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]

......
.
.
.
.
.
.

E
N
D
M

Example:-

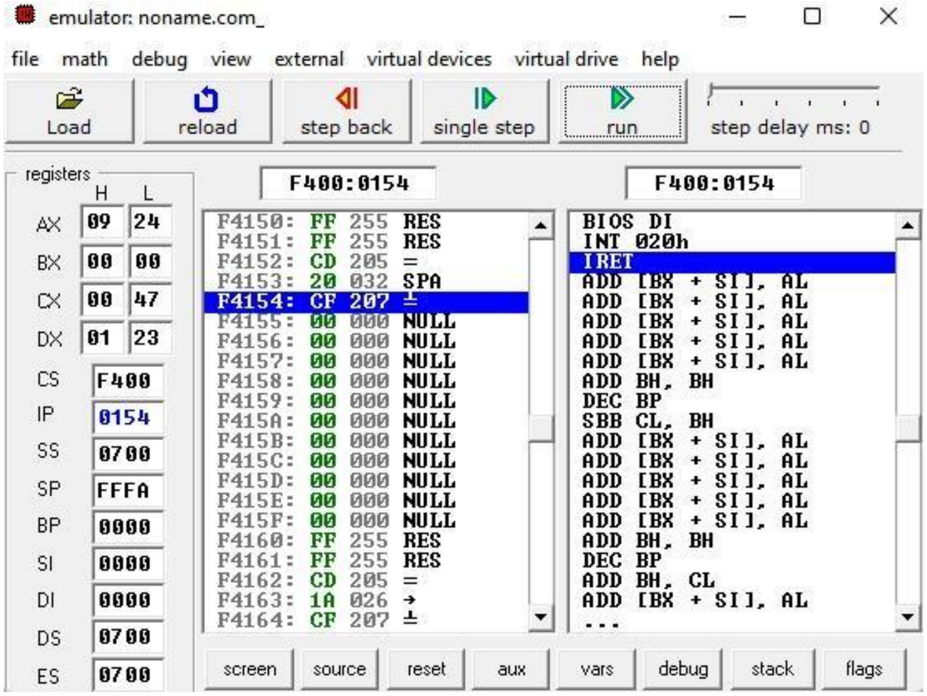
Display MACRO msg
.....
.
.
.
.
.

E
N
D
M

Program Code:

```
Macros
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
```

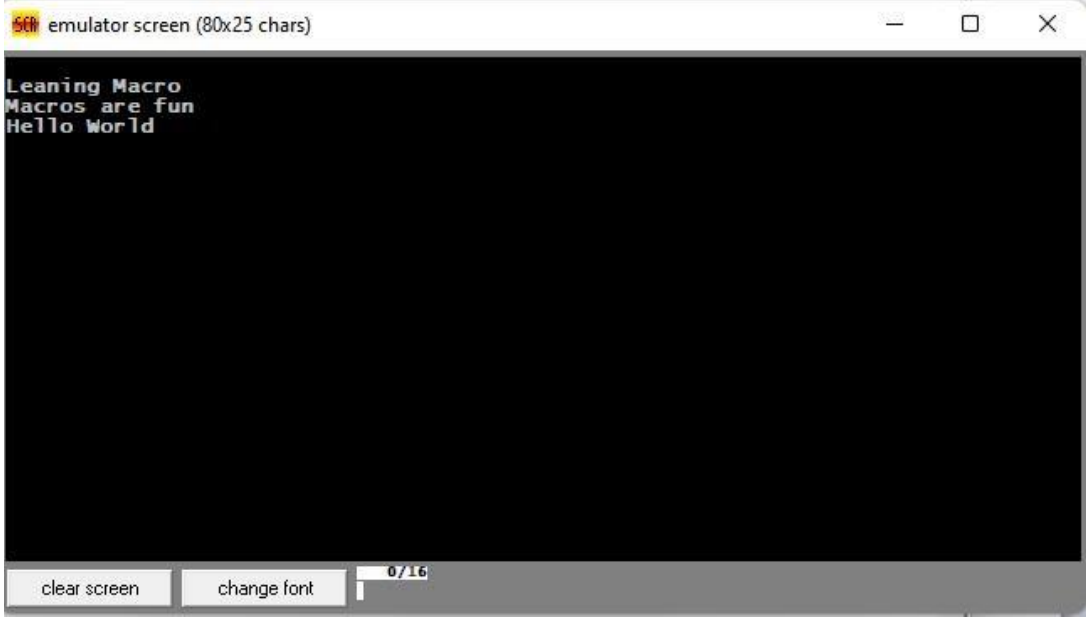


	H	L
AX	09	24
BX	00	00
CX	00	47
DX	01	23

CS	F400
IP	0154
SS	0700
SP	FFFA
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
F400:0154

