BX = 1234H - 0123H - 0 = 1111H

- DEC subtract 1 from its operand

- NEG BX (2's complement)

  00H – BX 0000 + 2's complement of BX

**EXAMPLE:**

BX = 3A H NEG BX 0011 1010

0000 H +FFC6H 1100 0101 +

1

1100 0110

C 6

## ➢ Multiplication and Division MUL, DIV

- **MUL CL**

  (AX)= AL* CL

- **MUL CX**

  (DX, AX) = AX * CX

- **DIV CL**

  (AH), (AL) = AX/CL 51

And AL the quotient

Where AH is the reminder

- **DIV CX**

DX, AX= (DX,AX)/CX

AX contain the Quotient

DX contain the reminder

**EXAMPLE:**

MUL CL where AL=-1 CL= -2

AX = FF H * FE H = FD02 H

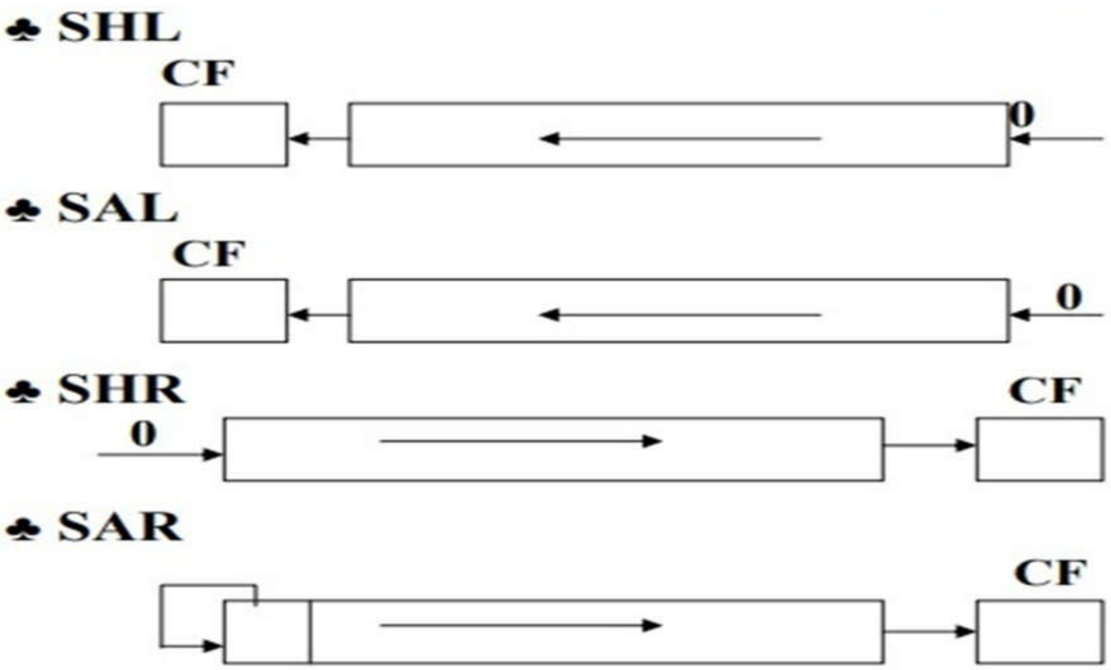
### 3.Logical Instructions (AND, OR, XOR, NOT)

| Instructions | AL |
|---|---|
| MOV AL, 0101 0101B | 0101 0101 B |
| AND AL,0001 1111B | 0001 0101 B |
| OR AL,1100 0000B | 1101 0101 B |
| XOR AL,0000 1111B | 1101 1010 B |
| NOT AL | 0010 0101 B |

### 4. Shift Instructions

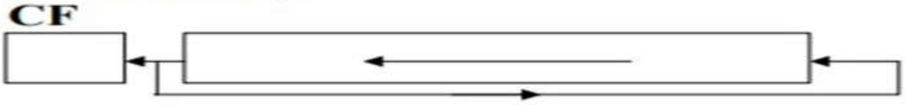
The four types of shift instructions can perform two basic types of shift operations. They are the logical shift and arithmetic shift. Each of these operations can be performed to the right or to the left.

