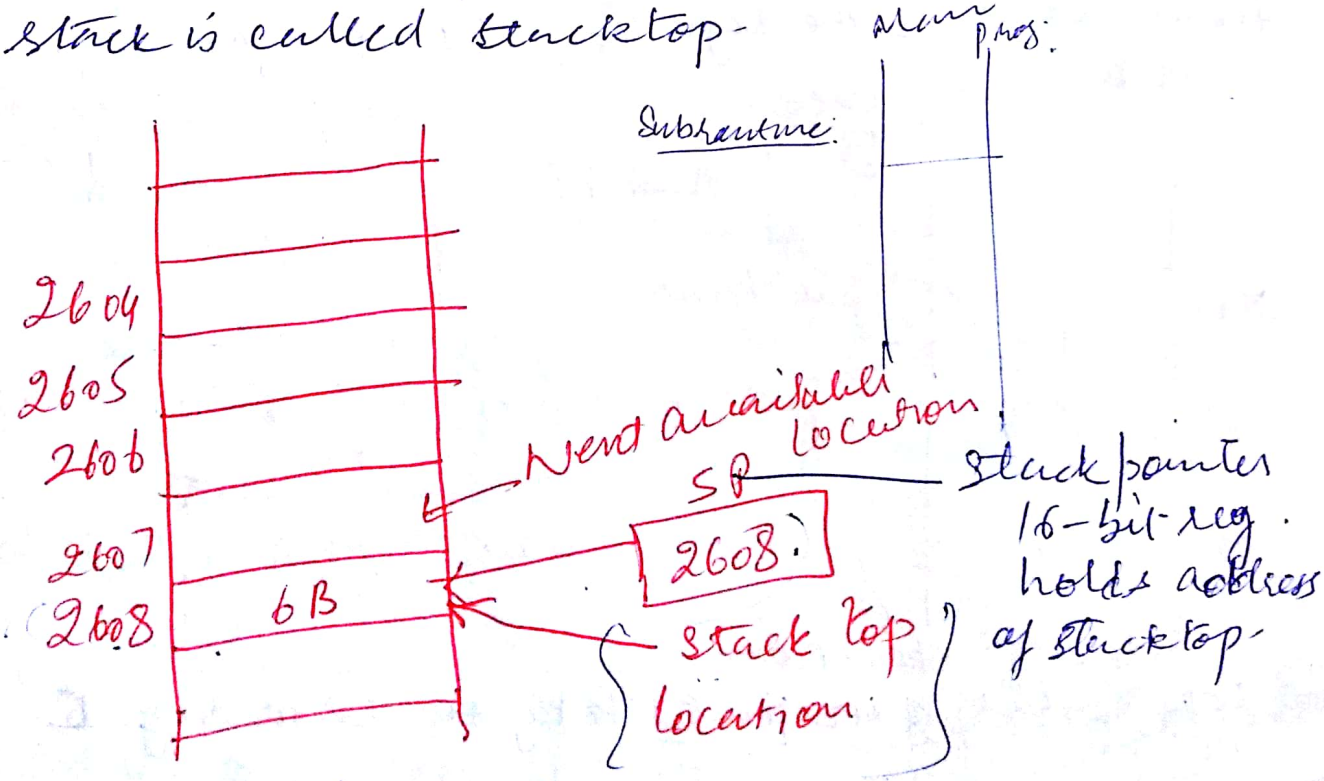
Stack & Subroutines

Stack

- The stack is an area of memory identified by the programmer for temp storage of info.
- during the execution of a prog. sometimes it becomes necessary to save the contents of certain reg. because the reg. are required for some other operation.
- These contents are moved to certain M.L's by Push operation.
- After completing operations, those contents which were saved in the Memory are transferred back to the registers by POP operation.
- Memo. loc. for this purpose are set aside in the beginning and called stack.
- The last M.L of the occupied portion of the stack is called Stacktop.

