

## **Lời cảm ơn!**

Trong thời gian ngồi trên ghế của giảng đường Đại Học Dân Lập Hải Phòng em đã nhận được sự chỉ bảo tận tình của các thầy cô giáo, được sự quan tâm của các thầy cô trong Ban giám hiệu nhà trường, bản thân em nói riêng và toàn thể sinh viên năm cuối nói chung đã trưởng thành học hỏi được nhiều điều bổ ích. Các thầy cô đã tạo điều kiện tốt nhất cho chúng em đi sâu thâm nhập vào thực tế. Đặc biệt, chúng em có cơ hội để kiểm chứng những điều đã được học hàng ngày ở nhà trường bằng những kinh nghiệm thực tiễn có thật.

Em xin gửi lời cảm ơn chân thành của mình tới toàn thể các thầy cô trong Ban giám hiệu nhà trường, các thầy cô đã giảng dạy chúng em trong suốt 4 năm học tại mái trường Đại Học Dân Lập Hải Phòng, các thầy cô giáo trong tổ bộ môn Công Nghệ Thông Tin. Em xin chúc các thầy cô luôn mạnh khỏe, công tác tốt, tiếp tục cống hiến cho sự nghiệp “trồng người” cao quý của dân tộc.

Sự hoàn thiện đề án này cũng là một cách thể hiện tình cảm của em tới gia đình, người thân, thầy cô và bạn bè đã giúp đỡ em cả về vật chất lẫn tinh thần, tạo điều kiện thuận lợi để em hoàn thành. Đặc biệt, em xin gửi lời cảm ơn sâu sắc nhất đến Thầy – Thạc sĩ Vũ Mạnh Khánh– người đã định hướng đề tài, tận tình hướng dẫn chỉ bảo em trong suốt quá trình hoàn thành đề án tốt nghiệp này.

Trong quá trình hoàn thành đề án mặc dù đã cố gắng, song do trình độ chuyên môn và kiến thức còn hạn chế nên những khiếm khuyết trong đề tài này không thể tránh khỏi. Vì vậy, em rất mong nhận được sự cảm thông và góp ý của các thầy cô cũng như bạn đọc để cho đề án của em được hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 2 Tháng 7 Năm 2011

Sinh viên

Vương Bá Ngọc

## MỤC LỤC

<b>DANH MỤC HÌNH VẼ .....</b>	<b>4</b>
<b>LỜI MỞ ĐẦU .....</b>	<b>5</b>
<b>CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ ASM.....</b>	<b>6</b>
1.1 Tổng quan về ASM: .....	6
1.2. Khung chương trình Assembly .....	7
1.2.1 Bộ ký tự của Assembly.....	7
1.2.2 Từ khóa.....	7
1.2.3 Tên tự đặt.....	7
1.2.4 Cấu trúc một lệnh Assembly.....	7
1.2.5 Các dạng hằng dùng trong Assembly.....	8
1.2.6 Khai báo biến và hằng .....	9
1.2.7 Khung của một chương trình Assembly .....	11
1.3. Biên soạn và dịch chương trình Assembly.....	17
<b>CHƯƠNG 2 : HDD-THƯ MỤC CỦA WINDOW .....</b>	<b>18</b>
2.1 Cấu trúc của HDD : .....	18
2.1.1 Tổng quan về ổ cứng : .....	18
2.1.2. Cấu tạo: .....	19
2.1.3. Hoạt động :.....	23
2.1.4. Các công nghệ sử dụng ổ đĩa cứng .....	24
2.2 Cách quản lý bộ nhớ ổ cứng :.....	26
2.2.1 Tốc độ truyền dữ liệu.....	26
2.2.2 Các số thông số về sản phẩm.....	27
2.2.3 Thiết đặt các chế độ hoạt động của đĩa cứng.....	28
2.2.4 Định dạng của phân vùng .....	30
<b>CHƯƠNG 3 : GIỚI THIỆU CHƯƠNG TRÌNH .....</b>	<b>34</b>
3.1 Mô tả bài toán.....	34
3.2 Sơ đồ phân rã chức năng .....	35
3.3 Các hàm và ngắt trong chương trình .....	36
<b>CHƯƠNG 4 : DEMO CHƯƠNG TRÌNH .....</b>	<b>39</b>
4.1. Kiểm tra dung lượng trống của ổ đĩa: .....	39
4.2. Kiểm tra trạng thái của ổ đĩa .....	40
4.3. Đọc bảng FAT: .....	41
4.4. Đọc bootrecord: .....	42
4.5. Hiển thị thư mục .....	43
4.6. Tạo thư mục.....	44

4.7. Xóa thư mục .....	45
<b>KẾT LUẬN .....</b>	<b>46</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>47</b>

## DANH MỤC HÌNH VẼ

Hình 2.1. Track và sector .....	21
Hình 3.1 Sơ đồ phân rã chức năng .....	35
Hình 4.1. Kiểm tra dung lượng trống.....	39
Hình 4.2. Kiểm tra trạng thái ổ đĩa.....	40
Hình 4.3. Đọc bảng Fat .....	41
Hình 4.4. Đọc bootrecord.....	42
Hình 4.5. Hiện thị thư mục.....	43
Hình 4.6. Tạo thư mục .....	44
Hình 4.7. Xóa thư mục.....	45

## LỜI MỞ ĐẦU

Với sự phát triển của kinh tế ngày nay thì máy tính là một thứ không thể thiếu trong xã hội bây giờ, chiếc máy tính bây giờ đã quá quen thuộc với mọi người, để sử dụng máy tính thì đơn giản nhưng để sử dụng đúng cách và bảo vệ máy tính là một điều không hề đơn giản.

Trong máy tính thì ổ cứng là một thứ không thể thiếu đối với tất cả các máy tính, nó lưu trữ dữ liệu, những tài liệu mật, những đoạn video, file nhạc, rất nhiều dữ liệu khác, Ổ đĩa cứng là một thiết bị rất quan trọng trong hệ thống bởi chúng chứa dữ liệu thành quả của một quá trình làm việc của những người sử dụng máy tính

Những sự hư hỏng của các thiết bị khác trong hệ thống máy tính có thể sửa chữa hoặc thay thế được, nhưng dữ liệu bị mất do yếu tố hư hỏng phần cứng của ổ đĩa cứng thường rất khó lấy lại được.

Vì vậy nên em chọn đề tài “ *Lập trình quản lý thư mục trong HDD bằng ngôn ngữ Assembly*” để hiểu rõ hơn về cơ chế lưu trữ dữ liệu của ổ cứng

Đề án gồm 4 chương :

- Chương 1 : Giới thiệu tổng quan về ASM
- Chương 2 : HDD-Thư mục của window
- Chương 3 : Giới thiệu chương trình
- Chương 4 : Demo chương trình

# CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ ASM

## 1.1 Tổng quan về ASM:

-**Ngôn ngữ assembly** (còn gọi là **hợp ngữ**) là một ngôn ngữ bậc thấp được dùng trong việc viết các chương trình máy tính. Ngôn ngữ assembly sử dụng các từ có tính gợi nhớ, các từ viết tắt để giúp ta dễ ghi nhớ các chỉ thị phức tạp và làm cho việc lập trình bằng assembly dễ dàng hơn. Mục đích của việc dùng các từ gợi nhớ là nhằm thay thế việc lập trình trực tiếp bằng ngôn ngữ máy được sử dụng trong các máy tính đầu tiên thường gặp nhiều lỗi và tốn thời gian. Một chương trình viết bằng ngôn ngữ assembly được dịch thành mã máy bằng một chương trình tiện ích được gọi là **assembler** (Một chương trình assembler khác với một trình biên dịch ở chỗ nó chuyển đổi mỗi lệnh của chương trình assembly thành một lệnh. Các chương trình viết bằng ngôn ngữ assembly liên quan rất chặt chẽ đến kiến trúc của máy tính. Điều này khác với ngôn ngữ lập trình bậc cao, ít phụ thuộc vào phần cứng.

-Sở dĩ ta dùng ngôn ngữ lập trình Assembly để viết phần mềm cho hệ vi xử lý là vì nó có các ưu điểm sau:

- Sử dụng trực tiếp tập lệnh của bộ vi xử lý nên quá trình điều hành chức năng rất sát với cấu trúc phần cứng của hệ thống, tận dụng triệt để khả năng của phần cứng mà không một ngôn ngữ lập trình bậc cao nào làm được.

- Có tốc độ thực hiện nhanh hơn nhiều so với các ngôn ngữ bậc cao. Do vậy nó rất thích hợp với các chức năng yêu cầu thời gian thực chẳng hạn như thao tác với các tín hiệu biến đổi nhanh.

Các chương trình viết bằng ngôn ngữ Assembly phải được dịch ra ngôn ngữ máy (dạng nhị phân) vì đây là dạng duy nhất mà hệ vi xử lý có thể hiểu được. Có nhiều chương trình biên dịch nhưng thông dụng nhất hiện nay Macro Assembler của hãng Microsoft và Turbo Assembler của hãng Borland. Chúng ta sẽ sử dụng Macro Assembler 6.0 để biên dịch các chương trình Assembly. Chương trình biên dịch MASM 6.0 có rất nhiều file nhưng tối thiểu cần những file sau:

- TASM.EXE để biên dịch chương trình sang ngôn ngữ máy
- LINK.EXE để liên kết các chương trình và tạo ra một chương trình chạy được có đuôi exe.
- EXE2BIN để chuyển chương trình đuôi exe sang đuôi com

## **1.2. Khung chương trình Assembly**

### **1.2.1 Bộ ký tự của Assembly**

Một ngôn ngữ bất kỳ từ ngôn ngữ giao tiếp của con người tới ngôn ngữ máy tính đều xây dựng trên một bộ ký tự. Các ký tự ghép lại thành các từ có nghĩa gọi là từ vựng. Các từ lại được viết thành các câu tuân theo cú pháp và ngữ pháp của ngôn ngữ để diễn tả hành động sự việc cần thực hiện. Bộ ký tự của Assembly gồm có:

- Các chữ cái latin: 26 chữ hoa A-Z, 26 chữ thường a-z.
- Các chữ số thập phân: '0' - '9'
- Các ký hiệu phép toán, các dấu chấm câu và các ký hiệu đặc biệt: + - \* / @ ? \$ , . : [ ] ( ) < > { } & % ! \ # v.v...
- Các ký tự ngăn cách: space và tab

### **1.2.2 Từ khóa**

Từ khóa là các từ của riêng Assembly như tên các thanh ghi, tên lệnh dạng gọi nhớ của bộ vi xử lý, tên toán tử... Các từ khóa này đòi hỏi người lập trình phải dùng đúng như Assembly quy định. Các từ khóa có thể viết bằng chữ hoa hoặc chữ thường đều được.

### **1.2.3 Tên tự đặt**

Tên là một dãy ký tự dùng để biểu thị tên hằng, tên biến, tên nhãn, tên chương trình con, tên đoạn nhớ... Tên do người lập trình tự đặt nhưng phải tuân theo quy tắc sau:

Quy tắc đặt tên: Tên chỉ gồm chữ cái, chữ số và một số ký tự đặc biệt như ? @ \_ \$ Chữ đầu của tên bắt buộc phải là chữ cái. Trong tên không có dấu cách. Tên có thể dài từ 1 đến 35 ký tự.

### **1.2.4 Cấu trúc một lệnh Assembly**

Một chương trình Assembly bao gồm các dòng lệnh, một dòng lệnh có thể là một lệnh thật dưới dạng gọi nhớ của bộ vi xử lý hoặc một hướng dẫn cho chương trình dịch (assembler directive, đôi khi gọi là lệnh giả). Lệnh thật sẽ được dịch ra mã máy còn lệnh giả thì không được dịch, vì nó chỉ có tác dụng chỉ dẫn cho chương trình dịch thực hiện công việc. Ta có thể viết các dòng lệnh bằng chữ hoa hoặc chữ thường đều được vì chúng được coi là tương đương nhau.

