**Aim:**

To implement logical and shifting operations and write programs to covert a BCD number to ASCII and to covert a BCD number to HEX.

**Tool Used:**

Assembler - MASM 611

**AND OPERATOR**

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit and.asm

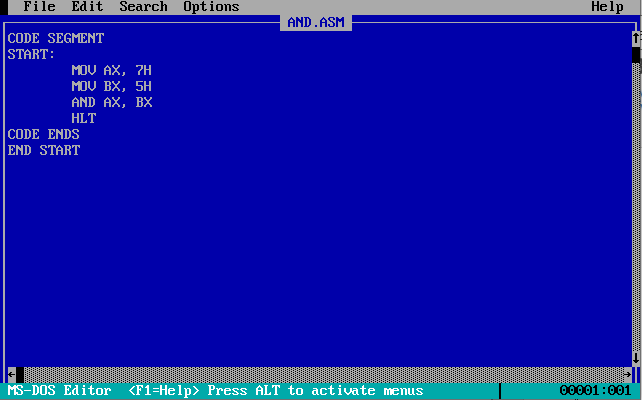
**STEP 3:** Type the program in the file and save as with .asm extension.

**STEP 4:** Move the values 07H in AX and 05H BX.

**STEP 5:** Apply AND operator between AX & BX and store the new value in AX.

**STEP 6:** To assemble and run follow this-

* masm and.asm
* link and.obj (then 4 enters continuously)
* debug and.exe
* -u
* -g 0764:0008



**Sample Input:**

AX =07H

BX = 05H

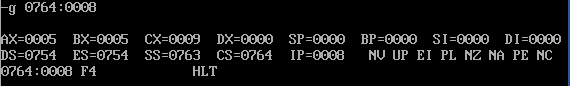
**Sample Output:**

AX = 0005

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



OR OPERATOR

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit or.asm

**STEP 3:** Type the program in the file and save as with .asm extension.

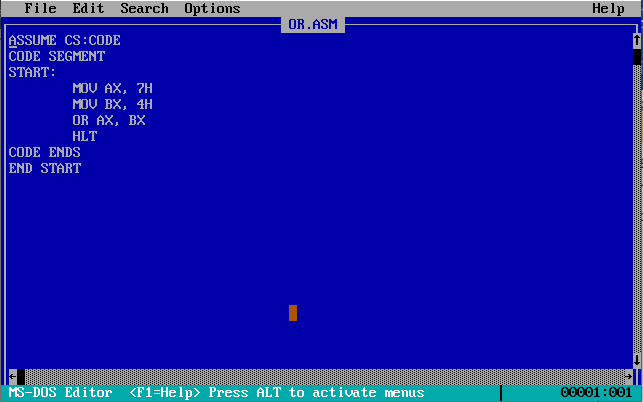
**STEP 4:** Move the values 07H in AX and 04H BX.

**STEP 5:** Apply OR operator between AX & BX and store the new value in AX.

**STEP 6:** To assemble and run follow this-

* masm or.asm
* link or.obj (then 4 enters continuously)
* debug or.exe
* -u
* -g 0764:0008

**Program:**



**Sample Input:**

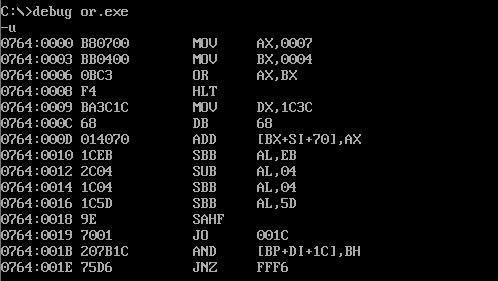
AX =07H

BX = 04H

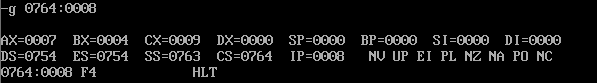
**Sample Output:**

AX = 0007

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



XOR OPERATOR

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit xor.asm

**STEP 3:** Type the program in the file and save as with .asm extension.

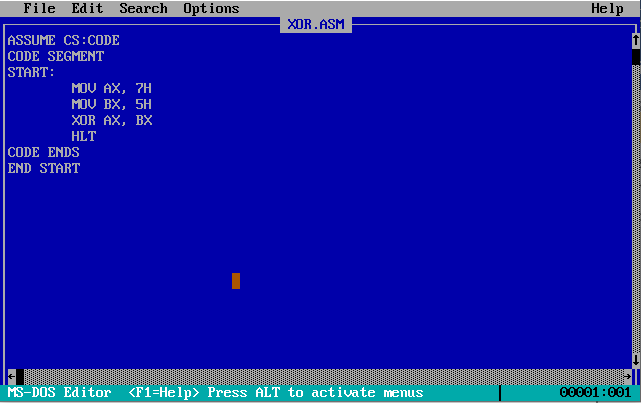
**STEP 4:** Move the values 07H in AX and 05H BX.

