




## Chương 6

# TẬP LỆNH CPU



## I. Giới Thiệu

- Các thành phần cấu thành chương trình hợp ngữ
  - Chương trình hợp ngữ được cấu thành từ 3 nhóm thành phần :
    - Tập lệnh của CPU và tập lệnh của bộ đồng xử lý toán học (coprocessor) : được thực hiện khi thi hành chương trình (tập lệnh này chạy trên hệ điều hành)
    - Các chỉ dẫn (directive) } **Được thực thi khi dịch**
    - Các toán tử (operators) } **(assembler)**
- Cấu trúc tổng quát lệnh của Intel



## Cấu trúc tổng quát lệnh của Intel

chiều thao tác

←

**[Label : ]      Operation [operand 1] [, operand 2] [;comment]**


↑

**Toán hạng đích**

↑

**Toán hạng nguồn**

- Lệnh có thể có 0, 1, 2 toán hạng, có thể hiểu ngầm toán hạng
- Kết quả (operation) nếu có được lưu vào toán hạng đích
- Operand 1 có thể nằm ở thanh ghi hoặc bộ nhớ
- Operand 2 có thể trong thanh ghi, bộ nhớ hoặc dữ liệu tức thời



- Toán hạng số tức thời có thể là số trong các hệ đếm khác nhau và được viết theo qui định như sau :
  - Số hệ 2 : xxxxxxxxB (x là 1 bit nhị phân).  
Ví dụ : 01101101B, 11111111B
  - Số hệ 10 : xxxxx , hay xxxxxD (x là một số thuộc hệ 10).  
Ví dụ : 65535, 1000
  - Số hệ 16 : xxxxH và bắt đầu bằng số ( là số thuộc hệ 16).  
Ví dụ : 1A59H, 0E05BH

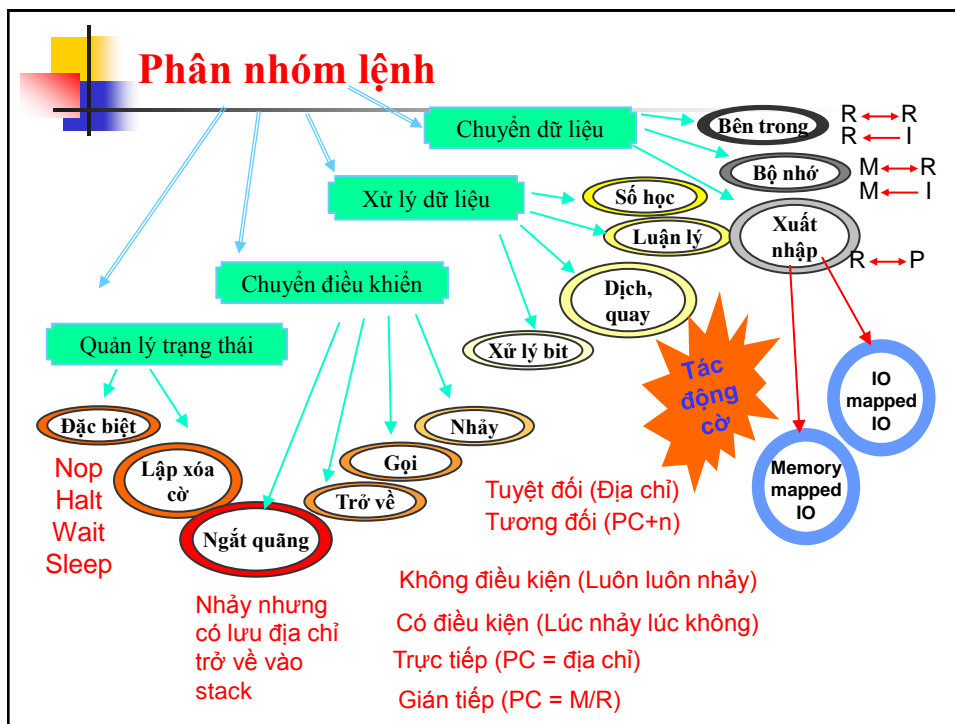


- Để thuận tiện trong vấn đề giải thích lệnh, ta qui ước thêm cách diễn tả sau :
  - Dữ liệu 8 bit của bộ nhớ : [địa chỉ]
  - Dữ liệu 16 bit của bộ nhớ : [địa chỉ +1, địa chỉ]
- Để xác định rõ hoạt động của bộ nhớ, ta phải dùng thêm toán tử PTR như sau :
  - Hoạt động 8 bit : BYTE PTR [1000h] là tham khảo 1 byte bộ nhớ có địa chỉ 1000h
  - Hoạt động 16 bit : WORD PTR [1000h] là tham khảo đến 2 byte bộ nhớ liên tiếp 1000h và 1001h



### Cấu trúc tổng quát lệnh của Intel

- Lưu ý :
  - Hai toán hạng không thể đồng thời nằm trong bộ nhớ (trừ các lệnh chuỗi)
  - Hai toán hạng phải có cùng chiều dài (trừ 1 số quy ước ngoại lệ)
  - Operation : cho biết lệnh công dụng làm gì
  - Label (nhãn) : khi dịch xong được gán 1 địa chỉ offset



## Các chữ viết tắt trong tập lệnh

<i>reg</i>	:	thanh ghi tổng quát.
<i>reg16</i>	:	thanh ghi 16 bit.
<i>segreg</i>	:	thanh ghi đoạn.
<i>accum</i>	:	thanh ghi bộ tích lũy <b>AX</b> hoặc <b>AL</b> .
<i>mem</i>	:	bộ nhớ (địa chỉ hiệu dụng).
<i>mem16</i>	:	bộ nhớ 2-byte-liên-tiếp (địa chỉ hiệu dụng).
<i>mem32</i>	:	bộ nhớ 4-byte-liên-tiếp (địa chỉ hiệu dụng).
<i>immed</i>	:	số tức thời.
<i>immed8</i>	:	số tức thời 8 bit.
<i>shortlabel</i>	:	nhãn ngắn (-128 byte ÷ +127 byte).
<i>nearlabel</i>	:	nhãn gần (2 byte OFFSET, trong đoạn).
<i>farlabel</i>	:	nhãn xa (4 byte SEGMENT:OFFSET, ngoài đoạn).
Src, dest	:	toán hạng nguồn, toán hạng đích
Left, right	:	toán hạng trái, toán hạng phải
Thn, thđ	:	viết tắt của toán hạng nguồn, toán hạng đích

## NHÓM LỆNH DI CHUYỂN DỮ LIỆU

### ■ Lệnh MOV

#### ■ Dạng lệnh : MOV thđ, thn

- MOV reg,reg                      MOV reg,immed
- MOV mem,reg                    MOV mem,immed
- MOV reg,mem                    MOV mem16,segreg
- MOV reg16,segreg              MOV segreg,mem16
- MOV segreg,reg16


#### ■ Giải thích : thđ $\leftarrow$ thn

#### ■ Tác động cờ : | | | | | | | | | |----|----|----|----|----|----|----|----| | OF | DF | IF | SF | ZF | AF | PF | CF | | | | | | | | | |

#### ■ Chép toán hạng nguồn vào toán hạng đích.

### Ví dụ :


- MOV AX,CX                      ; AX  $\leftarrow$  CX
- MOV DL,BH                    ; DL  $\leftarrow$  BH
- MOV [SI+1000h],BP          ; [SI+1001h, SI+1000h]  $\leftarrow$  BP
- MOV DX,[1000h]              ; DX  $\leftarrow$  [1001h,1000h]
- MOV DX,DS                    ; DX  $\leftarrow$  DS
- MOV ES,BX                    ; ES  $\leftarrow$  BX
- MOV DI,12h                   ; DI  $\leftarrow$  12h
- MOV AL,12h                   ; AL  $\leftarrow$  12h
- MOV BYTE PTR [1000h],12h   ; [1000h]  $\leftarrow$  12h
- MOV WORD PTR [2000h],1200h ; [2001h,2000h]  $\leftarrow$  1200h
- MOV [BX],DS                   ; [BX+1,BX]  $\leftarrow$  DS
- MOV SS,[2000h]               ; SS  $\leftarrow$  [2001h,2000h]



## ■ Lệnh PUSH : PUSH thn

- Dạng lệnh :
  - PUSH reg16            PUSH segreg
  - PUSH mem16
- Giải thích :
  - $SP \leftarrow SP - 2$
  - $[SS:SP+1, SS:SP] \leftarrow thn$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Đẩy toán hạng nguồn 16 bit vào đỉnh stack (địa chỉ đỉnh stack được xác định bởi SS:SP).




