This chapter includes two illustrative programs: The first illustrates the use of the stack-related instructions to examine and manipulate the flags; and the second illustrates the subroutine technique in a traffic-signal controller.

OBJECTIVES

- ☐ Define the stack, the stack pointer (register), and the program counter, and describe their uses.
- ☐ Explain how information is stored and retrieved from the stack using the instructions PUSH and POP and the stack pointer register.
- ☐ Demonstrate how the contents of the flag register can be displayed and how a given flag can be set or reset.
- ☐ Define a subroutine and explain its uses.
- ☐ Explain the sequence of a program execution when a subroutine is called and executed.
- ☐ Explain how information is exchanged between the program counter and the stack, and identify the contents of the stack pointer register when a subroutine is called.
- ☐ List and explain conditional Call and Return instructions.
- ☐ Illustrate the concepts in the following subroutines: multiple calling, nesting, and common ending.
- ☐ Compare similarities and differences between PUSH/POP and CALL/RET instructions.

9.1 STACK

The **stack** in an 8085 microcomputer system can be described as a set of memory locations in the R/W memory, specified by a programmer in a main program. These memory locations are used to store binary information (bytes) temporarily during the execution of a program.

The beginning of the stack is defined in the program by using the instruction LXI SP, which loads a 16-bit memory address in the stack pointer register of the microprocessor. Once the stack location is defined, storing of data bytes begins at the memory address that is one less than the address in the stack pointer register. For example, if the stack pointer register is loaded with the memory address 2099H (LXI SP,2099H), the storing of data bytes begins at 2098H and continues in reversed numerical order (decreasing memory addresses such as 2098H, 2097H, etc.). Therefore, as a general practice, the stack is initialized at the highest available memory location to prevent the program from being destroyed by the stack information. The size of the stack is limited only by the available memory.

Data bytes in the register pairs of the microprocessor can be stored on the stack (two at a time) in reverse order (decreasing memory address) by using the instruction PUSH. Data bytes can be transferred from the stack to respective registers by using the instruction POP. The stack pointer register tracks the storage and retrieval of the information. Because two data bytes are being stored at a time, the 16-bit memory address in the stack pointer register is decremented by two; when data bytes are retrieved, the address is incremented by two. An address in the stack pointer register indicates that the next two memory locations (in descending numerical order) can be used for storage.

The stack is shared by the programmer and the microprocessor. The programmer can store and retrieve the contents of a register pair by using PUSH and POP instructions. Similarly, the microprocessor automatically stores the contents of the program counter when a subroutine is called (to be discussed in the next section). The instructions necessary for using the stack are explained below.

INSTRUCTIONS

Opcode	Operand	
LXI	SP,16-bit	☐ Load Stack Pointer ☐ Load the stack pointer register with a 16-bit address. The LXI instructions were discussed in Chapter 7
PUSH	Rp	Store Register Pair on Stack ☐ This is a 1-byte instruction ☐ It copies the contents of the specified register pair on the stack as described below ☐ The stack pointer register is decremented, and the contents of the high-order register (e.g., register B) are copied in the location shown by the stack pointer register ☐ The stack pointer register is again decremented, and the contents of the low-order register (e.g., register C) are copied in that location ☐ The operands B, D, and H represent register pairs BC, DE, and HL, respectively ☐ The operand PSW represents Program Status Word, meaning the contents of the accumulator and the flags
PUSH	B	
PUSH	D	
PUSH	H	
PUSH	PSW	
POP	Rp	Retrieve Register Pair from Stack ☐ This is a 1-byte instruction ☐ It copies the contents of the top two memory locations of the stack into the specified register pair ☐ First, the contents of the memory location indicated by the stack pointer register are copied into the low-order register (e.g., register L), and then the stack pointer register is incremented by 1 ☐ The contents of the next memory location are copied into the high-order register (e.g., register H), and the stack pointer register is again incremented by 1
POP	B	
POP	D	
POP	H	
POP	PSW	

All three of these instructions belong to the data transfer (copy) group; thus, the contents of the source are not modified, and no flags are affected.

