Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Experiment 5:
Logical Operations and Conversions (Decimal to Hex)

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Microprocessor and Interfacing**	Code	:	**CSE2006**
Faculty	:	**Dr. Florence Gnana Poovathy J**	Slot	:	**L15+L16**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

Logical Operations and Conversion

Aim: To perform logical operations, shifts, BCD-ASCII conversion, and decimal to hexadecimal conversion.

Tool Used: Assembler – MASM611

Algorithm:

Step 1: First of all, mount the c drive using the command: **mount c c:\masm611\bin**

Step 2: After pressing **enter,** type **c:** and press enter.

Step 3: Now give a command, **filename.asm** for writing/editing the code and the write the code.

Step 4: A pop window appears; there we have to write out code(instructions) following the logic given below.

For AND:

1. Move an operand to AX register.
2. Move the next operand to BX register.
3. AND AX, BX performs bitwise AND, on the operands, and stores it in AX.
4. HLT brings the process to a halt.

For OR:

1. Move an operand to AX register.
2. Move the next operand to BX register.
3. OR AX, BX performs bitwise OR, on the operands, and stores it in AX.
4. HLT brings the process to a halt.

For NOT:

1. Move an operand to AX register.
2. NOT AX, performs bitwise NOT, on the operands, and stores it in AX.
3. HLT brings the process to a halt.

For XOR:

1. Move an operand to AX register.
2. Move the next operand to BX register.
3. XOR AX, BX performs bitwise XOR, on the operands, and stores it in AX.
4. HLT brings the process to a halt.

For SHL, SHR

1. Move an operand to AX register.
2. Move another operand to BX register.
3. Move to CL the required number of shifts you want.
4. SHR AX, CL and SHL AX, CL will do bitwise shifting to the right, and left respectively, for as many counts as mentioned in CL.
5. SHR BX, 1 and SHL BX, 1 will do bitwise shifting to the right and left respectively for 1 bit.
6. HLT brings the process to a halt.

For BCD to ASCII

1. Move the operand to AX register.
2. Move it to BX register, and perform AND operation with BX and 0FH.
3. Add 30H to the result in BX.
4. Move 04 to CL, and perform right rotation on AX for 4 bits.
5. Perform AND operation on AX and 0FH.
6. Add 30H to the result in AX.
7. HLT brings the process to a halt.

For Decimal to Hex

1. Move 2906H to AX register, and perform AND operation with AX and 0FH.
2. Move 1H to DX, and multiply AX with DX.
3. Move the result stored in AX to BX.
4. Move 2906H to AX again, and 4H to CL.
5. Perform ROR on AX for as many counts mentioned in CL, and perform AND operation with AX and 0FH.
6. Move 0AH to DX, and multiply DX with AX.
7. Add the result stored in AX to BX.
8. Move 2906H to AX again, and 8H to CL.
9. Perform ROR on AX for as many counts mentioned in CL, and perform AND operation with AX and 0FH.
10. Move 64H to DX, and multiply DX with AX.

11. Add the result stored in AX to BX.

12. Move 2906H to AX again, and 0CH to CL.

13. Perform ROR on AX for as many counts mentioned in CL, and perform AND operation with AX and 0FH.

14. Move 03E8H to DX, and multiply DX with AX.

15. Add the result stored in AX to BX.

16. HLT brings the process to a halt.

Step 5: Now give a command, **masm <filename>.asm** for running the code. The object file is created.

Step 6: Now give a command, **link <filename>.obj** to link the object file to library file present in the bin folder.

