

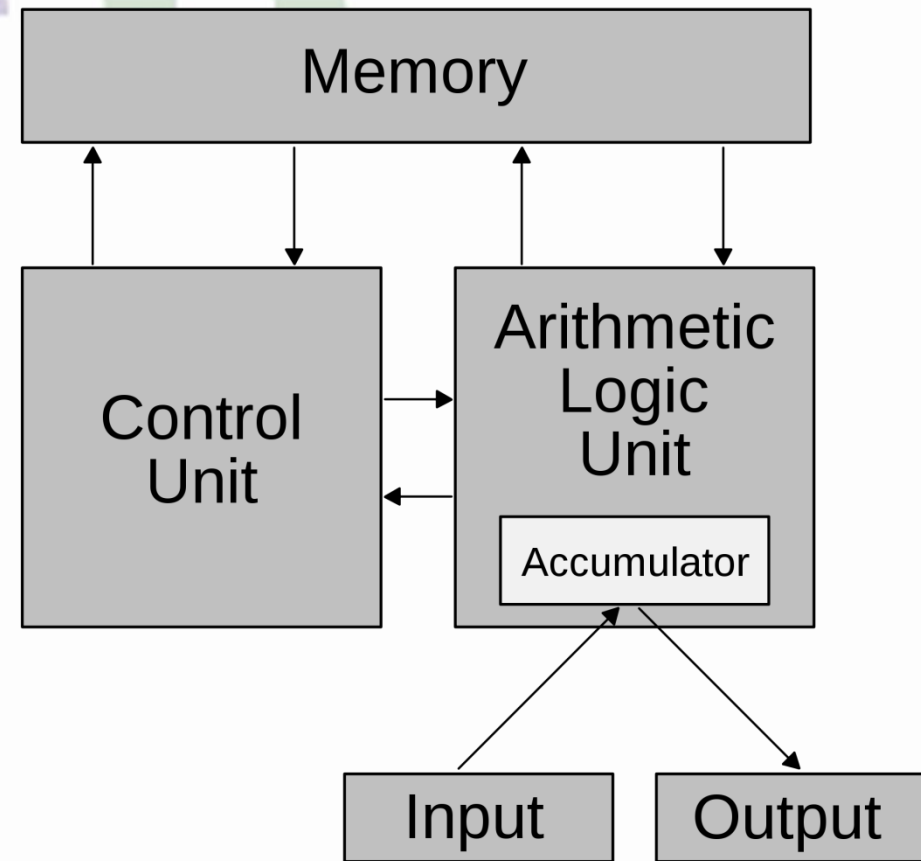
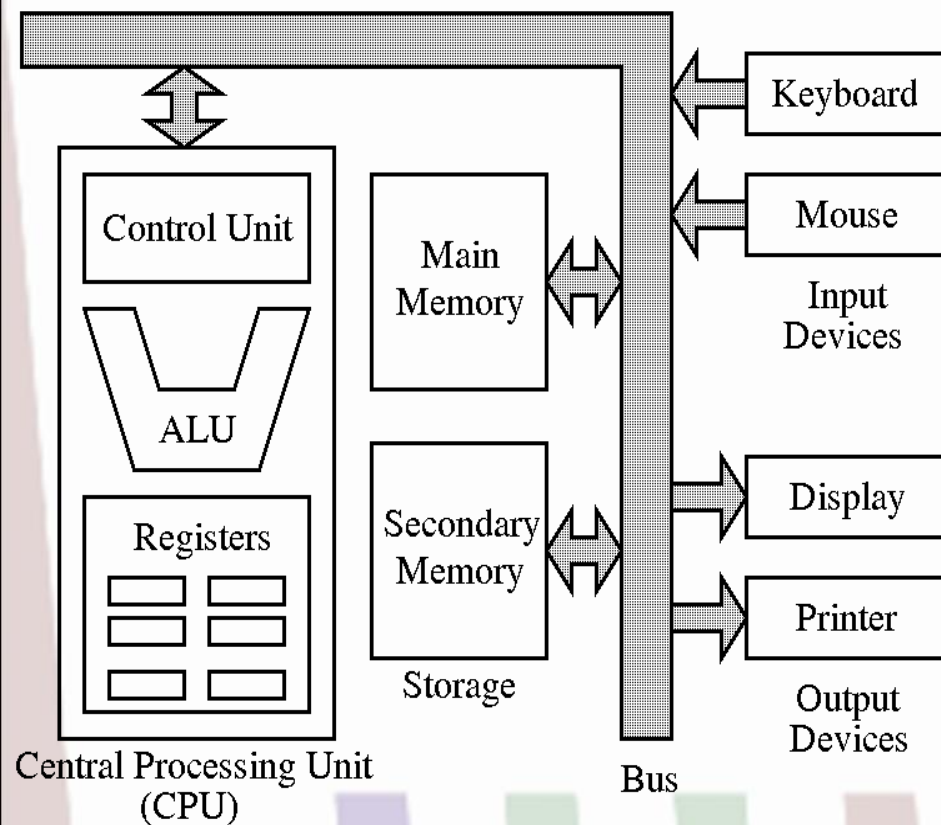
LẬP TRÌNH HỆ THỐNG.

- 1. Tổ chức máy tính.**
- 2. Ngắt (Interrupt).**
- 3. Hàm API (Application Programming Interface).**

1. Tổ chức máy tính.

1.1. Tổng quát.

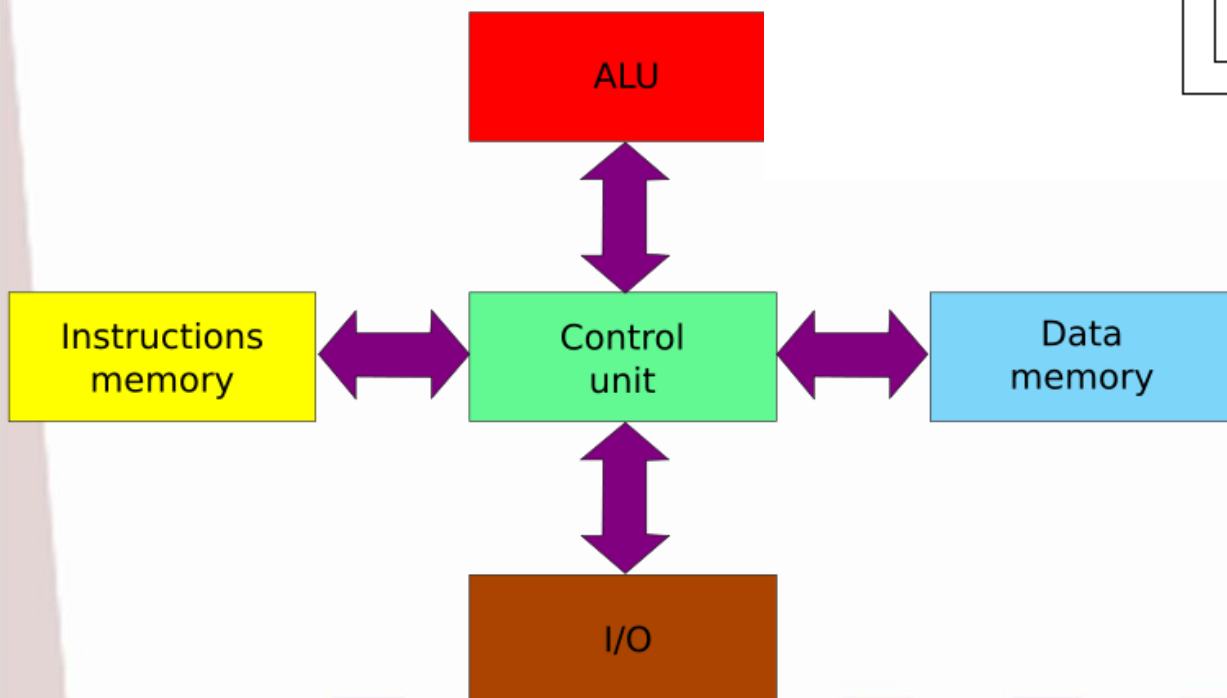
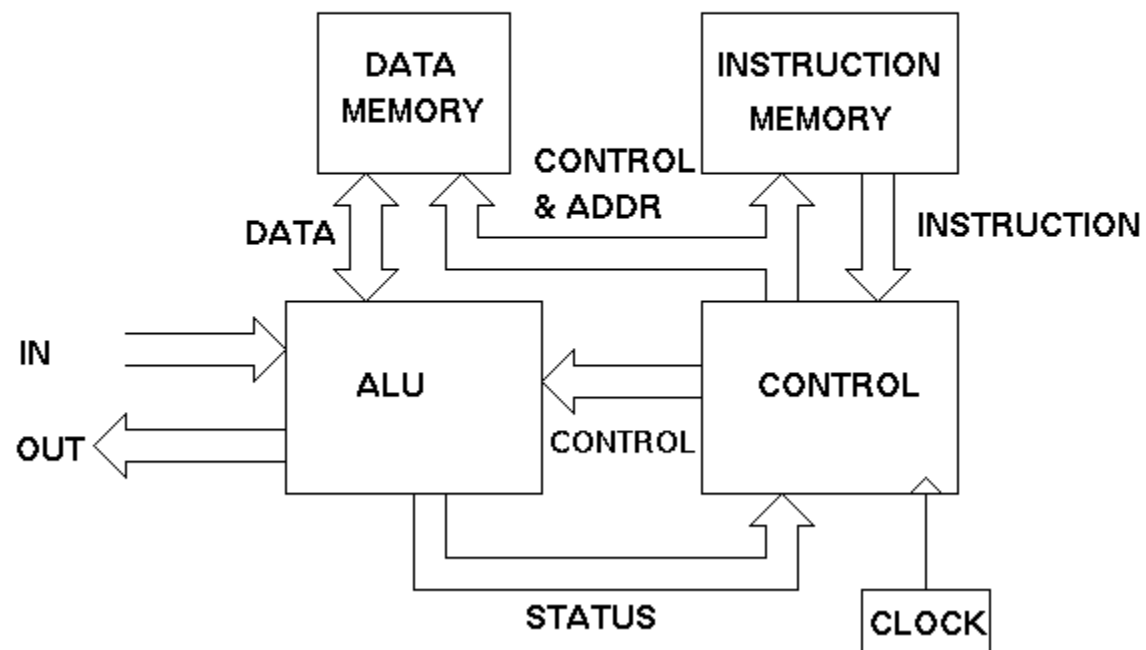
Kiến trúc Von Neumann.



