8.1 COUNTERS AND TIME DELAYS

Designing a counter is a frequent programming application. Counters are used primarily to keep track of events; time delays are important in setting up reasonably accurate timing between two events. The process of designing counters and time delays using software instructions is far more flexible and less time consuming than the design process using hardware.

COUNTER

A counter is designed simply by loading an appropriate number into one of the registers and using the INR (Increment by One) or the DCR (Decrement by One) instructions. A loop is established to update the count, and each count is checked to determine whether it has reached the final number; if not, the loop is repeated.

The flowchart shown in Figure 8.1 illustrates these steps. However, this counter has one major drawback; the counting is performed at such high speed that only the last count can be observed. To observe counting, there must be an appropriate time delay between counts.

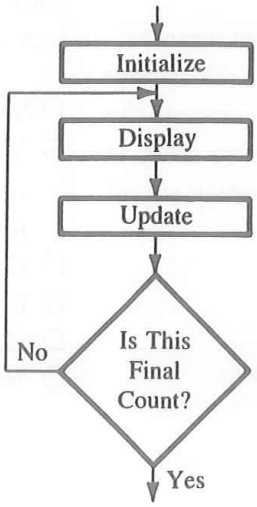
TIME DELAY

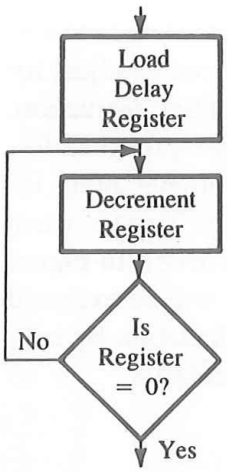
The procedure used to design a specific delay is similar to that used to set up a counter. A register is loaded with a number, depending on the time delay required, and then the register is decremented until it reaches zero by setting up a loop with a conditional Jump instruction. The loop causes the delay, depending upon the clock period of the system, as illustrated in the next sections.

8.1.1 Time Delay Using One Register

The flowchart in Figure 8.2 shows a time-delay loop. A count is loaded in a register, and the loop is executed until the count reaches zero. The set of instructions necessary to set up the loop is also shown in Figure 8.2.

FIGURE 8.1
Flowchart of a Counter





Label	Opcode	Operand	Comments	T-states
	MVI	C,FFH	;Load register C	7
LOOP:	DCR	C	;Decrement C	4
	JNZ	LOOP	;Jump back to ; decrement C	10/7

FIGURE 8.2
Time Delay Loop: Flowchart and Instructions

The last column in Figure 8.2 shows the T-states (clock periods) required by the 8085 microprocessor to execute each instruction. (See Appendix F for the list of the 8085 instructions and their T-states.) The instruction MVI requires seven clock periods. An 8085-based microcomputer with 2 MHz clock frequency will execute the instruction MVI in 3.5 μs as follows:

$$\text{Clock frequency of the system } f = 2 \text{ MHz}$$
$$\text{Clock period } T = 1/f = 1/2 \times 10^{-6} = 0.5 \ \mu s$$
$$\text{Time to execute MVI} = 7 \text{ T-states} \times 0.5$$
$$= 3.5 \ \mu s$$

However, if the clock frequency of the system is 1 MHz, the microprocessor will require 7 μs to execute the same instruction. To calculate the time delay in a loop, we must account for the T-states required for each instruction and for the number of times the instructions are executed in the loop.

In Figure 8.2, register C is loaded with the count FFH (255_{10}) by the instruction MVI, which is executed once and takes seven T-states. The next two instructions, DCR and JNZ, form a loop with a total of 14 (4 + 10) T-states. The loop is repeated 255 times until register C = 0.

The time delay in the loop T_L with 2 MHz clock frequency is calculated as

$$TL = (T \times \text{Loop T-states} \times N10)$$

where T_L = Time delay in the loop
T = System clock period
N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay register

$$T_L = (0.5 \times 10^{-6} \times 14 \times 255)$$
$$= 1785 \ \mu s$$
$$\approx 1.8 \text{ ms}$$

In most applications, this approximate calculation of the time delay is considered reasonably accurate. However, to calculate the time delay more accurately, we need to adjust for the execution of the JNZ instruction and add the execution time of the initial instruction.

