# Logic instructions and sample programs

## SECTION 3.3: LOGIC INSTRUCTIONS AND SAMPLE PROGRAMS

In this section we discuss the logic instructions AND, OR, XOR, SHIFT, and COMPARE. Instructions are given in the context of examples.

### AND

AND destination, source

This instruction will perform a logical AND on the operands and place the result in the destination. The destination operand can be a register or in memory. The source operand can be a register, in memory, or immediate.

| X | Y | X AND Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Example 3-5**

Show the results of the following:

```
MOV    BL,35H
AND    BL,0FH              ;AND BL with 0FH.  Place the result in BL.
```

**Solution:**

```
35H    0 0 1 1 0 1 0 1
0FH    0 0 0 0 1 1 1 1
05H    0 0 0 0 0 1 0 1     Flag settings will be:  SF = 0, ZF = 0, PF = 1, CF = OF = 0.
```

AND will automatically change the CF and OF to zero and PF, ZF and SF are set according to the result. The rest of the flags are either undecided or unaffected. As seen in Example 3-5, AND can be used to mask certain bits of the operand. It can also be used to test for a zero operand:

```
        AND     DH,DH
        JZ      XXXX
        ...
XXXX:   ...
```

The above will AND DH with itself and set ZF = 1 if the result is zero, making the CPU fetch from the target address XXXX. Otherwise, the instruction below JZ is executed. AND can thus be used to test if a register contains zero.

## OR

```
OR      destination,source
```

| X | Y | X OR Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

The destination and source operands are ORed and the result is placed in the destination. OR can be used to set certain bits of an operand to 1. The destination operand can be a register or in memory. The source operand can be a register, in memory, or immediate.

The flags will be set the same as for the AND instruction. CF and OF will be reset to zero and SF, ZF, and PF will be set according to the result. All other flags are not affected. See Example 3-6.

The OR instruction can also be used to test for a zero operand. For example, "OR BL,0" will OR the register BL with 0 and make ZF = 1 if BL is zero. "OR BL,BL" will achieve the same result.

**Example 3-6**

Show the results of the following:

```
MOV AX,0504          ;AX = 0504
OR  AX,0DA68H        ;AX = DF6C
```

**Solution:**

```
0504H     0000 0101 0000 0100
DA68H     1101 1010 0110 1000
DF6C      1101 1111 0110 1100
```

Flags will be: SF =1 , ZF = 0, PF = 1, CF = OF = 0.
Notice that parity is checked for the lower 8 bits only.

## XOR

XOR dest,src

| X | Y | X XOR Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The XOR instruction will eXclusive-OR the operands and place the result in the destination. XOR sets the result bits to 1 if they are not equal; otherwise, they are reset to 0. The flags are set the same as for the AND instruction. CF = 0 and OF = 0 are set internally and the rest are changed according to the result of the operation. The rules for the operands are the same as in the AND and OR instructions. See Examples 3-7 and 3-8.

**Example 3-7**

Show the results of the following:

```
MOV     DH,54H
XOR     DH,78H
```

**Solution:**

```
54H          0 1 0 1 0 1 0 0
78H          0 1 1 1 1 0 0 0
2C           0 0 1 0 1 1 0 0          Flag settings will be: SF = 0, ZF = 0, PF = 0, CF = OF = 0.
```

**Example 3-8**

The XOR instruction can be used to clear the contents of a register by XORing it with itself. Show how "XOR AH,AH" clears AH, assuming that AH = 45H.

**Solution:**

```
45H          01000101
45H          01000101
00           00000000          Flag settings will be: SF = 0, ZF = 1, PF =1 , CF = OF = 0.
```

XOR can also be used to see if two registers have the same value. "XOR BX,CX" will make ZF = 1 if both registers have the same value, and if they do, the result (0000) is saved in BX, the destination.

