Programming Assignment 2

Write an ARMv7 assembly program that implements a **two-digit up/down counter (00–59)** displayed on the **HEX1:HEX0** displays of the DE1-SoC (in CPUlator). The counter should respond to **pushbutton (KEY) inputs** and **switch control** while using proper **debouncing** to ensure stable operation.

Background

In embedded systems, mechanical pushbuttons do not make a clean electrical transition when pressed or released. The signal "bounces" between 0 and 1 for a few milliseconds. Without debouncing, a single button press may appear as multiple presses, causing erratic counting or toggling.

To avoid this, software debouncing ensures that a key press is recognized only after the signal is stable for a short time (5–10 ms).

Problem Statement

Implement a counter that behaves as follows:

Control	**Function**
KEY0	Start/Stop toggle — each press toggles between run and pause
KEY1	Reset counter to 00
SW0	Direction control — 0: count up, 1: count down

The counter must:

- Increment or decrement every second when running.
- Pause when stopped.
- Wrap correctly:
  - **Up**: 00 → 59 → 00
  - **Down**: 00 → 59 → 00
- Show the current value on **HEX1:HEX0** using **active-low encoding**.

Functional Requirements

1. **Start/Stop Toggle**
   - Each press of KEY0 toggles between "running" and "paused" states.
   - Must use **debouncing** to ensure only one toggle per press.
2. **Reset**
   - Pressing KEY1 resets counter to 00 immediately.

    - Also uses debouncing to prevent multiple resets.

3. **Direction Control**
    - Controlled by SW0 (read in each cycle).
    - SW0 = 0 → count up
    - SW0 = 1 → count down
4. **Counting**
    - Update once per second when running.
    - Wrap between 00 and 59.
    - Pause when stopped.
5. **Display**
    - Show the count on HEX1:HEX0 using a **7-segment lookup table** (digits 0–9).
    - Convert active-HIGH table to **active-LOW** by inverting bits before writing.

Debouncing

Before implementing the program, research and explain:

1. What mechanical switch bounce is and why it occurs.
2. What active-LOW means for the DE1-SoC KEY inputs.
3. How software debouncing works — including time delays, confirm-after-delay checks, and optional "wait for release."

Include a short section in your README.md summarizing:

- Why debouncing is essential in embedded systems.
- The method you implemented (with delay constant and reasoning).

Submission Guidelines

1. Upload the following files to your **GitHub repo**:
    - pa2_updown_counter.s
    - README.md including:
        - Short summary of debouncing and method used
        - YouTube (unlisted) video link showing your simulation
2. Submit your **repository URL** as your PA2 submission.