LẬP TRÌNH XỬ LÝ ĐĨA&FILE

- CƠ BẢN VỀ LƯU TRỮ TRÊN ĐĨA TỪ.
- MỘT ỨNG DỤNG HIỂN THỊ SECTOR
- MỘT ỨNG DỤNG HIỂN THỊ CLUSTER.
- CÁC CHỨC NĂNG VỀ FILE Ở MỨC HỆ THỐNG.
- QUẢN LÝ ĐĨA VÀ THƯ MỤC.
- TRUY XUẤT ĐĨA VỚI INT 13H CỦA ROMBIOS
- BÀI TẬP
- GiỚI THIỆU FILE VÀ LẬP TRÌNH XỬ LÝ FILE

CƠ BẢN VỀ LƯU TRỮ TRÊN ĐĨA TỪ

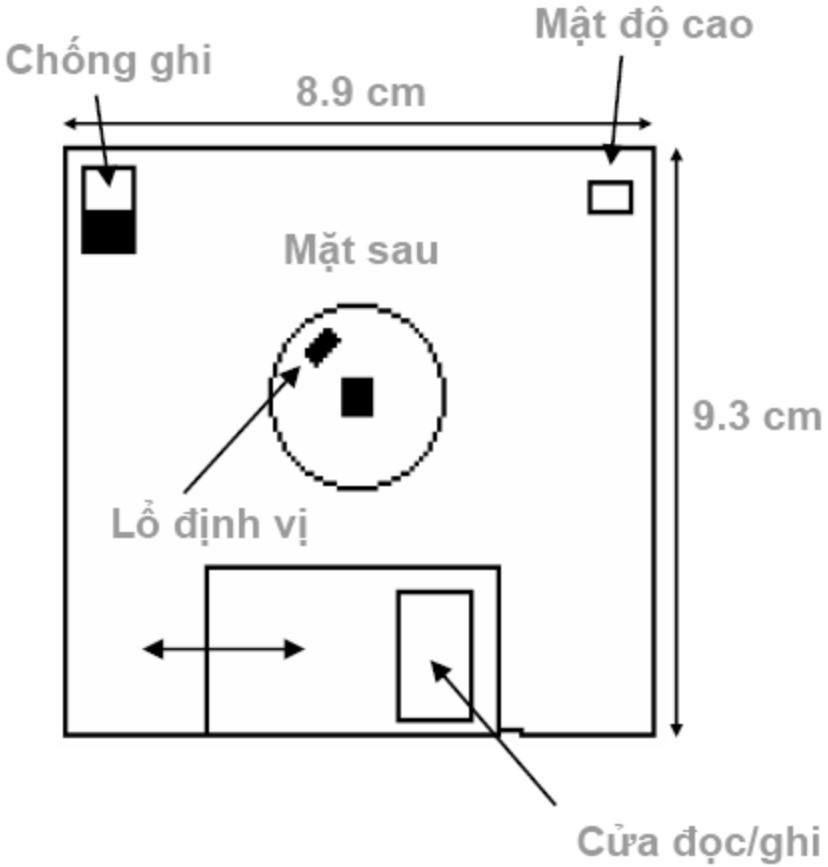
Ngôn ngữ ASM vượt trội hơn các ngôn ngữ khác về khả năng xử lý đĩa.

Ta xem xét việc lưu trữ thông tin trên đĩa theo 2 mức độ : mức phần cứng/BIOS và mức phần mềm/DOS.

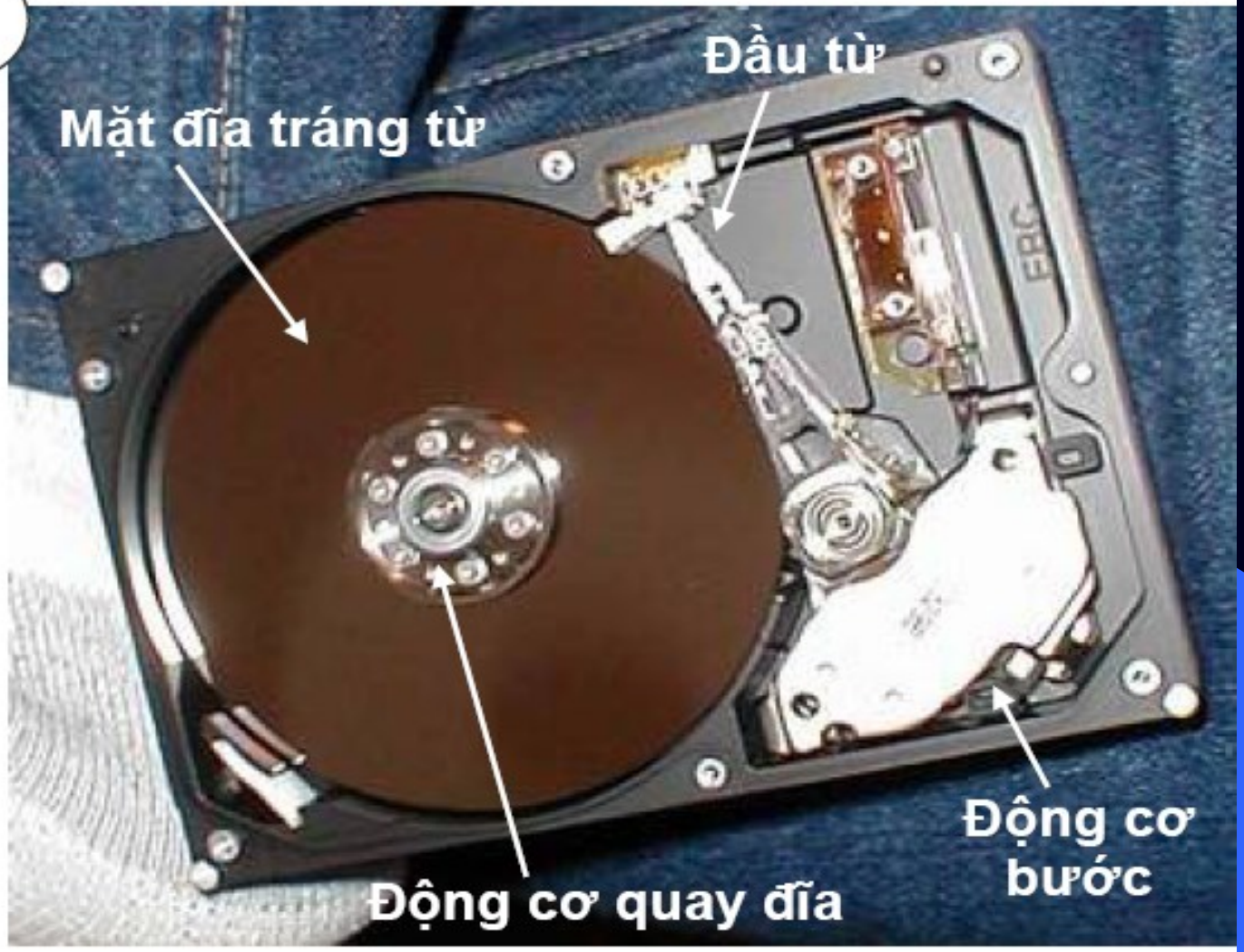
- **mức phần cứng :lưu trữ thông tin liên quan đến cách dữ liệu được lưu trữ 1 cách vật lý như thế nào trên đĩa từ?**

