

แผนบริหารการสอนประจำบทที่ 15

หัวข้อเนื้อหา

- คำสั่ง MOVS
- คำสั่ง REP
- คำสั่ง STOS
- คำสั่ง LODS
- คำสั่ง REPZ
- คำสั่ง CMPS
- คำสั่ง REPNZ
- คำสั่ง SCAS

วัตถุประสงค์เชิงพฤติกรรม

- มีความรู้และความเข้าใจเกี่ยวกับคำสั่งจัดการสายข้อมูลต่างๆ เช่น คำสั่ง MOVS, คำสั่ง REP, คำสั่ง STOS, คำสั่ง LODS, คำสั่ง REPZ, คำสั่ง CMPS, คำสั่ง REPNZ และคำสั่ง SCAS เป็นต้น

วิธีสอนและกิจกรรมการเรียนการสอน

- บรรยาย
- สืบเสาะหาความรู้
- ค้นคว้าเพิ่มเติม
- ตอบคำถาม

สื่อการเรียนการสอน

- สื่ออิเล็กทรอนิกส์
- ตอบคำถาม
- ภาพ
- เอกสารอ้างอิงประกอบการค้นคว้า

1. การวัดผลและประเมินผล

1.1 ใช้วิธีการสังเกตและจดบันทึกไว้เป็นระยะ

- สังเกตจากงานที่กำหนดให้ไปทำมาส่ง
- สังเกตจากการตอบคำถาม
- สังเกตจากการนำความรู้ไปใช้

2. การประเมินผล

2.1 วิธีตรวจผลงานต่างๆ ที่ให้ทำ

- ตรวจผลงานภาคปฏิบัติ
- ตรวจรายงาน
- ตรวจแบบฝึกหัด

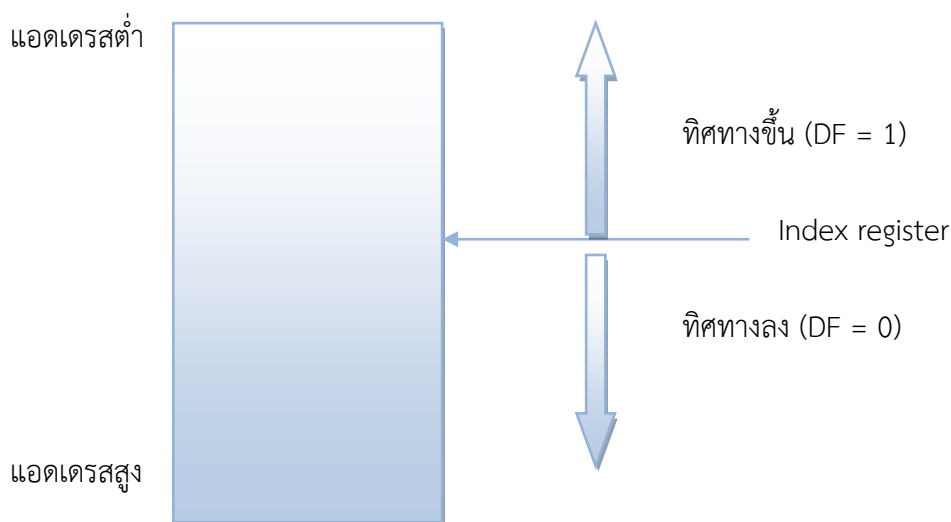
2.2 ใช้วิธีการออกข้อสอบข้อเขียน

บทที่ 15 คำสั่งจัดการกับสายข้อมูล (Character Instruction)

คำสั่งจัดการกับสายข้อมูล เป็นคำสั่งที่ใช้จัดการกับข้อมูลเป็นที่เป็นชุดเรียงต่อกัน คำสั่งในกลุ่มนี้จะระบุตำแหน่งของข้อมูลด้วย Index register คือ SI (Source Index) และ DI (Destination Index) ประกอบกับ DS และ ES โดย ข้อมูลต้นทางจะอยู่ที่ตำแหน่ง DS:SI และข้อมูลปลายทางจะอยู่ที่ ES:DI

