

**EEE 316**

**Microprocessor and Interfacing Sessional**

**Experiment 2**

**Name of the experiment: Logical Instructions  
and Jump Commands in Assembly Language**

**Venue: VLSI Lab/RNS Lab**

**Department of Electrical and Electronic  
Engineering, BUET**

### **Objective:**

- Using logical instructions in assembly language.
- Incorporating Jump commands in assembly programs.
- Writing simple assembly language programs with logical and Jump instructions.

### **Introduction:**

Both logical instructions and Jump commands are widely used in assembly language. These commands are explained below.

### **Logical instructions:**

Logical instructions include NOT, AND, OR, XOR, TEST etc. instructions. Their job is to compare the data values and make results according to logic specified. For example,

```
MOV  BX, 30H ; In binary 110000
NOT  BX      ; In binary 001111
```

This code takes BX value and then complements all the bits and stores the new value to BX. So it stores 0F value in BX after executing NOT operation. For another example,

```
MOV  BX, 70H ; In binary 1110000
MOV  CX, 40H ; In binary 1000000
AND  CX, BX  ; In binary 1000000
```

AND operation performs bit by bit AND operation and then stores the value in first operand. In upper code CX holds the final result.

```
MOV  BX, 70H ; In binary 1110000
MOV  CX, 40H ; In binary 1000000
OR   CX, BX  ; In binary 1110000
```

OR operation performs bit by bit OR operation and then stores the value in first operand. In upper code CX holds the final result. Similar case happens for XOR and it is given below,

```
MOV  BX, 70H ; In binary 1110000
MOV  CX, 40H ; In binary 1000000
XOR  CX, BX  ; In binary 0110000
```

Test operation is a little different from AND operation. It performs bit by bit AND operation but it does not change any operands value.

```
MOV  BX, 70H ; In binary 1110000
MOV  CX, 40H ; In binary 1000000
TEST CX, BX  ; In binary CX value is 1000000
```

All the logical instructions stated above upgrades all the flag register values except AF register. NOT command does not effect any flags. How flags are affected is stated below.

```
MOV  BX, 70H ; In binary 1110000
MOV  CX, 40H ; In binary 1000000
AND  CX, BX  ; In binary 1110000
```

After this operation Zero Flag is 0 (ZF = 0; as the value of CX is not 0), Carry Flag is 0 (CF = 0; as there is no carry), Parity Flag is 0 (PF = 0; as there are odd number of 1's), Sign Flag is 0 (SF = 1), Overflow Flag is 0 (OF = 0; as there is no overflow). In this all the flags can be determined.

Do not confuse yourself with semicolon given after each line in assembly codes above. Comments are written after semi colon ';' in assembly language.

### Exercise Part 1:

Write following codes and perform indicated operations.

#### (a) Program 1:

```
CODE SEGMENT
    ASSUME CS:CODE, DS:CODE

    MOV  BX, 3256H
    MOV  CX, 1554H
    AND  CX, BX

    HLT

CODE ENDS
    END
```

Observe content of CX register. What operation happened here?

---

---

#### (b) Program 2:

```
CODE SEGMENT
    ASSUME CS:CODE, DS:CODE
    MOV  BX, 3256H
    MOV  CX, 1554H
    XOR  CX, BX
```

```
        HLT  
CODE   ENDS  
      END
```

Observe content of CX register. What operation happened here?

---

---

### (c) Program 3:

```
CODE   SEGMENT  
      ASSUME CS:CODE, DS:CODE  
  
      MOV   AX, 1027H  
      MOV   BX, 5A27H  
      MOV   CX, 54A5H  
  
      OR    AX, BX  
  
      XOR   AX, CX  
  
      NOT   AX  
  
      TEST  CX, BX  
  
      AND   CX, AX  
  
      HLT  
CODE   ENDS  
      END
```

Perform this operation in single step mode and write the values of registers for every step. Obtain binary values for upper hexadecimal values and perform bit by bit operation for every step. Compare your hand calculation with obtained result.

