Stack and Subroutines

Stack is a set of memory locations in the Read/Write memory which is used for temporary storage of binary information during the execution of a program. It is implemented in the Last-in-first-out (LIFO) manner. i.e., the data written first can be accessed last, One can put the data on the top of the stack by a special operation known as PUSH. Data can be read or taken out from the top of the stack by another special instruction known as POP.

Stack is implemented in two ways. In the first case, a set of registers is arranged in a shift register organization. One can PUSH or POP data from the top register. The whole block of data moves up or down as a result of push and pop operations respectively. In the second case, a block of RAM area is allocated to the stack. A special purpose register known as stack pointer (SP) points to the top of the stack. Whenever the stack is empty, it points to the bottom address. If a PUSH operation is performed, the data are stored at the location pointed to by SP and it is

decremented by one. Similarly if the POP operation is performed, the data are taken out of the location pointed at by SP and SP is incremented by one. In this case the data do not move but SP is incremented or decremented as a result of push or pop operations respectively.

Application of Stack: Stack provides a powerful data structure which has applications in many situations. The main advantage of the stack is that,

We can store data (PUSH) in it with out destroying previously stored data. This is not true in the case of other registers and memory locations.

stack operations are also very fast

The stack may also be used for storing local variables of subroutine and for the transfer of parameter addresses to a subroutine. This facilitates the implementation of re-entrant subroutines which is a very important software property.

The disadvantage is, as the stack has no fixed address, it is difficult to debug and document a program that uses stack.

Stack operation: Operations on stack are performed using the two instructions namely PUSH and POP. The contents of the stack are moved to certain memory locations after PUSH instruction. Similarly, the contents of the memory are transferred back to registers by POP instruction.

For example let us consider a Stack whose stack top is 4506 H. This is stored in the 16-bit Stack pointer register as shown in Fig.29

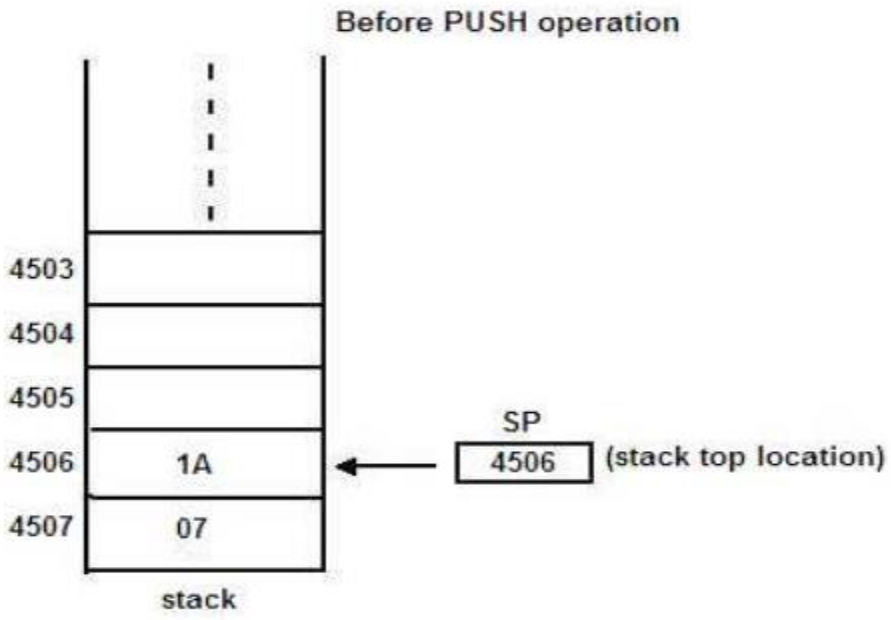
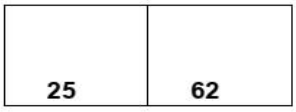


Figure.29 The PUSH operation of the Stack

Let us consider two registers (register pair) B & C whose contents are 25 & 62.