- **mức phần mềm : việc lưu trữ được quản lý bởi tiện ích quản lý File của HĐH DOS.**

Đĩa mềm 3.5”



Đĩa cứng

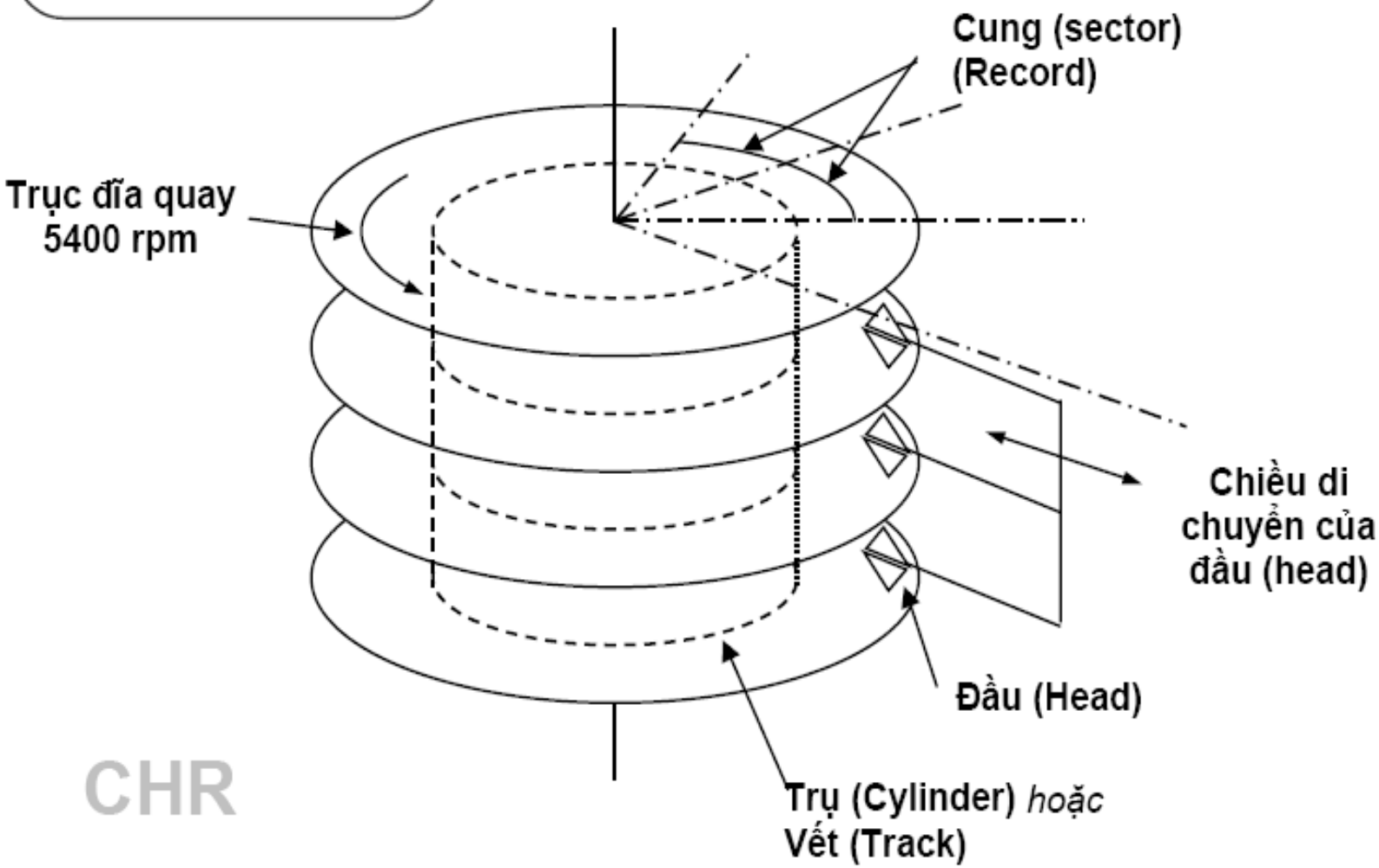


CÁC ĐẶC TÍNH LUẬN LÝ & VẬT LÝ CỦA ĐĨA TỪ

Ở mức vật lý : đĩa được tổ chức thành các Tracks, Cylinders, Sectors.

➔ Khả năng lưu trữ của đĩa được mô tả bằng 3 thông sô :
C (cylinder number)
H (Head side)
R (sector number)