## ■ Ví dụ :

- PUSH DI    ;  $[SS:SP+1, SS:SP] \leftarrow DI$
- PUSH CS    ;  $[SS:SP+1, SS:SP] \leftarrow CS$
- PUSH [SI]   ;  $[SS:SP+1, SS:SP] \leftarrow [SI+1, SI]$


## ■ Lệnh POP : POP thđ

- Dạng lệnh :
  - POP reg16 POP segreg
  - POP mem16
- Giải thích :
  - $thđ \leftarrow [SS:SP+1, SS:SP]$
  - $SP \leftarrow SP + 2$




- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
- Lấy dữ liệu từ đỉnh stack đưa vào toán hạng đích.
- Ví dụ :
  - POP AX ; AX  $\leftarrow$  [SS:SP+1,SS:SP]
  - POP ES ; ES  $\leftarrow$  [SS:SP+1,SS:SP]
  - POP [BX+1] ; [BX+2,BX+1]  $\leftarrow$  [SS:SP+1,SS:SP]




- **Lệnh XCHG : XCHG thđ, thn**
  - Dạng lệnh :
    - XCHG reg,reg XCHG mem,reg
    - XCHG accum,reg16 XCHG reg,mem
  - Giải thích : thđ  $\leftrightarrow$  thn
  - Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  - Trao đổi nội dung hai toán hạng nguồn và đích cho nhau.
  - Ví dụ :
    - XCHG AX,CX ; AX  $\leftrightarrow$  CX
    - XCHG AH,AL ; AH  $\leftrightarrow$  AL
    - XCHG [1000h],DX ; [1001h,1000h]  $\leftrightarrow$  DX



- **Lệnh IN : IN AL, IOPort**
  - **Dạng lệnh :**
    - IN accum,immed8
    - IN accum,DX
  - **Giải thích :** Btl  $\leftarrow$  [cổng IO]
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
  - **Nhập dữ liệu từ cổng xuất nhập vào thanh ghi bộ tích lũy (Btl) AL hay AX. Trường hợp AX sẽ nhập byte thấp trước, byte cao sau.**



- **Dạng lệnh có immed8 dùng trong trường hợp địa chỉ cổng xuất nhập 8 bit.**
- **Ví dụ :**
  - IN AL,61h
  - IN AX,40h
- **Dạng lệnh có thanh ghi DX dùng cho trường hợp địa chỉ cổng 16 bit. Tuy nhiên dạng này vẫn có thể dùng cho cổng xuất nhập có địa chỉ 8 bit và có lợi khi sử dụng địa chỉ cổng để nhập nhiều lần.**
- **Ví dụ :**
  - MOV DX,378h
  - IN AL,DX



## ■ Lệnh OUT : OUT Ioport, AL

- Dạng lệnh :
  - OUT immed8,accum
  - OUT DX,accum
- Giải thích : [cổng IO]  $\leftarrow$  btl
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Xuất dữ liệu từ thanh ghi bộ tích lũy AL hoặc AX ra cổng xuất nhập có địa chỉ 8 bit là số tức thời immed8 hay có địa chỉ 16 bit trong thanh ghi DX.
- Ví dụ :
  - OUT 20h,AL
  - MOV DX,2F8h      OUT DX,AL

## ■ Lệnh XLAT :

- Dạng lệnh : XLAT
- Giải thích : AL  $\leftarrow$  [DS:BX+AL]
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Tra bảng. Thanh ghi BX giữ địa chỉ đầu bảng. Thanh ghi AL giữ chỉ số của phần tử cần lấy ra.
- Lệnh XLAT có ứng dụng trong mã hóa dữ liệu.

**Ví dụ :**

- Bài toán tính bình phương một số nguyên có thể thực hiện bằng cách tra bảng như sau :

```

MOV    CX,1000h
MOV    DS,CX
MOV    BX,2000h ; địa chỉ đầu bảng
MOV    AL,5      ; chỉ số
XLAT    ; tra bảng

```

-----

Sau khi làm xong lệnh XLAT :


$AL = 25 = 5^2$

0	0
1	1
4	2
9	3
16	4
25	5
36	6
49	7
64	8
81	9

**Lệnh LEA :** LEA thđ, src

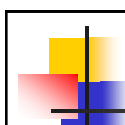
- Dạng lệnh : LEA reg16,mem
- Giải thích : thđ ← địa chỉ offset src
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Nạp địa chỉ offset của src vào thanh ghi 16 bit.
- Ví dụ :
  - LEA BX,[1000h] ; BX ← 1000h
  - LEA SI,[DI][BX][2000h] ; SI ← DI+BX+2000h




- **Lệnh LDS :**
  - **Dạng lệnh :** LDS thđ, mem32
  - **Giải thích :**
    - $DS \leftarrow [mem32+3, mem32+2]$
    - $thđ \leftarrow [mem32+1, mem32]$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
  - **Nạp 4 byte bộ nhớ (con trỏ) vào thanh ghi DS và một thanh ghi tổng quát (thđ).**
  - **Ví dụ :**
    - LDS BX,[1000h] ;  $DS \leftarrow [1003h, 1002h]$   
;  $BX \leftarrow [1001h, 1000h]$



- **Lệnh LES :**
  - **Dạng lệnh :** LES thđ, mem32
  - **Giải thích :**
    - $ES \leftarrow [mem32+3, mem32+2]$
    - $thđ \leftarrow [mem32+1, mem32]$
  - **Tác động cờ :**

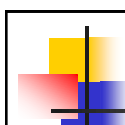
OF	DF	IF	SF	ZF	AF	PF	CF
  - **Nạp 4 byte bộ nhớ (con trỏ) vào thanh ghi ES và một thanh ghi tổng quát.**
  - **Ví dụ :** LES DI,[1000h] ;  $ES \leftarrow [1003h, 1002h]$   
;  $DI \leftarrow [1001h, 1000h]$



- **Lệnh LAHF :**
  - Dạng lệnh : LAHF
  - Giải thích :  $AH \leftarrow \text{FlagsL}$
  - Tác động cờ : 


OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  - Nạp 8 bit thấp của thanh ghi cờ vào thanh ghi AH.
- **Lệnh SAHF :**
  - Dạng lệnh : SAHF
  - Giải thích :  $\text{FlagsL} \leftarrow AH$
  - Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  - Cất thanh ghi AH vào 8 bit thấp của thanh ghi cờ.



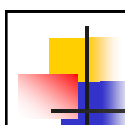
- **Lệnh PUSHF :**
  - Dạng lệnh : PUSHF
  - Giải thích :
    - $SP \leftarrow SP - 2$
    - $[SS:SP+1, SS:SP] \leftarrow \text{Flags}$
  - Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  - Đẩy thanh ghi cờ vào đỉnh stack.



- **Lệnh POPF :**
  - **Dạng lệnh :** POPF
  - **Giải thích :**
    - $\text{Flags} \leftarrow [\text{SS}:\text{SP}+1, \text{SS}:\text{SP}]$
    - $\text{SP} \leftarrow \text{SP} + 2$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x	x	x	x	x	x	x	x
  - **Lấy thanh ghi cờ từ đỉnh stack ra thanh ghi cờ.**




## Nhóm lệnh số học

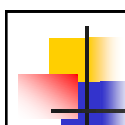
- **Lệnh ADD : ADD thđ, thn**
  - **Dạng lệnh :**

■ ADD reg,reg	ADD reg,immed
■ ADD mem,reg	ADD mem,immed
■ ADD reg,mem	ADD accum,immed
  - **Giải thích :**  $\text{thđ} \leftarrow \text{thđ} + \text{thn}$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x
  - **Cộng toán hạng nguồn vào toán hạng đích. Kết quả cất vào toán hạng đích.**
  - **Gọi là lệnh cộng không nhớ**



- Ví dụ :
  - `ADD CX,SI` ;  $CX \leftarrow CX + SI$
  - `ADD DH,BL` ;  $DH \leftarrow DH + BL$
  - `ADD [1000h],BX`  
;  $[1001h,1000h] \leftarrow [1001h,1000h] + BX$
  - `ADD [2000h],CL` ;  $[2000h] \leftarrow [2000h] + CL$
  - `ADD AL,[0000h]` ;  $AL \leftarrow AL + [0000h]$
  - `ADD BYTE PTR [SI+8],5` ;  $[SI+8] \leftarrow [SI+8] + 05h$



- **Lệnh ADC : (lệnh cộng có nhớ)**
  - **Dạng lệnh :** `ADC thđ, thn`
    - `ADC reg,reg`                      `ADC reg,immed`
    - `ADC mem,reg`                    `ADC mem,immed`
    - `ADC reg,mem`                    `ADC accum,immed`
  - **Giải thích :**  $thđ \leftarrow thđ + thn + CF$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x
  - Cộng toán hạng đích với toán hạng nguồn với cờ nhớ. Kết quả cất vào toán hạng đích.
  - ADC dùng cho phép cộng 2 số có chiều dài nhiều byte.
  - Ví dụ :
    - `ADC BX,AX` ;  $BX \leftarrow BX + AX + CF$
    - `ADC BYTE PTR [1000h],7Ah` ;  $[1000h] \leftarrow [1000h] + 7Ah + CF$



### ■ Lệnh INC : INC thđ

- Dạng lệnh : INC reg INC mem
- Giải thích :  $thđ \leftarrow thđ + 1$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	
- Tăng tức là cộng 1 vào toán hạng đích nhưng không ảnh hưởng cờ nhớ.
- Ví dụ :
  - INC CH
  - INC WORD PTR [1000h]



### ■ Lệnh AAA (ASCII Adjust for Addition):

- Dạng lệnh : AAA
- Cách làm việc
 

if low nibble of AL > 9 or AF = 1 then:

$$AL = AL + 6$$

$$AH = AH + 1$$


$$AF = 1$$

$$CF = 1$$

else

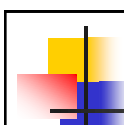
$$AF = 0$$

$$CF = 0$$
- in both cases: clear the high nibble of AL.



- Tác động cờ : 


OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×
- Chỉnh ASCII sau phép cộng. Chỉnh kết quả trong AL thành 2 số BCD không nén trong AH và AL.
- Ví dụ : kết quả : AH=00, AL= 0Dh, AF=0, CF=0
- sau khi chỉnh: AH=01h, AL=03h, AF=1, CF=1



## Nhóm Lệnh cộng

- **Ví dụ :** Cộng hai số BCD dạng không nén 9 và 3
  - MOV AX, 9
  - MOV BX, 3 ; 9 và 3 là hai số BCD dạng không nén
  - ADD AL, BL ; cộng 9 và 3 cho kết quả 0Ch trong AL
  - AAA ; chỉnh kết quả, làm cho AH=1 và AL =2, CF=1, AF=1
- **Ví dụ :** Trong đoạn được chỉ bởi DS cho 3 vùng nhớ var1, var2, var3. Vùng var1 và var2 dài 5 byte, mỗi vùng lưu trữ 1 số BCD dạng không nén gồm 5 ký số, vùng var3 dài 6 byte. Cộng var1 và var2, kết quả lưu vào var3 dưới dạng BCD không nén
  - Var1 DB 9, 2, 6, 4, 6
  - Len EQU \$-Var1 ;Len=Số Byte Định Nghĩa Cho Var1
  - Var2 DB 3, 4, 5, 6, 7
  - Var3 DB Len+1 DUP(0)






## Nhóm Lệnh cộng


---

- MOV BX, Len-1 ;BX=4
- MOV CX, Len ;CX=5
- CONT :
- MOV AL, Var1[BX]
- ADC AL,Var2[BX]
- AAA
- MOV Var3[BX+1],AL
- DEC BX
- LOOP CONT
- MOV AL,0
- ADC AL,AL
- MOV Var3[BX+1],AL



## ■ Lệnh DAA (Decimal Adjust for Addition):

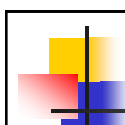
- Dạng lệnh : DAA
- Cách làm việc :
  - if low nibble of AL > 9 or AF = 1 then:
    - AL = AL + 6
    - AF = 1
  - if AL > 9Fh or CF = 1 then:
    - AL = AL + 60h
    - CF = 1
- Ví dụ : cộng 2 số BCD nén 55 và 26 cho nhau




---

- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×
- Chỉnh thập phân sau phép cộng. Chỉnh kết quả trong AL thành số BCD nén trong AL.



## Nhóm Lệnh cộng

---

- **Ví dụ :** Cộng hai số BCD 55h và 26h dạng nén
  - MOV AX, 5526H
  - ADD AL, AH
  - DAA ; Cho kết quả số BCD dạng nén là 81
- **Giải thích :** Cộng hai số BCD dạng nén 26 và 55 ta được
  - 00100110
  - 01010101
  - 01111011
- Kết quả nhị phân 7Bh (123), muốn chuyển sang BCD dạng nén, phải điều chỉnh bằng DAA, ta có AL=7Bh
- Vì AL and 0Fh = 0Bh > 9 nên AL=AL+6=7Bh+6=81h và AF=1
- Sau đó AL < 9F h nên kết quả là 81h



## Nhóm Lệnh cộng

- **Ví dụ :** Cộng hai số BCD dạng nén 59 và 63
  - MOV AX, 5963H
  - ADD AL,AH
  - DAA ; Cho kết quả số BCD dạng nén là 122
  - **Giải thích :** Cộng hai số 63h và 59h, ta được
    - 01100011
    - 01011001
    - 10111100
  - Kết quả nhị phân là số BCh, để chuyển sang BCD dạng nén, phải qua điều chỉnh DAA, ta có AL = 0BCh
  - Lần 1 : AL and 0Fh = 0C > 9 nên AL=AL+6=0BCh + 6 = C2h
  - Lần 2 : AL > 9Fh nên AL =AL+60h= C2h+60h=22h và CF=1
  - Kết quả số BCD dạng nén là 122



## ■ Lệnh SUB : SUB thđ, thn

- **Dạng lệnh :**
  - SUB reg,reg SUB reg,immed
  - SUB mem,reg SUB mem,immed
  - SUB reg,mem SUB accum,immed
- **Giải thích :** thđ ← thđ – thn
- **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x				x	x	x	x
- Trừ toán hạng đích cho toán hạng nguồn. Kết quả cất vào toán hạng đích.
- **Ví dụ :**
  - SUB DL,AL ; DL ← DL - AL
  - SUB CX,[DI] ; CX ← CX - [DI+1,DI]
  - SUB BP,4 ; BP ← BP - 4

## ■ **Lệnh SBB** : SBB thđ, thn

### ■ Dạng lệnh :

- SBB reg,reg            SBB reg,immed
- SBB mem,reg        SBB mem,immed
- SBB reg,mem        SBB accum,immed

### ■ Giải thích : thđ $\leftarrow$ thđ - thn - CF

### ■ Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x

### ■ Trừ toán hạng đích cho toán hạng nguồn và cờ nhớ. Kết quả cất vào toán hạng đích.

### ■ Ví dụ :

- SBB SI,BX ; SI  $\leftarrow$  SI - BX - CF
- SBB BYTE PTR [BX],2 ; [BX+1,BX]  $\leftarrow$  [BX+1,BX] - 2 - CF

## ■ **Lệnh DEC** : DEC thđ

### ■ Dạng lệnh : DEC reg                    DEC mem

### ■ Giải thích : thđ $\leftarrow$ thđ - 1


### ■ Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	

### ■ Giảm tức là trừ 1 vào toán hạng đích nhưng không ảnh hưởng cờ nhớ.


### ■ Ví dụ :

- DEC AX
- DEC BYTE PTR [SI][2000h]



- **Lệnh NEG : NEG thđ**
  - **Dạng lệnh :** NEG reg                      NEG mem
  - **Giải thích :** thđ  $\leftarrow$  bù 2(thđ)
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x
  - **Lấy bù 2 toán hạng đích.**
- **Lệnh CMP : CPM thđ, thn**
  - **Dạng lệnh :**
    - CMP reg,reg              CMP reg,immed
    - CMP mem,reg            CMP mem,immed
    - CMP reg,mem            CMP accum,immed



- **Cách làm việc :** thđ – thn  $\rightarrow$  kq, dùng kq để thiết lập các cờ
- **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x
- **So sánh.** Thực hiện trừ toán hạng đích cho toán hạng nguồn, không lưu lại kết quả mà chỉ giữ lại tác động của phép trừ lên các cờ.
- **Ví dụ :**
  - CMP AL,8 ; AL - 8
  - CMP WORD PTR [1000h], 3 ; [1001h,1000h] – 3

### ■ Lệnh AAS (ASCII Adjust for subtraction):

- Dạng lệnh : AAS
- Cách làm việc
- if low nibble of AL > 9 or AF = 1 then:
  - AL = AL - 6
  - AH = AH - 1
  - AF = 1
  - CF = 1
- else
  - AF = 0
  - CF = 0
- in both cases : clear the high nibble of AL.

