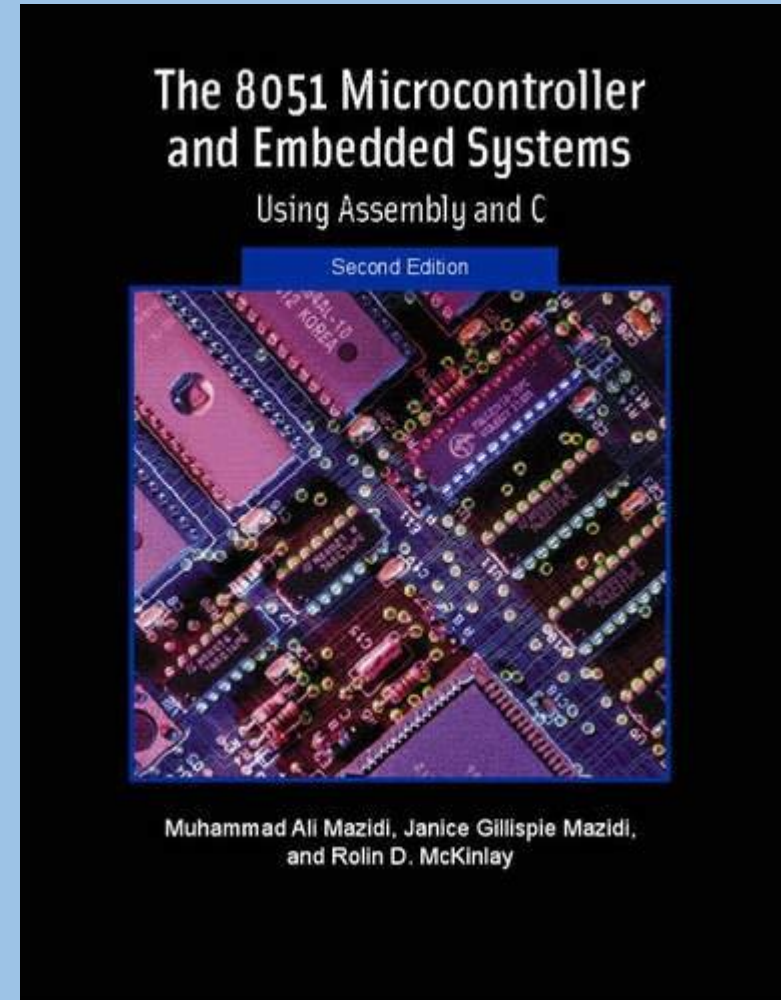


THE 8051 MICROCONTROLLER & Embedded Systems

Muhammad Ali Mazidi, Janice Mazidi
& Rolin McKinlay



I/O PORT PROGRAMMING

Chapter 4

Objectives

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

- >> List the 4 ports of the 8051**
- >> Describe the dual role of port 0 in providing both data and addresses**
- >> Code Assembly language to use the ports for input or output**
- >> Explain the dual role of port 0 and port 2**
- >> Code 8051 instructions for I/O handling**
- >> Code I/O bit-manipulation programs for the 8051**

Objectives

- **Discussing the I/O port programming of the 8051,**
- **Section 4.1, describing I/O access using byte-size data,**
- **Section 4.2, bit manipulation of the I/O ports is discussed in detail.**

