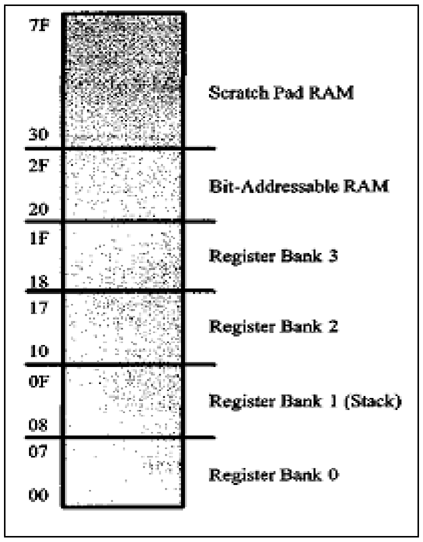
8051 REGISTER BANKS AND STACK

The 8051 microcontroller has a total of 128 bytes of RAM. In this section we discuss the allocation of these 128 bytes of RAM and examine their usage as registers and stack.

RAM memory space allocation in the 8051



There are 128 bytes of RAM in the 8051 (some members, notably the 8052, have 256 bytes of RAM). The 128 bytes of RAM inside the 8051 are assigned addresses 00 to 7FH. As we will see in Chapter 5, they can be accessed directly as memory locations. These 128 bytes are divided into three different groups as follows.

Figure 2-5. **RAM Allocation in the 8051**

1. A total of 32 bytes from locations 00 to IF hex are set aside for register banks and the stack.

2. A total of 16 bytes from locations 20H to 2FH are set aside for bit- addressable read/write memory. A detailed discussion of bit-address able memory and instructions is given in Chapter 8.

3. A total of 80 bytes from locations 30H to 7FH are used for read and write storage, or what is normally called a scratch pad. These 80 locations of RAM are widely used for the purpose of storing data and parameters by 8051 programmers. We will use them in future chapters to store data brought into the CPU via I/O ports. 1

Register banks in the 8051

As mentioned earlier, a total of 32 bytes of RAM are set aside for the register banks and stack. These 32 bytes are divided into 4 banks of registers in which each bank has 8 registers, RO – R7. RAM locations from 0 to 7 are set aside for bank 0 of RO – R7 where RO is RAM location 0, Rl is RAM location 1, R2 is location 2, and so on, until memory location 7, which belongs to R7 of bank 0. The second bank of registers RO – R7 starts at RAM location 08 and goes to location OFH. The third bank of RO – R7 starts at memory location 10H and goes to location 17H. Finally, RAM locations 18H to 1FH are set aside for the fourth bank of RO – R7. The following shows how the 32 bytes are allocated into 4 banks:

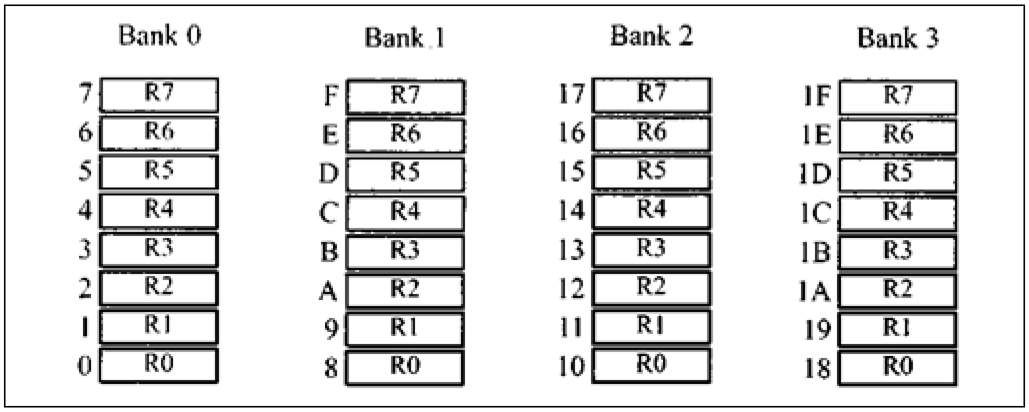
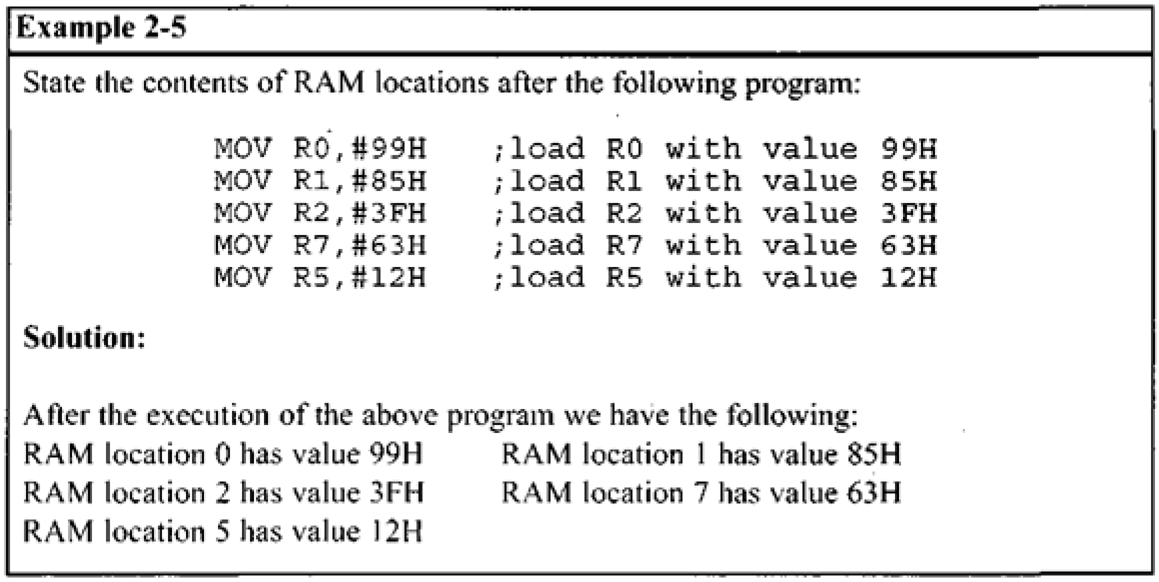


Figure 2-6. 8051 Register Banks and their RAM Addresses

As we can see from Figure 2-5, bank 1 uses the same RAM space as the stack. This is a major problem in programming the 8051. We must either not use register bank 1, or allocate another area of RAM for the stack. This will be discussed below.



Example 2-5

State the contents of RAM locations after the following program:

```
MOV  R0,#99H    ;load  R0  with  value  99H
MOV  R1,#85H    ;load  R1  with  value  85H
MOV  R2,#3FH    ;load  R2  with  value  3FH
MOV  R7,#63H    ;load  R7  with  value  63H
MOV  R5,#12H    ;load  R5  with  value  12H
```

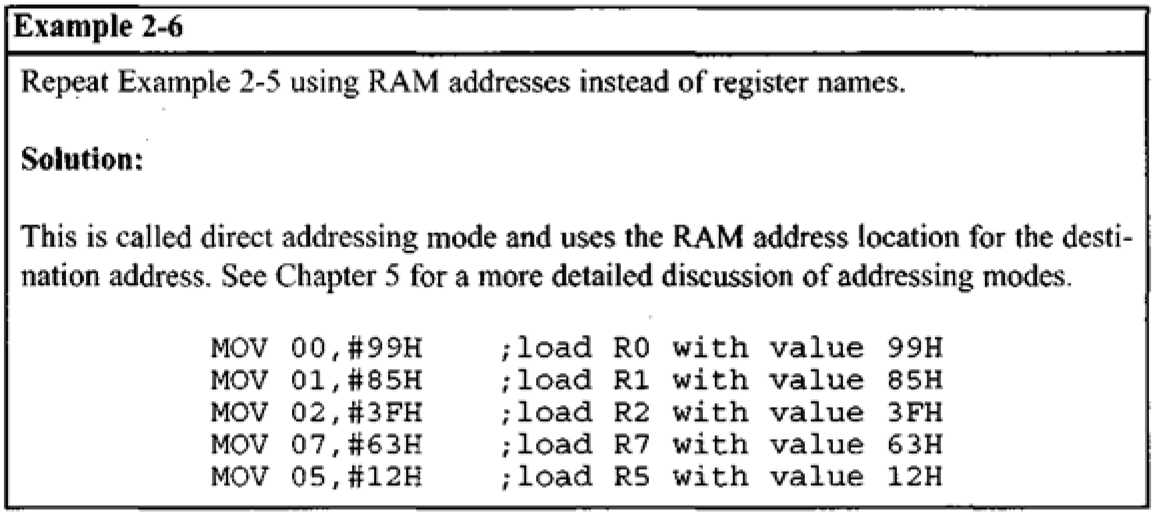
Solution:

After the execution of the above program we have the following:

RAM location 0 has value 99H	RAM location 1 has value 85H
RAM location 2 has value 3FH	RAM location 7 has value 63H
RAM location 5 has value 12H	

Default register bank

If RAM locations 00 – 1F are set aside for the four register banks, which •register bank of RO – R7 do we have access to when the 8051 is powered up? The answer is register bank 0; that is, RAM locations 0, 1,2, 3, 4, 5, 6, and 7 are accessed with the names RO, Rl, R2, R3, R4, R5, R6, and R7 when programming the 8051. It is much easier to refer to these RAM locations with names such as RO, R1, and so on, than by their memory locations. Example 2-6 clarifies this concept.



Example 2-6

Repeat Example 2-5 using RAM addresses instead of register names.

Solution:

This is called direct addressing mode and uses the RAM address location for the destination address. See Chapter 5 for a more detailed discussion of addressing modes.

```
MOV 00,#99H    ;load R0 with value 99H
MOV 01,#85H    ;load R1 with value 85H
MOV 02,#3FH    ;load R2 with value 3FH
MOV 07,#63H    ;load R7 with value 63H
MOV 05,#12H    ;load R5 with value 12H
```

How to switch register banks

As stated above, register bank 0 is the default when the 8051 is powered up. We can switch to other banks by use of the PSW (program status word) register. Bits D4 and D3 of the PSW are used to select the desired register bank as shown in Table 2-2.

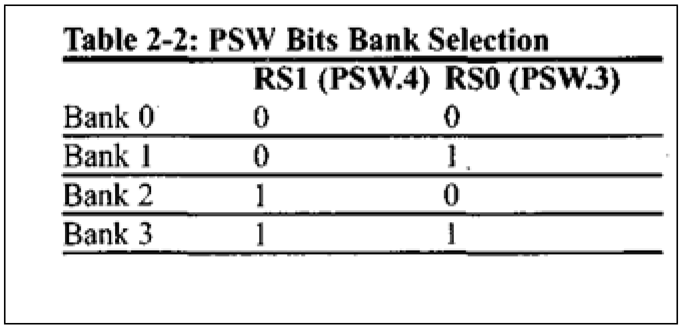
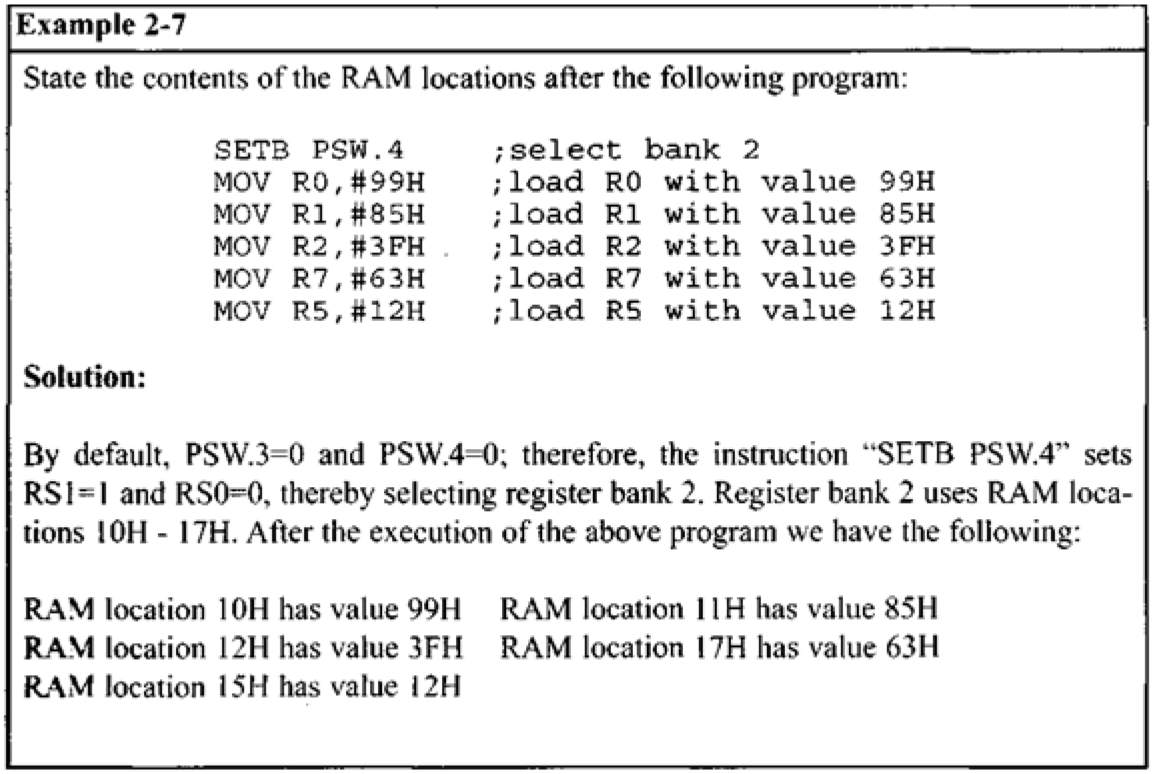