Example 9.1

In the following set of instructions (illustrated in Figure 9.1), the stack pointer is initialized, and the contents of register pair HL are stored on the stack by using the PUSH instruction. Register pair HL is used for the delay counter (actual instructions are not

shown); at the end of the delay counter, the contents of HL are retrieved by using the instruction POP. Assuming the available user memory ranges from 2000H to 20FFH, illustrate the contents of various registers when PUSH and POP instructions are executed.

Solution

In this example, the first instruction—LXI SP,2099H—loads the stack pointer register with the address 2099H (Figure 9.1). This instruction indicates to the microprocessor that memory space is reserved in the R/W memory as the stack and that the locations beginning at 2098H and moving upward can be used for temporary storage. This instruction also suggests that the stack can be initialized anywhere in the memory; however, the stack location should not interfere with a program. The next instruction—LXI H—loads data in the HL register pair, as shown in Figure 9.1.

When instruction PUSH H is executed, the following sequence of data transfer takes place. After the execution, the contents of the stack and the register are as shown in Figure 9.2.

1. The stack pointer is decremented by one to 2098H, and the contents of the H register are copied to memory location 2098H.
2. The stack pointer register is again decremented by one to 2097H, and the contents of the L register are copied to memory location 2097H.
3. The contents of the register pair HL are not destroyed; however, HL is made available for the delay counter.

After the delay counter, the instruction POP H restores the original contents of the register pair HL, as follows. Figure 9.3 illustrates the contents of the stack and the registers following the POP instruction.

1. 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 2098H.
2. 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.