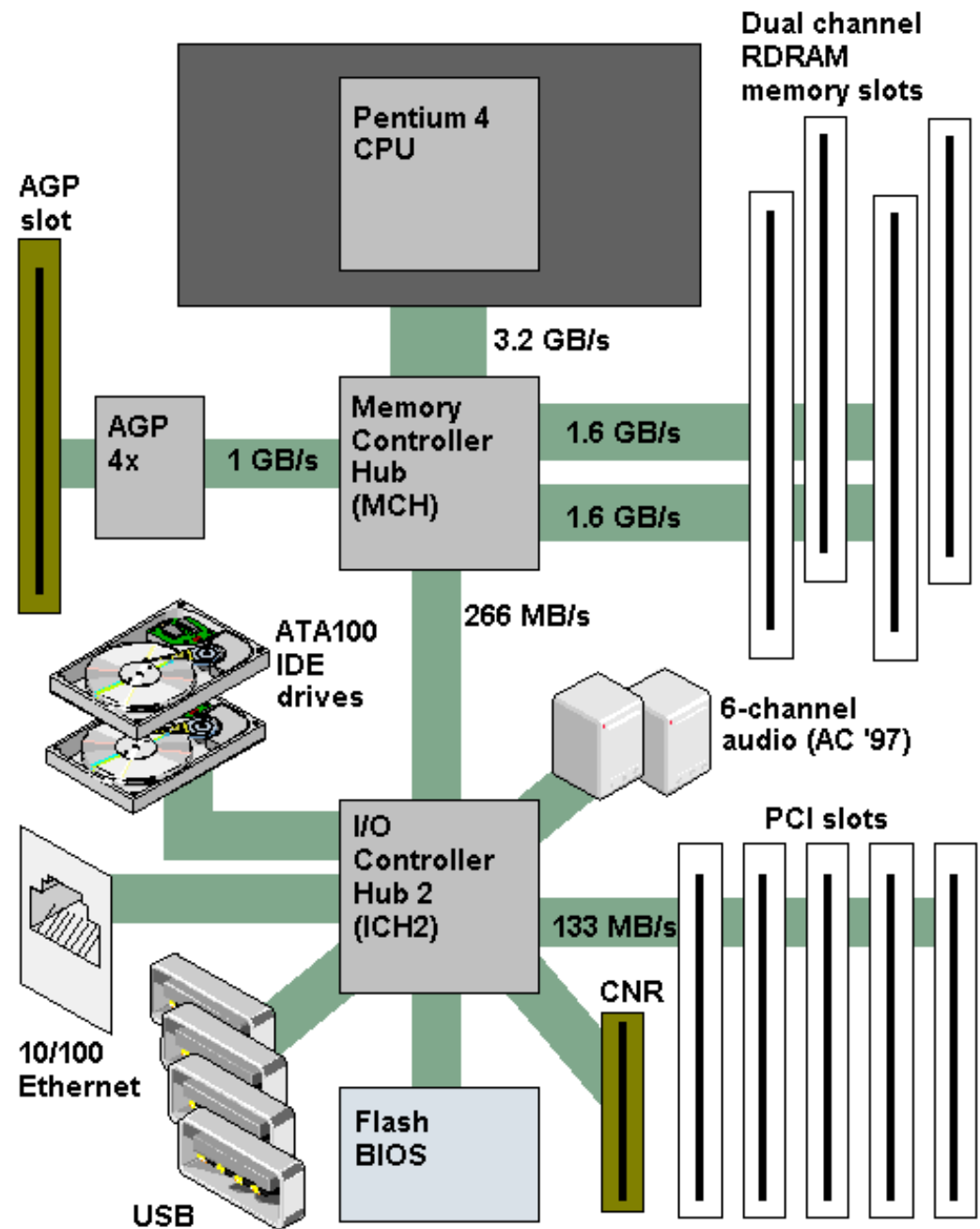
Kiến trúc Harvard.

HARVARD ARCHITECTURE

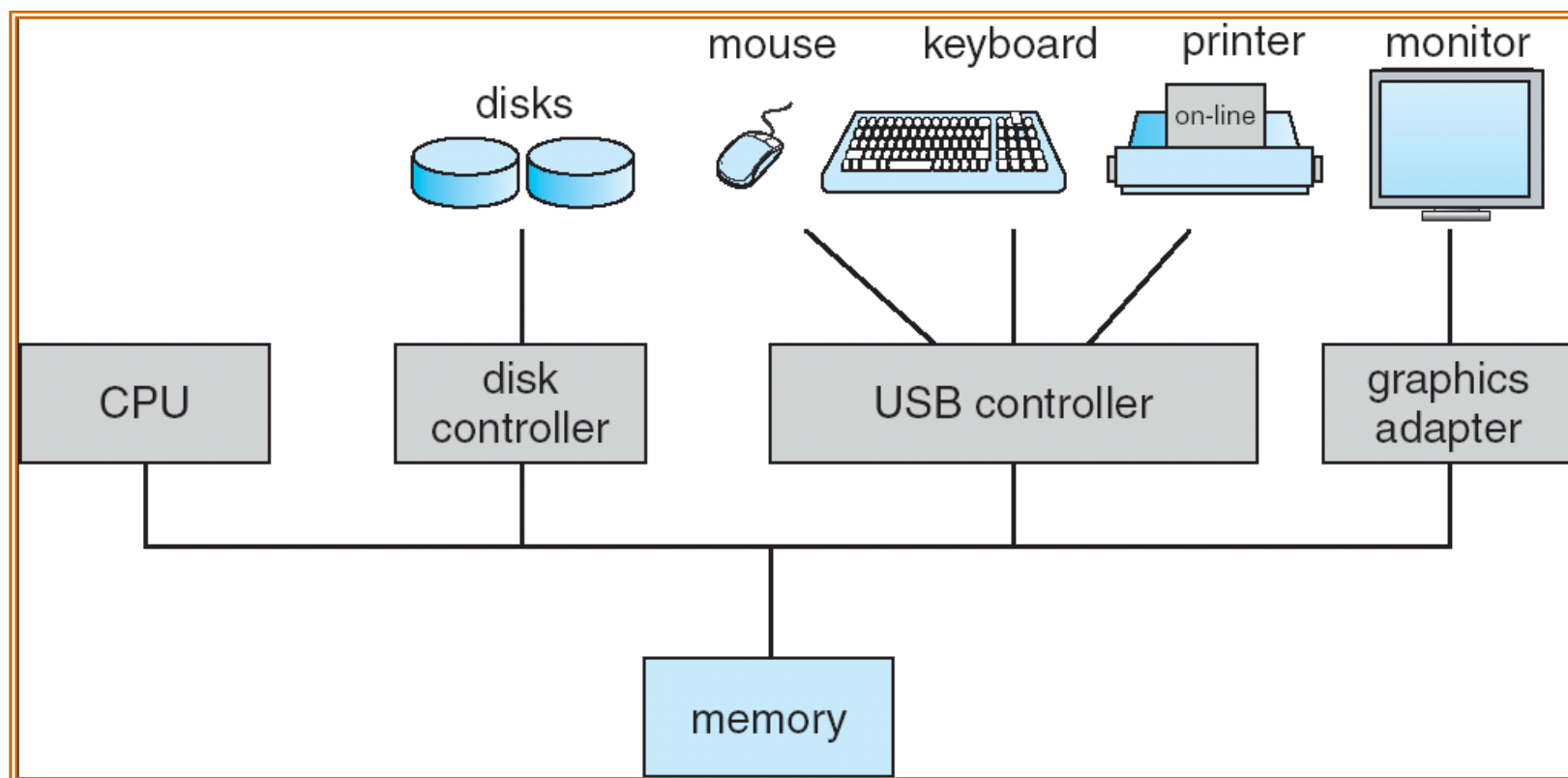
MICROPROCESSOR



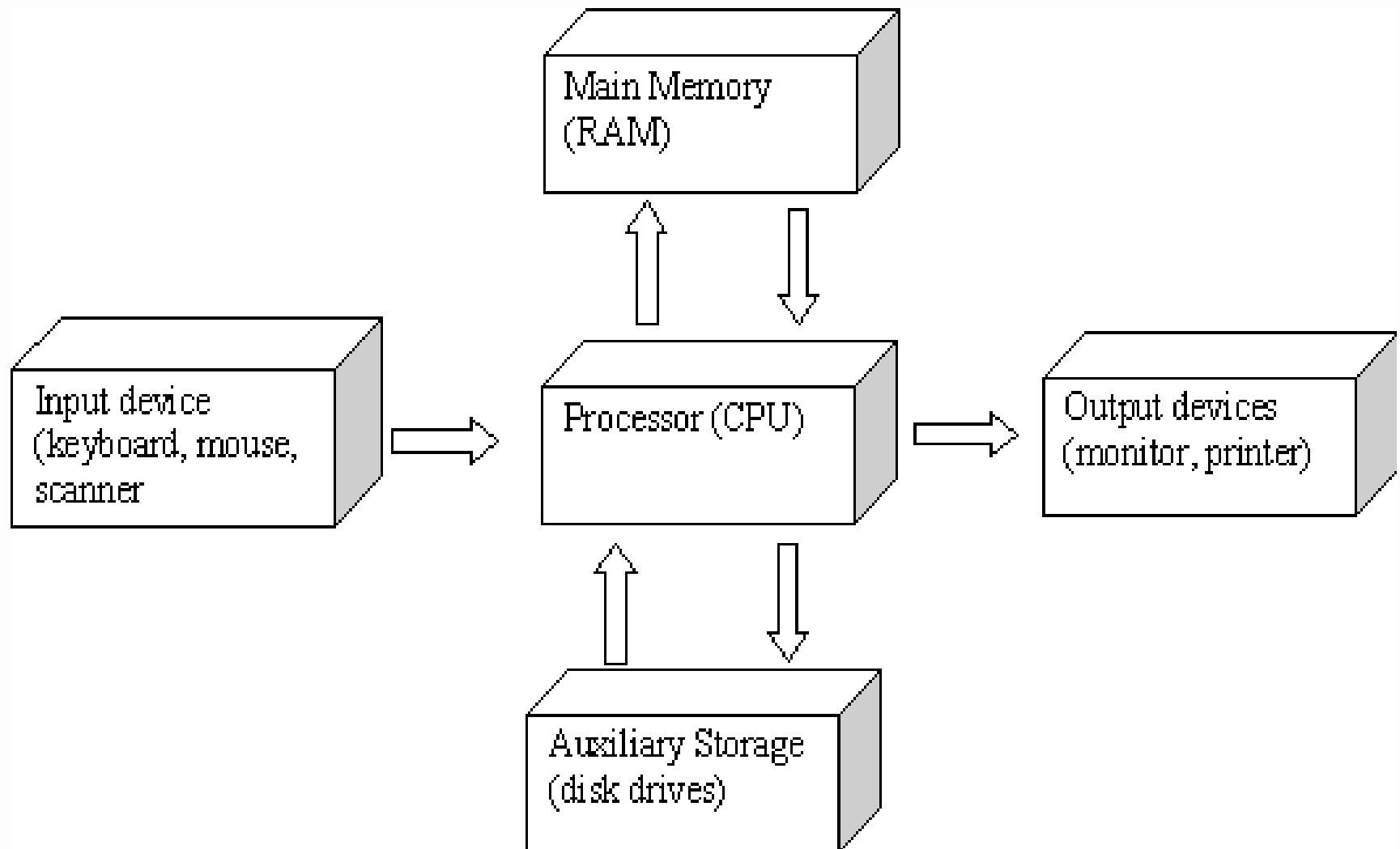
Kiến trúc 1 máy tính cụ thể.



Kiến trúc tổng quát máy PC.



Sơ đồ chức năng của MT



1.2. CPU

Chức năng : lấy lệnh-Giải mã lệnh-Đọc toán hạng-Thi hành lệnh

Cấu trúc : 8086/88 : EU (Execute Unit), BIU (Bus Interface Unit)

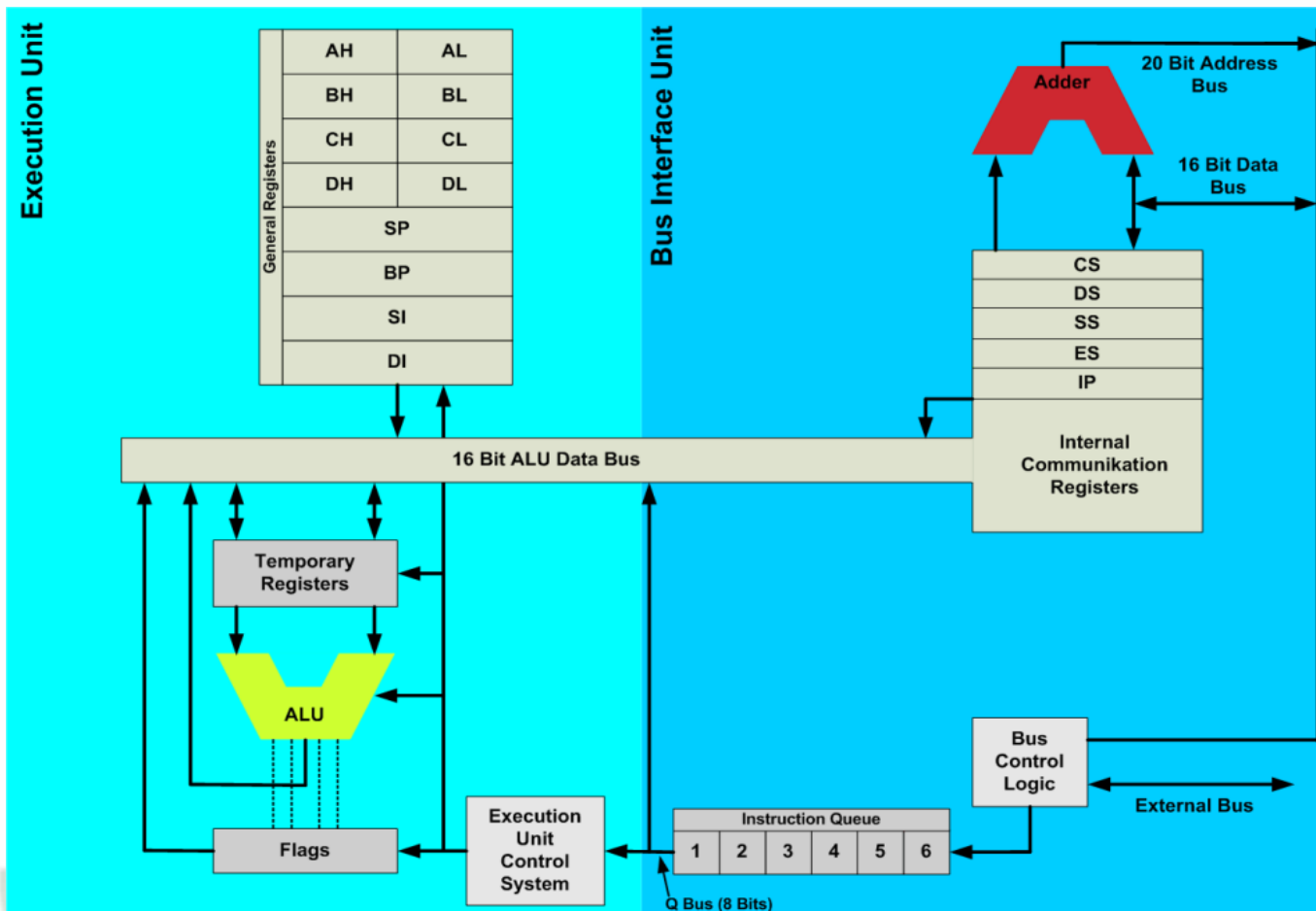
Các thành phần :

