

# **Experiment 1:**

Arithmetic Operations (Addition, Subtraction, Multiplication, Division) on 8-Bit and 16-Bit Numbers

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

Exp. 01

**Arithmetic Operations** 



# **Addition (8 bit)**

Aim: To Perform Addition of 8-Bit Numbers

# **Algorithm:**

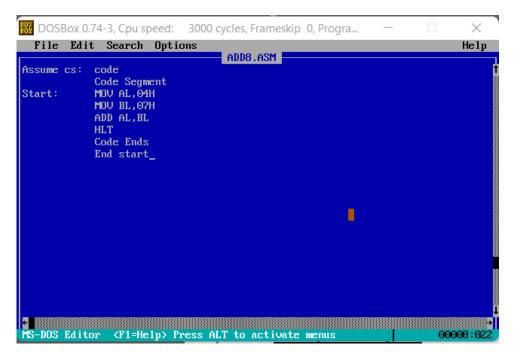
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- **Step 3:** Enter the command **edit add8.asm** for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Addition of 8-Bit Numbers.
  - i) Start.
  - ii) Let CS is Code and DS is Data.
  - iii) Code Segment starts.
  - iv) Move Data to AL.
  - v) Move Data to BL.
  - vi) Add Data of AL and BL.
  - vii) Code Segment ends.
  - viii) End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm add8.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link add8.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug add8.exe** to execute the code.

-u

**-g 0764:0006** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after addition of values in AL and BL registers.

## **Program:**



Sample Input:	Sample Output:
AL= 04H	AL=AL+BL
BL= 07H	AL=04H+07H=0BH
	Hence, AX=FF <mark>0B</mark>

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

```
C:\>debug add8.exe
-u
                                    AL,04
0764:0000 B004
                           MOV
0764:0002 B307
                                    BL,07
                           MOV
0764:0004 0203
                           ADD
                                    AL,BL
0764:0006 F4
                           HLT
0764:0007 1C04
0764:0009 BA3C1C
                                    AL,04
                           SBB
                                    DX,1C3C
                           MOV
0764:000C 68
                           DB
                                    68
0764:000D 014070
                                    [BX+SI+70],AX
                           ADD
0764:0010 1CEB
                                    AL,EB
                           SBB
0764:0012 2004
                           SUB
                                    AL,04
                           SBB
                                    AL,04
0764:0014 1004
0764:0016 1C5D
0764:0018 9E
                           SBB
                                    AL,5D
                           SAHF
0764:0019 7001
                           JO
                                    0010
0764:001B 207B1C
0764:001E 75D6
                                    [BP+DI+1C],BH
                           AND
                           JNZ
                                    FFF6
```

```
-g 0764:0006

AX=FF0B BX=0007 CX=0007 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000

DS=0754 ES=0754 SS=0763 CS=0764 IP=0006 NV UP EI PL NZ NA PO NC

0764:0006 F4 HLT
```

Exp. 01

**Arithmetic Operations** 



Register No.: 20BCE1975

# **Addition (16 bit)**

Aim: To Perform Addition of 16-Bit Numbers

## Algorithm:

Date: 12-01-2022

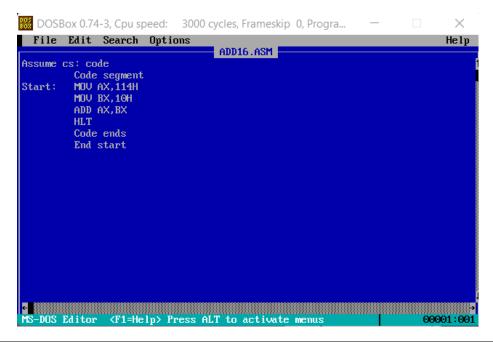
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- **Step 3:** Enter the command **edit add16.asm** for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Addition of 16-Bit Numbers.
  - i) Start.
  - ii) Let CS is Code and DS is Data.
  - iii) Code Segment starts.
  - Move Data to AX. iv)
  - Move Data to BX. V)
  - Add Data of AX and BX. vi)
  - Code Segment ends. vii)
  - viii) End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm add16.asm** for running the code. The object file is created.
- Step 6: Next, Type the command link add16.obj; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug add16.exe** to execute the code.

-u

-g 0764:0008 to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after addition of values in AX and BX registers.

### **Program:**



Sample Input:	Sample Output:
AX = 0114H	AX=AX+BX
BX = 0010H	AX=0114H+0010H=0124H
	Hence, AX=0124

## Register / Memory Contents for I/O:

```
C:\>debug add16.exe
-u
0764:0000 B81401
                            MOV
                                      AX,0114
0764:0003 BB1000
                            MOV
                                      BX,0010
0764:0006 0303
                            ADD
                                      AX,BX
0764:0008 F4
0764:0009 BA3C1C
                            HLT
                            MOV
                                      DX,1C3C
0764:000C 68
                            DB
                                      68
                                      [BX+SI+70],AX
AL,EB
0764:000D 014070
0764:0010 1CEB
                            ADD
                            SBB
0764:0012 2004
                            SUB
                                      AL,04
0764:0014 1004
                            SBB
                                      AL,04
0764:0016 1C5D
0764:0018 9E
                            SBB
                                      AL,5D
                            SAHF
0764:0019 7001
                            JO
                                      001C
0764:001B 207B1C
0764:001E 75D6
                            AND
                                      [BP+DI+1C],BH
                             JNZ
                                      FFF6
```

```
-g 0764:0008
AX=0124 BX=0010 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
```

Exp. 01

**Arithmetic Operations** 



# **Subtraction (8 bit)**

Aim: To Perform Subtraction of 8-Bit Numbers

# **Algorithm:**

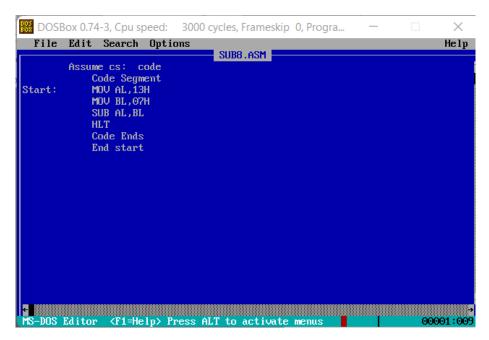
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- Step 3: Enter the command edit sub8.asm for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Subtraction of 8-Bit Numbers.
  - a. Start.
  - b. Let CS is Code and DS is Data.
  - c. Code Segment starts.
  - d. Move Data to AL.
  - e. Move Data to BL.
  - f. Subtract Data of AL and BL.
  - g. Code Segment ends.
  - h. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm sub8.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link sub8.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug sub8.exe** to execute the code.

-u

**-g 0764:0006** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

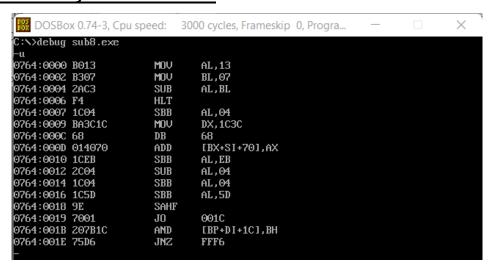
We check here the value stored in **AX register** which is the result obtained after subtraction of values in AL and BL registers.

## **Program:**



Sample Input:	Sample Output:
AL= 13H = 19 <sub>2</sub>	AL=AL-BL
$BL = 07H = 7_2$	$AL=13H-07H=19_2-7_2=12_2=0CH$
	Hence, AX=FF <mark>0C</mark>

# Register / Memory Contents for I/O:



```
-g 0764:0006

AX=FF0C BX=0007 CX=0007 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0754 ES=0754 SS=0763 CS=0764 IP=0006 NV UP EI PL NZ AC PE NC
0764:0006 F4 HLT
```

Exp. 01

**Arithmetic Operations** 



Register No.: 20BCE1975

# **Subtraction (16 bit)**

Aim: To Perform Subtraction of 16-Bit Numbers

## **Algorithm:**

Date: 12-01-2022

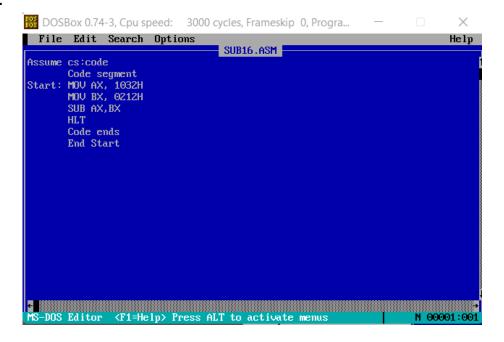
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- **Step 3:** Enter the command **edit sub16.asm** for writing/editing the code.
- Step 4: A pop window appears, write your code(instructions) there for Subtraction of 16-Bit Numbers.
  - a. Start.
  - b. Let CS is Code and DS is Data.
  - c. Code Segment starts.
  - d. Move Data to AX.
  - e. Move Data to BX.
  - f. Subtract Data of AX and BX.
  - g. Code Segment ends.
  - h. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm sub16.asm** for running the code. The object file is created.
- Step 6: Next, Type the command link sub16.obj; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug sub16.exe** to execute the code.

-u

-g 0764:0008 to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after subtraction of values in AX and BX registers.

### Program:



Sample Input:	Sample Output:
AX = 1032H = 4146 <sub>2</sub> BX = 0212H = 530 <sub>2</sub>	AX=AX-BX AX=1032H-0212H= 4146 <sub>2</sub> - 530 <sub>2</sub> = 3616 <sub>2</sub> = 0E20H
	Hence, AX=0E20

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

```
C:N>debug_sub16.exe
0764:0000 B83210
                                      AX,1032
BX,0212
                             MOV
0764:0003 BB1202
                             MOV
0764:0006 ZBC3
                             SUB
                                      AX,BX
0764:0008 F4
0764:0009 BA3C1C
                             HLT
                             MOV
                                      DX,1C3C
0764:000C 68
                             DB
                                      68
0764:000D 014070
                                       [BX+SI+70],AX
                             ADD
0764:0010 1CEB
                             SBB
                                      AL,EB
0764:0012 2004
                             SUB
                                      AL,04
0764:0014 1004
                             SBB
                                      AL,04
0764:0016 1C5D
0764:0018 9E
0764:0019 7001
                                      AL,5D
                             SBB
                             SAHF
                             JO
                                       0010
0764:001B 207B1C
0764:001E 75D6
                                       [BP+DI+1C],BH
                             AND
                             JNZ
                                      FFF6
```

```
-g 0764:0008

AX=0E20 BX=0Z12 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
```

Exp. 01

**Arithmetic Operations** 



# **Multiplication (8 bit)**

Aim: To Perform Multiplication of 8-Bit Numbers

# **Algorithm:**

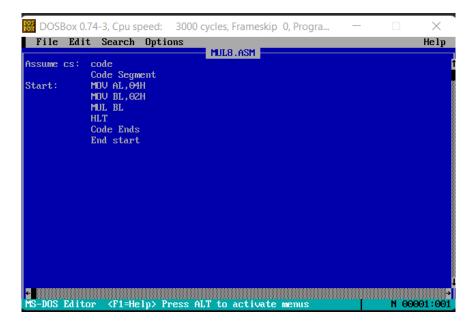
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- Step 3: Enter the command edit mul8.asm for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Multiplication of 8-Bit Numbers.
  - i. Start.
  - j. Let CS is Code and DS is Data.
  - k. Code Segment starts.
  - I. Move Data to AL.
  - m. Move Data to BL.
  - n. Multiply Data of AL and BL.
  - o. Code Segment ends.
  - p. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm mul8.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link mul8.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug mul8.exe** to execute the code.

-u

**-g 0764:0006** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after multiplication of values in AL and BL registers.

## **Program:**



Sample Input:	Sample Output:
AL= 04H	AL=AL*BL
BL= 02H	AL=04H * 02H = 08H
	Hence, AX=00 <mark>08</mark>

## Register / Memory Contents for I/O:

```
C:\>debug mul8.exe
-u
0764:0000 B004
                           MOV
                                    AL,04
0764:0002 B302
0764:0004 F6E3
                                    BL,02
                           MOV
                           MUL
                                    BL
0764:0006 F4
                           HLT
0764:0007 1004
                                    AL,04
                           SBB
0764:0009 BA3C1C
                           MOV
                                    DX,1C3C
0764:000C 68
                           DB
                                    68
0764:000D 014070
                           ADD
                                    [BX+SI+70],AX
0764:0010 1CEB
0764:0012 2C04
                                    AL,EB
                           SBB
                           SUB
                                    AL,04
0764:0014 1004
                           SBB
                                    AL,04
                                    AL,5D
0764:0016 1C5D
                           SBB
0764:0018 9E
0764:0019 7001
                           SAHF
                                    001C
                           JO
0764:001B 207B1C
                           AND
                                    [BP+DI+1C],BH
0764:001E 75D6
                           JNZ
                                    FFF6
```

```
- g 0764:0006
AX=0008 BX=000Z CX=0007 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0754 ES=0754 SS=0763 CS=0764 IP=0006 NV UP EI PL NZ NA PD NC
0764:0006 F4 HLT
```

Exp. 01

**Arithmetic Operations** 



# **Multiplication (16 bit)**

Aim: To Perform Multiplication of 16-Bit Numbers

## **Algorithm:**

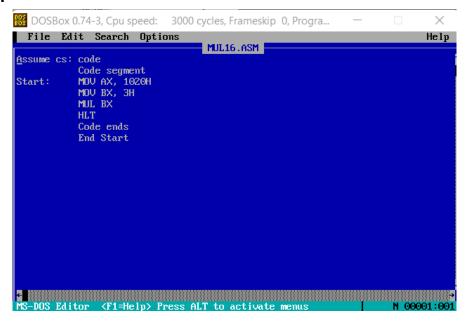
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- Step 2: Press Enter, Type c: to switch from z: to c: drive.
- Step 3: Enter the command edit mul16.asm for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Multiplication of 16-Bit Numbers.
  - i. Start.
  - j. Let CS is Code and DS is Data.
  - k. Code Segment starts.
  - I. Move Data to AX.
  - m. Move Data to BX.
  - n. Multiply Data of AX and BX.
  - o. Code Segment ends.
  - p. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm mul16.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link mul16.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug mul16.exe** to execute the code.

-u

**-g 0764:0008** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after multiplication of values in AX and BX registers.

## **Program:**



Sample Input:	Sample Output:
AX = 1020H	AX=AX*BX
BX = 0003H	AX=1020*3 = 3060H
	Hence, AX=3060

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

```
C:∖>debug mul16.exe
-u
0764:0000 B82010
                          MOV
                                   AX,1020
0764:0003 BB0300
                                   BX,0003
                          MOV
                                   BX
0764:0006 F7E3
                          MUL
0764:0008 F4
                          HLT
0764:0009 BA3C1C
                                   DX,1C3C
                          MOV
0764:000C 68
                          DB
                                   68
0764:000D 014070
                          ADD
                                   [BX+SI+70],AX
0764:0010 1CEB
                          SBB
                                   AL,EB
0764:0012 2C04
0764:0014 1C04
                                   AL,04
                          SUB
                          SBB
                                   AL,04
0764:0016 1C5D
                          SBB
                                   AL,5D
0764:0018 9E
                          SAHF
0764:0019 7001
0764:001B 207B1C
                          JO
                                   001C
                          AND
                                   [BP+DI+1C],BH
0764:001E 75D6
                          JNZ
                                   FFF6
```

```
-g 0764:0008
AX=3060 BX=0003 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
```

Exp. 01

**Arithmetic Operations** 



# **Division (8 bit)**

Aim: To Perform Division of 8-Bit Numbers

# **Algorithm:**

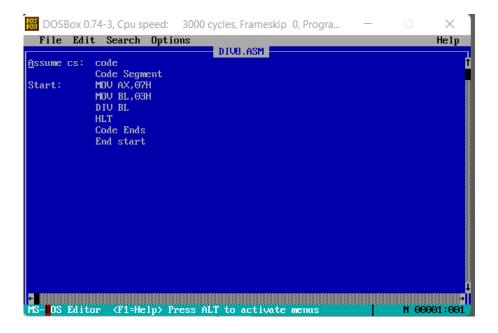
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- Step 3: Enter the command edit div8.asm for writing/editing the code.
- Step 4: A pop window appears, write your code(instructions) there for Division of 8-Bit Numbers.
  - q. Start.
  - r. Let CS is Code and DS is Data.
  - s. Code Segment starts.
  - t. Move Data to AL.(00 to AH)
  - u. Move Data to BL.
  - v. Divide Data of AX and BL.
  - w. Code Segment ends.
  - x. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm div8.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link div8.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug div8.exe** to execute the code.