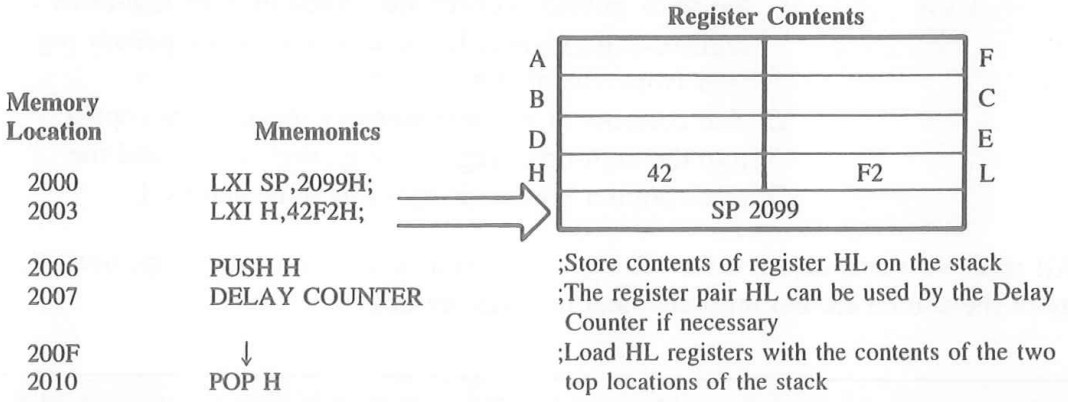


FIGURE 9.1
Instructions and Register Contents in Example 9.1

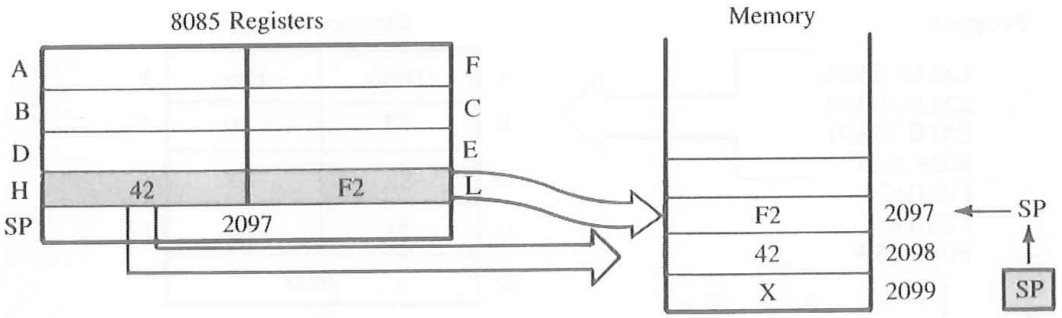


FIGURE 9.2
Contents on the Stack and in the Registers After the PUSH Instruction

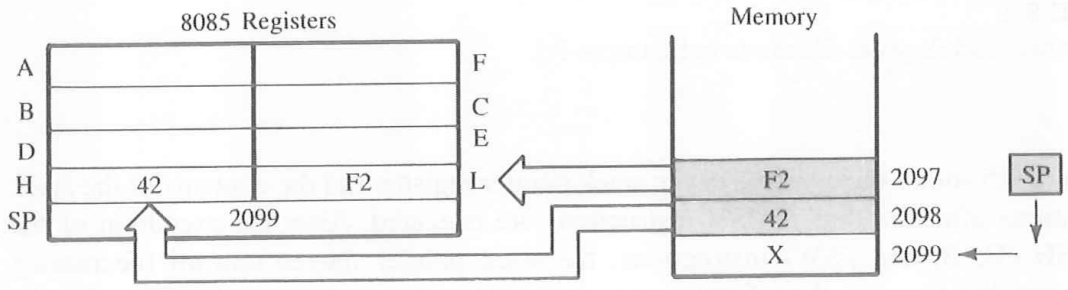


FIGURE 9.3
Contents on the Stack and in the Registers After the POP Instruction

3. The contents of memory locations 2097H and 2098H are not destroyed until some other data bytes are stored in these locations.

Example 9.2

The available user memory ranges from 2000H to 23FFH. A program of data transfer and arithmetic operations is stored in memory locations from 2000H to 2050H, and the stack pointer is initialized at location 2400H. Two sets of data are stored, starting at locations 2150H and 2280H (not shown in Figure 9.4). Registers HL and BC are used as memory pointers to the data locations. A segment of the program is shown in Figure 9.4.

1. Explain how the stack pointer can be initialized at one memory location beyond the available user memory.
2. Illustrate the contents of the stack memory and registers when PUSH and POP instructions are executed, and explain how memory pointers are exchanged.
3. Explain the various contents of the user memory.

Solution

1. The program initializes the stack pointer register at location 2400H, one location beyond the user memory (Figure 9.4). This procedure is valid because the initialized location is never used for storing information. The instruction PUSH first decrements the stack pointer register, and then stores a data byte.

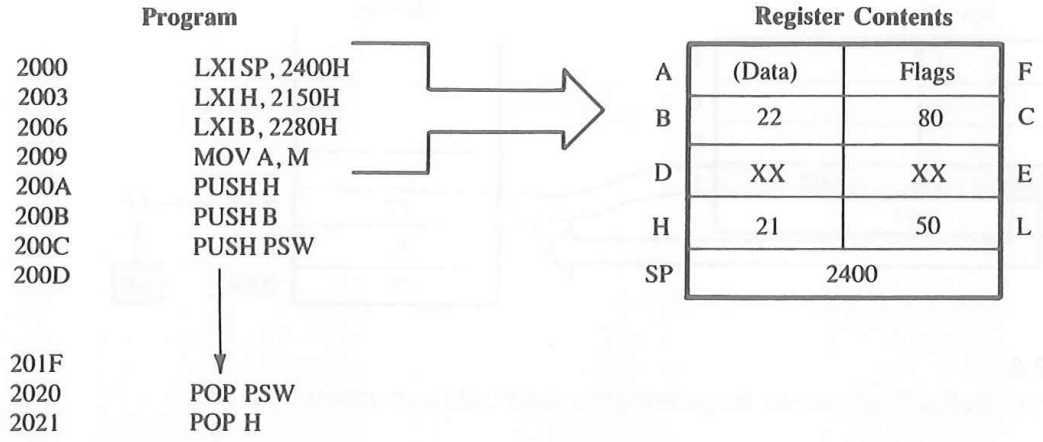


FIGURE 9.4
Instructions and Register Contents in Example 9.2

2. Figure 9.5 shows the contents of the stack pointer register and the contents of the stack locations after the three PUSH instructions are executed. After the execution of the PUSH (H, B, and PSW) instructions, the stack pointer moves upward (decreasing memory locations) as the information is stored. Thus the stack can grow upward in the user memory even to the extent of destroying the program.

