4- Shift instruction:

Shift instructions of the 8086 can perform two basic types of shift operations; the logical shift and the arithmetic shift. Each of these operations can be performed to the right or to the left. The shift instructions are:

- **SHL** Shift Logical Left
- **SAL** Shift Arithmetic Left
- **SHR** Shift Logical Right
- **SAR** Shift Arithmetic Right

These instructions are used to isolate bits of a byte or word so that it can be tested, and to perform simple multiply and divide computations.

Mnemonic	Meaning	Format	Operation	Flags Affected
SAL/SHL	Shift arithmetic Left/shift Logical left	SAL/SHL D, Count	Shift the (D) left by the number of bit positions equal to count and fill the vacated bits positions on the right with zeros	CF,PF,SF,ZF AF undefined OF undefined if count ≠1
SHR	Shift logical right	SHR D, Count	Shift the (D) right by the number of bit positions equal to count and fill the vacated bits positions on the left with zeros	CF,PF,SF,ZF AF undefined OF undefined if count ≠1
SAR	Shift arithmetic right	SAR D, Count	Shift the (D) right by the number of bit positions equal to count and fill the vacated bits positions on the left with the original most significant bit	CF,PF,SF,ZF AF undefined OF undefined if count ≠1

(a)

Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

(b)

The operation of the shift instructions is described in Fig (a), Note in Fig.(b)

- That the destination operand, the data whose bits are to be shifted, can be either the contents of an internal register or a storage location in memory
- The source operand can be specified in two ways:

  Count Value is 1 → Shift by One bit

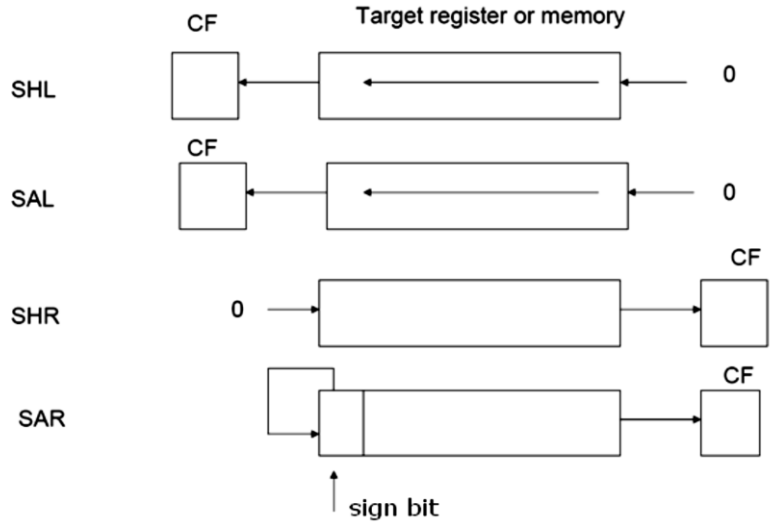
  Count Value is CL register → Shift by the value of CL register

❖ The **SHL** and **SAL** are identical. They shift the operand to left and fill the vacated bits to the right with **zeros**.

❖ The SHR instruction shifts the operand to right and fill the vacated bits to the left with **zeros.**

❖ The SAR instruction shifts the operand to right and fill the vacated bits to the left with the value of **MSB** (this operation used to shift the signed numbers)

As shown in fig (c)



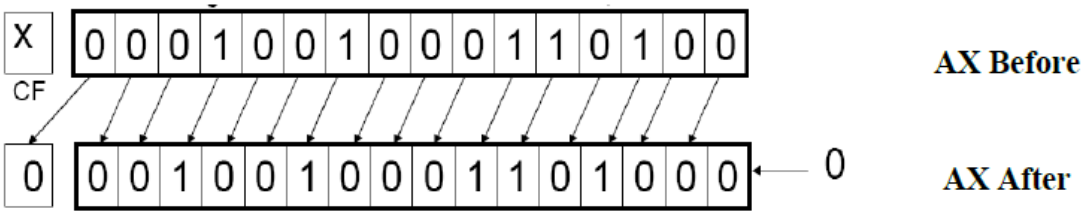
(c)

Ex: let AX=1234H what is the value of AX after execution of next instruction

SHL AX,1

Solution:

causes the 16-bit register to be shifted 1-bit position to the left where the vacated LSB is filled with zero and the bit shifted out of the MSB is saved in CF

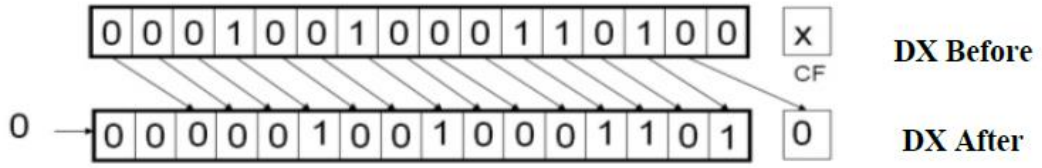


Ex:

MOV CL, 2H

SHR DX, CL

The two MSBs are filled with zeros and the LSB is thrown away while the second LSB is saved in CF.

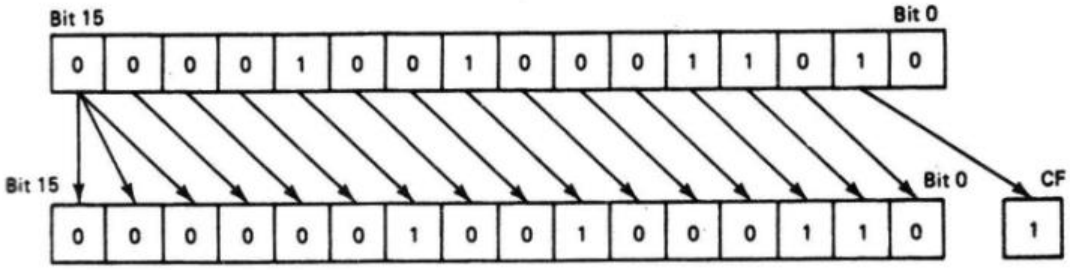


Ex: Assume CL= 2 and AX= 091AH.

Determine the new contents of AX and CF after the instruction

SAR AX, CL

is executed.



Ex: Multiply AX by 10 using shift instructions, assume AX = 0003 H

Solution:

```
SHL AX, 1
MOV BX, AX
MOV CL,2
SHL AX,CL
ADD AX, BX
```

instruction	(AX)H	(BX)H	(CL)H
Initiate	0003	0000	00
SHL AX , 1	0006	0000	00
MOV BX,AX	0006	0006	00
MOV CL,2	0006	0006	02
SHL AX,CL	0018	0006	02
ADD AX,BX	001E	0006	02

AX = 0003 H = 0000 0000 0000 0011 B = 3 Dec.
AX = 0006 H = 0000 0000 0000 0110 B = 6 Dec.
AX = 0018 H = 0000 0000 0001 1000 B = 24 Dec.
AX = 001E H = 0000 0000 0001 1110 B = 30 Dec.

Ex: What is the result of **SAR CL, 1** ,if CL initially contains B6H?
Solution: DBH

CL = **1**011 0110 Before
↓
CL = **11**01 1011 [0] After

Isolating the Value of a Bit in an Operand

If we need to isolate the value of one of the bits of a word or byte of data by shifting it into the carry flag.
The shift instructions may perform this operation on data either in a register or a storage location in memory.
The instructions that follow perform this type of operation on a byte of data stored in memory at address FLAG :

```
MOV AL, [FLAG]
MOV CL, 04H
SHR AL, CL
```

- The first instruction reads the value of the byte of data at address FLAG into AL.
- In the second instruction a shift count of four is loaded into CL,
- In third instruction the value in AL is shifted to the right four bit positions. Since the MSBs of AL are reloaded with zeros as part of the shift operation, the results are

(AL) = 0 0 0 0 B_7 B_6 B_5 B_4

and

(CF) = B_3

In this way, we see that bit B_3 of FLAG has been isolated by placing it inCF. Once this bit is in CF, it can be tested by other instructions and based on this value initiate another software operation.

5- Rotate Instructions

The rotate instructions, are similar to the shift instructions
As shown in Fig.(a), includes the:

- rotate left instruction **ROL**
- rotate right instruction **ROR**
- rotate left through carry instruction **RCL**
- rotate right through carry instruction **RCR**

They perform many of the same programming functions as the shift instructions, such as isolation of a bit of an element of data.

Fig.(b) shows, the rotate instructions are similar to the shift instructions in several ways. They have the ability to rotate the contents of either an internal register or a storage location in memory.

Also, the rotation that takes place can be from 1 to 255 bit positions to the left or to the right. Moreover, in the case of a multi bit rotate, the number of bit positions to be rotated is specified by the value in CL.

Their **difference** from the shift instructions lies in the fact that the bits moved out at either the MSB or LSB end are **not lost**; instead, they are reloaded at the other end.

Mnemonic	Meaning	Format	Operation	Flags Affected
ROL	Rotate Left	ROL D, Count	Rotate the (D) left by the number of bit positions equal to Count. Each bit shifted out from the left most bit goes back into the rightmost bit position.	CF OF undefined if count ≠ 1
ROR	Rotate Right	ROR D, Count	Rotate the (D) right by the number of bit positions equal to Count. Each bit shifted out from the rightmost bit goes back into the leftmost bit position.	CF OF undefined if count ≠ 1
RCL	Rotate Left through Carry	RCL D, Count	Same as ROL except carry is attached to (D) for rotation.	CF OF undefined if count ≠ 1
RCR	Rotate right through Carry	RCR D, Count	Same as ROR except carry is attached to (D) for rotation.	CF OF undefined if count ≠ 1

(a)

Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

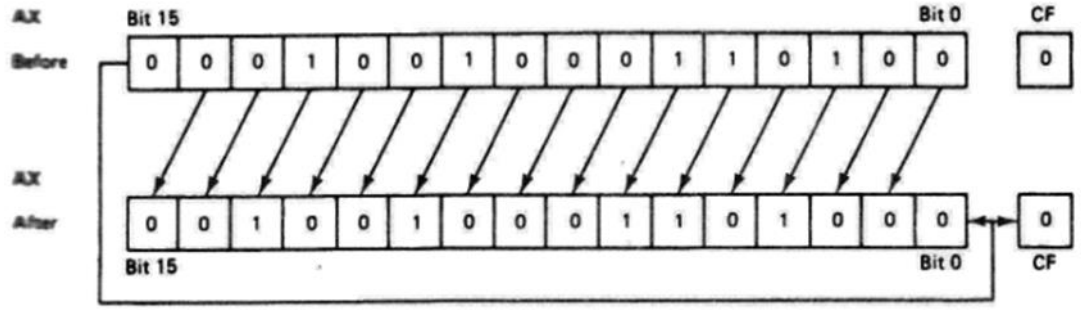
(b)

The Execution of **ROL** instruction causes the contents of the selected operand to be rotated left the specified number of bit positions. Each bit shifted out at the **MSB** end is reloaded at the **LSB** end. More- over, the content of CF reflects the state of the last bit that was shifted out.

Ex:

ROL AX, 1

This instruction causes a 1-bit rotate to the left. Figure (c) shows the result produced by executing this instruction. Note that the original value of bit 15 is zero. This value has been rotated into both CF and bit 0 of AX. All other bits have been rotated one bit position to the left.



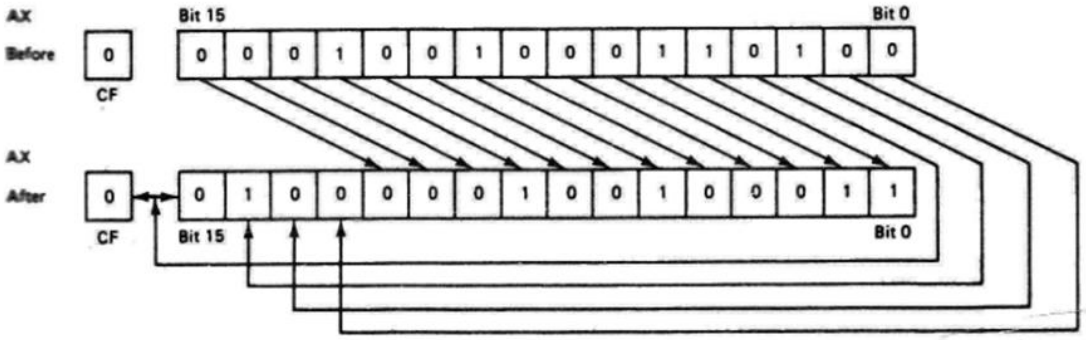
(c)

The **ROR** instruction operates the same way as ROL except that it causes data to be rotated to the right instead of to the left.

Ex:

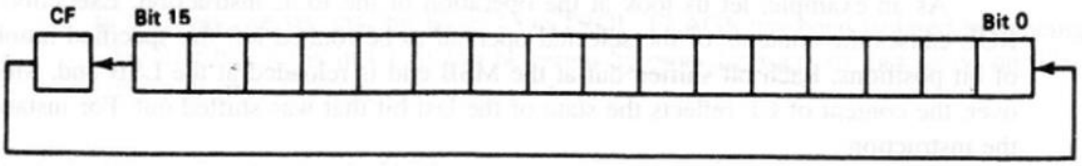
ROR AX, CL

causes the contents of AX to be rotated right by the number of bit positions specified in CL. Figure (d) illustrates the result for CL equal to four.



(d)

The other two rotate instructions, **RCL** and **RCR**, differ from **ROL** and **ROR** in that the bits are rotated through the carry flag. Figure (e) illustrates the rotation that takes place due to execution of the **RCL** instruction. Note that the value returned to bit 0 is the prior content of CF and not bit 15. The value shifted out of bit 15 goes into the carry Hag. Thus, the bits rotate through carry.



(e)

Ex:

What is the result in BX and CF after execution of the following instruction?

RCR BX, CL

Assume that (CL) = 04H, (BX) = 1234H, and (CF) = 0.

Solution:

The original contents of BX are

(BX) = 0001 0010 0011 0100 B = 1234H

Execution of the **RCR** instruction causes a 4-bit rotate right through carry to take place on the data in BX. The resulting contents of BX and CF are

(BX) = 1000 0001 0010 0011 B = 8123H

(CF) = 0B = 0H

	B(X)																
CF	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	Initial state
0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	1st shift
0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	2nd shift
1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	3rd shift
0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	4th shift