Activity No. 5

ARITHMETIC INSTRUCTIONS

1. Objective:

To create programs using the arithmetic instructions.

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Identify the arithmetic instructions

2.2 Create programs using DEBUG

2.3 Identify the flag bits

2.4 Describe how an arithmetic instructions affects the flag

3. Discussion :

The ARTHMETIC OPERATORS

The four basic arithmetic operators are the ADD, SUB, MUL, DIV.

The ADD is used for addition.

Syntax:
ADD REGISTER1, REGISTER2
ADD REGISTER, VALUE

Example 4.1:
```
MOV AX,5h
MOV BX,4Fh
ADD AX,BX          ; Adds AX and BX and stores the resulting value in AX.  AX=9h
```

Example 4.2:
```
MOV AX,5h
ADD AX,4Fh         ;AX is set to 5h, and 4Fh is added to it.
```
The SUB is used for subtraction.

Syntax :
SUB REGISTER1, REGISTER2
SUB REGISTER, VALUE

Example 4.3
```
MOV AX, 4F
MOV BX,4A
SUB BX, AX     ;AX=05h would
```

Example 4.4
```
MOV BX,4Fh
SUB BX,5h      ;BX = 4A
```

The MUL is used for multiplication.

Syntax:
```
MUL REGISTER
```

Example 4.5
```
MOV AX,5h
MOV BX,4Fh
MUL BX         ;AX=18B (4Fh x 5h = 18B).
```

The DIV is used for division.

Syntax:
```
DIV REGISTER
```

Example 4.6:
```
MOV AX,5h
MOV BX,4Fh
DIV BX         ;AX= Fh since 4Fh / 5h = Fh.
```

The Table 4.1 lists the Arithmetic Instructions

CODE	DESCRIPTION	CODE	DESCRIPTION
AAA	ASCII Adjust for Addition	DIV	Divide byte or word (unsigned)
AAD	ASCII Adjust for Division	IDIV	Integer divide byte or word
AAM	ASCII Adjust for Multiply	IMUL	Integer multiply byte or word
AAS	ASCII Adjust for Subtraction	INC	Increment byte or word by one
ADC	Add byte or word plus carry	MUL	Multiply byte or word (unsigned)
ADD	Add byte or word	NEG	Negate byte or word
CBW	Convert byte or word	SBB	Subtract byte or word and carry
CMP	Compare byte or word	SUB	Subtract byte or word
CWD	Convert word to double-word	CDQ	Convert double-word to quadword
DAA	Decimal Adjust for Addition	CWDE	Convert word to double-word

DAS	Decimal Adjust for Subtraction	CMPXCHG	Compare and Exchange
DEC	Decrement byte or word by one	XADD	Exchange and add

Table 4.1- The Arithmetic Instructions

THE FLAGS

The EFLAGS (or just *Flags*) register consists of individual binary bits that control the operation of the CPU or reflect the result of arithmetic and logical instructions. Some instructions test and manipulate individual processor flags.

A flag is *set* when it equals 1; it is *clear* (or reset) when it equals 0. The flag register is shown in Figure 4.1.

15	14	12	11	10	9	8	7	6	5	4	3	2	1	0
-	NT	IOPL	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

Figure 4.1 – The Flag Register

4. Resources:

PC
TASM Program

5. Procedure:

Using any text editor ,write the following programs . Execute and show the output. Use the space provide on the Data and Results.

1.

```
dosseg
    .model small
    .stack
    .data
        msg1 db 13,10,"Enter a string with dollar symbol as a break:$"
        msg2 db 13,10,"Reverse of the string is:$"
        strg db 20 DUP(0)
        restr db 20 DUP(0)
    .code
 main proc
    mov ax,@data
    mov ds,ax
    mov es,ax

    mov di,00

    lea dx,msg1
    mov ah,09h
    int 21h
```

```
 read:mov ah,01h
      int 21h
      cmp al,24h
      je next
      inc di
      mov strg[di],al
      jmp read

next: mov si,00

start:cmp di,0
      je dmsg2
      mov al,strg[di]
      mov restr[si],al
      inc si
      dec di
      jmp start

dmsg2:lea dx,msg2
      mov ah,09h
      int 21h

 dis:mov al,restr[di]
      cmp al,0
      je ou
      mov dl,al
      mov ah,02h
      int 21h
      inc di
      jmp dis

 ou: mov ax,4c00h
      int 21h

 main endp
      end
```

2.

```
dosseg

.model small

.stack

.data
```

```
        msg1 db 13,10,"Enter first number:$"
        msg2 db 13,10,"Enter second number:$"
        msg3 db 13,10,"Sum in decimal number:$"
        num1 db ?
        sum db ?
        res db 20 DUP('$')
    .code
main  proc
    mov ax,@data
    mov ds,ax

    lea dx,msg1
    mov ah,09h
    int 21h
    mov ah,01h
    int 21h
    sub al,'0'
    mov num1,al

    lea dx,msg2
    mov ah,09h
    int 21h
    mov ah,01h
    int 21h
    sub al,'0'
    add al,num1
    mov sum,al
```

```
        lea dx,msg3
        mov ah,09h
        int 21h

        mov si,offset res
        mov ax,00
        mov al,sum
        call hex2asc

        lea dx,res
        mov ah,09h
        int 21h

        mov ax,4c00h
        int 21h

main  endp
hex2asc proc near
        push ax
        push bx
        push cx
        push dx
        push si
        mov cx,00h
        mov bx,0Ah
```

```
rpt1: mov dx,00
      div bx

      add dl,'0'
      push dx

      inc cx
      cmp ax,0Ah
      jge rpt1

      add al,'0'
      mov [si],al
rpt2: pop ax
      inc si
      mov [si],al
      loop rpt2
      inc si
      mov al,'$'
      mov [si],al
      pop si
      pop dx
      pop cx
      pop bx
      pop ax
      ret
hex2asc endp
    end
```