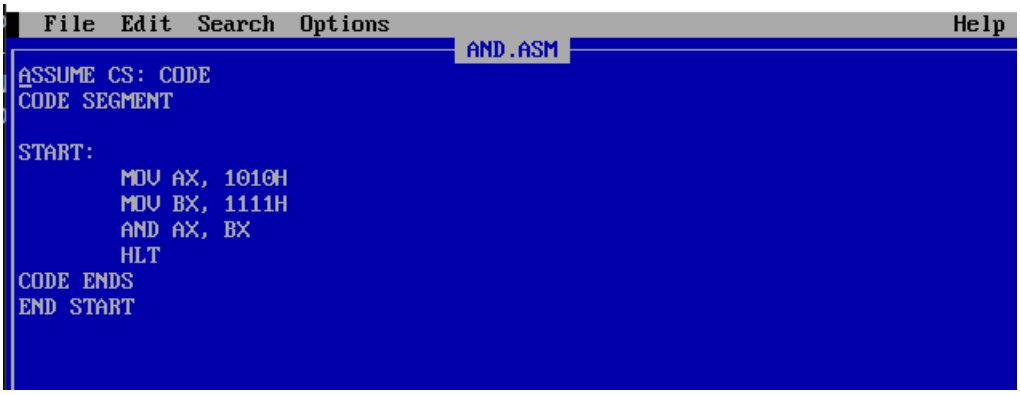
Step 7: Press **ENTER** four times.

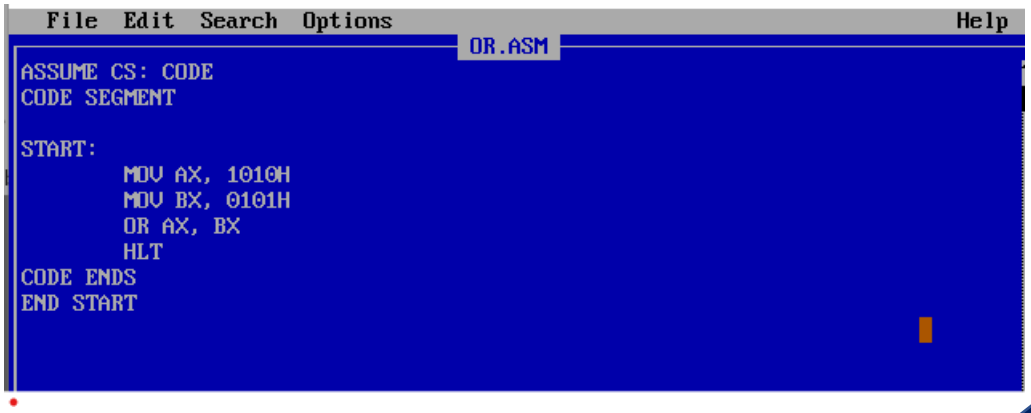
Step 8: Write **debug <filename>.exe**

-u

-g (followed by the **address of HLT or INT** to view the values in registers).

Program:





NOT.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 1010H
        NOT AX
        HLT
CODE ENDS
END START
```

File Edit Search Options Help

XOR.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 0102H
        MOV BX, 0403H
        XOR AX, BX
        HLT
CODE ENDS
END START
```

SHL.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 2906H
        MOV BX, 0110H
        MOV CL, 4

        SHL AX, CL
        SHL BX, 1
        HLT
CODE ENDS
END START
```

SHR.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 0296H
        MOV BX, 0110H
        MOV CL, 3

        SHR AX, CL
        SHR BX, 1
        HLT

CODE ENDS
END START
```

BCDASCII.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 29H
        MOV BX, AX
        AND BX, 0FH
        ADD BX, 30H

        MOV CL, 4
        ROR AX, CL

        AND AX, 0FH
        ADD AX, 30H
        HLT
CODE ENDS
END START
```

DECHEX.ASM

```
ASSUME CS: CODE
CODE SEGMENT

START:
        MOV AX, 2906H
        AND AX, 0FH
        MOV DX, 01H
        MUL DX
        MOV BX, AX

        MOV AX, 2906H
        MOV CL, 04H
        ROR AX, CL
        AND AX, 0FH
        MOV DX, 0AH
        MUL DX
        ADD AX, BX
        MOV BX, AX

        MOV AX, 2906H
        MOV CL, 08H
        ROR AX, CL
        AND AX, 0FH
        MOV DX, 64H
        MUL DX
        ADD AX, BX
        MOV BX, AX