Một dòng lệnh của Assembly có thể có những trường sau (không nhất thiết phải có đủ các trường):

<b>Tên</b>	<b>Mã lệnh</b>	<b>Các toán hạng</b>	<b>Chú giải</b>
------------	----------------	----------------------	-----------------

Ví dụ:

LAP: MOV AH,[BX] ; Copy nội dung của ô nhớ có địa chỉ DS:BX vào AH

Dòng lệnh trên có đủ 4 trường. Trường tên là nhãn LAP, trường mã lệnh là lệnh MOV, trường toán hạng là các thanh ghi AH và BX, trường chú giải đặt sau dấu chấm phẩy MAIN PROC và MAIN ENDP

Hai dòng lệnh này là hai lệnh giả, ở trường tên có tên thủ tục là MAIN, ở trường mã lệnh có lệnh giả PROC và ENDP. Đây là hai lệnh giả để bắt đầu và kết thúc một thủ tục có tên là MAIN.

- Trường tên

Trường tên có thể là tên nhãn, tên biến hoặc tên thủ tục (chương trình con). Các tên và nhãn này sẽ được trình biên dịch gán bằng các địa chỉ cụ thể của ô nhớ. Một nhãn kết thúc bằng dấu hai chấm (:).

- Trường mã lệnh

Chứa các lệnh thật hoặc lệnh giả

- Trường toán hạng

Đối với các lệnh thật thì trường này chứa các toán hạng của lệnh. Tùy từng loại lệnh mà có thể không có, có 1 hoặc 2 toán hạng trong một lệnh.

Đối với các lệnh giả thì trường này chứa các thông tin khác liên quan đến lệnh giả.

- Trường chú giải

Lời giải thích phải được bắt đầu bằng dấu chấm phẩy. Trường chú giải dành cho người lập trình để ghi các lời giải thích cho các lệnh của chương trình, giúp cho người đọc chương trình dễ hiểu các thao tác của chương trình lớn. Khi đọc thấy dấu chấm phẩy, chương trình dịch bỏ qua không dịch từ sau dấu chấm phẩy đến hết dòng. Người lập trình có thể lợi dụng đặc điểm này để loại bỏ một dòng lệnh nào đó trong chương trình.

### **1.2.5 Các dạng hằng dùng trong Assembly**

- Hằng số nhị phân: gồm một dãy các chữ số 0 và 1, kết thúc bằng chữ B.



- Hằng số hex: gồm một dãy các số từ 0 đến 9 và các chữ cái từ A đến F (a đến f), kết thúc bằng chữ H. Đối với các số bắt đầu bằng chữ thì phải thêm 0 đằng trước để báo cho chương trình dịch biết đó là số không phải là tên. Ví dụ: 7AC5H, 0ABH

- Hằng số thập phân: gồm một dãy các số từ 0 đến 9, có hoặc không có chữ D theo sau. Ví dụ: 34 hoặc 34D.

- Hằng ký tự: là một ký tự bất kỳ đặt giữa hai dấu phẩy trên. Ví dụ: 'A'

- Hằng xâu ký tự: là một dãy ký tự bất kỳ đặt giữa hai dấu phẩy trên.

### 1.2.6 Khai báo biến và hằng

#### - Khai báo biến

Biến là tên ô nhớ dùng để cất giữ dữ liệu. Khai báo biến là đặt tên cho ô nhớ và xác định ô nhớ có kích thước 1 byte, 1 từ hay 1 từ kép. Các tên biến sẽ được trình biên dịch gán cho một địa chỉ nhất định trong bộ nhớ khi dịch chương trình.

- Khai báo biến kiểu byte

Tên biến	DB	Giá trị khởi đầu
Ví dụ: B1	DB	4

Ví dụ trên định nghĩa biến kiểu byte có tên là B1 và dành 1 byte bộ nhớ cho nó, trong byte đó có chứa giá trị 4. Nếu không muốn biến chứa giá trị khởi đầu ta có thể dùng toán tử ? vào vị trí giá trị khởi đầu.

Ví dụ: B2            DB ?

Ví dụ trên chỉ định nghĩa biến byte có tên là B2 và dành 1 byte bộ nhớ cho nó.

- Khai báo biến kiểu từ

Tên biến	DW	Giá trị khởi đầu
Ví dụ: W1	DW	42H

Ví dụ này định nghĩa biến từ có tên là W1 và dành 2 byte bộ nhớ cho nó, trong đó chứa giá trị khởi đầu là 42H. Muốn biến không chứa giá trị khởi đầu ta dùng toán tử ? và vị trí giá trị khởi đầu.

Ví dụ: W2            DW ?

- Khai báo biến kiểu từ kép

<b>Tên biến</b>	<b>DD</b>	<b>Giá trị khởi đầu</b>
-----------------	-----------	-------------------------

Ví dụ: DW1	DD	1000
------------	----	------

- Khai báo biến mảng

Biến mảng là biến hình thành từ một dãy liên tiếp các phần tử (ô nhớ) có cùng kiểu byte từ hoặc từ kép. Khai báo biến mảng là đặt tên cho một dãy liên tiếp các byte từ hoặc từ kép trong bộ nhớ đồng thời cung cấp các giá trị ban đầu tương ứng. Số phần tử của mảng được xác định qua số giá trị khởi đầu.

<b>Tên biến mảng</b>	<b>DB/DW/DD</b>	<b>Các giá trị khởi đầu</b>
----------------------	-----------------	-----------------------------

Ví dụ: M1	DB	4,5,6,7,8,9
-----------	----	-------------

Ví dụ trên định nghĩa biến mảng có tên là M1 và dành 6 byte liên tiếp cho nó để chứa các giá trị khởi đầu tương ứng là 4, 5, 6, 7, 8, 9. Phần tử đầu của mảng là 4 và có địa chỉ trùng với địa chỉ của tên biến (M1), phần tử thứ hai là 5 và có địa chỉ là M1+1...Khi chúng ta muốn khởi đầu các phần tử của mảng với cùng một giá trị chúng ta có thể dùng thêm toán tử DUP. Toán tử DUP dùng để lặp lại các dữ liệu với số lần quy định. Cú pháp: Count DUP(Các dữ liệu) -> lặp lại các dữ liệu với số lần Count.

Ví dụ: M2	DB	20 DUP(0)
-----------	----	-----------

M3	DB	20 DUP(?)
----	----	-----------

Ví dụ trên định nghĩa một biến mảng có tên là M2 gồm 20 byte để chứa 20 giá trị khởi đầu bằng 0 và một biến mảng khác có tên là M3 gồm 20 byte nhưng không chứa giá trị khởi đầu.

- Khai báo biến kiểu xâu ký tự

Biến kiểu xâu ký tự là trường hợp đặc biệt của biến mảng kiểu byte, trong đó các phần tử của mảng là các ký tự. Một xâu ký tự có thể định nghĩa bằng các ký tự hoặc bằng mã ASCII của các ký tự đó.

Ví dụ:

Xaukt	DB	'ABCDE'
-------	----	---------

hoặc

Xaukt	DB	41h,42h,43h,44h,45h
-------	----	---------------------

hoặc

Xaukt	DB	41h,42h,'C','D',45h
-------	----	---------------------

### **- Khai báo hằng**

Các hằng trong chương trình Assembly được gán tên để làm cho chương trình dễ đọc hơn. Hằng có thể là kiểu số hoặc kiểu ký tự. Việc gán tên cho hằng được thực hiện bằng lệnh giả EQU như sau:

**Tên hằng    EQU    Giá trị của hằng**

Ví dụ:

CR EQU 0Dh

LF EQU 0Ah

CHAO EQU 'Hello'

Vì lệnh giả EQU không dành chỗ của bộ nhớ cho tên hằng nên ta có thể khai báo hằng ở bất kỳ đâu trong chương trình. Tuy nhiên người ta thường đặt các khai báo hằng trong đoạn dữ liệu.

### **1.2.7 Khung của một chương trình Assembly**

Một chương trình mã máy trong bộ nhớ thường bao gồm các vùng nhớ khác nhau để chứa mã lệnh, chứa dữ liệu của chương trình và một vùng nhớ được dùng làm ngăn xếp phục vụ hoạt động của chương trình. Chương trình viết bằng ngôn ngữ Assembly cũng phải có cấu trúc tương tự để khi dịch nó sẽ tạo ra mã máy có cấu trúc như trên, tức là đoạn mã lệnh sẽ được dịch và để trong vùng nhớ mã lệnh, đoạn dữ liệu sẽ được dịch và để trong vùng nhớ dữ liệu và đoạn ngăn xếp sẽ được dịch và tạo ra vùng nhớ ngăn xếp cho chương trình.

Trước khi tìm hiểu khung của một chương trình Assembly ta xem xét các khai báo có trong chương trình:

#### **-Khai báo quy mô sử dụng bộ nhớ**

Kích thước bộ nhớ dành cho đoạn mã và đoạn dữ liệu trong một chương trình được xác định bằng lệnh giả .MODEL. Lệnh này phải được đặt trước các lệnh khác trong chương trình nhưng đặt sau lệnh giả khai báo loại CPU. Cú pháp:

.MODEL Kiểu\_kích\_thước\_bộ\_nhớ

#### **- Khai báo đoạn ngăn xếp**

Việc khai báo đoạn ngăn xếp là để dành ra một vùng nhớ đủ lớn dùng làm ngăn xếp phục vụ cho hoạt động của chương trình. Cú pháp:

.STACK Kích\_thước

Kích\_thước quyết định số byte dành cho ngăn xếp. Thông thường với 100 - 256 byte là đủ để làm ngăn xếp và ta có thể khai báo kích thước cho ngăn xếp như sau:

```
.STACK 100 hoặc .STACK 100H
```

#### **- Khai báo đoạn dữ liệu**

Đoạn dữ liệu chứa toàn bộ các khai báo biến và hằng của chương trình. Các khai báo trong đoạn dữ liệu đặt sau lệnh giả .DATA

Ví dụ:

```
.DATA
```

```
MSG      DB      'Hello!$'
```

```
B1       DB      100
```

```
CR       EQU     0DH
```

```
LF       EQU     0AH
```

#### **- Khai báo đoạn mã**

Đoạn mã chứa mã lệnh của chương trình, tức là các lệnh của chương trình sẽ viết ở đây. Để bắt đầu đoạn mã ta dùng lệnh giả .CODE

Bên trong đoạn mã, các lệnh của chương trình có thể tổ chức thành chương trình chính và chương trình con như sau:

```
.CODE
```

```
Tên_CTChính      PROC
```

```
;Các lệnh của chương trình chính
```

```
.
```

```
.
```

```
CALL Tên_CTCon ;Gọi chương trình con
```

```
.
```

```
.
```

```
Tên_CTChính      ENDP
```

```
;Khai báo các chương trình con ở đây
```

```
Tên_CTCon PROC
```

;Các lệnh của chương trình con

**RET** ;Trở về

Tên\_CTCon            ENDP

**- Khung chương trình Assembly để dịch ra chương trình .EXE**

.MODEL SMALL

.STACK 100H

.DATA

;Các khai báo biến và hằng ở đây

.CODE

MAIN            PROC

;Khởi đầu cho đoạn DS

MOV AX,@DATA

MOV DS,AX

;Các lệnh của chương trình ở đây

.....

;Trở về DOS dùng hàm 4CH của INT 21H

MOV AH,4CH

INT 21H

MAIN            ENDP

;Các chương trình con (nếu có) khai báo tại đây

END MAIN            ;Kết thúc toàn bộ chương trình

Dòng cuối cùng của chương trình ta dùng lệnh giả END và tiếp theo là MAIN để kết thúc toàn bộ chương trình. Ta có nhận xét rằng MAIN là tên của chương trình chính nhưng về thực chất nó là nơi bắt đầu các lệnh của chương trình trong đoạn mã lệnh.

Khi một chương trình EXE được nạp vào bộ nhớ, DOS sẽ tạo ra một mảng gồm 256 byte làm đoạn mào đầu chương trình (Program Segment Prefix, PSP) dùng để chứa các thông tin liên quan đến chương trình và đặt nó vào ngay phía trước phần bộ nhớ chứa mã lệnh của chương trình. Trong khi đưa các thông số liên quan

đến chương trình vào PSP, DOS đã sử dụng đến các thanh ghi DS và ES. Do vậy DS và ES không chứa giá trị địa chỉ của đoạn dữ liệu của chương trình. Để chương trình có thể chạy đúng ta phải có các lệnh khởi tạo cho thanh ghi đoạn DS (hoặc cả ES nếu cần) để chứa địa chỉ đoạn dữ liệu của chương trình.

```
MOV AX,@DATA
```

```
MOV DS,AX
```

```
;MOV ES,AX ;Nếu cần trong đó @DATA là địa chỉ của đoạn dữ liệu.
```

### **- Khung chương trình Assembly để dịch ra chương trình .COM**

Chương trình đuôi .COM ngắn gọn và đơn giản hơn nhiều so với chương trình đuôi .EXE

Trong chương trình đuôi .COM, đoạn mã, đoạn dữ liệu và đoạn ngăn xếp được gộp lại trong một đoạn duy nhất là đoạn mã. Việc tạo ra tệp này không những tiết kiệm được thời gian và bộ nhớ khi chạy chương trình mà còn tiết kiệm được cả không gian nhớ khi lưu trữ chương trình trên ổ đĩa.