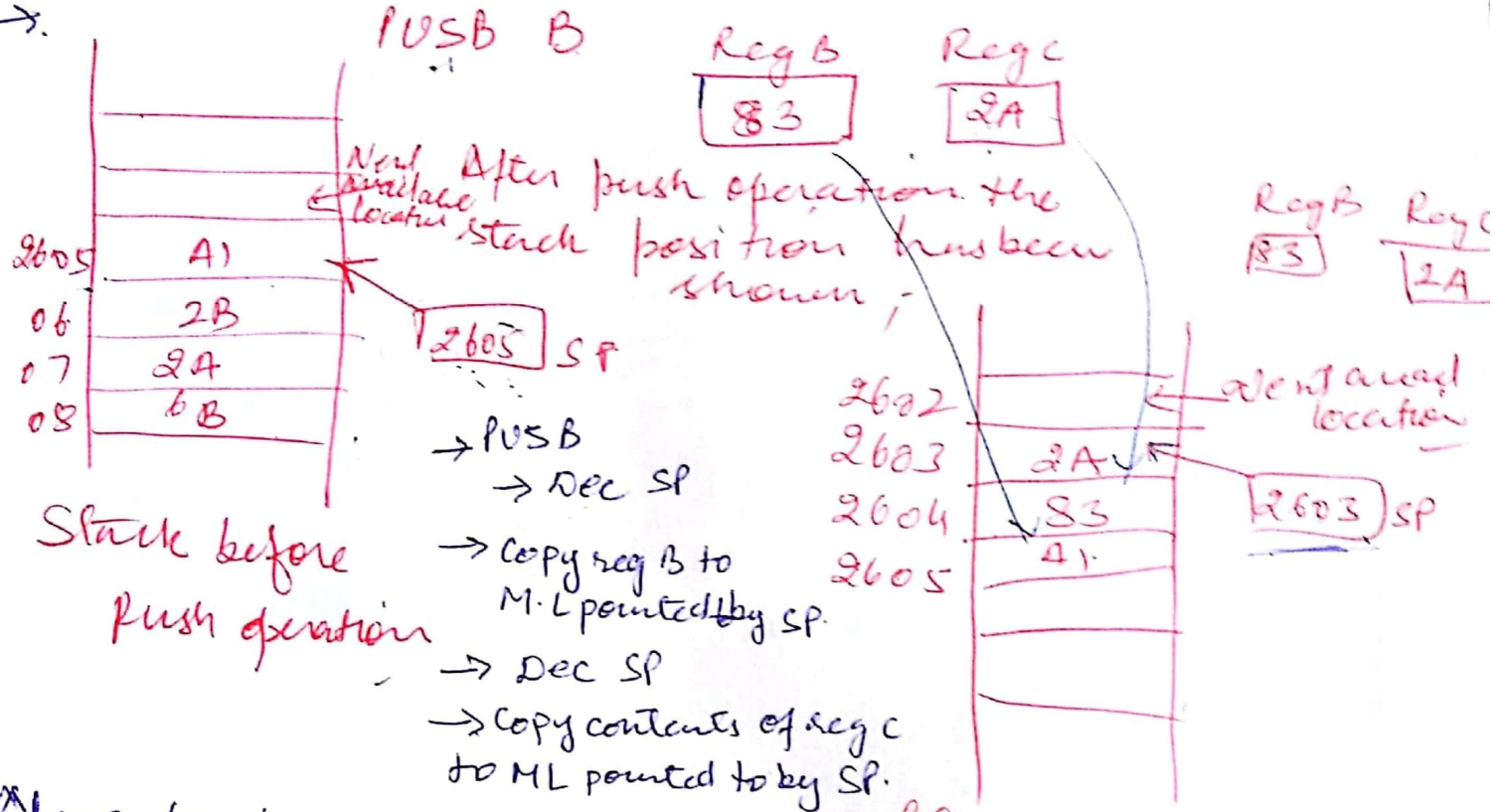


→ SP is initialized in the beginning of the prog. by LXI SP or SPHL inst.