Phân chia đĩa vật lý



CHR

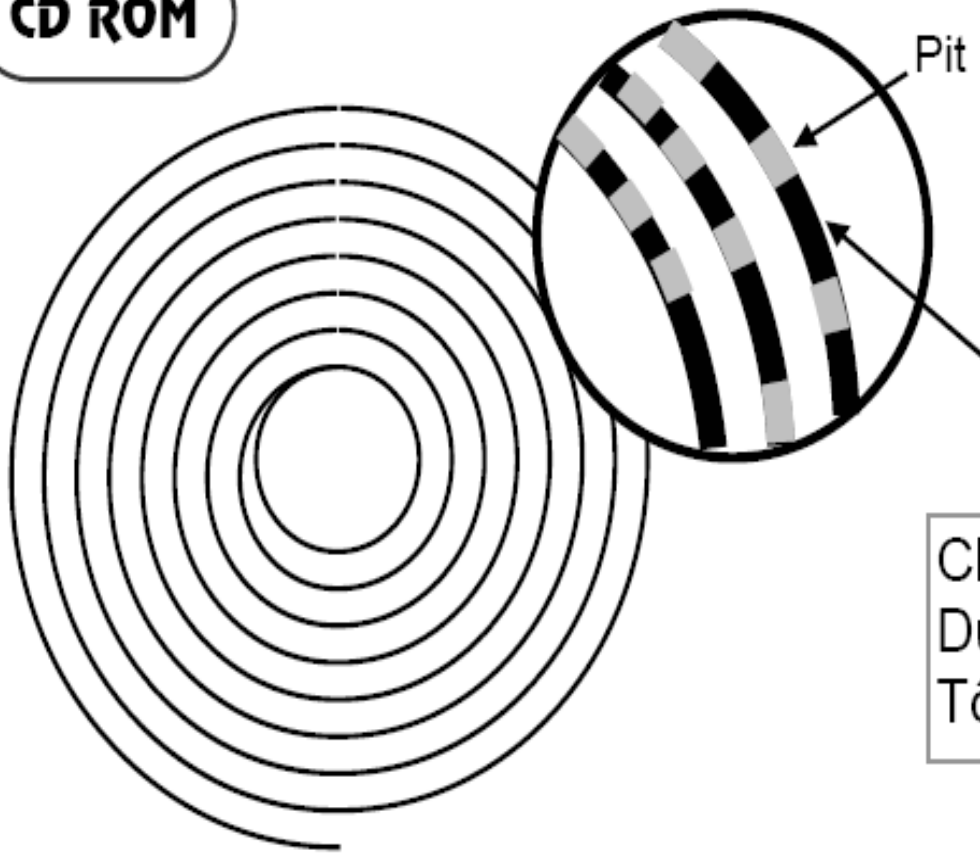
CÁC KHÁI NIỆM TRACK, CYLINDER, SECTOR

Tracks : là các vòng tròn đồng tâm được tạo ra trên bề mặt đĩa.

Cylinder : tập các tracks cùng bán kính trên 1 chồng đĩa. Mặt đĩa có bao nhiêu track thì sẽ có bấy nhiêu Cylinder.

Sector : là 1 đọan của track (cung từ) có khả năng lưu trữ 512 bytes dữ liệu. Các sector được đánh số bắt đầu từ 1 trên mỗi track → trên 1 đĩa tồn tại nhiều sector cùng số hiệu.

CD ROM



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.

Ở mức luận lý : đĩa được tổ chức thành các Clusters, các files mà DOS sẽ dùng để cấp phát vùng lưu trữ cho dữ liệu cần lưu trữ.

Cluster : là 1 nhóm gồm 2,4,6 các sector kề nhau. Đó chính là đơn vị cấp phát vùng lưu trữ cho dữ liệu (file). Các cluster được đánh số bắt đầu từ 0.

Nếu dữ liệu cần lưu trữ chỉ 1 byte thì hệ điều hành cũng cấp phát 1 cluster.

số bytes/cluster hay sector/cluster tùy thuộc vào từng loại đĩa.

TƯƠNG QUAN GIỮA SECTOR VẬT LÝ VÀ SECTOR LOGIC TRÊN ĐĨA MỀM

MẶT ĐĨA	TRACK	SECTOR	SECTOR LOGIC	THÔNG TIN
0	0	1	0	BOOT RECORD
0	0	2-5	1-4	FAT
0	0	6-9	5-8	Thư mục gốc
1	0	1-3	9-11	Thư mục gốc
1	0	4-9	12-17	Dữ liệu
0	1	1-9	18-26	Dữ liệu

BAD SECTOR

Trên bề mặt đĩa có thể tồn tại các sector mà HĐH không thể ghi dữ liệu vào đó hoặc không thể đọc dữ liệu từ đó. Các sector này gọi là Bad Sector.

Làm sao biết sector nào là bad sector

Kiểm tra giá trị của các phần tử (entry) trong bảng FAT, phần tử nào chứa giá trị (F)FF7H thì cluster tương ứng bị Bad

BẢNG FAT
FILE ALLOCATION TABLE

DOS quản lý các File nhờ vào 1 bảng gọi là bảng FAT.

Trong bảng FAT có ghi cluster bắt đầu của File này ở đâu ? Và đĩa còn bao nhiêu Clusters trống chưa cấp phát.

tổ chức luận lý của đĩa được mô tả như hình sau :

Thí dụ về bảng FAT

Đĩa mềm 3.5”” 360K thì :

Sector 0 : boot sector

Sector 1-4 : bảng FAT

Sector 5 – 11 : thư mục gốc

Sector 12-719 : vùng chứa data

BOOT RECORD

- Còn được gọi là Boot Sector. Ổ đĩa cứng gọi là Master boot, là Sector đầu tiên khi đĩa được format.
- chứa 1 chương trình nhỏ cho biết dạng lưu trữ trên đĩa và tên hệ thống MT, kiểm tra xem có các file hệ thống IO.SYS, MSDOS.SYS, COMMAND.COM hay không ?
- nếu có thì nạp chúng vào bộ nhớ (gọi là chương trình mồi của HĐH)

BOOT RECORD (tt)

- **Tọa độ vật lý :**

C=0, H=0, R =1 (C0H0R1) tức ở tại sector đầu tiên của track đầu tiên, mặt trên của đĩa đầu tiên trong ổ đĩa cứng.

- **Trong Master boot có chứa bảng PARTITION TABLE cho biết tầm địa chỉ vật lý (dung lượng) của ổ đĩa luận lý.**

Master boot không thuộc Partition nào

BOOT RECORD (tt)

- BOOT RECORD được ROM BIOS nạp vào địa chỉ 0000:7C00H.
- Nếu máy không bị Virus thì lệnh đầu tiên của chương trình BOOT là JMP 7C3EH, nghĩa là nhảy đến chương trình nạp mồi.
- chương trình nạp mồi (Bootstrap Loader) nạp thành phần cốt lõi của DOS lên RAM trong quá trình khởi động MT.

