```
50    A2        ;Data Bytes
51    FA
52    DF
53    E5
54    98
55    8B
```

PROGRAM DESCRIPTION AND OUTPUT

In this program, register B is used as a carry register, register C as a counter to count six data bytes, and the accumulator to add the data bytes and save the partial sum.

After the completion of the summation, the high-order byte (bits higher than eight bits) of the sum is saved in register B and the low-order byte is in the accumulator. Both are displayed at two different ports, or they can be stored at the memory locations XX70H and 71H.

See Questions and Programming Assignments 22–31 at the end of this chapter.

7.4 LOGIC OPERATIONS: ROTATE

In the last chapter, the logic instructions concerning the four operations AND, OR, Ex-OR, and NOT were introduced. This chapter introduces instructions related to rotating the accumulator bits. The opcodes are as follows:

- ☐ RLC: Rotate Accumulator Left
- ☐ RAL: Rotate Accumulator Left Through Carry
- ☐ RRC: Rotate Accumulator Right
- ☐ RAR: Rotate Accumulator Right Through Carry

7.4.1 Instructions

This group has four instructions; two are for rotating left and two are for rotating right. The differences between these instructions are illustrated in the following examples.

1. RLC: Rotate Accumulator Left
   - ☐ Each bit is shifted to the adjacent left position. Bit D_7 becomes D_0.
   - ☐ CY flag is modified according to bit D_7.

Example 7.8

Assume the accumulator contents are AAH and CY = 0. Illustrate the accumulator contents after the execution of the RLC instruction twice.

Solution

Figure 7.13 shows the contents of the accumulator and the CY flag after the execution of the RLC instruction twice. The first RLC instruction shifts each bit to the left by one position, places bit D_7 in bit D_0 and sets the CY flag because $D_7 = 1$. The accumulator byte AAH becomes 55H after the first rotation. In the second rotation, the byte is again AAH, and the CY flag is reset because bit D_7 of 55H is 0.

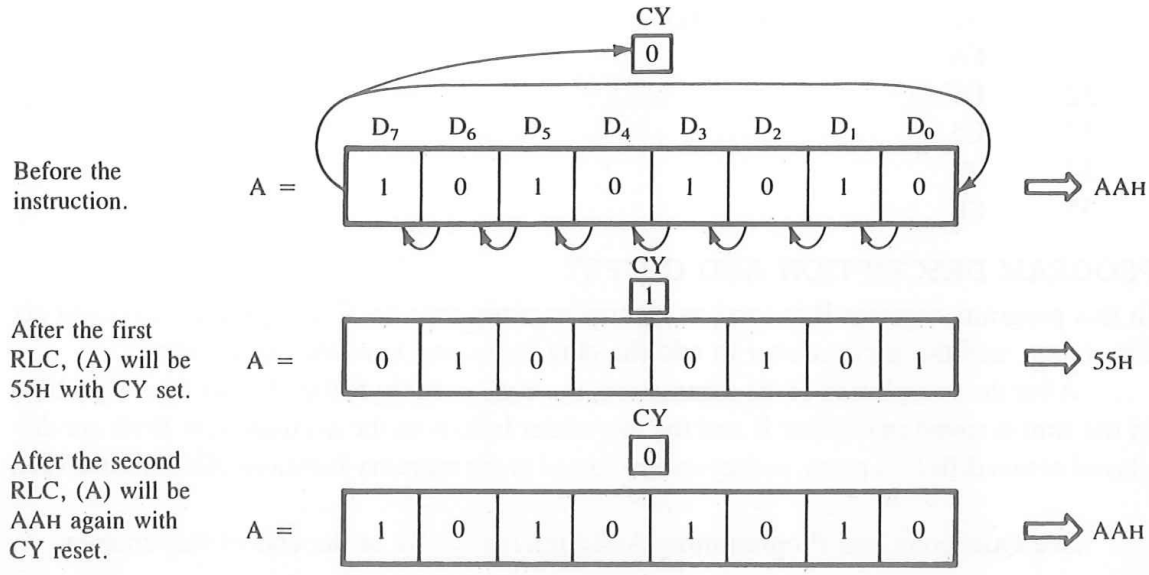


FIGURE 7.13
Accumulator Contents after RLC

2. RAL: Rotate Accumulator Left Through Carry
   - ☐ Each bit is shifted to the adjacent left position. Bit D_7 becomes the carry bit and the carry bit is shifted into D_0.
   - ☐ The Carry flag is modified according to bit D_7.