Control Unit : lấy lệnh từ bộ nhớ-giải mã lệnh-phân phối lệnh.

ALU (Arithmetic Logic Unit) → FPU

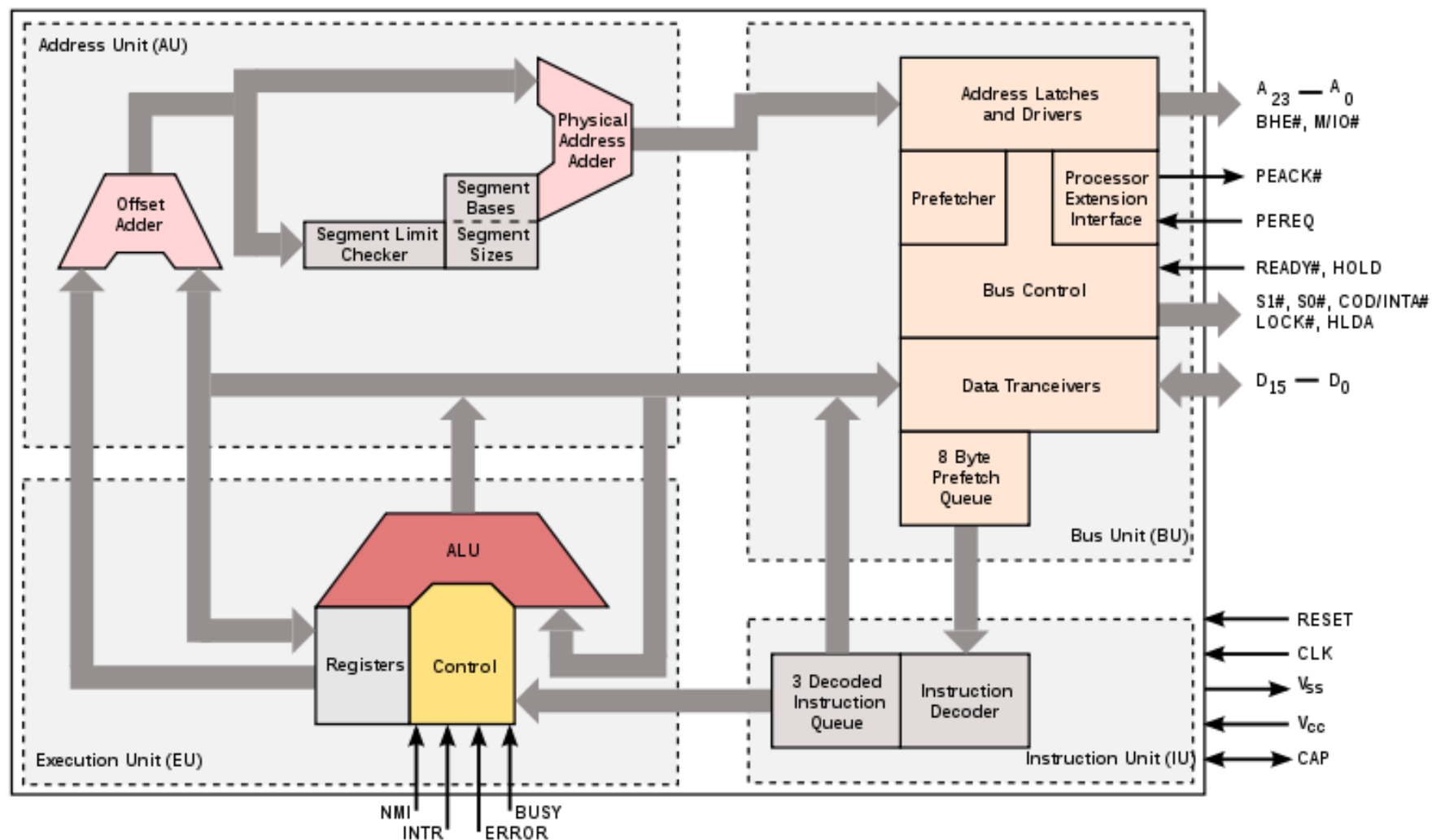
Các thanh ghi

Tập lệnh



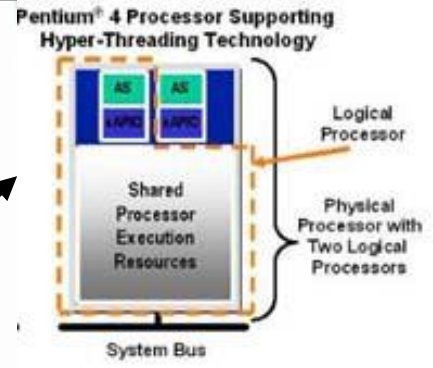
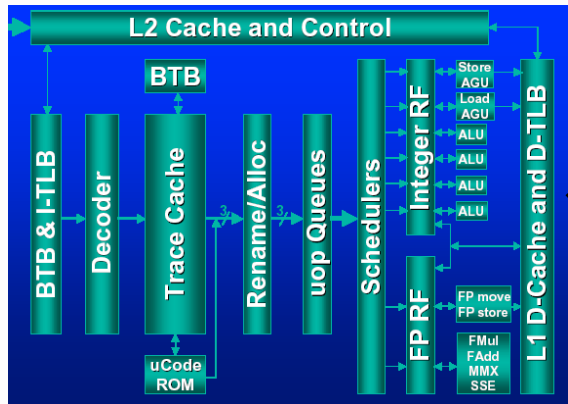
80286 : AU (Address Unit)
EU (Execution Unit)
BU (Bus Unit)
IU (Interface Unit)

Intel 80286 architecture

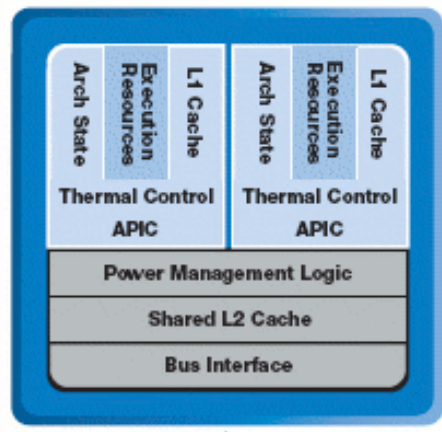


Phát triển KT CPU.

Intel@NetBrush

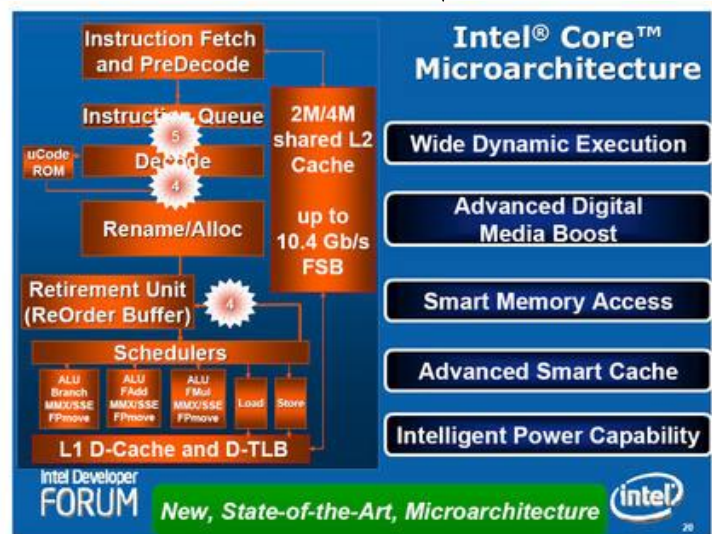


Core Solo



Core Duo

Core Solo



Core 2 Duo

Core 2 Quad

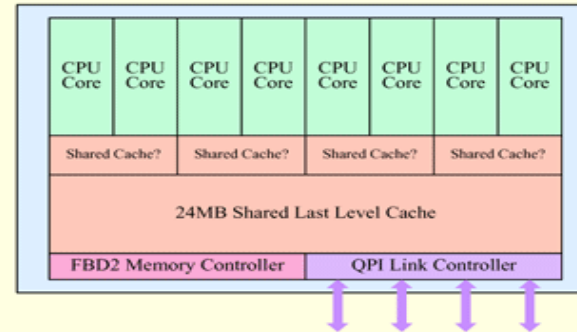
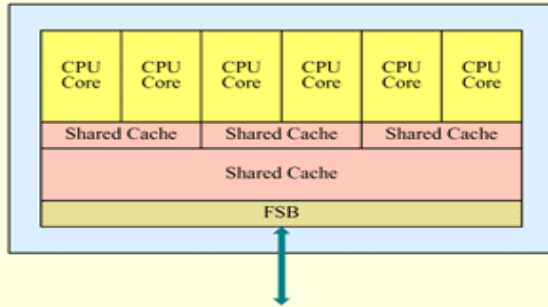
Core 2 Extreme

Nehalem (core i)

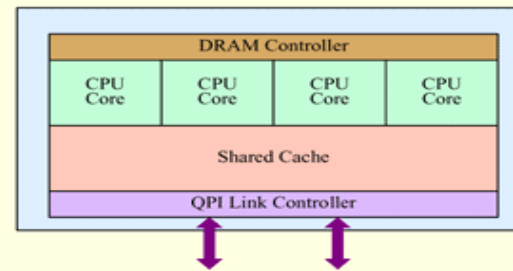
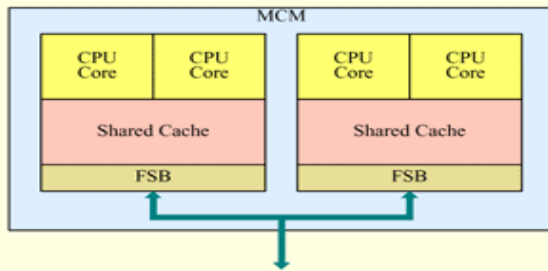
Core Microarchitecture

