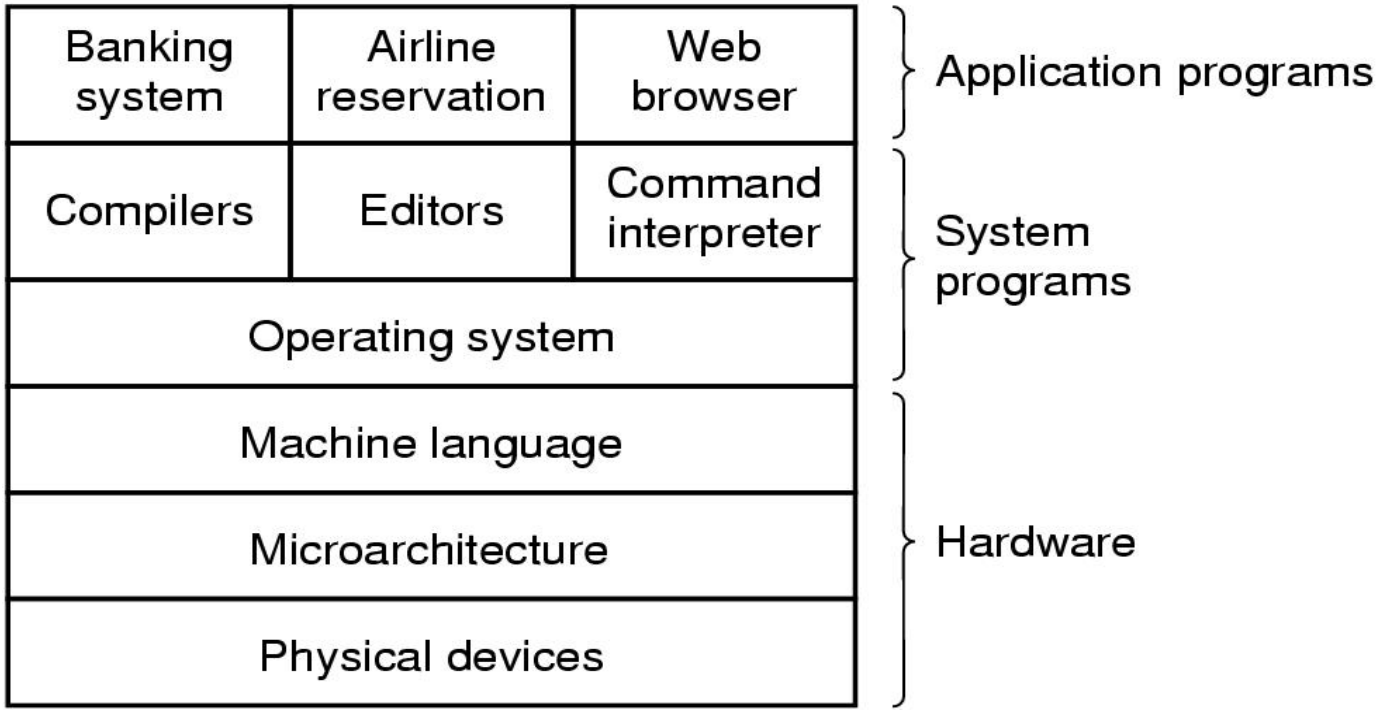
HỆ ĐIỀU HÀNH

3.1 Định nghĩa sơ lược về hệ điều hành
3.2 Lịch sử phát triển hệ điều hành
3.3 Phân loại các hệ điều hành
3.4 Nhắc lại phần cứng máy tính
3.5 Các khái niệm cơ bản về hệ điều hành
3.6 Các lời gọi dịch vụ HĐH "System call"
3.7 Kiến trúc của HĐH

Tài liệu tham khảo : chương 1, sách "Modern Operating Systems", Andrew S. Tanenbaum: , 2nd ed, Prentice Hall

3.1 Định nghĩa sơ lược về hệ điều hành

- Máy tính số là máy nhiều cấp, trong đó 3 cấp chính yếu là :
  - vật lý (phần cứng - hardware)
  - chương trình hệ thống (system programs)
  - chương trình ứng dụng (application programs)



Hệ điều hành là gì?

Hai định nghĩa được nhiều người đồng ý nhất :

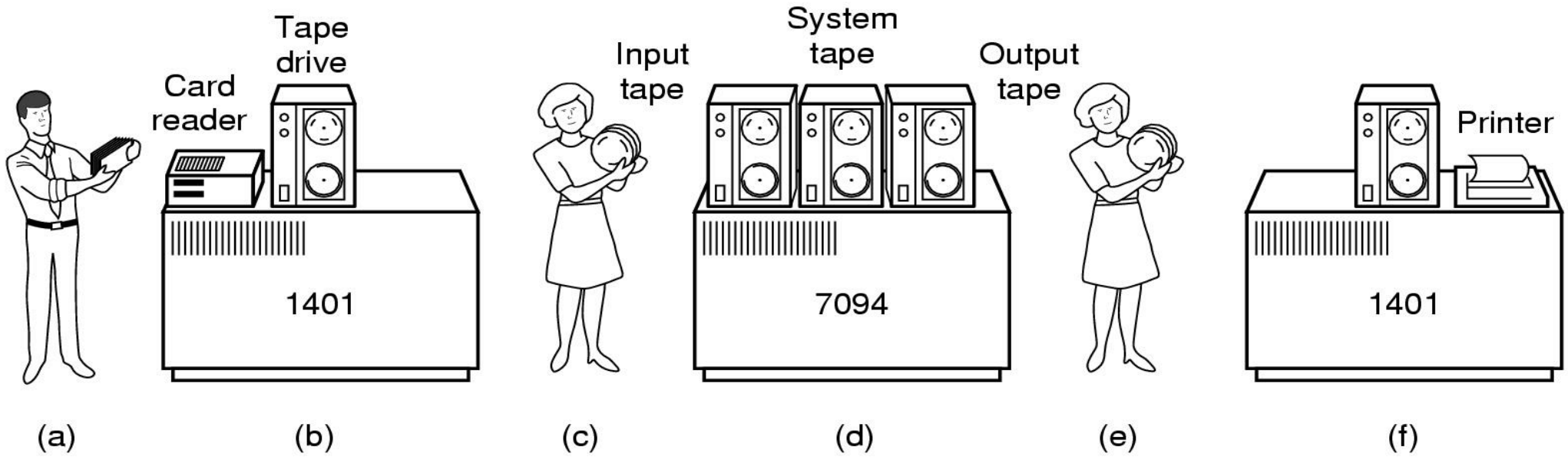
1. HĐH là 1 máy tính luận lý mở rộng (extended machine) : đây là góc nhìn từ ngoài vào.
   - dấu các chi tiết khó, phiền phức cần thực hiện.
   - cung cấp cho người dùng 1 máy luận lý dễ dùng hơn và độc lập với phần cứng (thông qua các lệnh *system calls*)
2. HĐH là 1 hệ quản lý các tài nguyên của máy : đây là góc nhìn bên trong
   - Phân chia việc dùng tài nguyên theo thời gian, mỗi chương trình dùng tài nguyên trong 1 khoảng thời gian rồi giao lại cho chương trình khác dùng (CPU, máy in).
   - Phân chia tài nguyên theo không gian : mỗi chương trình dùng 1 vùng nhỏ tài nguyên (bộ nhớ, đĩa).

3.2 Lịch sử hệ điều hành

Vì HĐH nằm trên cấp phần cứng nên lịch sử HĐH gắn liền với lịch sử phát triển phần cứng máy tính. Ở đây chúng ta tổng kết lại lịch sử phát triển máy tính số gồm 4 thế hệ sau :

1. First generation 1945 - 1955
   - vacuum tubes, plug boards
   - Inventors : Aiken (USA), Zuse (Germany)
   - chưa cần HĐH
2. Second generation 1955 - 1965
   - transistors
   - batch systems
3. Third generation 1965 – 1980
   - ICs (Integrated Circuits)
   - multiprogramming, spooling, time-sharing
4. Fourth generation 1980 – present
   - LSI (Large Scale Integration)
   - Hệ điều hành cho PC

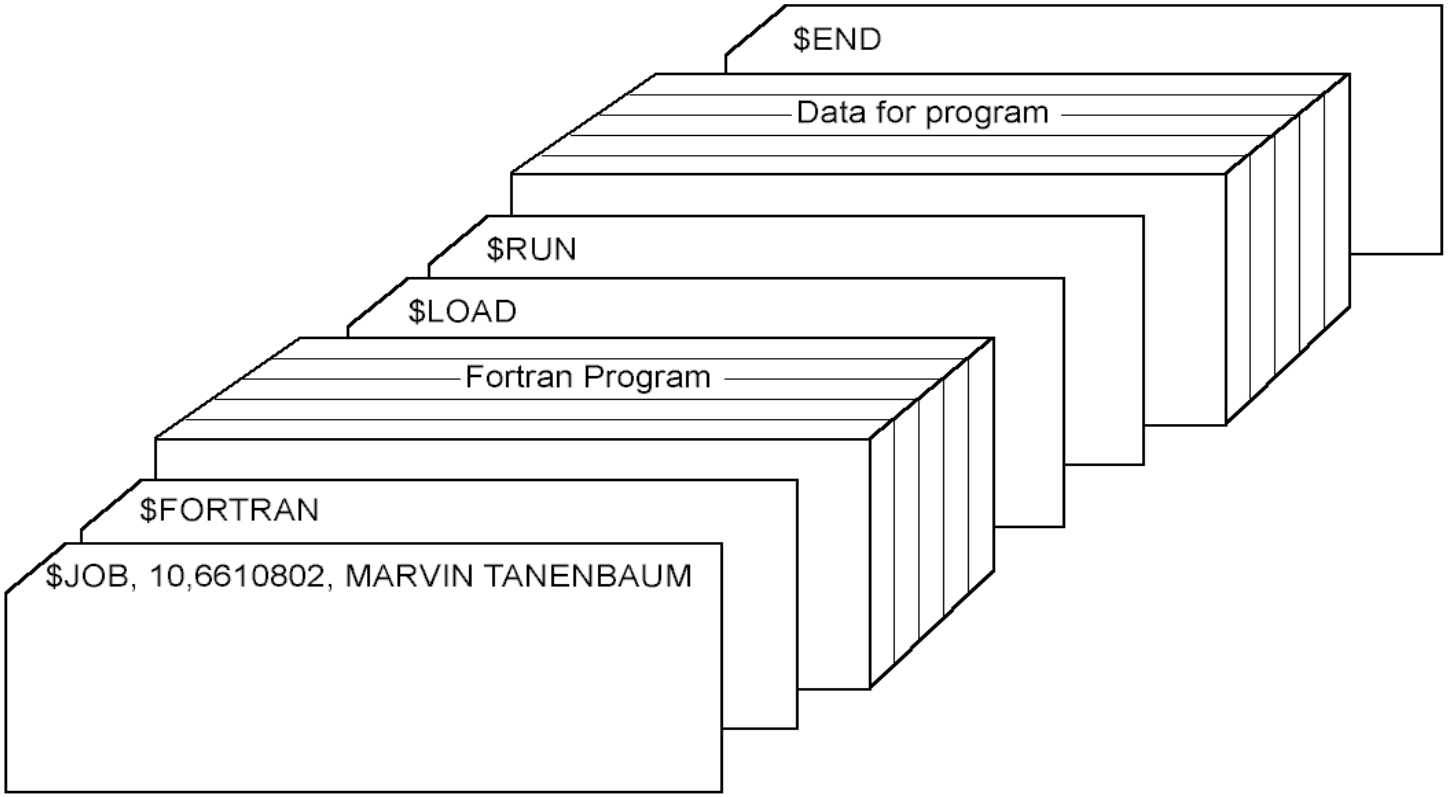
Lịch sử hệ điều hành - Thế hệ thứ 2



Early batch system (hệ thống xử lý bó)

- xuyên phiếu chuyển chương trình thành chồng card đục lỗ.
- để n chồng card theo thứ tự cho máy đọc card 1401 đọc và ghi lên băng từ.
- gắn băng từ cho máy 7094 xử lý tuần tự từng chương trình, kết quả của chương trình được ghi lên băng kết xuất.
- gắn băng kết xuất vào máy in 1401 để in ra giấy.

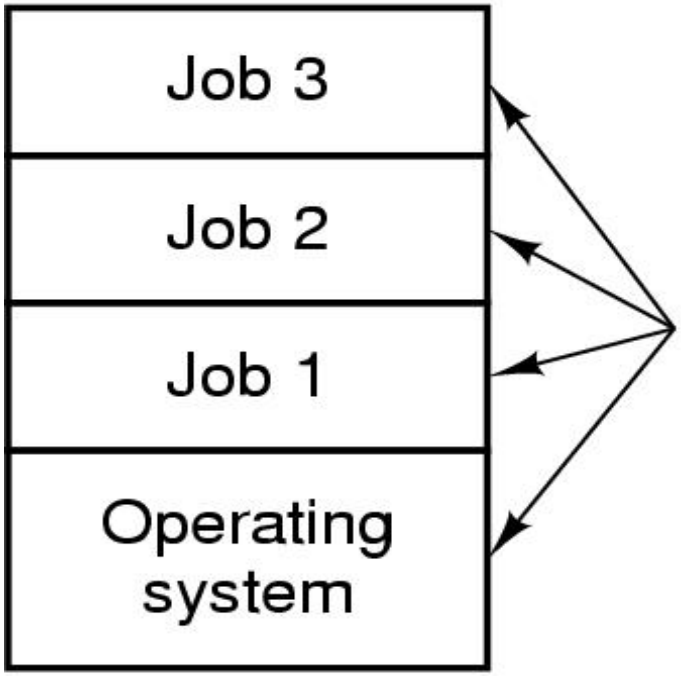
Lịch sử hệ điều hành - Thế hệ thứ 2



Cấu trúc điển hình của 1 job FMS

(FMS: Fortran Monitor System, hệ điều hành của IBM cho mainframe 7094)

Lịch sử hệ điều hành - Thế hệ thứ 3



OS/360 của IBM
MULTICS (MIT, Bell Labs)

Memory partitions

(Các vùng của bộ nhớ)

- Multiprogramming system
- Spooling (Simultaneous Peripheral Operation On Line)
- Time sharing

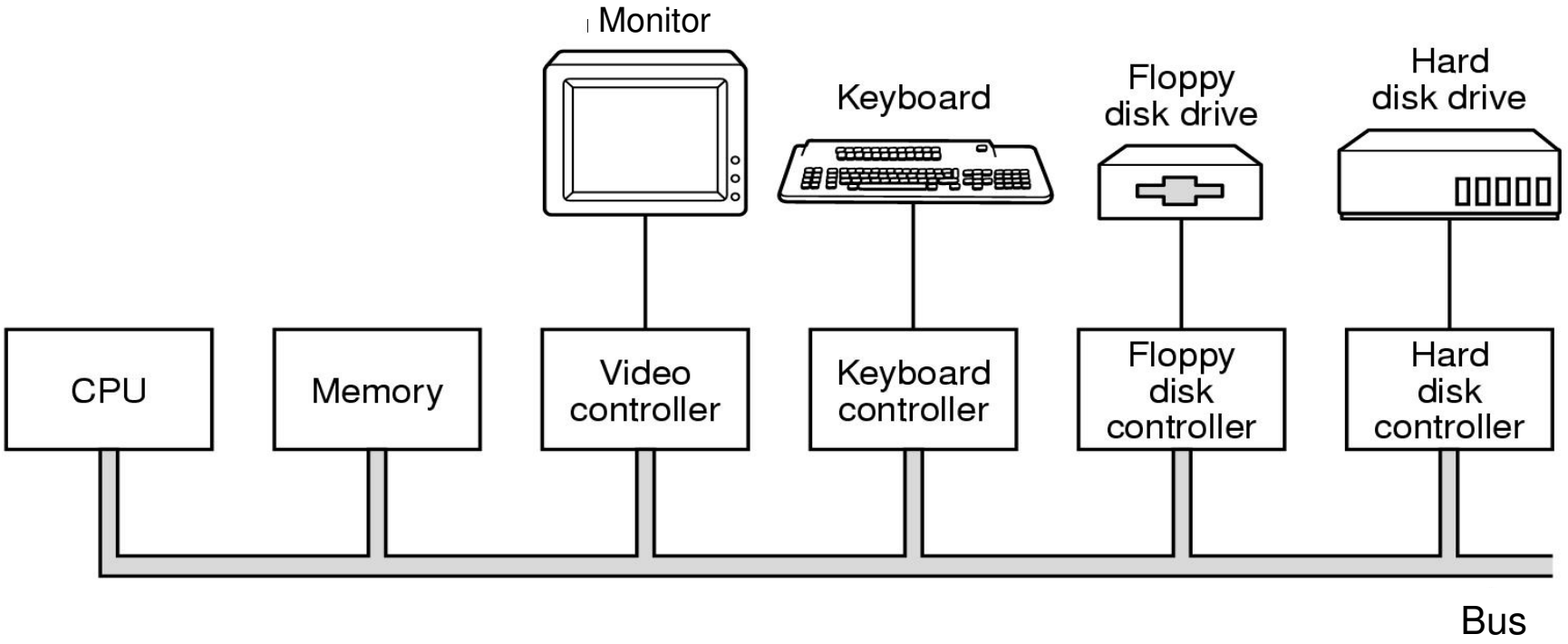
Lịch sử hệ điều hành - Thế hệ thứ 4

- 1974, first microcomputer
  - Intel 8080, first general-purposed 8-bit CPU
  - floppy disk
  - CP/M (Control Program for Microcomputers)
- early 1980s, IBM PC
  - DOS (Disk Operating System)
  - MS-DOS (Microsoft Disk Operating System)
- 1983, IBM PC/AT (Intel 80286 CPU)
- 1985-1995, Windows on top of MS-DOS
- Pentium PC
  - UNIX, Linux, Windows 2000
  - X Windows system (UNIX, Linux)

3.3 Phân loại các hệ điều hành

- Mainframe operating systems — High-end Web servers
  - OS/390
- Server operating systems — Web service, file service
  - UNIX, Linux, Windows 2000
- Multiprocessor operating systems
- Personal computer operating systems
  - Linux, Windows XP, Macintosh
- Real-time operating systems — Control systems
  - VxWorks, QNX
- Embedded operating systems — Mobile phones
  - uCLinux, PalmOS, Windows CE
- Smart card operating systems — Smart cards

3.4 Nhắc lại phần cứng máy tính

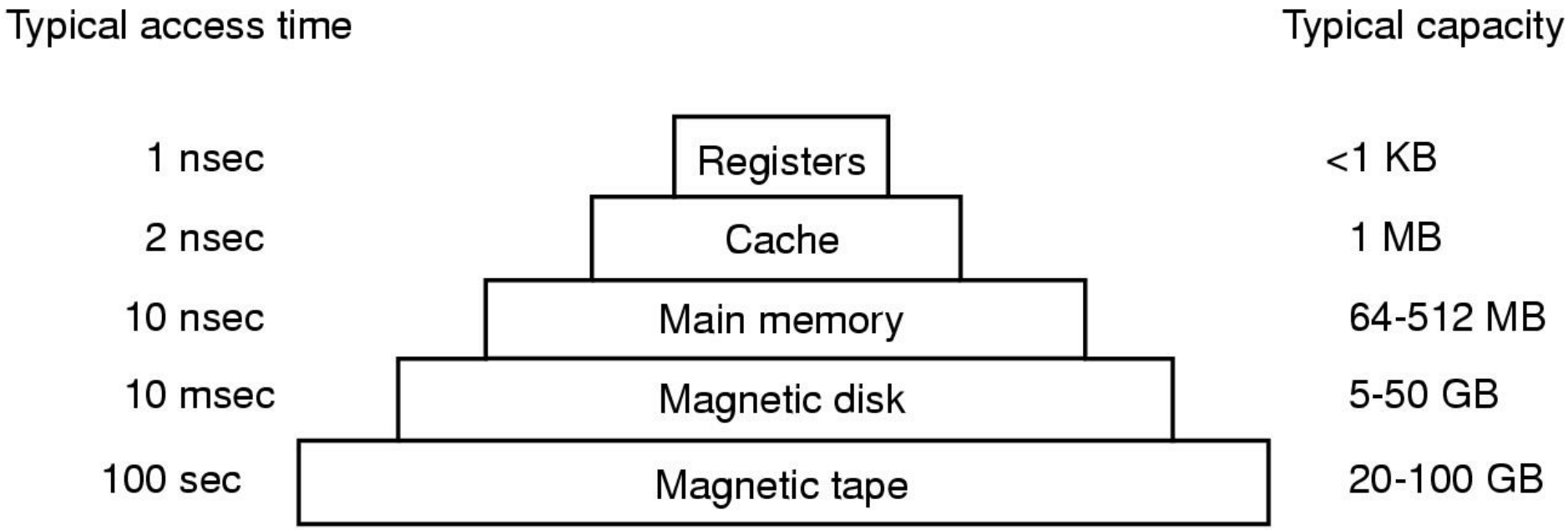


Các thành phần của một máy PC đơn giản

Nhắc lại phần cứng máy tính - Processors

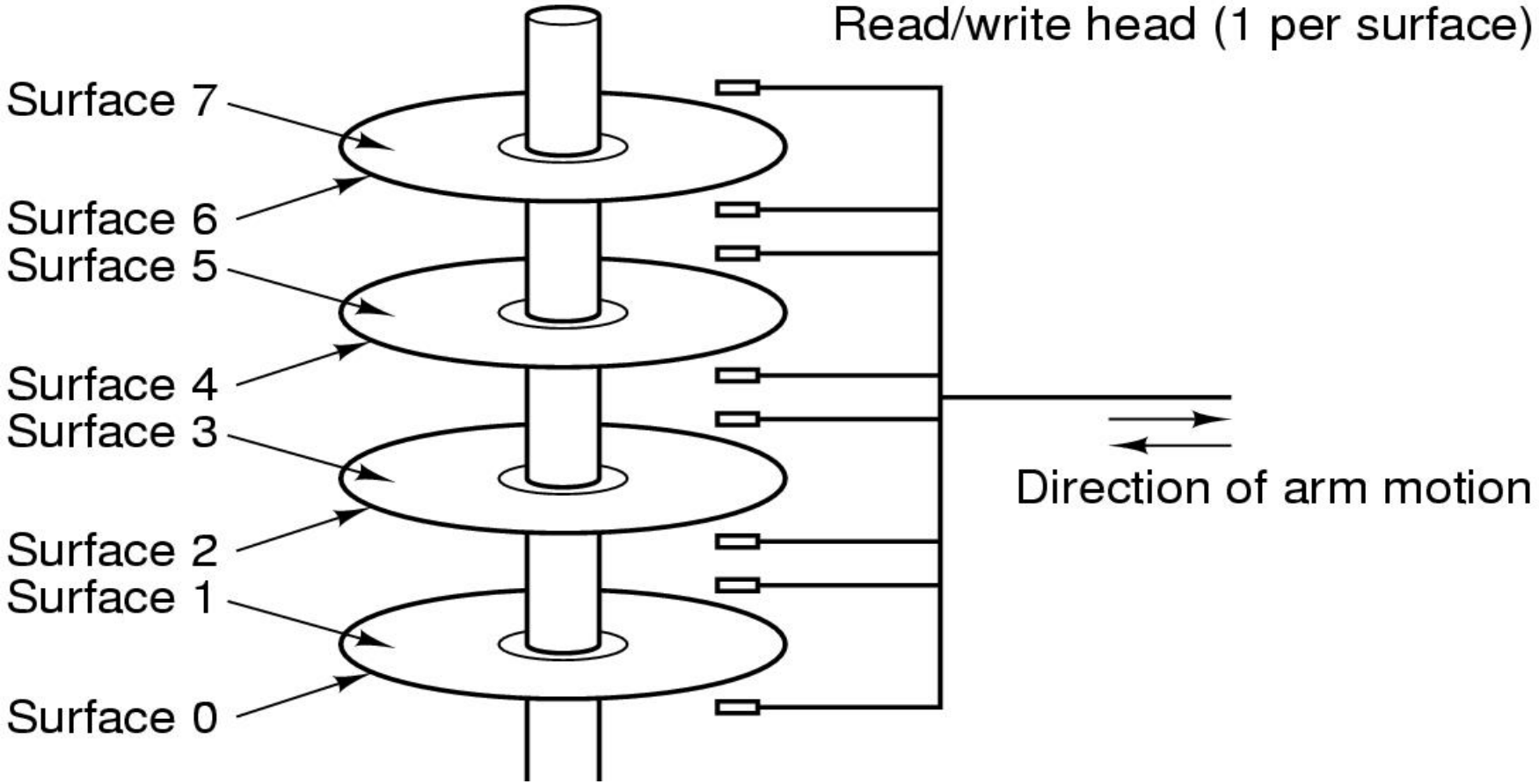
- Special registers
  - Program counter
  - Stack pointer
  - Program Status Word (PSW)
    - kernel mode
    - user mode
- TRAP instruction
  - System call

Nhắc lại phần cứng máy tính - Memory



Phân cấp điển hình các loại bộ nhớ
các giá trị chỉ có ý nghĩa xấp xỉ

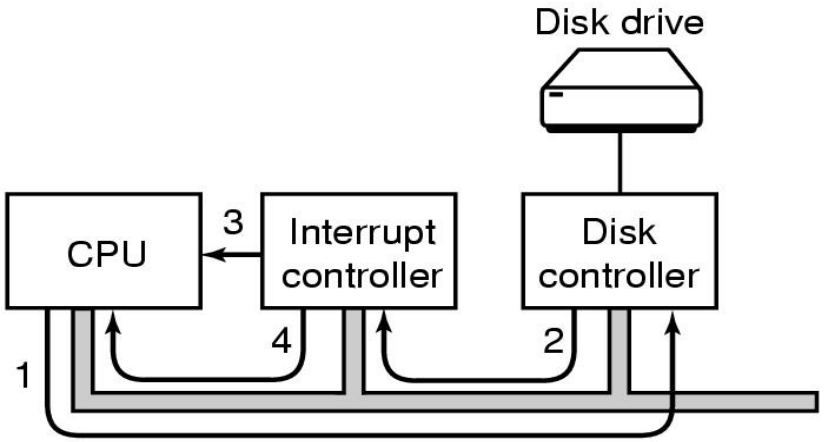
Nhắc lại phần cứng máy tính - Đĩa cứng



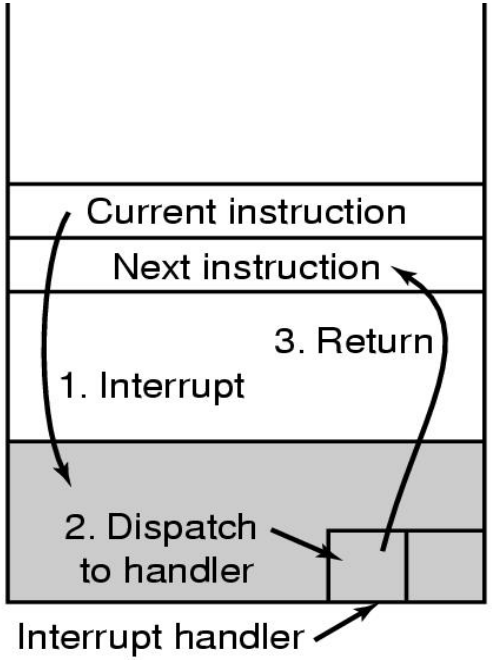
Cấu trúc của một ổ đĩa cứng

Nhắc lại phần cứng máy tính - Thiết bị I/O

Device driver là gì?



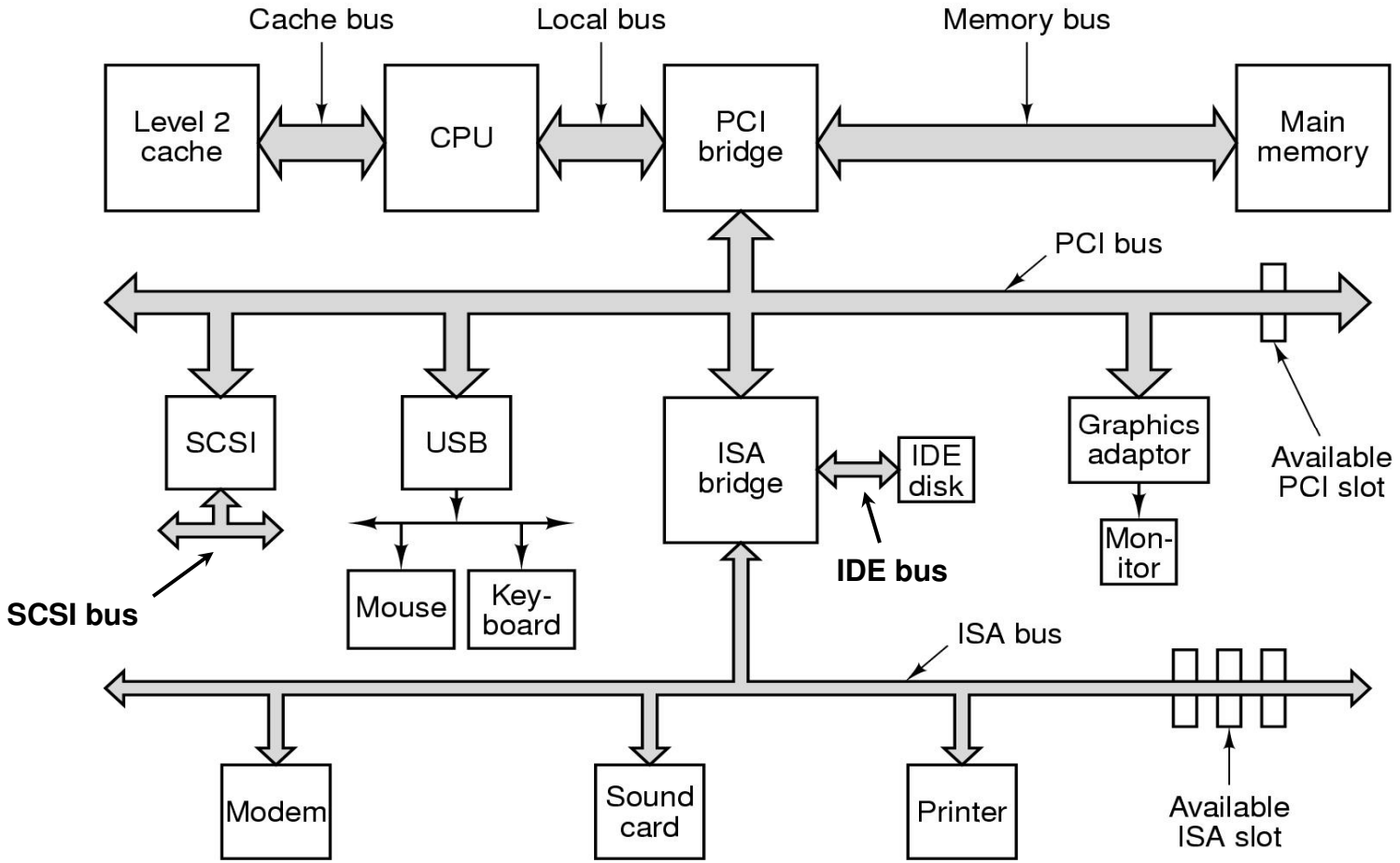
(a)



(b)

(a) Steps in starting an I/O device and getting interrupt
(b) Interrupt processing

Nhắc lại phần cứng máy tính - Bus



Cấu trúc của một hệ thống Pentium

3.5 Các ý niệm chủ đạo trong hệ điều hành

- Các tài nguyên của máy
- Quá trình (process)
- Lập thời biểu cho các quá trình chạy đồng thời (Scheduler)
- Cho phép các quá trình đồng thời truy xuất chung tài nguyên
- Deadlock và giải quyết
- Quản lý bộ nhớ (memory management)
- Hệ thống file (tập tin)
- Giao tiếp với thế giới ngoài (input/output)
- An ninh dữ liệu (security)
- The shell

Các ý niệm chủ đạo trong hệ điều hành - Tài nguyên

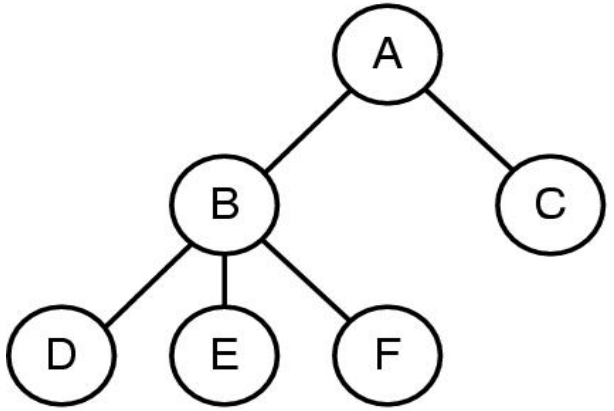
- Tài nguyên của 1 chương trình là bất kỳ thành phần nào của máy tính được sử dụng bởi chương trình đó.
  - Tài nguyên phần cứng : CPU, bộ nhớ, đĩa, CDROM, đĩa USB, màn hình, bàn phím, chuột, card mạng,...
  - Tài nguyên phần mềm : các file dữ liệu và các hệ thống phần mềm khác mà 1 chương trình cần truy xuất/tương tác.
- HĐH cần quản lý các tài nguyên sao cho việc sử dụng chúng bởi các chương trình được tin cậy, an toàn, hiệu quả và độc lập với tính chất vật lý của chúng.

Các ý niệm chủ đạo trong hệ điều hành - Process

File chương trình thường có 2 dạng : mã nguồn và mã thực thi. File thực thi (*.exe trên Windows) có thể được chạy trực tiếp bởi máy, nhưng nếu chưa chạy, nó vẫn là thành phần thụ động, ngủ yên và không tạo ra kết quả gì. Khi người dùng kích hoạt 1 file chương trình, nó được chạy bởi CPU, lúc này ta gọi nó bằng thuật ngữ "Tiến trình" (Process). Trong lúc hoạt động, process có thể tạo ra nhiều process khác (con) và cứ thế tiếp tục.