The T-states for JNZ instruction are shown as 10/7. This can be interpreted as follows: The 8085 microprocessor requires ten T-states to execute a conditional Jump instruction when it jumps or changes the sequence of the program and seven T-states when the program falls through the loop (goes to the instruction following the JNZ). In Figure 8.2, the loop is executed 255 times; in the last cycle, the JNZ instruction will be executed in seven T-states. This difference can be accounted for in the delay calculation by subtracting the execution time of three states. Therefore, the adjusted loop delay is

$$\begin{aligned} T_{LA} &= T_L - (3 \text{ T-states} \times \text{Clock period}) \\ &= 1785.0\ \mu s - 1.5\ \mu s = 1783.5\ \mu s \end{aligned}$$

Now the total delay must take into account the execution time of the instructions outside the loop. In the above example, we have only one instruction (MVI C) outside the loop. Therefore, the total delay is

$$\text{Total Delay} = \begin{matrix}\text{Time to execute instructions} \\ \text{outside loop}\end{matrix} + \begin{matrix}\text{Time to execute} \\ \text{loop instructions}\end{matrix}$$

$$\begin{aligned} T_D &= T_O + T_{LA} \\ &= (7 \times 0.5\ \mu s) + 1783.5\ \mu s = 1787\ \mu s \\ &\approx 1.8\ ms \end{aligned}$$

The difference between the loop delay T_L and these calculations is only 2 μs and can be ignored in most instances.

The time delay can be varied by changing the count FFH; however, to increase the time delay beyond 1.8 ms in a 2 MHz microcomputer system, a register pair or a loop within a loop technique should be used.

8.1.2 Time Delay Using a Register Pair

The time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (maximum FFFFH). The 16-bit number is decremented by using the instruction DCX. However, the instruction DCX does not set the Zero flag and, without the test flags, Jump instructions cannot check desired data conditions. Additional techniques, therefore, must be used to set the Zero flag.

The following set of instructions uses a register pair to set up a time delay.

Label	Opcode	Operand	Comments	T-states
	LXI	B,2384H	;Load BC with 16-bit count	10
LOOP:	DCX	B	;Decrement (BC) by one	6
	MOV	A,C	;Place contents of C in A	4
	ORA	B	;OR (B) with (C) to set Zero flag	4
	JNZ	LOOP	;If result ≠ 0, jump back to LOOP	10/7

In this set of instructions, the instruction LXI B,2384H loads register B with the number 23H, and register C with the number 84H. The instruction DCX decrements the entire number by one (e.g., 2384H becomes 2383H). The next two instructions are used only to set the Zero flag; otherwise, they have no function in this problem. The OR instruction sets the Zero flag only when the contents of B and C are simultaneously zero. Therefore, the loop is repeated 2384H times, equal to the count set in the register pair.

TIME DELAY

The time delay in the loop is calculated as in the previous example. The loop includes four instructions: DCX, MOV, ORA, and JNZ, and takes 24 clock periods for execution. The loop is repeated 2384H times, which is converted to decimals as

$$\begin{aligned} 2384H &= 2 \times (16)^3 + 3 \times (16)^2 + 8 \times (16)^1 + 4(16^0) \\ &= 9092_{10} \end{aligned}$$

If the clock period of the system = 0.5 μs, the delay in the loop T_L is

$$\begin{aligned} T_L &= (0.5 \times 24 \times 9092_{10}) \\ &\approx 109 \text{ ms (without adjusting for the last cycle)} \\ \text{Total Delay } T_D &= 109 \text{ ms} + T_O \\ &\approx 109 \text{ ms (The instruction LXI adds only 5 μs.)} \end{aligned}$$

A similar time delay can be achieved by using the technique of two loops, as discussed in the next section.

