I/O PORT PROGRAMMING

The 8051 Microcontroller and Embedded Systems: Using Assembly and C
Mazidi, Mazidi and McKinlay

Chung-Ping Young
楊中平

Home Automation, Networking, and Entertainment Lab

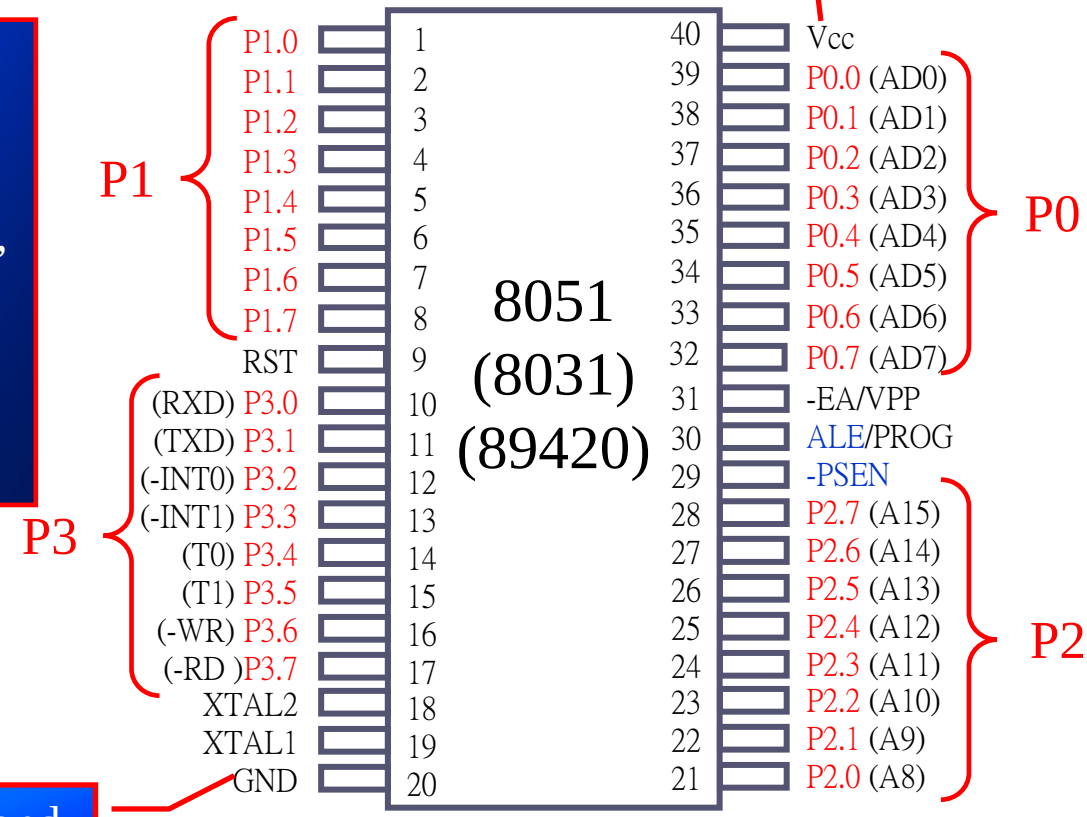
Dept. of Computer Science and Information Engineering
National Cheng Kung University, TAIWAN

I/O PROGRAMMING

8051 Pin Diagram

Provides +5V supply voltage to the chip

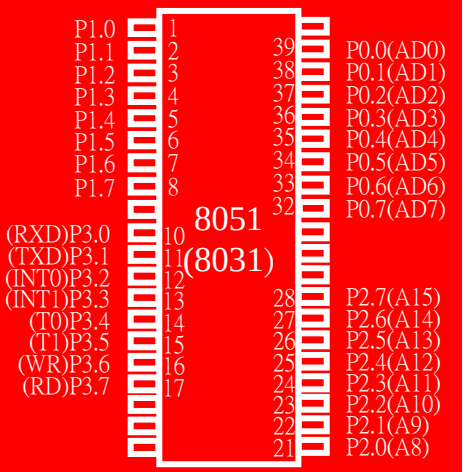
A total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each port takes 8 pins



