Lab 7: Interrupt & Timer

- **Link**
  - Document
  - Video(Interrupts)
  - Video(Timer)

- **Basic** (50%)
  - **Description**

    Design and implement a 4-bit bidirectional counter using Assembly language. The counter must display its binary value through four LEDs, with the counting direction controlled by a single push-button.

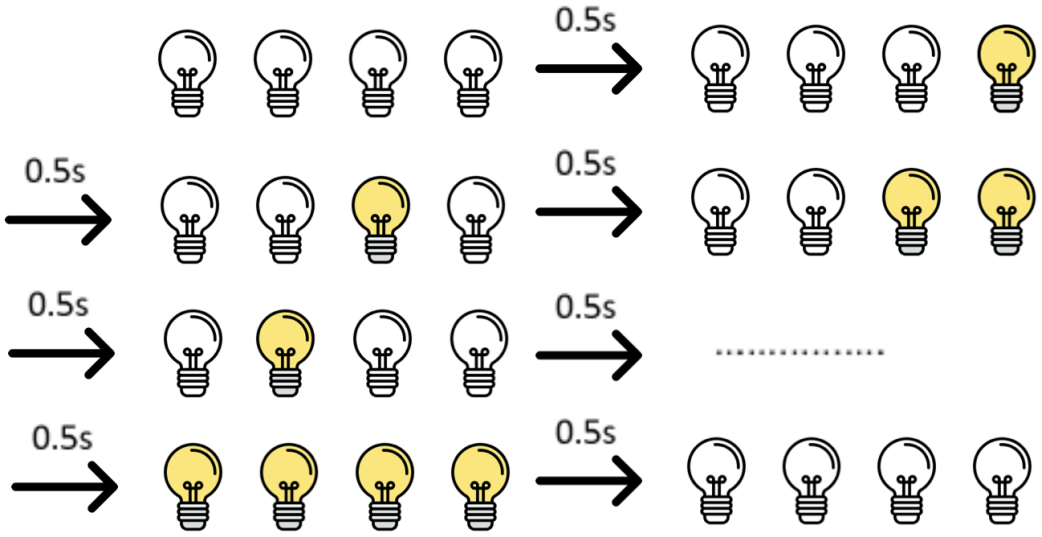
    The counter has two operating states.
    - **State 1 (Count-Up):** Cyclically increments the count from 0 (0000_2) to 15 (1111_2).
    - **State 2 (Count-Down):** Cyclically decrements the count from 15 (1111_2) to 0 (0000_2).

    A **push-button** is used to toggle between the two states.
    Each time the button is pressed, the counter must reset to the initial value of the newly selected state (**0** for count-up, **15** for count-down).

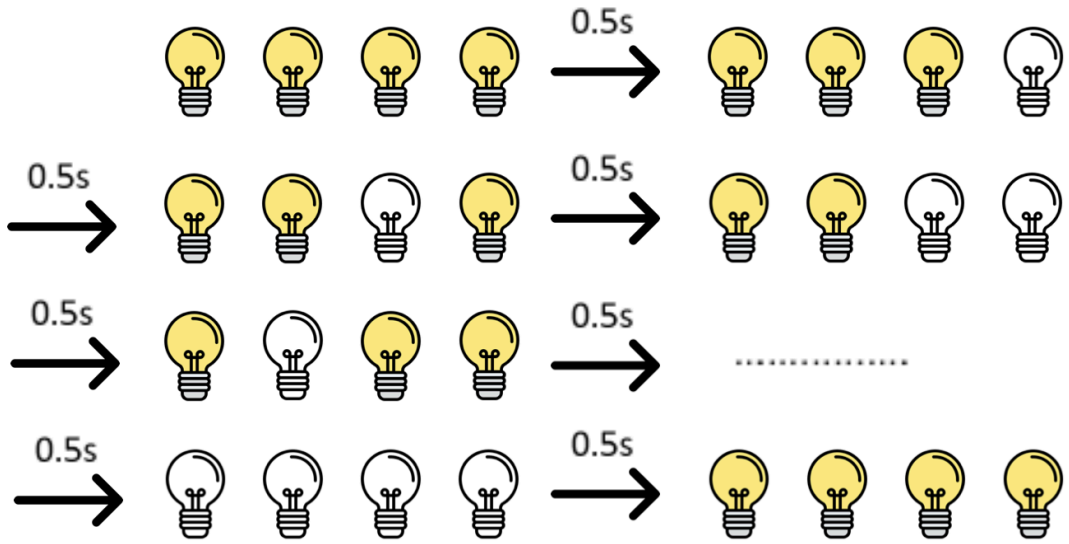
    The 4-bit output should be displayed on LEDs connected to RD7–RD4, with each count value held for exactly 0.5 seconds. The delay should be implemented using a delay macro or subroutine. Upon power-up, the counter must default to State 1 (Count-Up).

  - **State demonstration**

    State 1:



State 2:



- **Standard of grading**
  1. Proper implementation of delay using a macro or a subroutine.
  2. Correct operation of 4-bit output via LEDs RD7–RD4.
  3. Your program must be implemented in Assembly.

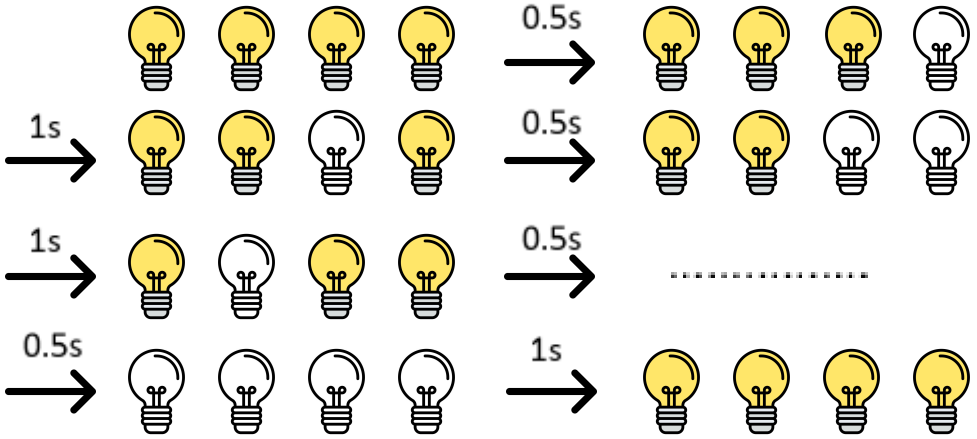
- **Advance** (30%)
  - **Description**

    Develop a **cyclic down counter** that continuously counts from **0x0F** down to **0x00**, then wraps around to 0x0F to repeat the cycle. The counter output must be displayed on **RD7–RD4** LEDs.

    The delay between consecutive counts must alternate as follows:
    - 0.5 seconds for one step,
    - 1.0 second for the next step,
    - Then repeat this alternating pattern continuously.

    The delay must be implemented using **TIMER2**, and the **DELAY macro is not allowed**.

  - **State demonstration**



- **Standard of grading**
  1. No usage of the DELAY macro (must use TIMER2)
  2. Proper output display through RD7–RD4 LEDs
  3. Your program must be implemented in Assembly.

- **Hard** (20%)
  - **Description**

    You will use four LEDs and one push-button to design a cyclic counter with two operational states. The LEDs are connected to RD7, RD6, RD5, and RD4 (from left to right), representing bits 3 to 0 of a 4-bit binary value. The push-button is connected to RB0.

    Design and implement a state-based cyclic counter controlled by one push-button. The counter must operate in two distinct modes in a cyclic manner. Pressing the push-button toggles between these two states. When switching states, the counter must always restart from the initial value corresponding to the new mode.

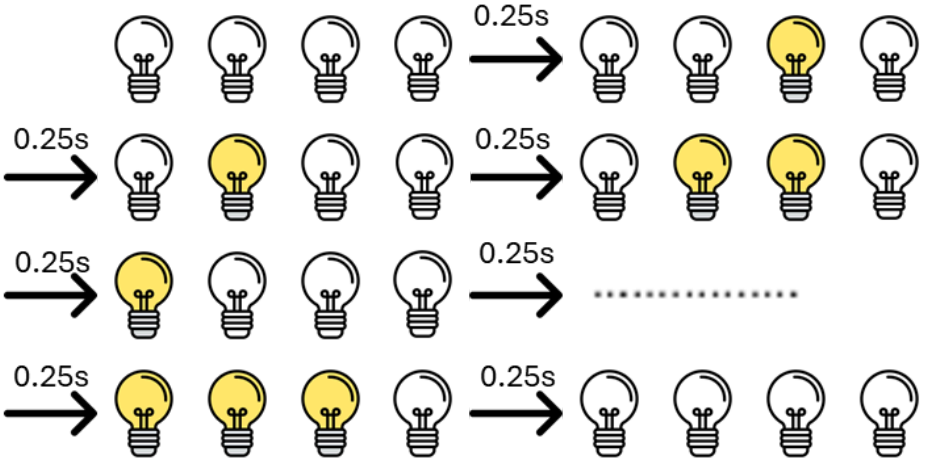
    State 1
    - Counter sequence: 0, 2, 4, 6, …, 14, 0, 2, … (even numbers)
    - Interval: 0.25 seconds

    State 2
    - Counter sequence: 1, 3, 5, 7, …, 15, 1, 3, … (odd numbers)
    - Interval: 1 second

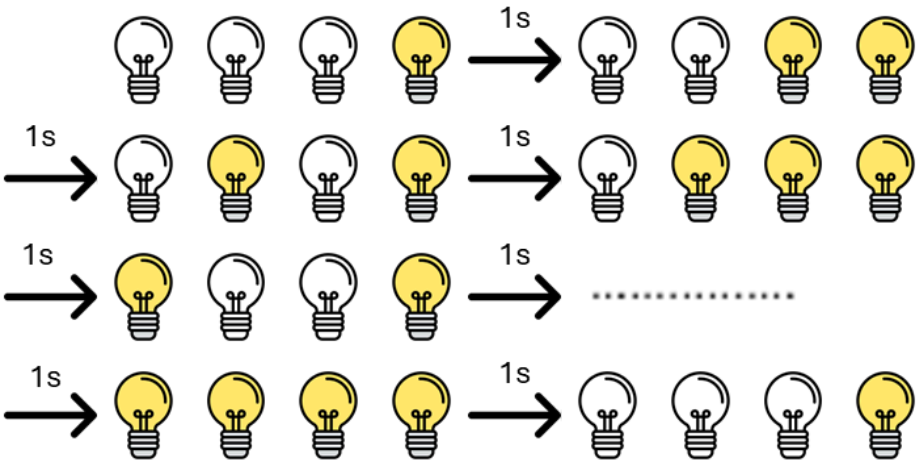
    State transitions:
    - Initial → State 1
    - Button click → State 2
    - Button click → State 1 (and so on)

  - **State demonstration**

    State 1:



State 2:



- **Standard of grading**
  1. Use ISR to handle the button event.
  2. Button: RB0, 4 LEDs: RD7-RD4.
  3. Your program must be implemented in Assembly.

- Note: If you have any questions, please send an email to the TA mailbox (mprocessor2025@gmail.com). Emails sent to individual TAs will not be answered.