8.1.3 Time Delay Using a Loop within a Loop Technique

A time delay similar to that of a register pair can also be achieved by using two loops; one loop inside the other loop, as shown in Figure 8.3(a). For example, register C is used in the inner loop (LOOP1) and register B is used for the outer loop (LOOP2). The following instructions can be used to implement the flowchart shown in Figure 8.3(a).

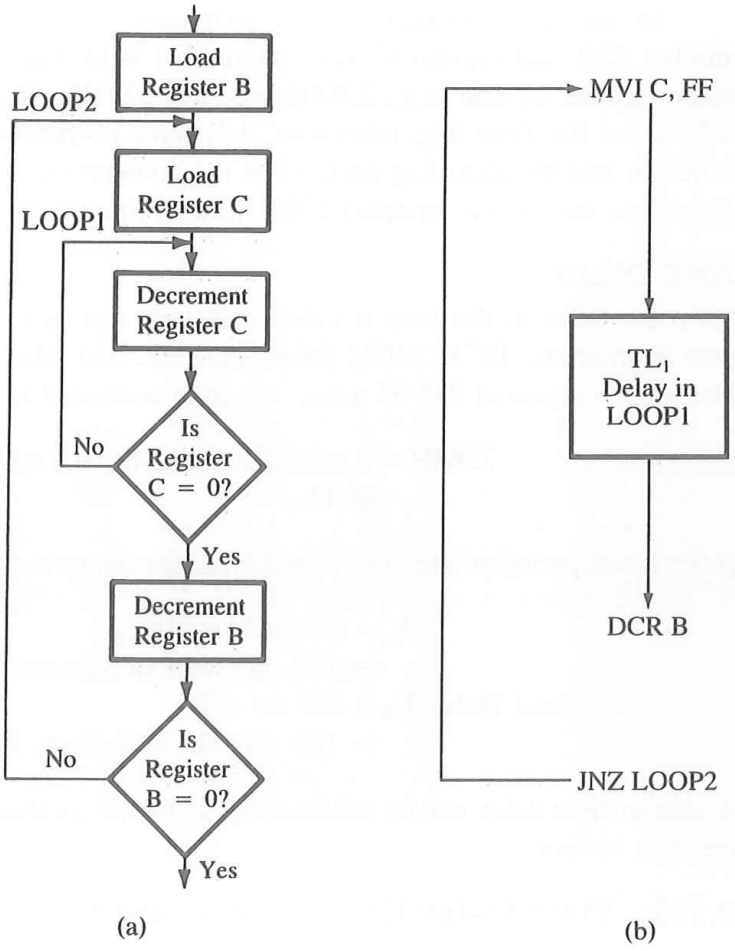
```
        MVI B,38H     7T
LOOP2:  MVI C,FFH     7T
LOOP1:  DCR C         4T
        JNZ LOOP1     10/7T
        DCR B         4T
        JNZ LOOP2     10/7T
```

DELAY CALCULATIONS

The delay in LOOP1 is T_{L1} = 1783.5 μs. These calculations are shown in Section 8.1.1. We can replace LOOP1 by T_{L1}, as shown in Figure 8.3(b). Now we can calculate the delay in LOOP2 as if it is one loop; this loop is executed 56 times because of the count (38H) in register B:

$$\begin{aligned} T_{L2} &= 56(T_{L1} + 21 \text{ T-states} \times 0.5 \text{ μs}) \\ &= 56(1783.5 \text{ μs} + 10.5 \text{ μs}) \\ &= 100.46 \text{ ms} \end{aligned}$$

FIGURE 8.3
Flowchart for Time Delay with Two Loops



The total delay should include the execution time of the first instruction (MVI B,7T); however, the delay outside these loops is insignificant. The time delay can be increased considerably by using register pairs in the above example.

Similarly, the time delay within a loop can be increased by using instructions that will not affect the program except to increase the delay. For example, the instruction NOP (No Operation) can add four T-states in the delay loop. The desired time delay can be obtained by using any or all available registers.