Một cây process (process tree)

- A đã tạo hai process con, B và C
- B đã tạo ba process con, D, E, và F



Các ý niệm chủ đạo trong hệ điều hành - Scheduler

- Để quản lý việc chạy các process đơn giản và dễ dàng nhất, người ta đã cho chúng chạy tuần tự, mỗi lần cho 1 chương trình chạy. Chỉ khi chương trình chạy xong (dù lâu hay mau) thì ta mới cho chương trình khác chạy,...
- Hầu hết các chương trình đều cần giao tiếp với người (hay I/O nói chung). Việc giao tiếp với I/O thường chậm hơn rất nhiều so với tốc độ của CPU, nghĩa là lúc chương trình dừng chờ I/O (chờ nhập phím), CPU phải ngủ chờ mất thời gian và hiệu suất làm việc của nó.
- Để sử dụng CPU hiệu quả hơn, người ta cố gắng cho nhiều chương trình chạy đồng thời. Cách thông thường nhất là dùng kỹ thuật phân chia thời gian (Time sharing) : chia trục thời gian làm nhiều khe nhỏ (quantum), cho mỗi chương trình chạy 1 khe nhỏ rồi dừng nó lại, chọn chương trình khác chạy trong khe nhỏ kế tiếp,...

Các ý niệm chủ đạo trong hệ điều hành - Scheduler

- Module của HĐH quản lý việc phân chia thời gian cho các chương trình chạy được gọi là trình lập thời biểu (Scheduler).

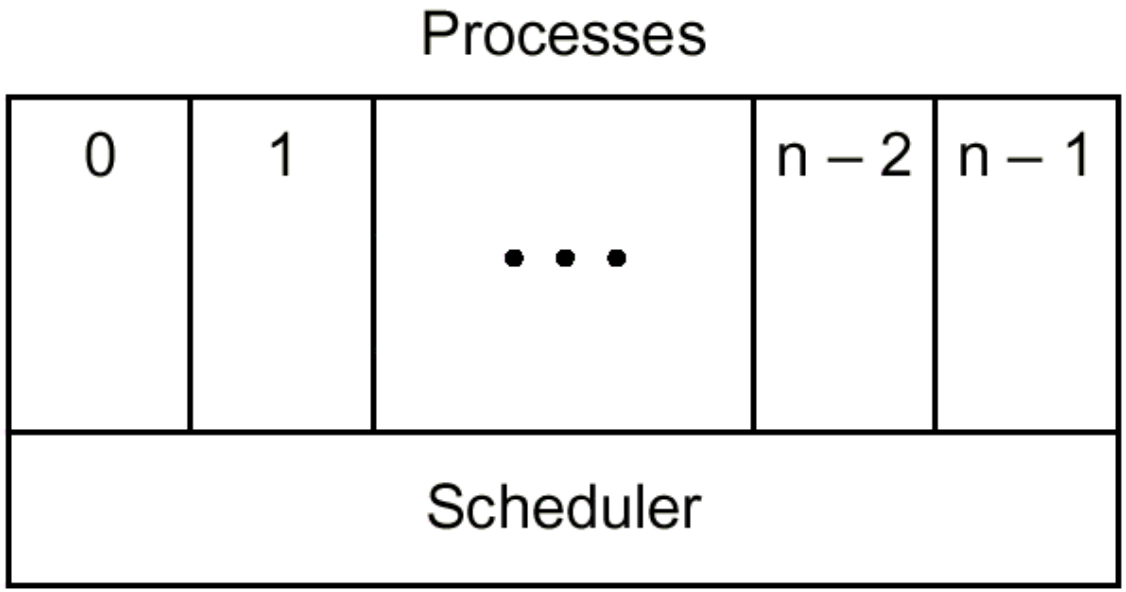


Figure 2-3. The lowest layer of a process-structured operating system handles interrupts and scheduling. Above that layer are sequential processes.

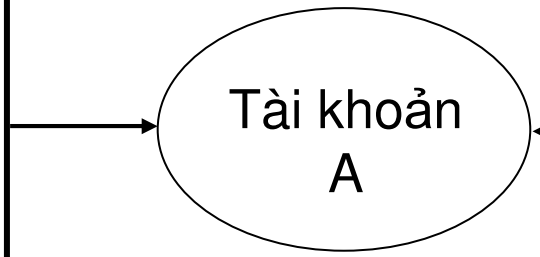
Vấn đề truy xuất 1 tài nguyên dùng chung

- Như vậy, về mặt vật lý chi ly, các chương trình không bao giờ chạy đồng thời trên 1 máy có 1 CPU vì tại từng khe thời gian, chỉ có 1 chương trình được chạy, các chương trình khác đều bị dừng chờ.
- Tuy nhiên theo góc nhìn người dùng (góc nhìn luận lý, góc nhìn ngữ nghĩa) thì ta cảm nhận rằng các chương trình chạy đồng thời.
- Nếu 2 hay nhiều chương trình chạy đồng thời và nếu chúng muốn truy xuất 1 tài nguyên (dùng chung) nào đó thì ta sẽ giải quyết ra sao ?
- Về nguyên tắc, ta phải cho chương trình truy xuất, nhưng nếu không kiểm soát việc truy xuất đồng thời vào cùng 1 tài nguyên thì có thể dẫn đến trình trạng "Race". Race là hiện tượng lỗi bất định có thể xảy ra khi 2 hay nhiều process truy xuất 1 tài nguyên dùng chung đồng thời.

Vấn đề truy xuất 1 tài nguyên dùng chung

Thí dụ 2 ứng dụng truy xuất tài khoản A đồng thời :

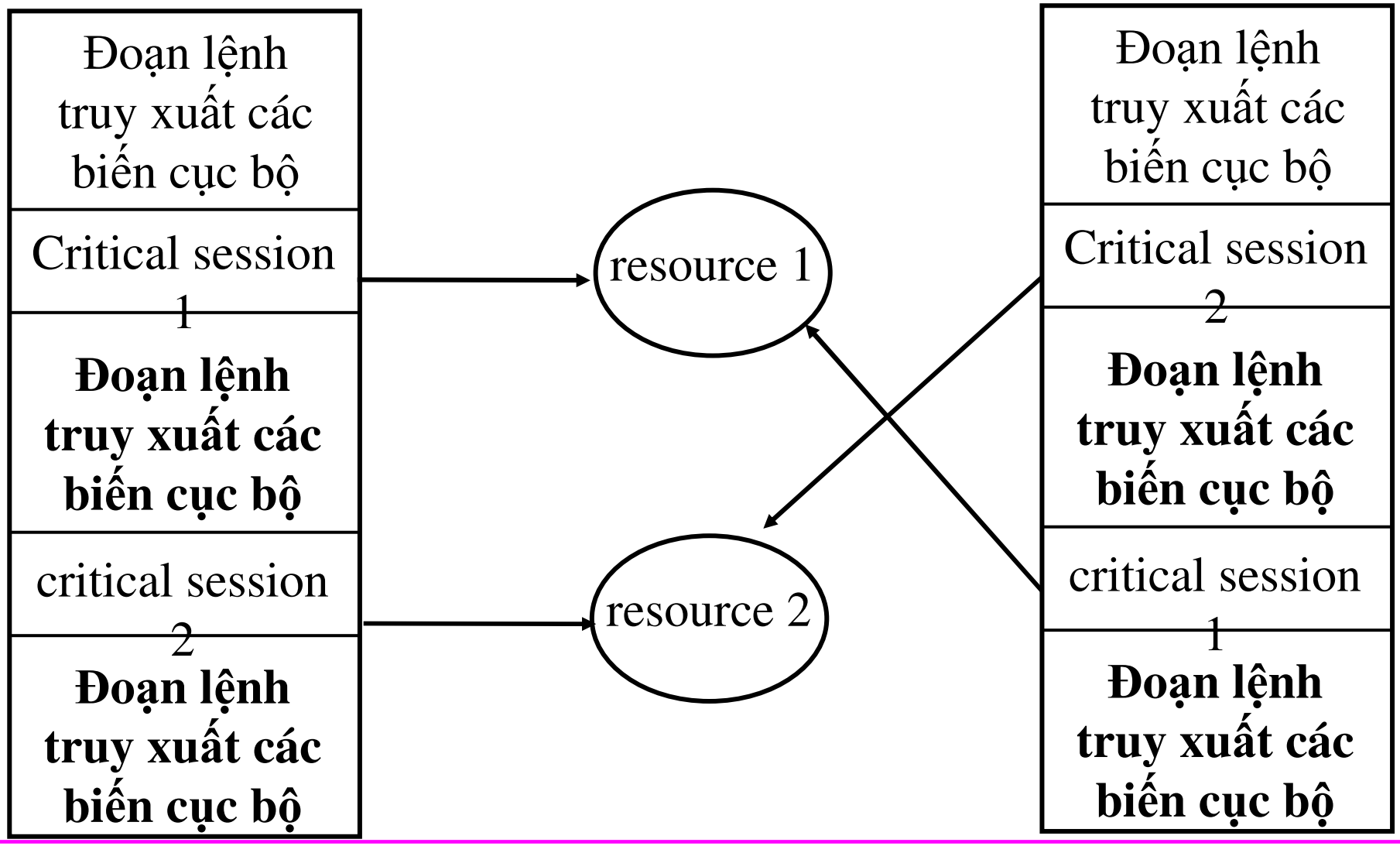
1. hiển thị giao diện & chờ người dùng ra lệnh
2. Người dùng ra lệnh nạp vào tài khoản A số tiền 700USD → xử lý :
   - 21a Đọc tài khoản A vào bộ nhớ,
   - 22a Tăng giá trị tài khoản trong bộ nhớ lên 700USD.
   - 23a Ghi lại giá trị mới.
3. Quay về bước 1



1. hiển thị giao diện & chờ người dùng ra lệnh
2. Người dùng ra lệnh rút tiền từ tài khoản A 500USD → xử lý :
   - 21b Đọc tài khoản A vào bộ nhớ,
   - 22b Giảm giá trị tài khoản trong bộ nhớ đi 500USD.
   - 23b Ghi lại giá trị mới.
3. Quay về bước 1

Nếu tài khoản A là 1000USD và HĐH điều khiển chạy 2 process P1 và P2 theo thứ tự 21a→22a→21b→22b→23b→23a thì kết quả tài khoản A sẽ là 1700USD (giá trị đúng là 1200USD).

Vấn đề truy xuất 1 tài nguyên dùng chung



Vấn đề truy xuất 1 tài nguyên dùng chung

- Để tránh tình trạng "race", ta sẽ loại trừ tương hỗ (Mutual Exclusion) các vùng code "critical session" truy xuất cùng 1 tài nguyên dùng chung của các process, nghĩa là không cho các vùng CS này chạy đồng thời mà phải tuần tự hóa việc chạy của chúng.
- Vì vùng CS thường rất hiếm trong chương trình và rất ngắn, nên việc tuần tự hóa việc chạy chúng không ảnh hưởng nhiều đến việc chạy đồng thời các process tương ứng.

Deadlock & giải quyết

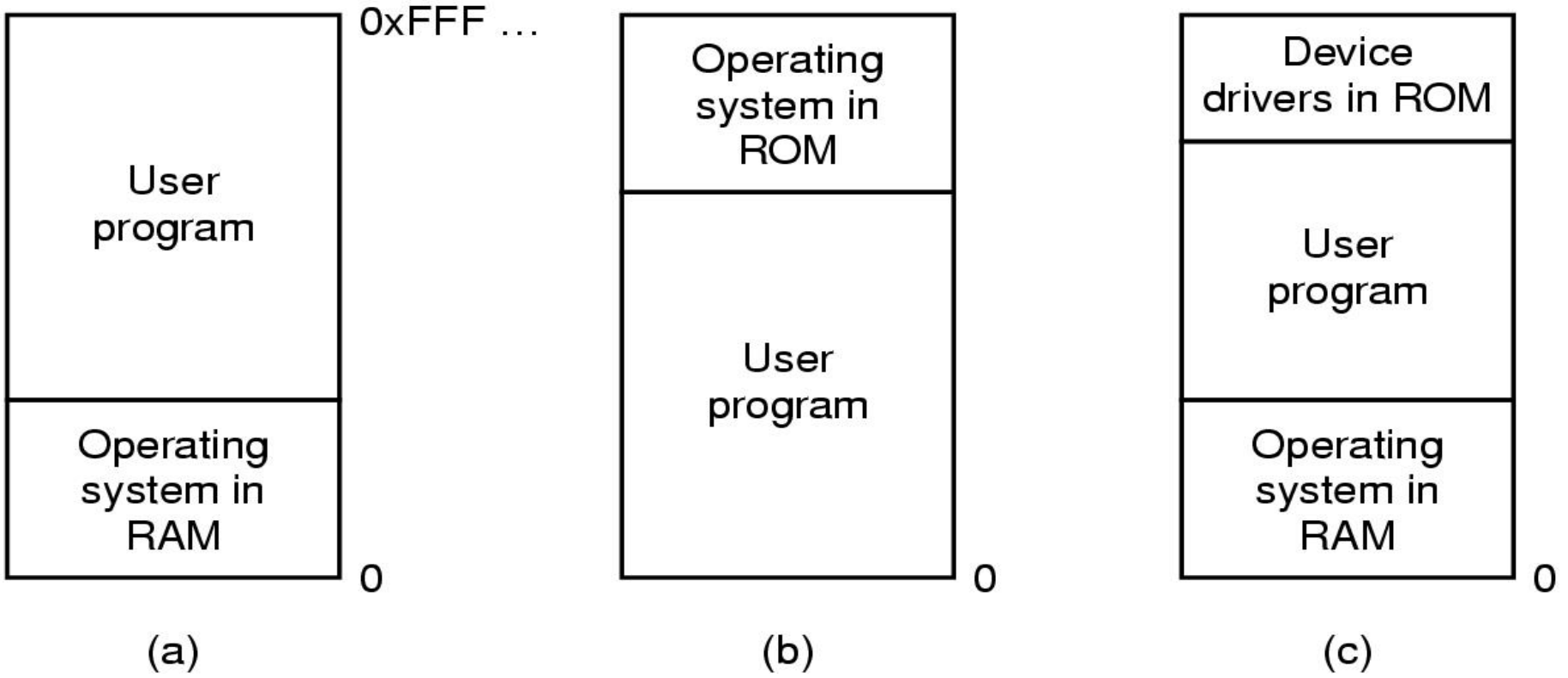
- Tuy nhiên kỹ thuật loại trừ tương hỗ các process lại thường dẫn đến mối nguy hiểm cho hệ thống mà người ta gọi là "deadlock".
- Deadlock là tình trạng của hệ thống mà ở đó có ít nhất 2 process đang dừng chờ lẫn nhau và bị kẹt mãi mãi ở trạng thái này. Trường hợp xấu nhất là mọi process đều bị dừng và chờ lẫn nhau, hệ thống sẽ bị tê liệt mãi mãi.

Deadlock & giải quyết

- Thí dụ giả sử có 2 process A và B đang chạy, theo giải thuật process A sẽ truy xuất file1 rồi file2, trong khi đó process B sẽ truy xuất file2 rồi file1 với tiến độ thời gian cụ thể như sau :
  - tại t1 : process A xin truy xuất file1 $\Rightarrow$ OK $\Rightarrow$ chạy tiếp.
  - tại t2 : process B xin truy xuất file2 $\Rightarrow$ OK $\Rightarrow$ chạy tiếp.
  - tại t3 : process A xin truy xuất file2 (vẫn còn truy xuất file1 nên chưa trả) $\Rightarrow$ không được $\Rightarrow$ phải dừng đợi process B.
  - tại t4 : process B xin truy xuất file1 (vẫn còn truy xuất file2 nên chưa trả) $\Rightarrow$ không được $\Rightarrow$ phải dừng đợi process A.
  - từ t4 trở đi : cả 2 process A và B đều bị dừng vì phải chờ lẫn nhau và chúng không bao giờ chạy được nữa.
- Cần phải giải quyết deadlock, chi tiết được trình bày trong môn HĐH.

Quản lý bộ nhớ trong hệ đơn chương

- Trong hệ đơn chương : 3 cách tổ chức bộ nhớ gồm vùng nhớ HĐH và vùng nhớ của 1 process đang chạy.

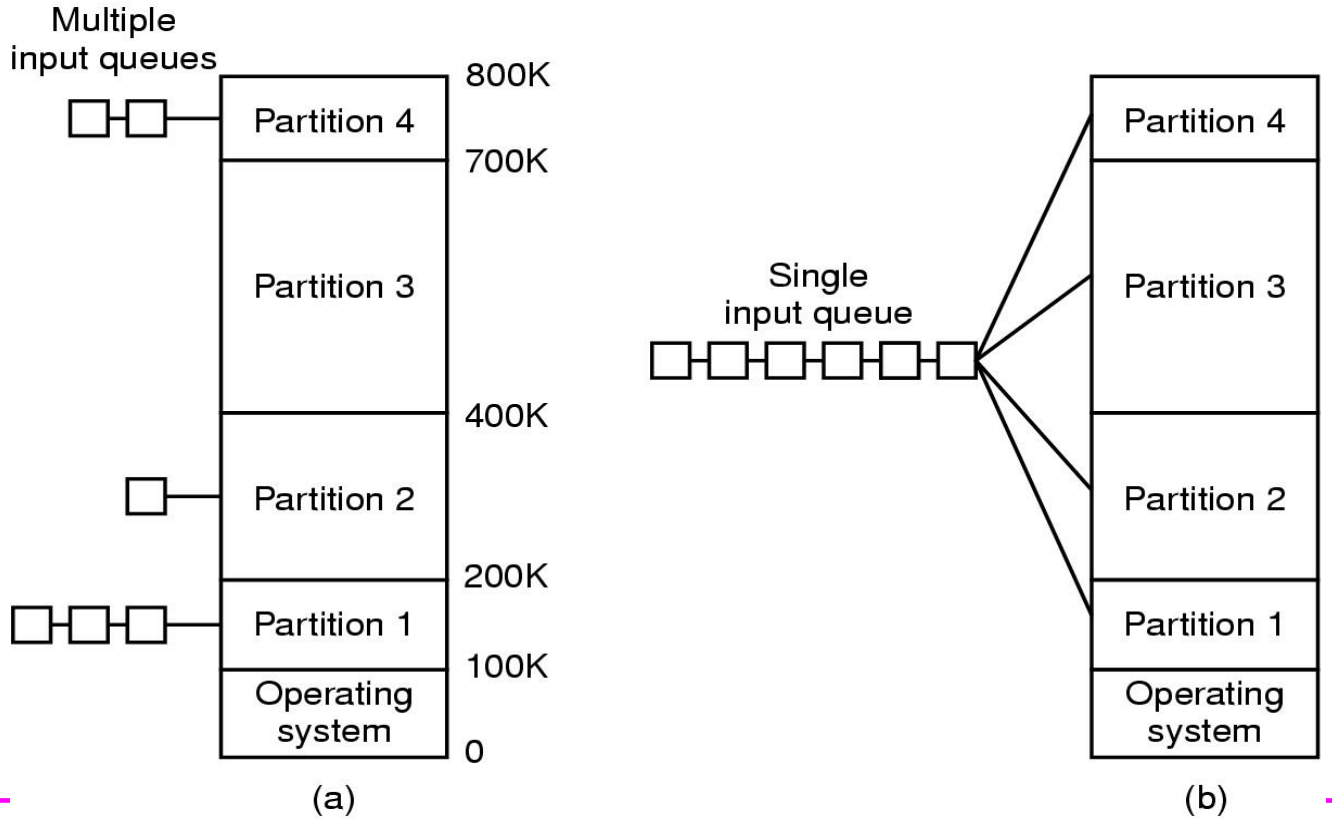


Quản lý bộ nhớ - Phân vùng tĩnh

Chia bộ nhớ ra nhiều partition với độ lớn khác nhau để chạy nhiều process đòi hỏi kích thước khác nhau.

(a) mỗi partition có hàng chờ các process đòi hỏi cùng dung lượng nhớ.

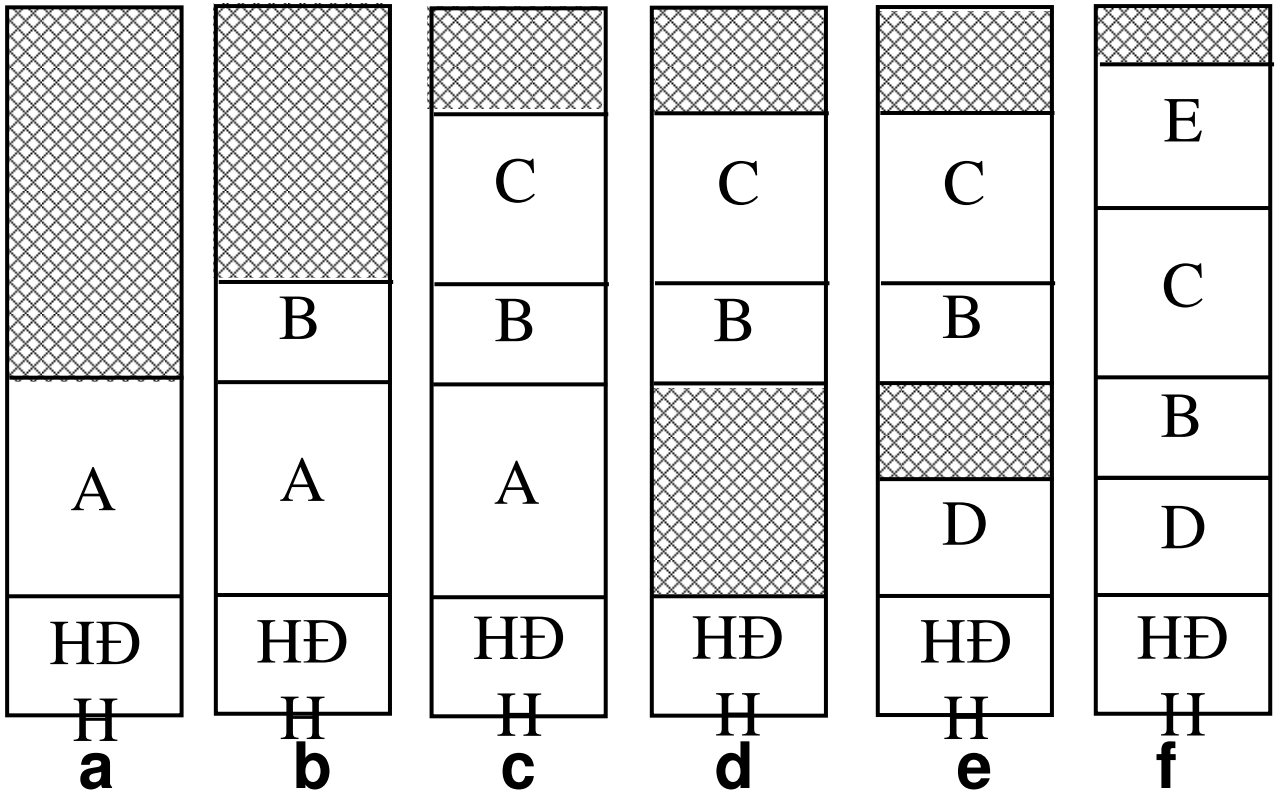
(b) dùng 1 hàng chờ cho mọi process



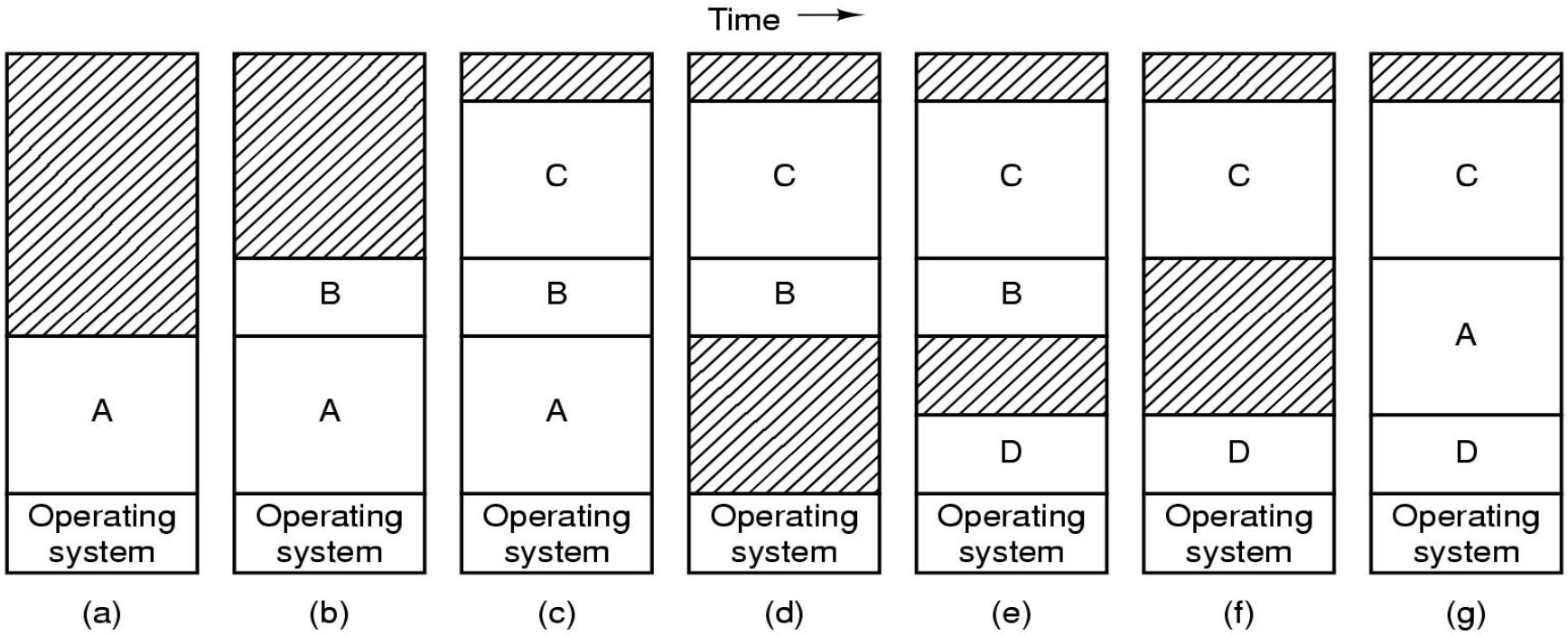
Quản lý bộ nhớ - Phân vùng động

Vùng nhớ lúc đầu để nguyên. Mỗi khi có process xin cấp phát vùng nhớ, hệ thống sẽ tạo 1 partition có kích thước vừa đúng theo yêu cầu, phần còn lại để trống... Theo thời gian, bộ nhớ có thể bị băm nát bởi nhiều vùng nhớ được trả lại bởi các process.

Ta có thể khắc phục vấn đề này bằng cách sắp xếp lại các vùng nhớ sao cho vùng nhớ trống là duy nhất & liên tục (compactage).



Quản lý bộ nhớ - Phân vùng động & Swapping



Các vùng bị thay đổi như sau :

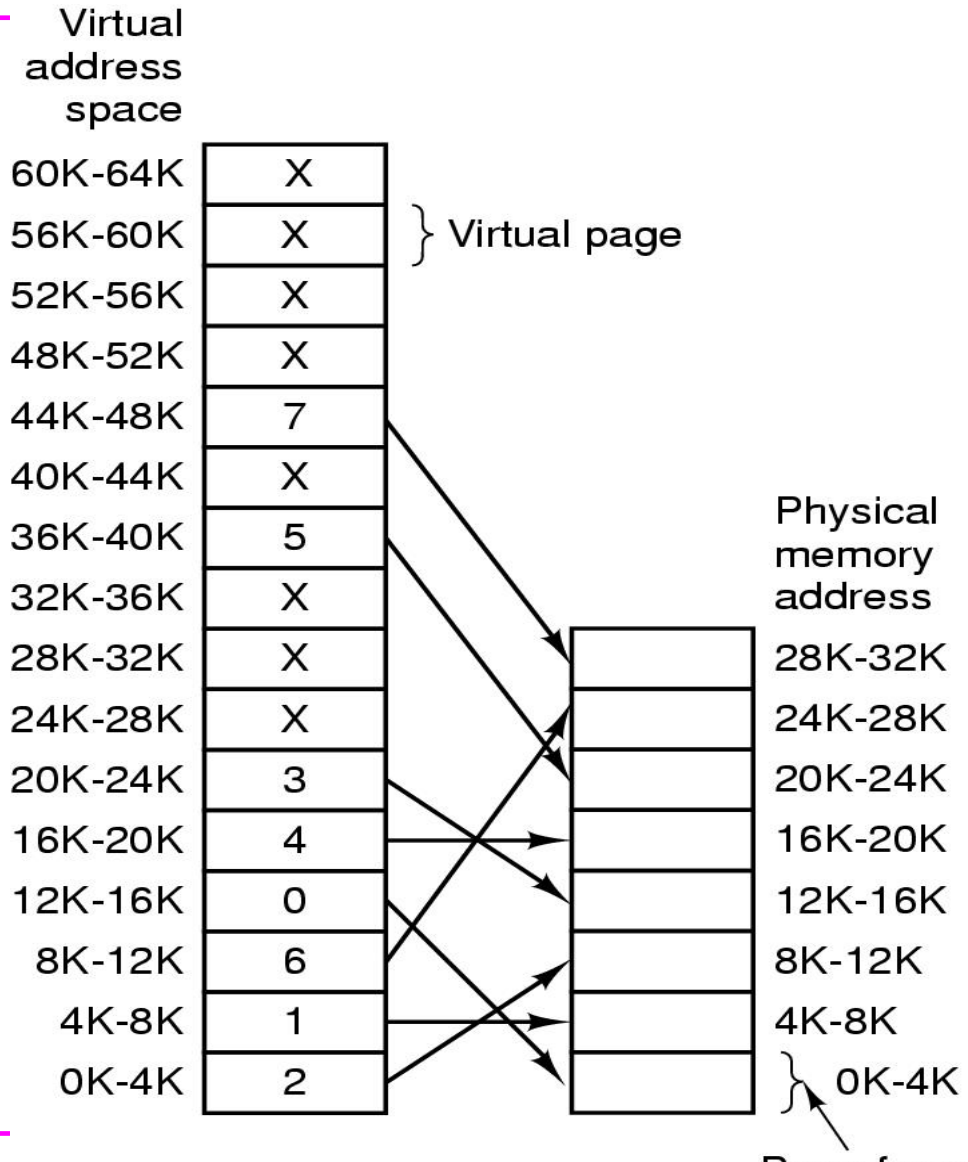
- process B được cấp 1 vùng nhớ để chạy (Fig. b)
- process A is swapped lên disk hay trả lại, Fig. (d)
- process A is swapped vào từ disk khi cần chạy tiếp, Fig. (g)

Quản lý bộ nhớ ảo (Virtual memory Man.)

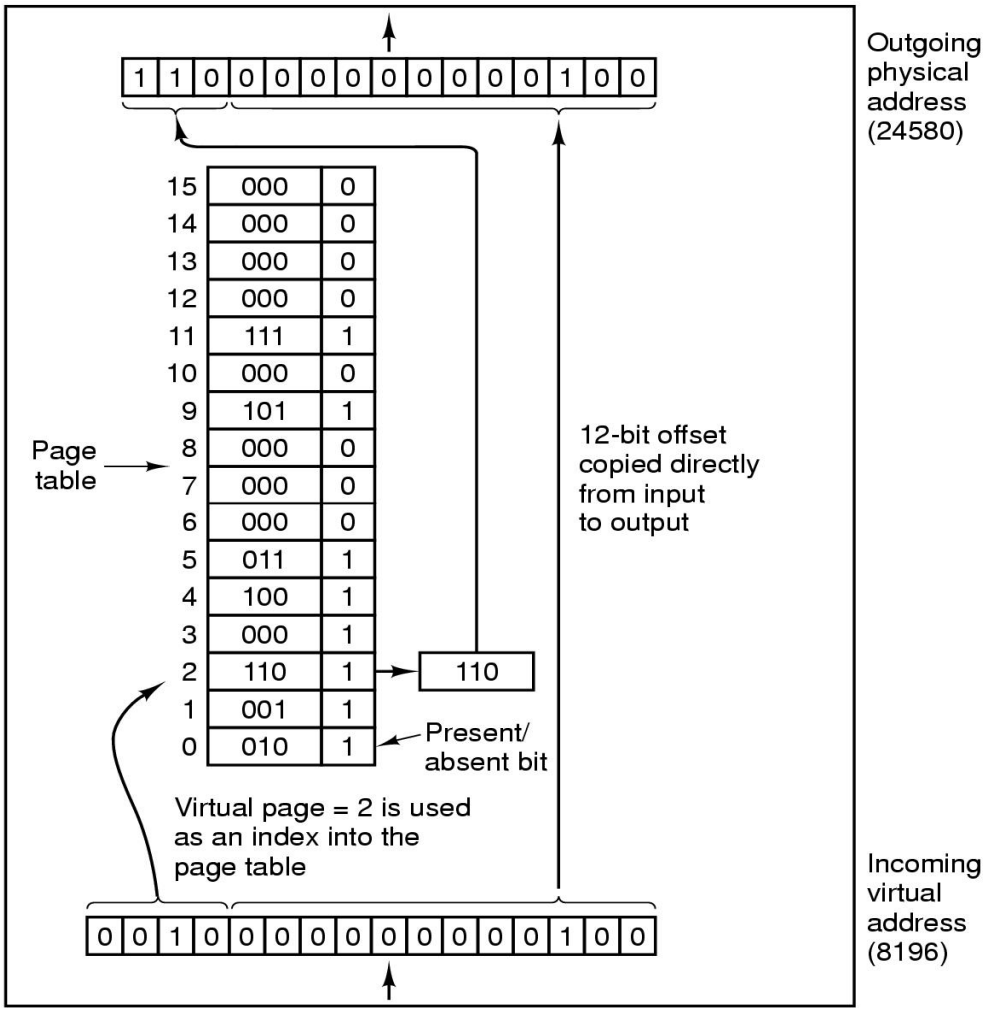
- Việc swap toàn vùng nhớ của 1 process ra/vào đĩa gặp khá nhiều phiền hà do kích thước của mỗi process rất khác nhau. Tuy nhiên ý tưởng này dẫn đến cơ chế quản lý bộ nhớ tinh vi mà hiện nay các HĐH đều dùng, đó là cơ chế quản lý bộ nhớ ảo.
- Ý tưởng cơ bản là tại từng thời điểm chương trình chạy, ta không cần nội dung của toàn chương trình và dữ liệu của nó trong bộ nhớ, ta chỉ cần nội dung của lệnh đang cần chạy và dữ liệu mà lệnh này cần truy xuất, mọi thứ khác có thể để trên đĩa.
- Như vậy để chạy được 1 process, ta chỉ cần 1 vùng rất nhỏ bộ nhớ bất chấp kích thước của process đó lớn hay nhỏ.
- Có 3 kỹ thuật quản lý bộ nhớ ảo : quản lý theo phân trang, quản lý theo phân đoạn và quản lý theo phân đoạn và phân trang. Chi tiết sẽ được trình bày trong môn HĐH.

