Pearson New International Edition

The 8051 Microcontroller and Embedded Systems

Mazidi Mazidi McKinlay

Second Edition

Pearson New International Edition

The 8051 Microcontroller and Embedded Systems

Mazidi Mazidi McKinlay

Second Edition

PEARSON

40. Assume that the lower four bits of P1 are connected to four switches. Write a program to send the following ASCII characters to P2 based on the status of the switches.

0000	‘0’
0001	‘1’
0010	‘2’
0011	‘3’
0100	‘4’
0101	‘5’
0110	‘6’
0111	‘7’
1000	‘8’
1001	‘9’
1010	‘A’
1011	‘B’
1100	‘C’
1101	‘D’
1110	‘E’
1111	‘F’

41. Find the checksum byte for the following ASCII message: “Hello”
42. True or false. If we add all the bytes, including the checksum byte, and the result is 00H, then there is no error in the data.
43. Write a program: (a) To get the data “Hello, my fellow World citizens” from code ROM, (b) to calculate the check-sum byte, and (c) to test the checksum byte for any data error.
44. Give three reasons you should write your programs in modules.
45. To display data on LCD or PC monitors, it must be in ______ (BIN, BCD, ASCII).
46. Assume that the lower four bits of P1 are connected to four switches. Write a program to send the following ASCII characters to P2 based on the status of the switches. Do not use the look-up table method.

0000	‘0’
0001	‘1’
0010	‘2’
0011	‘3’
0100	‘4’
0101	‘5’
0110	‘6’
0111	‘7’
1000	‘8’
1001	‘9’

ANSWERS TO REVIEW QUESTIONS

SECTION 6.1: ARITHMETIC INSTRUCTIONS

1. A, B
2. A, B
3. No. We must use registers A and B for this operation.
4. A, B
5. A, B
6. No. We must use registers A and B for this operation.
7. A, the accumulator
8. No. We must use registers A and B for this operation.
9.
```
MOV  A,R1
ADD  A,R2
```
10. A, the accumulator
11. (a) A = 00 and CY = 1 (b) A = FF and CY = 0
12.
```
   43H  0100 0011                       0100 0011
  -05H  0000 0101 2's complement  +  1111 1011
   3EH  0011 1110
```
13. A = 95H – 4FH – 1 = 45H

SECTION 6.2: SIGNED NUMBER CONCEPTS AND ARITHMETIC OPERATIONS

1. D7
2. 16H is 00010110 in binary and its 2's complement is 1110 1010 or −16H = EA in hex.
3. −128 to +127
4. +9 = 00001001 and −9 = 11110111 or F7 in hex.
5. An overflow is a carry into the sign bit (D7), but the carry is a carry out of register (D7).

SECTION 6.3: LOGIC AND COMPARE INSTRUCTIONS

1. (a) 02 (b) FFH (c) FDH
2. Zeros
3. One
4. All zeros
5. False
6. #53
7. 66H

SECTION 6.4: ROTATE INSTRUCTION AND DATA SERIALIZATION

1. 52H
2. 2AH
3. C0H
4. Because all the rotate instructions work with the accumulator only
5. 50H
6. CY = 0
7. CY = 1
8. CY = 1
9.
```
MOV C,P2.7     ;save status of P2.7 on CY
MOV 31,C       ;save carry in RAM bit location 06
```
10.
```
MOV C,9        ;save status of RAM bit 09 in CY
MOV P1.4,C     ;save carry in P1.4
```

SECTION 6.5: BCD, ASCII, AND OTHER APPLICATION PROGRAMS

1. (a) 15H = 0001 0101 packed BCD, 0000 0001,0000 0101 unpacked BCD
   (b) 99H = 1001 1001 packed BCD, 0000 1001,0000 1001 unpacked BCD
2. 3736H = 00110111 00110110B
   and in BCD we have 76H = 0111 0110B
3. No. We need to write it 54H (with the H) or 01010100B to make it in BCD. The value 54 without the "H" is interpreted as 36H by the assembler.
4. 36H, 37H
5. Yes, since A = 39H
6. ROM
7. 88H + 99H + AAH + BBH + CCH + DDH = 42FH. Dropping the carries we have 2FH, and its 2's complement is D1H.
8. False

CHAPTER 7

8051 PROGRAMMING IN C

OBJECTIVES

Upon completion of this chapter, you will be able to:

- **>>** **Examine the C data type for the 8051**
- **>>** **Code 8051 C programs for time delay and I/O operations**
- **>>** **Code 8051 C programs for I/O bit manipulation**
- **>>** **Code 8051 C programs for logic and arithmetic operations**
- **>>** **Code 8051 C programs for ASCII and BCD data conversion**
- **>>** **Code 8051 C programs for binary (hex) to decimal conversion**
- **>>** **Code 8051 C programs to use the 8051 code space**
- **>>** **Code 8051 C programs for data serialization**

Why program the 8051 in C?

Compilers produce hex files that we download into the ROM of the microcontroller. The size of the hex file produced by the compiler is one of the main concerns of microcontroller programmers, for two reasons:

1. Microcontrollers have limited on-chip ROM.
2. The code space for the 8051 is limited to 64K bytes.

How does the choice of programming language affect the compiled program size? While Assembly language produces a hex file that is much smaller than C, programming in Assembly language is tedious and time consuming. C programming, on the other hand, is less time consuming and much easier to write, but the hex file size produced is much larger than if we used Assembly language. The following are some of the major reasons for writing programs in C instead of Assembly:

1. It is easier and less time consuming to write in C than Assembly.
2. C is easier to modify and update.
3. You can use code available in function libraries.
4. C code is portable to other microcontrollers with little or no modification.

The study of C programming for the 8051 is the main topic of this chapter. In Section 7.1, we discuss data types and time delays. I/O programming is shown in Section 7.2. The logic operations AND, OR, XOR, inverter, and shift are discussed in Section 7.3. Section 7.4 describes ASCII and BCD conversions and checksums. In Section 7.5 we show how 8051 C compilers use the program (code) ROM space for data. Finally, in Section 7.6 data serialization for 8051 is shown.

SECTION 7.1: DATA TYPES AND TIME DELAY IN 8051 C

In this section we first discuss C data types for the 8051 and then provide code for time delay functions.

C data types for the 8051

Since one of the goals of 8051 C programmers is to create smaller hex files, it is worthwhile to re-examine C data types for 8051 C. In other words, a good understanding of C data types for the 8051 can help programmers to create smaller hex files. In this section we focus on the specific C data types that are most useful and widely used for the 8051 microcontroller.

Unsigned char

Since the 8051 is an 8-bit microcontroller, the character data type is the most natural choice for many applications. The unsigned char is an 8-bit data type that takes a value in the range of 0 - 255 (00 - FFH). It is one of the most widely used data types for the 8051. In many situations, such as setting a counter value, where there is no need for signed data we should use the unsigned char instead of the signed char. Remember that C compilers use the signed char as the default if we do not put the keyword *unsigned* in front of the char (see Example 7-1). We can also use the unsigned char data type for a string of ASCII characters, including extended ASCII characters. Example 7-2 shows a string of ASCII characters. See Example 7-3 for toggling ports.

In declaring variables, we must pay careful attention to the size of the data and try to use unsigned char instead of int if possible. Because the 8051 has a limited number of registers and data RAM locations, using the int in place of the char data type can lead to a larger size hex file. Such a misuse of the data types in compilers such as Microsoft Visual C++ for x86 IBM PCs is not a significant issue.

Example 7-1

Write an 8051 C program to send values 00 - FF to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char z;
    for(z=0;z<=255;z++)
      P1=z;
  }
```

Run the above program on your simulator to see how P1 displays values 00 - FFH in binary.

Example 7-2

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char mynum[]= "012345ABCD";
    unsigned char z;
    for(z=0;z<=10;z++)
      P1=mynum[z];
  }
```

Run the above program on your simulator to see how P1 displays values 30H, 31H, 32H, 33H, 34H, 35H, 41H, 42H, 43H, and 44H, the hex values for ASCII 0, 1, 2, and so on.