F4150: FF 255 RES
F4151: FF 255 RES
F4152: CD 205 =
F4153: 20 032 SPA
F4154: CF 207 ±
F4155: 00 000 NULL
F4156: 00 000 NULL
F4157: 00 000 NULL
F4158: 00 000 NULL
F4159: 00 000 NULL
F415A: 00 000 NULL
F415B: 00 000 NULL
F415C: 00 000 NULL
F415D: 00 000 NULL
F415E: 00 000 NULL
F415F: 00 000 NULL
F4160: FF 255 RES
F4161: FF 255 RES
F4162: CD 205 =
F4163: 1A 026 →
F4164: CF 207 ±
```

```
F400:0154

BIOS DI
INT 020h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
SBB CL, BH
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI], AL
...
```

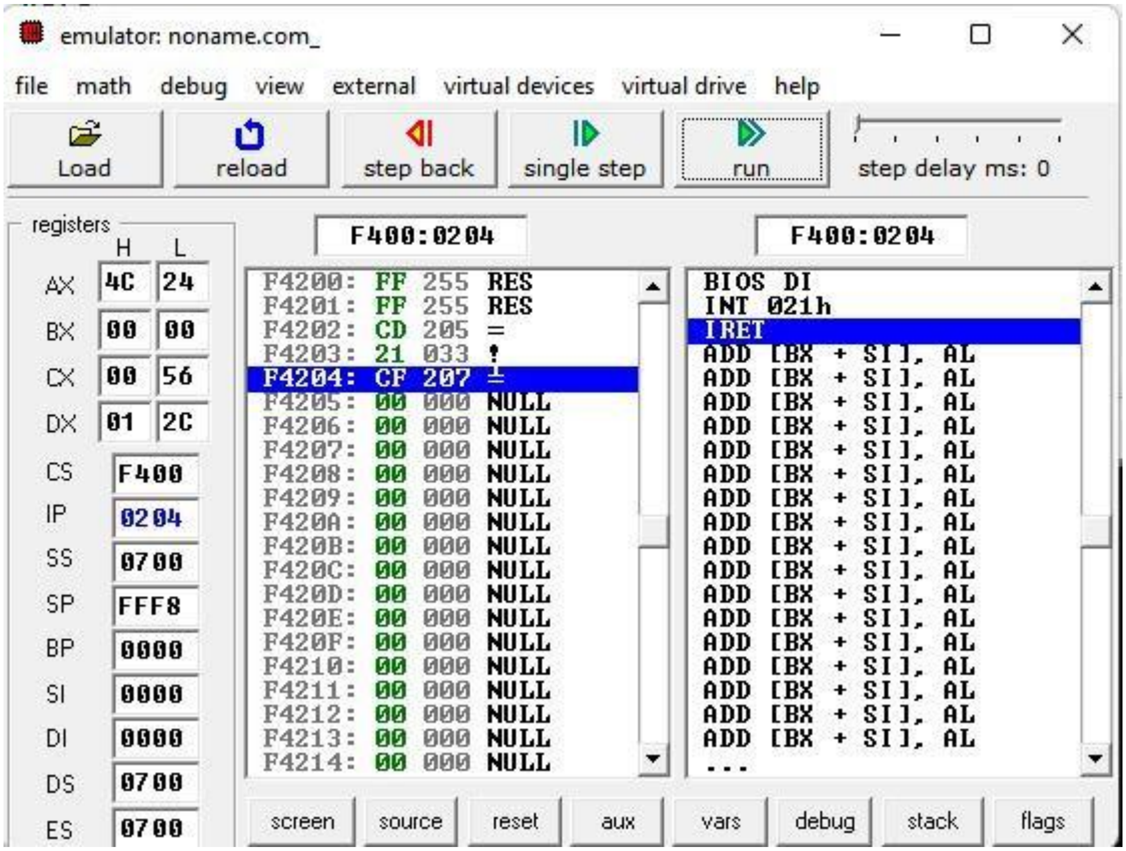