Quản lý bộ nhớ ảo (Virtual memory Man.)

Mỗi process có bảng quản lý trang, bảng này chứa thông tin về việc ánh xạ từng trang ảo của process vào từng trang thật bộ nhớ tại từng thời điểm theo thời gian.



Đổi địa chỉ ảo ra địa chỉ thật



Hệ thống file (File System)

- disk vật lý là không gian dữ liệu 3 chiều : disk = nhiều cylinder, mỗi cyclinder gồm nhiều track (head - vòng tròn chứa tin cùng bán kính), mỗi track chứa nhiều cung nhỏ chứa tin được truy xuất độc lập (sector). Sector là đơn vị truy xuất tin nhỏ nhất ở cấp vật lý, từ ngoài ta không thể truy xuất từng byte rời rạc trên đĩa được.
- Muốn truy xuất 1 sector, ta phải xác định được bộ ba chỉ số (C,H,S) $\Rightarrow$ rất khó dùng.
- Hơn nữa, dữ liệu có nghĩa cần lưu trên đĩa thường có kích thước rất khác nhau $\Rightarrow$ cần nhiều sector mới chứa đủ. Nếu việc quản lý 1 dữ liệu có nghĩa được chứa trên bao nhiêu sector đĩa và chỉ số cụ thể là gì được giao cho người dùng thì họ sẽ gặp rất nhiều rắc rối $\Rightarrow$ cần 1 giao tiếp sử dụng khác để sử dụng đĩa dễ dàng hơn.

Hệ thống file (File System)

- disk luận lý cấp #1 là không gian dữ liệu 1 chiều : disk = danh sách nhiều đơn vị chứa tin có độ dài cố định, mỗi đơn vị được gọi là cluster (hay block, sector luận lý). Độ dài của cluster cần độc lập với đĩa vật lý.
- Ở cấp độ này, muốn truy xuất 1 cluster, ta chỉ cần xác định 1 chỉ số của nó.
- Tuy nhiên, dữ liệu có nghĩa cần lưu trên đĩa thường có kích thước rất khác nhau $\Rightarrow$ cần nhiều cluster mới chứa đủ. Nếu việc quản lý 1 dữ liệu có nghĩa được chứa trên bao nhiêu cluster đĩa và chỉ số cụ thể là gì được giao cho người dùng thì họ sẽ gặp rất nhiều rắc rối $\Rightarrow$ cần 1 giao tiếp sử dụng khác để sử dụng đĩa dễ dàng hơn.

Hệ thống file (File System)

- disk luận lý cấp #2 là không gian dữ liệu 1 chiều : disk = danh sách nhiều đơn vị chứa tin có độ dài thay đổi theo yêu cầu của người dùng, mỗi đơn vị được gọi là file và được nhận dạng bằng tên gợi nhớ chứ không phải là chỉ số khó nhớ.
- Ở cấp độ này, muốn truy xuất 1 file, ta chỉ cần xác định tên gợi nhớ của nó.
- Dù dữ liệu có nghĩa cần lưu trên đĩa thường có kích thước rất khác nhau, nhưng chỉ cần 1 file là đủ để lưu 1 dữ liệu có nghĩa $\Rightarrow$ Việc quản lý dữ liệu trên đĩa trở nên dễ dàng hơn nhiều so với trước.
- Tuy nhiên vì 1 đĩa chứa 1 số rất lớn file (hàng triệu file) $\Rightarrow$ nếu dùng không gian phẳng để tổ chức các file thì cũng còn nhiều khó khăn trong việc đặt tên file, việc phân biệt các file của chương trình nào, của người nào $\Rightarrow$ cần 1 giao tiếp sử dụng khác để sử dụng đĩa dễ dàng hơn nữa.

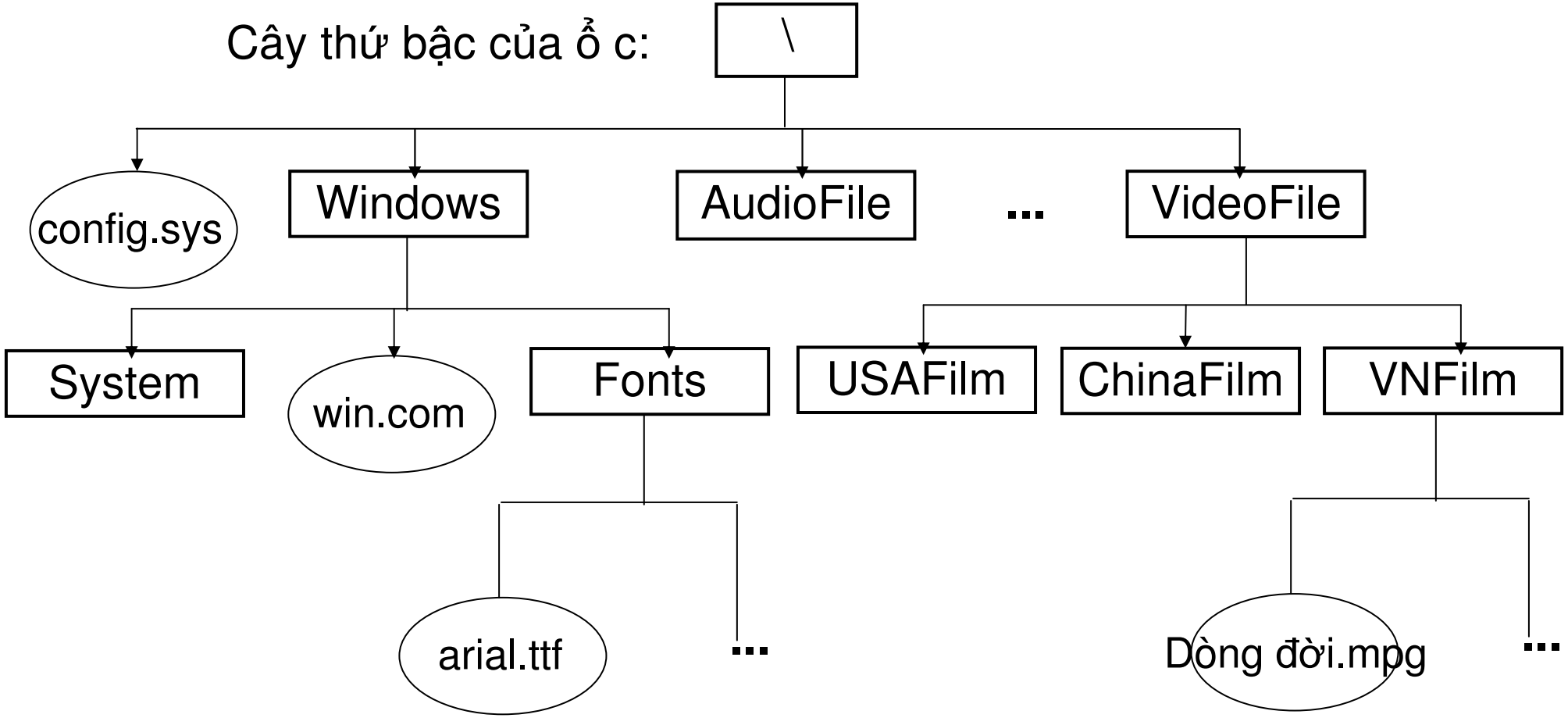
Hệ thống file (File System)

- disk luận lý cấp #3 là không gian dữ liệu dạng cây phân cấp : disk = thư mục gốc chứa nhiều phần tử con, mỗi phần tử con có thể là file hay thư mục khác...
- Trong cấp độ này, ta nhận dạng 1 phần tử bằng khái niệm đường dẫn (pathname). Có 2 loại pathname : tuyệt đối và tương đối. Tùy thuộc vào nhu cầu sử dụng cụ thể mà dạng nào sẽ thích hợp hơn.

Đường dẫn tuyệt đối và tương đối

- Đường dẫn (pathname) là thông tin để tìm kiếm (xác định) 1 phần tử từ 1 vị trí nào đó, nó chứa danh sách chính xác các tên gợi nhớ của các phần tử mà ta phải đi qua xuất phát từ vị trí đầu để đến phần tử cần tìm.
- ta dùng 1 dấu ngăn đặc biệt để ngăn cách 2 tên gợi nhớ liên tiếp nhau trong đường dẫn (trong Windows, dấu ngăn là '\')
- Tên thư mục gốc luôn là '\'.
- Có 2 khái niệm đường dẫn : đường dẫn tuyệt đối và đường dẫn tương đối. Đường dẫn tuyệt đối là đường dẫn xuất phát từ thư mục gốc, đường dẫn tương đối xuất phát từ thư mục làm việc (working directory).
- Trước khi ứng dụng bắt đầu chạy, hệ thống sẽ khởi động thư mục làm việc cho ứng dụng (theo cơ chế nào đó). Trong quá trình thực thi, ứng dụng có quyền thay đổi thư mục làm việc theo yêu cầu riêng.

Thí dụ về hệ thống file



Đường dẫn tuyệt đối và tương đối (tt)

- Xét cây thứ bậc của ổ c: trên slide trước, đường dẫn tuyệt đối sau sẽ nhận dạng chính xác file arial.ttf trong thư mục 'Fonts' :

  c:\Windows\Fonts\arial.ttf

- Nếu thư mục working của chương trình hiện là c:\Windows\Fonts thì ta có thể dùng đường dẫn tương đối sau đây để xác định file arial.ttf :

  arial.ttf

- Đường dẫn tuyệt đối thường dài hơn đường dẫn tương đối nhưng nó luôn có giá trị bất chấp ứng dụng đang ở thư mục working nào.
- Đường dẫn tương đối thường gọn hơn (đa số chỉ chứa tên file cần truy xuất vì ứng dụng sẽ thiết lập thư mục working là thư mục chứa các file mà ứng dụng truy xuất) nhưng chỉ có giá trị với 1 thư mục working cụ thể.
- Trong 1 vài trường hợp đặc biệt, ta phải dùng đường dẫn tương đối ngay cả nó dài và phức tạp hơn đường dẫn tuyệt đối!

Giao tiếp với thế giới ngoài

- chương trình khi hoạt động thỉnh thoảng phải giao tiếp với thế giới ngoài (thí dụ cần in ra máy in, cần giao tiếp mạng, cần truy xuất thông tin của các sensor đo thông số,..). Máy tính sẽ dùng 1 card chức năng (card I/O) để giao tiếp với thế giới ngoài.
- Có rất nhiều hãng sản xuất, mỗi hãng sản xuất rất nhiều model card I/O khác nhau, để đoạn code chương trình giao tiếp với I/O độc lập hoàn toàn với tính chất phần cứng của card I/O, ta sẽ xây dựng 1 module phần mềm đặc biệt : device driver. Mỗi card I/O có device driver riêng. Device driver phải chứa n hàm chức năng theo qui định của HĐH, chi tiết của từng hàm chức năng sẽ phụ thuộc vào phần cứng, còn việc sử dụng các hàm chức năng trong chương trình thì hoàn toàn độc lập với phần cứng.

An ninh hệ thống

- Máy tính có rất nhiều tài nguyên và cho phép nhiều người truy xuất $\Rightarrow$ Cần phải có cơ chế đảm bảo việc dùng tài nguyên bởi các người dùng, không cho phép việc truy xuất bất hợp pháp.
- An ninh hệ thống bao gồm 3 vấn đề chính :
  - Bảo mật dữ liệu : mỗi người chỉ được phép truy xuất 1 số tài nguyên qui định, không có khả năng truy xuất các tài nguyên khác.
  - Toàn vẹn dữ liệu : việc truy xuất tài nguyên của người dùng không được làm hư hỏng dữ liệu, dù chỉ 1 phần nhỏ.
  - Sẵn sàng dữ liệu : việc truy xuất tài nguyên của người dùng hợp pháp phải luôn được phục vụ trong khoảng thời gian ngắn nhất, bất cứ lúc nào, bất cứ ở đâu.
- Các biện pháp để bảo mật dữ liệu là quản lý người dùng theo account và mật mã hóa dữ liệu.

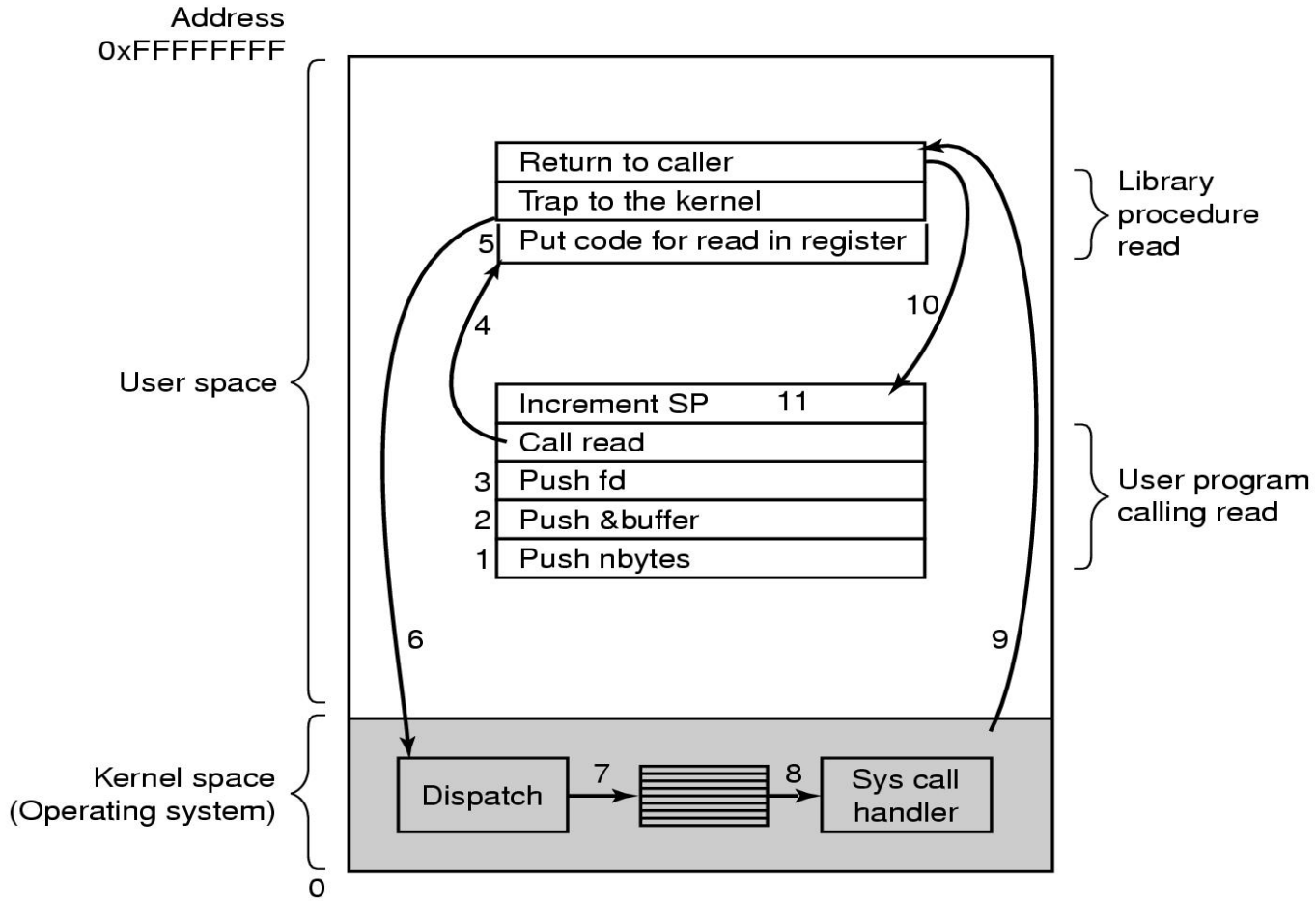
3.6 Các lời gọi dịch vụ HĐH "System call"

- HĐH cung cấp 1 giao tiếp sử dụng được gọi là "System Call", mỗi system call là 1 hàm thực hiện 1 chức năng xác định.
- Thường chỉ có code chương trình mới gọi System call, còn người dùng đầu cuối không thể gọi system call trực tiếp được.
- Người dùng đầu cuối sử dụng các dịch vụ HĐH gián tiếp thông qua từng ứng dụng cụ thể. Thí dụ để thực hiện các chức năng quản lý hệ thống file, người dùng trên Windows sẽ dùng trình "Windows Explorer", thông qua giao tiếp sử dụng đồ họa trực quan của chương trình, người dùng sẽ thực hiện các thao tác quản lý hệ thống file rất dễ dàng (duyệt file, tạo/hiệu chỉnh/xóa file/thư mục, di chuyển file/thư mục từ nơi này đến nơi khác,...)

Các lời gọi dịch vụ HĐH "System call"

- Gọi system call gần giống với gọi hàm bình thường, sự khác biệt lớn nhất là có sự thay đổi quyền truy xuất tài nguyên :
  - trước khi gọi system call, các lệnh của chương trình ứng dụng có quyền thấp.
  - khi gọi system call, quyền sẽ thay đổi thành rất cao (quyền hệ thống) để đoạn code của hàm "system call" có thể thực thi được chức năng đặc biệt của mình.
  - Sau khi gọi system call xong, quyền truy xuất được trả về mức cũ (thấp) của ứng dụng để đoạn code đi theo sau lệnh gọi system call chạy như cũ.

Các lời gọi dịch vụ HĐH "System call"



There are 11 steps in making the system call read(fd, buffer, nbytes)

Các lời gọi dịch vụ HĐH "System call"

Thí dụ 1 số lời gọi dịch vụ HĐH "System call" trên Windows và Linux :

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

3.7 Kiến trúc của HĐH

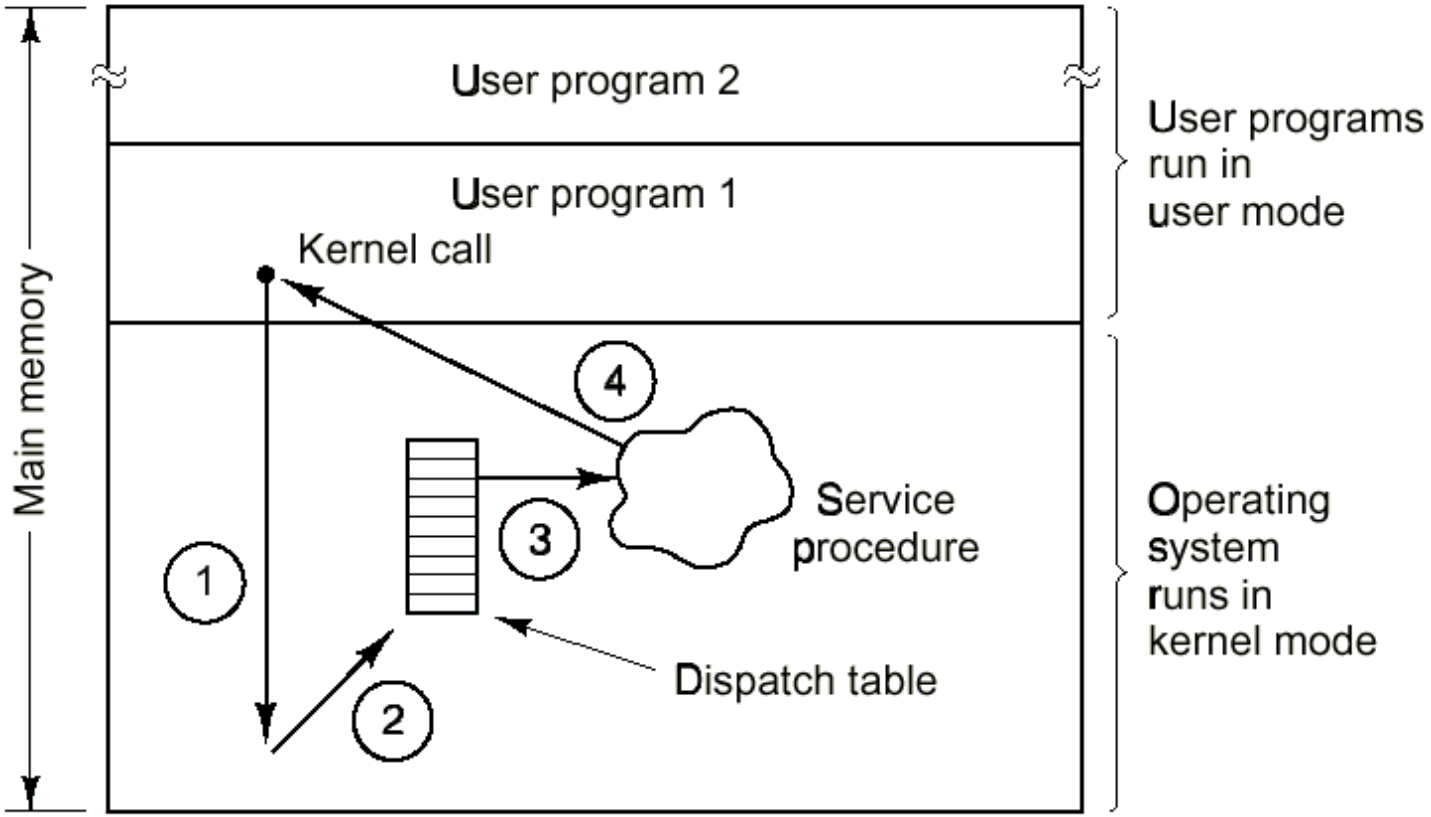
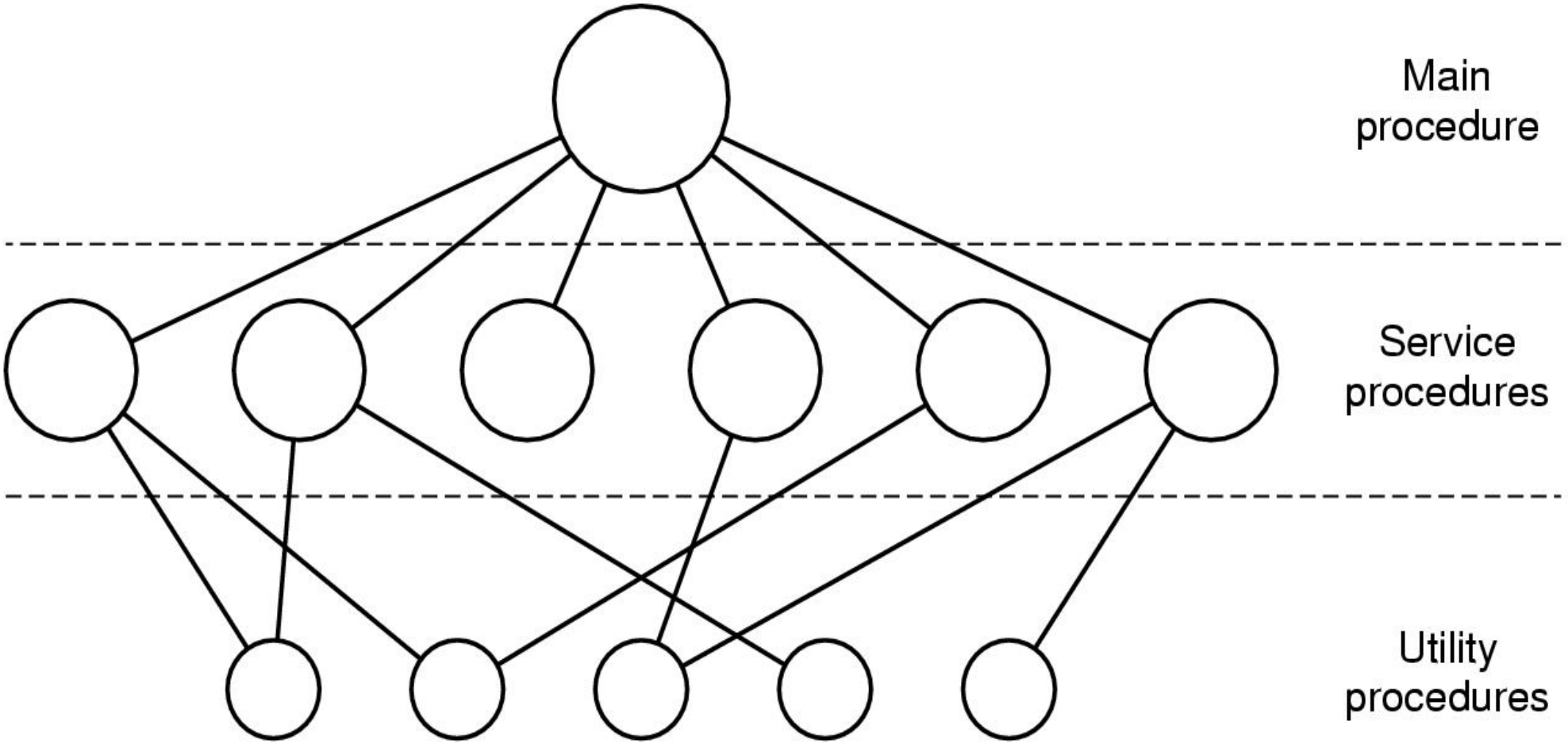
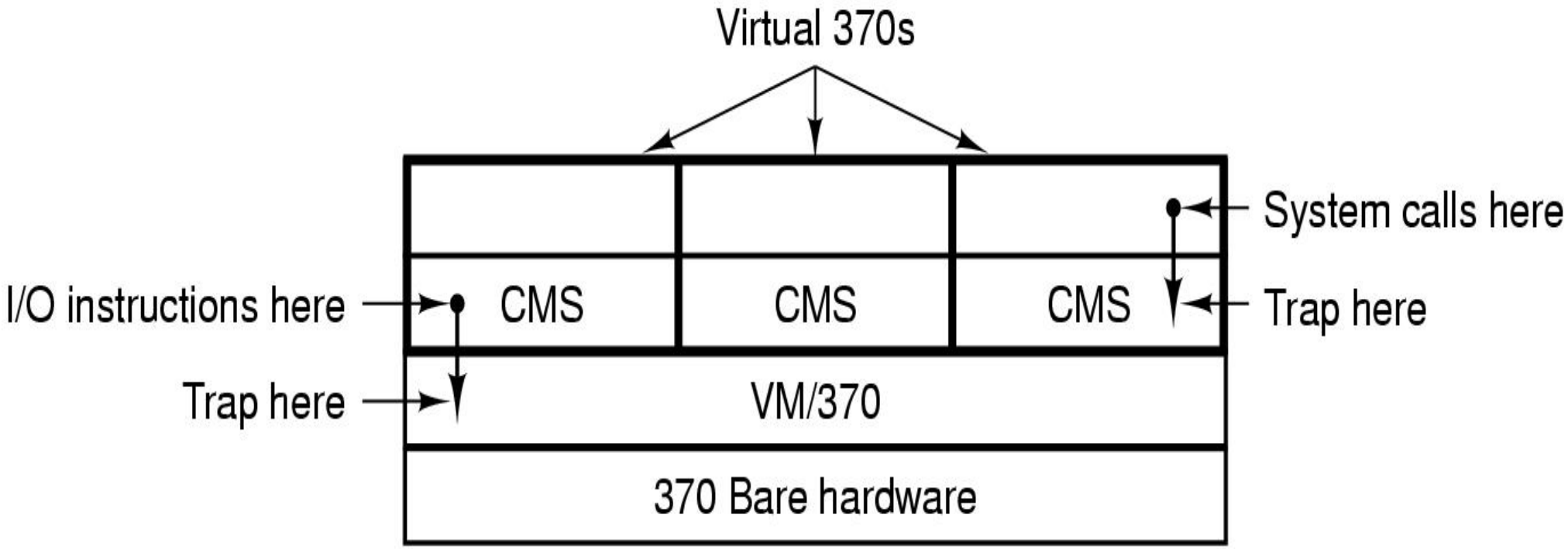


Figure 1-16. How a system call can be made: (1) User program traps to the kernel. (2) Operating system determines service number required. (3) Operating system calls service procedure. (4) Control is returned to user program.

