Procedures and Macros

- **Procedures:**

When a group of instructions are to be used several times to perform a same function in a program, then we can write them as separate subprogram called procedure or subroutine.

Procedures can be called in a program using CALL instructions.

Procedures are written and assembled as separate program modules and stored in the memory. The assembling of procedures will be done without considering the main program i.e., independent of calling program.

When procedure is called in main-program control is transferred to procedure and after execution the control is transferred back to main program, procedures are called using CALL instruction and returned back using RET instruction.

- **Label PROC Far / Near**

```
------
------
RET Label ENDP.
```

- **Types of Procedures:**
  i. Depending on Position of procedure in memory
     - Intra-segment Procedure
     - Inter-segment Procedure
  ii. Depending on occurrence in program
     - Single Procedure
     - Nested Procedure
     - Re-Cursive Procedure
     - Re-Entrant Procedure

- **Intra-Segment Procedure:**

CALL ➔ $(SP) \leftarrow (SP) - 2$
$((SP) + 1 : (SP)) \leftarrow (IP)$
$(IP) \leftarrow (EA)$

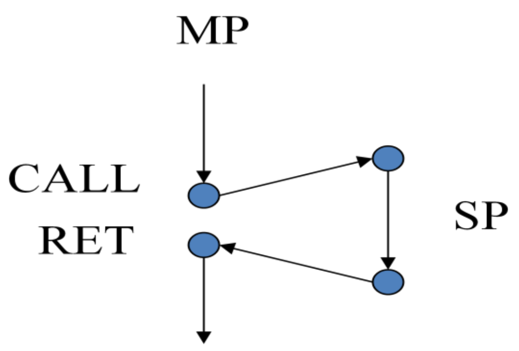
RET ➔ $(IP) \leftarrow ((SP) + 1 : (SP))$
$(SP) \leftarrow (SP) + 2$

- **Inter-Segment Procedure:**

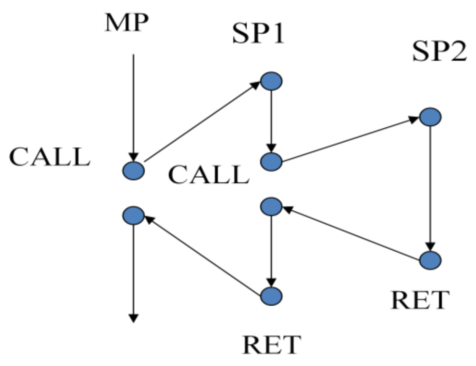
CALL ➔ $(SP) \leftarrow (SP) - 2$
$(((SP) + 1) : (SP)) \leftarrow (CS)$
$(SP) \leftarrow (SP) - 2$
$(((SP) + 1) : (SP)) \leftarrow (IP)$
$(IP) \leftarrow (EA)$
$(CS) \leftarrow$ Seg. Address

- RET ➔ $(IP) \leftarrow ((SP) + 1 : (SP))$
$(SP) \leftarrow (SP) - 2$
$(CS) \leftarrow ((SP) + 1 : (SP))$
$(SP) \leftarrow (SP) - 2$

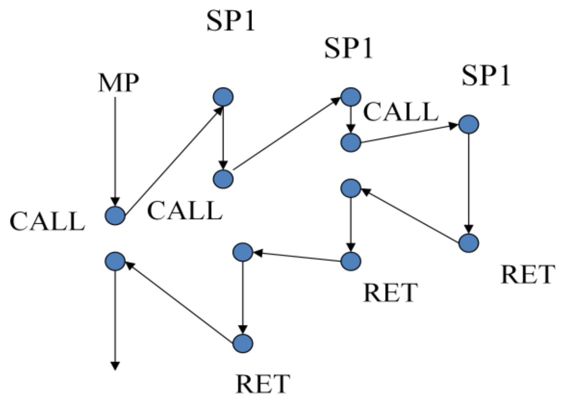
- Single Procedure:
  MP → Main Program
  SP → Sup-Program / Procedure



- **Nested Procedure**

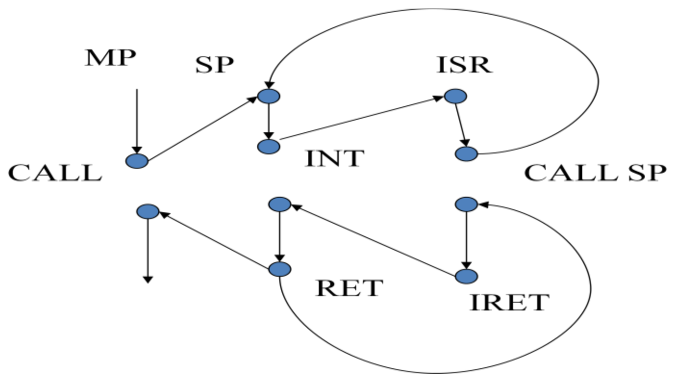


➢ **Re-Cursive Procedure:**



➢ The recursive procedures are implemented using procedure CALL itself, but care must be taken to assure that each successive call does not destroy the parameters and results generated by the previous CALL and make sure that procedure does not modify itself, i.e., each call must store its set of parameters, Registers and all temporary results in a different place in memory.

- Push all the flags and all registers used in the procedure.
- Should use only register or stack to pass parameters.
- **Re-Entrant Procedures:**



- **MACROS:**
- When a group of instructions are to be used several times to perform a same function in a program and they are too small to be written as a procedure, then they can be defined as a MACRO.
- A MACRO is a small group of instructions enclosed by the assembler directives MACRO and ENDM.
- The MACROS are identified by their names and usually defined at the start of a program.
- The Macro is called by its name in the program, whenever the MACRO is called in the program, the assembler will insert the defined group of instructions in place of CALL instruction.
- X1 MACRO

```
X1  MACRO
Mov SI, OFFSET  Arr
Mov DI, OFFSET Arr1

Mov AL, (SI)

Add AL, (DI)

Inc SI

Inc DI

ENDM
```

- Passing parameters in MACROS:

```
X1 MACRO Arr, Arr1
          :
END M
          :
X1 Arr2,Arr3
:
X1 Arr4, Arr5
```

- Advantages and Disadvantages Of Procedures and Macros:
  Procedures: Advantages:
  1. Machine codes for the group of instructions in the procedure has to be put in memory only once.
  2. Each set of instructions can be individually assembled and debugged.
- Disadvantages:
  1. Need stack operations performed.
  2. Over-head time required to call the procedure and return to the calling program.
- Macros Advantages:
  1. Simple
  2. No over-head time required.
- Disadvantages:
  Program may take up more memory due to insertion of machine codes in the program at the place of Macros.

- Comparison of Procedures and Macros:

S.No.	Procedures	Macros
1.	Machine code for instructions are stored in memory only once.	Machine codes are generated for instructions in the Macro each time it is called in the main program.
2.	Accessed by CALL and RET mechanisms during program execution.	Accessed during assembly with name given to Macro when defining it.
3.	Size is unlimited.	Size is limited to few lines.
4.	Parameters are passed in registers, Memory locations or stack.	Parameters are passed as part of statement which calls Macro.
5.	Can be present in Same (or) different segment as that of the main program.	Usually defined at the beginning of program in the same segment.