THÔNG TIN TRONG MASTER BOOT

BYTEBĐẦU	SỐ BYTES	THÔNG TIN
00H	3	chỉ thị nhảy về nới chứa CT nạp mồi
03H	8	Tên nhà sản xuất và hệ điều hành
0BH	2	Bytes/sector
0DH	1	Sector/block (mỗi block >=1 sector)
0EH	2	Số lượng Sectors không dùng đến kể từ sector 0.
10h	1	Số lượng bảng FAT

THÔNG TIN TRONG MASTER BOOT

BYTE BĐẦU	SỐ BYTES	THÔNG TIN
11H	2	Số Entry của thư mục gốc ổ đĩa.
13H	2	Tổng số sector của ổ đĩa logic này.
15H	1	Byte mô tả
16H	2	Số sector cho 1 bảng FAT
18H	2	Số Sectors trong 1 track.
1AH	2	Số lượng đầu đọc
1CH	4	Số lượng sector ẩn
20H	4	Tổng số sectors

THÔNG TIN TRONG MASTER BOOT

BYTE BĐẦU	SỐ BYTES	THÔNG TIN
3EH		Bootstrap
….		
1BEH	64	PARTITION TABLE
……		
1FEH	1	Giá trị 55H
1FFH	1	Giá trị 0AAH

THÔNG TIN TRONG MASTER BOOT

Từ thông tin trong bảng FORMAT, ta tính được địa chỉ của bảng FAT1, FAT2, Thư mục gốc ổ đĩa, địa chỉ bắt đầu của vùng dữ liệu.

BẢNG FAT

- Bảng chứa các danh sách liên kết các clusters. Mỗi danh sách trong bảng cho DOS biết rằng các clusters nào đã cấp phát, các clusters nào chưa dùng.
  - tùy theo ổ đĩa có thể có 1 hay 2 bảng FAT, bảng FAT2 để dự phỏng.
- có 2 loại bảng FAT :

  bảng có Entry 12 bit cho đĩa mềm.

  bảng có Entry 16 bit cho đĩa cứng.

PARTITON TABLE

64 Bytes của Partiton table được chia làm 4, mỗi phần 16 bytes mô tả cho 1 partition các thông tin sau :

Bytes	Mô tả
00H	active flag (=0 Non bootable =80H Bootable)
01H	starting head – Nơi bắt đầu Partittion
02H	starting cylinder

PARTION TABLE

Bằng FDISK của HĐH ta có thể chia không gian lưu trữ của đĩa cứng thành các phần khác nhau gọi là Partition.

DOS cho phép tạo ra 3 loại Partition :

Primary Dos, Extended Dos và None Dos

Ta có thể cài đặt các HĐH khác nhau lên các Partition khác nhau.

PARTITON TABLE

03H starting sector

04H parttition type :

0 Non Dos

1 cho đĩa nhỏ 12 bit FAT Entry

4 cho đĩa lớn 16 bit FAT Entry

5 Extended Dos

05H Ending nơi kết thúc Partition

06H Ending Cylinder

07H Ending Sector

08H, 0BH Starting sector for partition

0Ch,0FH Partition length in sectors

Một số thí dụ

kiểm tra Partition Active

- đọc sector đầu tiên của đĩa cứng lưu vào biến.
- kiểm tra offset 00 của 4 phần tử Partition trong Partition Table

```
MOV CX, 4
MOV SI, 1BEH
PACTIVE :
MOV AL, MBOOT [SI]
CMP AL, 80H
JE ACTIVE
ADD SI, 16
LOOP PACTIVE
NO_ACTIVE :
.....
ACTIVE : .....
```

Một số thí dụ

Đọc nội dung của BootSector ghi vào biến dem

- đọc sector đầu tiên của đĩa cứng lưu vào buffer.
- tìm partition active (phần tử trong bảng partition có offset 80h)
- đọc byte tại offset 01h và word tại offset 02h của phần tử partition tương ứng ở trên (head, sector, cylinder) để xác định số hiệu bắt đầu của partition active → boot sector của đĩa cứng.
- đọc nội dung của sector đọc được ở trên lưu vào buffer.

Một số thí dụ

ACTIVE :

```
MOV AX, 0201H ; đọc 1 sector
MOV CX, WORD PTR MBOOT [SI+2] ; sector cylinder
MOV DH, BYTE PTR MBOOT[SI+1] ; head
MOV DL, 80H ; đĩa cứng
MOV ES, CS ; trỏ về  đầu vùng buffer lưu
LEA BX, BUFFER
INT 13H
```

THƯ MỤC GỐC (ROOT DIRECTORY)

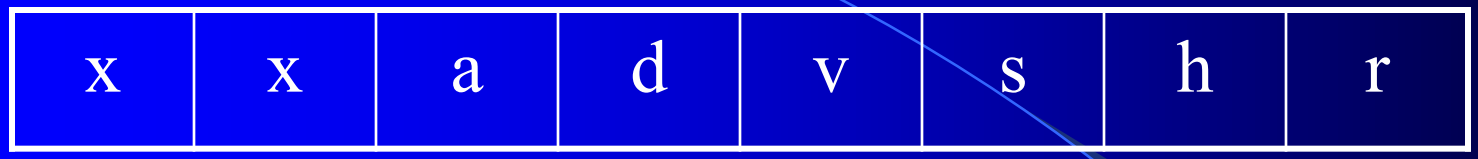
- Là danh sách tất cả các Files đã có trên đĩa, các thư mục cấp 1 đã có.
- Mỗi phần tử (32 bytes) trong bảng thư mục sẽ chứa thông tin về tên file hoặc là thư mục, kích thước, thuộc tính, cluster bắt đầu của file này hoặc cluster bắt đầu của thư mục thứ cấp (thư mục con).

mỗi bảng thư mục chứa tối đa 112 entry, mỗi entry là 32 bytes.