        MOV AX, 2906H
        MOV CL, 0CH
        ROR AX, CL
        AND AX, 0FH
        MOV DX, 03E8H
        MUL DX
        ADD AX, BX
        HLT
CODE ENDS
END START
```

Sample Input:

1.) For AND operation: AX=1010H, BX=1111H

2.) For OR operation: AX=1010H, BX=0101H

3.) NOT: AX=1010H

4.) XOR: AX=0102H, BX=0403H

5.) SHL: AX=2906, BX=0110H

6.) SHR: AX=0296H, BX=0110H

7.) BCD-ASCII: AX=0029H

8.) Decimal-Hex: AX = $(2906)_{10}$

Sample Output:

1. AND operation between AX=1010H and BX=1111H should give 1010.
2. OR operation between AX=1010H and BX=0101H should give 1111.
3. NOT should give EFEF for AX=1010.
4. XOR should give 0501 for AX=0102H, BX=0403H.
5. SHL should give 9060 for SHL AX, CL and 0220 for SHL BX,01.
6. SHR should give 0052 for SHR AX, CL and 0088 for SHL BX,01
7. BCD-ASCII for 29H, the expected result would be 32H for 2, and 39H for 3.
8. Decimal-Hex for 2906 in decimal result should be B5A in hex.

Registers and Flags:

```
C:\>debug and.exe
-u
0764:0000 B81010        MOV     AX,1010
0764:0003 BB1111        MOV     BX,1111
0764:0006 23C3          AND     AX,BX
0764:0008 F4            HLT
0764:0009 BA3C1C        MOV     DX,1C3C
0764:000C 68            DB      68
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:0008

AX=1010  BX=1111  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0008   NV UP EI PL NZ NA PO NC
0764:0008 F4            HLT
```

```
C:\>debug or.exe
-u
0764:0000 B81010        MOV     AX,1010
0764:0003 BB0101        MOV     BX,0101
0764:0006 0BC3          OR      AX,BX
0764:0008 F4            HLT
0764:0009 BA3C1C        MOV     DX,1C3C
0764:000C 68            DB      68
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:0008

AX=1111  BX=0101  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0008   NV UP EI PL NZ NA PE NC
0764:0008 F4            HLT
```

```
C:\>debug not.exe
-u
0764:0000 B81010        MOV     AX,1010
0764:0003 F7D0          NOT     AX
0764:0005 F4            HLT
0764:0006 041C          ADD     AL,1C
0764:0008 04BA          ADD     AL,BA
0764:000A 3C1C          CMP     AL,1C
0764:000C 68            DB      68
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:0005

AX=EFEF  BX=0000  CX=0006  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0005   NV UP EI PL NZ NA PO NC
0764:0005 F4            HLT
```

```
C:\>debug xor.exe
-u
0764:0000 B80201        MOV     AX,0102
0764:0003 BB0304        MOV     BX,0403
0764:0006 33C3          XOR     AX,BX
0764:0008 F4            HLT
0764:0009 BA3C1C        MOV     DX,1C3C
0764:000C 68            DB      68
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:0008

AX=0501  BX=0403  CX=0009  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0008   NV UP EI PL NZ NA PO NC
0764:0008 F4            HLT
```

```
C:\>debug shl.exe
-u
0764:0000 B80629        MOV     AX,2906
0764:0003 BB1001        MOV     BX,0110
0764:0006 B104          MOV     CL,04
0764:0008 D3E0          SHL     AX,CL
0764:000A D1E3          SHL     BX,1
0764:000C F4            HLT
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:000C

AX=9060  BX=0220  CX=0004  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=000C   NV UP EI PL NZ AC PO NC
0764:000C F4            HLT
```

```
C:\>debug shr.exe
-u
0764:0000 B89602        MOV     AX,0296
0764:0003 BB1001        MOV     BX,0110
0764:0006 B103          MOV     CL,03
0764:0008 D3E8          SHR     AX,CL
0764:000A D1EB          SHR     BX,1
0764:000C F4            HLT
0764:000D 014070        ADD     [BX+SI+70],AX
0764:0010 1CEB          SBB     AL,EB
0764:0012 2C04          SUB     AL,04
0764:0014 1C04          SBB     AL,04
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:000C

AX=0052  BX=0088  CX=0003  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=000C   NV UP EI PL NZ AC PE NC
0764:000C F4            HLT
```