Nehalem Microarchitecture

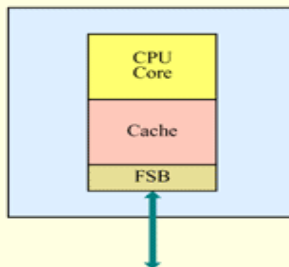
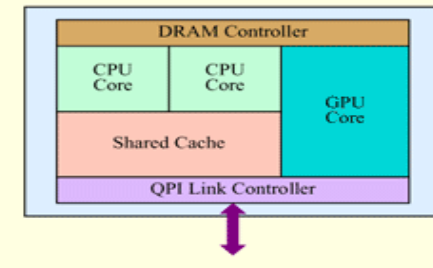
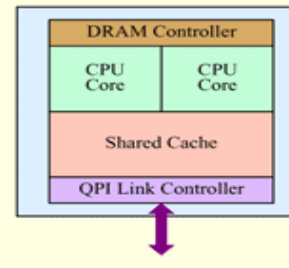
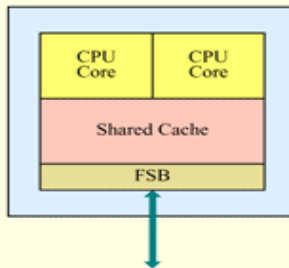
MP Server



DP/UP Server/Performance Desktop



Mainstream & Value Desktop



Các thanh ghi

Tập lệnh ASM16

Main registers

AH	AL	AX (primary accumulator)
BH	BL	BX (base, accumulator)
CH	CL	CX (counter, accumulator)
DH	DL	DX (accumulator, other functions)

Index registers

SI	Source Index
DI	Destination Index
BP	Base Pointer
SP	Stack Pointer

Status register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(bit position)
-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C	Flags

Segment register

CS	Code Segment
DS	Data Segment
ES	Extra Segment
SS	Stack Segment

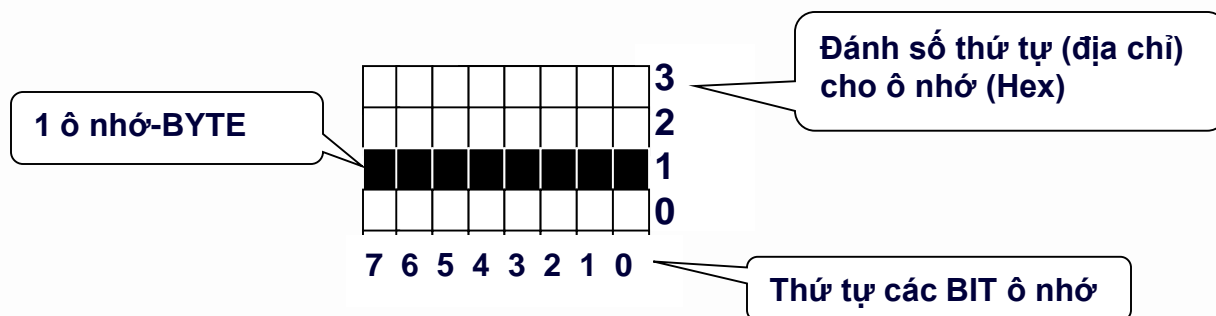
Instruction pointer

IP	Instruction Pointer
----	---------------------

The 8086 registers

1.3. Bộ nhớ

Tổ chức bộ nhớ vật lý : ô nhớ-địa chỉ ô nhớ-phân loại (ROM/RAM)



Tổ chức bộ nhớ của HDH MS DOS

Kiểu data : Byte, Word, Dword

Phân bố :

FFFF:FFFF F000:0000	ROM(thường trú)
EFFF:FFFF C000:000	ROM thiết bị
BFFF:FFFF A000:0000	Bộ nhớ màn hình
9FFF:FFFF 0000:0600	COMMAND.COM (thường trú) CT USER ...
0000:05FF 0000:0400	THAM SỐ ROM BIOS
0000:03FF 0000:0000	INTERRUPT VECTOR

Một số vùng đ/c cần lưu ý :

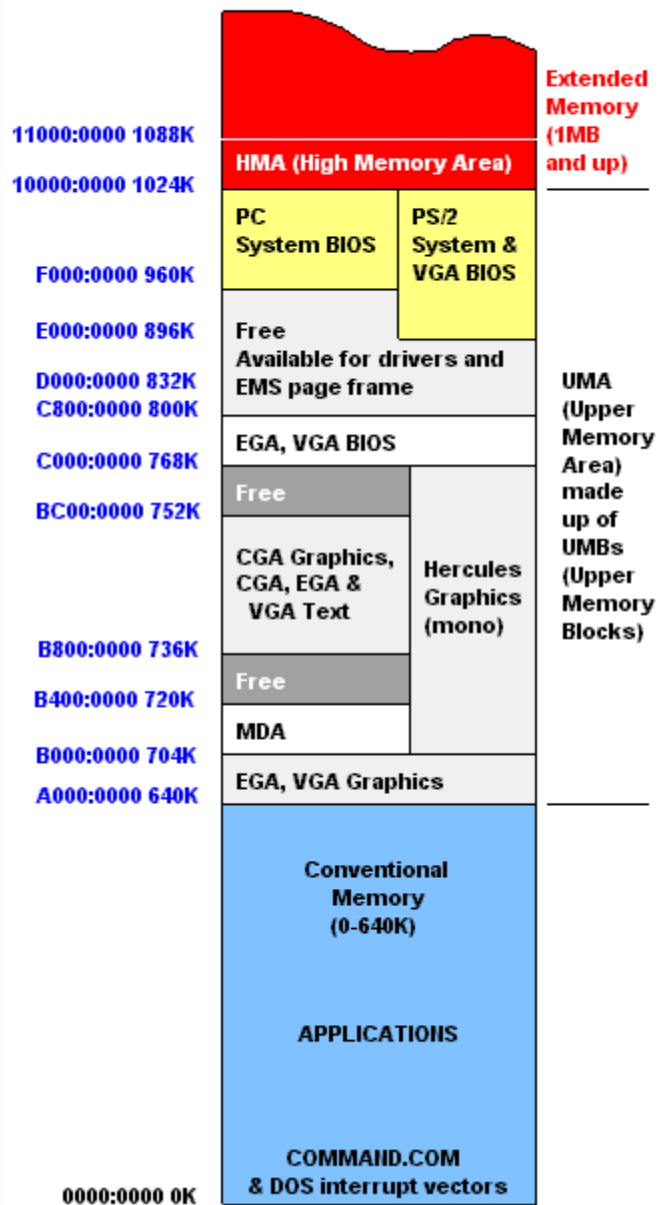
Interrupt Vector : 0000:0000-0000:03FF-256 Interrupt

