



## **Arithmetic and Logical instructions**

<u>AND instruction</u>: Logical AND between all bits of two operands. Result is stored in operand1.

## Algorithm:

Operand1 = operand1 AND operand2 (between all bits of two operands)
These rules apply:

0 AND 0 = 0 0 AND 1 = 0 1 AND 0 = 01 AND 1 = 1

As you see we get 1 only when both bits are 1.

Note: We use AND instruction to reset (clear), convert and check bits.

Ex1. Reset b 4,7 of DL register (assume DL =97).

MOV DL,97 AND DL,6F h

b7	b6	b5	b4	b3	b2	b1	b0	
1	0	0	1	0	1	1	1	AND
0	1	1	0	1	1	1	1	6F
0	0	0	0	0	1	1	1	

Ex2. Write instructions to convert 'a' to 'A'. Note that the 'a'= 61 h and 'A'= 41h.

MOV AL, 61 h; AL = 61 h = 0110 0001b = 'a' AND AL, DF h; AL = 41 h = 0100 0001b = 'A'

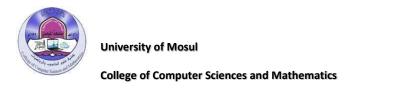
0110 1101		AND DF	
0100	0001		

<u>TEST instruction</u> The same as **AND** but **for flags only** (The result is not store in anywhere). Means Logical AND between all bits of two operands for flags only. These flags are effected: **ZF**, **SF**, **PF**.

## Algorithm:

Operand1 AND operand2 (the result is not store in anywhere)

After TEST we use conditional jump instructions.





**OR** instruction - Logical OR between all bits of two operands. Result is stored in first operand.

Algorithm:

Operand1 = operand1 OR operand2 (between all bits of two operands)

These rules apply: 0 OR 0 = 0

> 0 OR 1 = 11 OR 0 = 11 OR 1 = 1

As you see we get 1 every time when at least one of the bits is 1.

Note: We use OR instruction to set and convert bits.

Ex. Write instructions to set b1 and b15 of BX register. If you know BX= 76F0 h.

MOV BX, 76F0 h; BX = 76F0 hOR BX, 8002 h; BX = F6F2 h

0111 0110 1111 0000 OR 1000 0000 0000 0010 **1**111 0110 1111 00**1**0

**XOR** instruction: Logical XOR (exclusive OR) between all bits of two operands.

Algorithm:

Operand1 = operand1 XOR operand2 (between all bits of two operands)

These rules apply: 0 XOR 0 = 0

> 0 XOR 1 = 11 XOR 0 = 11 XOR 1 = 0

As you see we get 1 every time when bits are different from each other.

Note: We use XOR instruction to complement and convert bits.

Ex. Complement b1,2,3 of CL register assume CL=32 h

MOV CL, 32 h

0011 0010 XOR XOR CL,0E 0000 1110

0011 110 0