Faculty Member:____________________ **Dated: ________________**

Semester:_____________ **Section: ________________**

Department of Electrical Engineering and Computer Science

EE-222 Microprocessor Systems

Lab3: VARIABLES, STRING PRINTING, LOOP, LABEL AND ARITHMATIC OPERATIONS ADDITION AND SUBTRACTION

Name	Reg. No.	Viva / Quiz / Lab Performance 5 Marks	Analysis of data in Lab Report 5 Marks	Modern Tool usage 5 Marks	Ethics and Safety 5 Marks	Individual and Team Work 5 Marks	Total 25 Marks

EXPERIMENT 03

VARIABLES, STRING PRINTING AND

ARITHMATIC OPERATIONS ADDITION AND SUBTRACTION

OBJECTIVE:

- In this lab you would perform following arithmetic operations
    1. **Addition**
    2. **Subtraction**
- Getting introduced to the registers and basic commands used to perform arithmetic operations i.e. addition and subtraction in 8086 assembly language.
- To familiarize with the initialization of a variable, **loop**, **label** and **string** in assembly language.
- **Defining** and **displaying** a **variable** and **string**.

EQUIPMENT:

- **SOFTWARE:**
    1. Turbo assembler(TASM)

DISCUSSION:

VARIABLES

Variables are defined in the **.data** directive of the program structure.

Syntax: VariableName   Datasize   Value

In assembly language Value is called as **Initializer** and Datasize is called as **Initializer Directive**. For example, a variable is defined as follows

```
Var1        db      49              ; ASCII value of 49 is assigned to Var1
Var2        db      ?               ; no value is assigned to Var2, we can assign it inside CS
Var3        db      ‘A’             ; Indirectly ASCII of A is assign to Var3
Var4        db      ‘1’
Str1        db      ‘1234$’         ; printing numbers as a string
Str2        db      ‘hello world$’ ; printing hello world as a string
```

VariableName	should not be a reserved op-code or operand e.g AL, BL, CL, DL, Sub, Add etc.		
Initializer Directive	DB	Define Byte	1 byte, 8 bits
	DW	Define Word	2 bytes, 16 bits
	DD	Define Double Word	4 bytes, 32 bits
	DQ	Define Quad Word	8 bytes, 64 bits
	DT	Define Ten Bytes	10 bytes, 80 bits

DISPLAYING A STRING:

The function number 9 of int 21h is used to display a string.

```
Mov ah,9

INT 21h
```

Display a String

Input: DX = offset address of string

The string must end with $ character.

The LEA instruction:

The int 21h function 9 expects the offset address of string to be in DS. To get it there we use a new instruction:

```
LEA    destination, source
```

Where destination is a general register and source is a memory location. LEA stands for **load effective address**; this commands puts the source offset address into destination.

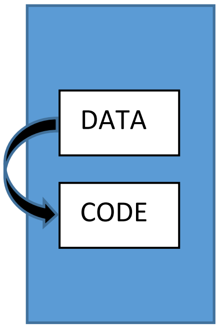
```
MOV AX, @DATA

MOV DS, AX
```

These two commands are used to translate the name @DATA into a number and moving into a segment register.

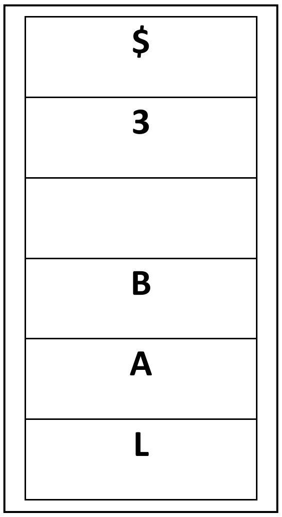
SAMPLE CODE :-

```
;TITLE MSG_DISPLAY
.model SMALL
.stack 100h
.data
        Var1 db 49
        msg db ‘LAB 3$'
.code
Main proc
; accessing address of data in code
            mov AX, @data              ; It moves memory location of  @DATA into AX (16-bit)
            mov DS, AX                  ; DS gets initialized as Heap memory to access variables
; Printing Var1
            mov dl,Var1
            mov ah,2
            int 21h
;Printing msg
            mov DX, offset msg          ; offset holds the beginning address of msg
      ; OR
            LEA DX , msg                ;Load Affective Address
            mov ah , 9
            int 21h
; terminating
           mov ah, 4ch
           int 21h
    main endp
end main
```



moving AX into DS, (register addressing only)

