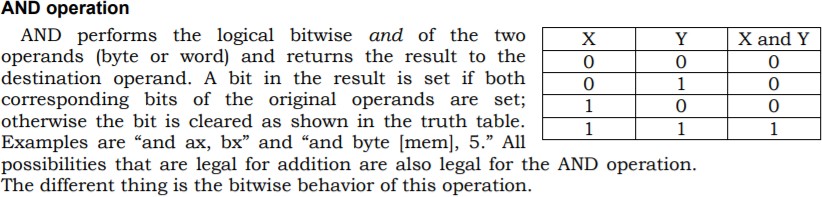
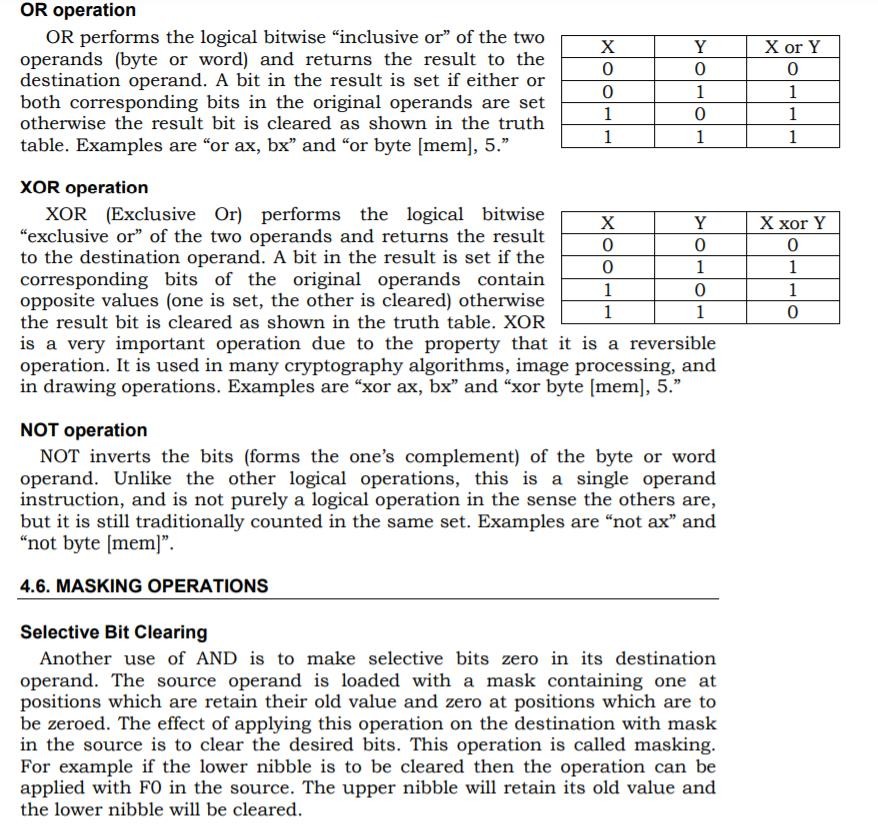
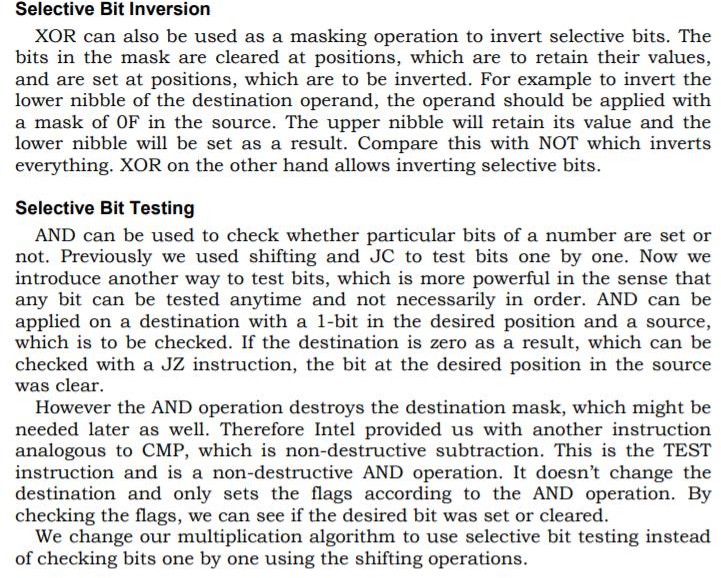
Computer Organization and Assembly Language

|  |  |
| --- | --- |
| **Lab 07** | |
| **Topic** | * Arithmetic & Logical instructions * Selective bit setting/clearing/complimenting * Extended addition and subtraction |