Khung của chương trình Assembly để dịch ra đuôi .COM như sau:

```
.MODEL TINY
```

```
.CODE
```

```
ORG 100H
```

```
START: JMP CONTINUE
```

```
;Các khai báo biến và hằng để tại đây
```

```
CONTINUE:
```

```
MAIN PROC
```

```
;Các lệnh của chương trình chính để tại đây
```

```
;Trở về DOS
```

```
INT 20H
```

```
MAIN ENDP
```

```
;Các chương trình con (nếu có) khai báo ở đây
```

```
END START
```

Ta nhận thấy ở ngay đầu đoạn mã là lệnh giả ORG 100H dùng để gán địa chỉ bắt đầu của chương trình tại 100h trong đoạn mã, chừa lại vùng nhớ 256 byte (từ địa chỉ 0 đến 255) cho đoạn mào đầu chương trình (PSP). Lệnh JMP sau nhãn START dùng để nhảy qua toàn bộ phần bộ nhớ dành cho việc khai báo dữ liệu. Đích của lệnh nhảy này là đầu chương trình chính. Khi kết thúc chương trình COM, để trở về DOS ta dùng ngắt INT 20H của DOS để làm cho chương trình gọn hơn.

Để kết thúc toàn bộ chương trình ta dùng lệnh giả END theo sau là nhãn START, vì START tương ứng với địa chỉ lệnh đầu tiên của chương trình trong đoạn mã.

Ví dụ: Chương trình hiện lên màn hình dòng chữ CHAO CAC BAN

```
.MODEL TINY.CODE  
  
ORG 100H  
  
START: JMP CONTINUE  
  
CRLF DB 13,10,'$'  
  
CHAO DB 'CHAO CAC BAN!$'  
  
CONTINUE:  
  
MAIN PROC  
  
;Xuong dong moi  
  
MOV AH,9  
  
LEA DX,CRLF  
  
INT 21H  
  
;Hien thi loi chao  
  
MOV AH,9  
  
LEA DX,CHAO  
  
INT 21H  
  
;Xuong dong moi  
  
MOV AH,9  
  
LEA DX,CRLF  
  
INT 21H
```

```
;Tro ve DOS  
INT 20H  
MAIN ENDP  
END START
```



### 1.3. Biên soạn và dịch chương trình Assembly

Để viết và dịch các chương trình Assembly ta theo các bước sau:

#### **Bước 1:** Soạn thảo chương trình

Dùng các phần mềm soạn thảo văn bản dạng TEXT (như NC, PASCAL, C) để tạo ra tệp văn bản chương trình Assembly. Sau đó ghi tệp chương trình Assembly ra đĩa với đuôi .ASM

#### **Bước 2:** Dịch chương trình sang ngôn ngữ máy

Dùng chương trình dịch MASM để dịch tệp chương trình đuôi .ASM sang mã máy dưới dạng tệp đuôi .OBJ. Nếu trong bước này chương trình có lỗi về cú pháp thì chương trình dịch sẽ báo lỗi và ta phải quay lại Bước 1 để sửa. Cách làm như sau: giả sử chương trình MASM nằm trên thư mục gốc ổ C, dấu nhắc của DOS là C:\>, khi đó từ dấu nhắc của DOS gõ :

```
MASM          Tên tệp chương trình; ↵
```

Dấu chấm phẩy sau tên tệp chương trình để báo cho MASM chỉ tạo tệp .OBJ, không tạo ra các tệp khác. Tên tệp chương trình có thể gõ đủ cả đuôi .ASM hoặc không gõ cũng được.

**Bước 3:** Liên kết các tệp đuôi .OBJ để tạo thành một tệp chương trình chạy được đuôi .EXE Cách làm như sau: giả sử chương trình liên kết LINK nằm trên thư mục gốc ổ C, dấu nhắc của DOS là C:\>, khi đó từ dấu nhắc của DOS ta gõ lệnh :

```
LINK Têntệp1 + Têntệp2 + ...; ↵
```

Chương trình liên kết sẽ lấy tên tệp đầu tiên (Têntệp1) để đặt tên cho tệp đuôi .EXE cuối cùng. Dấu chấm phẩy sau cùng để báo cho chương trình LINK không hỏi tên các tệp.

**Bước 4:** Nếu chương trình viết để dịch ra đuôi .COM thì ta phải dùng chương trình EXE2BIN của DOS để dịch tiếp tệp .EXE ra tệp chương trình chạy được đuôi .COM Cách làm như sau: giả sử chương trình EXE2BIN nằm trên thư mục gốc ổ C, dấu nhắc của DOS là C:\>, khi đó từ dấu nhắc của DOS ta gõ lệnh :

```
EX Têntệp.EXE Têntệp.COM ↵
```

## CHƯƠNG 2 : HDD-THƯ MỤC CỦA WINDOW

### 2.1 Cấu trúc của HDD :

#### 2.1.1 Tổng quan về ổ cứng :

-**Ổ đĩa cứng**, hay còn gọi là **ổ cứng** (tiếng Anh: *Hard Disk Drive*, viết tắt: **HDD**) là thiết bị dùng để lưu trữ dữ liệu trên bề mặt các tấm đĩa hình tròn phủ vật liệu từ tính.

Ổ đĩa cứng là loại bộ nhớ "không thay đổi" (*non-volatile*), có nghĩa là chúng không bị mất dữ liệu khi ngừng cung cấp nguồn điện cho chúng.

Ổ đĩa cứng là một thiết bị rất quan trọng trong hệ thống bởi chúng chứa dữ liệu thành quả của một quá trình làm việc của những người sử dụng máy tính. Những sự hư hỏng của các thiết bị khác trong hệ thống máy tính có thể sửa chữa hoặc thay thế được, nhưng dữ liệu bị mất do yếu tố hư hỏng phần cứng của ổ đĩa cứng thường rất khó lấy lại được.

*Ổ đĩa cứng là một khối duy nhất, các đĩa cứng được lắp ráp cố định trong ổ ngay từ khi sản xuất nên không thể thay thế được các "đĩa cứng" như với cách hiểu như đối với ổ đĩa mềm hoặc ổ đĩa quang.*

Ổ cứng thường được gắn liền với máy tính để lưu trữ dữ liệu cho dù chúng xuất hiện muộn hơn so với những chiếc máy tính đầu tiên.

Với sự phát triển nhanh chóng của công nghệ, ổ đĩa cứng ngày nay có kích thước càng nhỏ đi đến các chuẩn thông dụng với dung lượng thì ngày càng tăng lên. Những thiết kế đầu tiên ổ đĩa cứng chỉ dành cho các máy tính thì ngày nay ổ đĩa cứng còn được sử dụng trong các thiết bị điện tử khác như máy nghe nhạc kỹ thuật số, máy ảnh số, điện thoại di động thông minh (*SmartPhone*), máy quay phim kỹ thuật số, thiết bị kỹ thuật số hỗ trợ cá nhân...

Không chỉ tuân theo các thiết kế ban đầu, ổ đĩa cứng đã có những bước tiến công nghệ nhằm giúp lưu trữ và truy xuất dữ liệu nhanh hơn: ví dụ sự xuất hiện của các ổ đĩa cứng lai giúp cho hệ điều hành hoạt động tối ưu hơn, giảm thời gian khởi động của hệ thống, tiết kiệm năng lượng, sự thay đổi phương thức ghi dữ liệu trên các đĩa từ làm cho dung lượng mỗi ổ đĩa cứng tăng lên đáng kể.

### 2.1.2. Cấu tạo:

Ổ đĩa cứng gồm các thành phần, bộ phận có thể liệt kê cơ bản và giải thích sơ bộ như sau:

**Cụm đĩa:** Bao gồm toàn bộ các đĩa, trục quay và động cơ.

- Đĩa từ.
- Trục quay: truyền chuyển động của đĩa từ.
- Động cơ: Được gắn đồng trục với trục quay và các đĩa.

#### Cụm đầu đọc

- Đầu đọc (*head*): Đầu đọc/ghi dữ liệu
- Cần di chuyển đầu đọc (*head arm* hoặc *actuator arm*).

#### Cụm mạch điện

- Mạch điều khiển: có nhiệm vụ điều khiển động cơ đồng trục, điều khiển sự di chuyển của cần di chuyển đầu đọc để đảm bảo đến đúng vị trí trên bề mặt đĩa.

- Mạch xử lý dữ liệu: dùng để xử lý những dữ liệu đọc/ghi của ổ đĩa cứng.

- Bộ nhớ đệm (*cache* hoặc *buffer*): là nơi tạm lưu dữ liệu trong quá trình đọc/ghi dữ liệu. Dữ liệu trên bộ nhớ đệm sẽ mất đi khi ổ đĩa cứng ngừng được cấp điện.

- Đầu cắm nguồn cung cấp điện cho ổ đĩa cứng.

- Đầu kết nối giao tiếp với máy tính.

- Các cầu đấu thiết đặt (tạm dịch từ *jumper*) thiết đặt chế độ làm việc của ổ đĩa cứng: Lựa chọn chế độ làm việc của ổ đĩa cứng (SATA 150 hoặc SATA 300) hay thứ tự trên các kênh trên giao tiếp IDE (master hay slave hoặc tự lựa chọn), lựa chọn các thông số làm việc khác...

#### Vỏ đĩa cứng:

Vỏ ổ đĩa cứng gồm các phần: Phần đế chứa các linh kiện gắn trên nó, phần nắp đậy lại để bảo vệ các linh kiện bên trong.

Vỏ ổ đĩa cứng có chức năng chính nhằm định vị các linh kiện và đảm bảo độ kín khít để không cho phép bụi được lọt vào bên trong của ổ đĩa cứng.

Ngoài ra, vỏ đĩa cứng còn có tác dụng chịu đựng sự va chạm (ở mức độ thấp) để bảo vệ ổ đĩa cứng.

*Do đầu từ chuyển động rất sát mặt đĩa nên nếu có bụi lọt vào trong ổ đĩa cứng cũng có thể làm xước bề mặt, mất lớp từ và hư hỏng từng phần (xuất hiện các khối hư hỏng (bad block))... Thành phần bên trong của ổ đĩa cứng là không khí có độ sạch cao, để đảm bảo áp suất cân bằng giữa môi trường bên trong và bên ngoài, trên vỏ bảo vệ có các hệ lỗ thoáng đảm bảo cân bụi và cân bằng áp suất.*

### **Đĩa từ**

Đĩa từ (*platter*): Đĩa thường cấu tạo bằng nhôm hoặc thủy tinh, trên bề mặt được phủ một lớp vật liệu từ tính là nơi chứa dữ liệu. Tùy theo hãng sản xuất mà các đĩa này được sử dụng một hoặc cả hai mặt trên và dưới. Số lượng đĩa có thể nhiều hơn một, phụ thuộc vào dung lượng và công nghệ của mỗi hãng sản xuất khác nhau.

Mỗi đĩa từ có thể sử dụng hai mặt, đĩa cứng có thể có nhiều đĩa từ, chúng gắn song song, quay đồng trục, cùng tốc độ với nhau khi hoạt động.

