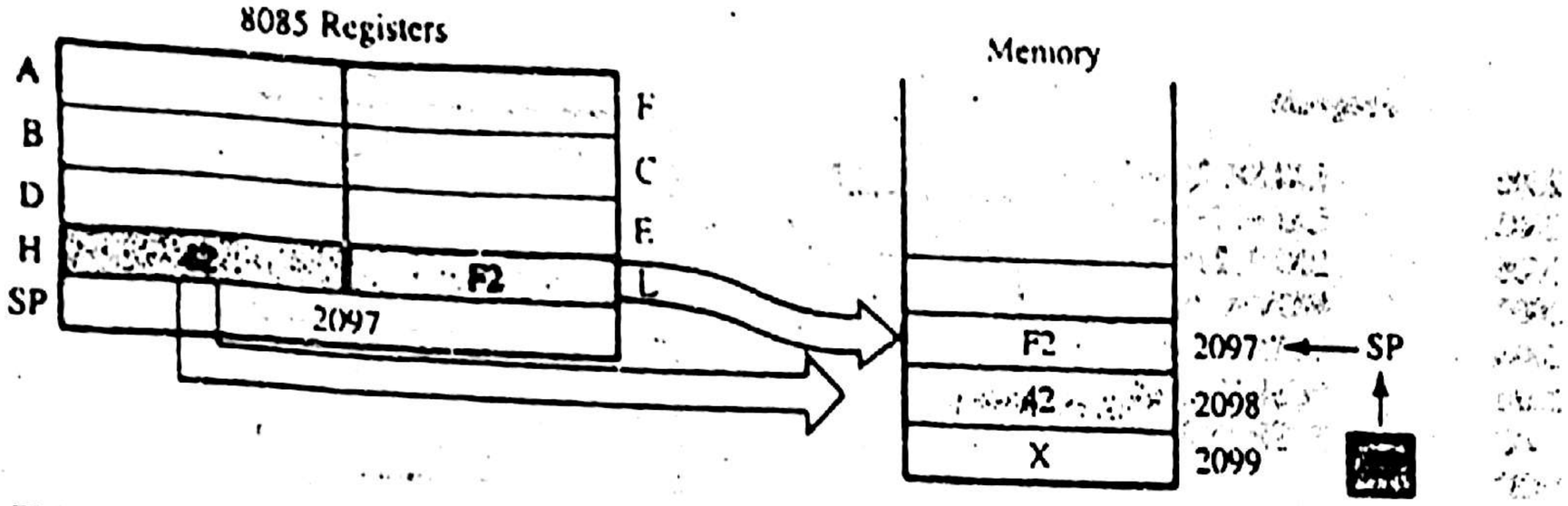


FIGURE 9.2
Contents on the Stack and in the Registers After the PUSH Instruction

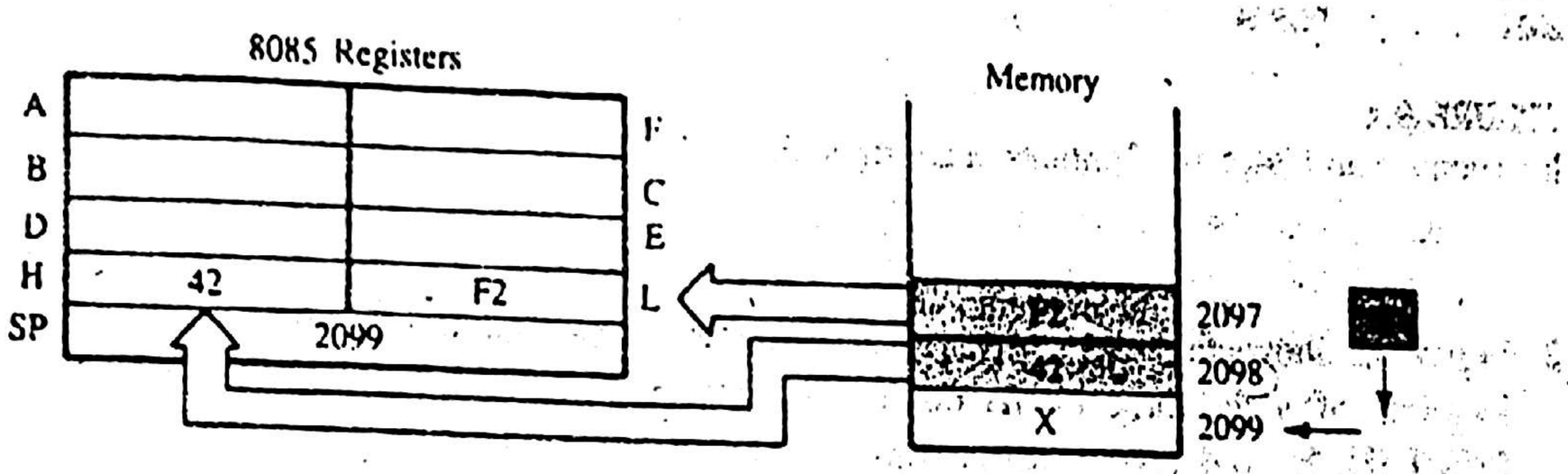


FIGURE 9.3
Contents on the Stack and in the Registers After the POP Instruction

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.

