Exp No: 1 **Date:** 23/08/2020

8-bit Arithmetic Operations

Name: Swetha Saseendran

Register Number: 185001183

Aim:

To write an assembly level program to perform basic arithmetic operation of adding, subtraction, multiplication and division of two 8-bit numbers using an 8086 microprocessor.

Procedure for executing MASM:

I. Open Dosbox and mount the
II. masm folder to the required drive using the command - ("mount drive-name location-of-masm-file")
III. Go to the mounted drive ("Drive-name:")
IV. Save the 8086 program with extension .asm in the same folder using command "edit" in Dosbox or use your desired editor and write your program and save in the same location where the masm file is located with extension asm.
V. Assemble it using the command ("masm filename.asm")
VI. Link the file using the command ("link filename.obj")
VII. Debug the file to execute and analyse the memory contents,
VIII. ("debug filename.exe").
IX. Now use command "u" to display the unassembled code.
X. Use command ("d segment:offset") to see the content of memory locations starting from segment:offset address
XI. Execute using the command "g" and check the outputs by repeating the previous step.
XII. Use command ("e segment:offset") to edit the variables.
XIII. Command "q" to exit from debug and command "exit" from command prompt to close dosbox.

Programs:

(i) 8-bit Addition:

Algorithm:

- Initialize the data segment
- Move data segment address to ds
- Load operand-1 to ah and operand-2 to bh
- Load 00h to ch register for carry
- Add ah and bh
- If there is no carry being generated, go to here segment else, increment ch by 1
- In here segment,
    - Load ah to result

    - Load ch to carry
- Terminate the program

Program	Comments
assume cs:code, ds:data	Using assume directive to declare data and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	
result db 00H	
carry db 00H	
data ends	
code segment	
org 0100h	Set location for code segment at 0100h
start: *mov ax,data*	Transferring address of data segment to ds
mov ds,ax	
mov ah,opr1	Value of opr1 is loaded to ah
mov bh,opr2	Value of opr2 is loaded to bh
mov ch,00h	Initializing the value of ch
add ah,bh	ah=ah+bh
jnc here	Jump to “here” segment if no carry is generated
here: *mov result,ah*	Load register value of ah to result
mov carry,ch	Load ch value to carry
mov ah,4ch	Termination of execution
int 21h	
code ends	Ending the segment with the segment name
end start	

Snapshot of sample input and output:

Sample Input:

opr1=FF

opr2=FE

```
-d 076a:0000
076A:0000  11 99 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   ................
-e 076a:0000
076A:0000  11.FF   99.FE

-d 076a:0000
076A:0000  FF FE 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   ................
-
```

Sample Output:

AH=FD

carry=01

```
-g

Program terminated normally
-d 076a:0000
076A:0000  FF FE FD 01 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   ................
```

(ii) 8-bit Subtraction

Algorithm:

- Initialize the data segment
- Move data segment address to ds
- Load operand-1 to ah and operand-2 to bh
- Load 00h to ch register
- Subtract ah and bh
- If ah is greater than bh, goto here segment else, increment ch by 1 and find the 2's complement of ah
- In here segment,

- Load ah to result
- Load ch to borrow

- Terminate the program

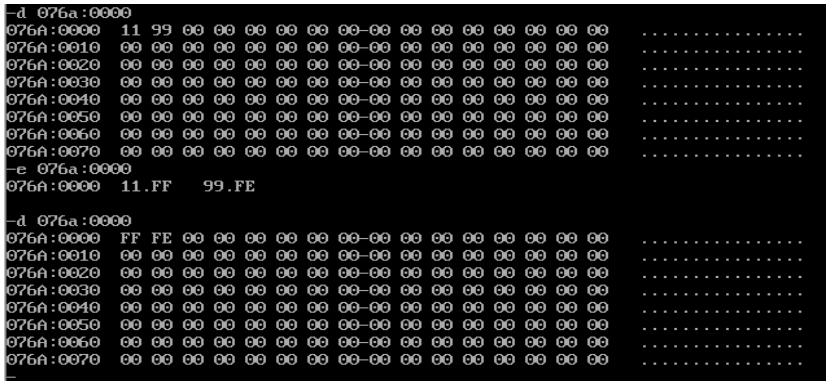
Program	Comments
assume cs:code, ds:data	Using assume directive to declare data and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	
result db 00H	
borrow db 00H	
data ends	
code segment	
org 0100h	Set location for code segment at 0100h
start: *mov ax,data*	Transferring address of data segment to ds
mov ds,ax	
mov ah,opr1	Value of opr1 is loaded to ah
mov bh,opr2	Value of opr2 is loaded to bh
mov ch,00h	Initializing the value of ch
sub ah,bh	ah=ah-bh
jnc here	Jump to “here” segment if ah>bh
inc ch	Increments ch by 1
neg ah	2’s complement of ah
here: *mov result,ah*	Load register value of ah to result
mov borrow,ch	Load ch value to borrow
mov ah,4ch	Termination of execution
int 21h	
code ends	Ending the segment with the segment name

Snapshot of sample input and output:

(i) Sample Input:

opr1=FF

opr2=FE



Sample Output:

AH=01

borrow=00

```
-g

Program terminated normally
-d 076a:0000
076A:0000  FF FE 01 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   ................
-
```

(ii) Sample Input:

opr1=FE

opr2=FF

```
-e 076a:0000
076A:0000  11.FE   99.FF

-d 076a:0000
076A:0000  FE FF 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   ................
-
```

Sample Output:

AH=01

borrow=01

```
-g

Program terminated normally
-d 076a:0000
076A:0000  FE FF 01 01 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   ................
-
```

(iii) 8-bit Multiplication:

Algorithm:

- Initialize the data segment
- Move data segment address to ds

- Load operand-1 to al and operand-2 to bl
- Multiply bl (ax=al x bl)
- Load ax to result
- Terminate the program

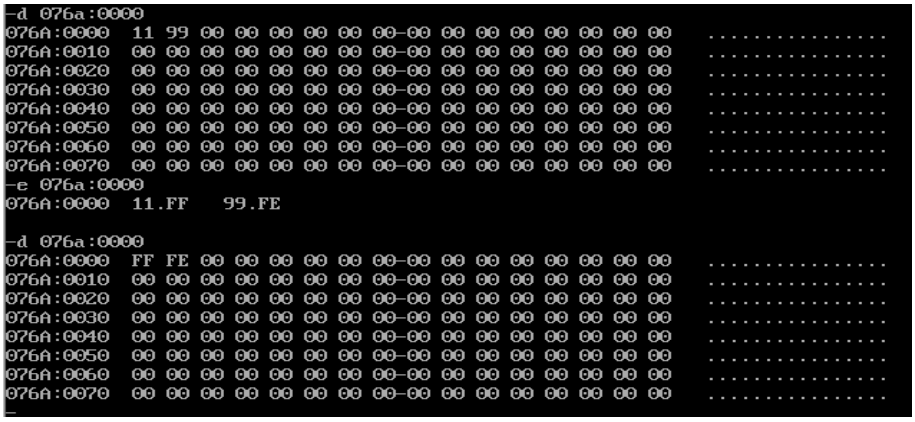
Program	Comments
assume cs:code, ds:data	Using assume directive to declare data and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	
result dw 0000H	
data ends	
code segment	
org 0100h	Set location for code segment at 0100h
start: mov ax,data	Transferring address of data segment to ds
mov ds,ax	
mov al,opr1	Value of opr1 is loaded to al
mov bl,opr2	Value of opr2 is loaded to bl
mul bl	ax=al x bl
mov result,ax	Load register value of ax to result
mov ah,4ch	
int 21h	Termination of execution
code ends	Ending the segment with the segment name

Snapshot of sample input and output:

Sample Input:

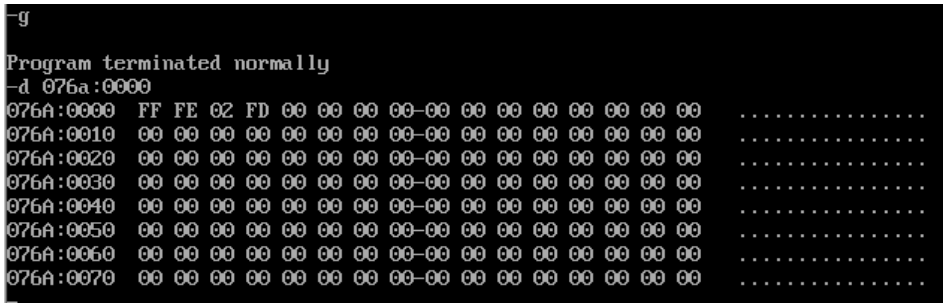
opr1=FF

opr2=FE



Sample Output:

AX=FD02



(iv) 16-bit by 8-bit Division:

Algorithm:

- Initialize the data segment
- Move data segment address to ds
- Load ah with 00
- Load operand-1 to ax and operand-2 to bl
- Divide bl (al=ax / bl ; remainder in ah)
- Load al to result
- Load ah to rem (remainder)
- Terminate the program

Program	Comments
assume cs:code, ds:data	Using assume directive to declare data and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	
quo db 00H	
rem db 00H	
data ends	
code segment	
org 0100h	Set location for code segment at 0100h
start: *mov ax,data*	Transferring address of data segment to ds
mov ds,ax	
mov ax,opr1	Register ah is loaded with 00
mov bl,opr2	Value of opr1 is loaded to ax
div bl	Value of opr2 is loaded to bl
mov quo,al	al = ax / bl
mov rem,ah	Load register value of al to result
mov ah,4ch	Load register value of ah to rem

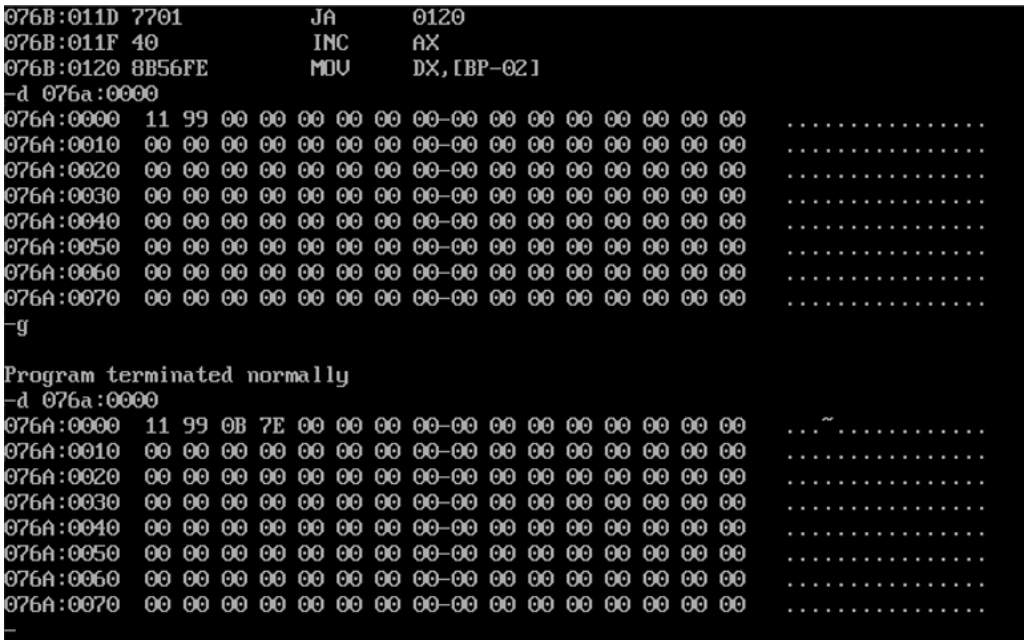
int 21h	Termination of execution
code ends	Ending the segment with the segment name

Snapshot of sample input and output:

Sample Input and Output:

opr1=11

opr2=99



Result:

The assembly level program to perform basic arithmetic operation of two 8-bit numbers using an 8086 microprocessor has been implemented.