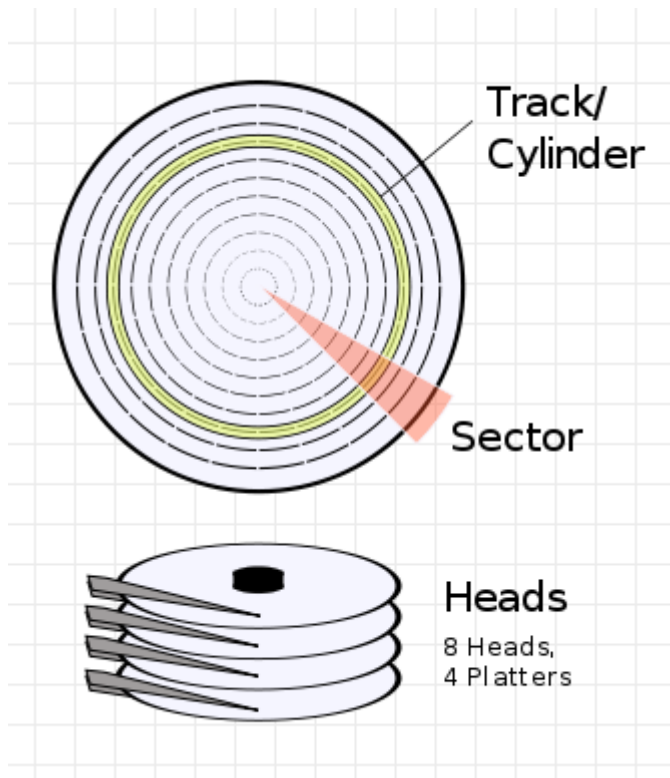
### **Track**

Trên một mặt làm việc của đĩa từ chia ra nhiều vòng tròn đồng tâm thành các **track**.

Track có thể được hiểu đơn giản giống các rãnh ghi dữ liệu giống như các đĩa nhựa (ghi âm nhạc trước đây) nhưng sự cách biệt của các rãnh ghi này không có các gờ phân biệt và chúng là các vòng tròn đồng tâm chứ không nối tiếp nhau thành dạng xoắn tròn ốc như đĩa nhựa. Track trên ổ đĩa cứng không cố định từ khi sản xuất, chúng có thể thay đổi vị trí khi định dạng cấp thấp ổ đĩa (*low format*).

*Khi một ổ đĩa cứng đã hoạt động quá nhiều năm liên tục, khi kết quả kiểm tra bằng các phần mềm cho thấy xuất hiện nhiều khối hư hỏng (bad block) thì có nghĩa là phần cơ của nó đã rơ rão và làm việc không chính xác như khi mới sản xuất, lúc này thích hợp nhất là format cấp thấp cho nó để tương thích hơn với chế độ làm việc của phần cơ*

## Sector



Hình 2.1. Track và sector

Trên *track* chia thành những phần nhỏ bằng các đoạn hướng tâm thành các **sector**. Các sector là phần nhỏ cuối cùng được chia ra để chứa dữ liệu. Theo chuẩn thông thường thì một sector chứa dung lượng 512 byte.

Số sector trên các track là khác nhau từ phần rìa đĩa vào đến vùng tâm đĩa, các ổ đĩa cứng đều chia ra hơn 10 vùng mà trong mỗi vùng có số sector/track bằng nhau.

Bảng sau cho thấy các khu vực với các thông số khác nhau và sự ảnh hưởng của chúng đến tốc độ truyền dữ liệu của ổ cứng Các khu vực ghi dữ liệu của ổ đĩa cứng Hitachi Travelstar 7K60 2,5".

## Cylinder

Tập hợp các *track* cùng cùng bán kính (cùng số hiệu trên) ở các mặt đĩa khác nhau thành các **cylinder**. Nói một cách chính xác hơn thì: khi đầu đọc/ghi đầu tiên làm việc tại một track nào thì tập hợp toàn bộ các track trên các bề mặt đĩa còn lại mà các đầu đọc còn lại đang làm việc tại đó gọi là cylinder (*cách giải thích này*

*chính xác hơn bởi có thể xảy ra thường hợp các đầu đọc khác nhau có khoảng cách đến tâm quay của đĩa khác nhau do quá trình chế tạo).*

Trên một ổ đĩa cứng có nhiều cylinder bởi có nhiều track trên mỗi mặt đĩa từ.

### **Trục quay**

Trục quay là trục để gắn các đĩa từ lên nó, chúng được nối trực tiếp với động cơ quay đĩa cứng. Trục quay có nhiệm vụ truyền chuyển động quay từ động cơ đến các đĩa từ.

Trục quay thường chế tạo bằng các vật liệu nhẹ (như hợp kim nhôm) và được chế tạo tuyệt đối chính xác để đảm bảo trọng tâm của chúng không được sai lệch - bởi chỉ một sự sai lệch nhỏ có thể gây lên sự rung lắc của toàn bộ đĩa cứng khi làm việc ở tốc độ cao, dẫn đến quá trình đọc/ghi không chính xác.

### **Đầu đọc/ghi**

Đầu đọc đơn giản được cấu tạo gồm lõi ferit (trước đây là lõi sắt) và cuộn dây (giống như nam châm điện). Gần đây các công nghệ mới hơn giúp cho ổ đĩa cứng hoạt động với mật độ xít chặt hơn như: chuyển các hạt từ sắp xếp theo phương vuông góc với bề mặt đĩa nên các đầu đọc được thiết kế nhỏ gọn và phát triển theo các ứng dụng công nghệ mới.

Đầu đọc trong đĩa cứng có công dụng đọc dữ liệu dưới dạng từ hoá trên bề mặt đĩa từ hoặc từ hoá lên các mặt đĩa khi ghi dữ liệu.

Số đầu đọc ghi luôn bằng số mặt hoạt động được của các đĩa cứng, có nghĩa chúng nhỏ hơn hoặc bằng hai lần số đĩa (nhỏ hơn trong trường hợp ví dụ hai đĩa nhưng chỉ sử dụng 3 mặt).

### **Cần di chuyển đầu đọc/ghi**

Cần di chuyển đầu đọc/ghi là các thiết bị mà đầu đọc/ghi gắn vào nó. Cần có nhiệm vụ di chuyển theo phương song song với các đĩa từ ở một khoảng cách nhất định, dịch chuyển và định vị chính xác đầu đọc tại các vị trí từ mép đĩa đến vùng phía trong của đĩa (phía trục quay).

Các cần di chuyển đầu đọc được di chuyển đồng thời với nhau do chúng được gắn chung trên một trục quay (đồng trục), có nghĩa rằng khi việc đọc/ghi dữ liệu trên bề mặt (trên và dưới nếu là loại hai mặt) ở một vị trí nào thì chúng cũng hoạt động cùng vị trí tương ứng ở các bề mặt đĩa còn lại.

Sự di chuyển cần có thể thực hiện theo hai phương thức:

- Sử dụng động cơ bước để truyền chuyển động.
- Sử dụng cuộn cảm để di chuyển cần bằng lực từ.

### 2.1.3. Hoạt động :

#### Giao tiếp với máy tính

Toàn bộ cơ chế đọc/ghi dữ liệu chỉ được thực hiện khi máy tính (hoặc các thiết bị sử dụng ổ đĩa cứng) có yêu cầu truy xuất dữ liệu hoặc cần ghi dữ liệu vào ổ đĩa cứng. Việc thực hiện giao tiếp với máy tính do bo mạch của ổ đĩa cứng đảm nhiệm.

Ta biết rằng máy tính làm việc khác nhau theo từng phiên làm việc, từng nhiệm vụ mà không theo một kịch bản nào, do đó quá trình đọc và ghi dữ liệu luôn luôn xảy ra, do đó các tập tin luôn bị thay đổi, xáo trộn vị trí. Từ đó dữ liệu trên bề mặt đĩa cứng không được chứa một cách liên tục mà chúng nằm rải rác khắp nơi trên bề mặt vật lý. Một mặt khác máy tính có thể xử lý đa nhiệm (thực hiện nhiều nhiệm vụ trong cùng một thời điểm) nên cần phải truy cập đến các tập tin khác nhau ở các thư mục khác nhau.

Như vậy cơ chế đọc và ghi dữ liệu ở ổ đĩa cứng không đơn thuần thực hiện từ theo tuần tự mà chúng có thể truy cập và ghi dữ liệu ngẫu nhiên tại bất kỳ điểm nào trên bề mặt đĩa từ, đó là đặc điểm khác biệt nổi bật của ổ đĩa cứng so với các hình thức lưu trữ truy cập tuần tự (như băng từ).

Thông qua giao tiếp với máy tính, khi giải quyết một tác vụ, CPU sẽ đòi hỏi dữ liệu (nó sẽ hỏi tuần tự các bộ nhớ khác trước khi đến đĩa cứng mà thứ tự thường là cache L1-> cache L2 ->RAM) và đĩa cứng cần truy cập đến các dữ liệu chứa trên nó. Không đơn thuần như vậy CPU có thể đòi hỏi nhiều hơn một tập tin dữ liệu tại một thời điểm, khi đó sẽ xảy ra các trường hợp:

1. Ổ đĩa cứng chỉ đáp ứng một yêu cầu truy cập dữ liệu trong một thời điểm, các yêu cầu được đáp ứng tuần tự.
2. Ổ đĩa cứng đồng thời đáp ứng các yêu cầu cung cấp dữ liệu theo phương thức riêng của nó.

Trước đây đa số các ổ đĩa cứng đều thực hiện theo phương thức 1, có nghĩa là chúng chỉ truy cập từng tập tin cho CPU. Ngày nay các ổ đĩa cứng đã được tích hợp các bộ nhớ đệm (*cache*) cùng các công nghệ riêng của chúng (TCQ, NCQ) giúp tối ưu cho hành động truy cập dữ liệu trên bề mặt đĩa nên ổ đĩa cứng sẽ thực hiện theo phương thức thứ 2 nhằm tăng tốc độ chung cho toàn hệ thống.

## **Đọc và ghi dữ liệu trên bề mặt đĩa**

Sự hoạt động của đĩa cứng cần thực hiện đồng thời hai chuyển động: Chuyển động quay của các đĩa và chuyển động của các đầu đọc.

Sự quay của các đĩa từ được thực hiện nhờ các động cơ gắn cùng trục (với tốc độ rất lớn: từ 3600 rpm cho đến 15.000 rpm) chúng thường được quay ổn định tại một tốc độ nhất định theo mỗi loại ổ đĩa cứng.

Khi đĩa cứng quay đều, cần di chuyển đầu đọc sẽ di chuyển đến các vị trí trên các bề mặt chứa phủ vật liệu từ theo phương bán kính của đĩa. Chuyển động này kết hợp với chuyển động quay của đĩa có thể làm đầu đọc/ghi tới bất kỳ vị trí nào trên bề mặt đĩa.

Tại các vị trí cần đọc ghi, đầu đọc/ghi có các bộ cảm biến với điện trường để đọc dữ liệu (và tương ứng: phát ra một điện trường để xoay hướng các hạt từ khi ghi dữ liệu). Dữ liệu được ghi/đọc đồng thời trên mọi đĩa. Việc thực hiện phân bố dữ liệu trên các đĩa được thực hiện nhờ các mạch điều khiển trên bo mạch của ổ đĩa cứng.

### **2.1.4. Các công nghệ sử dụng ổ đĩa cứng**

#### **S.M.A.R.T**

S.M.A.R.T (**S**elf-**M**onitoring, **A**nalysis, and **R**eporting **T**echnology) là công nghệ tự động giám sát, chuẩn đoán và báo cáo các hư hỏng có thể xuất hiện của ổ đĩa cứng để thông qua BIOS, các phần mềm thông báo cho người sử dụng biết trước sự hư hỏng để có các hành động chuẩn bị đối phó (như sao chép dữ liệu dự phòng hoặc có các kế hoạch thay thế ổ đĩa cứng mới).

Trong thời gian gần đây S.M.A.R.T được coi là một tiêu chuẩn quan trọng trong ổ đĩa cứng. S.M.A.R.T chỉ thực sự giám sát những sự thay đổi, ảnh hưởng của phần cứng đến quá trình lỗi xảy ra của ổ đĩa cứng (mà theo hãng Seagate thì sự hư hỏng trong đĩa cứng chiếm tới 60% xuất phát từ các vấn đề liên quan đến cơ khí): Chúng có thể bao gồm những sự hư hỏng theo thời gian của phần cứng: đầu đọc/ghi (mất kết nối, khoảng cách làm việc với bề mặt đĩa thay đổi), động cơ (xuống cấp, rơ rã), bo mạch của ổ đĩa (hư hỏng linh kiện hoặc làm việc sai).

*S.M.A.R.T không nên được hiểu là từ "smart" bởi chúng không làm cải thiện đến tốc độ làm việc và truyền dữ liệu của ổ đĩa cứng. Người sử dụng có thể bật (enable) hoặc tắt (disable) chức năng này trong BIOS (tuy nhiên không phải BIOS của hãng nào cũng hỗ trợ việc can thiệp này).*



## Ổ cứng lai

Ổ cứng lai (*hybrid hard disk drive*) là các ổ đĩa cứng thông thường được gắn thêm các phần bộ nhớ flash trên bo mạch của ổ đĩa cứng. Cụm bộ nhớ này hoạt động khác với cơ chế làm việc của bộ nhớ đệm (cache) của ổ đĩa cứng: Dữ liệu chứa trên chúng không bị mất đi khi mất điện.