Example 7.9

Assume the accumulator contents are AAH and CY = 0. Illustrate the accumulator contents after the execution of the instruction RAL twice.

Solution

Figure 7.14 shows the contents of the accumulator and the CY flag after the execution of the RAL instruction twice. The first RAL instruction shifts each bit to the left by one position, places bit D_7 in the CY flag, and the CY bit in bit D_0. This is a 9-bit rotation; CY is assumed to be the ninth bit of the accumulator. The accumulator byte AAH becomes 54H after the first rotation. In the second rotation, the byte becomes A9H, and the CY flag is reset.

Examining these two examples, you may notice that the primary difference between these two instructions is that (1) the instruction RLC rotates through eight bits, and (2) the instruction RAL rotates through nine bits.

3. RRC: Rotate Accumulator Right
   - ☐ Each bit is shifted right to the adjacent position. Bit D_0 becomes D_7.
   - ☐ The Carry flag is modified according to bit D_0.
4. RAR: Rotate Accumulator Right Through Carry
   - ☐ Each bit is shifted right to the adjacent position. Bit D_0 becomes the carry bit, and the carry bit is shifted into D_7.

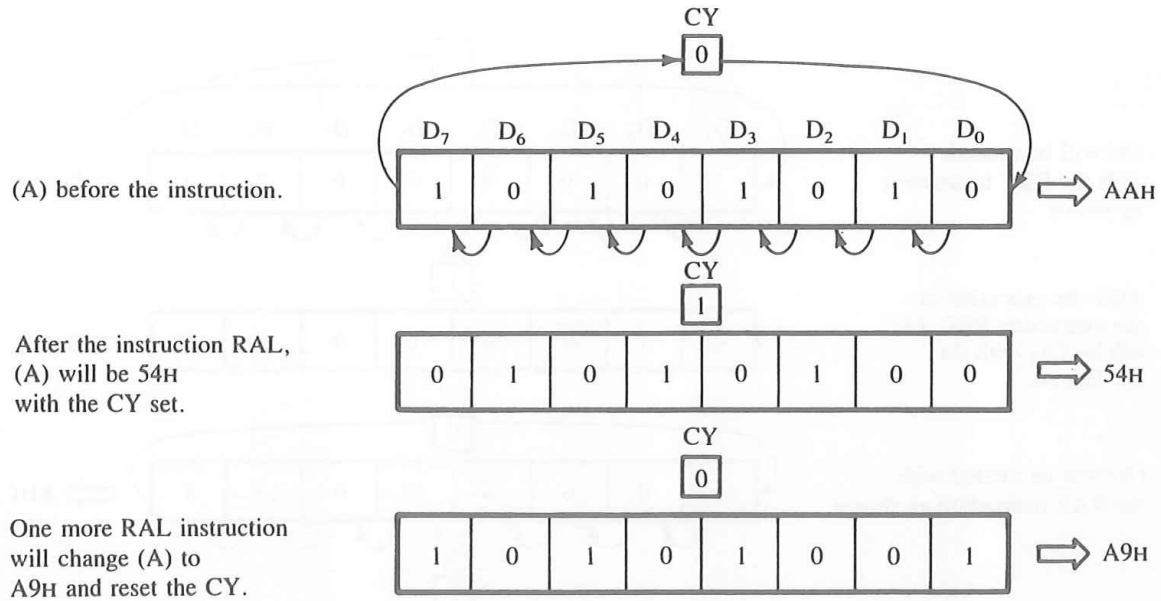


FIGURE 7.14
Accumulator Contents after RAL

Example 7.10

Assume the contents of the accumulator are 81H and CY = 0. Illustrate the accumulator contents after the RRC and RAR instructions.

Solution

Figure 7.15 shows the changes in the contents of the accumulator (81H) when the RRC instruction is used and when the RAR instruction is used. The 8-bit rotation of the RRC instruction changes 81H into C0H, and the 9-bit rotation of the RAR instruction changes 81H into 40H.

APPLICATIONS OF ROTATE INSTRUCTIONS

The rotate instructions are primarily used in arithmetic multiply and divide operations and for serial data transfer.

For example, if (A) is 0000 1000 = 08H,

- ☐ By rotating 08H right: (A) = 0000 0100 = 04H
  This is equivalent to dividing by 2
- ☐ By rotating 08H left: (A) = 0001 0000 = 10H
  This is equivalent to multiplying by 2 (10H = 16_{10})

However, these procedures are invalid when logic 1 is rotated left from D_7 to D_0 or vice versa. For example, if 80H is rotated left, it becomes 01H. Applications of serial data transfer are discussed in Chapter 16.

