ELEC 335 - Lab #2

Objective

The objective of this lab is to

- practice assembly language, and write programs that will require control flow and careful construction,
- work with seven segment displays,
- work with the debugging tools for analyzing program flow.

Setup

Install STM32CubeIDE and implement problems on the breadboard. For that, use the `asm` project from the g0 project repo (https://github.com/fcayci/stm32g0) and follow https://youtu.be/UYk_NAnDJlw. You can use one of the undergraduate laboratories for using the scope and measuring the necessary voltages if needed.

- **For each coding problem**, you are required to create a flowchart that shows software operation and a block/connection diagram that shows how things are connected and what modules are in use. You can use online tools to do that. One example is lucidchart (https://lucid.app). Alternatively, you can use a software you are familiar with.
- **All codes should be written in assembly.**
- If a specific pin number is not stated, you are free to pick any appropriate pin to connect your components. Make sure to include this in your block/connection diagram.

Submission

You should submit the following items organized in a folder:

- **Lab report** - Written in English. PDF file. Should include
  - a cover page
  - for each problem
    - a flow chart
    - a block/connection diagram
    - pictures/photos if any/requested
    - code listing
    - at least one paragraph explanation/comment about that problem, code listing
  - final lab conclusion about what you've learned.
- **Source files** - should have proper comments and pin connections if there are any external components.
- **Elf files** - generated elf file with debug information.

Problems

Problem 1 [40 pts]. For this problem, you are asked to implement a diamond pattern given in Table 1 using external LEDs.

- Connect 8 LEDs and 1 push button to the board.
- The 8th LED should be a different color to indicate the status of the program. Let's call this **the status LED.** (Not necessary, but advised)

Requirements:

- The button should be used to play or pause the pattern. You can assume the program has two modes: play, and pause, and the button is used to change modes.
- When in pause mode, the status LED should be on, and when in play mode, the status LED should be off.
- There should be around `125 ms` delay between transitions. (i.e., `t3-t2 ~= 125 ms`)
- All the patterns are given in Table1. You should repeat these patterns indefinitely.

Questions:

- Using an oscilloscope, capture LED1, show the ON and OFF times. You can use a trigger mechanism to do this capture by setting it to one-time capture.
  - What happens if you decrease this delay time to `10 ms` or less? Capture LED1 again and explain it.
- Capture both the button signal and the status LED. Then press the button for pause.
  - How long did it take to go from button press to status LED lighting up?
  - How about vice versa (press the button for play)?
  - Explain your findings.

	LED1	LED2	LED3	LED4	LED5	LED6	LED7
t0							
t1				🟥			
t2			🟥	🟥	🟥		
t3		🟥	🟥	🟥	🟥	🟥	
t4	🟥	🟥	🟥	🟥	🟥	🟥	🟥
t5		🟥	🟥	🟥	🟥	🟥	
t6			🟥	🟥	🟥		
t7				🟥			

Table 1. Pattern on LEDs. Rows represent time steps, and columns represent each LED.

Problem 2 [60 pts]. In this problem you are asked to write a **decimal counter** using Seven Segment Displays.

- Connect 1 x Seven Segment Displays, 2 x buttons, and 1 status LED.

Requirements:

- SSD should display the last digit of your school ID.
- One button should cycle through each project member's ID on the SSD.
- Second button should start the automatic counting down from that number down to 0.
  - It should roughly go down at 1 second intervals. If the ID is 0, treat it as 10 and count down from 10.
- Upon reaching 0, it should stay there and wait there for any button press.
  - If the cycle button is pressed, it should display the next ID.
  - If the counter button is pressed, it should count down from the original ID again.
- Status LED should be ON when the countdown operation is in progress. OFF otherwise.
- Make sure the buttons do not have bouncing (or minimal). You can implement any hardware or software debouncing methods.

Warnings:

- Careful with the SSD (and LEDs in general), you need resistors to limit the current, otherwise you might burn some of the segments.
- Do not try to implement everything at once. After creating a detailed flowchart with the appropriate subroutines, divide the work into smaller tasks, assign between team members and implement gradually.
- It will be easier if you implement a subroutine that displays a given number on the SSD, then you can just change that number and the subroutine will display that.

Questions:

- Using an oscilloscope, capture the status LED when the countdown operation is in progress, and show the ON time. Does it match the seconds in the requirements?
- Do all the buttons need debouncing? Explain the method you implemented.