Vùng tham số ROM BIOS : 0000:0400-0000:05FF

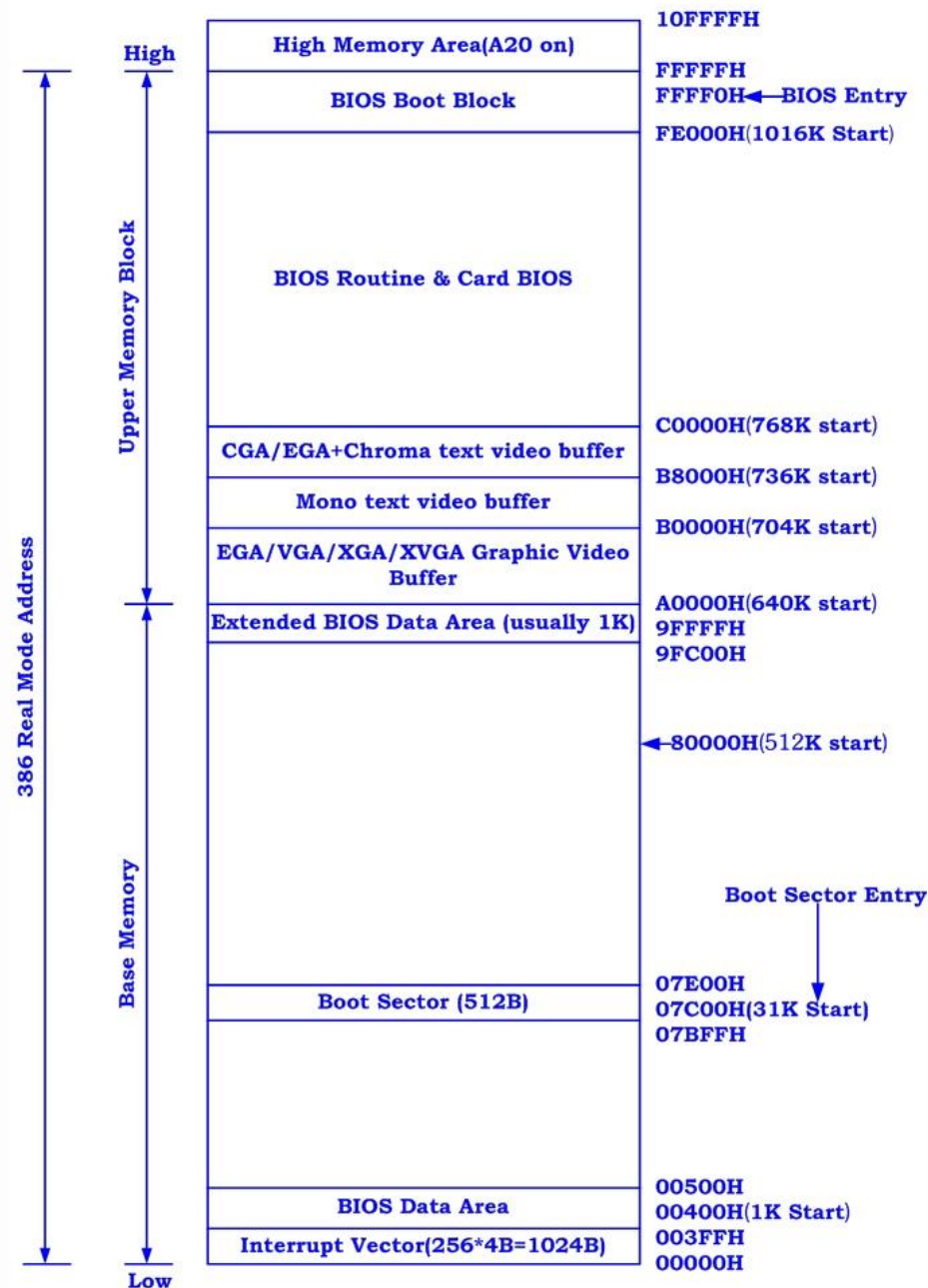
Danh sách thiết bị (2 Byte) : 0000:0410

Dung lượng bộ nhớ hệ thống : 0000:0413

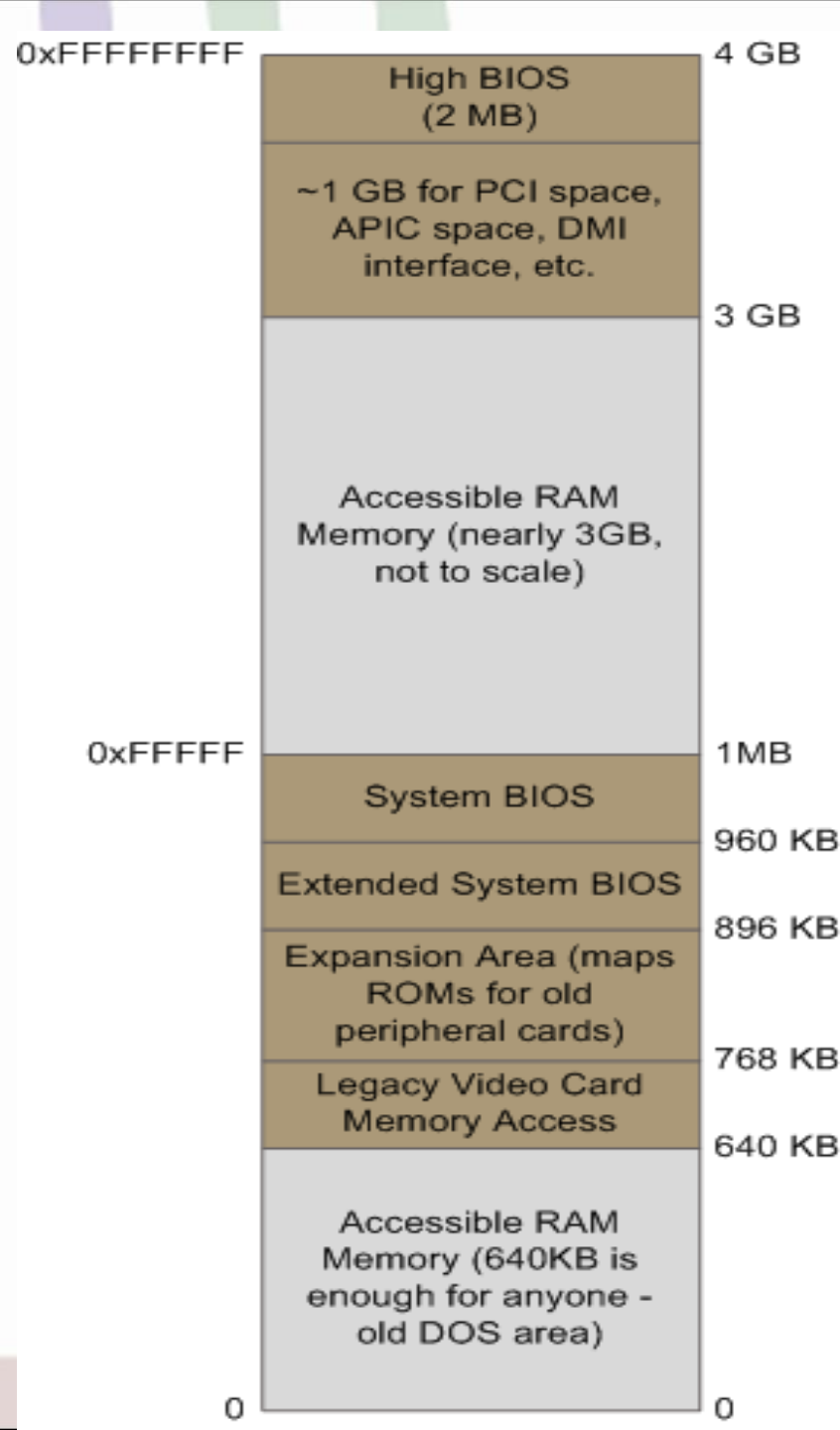
Nạp Boot sector : 0000:7C00



PC BIOS Memory Layout



Tổ chức bộ nhớ của HDH WINDOWS



Truy xuất bộ nhớ (đọc nội dung)

Truy xuất bộ nhớ màn hình VGA trong MS DOS

DS:BX → B800:0000 (LOOP, CX=số lần lặp)

DS:SI → B800:0000

ES:DI → BUFF (MOVESB, MOVSW, CX=số Byte/Word)

1.4. Thiết bị lưu trữ

Đĩa mềm

Tổ chức đĩa mềm : Head, Track, Sector

Tham số vật lý đĩa mềm 1.44MB

512 Byte/Sector

18 Sector/Track

80 Track/Head

2 Head

2880 Sector

Đánh số Sector

ROMBIOS (vật lý) : T(C),H,S; truy xuất đĩa bằng INT 13h

T : Track 0 – tổng số Track-1/Head

H : Head 0 – 1

S : Sector 1 – tổng số Sector/Track

DOS (logic) : 0 – tổng số Sector logic -1; đọc INT 25h, ghi INT 26h

(0,0,1) → 0

...

(0,0,18) → 17

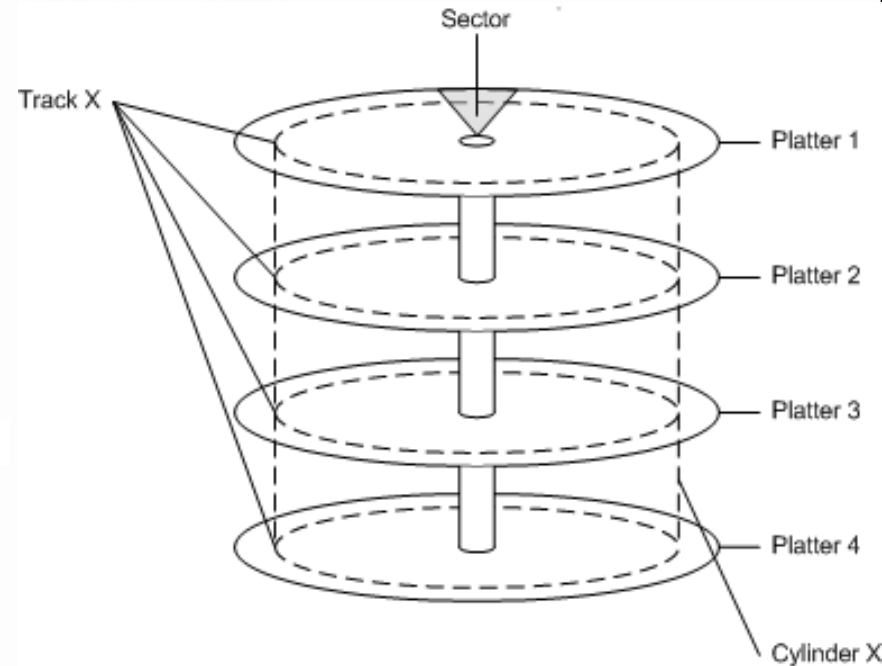
(0,1,1) → 18

...

(0,1,18) → 35

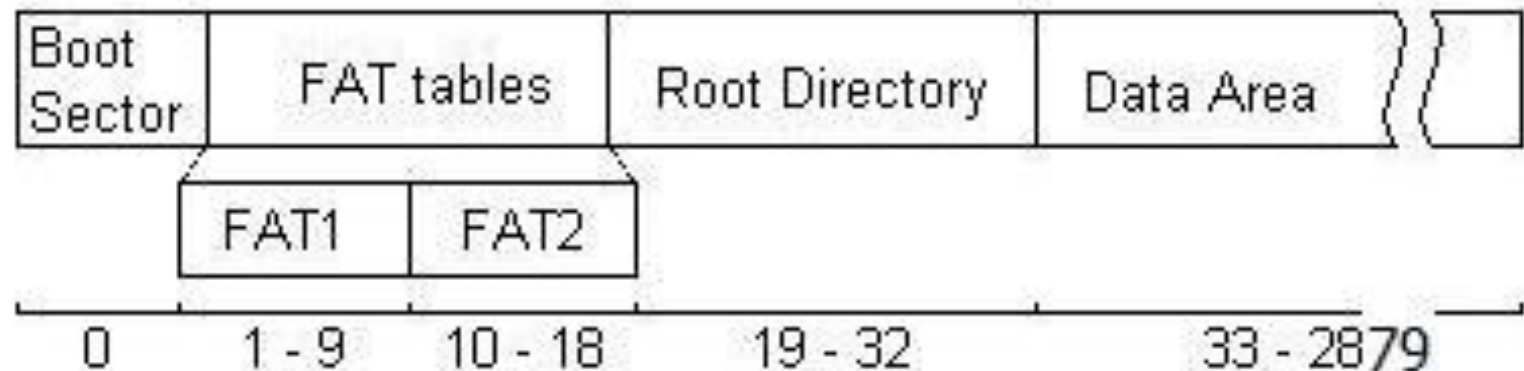
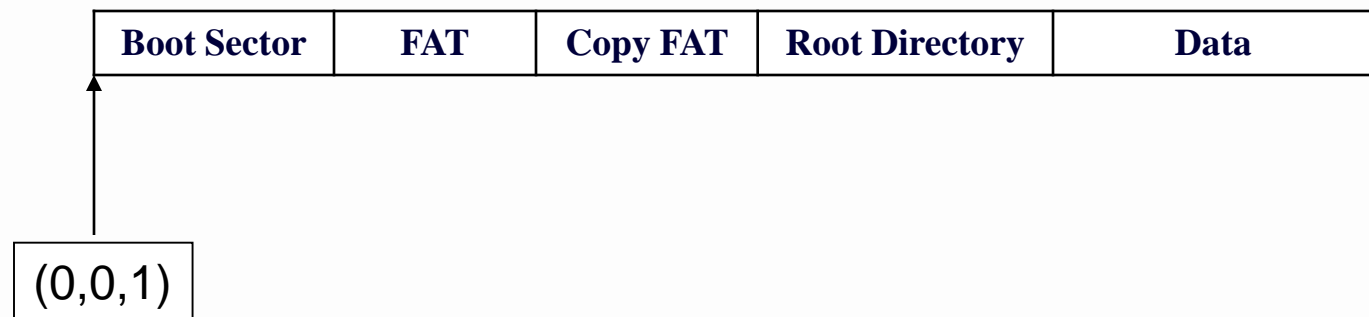
(1,0,1) → 36

...



Chuyển đổi Sector vật lý ↔ logic (tự NC)

Cấu trúc logic của đĩa mềm



Cấu trúc Boot Sector

Bảng BPB

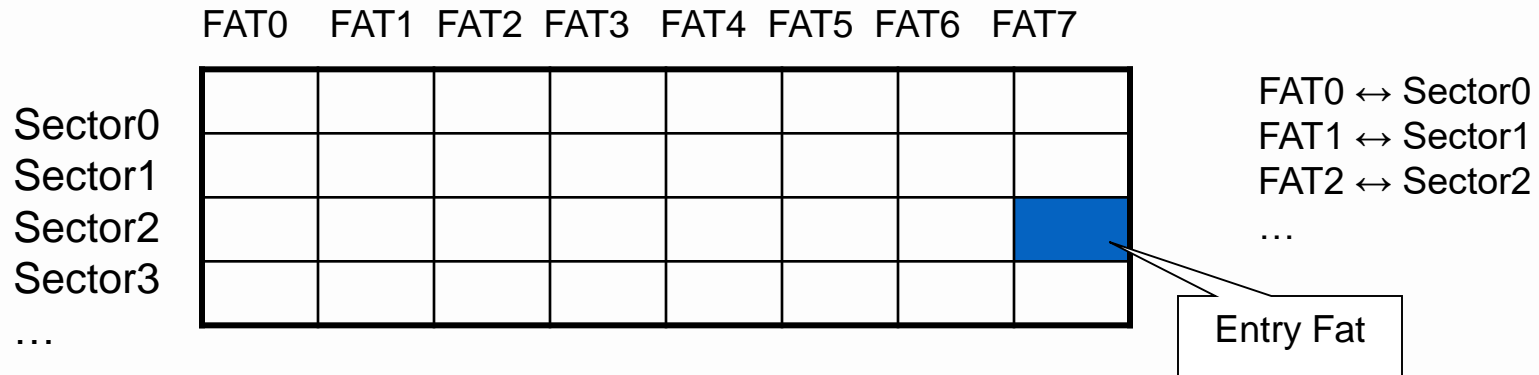
Offset (Hex)	Độ dài (Byte)	ý nghĩa
0	3	JMP đoạn mã Boot (3E)
3	8	Chuỗi tên của HDH (MSDOS5.0, NWIN4.1)
B	2	số Byte/Sector
D	1	số Sector/Cluster (Sector cùng vị trí ở các Head)
E	2	số Sector dành riêng (trước FAT kể cả Boot Sector)
10	1	số bảng FAT
11	2	số Entry trong Root Directory (max)
13	2	tổng số Sector (dung lượng nhỏ)/0 (offset 20h)
15	1	F0h (đĩa mềm)/F8h (đĩa cứng)
16	2	số Sector/FAT
18	2	số Sector/Track
1A	2	số Head/mặt đĩa
1C	4	0/số Sector ẩn (trước Boot Sector)
20	4	tổng số Sector (dung lượng lớn)
24	26	bảng BPB mở rộng
3E	448	đoạn mã Boot
1FE	2	055AAh

Bảng BPB Mở rộng

Offset (Hex)	Độ dài (Byte)	ý nghĩa
24	1	Số hiệu đĩa vật lý (00h – A:; 80h – C:)
25	1	Số hiệu đầu đọc hiện tại (NT)/Không sử dụng
26	1	28h/29h (Win NT)
27	4	Serial number của đĩa
2B	11	Nhãn đĩa
36	8	ID hệ thống file (FAT12/FAT16)

Tổ chức FAT (File Allocation Table)

FAT = 1 vùng đĩa, gồm nhiều Entry Fat



FAT12 = Entry Fat dài 12 bit

FAT16 = Entry Fat dài 16 bit

FAT32 = Entry Fat dài 32 bit

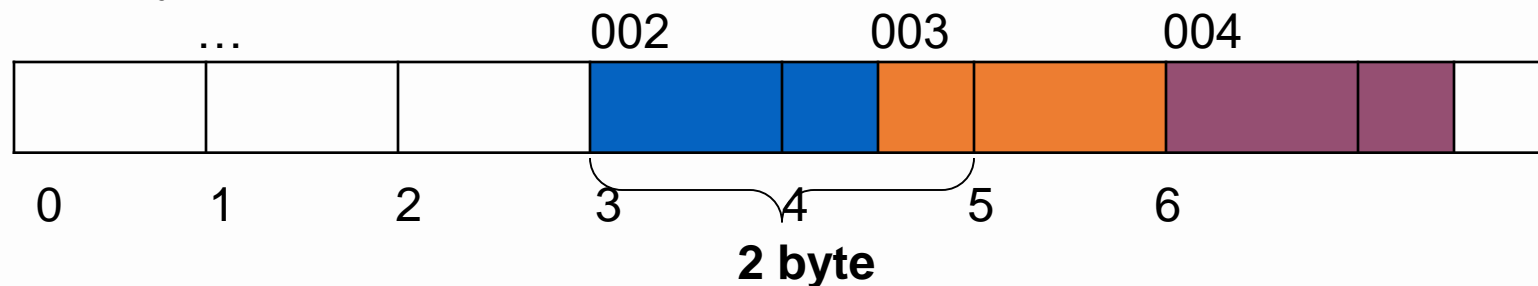
Đọc FAT

Đọc FAT : dựa vào BPB xác định Sector bắt đầu của FAT, độ lớn FAT → Buffer

Nội dung Entry FAT12/FAT16/FAT32

(0)000	Sector còn trống
(0)002 – (F)FEF	Số hiệu Sector tiếp theo của FILE(Sector đang chứa Data)
(F)FF0 – (F)FF6	Không sử dụng
(F)FF7	Sector hỏng
(F)FF8 – (F)FFF	Kết thúc tập tin.

