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)
**I/O PORTS: P0**

- **Port 0**
  - To use port 0 as both input and output, each pin must be connected externally to a 10K-Ohm pull-up resistor (Open drain)
  - Port 0 as output
    
    ```
    MOV A, #55H
    MOV P0, A
    ACALL DELAY
    MOV A, #0AAH
    MOV P0, A
    ACALL DELAY
    SJMP BACK
    ```
  - Port 0 as input
    - After P0 being used as output, if we want to use it as input, we must switch it back to input mode by writing 1 to all the bits
      ```
      MOV P0, #23H ; output 23H at P0
      MOV P0, #0FFH ; make P0 an input port
      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** in the previous instruction, 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

  • When power on, **P0 is in input mode by default**
  
  - At any moment, you can always write data to a port regardless it has been used as input or output in the previous instructions
  
  • 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, 2, 3

- Port 1, 2, 3
  - P1, P2, P3 can be used as both input and output
  - They do NOT require external pull-up resistors (they have internal pull-up resistors built inside the chip)
  - When power on, they 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.
• 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.
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 PSW.2, HERE` ; after execution, PSW.2 (overflow flag) ; will be 0
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
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
BIT MANIPULATION: CARRY FLAG

• **Read a single bit into the carry flag**
  – We can directly move a bit into carry flag in 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
      AGAIN: MOV C, P1.0 ; read P1.0 into C
                  MOV P2.7, C ; send C to P2.7
                  SJMP AGAIN
      ```

      • MOV P2.7, P1.0 is illegal !!
      • MOV P2, P1 is valid.
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.

<table>
<thead>
<tr>
<th>E/C</th>
<th>D</th>
<th>Q</th>
<th>(\bar{Q})</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>(Q_{\text{prev}})</td>
<td>(\bar{Q}_{\text{prev}})</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
</tbody>
</table>

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**.
BIT MANIPULATION: LATCH

Read latch

Internal CPU bus

Write to latch

Read pin

D       Q
P1.X
Clk      Q

TB2

Vcc

Load(L1)

M1

P1.X pin
BIT MANIPULATION: LATCH

- Writing ‘1’ to a pin

1. write a 1 to the pin

Internal CPU bus

Write to latch

Read latch

TB2

1

D Q
P1.X
Clk Q

Vcc

Load(L1)

P1.X pin

output 1

2. output pin is Vcc

Read pin

TB1

0
BIT MANIPULATION: LATCH

- Writing ‘0’ to a pin

1. write a 0 to the pin
2. output pin is ground

What will happen if we read a port while it’s in output mode?
- If it was written a “0”, CPU will always get “0” regardless of the status of the input pin.
- It can also damage the port since the pin is directly connected to ground!
Reading Input Pins vs. Port Latch

• **When reading ports, there are two possibilities:**
  – Read the status of the input pin.
  – Read the internal latch of the output port.

• **Instructions for reading input ports**
  – Must configure the port bits as input first
  – Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV A, PX</td>
<td></td>
</tr>
<tr>
<td>JNB PX.Y, Target</td>
<td></td>
</tr>
<tr>
<td>JB PX.Y, Target</td>
<td></td>
</tr>
<tr>
<td>MOV C, PX.Y</td>
<td></td>
</tr>
</tbody>
</table>

• **Reading the content of an internal port latch**
  – We can read the contents of port latch while it’s in **output mode**.
  – Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBC/DJNZ PX.Y, Target</td>
<td></td>
</tr>
<tr>
<td>MOV PX.Y, C</td>
<td></td>
</tr>
<tr>
<td>ANL/ORL/XRL PX, Addr/number</td>
<td></td>
</tr>
<tr>
<td>CPL/CLR/SETB PX.Y</td>
<td></td>
</tr>
<tr>
<td>INC/DEC PX</td>
<td></td>
</tr>
</tbody>
</table>
**Reading Latch for Output Port**

- **Read-Write-Modify instructions**
  - Read the contents in latch (read) \(\rightarrow\) change its value and write it back to latch (write) \(\rightarrow\) 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.

- Example

  ```
  MOV P1, #0FFH    ; change P1 to input mode
  MOV A, P1             ; read the value from the pins of P1
  MOV P1, #55H       ; output mode, 55H is written to the latch of P1, then the pins
  AGAIN:  CPL P1.1              ; Complement the value of P1.1 (reading the value of latch)
  ACALL DELAY
  SJMP AGAIN
  ```