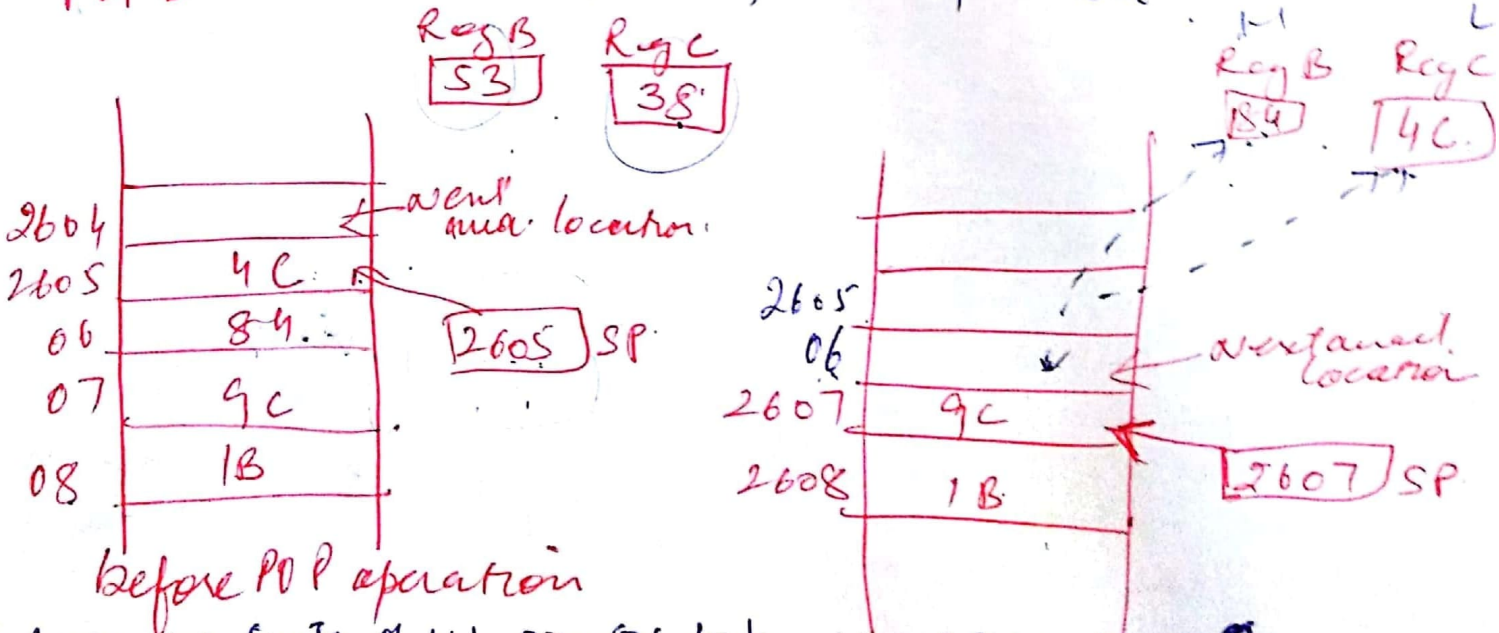
→ Any area of RAM can be used as stack.

→ data are stored in the stack on Last-in-first-out (LIFO) principle. faster than memory access.

→



Now to transfer the contents from stack to the register, POP operation



→ Copy contents of ML pointed to by the SP to reg C
→ Inc SP
→ Copy contents of ML pointed to by SP to reg B → Inc SP

* LIFO: The order of PUSH's and POP's must be opposite of each other in order to retrieve info back into its original location.

```
Push B
Push D

POP D
POP B.
```

* PSW reg. pair: - one additional reg. pair recognised in 8085. [Acc + Flag. reg.]
→ It is possible to push the PSW onto the stack. do whatever operations are needed, then POP it off of the stack.

* Subroutines: A subrout. is a group of inst's that will be used repeatedly in diff. locations of the program.
→ Rather than repeat the same inst several times, they can be grouped into a subroutine that is called from the diff locations.
→ In assembly language, a subroutine can exist anywhere in the code.
→ However, it is customary to place subrout. separately from main prog.