Trong quá trình làm việc của ổ cứng lai, vai trò của phần bộ nhớ flash như sau:

- Lưu trữ trung gian dữ liệu trước khi ghi vào đĩa cứng, chỉ khi máy tính đã đưa các dữ liệu đến một mức nhất định (tùy từng loại ổ cứng lai) thì ổ đĩa cứng mới tiến hành ghi dữ liệu vào các đĩa từ, điều này giúp sự vận hành của ổ đĩa cứng tối hiệu quả và tiết kiệm điện năng hơn nhờ việc không phải thường xuyên hoạt động.
- Giúp tăng tốc độ giao tiếp với máy tính: Việc đọc dữ liệu từ bộ nhớ flash nhanh hơn so với việc đọc dữ liệu tại các đĩa từ.
- Giúp hệ điều hành khởi động nhanh hơn nhờ việc lưu các tập tin khởi động của hệ thống lên vùng bộ nhớ flash.
- Kết hợp với bộ nhớ đệm của ổ đĩa cứng tạo thành một hệ thống hoạt động hiệu quả.

Những ổ cứng lai được sản xuất hiện nay thường sử dụng bộ nhớ flash với dung lượng khiêm tốn ở 256 MB bởi chịu áp lực của vấn đề giá thành sản xuất. Do sử dụng dung lượng nhỏ như vậy nên chưa cải thiện nhiều đến việc giảm thời gian khởi động hệ điều hành, dẫn đến nhiều người sử dụng chưa cảm thấy hài lòng với chúng. Tuy nhiên người sử dụng thường khó nhận ra sự hiệu quả của chúng khi thực hiện các tác vụ thông thường hoặc việc tiết kiệm năng lượng của chúng.

Hiện tại (2007) ổ cứng lai có giá thành khá đắt (khoảng 300 USD cho dung lượng 32 GB) nên chúng mới được sử dụng trong một số loại máy tính xách tay cao cấp. Trong tương lai, các ổ cứng lai có thể tích hợp đến vài GB dung lượng bộ nhớ flash sẽ khiến sự so sánh giữa chúng với các ổ cứng truyền thống sẽ trở lên khác biệt hơn.

## 2.2 Cách quản lý bộ nhớ ổ cứng :

### 2.2.1 Tốc độ truyền dữ liệu

Tốc độ của các chuẩn giao tiếp không có nghĩa là ổ đĩa cứng có thể đáp ứng đúng theo tốc độ của nó, đa phần tốc độ truyền dữ liệu trên các chuẩn giao tiếp thấp hơn so với thiết kế của nó bởi chúng gặp các rào cản trong vấn đề công nghệ chế tạo.

Các thông số sau ảnh hưởng đến tốc độ truyền dữ liệu của ổ đĩa cứng:

- Tốc độ quay của đĩa từ.
- Số lượng đĩa từ trong ổ đĩa cứng: bởi càng nhiều đĩa từ thì số lượng đầu đọc càng lớn, khả năng đọc/ghi của đồng thời của các đầu từ tại các mặt đĩa càng nhiều thì lượng dữ liệu đọc/ghi càng lớn hơn.
- Công nghệ chế tạo: Mật độ sít chặt của các track và công nghệ ghi dữ liệu trên bề mặt đĩa (phương từ song song hoặc vuông góc với bề mặt đĩa): dẫn đến tốc độ đọc/ghi cao hơn.
- Dung lượng bộ nhớ đệm: Ảnh hưởng đến tốc độ truyền dữ liệu tức thời trong một thời điểm.

### **Độ ồn**

Độ ồn của ổ đĩa cứng là thông số được tính bằng dB, chúng được đo khi ổ đĩa cứng đang làm việc bình thường.

Ổ đĩa cứng với các đặc trưng hoạt động là các chuyển động cơ khí của các đĩa từ và cần di chuyển đầu đọc, do đó chúng không tránh khỏi phát tiếng ồn. Do ổ đĩa cứng thường có độ ồn thấp hơn nhiều so với bất kỳ một quạt làm mát hệ thống nào đang làm việc nên người sử dụng có thể không cần quan tâm đến thông số này.

*Những tiếng “lắc rắc” nhỏ phát ra trong quá trình làm việc của ổ cứng một cách không đều đặn được sinh ra bởi cần đỡ đầu đọc/ghi di chuyển và dừng đột ngột tại các vị trí cần định vị để làm việc. Âm thanh này có thể giúp người sử dụng biết được trạng thái làm việc của ổ đĩa cứng mà không cần quan sát đèn trạng thái HDD.*

### **Chu trình di chuyển**

Chu trình di chuyển của cần đọc/ghi (*Load/Unload cycle*) được tính bằng số lần chúng khởi động từ vị trí an toàn đến vùng làm việc của bề mặt đĩa cứng và

ngược lại. Thông số này chỉ một số hữu hạn những lần di chuyển mà có thể sau số lần đó ổ đĩa cứng có thể gặp lỗi hoặc hư hỏng.

*Sau mỗi phiên làm việc (tắt máy), các đầu từ được di chuyển đến một vị trí an toàn nằm ngoài các đĩa từ nhằm tránh sự va chạm có thể gây xước bề mặt lớp từ tính, một số ổ đĩa có thiết kế cần di chuyển đầu đọc tự động di chuyển về vị trí an toàn sau khi ngừng cấp điện đột ngột. Nhiều người sử dụng năng động có thói quen ngắt điện trong một phiên làm việc trên nền DOS (bởi không có sự tắt máy chính thống) rồi tháo ổ đĩa cứng cho các công việc khác, quá trình di chuyển có thể gây va chạm và làm xuất hiện các khối hư hỏng (bad block).*

Chu trình di chuyển là một thông số lớn hơn số lần khởi động máy tính (hoặc các thiết bị sử dụng ổ đĩa cứng) bởi trong một phiên làm việc, ổ đĩa cứng có thể được chuyển sang chế độ tạm nghỉ (stand by) để tiết kiệm điện năng nhiều lần.

### **Chịu đựng sốc**

Chịu đựng sốc (*Shock - half sine wave*): Sốc (hình thức rung động theo nửa chu kỳ sóng, thường được hiểu là việc giao động từ một vị trí cân bằng đến một giá trị cực đại, sau đó lại trở lại vị trí ban đầu) nói đến khả năng chịu đựng sốc của ổ đĩa cứng khi làm việc.

Với các ổ cứng cho máy tính xách tay hoặc các thiết bị kỹ thuật số hỗ trợ cá nhân hay các ổ đĩa cứng ngoài thì thông số này càng cao càng tốt, với các ổ đĩa cứng gắn cho máy tính cá nhân để bàn thì thông số này ít được coi trọng khi so sánh lựa chọn giữa các loại ổ cứng bởi chúng đã được gắn cố định nên hiếm khi xảy ra sốc.

### **2.2.2 Các số thông số về sản phẩm**

Phản dưới đây giải thích một số thông số khác của các ổ đĩa cứng.

**Model:** Ký hiệu về kiểu sản phẩm của ổ đĩa cứng, model có thể được sử dụng chung cho một lô sản phẩm cùng loại có các đặc tính và thông số giống như nhau. Thông thường mỗi hãng có một cách ký hiệu riêng về thông số model để có thể giải thích sơ qua về một số thông số trên ổ đĩa cứng đó.

**Serial number:** Mã số sản phẩm, mỗi ổ đĩa cứng có một số hiệu này riêng. Thông số này thường chứa đựng thông tin đã được quy ước riêng của hãng sản xuất về thời gian sản xuất hoặc đơn thuần chỉ là thứ tự sản phẩm khi được sản xuất.

**Firmware revision:** Thông số về phiên bản firmware đang sử dụng hiện thời của ổ đĩa cứng. Thông số này có thể thay đổi nếu người sử dụng nâng cấp các phiên bản firmware của ổ đĩa cứng (nhưng việc nâng cấp này thường rất hiếm khi xảy ra).

*Một số hãng sản xuất phần mềm có thể sử dụng các thông số trên của ổ đĩa cứng để nhận dạng tình trạng bản quyền của phần mềm trên duy nhất một máy tính, tuy nhiên cách này không được áp dụng rộng rãi do việc đăng ký phức tạp, không thuận tiện cho quá trình nâng cấp ổ đĩa cứng của người sử dụng.*

### 2.2.3 Thiết đặt các chế độ hoạt động của đĩa cứng

#### Thiết đặt phần cứng thông qua cầu đầu

Cầu đầu (tạm dịch từ *jumper*) là một bộ phận nhỏ trên ổ đĩa cứng, chúng có tác dụng thiết đặt chế độ làm việc của các ổ đĩa cứng.

#### Thiết đặt kênh

Lựa chọn các kênh trên cable IDE: Các ổ đĩa cứng theo chuẩn giao tiếp ATA thường sử dụng hai kênh (trên cùng một cáp truyền dữ liệu), chúng có thể được đặt là kênh chính (*Master*) hoặc kênh phụ (*Slave*). Việc thiết lập chỉ đơn giản cần cắm các cầu đầu vào đúng vị trí của chúng trên các chân cắm. Về vị trí, chúng luôn được hướng dẫn trên phần nhãn hoặc viết tắt cạnh cầu đầu như sau: MA (hoặc chỉ M): Master, SL (hoặc chỉ S): Slave; CS (hoặc chỉ C): Cable select (tự động lựa chọn theo cáp truyền dữ liệu).

*Để giúp ổ đĩa cứng hoạt động tốt hơn, nên chọn các ổ đĩa cứng chứa các phân vùng có cài hệ điều hành làm kênh chính, các ổ đĩa cứng vật lý có tính dùng phụ, dùng cho lưu trữ hoặc các tập tin không được truy cập thường xuyên nên đặt tại ổ phụ (slave).*

#### Thiết đặt chuẩn giao tiếp

Một số ổ đĩa cứng sử dụng giao tiếp SATA thế hệ thứ 2 (300 MBps) có thể hoạt động phù hợp hơn với bo mạch chủ chỉ hỗ trợ giao tiếp SATA thế hệ đầu tiên (150 MBps) bằng cách đổi các cầu đầu thiết đặt. Hướng dẫn về cách đổi có thể được ghi trên nhãn đĩa hoặc chỉ có thể tìm thấy trong các phần hướng dẫn tại website của hãng sản xuất.

## Thiết đặt phần mềm

Thiết đặt phần mềm ở đây là các cài đặt, phân hoạch trên các ổ đĩa cứng giúp cho ổ đĩa cứng làm việc. Trong phạm vi bài viết về Ổ đĩa cứng, các mục dưới đây được trình bày tóm lược.

### Phân vùng (Partition)

Phân vùng (*partition*): là tập hợp các vùng ghi nhớ dữ liệu trên các cylinder gần nhau với dung lượng theo thiết đặt của người sử dụng để sử dụng cho các mục đích sử dụng khác nhau.

Sự phân chia phân vùng giúp cho ổ đĩa cứng có thể định dạng các loại tập tin khác nhau để có thể cài đặt nhiều hệ điều hành đồng thời trên cùng một ổ đĩa cứng. Ví dụ trong một ổ đĩa cứng có thể thiết lập một phân vùng có định dạng FAT/FAT32 cho hệ điều hành Windows 9X/Me và một vài phân vùng NTFS cho hệ điều hành Windows NT/2000/XP/Vista với lợi thế về bảo mật trong định dạng loại này (mặc dù các hệ điều hành này có thể sử dụng các định dạng cũ hơn).

Phân chia phân vùng không phải là điều bắt buộc đối với các ổ đĩa cứng để nó làm việc (một vài hãng sản xuất máy tính cá nhân nguyên chiếc chỉ thiết đặt một phân vùng duy nhất khi cài sẵn các hệ điều hành vào máy tính khi bán ra), chúng chỉ giúp cho người sử dụng có thể cài đặt đồng thời nhiều hệ điều hành trên cùng một máy tính hoặc giúp việc quản lý các nội dung, lưu trữ, phân loại dữ liệu được thuận tiện và tối ưu hơn, tránh sự phân mảnh của các tập tin.