Table 2-2: PSW Bits Bank Selection

	RS1 (PSW.4)	RS0 (PSW.3)
Bank 0	0	0
Bank 1	0	1
Bank 2	1	0
Bank 3	1	1

The D3 and D4 bits of register PSW are often referred to as PSW.4 and PSW.3 since they can be accessed by the bit-addressable instructions SETB and CLR. For example, “SETB
PSW.3″ will make PSW.3 = 1 and select bank register 1. See Example 2-7.



Example 2-7

State the contents of the RAM locations after the following program:

```
SETB PSW.4      ;select bank 2
MOV R0,#99H     ;load R0 with value 99H
MOV R1,#85H     ;load R1 with value 85H
MOV R2,#3FH     ;load R2 with value 3FH
MOV R7,#63H     ;load R7 with value 63H
MOV R5,#12H     ;load R5 with value 12H
```

Solution:

By default, PSW.3=0 and PSW.4=0; therefore, the instruction “SETB PSW.4” sets RS1=1 and RS0=0, thereby selecting register bank 2. Register bank 2 uses RAM locations 10H - 17H. After the execution of the above program we have the following:

RAM location 10H has value 99H	RAM location 11H has value 85H
RAM location 12H has value 3FH	RAM location 17H has value 63H
RAM location 15H has value 12H	

Stack in the 8051

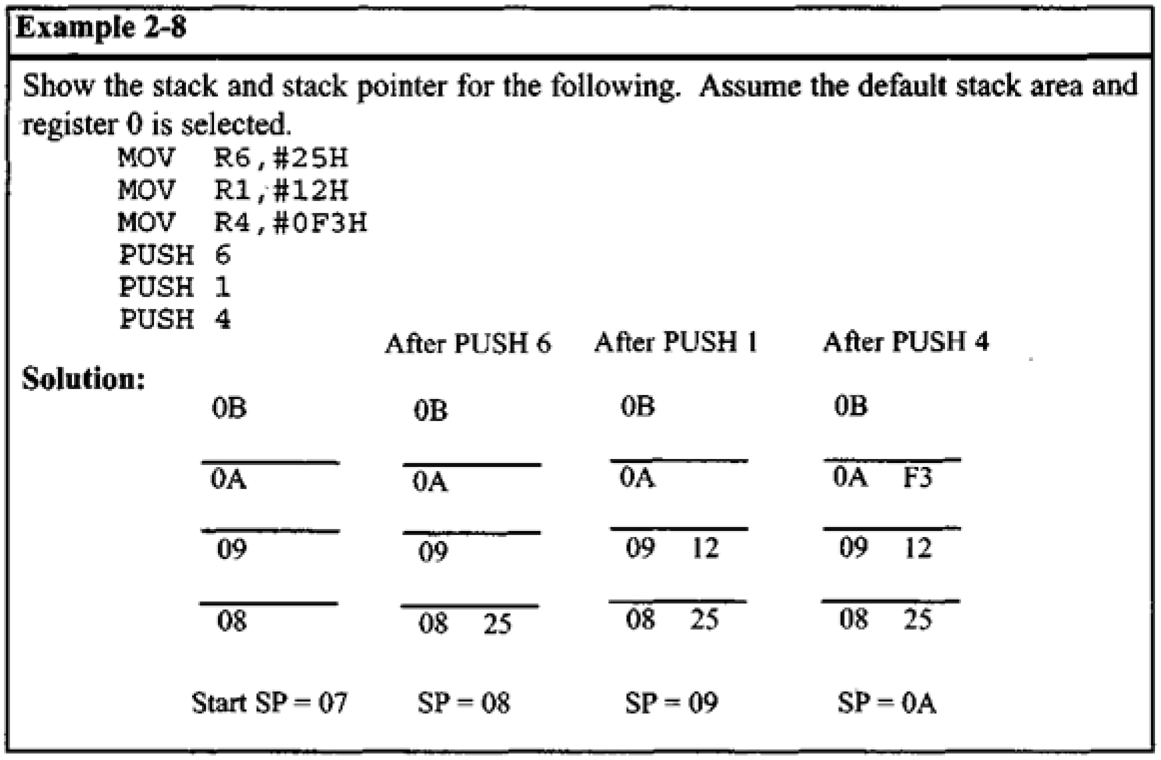
The stack is a section of RAM used by the CPU to store information temporarily. This information could be data or an address. The CPU needs this storage area since there are only a limited number of registers.

How stacks are accessed in the 8051

If the stack is a section of RAM, there must be registers inside the CPU to point to it. The register used to access the stack is called the SP (stack pointer) register. The stack pointer in the 8051 is only 8 bits wide, which means that it can take values of 00 to FFH. When the 8051 is powered up, the SP register contains value 07. This means that RAM location 08 is the first location used for the stack by the 8051. The storing of a CPU register in the stack is called a PUSH, and pulling the contents off the stack back into a CPU register is called a POP. In other words, a register is pushed onto the stack to save it and popped off the stack to retrieve it. The job of the SP is very critical when push and pop actions are performed. To see how the stack works, let’s look at the PUSH and POP instructions.

Pushing onto the stack

In the 8051 the stack pointer (SP) points to the last used location of the stack. As we push data onto the stack, the stack pointer (SP) is incremented by one. Notice that this is different from many microprocessors, notably x86 processors in which the SP is decremented when data is pushed onto the stack. Examining Example 2-8, we see that as each PUSH is executed, the content of the register are saved on the stack and SP is incremented by 1. Notice that for every byte of data saved on the stack SP is incremented only once. Notice also that to push the registers onto the stack we must use their RAM addresses. For example, the instruction “PUSH 1″ pushes register Rl onto the stack.



Example 2-8

Show the stack and stack pointer for the following. Assume the default stack area and register 0 is selected.

```
        MOV   R6,#25H
        MOV   R1,#12H
        MOV   R4,#0F3H
        PUSH  6
        PUSH  1
        PUSH  4
```

Solution:

	After PUSH 6	After PUSH 1	After PUSH 4
0B	0B	0B	0B
0A	0A	0A	0A F3
09	09	09 12	09 12
08	08 25	08 25	08 25
Start SP = 07	SP = 08	SP = 09	SP = 0A

Popping from the stack

Popping the contents of the stack back into a given register is the opposite process of pushing. With every pop, the top byte of the stack is copied to the register specified by the instruction and the stack pointer is decremented once. Example 2-9 demonstrates the POP instruction.

Example 2-9

Examining the stack, show the contents of the registers and SP after execution of the following instructions. All values are in hex.

```
    POP  3    ;POP stack into R3
    POP  5    ;POP stack into R5
    POP  2    ;POP stack into R2
```

Solution:

	After POP 3	After POP 5	After POP 2
0B 54	0B	0B	0B
0A F9	0A F9	0A	0A
09 76	09 76	09 76	09
08 6C	08 6C	08 6C	08 6C
Start SP = 0B	SP = 0A	SP = 09	SP = 08