Reg. B Reg. C

After PUSH  **operation the status of the Stack is as shown in Fig 3.30**

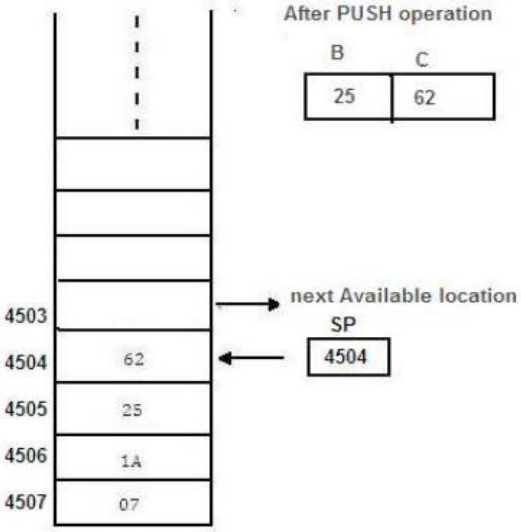


Figure .30 After PUSH operation the status of the stack

Let us now consider POP operation: The Figs 31 & 32 explains before and after the POP operation in detail

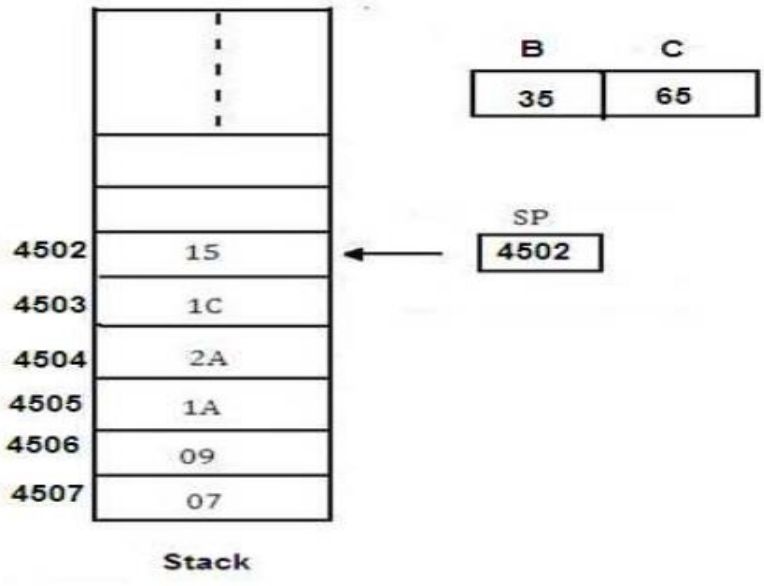


Figure 31 The POP operation of the Stack

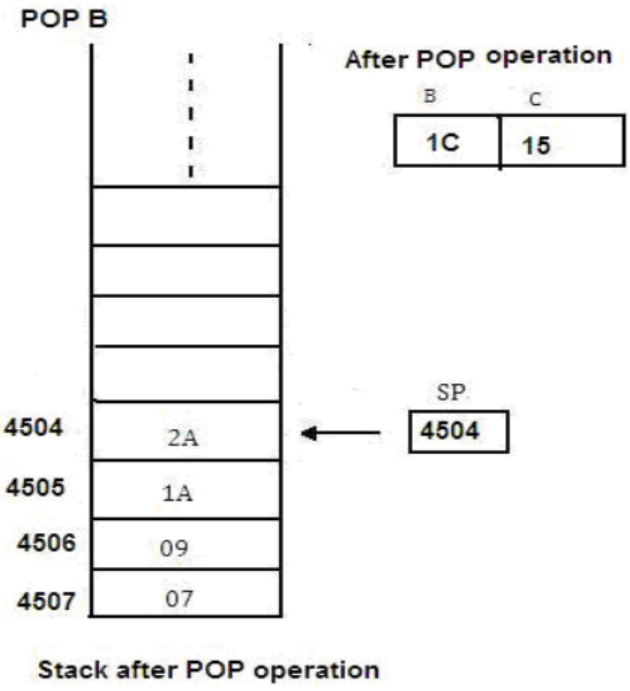


Figure 32 After POP operation the status of the stack

Before the operation the data 15 and 1C are in the locations 4502 & 4503 and after the pop operation the data is copied to B-C pair and now the SP register points to 4504 location. This is shown in Fig.3.32

Programming Example FOR PUSH & POP

Write a program to initialize the stack pointer (SP) and store the contents of the register pair H-L on stack by using PUSH instruction. Use the contents of the register pair for delay counter and at the end of the delay retrieve the contents of H-L using POP.

