การกระทำระดับบิต

- การกระทำระดับบิต
  - คำสั่งทางตรรกศาสตร์ AND, OR, XOR, NOT
  - คำสั่งเลื่อนบิต
  - คำสั่งหมุนบิต

คำสั่งทางตรรกศาสตร์

- คำสั่งกลุ่มนี้จะจัดการกับข้อมูลในระดับบิตต่อบิต
- **คำสั่ง AND**

```
mov   al,0ADh
and   al,55h
```

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

```
  1010 1101
  0101 0101  and
-----------
  0000 0101
```

- **คำสั่ง OR**

```
mov   al,0ADh
or    al,55h
```

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

```
  1010 1101
  0101 0101  or
-----------
  1111 1101
```

คำสั่งทางตรรกศาสตร์

- **คำสั่ง XOR**

```
mov  al,0ADh
xor  al,55h
```

A	B	A and B
0	0	0
0	1	1
1	0	1
1	1	0

```
  1010 1101
  0101 0101  xor
-------------
  1111 1000
```

- **คำสั่ง NOT**

```
mov  al,0ADh
not  al
```

A	not A
0	1
1	0

```
not 1010 1101
-------------
    0101 0010
```

- **คำสั่ง TEST**
  - เหมือนคำสั่ง AND แต่มีผลกับ flag เท่านั้น
  - นิยมใช้ในการทดสอบบิตต่าง ๆ โดยการ TEST แล้วทดสอบ zero flag.

คำสั่งทางตรรกศาสตร์

```
mov   ax,1234h
and   ax,2345h
```

```
      0001 0010 0011 0100
      0010 0011 0100 0101
-----------------------------
      0000 0010 0000 0100
```

AX = 0204h

```
or    ax,6789h
```

```
      0000 0010 0000 0100
      0110 0111 1000 1001
-----------------------------
      0110 0111 1000 1101
```

AX = 678Dh

```
and   ax,00FFh
```

AX = 008Dh

```
xor   ax,0F0Fh
```

AX = 0F02h

การประยุกต์ใช้คำสั่งทางตรรกศาสตร์

- **การเซ็ตบิต การเคลียร์บิต และการสลับบิต**

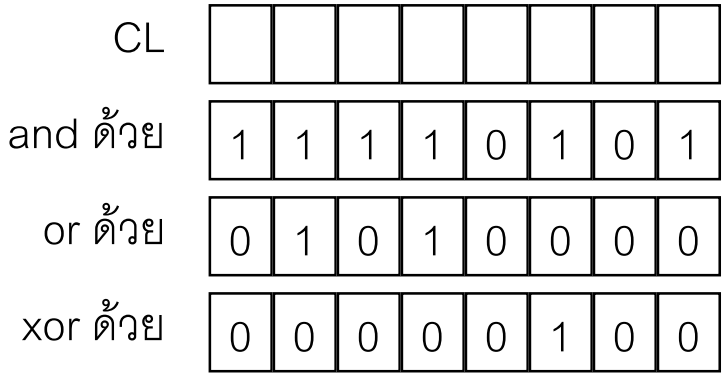
A or 1	1
A or 0	A
A and 1	A
A and 0	0
A xor 1	~A
A xor 0	A

การเซ็ตบิต : or ด้วย 1

การเคลียร์บิต : and ด้วย 0

การสลับบิต : xor ด้วย 1

EX จงเขียนส่วนของโปรแกรมที่ทำให้บิต 1 และ 3 ของ CL มีค่าเป็น 0 บิตที่ 4 และ 6 มีค่าเป็น 1 และ บิตที่ 2 มีค่าสลับกับค่าเดิม




```
and     cl,F5h
or      cl,50h
xor     cl,04h
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

การประยุกต์ใช้คำสั่งทางตรรกศาสตร์

- **การตรวจสอบค่าเท่ากับศูนย์**
  - ใช้การ OR

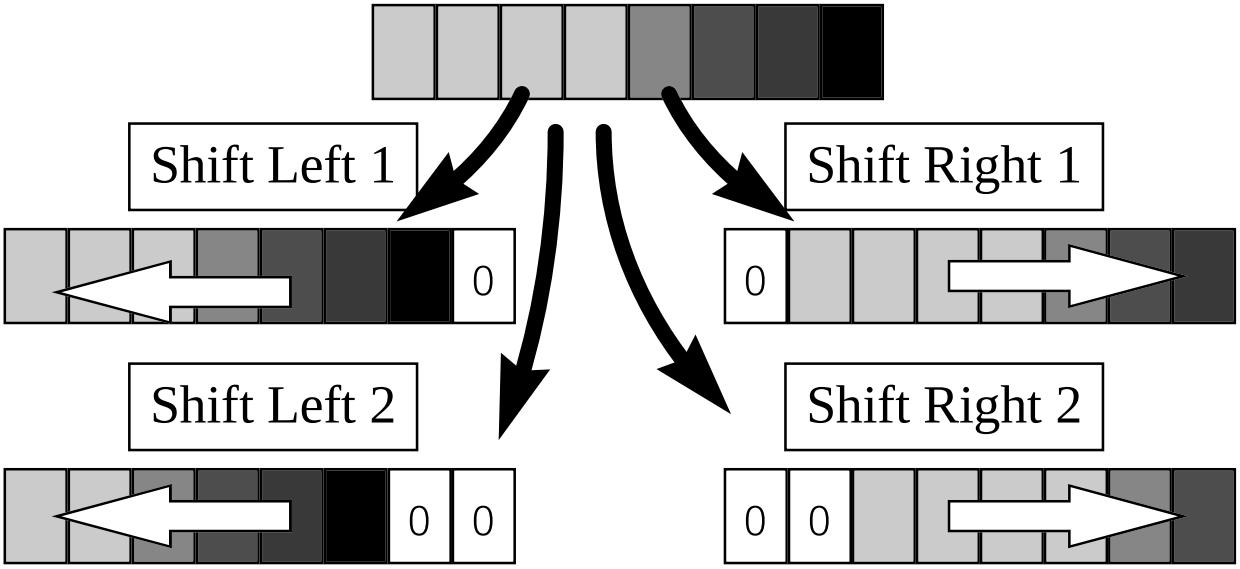
```
OR      AL,AL       ; al=0?
JZ      ZeroLabel
```

- **การกำหนดค่าให้เท่ากับศูนย์**
  - ใช้การ XOR

```
XOR     AL,AL       ; al<-0
```

คำสั่งเลื่อนบิต

- **คำสั่ง SHL (Shift Left) และ SHR (Shift Right)**



- **รูปแบบ**

```
SHL   reg,1      SHL   reg,CL
SHL   mem,1      SHL   mem,CL

SHL   reg,number  *
SHL   mem,number  *
```

* รูปแบบนี้ใช้ได้เฉพาะหน่วยประมวลผล 80286 ขึ้นไปเท่านั้น การใช้ในโปรแกรมต้องระบุ keyword ว่า .286 ในโปรแกรมด้วย

คำสั่งเลื่อนบิต

- **คำสั่ง SHL (Shift Left) และ SHR (Shift Right)**
- **รูปแบบ**

```
SHL   reg,1     SHL   reg,CL
SHL   mem,1     SHL   mem,CL

SHL   reg,number *
SHL   mem,number *
```

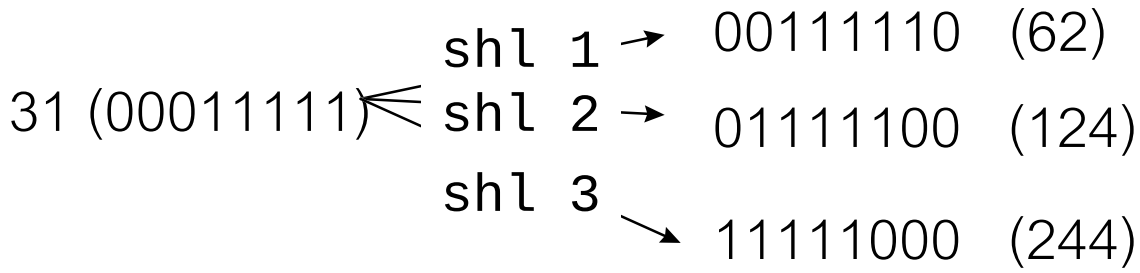
* รูปแบบนี้ใช้ได้เฉพาะหน่วยประมวลผล 80286 ขึ้นไปเท่านั้น

EX

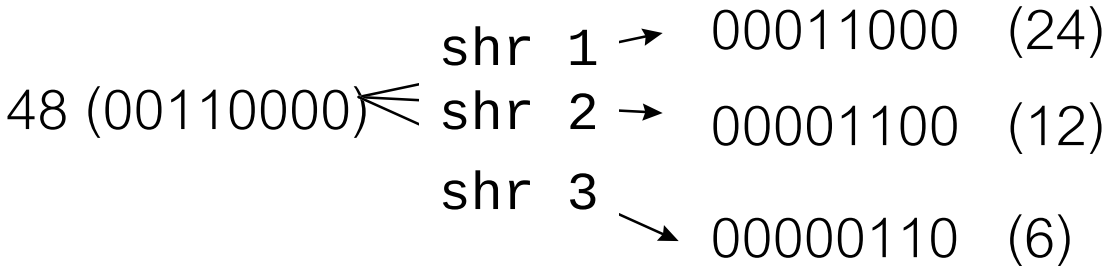
```
        mov   bx,0
        mov   cx,8
checkbit:
        test  al,1
        jz    bitzero
        inc   bx
bitzero:
        shr   al,1
        loop  checkbit
```

คำสั่งเลื่อนบิต

- **ความหมายทางคณิตศาสตร์ของคำสั่งเลื่อนบิต**



Shift left **คือ การคูณด้วยกำลังของสอง**



Shift right **คือ การหารด้วยกำลังของสอง**

คำสั่งเลื่อนบิต

- **คำสั่ง SAL (Shift Arithmetic Left) และ SAR (Shift Arithmetic Right)**
- มีลักษณะและรูปแบบเหมือน SHL และ SHR แต่จะใช้กับเลขคิดเครื่องหมาย
  - SAL กับ SHL เหมือนกันทุกประการ
  - SAR จะใส่บิตที่มีค่า 0 หรือ 1 ทางด้านซ้ายของข้อมูล โดยจะพิจารณาจากเครื่องหมาย

-126 (10000010)	`sar 1`	11000001	(-63)
40 (00101000)	`sar 2`	00001010	(10)

- **การประยุกต์ใช้คำสั่ง Shift ในการคำนวณ**

```
SHL     AL,1
MOV     DL,AL
SHL     AL,1
ADD     DL,AL
```

⇨ DL = AL*6

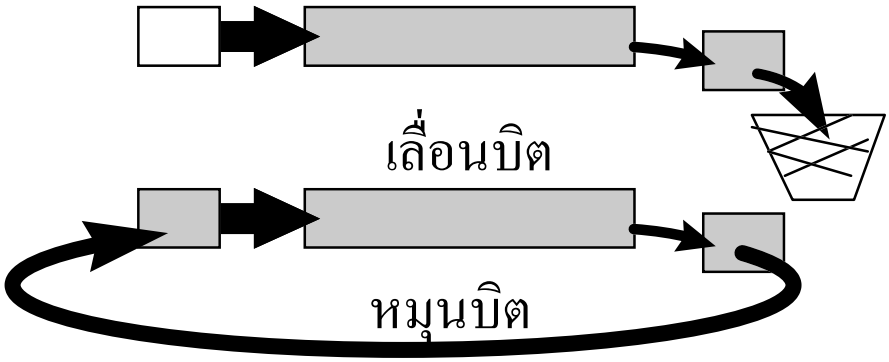
ตัวอย่าง

- **จงเขียนโปรแกรมแสดงค่ารหัสแอสกีของตัวอักษรที่รับจากผู้ใช้ ออกมาเป็นเลขฐานสอง (ให้แยกส่วนแสดงผลออกมาโปรแกรมย่อย)**

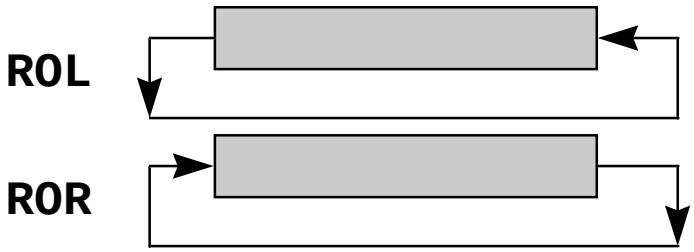
```
; Display Binary (input : al)
dispbin  proc  near
         push  ax
         push  cx
         push  dx
         mov   cx,8
printloop:
         test  al,80h ;test for 1
         jz    printzero
         mov   dl,'1'
         jmp   printit
printzero:
         mov   dl,'0'
printit: mov   ah,2
         int   21h

         shl   al,1
         loop  printloop
         pop   dx
         pop   cx
         pop   ax
         ret
dispbin  endp
```

คำสั่งหมุนบิต

- คำสั่งหมุนบิต และ คำสั่งเลื่อนบิต



- คำสั่งหมุนบิต
  - หมุนทางซ้าย ROL (Rotate Left)
  - หมุนทางขวา ROR (Rotate Right)

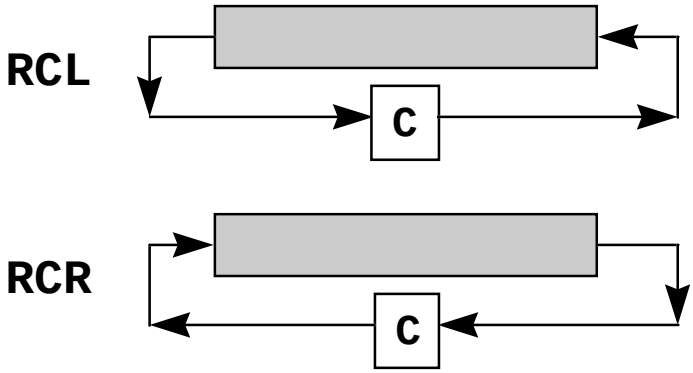


EX

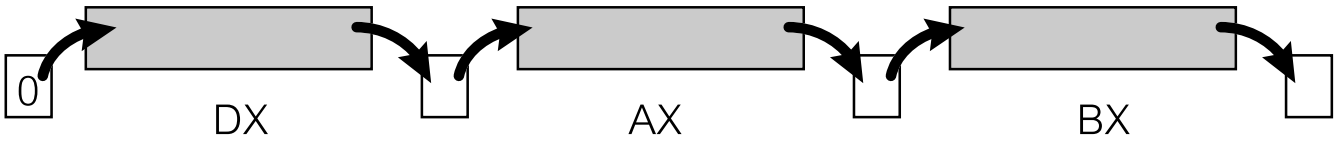
```
        mov   al,4Ah
        mov   cl,4
        ror   al,cl     ;al=0A4h

        mov   cx,8
        mov   dx,0
loophere:
        xor   dx,ax
        rol   ax,1
        loop  loophere
```

คำสั่งหมุนบิตผ่านแฟล็กทด

- **คำสั่ง RCL (Rotate with Carry Left) และ คำสั่ง RCR (Rotate with Carry Right)**



บิตที่เลื่อนออกมาจะถูกนำไปเก็บไว้ในแฟล็กทดก่อนที่จะนำไปแทนในบิตที่เลื่อนออกมาของข้อมูลตัวต่อไป. เรานิยมใช้คำสั่งหมุนบิตผ่านแฟล็กทดในการเลื่อนบิตของข้อมูลที่เก็บต่อเนื่องกันบนรีจิสเตอร์หลายตัว.




```
clc
rcr   dx,1
rcr   ax,1
rcr   bx,1
```

สังเกตว่า ก่อนที่เราจะหมุนบิตเราจะต้องกำหนดค่าให้กับแฟล็กทดเสียก่อน.

ตัวอย่างการประยุกต์ใช้งาน

- การพิมพ์ตัวเลขฐานสิบหก
  - หาค่าของหลักหน้าและหลักหลังของเลขฐานสิบหก
  - หารด้วย 16 : ผลลัพธ์ -> หลักแรก; เศษ -> หลักหลัง
  - ใช้การ shift และ คำสั่งทางตรรกศาสตร์

```
mov  bl,al
and  bl,0Fh    ;bl=lower digit
mov  bh,al
shr  bh,4      ;bh=higher digit
```

- การหาค่า parity ของข้อมูล
  - ถ้าข้อมูลมีจำนวนบิตที่เป็นหนึ่งเป็นเลขคู่ parity=1
  - เขียนโปรแกรมรับการกดปุ่มจากผู้ใช้และแสดงค่า parity ของข้อมูลที่มีค่าเท่ากับรหัส ASCII ของปุ่มที่ผู้ใช้กด.

```
     mov   bl,1     ;parity=1
     mov   cx,8
checkloop:
     mov   bh,al
     and   bh,1
     xor   bl,bh    ;xor with parity
     rol   al,1
     loop  checkloop
```

ตัวอย่างการประยุกต์ใช้งาน

- โปรแกรมคำนวณหาตัวเลขสำหรับตรวจสอบความถูกต้องของข้อมูลอย่างง่าย.

- ใช้ตัวเลขนี้มีค่าเท่ากับค่าของข้อมูลทั้งหมด XOR กัน.
- เขียนโปรแกรมรับข้อความจากผู้ใช้แล้วคำนวณเลขสำหรับตรวจสอบโดยใช้ค่าของรหัสแอสกีของอักษรในข้อความ. ให้แสดงผลลัพธ์เป็นอักขระที่มีค่ารหัสแอสกีเท่ากับค่าตัวเลขนี้