Grond

I/O PROGRAMMING

I/O Port Pins

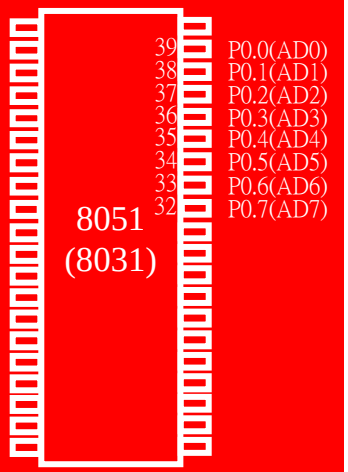


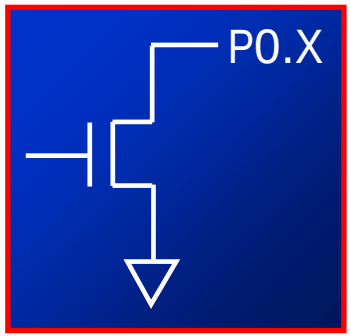
- The four 8-bit I/O ports P0, P1, P2 and P3 each uses 8 pins
- All the ports upon RESET are configured as input, ready to be used as input ports
  - When the first 0 is written to a port, it becomes an output
  - To reconfigure it as an input, a 1 must be sent to the port
    - To use any of these ports as an input port, it must be programmed

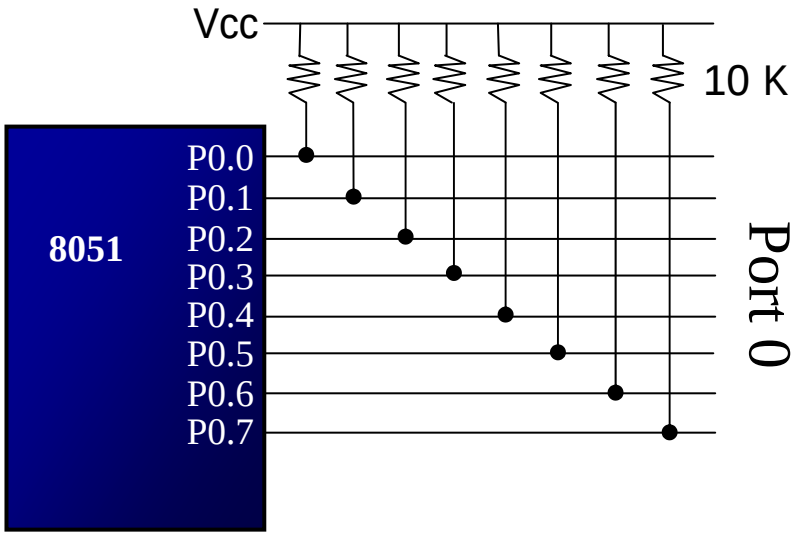
I/O PROGRAMMING

Port 0

- It can be used for input or output, each pin must be connected externally to a 10K ohm pull-up resistor
  - This is due to the fact that P0 is an open drain, unlike P1, P2, and P3
    - *Open drain* is a term used for MOS chips in the same way that open collector is used for TTL chips





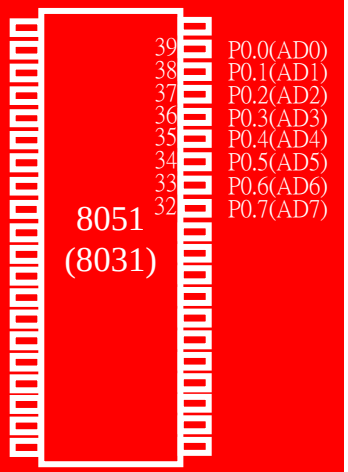


I/O PROGRAMMING

Port 0 (cont')

The following code will continuously send out to port 0 the alternating value 55H and AAH

```
BACK:   MOV     A,#55H
        MOV     P0,A
        ACALL   DELAY
        MOV     A,#0AAH
        MOV     P0,A
        ACALL   DELAY
        SJMP    BACK
```



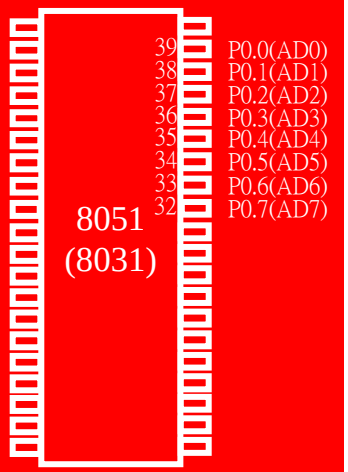
I/O PROGRAMMING

Port 0 as Input

- In order to make port 0 an input, the port must be programmed by writing 1 to all the bits

