Experiment 02

Learning Objective: Student should be able to develop a calculator (Addition and Subtraction) for a 16 bits number using macros and procedure. (Menu Based).

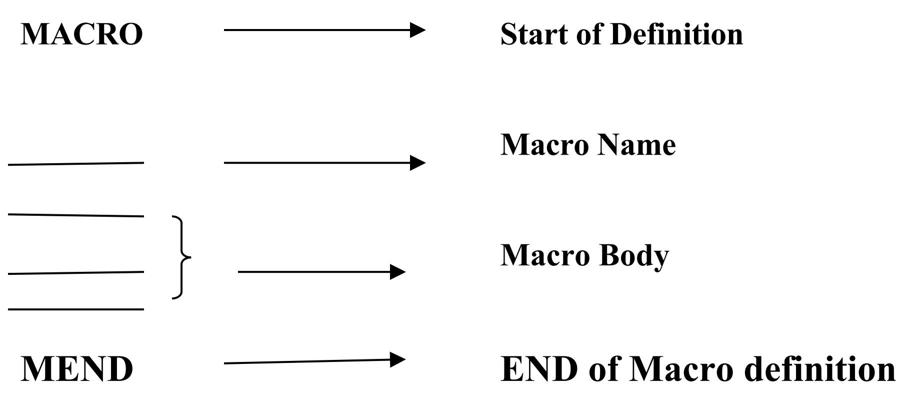
Tools: TASM/MASM

Theory:

Definition of Macro:

The assembly language programmer often finds certain statements being repeated in the program. The programmer can take the advantage of 'MACRO' facility where MACRO is defined to be –**Single line abbreviation for group of instructions.**

The template to be followed for defining a MACRO is as follows:



Definition & function of Macro processor:

- Macro processor is a program which is responsible for processing the macro.
- There are four basic tasks/ functions that any macro instruction processor must perform.

1. **Recognize macro definition:**

   A macro instruction processor must recognize macro definitions identified by the MACRO and MEND pseudo-ops.

1. **Save the definitions**:

   The processor must store the macro instruction definitions, which it will need for expanding macro calls.

2. **Recognize calls:**

The processor must recognize macro calls that appear as operation mnemonics. This suggests that macro names be handled as a type of op-code.

3. **Expand calls and substitute arguments**:

   The processor must substitute for dummy or macro definition arguments the corresponding arguments from a macro call; the resulting symbolic (in this case, assembly language) text is then substituted for the macro call. This text, of course, may contain additional macro definitions or calls.

   In summary: the macro processor must recognize and process macro definitions and macro calls.

The template to be followed for defining a **Procedure** is as follows:

PROC Proc_name $\longrightarrow$ **Start of Definition**

____

____

____

____

RET

Proc_name ENDP **END of procedure**

MACROS		PROCEDURE / Subroutine	
1	The corresponding machine code is written every time a macro is called in a program.	1	The Corresponding m/c code is written only once in memory
2	Program takes up more memory space.	2	Program takes up comparatively less memory space.
3	No transfer of program counter.	3	Transferring of program counter is

			required.
4	No overhead of using stack for transferring control.	4	Overhead of using stack for transferring control.
5	Execution is fast	5	Execution is comparatively slow.
6	Assembly time is more.	6	Assembly time is comparatively less.
7	More advantageous to the programs when repeated group of instruction is too short.	7	More advantageous to the programs when repeated group of instructions is quite large.

Application: Use of Macros and procedure in the Assembly Language programming to write modular program.

Design:

Result and Discussion:

Learning Outcomes: The student should have the ability to

LO1: Explain how to use macros and procedure in the program.
LO2: Compare Macro and procedure.
LO3:Apply macros and procedure to implement the program.

Course Outcomes: Upon completion of the course students will be able to make use of instructions of 8086 to build assembly and Mixed language programs.

Conclusion:

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				