Microprocessor

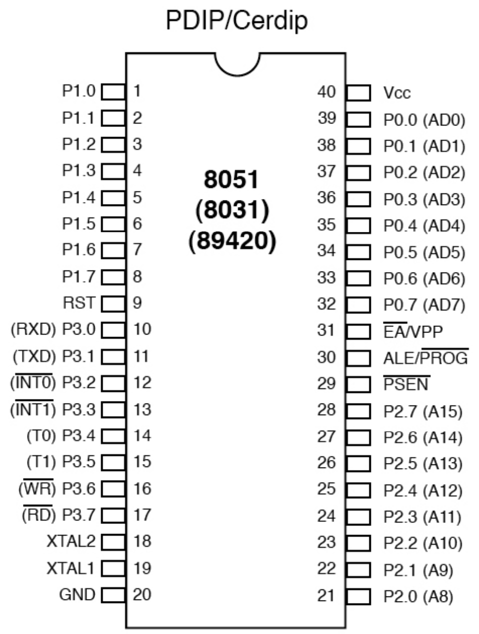
Ch.4 I/O Ports

OUTLINE

- **8051 I/O programming**
- **I/O bit manipulation programming**

I/O PORT

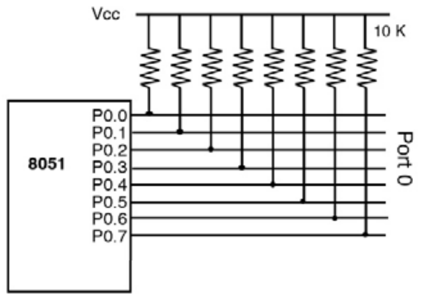
- **I/O Port**
  - 8051 has 40 pins
  - 32 pins are used for I/O ports
    - 4 I/O ports: P0, P1, P2, P3
    - Each port has 8 bits (8 pins)
- **Input mode and output mode**
  - When power on, all ports are used as input by default.
    - You can read data from the port
  - Change a port to output mode
    - Write all 0s to a port will change it into output mode (MOV P0, #00H).
  - If the ports are in output mode, and you want to read data from it, you must change it to input mode first
    - Write all 1s to a port will change it into input mode (MOV P1, #FFH)



$$\begin{tikzcd}
\boxed{\text{MOV P0, \#00H}} \arrow[r] & \boxed{\text{write instructions}} \arrow[r] & \boxed{\text{MOV P0, \#FFH}} \arrow[r] & \boxed{\text{read instructions}} \arrow[lll, bend left]
\end{tikzcd}$$

I/O PORTS: P0

- **Port 0**
  - To use port 0 as both input and output, each pin must be connected externally to 10K-Ohm pull-up resistor. (Open drain)
  - Port 0 as output



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

  - Port 0 as input
    - After P0 being used as output, we can switch it back to input mode by writing 1 to all the bits

```
        MOV A, #0FFH
        MOV P0, A           ; make P0 an input port
        MOV P1, #00H        ; make P1 an output port (optional)
BACK:   MOV A, P0           ; get data from P0
        MOV P1, A           ; sent it to port 1
        SJMP BACK
```

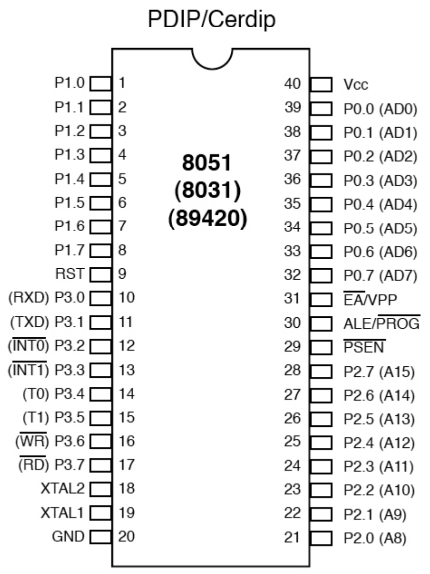
What will happen if we read a port while it's in output mode?

I/O PORTS: P0

- **Input mode and output mode**
  - If a port has been used as output, we must change it to input mode before we can read data from it
    - E.g. 1
      ```
      MOV P0, #23H     ; output
      MOV A, P0        ; input, invalid
      ```
    - E.g. 2
      ```
      MOV P0, #23H     ; output
      MOV P0, #0FFH    ; change P0 to input mode
      MOV A, P0        ; valid
      ```
  - At any moment, you can always write data to a port regardless it has been used as input or output in the previous instructions
    - The change to output mode is optional
    - E.g.
      ```
      MOV P0, #25H     ; P0 used as output
      MOV P0, #0FFH    ; change P0 to input mode
      MOV A, P0        ; P0 used as input
      MOV P0, #23H     ; P0 used as output (valid)
      ```

I/O PORTS: PORT 1

- **Port 1, 2, 3**
  - P1, P2, P3 can be used as both input and output
  - They do NOT require pull-up resistors (it has pull-up resistors inside the chip)
  - When power on, the are input ports by default
- **Dual roles of ports**
  - P0, P1, P2, P3 can be used as general I/O ports. They can also be used for some specific operations.
    - P0: when external memory is connected to 8051, we usually use Port 0 to serve as the interface for both address bus and data bus (AD0 – AD7)
    - P2: For system with larger external memory , P2 is used to serve as the interface for the high byte of address (A8 – A15)
    - P3: P3 is usually used to provide interrupt signals.



OUTLINE

- 8051 I/O programming
- I/O bit manipulation programming

BIT MANIPULATION

- **I/O port bit manipulation**
  - We can access each individual bit of the I/O port
  - E.g. the 3rd bit of P3: P3.2
  - Example

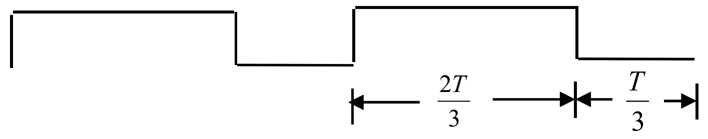
```
BACK:           SETB P1.2         ; set P1.2 to 1
                ACALL DELAY
                CPL P1.2          ; complement P1.2
                SJMP BACK
```

  - The ability to access single bit of I/O ports is one of the most powerful features of 8051.
    - It greatly increases program flexibility and is one of the main reasons many designers choose 8051.

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

BIT MANIPULATION

- **Example**
  - Create a square wave of 66% duty cycle on bit 3 of port 1.



BIT MANIPULATION: CONDITIONAL JUMP

- **Conditional jump**
  - We can jump to a location based on the value of a particular bit
  - Three instructions: JB, JNB, JBC
  - JB: (jump if bit)
    - JB *bit, target*
    - Jump if bit = 1
    - Example: JB P2.4 HERE
  - JNB: (jump if no bit)
    - JNB *bit, target*
    - Jump if bit = 0
    - Example: JNB P1.3 HERE
  - JBC: (jump if bit, then clear)
    - JBC *bit, target*
    - (1) Jump if bit = 1, (2) then clear bit
    - Example: JBC P0.4 HERE ; after execution, P0.4 will be 0

The bit must be in input mode while using the conditional jump!

BIT MANIPULATION: CONDITIONAL JUMP

- **Example**
  - Write a program to perform the following
    - Keep monitoring P1.2 bit until it becomes high
    - When P1.2 becomes high, write value 45H to port 0
    - Send a high-to-low pulse to P2.3

```
          SETB P1.2           ; change P1.2 to input mode

HERE:     JNB P1.2, HERE
          MOV P0, #45H

          CLR P2.3            ; change P2.3 to output mode
          SETB P2.3
          CLR P2.3
```

BIT MANIPULATION: CONDITIONAL JUMP

- **Example**
  - A switch is connected to P1.7. Write a program to check the status of the switch and perform the following
    - If SW = 0, send the ASCII code of letter ‘N’ to P2
    - If SW = 1, send the ASCII code of letter ‘Y’ to P2

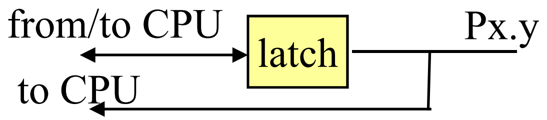
```
                SETB P1.7           ; make P1.7 as input
START:          JB P1.7 ONE         ; jump if SW = 1
                ACALL WRITE_N       ; if SW = 0
                SJMP START
ONE:            ACALL WRITE_Y       ; if SW = 1
                SJMP START
;-----------------------------------------------
                ORG 300H
WRITE_N:        MOV P2, #’N’        ; write the ASCII code of ‘N’ to P2
                RET
;-----------------------------------------------
                ORG 310H
WRITE_Y:        MOV P2, #’Y’        ; write the ASCII code of ‘Y’ to P2
                RET
;-----------------------------------------------
```

BIT MANIPULATION: CARRY FLAG

- **Read a single bit into the carry flag**
  - We can directly move a bit into carry flag in is the PSW register
  - MOV C, P1.2 ; read the value of P1.2 and save it into carry flag.
  - Example:
    - 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

```
                    SETB P1.0           ; set P1.7 to input mode
                    CLR P2.7
AGAIN:              MOV C, P1.0         ; read P1.0 into C
                    MOV P2.7, C         ; send C to P2.7
                    SJMP AGAIN
```

BIT MANIPULATION: LATCH



- **Latch and port**
  - Each pin is connected to a latch inside 8051
  - Review: latch is an digital device that can store one bit of information.
    - If you write to a port (e.g. MOV P0.3, C), the value will be first written to the latch, then the contents of the latch will change the signal at the pin.
    - If you read from a port (e.g. MOV C, P0.3), you need to write '1' to the port to change it to input mode, then the signal will be directly read to the CPU without using the latch.
- **Read-Write-Modify instructions**
  - Read the contents in latch (read) → change its value and write it back to latch (write) → the value in latch will change the signal at pin

  - E.g. CPL P1.2 ; complement the value of Pin P1.2
  - The execution of the instruction incurs the following sequence of actions
    - Reads the internal latch of the port, and brings that data into the CPU.
    - This data is complemented
    - The result is written back to the port latch
    - The port pin data is changed and now has the same value of port latch.

BIT MANIPULATION: LATCH

- **Read-Modify-Write instructions (Cont'd)**
  - XRL P1, A ; exclusive or logic
  - The execution of the instruction incurs the following sequence of actions
    - Reads the internal latch of the port, and brings that data into the CPU.
    - This data is EX-ORed with the contents of register A
    - The result is written back to the port latch
    - The port pin data is changed and now has the same value of port latch.
  - We can read the contents of port latch while it's in output mode.
  - Example

```
            MOV P1, #55H
            MOV A, #0FFH
AGAIN:      XRL P1, A             ;EX-OR P1 with 11111111
            ACALL DELAY
            SJMP AGAIN
```