CHƯƠNG 0

# Hệ Thống Máy Tính & Lập Trình Hệ Thống

## * Tổng quan về hệ thống máy tính

Kiến trúc máy tính gồm 2 phần chính:

+ HSA (Kiến trúc hệ thống phần cứng)

+ ISA (Kiến trúc bộ lệnh)

## * Các thành phần chính của hệ thống máy tính:

+ HSA (Kiến trúc hệ thống phần cứng)

Để có thể hoạt động, bắt buộc phải có:

- *- Hệ thống xử lý*
- *- Hệ thống lưu trữ*
- *- Hệ thống nhập xuất*

## Hệ thống lưu trữ

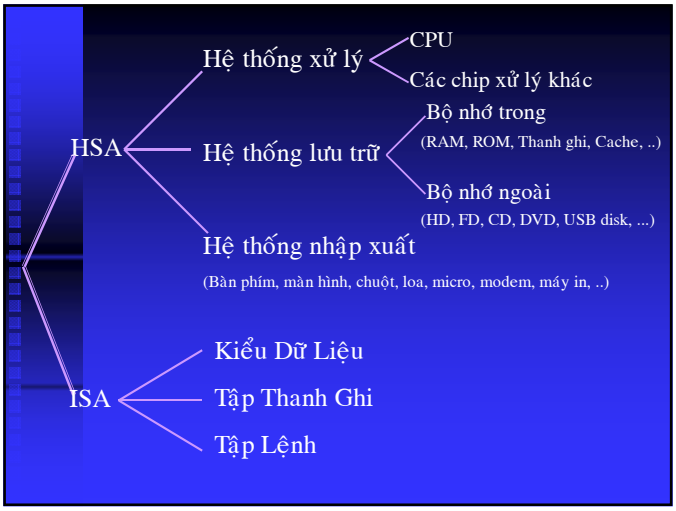
- *Bộ nhớ trong*
  *(RAM, ROM, Registers, Cache,...)*
- *Bộ nhớ ngoài*
  *(HD, FD, CD, DVD, USB disk, ...)*

## Các thành phần chính của hệ thống máy tính:

+ HSA (Kiến trúc hệ thống phần cứng)

+ ISA (Kiến trúc bộ lệnh)

- *Kiểu dữ liệu*
- *Tập thanh ghi*
- *Tập lệnh*




## Các hệ thống số

- **Hệ nhị phân (cơ số 2):**
  Hình thức thể hiện: 01001b, 11111001b, 11100b..
- **Hệ thập phân (cơ số 10):**
  Hình thức thể hiện: 9, 249, 28d, ..
- **Hệ thập lục phân (cơ số 16):**
  Hình thức thể hiện: 9h, 0F9h, 1Ch, ..

## Các kiểu dữ liệu trong máy tính

- **bit**: đơn vị lưu trữ nhỏ nhất
- **Byte**: đơn vị truy xuất của chương trình
- **Word**: đơn vị truy xuất của máy tính
  (có kích thước phụ thuộc vào CPU & lưu ngược theo đơn vị Byte – xem ví dụ)
- **Chuỗi ký tự**: lưu trữ theo thứ tự bình thường
- **Số BCD**: lưu trữ mỗi chữ số của 1 số thập phân bằng một (hoặc nửa) Byte

## Tổ chức dữ liệu trên bộ nhớ trong

- **Byte**: đơn vị truy xuất bộ nhớ trong của phần mềm (gồm 8 bit - bit phải nhất là bit 0 & bit trái nhất là bit 7)
- **Word**: đơn vị truy xuất của phần cứng (có kích thước phụ thuộc vào CPU) hoặc 1 kiểu dữ liệu của phần mềm (có kích thước phụ thuộc vào phần mềm tương ứng)
- **Chuỗi ký tự**: lưu trữ theo thứ tự bình thường
- **Số nguyên**: lưu ngược theo đơn vị Byte (khảo sát các ví dụ cụ thể)

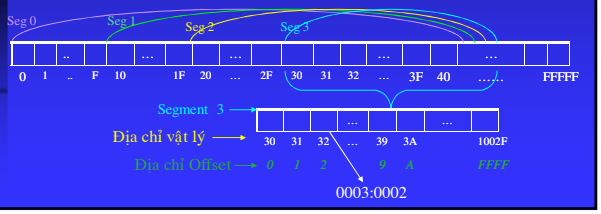
## Tổ chức bộ nhớ chính

- **Bộ nhớ chính**: phần 1 MB RAM đầu tiên (phần còn lại là bộ nhớ mở rộng)
- **Truy xuất bộ nhớ**: thông qua vị trí ô nhớ đầu vùng nhớ (mỗi ô nhớ là 1 Byte)
- **Địa chỉ vật lý**: chỉ số của ô nhớ đánh theo thứ tự tăng dần (1 con số 20 bit có giá trị từ 0 .. FFFFFh)
- **Địa chỉ logic**: gồm 2 con số 16bit (segment & offset) nói lên chỉ số của đoạn chứa ô nhớ & vị trí của ô trong đoạn đó.

## Tổ chức bộ nhớ chính

- **Địa chỉ vật lý**: (1MB = 100000h ô nhớ)

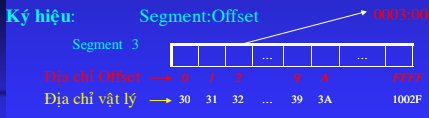

- **Địa chỉ logic**:




## Tổ chức bộ nhớ chính

- **Địa chỉ vật lý**: (1MB = 100000h ô nhớ)


- **Địa chỉ logic**:




**Công thức xác định địa chỉ vật lý từ địa chỉ logic:**

**Địa chỉ vật lý** = Segment * 10h + Offset

## * Tổ chức thanh ghi:

**+ Khái niệm thanh ghi:**

- Là các vùng nhớ nhỏ lưu dữ liệu trong các chip xử lý
- Có tốc độ truy xuất rất nhanh & tần suất sử dụng cao

**+ Kích thước thanh ghi:**

Tính theo đơn vị bit – tùy thuộc chức năng của chip

Các thanh ghi trong CPU 8bit có kích thước 8bit, trong CPU 16bit có kích thước 16bit, trong CPU 32bit có kích thước 32bit đồng thời có luôn các thanh ghi của CPU 16 bit

**+ Số lượng thanh ghi:**

Thường rất ít, tùy thuộc mức độ xử lý & thiết kế của chip

**CPU Intel 16 bit có 14 thanh ghi, phân thành 5 nhóm**

## * Tổ chức thanh ghi của CPU 16 bit:

**+ Nhóm thanh ghi đoạn (Segment register):**

- CS (Code Segment): lưu chỉ số của segment chứa CT ngôn ngữ máy.
- DS (Data Segment): lưu chỉ số segment chứa dữ liệu của CT.
- ES (Extra Segment): lưu chỉ số segment chứa dữ liệu bổ sung của CT.
- SS (Stack Segment): lưu chỉ số segment chứa ngăn xếp của CT.

(Trên CPU 32bit có thêm 2 thanh ghi FS, GS có chức năng tương tự như ES)

CT muốn truy xuất 1 vùng nhớ thì phải đưa chỉ số segment của vùng nhớ đó vào một thanh ghi đoạn

## * Tổ chức thanh ghi của CPU 16 bit:

**+ Nhóm thanh ghi con trỏ & chỉ mục (Pointer & Index Reg.)**

- IP (Instruction Pointer): lưu offset của ô nhớ chứa lệnh kế tiếp.
- BP (Base Pointer): lưu offset của ô nhớ cần truy xuất.
- SP (Stack Pointer): lưu offset đỉnh ngăn xếp.
- SI (Source Index): lưu offset vùng nhớ nguồn cần đọc lên.
- DI (Destination Index): lưu offset vùng nhớ đích cần ghi xuống.

**Mỗi thanh ghi trong nhóm này phải đi kèm với 1 thanh ghi trong nhóm thanh ghi đoạn mới biểu thị được vùng nhớ /ô nhớ cần truy xuất.**

## * Tổ chức thanh ghi của CPU 16 bit:

**+ Nhóm thanh ghi đa dụng (General Register)**

- AX (Accumulator Register): lưu các dữ liệu số, giá trị mặc định, …
- BX (Base Register): đóng vai trò chỉ số mảng,, cũng có thể lưu dữ liệu
- CX (Count Register): có thể dùng để định số lần lặp.
- DX (Data Register): lưu dữ liệu /kết quả tính toán (~AX).

**Mỗi thanh ghi trong nhóm này đều cho phép sử dụng đến từng Byte, bằng cách thay chữ 'X' thành chữ 'H' để chỉ Byte cao, hoặc 'L' để chỉ Byte thấp (AH, BL, CL)**

→ **Xem như có thêm 8 thanh ghi mới**

## Tổ chức thanh ghi của CPU 16 bit:

**+ Thanh ghi cờ (Flag Register)**

Không có tên, mỗi bit là 1 cờ, biểu diễn trạng trái của lệnh vừa thực hiện, hoặc đặt trạng trái cho lệnh thực hiện

