Name:	Preeti Rambahal Prajapati
Roll No:	48
Class/Sem:	SE/IV
Experiment No.:	10
Title:	Program for printing the string using procedure and macro.
Date of Performance:	
Date of Submission:	
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

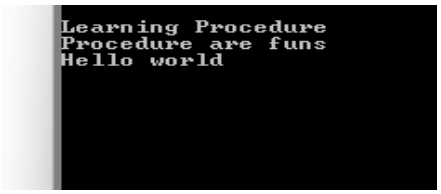
Thus, the program for printing the string is successfully implemented using procedure and macro.

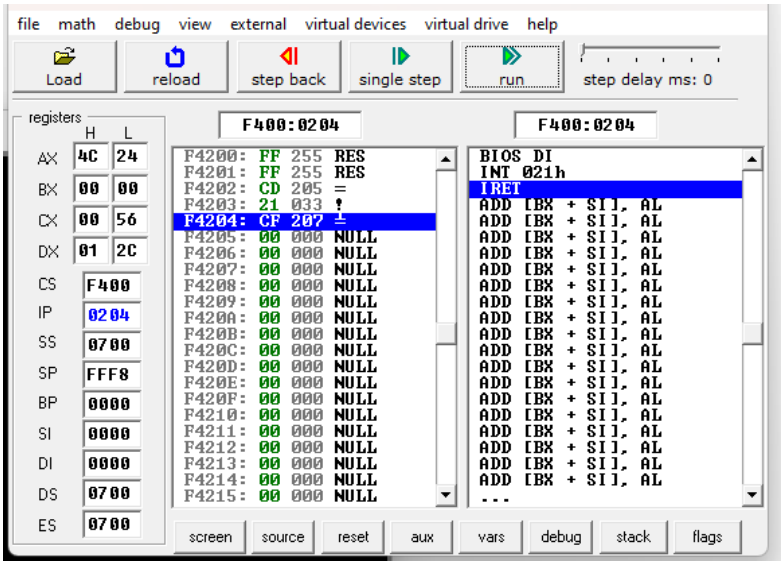
Assembly Code:

```
org 100h

.data

msg1 db 10,13,'Learning Procedure$'

msg2 db 10,13,'Procedure are funs$'
```

```
msg3 db 10,13,'Hello world$'

.code

lea dx, msg1

call print

lea dx, msg2

call print

lea dx, msg3

call print

mov ah, 4CH

int 21h

print PROC

mov ah,09h

int 21h

ret

print ENDP

ret
```

Output :





Conclusion:

- Differentiate between procedure and macros.

Ans. Procedures and macros are both used in assembly language programming to encapsulate and reuse code segments, but they have distinct differences:

- Procedures:
  - A procedure is a named block of code that performs a specific task or operation.
  - Procedures are defined using labels and typically terminated by a return instruction.
  - They can accept parameters passed through registers, stack, or memory.
  - Procedures support local variables and utilize stack space for parameter passing and local variable storage.
  - They offer flexibility in code organization and enable modular programming.
  - Procedures are usually more flexible and powerful but may have overhead due to parameter passing and stack manipulation.

- Macros:
  - Macros are code templates that are expanded inline at compile-time or assembly-time.
  - They are defined using macros directives, which specify the macro name and its replacement text.
  - Macros are invoked using their name and can accept arguments, similar to functions in higher-level languages.
  - They are expanded by the assembler wherever they are invoked, effectively replicating the macro's code at each invocation.

- Macros are useful for repetitive tasks, code generation, and providing syntactic sugar, but they do not support local variables or parameter passing like procedures.
- Macros offer efficiency in code size and execution speed as they eliminate the overhead of procedure calls, but they may result in larger source files.

In summary, procedures are reusable code segments that execute at runtime and support parameters and local variables, while macros are expanded inline during assembly and are suitable for code generation and repetition but lack the capabilities of procedures in terms of parameter passing and local variable storage.

- Explain CALL and RET instructions.

Ans. CALL and RET are essential instructions in assembly language programming for implementing subroutine calls and returns. Here's an explanation of each:

- CALL (Call):
  - CALL is used to transfer control to a subroutine or procedure.
  - It saves the return address (the address of the instruction immediately following the CALL instruction) onto the stack before transferring control to the subroutine.
  - CALL instructions are usually followed by the address of the subroutine to be called.
  - After executing the CALL instruction, the CPU begins executing the first instruction of the subroutine.

- RET (Return):
  - RET is used to return control from a subroutine back to the main program.
  - It retrieves the return address from the top of the stack and transfers control to that address.
  - RET instructions do not require any operands.
  - After executing the RET instruction, the CPU resumes execution at the instruction following the original CALL instruction.

Together, CALL and RET instructions allow for the implementation of subroutines or procedures, enabling modular programming and code reuse. Subroutines encapsulate specific tasks or operations, and CALL/RET facilitate their invocation and return, providing a structured way to organize code and manage program flow.