## (1) What is stack? Explain stack related instruction with example OR Give function of stack. OR What is stack? Explain the stack operations using examples.

→ The stack is a group of memory location in the R/W memory (RAM) that is used for temporary storage of data during the execution of a program.

→ Address of the stack is stored into the stack pointer register.

→ The 8085 provide two instructions PUSH & POP for storing information on the stack and reading it back.

- Data in the register pairs stored on the stack by using the instruction PUSH.
- Data is read from the stack by using the instruction POP.
- PUSH & POP both instruction works with register pairs only.
- The storage and retrieval of the content of registers on the stack fallows the LIFO(Last-In-First-Out) sequence.

### Operation of the stack by PUSH and POP Instruction

| 2000 | LXI SP, 2099H | ; this instruction define stack |
| 2003 | LXI H, 42F2H | ; this instruction store 42F2 in to the HL pair |
| 2006 | PUSH H | ; store HL pair on to the stack |
| 2010 | POP H | ; store data from top of the stack to HL pair |

### For PUSH H

→ The stack pointer is decremented by one to 2098H, and the contents of the h register are copied to memory location 2098H.The stack pointer register is again decremented by one to 2097H,and the contents of the L register are copied to memory location 2097H.The contents of the register pair HL are not destroyed.
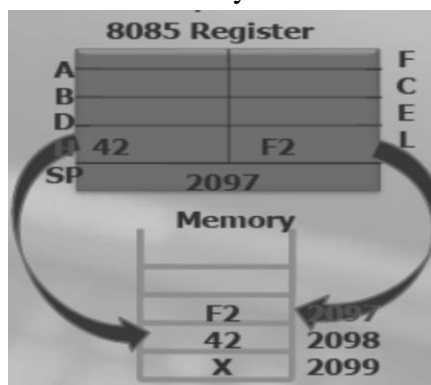
### For POP H

→ The contents of the top of the stack location shown by the stack pointer are copied in the L register and the stack pointer register is incremented by one to 2098 H. The contents of the top of the stack (now it is 2098H) are copied in the H register, and the stack pointer is incremented by one. The contents of memory location 2097H and 2098 are not destroyed until some other data bytes are stored in these location.
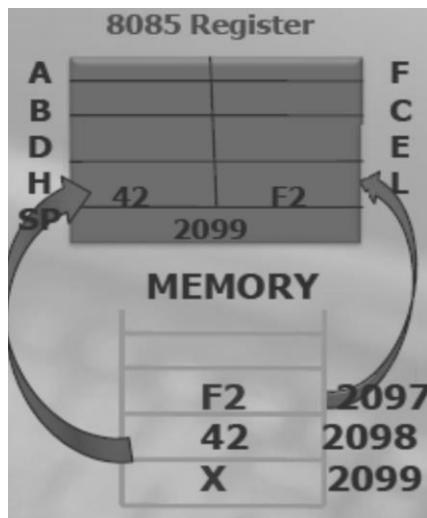


## (2) Explain Subroutine with CALL and RET Instruction.

→ A subroutine is a group of instructions that will be used repeatedly in different locations of the program. Rather than repeat the same instructions several times, they can be grouped into a one program which is called subroutine.

→ When main program calls a subroutine the program execution is transferred to the subroutine. After the completion of the subroutine, the program execution returns to the main program.

→ The microprocessor uses the stack to store the return address of the subroutine.

→ The 8085 has two instructions for dealing with subroutines.
  – The CALL instruction is used to CALL the subroutine.
  – The RET instruction is used to return to the main program at the end of the subroutine.