```
C:\>debug bcdascii.exe
-u
0764:0000 B82900        MOV     AX,0029
0764:0003 8BD8          MOV     BX,AX
0764:0005 83E30F        AND     BX,+0F
0764:0008 83C330        ADD     BX,+30
0764:000B B104          MOV     CL,04
0764:000D D3C8          ROR     AX,CL
0764:000F 83E00F        AND     AX,+0F
0764:0012 83C030        ADD     AX,+30
0764:0015 F4            HLT
0764:0016 1C5D          SBB     AL,5D
0764:0018 9E            SAHF
0764:0019 7001          JO      001C
0764:001B 207B1C        AND     [BP+DI+1C],BH
0764:001E 75D6          JNZ     FFF6
-g 0764:0015

AX=0032  BX=0039  CX=0004  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0015   NV UP EI PL NZ NA PO NC
0764:0015 F4            HLT
```

```
C:\>debug dechex.exe
-u
0764:0000 B80629        MOV     AX,2906
0764:0003 83E00F        AND     AX,+0F
0764:0006 BA0100        MOV     DX,0001
0764:0009 F7E2          MUL     DX
0764:000B 8BD8          MOV     BX,AX
0764:000D B80629        MOV     AX,2906
0764:0010 B104          MOV     CL,04
0764:0012 D3C8          ROR     AX,CL
0764:0014 83E00F        AND     AX,+0F
0764:0017 BA0A00        MOV     DX,000A
0764:001A F7E2          MUL     DX
0764:001C 03C3          ADD     AX,BX
0764:001E 8BD8          MOV     BX,AX
-
```

```
-u
0764:0020 B80629        MOV     AX,2906
0764:0023 B108          MOV     CL,08
0764:0025 D3C8          ROR     AX,CL
0764:0027 83E00F        AND     AX,+0F
0764:002A BA6400        MOV     DX,0064
0764:002D F7E2          MUL     DX
0764:002F 03C3          ADD     AX,BX
0764:0031 8BD8          MOV     BX,AX
0764:0033 B80629        MOV     AX,2906
0764:0036 B10C          MOV     CL,0C
0764:0038 D3C8          ROR     AX,CL
0764:003A 83E00F        AND     AX,+0F
0764:003D BAE803        MOV     DX,03E8
```

```
-u
0764:0040 F7E2          MUL     DX
0764:0042 03C3          ADD     AX,BX
0764:0044 F4            HLT
0764:0045 CE            INTO
0764:0046 9A1C041C20    CALL    201C:041C
0764:004B E504          IN      AX,04
0764:004D 0C44          OR      AL,44
0764:004F 64            DB      64
0764:0050 207F20        AND     [BX+20],BH
0764:0053 B41C          MOV     AH,1C
0764:0055 59            POP     CX
0764:0056 67            DB      67
0764:0057 041C          ADD     AL,1C
0764:0059 044D          ADD     AL,4D
0764:005B 90            NOP
0764:005C 1C04          SBB     AL,04
0764:005E 104460        ADC     [SI+60],AL
```

```
-u
0764:0040 F7E2          MUL     DX
0764:0042 03C3          ADD     AX,BX
0764:0044 F4            HLT
0764:0045 CE            INTO
0764:0046 9A1C041C20    CALL    201C:041C
0764:004B E504          IN      AX,04
0764:004D 0C44          OR      AL,44
0764:004F 64            DB      64
0764:0050 207F20        AND     [BX+20],BH
0764:0053 B41C          MOV     AH,1C
0764:0055 59            POP     CX
0764:0056 67            DB      67
0764:0057 041C          ADD     AL,1C
0764:0059 044D          ADD     AL,4D
0764:005B 90            NOP
0764:005C 1C04          SBB     AL,04
0764:005E 104460        ADC     [SI+60],AL
-g 0764:0044

AX=0B5A  BX=038A  CX=000C  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0754  ES=0754  SS=0763  CS=0764  IP=0044   NV UP EI PL NZ NA PE NC
0764:0044 F4            HLT
```

Result: The logical operations, shifting and conversions were executed successfully