### ■ ví dụ : trừ 2 số BCD không nén 13 và 4 cho nhau

- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×
- Chỉnh ASCII sau phép trừ. Chỉnh kết quả trong AL thành 2 số BCD không nén trong AH và AL.
- Ví dụ : kết quả : AH=00h, AL= 0Dh, AF=0, CF=0
- sau khi chỉnh: AH=01h, AL=03h, AF=1, CF=1




## Nhóm Lệnh trừ

- **Ví dụ :** Trừ hai số BCD không nén 13 và 4
  - MOV AX, 0103H
  - MOV BX, 04
  - SUB AL,BL ; lấy 3-4, kết quả -1 (0FFh)
  - AAS ; chỉnh số nhị phân 0FFh trong AL thành số BCD không nén 0 và 9 tương ứng trong AH và AL



## ■ Lệnh DAS (Decimal Adjust for Subtraction):

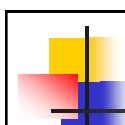
- Dạng lệnh : DAS
- Cách làm việc :
  - if low nibble of AL > 9 or AF = 1 then:
    - AL = AL - 6
    - AF = 1
  - if AL > 9Fh or CF = 1 then:
    - AL = AL - 60h
    - CF = 1
- Ví dụ : trừ 2 số BCD nén 57 và 28 cho nhau




---

OF DF IF SF ZF AF PF CF  
 ?          ×    ×    ×    ×    ×

- Tác động cờ :
- Chỉnh thập phân sau phép trừ. Chỉnh kết quả trong AL thành số BCD nén trong AL.
- Ví dụ : kết quả của  $(4 - 8)$  : AL= 0FCh, AF=1, CF=1
- sau khi chỉnh : AL=96h, AF=1, CF=1



## Nhóm Lệnh trừ

---

- **Ví dụ :** Trừ hai số BCD dạng nén 57 và 28
  - MOV AX, 2857H
  - SUB AL, AH
  - DAS ; Cho kết quả số BCD dạng nén là 29
- **Giải thích :** Trừ 2 số BCD dạng nén 57 và 28, ta được 2F, lưu trong AL, muốn chuyển sang số BCD dạng nén, ta phải dùng lệnh điều chỉnh DAS
- Ta có AL=2Fh, vì AL and 0Fh = Fh > 9
- Nên AL = AL -6 = 2F - 6 = 29 h, AL < 9Fh, nên kết quả là 29



## ■ Lệnh MUL : MUL thn

■ Dạng lệnh : MUL reg      MUL mem

■ Giải thích :

- Toán hạng nguồn 8 bit thì :  $AX \leftarrow AL * thn8$
- Toán hạng nguồn 16 bit thì :  $(DX : AX) \leftarrow AX * thn16$

■ Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			?	?	?	?	x

■ Nhân hai số không dấu 8 bit hay 16 bit. Số bit thực hiện được xác định bằng chiều dài của toán hạng nguồn.

- Phép nhân 8 bit : thực hiện nhân AL với toán hạng nguồn, kết quả 16 bit cất trong thanh ghi AX.
- Phép nhân 16 bit : thực hiện nhân AX với toán hạng nguồn, kết quả 32 bit cất trong 2 thanh ghi DX và AX. DX giữ 16 bit cao, AX giữ 16 bit thấp.

## ■ Ví dụ :

■ Nếu AL=5, CH=4, sau khi thực hiện lệnh MUL CH ta có  $AX = AL * CH = 5 * 4 = 20 = 0014h$ .

■ Nếu AX=500h, [1001h,1000h]=401h, sau khi thực hiện lệnh MUL WORD PTR [1000h]

ta có  $DXAX = AX * [1001h,1000h] = 500h * 401h = 00140500h$

Nghĩa là DX=0014h và AX=0500h.

### ■ Lệnh IMUL : IMUL thn

- Dạng lệnh : IMUL reg IMUL mem

- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			?	?	?	?	x

- Nhân hai số có dấu. Thực hiện giống hệt như lệnh MUL, chỉ có kết quả được xem là số có dấu.

### ■ Lệnh AAM (ASCII Adjust for Multiply):

- Dạng lệnh : AAM

- Giải thích :

- AH ← thương số của (AL/10)
- AL ← số dư của (AL/10)

- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
?			x	x	?	x	?

### ■ Ví dụ :

- MOV AL, 9
- MOV BL, 7
- MUL BL ; Cho kết quả AH=00 và AL=63
- AAM ; Điều chỉnh kết quả cho AH=6 và AL=3



- Chỉnh ASCII sau phép nhân. Có thể dùng để đổi số hex ra số BCD không nén.
- Ví dụ : kết quả : AH = 00, AL = 41h.
- sau khi chỉnh : AH = 06, AL = 05.
- Lệnh DIV : DIV thn
- Dạng lệnh : DIV reg      DIV mem
- Giải thích :
  - Toán hạng nguồn 8 bit thì : AL  $\leftarrow$  thương (AX / thn8)  
AH  $\leftarrow$  số dư của (AX / thn8)
  - Toán hạng nguồn 16 bit thì : AX  $\leftarrow$  thương (DX:AX / thn16)  
DX  $\leftarrow$  số dư của (DX:AX / thn16)



- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
?				?	?	?	?
- Chia hai số không dấu.
- Nếu toán hạng nguồn là thanh ghi hay bộ nhớ 8 bit, thực hiện chia số 16 bit trong thanh ghi AX cho toán hạng nguồn 8 bit. Kết quả 8 bit cất trong thanh ghi AL. Số dư 8 bit cất trong thanh ghi AH.
- Nếu toán hạng nguồn là thanh ghi hay bộ nhớ 16 bit, thực hiện chia số 32 bit trong 2 thanh ghi DX:AX cho toán hạng nguồn 16 bit. Kết quả 16 bit cất trong thanh ghi AX. Số dư 16 bit cất trong thanh ghi DX.



### ■ Ví dụ :

- Nếu  $AX=0024h$ ,  $[2000h]=05$  thì sau khi thực hiện lệnh `DIV BYTE PTR [2000h]`

ta có  $AX=0024h = 36 \rightarrow 36 : 5 \rightarrow$  thương 7, dư 1

$\rightarrow AL=07$  và  $AH=01$ .

- Nếu  $DX=0001h$ ,  $AX=0024h$ ,  $BX=0200h$  thì sau khi thực hiện lệnh `DIV BX`


ta có  $AX=0008$  và  $DX=0024h$ .

( $DX\ AX = 00010024\ h \rightarrow 00010024\ h : 0200h$ )



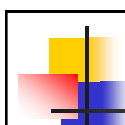
### ■ Lệnh IDIV

- Cú pháp : `IDIV Src`
- **Chức năng :** Chia số có dấu, kết quả là 1 số có dấu, lệnh này không ảnh hưởng tới thanh ghi cờ
  - Nếu  $Src$  là 8 bit thì :  $AX:Src \rightarrow (AL=Thương\ số, AH=Dư\ số)$
  - Nếu  $Src$  là 16 bit thì :  $(DX:AX):Src \rightarrow (AX=Thương, DX=Dư)$
- Các trường hợp xảy ra “divide overflow” đối với lệnh `IDIV` :
  - Thương số nằm ngoài khoảng  $(-128, +127)$  đối với phép chia cho  $src$  là 8 bit
  - Thương số nằm ngoài khoảng  $(-32768, +32767)$  đối với phép chia có dấu dạng word




- **Lệnh AAD (ASCII Adjust for Division) :**
  - **Dạng lệnh :** AAD
  - **Giải thích :**  $AL \leftarrow ((AH * 10) + AL)$   
 $AH \leftarrow 0$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
?			×	×	?	×	?
  - **Chỉnh ASCII trước phép chia DIV.** Có thể dùng lệnh này để đổi số BCD không nén trong AX ra thành giá trị nhị phân trong AL.
  - **Ví dụ :** nếu có : AL=03h, AH=05h
  - **sau khi chỉnh :** AL=35h, AH=00h




- **Ví dụ :** Tìm thương số của phép chia số BCD không nén 95 cho 3
  - `MOV AX, 0905H` ; Nạp số BCD 95 vào AX
  - `MOV BL, 3`
  - `AAD` ; Đổi số BCD không nén trong AX thành 5Fh
  - `DIV BL` ; Chia 5Fh cho 3, thương AL=1F và dư AH=2
  - `AAM` ; Chỉnh AL =1F thành số BCD không nén 31 trong AX



- **Lệnh CBW (Convert byte to word):**
  - **Dạng lệnh :** CBW
  - **Giải thích :** Nếu  $AL < 80h$  thì  $AH \leftarrow 00h$   
Nếu  $AL \geq 80h$  thì  $AH \leftarrow 0FFh$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
  - **Mở rộng dấu trước khi dùng lệnh chia.** Đổi số 1 byte có dấu trong AL thành số 2 byte có dấu trong AX.
- **Ví dụ (Đặt vấn đề) :**
  - Giả sử  $AL = -26$ ,  $BL = 5$ , ta muốn lấy  $-26$  chia cho 5
  - Không thể viết `IDIV BL` : sai, vì sao, làm sao khắc phục



- **Lệnh CWD (Convert Word to double Word):**
  - **Dạng lệnh :** CWD
  - **Giải thích :** Nếu  $AX < 8000h$  thì  $DX \leftarrow 0000h$   
Nếu  $AX \geq 8000h$  thì  $DX \leftarrow 0FFFFh$
  - **Tác động cờ :**

OF	DF	IF	SF	ZF	AF	PF	CF
  - **Mở rộng dấu trước khi dùng lệnh chia.** Đổi số 2 byte có dấu trong AX thành số 4 byte có dấu trong cặp DX AX.