Đọc Entry FAT12 : $\text{off} = \text{số hiệu FAT12} * 3 \text{ div } 2$



Đọc Entry FAT16 : $\text{off} = \text{số hiệu FAT16} * 2$

Đọc Entry FAT32 : $\text{off} = \text{số hiệu FAT32} * 4$

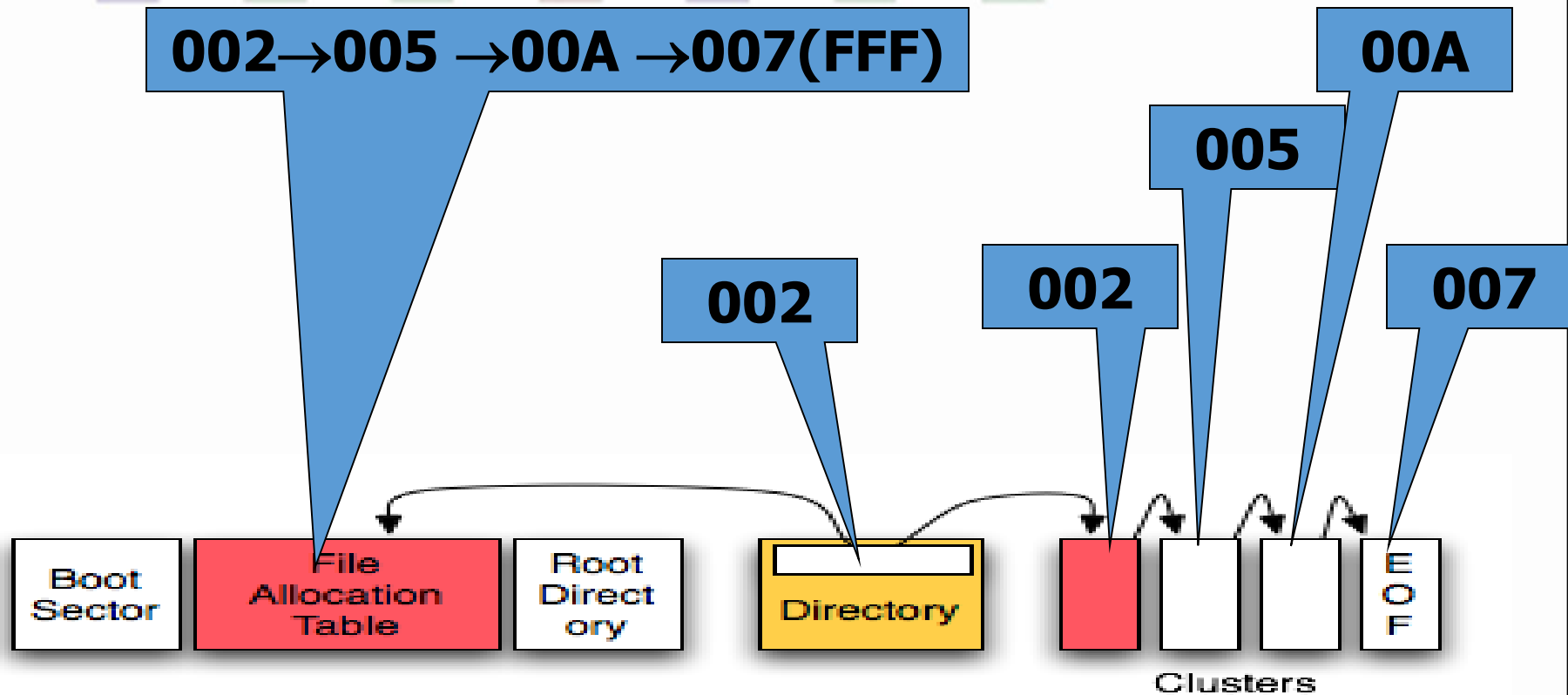
Tổ chức Root Directory (RD)

RD = 1 vùng đĩa, gồm nhiều Root Directory Entry (RDE)
mỗi RDE dài 32 byte.

Đọc RD : ...

Cấu trúc 1 RDE

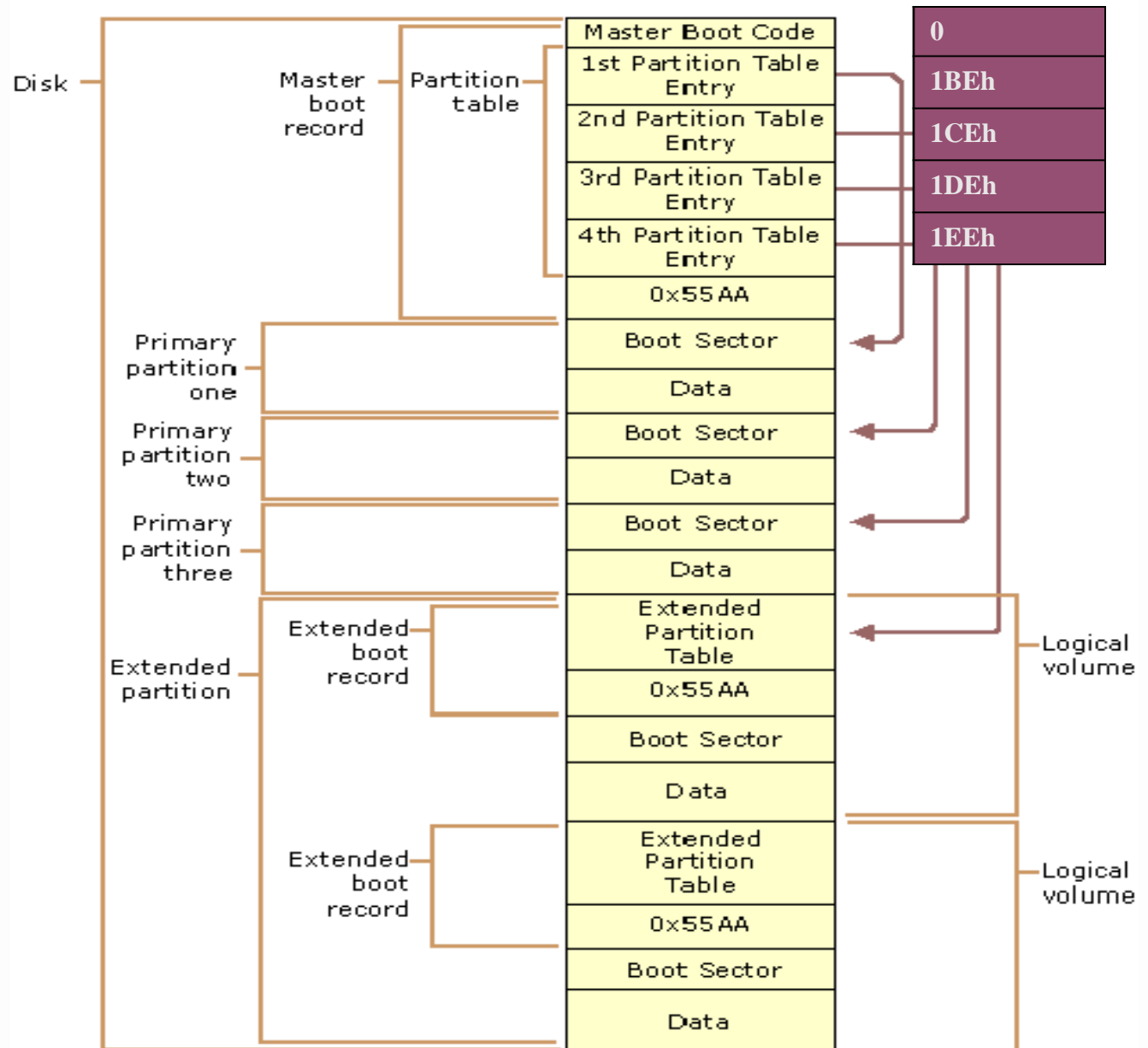
Offset (Hex)	Độ dài (Byte)	ý nghĩa
0	8	Tên File
8	3	Mở rộng File
B	1	Byte thuộc tính <div><div>0</div><div>0</div><div>A</div><div>D</div><div>V</div><div>S</div><div>H</div><div>R</div></div>
C	10	Chưa sử dụng
16	2	Giờ, phút, giây 0-23, 0-59, 0-29(s/2) <div><div><div>5</div><div></div><div></div><div></div><div>1</div><div>0</div><div></div><div></div><div></div><div></div><div>5</div><div>4</div><div></div><div></div><div></div><div>0</div></div></div>
18	2	Ngày, tháng, năm 1980+, 1-12, 1-31 <div><div><div>5</div><div></div><div></div><div></div><div></div><div></div><div>9</div><div>8</div><div></div><div></div><div>5</div><div>4</div><div></div><div></div><div></div><div>0</div></div></div>
1A	2	Số hiệu Entry FAT bắt đầu (Sector bắt đầu)
1C	4	Kích thước File (theo Byte)



Quản lý đĩa
(FAT)

VD: 002→005 →00A →007(FFF)

Cấu trúc logic của đĩa cứng - MBR Partition



Cấu trúc 1 Partition table Entry

Offset (Hex)	Độ dài (Byte)	ý nghĩa
0	1	0/80h
1	1	Head bắt đầu
2	2	Track(15 – 6)/Sector(5 – 0) bắt đầu
4	1	ID Partition (01h-FAT12, 04h-FAT16, 0Bh-FAT32, ...)
5	1	Head kết thúc
6	2	Track(15 – 6)/Sector(5 – 0) kết thúc
8	4	LBA bắt đầu
C	4	Độ dài Partition (tính theo Sector)

Cấu trúc 1 Partition

Boot Sector		FAT	Copy FAT	Root Directory	Data
-------------	--	-----	----------	----------------	------

Công cụ truy xuất Disk

INT 13h

AH=

00	Reset
02	Read
03	Write

INT 13

AH = 02 / 03

AL = number of sectors to read (1-128 dec.)

CH = track/cylinder number (0-1023 dec., see below)

CL = sector number (1-17 dec.)

DH = head number (0-15 dec.)

