MỘT SỐ KHÁI NIỆM CƠ BẢN VỀ HỢP NGỮ

1. Giới thiệu chung
2. Cấu trúc chương trình hợp ngữ
3. Cú pháp hợp ngữ
4. Các thành phần cơ bản của hợp ngữ
5. Tệp tin dạng .COM và . EXE
6. Sử dụng chương trình Macro Assembler 5.1

1. GIỚI THIỆU CHUNG

- Hợp ngữ *(Assembly Language)* là ngôn ngữ lập trình bậc thấp sử dụng dạng ký hiệu.
- Từ gợi nhớ *(Mnemonic)* tương đương 1-1 với lệnh CPU.
- Chương trình biên dịch (Assembler) dùng để biên dịch từ dạng ký hiệu sang mã máy.
  - ✓MacroAssembler của Microsoft.
  - ✓TurboAssembler của Borland.

Từ gợi nhớ (Mnemonic)

- Mnemonic – mã lệnh tượng trưng
- Assembler dịch mã lệnh sang mã máy
- Ví dụ: MOV, SUB, ADD
- Directive - chỉ dẫn cho trình biên dịch
- Assembler không dịch Directive mà dùng như hướng dẫn để biên dịch
- Ví dụ: ORG, PROC, ENDP, END, DB,...

Dữ liệu trong chương trình hợp ngữ

- **Số:**
  - Nhị phân: 10001110B, 10001110b
  - Thập phân: 12345, 12345D, 12345d
  - Hexa: 0ABCDH, 0ABCDh
- **Ký tự, chuỗi ký tự:** “hello!”, ‘hello!’
- **Biến:**
  - kiểu byte: Name db init_value
  - kiểu word: Name dw init_value
- **Mảng :** “hello!”, ‘hello!’

  B_array db 41h,42h,43h,44h
- **Hằng :** Name EQU Const

  LF EQU 0Ah

Chỉ dẫn hợp ngữ

- Khuôn dạng:

  [Tên] Từ gợi nhớ [Toán hạng] ;Chú giải

  Các chỉ dẫn thường dùng:

  1. Nhóm định nghĩa tên: **`Name EQU <text>`**
  2. Nhóm khai báo dữ liệu: **`[Name] DB data`**
  3. Nhóm khai báo đoạn: **`SEGMENT, ENDS`**
  4. Nhóm thủ tục: **`PROC, ENDP`**
  5. Chỉ dẫn ORG: **`ORG Expression`**
  6. Chỉ dẫn ASSUME: **`ASSUME CS:CODE`**

6. SỬ DỤNG MACROASSEMBLER

- *Bước 1*: Sử dụng các trình soạn thảo ASCII để viết chương trình nguồn và đặt tên mở rộng là .ASM.
- *Bước 2:* Sử dụng chương trình MASM.exe để dịch chương trình nguồn .ASM thành chương trình .OBJ.

  Cách viết: **masm sourcefile**
- *Bước 3:* Sử dụng chương trình link để liên kết các tệp tin mã máy dạng .OBJ thành tệp tin thi hành được dạng .EXE.

  Cách viết: **LINK objectfile**

  VD: **link chao**
- Nếu là chương trình .COM thì sử dụng exe2bin.exe để chuyển đổi tệp tin .EXE thành tệp tin .COM.

  VD: **exe2bin chao chao.com**

TỆP TIN .COM VÀ .EXE

1. Tệp tin .COM:

- Các đoạn Stack, Data, Code nằm trong 1 đoạn (64KB)
- Khi thực hiện các chương trình *.COM hệ điều hành sẽ tạo các thanh ghi segment có địa chỉ đầu như sau:
- Code segment CS: 0100
- Data segment DS: 0000
- Stack segment SS: FFFE
- Extra segment ES: 0000
- Tệp tin dạng COM được nạp vào bộ nhớ và thực hiện nhanh hơn tệp tin dạng EXE.

Cấu trúc chương trình dạng COM.

VD: hiển thị dòng " hello" trên màn hình. Tên tệp Hello.asm

```
code        segment
    assume      cs:code, ds:code
    org         100h
    Begin:
    mov         ah,9h
    mov         dx,offset mess
    int         21h
    int         20h
    mess        db 'hello.    $'
code        ends
end          begin
```

Tệp tin .EXE:

Chương trình có nhiều đoạn khác nhau

Kích thước chương trình lớn hơn 64K

Tận dụng được hết ưu điểm của máy 16 bits, không hạn chế về bộ nhớ. Phần mã lệnh CS, DS, SS có thể khai báo ở nhiều đoạn khác nhau.