Course: BET-CPET 2A	**Activity No.: 5**
Group No.:	**Section: 2A**
Group Members: Joseph C. Arenas	**Date Performed: 03/24/2025**
	Date Submitted: 03/25/2025
	Instructor: Ma'am Delos Trinos

6. DATA AND RESULTS:

Problem 1:

a. V
b. I
c. V
d. I
e. I
f. I
g. I

Problem 2:

a. ZF= 1
   CF= 1
   SF= 0
   OF= 1

b. ZF= 1
   CF= 0
   SF= 0
   OF= 0

c. ZF= 0
   CF= 1
   SF= 0
   OF= 0

d. ZF= 0
   CF= 0
   SF= 1
   OF= 0

Problem 3:

```
DOSSEG
.MODEL SMALL
.STACK 200H

.DATA

.CODE
START:
```

```
    MOV AX, @DATA
    MOV DS, AX

    MOV AX, 989680H
    MOV BX, 1
    DIV BX

    MOV AX, 4C00H
    INT 21H
END START


Problem 4:
DOSSEG
DOSSEG
.MODEL SMALL
.STACK 100H
.DATA
    var1 DW 6
    var2 DW 4
    var3 DW 3
    var4 DW ?

    msg1 DB 13,10, "Result (var4): $"
    res  DB 6 DUP('$')

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    MOV AX, var1
    IMUL WORD PTR -5

    MOV BX, var2
    NEG BX

    MOV CX, var3

    MOV DX, 0
    IDIV CX

    MOV BX, DX
    IDIV BX
    MOV var4, AX
```

```
    LEA DX, msg1
    MOV AH, 09H
    INT 21H

    MOV AX, var4
    CALL hex2asc

    LEA DX, res
    MOV AH, 09H
    INT 21H

    MOV AX, 4C00H
    INT 21H

hex2asc PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    MOV SI, OFFSET res
    MOV CX, 0
    MOV BX, 10

convert_loop:
    MOV DX, 0
    DIV BX
    ADD DL, '0'
    PUSH DX
    INC CX
    CMP AX, 0
    JNE convert_loop

print_loop:
    POP AX
    MOV [SI], AL
    INC SI
    LOOP print_loop

    MOV AL, '$'
    MOV [SI], AL

    POP SI
    POP DX
    POP CX
```

```
    POP BX
    POP AX
    RET
hex2asc ENDP

END START
```

Problem 5:

```
DOSSEG
     .MODEL SMALL
     .STACK
     .DATA
          msg1 DB 13,10,"Enter first number:$"
          msg2 DB 13,10,"Enter second number:$"
          msg3 DB 13,10,"Product in decimal number:$"
          num1 DB ?
          product DW ?
          res DB 20 DUP('$')
     .CODE
 MAIN  PROC
      MOV AX,@DATA
      MOV DS,AX

      LEA DX,msg1
      MOV AH,09H
      INT 21H

      MOV AH,01H
      INT 21H
      SUB AL,'0'
      MOV num1,AL

      LEA DX,msg2
      MOV AH,09H
      INT 21H

      MOV AH,01H
      INT 21H
      SUB AL,'0'

      XOR AH, AH
      MUL num1
      MOV product,AX

      LEA DX,msg3
      MOV AH,09H
```

```
    INT 21H

    MOV SI,OFFSET res
    MOV AX, product
    CALL hex2asc

    LEA DX,res
    MOV AH,09H
    INT 21H

    MOV AX,4C00H
    INT 21H

MAIN  ENDP

hex2asc PROC NEAR
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH SI
      MOV CX,00H
      MOV BX,0AH

 rpt1: MOV DX,00
      DIV BX

      ADD DL,'0'
      PUSH DX

      INC CX
      CMP AX,0AH
      JGE rpt1

      ADD AL,'0'
      MOV [SI],AL
 rpt2: POP AX
      INC SI
      MOV [SI],AL
      LOOP rpt2

      INC SI
      MOV AL,'$'
      MOV [SI],AL

      POP SI
```

```
        POP DX
        POP CX
        POP BX
        POP AX
        RET
hex2asc ENDP

    END
```

PROBLEMS:

1. Indicate whether each of the following is V- valid or I-invalid.
   a. ADD AX,BX
   b. ADD ECX,DX
   c. SUB SI,DI
   d. ADD BX, 90000
   e. SUB DS,1
   f. DEC IP
   g. SUB AH, 123H
2. What will be the value of the Flag after executing each of the following:
   a. MOV AX, 0FFFFH
      ADD AX,1
   b. MOV BH, 2
      SUB BH, 2
   c. MOV SI, 0B9F6H
      SUB SI, 9874H
   d. MOV DX,0
      DEC DX
3. Make a program that shows a **division overflow**.
4. A program to implement the expression: **var4 = (var1 * -5) / (-var2 % var3)**;
5. Write a program which will read two decimal numbers, then multiply them together , and finally print out the result (in decimal).

7. Conclusion:

In conclusion, various operands in assembly help create basic value operations in the programming language. This makes the programming language versatile on arithmetic and other functionalities that require numerical data manipulation.

8. Assessment (Rubric for Laboratory Performance):