Figure 4-1. 8051 Pin Diagram

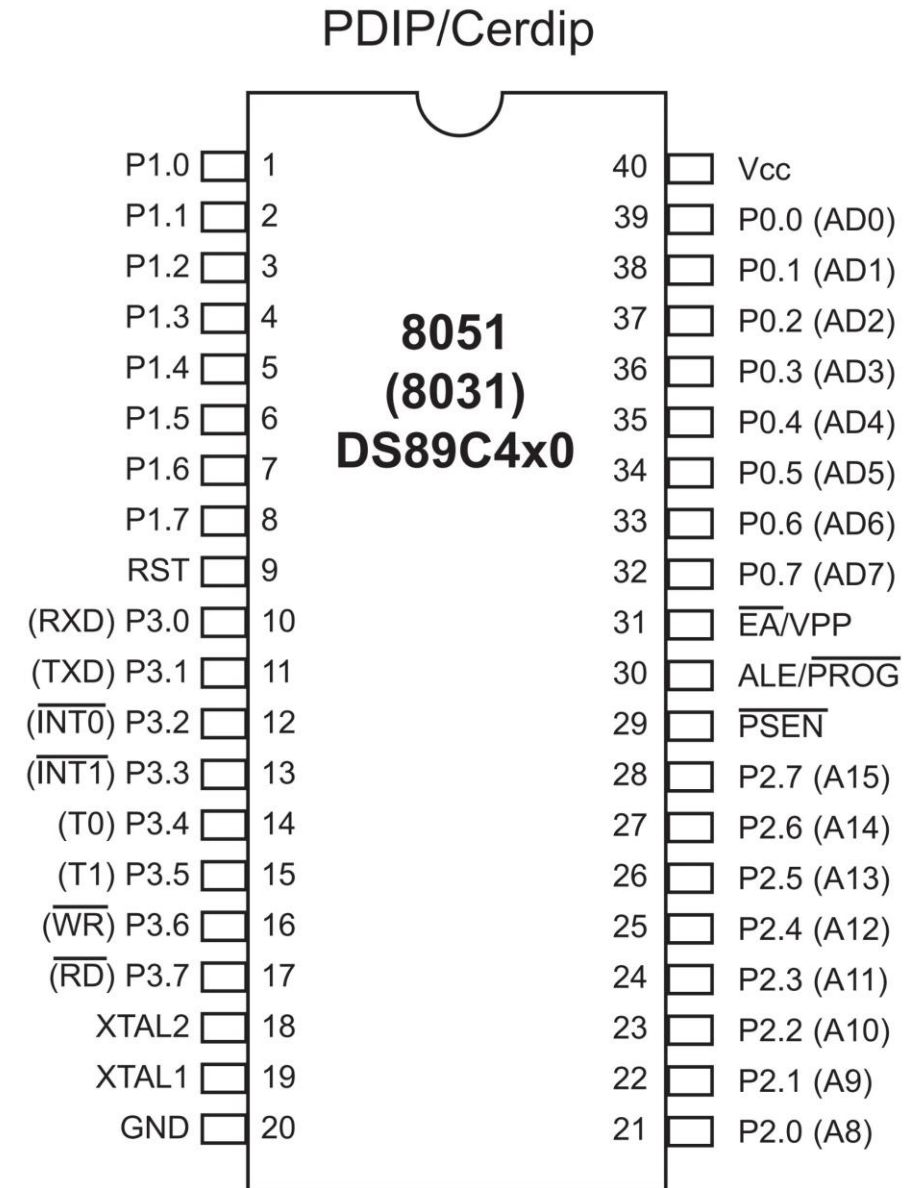


Figure 4-2. Port 0 with Pull-Up Resistors

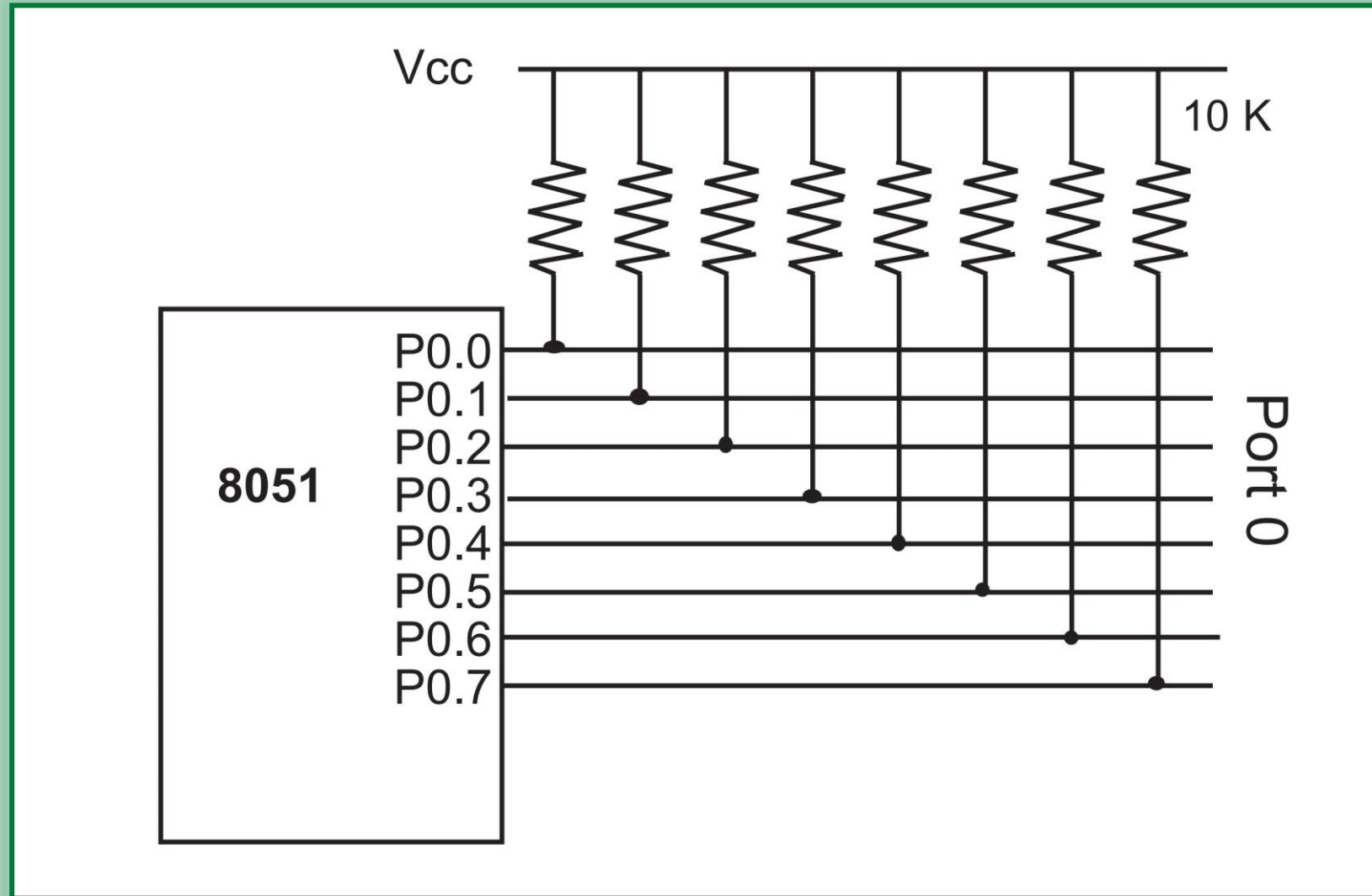


Table 4-1: Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Example 4-1

Write a test program for the DS89C4x0 chip to toggle all the bits of P0, P1, and P2 every 1/4 of a second. Assume a crystal frequency of 11.0592 MHz.

Solution:

```
;Tested for the DS89C4x with XTAL = 11.0592 MHz.
                                ORG          0
BACK:                          MOV          A,#55H
                                MOV          P0,A
                                MOV          P1,A
                                MOV          P2,A
                                ACALL        QSDELAY          ;Quarter of a second delay
                                MOV          A,#0AAH
                                MOV          P0,A
                                MOV          P1,A
                                MOV          P2,A
                                ACALL        QSDELAY
                                SJMP        BACK
;-----1/4 SECOND DELAY
QSDELAY:
                                MOV          R5, #11
H3:                             MOV          R4, #248
H2:                             MOV          R3, #255
H1:                             DJNZ        R3, H1          ;4 MC for DS89C4x0
                                DJNZ        R4, H2
                                DJNZ        R5, H3
                                RET
                                END
```

Delay = $11 \times 248 \times 255 \times 4 \text{ MC} \times 90 \text{ ns} = 250,430 \mu\text{s}$

Use an oscilloscope to verify the delay size.