Port 0 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

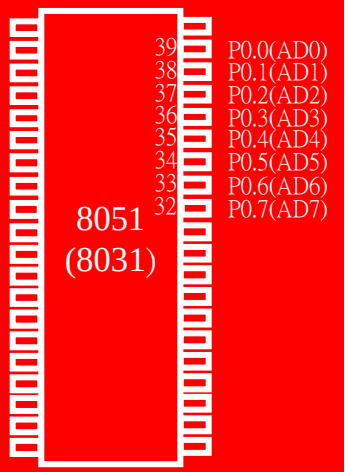
```
        MOV     A,#0FFH         ;A=FF hex
        MOV     P0,A            ;make P0 an i/p port
                                ;by writing it all 1s
BACK:   MOV     A,P0            ;get data from P0
        MOV     P1,A            ;send it to port 1
        SJMP    BACK            ;keep doing it
```



I/O PROGRAMMING

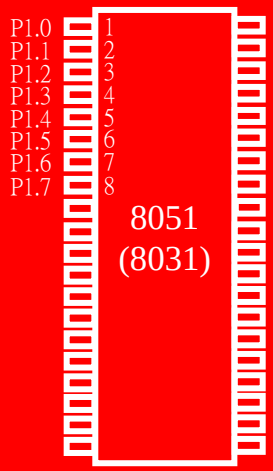
Dual Role of Port 0

- Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data
  - When connecting an 8051/31 to an external memory, port 0 provides both address and data



Port 1

- Port 1 can be used as input or output
  - In contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally
  - Upon reset, port 1 is configured as an input port



The following code will continuously send out to port 0 the alternating value 55H and AAH

```
        MOV     A,#55H
BACK:   MOV     P1,A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

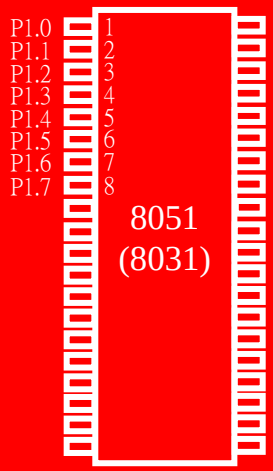
I/O PROGRAMMING

Port 1 as Input

- To make port 1 an input port, it must be programmed as such by writing 1 to all its bits

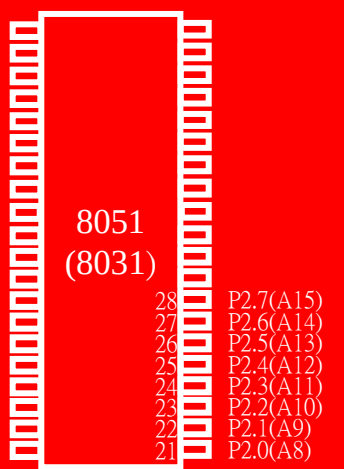
Port 1 is configured first as an input port by writing 1s to it, then data is received from that port and saved in R7 and R5

