Name:	Viraj Oza
Roll No:	69
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:-

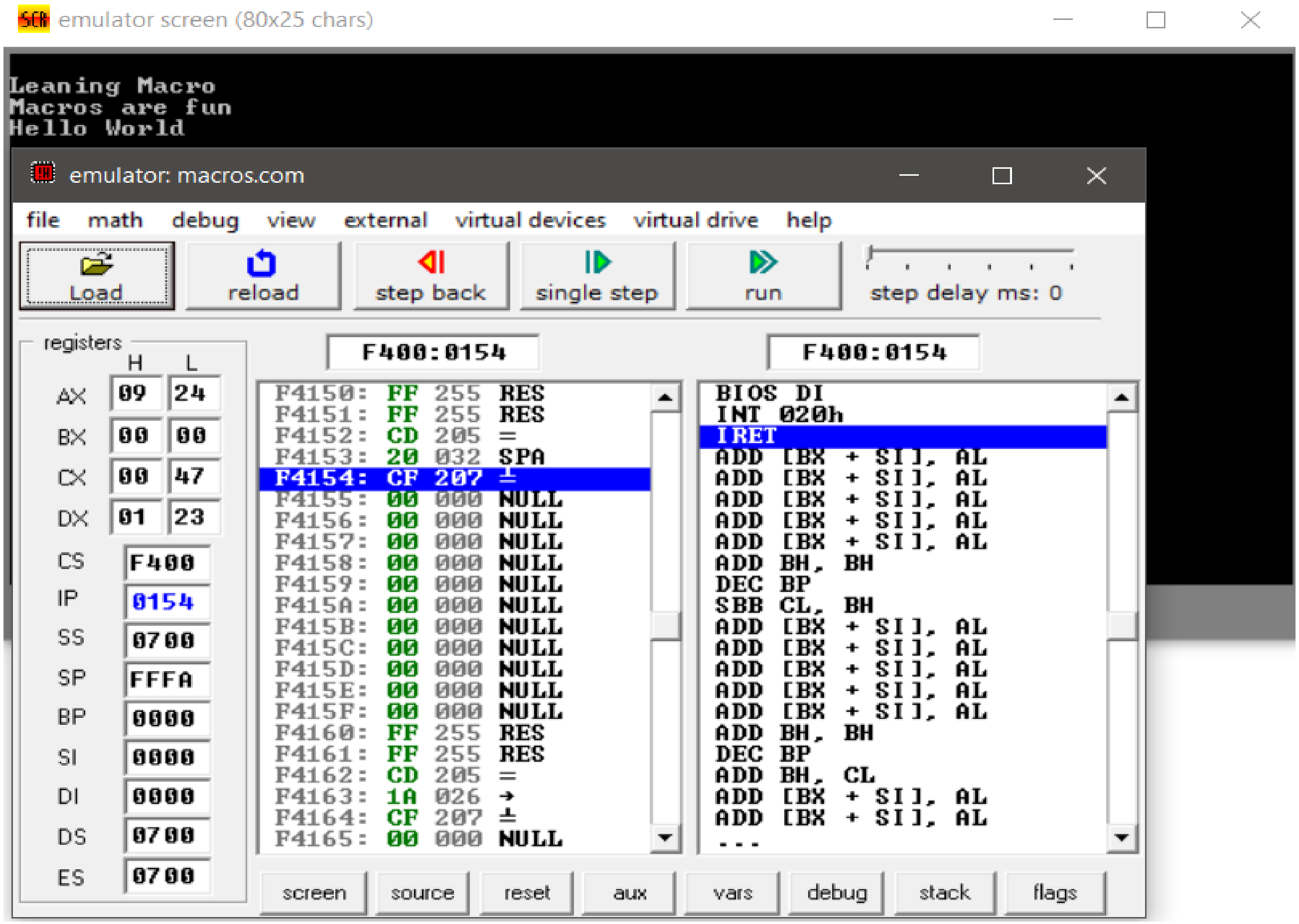
```
Display MACRO msg
.....
.....
ENDM
```

Programs:

String using 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
```

Output:



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