## ■ Lệnh SHL/SAL : SHL/SAL thđ, 1/CL

- Dạng lệnh : SHL reg,1                      SHL mem,1  
SHL reg,CL                      SHL mem,CL
- Giải thích : thđ  $\leftarrow$  (thđ) dịch trái 1 hay nhiều bit.
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x
- Dịch trái. Dạng SHL reg,1 dùng để dịch trái 1 bit.  
Dạng SHL reg,CL dùng để dịch trái nhiều bit. Lúc đó thanh ghi CL chứa số bit cần dịch.
- Ví dụ :
  - SHL DH,1
  - SAL CX,1
  - MOV CL,3
  - SHL WORD PTR [1000h],CL ; dịch trái 3 bit.



## ■ Lệnh SHR : SHR thđ, 1/CL

- Dạng lệnh : SHR reg,1                      SHR mem,1  
SHR reg,CL                      SHR mem,CL
- Giải thích : thđ  $\leftarrow$  (thđ) dịch phải luận lý 1 hay nhiều bit.
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x
- Dịch phải luận lý. Dạng có thanh ghi CL dùng để dịch nhiều bit.
- Ví dụ :
  - SHR AX,1
  - MOV CL,2
  - SHR BYTE PTR [1000h],CL ; dịch phải luận lý 2 bit.



## ■ Lệnh SAR :

- Dạng lệnh : SAR reg,1                      SAR mem,1  
                         SAR reg,CL                      SAR mem,CL

- Giải thích :  $thđ \leftarrow (thđ) \text{ dịch phải số học 1 hay nhiều bit.}$

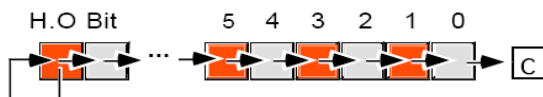
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x

- Dịch phải số học. Dạng có thanh ghi CL dùng để dịch nhiều bit.

- Ví dụ :

- SAR DX,1
- MOV CL,7
- SAR WORD PTR [2000h],CL ; dịch phải số học 7 bit.



## ■ Lệnh ROL :

- Dạng lệnh : ROL reg,1                      ROL mem,1  
                         ROL reg,CL                      ROL mem,CL

- Giải thích :  $thđ \leftarrow (thđ) \text{ quay trái không qua cờ nhớ 1 hay nhiều bit.}$

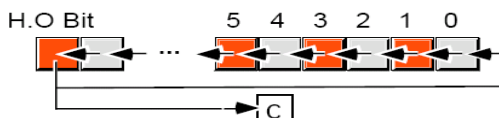
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x

- Quay trái không qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.

- Ví dụ :

- ROL DL,1
- MOV CL,6
- ROL WORD PTR [1000h],CL ; quay trái không qua cờ nhớ 6 bit.





## ■ Lệnh ROR :

- Dạng lệnh : ROR reg,1                  ROR mem,1  
                  ROR reg,CL                ROR mem,CL

- Giải thích :  $thđ \leftarrow (thđ) \text{ quay phải không qua cờ nhớ 1 hay nhiều bit.}$

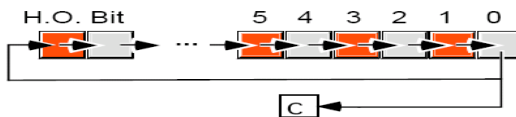
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x

- Quay phải không qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.

- Ví dụ :

- ROR SI,1
- MOV CL,3
- ROR WORD PTR [3000h],CL ; quay phải không qua cờ nhớ 3 bit



## ■ Lệnh RCL :

- Dạng lệnh : RCL reg,1                  RCL mem,1  
                  RCL reg,CL                RCL mem,CL

- Giải thích :  $thđ \leftarrow (thđ) \text{ quay trái qua cờ nhớ 1 hay nhiều bit.}$

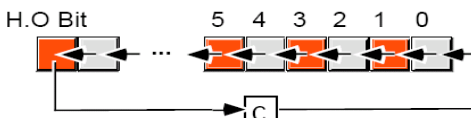
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x

- Quay trái qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.

- Ví dụ :

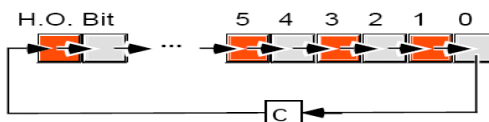
- RCL BX,1
- MOV CL,4
- RCL BYTE PTR [1000h],CL ; quay trái qua cờ nhớ 4 bit.



## ■ Lệnh RCR :

- Dạng lệnh : RCR reg,1                  RCR mem,1  
                                 RCR reg,CL                  RCR mem,CL
- Giải thích :  $thđ \leftarrow (thđ) \text{ quay phải qua cờ nhớ 1 hay nhiều bit.}$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	x
- Quay phải qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.
- Ví dụ :
  - RCR CH,1
  - MOV CL,2
  - RCR BYTE PTR [1800h],CL ; quay phải qua cờ nhớ 2 bit.



## Nhóm lệnh luận lý

### ■ Lệnh NOT : NOT thđ

- Dạng lệnh : NOT reg      NOT mem
- Giải thích :  $thđ \leftarrow \text{bù 1 của thđ}$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Đảo hay lấy bù 1.
- Ví dụ : NOT AL  
                  NOT WORD PTR [BX+1000h]

## ■ Lệnh AND : AND thđ, thn

- Dạng lệnh : AND reg,reg    AND reg,immed  
AND mem,reg    AND mem,immed  
AND reg,mem    AND accum,immed

- Giải thích : thđ  $\leftarrow$  thđ AND thn.

- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	0

- Và luận lý. Xóa cờ nhớ về 0.

- Ví dụ :

- AND CH,AH
- AND [SI],DX
- AND BYTE PTR [1000h],10000000b
- AND AX,0FFF0h

## ■ Lệnh TEST :

- Dạng lệnh : TEST reg,reg    TEST reg,immed  
TEST mem,reg    TEST mem,immed  
TEST reg,mem    TEST accum,immed

- Giải thích : thđ AND thn.


- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
0			x	x	?	x	0

- Và luận lý hai toán hạng nhưng không giữ lại kết quả mà chỉ lập các cờ. Xóa cờ nhớ và cờ tràn về 0. Thường dùng để kiểm tra bit. Lúc đó toán hạng nguồn là một mặt nạ bit cần thiết.

- Ví dụ :

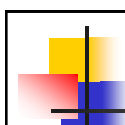
- TEST DX,1 ; kiểm tra bit 0
- TEST BYTE PTR [2000h],10000000b ; kiểm tra bit 7



■ **Lệnh OR : OR thđ, thn**

- Dạng lệnh :   OR reg,reg           OR reg,immed  
                  OR mem,reg       OR mem,immed  
                  OR reg,mem       OR accum,immed
- Giải thích : thđ  $\leftarrow$  thđ OR thn.
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	0
- Hay luận lý. Xóa cờ nhớ về 0.
- Ví dụ : OR DL,CH  
          OR BP,[2000h]  
          OR WORD PTR [1000h],000Fh



■ **Lệnh XOR : XOR thđ, thn**

- Dạng lệnh :   XOR reg,reg           XOR reg,immed  
                  XOR mem,reg       XOR mem,immed  
                  XOR reg,mem       XOR accum,immed
- Giải thích : thđ  $\leftarrow$  thđ XOR thn.
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	?	x	0
- Hay ngoại luận lý. Xóa cờ nhớ về 0.
- Ví dụ : XOR DL,80h ; đảo bit 7  
          XOR [2000h],AL ; [2000h]  $\leftarrow$  [2000h] XOR AL






- **nearLabel** : nhãn gần – là nhãn nằm trong phạm vi cùng segment với segment chứa lệnh JMP, phạm vi [-32768 32767] bytes
  - JMP Nhan
  - Mã máy chiếm 3 bytes trong bộ nhớ
  - Nhảy tới vị trí được chỉ bởi Nhan
  - Chỉ thay đổi nội dung thanh ghi IP, không thay đổi nội dung CS
- **farLabel** : nhãn xa – là nhãn nằm trong segment khác với segment chứa lệnh JMP
  - JMP far Nhan
  - Mã máy chiếm 5 bytes trong bộ nhớ
  - Nhảy tới vị trí được chỉ bởi Nhan
  - Thay đổi cả IP và CS




