**Lab # 9**

# Development of Macros in x86 Assembly Language

A procedure definition often includes parameters (also called formal parameters). These are associated with arguments (also called actual parameters) when the procedure is called. For a procedure's in (pass-by-value) parameters, values of the arguments are copied to the parameters when the procedure is called. These values are referenced in the procedure using their local names (the identifiers used to define the parameters).

Parameter values are normally passed on the stack. They are pushed in the reverse order from the argument list. As an example, consider the pseudocode: sum := add2(value1, value2) The 80x86 implementation follows: push cx ; assuming value2 in CX push value1 ; assuming value1 in memory

call add2 ; call procedure to find sum add sp, 8 ; remove parameters from stack mov sum, ax ; sum in memory

If the stack is not cleaned, and a program repeatedly calls a procedure, eventually the stack will fill up causing a runtime error with modern operating systems.

Macro Definition and Expansion :A macro expands to the statements it represents. Expansion is then assembled. It resembles a procedure call, but is different in the way that each time a macro appears in a code, it is expanded. In contrast, there is only one copy of procedure code. A macro is defined as follows:

*name* MACRO *list of parameters*

…..assembly language statements

ENDM

The Macro directive indicate the beginning of the macro definition and the ENDM directives signals the end. Code place between these two directives.

✓

Name must be unique and must follow assembly language conventions

✓

Arguments are names of parameters or even registers that are mentioned in the body.

After the Macro has been written it can be invoked or called by its name , and appropriate values are substituted for parameters

**Example:**

**String** Macro *Data1*

MOV AH,9

MOV DX, OFFSET DATA1

INT 21H

**ENDM**

The above code is the Macro Definition. Argument Data1 is mentioned in the body of Macro is already been defined in data segment as shown below :

Message1 DB “What is your name? $”

In the code segment, the Macro can be invoked by its name with the user‟s actual data **String Message1** ; Invoke the Macro

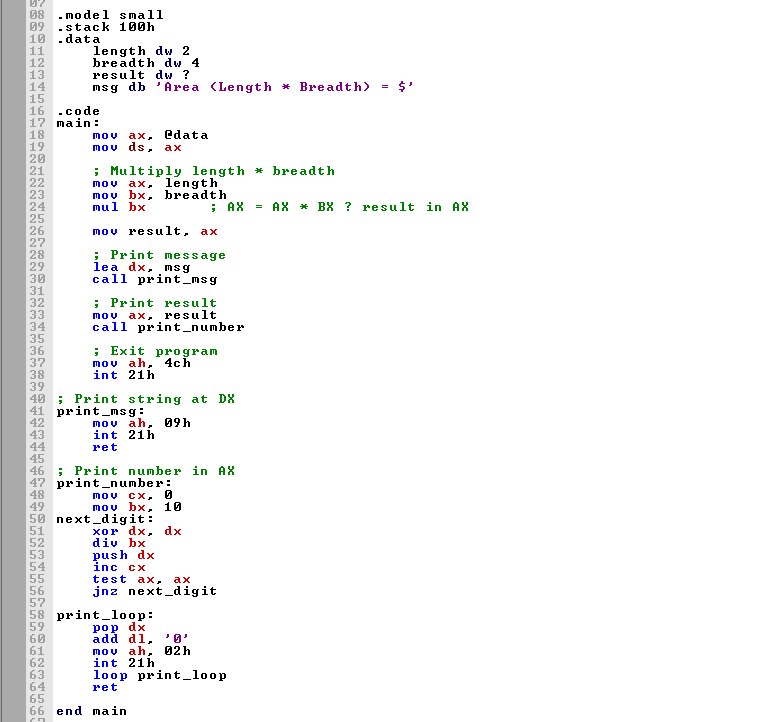
Parameters in the MACRO directive are ordinary symbols, separated by commas. The assembly language statements may use the parameters as well as registers, immediate operands, or symbols defined outside the macro.