THƯ MỤC GỐC (ROOT DIRECTORY)

Offset	Nội dung	Kích thước
00H	tên chính của File	8 bytes
08H	phần mở rộng của tên file	3 bytes
0BH	thuộc tính của File	1 byte
0CH	dự trữ	10 bytes
16H	giờ thay đổi thông tin cuối cùng	2 bytes
18H	ngày thay đổi thông tin cuối cùng	2 bytes
1Ah	cluster đầu tiên của File	2 bytes
1CH	Kích thước File	4bytes

BYTE THUỘC TÍNH



x : không sử dụng

a : thuộc tính lưu trữ (Archive)

d : thuộc tính thư mục con (Sub – Directory)

v : thuộc tính nhãn đĩa (Volume)

s : thuộc tính hệ thống (System)

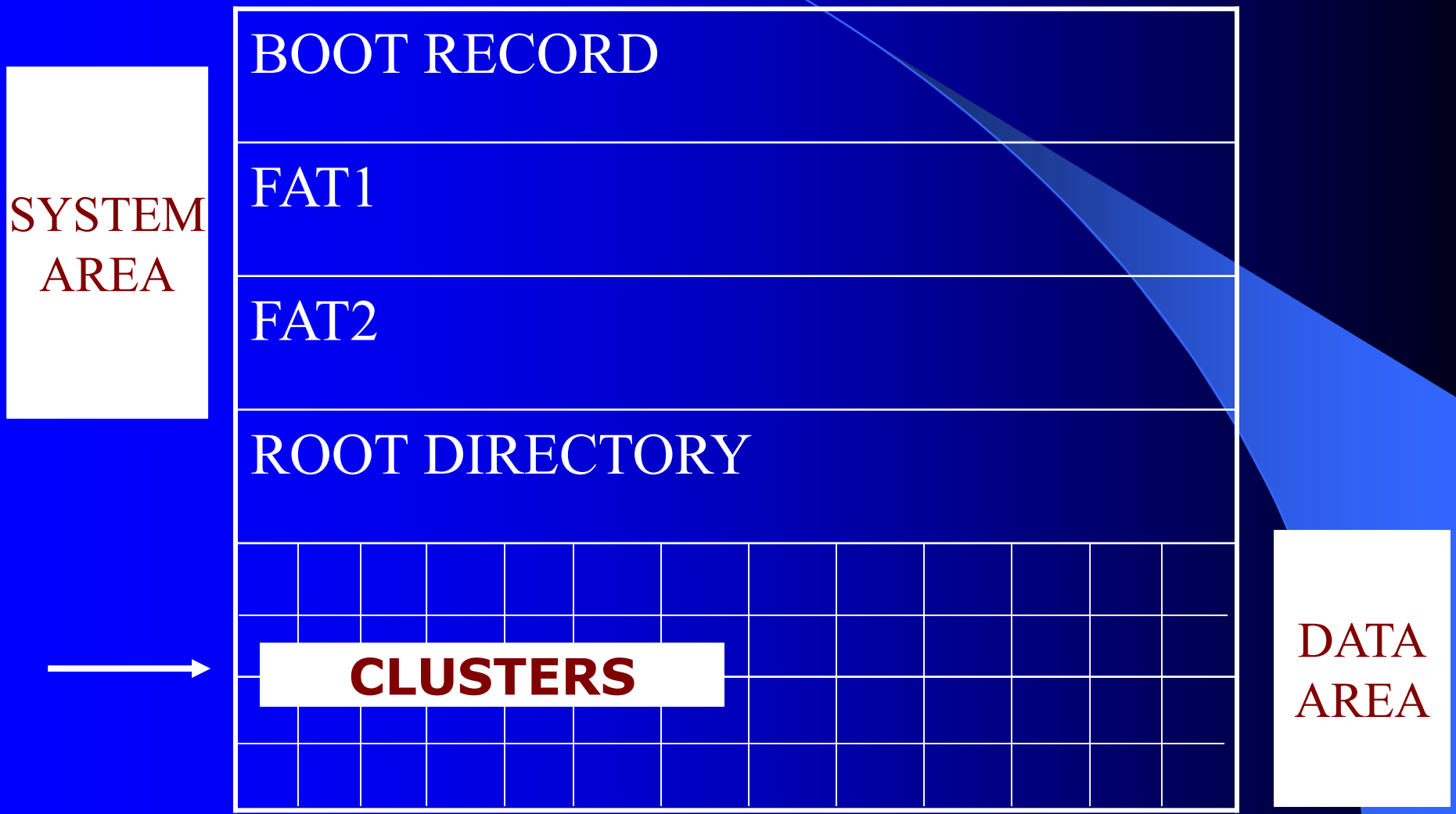
h : thuộc tính ẩn (Hidden)

r : thuộc tính chỉ đọc (Read Only)

VÙNG LƯU TRỮ

- là vùng dành cho việc lưu trữ dữ liệu.
- như vậy việc lưu trữ dữ liệu trên đĩa có cấu trúc là 1 danh sách liên kết mà bảng thư mục gốc là đầu của danh sách liên kết.
- đầu mỗi cluster luôn luôn chứa địa chỉ của cluster sau nó cho biết phần còn lại của file là cluster nào. Nếu giá trị này là 0 thì cluster này là cluster cuối cùng.

SỰ PHÂN VÙNG TRÊN ĐĨA



CÁC LOẠI ĐĨA

Disk Type	sides	track per side	sectors per track	total sector	cluster size	total bytes
360K	2	40	9	720	1,024	368,640
720K	2	80	9	1,440	512	737,280
1.2MB	2	80	15	2,400	512	1,228,800
1.4MB	2	80	18	2,880	512	1,474,560
32MB	6	614	17	62,610	2,048	32,056,832

TÍNH DUNG LƯỢNG ĐĨA

Công thức tính dung lượng đĩa :

Dung lượng đĩa (bytes) = số byte/1 sector * số sector/1 track * số track/ 1 mặt đĩa * số mặt đĩa.

MỘT SỐ HÀM THAO TÁC VỚI FILE VÀ ĐĨA INT 21H

HÀM 36H INT 21H :

Lấy số bytes còn trống trên đĩa

Input :

AH = 36H DL = O63 đĩa (0 : mặc định, 1 ổ A ….

Output :

Có lỗi AX = 0FFFFH

Không lỗi : AX = số sector / cluster

BX = số cluster còn trống

DX = tổng số cluster trên đĩa

CX = số bytes/cluster

BÀI TẬP

Viết chương trình tạo thư mục với yêu cầu tên thư mục (có thể bao gồm tên ổ đĩa, đường dẫn và tên thư mục) được nhập từ bàn phím, cho phép sửa sai khi gỏ nhầm tên thư mục.

Viết chương trình ghi dữ liệu vào file với yêu cầu :

- **Tên file nhập từ bàn phím**
- **Dữ liệu ghi vào file cũng gỏ từ bàn phím và kết thúc việc nhập bằng phím CTRL+Z**

Viết chương trình gộp nội dung 1 file vào cuối 1 file khác.