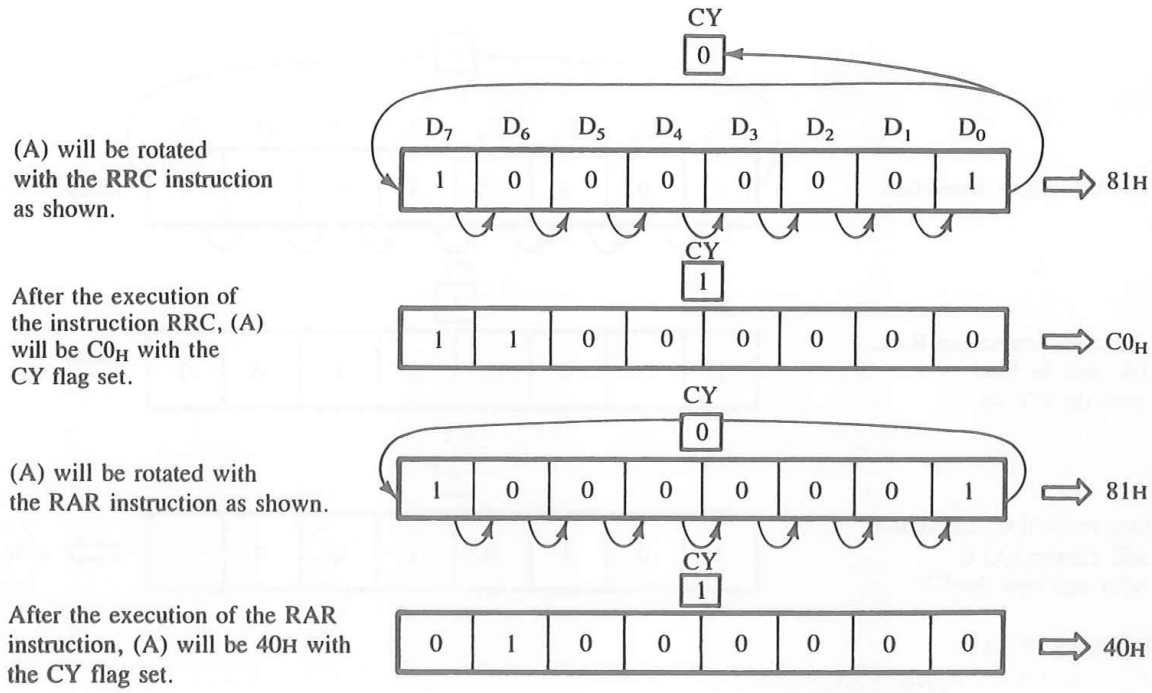


FIGURE 7.15
Rotate Right Instructions

7.4.2 Illustrative Program: Checking Sign with Rotate Instructions

PROBLEM STATEMENT

A set of ten current readings is stored in memory locations starting at XX60H. The readings are expected to be positive ($<127_{10}$). Write a program to