```
    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
```

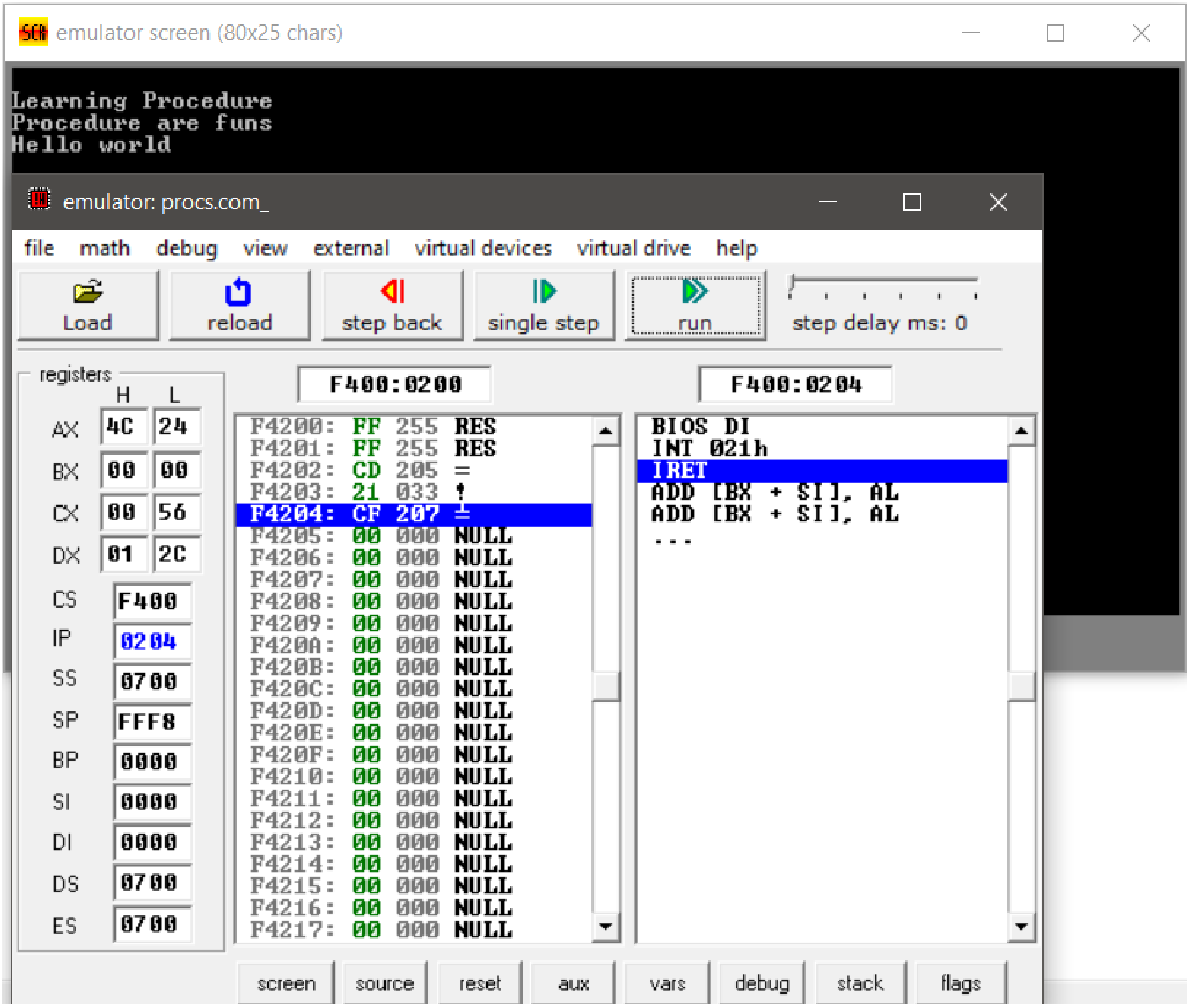
```
F400:0154
F4150: FF 255 RES
F4151: FF 255 RES
F4152: CD 205 ═
F4153: 20 032 SPA
F4154: CF 207 ╧
F4155: 00 000 NULL
F4156: 00 000 NULL
F4157: 00 000 NULL
F4158: 00 000 NULL
F4159: 00 000 NULL
F415A: 00 000 NULL
F415B: 00 000 NULL
F415C: 00 000 NULL
F415D: 00 000 NULL
F415E: 00 000 NULL
F415F: 00 000 NULL
F4160: FF 255 RES
F4161: FF 255 RES
F4162: CD 205 ═
F4163: 1A 026 →
F4164: CF 207 ╧
F4165: 00 000 NULL
```

```
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 BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

String using 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
```

Output:



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

	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	

```
F400:0200
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
F4215:  00 000  NULL
F4216:  00 000  NULL
F4217:  00 000  NULL
```

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

Conclusion:

Thus, the program for printing the string is successfully implemented using procedure and macro in assembly language. The program showcased two different approaches for achieving the same task, providing insights into the usage and benefits of each.

1. Differentiate between procedure and macros.

Ans. Procedures:

a. Named blocks of code that perform specific tasks or operations.
b. Encapsulate a sequence of instructions within a procedure definition using labels.
c. Organize code into logical units for better understanding, maintenance, and reuse.
d. Called using the `CALL` instruction, which transfers control to the beginning of the procedure.
e. Can accept parameters (arguments) passed to them by the calling code and return values back to the caller.
f. Parameters are typically passed via registers or the stack, and return values are often stored in specific registers or memory locations.

Macros:

g. Preprocessor directives that define sequences of instructions or statements for reuse.
h. Expanded inline during the assembly process wherever they are invoked.
i. Used to generate repetitive code or define complex sequences of instructions.
j. Offer flexibility and customization, allowing programmers to define parameters and customize the generated code.
k. Accept parameters specified within parentheses after the macro name, allowing customization when invoked.
l. Parameters are replaced with specified arguments during expansion, and the macro code is inserted directly into the program at each invocation point.

2. Explain CALL and RET instructions.

Ans. CALL (Call Procedure):

- The `CALL` instruction is used to transfer control from the current point in the program to a specific subroutine or procedure.
- Syntax: `CALL destination`
- `destination` specifies the target address of the subroutine or procedure to be called. It can be an immediate value, a label representing the address, or a register containing the address.
- When `CALL` is executed, the address of the instruction immediately following the `CALL` instruction is pushed onto the stack (the return address). Then, control is transferred to the specified destination, and execution continues from there.
- `CALL` is commonly used to invoke procedures or subroutines to perform specific tasks, providing modularity and code reuse in assembly language programs.

RET (Return from Procedure):

- The `RET` instruction is used to return control from a subroutine or procedure back to the calling code.
- Syntax: `RET`
- When `RET` is executed, the return address previously pushed onto the stack by the corresponding `CALL` instruction is popped from the stack and loaded into the instruction pointer (IP or EIP), transferring control back to the instruction immediately following the original `CALL` instruction.
- `RET` typically marks the end of a subroutine or procedure and allows the program flow to resume at the point where the subroutine was called.
- `RET` can also be used to return a value from a subroutine by placing the value in a specific register or memory location before executing `RET`.