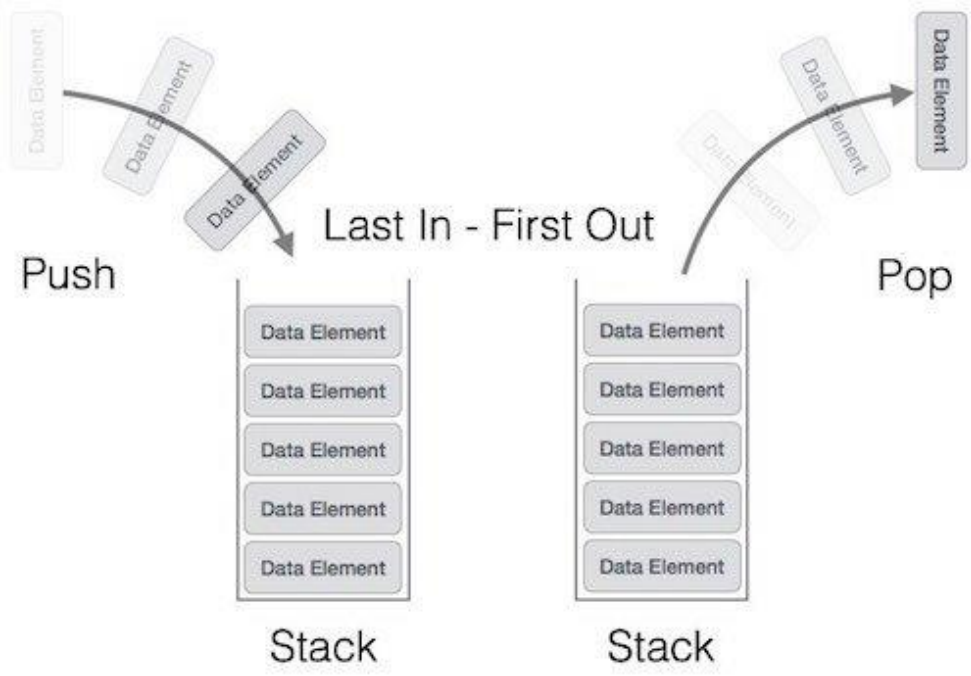
1. Stack Memory and Subroutines

1.1 subroutines

- **A subroutines** is a short sequence of instructions that perform a single task
- One advantage of using a subroutines is significant saving of memory space
- Subroutines also simplify the task of writing a program because it only appear in a program once, but are used often

1.2 stack memory

- **A stack memory** the stack is a storage device, used for storing information or data in a manner of LIFO (Last In First Out).



- **PUSH and POP instructions**: two instructions are used to data storage in stack memory which addressed by stack pointer.

- **PUSH operand** (Rp) Transferring the contents of operand (register pair) to the stack memory which beginning addressed by contents of stack pointer. Operand is one of register pair (BC, DE, HL, AF)

Exp: A- PUSH B Transferring the (BC) to stack memory. (One byte instruction)
B- PUSH D Transferring the (DE) to stack memory.

- **POP operand** transferring the contents of stack memory locations which beginning addressed by contents of stack pointer to operand. Operand is one of register pair (BC, DE, HL, AF).

Exp: A- POP B Transferring the contents of latest two stack memory locations to BC.
B- POP D Transferring the contents of latest two stack memory locations to DE.

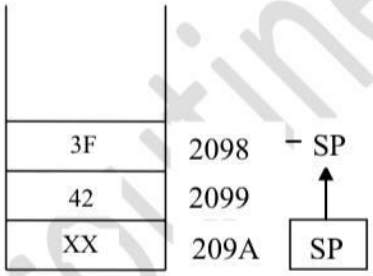
Example: write the contents of HL register and (SP) after execute the following instructions

```
LXI SP,209A
LXI H,423F
PUSH H
HLT
```

Solution:

First (**SP**)=209A (H)=42 (L)=3F

There for (SP)=2098 (H)=42 L=3F



2. Call instruction

- The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

- When 8085 executes a CALL instruction, two events occur:

1. The contents of PC are pushed onto the stack
2. The program continues at the address stored with CALL instruction.

2.1 Unconditional Call Instructions

CALL address: Call unconditionally a subroutine whose starting address given within the instruction and used to transfer program control to a subprogram or subroutine.

Example: CALL 2000H

2.2 Conditional Call Instructions

Conditional Calls

Instruction Code	Description	Condition for CALL
CC	Call on carry	CY=1
CNC	Call on not carry	CY=0
CP	Call on positive	S=0
CM	Call on minus	S=1
CPE	Call on parity even	P=1
CPO	Call on parity odd	P=0
CZ	Call on zero	Z=1
CNZ	Call on not zero	Z=0

3. Return instruction:

Return to the main program from subroutine

A-	**RET**	changing the execution of program from subroutine to the main program unconditionally
B-	**RC**	changing the execution of program from subroutine to the main program if **CY=1**
C-	**RNC**	changing the execution of program from subroutine to the main program if **CY=0**
D-	**RZ**	changing the execution of program from subroutine to the main program if **Z=1**
E-	**RNZ**	changing the execution of program from subroutine to the main program if **Z=0**
F-	**RPE**	changing the execution of program from subroutine to the main program if **P=1**
G-	**RPO**	changing the execution of program from subroutine to the main program if **P=1**
H-	**RP**	changing the execution of program from subroutine to the main program if **S=0**
I-	**RM**	changing the execution of program from subroutine to the main program if **S=1**