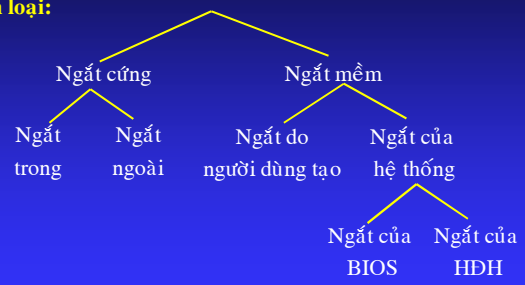
## Hệ thống ngắt (Interrupt):

**+ Khái niệm:**

- Là các chương trình con (thủ tục /hàm) có sẵn trong máy
- Các hàm ngắt không chỉ có sẵn trong ROM BIOS mà còn được Hệ Điều Hành bổ sung thêm
- Hàm ngắt không có tên mà thay vào đó là 1 con số (0..FF)
- Mỗi hàm ngắt lại chứa bên trong nó nhiều hàm con
- Tham số in./output của hàm ngắt được truyền qua thanh ghi

## Hệ thống ngắt (Interrupt):

**+ Phân loại:**




## Hệ thống ngắt (Interrupt):

**+ Chương trình ngắt:**

Thường gồm nhiều đoạn chương trình con bên trong, mỗi đoạn thực hiện 1 công việc cụ thể nào đó, chỉ số của đoạn thể hiện trong AH. Khi đó thân hàm ngắt có dạng (mã giả):

```
switch (AH) {
case 0:
        .. .. ..   // hàm con 0
        break;
case 1:
        .. .. ..   // hàm con 1
        break;
.. .. ..
case N:
        .. .. ..   // hàm con N
        break;
}
```

## Tổ chức dữ liệu trên đĩa từ

**+ Cấu trúc vật lý :**

- Hình tròn, gồm nhiều mặt, mỗi mặt có nhiều đường tròn đồng tâm, trên các đường tròn có các cung tròn, thông thường mỗi cung chứa 4096 điểm từ (=4096bit = 512 byte)
- Mỗi mặt có tương ứng 1 đầu đọc để đọc hoặc ghi dữ liệu.
- Mỗi lần đọc /ghi ít nhất 1 cung tròn (512 B)
- Các cung tròn, đường tròn & đầu đọc (hoặc mặt) có các từ gốc tương ứng là sector, track (hoặc cylinder) & head.
- Mỗi lần truy xuất (đọc hoặc ghi đĩa) chỉ có thể thực hiện trên N sector liên tiếp (N>=1)

## Tổ chức dữ liệu trên đĩa từ

**+ Cấu trúc vật lý :**

- Để truy xuất 1 sector cần phải chỉ ra vị trí của sector đó. Vị trí sector được thể hiện bằng 3 thông số: chỉ số sector, track & head
- Head được đánh số theo thứ tự từ trên xuống bắt đầu từ 0, Track được đánh số theo thứ tự từ ngoài vào bắt đầu từ 0, Sector được đánh chỉ số theo thứ tự bắt đầu từ 1 theo chiều ngược với chiều quay của đĩa.
- Địa chỉ của sector vật lý có ký hiệu : (sector, track, head)

## Tổ chức dữ liệu trên đĩa từ

**+ Tổ chức đĩa logic:**


- Là 1 dãy các sector được đánh chỉ số theo thứ tự tăng dần bắt đầu từ 0
- Đĩa thật sự là đĩa vật lý nhưng vì truy xuất phải dùng đến 3 tham số rất bất tiện nên khái niệm đĩa logic được đưa ra để dễ hiểu, dễ thao tác /tính toán hơn.
- Mỗi sector trên đĩa logic tương ứng với 1 sector duy nhất trên đĩa vật lý sao cho sau khi truy xuất sector K thì truy xuất tiếp sang sector K+1 là nhanh hơn so với tất cả các sector khác.

## Tổ chức dữ liệu trên đĩa từ

**+ Các thông số trên đĩa mềm 1.44 MB:**

- Đĩa có 2 head /disk, 80 track /head, 18 sector /track
- Dung lượng đĩa:

2 head/disk *80 track/head *18 sector/track =2880 sector/disk

= 0.5 KB/sector * 2880 sector/disk = 1440 KB/disk (~ 1.4MB)

➔ Các sector logic có chỉ số từ 0 đến 2879, và tương ứng với các sector vật lý như sau:

Sector 0 ..17 tương ứng với các sector vật lý (1,0,0) .. (18,0,0)

Sector 18..35 tương ứng với các sector vật lý (1,0,1).. (18,0,1)

Sector 36..53 tương ứng với các sector vật lý (1,1,0).. (18,1,0)

Sector 2879 tương ứng với sector vật lý (18, 79, 1)

## * Lập trình hệ thống

+ Khái niệm:

- Chương trình hệ thống là các CT can thiệp sâu vào hệ thống máy tính hoặc HĐH, là các CT điều khiển trực tiếp các thiết bị hoặc các CT có chức năng như các CT của HĐH

- Ví dụ về 1 số CT hệ thống: các CT trong HĐH (hoặc chính HĐH), CT virus hoặc CT diệt virus, CT cứu dữ liệu, CT điều khiển bàn phím để có thể gõ được 1 ngôn ngữ nào đó mà HĐH hiện tại chưa hỗ trợ (ví dụ như VietKey, UniKey,..)