- **Xét JMP target**
  - Nhảy tới vị trí được chỉ bởi đích target
  - Target có thể là thanh ghi hay bộ nhớ
  - VD : JMP AX ; AX → IP, nhảy tới vị trí được xác định trong CS:IP
  - VD : JMP WORD PTR [BX] ; [BX+1,BX] → IP, nhảy tới vị trí được xác định trong CS:IP
  - VD : JMP DWORD PTR [BX] ; [BX+3, BX+2] → CS, [BX+1,BX] → IP, nhảy tới vị trí được cho bởi CS:IP



## Lệnh Call

- gọi và thực hiện chương trình con tại vị trí được xác định bởi Nhan hoặc target
- Cú pháp : CALL Nhan    hoặc CALL target
- Cho dù gọi tới nhãn hay đích (target) thì cũng rơi vào 1 trong 2 trường hợp : Near hay far.
- Xét CALL Nhan
  - Dạng Near :
    - Chiếm 2 bytes trong bộ nhớ
    - Nhan phải nằm trong cùng segment với segment chứa lệnh CALL



## Hoạt động :

- Push IP → stack
- Offset (Nhan) → IP
- Thi hành lệnh tại địa chỉ CS:IP
- Quay về lệnh kế tiếp sau lệnh call (POP stack → IP)

■ Ví dụ :



■ 1100:03E0		MOV DX, OFFSET Str
■ 1100:03E3		CALL PrintSt
■ 1100:03E6	Next :	INT 20h
■ 1100:03E8	Str	DB 'KHOA TIN HOC'
-----		
■ 1100:0600	PrintSt	PROC NEAR
■		MOV AH, 09H
■		INT 21H
■		RET
■	PrintSt	ENDP




### ■ Dạng far

- Chiếm 5 bytes trong bộ nhớ
- Nhan nằm trong segment khác với segment chứa lệnh CALL

### ■ Hoạt động :


- Push CS và IP → stack
- Nạp địa chỉ : Segment(Nhan) → CS, Offset (Nhan) → IP
- Thi hành lệnh tại địa chỉ CS:IP
- Quay về lệnh kế tiếp sau lệnh call (POP stack → IP, CS)






**Ví dụ :**

■ 1100:03E0		MOV DX, OFFSET Str
■ 1100:03E3		CALL FAR PTR PrintSt
■ 1100:03E6	Next :	INT 20h
■ 1100:03E8	Str	DB 'KHOA TIN HOC'
■	-----	
■ 4100:0600	PrintSt	PROC FAR
■		MOV AH, 09H
■		INT 21H
■		RET
■	PrintSt	ENDP




- Xét CALL target
- VD : CALL BX
  - Push IP→Stack
  - BX → IP
  - Thi hành chương trình tại địa chỉ CS:IP
  - Quay về lệnh kế tiếp
- VD : CALL WORD PTR[BX]
  - Push IP→Stack
  - [BX] → IP, thực thi chương trình tại địa chỉ CS:IP
- VD : CALL DWORD PTR[BX]
  - Push CS, IP →Stack
  - [BX+3, BX+2] →CS, [BX+1,BX] →IP



## Lệnh gọi ngắt INT

---

- Người ta đã viết sẵn các chương trình cho phép người dùng sử dụng, đó là các ngắt.
- Gọi thông qua lệnh INT n – trong đó n là số hiệu ngắt
- Mỗi ngắt có nhiều chức năng khác nhau, mỗi chức năng được đánh số
- Khi thực hiện sẽ tác động lên dữ liệu nào đó
- Kết quả được cho ở một nơi thích hợp
- Vì vậy trước khi gọi ngắt, phải thực hiện các thao tác chuẩn bị gồm :




---

- Đưa chức năng tương ứng vào thanh ghi AH
- Đưa dữ liệu cần tác động vào nơi thích hợp
- Chuẩn bị nơi chứa kết quả
- Gọi ngắt INT n
- Cách làm việc của lệnh INT
  - Push thanh ghi cờ vào stack
  - Cất nội dung thanh ghi CS, IP vào stack
  - Nạp địa chỉ chương trình con phục vụ ngắt vào CS:IP




- Số ngắt cũng theo qui ước của hệ thống như sau :
  - $00h \div 07h$  : ngắt hệ thống.
  - $08h \div 0Fh, 70h \div 77h$  : ngắt cứng.
  - Còn lại : ngắt mềm.
- Một số ngắt thông dụng :
  - INT 10h : màn hình.
  - INT 13h : đĩa.
  - INT 14h : thông tin liên lạc.
  - INT 16h : bàn phím.
  - INT 17h : máy in
  - INT 21h : các phục vụ của MS-DOS.
  - INT 20h : kết thúc chương trình, trở về DOS.




## Lệnh nhảy có điều kiện

- Cú pháp : J<ĐK> Nhan
- Nhảy tới Nhan nếu điều kiện thỏa
- Nhan : là nhãn gần
- Ví dụ :
  - JC Nhan
  - JZ
  - JNC
  - JNZ
  - JG
  - JL



Mã lệnh	Giải thích	Điều kiện
JE/JZ	Nhảy nếu bằng/không	ZF = 1
JL/JNGE	Nhảy nếu nhỏ hơn/không lớn hơn hoặc bằng	(SF xor OF) = 1
JLE/JNG	Nhảy nếu nhỏ hơn hoặc bằng /không lớn hơn	((SF xor OF) or ZF) = 1
JB/JNAE/JC	Nhảy nếu dưới /không trên hoặc bằng/nhỏ	CF = 1
JBE/JNA	Nhảy nếu dưới hoặc bằng /không trên	(CF or ZF) = 1
JP/JPE	Nhảy nếu kiểm tra / kiểm tra chẵn	PF = 1
JO	Nhảy nếu tràn	OF = 1
JS	Nhảy nếu dấu	SF = 1
JNE/JNZ	Nhảy nếu không bằng/khác không	ZF = 0
JNL/JGE	Nhảy nếu không nhỏ hơn/lớn hơn hoặc bằng	(SF xor OF) = 0
JNLE/JG	Nhảy nếu không nhỏ hơn hoặc bằng /lớn hơn	((SF xor OF) or ZF) = 0
JNB/JAE/JNC	Nhảy nếu không dưới /trên hoặc bằng/không nhỏ	CF = 0
JNBE/JA	Nhảy nếu không dưới hoặc bằng /trên	(CF or ZF) = 0
JNP/JPO	Nhảy nếu không kiểm tra / kiểm tra lẻ	PF = 0
JNO	Nhảy nếu không tràn	OF = 0
JNS	Nhảy nếu không dấu	SF = 0

- 
- Ví dụ : MOV CX,3 ; thực hiện một vòng lặp làm 3 lần.


```


MOV AX,0
Nhan:  ADD AX,12
        DEC CX
        JNZ Nhan ; nhảy đến lệnh tại vị trí "Nhan" nếu CX ≠ 0.
        MOV [3000h],AX

```

### ■ Lệnh RET :

- Dạng lệnh :     RET                     RETF  
                   RET immedi8       RETF immedi8
- Giải thích :
- RET :           POP IP
- RETF :          POP IP  
                   POP CS
- RET immedi8 :   POP IP  
                                  SP ← SP + immedi8
- RETF immedi8 : POP IP  
                                  POP CS  
                                  SP ← SP + immedi8

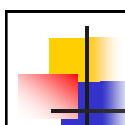
- 
- | OF | DF | IF | SF | ZF | AF | PF | CF |
|----|----|----|----|----|----|----|----|
|    |    |    |    |    |    |    |    |
- Tác động cờ :
  - Trở về từ chương trình con. Lệnh trở về là lệnh dùng để kết thúc một chương trình con.
  - Lệnh RET để kết thúc một chương trình con gần.
  - Lệnh RETF để kết thúc một chương trình con xa.
  - Dạng lệnh trở về có toán hạng immedi8 dùng cho các chương trình con có sử dụng thông số trong chồng. Khi đó, toán hạng nguồn immedi8 sẽ được cộng vào thanh ghi SP để chỉnh lại vị trí đỉnh chồng sau khi gọi chương trình con, tránh thất thoát bộ nhớ dùng cho chồng.



## ■ Lệnh IRET

- Giống lệnh RET
- Trở về chương trình chính từ chương trình con phục vụ ngắt
- Hoạt động
  - POP IP
  - POP CS
  - POPF
- Tác động cờ :
 

OF	DF	IF	SF	ZF	AF	PF	CF
x	x	x	x	x	x	x	x



## ■ Lệnh lặp

- Gồm : LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ
- Lệnh LOOP :
  - Dạng lệnh 1 : LOOP shortlabel
  - Giải thích : giảm CX, lặp vòng (nhảy) nếu  $CX \neq 0$   
 $IP \leftarrow \text{địa chỉ lệnh kế} + \text{độ dời (mở rộng dấu 16 bit)}$
  - Tác động cờ :
 

OF	DF	IF	SF	ZF	AF	PF	CF
  - Lặp vòng không điều kiện. CX giữ số lần lặp. Rất tiện dụng trong việc tạo ra các vòng lặp. Chẳng hạn như ví dụ trong phần lệnh nhảy có điều kiện có thể viết lại gọn hơn như sau :



```

MOV CX,3
MOV AX,0
Nhan:  ADD AX,12
        LOOP Nhan
        MOV [3000h],AX

```

- Một trong những ứng dụng phổ biến của lệnh lặp vòng là tạo ra các vòng làm trễ.