```
Leaning Macro
Macros are fun
Hello World
```

Procedure:

```
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
```

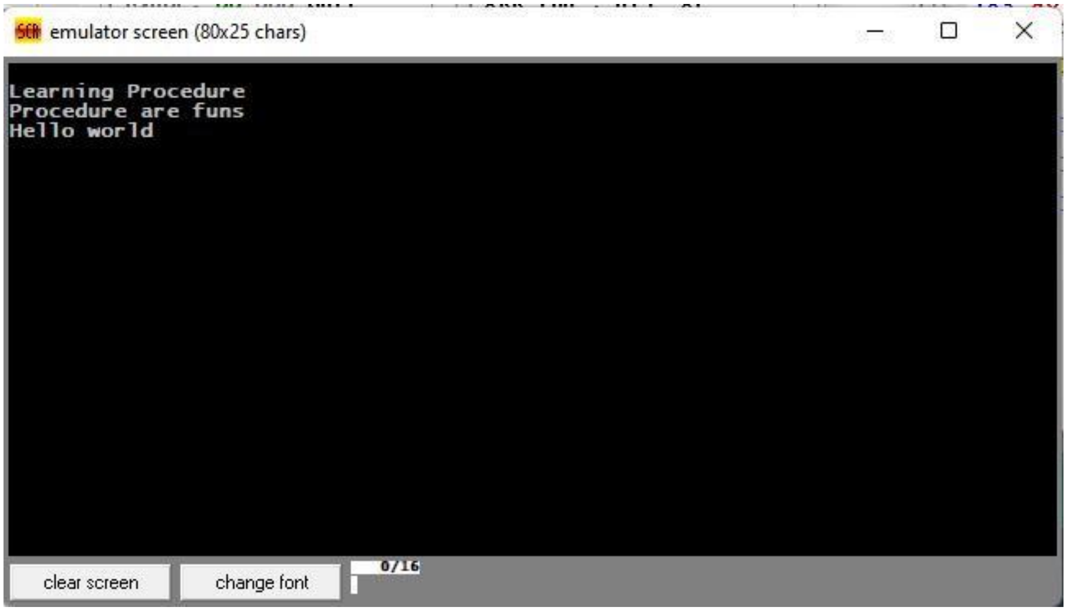
Ret