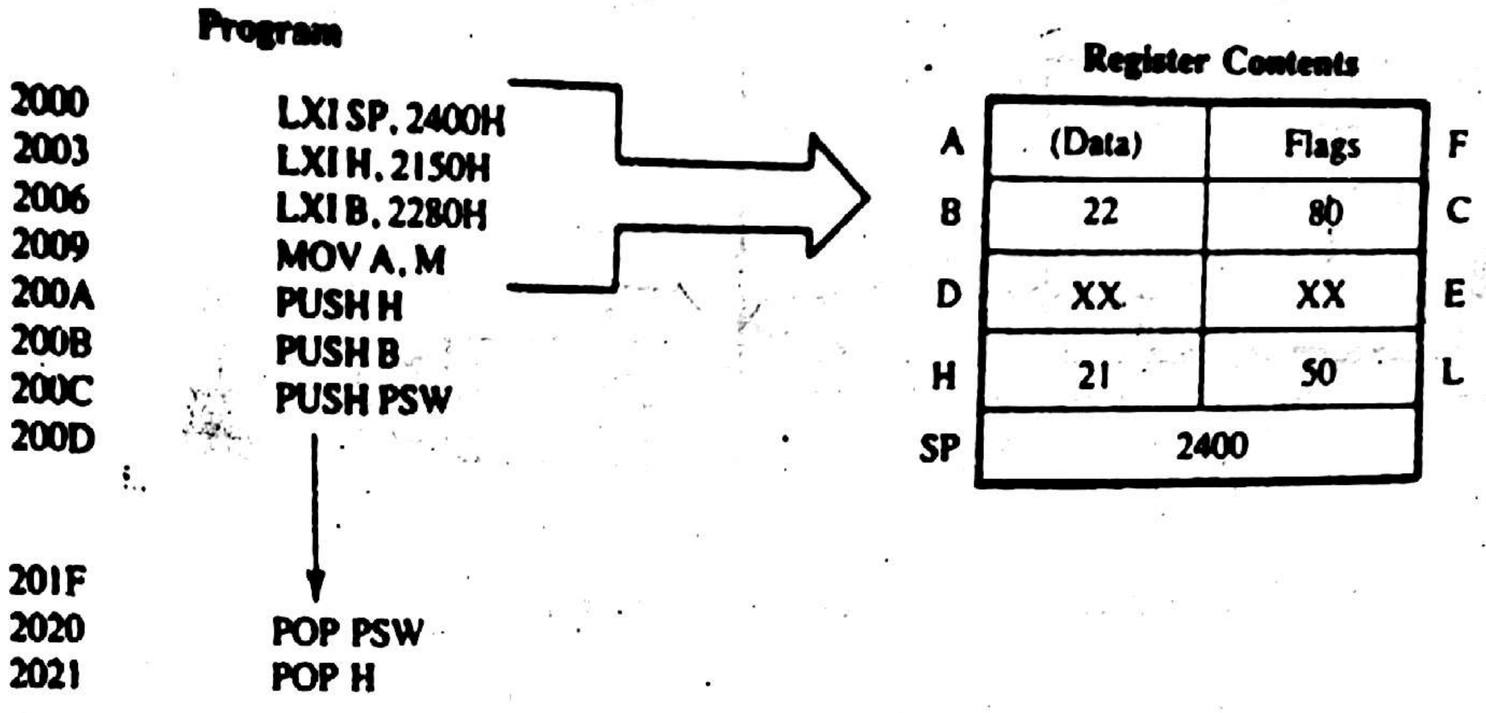


FIGURE 9.4
Instructions and Register Contents in Example 9.2

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.