Two inst

Call (CALL)

→ used to redirect program execution to the subroutine

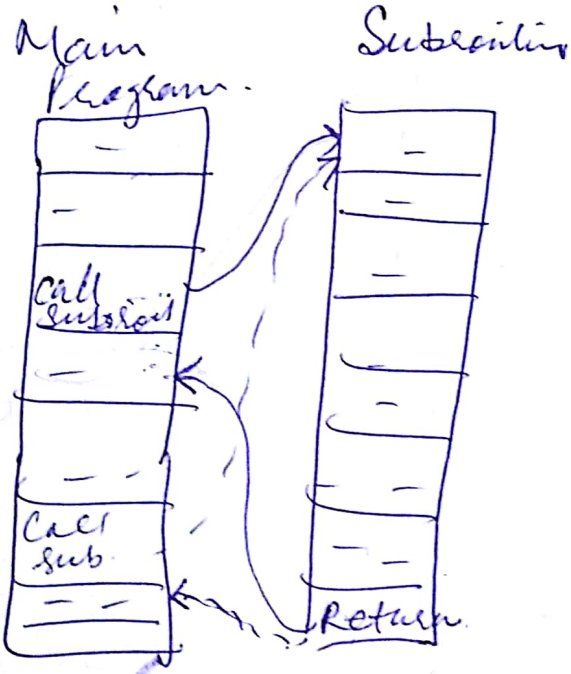
Return (RTE)

→ used to return the execution to the calling routine.

→ CALL 2003H.

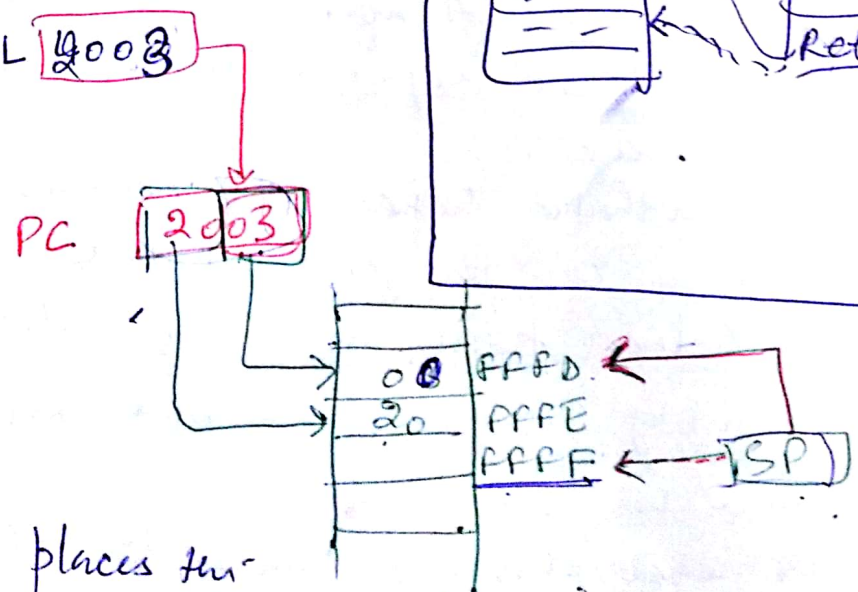
→ Push the address of the inst immediately following the CALL onto the stack.

→ Load the prog. counter with 16-bit add. supplied with the CALL inst.



2000 CALL 2003

2003

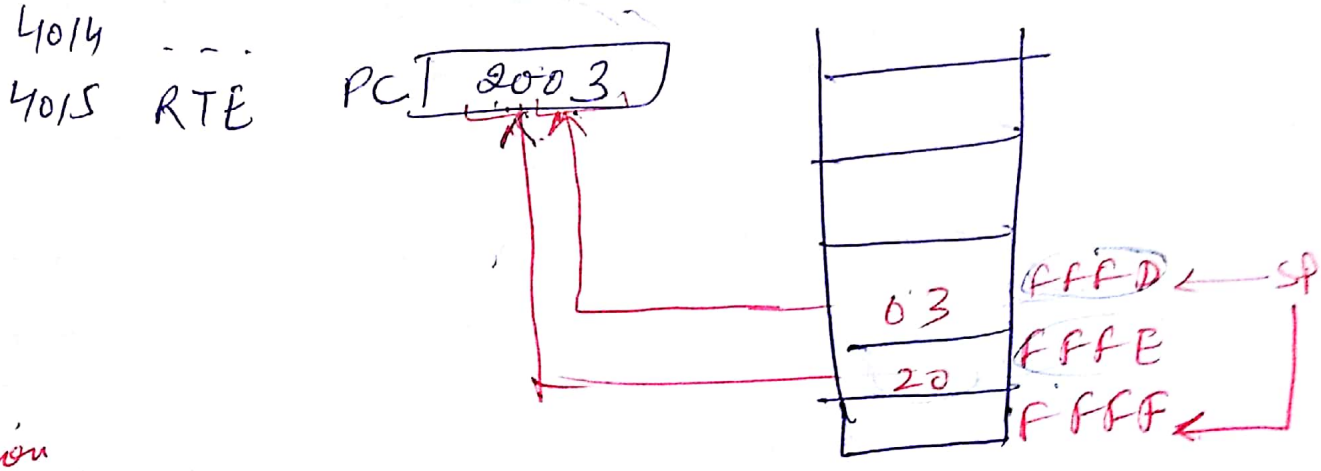


Cautions:

The CALL inst places the return address at the two mem. locations immediately before where the stack pointer is pointing.

→ Set the SP correctly before using the CALL inst.

* RTE
  → retrieve the return address from the top of stack.
  → load the PC with the return add.



Caution
The RTE inst. takes the contents of two M L at top of the stack and uses these as return address.
→ do not modify the SP in a Subroutines. Otherwise you will loose the return address.

* Passing data to a subroutines:

① data is stored in
↓
reg, by calling program and subroad. uses the values from the reg.

② The calling program stores data in
↓
M.em. location
the subroutine retrieves the data from the location and uses it.

Call by Reference

→ If the subrout performs operation on the contents of registers, then these modifications will be transferred back to the calling prog. upon returning from a subroutine.

Call by value.

If it is not desired, the subrout should PUSH all the registers it needs on the stacks on entry and POP them on return.

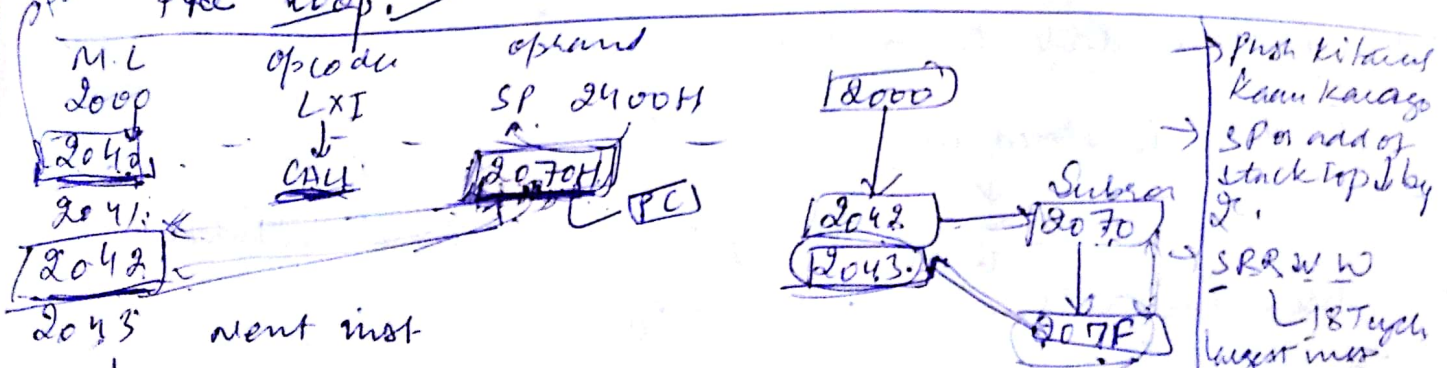
→ the original values are restored before execution returns to the calling program.

* **Cautions with PUSH & POP!**

(1) PUSH and POP should be used in opposite order.

(2) PUSH = POP.
as many.

(3) It is not advisable to use PUSH and POP inside the loop.



2070 → 1st Subr. inst

207F → RET

[call = Push & Jump]

Call 6000H → PC

Next inst. add → Stack top

6015H CALL 2019H → PC but before this 6018 is inst

6018H MOV B,C

↑ store to stack top