

# แบบการอ้างอิงเอกสาร

---

- ทบทวน : การอ้างอิงเอกสาร
- รูปแบบการอ้างอิงเอกสารแบบต่าง ๆ

# การอ้างแอดเดรส

---

- การอ้างแอดเดรสใน 8086 นั้นจะระบุตำแหน่งโดยใช้ตัวเลข 16 บิตสองตัว : เซกเมนต์ และ ออฟเซต
- ในการอ้างข้อมูลทั่วไป ออฟเซตที่เราจะระบุจะคิดเทียบกับ DS
  - เราสามารถระบุให้คิดเทียบกับเซกเมนต์รีจิสเตอร์ตัวอื่นได้

```
mov    ax,[es:100h]  
mov    bl,es:[bx]  
mov    ss:[bx+10],cx
```

```
mov    ax,0B800h  
mov    es,ax  
mov    dx,0F41h  
mov    [es:00],dx
```

# แบบการอ้างแอดเดรส

---

- อ้างแบบรีจิสเตอร์ (Register addressing)
- อ้างแบบค่าคงที่ (Immediate addressing)
- อ้างโดยตรง (Direct addressing)

– ระบุตำแหน่งของ offset ลงไปโดยตรง

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

mov    cl,total
mov    sum,ax
```

- อ้างทางอ้อมโดยใช้รีจิสเตอร์ (Register indirect addressing)

– แอดเดรสของข้อมูลจะอยู่ในรีจิสเตอร์ :

➡ BX, BP, SI, หรือ DI

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

การอ้างด้วย  
BP แอดเดรสจะ  
คิดเทียบกับ SS

# แบบการอ้างแอดเดรส

---

- อ้างแบบดัชนีโดยตรง (Direct indexed addressing)

- แอดเดรสของข้อมูลจะได้จากการนำค่าของรีจิสเตอร์ดัชนี (SI หรือ DI) มาบวกกับเลขคี่เครื่องหมายขนาดแปดบิต หรือเลขไม่คี่เครื่องหมายขนาดสิบหกบิต.

**.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
```

# แบบการอ้างแอดเดรส

---

- อ้างแบบสัมพันธ์กับฐาน (Base relative addressing)
  - ตำแหน่งของข้อมูลจะได้จากการนำค่าคงที่ไปบวกกับค่าในรีจิสเตอร์ BX หรือ BP.

**EX**

มี array ของ

```
record
    x,y,z,c :integer;
end;
```

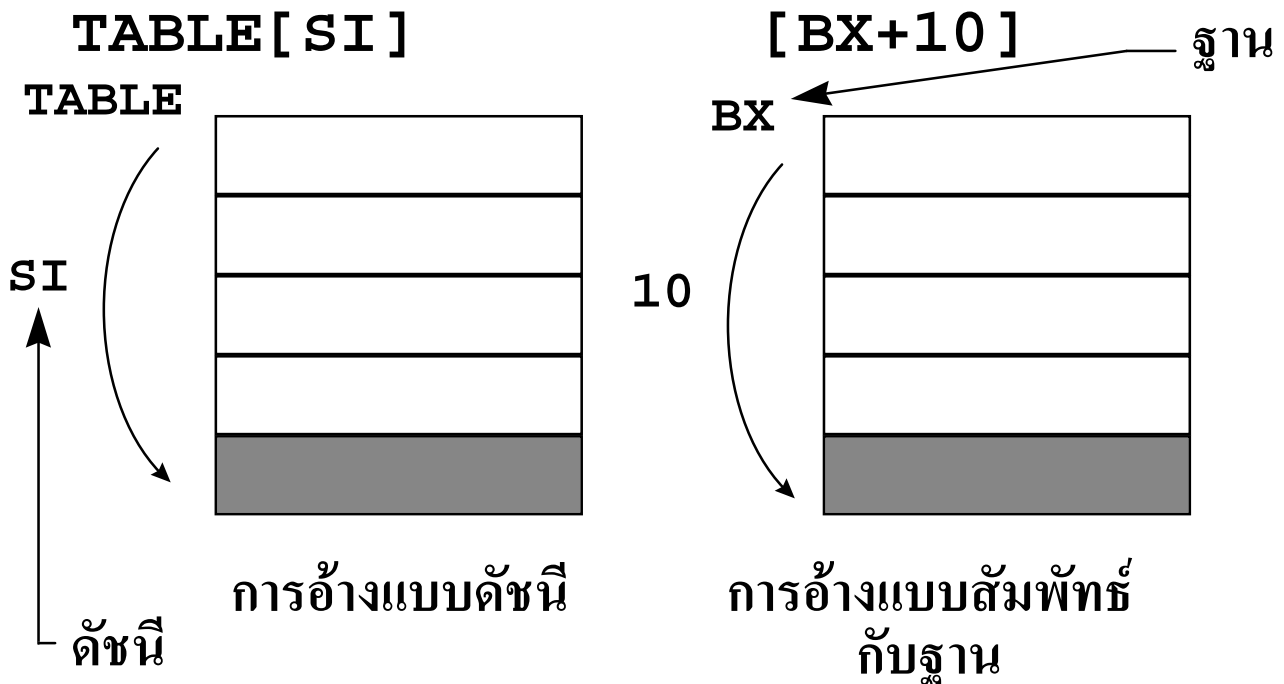
```
.data
rec dw 10 dup (4 dup(?))
```

```
...
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
```

# แบบการอ้างแอดเดรส

- การอ้างแบบดัชนีโดยตรง VS การอ้างแบบสัมผัสกับฐาน

- การอ้างแอดเดรสทั้งสองแบบมีความคล้ายคลึงกันมาก จนสามารถใช้แทนกันได้.



ในหน่วยประมวลผลรุ่นใหม่ ๆ (เช่น 80386) รูปแบบการใช้งานของแบบการอ้างแอดเดรสทั้งสองแบบนี้จะต่างกันมาก.

# แบบการอ้างแอดเดรส

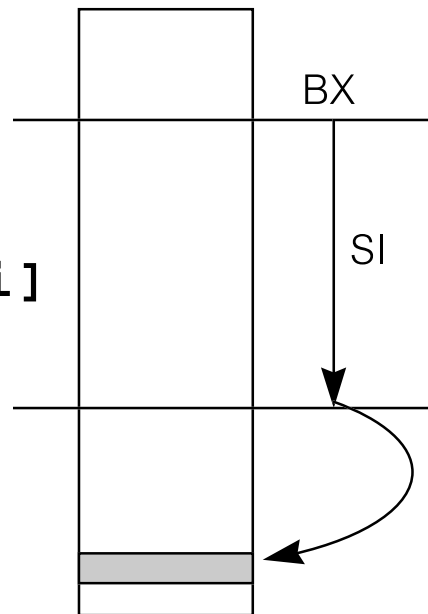
- อ้างแบบดัชนีกับฐาน (Base indexed addressing)

- ตำแหน่งของข้อมูลจะคิดเทียบกับค่าของรีจิสเตอร์ฐาน (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
```



ถ้าคิดสัมพันธ์กับ BP ออฟเซตที่ได้จะคิดเทียบกับ  
รีจิสเตอร์ SS (Stack Segment.)

# ตัวอย่างจริง ๆ

- โปรแกรมแสดงตัวอักษรแบบใหญ่

EX

```
      #      #####      #####  
      # #    #      # #      #  
      #  #   #      # #      #  
      #      # #####      #  
      ##### #      # #      #  
      #      # #      # #      #  
      #      # #####      #####
```

- เขียนโปรแกรมย่อยที่แสดงตัวอักษร A ถึง Z แบบใหญ่

- ต้องมีการเก็บรูปแบบตัวอักษร (font) ที่จะพิมพ์ไว้ในหน่วยความจำ
- ประกาศ font [ใช้แค่ 8x8 แล้วกัน :)]
- จะเก็บ font อย่างไร??



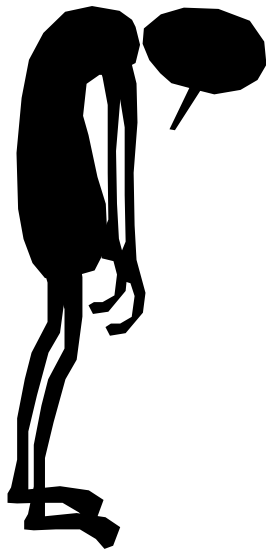
“เก็บเป็นตาราง 8x8 ช่องละไบต์ถ้า  
พิมพ์ให้เก็บค่า 1 ถ้าว่างก็เก็บเป็น 0  
แล้วกัน”



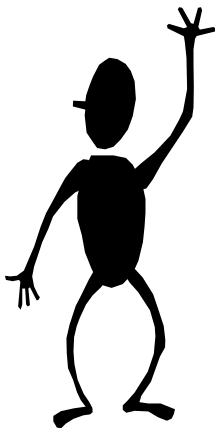
## ตัวอย่างจริง ๆ (ต่อ)

- ตารางประมาณนี้ !!!

.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		
db	1	,	1	,	1	,	1	,	1	,	1	,	0	,	0		; B
db	1	,	0	,	0	,	0	,	0	,	0	,	1	,	0		
db	1	,	0	,	0	,	0	,	0	,	0	,	1	,	0		
db	1	,	1	,	1	,	1	,	1	,	1	,	0	,	0		
db	1	,	0	,	0	,	0	,	0	,	0	,	1	,	0		
db	1	,	0	,	0	,	0	,	0	,	0	,	1	,	0		
db	1	,	1	,	1	,	1	,	1	,	1	,	0	,	0		
db	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0		
db	...																(อีกยาวเลย)



“ เก็บเป็นตาราง 8x8 ช่องละไบต์เก็บเป็น  
ตัวอักษรที่จะพิมพ์เลย ‘#’ กับ ‘ ‘!!!’ ”

## ตัวอย่างจริง ๆ (ต่อ)

- ตารางคือยเป็นผู้เป็นคนจนมาหน่อย :)

[illegible]

- จะพิมพ์ยังไงดีเนี่ยะ ??
  - เขียนโปรแกรมย่อยแสดงตัวอักษร
  - ใส่รหัส ASCII ของตัวอักษรทางรีจิสเตอร์ DL

## ตัวอย่างจริง ๆ (ต่อ)

---

- จะหาตำแหน่งเริ่มต้นของรูปแบบของตัวอักษรที่ต้องการอย่างไร?

- ตัวอักษรตัวแรกเริ่มที่ตัว 'A' เริ่มที่ offset fontbuf
- แต่ละตัวตำแหน่งเพิ่มขึ้น 64 ไบต์









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

- จะพิมพ์ตัวอักษรออกมาได้อย่างไร ??

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

## ตัวอย่างจริง ๆ (ต่อ)

- สังเกตว่าข้อมูล font ของเราสามารถเก็บในรูปแบบอื่นเพื่อให้มีขนาดเล็กลงได้.
  - เก็บเป็นบิต. หนึ่งบรรทัดมี 8 ตัวอักษร -> 1 ไบต์

	10h
	28h
	44h
	FEh
	82h
	82h
	82h
	00h

- ข้อมูลที่ได้จะมีขนาดเล็กลงมาก

```
.data
fontbuf  db  10h, 28h, 44h, 0FEh    ; A
          db  82h, 82h, 82h, 00h

          db  0FCh, 82h, 82h, 0FCh   ; B
          db  82h, 82h, 0FCh, 00h

          db  . . . (แต่อย่างนี้เขียนเองไม่ไหว)
```

## ตัวอย่างจริง ๆ (ต่อ)

- แก้ให้แสดงผลกับข้อมูลแบบใหม่ได้
- หาคำแหน่งเริ่มต้นใหม่ : ตัวอักษรตัวหนึ่งใช้แค่ 8 ไบต์ (คูณแค่ 8)
- แสดงผลแบบใหม่
  - ต้องทดสอบบิต

```
        mov     si,0
        mov     bl,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     bl
        cmp     bl,8
        jnz     printline
```

# ตัวอย่างจริง ๆ ยังไม่จบแค่นี้

- จริง ๆ แล้วไม่ต้องสร้าง font เอง
  - ในหน่วยความจำตำแหน่งที่ 0F000:0FA6Eh จะมีรูปของตัวอักษรต่าง ๆ ตั้งแต่ตัวอักษรที่มีรหัส ASCII=0 ถึงรหัส 255. โดยเก็บในลักษณะเดียวกันกับที่เราใช้ (เก็บเป็นบิต).

-d f000:fa6e

F000:FA60																00	00
F000:FA70	00	00	00	00	00	00	00	7E	81-A5	81	BD	99	81	7E		7E	FF
F000:FA80	DB	FF	C3	E7	FF	7E	6C	FE-FE	FE	7C	38	10	00	10	38		
F000:FA90	7C	FE	7C	38	10	00	38	7C-38	FE	FE	7C	38	7C	10	10		
F000:FAA0	38	7C	FE	7C	38	7C	00	00-18	3C	3C	18	00	00	FF	FF		
F000:FAB0	E7	C3	C3	E7	FF	FF	00	3C-66	42	42	66	3C	00	FF	C3		
F000:FAC0	99	BD	BD	99	C3	FF	0F	07-0F	7D	CC	CC	CC	78	3C	66		
F000:FAD0	66	66	3C	18	7E	18	3F	33-3F	30	30	70	F0	E0	7F	63		
F000:FAE0	7F	63	63	67	E6	C0	99	5A-3C	E7	E7	3C	5A	99				
-																	

EX

- ทำอย่างไรจึงจะพิมพ์ตัวอักษรหลายตัวในบรรทัดเดียวกันได้ (เช่นตอนต้น 'ABC')

