8 Bit Arithmetic Operations

Expt No:	1	**Name:**	Mahesh Bharadwaj K
Date :	20/08/2020	**Reg No:**	185001089

Aim:

To write and execute 8086 programs for 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 to the AX register and then move it to the DS register.
* Move the first operand to AH register.
* Move the second operand to the BH 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 AH and CH should be moved to RESULT and CARRY respectively.

2. Subtraction

* Move the data segment to the AX register and then move it to the DS register.
* Move the first operand to AH register.
* Move the second operand to the BH 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 AH > BH and hence no need to increment CH and no need to complement AH.
* Else, AH<BH. Hence we have to take 2's complement of AH using NEG AH and also increment CH by 1 using INC CH.
* The result and carry stored in AH and CH should be moved to RESULT and CARRY respectively.

3. Multiplication

* Move the data segment to the AX register and then move it to the DS register.
* Move the first operand to AL register.
* Move the second operand to the BL register.
* Then multiply using MUL BL.(Since AL is default operand register for MUL instruction we only need to specify the other operand register.)
* The result stored in AX register(16 bit- because multiplication of two 8 bit numbers yields a 16 bit number) should now be moved to RESULT.

4. Division

* Move the data segment to the AX register and then move it to the DS register.
* Now, set AH register to 00h and move first operand to AL register. (Since we can't directly divide a 8 bit number by 8 bit number in 8086, we now make our dividend 16 bit by storing 00h in AH register and the 8-bit operand 1 in AL register).
* Move the second operand to the BL register.
* Now divide using DIV BL.(It will perform AX / BL. Because AH is 00h, what actually happens is the division of a 16 bit number by a 8 bit number).
* The quotient and remainder stored in AL and AH should be moved to QUOTIENT and REMAINDER respectively

8 Bit Addition

Program:

Program	Comments
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ah,opr1	Move contents of opr1 to AH register
mov bh,opr2	Move contents of opr2 to BH register
mov ch,00h	Move hex value 00 to CH register
add ah,bh	AH = AH + BH
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,ah	Move contents of AH 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 8BITADD.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 02E7          ADD     AH,BH
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 88260200      MOV     [0002],AH
076B:0119 882E0300      MOV     [0003],CH
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
```

Input and Output:

Figure 1: **Input:** opr1 - 11h, opr2 - 99h

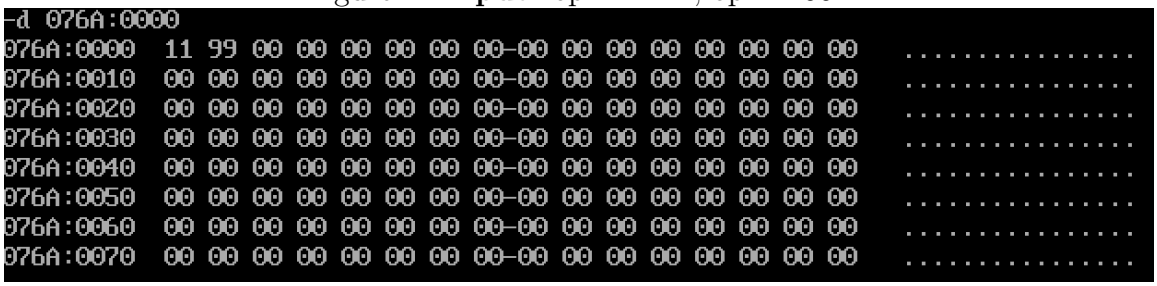
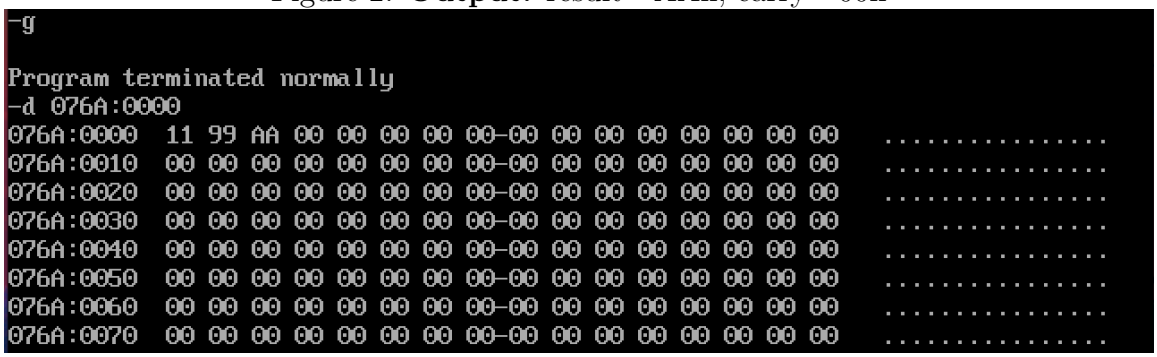


Figure 2: **Output:** result - AAh, carry - 00h



8 Bit Subtraction

Program:

Program	**Comments**
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ah,opr1	Move contents of opr1 to AH register
mov bh,opr2	Move contents of opr2 to BH register
mov ch,00h	Move hex value 00 to CH register
sub ah,bh	AH = AH - BH
jnc here	Jump to the label here, if there is no carry
inc ch	Increment value of CH if there is a carry
neg ah	Negate the contents of the AH register
here: mov result,ah	Move contents of AH 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 8BITSUB.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 2AE7          SUB     AH,BH
076B:0111 7304          JNB     0117
076B:0113 FEC5          INC     CH
076B:0115 F6DC          NEG     AH
076B:0117 88260200      MOV     [0002],AH
076B:011B 882E0300      MOV     [0003],CH
076B:011F B44C          MOV     AH,4C
```

Input and Output:

Figure 3: **Input:** opr1 - 99h, opr2 - 11h

```
-d 076A:0000
076A:0000  99 11 00 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 4: **Output:** result - 88h, sign - 00h

```
-g

Program terminated normally
-d 076A:0000
076A:0000  99 11 88 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   ................
```

8 Bit Multiplication

Program:

Program	Comments
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov al,opr1	Move contents of opr1 to AL register
mov ah, 00H	Move hex value 00 to AH register
mov bl,opr2	Move contents of opr2 to BL register
mul bl	AX = AL * BL
mov result,ax	Move contents of AX register to result
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 8BITMUL.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00000        MOV     AL,[0000]
076B:0108 8A1E0100      MOV     BL,[0001]
076B:010C F6E3          MUL     BL
076B:010E A30200        MOV     [0002],AX
076B:0111 B44C          MOV     AH,4C
076B:0113 CD21          INT     21
076B:0115 F6DC          NEG     AH
076B:0117 88260200      MOV     [0002],AH
076B:011B 882E0300      MOV     [0003],CH
076B:011F B44C          MOV     AH,4C
```

Input and Output:

Figure 5: **Input:** opr1 - 02h, opr2 - 13h

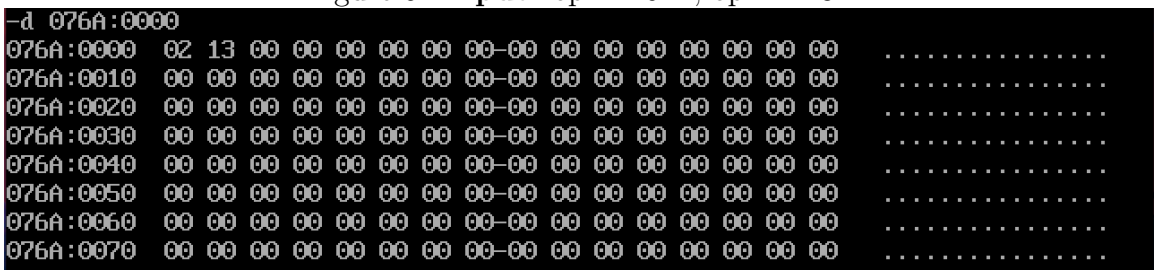
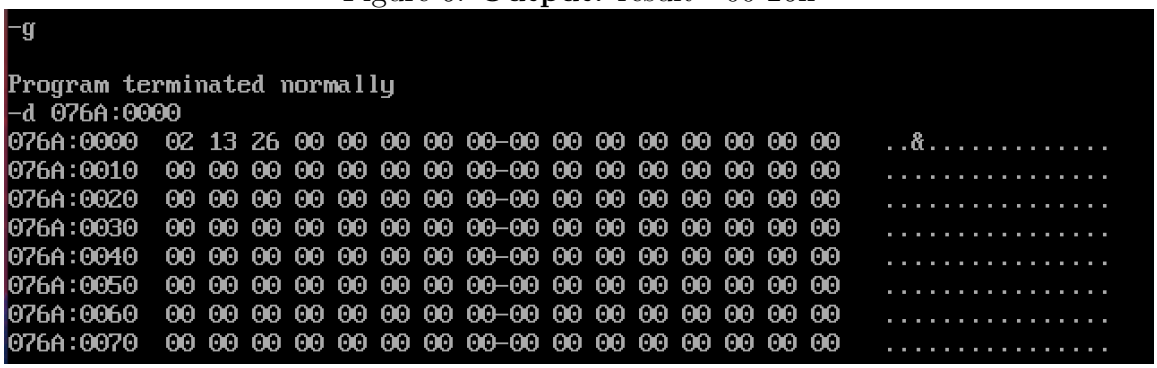


Figure 6: **Output:** result - 00 26h



8 Bit Division

Program:

Program	**Comments**
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov al,opr1	Move contents of opr1 to AL register
mov ah, 00h	Move hex value 00 to AH register
mov bl,opr2	Move contents of opr2 to BL register
div bl	AX = AL / BL
mov quotient,al	Move contents of AL register to quotient
mov remainder,ah	Move contents of AH register to remainder
mov ah,4ch	
int 21h	Request interrupt routine

Unassembled Code:

```
D:\>debug 8BITDIV.EXE
-U
076B:0100 B86A07          MOV     AX,076A
076B:0103 8ED8            MOV     DS,AX
076B:0105 A00000          MOV     AL,[0000]
076B:0108 B400            MOV     AH,00
076B:010A 8A1E0100        MOV     BL,[0001]
076B:010E F6F3            DIV     BL
076B:0110 A20200          MOV     [0002],AL
076B:0113 882E0300        MOV     [0003],AH
076B:0117 B44C            MOV     AH,4C
076B:0119 CD21            INT     21
076B:011B 882E0300        MOV     [0003],CH
076B:011F B44C            MOV     AH,4C
```

Input and Output:

Figure 7: **Input:** opr1 - 99h, opr2 - 11h

```
-d 076A:0000
076A:0000  99 11 00 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 - 09h, Remainder - 00h

```
-g

Program terminated normally
-d 076A:0000
076A:0000  99 11 09 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   ................
```

Result:

The 8086 programs were written to perform 8-bit arithmetic operations, and the results observed.