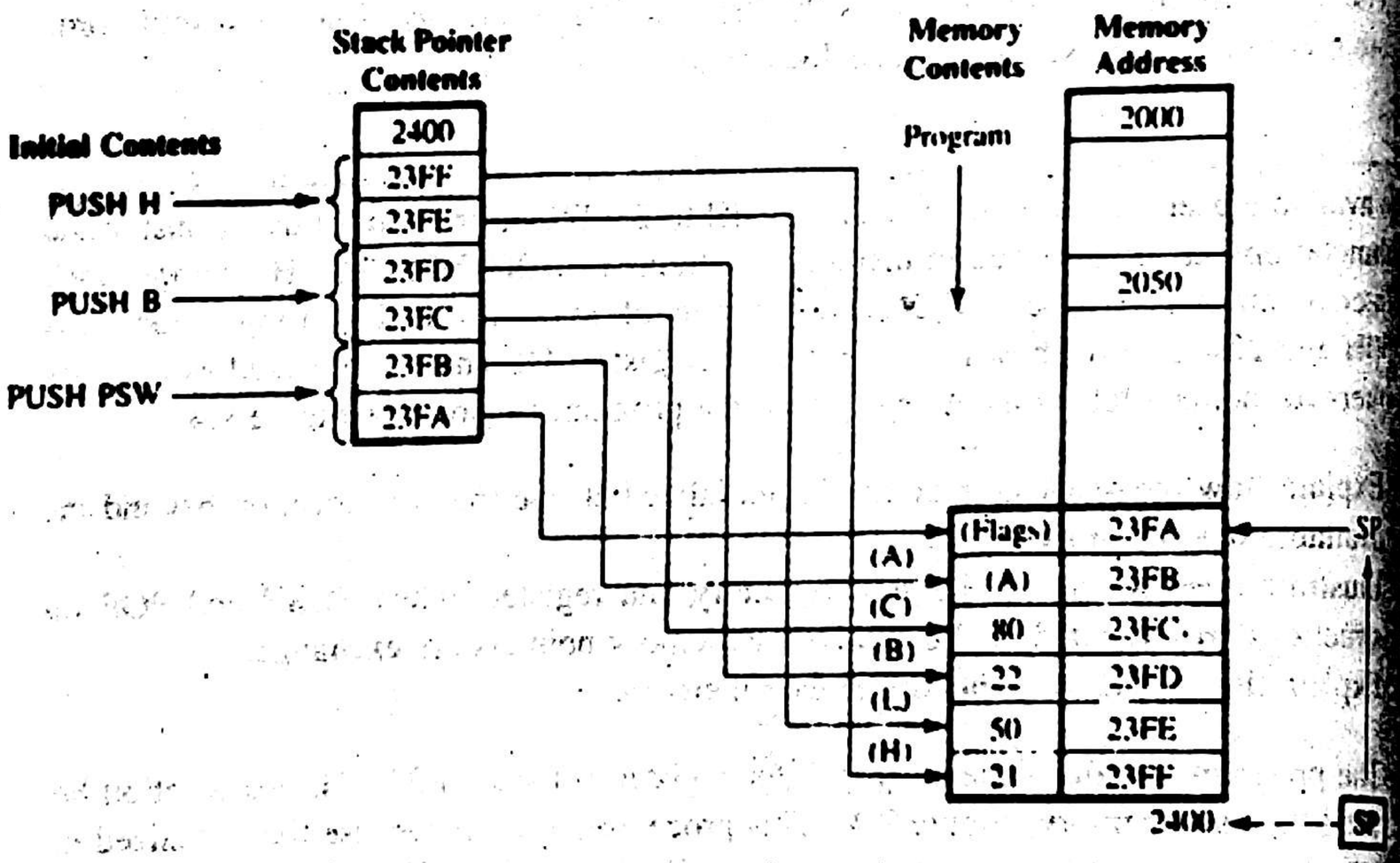


FIGURE 9.5
Stack Contents After the Execution of PUSH Instructions

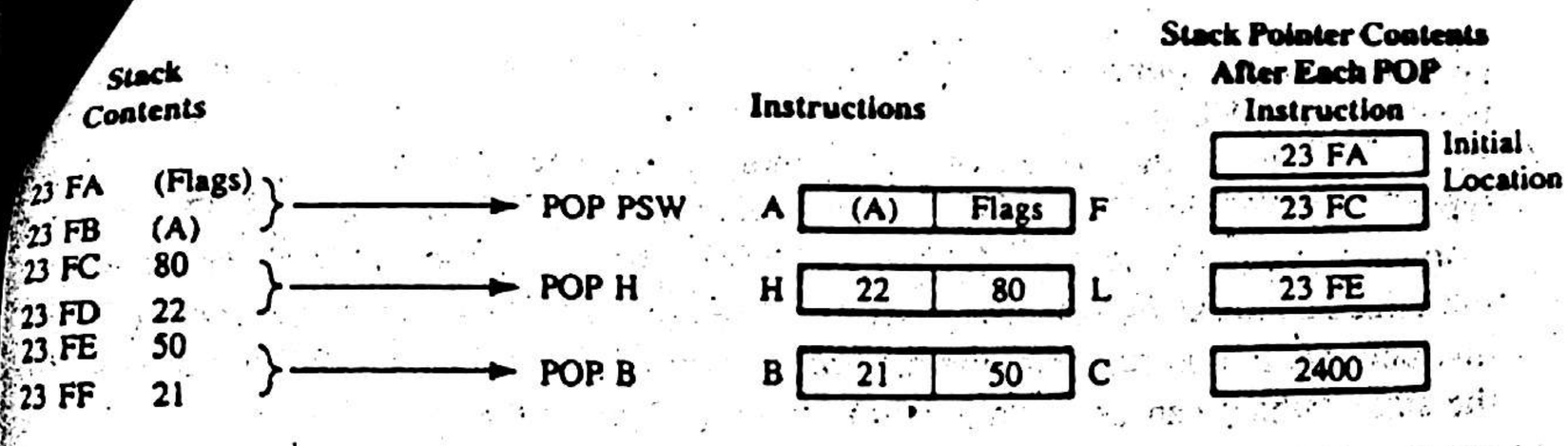
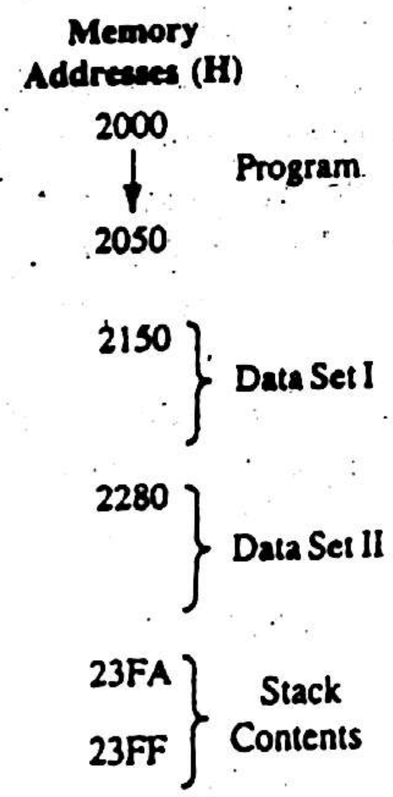


FIGURE 9.6
Register Contents After the Execution of POP Instructions

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: