คำสั่งจัดการกับสายข้อมูล

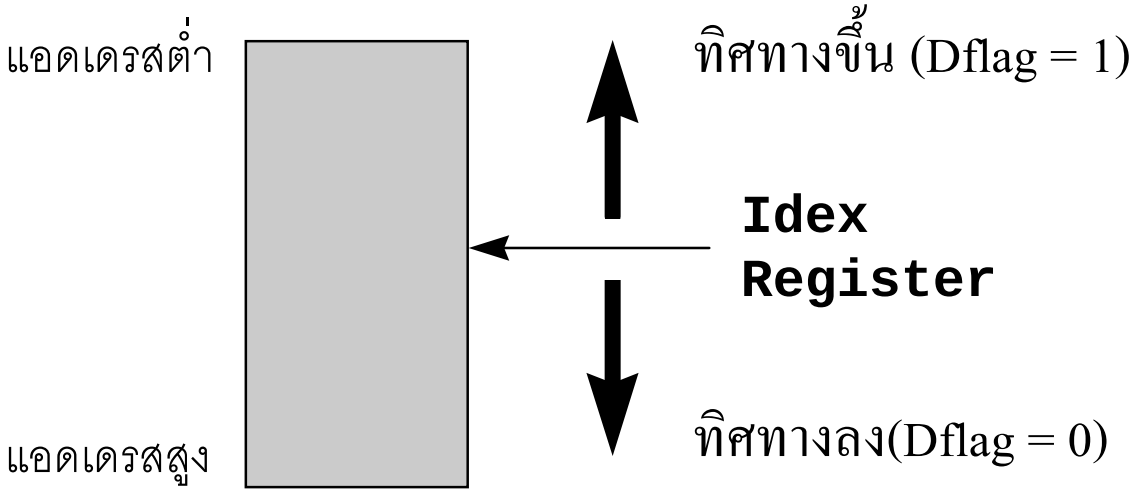
- คำสั่งในกลุ่มนี้จะจัดการกับข้อมูลเป็นชุด.
- ตำแหน่งของข้อมูลระบุโดยใช้ Index register (SI : Source Index, DI : Destination Index) ประกอบกับ DS และ ES ตามลำดับ.
  - ข้อมูลต้นทางจะอยู่ที่ DS:SI
  - ข้อมูลปลายทางจะอยู่ที่ ES:DI
- เมื่อคำสั่งนี้ทำงานเรียบร้อยแล้วจะมีการ**ปรับค่าของรีจิสเตอร์ดัชนีให้โดยอัตโนมัติ**.
- คำสั่งต่าง ๆ ในกลุ่มนี้

```
MOVSx - Move String
LODSx - Load String
STOSx - Store String
SCASx - Scan String
CMPSx - Compare String
```

```
x = B : Byte
  = W : Word
```

ทิศทางในการปรับค่ารีจิสเตอร์ดัชนี

- ลักษณะพิเศษของคำสั่งกลุ่มนี้คือจะมีการปรับค่ารีจิสเตอร์ที่เก็บตำแหน่งของข้อมูลโดยอัตโนมัติ. ทิศทางในการปรับค่า (เพิ่มตำแหน่ง, ลดตำแหน่ง) จะขึ้นกับค่าในแฟล็กทิศทาง (Direction flag)



  - Dflag =1 จะปรับค่าของรีจิสเตอร์ดัชนีลดลง
  - Dflag =0 จะปรับค่าของรีจิสเตอร์ดัชนีเพิ่มขึ้น

- คำสั่งที่ใช้กำหนดค่าแฟล็กทิศทาง

STD **CLD**

คำสั่ง MOVS

- คำสั่ง MOVS จะคัดลอกข้อมูลจากแอดเดรส DS:SI ไปยัง ES:DI พร้อมทั้งปรับค่ารีจิสเตอร์ SI และ DI.

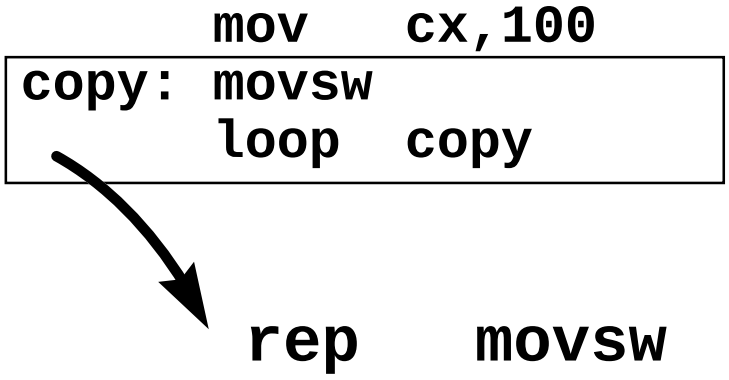
EX เขียนโปรแกรมคัดลอกข้อมูลจำนวน 100 ตัวที่เริ่มต้นที่เลเบล d1 ไปยังเลเบล d2

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

.code
        mov     ax,@data
        mov     ds,ax
        ...
        mov     si,offset d1    ;ตั้งค่าแอดเดรสเริ่มต้น
        mov     ax,@data
        mov     es,ax
        mov     di,offset d2
        cld                     ;ตั้งทิศทาง
        mov     cx,100
copy:   movsw
        loop    copy
        ...
```

คำสั่ง MOVS และ REP

- เราสามารถใช้คำสั่ง REP (repeat) ประกอบกับคำสั่ง MOVS ได้



คำสั่ง REP ที่ใช้เป็นคำสั่งนำหน้าคำสั่งประมวลผลชุดข้อมูล จะมีลักษณะการทำงานเหมือนกับคำสั่ง LOOP นั่นคือจะลดค่าของรีจิสเตอร์ CX และเปรียบเทียบกับ 0. และจะกระทำคำสั่งนั้นซ้ำจนกว่าค่าในรีจิสเตอร์ CX ที่ลดมาจะเท่ากับ 0.

คำสั่ง STOS

- คำสั่งนี้จะคัดลอกข้อมูลจากรีจิสเตอร์ AX หรือ AL ไปยังหน่วยความจำที่ชี้โดย ES:DI พร้อมทั้งปรับค่ารีจิสเตอร์ DI.

EX เขียนโปรแกรมกำหนดค่าเริ่มต้นให้กับข้อมูล 100 ตัว ที่จองไว้เริ่มที่ d1. กำหนดค่าให้เป็น 0.

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

.code
        mov     ax,@data
        mov     ds,ax
        ...
        mov     es,ax                   ;ตั้งค่าแอดเดรสเริ่มต้น
        mov     di,offset d1
        cld                             ;ตั้งทิศทาง
        mov     cx,100
        mov     ax,0
rep     stosw
        ...
```

คำสั่ง LODS

- คำสั่งนี้จะคัดลอกข้อมูลจากหน่วยความจำที่ชี้โดย DS:SI ไปยังรีจิสเตอร์ AX หรือ AL พร้อมทั้งปรับค่ารีจิสเตอร์ SI.

EX เขียนโปรแกรมหาค่าผลรวมของข้อมูลแบบไบต์ 100 ตัวที่จองไว้เริ่มต้นที่ d1. เก็บผลรวมที่ได้ไว้ใน DX

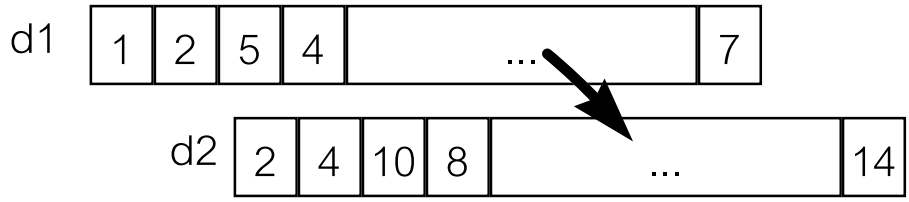
```
.data
d1      db      100 dup (?)

.code
        mov     ax,@data
        mov     ds,ax
        ...
        mov     si,offset d1    ;ตั้งค่าแอดเดรสเริ่มต้น
        cld
        mov     cx,100          ;ตั้งทิศทาง
        mov     dx,0
calsum:
        lodsb
        add     dl,al
        adc     dh,0
        loop    calsum
        ...
```

โดยปกติเราจะไม่พบการใช้คำสั่ง REP กับประกอบกับคำสั่ง LODS

ตัวอย่าง

EX เขียนโปรแกรมคำนวนค่า 100 จำนวนใน d2 โดยมีค่าเท่ากับค่าใน d1 ที่มีลำดับเท่ากันคูณด้วย 2.




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

.code
        mov     ax,@data
        mov     ds,ax
        ...
        mov     ax,ds               ;ตั้งค่าแอดเดรสเริ่มต้น
        mov     es,ax
        mov     si,offset d1
        mov     di,offset d2
        cld                         ;ตั้งทิศทาง
        mov     cx,100
calsum:
        lodsb
        shl     al,1
        stosb
        loop    calsum
        ...
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

คำสั่ง REPZ และ REPNZ

- คำสั่ง REPZ และ REPNZ จะเป็นคำสั่งที่ใช้ประกอบกับคำสั่งกลุ่มจัดการกับชุดข้อมูล โดยจะทำงานเหมือนคำสั่ง REP แต่จะมีการตรวจสอบ Zero flag ด้วย.
  - คำสั่ง REPZ จะกระทำซ้ำเมื่อ Zero flag มีค่าเป็น 1 และค่าใน CX มีค่าไม่เท่ากับศูนย์
  - คำสั่ง REPNZ จะกระทำซ้ำเมื่อ Zero flag มีค่าเป็น 0 และค่าใน CX มีค่าไม่เท่ากับศูนย์
- โดยปกติเงื่อนไขในการหยุดการกระทำซ้ำของคำสั่งทั้งสองมีสองเงื่อนไขคือ
  - CX = 0
  - Zero flag มีค่าไม่ตรงกับที่ต้องการ
- เราสามารถใช้คำสั่ง JZ เพื่อทดสอบเงื่อนไขได้.
- คำสั่ง REPZ และ REPNZ นิยมใช้ประกอบกับคำสั่ง SCAS และ CMPS

คำสั่ง SCAS

- คำสั่งนี้จะนำค่าในรีจิสเตอร์ AX หรือ AL เปรียบเทียบกับค่าในหน่วยความจำที่ระบุตำแหน่งโดย ES:DI จากนั้นจะปรับค่าของรีจิสเตอร์ DI โดยอัตโนมัติ.
- โดยปกติคำสั่งนี้จะใช้ควบคู่กับคำสั่ง REPZ และ REPNZ.

EX เขียนโปรแกรมหาข้อมูลในรีจิสเตอร์ AL จากข้อมูล 100 ตัวในหน่วยความจำที่เริ่มที่ d1.

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

.code
        mov     ax,@data
        mov     ds,ax
        ...
        mov     bx,ds           ;ตั้งค่าแอดเดรสเริ่มต้น
        mov     es,bx
        mov     di,offset d1
        cld                     ;ตั้งทิศทาง
        mov     cx,100
repnz   scasb
        jz      found           ;พบข้อมูล
        ...
```

คำสั่ง CMPZ

- คำสั่งนี้จะนำค่าในหน่วยความจำที่ระบุตำแหน่งโดย DS:SI มาเปรียบเทียบกับค่าในหน่วยความจำที่ระบุตำแหน่งโดย ES:DI จากนั้นจะปรับค่าของรีจิสเตอร์ SI และ DI โดยอัตโนมัติ.
- โดยปกติคำสั่งนี้จะใช้ควบคู่กับคำสั่ง REPZ และ REPNZ.

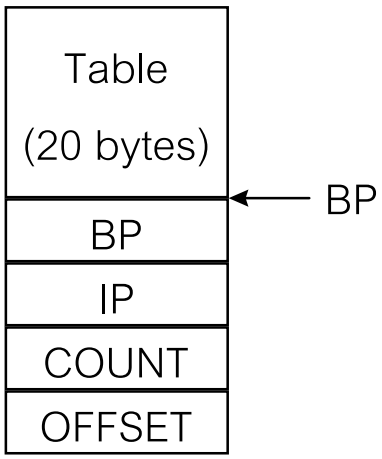
EX เขียนโปรแกรมเปรียบเทียบข้อมูลในหน่วยความจำที่เริ่มที่ d1 กับข้อมูลในหน่วยความจำที่เริ่มต้นที่ d2.

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

.code
        mov     ax,@data
        mov     ds,ax
        ...                             ;ตั้งค่าแอดเดรสเริ่มต้น
        mov     bx,ds
        mov     es,bx
        mov     si,offset d1    ;ตั้งทิศทาง
        mov     di,offset d2
        cld
        mov     cx,100
repz    cmpsw
        jz      same                    ;เหมือนกัน
        ...
```

ตัวอย่างการใช้งาน คำสั่งจัดการกับชุดข้อมูล

- โปรแกรมย่อยที่หาข้อมูลที่มีความถี่สูงสุด (ตัวอย่างในบทที่แล้ว)
- ขั้นตอนการทำงาน
  - จองข้อมูลจำนวน 20 ไบต์ในแสต็ก.
  - กำหนดค่าเริ่มต้นให้กับข้อมูล (STOS)
  - อ่านข้อมูลที่จะนับเข้ามาเขียนค่าในเนื้อที่ที่จองไว้ (LODS)
  - หาค่าที่มีความถี่สูงสุด *(ไม่มีเฉลย)*
- ลักษณะของแสต็กที่จองไว้เป็นดังนี้



ตัวอย่าง

- กำหนดค่าเริ่มต้นให้กับข้อมูลในตารางในแสต็ก

```
        mov   ax,ds
        mov   es,ax
        mov   di,bp
        sub   di,2
        mov   ax,0
        mov   cx,10
        cld
rep     stosw
```

ทำไมต้องลบค่า di ด้วย 2

- อ่านข้อมูลมาปรับค่าในตาราง

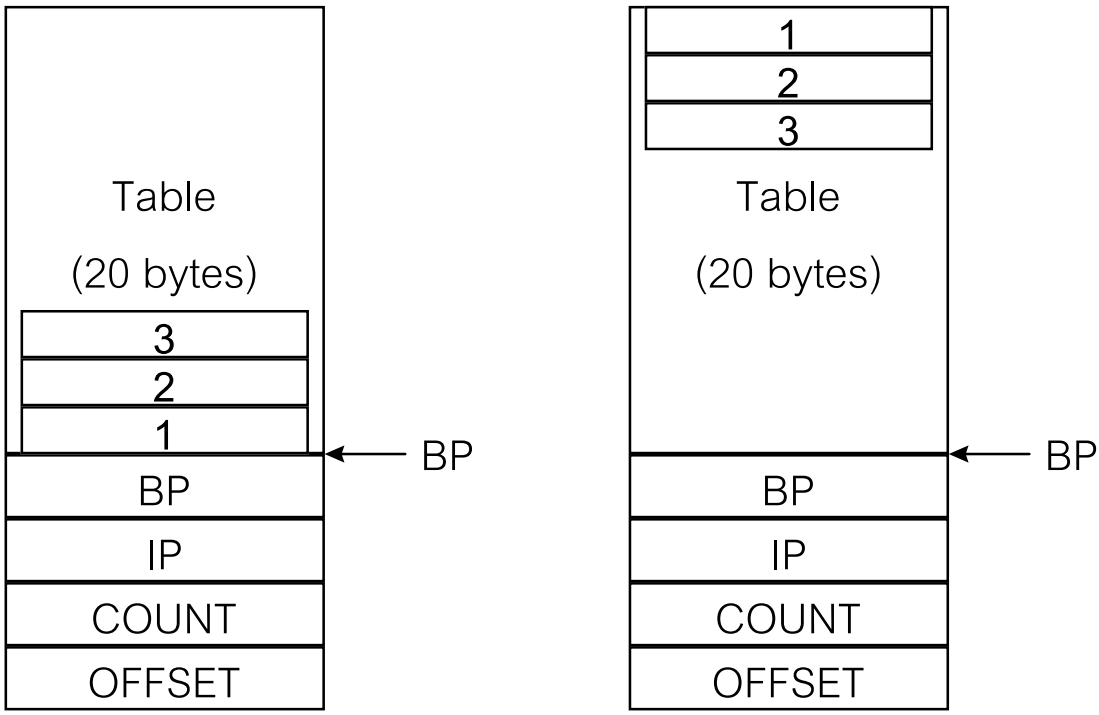
```
;คืนค่าของ CX ที่ถูกใช้เสียก่อน
        mov   si,bx
        cld
readdata:
        lodsb
```

นำค่าของ AL ไปปรับค่าของตารางในแสต็ก

```
        loop  readdata
```

ตัวอย่าง

- **การปรับค่าในตาราง**
- วิธีการปรับขึ้นกับการเรียงลำดับของข้อมูลในตารางในแสต็ก




```
MOV AH,0
MOV DI,AX
NEG DI
INC BYTE PTR[BP+DI]
```

```
MOV AH,0
MOV DI,AX
INC BYTE PTR
    [BP+DI-21]
```

HW เขียนโปรแกรมย่อยนี้ให้สมบูรณ์