8.1.4 Additional Techniques for Time Delay

The disadvantages in using software delay techniques for real-time applications in which the demand for time accuracy is high, such as digital clocks, are as follows:

1. The accuracy of the time delay depends on the accuracy of the system's clock.
2. The microprocessor is occupied simply in a waiting loop; otherwise it could be employed to perform other functions.
3. The task of calculating accurate time delays is tedious.

In real-time applications, timers (integrated timer circuits) are commonly used. The Intel 8254 (described in Chapter 15) is a programmable timer chip that can be interfaced with the microprocessor and programmed to provide timings with considerable accuracy. The disadvantages of using the hardware chip include the additional expense and the need for an extra chip in the system.

8.1.5 Counter Design with Time Delay

To design a counter with a time delay, the techniques illustrated in Figures 8.1 and 8.2 can be combined. The combined flowchart is shown in Figure 8.4.

The blocks shown in the flowchart are similar to those in the generalized flowchart in Figure 7.3. The block numbers shown in Figure 8.4 correspond to the block numbers in the generalized flowchart. Compare Figure 8.4 with Figure 7.3 and note the following points about the counter flowchart (Figure 8.4):

1. The output (or display) block is part of the counting loop.
2. The data processing block is replaced by the time-delay block.
3. The save-the-partial-answer block is eliminated because the count is saved in a counter register, and a register can be incremented or decremented without transferring the count to the accumulator.

The flowchart in Figure 8.4 shows the basic building blocks. However, the sequence can be changed, depending upon the nature of the problem, as shown in Figures 8.5(a) and 8.5(b).

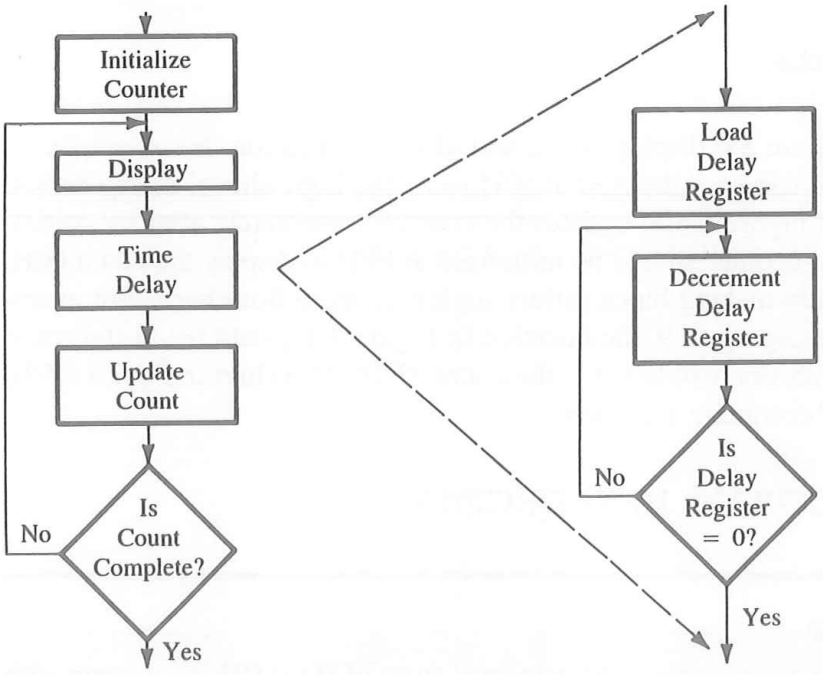


FIGURE 8.4
Flowchart of a Counter with a Time Delay

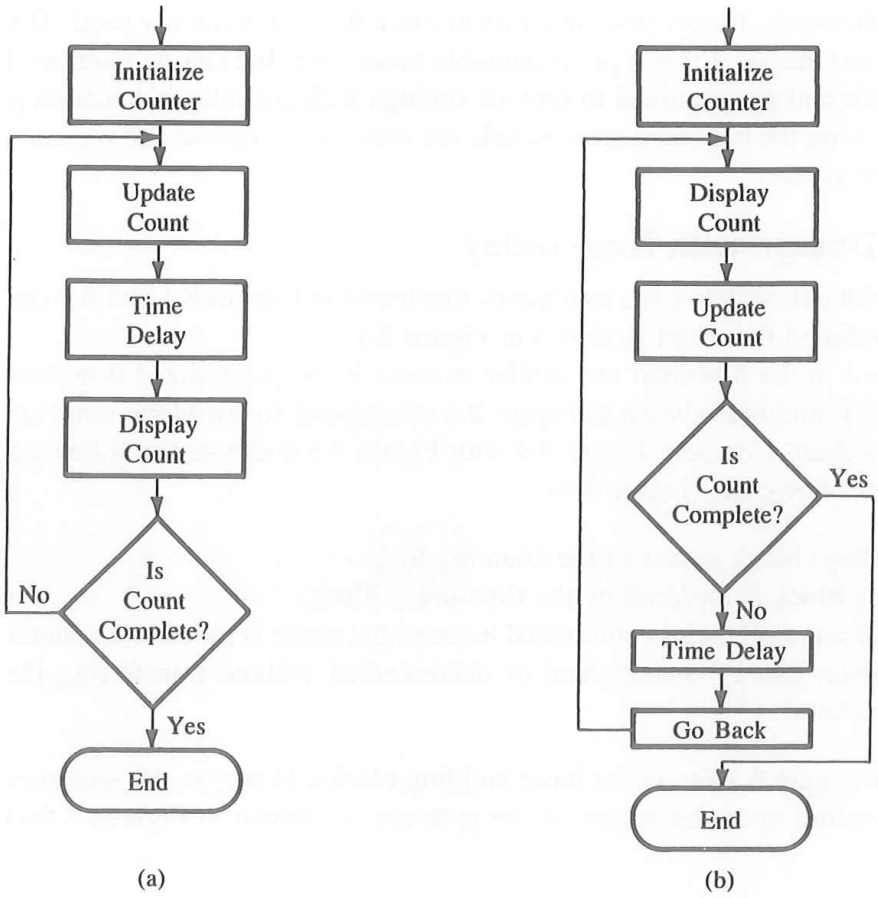


FIGURE 8.5
Variations of Counter Flowchart

The flowchart in Figure 8.4 displays the count after initialization. For example, an up-counter starting at zero can be initialized at 00H using the logic shown in Figure 8.4. However, the flowchart in Figure 8.5(a) updates the counter immediately after the initialization. In this case, an up-counter should be initialized at FFH to display the count 00H.

Similarly, the decision-making block differs slightly in these flowcharts. For example, if a counter was counting up to 9, the question in Figure 8.4 would be: Is the count 10? In Figure 8.5(a) the question would be: Is the count 9? The flowchart in Figure 8.5(b) illustrates another way of designing a counter.

8.2 ILLUSTRATIVE PROGRAM: HEXADECIMAL COUNTER

PROBLEM STATEMENT

Write a program to count continuously in hexadecimal from FFH to 00H in a system with a 0.5 μs clock period. Use register C to set up a one millisecond (ms) delay between each count and display the numbers at one of the output ports.

PROBLEM ANALYSIS

The problem has two parts; the first is to set up a continuous down-counter, and the second is to design a given delay between two counts.

1. The hexadecimal counter is set up by loading a register with an appropriate starting number and decrementing it until it becomes zero (shown by the outer loop in the flowchart, Figure 8.6). After zero count, the register goes back to FF because decrementing zero results in a (–1), which is FF in 2's complement.
2. The one millisecond (ms) delay between each count is set up by using the procedure explained previously in Section 8.1.1—Time Delay Using One Register. Figure 8.2 is identical with the inner loop of the flowchart shown in Figure 8.6. The delay calculations are shown later.

