Shift and Rotate Instructions

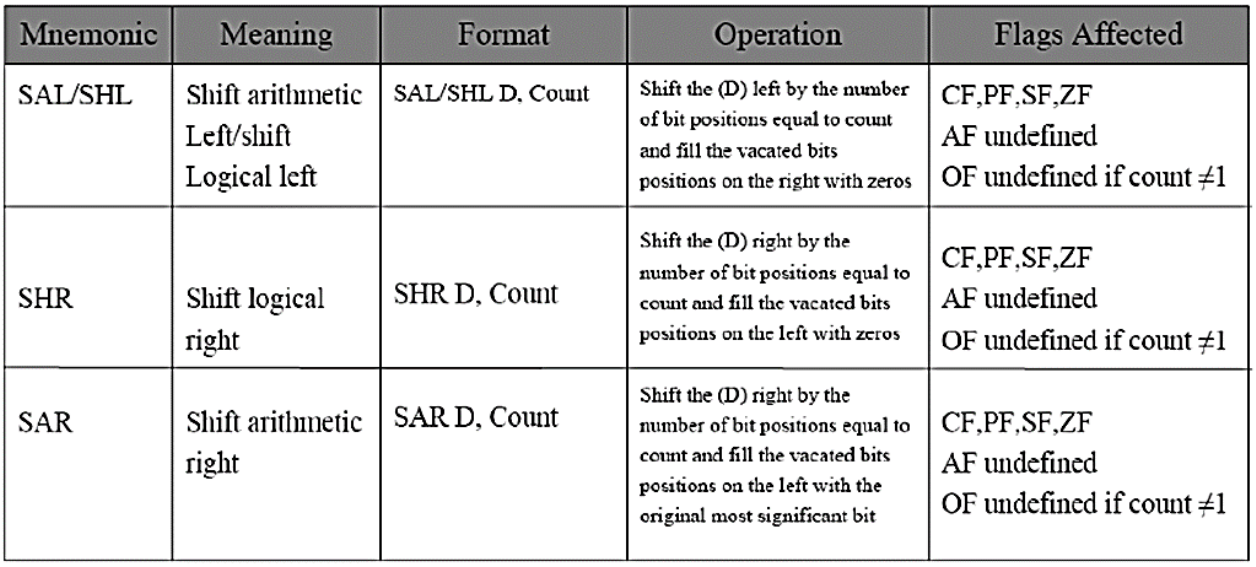
Shift Instructions:

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:

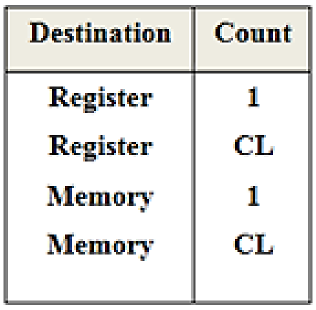
- **SHL** Shift Logical Left
- **SAL** Shift Arithmetic Left
- **SHR** Shift Logical Right
- **SAR** Shift Arithmetic Right

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)

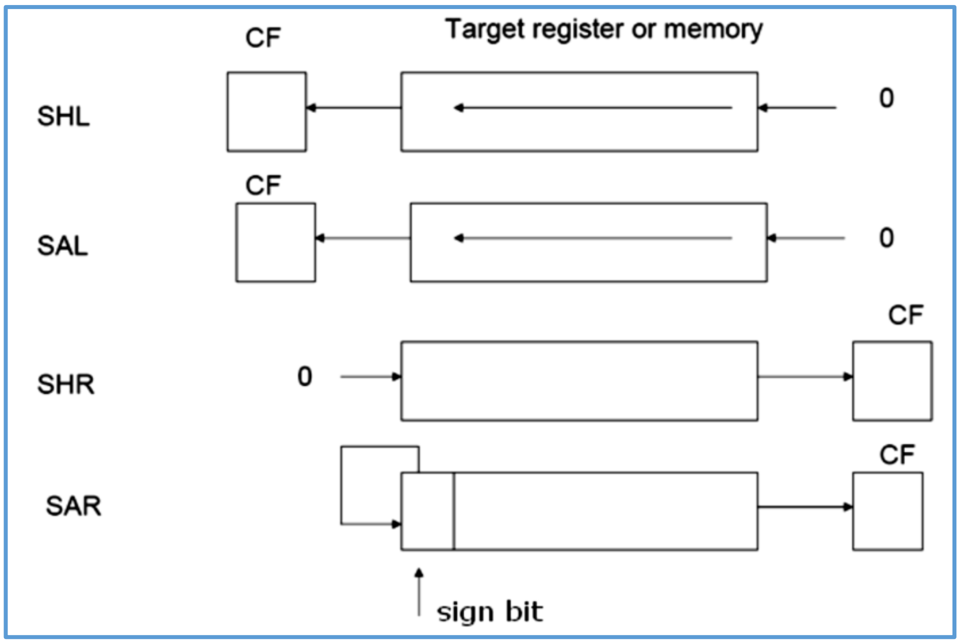
The operation of the shift instructions is described in Figure (a), Note in Fig (b) that:

- 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.

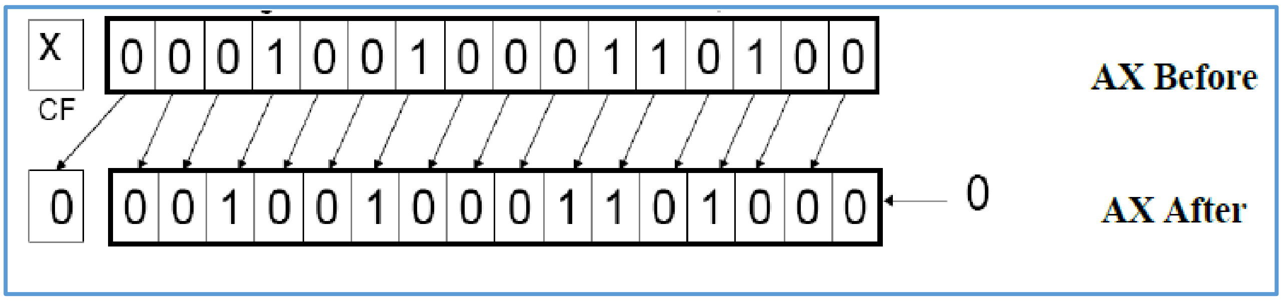
The **SAR** instruction shifts the operand to right and fill the vacated bits to the left with the value of **MSB** (this operation used to shift the signed numbers) as shown in Figure (c).



(c)

Example:
Let AX=1234H what is the value of AX after execution of next instruction
SHL AX, 1

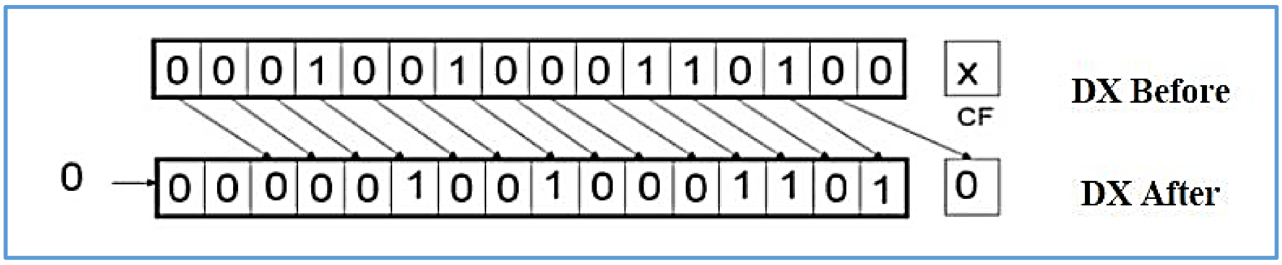
Solution:
Causes the 16-bit register to be shifted 1-bit position to the left where the vacated LSB is filled with zero and the bit shifted out of the MSB is saved in CF.



Example:

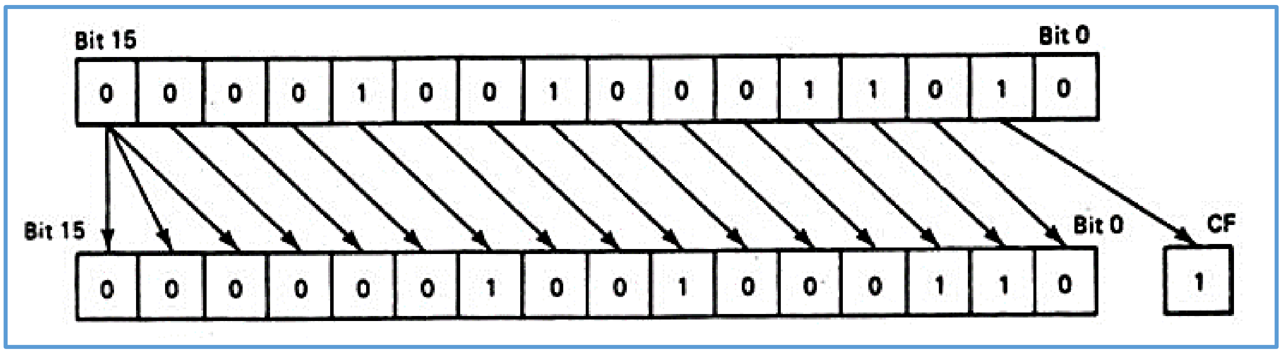
MOV CL, 2H
SHR DX, CL

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



Example: Assume CL= 2 and AX= 091AH.
Determine the new contents of AX and CF after the below instruction is executed.

SAR AX, CL



Rotate Instructions:

The rotate instructions, are similar to the shift instructions, as shown in Figure (a), includes the:

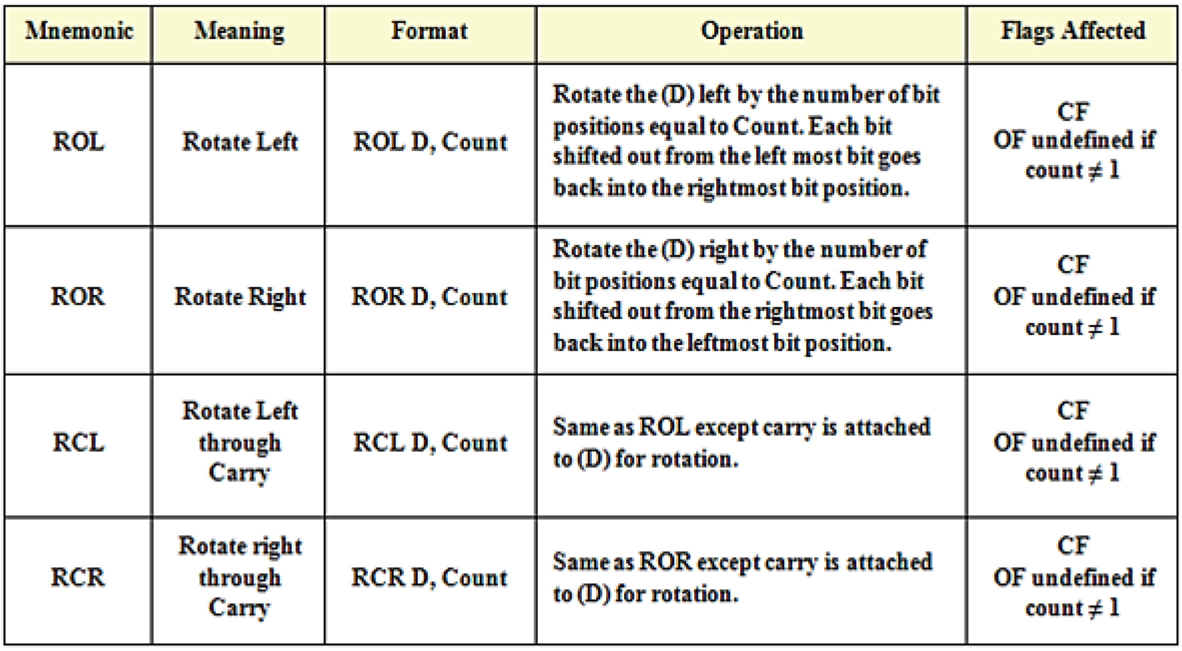
- **ROL** rotate left
- **ROR** rotate right
- **RCL** rotate left through carry
- **RCR** rotate right through carry

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

Figure (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.

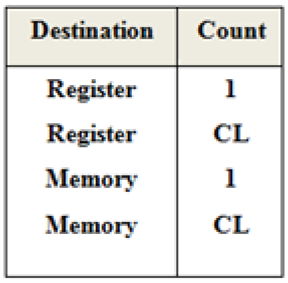
Also, the rotation that takes place can be from 1 to 255 bit positions to the left or to the right. Moreover, in the case of a multi bit rotate, the number of bit positions to be rotated is specified by the value in CL.

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 $\neq$ 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 $\neq$ 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 $\neq$ 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 $\neq$ 1

(a)



Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

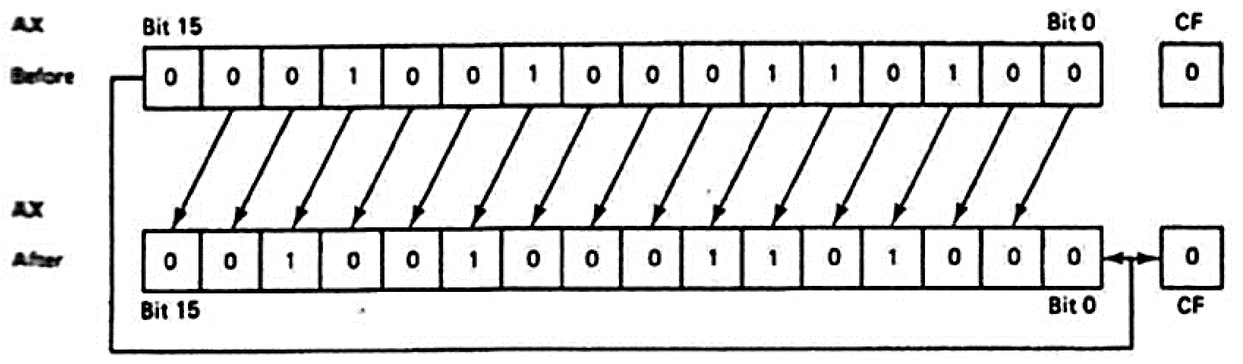
(b)

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.

Example:

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.



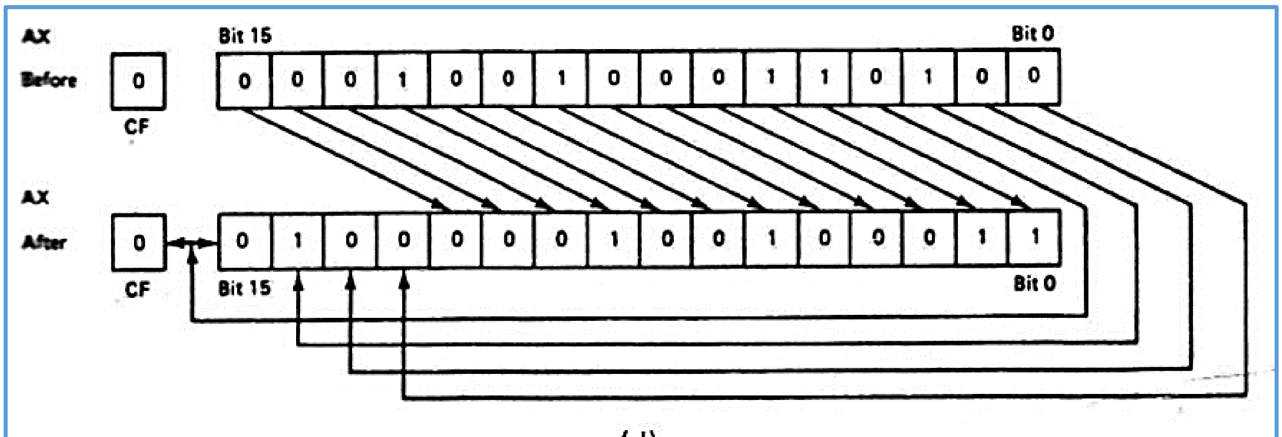
(c)

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.

Example:

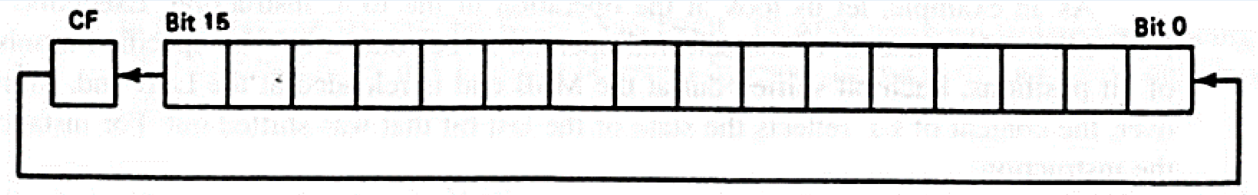
ROR AX, CL

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)