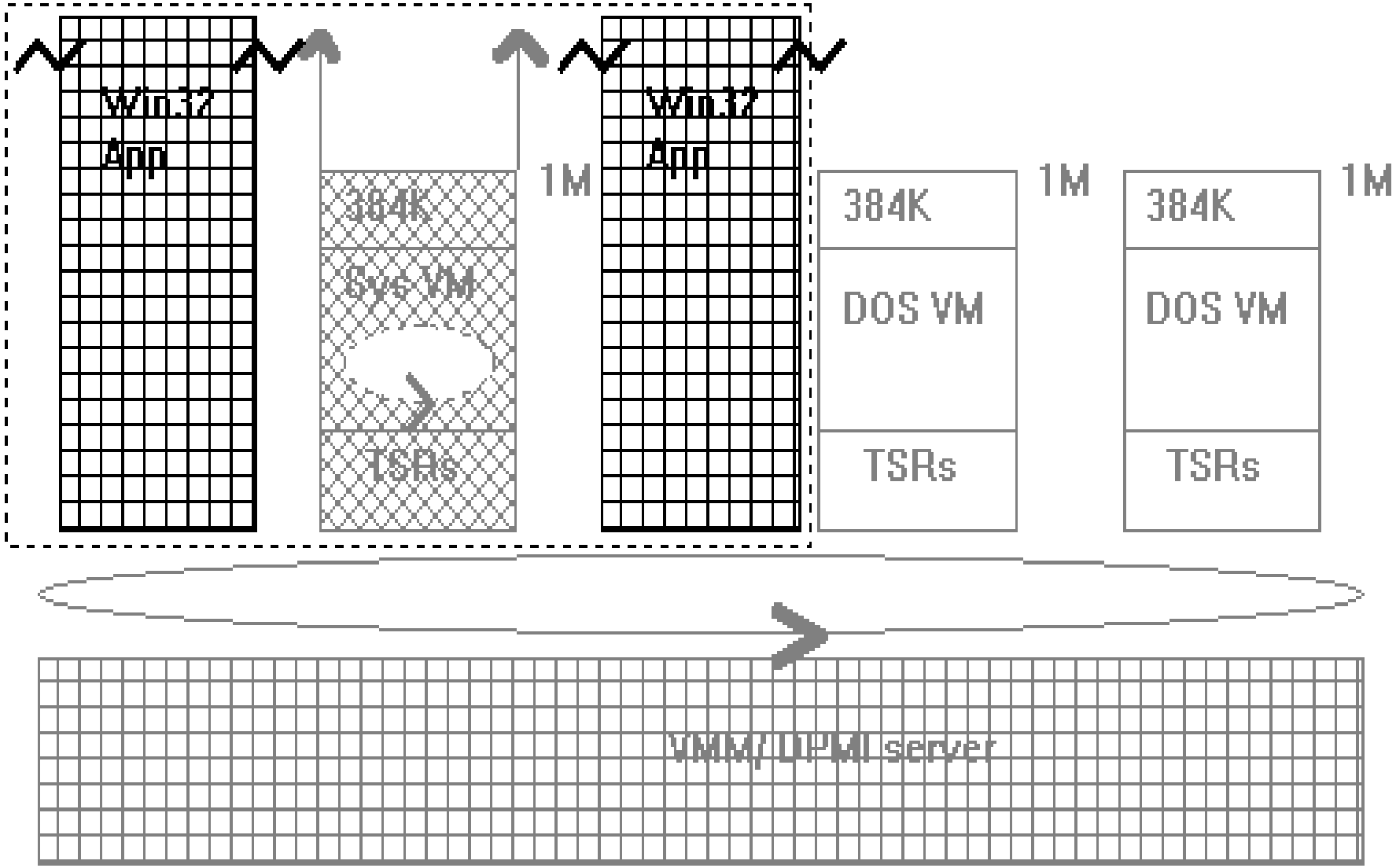
Kiến trúc phân cấp



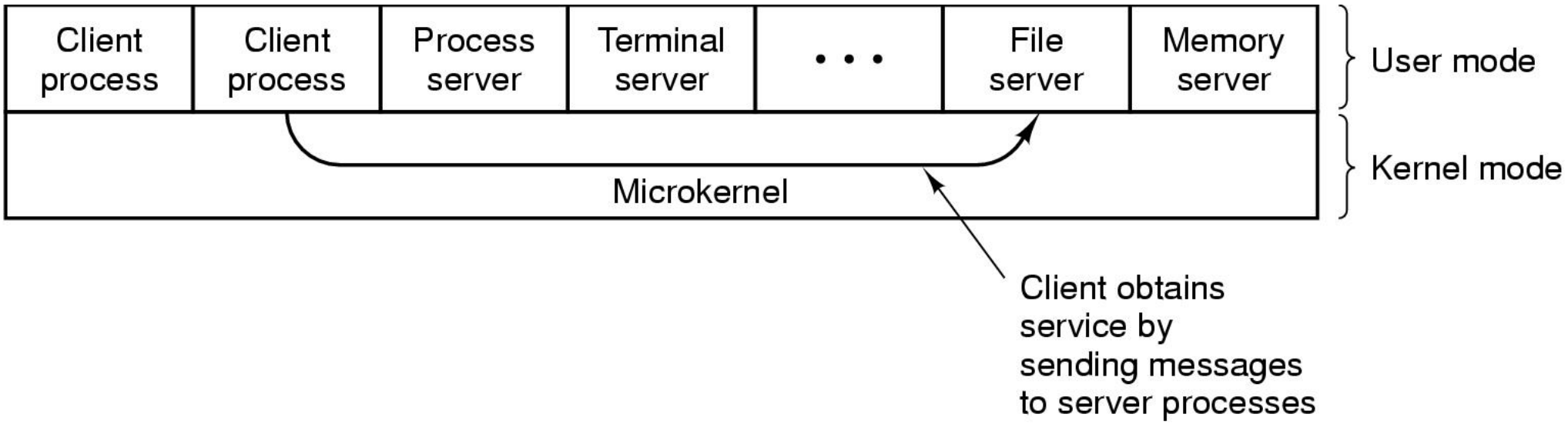
Kiến trúc máy ảo



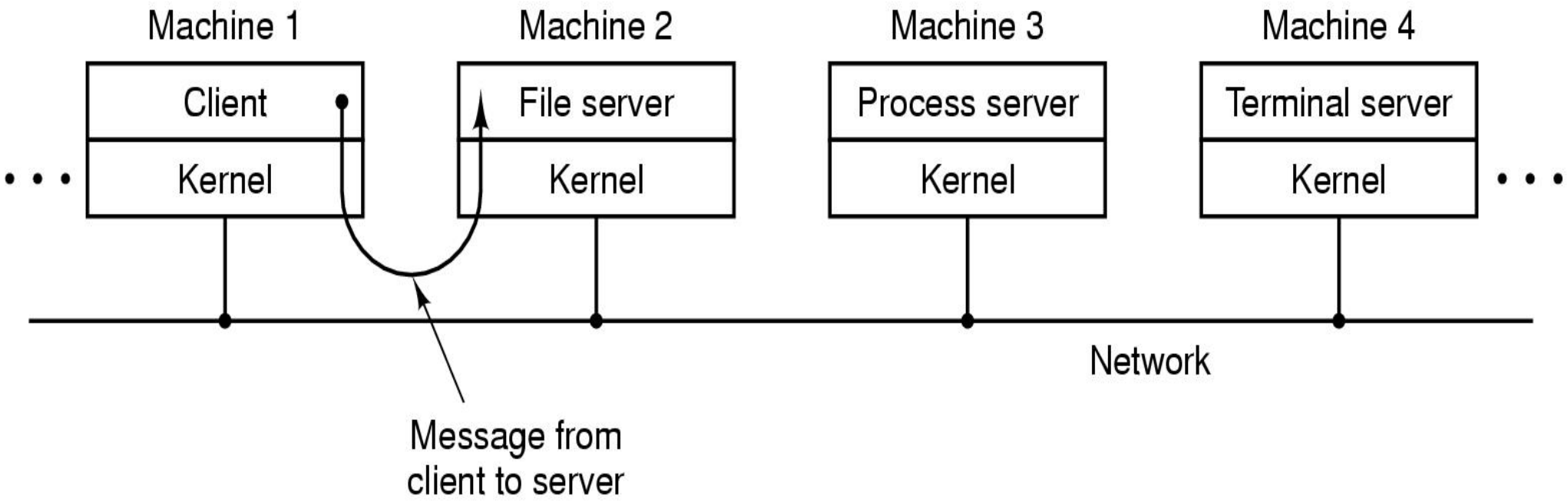
Thí dụ về kiến trúc máy ảo



Kiến trúc client-server



Kiến trúc client-server



Chương 4

LẬP TRÌNH

4.1 Lập trình với ngôn ngữ cấp cao
4.2 Xử lý ngôn ngữ
4.3 Phát triển phần mềm
4.4 Tài liệu hoá chương trình

4.1 Lập trình với ngôn ngữ cấp cao

- Ngôn ngữ lập trình:
  - Trong chương 3, ta đã thấy máy tính số là máy nhiều cấp, mỗi cấp là 1 máy tính (vật lý hay luận lý) thực hiện được tập lệnh máy của cấp mình.
  - Về nguyên lý, bất kỳ bài toán (vấn đề) cần giải quyết ngoài đời nào cũng có thể được miêu tả chính xác thành 1 chuỗi các lệnh máy (thuộc 1 máy luận lý xác định). Chuỗi các lệnh máy này được gọi là chương trình (program) giải quyết bài toán tương ứng.
  - Lập trình (programming) hay tổng quát hơn là phát triển phần mềm (software developping) là qui trình thực hiện các công việc để tạo được chương trình cụ thể từ 1 bài toán cần giải quyết.
  - Chương trình được miêu tả bằng 1 ngôn ngữ cụ thể. Ta gọi ngôn ngữ được dùng để miêu tả chương trình là ngôn ngữ lập trình, đây là ngôn ngữ mà máy tính (ở cấp tương ứng) hiểu và thực thi được.

Ngôn ngữ máy

- Ngôn ngữ máy :
  - Ta thường dùng thuật ngữ "ngôn ngữ máy" để nói về ngôn ngữ của máy tính vật lý mà người dùng có thể lập trình được (còn có ngôn ngữ máy thấp hơn nữa như vi lệnh)
- Lệnh máy :
  - Mỗi lệnh máy chỉ thực hiện một tác vụ rất đơn giản như 1 phép tính số học hay 1 hoạt động đọc/ghi vùng nhớ/thanh ghi CPU.
  - Một lệnh máy bao gồm 2 phần : mã lệnh và toán hạng. Mã lệnh (opcode) là một chuỗi các bit 0 và 1. Mỗi chuỗi bit miêu tả 1 số, mỗi số miêu tả 1 lệnh máy cụ thể. Thí dụ máy có n lệnh (n <256), ta có thể miêu tả mỗi lệnh máy bằng 1 byte (8bit), byte này được gọi là mã lệnh. Toán hạng xác định dữ liệu nào sẽ bị xử lý bởi lệnh máy tương ứng. Toán hạng cũng là chuỗi bit nhị phân, nhưng định dạng và ngữ nghĩa của nó phụ thuộc vào từng lệnh máy cụ thể.

Ví dụ về ngôn ngữ máy

Giả sử ta có 2 biến nguyên 16 bit, biến nguyên thứ nhất (i) nằm ở vị trí nhớ 200h, biến nguyên thứ 2 (j) nằm ở vị trí nhớ 202h. Đoạn lệnh máy (Intel 80x86) sau đây sẽ thiết lập nội dung cho biến i = 5 rồi thiết lập nội dung của biến j theo công thức i+10 :

10111000 00000101 00000000	b8 05 00
10100011 00000000 00000002	a3 00 02
10100001 00000000 00000002	a1 00 02
00000101 00001010 00000000	05 0a 00
10100011 00000010 00000010	a3 02 02

$\Rightarrow$Con người rất khó lập trình (rất khó viết và đọc) giải quyết bài toán ngoài đời (thường khá phức tập) trực tiếp bằng ngôn ngữ máy vì quá xa lạ với ngôn ngữ tự nhiên mà con người đã từng dùng.

Ngôn ngữ lập trình cấp thấp

- Cấu trúc điều khiển : Một cấu trúc ngôn ngữ quy định thứ tự thực hiện các lệnh trong chương trình.
- Ngôn ngữ máy chỉ có hai cấu trúc điều khiển cơ bản để thực hiện các lệnh : tuần tự và nhảy. Cấu trúc tuần tự là mặc định : sau khi thực hiện xong lệnh máy hiện hành sẽ thi hành tiếp lệnh đi ngay sau lệnh hiện hành trong chương trình. Lệnh nhảy cho phép người lập trình xác định lệnh kế tiếp được thi hành ở đâu trong chương trình. Đ số các lệnh nhảy đều có kèm theo điều kiện (kết quả vừa tính là âm/bằng 0/dương...
- Ta dùng thuật ngữ "ngôn ngữ lập trình cấp thấp" để miêu tả các ngôn ngữ của các máy nằm thấp dưới đáy chồng các máy nhiều cấp. Thí dụ ngôn ngữ máy là ngôn ngữ lập trình cấp thấp.

Ngôn ngữ lập trình cấp cao

- Tương tự, ta dùng thuật ngữ "ngôn ngữ lập trình cấp cao" để miêu tả các ngôn ngữ của các máy nằm cao trên chồng các máy nhiều cấp. Thí dụ ngôn ngữ C# là ngôn ngữ lập trình cấp cao.
- Ngôn ngữ lập trình cấp cao cho phép dùng nhiều kiểu diễu dữ liệu và nhiều cấu trúc điều khiển hơn so với những gì được cung cấp bởi ngôn ngữ cấp thấp, đồng thời cách biểu diễn các lệnh (phát biểu) cũng gần với ngôn ngữ tự nhiên hơn.
- Phân loại các ngôn ngữ lập trình cấp cao :
  - Ngôn ngữ đ mục đích: Basic, C, C++, C#, Java, Fortran, Pascal
  - Ngôn ngữ lập trình stack : TrueType, Postscript,...
  - Lập trình khai báo : C, Pascal,...
  - Ngôn ngữ lập trình logic, lập trình thủ tục & lập trình hàm : Prolog, Lisp,..
  - Ngôn ngữ lập trình hướng đối tượng : C++, C#, Java,..

Ví dụ về ngôn ngữ lập trình cấp cao : C

Ngôn ngữ máy dạng nhị phân	NNM dạng Hex	NN Assembly
10111000 00000101 00000000	b8 05 00	mov ax, 5
10100011 00000000 00000002	a3 00 02	mov [200], ax
10100001 00000000 00000002	a1 00 02	mov ax, [200]
00000101 00001010 00000000	05 0a 00	add ax, 10
10100011 00000010 00000010	a3 02 02	mov [202],ax

Ngôn ngữ cấp cao C :

```
short i, j;      // khai báo 2 biến i, j thuộc kiểu nguyên 16 bit
i = 5;           // chứa 5 vào biến i
j = i +10;       // chứa kết quả tính công thức i + 10 vào biến j
```

Đánh giá :

- Con người rất khó viết và đọc chương trình viết bằng ngôn ngữ máy (dù ở dạng nhị phân hay ở dạng hexadecimal).
- Nhưng nếu ở dạng assembly, con người dễ dàng viết và đọc hơn.
- Và nếu ở dạng ngôn ngữ cấp cao, con người sẽ rất dễ dàng viết và đọc.

⇒ Con người cố gắng định nghĩa nhiều ngôn ngữ cấp cao và dùng ngôn ngữ cấp cao để viết chương trình.

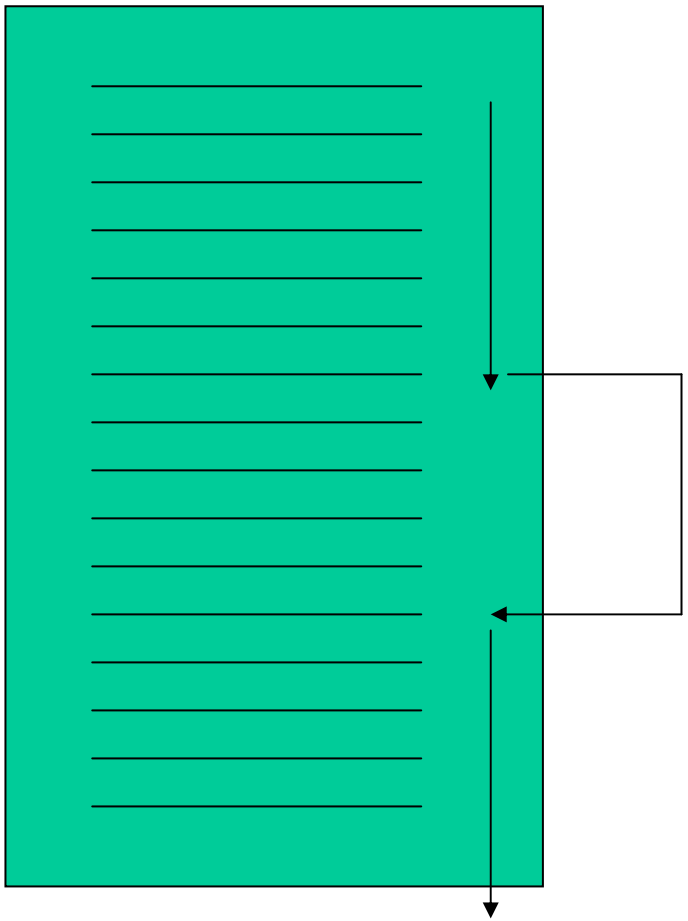
Cấu trúc điều khiển

- Cấu trúc điều khiển : Một cấu trúc ngôn ngữ quy định thứ tự thực hiện các lệnh
- Ngôn ngữ máy : Tuần tự và nhảy
- Ngôn ngữ cấp cao cung cấp thêm :
  - Rẽ nhánh
  - Lặp

Cấu trúc tuần tự và nhảy

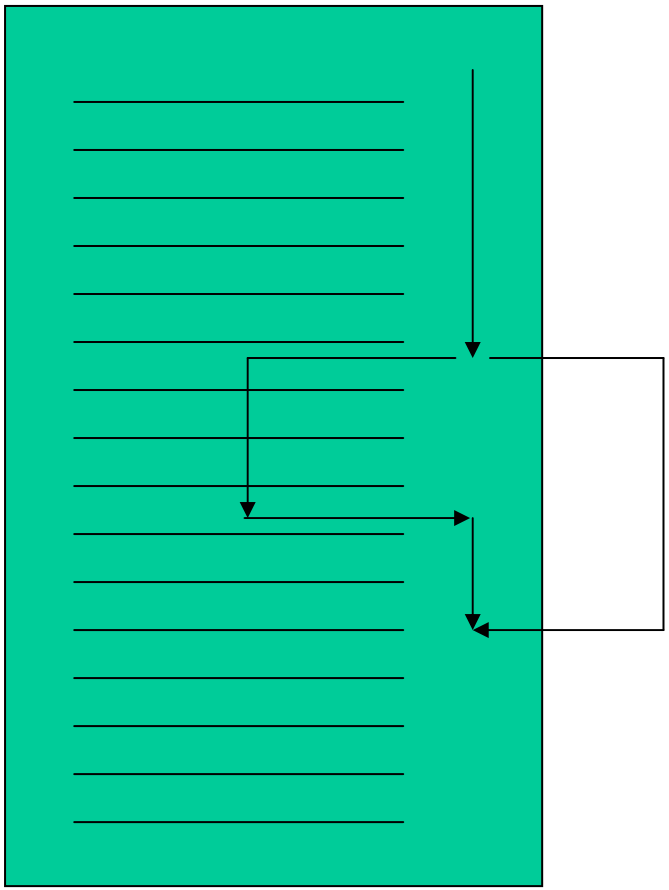
```
    A = 1;              // tuần tự
    Goto Lable1;        //nhảy
    A = A*2;            // tuần tự
Label1:
    A = 3               // tuần tự
```

A=?



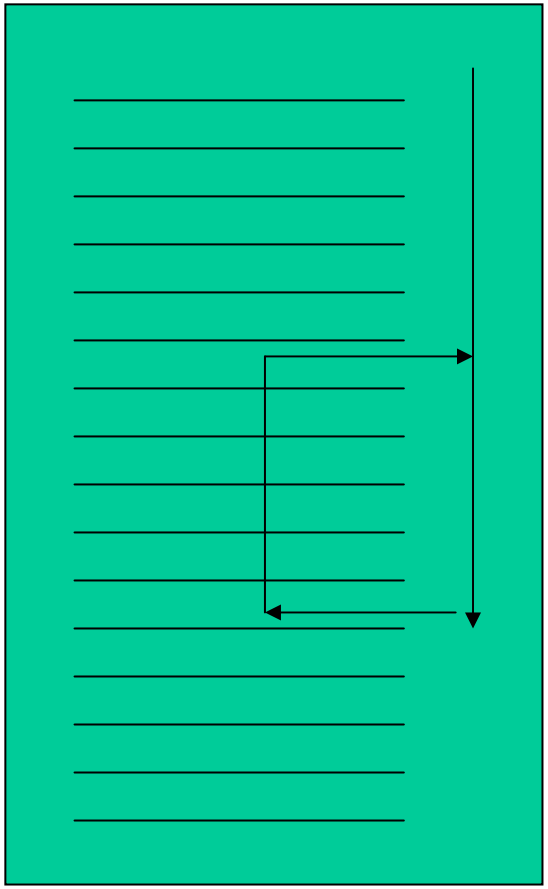
Cấu trúc rẽ nhánh

```
if (x < y) {
        printf ("x is smaller"); //nhánh 1
} else {
        printf ("x is greater"); //nhánh 2
}
```



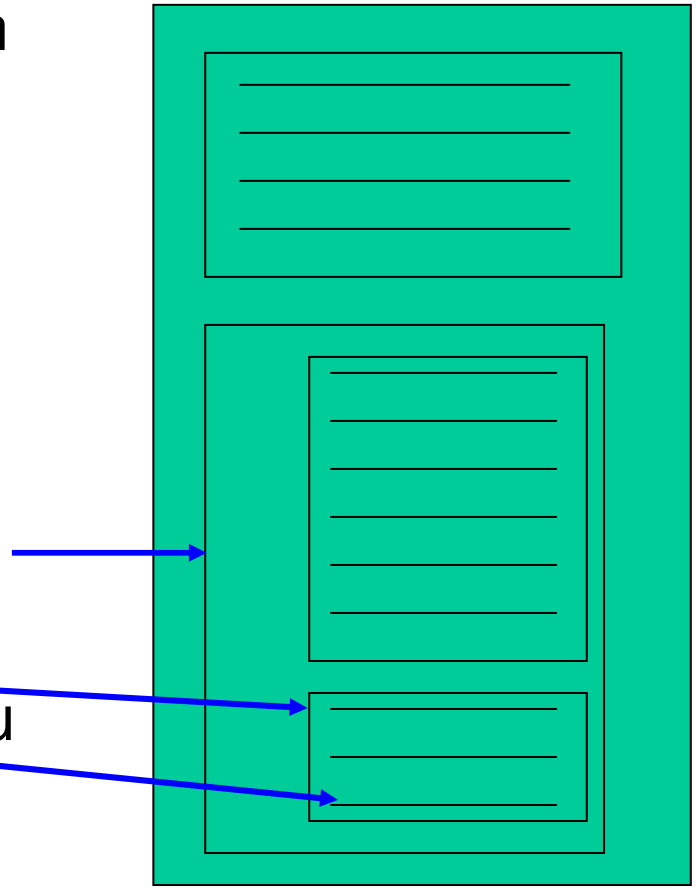
Cấu trúc lặp

```
i = 1;
while (i < 5) do {
    printf (i);
    i = i + 1;
 }
```



Cấu trúc khối & các lệnh lồng nhau

- Cấu trúc khối cho phép ta gộp nhiều lệnh thành 1 thành phần duy nhất. Lệnh miêu tả cấu trúc khối thường được gọi là lệnh kép (compound statement). Mỗi lệnh kép chứa nhiều lệnh trong thân của nó, mỗi lệnh trong thân của 1 lệnh kép có thể là lệnh kép khác,... Kết quả các lệnh của chương trình được tổ chức theo dạng phân cấp, lệnh ngoài cùng (cấp 1) có thể chứa nhiều lệnh cấp 2, mỗi lệnh cấp 2 có thể chứa nhiều lệnh cấp 3,...



Hàm và chương trình con

- Các phần chương trình nhỏ, có tên và có thể được gọi bởi tên ở các phần khác của chương trình.
- Thực hiện một công việc chuyên nhiệm và lập lại nhiều lần trong chương trình (hay cần dùng bởi nhiều chương trình khác nhau).
- Cho phép chương trình được thiết kế thành nhiều thành phần nhỏ.
- Có thể định nghĩa biến cục bộ riêng.
- Hàm (function) trả về kết quả khi được gọi, nếu không trả về kết quả thì ta gọi là thủ tục (subroutine, procedure).

Ví dụ

```
//hàm tìm giá trị lớn nhất trong 2 giá trị
int max(int a, int b) {
        if (a < b)
           return □ ;
        else
           return b;
}

//điểm nhập của chương trình viết bằng ngôn ngữ C
void main() {
          int a;
          a = max(1,2);
}
```

Các thế hệ ngôn ngữ lập trình

- Thế hệ thứ nhất:
  - Xuất hiện vào thập niên 60
  - Tập lệnh gần giống như tập lệnh máy (machine code)
  - Đại diện tiêu biểu: Fortran
- Thế hệ thứ hai
  - Phát triển các cấu trúc dữ liệu từ thế hệ thứ nhất
  - Xuất hiện cấu trúc khối (block structure), các cấu trúc điều khiển (control structures) và các dạng cú pháp linh hoạt hơn
  - Đại diện tiêu biểu: Algol-60

Các thế hệ ngôn ngữ lập trình (tt)

- Thế hệ thứ ba:
  - Xuất hiện các kiểu dữ liệu do người sử dụng định nghĩa (user-defined data types)
  - Các dạng cấu trúc điều khiển tiếp tục được bổ sung hiệu quả hơn.
  - Ngôn ngữ độc lập hơn với kiến trúc máy tính.
  - Đại diện tiêu biểu: Pascal

Các thế hệ ngôn ngữ lập trình (tt)

- Thế hệ thứ tư: (Fourth Generation Languages – 4GL)
  - Dễ sử dụng hơn, đặc biệt dành cho những người không phải là chuyên gia
  - Cho phép đưa ra những giải pháp nhanh để xử lý dữ liệu
  - Xúc tích hơn
  - Gần với ngôn ngữ tự nhiên
  - Gần gũi với người sử dụng
  - Không có dạng thủ tục (non-procedural)
  - Đại diện tiêu biểu: Structured Query Language (SQL)
- Thế hệ thứ năm:
  - Các ngôn ngữ được chuyên dụng hoá, độc lập với kiến trúc máy tính, phục vụ các nhu cầu lập trình đặc trưng.
  - Hỗ trợ nhiều cấu trúc điều khiển và có các dạng cú pháp tương đối dễ đọc.

4.2 Xử lý ngôn ngữ

- Máy tính chỉ có thể hiểu và thực thi được một chương trình khi các lệnh của chương trình được viết một cách tuyệt đối chính xác và rõ ràng về ngữ nghĩa theo ngôn ngữ mà máy đó qui định.
- Để viết được một chương trình như vậy, ngôn ngữ lập trình cũng phải được định nghĩa theo một hình thức rõ ràng và chính xác.
- Ngôn ngữ dùng để định nghĩa ngôn ngữ lập trình là siêu ngôn ngữ (meta-language).

Dịch ngôn ngữ máy tính

- Máy tính vật lý chỉ có thể hiểu và thực thi được chương trình viết bằng ngôn ngữ máy.
- Nhưng con người thường dùng 1 trong các ngôn ngữ lập trình cấp cao để viết chương trình vì dễ thể hiện ý tưởng của mình hơn nhiều.
- Để máy tính thực hiện được một chương trình viết bằng ngôn ngữ lập trình cấp cao, chương trình đó cần phải được dịch sang ngôn ngữ máy.
- Dịch (hoặc xử lý) ngôn ngữ máy tính là chuyển đổi một chương trình viết bằng ngôn ngữ lập trình sang một dạng ngôn ngữ khác (thường là ngôn ngữ máy).
- Có 2 cách thức dịch : biên dịch (compiler) và thông dịch (interpreter).

Trình biên dịch (Compiler)

- Chương trình biên dịch nhận một chương trình nguồn (thường được viết bằng ngôn ngữ cấp cao) và tạo ra một chương trình đối tượng tương ứng về chức năng nhưng thường được viết bằng ngôn ngữ cấp thấp (thường là ngôn ngữ máy).
- Nếu có lỗi xảy ra trong lúc dịch, trình biên dịch sẽ báo lỗi, cố gắng tìm vị trí đúng kế tiếp rồi tiếp tục dịch… Nhờ vậy, mỗi lần dịch 1 chương trình, ta sẽ xác định được nhiều lỗi nhất có thể có.
- Sau mỗi lần dịch, nếu không có lỗi, trình biên dịch sẽ tạo ra file chứa chương trình đối tượng (thí dụ file chương trình khả thi *.exe trên Windows).
- Để chạy chương trình, người dùng chỉ cần kích hoạt file khả thi (người dùng không biết và không cần quan tâm đến file chương trình nguồn).

Trình thông dịch (Interpreter)

- Chương trình thông dịch không tạo ra và lưu giữ chương trình đối tượng.
- Mỗi lần thông dịch 1 chương trình nguồn là 1 lần cố gắng chạy chương trình này theo cách thức sau :
  - dịch và chuyển sang mã thực thi từng lệnh một rồi nhờ máy chạy đoạn lệnh tương ứng.
  - Nếu có lỗi thì báo lỗi, nếu không có lỗi thì thông dịch lệnh kế tiếp... cho đến khi hết chương trình.
  - Như vậy, mỗi lần thông dịch chương trình, trình thông dịch chỉ thông dịch các lệnh trong luồng thi hành cần thiết chứ không thông dịch hết mọi lệnh của chương trình nguồn. Do đó, sau khi thông dịch thành công 1 chương trình, ta không thể kết luận rằng chương trình này không có lỗi.

So sánh trình biên dịch & trình thông dịch

- Mọi hoạt động xử lý trên mọi mã nguồn của chương trình (kiểm tra lỗi, dịch ra các lệnh đối tượng tương đương,...) đều được chương trình biên dịch thực hiện để tạo được chương trình đối tượng thực thi. Do đó sau khi dịch các file mã nguồn của chương trình, nếu không có lỗi, ta có thể kết luận chương trình không thể có lỗi thời điểm dịch (từ vựng, cú pháp). Quá trình biên dịch và quá trình thực thi chương trình là tách rời nhau : biên dịch 1 lần và chạy nhiều lần cho đến khi cần cập nhật version mới của chương trình.
- Chương trình thông dịch sẽ thông dịch từng lệnh theo luồng thi hành của chương trình bắt đầu từ điểm nhập của chương trình, thông dịch 1 lệnh gồm 2 hoạt động : biên dịch lệnh đó và thực thi các lệnh kết quả. Nếu 1 đoạn lệnh cần được thực thi lặp lại thì trình thông dịch sẽ phải thông dịch lại tất cả đoạn lệnh đó. Điều này sẽ làm cho việc chạy chương trình trong chế độ thông dịch không hiệu quả.
- Việc chạy chương trình bằng cơ chế thông dịch đòi hỏi chương trình thông dịch và chương trình ứng dụng cần chạy phải tồn tại đồng thời trong bộ nhớ máy tính, do đó có nguy cơ chạy không được các chương trình lớn nếu tài nguyên của máy không đủ cho cả 2 chương trình thông dịch và chương trình ứng dụng.

Hoạt động liên kết (Link)

- Một chương trình thường bao gồm nhiều module chức năng, mỗi module gồm nhiều file mã nguồn. Các file thường có liên quan (phụ thuộc) với nhau qua thao tác gọi hàm/thủ tục hay truy xuất 1 số dữ liệu của nhau. Các module của chương trình có thể do người lập trình chương trình đó viết hay là module thư viện đã có (của các hãng phần mềm và của người khác).
- Chương trình dịch cho phép dịch từng file mã nguồn rời rạc và độc lập nhau. File mã đối tượng (object file) được tạo ra khi dịch 1 file mã nguồn còn chứa nhiều chỗ chưa hoàn chỉnh, đó là những lệnh gọi hàm/thủ tục của module khác, hay đó là địa chỉ của biến dữ liệu trong module khác...
- Chương trình liên kết (Linker) có nhiệm vụ tổng hợp các file mã đối tượng của chương trình lại thành 1 thể thống nhất và hoàn chỉnh lại các vị trí chưa có thông tin đầy đủ, hầu có thể chạy được.

Hoạt động liên kết (tt)

- Có 2 cơ chế liên kết các file của 1 chương trình là liên kết tĩnh (static link) và liên kết động (dynamic link).
- Liên kết tĩnh là hoạt động liên kết xảy ra tại thời điểm dịch, trước khi chương trình chạy, tất cả các vị trí chứa thông tin chưa hoàn chỉnh đều phải được hiệu chỉnh lại cho hoàn chỉnh.
- Liên kết động là hoạt động liên kết xảy ra tại thởi điểm chạy chương trình, cụ thể tại lần đầu tiên chạy lệnh chứa thông tin chưa hoàn chỉnh (hay mỗi lần chạy lại). Mỗi lần chương trình chạy đến lệnh chứa thông tin chưa hoàn chỉnh, hệ thống sẽ dừng tạm thời chương trình, cố gắng liên kết với module liên quan, hiệu chỉnh lại lệnh hiện hành sao cho có thể chạy được rồi tiếp tục chạy chương trình từ lệnh này.
- Liên kết động có nhiều ưu điểm hơn liên kết tĩnh và hầu hết các hệ thống hiện nay (Windows, Linux) đều sử dụng chủ yếu cơ chế liên kết động.

4.3 Phát triển phần mềm

- Phần mềm phục vụ nhu cầu cho người dùng hiện nay khá phức tạp, khá lớn nên người ta không thể viết ngay ra mã nguồn chương trình ngay sau khi được đặt hàng về bài toán cần giải quyết.
- Từ bài toán cần giải quyết đến khi có được chương trình giải quyết bài toán đó, người ta phải thực hiện nhiều công việc khác nhau, tốn nhiều thời gian, công sức,...
- Người ta dùng thuật ngữ "qui trình phát triển phần mềm" (Software Development Process) để miêu tả cụ thể, chi tiết trình tự các công việc cần phải thực hiện để xây dựng được chương trình từ bài toán cần giải quyết.
- Người ta đã đưa ra và dùng nhiều qui trình phát triển khác nhau để xây dựng phần mềm, trong đó qui trình phát triển phần mềm hợp nhất (Unified Software Development Process) hiện được dùng phổ biến nhất.

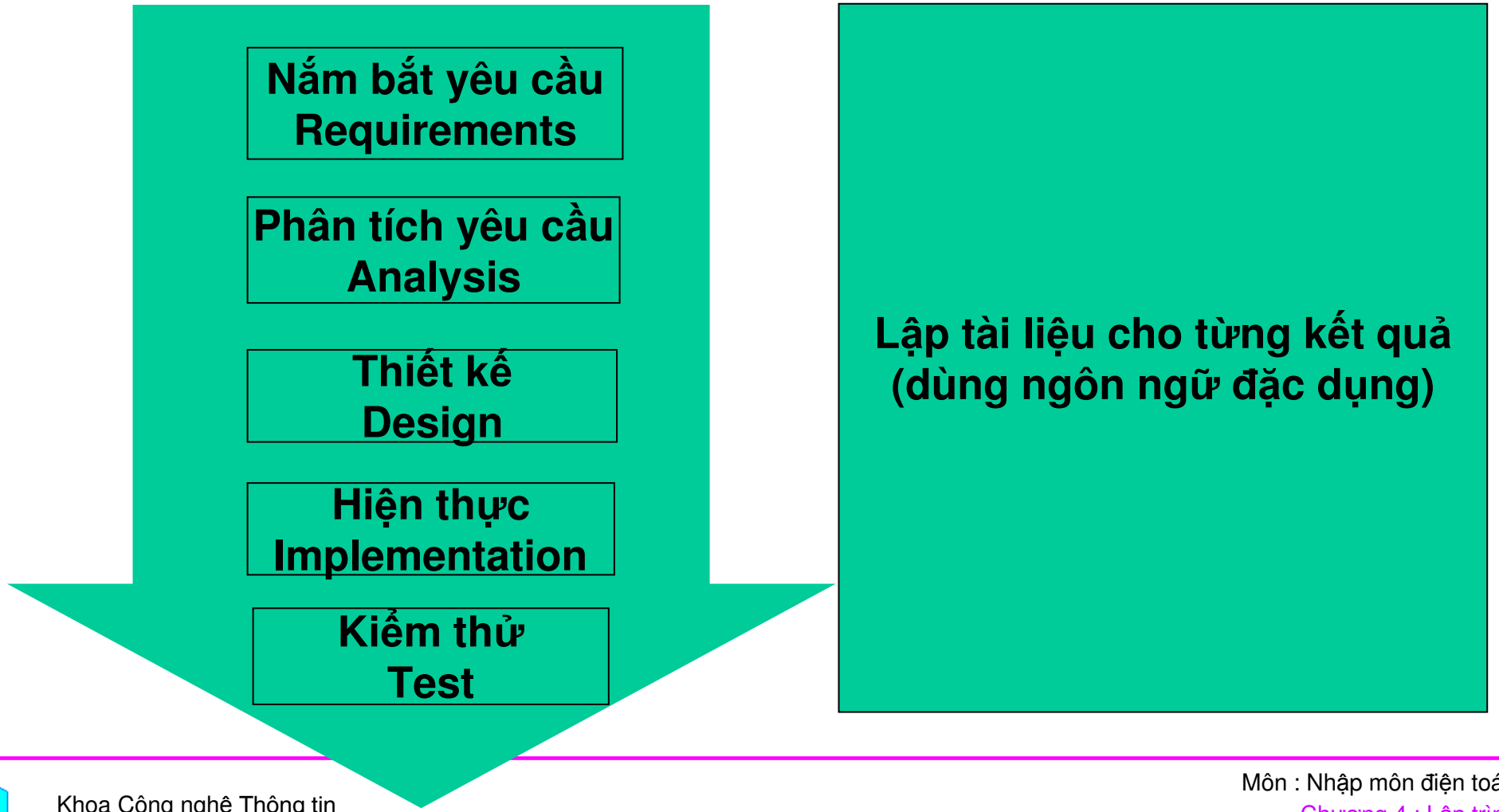
Phát triển phần mềm (tt)

Để xây dựng 1 chương trình, qui trình phát triển phần mềm hợp nhất (Unified Software Development Process) sẽ xác định rõ ràng các thông tin sau :

- bao nhiêu loại người (role) sẽ tham gia thực hiện, thí dụ như kiến trúc sư phần mềm, phân tích viên, kỹ sư thiết kế, lập trình viên, kiểm lỗi viên,...
- mỗi loại người sẽ phải thực hiện các công việc gì cụ thể, thí dụ lập trình viên A phải viết code cho bao nhiêu hàm, các hàm đó cụ thể là gì ?
- mỗi công việc sẽ được thực hiện khi nào ?
- mỗi công việc sẽ được thực hiện bằng cách nào ?
- kết quả mỗi công việc sẽ được miêu tả theo định dạng nào, bằng ngôn ngữ miêu tả nào ?