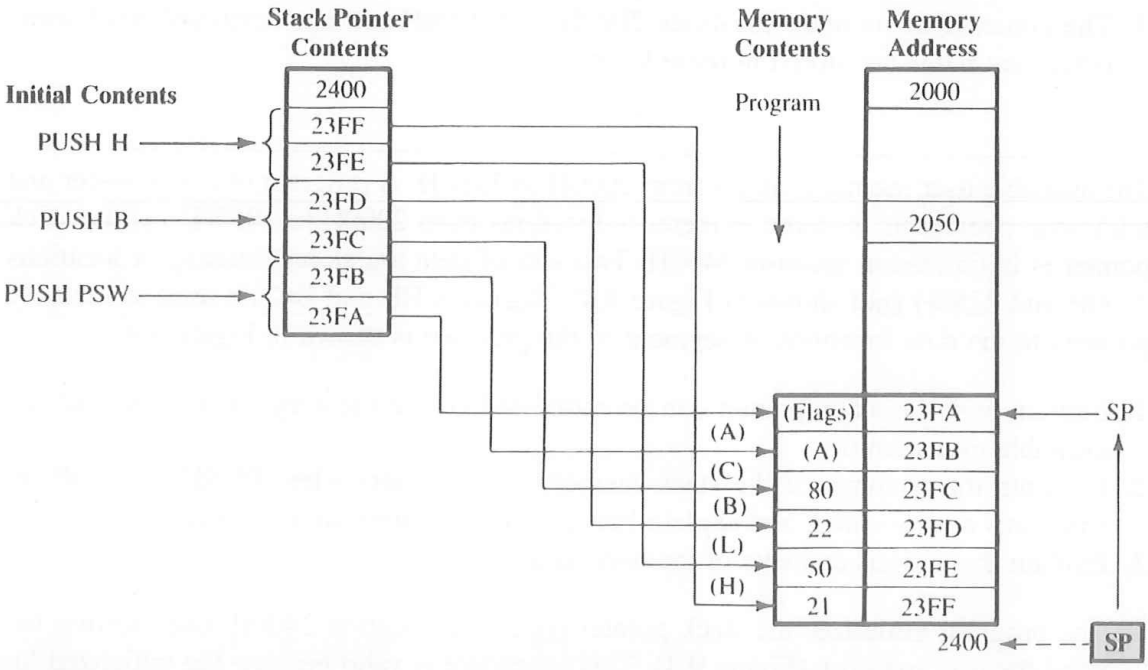


FIGURE 9.5
Stack Contents After the Execution of PUSH Instructions

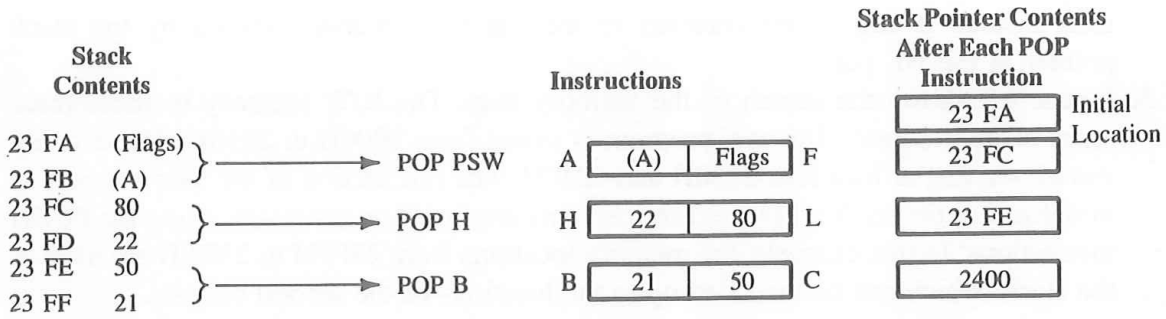
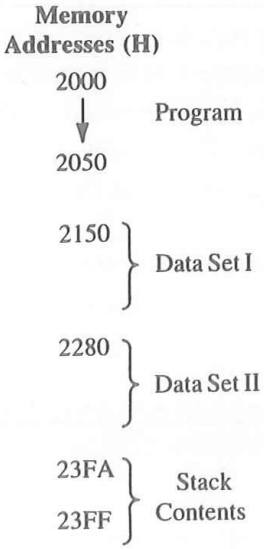


FIGURE 9.6
Register Contents After the Execution of POP Instructions

Figure 9.6 shows how the contents of various register pairs are retrieved. To restore the original contents in the respective registers, follow the sequence Last-In–First-Out (LIFO). In the example, the register contents were pushed on the stack in the order of HL, BC, and PSW. The contents should have been restored in the order of PSW, BC, and HL. However, the order is altered in this example to demonstrate how register contents are exchanged.

The instruction POP PSW copies the contents of the two top locations to the flag register and the accumulator, respectively, and increments the stack pointer by two to 23FCH. The next instruction, POP H, takes the contents of the top two locations (23FC and 23FD), and copies them in registers L and H, respectively, while incrementing the stack pointer by two to 23FEH. The instruction POP B copies the contents of the next two locations in registers C and B, incrementing the stack pointer to 2400H. By reversing the positions of two instructions, POP H and POP B, the contents of the BC pair are exchanged with those of the HL pair. It is important to remember that the instruction POP H does not restore the original contents of the HL

FIGURE 9.7
R/W Memory Contents



pair; instead it copies the contents of the top two locations shown by the stack pointer in the HL pair.

3. Figure 9.7 shows the sketch of the memory map. The R/W memory includes three types of information. The user program is stored from 2000H to 2050H. The data are stored, starting at locations 2150H and 2280H. The last section of the user memory is initialized as the stack where register contents are stored as necessary, using the PUSH instructions. In this example, the memory locations from 23FFH to 23FAH are used as the stack, which can be extended up to the locations of the second data set.

9.1.1 Review of Important Concepts

The following points can be summarized from the preceding examples:

1. Memory locations in R/W memory can be employed as temporary storage for information by initializing (loading) a 16-bit address in the stack pointer register; these memory locations are called the stack. The terms *stack* and *stack pointer* appear similar; however, they are not the same. The stack is memory locations in R/W memory; the stack pointer is a 16-bit register in the 8085 microprocessor.
2. Read/Write memory generally is used for three different purposes:
   a. to store programs or instructions;
   b. to store data;
   c. to store information temporarily in defined memory locations called the stack during the execution of the program.
3. The stack space grows upward in the numerically decreasing order of memory addresses.
4. The stack can be initialized anywhere in the user memory map. However, as a general practice, the stack is initialized at the highest user memory location so that it will be less likely to interfere with a program.
5. A programmer can employ the stack to store contents of register pairs by using the instruction PUSH and can restore the contents of register pairs by using the instruction POP. The address in the stack pointer register always points to the top of the stack, and the address is decremented or incremented as information is stored or retrieved.
6. The contents of the stack pointer can be interpreted as the address of the memory location that is already used for storage. The retrieval of bytes begins at the address in the stack pointer; however, the storage begins at the next memory location (in the decreasing order).
7. The storage and retrieval of data bytes on the stacks should follow the LIFO (Last-In–First-Out) sequence. Information in stack locations is not destroyed until new information is stored in those locations.