DL = drive number (00=A:, 01=2nd floppy, 80h=drive 0, 81h=drive 1)

ES:BX = pointer to buffer

on return:

AH = status

AL = number of sectors read

CF = 0 if successful; = 1 if error

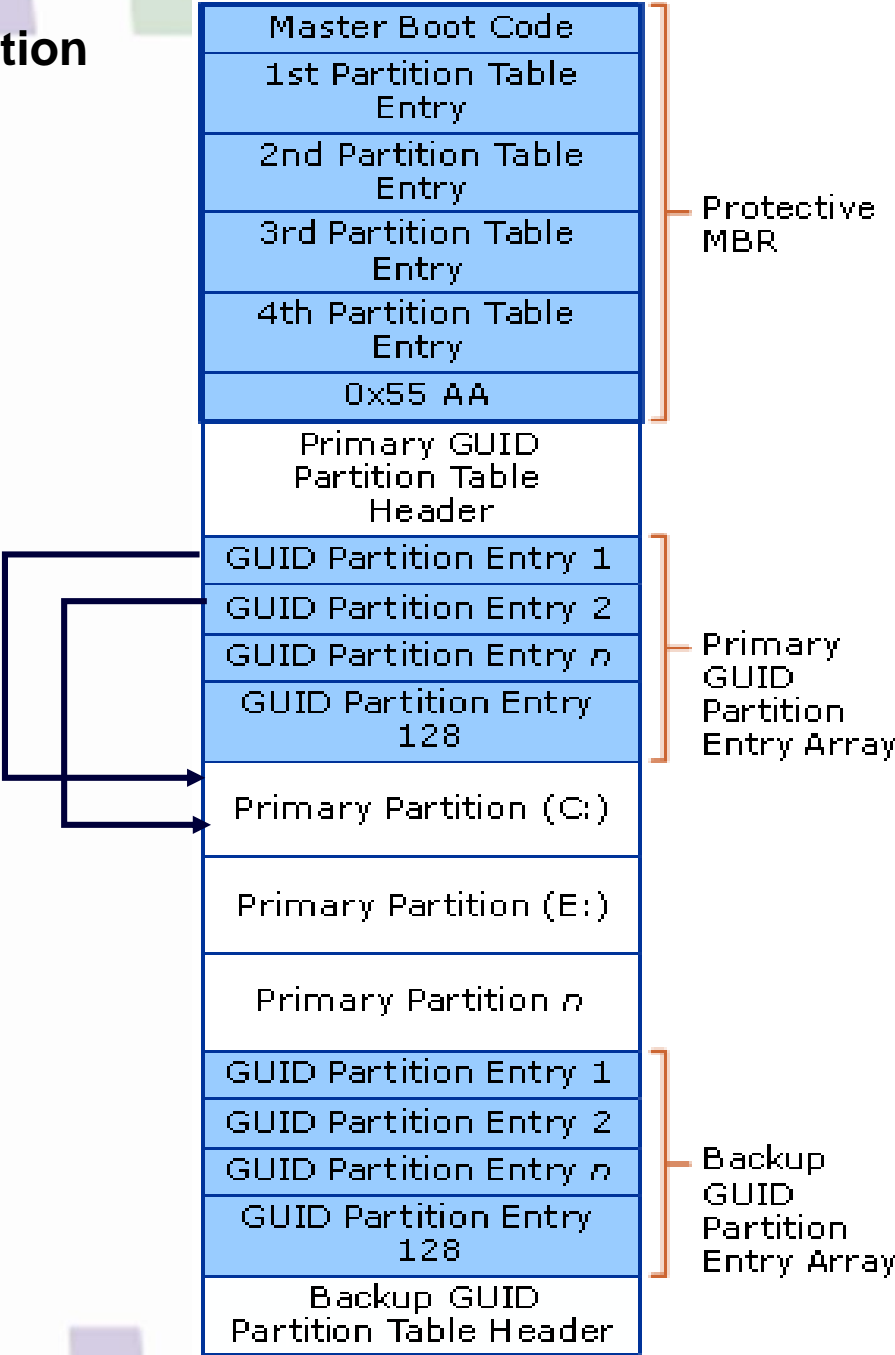
```
CX =      ---CH--- ---CL---  
cylinder : 76543210 98  
sector   :           543210
```

```
CX := ((cylinder and 255) shl 8) or (( cylinder and 768 ) shr 2) or sector;  
cylinder := ( (CX and 0xFF00) shr 8 ) or ( (CX and 0xC0) shl 2)  
sector := CX and 63;
```

INT 25h / 26h

Action:	Provides a direct link into the BIOS to allow data to be read from a specified memory location to disc, starting at a specified logical disc sector into a specified memory location.
Input:	AL = Drive number (0 = A: ; 1 = B: ; 2=C:; etc) CX = number of sectors to read DX = start sector relative (logical) number DS:BX = Buffer
Output:	Carry clear if successful Carry set if failed, AX = Error code, as follows

Cấu trúc logic của đĩa cứng - GPT Partition



1.5. Thiết bị I/O

- **Bàn phím**

Tổ chức bàn phím

Xử lý nhập ký tự từ bàn phím

Phím → bộ đệm

Scan code	ASCII code
-----------	------------

Xử lý phím bằng INT 21h (01h, 02h, 08h, 09h, 0ah)

Phím thông thường : Scan code=0

Phím mở rộng : ASCII code=0

Đọc 1 ký tự từ bàn phím

AH=01

...

AL=ký tự nhập (ASCII Code).

AL=ký tự nhập (Scan Code).



VD : đọc phím

...

Mov ah,01

Int 21h ;AL=KT nhập (ASCII code)

Cmp al,0

Jz PMR

;XL ASCII code

;

:PMR

Int 21h ;KT nhập (Scancode)

;XL Scancode

;

Xử lý phím bằng INT 16h
INT 16h (00, 01, ...)

AH=00-đọc 1 phím

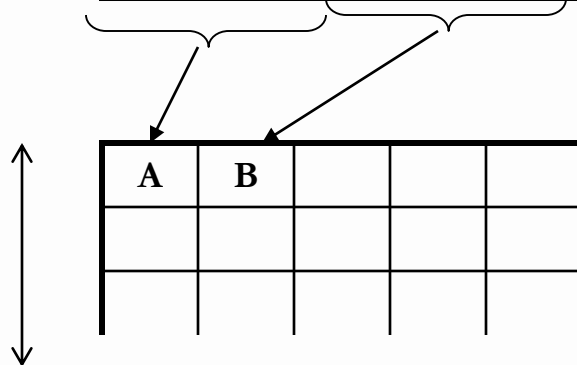
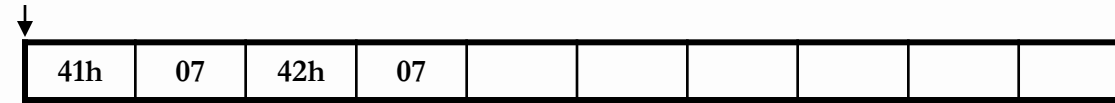
AL=ASCII Code

AH=Scancode

- **Màn hình**

Tổ chức màn hình và bộ nhớ màn hình : 1kt→2byte; B800:0000-địa chỉ bắt đầu

B800:0000



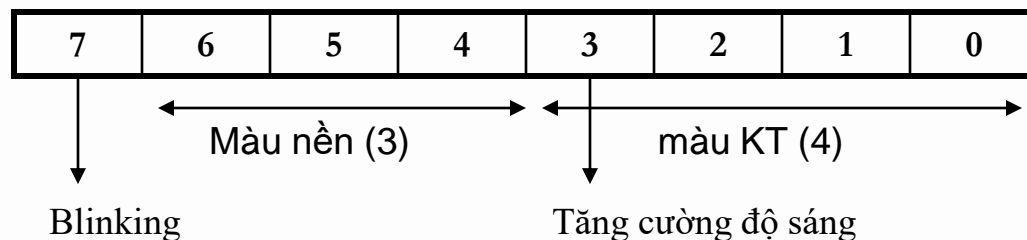
25 dòng

80 cột

VD về lưu trữ ký tự A, B màu trắng nền đen ở màn hình VGA 25x80

- Màn hình

Tổ chức màn hình và bộ nhớ màn hình : 1kt→2byte; B800:0000-địa chỉ bắt đầu Byte thuộc tính



INT 10h

AH=

00h	Set Video mode
01h	Set kthước con trỏ (text mode)
02h	Set vt con trỏ
03h	Lấy vt con trỏ
05h	Chọn trang MH (lật trang)
06h, 07h	Cuộn MH lên, xuống
08h	Đọc KT và thuộc tính
09h	Ghi KT và thuộc tính
0ah	Ghi KT

(*)

- **Mouse**

TB mouse

Xử lý Mouse INT 33h

AH=0

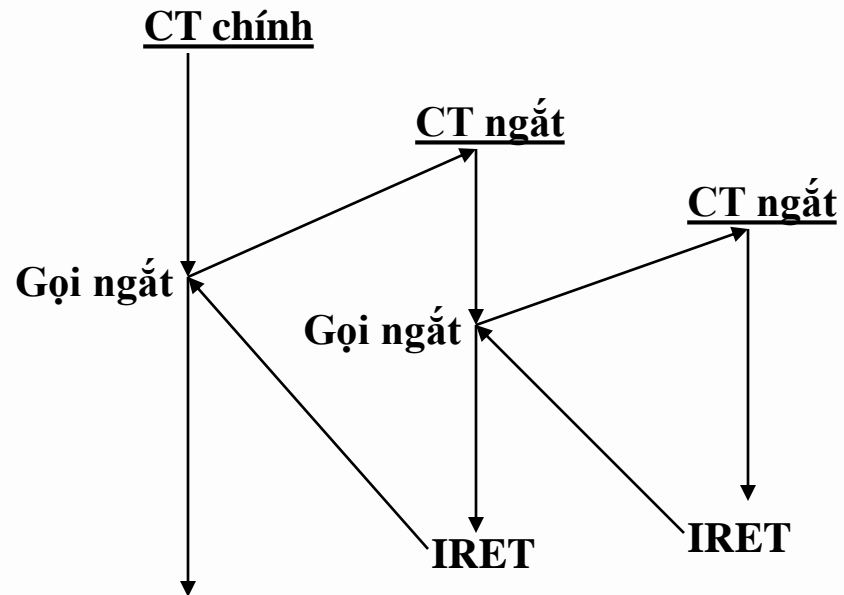
AL=

00h	Set/Reset mouse
01h	Hiện Mouse
02h	Ẩn Mouse
03h	Lấy vt và nút bấm
04h	Đặt vt Mouse
05h	Data nút bấm
06h	Data nút thả
07h	Xác định đường ngang
08h	Xác định đường dọc
09h	Mouse dạng đồ họa

2. Ngắt (Interrupt).

2.1. Ngắt

- Sơ đồ xử lý ngắt.



Bảng Vector ngắt
0000:0000-0000:03FF-256 Interrupt

Tính địa chỉ ngắt
mỗi Interrupt chiếm 4 Byte (offset:seg)→cách tính địa chỉ
của Interrupt

VD : int 13h → 13hx4 → 4Ch → 0000:004C
Offset = 004C ; Seg = 004E

Phân loại ngắt

Ngắt cứng : do tác động của phần cứng; 2h, 8h, 9h, Bh-Fh

Ngắt mềm : Bios, Dos (gọi trong CT)

Bios : 5, 10h-1Ch, 48h

Dos : còn lại (INT 21h)

Các ngắt cụ thể

Int 21h, INT 33h, INT 10h, INT 13h (Interrupt Help)

Lập trình sửa ngắt→thay đổi chức năng phần cứng

INT 21h (25h, 35h)

INT 27h

2.2. Lập trình sử dụng ngắt

- **Assembly**
Chương trình viết bằng Assembly
Gọi ngắt trong Assembly

Nhập (Input) : các giá trị gán cho các thanh ghi theo đúng qui định

Gọi ngắt

GT trả về (Output) : nếu có chứa trong các thanh ghi qui định

VD :

***Int 21h; AH=01h**

Input:

AH = 01h

Output: AL = character read

```
...  
mov ah,01  
int 21h; al=KT nhập  
...
```

***Int 21h; AH=0ah**

Input.:

AH = 0Ah

DS:DX -> buffer

Output: buffer filled with user input

Offset	Size	Description
00h	BYTE	Độ dài (N+2)
01h	BYTE	Số KT thực nhập
02h	N BYTEs	data nhập

```
.code  
...  
mov ax,@data  
mov ds,ax  
mov dx,offset buffer  
mov ah,0ah  
int 21h  
...  
.data  
buffer db n+2  
n_char db 0  
s_char db n dup(?)  
...
```


- C

***Chương trình C for DOS**

***Chuyển đổi địa chỉ :**

Địa chỉ thực → địa chỉ phân đoạn

unsigned FP_SEG(đ/c thực)

unsigned FP_OFF(đ/c thực)

VD :

...

char buff[100];

unsigned ds,dx;

...

ds=FP_SEG(buff);

dx=FP_OFF(buff);

...