| Instructions | 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 | OF, CF |
| SHR | Shift logical right | SHR D, Count | Shift the D right by the number of bit position equal to count and fill the vacated bit positions on the left with zeros | OF, CF |
| SAR | Shift arithmetic right | SAR D, Count | Shift the D right by the number of bit positions equal to count and fill the vacated bit positions on the left with the original most significant bit | OF,SF, ZF, AF, PF, CF |

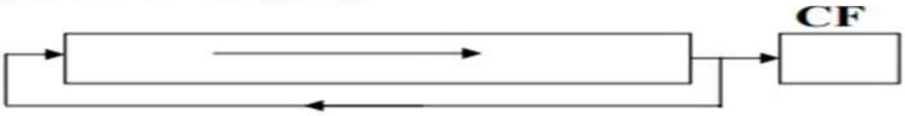


### 5.Rotate Instructions

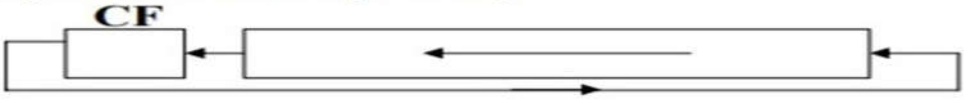
♣ **ROL (Rotate Left)**



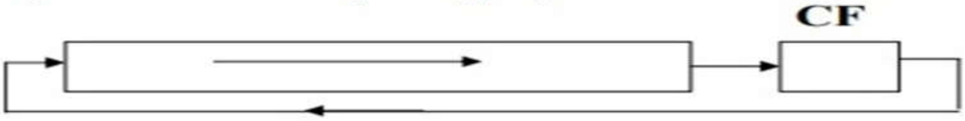
♣ **ROR (Rotate Right)**



♣ **RCL (Rotate Carry Left)**



♣ **RCR (Rotate Carry Right)**



## 6. Advance instruction (Program and Control Instruction)

In this section many of instructions that can be executed by the 8086 microprocessor are described, furthermore, these instructions use to write simple programs. The following topics are discussed in this section:

1. Flag control instructions
2. Compare instruction
3. Jump instructions
4. Push and POP Instruction
5. String instruction

### 1.Flag Control Instruction

The 8086 microprocessor has a set of flags which either monitor the status of executing instruction or control options available in its operation. The instruction set includes a group of instructions which when execute directly affect the setting of the flags. The instructions are:

**LAHF:** load AH from flags

**SAHF:** store AH into flags

**CLC:** clear carry, CF=0

**STC:** set carry, CF=1

**CMC:** complement carry, CF= CF

**CLI:** clear interrupt, IF=0

**STI:** set interrupt, IF=1

**EXAMPLE:-**

Write an instruction to save the current content of the flags in memory location MEM1 and then reload the flags with the contents of memory location MEM2

**Solution:**

LAHF
MOV MEM1, AH

MOV AH, MEM2

SAHF

### 2.Compare Instruction

There is an instruction included instruction set which can be used to compare two 8-bit number or 16-bit numbers. It is the compare (CMP) instruction. The operands can reside in a storage location in memory, a register within the MPU.

| Instruction | Meaning | Format | Operation | Flag affected |
|---|---|---|---|---|
| CMP | Compare | CMP D,S | D-S | CF,AF,OF,PF,SF |

| Destination | Source |
|---|---|
| Register | Register |
| Register | Memory |
| Memory | Register |
| Register | Immediate |
| Memory | Immediate |
| Accumulator | Immediate |

The process of comparison performed by the CMP instruction is basically a subtraction operation. The source operand is subtracted from the destination operand. However the result of this subtraction is not saved. Instead, based on the result the appropriate flags are set or reset.

<u>**EXAMPLE:**</u> lets the destination operand equals 100110012 and that the source operand equals 000110112. Subtraction the source from the destination, we get

```
   ١٠٠١١٠٠١
-  ٠٠٠١١٠١١
_____________
```

Replacing the destination operand with its 2's complement and adding

```
   10011001
+  11100101
___________
  011111102
```

1. No carry is generated from bit 3 to bit 4, therefore, the auxiliary carry flag AF is at logic 0.
2. There is a carry out from bit 7. Thus carry flag CF is set.
3. Even through a carry out of bit 7 is generated; there is no carry from bit 6 to bit 7. This is an overflow condition and the OF flag is set.