แผนบริหารการสอนประจำบทที่ 13

หัวข้อเนื้อหา

- การอ้างแอดเดรส
- แบบการอ้างแอดเดรส

วัตถุประสงค์เชิงพฤติกรรม

- มีความรู้และความเข้าใจเกี่ยวกับการอ้างแอดเดรสใน เซกเมนต์ทั้งสี่ได้แก่ CS, DS, SS และ ES
- สามารถประยุกต์ใช้งานคำสั่งในการอ้างแอดเดรสแบบต่าง ๆ ได้

วิธีสอนและกิจกรรมการเรียนการสอน

- บรรยาย
- สืบเสาะหาความรู้
- ค้นคว้าเพิ่มเติม
- ตอบคำถาม

สื่อการเรียนการสอน

- สื่ออิเล็กทรอนิกส์
- ตอบคำถาม
- ภาพ
- เอกสารอ้างอิงประกอบการค้นคว้า

การวัดผลและประเมินผล

ใช้วิธีการสังเกตและจดบันทึกไว้เป็นระยะ

- สังเกตจากงานที่กำหนดให้ไปทำมาส่ง
- สังเกตจากการตอบคำถาม
- สังเกตจากการนำความรู้ไปใช้

การประเมินผล

วิธีตรวจผลงานต่างๆ ที่ให้ทำ

- ตรวจผลงานภาคปฏิบัติ
- ตรวจรายงาน
- ตรวจแบบฝึกหัด

ใช้วิธีการออกข้อสอบข้อเขียน

บทที่ 13 การอ้างแอดเดรส (Addressing Mode)

13.1 การอ้างแอดเดรส

8086 จะมองลักษณะของหน่วยความจำ โดยแบ่งหน่วยความจำเป็นกลุ่มๆในรูปแบบของ เซกเมนต์ ในหนึ่งเซกเมนต์จะชี้ได้ถึง 64 กิโลไบต์ เซกเมนต์ทั้งสี่ได้แก่ CS, DS, SS และ ES จะแสดงแอดเดรสของหน่วยความจำที่จะติดต่อด้วย CS จะบรรจุค่าแสดงแอดเดรสเริ่มต้นของโปรแกรม DS จะเก็บค่าดาตาเซกเมนต์ขณะนั้น SS ก็จะเก็บค่าสแต็กเซกเมนต์ขณะนั้น และ ES จะกำหนดเซกเมนต์ของข้อมูลรวมที่เรียกว่า global data segment

จุดสำคัญอีกจุดหนึ่งก็คือ เซกเมนต์จะแสดงตำแหน่งเหมือนกับเป็นพารากราฟ (paragraph) โดยจะเลื่อนไปทางซ้าย 4 บิต เพื่อที่จะกำหนดหรืออ้างแอดเดรสให้ครบ 20 เส้น โดยจุดเริ่มต้นของพารากราฟจะต้องมี 4 บิตหลังสุดเป็น 0 เช่น เป็น 00000H, 00010H, 00020H เป็นต้น

เพื่อที่จะทำการติดต่อกับข้อมูลหนึ่งไบต์หรือหนึ่งเวิร์ดนั้น 8086 ได้เตรียมค่า ออฟเซต เพื่อใช้อ้างตำแหน่งตั้งแต่จุดเริ่มต้นของเซกเมนต์แอดเดรส ตำแหน่งใดๆจะได้มาจากการบวกค่าเซกเมนต์รีจิสเตอร์กับค่าของออฟเซต 16 บิต เช่นถ้าเซกเมนต์มีค่า E89F และให้ออฟเซตมีค่า 0003H จะทำให้การอ้างแอดเดรสไปที่ E89F3H

ลักษณะในการอ้างแอดเดรส

- การอ้างแอดเดรสโดยระบุตำแหน่งด้วย เซกเมนต์และออฟเซต เช่น

```
mov   ax, [es:100h]
mov   bl, es:[bx]
```

- การอ้างข้อมูลทั่วๆไปโดยไม่ระบุเซกเมนต์ ออฟเซตที่ระบุจะคิดเทียบกับ DS เช่น

```
mov   ax, B800h
mov   es, ax
```

13.2 แบบการอ้างแอดเดรส

- อ้างแบบรีจิสเตอร์ (Register addressing) ข้อมูลอยู่ในรีจิสเตอร์ เช่น

```
mov   ax, bx
```

- อ้างแบบค่าคงที่ (Immediate addressing) ข้อมูลอยู่ในคำสั่ง เช่น

```
mov   ax, 10
```

- อ้างโดยตรง (Direct addressing) เป็นการอ้างแอดเดรสแบบที่ระบุค่าออฟเซตโดยตรง เช่น

```
mov   ax, [100h]
mov   [200h], cl
mov   cl, total
mov   sum, ax
```

- อ้างทางอ้อมโดยใช้รีจิสเตอร์ (Register indirect addressing) เป็นการอ้างแอดเดรสโดยแอดเดรสของข้อมูลอยู่ในรีจิสเตอร์ เช่น

```
mov    bx, offset buffer
mov    al, [bx]
mov    di, offset total
add    [di],al
```

- อ้างแบบดัชนีโดยตรง (Direct indexed addressing) เป็นการอ้างแอดเดรสโดยแอดเดรสของข้อมูลได้จากการนำค่ารีจิสเตอร์ดัชนี (SI หรือ DI) มาบวกกับเลขคิดเครื่องหมายขนาดแปดบิต หรือเลขไม่คิดเครื่องหมายขนาดสิบหกบิต

```
.data
balance             dw      10 dup(?)
credit      dw      10 dup(?)
debit       dw      10 dup(?)
            ...
            mov     cx, 10
            mov     si, 0
calloop:            mov     ax, balance[si]
            sub     ax, credit[si]
            add     ax, debit[si]
            mov     balance[si], ax
            inc     si
            inc     si
            loop    calloop
```

- อ้างแบบสัมพัทธ์กับฐาน (Base relative addressing) เป็นการอ้างแอดเดรสโดยแอดเดรสของข้อมูลได้จากการนำค่าคงที่ไปบวกกับค่าในรีจิสเตอร์ BX หรือ BP เช่น

```
.data
rec     dw      10 dup(4 dup(?))
        ...
        mov     cx, 10
        mov     bx, offset rec
updateloop:
        mov     ax, [bx]        ; x
        add     ax, [bx+2]      ; +y
        add     ax, [bx+4]      ; +z
        mov     [bx+6], ax      ; c = x+y+z
        add     bx, 8
        loop    updateloop
```

ถ้าคิดสัมพัทธ์กับ BP ออฟเซตที่ได้จะคิดเทียบกับรีจิสเตอร์ SS (Stack Segment)

- อ้างแบบดัชนีกับฐาน (Base indexed addressing) เป็นการอ้างแอดเดรสโดยแอดเดรสของข้อมูลได้จากการนำค่าของรีจิสเตอร์ฐาน (BX หรือ BP) รวมกับค่าของรีจิสเตอร์ดัชนี (SI หรือ DI)

```
mov     ax, [bx+si+2]
mov     [bx+di], al
inc     byte ptr [bx+si]
mov     dx, [bp+si+2]
mov     [bp+di+2], dx
```

ตัวอย่างโปรแกรมแสดงตัวอักษรแบบใหญ่

การแสดงตัวอักษรใหญ่ๆนั้น ต้องมีการเก็บตัวอักษรที่จะพิมพ์ไว้ในหน่วยความจำ โดยการควรเก็บเป็นตารางจะง่ายต่อความเข้าใจ เช่น ตารางขนาด 8*8

```
.data
fontbuf db      0, 0, 0, 1, 0, 0, 0, 0          ;A
        db      0, 0, 1, 0, 1, 0, 0, 0
        db      0, 1, 0, 0, 0, 1, 0, 0
        db      1, 0, 0, 0, 0, 0, 1, 0
        db      1, 1, 1, 1, 1, 1, 1, 0
        db      1, 0, 0, 0, 0, 0, 1, 0
        db      1, 0, 0, 0, 0, 0, 1, 0
        db      0, 0, 0, 0, 0, 0, 0, 0
โดยต้องทำแบบนี้ทุกตัวอักษร( A – Z )
```

เราสามารถเปลี่ยนจากการเก็บ 0 และ 1 เป็น ช่องว่าง และ # แทนตามลำดับ

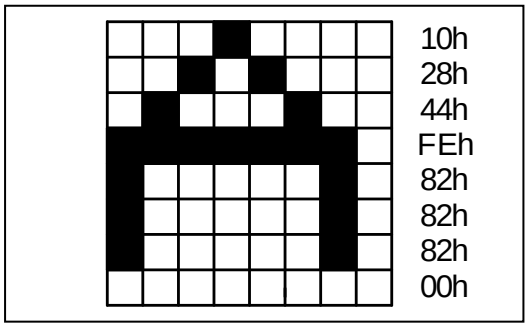
โปรแกรมย่อยสำหรับแสดงตัวอักษร หาตำแหน่งเริ่มต้นของตัวอักษร

```
        mov     bx, offset fontbuf
        mov     dh, 0
        sub     dx, 'A'
        mov     cl, 6
        shl     dx, cl              ; dx = dx*16
        add     bx, dx              ; bx = buf addr
```

โดยตัวอักษรตัวแรกเริ่มที่ 'A' เริ่มที่ offset fontbuf แต่ละตัวตำแหน่งเพิ่มขึ้นทีละ 64 ไบต์ซึ่งมาจากการเก็บตัวอักษรในรูปของตาราง 8*8 ที่แสดงข้างต้น

การพิมพ์ตัวอักษรออกมา

```
        mov     si, 0
        mov     dh, 0
printline:
        mov     cx, 8
printonechar:
        mov     dl, [bx+si]
        call    printchar
        inc     si
        loop    printonechar
        call    printnewline
        inc     dh
        cmp     dh, 8
        jnz     printline
```



ซึ่งจากวิธีนี้ทำให้ใช้ 8 ไบต์เท่านั้นในการเก็บสำหรับหนึ่งตัวอักษร

```
.data
fontbuf  db      10h, 28h, 44h, 0Feh                ;A
         db      82h, 82h, 82h, 00h
โดยที่ต้องสำหรับทุกตัวอักษร ( A – Z )
```

จากการที่ได้เปลี่ยนวิธีการเก็บตัวอักษรคือจากขนาด 64 ไบต์ เป็น 8 ไบต์ จึงต้องมีการแก้ไขส่วนของโปรแกรมแสดงผลใหม่ ดังนี้

```
            mov     si, 0
            mov     di, 0
printline:
            mov     dh, [bx+si]
            mov     cx, 8
printonechar:
            test    dh, 80h
            jz      printzero
            mov     dl, '#'
            jmp     printit
printzero:
            mov     dl, ' '
printit:
            call    printchar
            rol     dh, 1
            loop    printonechar
            call    printnewline
            inc     si
            inc     di
            cmp     di, 8
            jnz     printline
```

สรุป

8086 จะมองลักษณะของหน่วยความจำ โดยแบ่งหน่วยความจำเป็นกลุ่มๆในรูปแบบของ เซกเมนต์ ในหนึ่งเซกเมนต์จะชี้ได้ถึง 64 กิโลไบต์ เซกเมนต์ทั้งสี่ได้แก่ CS, DS, SS และ ES จะแสดงแอดเดรสของหน่วยความจำที่จะติดต่อด้วย CS จะบรรจุค่าแสดงแอดเดรสเริ่มต้นของโปรแกรม DS จะเก็บค่าดาตาเซกเมนต์ขณะนั้น SS ก็จะเก็บค่าสแต็กเซกเมนต์ขณะนั้น และ ES จะกำหนดเซกเมนต์ของข้อมูลรวมที่เรียกว่า global data segment

คำถามทบทวน

1. ลักษณะการอ้างอิงแอดเดรสมีกี่แบบอะไรบ้าง
2. จงเขียนคำสั่งเพื่ออ้างอิงแอดเดรสต่อไปนี้
   2.1 อ้างแบบรีจิสเตอร์ (Register addressing)
   2.2 อ้างแบบค่าคงที่ (Immediate addressing)
   2.3 อ้างโดยตรง (Direct addressing)
   2.4 อ้างทางอ้อมโดยใช้รีจิสเตอร์ (Register indirect addressing)
   2.5 อ้างแบบดัชนีโดยตรง (Direct indexed addressing)
   2.6 อ้างแบบสัมพัทธ์กับฐาน (Base relative addressing)
   2.7 อ้างแบบดัชนีกับฐาน (Base indexed addressing)
3. จงเขียนโปรแกรมแสดงตัวอักษร B และมีการเก็บตัวอักษรที่จะพิมพ์ไว้ในหน่วยความจำโดยเก็บเป็นตาราง ขนาด 8*8