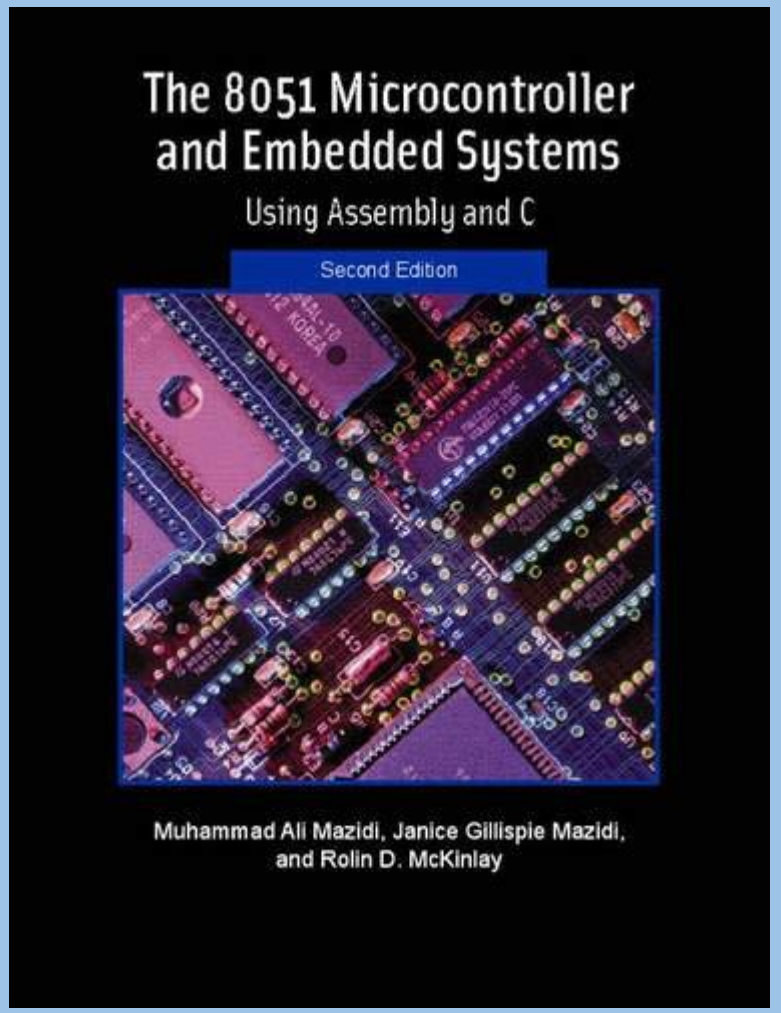
THE 8051 MICROCONTROLLER & Embedded Systems

Muhammad Ali Mazidi, Janice Mazidi
& Rolin McKinlay



JUMP, LOOP, AND CALL INSTRUCTIONS

Chapter 3

Objectives

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

- >> Code 8051 Assembly language instructions using loops
- >> Code 8051 Assembly language conditional jump instructions
- >> Explain conditions that determine each conditional jump instruction
- >> Code long jump instructions for unconditional jumps
- >> Code short jump instructions for unconditional short jumps
- >> Calculate target addresses for jump instructions
- >> Code 8051 subroutines
- >> Describe precautions in using the stack in subroutines
- >> Discuss crystal frequency versus machine cycle
- >> Code 8051 programs to generate a time delay

Objectives

- Discussing the control transfer instructions available in 8051 Assembly language,
- Section 3.1, instructions used for looping, as well as instructions for conditional and unconditional jumps,
- Section 3.2, CALL instructions and their uses,
- Section 3.3, time delay subroutine.

Loop instructions

Looping in the 8051

- Repeating a sequence of instructions a certain number of times is called a *loop*.
- Loop action is performed by instruction: “DJNZ reg, label”
- DJNZ: Decrement and jump if register ≠ 0
- Register is decremented; if it is not zero, it jumps to the target address referred to by the label.
- In DJNZ instruction, register can be any of R0-R7

Example 3-1: Looping in the 8051

Write a program to
(a) clear ACC, then
(b) add 3 to the accumulator ten times.

Solution:

;This program adds value 3 to the ACC ten times

```
          MOV   A,#0        ;A=0, clear ACC
          MOV   R2,#10      ;load counter R2=10
AGAIN:    ADD   A,#03       ;add 03 to ACC
          DJNZ  R2,AGAIN    ;repeat until R2=0(10 times)
          MOV   R5,A        ;save A in R5
```

Example 3-2

What is the maximum number of times that the loop in Example 3-1 can be repeated?

Solution:

Since R2 holds the count and R2 is an 8-bit register, it can hold a maximum of FFH (255 decimal); therefore, the loop can be repeated a maximum of 256 times.

```
Note: Nested loop i.e. loop inside a loop, can
be used to repeat action more times than 256
```

Example 3-3: Loop Inside a Loop

Write a program to (a) load the accumulator with the value 55H, and (b) complement the ACC 700 times.

Solution:

Since 700 is larger than 255 (the maximum capacity of any register), we use two registers to hold the count. The following code shows how to use R2 and R3 for the count.

```
            MOV   A,#55H        ;A=55H
            MOV   R3,#10        ;R3=10, the outer loop count
NEXT: MOV   R2,#70              ;R2=70, the inner loop count
AGAIN:      CPL   A             ;complement A register
            DJNZ  R2,AGAIN      ;repeat it 70 times (inner loop)
            DJNZ  R3,NEXT
```

In this program, R2 is used to keep the inner loop count. In the instruction "DJNZ R2,AGAIN", whenever R2 becomes 0 it falls through and "DJNZ R3,NEXT" is executed. This instruction forces the CPU to load R2 with the count 70 and the inner loop starts again. This process will continue until R3 becomes zero and the outer loop is finished.

Jump instructions

Instruction	Action
JZ	Jump if A = 0
JNZ	Jump if A ≠ 0
DJNZ	Decrement and jump if register ≠ 0
CJNE A, data	Jump if A ≠ data
CJNE reg, #data	Jump if byte ≠ #data
JC	Jump if CY = 1
JNC	Jump if CY = 0
JB	Jump if bit = 1
JNB	Jump if bit = 0
JBC	Jump if bit = 1 and clear bit

Table 3-1: 8051 Conditional Jump Instructions

Example 3-4

Write a program to determine if R5 contains the value 0. If so, put 55H in it.

Solution:

```
        MOV   A,R5        ;copy R5 to A
        JNZ   NEXT        ;jump if A is not zero
        MOV   R5,#55H
NEXT:...
```

Example 3-5

Find the sum of the values 79H, F5H, and E2H. Put the sum in registers R0 (low byte) and R5 (high byte).

Solution:

```
        MOV   A,#0        ;clear A(A=0)
        MOV   R5,A        ;clear R5
        ADD   A,#79H      ;A=0+79H=79H
        JNC   N_1         ;if no carry, add next number
        INC   R5          ;if CY=1, increment R5
N_1:    ADD   A,#0F5H     ;A=79+F5=6E and CY=1
        JNC   N_2         ;jump if CY=0
        INC   R5          ;If CY=1 then increment R5(R5=1)
N_2:    ADD   A,#0E2H     ;A=6E+E2=50 and CY=1
        JNC   OVER        ;jump if CY=0
        INC   R5          ;if CY=1, increment 5
OVER: MOV   R0,A          ;Now R0=50H, and R5=02
```

Example 3-6

Using the following list file, verify the jump forward address calculation.

```
Line    PC              Opcode                  Mnemonic Operand
01      0000                                    ORG      0000
02      0000    7800                            MOV      R0,#0
03      0002    7455                    MOV     A,#55H
04      0004    6003                    JZ      NEXT
05      0006    08                              INC      R0
06      0007    04              AGAIN:  INC     A
07      0008    04                      INC     A
08      0009    2477            NEXT:   ADD     A,#77h
09      000B    5005                    JNC     OVER
10      000D    E4                      CLR     A
11      000E    F8                      MOV     R0,A
12      000F    F9                      MOV     R1,A
13      0010    FA                      MOV     R2,A
14      0011    FB                      MOV     R3,A
15      0012    2B      OVER:   ADD     A,R3
16      0013    50F2                    JNC     AGAIN
17      0015    80FE    HERE:   SJMP    HERE
18      0017                                    END
```

Example 3-6

Solution:

First notice that the JZ and JNC instructions both jump forward. The target address for a forward jump is calculated by adding the PC of the following instruction to the second byte of the short jump instruction, which is called the relative address. In line 4 the instruction "JZ NEXT" has opcode of 60 and operand of 03 at the addresses of 0004 and 0005. The 03 is the relative address, relative to the address of the next instruction INC R0, which is 0006. By adding 0006 to 3, the target address of the label NEXT, which is 0009, is generated. In the same way for line 9, the "JNC OVER" instruction has opcode and operand of 50 and 05 where 50 is the opcode and 05 the relative address. Therefore, 05 is added to 000D, the address of instruction "CLR A", giving 12H, the address of label OVER.

Example 3-7

Verify the calculation of backward jumps in Example 3-6.

Solution:

In that program list, "JNC AGAIN" has opcode 50 and relative address F2H. When the relative address of F2H is added to 15H, the address of the instruction below the jump, we have 15H + F2H = 07 (the carry is dropped). Notice that 07 is the address of label AGAIN. Look also at "SJMP HERE", which has 80 and FE for the opcode and relative address, respectively. The PC of the following instruction, 0017H, is added to FEH, the relative address, to get 0015H, address of the HERE label (17H + FEH = 15H). Notice that FEH is -2 and 17H + (-2) = 15H. For further discussion of the addition of negative numbers, see Chapter 6.