LẬP TRÌNH XỬ LÝ FILE

GiỚI THIỆU FILE
CÁC HÀM CHỨC NĂNG XỬ LÝ FILE CỦA INT 21H CỦA DOS

GIỚI THIỆU FILE

- Trong quản lý File, Dos vay mượn khái niệm Handle trong HĐH Unix để truy xuất File và thiết bị.

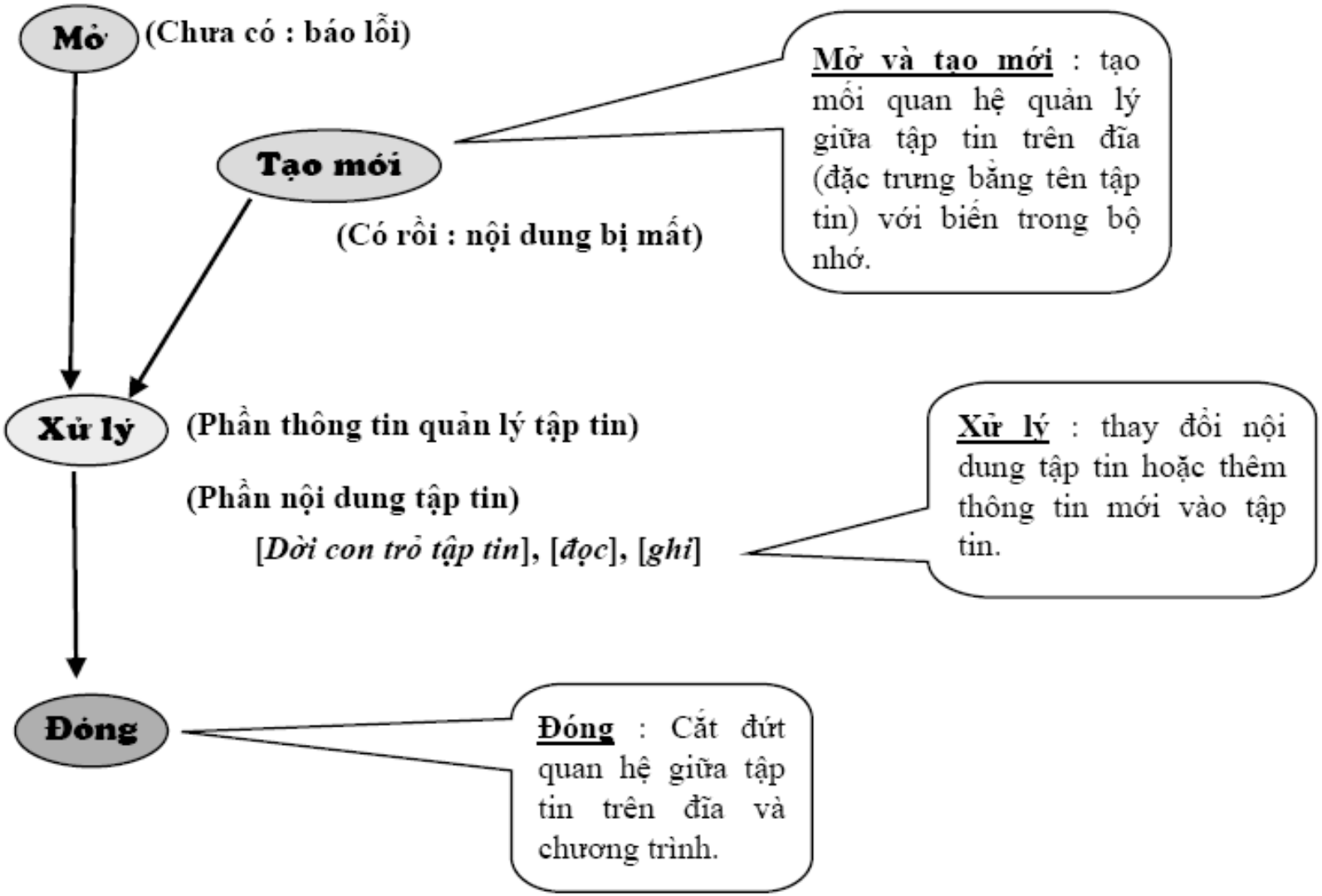
- Handle là 1 số 16 bits được Dos sử dụng để nhận biết File đã mở hoặc 1 thiết bị trong hệ thống.

GIỚI THIỆU FILE

- **Có 5 Handle thiết bị chuẩn được Dos nhận dạng.**

Handle	Thiết bị
0	Keyboard, standard input
1	Console, standard output
2	Error output thiết bị xuất lỗi – màn hình
3	Auxiliary device asynchronous
4	Printer

CÁC THAO TÁC XỬ LÝ FILE



CÁC CHỨC NĂNG CƠ BẢN VỀ XỬ LÝ FILE CỦA INT 21H

Chức năng	Tác vụ
3Ch	Tạo File mới
3Dh	Mở File đã có để xuất/nhập/vừa nhập vừa xuất
3Eh	Đóng thẻ File
3Fh	Đọc từ File hay đọc từ thiết bị 1 số bytes định trước
40h	Ghi vào File hay đọc từ thiết bị 1 số bytes định trước
42h	di chuyển con trỏ File trước khi đọc/ ghi

CHỨC NĂNG TẠO FILE 3Ch
CREATE FILE FUNCTION 3Ch

Chức năng : Mở 1 File mới để đọc ghi. Nếu file đã có thì file cũ sẽ bị xóa.

AH = 3Ch

DS:DX địa chỉ của tên File muốn mở (ASCIIZ String)

CX = thuộc tính File

(0 normal 1 ReadOnly 2 Hidden 4 System)

Xuất : không lỗi CF =0 AX = File Handle Có lỗi CF =1.

Mã lỗi trong AX (3,4,5).

CHỨC NĂNG TẠO FILE 3Ch
CREATE FILE FUNCTION 3Ch

Ex :

```
CREATE_FILE :
MOV AH, 3CH
MOV DX, OFFSET NEWFILE
MOV CX, 0
INT 21H
JC DISPLAY_ERROR
MOV NEWFILEHANDLE, AX
...
NEWFILE DB ‘ FILE1.DOC ’,0
NEWFILEHANDLE  DW ?
```

CHỨC NĂNG TẠO FILE 3Ch
CREATE FILE FUNCTION 3Ch

Ex :

CHỨC NĂNG 3Ch CÓ 1 KHUYẾT ĐIỂM LÀ NẾU CÓ 1 FILE CÙNG TÊN(CÙNG ĐƯỜNG DẪN) ĐÃ TỒN TẠI THÌ FILE CŨ SẼ BỊ XÓA.