Another widely used application of XOR is to toggle bits of an operand. For example, to toggle bit 2 of register AL:

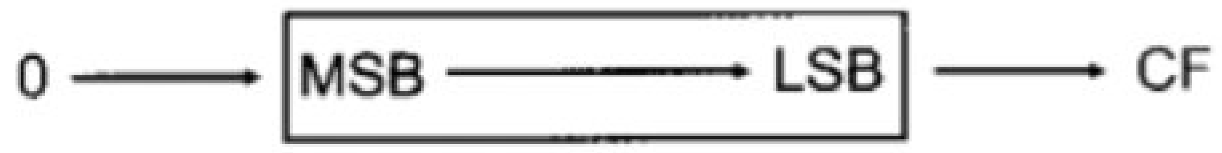
```
XOR     AL,04H          ;XOR  AL with 0000 0100
```

This would cause bit 2 of AL to change to the opposite value; all other bits would remain unchanged.

## SHIFT

There are two kinds of shift: logical and arithmetic. The logical shift is for unsigned operands, and the arithmetic shift is for signed operands. Logical shift will be discussed in this section and the discussion of arithmetic shift is postponed to Chapter 6. Using shift instructions shifts the contents of a register or memory location right or left. The number of times (or bits) that the operand is shifted can be specified directly if it is once only, or through the CL register if it is more than once.

**SHR**: This is the logical shift right. The operand is shifted right bit by bit, and for every shift the LSB (least significant bit) will go to the carry flag (CF) and the MSB (most significant bit) is filled with 0. Examples 3-9 and 3-10 should help to clarify SHR.



**Example 3-9**

Show the result of SHR in the following:

```
MOV     AL,9AH
MOV     CL,3        ;set number of times to shift
SHR     AL,CL
```

**Solution:**

```
9AH = 10011010
      01001101        CF=0    (shifted once)
      00100110        CF=1    (shifted twice)
      00010011        CF=0    (shifted three times)
```

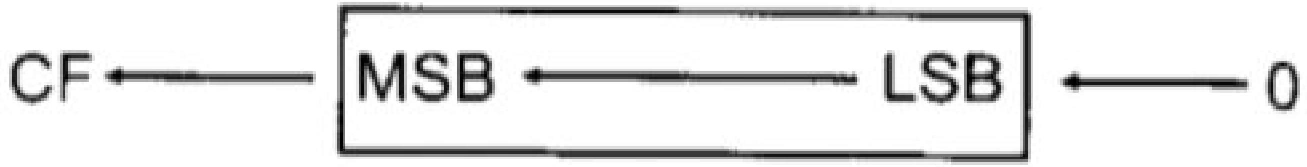
After three times of shifting right, AL = 13H and CF = 0.

If the operand is to be shifted once only, this is specified in the SHR instruction itself rather than placing 1 in the CL. This saves coding of one instruction:

```
MOV   BX,0FFFFH              ;BX=FFFFH
SHR   BX,1                   ;shift right BX once only
```

After the shift above, BX = 7FFFH and CF = 1. Although SHR does affect the OF, SF, PF, and ZF flags, they are not important in this case. The operand to be shifted can be in a register or in memory, but immediate addressing mode is not allowed for shift instructions. For example, "SHR 25,CL" will cause the assembler to give an error.

**SHL**: Shift left is also a logical shift. It is the reverse of SHR. After every shift, the LSB is filled with 0 and the MSB goes to CF. All the rules are the same as SHR.



## Example 3-11

Show the effects of SHL in the following:

```
MOV     DH,6
MOV     CL,4
SHL     DH,CL
```

**Solution:**

```
                   00000110
CF=0               00001100              (shifted left once)
CF=0               00011000
CF=0               00110000
CF=0               01100000              (shifted four times)
```

After the four shifts left, the DH register has 60H and CF = 0.

Example 3-11 could have been coded as

```
MOV   DH,6
SHL   DH,1
SHL   DH,1
SHL   DH,1
SHL   DH,1
```