FLOWCHART AND PROGRAM

The flowchart in Figure 8.6 shows the two loops discussed earlier—one for the counter and another for the delay. The counter is initialized in the first block, and the count is displayed

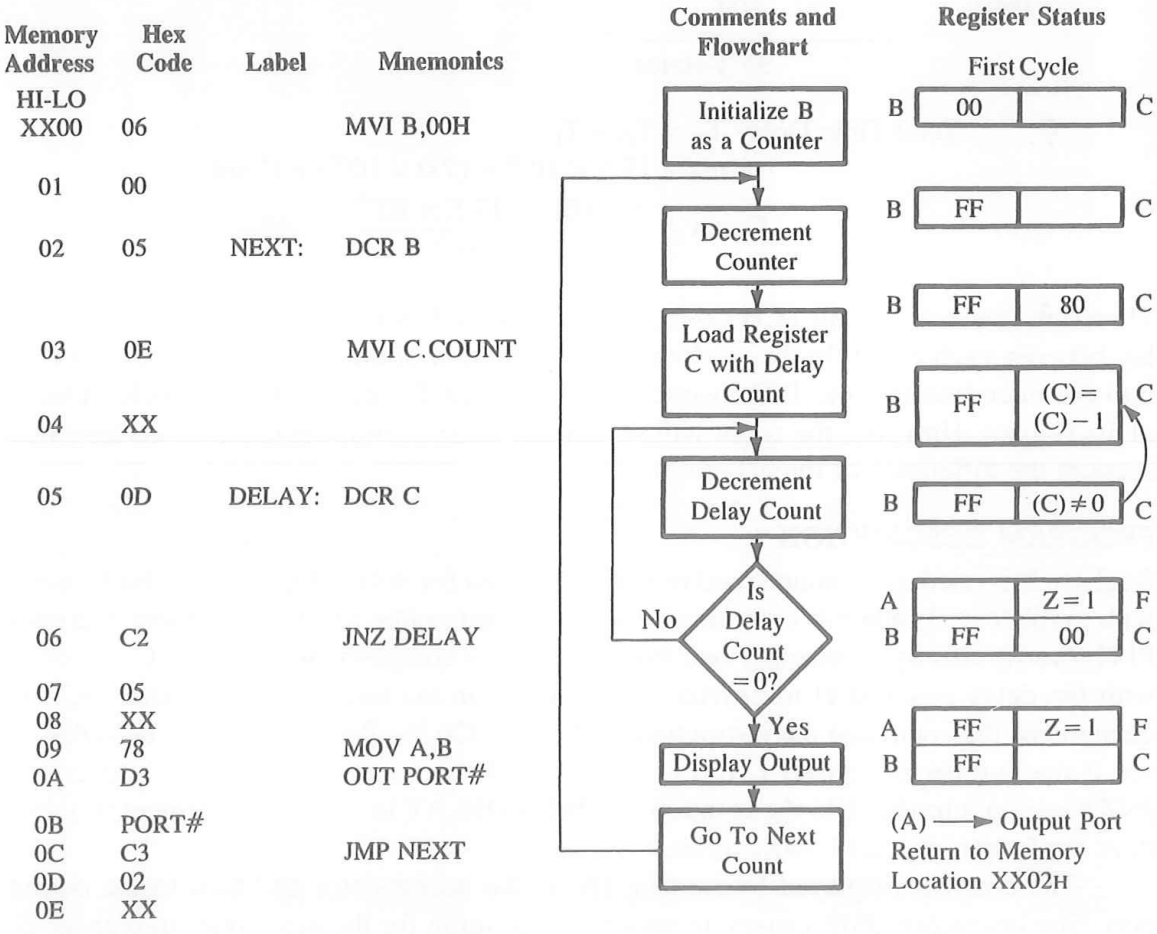


FIGURE 8.6
Program and Flowchart for a Hexadecimal Counter

in the outer loop. The delay is accomplished in the inner loop. This flowchart in Figure 8.6 is similar to that in Figure 8.5(a). You should study the flowchart carefully to differentiate between the counter loop and the delay loop because they may at first appear to be similar.