# Example 1:

**Let the binary of a number (0XABCD) is 10101011 11001101.**

# Set the fourth bit from 0 to so on.

mov ax,0xABCD

or ax,0000000000010000b

mov ax,0x4c00 int 21h

# Clear the L.S.B.

mov ax,0xABCD

and ax,1111111111111110b

mov ax,0x4c00 int 21h

# Invert the M.S.B.

mov ax,0xABCD

xor ax,1000000000000000b

mov ax,0x4c00 int 21h

# Set the 4th ,7th and 13th bit.

mov ax,0xABCD

or ax,0010000010010000b

mov ax,0x4c00 int 21h

***Note: logical operations are bitwise operations.***

**Example 4(Extended addition)**

MOV AX, [Num1] ;loads two bytes into AX register, AX=FFFF

MOV BX, [Num1+2] ;loads Next two bytes into AX register, AX=0001 ADD AX, [Num2] ; adds into AX; AX=AX+0002;

ADC BX, [Num2+2]; Add with carry instruction

MOV [SUM],AX ; Move the lower bits into Sum variable

MOV [SUM+2],BX ; Move the higher bits into Sum variable higher bits

mov ax,0x4c00 int 21h

Num1: dd 0x0001FFFF Num2: dd 0x00010002 SUM: dd 0

# Example 5(Extended subtraction)

MOV AX,[Num2] ;loads two bytes into AX register, AX=0002

MOV BX, [Num2+2] ;loads Next two bytes into AX register, AX=0001 SUB AX, [Num1] ; sub into AX; AX=AX-FFFF;

SBB BX, [Num1+2]; Subtraction with borrow.

MOV [SUM],AX ; Move the lower bits into Sum variable

MOV [SUM+2],BX ; Move the higher bits into Sum variable higher bits

mov ax,0x4c00 int 21h

Num1: dd 0x0001FFFF Num2: dd 0x00010002 SUM: dd 0

|  |  |
| --- | --- |
| a = 10, b = 20, c = 5 , sum=0;  if (a < =b)  {  // L1  if (a < =c)  {  // L2  if (b>c)  {  // L3  sum = a + b + c;  }  else  {  // L4  sum = a - b - c;  }  }  else  {  // L5  sum = a + b - c;  }  }  else  {  // L6  sum = a - b + c;  } | [org 0x100] mov al,[a]  mov bl,[b]  mov cl,[c]  cmp al,bl jng l1  jg l6 l1:  cmp al,cl jng l2  jg l5 l2:  cmp bl,cl jg l3  jng l4 l3:  mov [sum],al add [sum],bl add [sum],cl jmp exit  l4:  mov [sum],al sub [sum],bl sub [sum],cl jmp exit  l5:  mov [sum],al add [sum],bl sub [sum],cl jmp exit  l6:  mov [sum],al sub [sum],bl add [sum],cl  exit:  mov ax,0x4c00 int 21h  a: db 10  b: db 20  c: db 5  sum: db 0 |

**Question#1**

Calculate the number of one bits in BX and complement an equal number of least significant bits in AX. Any random value in AX and BX.

HINT: Use the XOR instruction

**Question#2**

Write a program that will first invert the 9th, 10th and 4th bit of Ax register.

1. Now Set the 8th, 12th and 3rd bit of Ax.

**Question#3**

We want to swap two numbers using bit masking xor operation.

Eg: a = 4 (100 in binary), b = 6 (110 in binary)

We want a = 6 and b = 4 as our answer.

Hint: 6 xor 6= 0 and 4 xosssr 4=0

**Question#4**

Convert the following c++ code into assembly code

a = 5, b = 10, c = 20 , sum=0;

if (a < b)

{

// L1

if (a < c)

{

// L2

if (b>=c)

{

// L3

sum = a - b + a;

}

else

{

// L4

sum = a + b - b;

}

}

else

{

// L5

sum = a - b + c;

}

}

else

{

// L6

sum = a + b - c;

}