Shift and Rotate Instructions

shift instructions are among the most characteristic of assembly language. *Shifting and rotating* means to move bits right and left inside an operand. 8086 processor provides a particularly rich set of instructions in this area (Table 1), all affecting the Overflow and Carry flags:

Table 1: shift and Rotate instructions.

SHL	Shift left
SHR	Shift right
SAL	Shift arithmetic left
SAR	Shift arithmetic right
ROL	Rotate left
ROR	Rotate right
RCL	Rotate carry left
RCR	Rotate carry right

The basic syntax of these instructions:

```
Inst    op1,op2
```

The types of operands permitted by these instructions as follows (Same for all shift and rotate instructions):

	Operand1 (destination)		*Operand2 (source is count)*
Inst	Memory	,	imm8
	REG	,	imm8
	memory	,	CL
	REG	,	CL

The second operand (source contains the shift or rotate count, which is an imm8 (immediate-8bit) or CL register. The Formats shown here also apply to the SHL, SHR, SAL, SAR, ROR, ROL, RCR, and RCL instructions:

1.The format for single shift or rotate is:

```
Inst   dest ,1
```

2. The format for shift or rotate of N positions is:

```
Inst   dest, CL
```

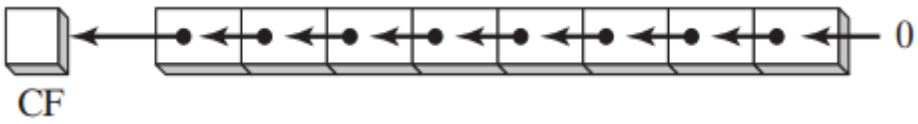
Where CL contain N.

In both cases, the first operand (dest) is an 8- or 16- bit register or memory location.

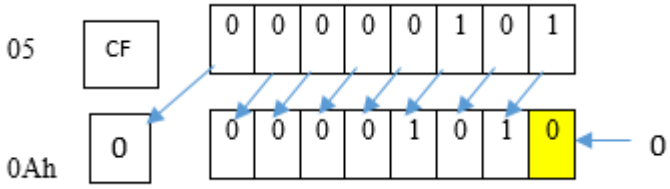
Shifts instructions

(a) .Shifting bits left instructions: (SHL and SAL instructions)

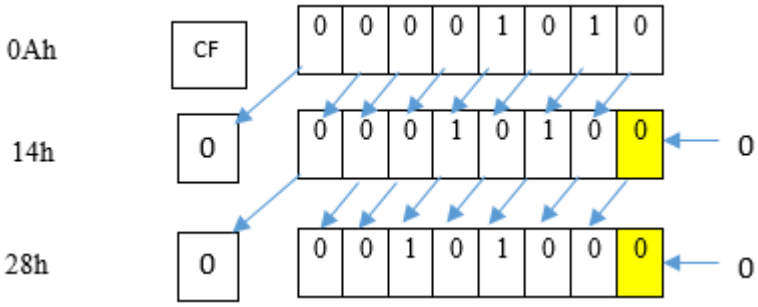
SHL instruction : Shift Left for logical unsigned data performs a logical left shift on the destination operand. For each shift count, SHL shifts each bit in the destination operand to the next highest bit position. The lowest bit is assigned 0. The highest bit is moved to the Carry flag, and the bit that was in the Carry flag is discarded:



Ex1: MOV BL , 05 ; BL = 05 h
SHL BL , 01 ; shift left 1 BL = 0A h , CF =0



Ex2: MOV CL , 02 ; set shift value CL = 02
MOV BH, 0Ah
SHL BH , CL ; shift left 2 BL = 28h h , CF =0



SAL instruction: Shift Arithmetic Left for arithmetic signed data works the same as the SHL instruction.

Both instructions (SHL and SAL) are identical in their operation (SHL=SAL).

Fast Multiplication

Shifting left 1 bit multiplies a number by 2

```
mov dl,5
shl dl,1
```

Before: 00000101 = 5
After: 00001010 = 10

Shifting left *n* bits multiplies the operand by 2^n

For example, $5 * 2^2 = 20$

Ex3: MOV DL, 5
MOV CL,2
SHL DL, CL ; DL= 5*4=20 = 14h

The same as when using: SAL DL, CL

Note:

Shift left 1 bit means multiply by 2 (2^1)

Shift left 2 bit means multiply by 4 (2^2)

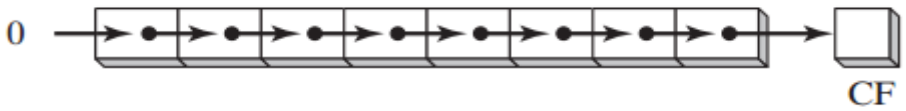
Shift left 3 bit means multiply by 8 (2^3)

Shift left 4 bit means multiply by 16 (2^4)

Shift left 5 bit means multiply by 32 (2^5) and so on.

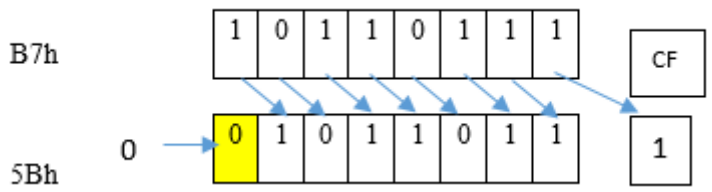
(b) .Shifting bits right instructions:(SHR and SAR instructions)

SHR instruction : Shift Right instruction for logical unsigned data performs a logical right shift on the destination operand, replacing the highest bit with a 0. The lowest bit is copied into the Carry flag, and the bit that was previously in the Carry flag is lost:



In the following example, the 0 from the lowest bit in AL is copied into the Carry flag, and the highest bit in AL is filled with a zero:

Ex1: MOV DH , B7h ; DH = B7 h
SHR DH , 01 ; shift right 1 DH = 5B h , CF =1



Fast Divition

Shifting right 1 bit divide a number by 2

```
mov dl,80h
shr dl,1
```

Before:	10000000	= 80
After:	01000000	= 40

Shifting right n bits divides the operand by 2^n

Ex3: In the following example, we use SHL inst. to divide DH register over 4.

```
MOV   DH , 5B h     ;  DH = 5B h,
MOV   CL , 02       ; set shift value     CL = 02
SHR   DH , CL       ; shift right 2, means divide DH over 4;
                    ;   DH = 16 h , CF =1
```

Note:

Shift right 1 bit means divide by 2 (2^1)

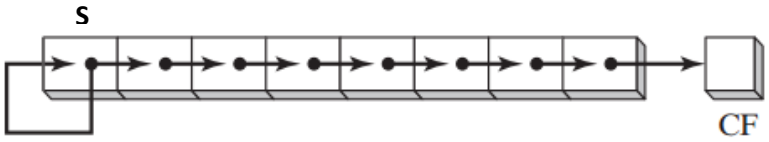
Shift right 2 bit means divide by 4 (2^2)

Shift right 3 bit means divide by 8 (2^3)

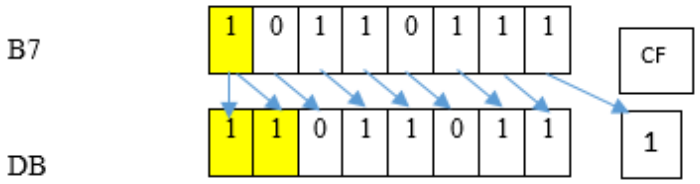
Shift right 4 bit means divide by 16 (2^4)

Shift right 5 bit means divide by 32 (2^5) and so on.

SAR instruction: Shift Arithmetic Right instruction provides for arithmetic signed data. The ***SAR*** differs from SHR in one important way : SAR uses the sign bit to fill leftmost vacated bits. In this way , positive and negative values retain their signs.




```
Ex1:  MOV   DH , B7 h     ;  DH = B7 h
      SAR   DH , 01       ;  shift right 1      DH = DB h   , CF =1
```



Signed Division You can divide a signed operand by a power of 2, using the SAR instruction. In the following example, DBh (–37) is divided by 2^2. The quotient is –10 (F6h):

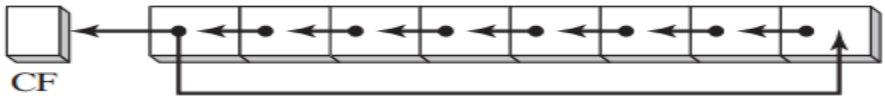
```
EX2:   MOV   DH , DBh        ;  DH = DBh = 1101 1011 b
       MOV   CL , 02         ;  CL = 02
       SAR   DH , CL         ;  shift arithmetic right 2 ;  DH = F6 H , CF =1
```

Note : SHL= SAL but SHR ≠ SAR

Rotating instructions

1.Rotating bits left instructions: (ROL and RCL instructions)

ROL* instruction:** Rotate Left instruction provides for logical unsigned data. The ***ROL operation rotate each bits to the left. The highest bit is copied into the Carry flag and the lowest bit position.



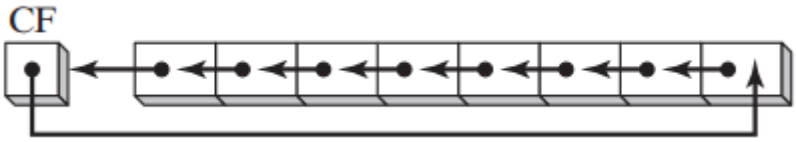
Bit rotation does not lose bits. A bit rotated off one end of a number appears again at the other end. Note in the following example how the high bit is copied into both the Carry flag and bit position 0:

```
Ex1: MOV AL,40H          ; AL = 01000000  b
     ROL AL,1            ; CF = 0 , AL = 10000000  b
     ROL AL,1            ; CF = 1 , AL = 00000001  b
     ROL AL,1            ; CF = 0 , AL =  00000010  b
```

The above example we can write it as follows:

```
MOV AL,40H          ; AL = 01000000  b
MOV  CL,3
ROL AL,CL           ; CF = 0 , AL = 00000010  b
```

RCL* instruction:** Rotate with Carry Left instruction provides for arithmetic signed data. The ***RCL differs from ***ROL*** in this way: Each bit rotated off on the left first moves into the Carry Flag, and the Carry Flag bit moves into LSB, and copies the MSB into the Carry flag:



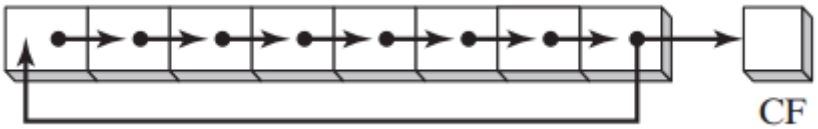

```
CLC                 ; CF = 0
MOV   BL,88h        ; CF = 0  ,  BL =  10001000 b
RCL   BL,1          ; CF = 1  ,  BL =  00010000 b
RCL   BL,1          ; CF = 0  ,  BL =  00100001 b
```

The above example we can write it as follows:

```
CLC                 ; CF = 0
MOV   BL,88h        ; CF= 0  ,  BL =  10001000 b
MOV   CL, 2
RCL   BL,CL          ; CF = 0  ,  BL =  00100001 b
```

2.Rotating bits right : (ROR and RCR instructions)

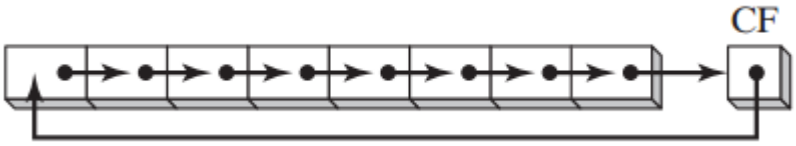
ROR* instruction:** Rotate Logical Right instruction provides for logical unsigned data. The ***ROR operation rotate bits to the right and copies the lowest bit into the Carry flag and the highest bit position. Each bit rotated off enters the carry flag:



In the following examples, note how the lowest bit is copied into both the Carry flag and the highest bit position of the result:

```
MOV     AL,01     ; AL = 00000001  b
ROR     AL,1      ; AL = 10000000  b , CF = 1
ROR     AL,1      ; AL = 01000000  b , CF = 0
```

RCR instruction: Rotate with Carry Right instruction provides for arithmetic signed data. The ***RCR*** differs from ***ROR*** in this way: Each bit rotated off on the right first moves into the CF, and the CF bit moves into the vacated bit position on the left.




```
STC                 ; CF = 1
MOV   DL,10h        ; DL= 00010000    CF = 1
RCR   DL,1          ; DL= 10001000    CF =0
```

Notes: Effects of shift and rotate instructions on Flags:

1. Always CF contains the last bit shifted out. After the shift and rotate operations, you can use the JC (jump if Carry) instruction to test the bit shifted or rotated into the CF.

2. SF and ZF are affected according to result.

3. Effect on OF for all (left and right) shift/rotate:
   - For any single-bit shift/rotate: OF = 1 if the shift/rotate changes the sign-bit.
   - For multiple-bit shift: OF is undefined.