- Ví dụ :

```

MOV CX,10
Nhan:  LOOP Nhan ; vòng lặp làm 10 lần lệnh LOOP
MOV CX,0
Nhan:  LOOP Nhan ; vòng lặp làm 65536 lần lệnh LOOP

```



- Dạng lệnh 2 : LOOPE/LOOPZ shortlabel
- Giải thích : giảm CX, lặp vòng (nhảy) nếu  $CX \neq 0$  và  $ZF = 1$
- $IP \leftarrow \text{địa chỉ lệnh kế} + \text{độ dời (mở rộng dấu 16 bit)}$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
- Lặp vòng nếu bằng / nếu không. Hai điều kiện  $CX \neq 0$  và  $ZF = 1$  phải thỏa đồng thời thì lệnh mới lặp vòng. Nếu không, không làm gì cả (qua lệnh kế).
- Đôi khi người ta xét các điều kiện để không lặp còn gọi là điều kiện thoát khỏi vòng lặp :  $CX = 0$  hay  $ZF = 0$ . Có thể xem đó là các nguyên nhân gây ra kết thúc vòng lặp.



- Dạng lệnh 3 : LOOPNE/LOOPNZ shortlabel
- Giải thích : giảm CX, lặp vòng (nhảy) nếu  $CX \neq 0$  và  $ZF = 0$
- $IP \leftarrow$  địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Lặp vòng nếu không bằng / nếu khác không. Hai điều kiện  $CX \neq 0$  và  $ZF = 0$  phải thỏa đồng thời thì lệnh mới lặp vòng. Nếu không, không làm gì cả (qua lệnh kế).
- Đôi khi người ta xét các điều kiện để không lặp còn gọi là điều kiện thoát khỏi vòng lặp :  $CX = 0$  hay  $ZF = 1$ . Có thể xem đó là các nguyên nhân gây ra kết thúc vòng lặp.



#### ■ Lệnh JCXZ :

- Dạng lệnh : JCXZ shortlabel
- Giải thích : Nếu  $CX = 0$  thì
- $IP \leftarrow$  địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Nhảy nếu  $CX=0$ . Thường dùng sau LOOPE, LOOPNE, REPE, REPNE để xác định nguyên nhân kết thúc vòng lặp.
- Ví dụ : REPE CMPSB ; so sánh 2 chuỗi  
JCXZ nhan  
( đoạn chương trình xử lý cho trường hợp chuỗi khác nhau)  
**nhan:** ( đoạn chương trình xử lý cho trường hợp chuỗi giống nhau)



## Nhóm lệnh xử lý chuỗi

- Các bước xử lý chuỗi
- B1 : đưa số phần tử cần xử lý vào thanh ghi CX
- B2 : Xác định hướng xử lý chuỗi thông qua cờ DF, cụ thể
  - Nếu cờ DF = 0 : xử lý theo hướng tăng dần địa chỉ
  - Nếu cờ DF = 1 : xử lý theo hướng giảm dần địa chỉ
- B3 : Đưa địa chỉ chuỗi nguồn vào cặp thanh ghi DS:SI, đưa địa chỉ chuỗi đích vào cặp thanh ghi ES:DI
- B4 : chọn tiếp đầu lệnh : REP, REPE/REPZ, REPNE/REPZ
- B5 : đặt lệnh xử lý chuỗi sau tiếp đầu lệnh

- Để cho cờ DF = 0 ta dùng lệnh CLD
- Để cho cờ DF = 1 ta dùng lệnh STD
- Các tiếp đầu lệnh gồm
  - REP
  - REPE/REPZ
  - REPNE/REPZ
- Các lệnh xử lý chuỗi gồm : MOVS, LODS, S



## Các tiếp đầu lệnh

### ■ Tiếp đầu lệnh REP :


- **Dạng 1** : REP lệnh xử lý chuỗi
- Giải thích : giảm CX, lặp lại lệnh theo sau Nếu CX  $\neq$  0
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Lặp lại không điều kiện. Thanh ghi CX giữ số lần lặp. Thường dùng với lệnh chép chuỗi MOVS.
- Ví dụ :
  - MOV CX,10
  - REP MOVSB ; thực hiện lệnh MOVSB 10 lần.




- **Dạng 2** : REPE / REPZ lệnh xử lý chuỗi
- Giải thích : Giảm CX, lặp lại lệnh theo sau Nếu CX  $\neq$  0 và ZF = 1
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Lặp lại nếu bằng / nếu không. Hai điều kiện CX  $\neq$  0 và ZF = 1 phải thỏa đồng thời thì lệnh theo sau mới được lặp lại. Nếu không, làm qua lệnh kế. Thường dùng với lệnh so sánh chuỗi CMPS để tìm chuỗi con trong chuỗi lớn.
- Ví dụ :
  - MOV CX,10
  - REPE CMPSB ; thực hiện lệnh CMPSB nếu chưa đủ 10 ; lần và hai chuỗi vẫn còn bằng nhau.



- Dạng 3 : REPNE / REPNZ lệnh xử lý chuỗi
- Giải thích : giảm CX, lặp lại lệnh theo sau Nếu CX  $\neq$  0 và ZF = 0
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF
- Lặp lại nếu bằng / nếu không. Hai điều kiện CX  $\neq$  0 và ZF = 1 phải thỏa đồng thời thì lệnh theo sau mới được lặp lại. Nếu không, làm qua lệnh kế. Thường dùng với lệnh quét chuỗi SCAS để dò tìm ký tự trong chuỗi.
- Ví dụ :
  - MOV CX,20
  - REPNE SCASB ; thực hiện lệnh CMPSB nếu chưa đủ 20; lần và chưa tìm ra ký tự trong chuỗi.



## Các lệnh xử lý chuỗi

- Lệnh MOVS :
- Công dụng : di chuyển (sao chép) nội dung chuỗi nguồn vào chuỗi đích
- Có 2 dạng : MOVSB và MOVSW
- Nội dung chuỗi nguồn được chỉ bởi cặp DS:SI, chuỗi đích được chỉ bởi cặp thanh ghi ES:DI
- MOVSB : [ES:DI]  $\leftarrow$  [DS:SI]
  - Mỗi lần di chuyển 1 byte
  - Nếu DF=0 thì : SI  $\leftarrow$  SI + 1, DI  $\leftarrow$  DI + 1
  - ngược lại thì : SI  $\leftarrow$  SI - 1, DI  $\leftarrow$  DI - 1
- MOVSW : [ES:DI+1,ES:DI]  $\leftarrow$  [DS:SI+1,DS:SI]
  - Mỗi lần di chuyển 2 byte
  - Nếu DF=0 thì : SI  $\leftarrow$  SI + 2, DI  $\leftarrow$  DI + 2
  - ngược lại thì : SI  $\leftarrow$  SI - 2, DI  $\leftarrow$  DI - 2
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF



- Ví dụ 1 chép byte : chép 100 byte từ địa chỉ 3000:1000 sang địa chỉ 4800:C200
  - MOV AX,3000h
  - MOV DS,AX
  - MOV SI,1000h ; địa chỉ chuỗi nguồn.
  - MOV AX,4800h
  - MOV ES,AX
  - MOV DI,0C200h ; địa chỉ chuỗi đích.
  - MOV CX,100 ; số lần chép.
  - CLD ; xóa cờ DF, xử lý tăng địa chỉ.
  - REP MOVSB




- Ví dụ 2 chép word : yêu cầu như ví dụ 1
  - MOV AX,3000h
  - MOV DS,AX
  - MOV SI,1000h ; địa chỉ chuỗi nguồn.
  - MOV AX,4800h
  - MOV ES,AX
  - MOV DI,0C200h ; địa chỉ chuỗi đích.
  - MOV CX,40h ; số lần chép.
  - CLD ; xóa cờ DF, xử lý tăng địa chỉ.
  - REP MOVSW