Table 4-2: Reset Value of Some 8051 Ports

Register	Reset Value (Binary)
P0	11111111
P1	11111111
P2	11111111
P3	11111111

Table 4-3: Single-Bit Addressability of Ports

P0	P1	P2	P3	Port Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Example 4-2

Write the following programs.

- (a) Create a square wave of 50% duty cycle on bit 0 of port 1.
- (b) Create a square wave of 66% duty cycle on bit 3 of port 1.

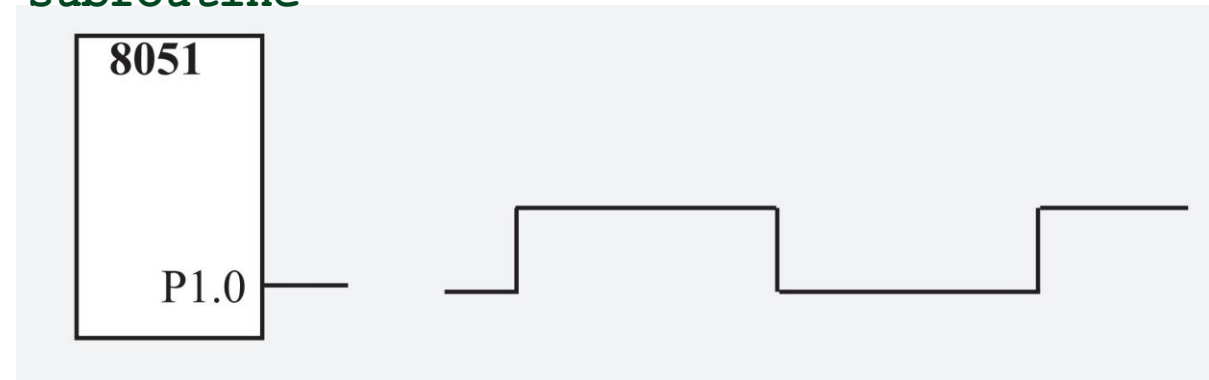
Solution:

- (a) The 50% duty cycle means that the “on” and “off” states (or the high and low portions of the pulse) have the same length. Therefore, we toggle P1.0 with a time delay in between each state.

```
HERE:  SETB    P1.0           ;set to high bit 0 of port 1
        LCALL  DELAY          ;call the delay subroutine
        CLR    P1.0           ;P1.0=0
        LCALL  DELAY
        SJMP   HERE           ;keep doing it
```

Another way to write the above program is:

```
HERE:  CPL     P1.0           ;complement bit 0 of port 1
        LCALL  DELAY          ;call the delay subroutine
        SJMP   HERE           ;keep doing it
```



Example 4-2

Write the following programs.

- (a) Create a square wave of 50% duty cycle on bit 0 of port 1.
- (b) Create a square wave of 66% duty cycle on bit 3 of port 1.

Solution:

- (b) The 66% duty cycle means the “on” state is twice the “off” state.

```
BACK:  SETB    P1.3      ;set port 1 bit 3 high
        LCALL DELAY      ;call the delay subroutine
        LCALL DELAY      ;call the delay subroutine again
        CLR     P1.3      ;clear bit 2 of port 1(P1.3=low)
        LCALL DELAY      ;call the delay subroutine
        SJMP    BACK      ;keep doing it
```

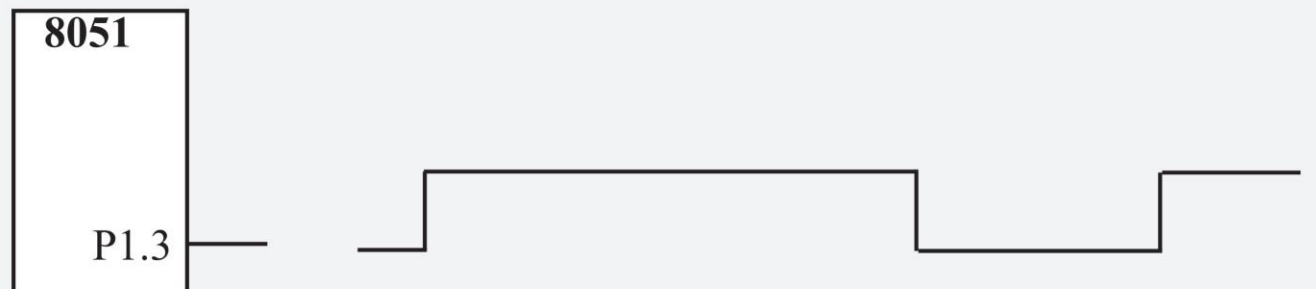


Table 4-4: Single-Bit Instructions

Instruction		Function
SETB	bit	Set the bit (bit = 1)
CLR	bit	Clear the bit (bit = 0)
CPL	bit	Complement the bit (bit = NOT bit)
JB	bit,target	Jump to target if bit = 1 (jump if bit)
JNB	bit,target	Jump to target if bit = 0 (jump if no bit)
JBC	bit,target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

Example 4-3

Write a program to perform the following:

- (a) keep monitoring the P1.2 bit until it becomes high
- (b) when P1.2 becomes high, write value 45H to port 0
- (c) send a high-to-low (H-to-L) pulse to P2.3

Solution:

```
                SETB    P1.2                ;make P1.2 an input
                MOV     A,#45H              ;A=45H
AGAIN:          JNB     P1.2,AGAIN          ;get out when P1.2=1
                MOV     P0,A                ;issue A to P0
                SETB    P2.3                ;make P2.3 high
                CLR     P2.3                ;make P2.3 low for H-to-L
```

In this program, instruction “JNB P1.2,AGAIN” (JNB means jump if no bit) stays in the loop as long as P1.2 is low. When P1.2 becomes high, it gets out of the loop, writes the value 45H to port 0, and creates an H-to-L pulse by the sequence of instructions SETB and CLR.

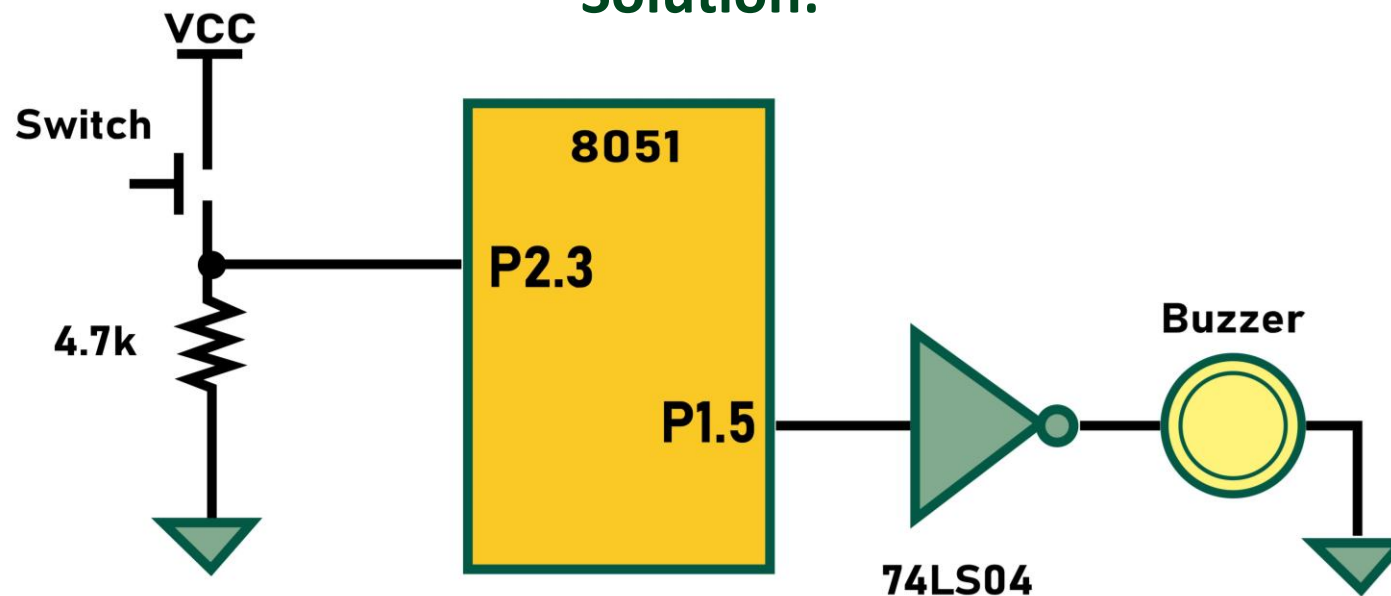
Table 4-5: Instructions For Reading the Status of an Input Port

