CÁC LỆNH TRAO ĐỔI DỮ LIỆU

Lệnh	Cú pháp	Chức năng	Ví dụ
MOV	**MOV** đích, gốc	Lệnh gán giá trị của gốc vào đích	MOV AL, BL ;AL = BL MOV CX, 123FH ;CX = 123FH
LEA	**LEA** đích, gốc	Lệnh gán địa chỉ của gốc vào đích	LEA SI, a ;nạp địa chỉ biến a vào thanh ghi SI LEA CX, [BX] ;nạp địa chỉ ô nhớ có địa chỉ [DS:BX] vào CX (hay CX = BX)
PUSH	**PUSH** gốc	Đẩy giá trị của gốc vào ngăn xếp	PUSH AX ;đẩy giá trị của AX vào ngăn xếp
POP	**POP** gốc	Lấy giá trị trên cùng của ngăn xếp và gán vào gốc	POP AX ;lấy giá trị trên cùng của ngăn xếp và gán vào AX
LOOP	**LOOP**	Vòng lặp có số lần lặp phụ thuộc vào CX	
LODSB	**LODSB**	AL ← [SI] Nếu DF=0 thì : SI ← SI + 1 ngược lại thì : SI ← SI – 1 Mặc định DF=0	

Một cách dễ hiểu, LEA tải một con trỏ đến mục bạn đang xử lý trong khi MOV tải giá trị thực tại địa chỉ đó.

Note: *khi muốn truy cập tới địa chỉ đích mà phải dùng MOV (thường thì ta sẽ dùng LEA), ta có thể thêm câu lệnh offset ngay trước gốc.*

CÁC PHÉP TOÁN

Lệnh	Cú pháp	Chức năng	Ví dụ
ADD	**ADD** đích, gốc	Cộng hai toán hạng	SUB AL, 20H ;AL = AL + 20H SUB DL, [SI] ;DL = DL + [DS:SI]
SUB	**SUB** đích, gốc	Trừ hai toán hạng	SUB AL, 20H ;AL = AL - 20H SUB DL, [SI] ;DL = DL - [DS:SI]
MUL	**MUL** gốc	Nhân hai toán hạng - Gốc 8 bit: AX = AL x Gốc - Gốc 16 bit: DXAX = AX x gốc	MUL CL ;AX = AL x CL MUL BX ;DXAX = AX x BX
DIV	**DIV** gốc	Chia hai toán hạng - Gốc 8 bit: AL = AX / Gốc AH = AX % Gốc - Gốc 16 bit: AX = DXAX / gốc DX = DXAX % gốc Thương được làm tròn theo số nguyên dưới (VD: AX / gốc = 4,9 => AL = 4)	DIV BL ;AL = AX / BL ;AH = AX % BL DIV [SI] ;AL = AX / [DS:SI] ;AH = AX % [DS:SI] DIV BX ;AX = DXAX / BX ;DX = DXAX % BX

LỆNH NGẮT INT 21H

- Cú pháp: **INT 21H**
- Chức năng của lệnh dựa theo giá trị của AH

Ngắt	Chức năng	Thực hiện khi AH=?	Cách emu8086 xử lý	Ví dụ

Ngắt loại 1	Đọc một ký tự từ bàn phím	AH = 1	Đọc một ký tự được nhập vào từ bàn phím, AL sẽ lưu mã ASCII của phím vừa nhập. Nếu phím vừa nhập là phím chức năng, AL = 0	MOV AH, 1; INT 21H; => chương trình sẽ ngừng lại đến khi bạn nhập vào một phím

Ngắt loại 2	Hiện một kí tự lên màn hình	AH = 2	Hiện một ký tự có mã ASCII là giá trị của DL lên màn hình	MOV AH, 2;
				MOV DL, 30h; (30h là mã ASCII của '0')
				INT 21H;
				=> màn hình sẽ in ra ký tự '0'

Ngắt loại 9	Hiện xâu ký tự	AH = 9	Hiện xâu ký tự có địa chỉ lệnh là giá trị của DX	tb BD 'co lam thi moi co an$'; MOV AH, 9; LEA DX, tb; INT 21H; => màn hình in ra xâu tb (ko hiện ký tự '$')
Ngắt loại 10 (ngắt 0AH)	Đọc một chuỗi từ bàn phím	AH = 10 (0AH)	Nhập một xâu kí tự vào một biến đệm cho trước trong chương trình, biến này phải được chỉ bởi thanh ghi DX. Và biến đệm phải có dạng như sau: - Byte 0: chứa số kí tự tối đa của xâu nhập vào - Byte 1: chứa một trị không (= 0) - Byte 2, byte 3, byte 4, ... chứa một trị rỗng (để trống), để chứa các kí tự sẽ được nhập vào sau này (khi chương trình thực hiện).	VD1: Xau DB 256, 0, 256 Dup (' '); => Xau là một biến kiểu byte, gồm 258 byte. Byte đầu tiên (byte) chứa trị 256, byte 1 chứa trị 0, 256 byte tiếp theo chứa kí tự trống. Giả sử sau đó ta nhập vào Xau một chuỗi là 'Test' thì Byte đầu tiên chứa trị 256, byte 1 chứa trị 4 là độ dài chuỗi, 4 byte tiếp theo chứa 4 kí tự 'Test'
Ngắt chương trình	Dừng chương trình	AH = 4 CH	Dừng chương trình	MOV AH, 4CH

KHAI BÁO BIẾN, HẰNG, CHUỖI KÝ TỰ, MẢNG

1. Biến:

cú pháp: <tên biến> <kiểu dữ liệu> <giá trị khởi đầu>

Trong đó:

- tên biến: là tên của biến
- kiểu dữ liệu: là miền giá trị của biến, có 3 kiểu dữ liệu:

DB (define byte): biến byte, độ dài 8 bit

DW (define word): biến word, độ dài 16 bit

DD (define double word): biến double word, độ dài 32 bit

- giá trị khởi đầu: là giá trị được gán vào khi biến được khởi tạo. Nếu muốn khởi tạo biến mà không có giá trị ban đầu, sử dụng ký tự '?'