Điểm vào của một chương trình .EXE nằm ở bất cứ một nơi nào trong vùng mã lệnh, được xác định khi khai báo vùng mã lệnh của chương trình (do người lập trình quyết định).

Với kích thước có thể lớn tuỳ ý phụ thuộc vào kích thước bộ nhớ máy tính. File EXE được sử dụng để xây dựng các chương trình lớn hơn 64 KB.

Cấu trúc chương trình dạng EXE

```
stack       segment stack
            dw      80      dup(?)
stack       ends
data        segment
            mess    db      'chao.$'
data        ends
code        segment
assume      cs:code,ds:data
begin:      mov     ax,data
            mov     ds,ax
            mov     ah,9
            mov     dx,offset mess
            int     21h
            mov     ah,4ch
            int     21h
code        ends
end         begin
```

Khi chạy file .EXE sẽ khởi sinh: CS : IP, SS : SP và DS, ES → PSP

Ghi chú: trong chương trình có 3 đoạn khác nhau là code, data, stack.

Chương trình .COM và .EXE sẽ trở về DOS bằng lệnh INT 21h với Ah = 4ch. Một số hàm ngắt int 21h thường dùng:

Ah = 02	In ký tự có mã ASCII trong DL ra màn hình
Ah = 09	In một dãy ký tự ra màn hình Địa chỉ đầu của dãy ký tự: DS:DX Kết thúc dãy ký tự là dấu $
Ah = 4ch	Thoát về DOS
Ah = 01	Nhận một ký tự từ bàn phím và hiện nó ra màn hình.
Ah = 08	Nhận một ký tự từ bàn phím, mã ASCII của nó trong AL
Ah = 0Ah	Nhập chuỗi dài tối đa 255 ký tự từ bàn phím, phím Enter kết thúc chuỗi

Ngắt Int 10h với các giá trị của AH

AH=00h	Đặt chế độ Video; AL = chế độ
AH=01h	Xác lập dòng trên cùng và dòng dưới cùng của con trỏ. Khi đó CH và CL chứa dòng bắt đầu và kết thúc của con trỏ.
AH=02h	Đặt vị trí con trỏ, Dh hàng, Dl cột
AH=03h	Xác định vị trí hiện thời của con trỏ (Dh: hàng, Dl: cột)
AH=06h	Xoá màn hình (cuốn màn hình từ 1 điểm có toạ độ Cl = cột, Ch = dòng đến 1 toạ độ mới Dl = cột và Dh = hàng)
AH=09h	Ghi ký tự có thuộc tính trong Bl và mã ASCCI trong Al ra màn hình. BH: Trang màn hình; CX: Số lần viết ký tự .
AH=0Eh	Ghi ký tự có mã ASCCI trong Al ra màn hình tại vị trí hiện thời của con trỏ và dịch con trỏ sang vị trí tiếp theo.

AH=13h	Hiện thị xâu ký tự mã ASCCI có địa chỉ đầu tại ES: BP Với:AL = 0: BL chứa thuộc tính, không cập nhật vị trí con trỏ AL = 1: BL chứa thuộc tính, có cập nhật vị trí con trỏ AL = 2: Xâu chứa cả ký tự và thuộc tính. Byte sau ký tự là bye thuộc tính, không cập nhật vị trí con trỏ. AL = 3: Xâu chứa cả ký tự và thuộc tính. Byte sau ký tự là bye thuộc tính, vị trí con trỏ được cập nhật. BH: Số trang; CX: độ dài xâu; DH: Tọa độ hàng; DL: Tọa độ cột.

Ngắt INT 16 với các hàm

AH = 00 h	Đọc 1 ký tự. Ah chứa mã quét, Al chứa mã ASCI
AH = 01 h	Đọc 1 ký tự. Ah chứa mã quét, Al chứa mã ASCII. Cờ ZF = 0
AH = 02h	Al sẽ chứa byte trạng thái bàn phím hiện hành
AH = 10h	Đọc 1 ký tự cả trên bàn phím mở rộng. Ah chứa mã quét, Al chứa mã ASCII
AH = 11 h	Đọc 1 ký tự cả trên bàn phím mở rộng. Ah chứa mã quét, Al chứa mã ASCII. Cờ ZF = 0
AH = 12 h	Al sẽ chứa byte trạng thái bàn phím hiện hành kể cả trên bàn phím mở rộng

Bài tập.