ĐỂ BẢO VỆ FILE, CÓ 2 CÁCH :

C1 : MỞ FILE BẰNG CHỨC NĂNG 3Dh, NẾU FILE CHƯA CÓ THÌ TRẢ VỀ LỖI SỐ 2 (FILE NOT FOUND) ➔ YÊN TÂM MỞ FILE MỚI.

C2 : DÙNG CHỨC NĂNG 5Bh MỞ FILE CÓ KIỂM TRA TÊN FILE NÀY ĐÃ CÓ CHƯA.

CHỨC NĂNG 5Bh
TẠO FILE MỚI CÓ KIỂM TRA

ĐIỀU KIỆN : GIỐNG CHỨC NĂNG 3Ch

NẾU FILE NÀY ĐÃ CÓ THÌ KHÔNG MỞ FILE MỚI MÀ TRẢ VỀ LỖI 50h

```
CREATE_FILE :
MOV AH,5BH
MOV DX, OFFSET FILENAME
MOV CX, 0
INT 21H
JC  ERROR
 .
FILENAME  DB 'FILE1.DOC' , 0
```

CÁC LỖI KHI MỞ FILE

MÃ LỖI DIỄN GIẢI

2 FILE NOT FOUND KHÔNG TÌM THẤY FILE, CÓ THỂ ĐƯỜNG DẪN KHÔNG ĐÚNG HOẶC TÊN FILE MÔ TẢ KHÔNG HỢP LỆ.

3 PATH NOT FOUND ĐƯỜNG DẪN KHÔNG CÓ.

4 TOO MANY OPEN FILES CÓ THỂ DO LỆNH PATH XX TRONG CONFIG.SYS QUÁ NHỎ KHÔNG CHO PHÉP MỞ NHIỀU FILE.

5 ACCESS DENIED TỪ CHỐI TRUY XUẤT. CÓ THỂ TA MUỐN XOÁ FILE ĐANG MỞ, HAY FILE NÀY CÓ THUỘC TÍNH CHỈ ĐỌC.

CH Mã truy nhập không hợp lệ.

FH Ổ đĩa không hợp lệ

10h Đang tìm cách xóa thư mục hiện thời

CÁC LỖI KHI MỞ FILE

MÃ LỖI	DIỄN GIẢI
11H	Không cùng thiết bị
12H	Không tìm được thêm File nào

CHỨC NĂNG MỞ FILE ĐÃ CÓ 3Dh Int 21h
OPEN FILE

ĐIỀU KIỆN :

AH = 3DH DS:DX ĐỊA CHỈ TÊN FILE

AL = MODE

0: INPUT (MỞ CHỈ ĐỌC)
1 : OUTPUT (MỞ ĐỂ GHI)

2 : INPUT OUTPUT (MỞ VỪA ĐỌC VỪA GHI)

XUẤT :

KHÔNG LỖI CF = 0 AX = FILE HANDLE

CÓ LỖI CF = 1 AX ← mã lỗi (2,4,512)

MỞ FILE HÀM 3CH INT 21H

- Trước khi sử dụng 1 file, ta phải mở nó.
- Để tạo 1 file mới hay ghi lại 1 file cũ, ta sử dụng tên file và thuộc tính của File.
- ➔ DOS trả về thẻ file

MỞ FILE HÀM 3CH INT 21H

AH = 3CH
DS:DX địa chỉ của chuỗi ASCII
(chuỗi tên File kết thúc bằng byte 0)
CL = thuộc tính File
Nếu thành công, AX = thẻ File
Nếu CF được set thì có lỗi, mã lỗi chứa trong AX
(lỗi 3,4,5)

Viết code mở 1 File mới với thuộc tính chỉ đọc, tên File là FILE1

```
Fname DB ‘FILE1’,0
FHANDLE DW ?
MOV AX,@DATA
MOV DS,AX
MOV AH,3CH
MOV CL,1
LEA DX,FNAME
INT 21H
MOV FHANDLE, AX
JC OPEN_ERROR
……
```

CHỨC NĂNG MỞ FILE ĐÃ CÓ SẴN
HÀM 3Dh INT 21H
OPEN FILE

AH = 3DH

DS:DX = địa chỉ của chuỗi ASCII
(chuỗi tên File kết thúc bằng byte 0)

AL = mã truy cập

0 : mở để đọc

1 : mở để ghi

2 : mở để đọc và ghi

➔Thành công, AX = Fhandle

➔Có lỗi. Mã lỗi chứa trong AX (2,4,5,12)

CHỨC NĂNG MỞ FILE ĐÃ CÓ SẴN
HÀM 3Dh INT 21H
OPEN FILE

```
MOV AH, 3DH
MOV AL, 0
MOV DX, OFFSET FILENAME
INT 21H
JC DISPLAY_ERROR
MOV  INFILEHANDLE, AX
 .....
INFILE  DB ‘ D:\FILE1.DOC’, 0
INFILEHANDLE  DW  ?
```

CHỨC NĂNG 3EH ĐÓNG FILE

ĐIỀU KIỆN :

AH = 3EH BX = FILE HANDLE CẦN ĐÓNG

XUẤT :

KHÔNG LỖI CF = 0   CÓ LỖI CF = 1

EX :

```
MOV AH, 3EH
MOV BX, INFILEHANDLE
INT 21H
JC DISPLAY_ERROR
..
INFILE  DB 'D:\FIEL1.DOC', 0
INFILEHANDLE  DW ?
```

LỖI SỐ 6 : INVALID HANDLE

FILE HANDLE TRONG BX KHÔNG PHẢI LÀ THẺ FILE CỦA FILE ĐÃ MỞ.

CHỨC NĂNG 3FH ĐỌC FILE

ĐỌC 1 SỐ BYTES TỪ FILE LƯU VÀO BỘ NHỚ

ĐIỀU KIỆN :

AH = 3FH BX = FILE HANDLE , CX = SỐ BYTES CẦN ĐỌC

DS:DX : ĐỊA CHỈ BỘ ĐỆM.

XUẤT :

AX = SỐ BYTES ĐỌC ĐƯỢC, NẾU AX = 0 HAY AX<CX FILE ĐÃ KẾT THÚC.

NẾU CỜ CF ĐƯỢC LẬP ➔ CÓ LỖI, MÃ LỖI CHỨA TRONG AX(5,6)

CHỨC NĂNG 3FH ĐỌC FILE

EX : ĐỌC 1 SECTOR 512 BYTES TỪ FILE

```
.DATA
HANDLE DW ?
BUFFER DB 512 DUP(?)
MOV AX, @DATA
MOV DS, AX
MOV AH, 3FH
MOV CX, 512
MOV BX, HANDLE
MOV CX, 512
INT 21H
JC READ_ERROR
```

NẾU CẦN ĐỌC HẾT CÁC SECTOR CHO ĐẾN HẾT FILE ➔ EOF

```
CMP AX, CX
JL  EXIT
JMP READ_LOOP
```

CHỨC NĂNG 40H GHI FILE

GHI 1 SỐ BYTES LÊN FILE HAY THIẾT BỊ

INPUT :

AH =40H BX = THẺ FILE CX = SỐ BYTES CẦN GHI

DS:DX : ĐỊA CHỈ VÙNG ĐỆM.

OUTPUT :

AX : SỐ BYTES GHI ĐƯỢC, NẾU AX<CX, CÓ LỖI (ĐĨA ĐẦY).NẾU CF ĐƯỢC LẬP → CÓ LỖI, MÃ LỖI TRONG AX (5,6).

HÀM 40H CŨNG CÓ THỂ DÙNG ĐỂ ĐƯA DỮ LIỆU RA MÀN HÌNH

CON TRỎ FILE

- DÙNG ĐỂ ĐỊNH VỊ TRONG FILE.
- KHI FILE ĐƯỢC MỞ, CON TRỎ FILE NẰM Ở ĐẦU FILE.
- SAU MỖI THAO TÁC ĐỌC, CON TRỎ FILE SẼ DI CHUYỂN ĐẾN BYTE KẾ.
- SAU KHI GHI 1 FILE MỚI CON TRỎ CHỈ ĐẾN CUỐI FILE (EOF).
- ĐỂ DI CHUYỂN CON TRỎ FILE HÀM 42H

MINH HỌA LẬP TRÌNH FILE

Viết chương trình cho phép User gỏ vào tên File (có thể có kèm theo tên ổ đĩa, thư mục chứa fiel), chương trình sẽ đọc và hiển thị nội dung File ra màn hình.

DỊCH CHUYỂN CON TRỎ FILE
HÀM 42H INT 21H

AH = 42H AL = PHƯƠNG THỨC TRUY NHẬP

0 DỊCH CHUYỂN TƯƠNG ĐỐI SO VỚI ĐẦU FILE.

1 DỊCH CHUYỂN TƯƠNG ĐỐI SO VỚI VỊ TRÍ HIỆN THỜI CỦA CON TRỎ.

2 DỊCH CHUYỂN TƯƠNG ĐỐI SO VỚI CUỐI FILE.

BX = THẺ FILE.

CX : DX SỐ BYTES CẦN DỊCH CHUYỂN.

OUTPUT :

DX:AX : VỊ TRÍ MỚI CỦA CON TRỎ FILE TÍNH BẰNG BYTE TỪ ĐẦU FILE.

NẾU CF =1 MÃ LỖI TRONG AX (1, 6).

DỊCH CHUYỂN CON TRỎ FILE HÀM 42H INT 21H

CX : DX CHỨA SỐ BYTES ĐỂ DI CHUYỂN CON TRỎ. NẾU LÀ SỐ DƯƠNG ➔ CHUYỂN VỀ CUỐI FILE.

NẾU LÀ SỐ ÂM ➔CHUYỂN VỀ ĐẦU FILE.

DI CHUYỂN CON TRỎ FILE ĐẾN CUỐI FILE VÀ XÁC ĐỊNH KÍCH THƯỚC FILE

```
MOV AH, 42H          ; DI CHUYỂN CON TRỎ FILE
MOV BX, HANDLE   ; LẤY THẺ FILE
XOR DX, DX
XOR CX, CX   ; DỊCH CHUYỂN 0 BYTE
MOV AL, 2     ; TÍNH TỪ CUỐI FILE
INT 21H  ; CHUYỂN CON TRỎ ĐẾN CUỐI FILE, DX:AX KÍCH THƯỚC FILE
JC MOVE_ERROR
```

THAY ĐỔI THUỘC TÍNH FILE
HÀM 43H INT 21H

INPUT :

AH = 43H DS :DX = ĐỊA CHỈ CHUỖI ASCII STRING

AL = 0 ĐỂ LẤY THUỘC TÍNH FILE AL =1 ĐỂ THAY ĐỔI THUỘC TÍNH FILE, CX = THUỘC TÍNH FILE MỚI (NẾU AL =1)

OUTPUT :

NẾU THÀNH CÔNG, CX = THUỘC TÍNH HIỆN THỜI

NẾU CF ĐƯỢC LẬP ➔ CÓ LỖI, MÃ LỖI TRONG AX (2,3,5).

Ex : thay đổi thuộc tính File thành hidden file

```
MOV AH, 43H          ; Hàm lấy / đổi thuộc tính File
MOV AL, 1            ; tuỳ chọn thay đổi thuộc tính
LEA DX, FILENAME     ; lấy tên file kế cả đường dẫn.
MOV CX, 1            I; thuộc tính Hideen
INT 21H              ; đổi thuộc tính
JC ATT_ERROR         ; thoát nếu có lỗi, mã lỗi trong AX
```

LẬP TRÌNH FILE

1. Viết chương trình chép một file nguồn đến một file đích trong đó thay chữ thường bằng chữ hoa.
2. Viết chương trình đọc 2 file và hiển thị chúng bên cạnh nhau trên màn hình. Chú ý có chức năng dừng từng trang màn hình nếu file quá dài.
3. Viết chương trình ghép nội dung 1 file vào cuối 1 file khác đã có.
4. Viết chương trình tạo 1 thư mục, tên thư mục được gỏ từ bàn phím (tên thư mục có thể bao gồm tên ổ đĩa, đường dẫn).