Những lời khuyên dưới đây giúp sử dụng ổ đĩa cứng một cách tối ưu hơn:

- Phân vùng chứa hệ điều hành chính: Thường nên thiết lập phân vùng chứa hệ điều hành tại các vùng chứa phía ngoài rìa của đĩa từ (*outer zone*) bởi vùng này có tốc độ đọc/ghi cao hơn, dẫn đến sự khởi động hệ điều hành và các phần mềm khởi động và làm việc được nhanh hơn. Phân vùng này thường được gán tên là **C**.

Phân vùng chứa hệ điều hành không nên chứa các dữ liệu quan trọng bởi chúng dễ bị virus tấn công (hơn các phân vùng khác), việc sửa chữa khắc phục sự cố nếu không thận trọng có thể làm mất toàn bộ dữ liệu tại phân vùng này.

- Phân vùng chứa dữ liệu thường xuyên truy cập hoặc thay đổi: Những tập tin đa phương tiện (*multimedia*) nếu thường xuyên được truy cập hoặc các dữ liệu làm việc khác nên đặt tại phân vùng thứ hai ngay sau phân

vùng chứa hệ điều hành. Sau khi quy hoạch, nên thường xuyên thực thi tác vụ chống phân mảnh tập tin trên phân vùng này.

- Phân vùng chứa dữ liệu ít truy cập hoặc ít bị sửa đổi: Nên đặt riêng một phân vùng chứa các dữ liệu ít truy cập hoặc bị thay đổi như các bộ cài đặt phần mềm. Phân vùng này nên đặt sau cùng, tương ứng với vị trí của nó ở gần khu vực tâm của đĩa (*inner zone*).

*Có nhiều phần mềm có thể sử dụng để quy hoạch các phân vùng đĩa cứng: fdisk trong DOS, Disk Management của Windows (2000, XP) và một số phần mềm của các hãng khác, nhưng có thể chúng chỉ đơn thuần là tạo ra các phân vùng, xóa các phân vùng mà không thay đổi kích thước phân vùng đang tồn tại, chúng thường làm mất dữ liệu trên phân vùng thao tác. Partition Magic (hiện tại của hãng Symantec) thường được nhiều người sử dụng bởi tính năng mạnh mẽ, giao diện thân thiện (sử dụng chuột, giống các phần mềm trong môi trường 32 bit) và đặc biệt là không làm mất dữ liệu khi thao tác với các phân vùng*

#### **2.2.4 Định dạng của phân vùng**

Lựa chọn định dạng các phân vùng là hành động tiếp sau khi quy hoạch phân vùng ổ đĩa cứng. Tùy thuộc vào các hệ điều hành sử dụng mà cần lựa chọn các kiểu định dạng sử dụng trên ổ đĩa cứng. Một số định dạng sử dụng trong các hệ điều hành họ Windows có thể là:

- **FAT (File Allocation Table):** Chuẩn hỗ trợ DOS và các hệ điều hành họ Windows 9X/Me (và các hệ điều hành sau ). Phân vùng FAT hỗ trợ độ dài tên 11 ký tự (8 ký tự tên và 3 ký tự mở rộng) trong DOS hoặc 255 ký tự trong các hệ điều hành 32 bit như Windows 9X/Me. FAT có thể sử dụng 12 hoặc 16 bit, dung lượng tối đa một phân vùng FAT chỉ đến 2 GB dữ liệu.
- **FAT32 (File Allocation Table, 32-bit):** Tương tự như FAT, nhưng nó được hỗ trợ bắt đầu từ hệ điều hành Windows 95 OSR2 và toàn bộ các hệ điều hành sau này. Dung lượng tối đa của một phân vùng FAT32 có thể lên tới 2 TB (2.048 GB).
- **NTFS (Windows New Tech File System):** Được hỗ trợ bắt đầu từ các hệ điều hành họ NT/2000/XP/Vista. Một phân vùng NTFS có thể có dung lượng tối đa đến 16 exabytes.

Không chỉ có thế, các hệ điều hành họ Linux sử dụng các loại định dạng tập tin riêng.

### **Format**

Format là sự định dạng các vùng ghi dữ liệu của ổ đĩa cứng. Tùy theo từng yêu cầu mà có thể thực hiện sự định dạng này ở các thể loại cấp thấp hay sự định dạng thông thường.

#### **Format cấp thấp**

Format cấp thấp (*low-level format*) là sự định dạng lại các track, sector, cylinder (bao gồm cả các ‘khu vực’ đã trình bày trong phần sector). Format cấp thấp thường được các hãng sản xuất thực hiện lần đầu tiên trước khi xuất xưởng các ổ đĩa cứng. Người sử dụng chỉ nên dùng các phần mềm của chính hãng sản xuất để format cấp thấp (cũng có các phần mềm của hãng khác nhưng có thể các phần mềm này không nhận biết đúng các thông số của ổ đĩa cứng khi tiến hành định dạng lại).

Khi các ổ cứng đã làm việc nhiều năm liên tục hoặc có các khối hư hỏng xuất hiện nhiều, điều này có hai khả năng: sự lão hoá tổng thể hoặc sự rơ rã của các phần cơ khí bên trong ổ đĩa cứng. Cả hai trường hợp này đều dẫn đến một sự không đáng tin cậy khi lưu trữ dữ liệu quan trọng trên nó, do đó việc định dạng cấp thấp có thể kéo dài thêm một chút thời gian làm việc của ổ đĩa cứng để lưu các dữ liệu không mấy quan trọng. Format cấp thấp giúp cho sự đọc/ghi trên các track đang bị lệch lạc trở thành phù hợp hơn khi các track đó được định dạng lại (có thể hiểu đơn giản rằng nếu đầu đọc/ghi bắt đầu làm việc dịch về một biên phía nào đó của track thì sau khi format cấp thấp các đầu đọc/ghi sẽ làm việc tại tâm của các track mới).

Không nên lạm dụng format cấp thấp nếu như ổ đĩa cứng của bạn đang hoạt động bình thường bởi sự định dạng lại này có thể mang lại sự rủi ro: Sự thao tác sai của người dùng, các vấn đề xử lý trong bo mạch của ổ đĩa cứng. Nếu như một ổ đĩa cứng xuất hiện một vài khối hư hỏng thì người sử dụng nên dùng các phần mềm che dấu nó bởi đó không chắc đã do sự hoạt động rơ rã của phần cứng.

#### **Format thông thường**

Định dạng mức cao (*high-level format*) là các hình thức format thông thường mà đa phần người sử dụng đã từng thực hiện (chúng chỉ được gọi tên như vậy để phân biệt với format cấp thấp) bởi các lệnh sẵn có trong các hệ điều hành (DOS hoặc Windows), hình thức format này có thể có hai dạng:

- Format nhanh (quick): Đơn thuần là xoá vị trí lưu trữ các ký tự đầu tiên để hệ điều hành hoặc các phần mềm có thể ghi đè dữ liệu mới lên các dữ liệu cũ. Nếu muốn format nhanh: sử dụng tham số “/q” với lệnh trong DOS hoặc chọn “quick format” trong hộp lựa chọn của lệnh ở hệ điều hành Windows.
- Format thông thường. Xoá bỏ các dữ liệu cũ và đồng thời kiểm tra phát hiện khối hư hỏng (*bad block*), đánh dấu chúng để chúng không còn được vô tình sử dụng đến trong các phiên làm việc sắp tới (nếu không có sự đánh dấu này, hệ điều hành sẽ ghi dữ liệu vào khối hư hỏng mà nó không báo lỗi - tuy nhiên khi đọc lại dữ liệu đã ghi đó mới là vấn đề nghiêm trọng). Đối với bộ nhớ Flash thì cũng không nên format nhiều để làm hỏng ổ đĩa.

#### Tham số khi format

Ở dạng format cấp thấp, các thông số thiết đặt phần nhiều do phần mềm của hãng sản xuất xác nhận khi bạn nhập vào các thông số nhìn thấy được trên ổ đĩa cứng (Model, serial number...) nên các thông số này cần tuyệt đối chính xác nhằm tránh sự thất bại khi tiến hành.

Ở dạng format thông thường, nếu là hình thức format nhanh (quick) thì các thông số được giữ nguyên như lần format gần nhất, còn lại có một thông số mà người tiến hành format cần cân nhắc lựa chọn là kích thước đơn vị (nhỏ nhất) của định dạng là cluster (trong Windows XP mục *Allocation unit size* trong hộp thoại lựa chọn format).

Kích thước cluster có thể lựa chọn bắt đầu từ 512 byte bởi không thể nhỏ hơn kích thước chứa dữ liệu của một sector (với kích thước một sector thông dụng nhất là 512 byte). Các kích thước còn lại có thể là: 1024, 2048, 4096 với quy định giới hạn của từng loại định dạng (FAT/FAT32 hay NTFS).

Sự lựa chọn quan trọng nhất là phân vùng cần định dạng sử dụng chủ yếu để chứa các tập tin có kích thước như thế nào. Để hiểu hơn về lựa chọn, xin xem một ví dụ sau: Nếu lưu một tập tin text chỉ có dung lượng 1 byte (bạn hãy thử tạo một tập tin text và đánh 1 ký tự vào đó) thì trên ổ đĩa cứng sẽ phải dùng đến ít nhất 512 byte để chứa tập tin này với việc lựa chọn kích thước đơn vị là 512 byte, còn nếu lựa chọn cluster bằng 4096 byte thì kích thước lãng phí sẽ là  $4096 - 1 = 4095$  byte.

Nếu như lựa chọn kích thước cluster có kích thước khá nhỏ thì các bảng FAT hoặc các tập tin MFT (Master File Table) trong định dạng NTFS lại trở lên lớn hơn.



Như vậy ta nhận thấy: Nếu ổ đĩa cứng sử dụng cho các tập tin do các phần mềm văn phòng thường ngày (Winword, bảng tính excel...), nên chọn kích thước nhỏ: 1024 hoặc 2048 byte. Nếu chứa các tập tin là dạng các bộ cài đặt phần mềm hoặc các tập tin video, nên chọn kích thước này lớn hơn. Đặc biệt ở các ổ cứng nhỏ dành cho thiết bị di động thì sự lựa chọn thường là 512 byte (đây cũng thường là lựa chọn khi format các loại thẻ nhớ).

## CHƯƠNG 3 : GIỚI THIỆU CHƯƠNG TRÌNH

### 3.1 Mô tả bài toán

- Quản lý tập tin và thư mục là việc cơ bản đầu tiên cần phải biết đối với người sử dụng máy vi tính.

- Bài toán lập trình quản lý thư mục trong HDD sẽ mô tả cách quản lý thư mục gốc, thư mục con, và các tệp tin có trong ổ cứng

- Cụ thể hơn đó là đọc ,tạo ,xóa, đổi tên thư mục tên file, xem thông tin có trong ổ đĩa, format, đọc bảng fat