-u

**-g 0764:0007** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after division of values in AL and BL registers.

## **Program:**



Sample Input:	Sample Output:
AX= 07H	AX = AX/BL
BL= 03H	Remainder(AH) = 01; Quotient(AL) = 02
	Hence, AX=0102

## Register / Memory Contents for I/O:

```
C:∖>debug div8.exe
-u
0764:0000 B80700
                         MOV
                                  AX,0007
0764:0003 B303
                         MOV
                                  BL,03
0764:0005 F6F3
                         DIV
                                  BL
0764:0007 F4
                         HLT
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 2004
                         SUB
                                  AL,04
0764:0014 1004
                                  AL,04
                         SBB
                         SBB
                                  AL,5D
0764:0016 1C5D
0764:0018 9E
0764:0019 7001
                         SAHF
                                  001C
                         JO
0764:001B 207B1C
                         AND
                                  [BP+DI+1C],BH
0764:001E 75D6
                         JNZ
                                  FFF6
```

```
-g 0764:0007
AX=0102 BX=0003 CX=0008 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0754 ES=0754 SS=0763 CS=0764 IP=0007 NV UP EI PL NZ NA PO NC
0764:0007 F4 HLT
```

Exp. 01

**Arithmetic Operations** 



# **Division (16 bit)**

Aim: To Perform Division of 16-Bit Numbers

## **Algorithm:**

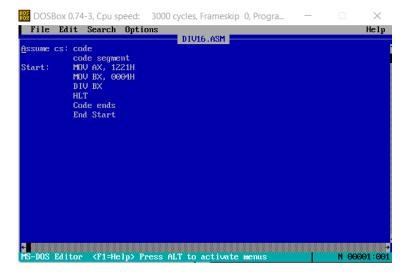
- Step 1: Mount the c drive using the command: mount c c:\masm611\bin
- **Step 2:** Press **Enter**, Type **c:** to switch from z: to c: drive.
- Step 3: Enter the command edit div16.asm for writing/editing the code.
- **Step 4:** A pop window appears, write your code(instructions) there for Division of 16-Bit Numbers.
  - q. Start.
  - r. Let CS is Code and DS is Data.
  - s. Code Segment starts.
  - t. Move Data to AX.
  - u. Move Data to BX.
  - v. Divide Data of AX and BX.
  - w. Code Segment ends.
  - x. End.
- **Step 5:** Save the Code, and exit the editor. Type the command **masm div16.asm** for running the code. The object file is created.
- **Step 6:** Next, Type the command **link div16.obj**; to link the object file to library file present in the bin folder.
- **Step 7:** Type **debug div16.exe** to execute the code.

-u

**-g 0764:0008** to view the result in all the registers specifically at that HLT Position. (-g <Address of HLT Command>)

We check here the value stored in **AX register** which is the result obtained after division of values in AX and BX registers.

### **Program:**



Sample Input:	Sample Output:
AX = 1221H = 4641 <sub>2</sub>	AX=AX/BX
$BX = 0004H = 4_2$	$AX=4641/4 = 1160_2 = 0488H$
	Remainder: $DX = 0001H$
	Hence, AX=0488H; DX=0001H

## Register / Memory Contents for I/O:

```
C:N>debug div16.exe
-u
0764:0000 B82112
                         MOV
                                  AX,1221
0764:0003 BB0400
                         MOV
                                  BX,0004
0764:0006 F7F3
0764:0008 F4
                         DIV
                                  BX
                         HLT
0764:0009 BA3C1C
                         MOV
                                  DX,1C3C
0764:000C 68
                                  68
                         DB
0764:000D 014070
                         ADD
                                  [BX+SI+70],AX
0764:0010 1CEB
                         SBB
                                  AL,EB
0764:0012 2004
                         SUB
                                  AL,04
                                  AL,04
0764:0014 1004
                         SBB
                         SBB
0764:0016 1C5D
                                  AL,5D
0764:0018 9E
                         SAHF
                                  001C
0764:0019 7001
                         JO
                                  [BP+DI+1C],BH
0764:001B 207B1C
                         AND
0764:001E 75D6
                                  FFF6
                         JNZ
```

## **Output:**

```
-g 0764:0008
AX=0488 BX=0004 CX=0009 DX=0001 SP=0000 BP=0000 SI=0000 DI=0000
DS=0754 ES=0754 SS=0763 CS=0764 IP=0008 NU UP EI PL NZ NA PO NC
0764:0008 F4 HLT
```

**Result:** The arithmetic operations are performed in accordance with the calculated values. The assembly code functions as expected