### JUMP Commands:

Sometimes it is necessary to go from one line of program to another line without executing some intermediate lines. For this Jump commands are used. We can explain this with a simple example.

```

MOV  AX, 3254H
MOV  BX, 1F4BH
MOV  CX, 412AH

ADD  AX, CX

JMP  L3T2

SUB  AX, BX

L3T2: AND  AX, BX

      HLT
    
```

In this example L3T2 is a level. As we can see in fifth line JMP command is used. It makes the program to go from fifth line to L3T2 level that is seventh line. So sixth line is not executed.

There are two types of Jump commands. These are (i) Conditional jump and (ii) Unconditional Jump. Previous example is an unconditional jump. Conditional Jumps are like if statements. If some flags are affected only then these jump instructions executed. We can look at the following example,

```

MOV  AX, 125BH
MOV  BX, 125BH
MOV  CX, 412AH

SUB  AX, BX

JZ   L3T2

DIV  BX

L3T2: AND  AX,CX

      HLT
    
```

Clearly observe the code. In fourth line subtraction operation is performed. As both AX and BX have same value. Their subtracted value is 0. So ZF is set to 1. In fifth line JZ L3T2 is written. It means if ZF = 1 then go to L3T2: . Otherwise continue. As ZF = 1, program moves to eighth line. This is a conditional Jump. Some other conditional jumps are,

Command	Condition of Jump
JA/JNBE	CF = 0, ZF = 0
JBE/JNA	CF = 0 or ZF = 0
JNB/JAE/JNC	CF = 0
JB/JNAE/JC	CF = 1

JG/JNLE	SF $\nabla$ OF = 0, ZF = 0
JLE/JNG	SF $\nabla$ OF = 0, ZF = 1
JGE/JNL	SF $\nabla$ OF = 0
JL/JNGE	SF $\nabla$ OF = 1
JZ/JE	ZF = 1
JNZ/JNE	ZF = 0
JS	SF = 1
JNS	SF = 0
JPE/JP	PF = 1
JPO/JNP	PF = 0
JO	OF = 1
JNO	OF = 0
JCXZ	CX = 0

### Exercise Part 2:

Write following codes and perform indicated operations.

#### (a) Program 1:

```

CODE SEGMENT
    ASSUME CS:CODE, DS:CODE

        MOV  AX, 7A24H
        MOV  BX, 15A3H

        SUB  AX, BX

        JMP  L3T2

EEE316:  DIV  BX

        JMP  Last

L3T2:   MOV  CX, 45B1H

        AND  AX, CX
        TEST AX, BX

        JMP  EEE316

Last:   HLT
    
```

```
CODE ENDS  
END
```

Perform this operation in single step mode and write the values of registers for every step. Explain why we need 'Last' termed level? What will happen if it is not used?

**(b) Program 1:**

```
CODE SEGMENT  
ASSUME CS:CODE, DS:CODE  
  
        MOV  AX, 7A24H  
        MOV  BX, 95A3H  
  
        ADD  AX, BX  
  
        JC   L3T2  
  
EEE316: OR   AX, 23H  
  
        JNZ  Last  
  
L3T2:   MOV  CX, 0FC7H  
  
        SUB  AX, CX  
  
        JZ   EEE316  
  
Last:   HLT  
  
CODE ENDS  
END
```

Update the register values in every step.

**Home Task:**

1. Write an assembly code that will determine whether a number is greater than 5 or equal of less, and put 0 or 1 or 2 for the conditions in DX.
2. Subtract 86B1H from 3F42H and store 0 in CX if overflow occurs and 1 if no overflow occurs.
3. Take 2 arbitrary numbers x and y. If  $x > 1000H$  perform  $x+y$ . If  $y < 1000H$  perform  $x-y$ . If  $x > 1000H$  and  $y < 100H$  perform  $x = x'$ .