Workflows (Luồng công việc)

Thường để phát triển 1 chương trình, ta cần thực hiện các luồng công việc chức năng sau đây :



Nắm bắt yêu cầu (Requirements)

- Nhiệm vụ của workflow này là xác định chính xác, rõ ràng và đầy đủ các thông tin sau liên quan đến chương trình :
  - các chức năng của chương trình cần đáp ứng
  - chương trình sẽ tương tác với các thành phần nào : loại người nào, phần mềm nào, thiết bị nào,...
- Thí dụ xây dựng chương trình "hoa hóa" các từ trong 1 file dữ liệu. Sau khi nắm bắt yêu cầu, ta có được kết quả sau :
  - ai cũng có thể dùng chương trình với chức năng giống nhau : chương trình chỉ cung cấp chức năng đổi thành chữ hoa ký tự đầu từ của tất cả các từ trong 1 file do người dùng xác định.
  - chương trình chỉ cần tương tác với hệ thống để nhờ thực hiện 1 số chức năng truy xuất file.

Phương pháp phân tích từ-trên-xuống

Trong quá khứ, phương pháp thường sử dụng để phân tích bài toán là phương pháp từ-trên-xuống (top-down analysis). Phương pháp này cũng được dùng cho workflow thiết kế, hiện thực,...

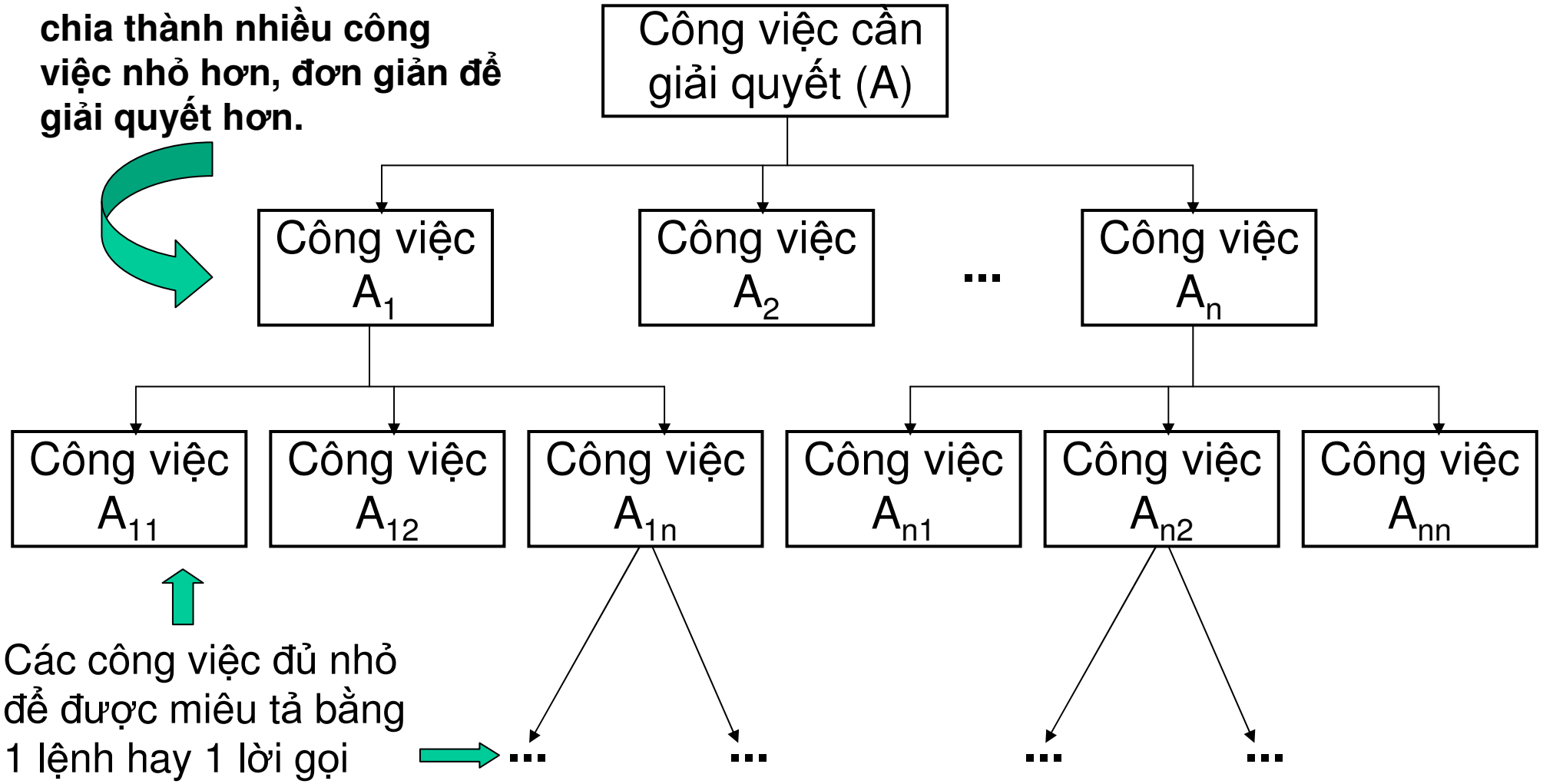
Nội dung của phương pháp này là xét xem, muốn giải quyết vấn đề nào đó thì cần phải làm những công việc nhỏ hơn nào. Mỗi công việc nhỏ hơn tìm được lại được phân thành những công việc nhỏ hơn nữa, cứ như vậy cho đến khi những công việc phải làm là những công việc thật đơn giản, có thể thực hiện dễ dàng.

Thí dụ việc học lấy bằng kỹ sư CNTT khoa CNTT ĐHBK TP.HCM có thể bao gồm 9 công việc nhỏ hơn là học từng học kỳ từ 1 tới 9, học học kỳ i là học n môn học của học kỳ đó, học 1 môn học là học m chương của môn đó,...

Hình vẽ của slide kế cho thấy trực quan của việc phân tích top-down.

Phương pháp phân tích từ-trên-xuống (tt)

chia thành nhiều công việc nhỏ hơn, đơn giản để giải quyết hơn.



Phân tích yêu cầu (Analysis)

- Nhiệm vụ của workflow này là phát họa sơ lược cách giải quyết từng chức năng của chương trình :
  - lặp phân tích từng chức năng theo 1 thứ tự nào đó.
- Thí dụ chương trình "hoa hóa" các từ trong 1 file dữ liệu, chương trình chỉ có 1 chức năng. Sau khi phân tích chức năng này, ta phát hoạ được sơ lược cách giải quyết nó như sau :
  - tương tác với người dùng để họ xác định được file cần xử lý.
  - đọc nội dung file vào bộ nhớ.
  - duyệt nội dung file trong bộ nhớ, xác định từng từ rồi "hoa hóa" ký tự đầu từ.
  - ghi nội dung xử lý được lên file.

Thiết kế (Design)

- Nhiệm vụ của workflow này là chi tiết hóa cách giải quyết từng chức năng của chương trình đến mức độ dễ dàng viết code nhất có thể có.
  - lặp thiết kế từng chức năng theo 1 thứ tự xác định.
- Thí dụ chương trình "hoa hóa" các từ trong 1 file dữ liệu, chương trình chỉ có 1 chức năng. Sau khi thiết kế chức năng này, ta miêu tả được cách giải quyết nó như sau :
  - dùng đối tượng CFileDialog để tương tác với user để user duyệt hệ thống file trên đĩa và xác định file cần xử lý.
  - dùng đối tượng CFile để quản lý việc truy xuất nội dung file.
  - duyệt nội dung file trong bộ nhớ, xác định từng từ rồi "hoa hóa" ký tự đầu từ (theo giải thuật chi tiết ở silde kế tiếp).
  - ghi nội dung xử lý được lên file.

Thiết kế (Design)

- Thiết kế thuật giải "hoa hóa" 1 chuỗi văn bản :

Lặp "hoa hóa" từng từ cho đến khi hết chuỗi :

- lặp tìm từng ký tự dấu ngăn đi trước từ sắp thấy
- chuyển ký tự chữ đầu từ thành chữ hoa
- lặp tìm từng ký tự chữ của từ hiện hành

Kết thục lặp

Hiện thực (Implementation)

- Nhiệm vụ của workflow này là dịch bản thiết kế chi tiết thành mã nguồn của ngôn ngữ lập trình xác định, từ đó dịch ra mã máy để tạo thành chương trình khả thi có thể chạy trên máy tính.
  - lặp hiện thực từng chức năng thiết kế (hay từng phần nhỏ của chức năng) theo 1 thứ tự xác định.
- Thí dụ chương trình "hoa hóa" các từ trong 1 file dữ liệu, chương trình chỉ có 1 chức năng. Sau khi hiện thực chức năng này từ bản thiết kế trong slide trước bằng ngôn ngữ VC++, ta có được đoạn chương trình như sau :

Hiện thực (Implementation)

```
//dùng đối tượng CFileDialog để tương tác với user
CFileDialog dlg(TRUE);
if (dlg.DoModal() != IDOK) return;
//dùng đối tượng CFile để quản lý việc truy xuất nội dung file
CFile file;
file.Open(dlg.GetFileName(),CFile::modeRead | CFile::shareExclusive);
int flen= file.GetLength();
unsigned char* fbuf = (unsigned char*) malloc(flen+1);
file.Read (fbuf,flen);
file.Close();
//duyệt xử lý nội dung file trong bộ nhớ.
int i = 0;
unsigned char ch;
```

Hiện thực (Implementation)

```
while (i < flen) { //Lặp "hoa hóa" từng từ cho đến khi hết chuỗi
    //lặp tìm từng ký tự dấu ngăn đi trước từ sắp thấy
    while (i <flen) {
        ch = fbuf[i];
        if (('A' <= ch && ch <= 'Z') || ('a' <= ch && ch <= 'z')) break;
        i++;
    }
    if (i == flen) break;
    //chuyển ký tự chữ đầu từ thành chữ hoa
    if ('a' <= ch && ch <= 'z') fbuf[i++] -= 32;
    //lặp tìm từng ký tự chữ của từ hiện hành
    while (i <flen) {
        ch = fbuf[i];
        if (!(('A' <= ch && ch <= 'Z') || ('a' <= ch && ch <= 'z'))) break;
        i++;
    }
} //Kết thục lặp
```

Hiện thực (Implementation)

```
//ghi nội dung xử lý được lên file
file.Open(dlg.GetFileName()+".Hoa",
    CFile::modeCreate | CFile::modeWrite |CFile::shareExclusive);
file.Write(fbuf,flen);
file.Close();
```

Kiểm thử (Testing)

- Nhiệm vụ của workflow này kiểm tra và thử nghiệm chương trình thực thi xem nó có lỗi không, nếu có thì lỗi cụ thể nằm ở lệnh nào, tại sao bị lỗi và sữa lỗi.
  - lặp kiểm thử từng hàm chức năng theo 1 thứ tự xác định.
- Có 2 loại kiểm thử trên từng thành phần chương trình :
  - kiểm thử hộp đen (black-box testing) : kiểm thử thành phần theo góc nhìn từ ngoài xem hành vi của thành phần có thỏa mãn đặc tả sử dụng không ? Thí dụ ta thử gọi hàm cos(0) xem hàm có trả về 1 không, nếu hàm trả về giá trị khác 1, ta nói hàm cos bị lỗi.
  - kiểm thử hộp trắng (white-box testing) : kiểm thử thành phần theo góc nhìn bên trong xem từng lệnh của thành phần có chạy đúng theo giải thuật thiết kế không ? Thường khi kiểm tra hộp đen 1 thành phần nào đó bị lỗi thì ta mới tiến hành kiểm thử hộp trắng để xác định chính xác các lệnh gây lỗi trong thành phần đó.

Tài liệu hóa qui trình phát triển phần mềm

- Trong qui trình phát triển phần mềm, ta đã thực hiện nhiều workflow, thực hiện mỗi workflow sẽ tạo ra nhiều kết quả, ta phải quản lý, bảo trì các kết quả này theo thời gian nhằm phục vụ cho việc nghiên cứu, hiệu chỉnh, nâng cấp phầm mềm sau này. Một trong các việc quản lý, bảo trì các kết quả tạo được là lập tài liệu. Ta phải dùng 1 ngôn ngữ thích hợp để lập tài liệu cho các kết quả sao cho việc quản lý, bảo trì, chuyển giao phần mềm được dễ dàng, tin cậy và hiệu quả...
- Hiện nay, ngôn ngữ mô hình UML (Unified Modeling Language) được sử dụng rất phổ biến để đặc tả, quản lý các tài liệu trong quá trình phát triển phần mềm.

Chương 5

CƠ SỞ DỮ LIỆU

5.1 Dữ liệu & Hệ thống file
5.2 Các khái niệm cơ bản về database
5.3 Hệ quản trị CSDL
5.4 Các ý niệm cơ bản về cơ sở dữ liệu quan hệ
5.5 Ngôn ngữ SQL
5.6 Cơ sở dữ liệu phân tán

5.1 Dữ liệu & Hệ thống file

- Trong chương 3, chúng ta đã giới thiệu module "Hệ thống file" của HĐH và các dịch vụ truy xuất file/thư mục của nó. Ở mức độ HĐH, mỗi file có cấu trúc đơn giản : chuỗi byte với độ dài theo nhu cầu.
- Mỗi phần mềm dùng dịch vụ Hệ thống file của HĐH để tạo file, thêm/bớt, hiệu chỉnh dữ liệu theo cách riêng của mình. Tùy theo nhu cầu xử lý dữ liệu, mỗi ứng dụng tự đặt ra 1 định dạng riêng để chứa dữ liệu lên file. Thí dụ định dạng *.txt, *.doc, *.xls, *.mp3, *.wav, *.mpg, *.exe...
- Hầu hết các ứng dụng hiện nay, nhất là các ứng dụng nghiệp vụ (ứng dụng giải quyết 1 bài toán nghiệp vụ ngoài đời như quản lý cán bộ, quản lý thư viện, quản lý vật tư,...), đều phải truy xuất rất nhiều dữ liệu có cùng định dạng, cấu trúc (mặc dù nội dung cụ thể thì khác nhau). Thí dụ file chứa các hồ sơ sinh viên, file chứa các hồ sơ nhà, file chứa các hồ sơ đường xá...

Dữ liệu & Hệ thống file (tt)

- Mỗi dữ liệu trong file dữ liệu có định dạng đồng nhất được gọi là record. Việc quản lý các file chứa nhiều record bao gồm nhiều tác vụ như tạo file mới với cấu trúc record cụ thể, thêm/bớt/hiệu chỉnh/duyệt các record, tìm kiếm các record thỏa mãn 1 tiêu chuẩn nào đó,... Để thực hiện các tác vụ trên (nhất là tìm kiếm record) hiệu quả, tin cậy, ta cần nhiều kiến thức khác nhau và phải tốn nhiều công sức.
- Để giải phóng các ứng dụng khỏi việc quản lý phiền hà, nặng nhọc trên, người ta đã xây dựng ứng dụng đặc biệt : Hệ quản trị cơ sở dữ liệu - DBMS (Database Management System). DBMS sẽ thực hiện mọi tác vụ quản lý các file dữ liệu có định dạng đồng nhất (mà ta gọi là database), còn ứng dụng sẽ nhờ DBMS để truy xuất database dễ dàng, tin cậy, hiệu quả...

5.2 Các khái niệm cơ bản về database

- Cơ sở dữ liệu (database) là sự tập hợp có tổ chức các dữ liệu có liên quan luận lý với nhau.
- Dữ liệu (data) là sự biểu diễn của các đối tượng và sự kiện được ghi nhận và được lưu trữ trên các phương tiện của máy tính.
  - Dữ liệu có cấu trúc : số, ngày, chuỗi ký tự, …
  - Dữ liệu không có cấu trúc: hình ảnh, âm thanh, đoạn phim, …
- Có tổ chức (organized) : người sử dụng có thể dễ dàng lưu trữ, thao tác và truy xuất dữ liệu.
- Có liên quan luận lý (logically related) : dữ liệu mô tả một lĩnh vực mà nhóm người sử dụng quan tâm và được dùng để trả lời các câu hỏi liên quan đến lĩnh vực này.

Các khái niệm cơ bản về database

- Thông tin (information) là dữ liệu đã được xử lý để làm tăng sự hiểu biết của người sử dụng.
  - Dữ liệu trong ngữ cảnh.
  - Dữ liệu được tổng hợp/xử lý.

Các khái niệm cơ bản về database

Dữ liệu

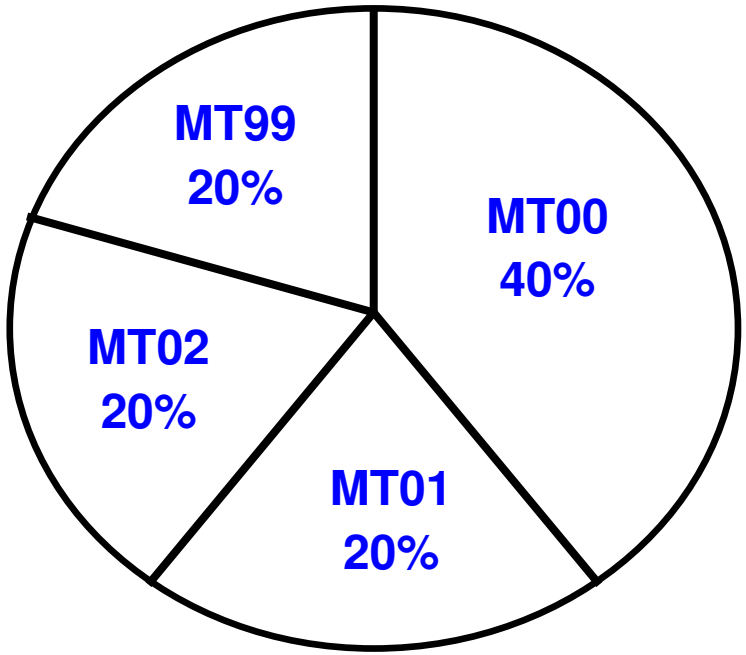
50010273	Nguyễn Trung Tiến	MT00	20
50100298	Lê Việt Hùng	MT01	19
59900012	Trần Hùng Việt	MT99	21
50200542	Hồ Xuân Hương	MT02	18
50000075	Bùi Đức Duy	MT00	20

Thông tin: dữ liệu trong ngữ cảnh

Mã sinh viên	Họ và tên sinh viên	Lớp	Tuổi
50010273	Nguyễn Trung Tiến	MT00	20
50100298	Lê Việt Hùng	MT01	19
59900012	Trần Hùng Việt	MT99	21
50200542	Hồ Xuân Hương	MT02	18
50000075	Bùi Đức Duy	MT00	20

Các khái niệm cơ bản về database

Thông tin : dữ liệu được tổng hợp / xử lý



Các khái niệm cơ bản về database

- Siêu dữ liệu (*metadata)* là dữ liệu dùng để mô tả các tính chất/đặc tính của dữ liệu khác (dữ liệu về dữ liệu).
  - Các đặc tính : định nghĩa dữ liệu, cấu trúc dữ liệu, qui tắc/ràng buộc.

Siêu dữ liệu cho Sinh_viên

Data Item						
Name	**Type**	**Length**		**Min**	**Max**	**Description**
MaSV	Character	8				Mã sinh viên
Hoten	Character	30				Họ tên sinh viên
Lop	Character	3				Lớp
Tuoi	Number	2	17	25		Tuổi

Ưu điểm của cách tiếp cận CSDL

- Độc lập dữ liệu – chương trình (*data - program independence*).
  - DBMS chứa siêu dữ liệu (*metadata*), do đó các ứng dụng không cần quan tâm đến các dạng thức của dữ liệu.
  - DBMS quản lý các truy vấn và cập nhật dữ liệu, do đó ứng dụng không cần xử lý việc truy xuất dữ liệu.
- Giảm tối thiểu sự dư thừa dữ liệu (*data redundancy*).
- Nâng cao tính nhất quán (*data consistency*) / toàn vẹn dữ liệu (*data integrity*).
- Nâng cao việc dùng chung dữ liệu (*data sharing*).
  - Những người sử dụng khác nhau có những cái nhìn khác nhau về dữ liệu.
- Tăng hiệu suất phát triển ứng dụng.

Ưu điểm của cách tiếp cận CSDL

- Tuân thủ các tiêu chuẩn.
  - Tất cả các truy xuất dữ liệu đều được thực hiện theo cùng một cách.
- Nâng cao chất lượng của dữ liệu.
  - Các ràng buộc (*constraint*), các qui tắc hợp lệ của dữ liệu (*data validation rule*).
- Nâng cao tính truy xuất và tính đáp ứng của dữ liệu.
  - Sử dụng ngôn ngữ truy vấn dữ liệu chuẩn (SQL - *Structured Query Language*).
- Giảm chi phí bảo trì chương trình.
- Bảo mật (*security*).
- Chép lưu (*backup*) và phục hồi (*recovery*).
- Điều khiển tương tranh (*concurrency control*).

Chi phí và rủi ro của cách tiếp cận CSDL

- Chi phí b□ n đầu
  - Chi phí cài đặt và quản lý
  - Chi phí chuyển đổi (*conversion cost*)
- Chi phí vận hành
  - Cần nhân viên mới có chuyên môn.
  - Cần phải chép lưu và phục hồi.
- Mâu thuẫn về mặt tổ chức
  - Rất khó th□ y đổi các thói quen cũ.

Các loại cơ sở dữ liệu

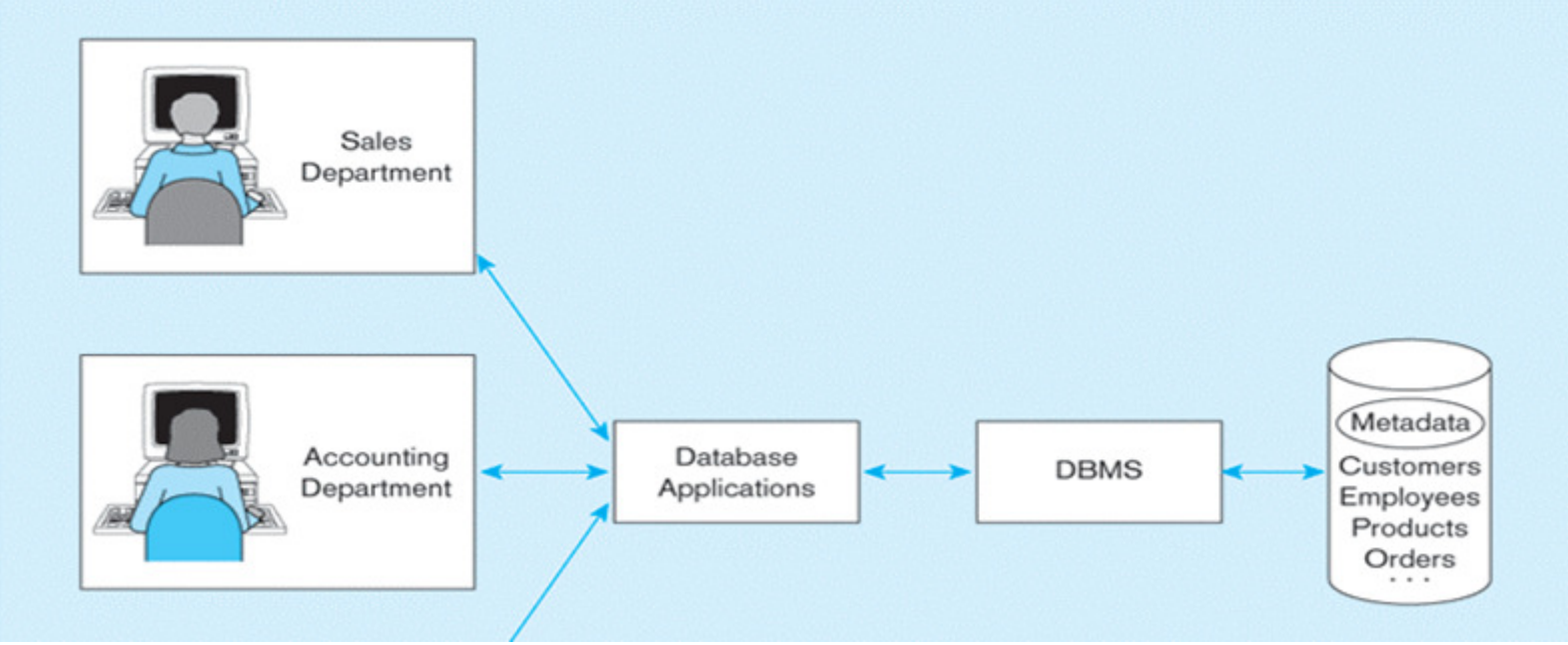
- CSDL cá nhân
  - *personal database*
  - CSDL riêng.
- CSDL nhóm làm việc
  - *workgroup database*
  - Mạng cục bộ (ít hơn 25 người sử dụng)
- CSDL phòng ban
  - *department database*
  - Mạng cục bộ (từ 25 đến 100 người sử dụng)
- CSDL xí nghiệp
  - *enterprise database*
  - Mạng diện rộng (hàng trăm hoặc hàng ngàn người sử dụng)

Các loại cơ sở dữ liệu

Type of Database	*Typical Number of Users*	*Typical Architecture*	*Typical Size of Database*
Personal	1	Desktop/laptop computer, PDA	Megabytes
Workgroup	5–25	Client/server (two-tier)	Megabytes–gigabytes
Department	25–100	Client/server (three-tier)	Gigabytes
Enterprise	>100	Client/server (distributed or parallel server)	Gigabytes–terabytes
Internet	>1000	Web server and application servers	Megabytes–gigabytes

5.3 Hệ quản trị cơ sở dữ liệu

- Hệ quản trị CSDL (DBMS – DataBase Management System) là tập hợp các chương trình dùng để quản lý cấu trúc và dữ liệu của CSDL và điều khiển truy xuất dữ liệu trong CSDL.
  - Cho phép người sử dụng định nghĩa, tạo lập, bảo trì CSDL và cung cấp các truy xuất dữ liệu.



Các chức năng của hệ quản trị CSDL

- Lưu trữ, truy xuất và cập nhật dữ liệu
  - Ngôn ngữ định nghĩa dữ liệu (DDL - Data Definition Language)
  - Ngôn ngữ thao tác dữ liệu (DML - Data Manipulation Language).
- Quản lý giao tác (transaction management).
- Điều khiển tương tranh (concurrency control)
- Chép lưu và phục hồi dữ liệu.
- Bảo mật dữ liệu
  - Ngôn ngữ điều khiển dữ liệu (DCL - Data Control Language).
- Hỗ trợ truyền thông dữ liệu.
- Duy trì tính toàn vẹn / nhất quán dữ liệu.
- Cung cấp các tiện ích.

5.4 Các ý niệm cơ bản về cơ sở dữ liệu quan hệ

- Quan hệ (relation) là một bảng dữ liệu hai chiều bao gồm nhiều hàng (mẩu tin) và nhiều cột (thuộc tính hoặc vùng tin).
  - Mỗi hàng là duy nhất: không thể có hai hàng có cùng các giá trị ở tất cả vùng tin.
  - Thứ tự của các hàng là không quan trọng.
  - Thứ tự của các cột là không quan trọng.
  - Không phải mọi bảng đều là quan hệ. Quan hệ là một bảng không chứa các hàng giống hệt nhau.

Quan hệ

Quan hệ: Supplier

Snum	Name	City
S1	Nguyễn Trung Tiến	SF
S2	Trần Thị Yến	LA
S3	Nguyễn Văn An	SF

Khóa (Key)

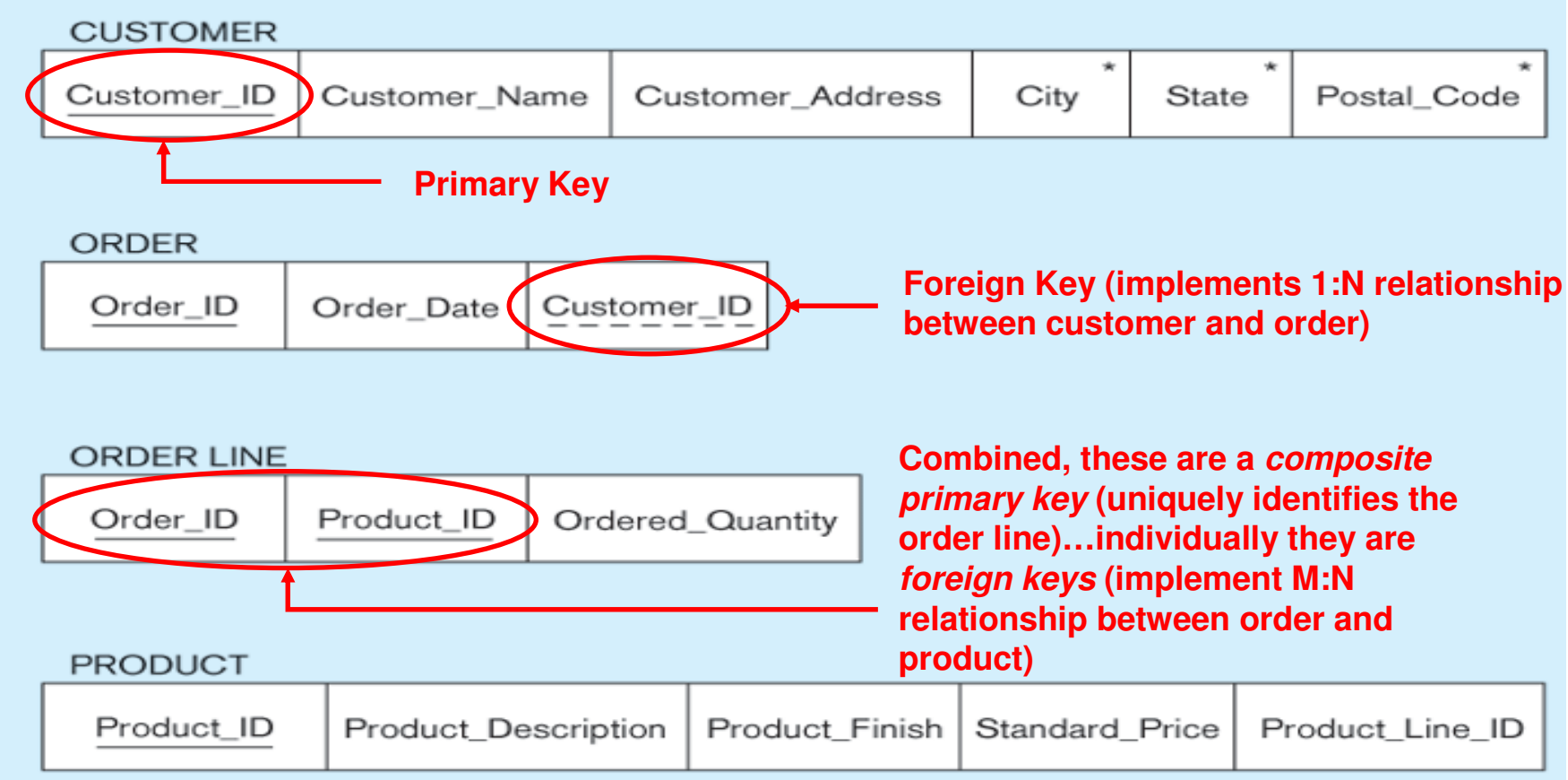
- Khóa quan hệ là một tập nhỏ nhất các thuộc tính dùng để xác định duy nhất một hàng.
- Một khóa chỉ có một thuộc tính được gọi là khóa đơn (simple key).
- Một khóa có nhiều thuộc tính được gọi là khóa phức hợp (composite key).
- Khóa thường được sử dụng làm chỉ mục (index) của bảng dữ liệu để làm tăng tốc độ xử lý câu truy vấn.
- Một quan hệ phải có ít nhất một khóa và có thể có nhiều khóa.
- Các thuộc tính thuộc một khóa được gọi là thuộc tính khóa (prime attribute), các thuộc tính còn lại trong lược đồ quan hệ được gọi là các thuộc tính không khóa (nonprime attribute).

Khóa

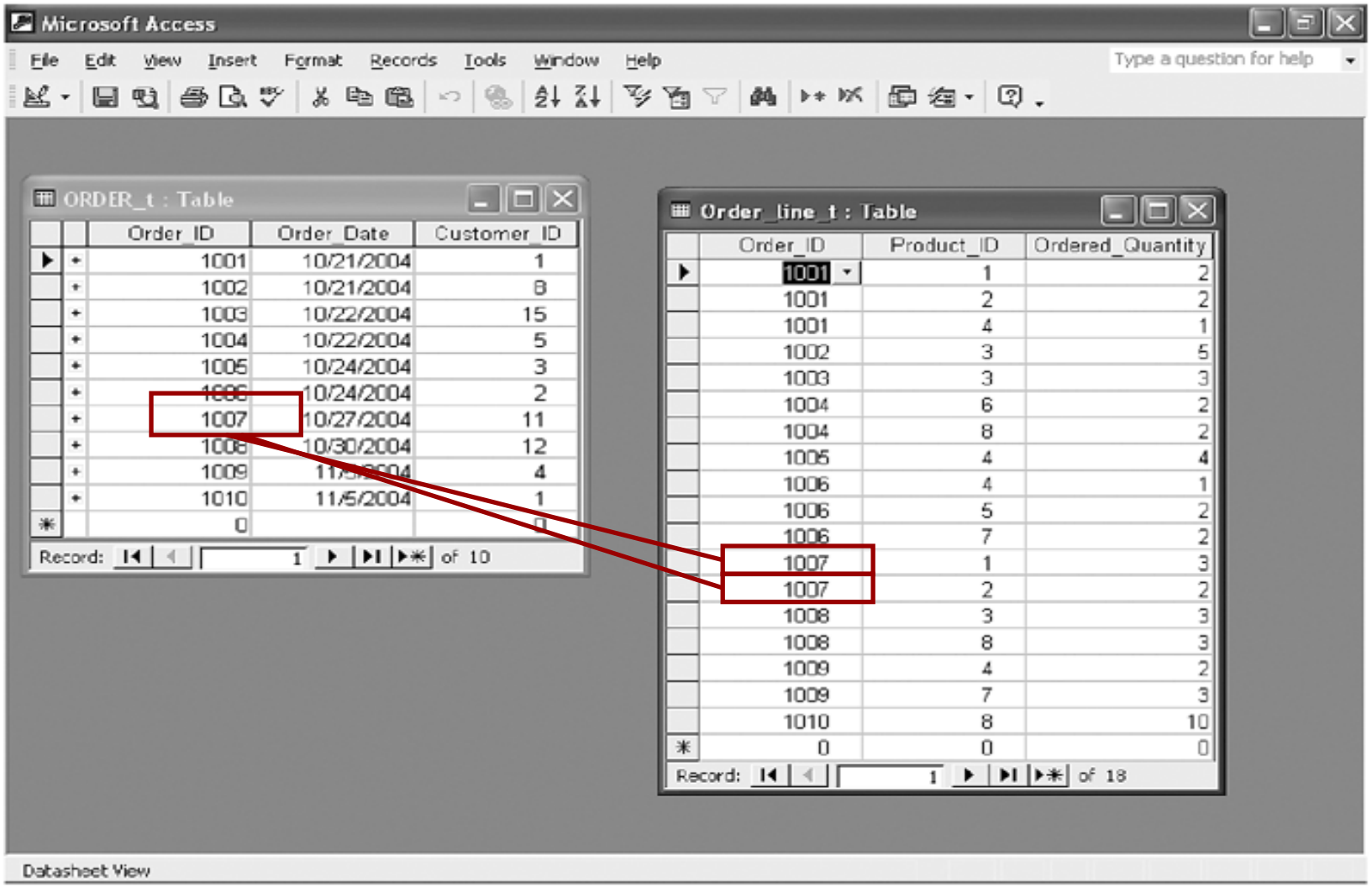
- Các thuộc tính khóa được gạch dưới.
- Các thuộc tính khóa không được có giá trị rỗng (*null value*).
- Tất cả các khóa của một quan hệ được gọi là *khóa dự tuyển* (*candidate key*).
- Một trong các khóa dự tuyển được chọn làm khóa tiêu biểu, khóa này được gọi là *khóa chính* (*primary key*).
- Một quan hệ chỉ có một khóa chính và có thể có nhiều khóa dự tuyển.
- Trong một quan hệ, một hoặc nhiều thuộc tính được gọi là *khóa ngoại* (*foreign key*) nếu chúng là khóa chính của một quan hệ khác.

Cơ sở dữ liệu quan hệ

- Cơ sở dữ liệu quan hệ (*relational database*) bao gồm các bảng (quan hệ) biểu diễn các thực thể và các khóa chính / khóa ngoại biểu diễn các mối liên kết.



Cơ sở dữ liệu quan hệ



Lược đồ cơ sở dữ liệu

- Lược đồ cơ sở dữ liệu (database schema) là một tập hợp các lược đồ quan hệ.
  - Trong một lược đồ cơ sở dữ liệu, các tên lược đồ quan hệ là duy nhất.

Lược đồ cơ sở dữ liệu:

Emp (***Empnum***, ***Name, Sal, Tax, Mgrnum, Deptnum***)

Dept (***Deptnum***, ***Name, Area, Mgrnum***)

Supplier (***Snum***, ***Name, City***)

Supply (***Snum, Pnum***, ***Deptnum, Quan***)

Ràng buộc toàn vẹn

- Ràng buộc toàn vẹn
  - *integrity constraint*
  - *Ràng buộc toàn vẹn* là một qui tắc mà tất cả các dữ liệu trong CSDL phải thỏa mãn qui tắc này.
- Ràng buộc miền trị
  - *domain constraint*
  - Các giá trị cho phép của một thuộc tính.
- Toàn vẹn thực thể
  - *entity integrity*
  - Thuộc tính khóa chính không có giá trị rỗng (*null value*).

Ràng buộc toàn vẹn

- Qui tắc hoạt động
  - *action assertion*
  - Các qui tắc nghiệp vụ (*business rule*).

Ràng buộc toàn vẹn

Attribute	*Domain Name*	*Description*	*Domain*
Customer_ID	Customer_IDs	Set of all possible customer IDs	character: size 5
Customer_Name	Customer_Names	Set of all possible customer names	character: size 25
Customer_Address	Customer_Addresses	Set of all possible customer addresses	character: size 30
City	Cities	Set of all possible cities	character: size 20
State	States	Set of all possible states	character: size 2
Postal_Code	Postal_Codes	Set of all possible postal zip codes	character: size 10
Order_ID	Order_IDs	Set of all possible order IDs	character: size 5
Order_Date	Order_Dates	Set of all possible order dates	date format mm/dd/yy
Product_ID	Product_IDs	Set of all possible product IDs	character: size 5
Product_Description	Product_Descriptions	Set of all possible product descriptions	character size 25
Product_Finish	Product_Finishes	Set of all possible product finishes	character: size 15
Standard_Price	Unit_Prices	Set of all possible unit prices	monetary: 6 digits
Product_Line_ID	Product_Line_IDs	Set of all possible product line IDs	integer: 3 digits
Ordered_Quantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Định nghĩa miền trị cho các thuộc tính

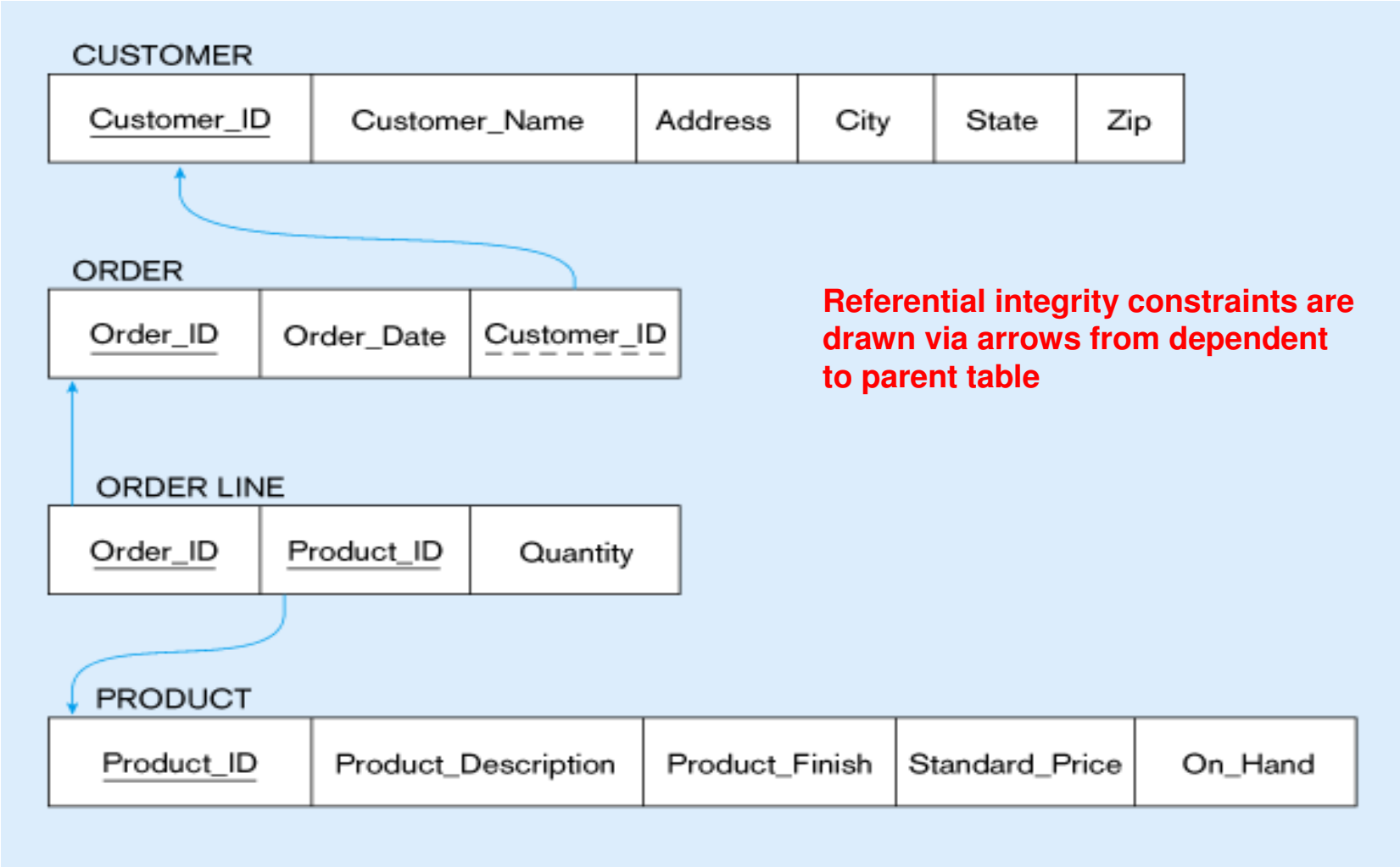
Ràng buộc toàn vẹn

- Ràng buộc toàn vẹn tham chiếu
  - *referential integrity constraint*
  - *Ràng buộc toàn vẹn tham chiếu* là một qui tắc mà tất cả các giá trị của khóa ngoại (nếu khác *null*) trong quan hệ bên phía *nhiều* phải có trong các giá trị của khóa chính trong quan hệ bên phía *một*.

Ràng buộc toàn vẹn

- Ràng buộc toàn vẹn tham chiếu
  - Qui tắc xóa các hàng dữ liệu
    - Hạn chế (*restrict*): không cho phép xóa các hàng bên phía cha (*parent*) nếu tồn tại các hàng liên quan bên phía phụ thuộc (*dependent*).
    - Tầng (*cascade*): tự động xóa các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha.
    - Gán null (*set-to-null*): gán *null* cho khóa ngoại của các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha. Không áp dụng cho các thực thể yếu.

Ràng buộc toàn vẹn



Ví dụ về ràng buộc toàn vẹn tham chiếu

5.5 Ngôn ngữ SQL

- Ngôn ngữ truy vấn có cấu trúc (SQL -Structured Query Language) là ngôn ngữ chuẩn được dùng để tạo lập và truy vấn các cơ sở dữ liệu quan hệ, nó được hỗ trợ bởi hầu hết các hệ quản trị CSDL quan hệ (RDBMS - *Relational* DBMS).
- Ngôn ngữ SQL là một ngôn ngữ tựa tiếng Anh (English-like language), sử dụng các từ như select, insert, delete trong tập lệnh.
- Ngôn ngữ SQL là một ngôn ngữ phi thủ tục (nonprocedural language).
  - Chỉ ra các thông tin gì cần thiết (what).
  - Không cần phải chỉ ra cách thực hiện như thế nào (how) để có được các thông tin này.
- SQL xử lý các tập hợp mẩu tin (bảng) hơn là mỗi lần một mẩu tin đơn lẻ.

Ngôn ngữ SQL

- Nhiều loại người có thể sử dụng SQL: người quản trị CSDL (DBA), người lập trình ứng dụng, người quản lý, người sử dụng cuối cùng (end user).
  - SQL cung cấp nhiều lệnh cho nhiều công việc khác nhau:
  - Truy vấn dữ liệu.
  - Thêm vào, cập nhật và xóa bỏ các hàng của bảng.
  - Tạo lập, thay đổi và xóa bỏ các đối tượng CSDL.
  - Điều khiển truy xuất cơ sở dữ liệu và các đối tượng CSDL.
- Bảo đảm tính nhất quán của CSDL.

Ngôn ngữ SQL

- Ngôn ngữ định nghĩa dữ liệu
  - DDL - Data Definition Language
  - Các lệnh dùng để định nghĩa CSDL: tạo lập (create), thay đổi (alter) và hủy bỏ (drop) các đối tượng dữ liệu, thiết lập các ràng buộc.
- Ngôn ngữ thao tác dữ liệu
  - DML - Data Manipulation Language
  - Các lệnh dùng để bảo trì và truy vấn CSDL: thêm (insert), sửa (update), xóa (delete) dữ liệu của bảng, truy vấn (select).
- Ngôn ngữ điều khiển dữ liệu
  - DCL - Data Control Language
  - Các lệnh dùng để điều khiển CSDL: quản trị các quyền (grant, revoke).

Một số lệnh SQL thường dùng : CREATE

- Lệnh Create dùng để một bảng dữ liệu mới với cấu trúc và những tính chất xác định.

Thí dụ :

Lệnh Create dưới đây tạo 1 bảng mới có tên là PRODUCT_T, mỗi record của bảng có 5 field Product_ID, Product_Description, Product_Finish, Standard_Price, Product_Line_ID.

```
CREATE TABLE PRODUCT_T
          (PRODUCT_ID              INTEGER        NOT NULL,
           PRODUCT_DESCRIPTION     VARCHAR2(50),
           PRODUCT_FINISH          VARCHAR2(20)
                    CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
           STANDARD_PRICE          DECIMAL(6,2),
           PRODUCT_LINE_ID         INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

Định nghĩa các cột và kiểu dữ liệu của các cột.

Một số lệnh SQL thường dùng : SELECT

- Lệnh Select dùng để truy vấn dữ liệu của một bảng hoặc nhiều bảng.

  SELECT [DISTINCT] <*list of expressions*> [INTO <*list of variables*>]

  FROM <*list of tables*> [WHERE <*row conditions*>]

  [GROUP BY <*list of expressions*> [HAVING <*group conditions*>]]

  [ORDER BY <*list of expressions*>];

trong đó :

SELECT: liệt kê các cột (các biểu thức) của kết quả.

FROM: các bảng hoặc các khung nhìn chứa dữ liệu cần thiết cho truy vấn.

WHERE: điều kiện xử lý các hàng để tạo ra kết quả.

GROUP BY: gom nhóm các hàng.

HAVING: điều kiện xử lý các nhóm.

ORDER BY: sắp thứ tự kết quả.

Một số lệnh SQL thường dùng : INSERT

- Thí dụ :

  Select * from Sinhvien where namsinh = 1988

  sẽ lọc tất cả sinh viên sinh nam 1988 trong bảng Sinhvien.

- Lệnh Insert dùng để thêm 1 hay nhiều record dữ liệu vào một bảng.

  INSERT INTO <*table name*> [(<*list of columns*>)]

  VALUES (<*list of expressions*>);

- Thí dụ :

  Insert into Sinhvien Values ('Nguyen Van A', 1987, '245 Ly Thai To, P15, Q10, Tp.HCM');

  sẽ thêm 1 sinh viên mới (với các field dữ liệu được đặc tả trong danh sách) vào bảng Sinhvien.

Một số lệnh SQL thường dùng : DELETE

- Lệnh Delete dùng để xóa 1 hay nhiều record dữ liệu trong một bảng.

  DELETE [FROM] <table name>

  [WHERE <row conditions>];

- Thí dụ :

  Delete from Sinhvien where namsinh = 1988;

  sẽ xóa các sinh viên trong bảng Sinhvien mà có field namsinh = 1988.

Một số lệnh SQL thường dùng : UPDATE

- Lệnh Update dùng để cập nhật nội dung 1 hay nhiều record dữ liệu trong một bảng.

  UPDATE <table name> [<alias>]

  SET <column1> = {<expression>, <subquery>}

  [, <column2> = {<expression>, <subquery>} …]

  [WHERE <row conditions>];

- Thí dụ :

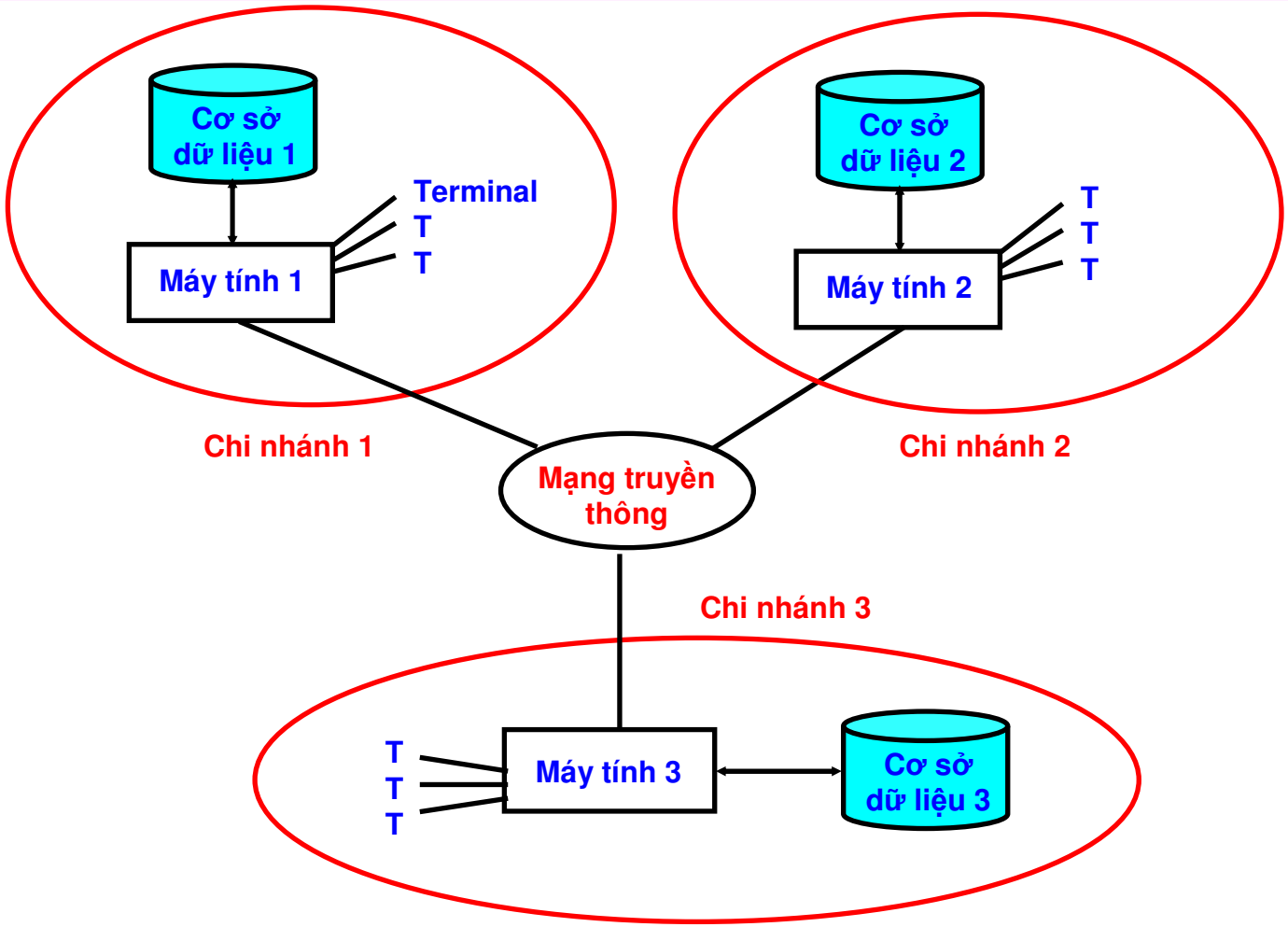
  Update Sinhvien Set namsinh = 1990 where namsinh = 1988;

  sẽ cập nhật lưu giá trị 1990 vào field namsinh của các sinh viên trong bảng Sinhvien mà có field namsinh = 1988.

5.6 Cơ sở dữ liệu phân tán

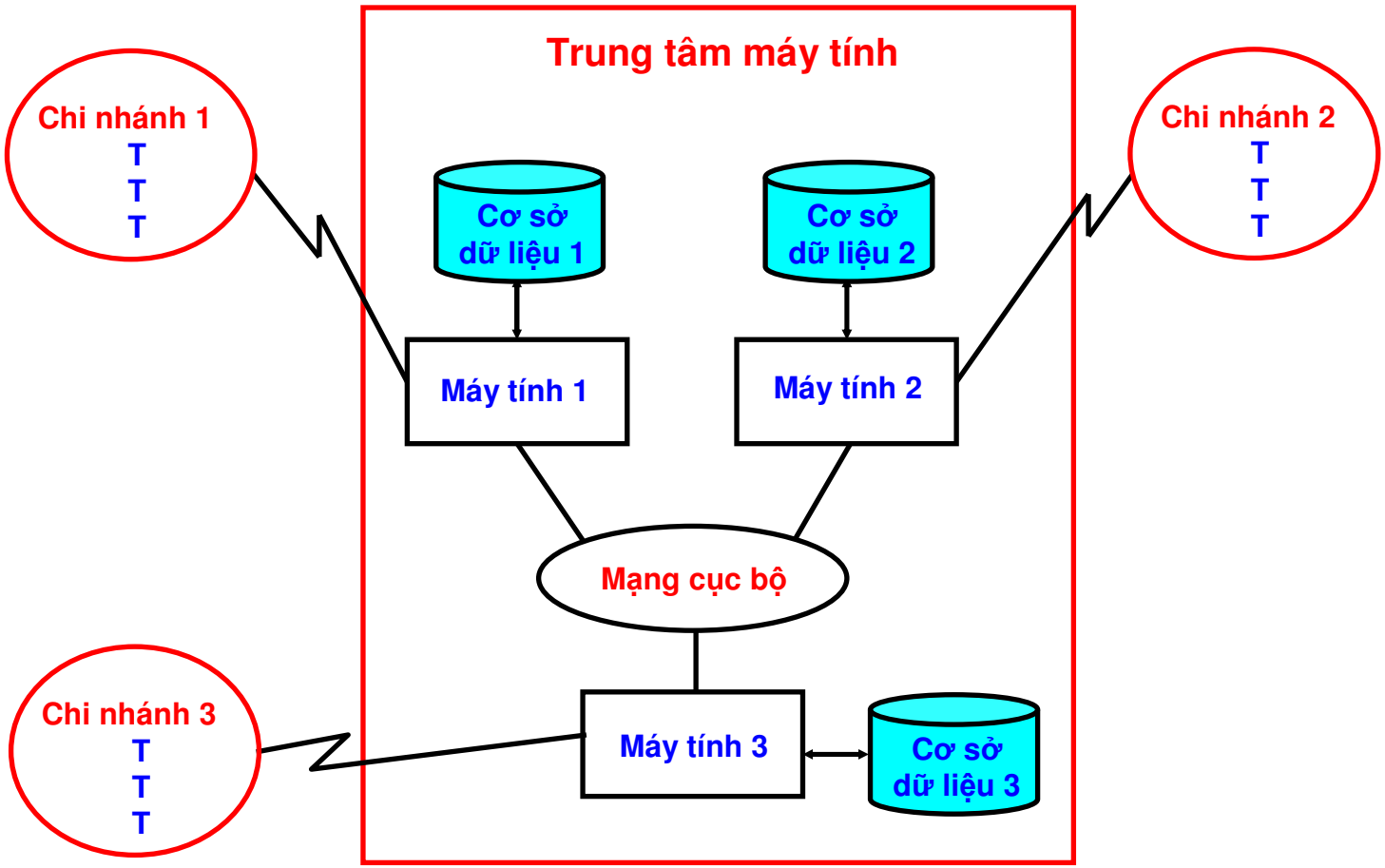
- Định nghĩa 1 : *Cơ sở dữ liệu phân tán* (*distributed database*) là sự tập hợp dữ liệu mà về mặt luận lý chúng *thuộc cùng một hệ thống* nhưng *được đặt* ở nhiều nơi (*site*) của một mạng máy tính.
  - *Sự phân tán dữ liệu* (*data distribution*): dữ liệu phải được phân tán ở nhiều nơi.
  - *Sự tương quan luận lý* (*logical correlation*): dữ liệu của các nơi được sử dụng chung để cùng giải quyết một vấn đề.
- Ví dụ
  - Một ngân hàng có ba chi nhánh ở các vị trí địa lý khác nhau.
  - Tại mỗi chi nhánh có một máy tính và một cơ sở dữ liệu tài khoản, tạo thành một *nơi* (*site*) của cơ sở dữ liệu phân tán.
  - Các máy tính được kết nối với nhau thông qua một mạng máy tính truyền thông.
  - Một khách hàng có thể gửi tiền và rút tiền tại các chi nhánh.

Cơ sở dữ liệu phân tán



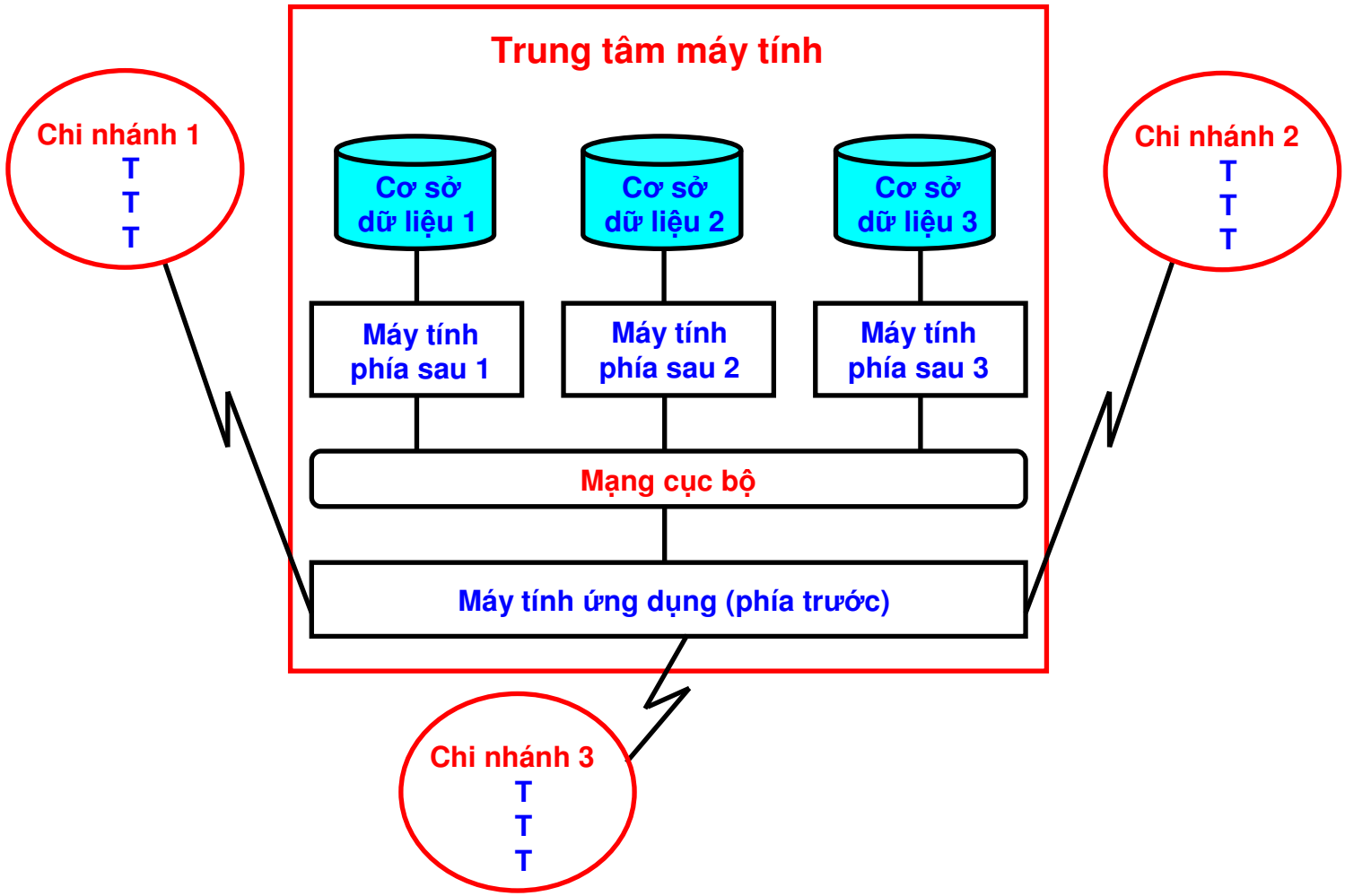
Cơ sở dữ liệu phân tán trên một mạng phân tán địa lý.

Cơ sở dữ liệu phân tán



Cơ sở dữ liệu phân tán trên một mạng cục bộ.

Cơ sở dữ liệu phân tán



Hệ thống đa xử lý (*multiprocessor system*).

Cơ sở dữ liệu phân tán

- Định nghĩa 2 : *Cơ sở dữ liệu phân tán* là sự tập hợp dữ liệu *được phân tán* trên các máy tính khác nhau của một mạng máy tính. Mỗi nơi của mạng máy tính có khả năng xử lý tự trị và có thể *thực hiện* các ứng dụng cục bộ. Mỗi nơi cũng *tham gia thực hiện* ít nhất một ứng dụng toàn cục, mà nơi này yêu cầu truy xuất dữ liệu ở nhiều nơi bằng cách dùng hệ thống truyền thông con.
  - Sự phân tán dữ liệu (data distribution): dữ liệu phải được phân tán ở nhiều nơi.
  - Ứng dụng cục bộ (local application): ứng dụng được chạy hoàn thành tại một nơi và chỉ sử dụng dữ liệu cục bộ của nơi này.
  - Ứng dụng toàn cục (hoặc ứng dụng phân tán) (global application / distributed application): ứng dụng được chạy hoàn thành và sử dụng dữ liệu của ít nhất hai nơi.

Chương 6

PHẦN MỀM ỨNG DỤNG

Một số ý niệm tổng quát

- Với đặc tính của máy tính số, nó có thể giải quyết bất kỳ bài toán nào thuộc lĩnh vực gì nếu con người biết được giải thuật giải quyết bài toán đó và miêu tả được giải thuật bằng ngôn ngữ lập trình cho máy tính hiểu.
- Hiện nay, máy tính số (hay lĩnh vực công nghệ thông tin) đã và đang được sử dụng rộng rãi và phổ biến trong hầu hết các cá nhân, đơn vị, địa phương, vùng miền... Mỗi vị trí sử dụng máy tính thường sử dụng chủ yếu 1 số ít ứng dụng liên quan đến lĩnh vực mình cần.
- Tóm lại, số lượng ứng dụng mà con người đã viết, sử dụng là rất lớn và đa dạng, phong phú về chức năng xử lý. Tuy nhiên, ứng với vị trí sử dụng cụ thể của 1 đối tượng cụ thể, chỉ 1 số rất ít ứng dụng liên quan mật thiết đến lĩnh vực xử lý mới được dùng thường xuyên..
- Trong chương này, chúng ta chỉ giới thiệu 1 số ứng dụng điển hình và phổ biến.

1. Hệ điều hành

- Hệ điều hành (Operating System) là phần mềm quản lý các tài nguyên cấp thấp (thường là phần cứng), che dấu các tính chất vật lý của chúng (thường rất khó hiểu và sử dụng), rồi cung cấp lại một interface sử dụng chúng với các lợi điểm như an toàn, tin cậy, thân thiện, hiệu quả và nhất là độc lập với tính chất vật lý của tài nguyên được sử dụng. Người ta còn gọi HĐH là máy ảo (máy luận lý).
- Hiện 2 HĐH được sử dụng phổ biến nhất là Windows (XP, Vista) và Linux.
- ROM BIOS của máy PC có thể được xem là HĐH quản lý các tài nguyên vật lý của máy PC, Windows hay Linux là HĐH chạy trên ROM BIOS. Ứng dụng cụ thể sẽ chạy trên HĐH. Người dùng sẽ làm việc với ứng dụng.

2. Chương trình dịch

- Máy tính chỉ có thể chạy trực tiếp các chương trình viết bằng lệnh máy. Nhưng lập trình bằng ngôn ngữ máy rất khó, tốn nhiều công sức, thời gian mà độ tin cậy, đúng đắn của chương trình lại thấp, chi phí bảo trì và nâng cấp rất cao. Do đó, hầu hết các ứng dụng đều được viết bằng ngôn ngữ cấp cao như C++, Java,...
- Cần phải có chương trình dịch chương trình từ mã nguồn sang mã máy. Có 2 loại chương trình dịch : trình biên dịch (compiler) và trình thông dịch (interpreter)
- Mỗi lần chạy, trình biên dịch sẽ dịch các file mã nguồn sang dạng mã máy tương đương (thường được link lại thành file khả thi - *.exe). Mỗi lần chạy ứng dụng, ta chỉ kích hoạt file khả thi.
- Mỗi lần chạy, trình thông dịch sẽ thực thi từng lệnh mã nguồn bằng cách dịch lệnh ấy sang danh sách lệnh máy tương đương rồi nhờ máy thực thi danh sách lệnh máy tương đương này. Như vậy, mỗi lần thông dịch là 1 lần chạy ứng dụng mã nguồn. Muốn chạy lại lần nữa, phải thông dịch lại từ đầu.

3. Ứng dụng văn phòng

- Cho phép người dùng thực hiện 1 số chức năng thông thường liên quan đến văn phòng. Microsoft Office là ứng dụng văn phòng được sử dụng phổ biến nhất. Open Office là ứng dụng văn phòng mã nguồn mở nhưng yếu hơn và thiếu ổn định hơn
- Microsoft Office là tập các ứng dụng độc lập : Word cho phép xử lý tài liệu văn bản ; Excel cho phép xử lý các bảng tính số liệu ; PowerPoint cho phép xử lý các slide bài giảng, thuyết trình ; Access cho phép xử lý database...
- Thật ra Microsoft đã nâng cấp các ứng dụng văn phòng để từng ứng dụng riêng lẻ trong bộ Office trở thành chương trình đa mục tiêu :

3. Ứng dụng văn phòng (tt)

- Thí dụ Word được dùng chủ yếu như là 1 ứng dụng xây dựng và xử lý tài liệu văn bản (đơn từ, giấy tờ, sách báo, thuyết minh đề án, luận văn,..).
- Nhưng nhờ khả năng macro và cho phép người dùng thiết lập lại hệ thống menu bar và toolbar nên người dùng có thể biến Word nguyên thủy thành 1 ứng dụng với chức năng riêng biệt nào đó. Ta nói Word là 1 ứng dụng tổng quát hóa.
- Ngoài ra, trong tài liệu Word mà người dùng xây dựng không chỉ chứa các nội dụng văn bản, hình ảnh tĩnh, mà còn được phép chèn vào vị trí cần thiết đối tượng giao diện (button, TextBox,..) để biến tài liệu Word thành giao diện trực quan của ứng dụng cụ thể cho người dùng. Như vậy tài liệu Word trở thành phần mềm và Word được xem như là môi trường thiết kế trực quan phần mềm.

4. Ứng dụng nghiệp vụ & Database server

- Ứng dụng nghiệp vụ thực hiện các yêu cầu nghiệp vụ trong các cơ quan, đơn vị như quản lý nhân viên, quản lý tài sản, quản lý điểm, quản lý bệnh nhân và bệnh án,...
- Trong hầu hết các ứng dụng nghiệp vụ, dữ liệu cần lưu trữ và xử lý là rất lớn. Vấn đề lưu trữ và quản lý những dữ liệu lớn sao cho nhất quán, an toàn tin cậy,... đòi hỏi nhiều kiến thức chuyên sâu và nhiều thời gian công sức hiện thực.
- Database server ra đời nhằm giải phóng ứng dụng khỏi việc lưu trữ và quản lý khối dữ liệu lớn mà mình muốn sử dụng.
- Có nhiều database server với qui mô khác nhau như Excel, FoxPro, Access, MySQL, SQL, Oracle,... Tùy mức độ quản lý dữ liệu và độ lớn dữ liệu cần quản lý, ta nên chọn database server phù hợp.

5. Biên tập & chơi multimedia

- Multimedia là dữ liệu đa phương tiện như văn bản, hình ảnh, âm thanh, film,... Dữ liệu đa phương tiện giúp người dùng thích thú hơn khi nghiên cứu về 1 vấn đề nào đó.
- Các ứng dụng biên tập dữ liệu đa phương tiện cho phép ta xây dựng, thêm/bớt/hiệu chỉnh thông tin và file đa phương tiện tương ứng. Thí dụ trình Photoshop cho ta xử lý ảnh tĩnh, SoundGold cho phép ta xử lý âm thanh, Photo Premiere cho ta biên tập film...
- Các ứng dụng chơi multimedia cho phép người dùng tham khảo file multimedia đã có. Thí dụ trình Window Multimedia Player của Microsoft cho ta chơi hầu hết các định dạng file multimedia khác nhau từ ảnh tĩnh, âm thanh hay film. File multimedia cần chơi có thể nằm trên máy đơn hay trên 1 server multimedia nào đó trong mạng Internet.

6. Game

- Game (ứng dụng trò chơi) là những ứng dụng dễ lôi cuốn người dùng nhất.
- Có 2 thể loại game phổ biến : game hành động và game trí tuệ.
- Loại game hành động đòi hỏi chủ yếu sự lanh lẹ, kịp thời trong các thao tác của người chơi. Nhưng thường để có phản ứng lanh lẹ, kịp thời, người chơi phải tích lũy rất nhiều thời gian chơi để có được phản ứng không điều kiện (theo phản xạ). Võ lâm truyền kỳ là 1 game khá phổ biến ở nước ta trong thời gian qua.
- Loại game trí tuệ đòi hỏi khả năng tư duy cao, sự kiên nhẫn và trầm tĩnh của người chơi. Nói chung người chơi có óc suy luận cao, có khả năng toán học tốt thường thích hợp cho những trò chơi trí tuệ này. Cờ tướng, cờ vua,... là những game trí tuệ rất phổ biến.

7. Các ứng dụng trên mạng Internet

- Internet là mạng nối các máy tính của nhiều người trên toàn thế giới lại với nhau. Hiện tuyệt đại đa số các máy của người dùng đều được nối mạng Internet (hoặc online hay offline).
- Ứng dụng mạng là ứng dụng sử dụng nhiều tài nguyên của nhiều máy khác nhau trên mạng. Ứng dụng mạng gồm nhiều module chức năng, mỗi module chạy trên 1 máy.
- Thường ứng dụng mạng dùng mô hình hoạt động client/server, mỗi module sẽ đóng vai trò hoặc server, hoặc client. Module server sẽ quản lý các tài nguyên liên quan trên máy mình đang chạy và cung cấp dịch vụ truy xuất các tài nguyên này cho các module ở các máy khác. Module client sẽ chạy trên máy người dùng, cung cấp giao tiếp sử dụng thân thiện, dễ dàng, an toàn,...
- Các module server/client cửa 1 ứng dụng mạng thường tuân thủ 1 giao thức xác định nào đó. Giao thức (protocol) là tập các thông báo request/reply cùng định dạng cụ thể của từng thông báo mà client/server sẽ gởi/nhận cho nhau.

7. Các ứng dụng trên mạng Internet (tt)

- Mỗi khi người dùng yêu cầu 1 chức năng nào đó, client sẽ xây dựng 1 thông báo request chứa thông tin về chức năng đó gởi đến server. Server nhận, phân tích và thực thi. Kết quả sẽ được đóng gói thành 1 thông báo reply để gởi về client.

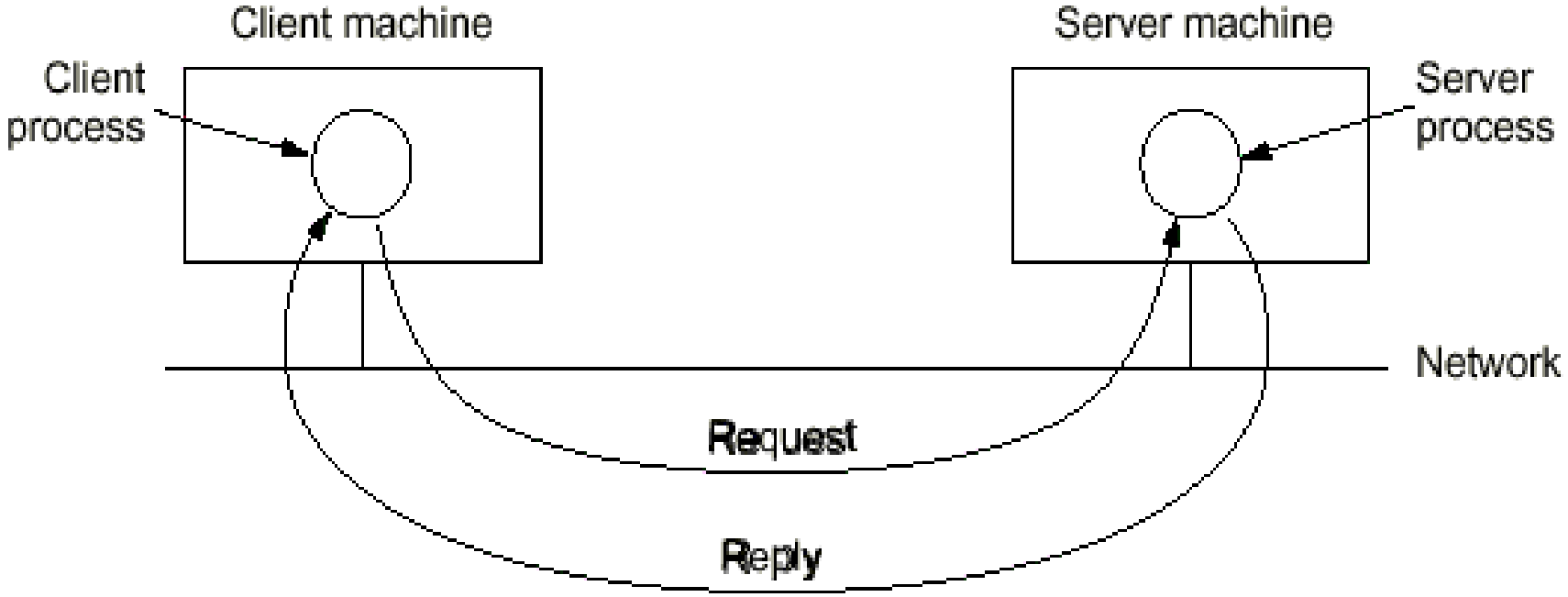


Fig. 1-1. The client-server model.

7. Các ứng dụng trên mạng Internet (tt)

- Mỗi máy tính cần từ 1 tới nhiều card giao tiếp mạng để nối máy với mạng. Mỗi card mạng sẽ được nhận dạng bởi 1 địa chỉ IP duy nhất. Địa chỉ IP là 1 số nguyên 4 byte (32 bit).
- Mỗi máy tính có thể chạy đồng thời nhiều ứng dụng, các ứng dụng này có thể là ứng dụng mạng. Mỗi ứng dụng mạng được nhận dạng duy nhất trong Internet bởi địa chỉ TCP của nó. Địa chỉ TCP là sự nối kết 2 thông tin : địa chỉ IP của máy + port giao tiếp của phần mềm (số nguyên 2 byte).
- Mạng Internet chứa rất nhiều tài nguyên, mỗi tài nguyên được nhận dạng 1 cách duy nhất nhờ tên nhận dạng của nó (URL)

Khái niệm URL

URL (Uniform Resource Locator) là phương tiện giải quyết đồng thời 3 chức năng :

- Xác định giao thức được dùng để truy xuất tài nguyên.
- Xác định địa chỉ máy (thường là địa chỉ DNS) chứa tài nguyên.
- Xác định vị trí cụ thể của tài nguyên trên máy (pathname).

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	/usr/suzanne/prog.c
news	News group	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending email	mailto:kim@acm.org
telnet	Remote login	telnet://www.w3.org:80

Các ứng dụng mạng phổ biến

7.1 Hệ thống DNS (Domain Name System)

7.2 Hệ thống E-mail

7.3 Hệ thống FTP

7.4 Hệ thống WWW

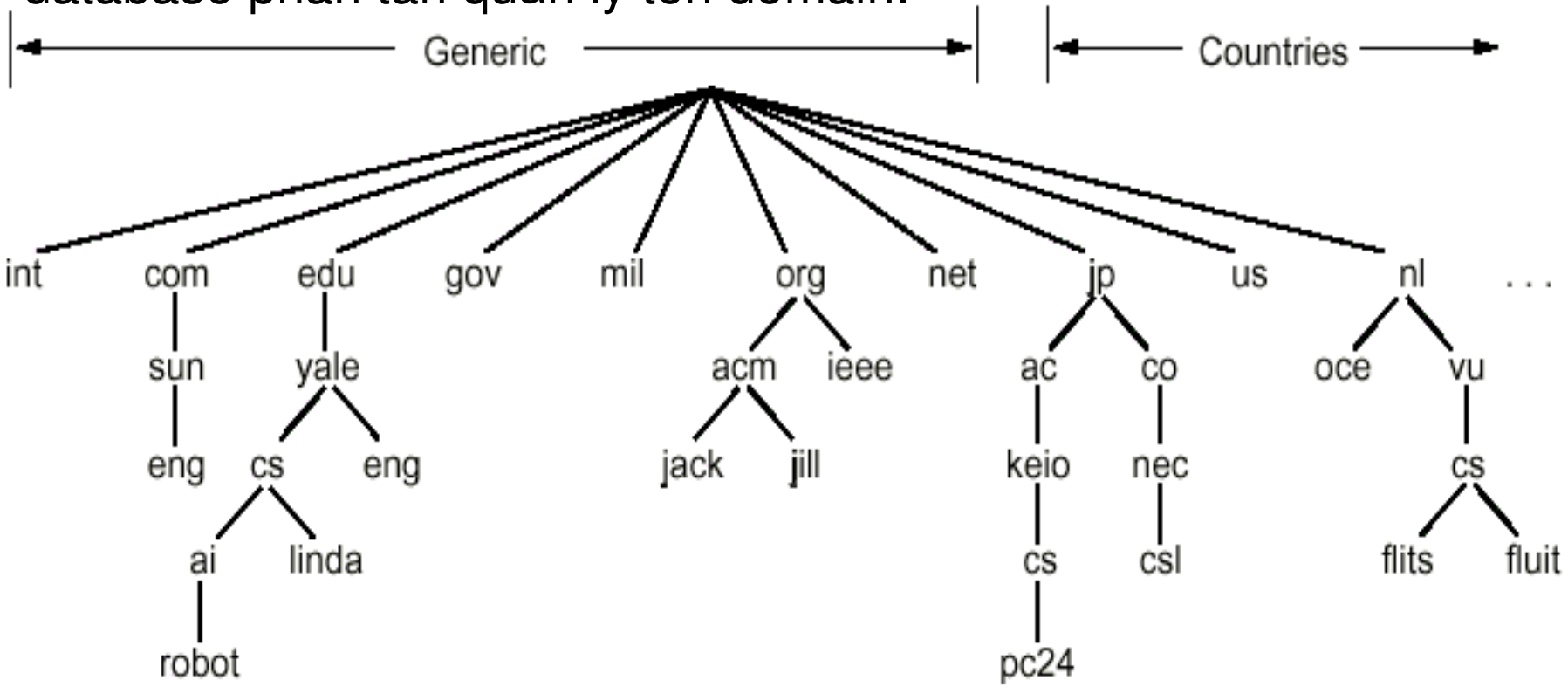
7.5 Hệ thống Chat

7.1 Hệ thống DNS

Nhiệm vụ : đổi địa chỉ từ gợi nhớ sang IP

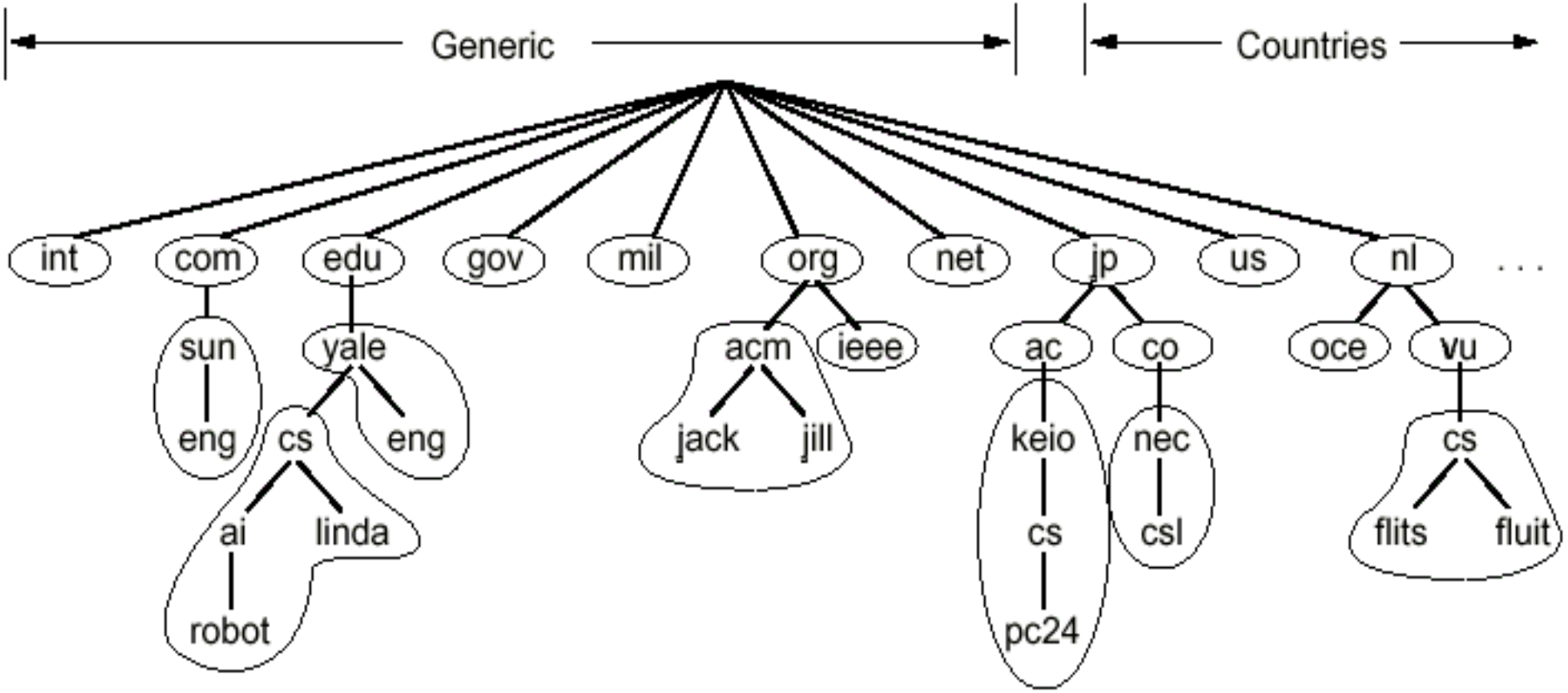
Gồm 2 thành phần :

- không gian tên dạng thứ bậc dùng khái niệm domain (miền).
- database phân tán quản lý tên domain.



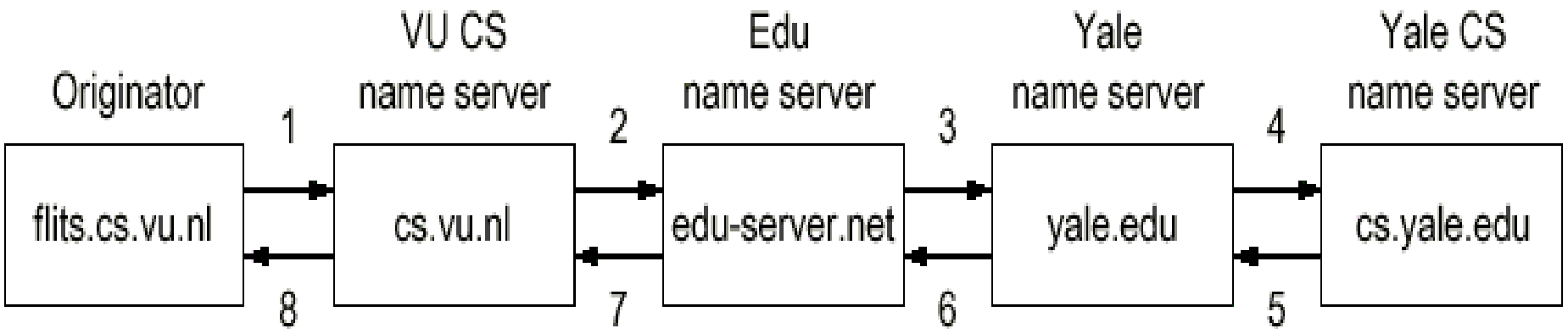
7.1 Hệ thống DNS (tt)

Database phÝn t¾n quÀn lû tãn domain : gëm nhiảu zone, míi zone chöa c¾c record vả c¾c domain ½õôc chöa trong zone, cÍ 1 server chÈnh (primary server) v¿ nhiảu server phÖ quÀn lû (Secondary server).

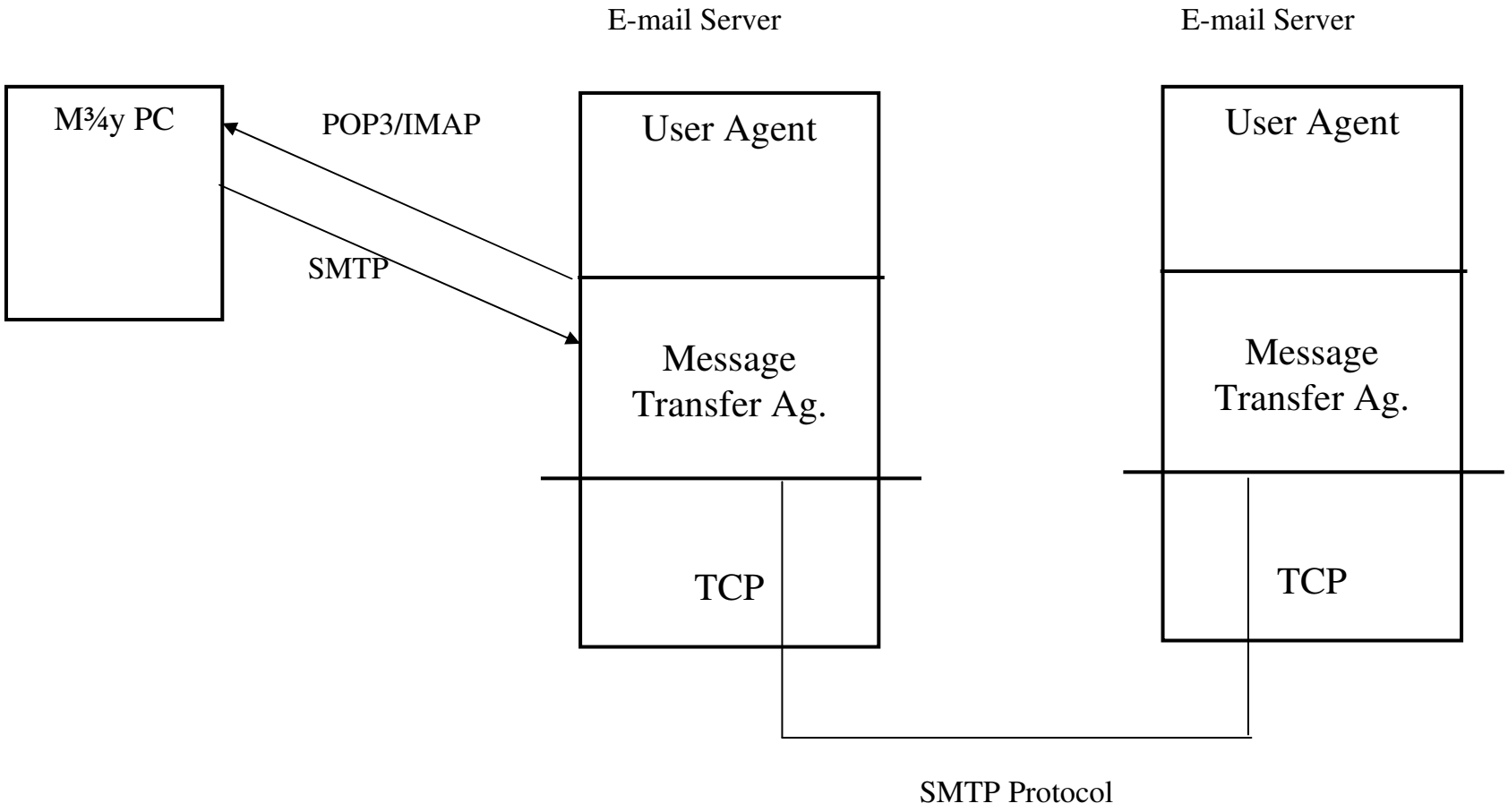


7.1 Hệ thống DNS

TrÉnh tú gòi request/reply dÎ hÏi théng tin ½iæn hÉnh :



7.2 Hệ thống E-Mail : Mô hình hoạt động



Hệ thống E-Mail : Mô hình hoạt động

- Server quản lý n account email, mỗi account được quản lý bởi 1 record dữ liệu :

Username
Password
Inbox (mailbox nhêån)
Outbox (mailbox gúãi)
caác thöng tin khaác

- Khi có yêu cầu, các server e-mail sẽ tạo cầu nối TCP với nhau (dùng port 25), trên cầu nối này, chúng sẽ giao tiếp với nhau bằng giao thức SMTP (Simple Mail Transfer Protocol) để gởi/nhận e-mail của nhau.
- User thường sẽ chạy trên máy PC hay Workstation, dùng 1 phần mềm mail-client nào đó (Outlook,...) để truy cập account của mình trên máy server thông qua các giao thức POP3, IMAP4, SMTP,...

Hệ thống E-Mail : Cấu trúc 1 email

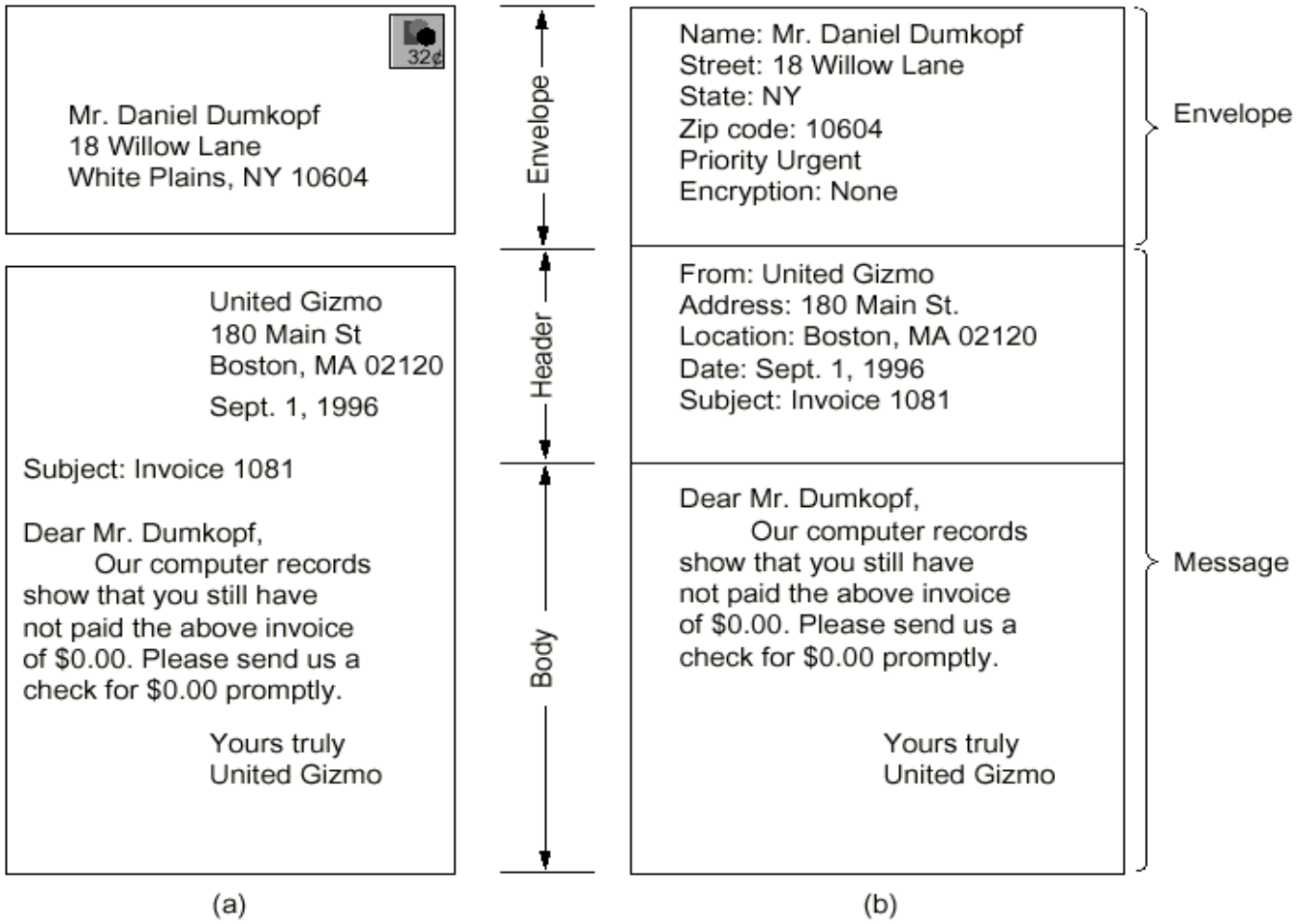
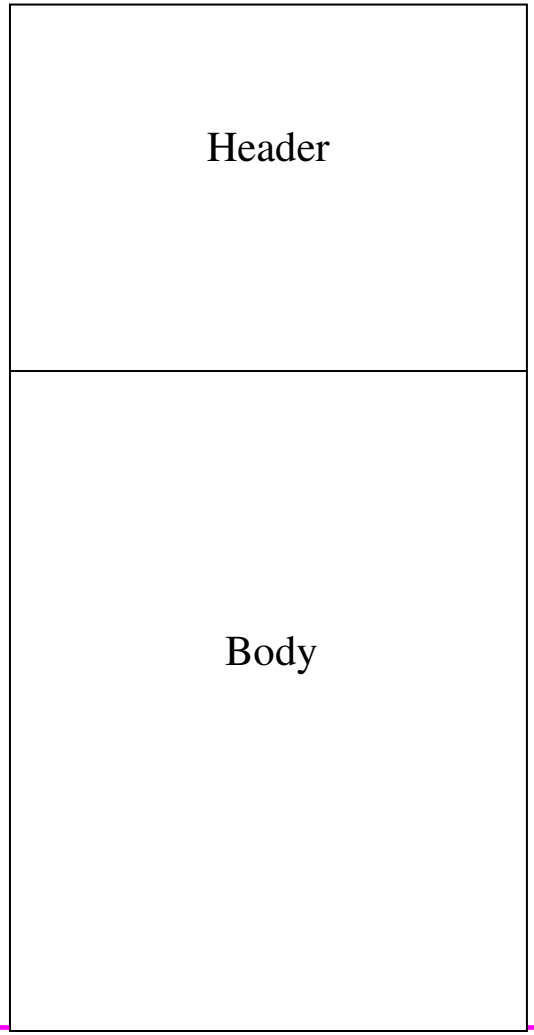


Fig. 7-39. Envelopes and messages. (a) Postal email. (b) Electronic mail.

Hệ thống E-Mail : Cấu trúc 1 email



<tïn liånh 1> : giaá trõ
<tïn liånh 1> : giaá trõ
<tïn liånh 1> : giaá trõ
.....

paragraph 1 cuãa nöåi dung
paragraph 2 cuãa nöåi dung
paragraph 3 cuãa nöåi dung
.....
paragraph n cuãa nöåi dung

Hệ thống E-Mail : Các lệnh trong header

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Fig. 7-42. RFC 822 header fields related to message transport.

Hệ thống E-Mail : Các lệnh trong header

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User chosen keywords
Subject:	Short summary of the message for the one-line display

Fig. 7-43. Some fields used in the RFC 822 message header.

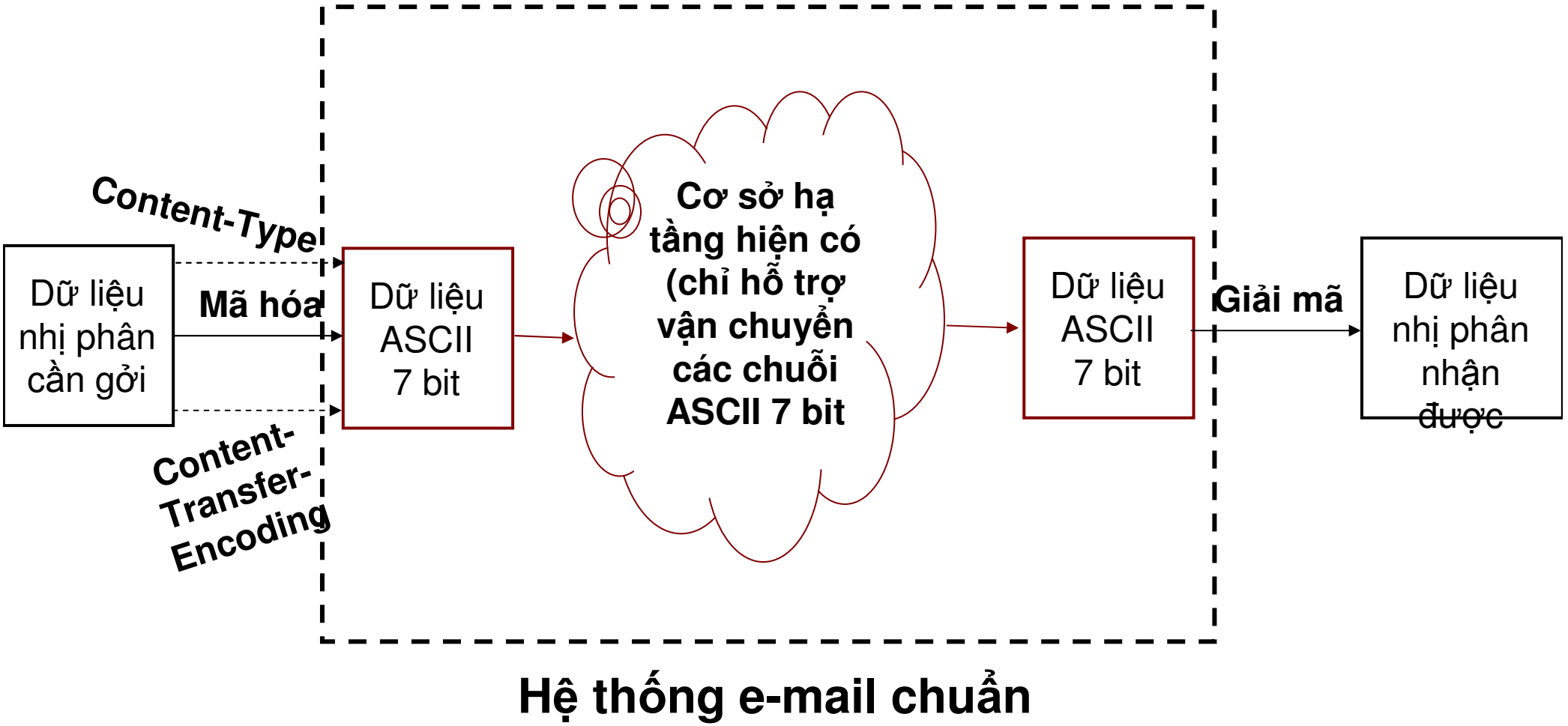
Hệ thống E-Mail : Các lệnh trong header

Format chuẩn của e-mail chỉ chứa các ký tự ASCII 7 bit, để có thể gởi/nhận các file nhị phân, ta dùng chuẩn nới rộng MIME (Multipurpose Internet Mail Extension) với các lệnh tăng cường sau đây :

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Nature of the message

Fig. 7-44. RFC 822 headers added by MIME.

Hệ thống E-Mail : Nguyên tắc làm việc của chuẩn MIME



Hệ thống E-Mail : Các giá trị điển hình của Content-Type

Type	Subtype	Description
Text	Plain	Unformatted text
	Richtext	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

Hệ thống E-Mail : Các giá trị của Content-Transfer-Encoding

Các phương pháp mã hóa thông thường & giá trị tương ứng trong lệnh Content-Transfer-Encoding:

1. **ASCII 7 bit** : e-mail nhận được là 1 chuỗi <= 1000 ký tự 7 bit ASCII
2. **ASCII 8 bit** : e-mail nhận được là 1 chuỗi <= 1000 ký tự 8 bit ASCII
3. **Binary** : e-mail nhận được là 1 chuỗi byte 8 bit có độ dài tùy ý.
4. **Base64** : e-mail nhận được đã bị mã hóa bằng phương pháp Base64
5. **Quoted-printed** : e-mail nhận được đã bị mã hóa bằng phương pháp Quoted-printed
6. **User-defined** : e-mail nhận được đã bị mã hóa bằng phương pháp riêng của người gởi.

Hệ thống E-Mail : Giao thức SMTP

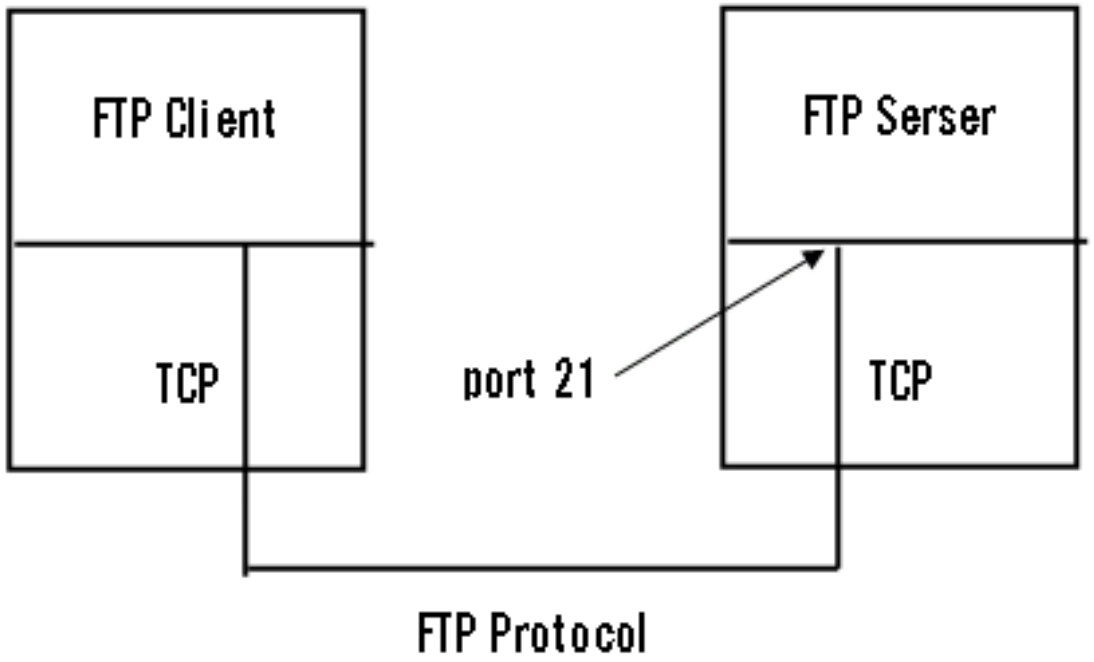
1. HELLO (HELO)
   HELO dit.hcmut.edu.vn
2. MAIL (MAIL)
   MAIL FROM:<hiep@dit.hcmut.edu.vn >
3. RECIPIENT (RCPT)
   RCPT TO:< nvhung@vnn.vn>
4. DATA (DATA)
   DATA
5. SEND (SEND)
   SEND FROM:<hiep@dit.hcmut.edu.vn >
6. SEND OR MAIL (SOML)
   SOML FROM:<hiep@dit.hcmut.edu.vn >
7. SEND AND MAIL (SAML)
   SAML FROM:<hiep@dit.hcmut.edu.vn >
8. RESET (RSET)
   RSET
9. VERIFY (VRFY)
   VRFY nvhung
10. EXPAND (EXPN)
    EXPN mailing-list name
11. HELP (HELP)
    HELP
12. NOOP (NOOP)
    NOOP
13. QUIT (QUIT)
    QUIT
14. TURN (TURN)

Hệ thống E-Mail : Giao thức SMTP

Một session e-mail đơn giản :

S: 220 xyz.com SMTP service ready
C: HELO abc.com
S: 250 xyz.com says hello to abc.com
C: MAIL FROM: <elinor@abc.com>
S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
S: 250 recipient ok
C: DATA
S: 354 Send mail; end with "." on a line by itself
C: From: elinor@abc.com
C: To: carolyn@xyz.com
....
C: .
S: 250 message accepted
C: QUIT
S: 221 xyz.com closing connection

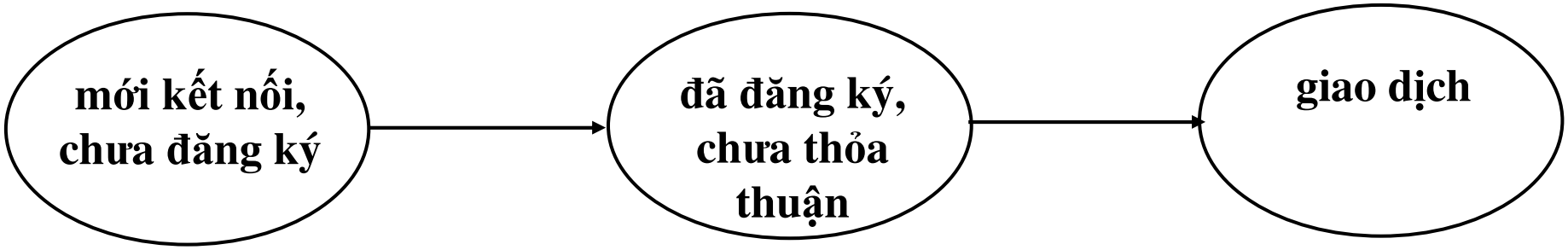
7.3 Hệ thống FTP : Mô hình hoạt động



- FTP Server là chương trình quản lý hệ thống cây thư mục các file trên máy mình và phục vụ việc truy xuất các file này từ xa.
- FTP Client giao tiếp với người dùng cho phép họ truy xuất cây thư mục của các FTP Server từ xa.
- FTP Server và FTP Client giao tiếp nhau bằng giao thức FTP (File Transfer Protocol).

Hệ thống FTP : Giao thức FTP

Mỗi session làm việc của người dùng trải qua 3 trạng thái :



Các lệnh request FTP được chia làm 3 nhóm chức năng tương ứng với 3 trạng thái :

1. ACCESS CONTROL COMMANDS

USER NAME: USER <SP> <username> <CRLF>

PASSWORD: PASS <SP> <password> <CRLF>

ACCOUNT: ACCT <SP> <account-information> <CRLF>

CHANGE WORKING DIRECTORY: CWD <SP> <pathname> <CRLF>

CHANGE TO PARENT DIRECTORY: CDUP <CRLF>

STRUCTURE MOUNT: SMNT <SP> <pathname> <CRLF>

REINITIALIZE: REIN <CRLF>

LOGOUT: QUIT <CRLF>

Hệ thống FTP : Giao thức FTP

2. TRANSFER PARAMETER COMMANDS

DATA PORT: PORT <SP> h1,h2,h3,h4,p1,p2 <CRLF>

PASSIVE: PASV <CRLF>

REPRESENTATION TYPE: TYPE <SP> <type-code> <CRLF>

\ /

A - ASCII | | N - Non-print

|-><-| T - Telnet format effectors

E - EBCDIC | | C - Carriage Control (ASA)

/ \

I - Image

L <byte size> - Local byte Byte size

FILE STRUCTURE: STRU <SP> <structure-code> <CRLF>

F - File (no record structure)

R - Record structure

P - Page structure

TRANSFER MODE: MODE <SP> <mode-code> <CRLF>

S - Stream

B - Block

C - Compressed

Hệ thống FTP : Giao thức FTP

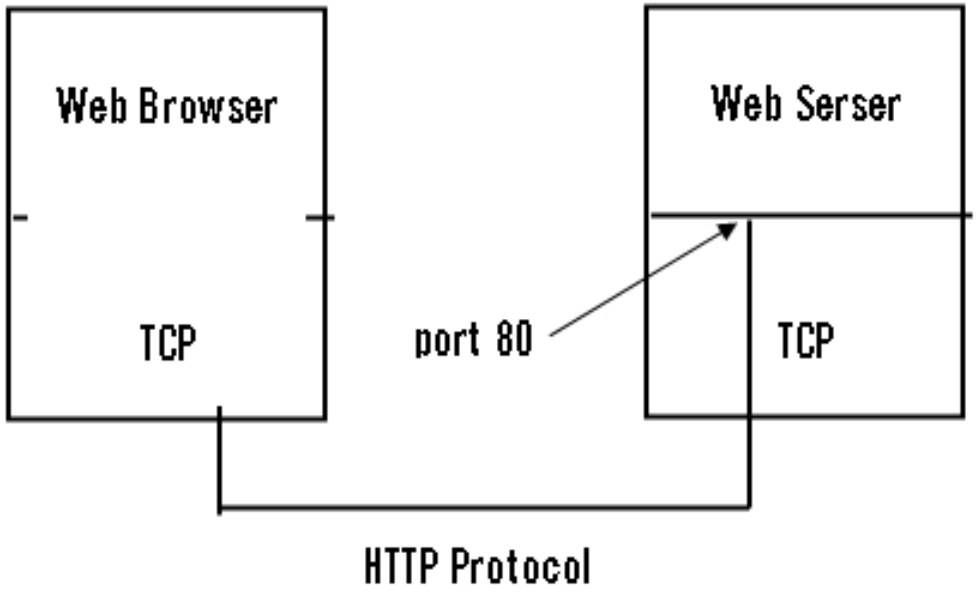
RETRIEVE: RETR <SP> <pathname> <CRLF>
STORE: STOR <SP> <pathname> <CRLF>
STORE UNIQUE: STOU <CRLF>
APPEND (with create) : APPE <SP> <pathname> <CRLF>
ALLOCATE: ALLO <SP> <decimal-integer> [<SP> R <SP> <decimal-integer>] <CRLF>
RESTART: REST <SP> <marker> <CRLF>
RENAME TO: RNTO <SP> <pathname> <CRLF>
RENAME FROM : RNFR <SP> <pathname> <CRLF>
ABORT: ABOR <CRLF>
DELETE: DELE <SP> <pathname> <CRLF>
REMOVE DIRECTORY: RMD <SP> <pathname> <CRLF>
MAKE DIRECTORY: MKD <SP> <pathname> <CRLF>
PRINT WORKING DIRECTORY: PWD <CRLF>
LIST: LIST [<SP> <pathname>] <CRLF> STAT [<SP>
NAME LIST: NLST [<SP> <pathname>] <CRLF> pathname>] <CRLF>
SITE PARAMETERS: SITE <SP> <string> <CRLF> HELP: HELP [<SP> <string>]
SYSTEM: SYST <CRLF> <CRLF>
NOOP: NOOP <CRLF>

Hệ thống FTP : Giao thức FTP

1yz Positive Preliminary reply
2yz Positive Completion reply
3yz Positive Intermediate reply
4yz Transient Negative Completion reply
5yz Permanent Negative Completion reply
x0z Syntax - These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands.
x1z Information - These are replies to requests for information, such as status or help.
x2z Connections - Replies referring to the control and data connections.
x3z Authentication and accounting - Replies for the login process and accounting procedures.
x4z Unspecified as yet.
x5z File system - These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action.

7.4 Hệ thống WWW : Mô hình hoạt động

IV.1 Mô hình hoạt động



Gồm 2 loại phần tử :

- Web server : chương trình quản lý một cây thứ bậc các trang Web và phục vụ yêu cầu truy xuất chúng từ các client từ xa.
- Web Browser : chương trình giao tiếp với người dùng, nhận yêu cầu từ user rồi truy xuất trang Web ở server tương ứng để phân giải và hiển thị nội dung lên màn hình.

Hệ thống WWW : Giao thức HTTP

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
LINK	Connects two existing resources
UNLINK	Breaks an existing connection between two resources

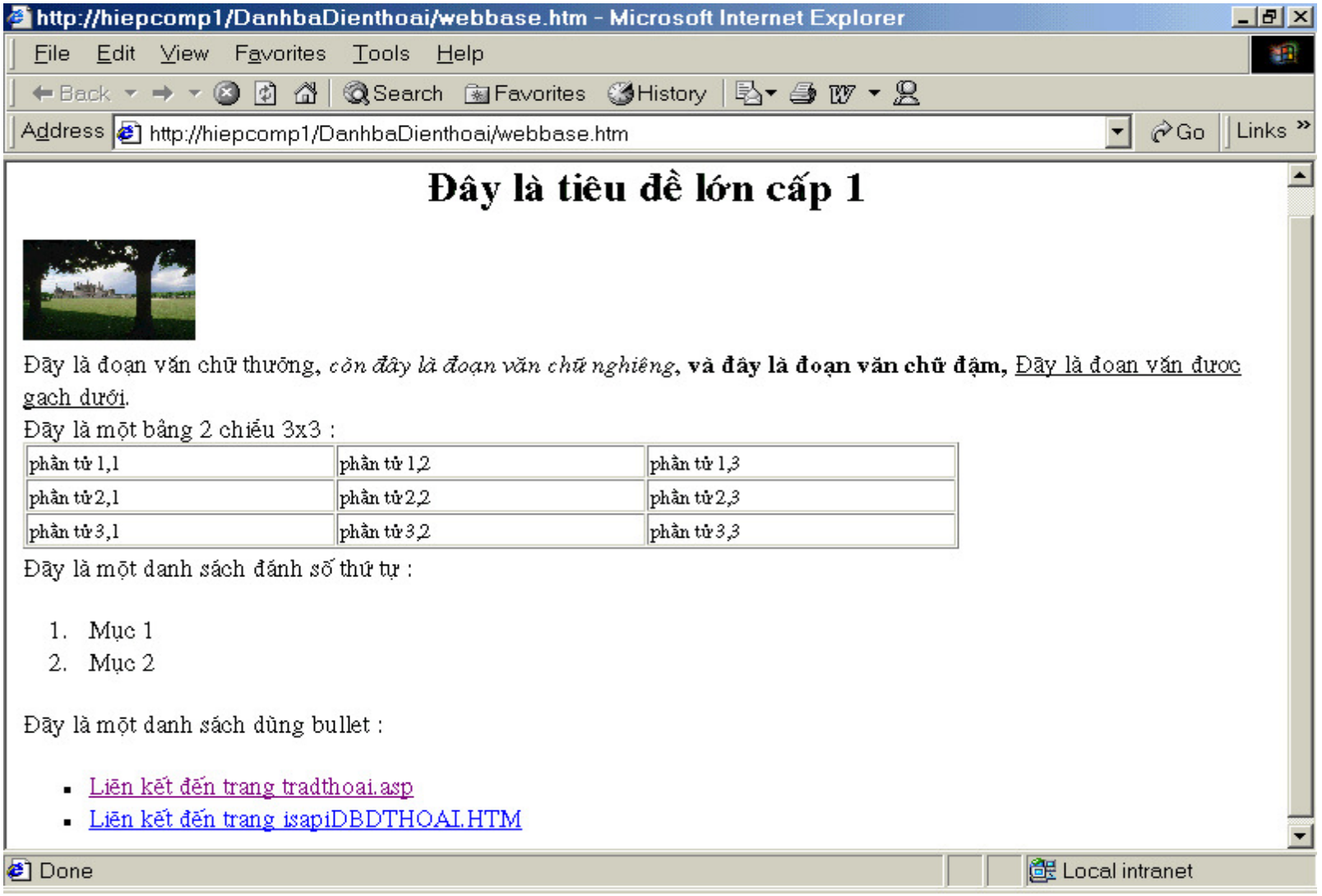
Hệ thống WWW : Các tag lệnh HTML cơ bản

<HTML> ... </HTML>	Declares the Web page to be written in HTML
<HEAD> ... </HEAD>	Delimits the page's head
<TITLE> ... </TITLE>	Defines the title (not displayed on the page)
<BODY> ... </BODY>	Delimits the page's body
<H*n*> ... </H*n*>	Delimits a level *n* heading
<B> ... </B>	Set ... in boldface
<I> ... </I>	Set ... in italics
<UL> ... </UL>	Brackets an unordered (bulleted) list
<OL> ... </OL>	Brackets a numbered list
<MENU> ... </MENU>	Brackets a menu of <LI> items
<LI>	Start of a list item (there is no </LI>)
 	Force a break here
<P>	Start of paragraph
<HR>	Horizontal rule
<PRE> ... </PRE>	Preformatted text; do not reformat
<IMG SRC="...">	Load an image here
<A HREF="..."> ... </A>	Defines a hyperlink

Hệ thống WWW : Các tag lệnh HTML cơ bản

<TABLE> .... </TABLE>	Bảng thông tin
<TR> .... </TR>	hàng thuộc bảng
<TH> ... </TH>	phần tử header
<TD> ... </TD>	phần tử dữ liệu
<OBJECT> .... </OBJECT>	phần tử được chèn vào như ActiveX Control,...
<APPLET code=URL ...>	Applet Java
<SCRIPT> ..... </SCRIPT>	đoạn chương trình bằng ngôn ngữ script
.....	

Hệ thống WWW : Trang Web cơ bản



Hệ thống WWW : Source code của trang Web

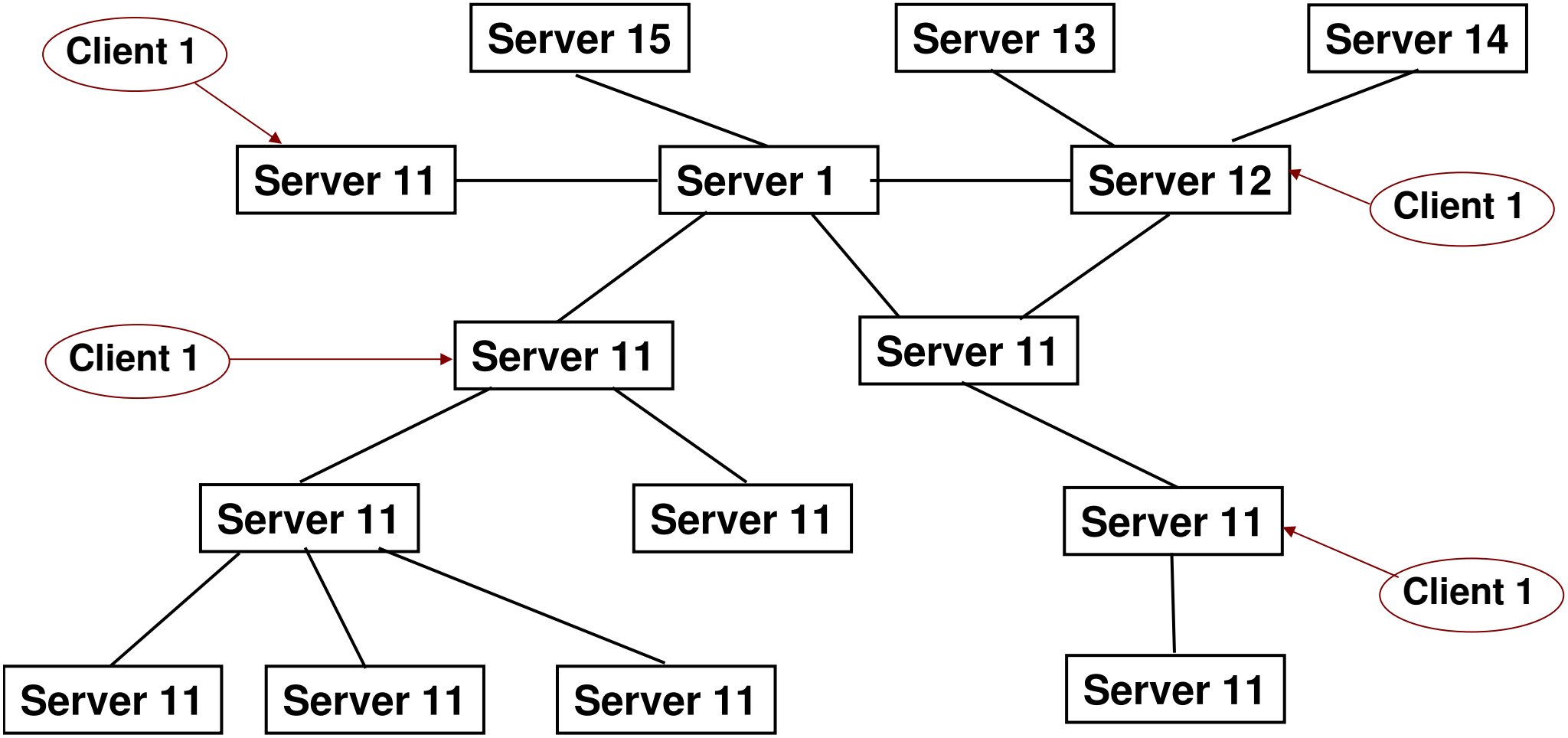
Source code HTML của trang Web cơ bản ở slide trước :