1. Viết chương trình in ra màn hình 6 ký tự đầu tiên của bảng mã ASCII mỗi ký tự cách nhau hai khoảng trống.

Đoạn chương trình

2. Viết chương trình hiển thị dãy số 324.

Sử dụng ngắt int 21h với ah=09h. (ngắt in21h)

Đoạn chương trình

3. Viết chương trình hiển thị chữ A nền đỏ chữ trắng tại vị trí cột 5 dòng 5. Chương trình sẽ kết thúc khi ấn phím Enter

Sử dụng ngắt int 10h; ah=09; Al=ASCII; Bl=thuộc tính

Đoạn chương trình

Chương trình gỡ rối DEBUG

- Cho phép truy xuất và thay đổi từng byte của bộ nhớ và nội dung các thanh ghi của CPU.

  - Cho phép truy xuất và thay đổi từng sector của đĩa.

  - Cho phép xem và thay đổi nội dung của mọi tập tin cả tập tin văn bản lẫn mã nhị phân.

  - Cho phép dịch các lệnh mã máy sang dạng gợi nhớ và ngược lại.

  - Cho phép nạp và thực hiện các tập tin thi hành được, thậm chí có thể vừa sửa vừa thực hiện từng bước chương trình.

DEBUG có 19 lệnh. các lệnh này sử dụng chữ cái tiếng Anh để gợi nhớ ý nghĩa của lệnh. Lần lượt chúng ta sẽ xét các lệnh cơ bản của DEBUG.

1. Lệnh R

Công dụng:

Dạng 1: Cho phép xem nội dung tất cả các thanh ghi của CPU.

Dạng 2: Xem và sửa nội dung thanh ghi được liệt kê tên.

Ví dụ:

1. Xem nội dung tất cả các thanh ghi:

R ↵

2. RAX

Sẽ hiện:AX 00D2

:

Hiện giá trị của AX và chờ người sử dụng hiệu chỉnh. Nếu không muốn hiệu chỉnh, ấn Enter.

2. Lệnh A:

Công dụng: Cho phép nhập chỉ thị tại mỗi địa chỉ. Nếu sau A không có offset thì Offset bằng 100 (nếu khi mới khởi động chương trình DEBUG) còn không thì tiếp nhận offset hiện hành

Ví dụ: A 100 ↵

xxxx:0100 mov dl,04

xxxx:0102 mov ah,02

xxxx: 0104 int21

xxxx: 0106 int 20

↵

G ↵

3. Lệnh G (Go) G [=address1][address2]

Công dụng: Thực hiện đoạn chương trình trong bộ nhớ bắt đầu tại địa chỉ address1 và kết thúc tại địa chỉ address2.

Nếu không có địa chỉ address1 thì chương trình sẽ bắt đầu tại địa chỉ xác định bởi CS:IP. (Code Segment và Instruction Pointer)

Nếu không có địa chỉ address2 thì chương trình sẽ thực hiện cho đến khi gặp lệnh kết thúc chương trình.

4. Lệnh U (UnASEMBLER) U [Address]

Công dụng: In 32 byte mã máy của chương trình trong bộ nhớ ra màn hình dưới dạng lệnh gợi nhớ.

- Nếu là lệnh U không có tham số thì lệnh U hiển thị 32 byte mã máy tại địa chỉ CS:100. Các lệnh U sau đó sẽ hiển thị các mã máy tiếp theo.

- Nếu trong lệnh U không chỉ rõ địa chỉ đoạn thì DEBUG sẽ lấy đoạn được chỉ bởi CS.

Ví dụ: U 100 124

Sẽ in ra màn hình lệnh mã máy dưới dạng từ gợi từ địa chỉ CS:100 đến CS:124.

5. Lệnh D (Dump) D [address] D [range]

Ý nghĩa: Liệt kê nội dung bộ nhớ ra màn hình dưới dạng số HEX.

- Dạng 1: in nội dung vùng nhớ gồm 128 byte bắt đầu tại địa chỉ do address qui định.
- Dạng 2 in toàn bộ nội dung vùng nhớ do range qui định.
- Không có tham số:

  + Nếu là lệnh D đầu tiên thì DEBUG in nội dung vùng nhớ dài 128 byte bắt đầu tại địa chỉ DS:100.

  + Các lệnh D tiếp theo DEBUG in tiếp các vùng nhớ tương ứng tiếp theo

Ví dụ:

- D 100 ↵
- D CS:100 120 Xem nội dung bộ nhớ địa chỉ CS:100 đến CS:120
- D F000:100 L7 Xem nội dung 7 byte

6. Lệnh N (Name) N Filename [Arglist]

Công dụng: Xác định tên tệp tin cần đọc hay cần ghi trước khi dùng lệnh L hay W

Ví dụ:

- N c:\Autoexec.bat
- l ↵
- d ↵

Nạp và hiển thị dưới dạng mã tệp Autoexec.bat.

7. Lệnh L (Load) L [Address] hoặc L Address drive fistsector number

Công dụng:

- Dạng 1: Nạp nội dung tệp tin có tên xác định bởi lệnh N vào bộ nhớ tại địa chỉ address. Cặp thanh ghi bx:cx sẽ chứa kích thước tệp tin. Nếu không có address thì tệp tin được nạp bắt đầu tại địa chỉ CS:100.

-Dạng 2: Nạp nội dung các sector trên đĩa vào bộ nhớ bắt đầu tại địa chỉ do address qui định. sector đầu tiên trên đĩa cần đọc xác định trong FistSector. Number là số sector cần đọc.

Ví dụ: -L ds:0 0 1 5

8. Lệnh W (Write) W [address] hoặc W address dive firstSector Number

Công dụng:

- Dạng 1: Ghi 1 vùng nhớ có địa chỉ bắt đầu do address qui định có chiều dài được cho trong cặp thanh ghi BX:CX lên 1 tệp tin có tên xác định bởi N. Nếu không ghi địa chỉ thì tệp tin được ghi bắt đầu từ địa chỉ CS:100.

- Dạng 2: Ghi nội dung các sector trong bộ nhớ bắt đầu tại địa chỉ qui định bởi address lên đĩa. FirstSector xác định sector đầu tiên trên đĩa sẽ được ghi. Number sẽ xác định số các sector cần ghi.

Chú ý: Nếu không có địa chỉ đoạn thì địa chỉ được xác định bởi CS.

Ví dụ 1: Ghi 8 sector trong bộ nhớ bắt đầu tại địa chỉ CS:0 vào sector thứ 200 của đĩa A.

-W 0 0 200 8

9. Lệnh E E address [list]

Ý nghĩa: Thay đổi nội dung bộ nhớ. Danh sách được nêu ở Litst sẽ được thay vào vùng nhớ tại địa chỉ được xác định bởi address.

Nếu không có tham số list, debug sẽ hiện địa chỉ và nội dung của ô nhớ ra màn hình, chờ người sử dụng nhập trị cần đổi vào. Muốn đổi ô nhớ tiếp ấn Space, muốn sửa ô trước, ấn -, và muốn kết thúc ấn Enter.

Ví dụ:

- E 100 "bcdefg"

-> đổi 6 byte thành bcdefg

-E xxxx:100 40 41 42 43

-> thay 4 byte thành 40 41 42 43

10. Lệnh I (Input) I Port

Ý nghĩa: Đọc giá trị 8 bit từ cổng vào có địa chỉ cổng ấn định bởi Port.

Ví dụ: I 3FF

Nếu tại địa chỉ 3FF có nội dung là 5A thì nội dung 5A sẽ được chứa trong thanh ghi AL.

11. Lệnh O (Output) O Port byte

Ý nghĩa: Xuất giá trị byte ra cổng ở địa chỉ Port

Ví dụ: Xuất giá trị 00 ra cổng 61

- o 61 00.

12. Lệnh M (move) M range address

Ý nghĩa: Dùng để sao chép nội dung vùng nhớ nguồn (range) sang vùng nhớ đích (address).

Nếu trong range hay address không qui định địa chỉ đoạn thì debug sẽ lấy đoạn được chỉ bởi DS.

Ví dụ: M 100 L100 4000:000 thực hiện chép vùng nhớ từ địa chỉ DS:100 với độ dài 100 byte sang vùng nhớ 4000:000

13. Lệnh T (trace) và P (Procced) T [=address] [value]

P [=address] [value]

Ý nghĩa: Dùng để thực hiện từng bước một hay nhiều lệnh bắt đầu tại địa chỉ do *address* qui định. Sau mỗi lệnh nội dung của tất cả các thanh ghi đều hiện ra màn hình. *Value* xác định số các lệnh cần thực hiện.

Nếu không có *Address* thì lệnh P hay T thực hiện từ địa chỉ CS:IP.

Nếu không có *Value* thì thực hiện 1 lệnh.

Ví dụ: T =100 5

Bài tập:

Vào chương trình Debug thực hiện các lệnh sau:

1. Lệnh R.

Giải thích ý nghĩa của các thông tin được hiển thị ra màn hình.

2. Dùng lệnh A gõ một vài dòng lệnh ASM thực hiện một chức năng nào đó. Sau đó dùng lệnh G để thực hiện các lệnh vừa gõ. Nhận xét và giải thích những gì nhận được.

3. Thực hiện lệnh D 0040:0000 L10. Ghi lại những thông tin nhận được trên màn hình.

4. Thực hiện lệnh I 03fd sau đó thực hiện lệnh R AX.

Ghi nhận giá trị của AL mà màn hình thông báo.

Bài tập: