**Lab 02: Arithmetic Operations**

**OBJECTIVE**

To learn the basic arithmetic commands and their use.

**Arithmetic Operations**

**1.** **ADD**: A source operand is added to a destination operand and the sum is stored in the destination. Operands must be the same size.

Syntax: **Add** *destination, source*

Instruction Formats:

Add reg,reg Add reg,operand

Add mem,reg Add mem,operand

Add reg,mem

Example: Add al,20h ; al = al + 20h

Add al,bl ; al = al + bl

Add eax,[ebx] ; Add the 4 bytes in memory at the address contained in EBX into EAX

**2.** **Sub:** Subtracts the source operand from the destination operand.

Syntax: **Sub** *destination, source*

Instruction Formats:

Sub reg,reg Sub reg,operand

Sub mem,reg Sub mem,operand

Sub reg,mem

Example: Sub al,20h ; al = al – 20h

Sub al,bl ; al = al – bl

Sub eax,[ebx] ; Sub the 4 bytes in memory at the address contained in EBX into EAX

**3.** **Inc:** Increments or adds 1 to a register or memory operand.

Syntax: **inc** *destination*

Instruction Formats:

inc reg inc mem

Example: inc al ; al = al + 1

**4.** **Dec:** Decrements or subtracts 1 from an operand. Does not affect carry.

Syntax: **dec** *destination*

Instruction Formats:

dec reg dec mem

Example: dec al ; al = al – 1

**5.** **Neg:** It reverses the sign of a number by converting the number to its two’s complement and stores the result in the destination.

Syntax: **Neg** *destination*

Instruction Formats:

neg reg neg mem

Example: neg al ; al = -5 if it initially contined 5

**Code: (Add, Sub, Inc, Dec)**

.model small

.stack 100h

.data

.code

main proc

mov al,5

add al,2

;inc al

;sub al,1

;dec al

mov ah,2

mov dl,al

add dl,30h

int 21h

main endp

end main

**6.** **Mul:** Multiplies AL or AX by a *source* operand.

1. If the source is 8 bits, it is multiplied by AL and the product (16 bit) is stored in AX.
2. If the source is 16 bits, it is multiplied by AX and the product (32 bit) is stored in DX:AX.

Syntax: **Mul** *source* ; the destination is AX where AL is mul by source.

Instruction Formats:

Table of 2: (Columns 1, 2 and 3)

Mov al,2 ; mov ax,2 ; mov ax,2

Mov bl,1 ; inc bl ; inc bl

Mul bl ; mul bl ; mul bl

mul reg mul mem

Example: mov al,2

mov bl,4

mul bl ; Ax = al\*bl = 8

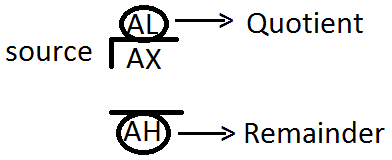
**7.** **Div:** Performs either 8 or 16-bit unsigned integer division.

1. If the divisor is 8 bits, the dividend is in AX, the quotient goes to AL, and the remainder to AH i.e. (AL = AX/operand & AH = remainder (modulus).
2. If the divisor is 16 bits, the dividend is DX:AX, the quotient is AX, and the remainder is DX.

Syntax: **Div** *source*

Instruction Formats:

div reg div mem



Example: mov ax,9

mov bl,2

div bl ; AL = 4, AH =1

**Code for Division:**

.model small

.stack 100h

.data

.code

main proc

mov ax,100h

mov bl,8h

div bl

;we want to display QUOTIENT digits, so divide quotient by 10

mov bh,10h ; to display values in separate digit we divide by 10

div bh ; e.g. 20/10 now al=2

mov bh,ah ;remainder moved to bh to avoid loss of contents

mov ah,2

mov dl,al ; display quotient 20/10 = 2 will display

add dl,30h ; ascii conversion

int 21h

mov ah,2

mov dl,bh ; now display the remainder contained in bh=0

add dl,30h ; ascii conversion

int 21h

main endp

end main

**Lab Code 01:** Perform the following arithmetic expression

Result = -5 + (8 - 2)

**Hint:** Use neg, sub and add commands.

.model small

.stack 100h

.data

.code

main proc

mov al,5

neg al ; al = -5

mov bl,8

sub bl,2 ; (8-2)=6

add al,bl ; -5 + 6 =1

mov ah,2

mov dl,al

add dl,30h

int 21h

main endp

end main

**Lab Code 02:** To input a character at run time, convert its case (from lower to upper) and display it.

**Hint:** ASCII codes for ‘upper case’ and ‘lower case’ characters differ by 32d or 20h. (a=61h & A=41h).

.model small

.stack 100h

.data

.code

main proc

mov ah,1

int 21h

sub al,20h ; case converted

mov ah,2

;mov dl,0dh ; new line generation

;int 21h

;mov dl,0ah

;int 21h

mov dl,al

int 21h

mov ah,4ch ; terminate program and return to DOS

int 21h

main endp

end main

**The-End**