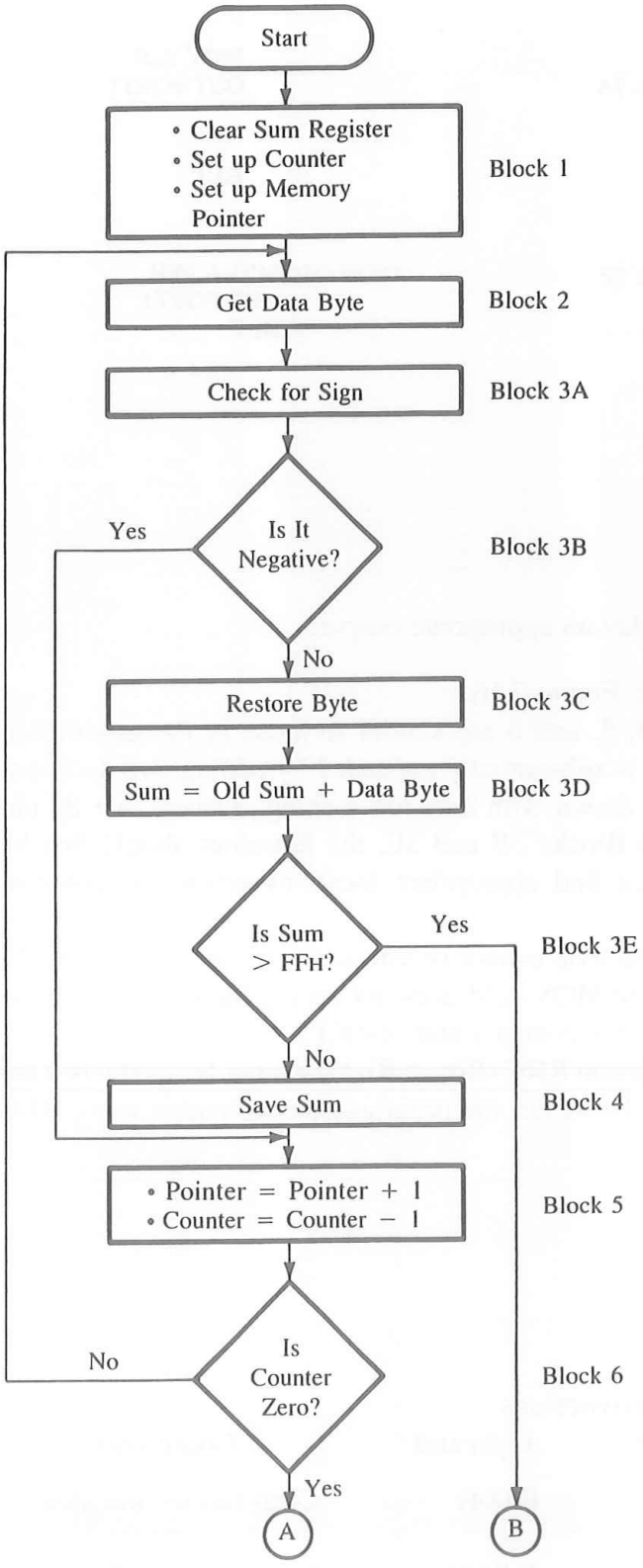
1. check each reading to determine whether it is positive or negative.
2. reject all negative readings.
3. add all positive readings.
4. output FFH to PORT1 at any time when the sum exceeds eight bits to indicate overload; otherwise, display the sum. If no output port is available in the system, go to step 5.
5. store FFH in the memory location XX70H when the sum exceeds eight bits; otherwise, store the sum.

Data(H) 28, D8, C2, 21, 24, 30, 2F, 19, F2, 9F

PROBLEM ANALYSIS

This problem can be divided into the following steps:

1. Transfer a data byte from the memory location to the microprocessor, and check whether it is a negative number. The sign of the number can be verified by rotating bit D_7 into the Carry position and checking for CY. (See Assignment 38 at the end of this chapter to verify the sign of a number with the Sign flag.)
2. If it is negative, reject the data and get the next data byte.



Mnemonics

```
        MVI B,00H
        MVI C,0AH
        LXI H,XX60H
NEXT:   MOV A,M
        RAL
        JC REJECT
        RAR
        ADD B
        JC OVRLOD
        MOV B,A
REJECT: INX H
        DCR C
        JNZ NEXT
```

FIGURE 7.16
Flowchart for Checking Sign with Rotate Instructions (*continued on next page*)

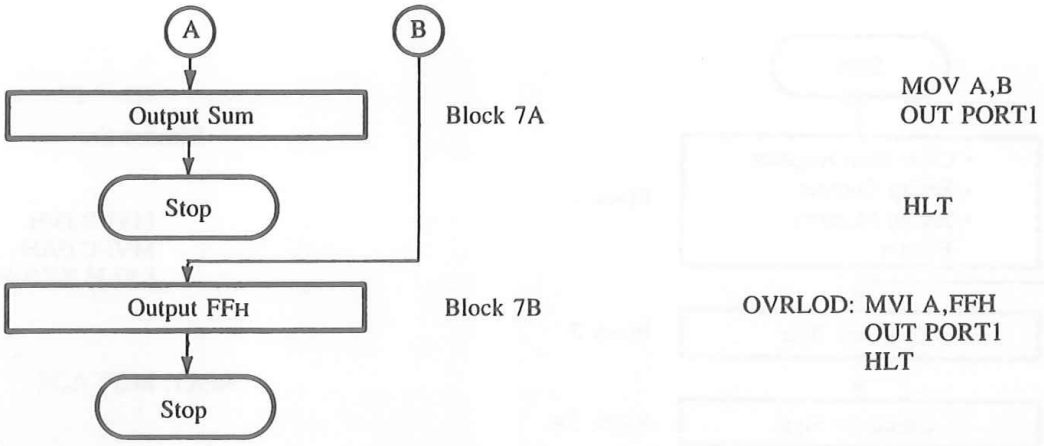


FIGURE 7.16
(continued)

3. If it is positive, add the data byte.
4. Check the sum for a carry and display an appropriate output.

The steps are shown in the flowchart in Figure 7.16.