การทำงานคำสั่งในกลุ่มนี้จะมีการปรับค่าของรีจิสเตอร์ที่เก็บตำแหน่งของข้อมูลให้โดยอัตโนมัติ เพื่อที่เราจะสามารถใช้คำสั่งนั้นกระทำกับข้อมูลที่ตำแหน่งติดกันถัดไปได้โดยไม่ต้องปรับค่าของรีจิสเตอร์เอง ทิศทางในการปรับจะขึ้นกับค่าที่กำหนดในแฟล็กทิศทาง (DF : Direction flag)

ถ้า DF เป็น 0 แสดงว่ามีการเพิ่มตำแหน่งของการทำงาน โดยจะชี้ไปที่ข้อมูลตัวถัดไป คือ ค่าของรีจิสเตอร์ดัชนีจะเพิ่มขึ้น แต่ถ้า DF เป็น 1 แสดงว่าการทำงานจะเป็นการลดตำแหน่ง โดยจะชี้ไปที่ข้อมูลที่อยู่ก่อนหน้านี้ คือรีจิสเตอร์ดัชนีจะมีค่าลดลง



การกำหนดค่าของ DF สามารถทำได้ด้วยคำสั่ง STD (set direction flag) ซึ่งจะทำให้ ค่าใน DF เป็น 1 และ CLD (clear direction flag) ซึ่งจะทำให้ค่าใน DF เป็น 0
คำสั่งต่างๆ ในกลุ่มนี้

MOVSw	Move string
LODSw	Load string
STOSw	Store string
SCASw	Scan string
CMPSw	Compare string

x แทนได้ถึง B (byte) หรือ W (word) หรือ D (double) ซึ่งหมายถึงขนาดของข้อมูลที่ถูกกระทำด้วยคำสั่งนั้นๆ

คำสั่ง MOVSB

คำสั่ง MOVSB เป็นคำสั่งที่ใช้สำหรับการคัดลอกข้อมูลจากแหล่งข้อมูลต้นทางไปยังแหล่งข้อมูลปลายทาง โดยมีรูปแบบดังนี้

MOVSB MOVSW MOVSD

คำสั่ง MOVSB จะทำการคัดลอกข้อมูลจากตำแหน่ง DS:SI ไปยังตำแหน่ง ES:DI พร้อมทั้งปรับค่าในรีจิสเตอร์ DI และ SI คำสั่ง MOVSB ใช้ในการคัดลอกค่าขนาดไบต์ คำสั่ง MOVSW ใช้คัดลอกค่าขนาดเวิร์ด และคำสั่ง MOVSD ใช้คัดลอกค่าขนาดดับเบิลเวิร์ด

ตัวอย่าง โปรแกรมคัดลอกข้อมูลจำนวน 100 ตัวที่เริ่มต้นที่เลเบล d1 ไปยังเลเบล d2

```
.data
d1      dw    100 dup (N)
d2      dw    100 dup (?)

.code
mov     ax,@data
mov     ds,ax
mov     es,ax
mov     si,offset d1      ; source address
mov     di,offset d2      ; destination address
cld                                ; move in forward direction
mov     cs,100            ; 100 byte to be moved
copy:   movsw
        loop copy
...
```

คำสั่ง REP

คำสั่งการกระทำกับสายข้อมูลนั้นมักจะต้องทำงานซ้ำเป็นวงรอบเพราะในการเรียกคำสั่งแต่ละครั้งนั้นจะเป็นการกระทำกับข้อมูลที่ละ 1 ตัวเท่านั้น ดังเช่นในตัวอย่างข้างต้นเป็นการใช้คำสั่ง MOVSW ร่วมกับคำสั่ง LOOP แต่เราก็อาจนำคำสั่ง REP มาใช้แทนได้

คำสั่ง REP ใช้หน้าคำสั่งอื่นๆ เพื่อให้ทำคำสั่งนั้นซ้ำ โดยในการกระทำคำสั่งแต่ละครั้งนั้นจะมีการลดค่าของรีจิสเตอร์ CX ลงครั้งละ 1 จนกว่าค่าของรีจิสเตอร์ CX จะเท่ากับ 0 ซึ่งก็คล้ายกับการทำงานของคำสั่ง LOOP นั่นเอง

ตัวอย่าง

```
mov cx,100
copy: movsw
loop copy
```

เราสามารถแทนบรรทัดข้างต้นทั้ง 3 ได้ด้วย

```
mov cx,500
rep movsw
```

คำสั่ง STOS

คำสั่ง STOS เป็นคำสั่งที่ใช้สำหรับการคัดลอกข้อมูลขนาดไบต์จากรีจิสเตอร์ AL หรือ ขนาดเวิร์ดจากรีจิสเตอร์ AX หรือ ขนาดดับเบิลเวิร์ดจากรีจิสเตอร์ EAX ไปยังหน่วยความจำตำแหน่ง ES:DI พร้อมทั้งปรับค่ารีจิสเตอร์ DI โดยมีรูปแบบดังนี้

```
STOSB
STOSW
STOSD
```

ตัวอย่าง โปรแกรมกำหนดค่าเริ่มต้นเท่ากับ 0 ให้กับข้อมูล 100 ตัวที่เริ่มต้นที่หน่วยความจำตำแหน่ง d1

```
.data
d1 dw 100 dup (?)

.code
mov ax,@data
mov ds,ax
mov es,ax
mov ax,0
```

```

mov  di,offset d1      ; destination address
cld                                ; fill them forwards
mov  cx,100            ; number of locations to be filled
rep  stosw
...

```

คำสั่ง LODS

คำสั่ง LODS เป็นคำสั่งที่ใช้สำหรับการคัดลอกข้อมูลขนาดไบต์จากหน่วยความจำตำแหน่ง DS:SI ไปยังรีจิสเตอร์ AL หรือ ขนาดเวิร์ดไปยังรีจิสเตอร์ AX หรือ ขนาดดับเบิลเวิร์ดไปยังรีจิสเตอร์ EAX ไปยัง พร้อมทั้งปรับค่ารีจิสเตอร์ SI โดยมีรูปแบบดังนี้

```

LODSB
LODSW
LODSD

```

ตัวอย่าง โปรแกรมหาค่าผลรวมของข้อมูลแบบไบต์ 100 ตัวที่เริ่มต้นที่ d1 และเก็บผลรวมที่ได้ไว้ใน DX

```

.data
d1      db  100 dup (?)

.code
mov  ax,@data
mov  ds,ax
...
mov  dx,0
mov  si,offset d1      ; destination address
cld                                ; forward direction
mov  cx,100            ; number of bytes to be moved
calsum: lodsb          ;move the next byte into AL
add  dl,al             ;add AL to DX
adc  dh,0
loop calsum           ;if more to do, go back and repeat
...

```

โดยปกติเราจะไม่พบการใช้คำสั่ง REP ร่วมกับคำสั่ง LODS

ตัวอย่าง

```
.data
d1      dw      100 dup (?)

.code
mov     ax,@data
mov     ds,ax
...
mov     di,offset d1
cld
mov     cx,100
rep     lodsw
...
```

ข้างต้นเป็นตัวอย่างการใช้คำสั่ง LODS อ่านค่าจากหน่วยความจำตำแหน่ง d1 ทีละเวิร์ดไปเก็บยังรีจิสเตอร์ AX ซึ่งจะเห็นได้ว่าค่าในรีจิสเตอร์ AX จะถูกเขียนทับไปเรื่อยๆ โดยในที่สุดก็มีเพียงค่าสุดท้ายเท่านั้น ไม่มีประโยชน์อันใด

ตัวอย่าง โปรแกรมคำนวณค่าจำนวน 100 ค่าเพื่อใส่ลงในหน่วยความจำที่เริ่มต้นที่ตำแหน่ง d2 โดยมีค่าเท่ากับค่าในหน่วยความจำที่เริ่มต้นที่ตำแหน่ง d1 ที่มีลำดับเท่ากับคุณด้วย 2

```
.data
d1      db      100 dup (?)
d2      db      100 dup (?)

.code
mov     ax,@data
mov     ds,ax
...
mov     ax,ds
mov     es,ax
mov     si,offset d1      ;source address
mov     di,offset d2      ;destination address
```

cld		;forward direction
mov	cx,100	;number of bytes to be moved
calsum:	lods b	;move next byte to AL
shl	al,1	;multiply AL by 2
stosb		;store it in the new location
loop	calsum	;if more to do ,go back and repeat
...		

คำสั่ง REPZ

คำสั่ง REPZ มีการทำงานคล้ายกับคำสั่ง REP แต่คำสั่ง REPZ จะกระทำซ้ำเมื่อ zero flag มีค่าเป็น 1 และ CX มีค่าไม่เท่ากับ 0

คำสั่ง CMPS

คำสั่ง CMPS จะนำค่าในหน่วยความจำตำแหน่ง ES:DI มาลบออกจากค่าในหน่วยความจำตำแหน่ง DS:SI แล้วปรับค่าแฟล็กต่างๆ ที่เกี่ยวข้องโดยค่าของข้อมูลในหน่วยความจำยังคงเดิม และปรับค่าของรีจิสเตอร์ DI และ SI โดยอัตโนมัติ คำสั่ง CMPS มีรูปแบบดังนี้

CMPSB
CMPSW
CMPSD

คำสั่ง CMPS มักจะใช้ในการเปรียบเทียบ 2 สตริงว่าเหมือนกันหรือไม่ ในการจะค้นหาว่าข้อมูลของสตริง 2 ตัวมีค่าเท่ากันหรือไม่นั้น เราจะต้องทำการเปรียบเทียบข้อมูลไปที่ละตัวจนกว่าจะหมดข้อมูลหรือพบข้อมูลที่ไม่เท่ากันเป็นครั้งแรก จากที่ผ่านมาเราใช้คำสั่ง REP ร่วมกับคำสั่งอื่นเพื่อให้คำสั่งนั้นซ้ำเท่าจำนวนที่เรากำหนดในรีจิสเตอร์ CX) แต่ถ้าเราต้องการให้หยุดกระทำคำสั่งเมื่อพบจุดที่แตกต่าง นั่นก็คือเราจะใช้คำสั่ง REPZ แทนคำสั่ง REP เพราะคำสั่ง REPZ จะหยุดกระทำเมื่อ Z-flag เป็น 0 นั่นคือเมื่อคำสั่ง CMPS ทำการเปรียบเทียบพบข้อมูลที่ต่างกันนั่นเอง

ตัวอย่าง โปรแกรมเปรียบเทียบข้อมูลในหน่วยความจำที่ตำแหน่งเริ่มต้นที่ d1 และในหน่วยความจำตำแหน่งเริ่มต้นที่ d2

.data		
d1	db	100 dup (?)
d2	db	100 dup (?)

```

.code
mov ax,@data
mov ds,ax
...
mov bx,ds
mov es,bx
mov si, offset d1 ;start address of first string
mov di, offset d2 ;start address of second string
cld ;forward direction
mov cx,100 ;the number of words to be compare
repz cmpsw
jnz differ
same: ...
...
jmp done
differ: ...
...
done: ...
...

```

คำสั่ง REPZ

คำสั่ง REPZ มีการทำงานคล้ายกับคำสั่ง REP แต่คำสั่ง REPZ จะกระทำซ้ำเมื่อ zero flag มีค่าเป็น 0 และ CX มีค่าไม่เท่ากับ 0

คำสั่ง SCAS

คำสั่ง SCAS จะค้นหาข้อมูลสำหรับค่าขนาดไบต์ที่กำหนดใน AL หรือขนาดเวิร์ดใน AX หรือขนาดดับเบิลเวิร์ดใน EAX โดยเริ่มต้นที่หน่วยความจำตำแหน่ง ES:DI จากนั้นจะมีการปรับค่าของรีจิสเตอร์ DI โดยอัตโนมัติ โดยมีรูปแบบดังนี้

```

SCASB
SCASW
SCASD

```

คำสั่ง SCAS มักจะใช้ในการค้นหาข้อมูลภายในสตริง ในการจะค้นหานั้น เราจะต้องทำการเปรียบเทียบข้อมูลไปทีละตัวจนกว่าจะหมดข้อมูลหรือพบข้อมูลที่ต้องการ ในทำนองเดียวกับคำสั่ง CMPS เราไม่สามารถใช้คำสั่ง REP ได้เพราะเราต้องการให้หยุดกระทำคำสั่งเมื่อพบข้อมูลที่ต้องการ เราจึงใช้คำสั่ง REPZ แทนคำสั่ง

REP เพราะคำสั่ง REPZ จะหยุดกระทำเมื่อ Z-flag เป็น 1 นั่นคือเมื่อคำสั่ง SCAS ทำการเปรียบเทียบพบข้อมูลที่เหมือนกันนั่นเอง

ตัวอย่าง โปรแกรมค้นหาอักขระ 'X' จากข้อมูล 100 ตัว

```
.data
d1      db      100 dup (?)
.code
mov     ax,@data
mov     ds,ax
...
mov     bx,ds
mov     es,bx
mov     di,offset d1      ;first location to be searched
cld                      ;forward direction
mov     cx,100           ;the number of locations to be searched
        mov     al, 'X'
repnz   scasb            ;do the search
jz      found
notfnd: ...
        jmp     done
found:  ...
        ...
done:   ...
```

ตัวอย่าง โปรแกรมสำหรับนำข้อมูลที่เริ่มต้นที่ d2 จำนวน 100 ตัวไปต่อท้ายข้อมูลที่เริ่มต้นที่ d1 โดยถือว่าข้อมูลที่เริ่มต้นที่ d1 ถูกปิดท้ายด้วย null string

```
...
mov     ax,@data
mov     ds,ax
mov     bx,ds
mov     es,bx
mov     al, 0
mov     di, offset d1      ;first location to be searched
cld                      ;forward direction
repnz   scasb            ;do the search
```

mov	si, offset d2	;start address of second string
mov	cx,100	;number of bytes to be moved
rep	movsb	;do the move
...		

สรุป

คำสั่งจัดการกับสายข้อมูล เป็นคำสั่งที่ใช้จัดการกับข้อมูลที่เป็นชุดเรียงต่อกัน คำสั่งในกลุ่มนี้จะระบุตำแหน่งของข้อมูลด้วย Index register คือ SI (Source Index) และ DI (Destination Index) ประกอบกับ DS และ ES โดย ข้อมูลต้นทางจะอยู่ที่ตำแหน่ง DS:SI และข้อมูลปลายทางจะอยู่ที่ ES:DI การทำงานคำสั่งในกลุ่มนี้ จะมีการปรับค่าของรีจิสเตอร์ที่เก็บตำแหน่งของข้อมูลให้โดยอัตโนมัติ เพื่อว่าเราจะสามารถใช้คำสั่งนั้นกระทำกับข้อมูลตำแหน่งติดกันถัดไปได้โดยไม่ต้องปรับค่าของรีจิสเตอร์เอง ทิศทางในการปรับจะขึ้นกับค่าที่กำหนดในแฟล็กทิศทาง (DF : Direction flag)

คำถามทบทวน

- จงอธิบายการทำงานของรีจิสเตอร์ต่อไปนี้ SI DI DS ES และ DF
- จงแสดงและจงอธิบายการทำงานของคำสั่งต่อไปนี้
 - คำสั่ง MOVS
 - คำสั่ง REP
 - คำสั่ง STOS
 - คำสั่ง LODS
 - คำสั่ง REPZ
 - คำสั่ง CMPS
 - คำสั่ง REPNZ
 - คำสั่ง SCAS
- จงเขียนโปรแกรมสำหรับนำข้อมูลที่เริ่มต้นที่ data2 จำนวน 25 ตัวไปต่อท้ายข้อมูลที่เริ่มต้นที่ data1 โดยถือว่าข้อมูลที่เริ่มต้นที่ data1 ถูกปิดท้ายด้วย null string