ECEN 260 Lab Report

Lab #5: Switch-Controlled LEDs with Assembly

Name: Brodric Young

Lab Partner: Karl Richards

Date: 2/5/2025

Instructor: Brother Allred

Semester: Winter 2025

Lab Overview

Replace this text with a paragraph that provides the reader with a clear and brief introduction to the lab. It should also effectively communicate the objectives/goals of the lab and set the appropriate context for the report

This lab focused on using GPIO registers to read inputs from buttons and control LEDs using assembly language on a microcontroller board. The objectives of the lab were to configure GPIO ports for input and output, wire a circuit on a breadboard, and write assembly code to make button controlled LEDs. It required configuring another button and LED and adding some more code to turn on which LED for which button combination was pressed.

Code

```
// ***** Configure button 3 (PA1) as input with pull up resistor *****
// MODER
MOV R1, #0xC // 0000 0000 0000 0000 0000 0000 0000 1100
MVN R1, R1   // 1111 1111 1111 1111 1111 1111 1111 0011
LDR R0, [R4] // load in Port A MODER
AND R0, R1 // bit mask to reset bits 3 and 4
STR R0, [R4] // store PA MODER

// PUPDR
LDR R0, [R4, #0xC] // load in PA PUPDR
MOV R1, #0x4 // 0000 0000 0000 0000 0000 0000 0000 0100
ORR R0, R1 // set bit 2
STR R0, [R4, #0xC] // store PA PUPDR

// ************ Configure LED4 (PA0) as output **************
// Create bitmask to reset bit 1 of PA MODER (bit 0 initializes to 1)
MOV R1, #0x2 // 0000 0000 0000 0010
MVN R1, R1   // 1111 1111 1111 1101
// read PA MODER value, reset bit 1 and write back to PA MODER
LDR R0, [R4] // load PA MODER
AND R0, R1 // bitmask to reset bit 1
STR R0, [R4] // store PA MODER
```

```
loop:
      // *********** Read status of button 1 (PC13) ***********
      MOV R1, #0x2000 // create bitmask to read bit 13 pf PC IDR
      // read PC IDR value, reset all bits except bit 13
      LDRH R0, [R5, #0x10] // load PC IDR
      AND R0, R1 // bitmask to reset all bits excpet bit 13
      CMP R0, #0 // Zero = button pushed. NonZero = button not pushed
      BNE elseLED2 //if not pushed, go to "elseLED2", if not continue to "if"

if:
      // ************** Read status of button 3 **************
      MOV R1, #0x2 // bitmask to reset all bits but bit 1
      LDRH R0, [R4, #0x10] // load PA IDR
      AND R0, R1 // reset all bits but bit 1

      CMP R0, #0 // 0 = button pushed, nonzero = not pushed
      BNE ifLED2andelseLED4 // if not pushed, branch to turn on LED4 and off
LED2
      BL turnonLED2 // both button 1 and 3 pushed
      BL turnoffLED4
      B loop

// this function is called if button 1 is not pressed.
// It reads the status on button 3 and if it is pressed, turns on LED4
// and LED2 off. otherwise branches to turn off both
elseLED2:
      // ************** Read status of button 3 **************
      MOV R1, #0x2 // bitmask to reset all bits but bit 1
      LDRH R0, [R4, #0x10] // load PA IDR
      AND R0, R1 // reset all bits but bit 1

      CMP R0, #0 // 0 = button pushed, nonzero = not pushed
      BNE elseLED2and4
```

```
    BL turnonLED4
    BL turnoffLED2
    B loop

// this function turns of both LEDs called when no button is pressed
elseLED2and4:
    BL turnoffLED2
    BL turnoffLED4
    B loop

// this function turns on LED4 and turns off LED2 called when exactly one
// button is pressed
ifLED2andelseLED4:
    BL turnoffLED2
    BL turnonLED4
    B loop
```

Conclusion

In this lab, I configured GPIO registers to read button inputs and control LEDs based on button presses using assembly language. I wired the circuit, adding another button and LED then enabled an internal pull-up resistor in assembly, and made conditional branching in assembly to create the desired LED behavior for both buttons and LEDs. I learned more about how to use functions in assembly, using bit masks, and the GPIO registers.

In class we had learned about branching and functions in assembly which was crucial to implement the logic we needed for the desired LED behavior. We also learned about the pull-up/down resistors and active low/high inputs which we used in this lab. In another class I saw how bit masks could be used to preform different functions which connected to this lab where we used them to manipulate specific parts of a GPIO register. After doing this coding in assembly language, it will be much easier to program in C later on and it will make much more sense why we do certain things in C.