Mnemonic	Example	Description
MOV A, PX	MOV A, P2	Bring into A the data at P2 pins
JNB PX.Y, ..	JNB P2.1, TARGET	Jump if pin P2.1 is low
JB PX.Y, ..	JB P1.3, TARGET	Jump if pin P1.2 is high
MOV C, PX.Y	MOV C, P2.4	Copy status of pin P2.4 to CY
CJNE A, PX, ..	CJNE A, P1, TARGET	Compare A with P1 and jump if not equal

Example 4-4

Assume that bit P2.3 is an input and represents the condition of an oven. If it goes high, it means that the oven is hot. Monitor the bit continuously. Whenever it goes high, send a high-to-low pulse to port P1.5 to turn on a buzzer.

Solution:



```
HERE: JNB    P2.3, HERE    ;keep monitoring for high
      SETB   P1.5          ;set bit P1.5=1
      CLR    P1.5          ;make high-to-low
      SJMP   HERE          ;keep repeating
```

Example 4-5

A switch is connected to pin P1.7. Write a program to check the status of SW and perform the following:

- (a) If SW=0, send letter 'N' to P2.
- (b) If SW=1, send letter 'Y' to P2.

Solution:

```
                SETB  P1.7          ;make P1.7 an input
AGAIN:          JB    P1.2,OVER      ;jump if P1.7=1
                MOV    P2,#'N'       ;SW=0, issue 'N' to P2
                SJMP   AGAIN          ;keep monitoring
OVER:           MOV    P2,#'Y'       ;SW=1, issue 'Y' to P2
                SJMP   AGAIN          ;keep monitoring
```

Example 4-6

A switch is connected to pin P1.7. Write a program to check the status of the switch and perform the following:

(a) If switch = 0, send letter 'N' to P2.

(b) If switch = 1, send letter 'Y' to P2.

Use the carry flag to check the switch status. This is a repeat of the last example.

Solution:

	SETB	P1.7	;make P1.7 an input
AGAIN:	MOV	C,P1.2	;read the SW status into CF
	JC	OVER	;jump if SW = 1
	MOV	P2,#'N'	;SW = 0, issue 'N' to P2
	SJMP	AGAIN	;keep monitoring
OVER:	MOV	P2,#'Y'	;SW = 1, issue 'Y' to P2
	SJMP	AGAIN	;keep monitoring

Example 4-7

A switch is connected to pin P1.0 and an LED to pin P2.7. Write a program to get the status of the switch and send it to the LED.

Solution:

```
                SETB  P1.7          ;make P1.7 an input
AGAIN:          MOV   C,P1.0        ;read the SW status into CF
                MOV   P2.7,C        ;send the SW status to LED
                SJMP  AGAIN          ;keep repeating
```

Note: The instruction "MOV P2.7, P1.0" is wrong since such an instruction does not exist. However, "MOV P2, P1" is a valid instruction.

Table 4-6: Instructions Reading a Latch (Read-Modify-Write)

Mnemonic		Example	
ANL	Px	ANL	P1, A
ORL	Px	ORL	P2, A
XRL	Px	XRL	P0, A
JBC	PX.Y, TARGET	JBC	P1.1, TARGET
CPL	PX.Y	CPL	P1.2
INC	Px	INC	P1
DEC	Px	DEC	P2
DJNZ	PX.Y, TARGET	DJNZ	P1, TARGET
MOV	PX.Y, C	MOV	P1.2, C
CLR	PX.Y	CLR	P2.3
SETB	PX.Y	SETB	P2.3

Note: x is 0, 1, 2, or 3 for P0 - P3.