## Lệnh CMPS :

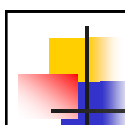
- Công dụng : so sánh nội dung chuỗi nguồn và nội dung chuỗi đích
- Gồm 2 dạng : CMPSB và CMPSW
- Địa chỉ chuỗi nguồn chứa trong cặp thanh ghi DS:SI, địa chỉ chuỗi đích chứa trong cặp ES:DI
- **CMPSB :**
  - [DS:SI] - [ES:DI]
  - Nếu DF=0 thì :  $SI \leftarrow SI + 1, DI \leftarrow DI + 1$
  - ngược lại thì :  $SI \leftarrow SI - 1, DI \leftarrow DI - 1$
- **CMPSW :**
  - [DS:SI+1,DS:SI] - [ES:DI+1,ES:DI]
  - Nếu DF=0 thì :  $SI \leftarrow SI + 2, DI \leftarrow DI + 2$
  - ngược lại thì :  $SI \leftarrow SI - 2, DI \leftarrow DI - 2$
- Tác động cờ :
 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x

- Ví dụ : kiểm tra xem 100 byte bắt đầu ở địa chỉ 3000:100 và 100 byte bắt đầu tại địa chỉ 4800:200 xem có giống nhau không ?
- Nguyên tắc : so sánh từng byte ở địa chỉ 3000:100 và 4800:200, nếu thấy vị trí nào khác nhau là ngưng và kết luận
- Chương trình
  - Mov CX,100
  - CLD




- Mov AX, 3000
- Mov DS, AX
- Mov SI, 100
- Mov AX, 4800
- Mov ES, AX
- Mov DI, 200
- REPE CMPSB
- JZ giống
- JMP khác
- Giống:
- Khác :




## NHÓM LỆNH XỬ LÝ CHUỖI

- **Ví dụ :** Tìm phần tử khác nhau đầu tiên của hai chuỗi Str1 và Str2 trong đoạn chỉ bởi DS, biết rằng Str1 và Str2 có cùng chiều dài 500 byte.
  - PUSH DS
  - POP ES
  - CLD
  - MOV CX, 500
  - MOV SI, OFFSET Str1
  - MOV DI, OFFSET Str2
  - REPE CMPSB
  - JCXZ Match
  - DEC SI
  - LODSB ; nạp phần tử khác nhau đầu tiên vào AL
  - -----
  - Match : ; hai chuỗi Str1 và Str2 bằng nhau



■ **Lệnh SCAS :**

- Quét để xem trong chuỗi đích xem có phần tử nào đó hay không
- Địa chỉ chuỗi đích chứa trong cặp thanh ghi ES:DI
- Phần tử cần tìm chứa trong AL hoặc AX
- Có 2 dạng : SCASB và SCASW
- Nếu SCASB
  - Cách làm việc : AL - [ES:DI]
  - Mỗi lần quét 1 byte, phần tử cần tìm chứa trong AL
  - Nếu DF=0 thì :  $DI \leftarrow DI + 1$
  - ngược lại thì :  $DI \leftarrow DI - 1$



■ **SCASW**

- Cách làm việc : AX - [ES:DI+1,ES:DI]
- Mỗi lần quét 2 byte, phần tử cần tìm chứa trong AX
- Nếu DF=0 thì :  $DI \leftarrow DI + 2$
- ngược lại thì :  $DI \leftarrow DI - 2$
- Tác động cờ :
 

OF	DF	IF	SF	ZF	AF	PF	CF
x			x	x	x	x	x

- **Ví dụ :** Tìm vị trí của ký tự 'a' trong vùng nhớ mem có chiều dài 100 byte
  - CLD ; thực hiện từ trái sang
  - MOV CX, 100
  - MOV AX, SEG Mem
  - MOV ES, AX
  - MOV DI, OFFSET Mem
  - MOV AL, 'a'
  - REPNE SCASB
  - JCXZ Not\_Found
  - DEC DI ; tìm thấy được ở vị trí xác định bởi DI
  - -----
  - Not\_Found :

### ■ **Lệnh LODS :**

- Nạp nội dung chuỗi nguồn vào thanh ghi AL hoặc AX
- Địa chỉ chuỗi nguồn chứa trong cặp thanh ghi DS:SI
- Có 2 dạng : LODSB và LODSW
- LODSB : mỗi lần nạp 1 byte
  - $AL \leftarrow [DS:SI]$
  - Nếu DF=0 thì :  $SI \leftarrow SI + 1$
  - ngược lại thì :  $SI \leftarrow SI - 1$
- LODSW : mỗi lần nạp 2 byte
  - $AX \leftarrow [DS:SI+1, DS:SI]$
  - Nếu DF=0 thì :  $SI \leftarrow SI + 2$
  - ngược lại thì :  $SI \leftarrow SI - 2$
- Tác động cờ :
 

OF	DF	IF	SF	ZF	AF	PF	CF
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>




## ■ Lệnh STOS :

- Cất nội dung trong AL hoặc AX vào chuỗi đích
- Địa chỉ chuỗi đích chứa trong cặp thanh ghi ES:DI
- Có 2 dạng : STOSB và STOSW
- STOSB :
  - $[ES:DI] \leftarrow AL$
  - Nếu DF=0 thì :  $DI \leftarrow DI + 1$
  - ngược lại thì :  $DI \leftarrow DI - 1$
- STOSW :
  - $[ES:DI+1, ES:DI] \leftarrow AX$
  - Nếu DF=0 thì :  $DI \leftarrow DI + 2$
  - ngược lại thì :  $DI \leftarrow DI - 2$
- Tác động cờ : 

OF	DF	IF	SF	ZF	AF	PF	CF

## NHÓM LỆNH XỬ LÝ CHUỖI


- **Ví dụ :** Thay thế phần tử có trị 0 đầu tiên của chuỗi Str bằng ký tự '\$', chiều dài 200 byte.
  - PUSH DS
  - POP ES
  - CLD
  - MOV CX, 200
  - MOV DI, OFFSET Str
  - MOV AL, 0
  - REPNE SCASB
  - JCXZ Not\_Found
  - DEC DI
  - MOV AL, '\$'
  - STOSB ; Thay thế ký tự '\$' vào vùng nhớ
  - -----
  - Not\_found :



## Nhóm lệnh điều khiển bộ xử lý

---

- Lệnh CLC :
  - Dạng lệnh : CLC
  - Giải thích :  $CF \leftarrow 0$
- Lệnh STC :
  - Dạng lệnh : STC
  - Giải thích :  $CF \leftarrow 1$
- Lệnh CMC :
  - Dạng lệnh : CMC
  - Giải thích :  $CF \leftarrow \text{bù 1 của } CF$



## Nhóm lệnh điều khiển bộ xử lý

---


- Lệnh NOP :
  - Dạng lệnh : NOP
  - Giải thích : không làm gì cả
  - Không làm gì cả. Dùng để tạo ra các khoảng làm trễ ngắn.
- Lệnh CLD :
  - Dạng lệnh : CLD
  - Giải thích :  $DF \leftarrow 0$
  - Xóa cờ định hướng về 0. Xử lý tăng địa chỉ trong các lệnh xử lý chuỗi.



- **Lệnh STD :**
  - **Dạng lệnh :** STD
  - **Giải thích :**  $DF \leftarrow 1$
  - **Lập cờ định hướng lên 1.** Xử lý giảm địa chỉ trong các lệnh xử lý chuỗi.
- **Lệnh CLI :**
  - **Dạng lệnh :** CLI
  - **Giải thích :**  $IF \leftarrow 0$
  - **Xóa cờ ngắt quãng về 0.** Cấm ngắt quãng cứng.

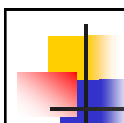


- **Lệnh STI :**
  - **Dạng lệnh :** STI
  - **Giải thích :**  $IF \leftarrow 1$
  - **Lập cờ ngắt quãng lên 1.** Cho phép ngắt quãng cứng
- **Lệnh HLT :**
  - **Dạng lệnh :** HLT
  - **Giải thích :** CPU vào trạng thái dừng.
  - **Dừng CPU, chờ một ngắt quãng cứng xảy ra (INTR hay NMI).**




---

- **Lệnh WAIT :**
  - Dạng lệnh : WAIT
  - Giải thích : CPU vào trạng thái đợi.
  - CPU vào trạng thái đợi cho đến khi ngỏ TEST tác động.
- **Tiếp đầu lệnh LOCK :**
  - Dạng lệnh : LOCK lệnh
  - Giải thích : Khóa các tuyến trong khi thi hành lệnh theo sau.
  - Khóa các tuyến khi thi hành lệnh theo sau. Không cho phép các vi xử lý khác yêu cầu tuyến (chẳng hạn DMA).




---

- **Lệnh ESC :**
- Dạng lệnh : ESC immed,reg  
ESC immed,mem
- Giải thích : đưa lệnh ra tuyến dữ liệu.
- Phát ra một lệnh cho vi mạch đồng xử lý 8087.
- Ví dụ : ESC 6,AL
- ESC 4,[2000h]