**STEP 5:** Apply XOR operator between AX & BX and store the new value in AX.

**STEP 6:** To assemble and run follow this-

* masm xor.asm
* link xor.obj (then 4 enters continuously)
* debug xor.exe
* -u
* -g 0764:0008

**Program:**



**Sample Input:**

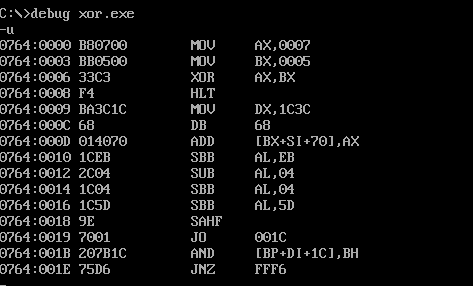
AX =07H

BX = 05H

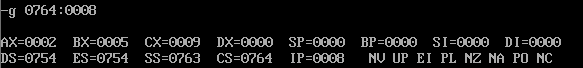
**Sample Output:**

AX = 0002

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



NOT OPERATOR

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit not.asm

**STEP 3:** Type the program in the file and save as with .asm extension.

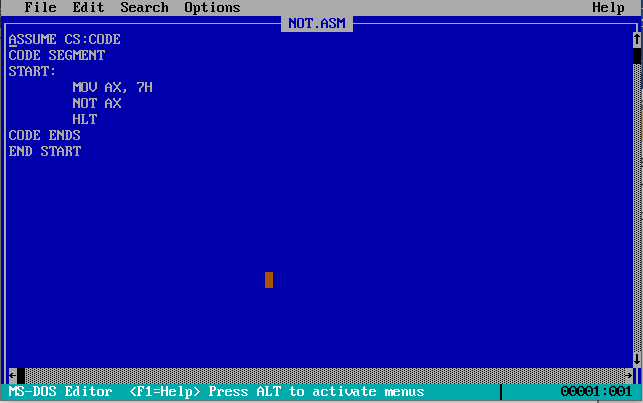
**STEP 4:** Move the value 07H in AX.

**STEP 5:** Apply NOT operator on AX and store the new value in AX.

**STEP 6:** To assemble and run follow this-

* masm not.asm
* link not.obj (then 4 enters continuously)
* debug not.exe
* -u
* -g 0764:0005

**Program:**



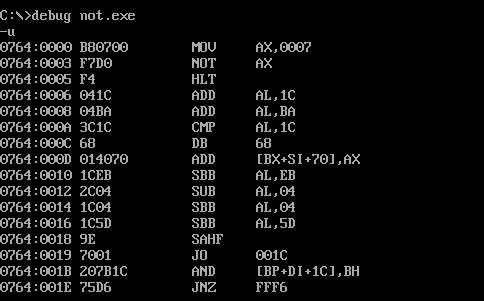
**Sample Input:**

AX =07H

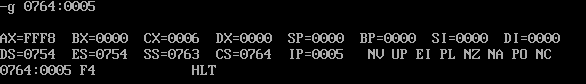
**Sample Output:**

AX = FFF8

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



SHR

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit shr.asm

**STEP 3:** Type the program in the file and save as with .asm extension.

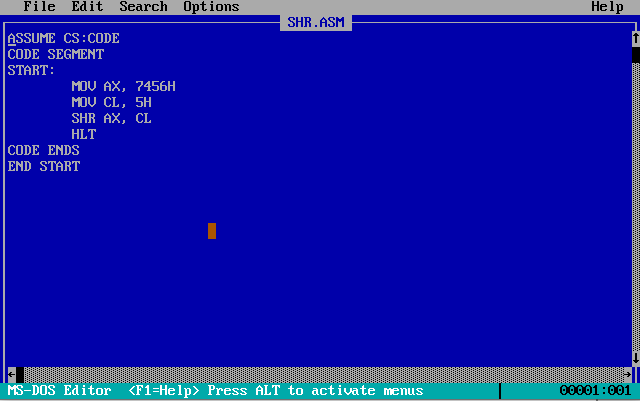
**STEP 4:** Move the values 7456H in AX and 05H in CL.

**STEP 2:** SHR AX by CL and then store the result in AX.

**STEP 6:** To assemble and run follow this-

* masm shr.asm
* link shr.obj (then 4 enters continuously)
* debug shr.exe
* -u
* -g 0764:0007