Memory Location	Label	Mnemonics	Operand	Comments
8000		LXI	SP, 4506 H	Initialize Stack pointer
8003		LXI	H,2565 H	
8006		PUSH	H	
8007				
.		DELAY	CALL	Push the
.		.	.	contents.
.		.	.	
.		.	.	
8.00A		POP	H	

Subroutine: It is a set of instructions written separately from the main program to execute a function that occurs repeatedly in the main program.

For example, let us assume that a delay is needed three times in a program. Writing delay programs for three times in a main program is nothing but repetition. So, we can write a subroutine program called 'delay' and can be called any number of times we need

Similarly, in 8085 microprocessor we do not find the instructions for multiplication and division. For this purpose we write separate programs. So, in any main program if these operations are needed more than once, the entire program will become lengthy and complex. So, we write subroutine programs MUL & DIV separately from main program and use the instruction CALL MUL (or) CALL DIV in the main program. This can be done any number of times. At the end of every subroutine program there must be an instruction called 'RET'. This will take the control back to main program.

The 8085 microprocessor has two instructions to implement the subroutines. They are CALL and RET. The CALL instruction is used in the main program to call a subroutine and RET instruction is used at the end of the subroutine to return to the main program. When a subroutine is called, the contents of the program counter, which is the address of the instruction following the CALL instruction is stored on the stack and the program execution is transferred to the subroutine address. When the RET instruction is executed at the end of the subroutine, the memory address stored on the stack is retrieved and the sequence of execution is resumed in the main program.

Diagrammatic representation

Let us assume that the execution of the main program started at 8000 H. It continues until a CALL subroutine instruction at 8020 H is encountered. Then the program execution transfers to 8070 H. At the end of the subroutine 807B H. The RET instruction is present. After executing this RET, it comes back to main program at 8021 H as shown in the following Fig. 33

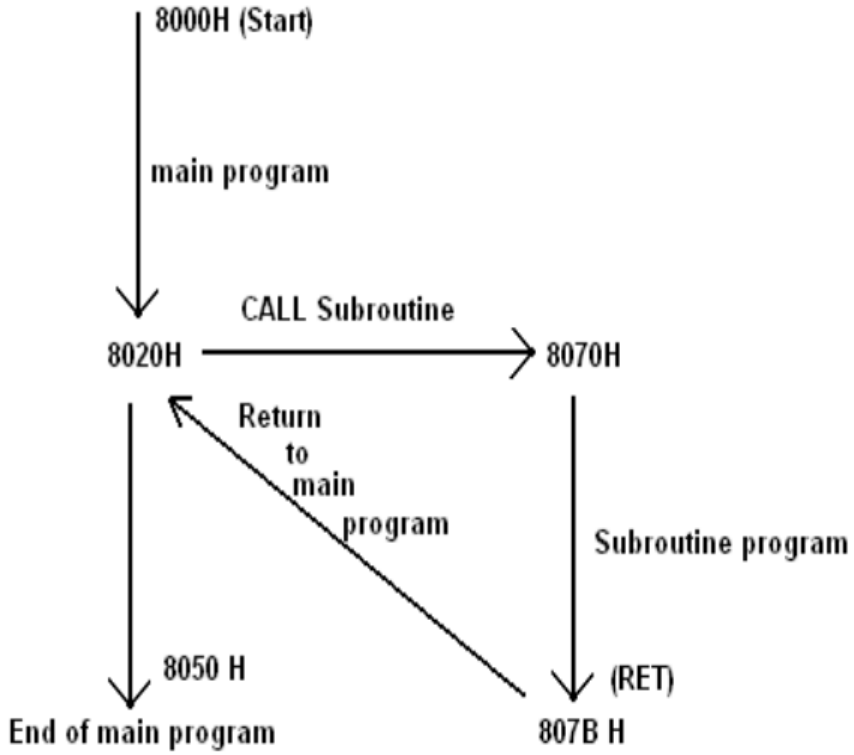


Fig.33 Diagrammatic representation of subroutine program execution

The same is explained using the assembly language program example.

Program Example:

Memory Address	Mnemonics	Operand	Comments
8000 | | | 8020	LXI	SP, 8400 H	Initialize the Stack pointer at 8400 H
8021 8022	CALL	8070 H	Call a subroutine program stored at the location 8070 H. (It is a three by Instruction)
8023 | | | 802F	Next instruction | | | HLT		The address of the next instruction following CALL instruction. End of the main program .