9.1.2 Illustrative Program: Resetting and Displaying Flags

PROBLEM STATEMENT

Write a program to perform the following functions:

1. Clear all the flags.
2. Load 00H in the accumulator, and demonstrate that the Zero flag is not affected by the data transfer instruction.
3. Logically OR the accumulator with itself to set the Zero flag, and display the flag at PORT1 or store all the flags on the stack.

PROBLEM ANALYSIS

The problem concerns examining the Zero flag after instructions have been executed. There is no direct way of observing the flags; however, they can be stored on the stack by using the instruction PUSH PSW. The contents of the flag register can be retrieved in any one of the registers by using the instruction POP, and the flags can be displayed at an output port. In this example, the result to be displayed is different from the result to be stored on the stack memory. To display only the Zero flag, all other flags should be masked; however, the masking is not necessary to store all the flags.

PROGRAM

Memory Address	Machine Code	Instructions	Comments
XX00	31	LXI SP,XX99H	;Initialize the stack
01	99		
02	XX		
03	2E	MVI L,00H	;Clear L
04	00		
05	E5	PUSH H	;Place (L) on stack
06	F1	POP PSW	;Clear flags
07	3E	MVI A,00H	;Load 00H
08	00		
09	F5	PUSH PSW	;Save flags on stack
0A	E1	POP H	;Retrieve flags in L
0B	7D	MOV A,L	
0C	D3	OUT PORT0	;Display flags
0D	PORT0		
0E	3E	MVI A,00H	;Load 00H again
0F	00		
10	B7	ORA A	;Set flags and reset CY, AC
11	F5	PUSH PSW	;Save flags on stack
12	E1	POP H	;Retrieve flags in L
13	7D	MOV A,L	
14	E6	ANI 40H	;Mask all flags except Z
15	40		
16	D3	OUT PORT1	
17	PORT1		
18	76	HLT	;End of program

Storing in Memory: Alternative to Output Display

```
XX0A   3E   MVI A,00H    ;Load 00H again
  0B   00
  0C   B7   ORA A         ;Set flags and reset CY and AC
  0D   F5   PUSH PSW      ;Save flags on stack
  0E   76   HLT           ;End of program
```

Program Description The stack pointer register is initialized at XX99H. The instruction MVI L clears (L), and (L) is placed on the stack, which is subsequently placed into the flag register to clear all the flags.

To verify the flags after the execution of the MVI A instruction, the PUSH and POP instructions are used in the same way as these instructions were used to clear the flags, and the flags are displayed at PORT0. Similarly, the Zero flag is displayed at PORT1 after the instructions MVI and ORA.

Program Output Data transfer (copy) instructions do not affect the flags; therefore, no flags should be set after the instruction MVI A, even if (A) is equal to zero. PORT0 should display 00H. However, the instruction ORA will set the Zero and the Parity flags to reflect the data conditions in the accumulator, and it also resets the CY and AC flags. In the flag register, bit D_6 represents the Z flag, and the ANI instruction masks all flags except the Z flag. PORT1 should display 40H as shown below.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	0	0	1	0	0	= 40H
S	Z		AC		P		CY	

Storing Output in Memory If output ports are not available, the results can be stored in the stack memory. The machine code (F5) at memory location XX09H saves the flags affected by the instruction MVI A,00H. Then the instructions can be modified starting from memory location XX0AH. The alternative set of instructions is shown above; it sets the flags (using ORA instruction), and saves them on the stack without masking. The result (44H) includes the parity flag. The contents of the stack locations should be as shown in Figure 9.8.

Instructions	Stack Memory	Contents	
	XX99	←	Stack Pointer Initialization
XX09 PUSH PSW:	XX98	(A) = 00H	
	XX97	(F) = 00H	
XX0D PUSH PSW:	XX96	(A) = 00H	
	XX95	(F) = 44H	

FIGURE 9.8
Output Stored in Stack Memory