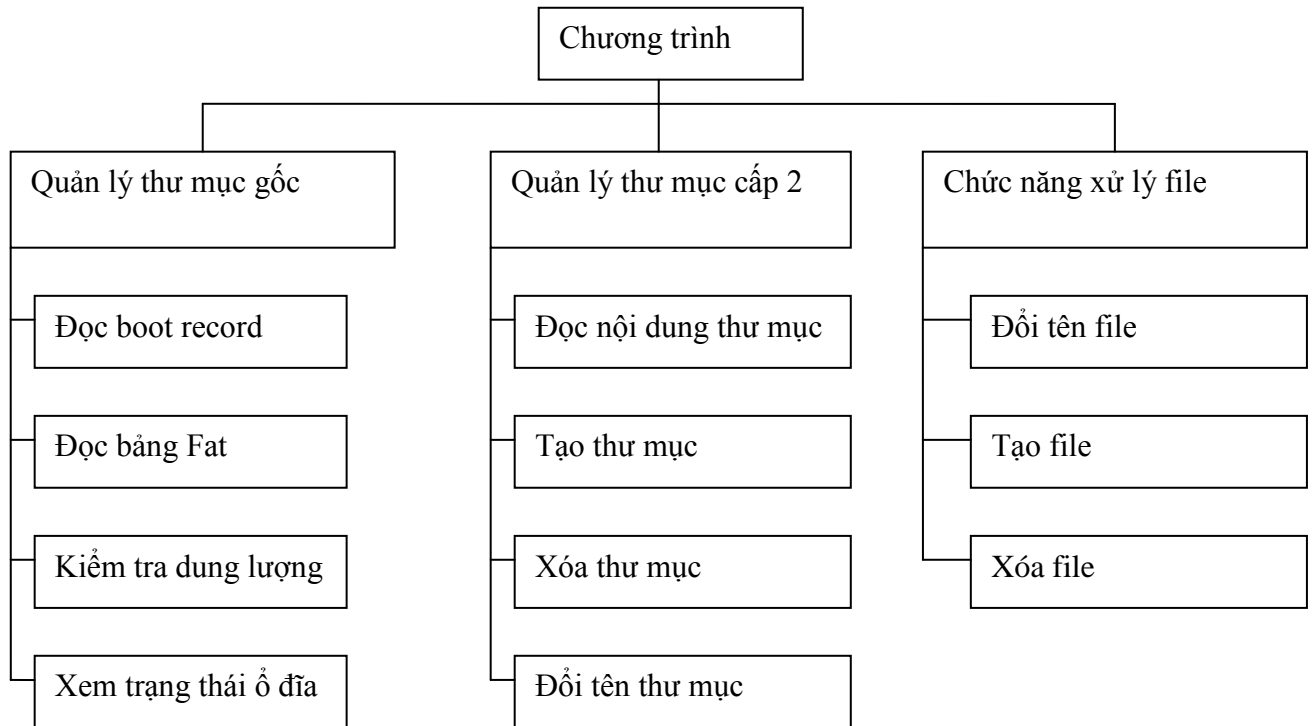
-Thư mục (Folder, Directory) là một dạng tập tin đặc biệt có công dụng như là một ngăn chứa, được dùng trong việc quản lý và sắp xếp các tập tin. Thư mục có thể chứa các tập tin và các thư mục phụ (Sub Folder) bên trong, các thư mục phụ này cũng có thể chứa thêm các tập tin và các thư mục phụ khác nữa... Có thể tạo nhiều thư mục dùng để chứa các tập tin khác nhau giúp phân loại chúng để thuận tiện trong việc tìm kiếm và sử dụng.

- Cũng giống như tập tin, thư mục có thể được đặt tên tùy ý nhưng không cần phải có phần mở rộng, độ dài của tên cũng tùy thuộc vào hệ thống tập tin và Hệ điều hành, trong một số trường hợp có thể đặt tên có dấu tiếng Việt.

- Quản lý thư mục đóng một vai trò rất quan trọng việc lưu trữ, tìm kiếm và xử lý dữ liệu. Nếu ta quản lý tốt, sắp xếp khoa học thì dữ liệu sẽ dễ dàng tìm kiếm và sử dụng nó.

- Quản lý thư mục gốc ( ổ đĩa ) quan trọng nhất sau đó đến quản lý thư mục con của nó và các file có trong thư mục con đó. Hiện nay có rất nhiều phần mềm quản lý thư mục ổ cứng một cách khoa học, chương trình này sẽ giúp chúng ta hiểu rõ hơn một phần nào đó về cách quản lý thư mục

### 3.2 Sơ đồ phân rã chức năng



Hình 3.1 Sơ đồ phân rã chức năng

### 3.3 Các hàm và ngắt trong chương trình

#### Ngắt 21h

**-Hàm 39h:** tạo thư mục

Vào: AH=39h

DX= địa chỉ offset của tên thư mục

Ra: Nếu thành công, thư mục được tạo ra

Nếu không thành công, CF=1 và AX = mã lỗi.

**-Hàm 3Ah:** xóa thư mục

Vào : AH = 3Ah

DX = địa chỉ offset của tên thư mục

Ra : Nếu thành công, thư mục được xóa, ngược lại không thành công, CF=1 và AX= mã lỗi

**-Hàm 3ch :** tạo file

Vào : AH = 3Ch

DX = địa chỉ offset của tên file

CX = thuộc tính file

Ra : Nếu thành công, file được tạo ra, CF=0 và AX=thẻ file (file handle)

Nếu không thành công, CF=1 và AX= mã lỗi.

**-Hàm 36h:** Lấy dung lượng còn trống trên đĩa

Vào : AH = 36h

DL = Ổ đĩa (0=mặc định, 1=A....)

Ra : BX = Số cluster khả dụng

DX = Số cluster/đĩa

CX = Số byte/sector

AX = FFFFh nếu ổ đĩa trong DL không hợp lệ, các trường hợp khác bằng số sector trong một cluster

**-Hàm 47h :** Lấy thư mục hiện thời

Đưa vào tên đường dẫn đầy đủ (bắt đầu từ thư mục gốc) của thư mục hiện tại trong ổ đĩa cho trước vào vùng có địa chỉ DS:SI.

Vào: AH = 48h

BX = Số paragraph nhớ yêu cầu

Ra: AX:0 = địa chỉ khối nhớ cấp phát.

AX = mã lỗi nếu cò nhớ được lập.

BX = kích thước của khối lớn nhất của bộ nhớ khả dụng nếu sự cấp phát không thành công.

**Ngắt 13h: Vào ra đĩa**

-Hàm 01h : Kiểm tra Trạng thái ổ đĩa

**Parameters:**

\$Bit 7=0 for floppy drive, bit 7=1 for fixed drive

**-Hàm 02h :** Đọc sector

Đọc một hay nhiều sector

Vào : AH=2h

AL = Số sector

CH = xi lanh (Track)

CL = sector

DH = đầu từ

DL = ổ đĩa ( 0-7Fh = đĩa mềm, 80h-ffh = đĩa cứng)

ES: BX = địa chỉ segment: offset của bộ đệm

Ra : Nếu thành công :

CF = xóa

AH = 0

AL = Số sector được đọc

Nếu không thành công :

CF = thiết lập

AH = mã lỗi

**-Hàm 03h:** Viết sector

Vào:

AH = 3h

AL = số sector

BX = thanh ghi màu thứ nhất

CH = xi lanh

CL = sector

DH = đầu từ

DL = ổ đĩa ( 0-7Fh = đĩa mềm, 80h-FFh = đĩa cứng)

ES :BX = địa chỉ segment : offset của bộ đệm

Ra: Nếu thành công:

CF = thiết lập

AH = mã lỗi

**Ngắt 25h :** Đọc tuyệt đối đĩa

Vào: AL = Số hiệu ổ đĩa

CX = Số sector đọc

DX = số hiệu sector logic bắt đầu.

DS : BX = địa chỉ chuyển đến.

Ra : Nếu thành công CF =0

Nếu không thành công CF =1 và AX chứa mã lỗi.

**Ngắt 26h :** Ghi tuyệt đối đĩa

Vào : AL = Số hiệu ổ đĩa

CX = Số sector ghi

DX = Số hiệu sector logic bắt đầu

DS : BX = địa chỉ chuyển đi

Ra: Nếu thành công CF =0, ngược lại CF =1

## CHƯƠNG 4 : DEMO CHƯƠNG TRÌNH

### 4.1. Kiểm tra dung lượng trống của ổ đĩa:

- **Chức năng:** Kiểm tra số cluster, sector

-**Hàm sử dụng: Hàm 36h**, Lấy dung lượng còn trống trên đĩa

Vào : AH = 36h

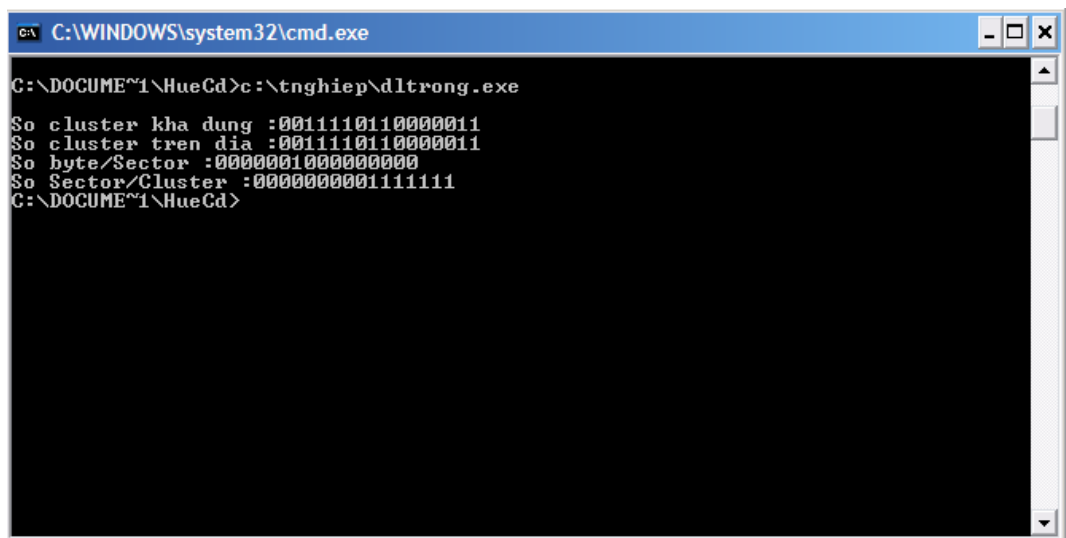
DL = Ổ đĩa (0=mặc định, 1=A....)

Ra : BX = Số cluster khả dụng

DX = Số cluster/đĩa

CX = Số byte/sector

AX = FFFFh nếu ổ đĩa trong DL không hợp lệ, các trường hợp khác bằng số sector trong một cluster



```
C:\WINDOWS\system32\cmd.exe
C:\DOCUMENT~1\HueGd>c:\tngghiep\dlttrong.exe
So cluster kha dung :0011110110000011
So cluster tren dia :0011110110000011
So byte/Sector :0000001000000000
So Sector/Cluster :0000000001111111
C:\DOCUMENT~1\HueGd>
```

Hình 4.1. Kiểm tra dung lượng trống

## 4.2. Kiểm tra trạng thái của ổ đĩa

- Chức năng : Kiểm tra trạng thái của ổ đĩa

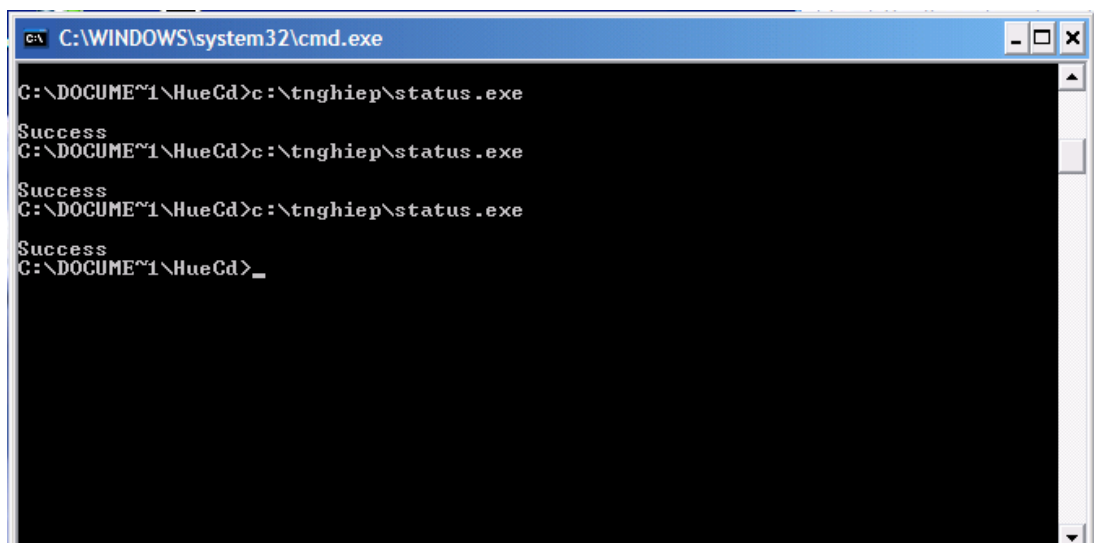
-Hàm sử dụng:

**Ngắt 13h:** Vào ra đĩa

**Hàm 01h :** Kiểm tra Trạng thái ổ đĩa

**Parameters:**

\$Bit 7=0 for floppy drive, bit 7=1 for fixed drive



```
C:\WINDOWS\system32\cmd.exe
C:\DOCUMENT1\HueCd>c:\tnghiep\status.exe
Success
C:\DOCUMENT1\HueCd>c:\tnghiep\status.exe
Success
C:\DOCUMENT1\HueCd>c:\tnghiep\status.exe
Success
C:\DOCUMENT1\HueCd>_
```

Hình 4.2. Kiểm tra trạng thái ổ đĩa



### 4.3. Đọc bảng FAT:

- **Chức năng:** bảng này liệt kê tuần tự số thứ tự của các cluster dành cho file lưu trữ trên đĩa

- **Ngắt sử dụng:** Ngắt 25h, đọc tuyệt đối đĩa

Vào: AL = Số hiệu ổ đĩa

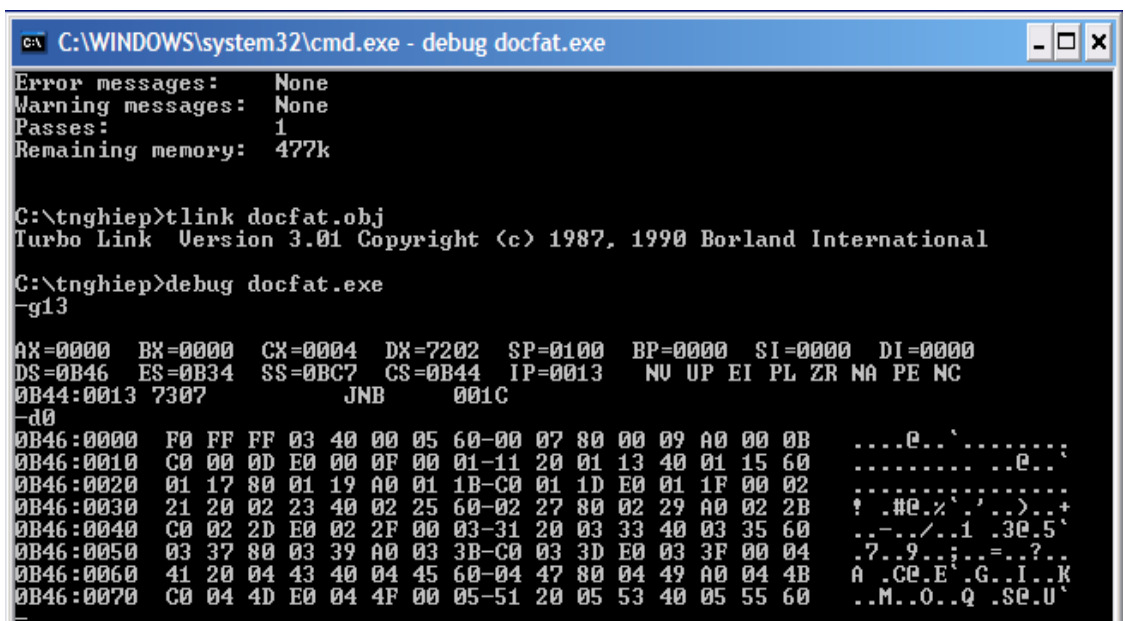
CX = Số sector đọc

DX = số hiệu sector logic bắt đầu.

DS : BX = địa chỉ chuyển đến.

Ra : Nếu thành công CF =0

Nếu không thành công CF =1 và AX chứa mã lỗi.



```
C:\WINDOWS\system32\cmd.exe - debug docfat.exe
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 477k

C:\tnghiep>tlink docfat.obj
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\tnghiep>debug docfat.exe
-g13
AX=0000 BX=0000 CX=0004 DX=7202 SP=0100 BP=0000 SI=0000 DI=0000
DS=0B46 ES=0B34 SS=0BC7 CS=0B44 IP=0013 NU UP EI PL ZR NA PE NC
0B44:0013 7307 JNB 001C
-d0
0B46:0000 F0 FF FF 03 40 00 05 60-00 07 80 00 09 A0 00 0B .....e..
0B46:0010 C0 00 0D E0 00 0F 00 01-11 20 01 13 40 01 15 60 .....e..
0B46:0020 01 17 80 01 19 A0 01 1B-C0 01 1D E0 01 1F 00 02 .....
0B46:0030 21 20 02 23 40 02 25 60-02 27 80 02 29 A0 02 2B ?.#e.%'.>..+
0B46:0040 C0 02 2D E0 02 2F 00 03-31 20 03 33 40 03 35 60 ..-./..1 .3e.5'
0B46:0050 03 37 80 03 39 A0 03 3B-C0 03 3D E0 03 3F 00 04 ?..9...;..=..?..
0B46:0060 41 20 04 43 40 04 45 60-04 47 80 04 49 A0 04 4B A .Ce.E .G. .I. .K
0B46:0070 C0 04 4D E0 04 4F 00 05-51 20 05 53 40 05 55 60 ..M..O..Q .Se.U'
```

Hình 4.3. Đọc bảng Fat

#### 4.4. Đọc bootrecord:

- Chức năng: Cho biết ổ đĩa nào là ổ đĩa khởi động

- Ngắt sử dụng: Ngắt 25h

```
C:\WINDOWS\system32\cmd.exe - debug bootrec.exe
Error messages:      None
Warning messages:   None
Passes:              1
Remaining memory:   477k

C:\tngghiep>tlink bootrec.obj
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\tngghiep>debug bootrec.exe
-g13
AX=0000 BX=0000 CX=0001 DX=7202 SP=0100 BP=0000 SI=0000 DI=0000
DS=0B46 ES=0B34 SS=0B67 CS=0B44 IP=0013  NU UP EI PL ZR NA PE NC
0B44:0013 7307          JNB      001C
-d0
0B46:0000 EB 3C 90 2A 2D 76 34 56-49 48 43 00 02 01 01 00  .<.*-v4UIHC.....
0B46:0010 02 E0 00 40 0B F0 09 00-12 00 02 00 00 00 00 00  ...e.....
0B46:0020 00 00 00 00 00 00 29 E1-6C 87 2A 20 20 20 20 20  .....>.l.*
0B46:0030 20 20 20 20 20 20 46 41-54 31 32 20 20 20 33 C9          FAT12  3.
0B46:0040 8E D1 BC FC 7B 16 07 BD-78 00 C5 76 00 1E 56 16  ....<...x...v..U.
0B46:0050 55 BF 22 05 89 7E 00 89-4E 02 B1 0B FC F3 A4 06  U."...~..N.....
0B46:0060 1F BD 00 7C C6 45 FE 0F-38 4E 24 7D 20 8B C1 99  ...!.E..8N$>...
0B46:0070 E8 7E 01 83 EB 3A 66 A1-1C 7C 66 3B 07 8A 57 FC  .~...:f...lf;..W.
```

Hình 4.4. Đọc bootrecord

#### 4.5. Hiển thị thư mục

- Chức năng: Hiển thị nội dung thư mục

- Hàm sử dụng:

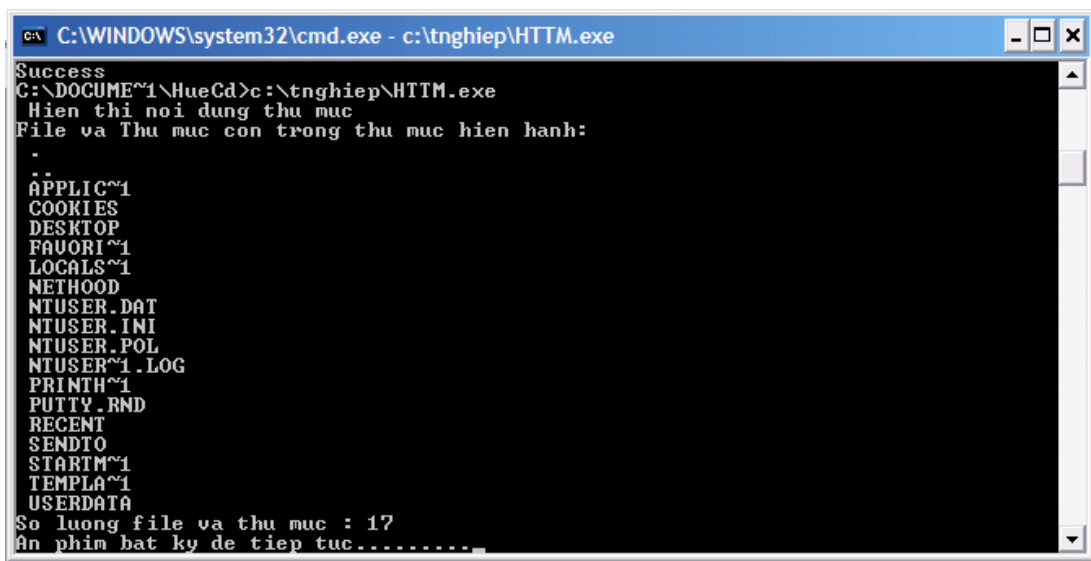
Ngắt 21h, hàm 09h: in chuỗi

Đưa chuỗi ký tự tới thiết bị chuẩn

Vào: AH =5h

DL =ký tự

Ra: Không



```
C:\WINDOWS\system32\cmd.exe - c:\tnghiep\HTTM.exe
Success
C:\DOCUMENTS\HueCd>c:\tnghiep\HTTM.exe
Hien thi noi dung thu muc
File va Thu muc con trong thu muc hien hanh:
.
.
APPLICATIONS
COOKIES
DESKTOP
FAVORITES
LOCALS
NETHOOD
NTUSER.DAT
NTUSER.INI
NTUSER.POL
NTUSER.NTUSER.LOG
PRINTH
PUTTY.RND
RECENT
SENDTO
STARTMENU
TEMPLATE
USERDATA
So luong file va thu muc : 17
An phim bat ky de tiep tuc.....
```

Hình 4.5. Hiển thị thư mục

#### 4.6. Tạo thư mục

-**Chức năng:** tạo ra các thư mục con

-**Hàm sử dụng:** Hàm 39h của ngắt 21h

Vào: AH=39h

DX= địa chỉ offset của tên thư mục

Ra: Nếu thành công, thư mục được tạo ra

Nếu không thành công, CF=1 và AX = mã lỗi

```
Nhập tên thư mục: ngọc
Chúc mừng, bạn đã tạo thư mục thành công
```



Hình 4.6. Tạo thư mục

#### 4.7. Xóa thư mục


-**Chức năng:** Xóa thư mục

-**Hàm sử dụng:** -Ngắt 21h, hàm 3Ah: xóa thư mục

Vào : AH = 3Ah

DX = địa chỉ offset của tên thư mục

Ra : Nếu thành công, thư mục được xóa, ngược lại không thành công, CF=1 và AX= mã lỗi



```
Nhap ten thu muc: vuongngoc
Xoa thu muc khong thanh cong
```

Hình 4.7. Xóa thư mục

## KẾT LUẬN

Trong quá trình nghiên cứu tài liệu và thực hiện đồ án dưới sự hướng dẫn của thầy Vũ Mạnh Khánh , em thấy bản thân đã đạt được một số kết quả như sau:

- ✓ Tìm hiểu và vận dụng được ngôn ngữ Assembly, các hàm các ngắt liên quan đến ổ cứng, thư mục, file.
- ✓ Tìm hiểu được tổng quan về ổ cứng, cấu trúc, cách vận hành ổ cứng, thư mục và file.
- ✓ Viết được các modulo quản lý thư mục, ổ đĩa , tệp tin.
- ✓ Ngoài ra, trong quá trình nghiên cứu em cũng tự tích lũy thêm cho mình các kiến thức về toán học, về kỹ thuật lập trình,...

Bên cạnh những kết quả đạt được em tự thấy bản đồ án vẫn còn một số hạn chế:

- ✓ Trong khuôn khổ một đồ án tốt nghiệp ,em mới chỉ trình bày lại các kiến thức tìm hiểu được chứ chưa đề xuất được một phương pháp hoàn toàn mới.
- ✓ Chỉ đọc bảng fat, thư mục gốc mà chưa ghi được
- ✓ Chưa có giao diện chương trình mà chỉ làm Assembly thuần túy trên DOS

## TÀI LIỆU THAM KHẢO

### Tài liệu tiếng việt:

[ 1 ] ThS. Phạm Văn Cường, “ Lập trình hệ thống và điều khiển thiết bị ”, Học viện công nghệ bưu chính viễn thông

[ 2 ] Đỗ Xuân Toàn, “ Kỹ thuật vi xử lý và lập trình Assembly cho hệ vi xử lý”, NXB khoa học và kỹ thuật

### Website :

Chương 1 và 2 tham khảo “ [Http://vi.wikipedia.org/wiki/Ổ\\_đĩa\\_cứng](http://vi.wikipedia.org/wiki/Ổ_đĩa_cứng) ”