# Microprocessor

**Just Note: Logic instructions**

**Logic instruction— AND, OR, XOR, NOT**

Example

Describe the result of executing the following sequence of instructions

```
MOV AL,01010101B
AND AL,00011111B
OR AL,11000000B
XOR AL,00001111B
NOT AL
```

| AND | OR | XOR | NOT |
|---|---|---|---|
| 01010101 | 00010101 | 11010101 | |
| 00011111 | 11000000 | 00001111 | 11011010 |
| 00010101 | 11010101 | 11011010 | 00100101 |
| $15_{16}$ | $D5_{16}$ | $DA_{16}$ | $25_{16}$ |

**-a 0100**
13B2:0100 **mov al,55**
13B2:0102 **and al,1f**
13B2:0104 **or al,c0**
13B2:0106 **xor al,0f**
13B2:0108 **not al**
13B2:010A
**-t**
**AX=0055** BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=0102 NV UP EI PL NZ NA PO NC
13B2:0102 241F **AND AL,1F**

Mask off the 3 most significant bits of AL

**-t**
**AX=0015** BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=0104 NV UP EI PL NZ NA PO NC
13B2:0104 0CC0 **OR AL,C0**

Setting the two most significant bits of AL

**-t**
**AX=00D5** BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=0106 NV UP EI NG NZ NA PO NC
13B2:0106 340F **XOR AL,0F**
**-t**
**AX=00DA** BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=0108 NV UP EI NG NZ NA PO NC
13B2:0108 F6D0 **NOT AL**
**-t**
**AX=0025** BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=010A NV UP EI NG NZ NA PO NC
13B2:010A 0000 ADD [BX+SI],AL DS:0000=CD

A common use of logic instructions is to mask a group of bits of a byte, word, or double word of data. By mask, we mean to clear the bit or bits to zero. Remember that when a bit is ANDed with another bit that is at logic 0, the result will always be 0. On the other hand, if a bit is ANDed with a bit that is at logic 1, its value will remain unchanged. Thus we see that the bits that are to be masked must be set to 0 in the mask
For instance, in the instruction
**AND AX,000FH**
It would mask off the upper 12 bits of the word of data in the destination AX

The OR instruction can be used to set a bit or bits in a register or storage location in memory to logic 1.

**SHIFT Instructions**

| | |
|---|---|
| SHL | logical Shift left |
| SHR | logical shift right |
| SAL | shift arithmetic left |
| SAR | shift arithmetic right |
| SHLD | Double-precision shift left |
| SHRD | Double-precision shift right |

**SHL AX,1**

Causes the 16-bit contents of the AX register to be shifted 1 bit position to the left. Here we see that the vacated LSB location is filled with zero and the bit shifted out of the MSB is saved in CF

**SHR AX,CL**

Assuming that CL contains the value $02_{16}$ , two MSBs have been filled with 0s, and the last bit shifted out at the LSB, which is zero, is placed in the carry flag

**Rotate Instructions**

| | |
|---|---|
| ROL | Rotate left |
| ROR | Rotate Right |
| RCL | Rotate left through carry |
| RCR | Rotate right through carry |

Example

What is the result in BX and CF after execution of the following instruction?
**`RCR BX,CL`**
Assume that prior to execution of the instruction, CL = $04_{16}$, BX = $1234_{16}$, and CF = 0

Solution
**-r cx**
CX 0000
:**0004**
**-r bx**
BX 0000
:**1234**
**-r f**

NV UP EI PL NZ NA PO NC -
**-a 0100**
13B2:0100 **rcr bx,cl**
13B2:0102
**-t**
AX=0000 **BX=8123** CX=0004 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13B2 ES=13B2 SS=13B2 CS=13B2 IP=0102 NV UP EI PL NZ NA PO NC
13B2:0102 241F AND AL,1F

**BIT TEST AND BIT SCAN INSTRUCTIONS**

The bit test and bit scan instructions enable a programmer to test the logic value of a bit in either a register or storage location in memory

BT → Bit Test: Have two operands. The first operand identifies the register or memory location that contains the bit that is to be tested. The second operand contains an index that selects the bit that is to be tested. When the instruction is executed, the value of the tested bit is saved in the carry flag.

Once the state of the bit is saved in CF, it can be tested through software. For instance, a conditional jump instruction could be used to test the value in CF, and if CF equals 1, program control could be passed to a service routine

| | | | |
|---|---|---|---|
| **BT** | Bit Test | BT D,S | Saves the value of the bit in D specified by the value in S in CF |
| **BTR** | Bit Test and Reset | BTR D,S | Saves the value of the bit in D specified by the value in S in CF and then **resets** the bit in D |
| **BTS** | Bit Test and set | BTS D,S | Saves the value of the bit in D specified by the value in S in CF and then **sets** the bit in D |
| **BTC** | Bit Test and complement | BTC D,S | Saves the value of the bit in D specified by the value in S in CF and then **complements** the bit in D |

Example

Describe the operation that is performed by the instruction
BTC BX,7
Assume that register BX contains the value $03F0_{16}$

Solution
0000 0011 **1**111 0000 → 0000 0011 **0**111 0000 = 0370
CF = 1