  VD:

```
B1      DB      16      ;khởi tạo biến B1 có giá trị là 16
x       DB      ?       ;khởi tạo biến x không có giá trị ban đầu
```

2. Hằng:

cú pháp: <tên hằng> EQU <giá trị>

- Ví dụ:

```
CRLF  EQU   13, 10, '$'     ;khai báo hằng CRLF
a     EQU   'Hi '           ;khai báo hằng a = 'Hi'
```

3. Chuỗi:

cú pháp: <tên xâu> <kiểu dữ liệu> <xâu>

- Là một kiểu đặc biệt của mảng, trong đó, mỗi ký tự của xâu là 1 phần tử của mảng. giá trị của mỗi phần tử chính là mã ASCII của ký tự đó.
- ký tự '$' báo hiệu đã hết xâu.
- VD:

```
xau1    DB      'ahihi$'        ;xau1 = 'ahihi'
xau2    DB      30h, 31h, 32h, 33h,'$'          ;xau2 = '0123$'
CRLF    DB      13,10,'$'       ;xâu dùng để xuống dòng và về đầu dòng
```

Note: *13 là ký tự về đầu dòng (CR - carriage return)*

10 là ký tự thêm dòng mới (LF - line feed),

hiểu đơn giản CRLF có tác dụng như "\n" trong c\c++

4. Mảng:

cú pháp: <tên mảng> <kiểu dữ liệu> <giá trị 1>, <giá trị 2>, …

Trong đó:

- tên mảng: là tên của mảng
- kiểu dữ liệu: là miền giá trị của biến, có 3 kiểu dữ liệu (tương tự kiểu dữ liệu của biến)
- giá trị: là giá trị được gán vào khi các biến được khởi tạo. Nếu muốn khởi tạo biến mà không có giá trị ban đầu, sử dụng ký tự '?'. Nếu muốn khởi tạo nhiều biến có cùng 1 giá trị, sử dụng lệnh:

  <số lượng phần tử> DUP(<giá trị>), lệnh DUP có thể lồng nhau.

Ví dụ:

```
mang    DB      1,2,3,4,5,6,7           ;khởi tạo mảng có 7 phần tử có gía trị từ 1 đến 7
a       DB      100 DUP (?)             ;khởi tạo mảng có 100 phần tử chưa có giá trị
b       DB      1, 2, 3 DUP(4)          ;khởi tạo mảng có 5 phần tử có gtrị là: 1,2,4,4,4
```

```
c       DB      1,2, 2 DUP(4 , 3 DUP(5), 6)   ;các phần tử của c là 1, 2, 4, 5, 5, 5, 6, 4, 5, 5, 5,6
```

CÁC LỆNH NHẢY

Lệnh nhảy	Tên Tiếng Anh	Chức năng
JMP	Jump	nhảy không điều kiện
JG	Jump if Greater	nhảy nếu lớn hơn
JNG	Jump if Not Greater	nhảy nếu không lớn hơn (nhảy nếu bé hơn hoặc bằng)
JLE	Jump if Lower or Equal	tương tự JNG
JL	Jump if Lower	nhảy nếu bé hơn
JNL	Jump if Not Lower	Nhảy nếu không bé hơn (nhảy nếu lớn hơn hoặc bằng)
JGE	Jump if Greater or Equal	tương tự JNL
JE	Jump if Equal	nhảy nếu bằng
JNE	Jump if Not Equal	nhảy nếu không bằng
JZ	Jump if Zero	nhảy nếu bằng 0 (lệnh này tương tự JE)
JNZ	Jump if Not Zero	nhảy nếu khác 0 (lệnh này tương tự JNE)
JS	Jump if Signed	nhảy nếu có dấu (nhảy nếu SF == 1)
JNS	Jump if Not Signed	nhảy nếu không có dấu (nhảy nếu kết quả dương) (nhảy nếu SF == 0)
JC	Jump if Carry	nhảy nếu có nhớ (nhảy nếu CF == 1)
JNC	Jump if Not Carry	nhảy nếu không có nhớ (nhảy nếu CF == 0)
JO	Jump if Overflow	nhảy nếu tràn (nhảy nếu OF == 1)
JNO	Jump if Not Overflow	nhảy nếu không tràn (nhảy nếu OF == 0)

CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH HỢP NGỮ (.EXE)

```
.MODEL SMALL
.STACK 100H
.DATA
;khai báo các biến và hằng
.CODE
        MAIN PROC
                ;nạp các biến đã được khai báo trong .CODE vào thanh ghi đoạn DS
                MOV AX, @data
                MOV DS, AX
                ;code nằm ở đây
                ;kết thúc chương trình
                MOV AH, 4CH
                INT 21H
        MAIN ENDP
        ;các chương trình con (nếu có)
END MAIN
```