Delay Calculations The delay loop includes two instructions: DCR C and JNZ with 14 T-states. Therefore, the time delay T_L in the loop (without accounting for the fact that JNZ requires seven T-states in the last cycle) is

$$\begin{aligned} T_L &= 14 \text{ T-states} \times T \text{ (Clock period)} \times \text{Count} \\ &= 14 \times (0.5 \times 10^{-6}) \times \text{Count} \\ &= (7.0 \times 10^{-6}) \times \text{Count} \end{aligned}$$

The delay outside the loop includes the following instructions:

DCR B	4T
MVI C,COUNT	7T
MOV A,B	4T
OUT PORT	10T
JMP	10T
	35 T-states

Delay outside the loop: $T_O = 35 \text{ T-states} \times T = 35 \times (0.5 \times 10^{-6}) = 17.5\ \mu s$

$$\text{Total Time Delay } T_D = T_O + T_L$$

$$1 \text{ ms} = 17.5 \times 10^{-6} + (7.0 \times 10^{-6}) \times \text{Count}$$

$$\text{Count} = \frac{1 \times 10^{-3} - 17.5 \times 10^{-6}}{7.0 \times 10^{-6}} \approx 140_{10}$$

Therefore, the delay count 8CH (140_{10}) must be loaded in register C to obtain 1 ms delay between each count. In this problem, the calculation of the delay count does not take into consideration that the JNZ instruction takes seven T-states in the last cycle, instead of 10 T-states. However, the count will remain the same even if the calculations take into account the difference of three T-states.

PROGRAM DESCRIPTION

Register B is used as a counter, and register C is used for delay. Register B initially starts with number 00H; when it is decremented by the instruction DCR, the number becomes FFH. (Verify this by subtracting one from zero in 2's complement.) Register C is loaded with the delay count 8CH to provide a 1 ms delay in the loop. The instruction DCR C decrements the count and the instruction JNZ (Jump On No Zero) checks the Zero flag to see if the number in register C has reached zero. If the number is not zero, instruction JNZ causes a jump back to the instruction labeled DELAY in order to decrement (C) and, thus, the loop is repeated 140_{10} times.

The count is displayed by moving (B) to the accumulator and then to the output port. The instruction JMP causes an unconditional jump for the next count in register B, forming a continuous loop to count from FFH to 00H. After the count reaches zero, (B) is decremented, becoming FFH, and the counting cycle is repeated.

PROGRAM OUTPUT

When the program is executed, the actual output seen may vary according to the device used as the output for the display. The eye cannot see the changes in a count with a 1 ms delay. If the output port has eight LEDs, the LEDs representing the low-order bits will appear to be on continuously, and the LEDs with high-order bits will go on and off according to the count. If the output port is a seven-segment display, all segments will appear to be on; a slight flicker in the display can be noticed. However, the count and the delay can be measured on an oscilloscope.

8.3 ILLUSTRATIVE PROGRAM: ZERO-TO-NINE (MODULO TEN)* COUNTER

PROBLEM STATEMENT

Write a program to count from 0 to 9 with a one-second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Use register pair HL to set up the delay, and display each count at one of the output ports. Assume the clock frequency of the microcomputer is 1 MHz.

Instructions Review the following instructions.

- ☐ LXI: Load Register Pair Immediate
- ☐ DCX: Decrement Register Pair
- ☐ INX: Increment Register Pair

These instructions manipulate 16-bit data by using registers in pairs (BC, DE, and HL). However, the instructions DCX and INX do not affect flags to reflect the outcome of the operation. Therefore, additional instructions must be used to set the flags.

PROBLEM ANALYSIS

The problem is similar to that in the Illustrative Program for a hexadecimal counter (Section 8.2) except in two respects: the counter is an up-counter (counts *up* to the digit 9), and the delay is too long to use just one register.

1. The counter is set up by loading a register with the appropriate number and incrementing it until it reaches the digit 9. When the counter register reaches the final count, it is reset to zero. This is an additional step compared to the Illustrative Program of Section 8.2 in which the counter resets itself. (Refer to the flowchart in Figure 8.7 to see how each count is checked against the final count.)

*The counter goes through ten different states (0 to 9) and is called a *modulo ten* counter.