```
MOV     A,#0FFH     ;A=FF hex
MOV     P1,A        ;make P1 an input port
                    ;by writing it all 1s
MOV     A,P1        ;get data from P1
MOV     R7,A        ;save it to in reg R7
ACALL   DELAY       ;wait
MOV     A,P1        ;another data from P1
MOV     R5,A        ;save it to in reg R5
```



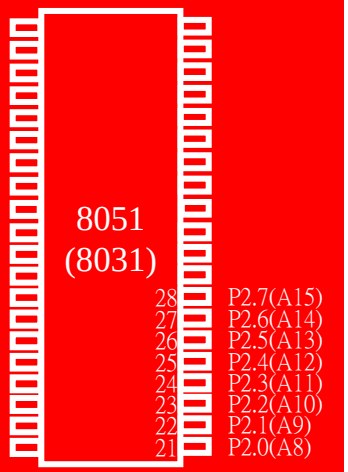
Port 2

- Port 2 can be used as input or output
  - Just like P1, port 2 does not need any pull-up resistors since it already has pull-up resistors internally
  - Upon reset, port 2 is configured as an input port



I/O PROGRAMMING

Port 2 as Input or Dual Role

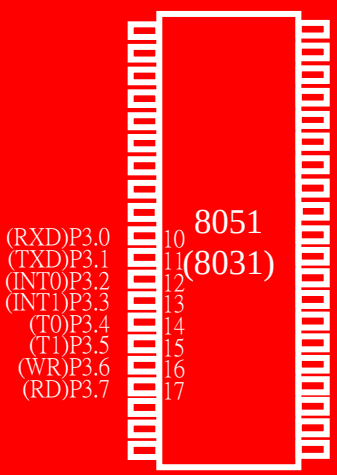


- To make port 2 an input port, it must be programmed as such by writing 1 to all its bits
- In many 8051-based system, P2 is used as simple I/O
- In 8031-based systems, port 2 must be used along with P0 to provide the 16-bit address for the external memory
  - Port 2 is also designated as A8 – A15, indicating its dual function
  - Port 0 provides the lower 8 bits via A0 – A7

I/O PROGRAMMING

Port 3

- Port 3 can be used as input or output
  - Port 3 does not need any pull-up resistors
  - Port 3 is configured as an input port upon reset, this is not the way it is most commonly used



I/O PROGRAMMING

Port 3 (cont')

- Port 3 has the additional function of providing some extremely important signals

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

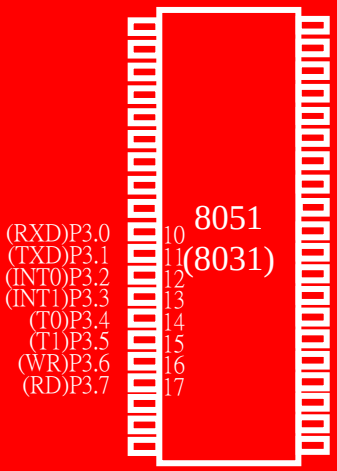
Serial communications

External interrupts

Timers

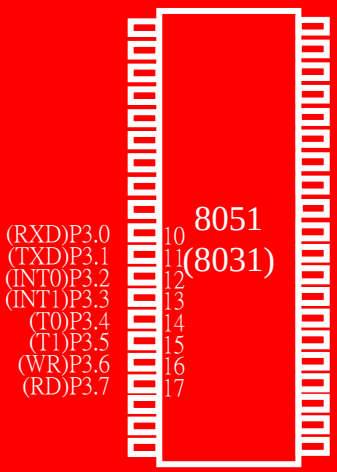
Read/Write signals of external memories

In systems based on 8751, 89C51 or DS89C4x0, pins 3.6 and 3.7 are used for I/O while the rest of the pins in port 3 are normally used in the alternate function role



I/O PROGRAMMING

Port 3 (cont')



Write a program for the DS89C420 to toggle all the bits of P0, P1, and P2 every 1/4 of a second

```
        ORG     0
BACK:   MOV     A,#55H
        MOV     P0,A
        MOV     P1,A
        MOV     P2,A
        ACALL   QSDELAY     ;Quarter of a second
        MOV     A,#0AAH
        MOV     P0,A
        MOV     P1,A
        MOV     P2,A
        ACALL   QSDELAY
        SJMP    BACK
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
$= 11 \times 248 \times 255 \times 4$ MC $\times$ 90 ns
$= 250{,}430\ \mu s$

I/O PROGRAMMING

Different ways of Accessing Entire 8 Bits

The entire 8 bits of Port 1 are accessed

```
BACK:   MOV     A,#55H
        MOV     P1,A
        ACALL   DELAY
        MOV     A,#0AAH
        MOV     P1,A
        ACALL   DELAY
        SJMP    BACK
```

Rewrite the code in a more efficient manner by accessing the port directly without going through the accumulator

```
BACK:   MOV     P1,#55H
        ACALL   DELAY
        MOV     P1,#0AAH
        ACALL   DELAY
        SJMP    BACK
```

Another way of doing the same thing

```
        MOV     A,#55H
BACK:   MOV     P1,A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

I/O BIT MANIPULATION PROGRAMMING

I/O Ports and Bit Addressability

- Sometimes we need to access only 1 or 2 bits of the port

```
BACK:   CPL     P1.2            ;complement P1.2
        ACALL   DELAY
        SJMP    BACK