**Program:**



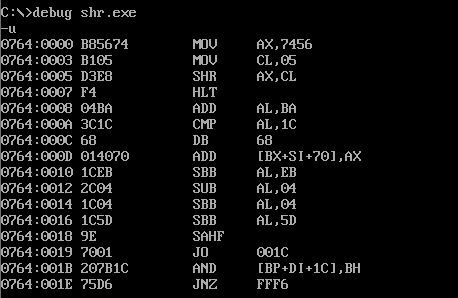
**Sample Input:**

AX =7456H

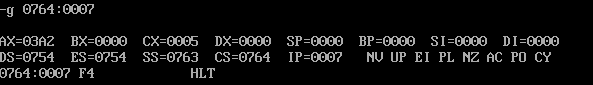
**Sample Output:**

AX = 03A2

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



SHL

**Algorithm:**

**STEP 1:** Open DOSBOX.

**STEP 2:** Type the following in the command prompt:

* mount c c:\masm611\bin.
* c:
* edit shl.asm

**STEP 3:** Type the program in the file and save as with .asm extension.

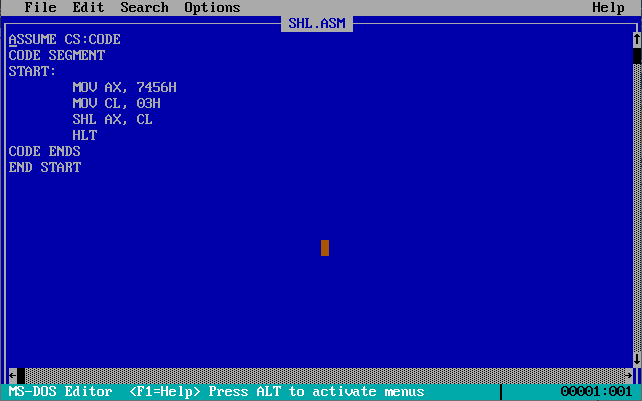
**STEP 4:** Move the values 7456H in AX and 03H in CL.

**STEP 2:** SHL AX by CL and then store the result in AX.

**STEP 6:** To assemble and run follow this-

* masm shl.asm
* link shl.obj (then 4 enters continuously)
* debug shl.exe
* -u
* -g 0764:0007

**Program:**



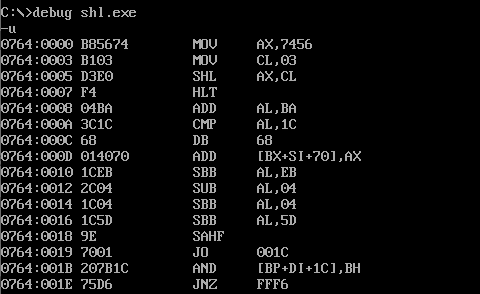
**Sample Input:**

AX =7456H

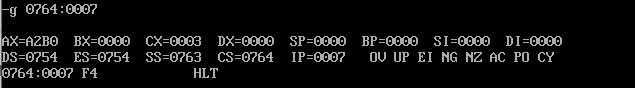
**Sample Output:**

AX = A2B0

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



BCD TO ASCII

**Algorithm:**

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

ASSUME CS:CODE

CODE SEGMENT

  START:

        MOV AX,75H

        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

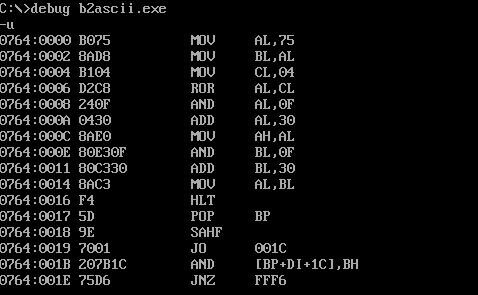
**Sample Input:**

AX =75H

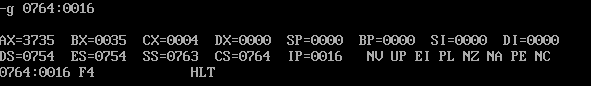
**Sample Output:**

AX = 2625

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



BCD TO HEX

**Algorithm:**

Move 1772H to AX register, and perform AND operation with AX and 0FH.

Move 1H to DX, and multiply AX with DX.

Move the result stored in AX to BX. Move 1772H to AX again, and 4H to CL.

For as many counts mentioned in CL, and perform AND operation with AX and 0FH.

Move 0AH to DX, and multiply DX with AX. Add the result stored in AX to BX.

Move 1772H to AX again, and 8H to CL. Perform ROR on AX for as many counts mentioned in CL, and perform AND operation with AX and 0FH.

Move 64H to DX, and multiply DX with AX.

Add the result stored in AX to BX. Move 1772H to AX again, and 0CH to CL.

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

Move 03E8H to DX, and multiply DX with AX. Add the result stored in AX to BX. HLT brings the process to a halt.

DATA SEGMENT

    X    DW 17H

    B    DW 0AH

DATA ENDS

CODE SEGMENT

          ASSUME CS:CODE, DS:DATA

    START:

          MOV    AX,DATA

          MOV    DS,AX

          MOV    AX,X

          AND    AX,0FH

          ADD    BX,AX

    LOOP1:

          MOV    CL,04H

          SHR    X,CL

          MOV    AX,X

          AND    AX,0FH

          MUL    B

          ADD    BX,AX

          MOV    AX,B

          MOV    B,0AH

          MUL    B

          MOV    B,AX

          CMP    X,0000H

          JNZ    LOOP1

          HLT

CODE ENDS

END START

**Sample Input:**

AX = 17H

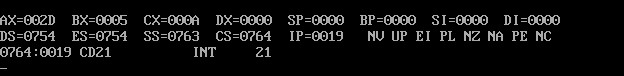
**Sample Output:**

AX = 002D

**Register/ Memory Contents for I/O:**



**Snapshot of the Output:**



**Result:**

All logical and shift operation are successfully implemented

BCD to ASCII conversion is successfully executed

BCD to HEX conversion is successfully executed