It is an indirect instruction used to points the address of source operand into destination operand



ARITHMATIC OPERATIONS:

The first two kinds of instructions which are used for basic arithmetic operations are

1. Addition
2. Subtraction

Addition and Subtraction instructions:

The **ADD** and **SUB** instructions are used to add or subtract the contents of two registers, a register and a memory location, or to add (subtract) a number to (from) a register or memory location. The syntax is

ADD destination, source

SUB destination, source

For example

ADD word1, AX

This instruction " ADD AX to word1" causes the contents of AX and memory word1 to be added and the sum is stored in word1, AX is unchanged.

SUB AX, DX

This instruction "subtract DX from AX" the value of DX is subtracted from the value of AX, with the difference being stored in AX, DX is unchanged.

Legal combinations for operands for ADD and SUB

OP-CODE	OPERANDS		
ADD	Destination	,	Source
	REG	,	Memory
	Memory	,	REG
	REG	,	REG
	Memory	,	Immediate
	REG	,	Immediate
SUB	Destination	,	Source
	REG	,	Memory
	Memory	,	REG
	REG	,	REG
	Memory	,	Immediate
	REG	,	Immediate

Loop, Label, Inc and Dec instructions:

Loop is a series of instructions that are repeated until a terminating condition is reached.
Label is a name at particular location inside the assembly program where we want the program to go. It is just like the “goto” instruction we use in C++. Labels are normally used with jump commands we will see jump command in coming labs.

1. Label name should not be a reserved command e.g mov: or add: or db: etc.
2. Label should not be started by a number e.g 1test:, 1L: etc.
3. Colon must be used after the label name while initializing inside assembly language, e.g. test: or L1: etc. but not while calling.

Up till now loop along with the label are defined, now how many times this loop will run? This will depend upon the value placed in counter register (CX).

For example, the following instructions are used to print “*” but what to do if we have to print Asterisk for 5 times.

```
;TITLE PRINTING ASTERIC
.model SMALL
.stack 100h
.data
.code
Main proc
        mov ah,2
        mov dl,'*'
        int 21h
  ; repeat above three instructions 5 times
   main endp
end main
```

```
;TITLE PRINTING ASTERIC
.model SMALL
.stack 100h
.data
.code
Main proc
        mov cx,5
        L1:              ; Label L1 Defined
        mov dl,'*'
        mov ah,2
        int 21h
        loop L1          ; Loop calling label
   mov ah,4ch
   int 21h
     main endp
end main
```

INC is used to add 1 to the contents of register or memory location

DEC subtracts 1 from a register or memory location.

The syntax is

INC destination

DEC destination

For example:

INC WORD1: Adds 1 to the contents of WORD1.

DEC BYTE1: Subtracts 1 from the contents of BYTE1.

PROCEDURE:

- As per discussion in class familiarize with defining a string in assembly language, displaying it on screen.
- Write down the commands which you learnt today about defining a string and addressing modes.

EXERCISES:

1. Write an assembly code for displaying two messages both in new line as shown below

   ```
   Hello
   World!
   ```

2. Write an assembly code to print 0 to 9 digits on output using loop.
3. Write an assembly code to print A to Z alphabets on output using loop.
4. Input two numbers by displaying a string “Input number 1: ” and “Input Number 2: ” both in new line, then perform their addition and subtraction (Subtract Number 2 from Number1) and display the results in next line using string as shown “Sum : ” and “Difference : ”

   NOTE: Numbers should be less than 5 and 1st number should be greater than 2nd number.