;another variation of the above program
AGAIN:  SETB    P1.2            ;set only P1.2
        ACALL   DELAY
        CLR     P1.2            ;clear only P1.2
        ACALL   DELAY
        SJMP    AGAIN
```

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.
Create a square wave of 50% duty cycle on bit 0 of port 1.

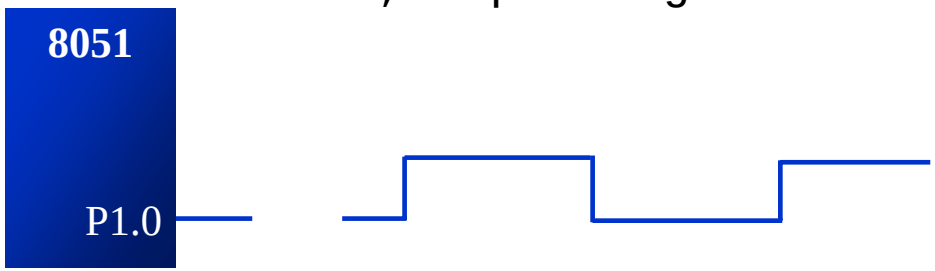
Solution:

The 50% duty cycle means that the "on" and "off" state (or the high and low portion 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   ;set to high bit 0 of port 1
       LCALL  DELAY  ;call the delay subroutine
       SJMP   HERE   ;keep doing it
```



- Instructions that are used for signal-bit operations are as following

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)

I/O BIT MANIPULATION PROGRAMMING

Checking an Input Bit

- The JNB and JB instructions are widely used single-bit operations
  - They allow you to monitor a bit and make a decision depending on whether it's 0 or 1
  - These two instructions can be used for any bits of I/O ports 0, 1, 2, and 3
    - Port 3 is typically not used for any I/O, either single-bit or byte-wise

Instructions for Reading an Input Port

Mnemonic	Examples	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 P1.3 is high
MOV C,PX.Y	MOV C,P2.4	Copy status of pin P2.4 to CY

I/O BIT MANIPULATION PROGRAMMING

Checking an Input Bit (cont')

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
```

I/O BIT MANIPULATION PROGRAMMING

Checking an Input Bit (cont')

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
```

I/O BIT MANIPULATION PROGRAMMING

Checking an Input Bit (cont')

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
```

I/O BIT MANIPULATION PROGRAMMING

Reading Single Bit into Carry Flag

Example 4-6

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
Use the carry flag to check the switch status.

Solution:

```
        SETB P1.7       ;make P1.7 an input
AGAIN:  MOV  C,P1.2     ;read 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
```

I/O BIT MANIPULATION PROGRAMMING

Reading Single Bit into Carry Flag (cont')

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 SW status into CF
        MOV  P2.7,C      ;send SW status to LED
        SJMP AGAIN       ;keep repeating
```

The instruction '`MOV P2.7,P1.0`' is wrong , since such an instruction does not exist

However '`MOV P2,P1`' is a valid instruction

I/O BIT MANIPULATION PROGRAMMING

Reading Input Pins vs. Port Latch

- In reading a port
  - Some instructions read the status of port pins
  - Others read the status of an internal port latch
- Therefore, when reading ports there are two possibilities:
  - Read the status of the input pin
  - Read the internal latch of the output port
- Confusion between them is a major source of errors in 8051 programming
  - Especially where external hardware is concerned

READING INPUT PINS VS. PORT LATCH

Reading Latch for Output Port

- Some instructions read the contents of an internal port latch instead of reading the status of an external pin
  - For example, look at the `ANL P1,A` instruction and the sequence of actions is executed as follow
    1. It reads the internal latch of the port and brings that data into the CPU
    2. This data is ANDed with the contents of register A
    3. The result is rewritten back to the port latch
    4. The port pin data is changed and now has the same value as port latch

READING INPUT PINS VS. PORT LATCH

Reading Latch for Output Port (cont')

- *Read-Modify-Write*
  - The instructions read the port latch normally read a value, perform an operation then rewrite it back to the port latch

Instructions Reading a latch (Read-Modify-Write)

Mnemonics	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

- The ports in 8051 can be accessed by the Read-modify-write technique
  - This feature saves many lines of code by combining in a single instruction all three actions
    1. Reading the port
    2. Modifying it
    3. Writing to the port

```
        MOV    P1,#55H   ;P1=01010101
AGAIN:  XRL    P1,#0FFH  ;EX-OR P1 with 1111 1111
        ACALL  DELAY
        SJMP   BACK
```