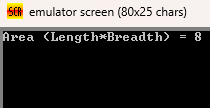
**Comments in Macro:**

There are basically two types of comments in the Macro: List able and non-list able. If comments are precedes by a single semicolon (;) , they will show up in the .lst file , but if comment precedes by double semicolon(;;) ,they will not appear in the .lst file when the programmed is assembled.

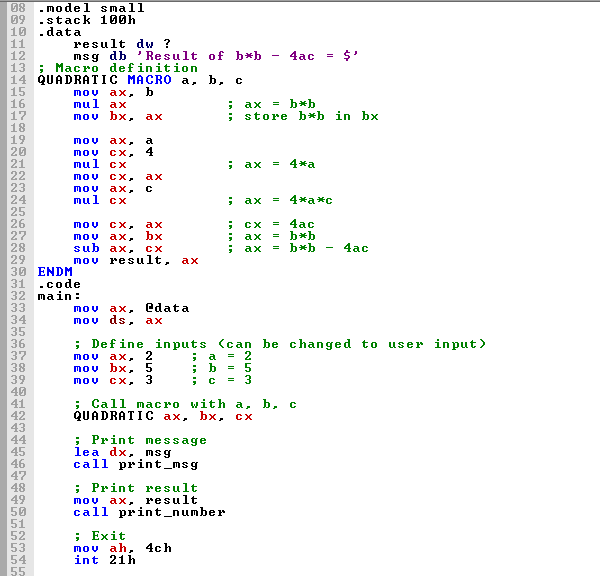
**EXERCISE:**

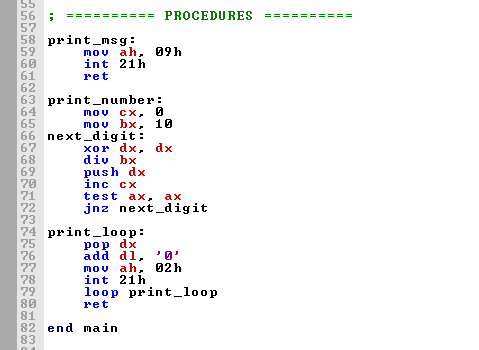
Q1) Write code that calculate and print value of Length \* Breadth where length and breadth are 2 & 4 respectively.

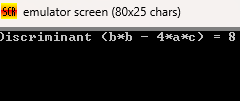




Q2) Repeat Q1-lab # 8, Write Macro definition of a procedure having 3 arguments a,b,c and called it from main procedure.



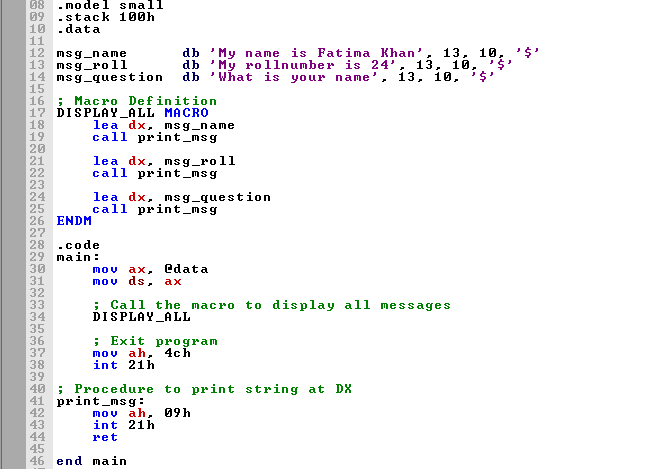


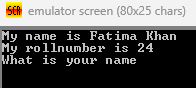


Q3) Write down the assembly code of following output using Only One Macro definition

My name is xxxxx

My rollnumber is yyyyy What is Your name





Q4) Using Macros write down the code to solve following expression: x –2 y + 2z Where x,y,z are values provided by the user.Code should have at least two Macros