→ Subroutine process is shown in figure below.
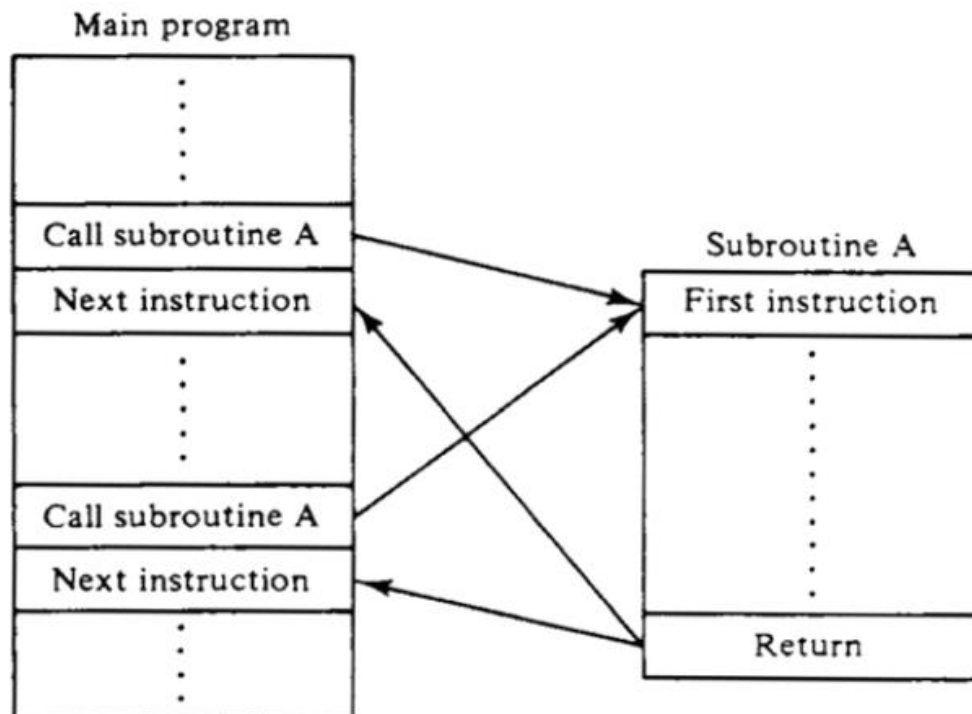


**The CALL Instruction**

CALL 16-bit address
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
Example: CALL 2034H or CALL XYZ

We can also call the subroutine by using conditional CALL instruction. For Example,

| | |
|---|---|
| CC 16-bit address | Call on if CY = 1 |
| CNC16-bit address | Call on no Carry CY = 0 |
| CP16-bit address | Call on positive S = 0 |
| CM16-bit address | Call on minus S = 1 |
| CZ 16-bit address | Call on zero Z = 1 |
| CNZ16-bit address | Call on no zero Z = 0 |
| CPE16-bit address | Call on parity even P = 1 |
| CPO16-bit address | Call on parity odd P = 0 |

**RET Instruction**

RET none

The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RET

We can also return from the subroutine by using conditional RET instruction. For Example,

RC 16-bit address     Return if CY = 1
RNC16-bit address   Return if CY = 0
RP16-bit address      Return if S = 0
RM16-bit address     Return if S = 1
RZ 16-bit address     Return if Z = 1
RNZ16-bit address    Return if Z = 0
RPE16-bit address    Return if P = 1
RPO16-bit address    Return if P = 0

## (3) Describe the looping and counting techniques. OR Explain looping, counting & indexing with an example.

The Programming Technique used to instruct the microprocessor to repeat task is called looping. This process is accomplished by using jump instructions.

A loop can be classified into two groups:

Continuous loop- repeats a task continuously

Conditional loop-repeats a task until certain data condition are met

**Continuous loop**

A continuous loop is set up by using the unconditional jump Instruction shown in the flowchart. A program with Continuous loop does not stop repeating the tasks until the system is reset.

**Conditional Loop**

A Conditional loop is setup by the conditional jump instructions. These instructions Check flags (zero, carry, etc.) and repeat the specified task if the conditions are satisfied. These loops usually include counting and indexing. Conditional loop is shown by the Flowchart as follow.
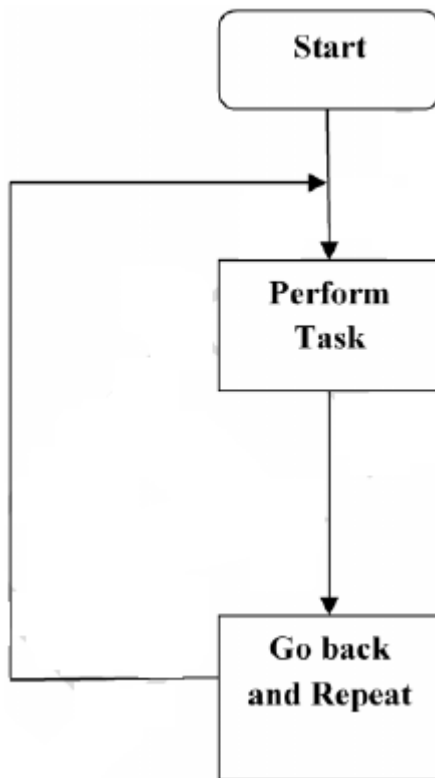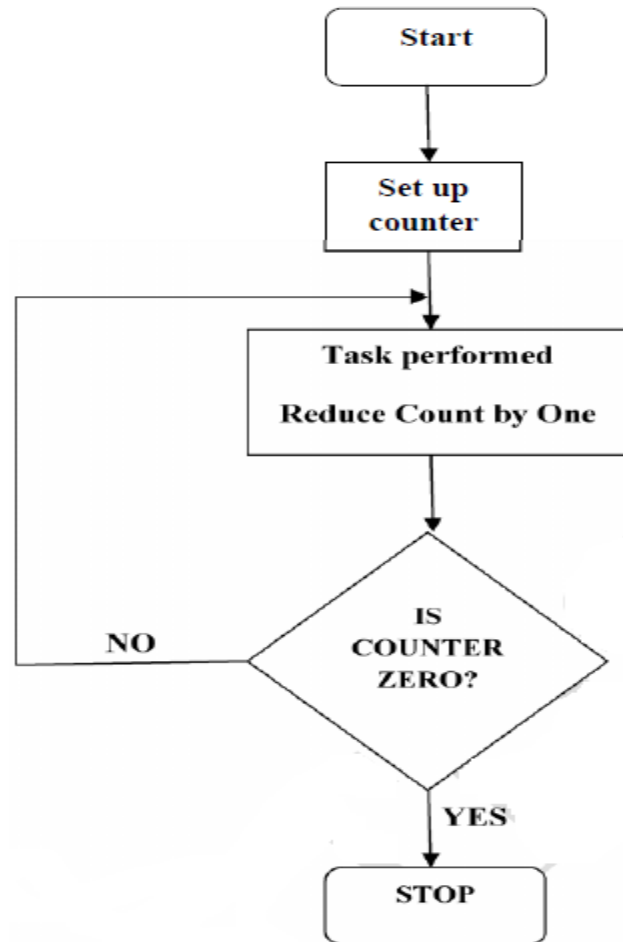
Fig: - Continuous Loop                    Fig: - Conditional Loop

The Flowchart is translated into the program as follows:

1. Counter is setup by loading an appropriate count in a register.
2. Counting is performed by either incrementing or decrementing the counter.
3. Loop is set up by a conditional jump instruction.
4. End of counting is indicated by a flag.

## (4) Explain MACRO with Example.

➢ **Macros:** A macro is a group of repetitive instructions in a program which are codified only once and can be used as many times as necessary.

➢ At the moment when the macro is executed, each parameter is substituted by the name or value specified at the time of the call.

➢ We can say that a procedure is an extension of a determined program, while the macro is a module with specific functions which can be used by different programs.

➢ The parts which make a macro are:
   i.   Declaration of the macro.

ii.     Code of the macro.

iii.     Macro termination directive

➢ Declaration of the macro: NameOfMacro MACRO [parameter1, parameter2...]

➢ Even though we have the functionality of the parameters it is possible to create a macro which does not need them.

➢ The directive for the termination of the macro is: ENDM

➢ An example of a macro, to place the cursor on a determined position on the screen is:

➢ Position MACRO Row, Column

    PUSH AX

    PUSH BX

    PUSH DX

    MOV AH, 02H

    MOV DH, Row

    MOV DL, Column

    MOV BH, 0

    INT 10H

    POP DX

    POP BX

    POP AX

    ENDM

➢ To use a macro it is only necessary to call it by its name, as if it were another assembler instruction, since directives are no longer necessary as in the case of the procedures.

    Example: Position 8, 6.

### Macro Libraries

➢ One of the facilities that the use of macros offers is the creation of libraries, which are groups of macros which can be included in a program from a different file.

➢ The creation of these libraries is very simple, we only have to write a file with all the macros which will be needed and save it as a text file.

➢ To call these macros it is only necessary to use the following instruction Include NameOfTheFile, on the part of our program where we would normally write the macros, this is, at the beginning of our program, before the declaration of the memory model.