Lab sheet-03

🧾 Lab Sheet: Basic structure of Assembly code and string operations

🎯 Lab Objectives

By the end of this lab, students will be able to:

1. Understand how to display (print) a string using **INT 21H**.
2. Take string input from the user.
3. Understand the purpose of **DATA**, **CODE**, and **STACK** segments.

🧩 1. Introduction to Segments

The 8086 microprocessor divides memory into parts called **segments**.

They help organize code, data, and stack efficiently.

Segment Register	Purpose
CS	Code Segment – stores program instructions
DS	Data Segment – stores variables and data
SS	Stack Segment – temporary storage
ES	Extra Segment – used for extra data or string operations

When we write `.MODEL SMALL` in EMU8086, it means:

> One code segment and one data segment will be used.

🧩 2. Basic Program Structure

A simple EMU8086 program looks like this:

```
.MODEL SMALL
.STACK 100H

.DATA
    ; variables and messages go here

.CODE
MAIN PROC
    MOV AX, @DATA   ; load address of data segment
    MOV DS, AX      ; move it into DS register

    ; your code goes here

    MOV AH, 4CH     ; exit program
    INT 21H
MAIN ENDP
END MAIN
```

Explanation:

- `.MODEL SMALL` → tells the assembler to use one data and one code segment.
- `.DATA` → section to store variables and strings.
- `.CODE` → section to write program instructions.
- `MOV AX, @DATA` and `MOV DS, AX` → prepare the data segment for use.
- `INT 21H` → performs DOS operations (input/output).

🧩 3. Displaying (Printing) a String

We can display a string using **INT 21H**, function **09H**.

Syntax:

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

```
INT 21H
```

The string must **end with** `$` symbol.

Example 1: Display a Message

```
.MODEL SMALL
.STACK 100H

.DATA
   MSG DB 'HELLO, WORLD!$'

.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX

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

   MOV AH, 4CH
   INT 21H
MAIN ENDP
END MAIN
```

Output:

```
HELLO, WORLD!
```

🧩 4. Taking String Input from User

To take a string from the keyboard, we can use **INT 21H** with function **0AH**.

Structure for Input:

```
BUFFER DB 20        ; maximum characters allowed
DB ?                ; will store number of characters entered
DB 20 DUP(?)        ; actual input storage
```

Example 2: Input and Display a String

```
.MODEL SMALL
.STACK 100H

.DATA
    MSG1 DB 'Enter your name: $'
    MSG2 DB 0DH,0AH,'You entered: $'
    BUFFER DB 20, ?, 20 DUP(?)

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; display message
    MOV AH, 09H
    LEA DX, MSG1
    INT 21H

    ; take input
    MOV AH, 0AH
    LEA DX, BUFFER
    INT 21H

    ; display "You entered:"
    MOV AH, 09H
    LEA DX, MSG2
    INT 21H
```

```
    ; print user input
    MOV AH, 09H
    LEA DX, BUFFER+2
    INT 21H

    ; exit
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

Output Example:

```
Enter your name: Alice
You entered: Alice
```

🧮 5. Summary

Operation	INT 21H Function	Purpose
Display string	AH = 09H	Prints string ending with `$`
Take string input	AH = 0AH	Takes keyboard input into buffer
Program exit	AH = 4CH	Terminates the program

🧠 6. Lab Task

Task Title: Input and Display User Message

Description:

Write an assembly program that:

1. Asks the user to enter a message.
2. Displays the same message back on the next line.

Hint:

Use **INT 21H (AH = 0AH)** for input and **INT 21H (AH = 09H)** for output.

Expected Output:

```
Enter your message: I love Assembly!
You entered: I love Assembly!
```
