|  |  |
| --- | --- |
| **Name:** | Rahul Yadav |
| **Roll No:** | 76 |
| **Class/Sem:** | SE/IV |
| **Experiment No.:** | 10 |
| **Title:** | Program for printing the string using procedure and macro. |
| **Date of Performance:** | 05/04/2024 |
| **Date of Submission:** | 12/04/2024 |
| **Marks:** |  |
| **Sign of Faculty:** |  |

**Aim:** Program for printing the string using procedure and macro.

**Theory:**

**Procedures:-**

• Procedures are used for large group of instructions to be repeated.

• Object code generated only once. Length of the object file is less the memory

• CALL and RET instructions are used to call procedure and return from procedure.

• More time required for its execution.

• Procedure Can be defined as:

Procedure\_name PROC

……

……

Procedure\_name ENDP

Example:

Addition PROC near

……

……

Addition ENDP

**Macro:-**

• Macro is used for small group of instructions to be repeated.

• Object code is generated every time the macro is called.

• Object file becomes very lengthy.

• Macro can be called just by writing.

• Directives MACRO and ENDM are used for defining macro.

• Less time required for its execution.

• Macro can be defined as:

Macro\_name MACRO [Argument, .... , Argument N]

......

......

ENDM

Example:-

Display MACRO msg

.....

.....

ENDM

**Implementation:**

**Program :**

**org 100h**

**print macro p1**

**lea dx,p1**

**mov ah,09h**

**int 21h**

**endm**

**.data**

**m1 db 10,13,"Leaning Macro$"**

**m2 db 10,13,"Macros are fun$"**

**m3 db 10,13,"Hello World$"**

**.code**

**print m1**

**print m2**

**print m3**

**ret**

**Output:**



**Conclusion:**

Thus, the program for printing the string is successfully implemented using procedure and macro.

* Differentiate between procedure and macros.



* Explain CALL and RET instructions.

CALL:

* Initiates a subroutine or procedure by transferring control to a specified memory address.
* Saves the return address (the address of the instruction following the CALL) onto the stack before transferring control.

RET :

* Returns control from a subroutine or procedure back to the calling code.
* Retrieves the return address from the stack and sets the instruction pointer to that address, effectively resuming execution from the point where the subroutine was called.