```
<HTML><HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=x-user-defined">
<TITLE>Trang Web demo các phần tử cơ bản</TITLE>
</HEAD>
<BODY>
<FONT face=VnTimes size=4>
<H1 align=center>Đây là tiêu đề lớn cấp 1</H1>
<IMG style="WIDTH: 105px; HEIGHT: 65px" height=431 src="Chateau.jpg" width=426 >
<BR>Đây là đoạn văn chữ thường, <EM>còn đây là đoạn văn chữ nghiêng</EM>,
<STRONG>và đây là đoạn văn chữ đậm, </STRONG>
<U>đây là đoạn văn được gạch dưới</U>.
<BR>Đây là 1 mảng 2 chiều 3x3 :
<TABLE cellSpacing=1 cellPadding=1 width="75%" border=1>
  <TR>
    <TD>phần tử 1,1</TD>
    <TD>phần tử 1,2</TD>
    <TD>phần tử 1,3</TD></TR>
```

Hệ thống WWW : Source code của trang Web

```
  <TR><TD>phần tử 2,1</TD> <TD>phần tử 2,2</TD> <TD>phần tử
2,3</TD></TR>
  <TR>
    <TD>phần tử 3,1</TD>
    <TD>phần tử 3,2</TD>
    <TD>phần tử 3,3</TD></TR>
</TABLE>
Đây là danh sách được đánh số thứ tự :
<OL>
  <LI>MÖc 1
  <LI>MÖc 2
</OL>
Đây là danh sách dùng bullet :
<UL>
  <LI><A href="tradthoai.asp">Liên kết đến trang tradthoai.asp </A>
  <LI><A href="isapiDBDTHOAI.HTM">Liên kết đến trang isapidbdthoai.htm </A>
</UL> </FONT>
</BODY></HTML
```

7.5 Hệ thống Chat : Mô hình hoạt động



7.5 Hệ thống Chat : Mô hình hoạt động

- Gồm nhiều server được nối kết lẫn nhau hầu có thể trao đổi thông tin lẫn nhau.
- Gồm nhiều user, mỗi user chạy trên 1 máy client, nối kết vào 1 chat server mình thích, đăng ký nickname, nối kết vào 1 nhóm chat xác định (nếu nhóm này chưa có thì hệ thống sẽ tạo mới và user sẽ trở thành admin của nhóm này).
- Admin của 1 nhóm chat có quyền thiết lập chế độ làm việc của nhóm, cho phép user khác nối kết vào, gỡ bỏ user ra khỏi nhóm,..
- User bình thường của nhóm chỉ có quyền trao đổi thông tin với các thành viên khác của nhóm.
- 1 User có thể đăng ký đồng thời vào nhiều nhóm khác nhau (max là 10).

8. Các ứng dụng CAD (Computer-Aided Design)

- Giúp người thiết kế xây dựng/hiệu chỉnh bản thiết kế trực tiếp trên máy tính với sự giám sát/kiểm soát/trợ giúp ngày càng nhiều của phần mềm, nhờ đó ta có nh□ nh được bản thiết kế đúng đắn, khoa học, bài bản,...
- Bản vẽ có thể được phân tích tự động để xác định một số thông tin cần thiết
- CAD được dùng để thiết kế chi tiết cơ khí, mạch điện tử, tàu thủy, xe hơi, nhà cửa, quần áo,…
- Một số phần mềm tiêu biểu: OrCAD, AutoCAD,…

9. Ứng dụng CAM (Computer-Aided Manufacture)

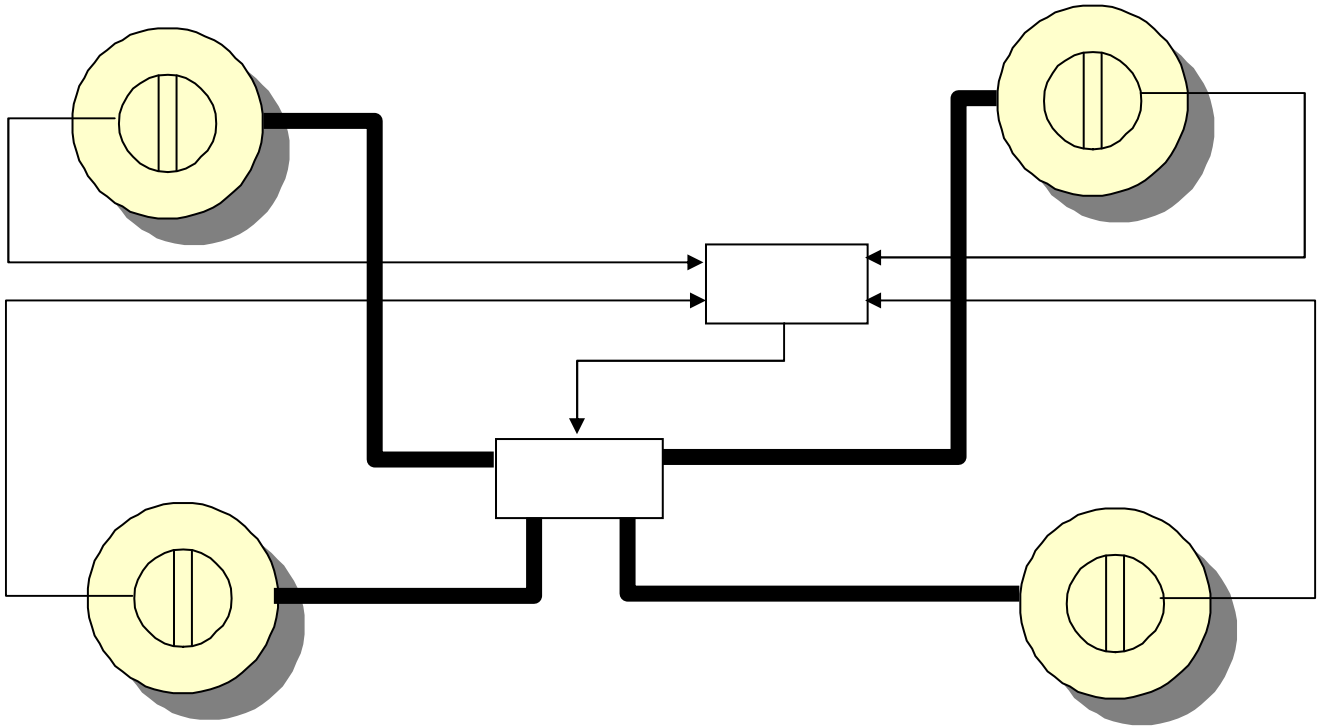
- Phần mềm dùng máy tính hỗ trợ quá trình sản xuất và chế tạo linh kiện/thiết bị.
- Một số dạng CAM như Robot công nghiệp, CNC, CAD/CAM, hệ thống điều khiển...
- Robot công nghiệp thay thế con người trong việc thực hiện 1 công việc cụ thể xác định. Robot có 3 thành phần chính : tay máy, bộ điều khiển và bộ cung cấp năng lượng. Robot có khả năng cảm nhận môi trường xung quanh (nhờ các sensor) và đáp ứng kịp thời với sự th y đổi trạng thái của môi trường.
- CNC (Computer numerical control) được sử dụng rộng rãi trong các thiết bị máy, nó điều khiển hoạt động của máy dựa trên chương trình viết sẵn. Mạch vi xử lý sẽ thu thập các thông tin liên quan (thông qua các mạch D/A), dựa trên thông tin thu thập được, nó điều khiển thiết bị máy thay đổi hành vi theo sự tác động của bên ngoài.

9. Ứng dụng CAM (Computer-Aided Manufacture)

- Kết hợp CAD/CAM : Bản thiết kế được tạo ra nhờ phần mềm CAD sẽ được chuyển thành thông tin/chương trình điều khiển máy CNC chế tạo ra linh kiện/thiết bị tương ứng. Thí dụ AutoCAD cho ta thiết kế chi tiết 1 bulon, rồi dùng thông tin của bản thiết kế bulon để điều khiển "máy in" chế tạo bulon đó. Các phần mềm CAD thường có khả năng kiểm tra sự tương thích của bản thiết kế và đặc tả của "máy in".

9. Ứng dụng CAM (Computer-Aided Manufacture)

- Hệ thống điều khiển sẽ điều khiển hoạt động của một quá trình hoạt động nào đó. Thí dụ hệ thống chống bó cứng má phanh trên các xe ô tô (ABS - Anti-locking Brake System).



10. Ứng dụng nhúng (embedded apllications)

- Là các ứng dụng được cài đặt vào 1 thiết bị cụ thể có trang bị máy tính số với các tài nguyên hạn chế. Nhiệm vụ của ứng dụng nhúng là điều khiển quá trình hoạt động của thiết bị. Thí dụ chương trình điều khiển máy in laser, máy vẽ, máy photocopy, máy giặt,...

11. Mô hình hóa & mô phỏng

- Mô hình hóa được dùng để xem xét, phân tích hoặc lên kế hoạch cho một công việc phức tạp. Mô hình hóa thường dùng các công thức toán học để mô tả một công việc hay một quá trình phức tạp. Ví dụ : $s = 0.5\ at^2$ với s là quảng đường đi được, a là gia tốc và t là thời gian.
- Mô phỏng dùng máy tính để mô hình hóa và xem xét hệ thống : Kiểm tra, An toàn, Dự đoán, Nhanh chóng và linh hoạt
- Thường phần mềm mô hình hóa & mô phỏng được sử dụng nhiều trong các lĩnh vực Giao thông, Xã hội học, Dự đoán thời tiế, Thủy lực học, Sinh học, Giáo dục học, Mô hình 3 chiều.

Chương 7

CÁC VẤN ĐỀ TỔ CHỨC & XÃ HỘI

1. Ảnh hưởng của CNTT đến xã hội

- Đa số người đồng ý rằng máy tính và các công nghệ liên quan đã, đang và sẽ mang lại những biến đổi to lớn trong xã hội con người.
- Tuy nhiên có nhiều ý kiến khác nhau về những biến đổi đó là gì, tốc độ biến đổi ra sao và mức độ lợi ích mà nó mang lại.
- Các lợi ích của CNTT cho xã hội :
  - Tạo ra 1 ngành nghề hoạt động mới.
  - Hỗ trợ đắc lực và hiệu quả nhất cho mọi ngành nghề khác phát triển.
  - Cải thiện chất lượng cuộc sống ngày càng cao.
  - Ảnh hưởng đến sinh hoạt, suy nghĩ của con người.

Tạo ra 1 ngành nghề hoạt động mới

- Ngành CNTT mặc dù còn rất non trẻ so với nhiều ngành nghề khác, nhưng nó thu hút 1 lượng lớn nhân lực trong xã hội làm việc hoặc trên lĩnh vực phần cứng hoặc trên phần mềm, nhờ đó góp 1 phần quan trọng trong việc giải quyết thất nghiệp (tuy nhiên kết quả ứng dụng CNTT có thể làm nhiều người mất việc làm).
- Tỉ lệ tài chính mà CNTT góp vào kinh tế của xã hội là rất cao (nhiều tỉ phú hàng đầu đều làm việc trong lĩnh vực CNTT, như Bill Gate, ...).

2. Ảnh hưởng của CNTT đến tổ chức

- Mô hình hóa được dùng để xem xét, phân tích hoặc lên kế hoạch cho một công việc phức tạp. Mô hình hóa thường dùng các công thức toán học để mô tả một công việc hay một quá trình phức tạp. Ví dụ : $s = 0.5\ at^2$ với s là quảng đường đi được, a là gia tốc và t là thời gian.
- Mô phỏng dùng máy tính để mô hình hóa và xem xét hệ thống : Kiểm tra, An toàn, Dự đoán, Nhanh chóng và linh hoạt
- Thường phần mềm mô hình hóa & mô phỏng được sử dụng nhiều trong các lĩnh vực Giao thông, Xã hội học, Dự đoán thời tiế, Thủy lực học, Sinh học, Giáo dục học, Mô hình 3 chiều.