8070	Instructions		Beginning of the Subroutine.	Subroutine Program:
|				
|				Delay programs:
|				
807B	RET		End of the program	
807C	Next Subroutine		Instructions of next subroutine if any	
807F	RET		End of the subroutine.	

In many situations it may be desired to provide some delay between the execution of two instructions by a microprocessor. The delay can be produced by either hardware chip like 8253 or by writing a software program using registers of the processor. Here we will discuss the software delay program. This delay program is not a part of the main program. Hence it is called delay sub-routine program. For small delays we can use only one register. But for longer delays one has to use two or three registers. The technique involved here is, a register is loaded with a number and then decremented by using the instruction DCR until it becomes zero. The time of execution of the microprocessor is equal to the delay time produced.

For example, we have constructed a display system where the LEDs receive the input from a microprocessor. Since the microprocessor is a very fast device it sends the signal at very high speeds there by our eye cannot recognize the display pattern. So, if you provide some delay between two input signals, the display can be visualized clearly. Similarly to observe the rotations of a stepper motor, a delay is needed between every two excitation signals applied to the motor.

Delay Subroutine with one register:

Program

Address	Label	Machine code	Mnemonics	Operand	Comments
9000			MVI	A, FF	Get FF in register A

9002	**LOOP**		**DCR**	**A**	**Decrement register A.**
9003			**JNZ**	**LOOP**	**Has the content of register B becomes zero? No, jump to LOOP. Yes, proceed ahead.**
9006			**RET**		**Return to main program**

Calculation of Delay time for the above program:

In the above program register A is loaded by FFH B(255 decimal) and it is decremented in a loop until it becomes zero. The delay produced by this program is as follows

We should know the number of times each instruction of the above program is being executed. The number of states required for the execution of each instruction is as follows:

Instructions	States
MVI A, FFH	**7**
(loop) DCR A	**4**
JNZ loop	**7/10**
RET	**10**

Instruction	No. of times the Instruction is executed	states	Total states
MVI A, FF	1	7x1	7
loop DCR A	255	4x255	1020
JNZ loop	255	(10x254)+(7x1)	2547
RET	1	10x1	10

3584 States

Total T States=3584

The time required for one T-state in INTEL 8085 microprocessor is nearly 330n.sec

Delay time is= 3584 x 333n.sec

= 3.584 x 0.333 x 10^{-3} seconds

= 1.18272 x 10^{-3} seconds

= 1. 193472 milliseconds

Delay Subroutine with two registers

Program:

Address	Label	Machine Code	Mnemonic	Operand	Comments
8400			MVI	B, 10H	Get desired number in register B
8402	LOOP1		MVI	C, 56H	Get desired number in register
8404	LOOP2		DCR	C	Decrement C.
8405			JNZ	LOOP2	Is [C] zero? No, go to LOOP2. Yes, proceed further
8408			DCR	B	Decrement register B
8409			JNZ	LOOP1	Is [B] zero? No, go to LOOP1. Yes, proceed further
840C			RET		Return to main program.

Delay Subroutine using register pair

Program:

Address	Label	Machine Code	Mnemonic	Operand	Comments
8000			LXI	D, FFFF	Get FFFF in register pair D-E
	LOOP		DCX	D	Decrement count

			MOV	A, D	Move the contents of register D to accumulator
			ORA	E	Check if D and E are zero.
			JNZ	LOOP	If D-E is not zero, jump to LOOP
			RET		Return to main program

Delay Subroutine using three registers

Program:

Address	Label	Machine Code	Mnemonic	Operand	Comments
8400			MVI	A, 98H	Get control word
8402			OUT	03	Initialize port foe LED Display
8404			MVI	B, 50H	
8406			MVI	C, FFH	
8408			MVI	D, FFH	
840A			DCR	D	Delay Subroutine with three registers
840B			JNZ	LOOP3	
840E			DCR	C	
840F			JNZ	LOOP2	
8412			DCR	B	
8413			JNZ	LOOP1	
8416			MVI	A, 01	
8418			OUT	01	Output for LED
8419			HLT		Stop.

From the above discussion it is clear that with increase of T-states required for a delay subroutine ,the delay time also increases.