ELEC 2220 – Final Project – Parts 1 and 2 – Due April 22 and 28, respectively

The project is described below. Each of these will be submitted separately on Canvas, with Part 1 being the project description as a flow chart or commented/skeleton code (see below) and Part 2 being the implementation of the project as a *.s file.

The final project will entail writing a program with three modes:

Mode A – The lights will start all off, then turn on (and stay on) one at a time: orange, then red, then blue, and lastly green. They will then turn off (and stay off) one at a time in that same order. Once all are off, the cycle will repeat. Put a 0.25 second pause between each change. (For example, if all lights are off, the orange LED will come on and stay on, then 0.25 seconds later the red LED will also come on. 0.25 seconds after that, the blue LED will come on, etc.)

Mode B – The blue light will toggle/blink once per second, the red light will toggle every 0.75 seconds, the orange light will toggle every half second, and the green light will toggle every 0.25 seconds.

Mode C – The program will read in byte values from a null-terminated, read-only array called ModeCVals and display them on the lights. If the value is between 1 and 15, the bits will correspond such that PD 15 represents bit 3, PD14 for bit 2, PD13 for bit 1, and PD12 for bit 0. For example, if the value to be displayed is 6, that's 2_00000110 so PD15 would be off, PD14 and PD13 would be on, and PD12 would be off. If the value in the array is larger than decimal fifteen (0x0F), do not turn on any of the lights for its representation. Each value's LED representation should display for one second before changing directly to the next value. The null-termination character counts as a value (it's zero). After displaying the final (0x00) value, loop around to the beginning of the ModCVals array and start displaying the values from the beginning again.

The user will switch between these modes by pressing the PA0 input button. The change in mode can occur at any point during the mode (as opposed to finishing the cycle of blinks before starting the next. When the button is pressed while the program is on Mode C, it should "loop around" and begin Mode A again.

Hints: Use an interrupt for the input. Debounce your input. If a timer triggers every 0.25 seconds, then something that occurs every third timer trigger will occur once every 0.75 seconds. Moving bits 3 through 0 to bits 15 through 12 is a matter of shifting. Make DATA area values to be used as counters or to track what mode you're in.

Part 1 (3 points)

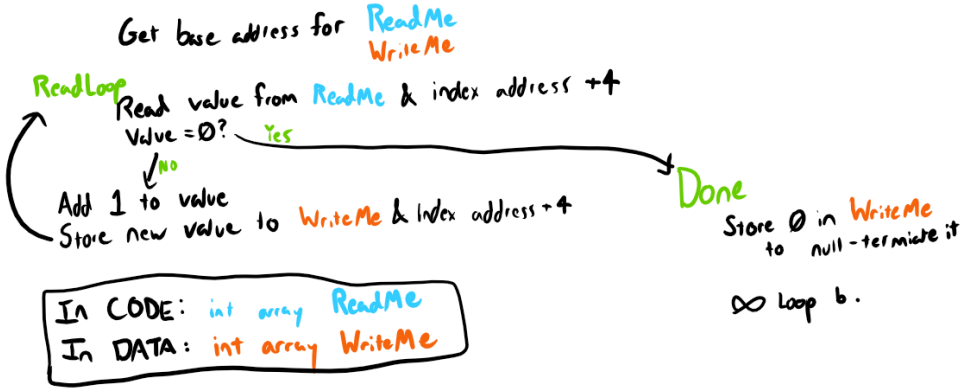
Each functionality of the program (including initializations) should be defined in detail such that the reader can follow the operation of the code. Labels for branches, arrays, and subroutines should be defined. You do not need to specify which register is used for which values; just provide a follow-able overview of the program plan. You also do not need to write op codes or specific operands; just describe your plan for the program (for example, "multiply by 3" is sufficient rather than writing out "mov rx, #3 then mul rz, ry, rx").

This portion of the project is assigned to make sure you think about the flow of the program and its functionalities prior to beginning to write code. If, once you begin actually coding the project, you realize you want to deviate from your Part 1 plan, that is okay.

For Part 1, turn in a **single PDF** with your project description of choice. Two examples are provided below. Full credit will be given for submissions which address all components of the project; when in doubt, more detail is better but aim for **one page in length**.

See below for an example of 1) a flow chart and 2) skeleton code for a program designed to read in values from a null-terminated array, add one to them, then store them to another null-terminated array. Hopefully this will help illustrate the approximate level of detail expected.

1)



2)

```
        ;get ReadMe base address
        ;get WriteMe base address
ReadLoop      ;stay in loop while reading values
        ;get value from ReadMe and index address by 4 (because int)
        ;compare that value to 0 and if it's 0, exit to Done label
        ;if you're still here, add one to the value
        ;store the value to WriteMe and index address by 4 (b/c int)
        ;do ReadLoop again
Done    ;it you're here, reading is done
        ;store 0x00 (null) to WriteMe
        b .     ;infinite loop
;In CODE section: ReadMe int array
;In DATA section: WriteMe int array
```

Part 2 (7 points)

Submit your code as a single *.s file on Canvas along with a PDF including a screenshot of uVision in debug mode. These screenshots should show the full window, from file path in the top left to memory window in the bottom right. Make sure the center of the screen is open to your code tab (as opposed to just showing startup code or the .ini file tabs).

Uncommented code and/or plagiarized work will not be graded and a zero will be given for the full project.