16 Bit Arithmetic Operations

Expt No:	2	**Name:**	Mahesh Bharadwaj K
Date :	27/08/2020	**Reg No:**	185001089

Aim:

To write and execute 8086 programs for 16 bit arithmetic operations like addition, subtraction, multiplication and division.

Procedure:

- Mount masm folder to a drive on DOSBOX.
- Navigate to mounted drive using 'dir' .
- Save 8086 program with the extension **'.asm'** in the same folder using the command **'edit'**.
- Assemble the **.asm** file using the command **'masm filename.asm'**.
- Link the assmebled **.obj** file using the command **'link filename.obj'**.
- Debug the executable file **.exe** with the **'debug filename.exe'** command.
    i. **U:** To view the un-assembled code.
    ii. **D:** Used as 'D segment:offset' to see the content of memory locations starting from segment:offset address.
    iii. **E:** To change the values in memory.
    iv. **G:** Execute the program using command.
    v. **Q** exits from the debug session.

Algorithm:

1. Addition

* Move the data segment address to the AX register and then move it to the DS register.
* Move the first operand to AX register.
* Move the second operand to the BX register.
* Initially set the CH register to 00h.
* Then add using ADD AH,BH.
* Using JNC instruction check for carry and if there is no carry, no need to increment CH.

* Else, increment CH by 1.
* The result and carry stored in AX and CH should be moved to RESULT and CARRY respectively.

2. Subtraction

* Move the data segment address to the AX register and then move it to the DS register.
* Move the first operand to AX register.
* Move the second operand to the BX register.
* Initially set the CH register to 00h.
* Then subtract using SUB AH,BH.
* Check for carry using JNC instruction. If no carry then it means AX > BX and hence no need to increment CH and no need to complement AH.
* Else, AX<BX. Hence we have to take 2's complement of AX using NEG AX and also increment CH by 1 using INC CH.
* The result and sign stored in AX and CH should be moved to RESULT and CARRY respectively.

3. Multiplication

* Move the data segment address to the AX register and then move it to the DS register.
* Move the first operand to AX register.
* Move the second operand to the BX register.
* Then multiply using MUL BX.(Since AX is default operand register for MUL instruction we only need to specify the other operand register.)
* The result stored in DX AX register(32 bit- because multiplication of two 16 bit numbers yields a 32 bit number) should now be moved to RESULT_H & RESULT_L respectively.

4. Division

* Move the data segment address address to the AX register and then move it to the DS register.
* Set value of DX register to 0000H.(No dedicated 16bit by 16bit instruction)
* Move first operand to AX register.
* Move the second operand to the BX register.
* Now divide using DIV BX (Performs DX AX / BX).
* The quotient and remainder stored in AX and DX should be moved to QUOTIENT and REMAINDER respectively

16 bit Addition

Program:

Program	Comments
start: mov ax,data	Move data segment address contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
mov ch,00h	Move hex value 00 to CH register
add ax,bx	AX = AX + BX
jnc here	Jump to the label here, if there is no carry
inc ch	Increment value of CH if there is a carry
here: mov result,ax	Move contents of AX register to result
mov carry,ch	Move contents of CH register to carry
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 16BITADD.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C B500          MOV     CH,00
076B:010E 02E7          ADD     AH,BH
076B:0110 7302          JNB     0114
076B:0112 FEC5          INC     CH
076B:0114 A30400        MOV     [0004],AX
076B:0117 882E0600      MOV     [0006],CH
076B:011B B44C          MOV     AH,4C
076B:011D CD21          INT     21
```

Input and Output:

Figure 1: **Input:** opr1 - 0110h, opr2 - 0990h

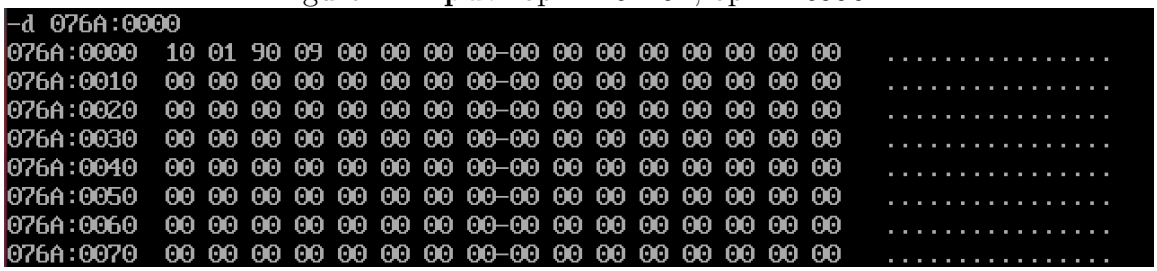


Figure 2: **Output:** result - 0AA0h, carry - 00h

16 bit Subtraction

Program:

Program	**Comments**
start: mov ax,data	Move data segment address contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
mov ch,00h	Move hex value 00 to CH register
sub ax,bx	AX = AX - BX
jnc here	Jump to the label here, if there is no carry
inc ch	Increment value of CH if there is a carry
neg ax	Negate the contents of the AH register
here: mov result,ax	Move contents of AX register to result
mov sign,ch	Move contents of CH register to sign
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 16BITSUB.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C B500          MOV     CH,00
076B:010E 2BC3          SUB     AX,BX
076B:0110 7304          JNB     0116
076B:0112 FEC5          INC     CH
076B:0114 F7D8          NEG     AX
076B:0116 A30400        MOV     [0004],AX
076B:0119 882E0600      MOV     [0006],CH
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
```

Input and Output:

Figure 3: **Input:** opr1 - 0110h, opr2 - 0990h

```
-d 076A:0000
076A:0000  10 01 90 09 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Figure 4: **Output:** result - 0880h, sign - 01h

```
-g

Program terminated normally
-d 076A:0000
076A:0000  10 01 90 09 80 08 01 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

16 bit Multiplication

Program:

Program	Comments
start: mov ax,data	Move data segment address contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
mul bx	DX AX = AX * BX
mov result_h, dx	Move contents of DX register to result_h
mov result_l, ax	Move contents of AX register to result_l
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 16BITMUL.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C F7E3          MUL     BX
076B:010E 89160400      MOV     [0004],DX
076B:0112 A30600        MOV     [0006],AX
076B:0115 B44C          MOV     AH,4C
076B:0117 CD21          INT     21
076B:0119 4C            DEC     SP
076B:011A CD21          INT     21
```

Input and Output:

Figure 5: **Input:** opr1 - 0130h, opr2 - 0030h

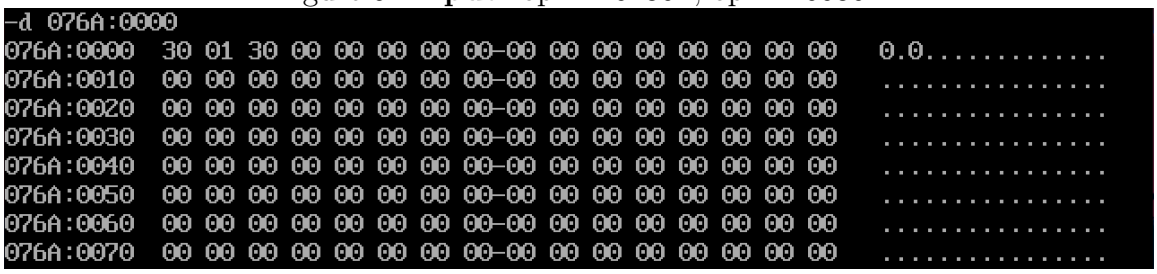
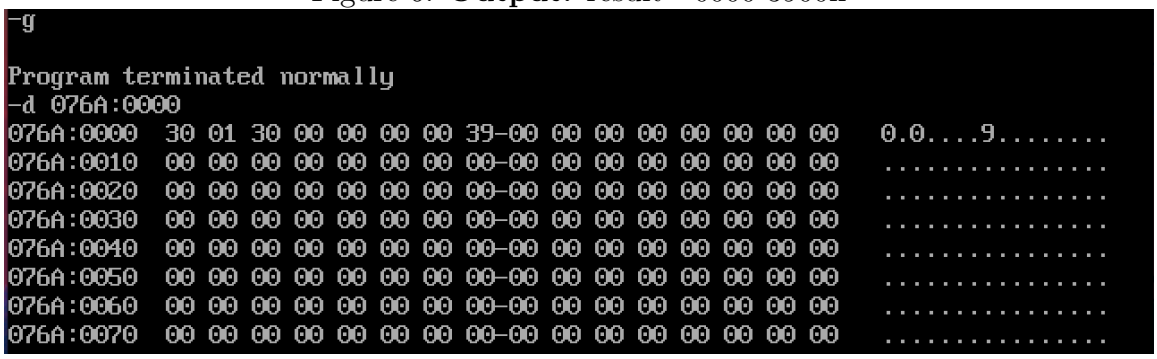


Figure 6: **Output:** result - 0000 3900h



16 bit Division

Program:

Program	**Comments**
start: mov ax,data	Move data segment address contents to AX register
mov ds,ax	Move data in AX register to DS register
mov dx, 0000h	Set value of DX to 0000h
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
div bX	DX AX = DX AX / BX
mov quotient,ax	Move contents of AX register to quotient
mov remainder,dx	Move contents of DX register to remainder
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 16BITDIV.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 BA0000        MOV     DX,0000
076B:0108 A10000        MOV     AX,[0000]
076B:010B 8B1E0200      MOV     BX,[0002]
076B:010F F7F3          DIV     BX
076B:0111 A30400        MOV     [0004],AX
076B:0114 89160600      MOV     [0006],DX
076B:0118 B44C          MOV     AH,4C
076B:011A CD21          INT     21
```

Input and Output:

Figure 7: **Input:** opr1 - 0040h, opr2 - 0020h

```
-d 076A:0000
076A:0000  40 00 20 00 00 00 00 00-00 00 00 00 00 00 00 00   @. ............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Figure 8: **Input:** Quotient - 0002h, Remainder - 0000h

```
-g

Program terminated normally
-d 076A:0000
076A:0000  40 00 20 00 02 00 00 00-00 00 00 00 00 00 00 00   @. ............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Result:

The 8086 programs were written to perform 16-bit arithmetic operations, and the results observed.