In this flowchart, Blocks 1, 2, 4, 5, and 6 are similar to those in the generalized flowchart. Block 3—data processing—is substantially expanded by adding two decision-making blocks. This flowchart is first drawn with decision-making answers that do not change the sequence. For example, in Blocks 3B and 3E, the flowchart should first be continued with the answers NO. Then find appropriate locations where the program should be directed to the answers YES.

In this program, the sign of a data byte cannot be checked with the instruction JM (Jump On Minus) because the instruction MOV A,M does not set any flags. (However, the flags can be set by ORing the accumulator contents with itself.)

Similarly, in Block 3C, the instruction RRC (Rotate Right) cannot be used when the instruction RAL is used to rotate left. However, the program can be written using RLC and RRC in both places (3A and 3C).

PROGRAM

Memory Address HI-LO	Machine Code	Label	Instructions Opcode	Operand	Comments
XX00	06		MVI	B,00H	;Clear (B) to save sum
01	00				
02	0E		MVI	C,0AH	;Set up register C
					; as a counter

```
03      0A
04      21              LXI     H,XX60H     ;Set up HL as memory
                                            ;   pointer
05      60
06      XX
07      7E      NEXT:   MOV     A,M         ;Get byte
08      17              RAL                 ;Shift D7 into CY
09      DA              JC      REJECT      ;If D7 = 1, reject byte and
0A      12                                  ;   go to increment
                                            ;   pointer
0B      XX
0C      1F              RAR                 ;If byte is positive, re-
                                            ;   store it
0D      80              ADD     B           ;Add previous sum to (A)
0E      DA              JC      OVRLOD      ;If sum >FFH, it is over-
                                            ;   load;
0F      1C                                  ; turn on emergency
10      XX
11      47              MOV     B,A         ;Save sum
12      23      REJECT: INX     H           ;Point to next reading
13      0D              DCR     C           ;One reading is checked;
                                            ;   decrement counter
14      C2              JNZ     NEXT        ;If all readings are not
15      07                                  ;   checked, go back to
                                            ;   transfer next byte
16      XX
                ;Output Display Section
17      78              MOV     A,B
18      D3              OUT     PORT1       ;Display sum
19      PORT1
1A      76              HLT                 ;End of program
1B      00              NOP                 ;To match Jump location,
                                            ;   OVRLOD in memory
                                            ;   storage
1C      3E      OVRLOD: MVI     A,FFH       ;It is an overload
1D      FF
1E      D3              OUT     PORT1       ;Display overload signal
1F      PORT1                               ;   at PORT1
20      76              HLT
```

```
                    ;Storing Result in Memory—Alternative to Output Display
18      32                      STA     XX70H       ;Store sum in memory
                                                    ;  XX70H
19      70
1A      XX
1B      76
1C      3E          OVRLOD:     MVI     A,FFH       ;Store overload signal in
1D      FF                                          ;  memory XX70
1E      32                      STA     XX70H
1F      70
20      XX
21      76                      HLT
```

```
XX60    28      ;Current Readings
  61    D8
  62    C2
  63    21
  64    24
  65    30
  66    2F
  67    19
  68    F2
  69    9F
```

PROGRAM DESCRIPTION AND OUTPUT

In this program, register C is used as a counter to count ten bytes. Register B is used to save the sum. The sign of the number is checked by verifying whether D_7 is 1 or 0. If the Carry flag is set to indicate the negative sign, the program rejects the number and goes to Block 5, Getting Ready for Next Operation.

The program should reject the data bytes D8, C2, F2, and 9F, and should add the rest. The answer displayed should be E5.

See Questions and Programming Assignments 32–40 at the end of this chapter.

7.5 LOGIC OPERATIONS: COMPARE

The 8085 instruction set has two types of Compare operations: CMP and CPI.

- ☐ CMP: Compare with Accumulator
- ☐ CPI: Compare Immediate (with Accumulator)

The microprocessor compares a data byte (or register/memory contents) with the contents of the accumulator by subtracting the data byte from (A), and indicates