```
registers
     H   L
AX   4C  24
BX   00  00
CX   00  56
DX   01  2C
CS   F400
IP   0204
SS   0700
SP   FFF8
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
F400:0204
F4200:  FF 255  RES
F4201:  FF 255  RES
F4202:  CD 205  =
F4203:  21 033  !
F4204:  CF 207  ╧
F4205:  00 000  NULL
F4206:  00 000  NULL
F4207:  00 000  NULL
F4208:  00 000  NULL
F4209:  00 000  NULL
F420A:  00 000  NULL
F420B:  00 000  NULL
F420C:  00 000  NULL
F420D:  00 000  NULL
F420E:  00 000  NULL
F420F:  00 000  NULL
F4210:  00 000  NULL
F4211:  00 000  NULL
F4212:  00 000  NULL
F4213:  00 000  NULL
F4214:  00 000  NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ADD [BX + SI], AL
 ...
```

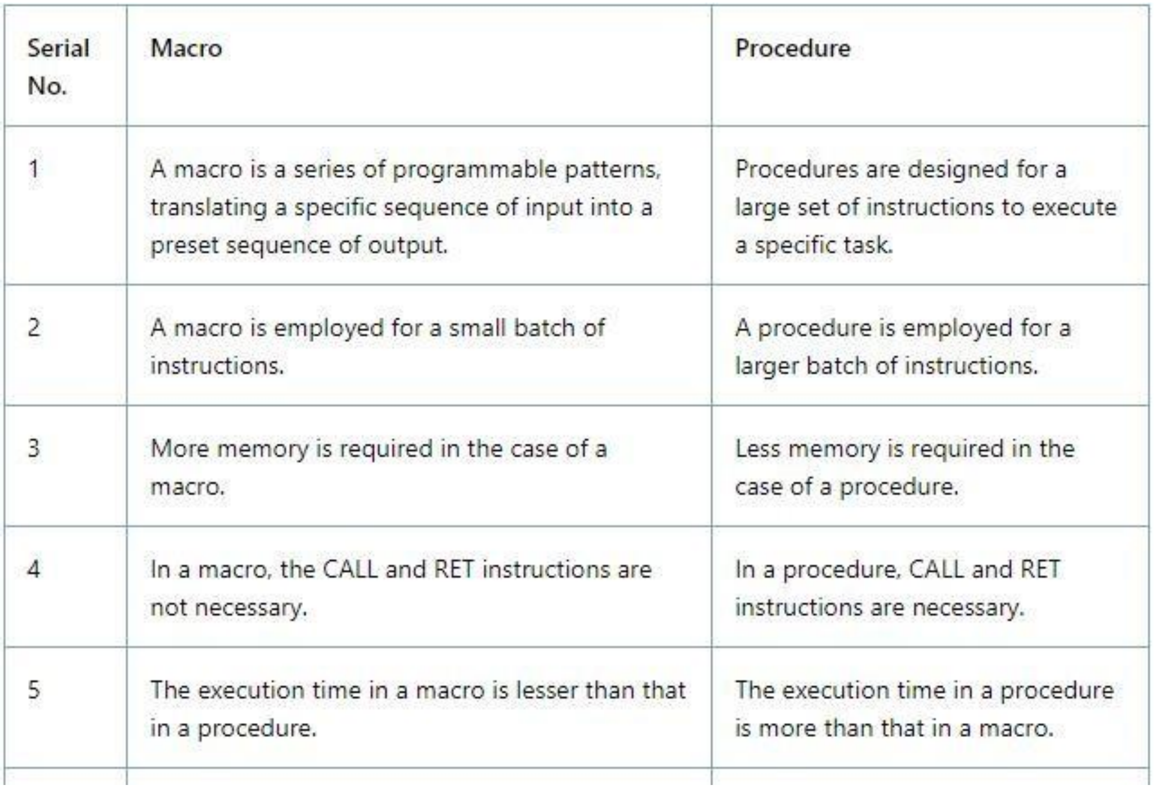


```
Learning Procedure
Procedure are funs
Hello world
```

Conclusion:

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

Q1). Differentiate between procedure and macros.



Serial No.	Macro	Procedure
1	A macro is a series of programmable patterns, translating a specific sequence of input into a preset sequence of output.	Procedures are designed for a large set of instructions to execute a specific task.
2	A macro is employed for a small batch of instructions.	A procedure is employed for a larger batch of instructions.
3	More memory is required in the case of a macro.	Less memory is required in the case of a procedure.
4	In a macro, the CALL and RET instructions are not necessary.	In a procedure, CALL and RET instructions are necessary.
5	The execution time in a macro is lesser than that in a procedure.	The execution time in a procedure is more than that in a macro.

2. Explain CALL and RET instructions.

Ans. The CALL and RET instructions are fundamental to procedural programming and are commonly used in assembly language and low-level programming. They are used for controlling program flow and implementing subroutines or functions.

<u>CALL Instruction:</u>

The CALL instruction is used to transfer control from the current point in the program to a specified subroutine or function.

When a CALL instruction is encountered, the address of the next instruction (the return address) after the CALL is pushed onto the stack.

Then, the program counter (PC) is set to the address of the subroutine or function specified in the CALL instruction.

After the subroutine or function completes its execution, it typically contains a RET instruction to return control to the instruction immediately following the original CALL.

RET Instruction:

The RET (return) instruction is used to return control from a subroutine or function back to the instruction immediately following the corresponding CALL.

When a RET instruction is executed, the address at the top of the stack (the return address) is popped off the stack and loaded into the program counter (PC).

This causes the program to resume execution at the instruction following the original CALL.

Additionally, any parameters or local variables that were pushed onto the stack by the CALL instruction can be cleaned up before the RET instruction is executed, depending on the specific implementation.