Địa chỉ phân đoạn → địa chỉ thực

void far*MK_FP(SEG,OFF)

VD :

...

char *pchar;

...

pchar=(char*)MK_FP(es,bx);

...

***Hàm gọi ngắt :**

void geninterrupt(int num_intr);

Sử dụng các giả thanh ghi (AX→_AX, ...).

Goi ngắt tương tự như Assembly

VD :

#include <dos.h>

...

char buff[n];

int i,m;

...

_DS=FP_SEG(buff);

_DX=FP_OFF(buff);

_AH=0xA;

geninterrupt(0x21);

m=buff[1];

for(i=0, i<m, i++)

buff[i]=buff[i+2];

buff[m]=0;

...

int int86(int intr_num, union REGS *inregs, union REGS *outregs)
int int86x(int intr_num, union REGS *inregs, union REGS *outregs, struct SREGS *segregs)

int intdos(union REGS *inregs, union REGS *outregs)
int intdosx(union REGS *inregs, union REGS *outregs, struct SREGS *segregs)

void intr(int intr_num, struct REGPACK *preg)

***Cấu trúc data :**

struct WORDREGS (ax,bx,cx,dx,si,di,cflag,flags)
struct BYTEREGS (al,ah,bl,bh,cl,ch,dl,dh)
union REGS (WORDREGS x, BYTEREGS h)
struct SREGS (es,cs,ss,ds)
struct REGPACK (r_ax,r_bx,r_cx,r_dx,r_bp,r_si,r_di,r_ds,r_es,r_flags)

```
VD :
#include <dos.h>
...
char buff[n];
int i,m;
union REGS v,r;
struct SREGS s;
...
s.ds=FP_SEG(buff);
v.x.dx=FP_OFF(buff);
v.h.ah=0xA;
int86x(0x21,&v,&r,&s);
m=buff[1];
for(i=0, i<m, ++i)
    buff[i]=buff[i+2];
buff[m]=0;
...
```

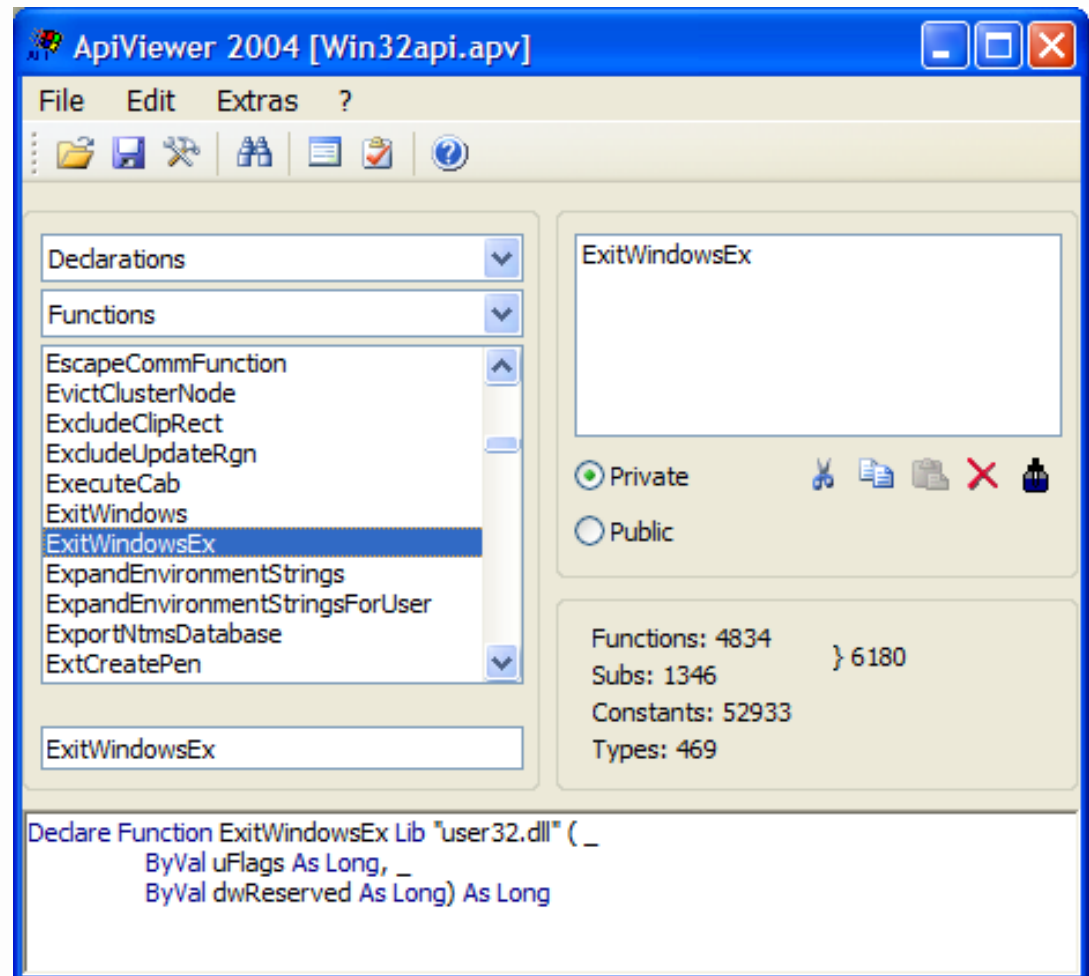
3. Hàm API (Application Programming Interface).

3.1. Hàm API

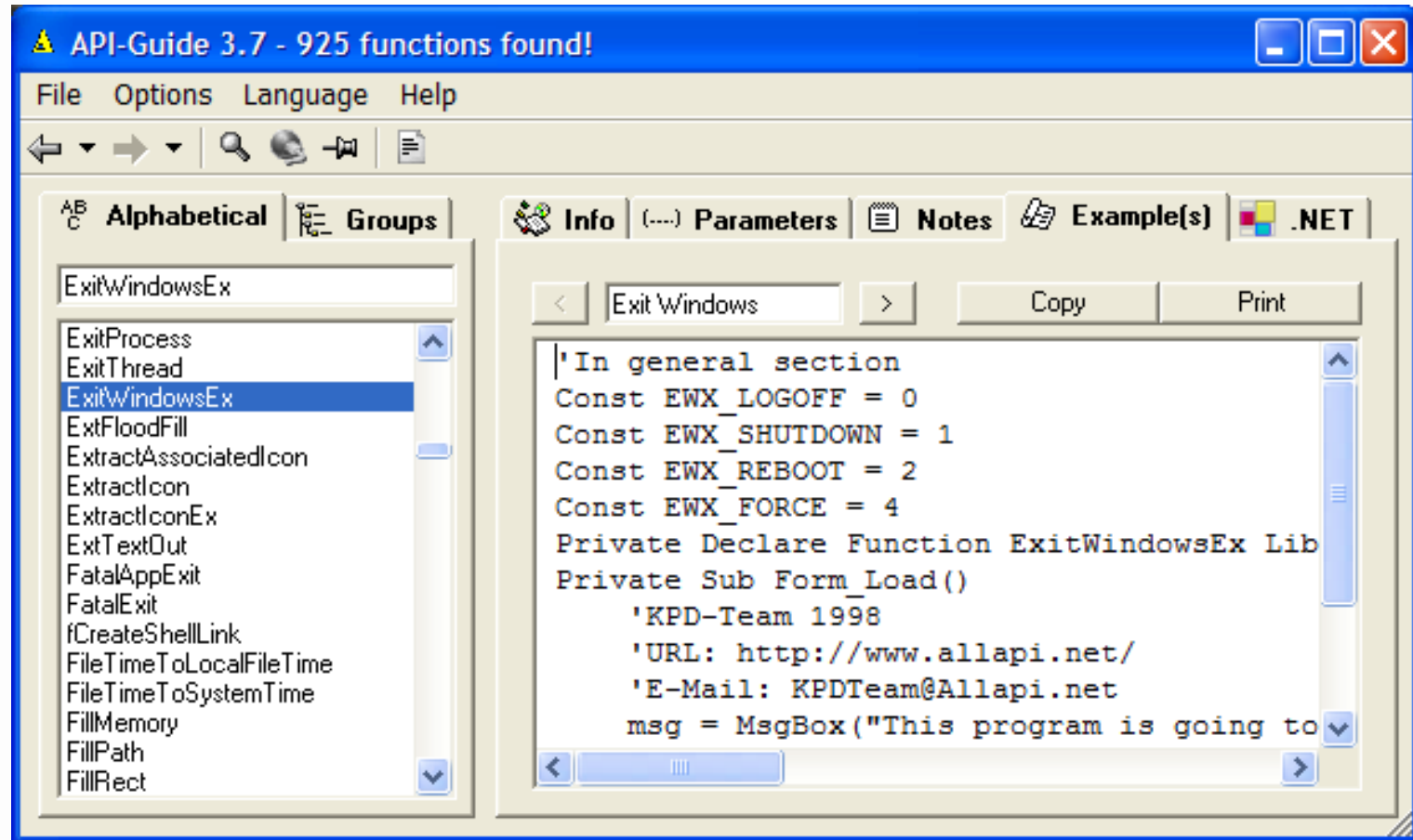
Hàm API (Application Programming Interface) được lưu trong các tập tin thư viện liên kết động (*.DLL - Dynamic Link Library).

Cung cấp 1 công cụ lập trình trong Windows

Sử dụng API Viewer để lấy ra các khai báo về hàm API.



Sử dụng API Guide để lấy ra các khai báo về hàm API



3.2. Lập trình sử dụng hàm API

VD : hàm ExitWindowsEx()

***Khai báo (API Viewer)**

**Declare Function ExitWindowsEx Lib "user32.dll" (_ ByVal uFlags As
Long,_ ByVal dwReserved As Long) As Long**

The **ExitWindowsEx** function either logs off, shuts down, or shuts down and restarts the system.

```
BOOL ExitWindowsEx(  
    UINT uFlags,           // shutdown operation  
    DWORD dwReserved      // reserved  
);
```

Parameters

uFlags

Specifies the type of shutdown. This parameter must be some combination of the following values:

Value	Meaning
EWX_FORCE	Forces processes to terminate. When this flag is set, Windows does not send the messages WM_QUERYENDSESSION and WM_ENDSESSION to the applications currently running in the system. This can cause the applications to lose data. Therefore, you should only use this flag in an emergency.
EWX_LOGOFF	Shuts down all processes running in the security context of the process that called the ExitWindowsEx function. Then it logs the user off.
EWX_POWEROFF	Shuts down the system and turns off the power. The system must support the power-off feature. Windows NT: The calling process must have the SE_SHUTDOWN_NAME privilege. For more information, see the following Remarks section. Windows 95: Security privileges are not supported or required.
EWX_REBOOT	Shuts down the system and then restarts the system. Windows NT: The calling process must have the SE_SHUTDOWN_NAME privilege. For more information, see the following Remarks section. Windows 95: Security privileges are not supported or required.
EWX_SHUTDOWN	Shuts down the system to a point at which it is safe to turn off the power. All file buffers have been flushed to disk, and all running processes have stopped. Windows NT: The calling process must have the SE_SHUTDOWN_NAME privilege. For more information, see the following Remarks section. Windows 95: Security privileges are not supported or required.

dwReserved

Reserved; this parameter is ignored.

Return Values

If the function succeeds, the return value is nonzero.

Sử dụng trong VB

‘Khai báo chung

**Declare Function ExitWindowsEx Lib "user32.dll" (_ ByVal uFlags As Long, _
ByVal dwReserved As Long) As Long**

....

Public Sub ThoatVaTatMay ()

'Thoát và tắt máy

Dim thoat

Thoat = ExitWindowsEX(2,0)

End Sub

...

‘Gọi trong CT

...

ThoatVaTatMay