- Lập trình hệ thống là làm ra các chương trình hệ thống!

## * Lập trình hệ thống

+ Cách truy cập trực tiếp bộ nhớ chính (trong C):

- Lấy địa chỉ segment của 1 đối tượng: dùng macro **FP_SEG**

- Lấy địa chỉ offset của 1 đối tượng: dùng macro **FP_OFF**

- Cho 1 con trỏ trỏ đến 1 địa chỉ logic: dùng macro **MK_FP**

- Cú pháp sử dụng:

```
unsigned  FP_OFF (void far *p);     // p là tên của đối
unsigned  FP_SEG (void far *p);     // tượng cần lấy địa chỉ
void far  * MK_FP (unsigned segment, unsigned offset);
```

- Ví dụ: xem file LuuSo-M.cpp

## * Lập trình hệ thống

+ Cách điều khiển trực tiếp màn hình :
( Chế độ văn bản 80 cột * 25 dòng * 16 màu )

- Mỗi ô trên màn hình được biểu diễn bằng 2 byte trên bộ nhớ

- 2000 ô trên màn hình chiếm 4000 byte trên bộ nhớ chính - từ vị trí B800:0000 đến B800:0F9F (F9Fh = 3999d)

- Ký tự trong ô được lưu ở byte đầu, màu của ký tự lưu ở byte sau (với 4 bit thấp là màu chữ & 4 bit cao là màu nền)

- Ô (x,y) trên màn hình tương ứng với 2 offset k và (k+1), với k = (y-1)*160 + (x-1)*2

## * Lập trình hệ thống

+ Cách điều khiển trực tiếp màn hình : (Chế độ 80*25 * 16)

Ví dụ:

```
// Xuất ký tự 'A' ra ô (x,y) trên màn hình:
        unsigned char far * p;
        p = (unsigned char far *) MK_FP (0xB800, 0);
        p [ (y-1)*160 + (x-1)*2 ] = 'A';
// đổi màu của ký tự trong ô (x,y) thành nền xanh chữ trắng:
  p [ (y-1)*160 + (x-1)*2 + 1 ] = 0x1F; // 1: màu xanh, F: màu trắng
// chuyển ký tự trong ô (x,y) sang ký tự kế tiếp:
  p [ (y-1)*160 + (x-1)*2 ] ++;
```

## Lập trình hệ thống

+ Cách điều khiển trực tiếp màn hình : (Chế độ 80*25 * 16)

Ví dụ:

```
// Xuất ký tự 'A' ra ô (x,y) với nền xanh chữ trắng:
    unsigned char far * p;
    p = (unsigned char far *) MK_FP (0xB800, 0);
    p [ (y-1)*160 + (x-1)*2 ] = 'A';
// đổi màu của ký tự trong ô (x,y) thành nền xanh chữ trắng:
  p [ (y-1)*160 + (x-1)*2 + 1 ] = 0x1F; // 1: màu xanh, F: màu trắng
// chuyển ký tự trong ô (x,y) sang ký tự kế tiếp:
  p [ (y-1)*160 + (x-1)*2 + 1 ] ++;
```

## Lập trình hệ thống

+ Lập trình sử dụng ngắt trong C:

Dùng hàm int86/ int86x, geninterrupt hoặc chèn mã hợp ngữ

Ví dụ: // Xuất ký tự 'A' ra màn hình

```
    _AH = 2;
    _DL = 'A';
    geninterrupt (0x21);
```

// Xem thêm ở file ví dụ Ngat.cpp

## Lập trình hệ thống

+ Cách truy xuất đĩa trực tiếp (trong C):

- Đọc nội dung sector trên đĩa logic: dùng hàm **absread**
- Ghi nội dung vào sector của đĩa logic: dùng hàm **abswrite**
- Thực hiện các thao tác trên đĩa vật lý: dùng hàm **biosdisk**
- Cú pháp sử dụng:

```
int absread (int drive, int nsects, long lsect, void *buffer);
int abswrite (int drive, int nsects, long lsect, void *buffer);
int biosdisk  (int cmd, int drive, int head, int track, int sector,
               int nsects, void *buffer);
```

- Ví dụ: xem file Sector.cpp