Terna Engineering College
Computer Engineering Department

Program: Sem V

Course: Microprocessor Lab

Faculty: ARATHI BOYANAPALLI

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No. 8

A.1 Aim:
Write an Assembly program to display strings using a macro. [Use BIOS/DOS interrupts to read input and display results.]

A.2 Prerequisite: Basic knowledge of 8086 instruction set, interrupts, procedures and macros in 8086.

A.3 Outcome:
After successful completion of this experiment, students will be able to -

1. Use appropriate instructions to program microprocessors to perform various tasks.
2. Develop the program in assembly/ mixed language for Intel 8086 processor.
3. Demonstrate the execution and debugging of assembly/ mixed language programs.

A.4 Theory:
A macro is an extension to the basic ASSEMBLER language. They provide a means for generating a commonly used sequence of assembler instructions/statements. The sequence of instructions/statements will be coded ONE time within the macro definition.

The Syntax for macro definition –

```
%macro macro_name  number_of_params
<macro body>
%endmacro
```

Where number_of_params specifies the number parameters, macro_name specifies the name of the macro.
The macro is invoked by using the macro name along with the necessary parameters. When you need to use some sequence of instructions many times in a program, you can put those instructions in a macro and use it instead of writing the instructions all the time.

A.5 Algorithm:

1. Start.
2. Initialize data segment and code segment.
3. Start the macro body.
4. Write the string to display with macro.
   eg: msg1 DB “message 1 $”.
5. display macro msg1.
6. Store the effective address of the string in the register.
7. End of Macro String.
8. Start the code segment.
9. Invoke the macro to display the string.
10. End.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)

Roll No.: 50	Name: Amey Thakur
Class: TE-Comps B	Batch: B3
Date of Experiment: 08/09/2020	Date of Submission: 08/09/2020
Grade:	

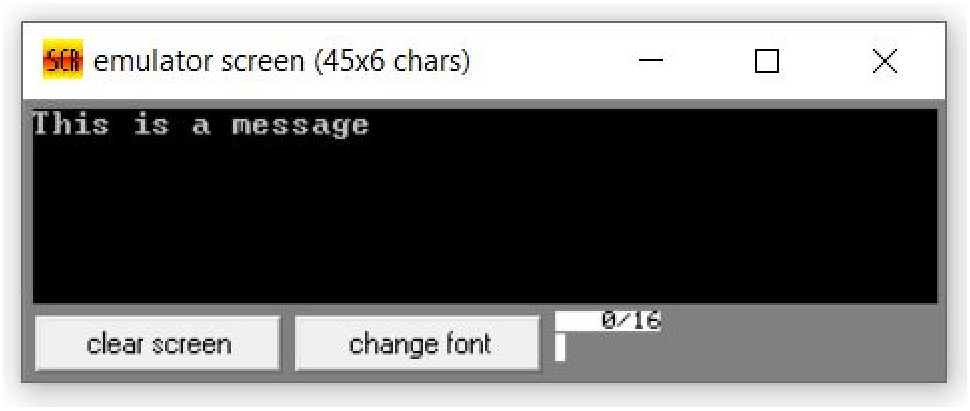
B.1 Observations and learning:

(Software Code written by a student and output of the program)

- **Program to display string using a macro**

```
data segment
        msg1 db "This is a message $"
        display macro msg1
        push ax
        push dx
        lea dx,msg1
        mov ah, 09h
        int 21h
        pop dx
        pop ax
        endm
data ends

code segment
assume cs:code, ds:data
        start:
        mov ax, data
        mov ds,ax
        display msg1
        mov ah, 4ch
        int 21h
        code ends
end start
```

Output:



```
This is a message
```

B.2 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.1)

In brief, assembly language is a common programming language used for microprocessor programming, and Macro and Procedure are two concepts in Assembly. The main difference between Macro and Procedure is that a macro is used for a small number of instructions; mostly, less than ten instructions, while the procedure is used for a large number of instructions; mostly, higher than ten instructions.

B.5 Question of Curiosity

Q1. What are macros in 8086?

Ans:

- ➔ A Macro is a set of instructions grouped under a single unit. It is another method for implementing modular programming in the 8086 microprocessors (The first one was using Procedures).
- ➔ The Macro is different from the Procedure in a way that unlike calling and returning the control as in procedures, the processor generates the code in the program every time whenever and wherever a call to the Macro is made.
- ➔ A Macro can be defined in a program using the following assembler directives: MACRO (used after the name of Macro before starting the body of the Macro) and ENDM (at the end of the Macro). All the instructions that belong to the Macro lie within these two assembler directives. The following is the syntax for defining a Macro in the 8086 Microprocessor:

```
Macro_name  MACRO  [ list of parameters ]
    Instruction 1
    Instruction 2
    - - - - - - - - - - -
    - - - - - - - - - - -
```

```
            - - - - - - - - - - -
            Instruction n
          ENDM
```

- ➔ And a call to Macro is made just by mentioning the name of the Macro:

  Macro_name [list of parameters]
- ➔ It is optional to pass the parameters in the Macro. If you want to pass them to your macros, you can simply mention them all in the very first statement of the Macro just after the directive: MACRO.

Q2. Using Macro write an assembly level language program for the addition of 16bit nos.

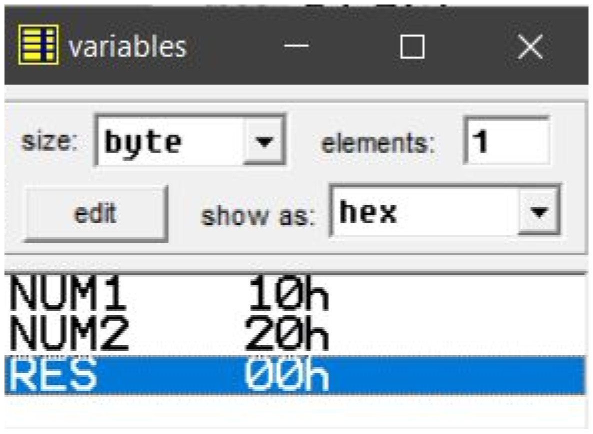
Ans:

Input -

```
assume cs: code
assume ds: data

addition macro a,b,result
   mov al,a
   mov bl,b
   add al,bl
   mov result,al
endm

data segment
   num1 db 10h
   num2 db 20h
   res db ?
   data ends

code segment
   start:
   mov ax,data
   mov ds,ax
   mov ax,0000h
   addition num1,num2,res
   int 3
   code ends

end start
```

Output -

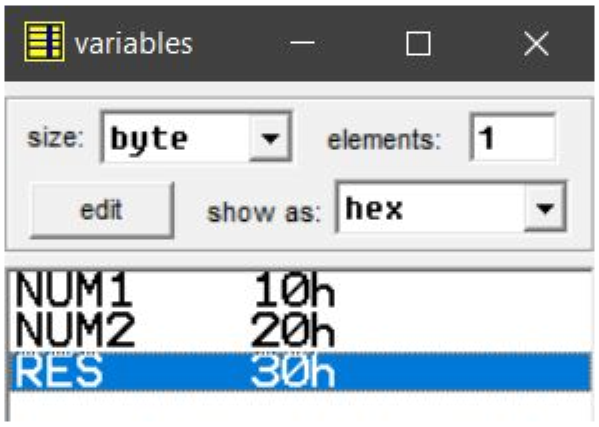
- ➔ **Before Execution**



size: byte elements: 1
show as: hex

```
NUM1        10h
NUM2        20h
RES         00h
```

- ➔ **After Execution**



size: byte elements: 1
show as: hex

```
NUM1        10h
NUM2        20h
RES         30h
```