Introduction:

During this lab, we will be working with Code Composer Studio and running various assembly programs. We will be analyzing their behavior and then we will be creating our own program to control the red led on port 1.0.

Procedure:

1. **Analyze Arm Assembly Code**
   - **b) Example Projects**
     - **i.** GPIO_MSP432asm

       This program initializes port 4.0-4.3 as outputs to the LEDs. During the main loop, it cycles between the 4 LEDs.

     - **ii.** InputOutput_MSP432asm

       This program initializes port 2.0-2.2 for the RGB LED and P1.1 & P1.4 for the switches. The main loop reads the switches and then depending on which are pressed, it changes the color of the RGB LED.

     - **iii.** SquareWaves_MSP432asm

       This program initializes port 2.1-2.2 for the green and blue LED. During the loop, it toggles the color of the LED between blue and green.

     - **iv.** SSR_MSP432asm

       This program initializes port 2.2 for the motor output, and port 1.1 & 1.4 for the switches. The main loop makes the motor turn when you press button 1, and turns the motor off.

     - **v.** Switch_MSP432asm

       This program initializes port 2.0-2.2 for the RGB LEDs, P1.0 for red LED, and P1.1 & 1.4 for the switches. The main loop detects the switches and sets the RGB LED to a certain color depending on which switches are pressed. The last part of the loop checks the status of P1.5 and moves it to the red LED.

2. **Write ARM Assembly Program to Control an LED**
   - **a)** Write program to control the red LED from the two switches as inputs.
   - **b)** The sample projects I chose to use is, InputOutput for the design and better style of coding for the inputs, and Switch for the better design of modularizing the code and using a more efficient way to mask bits for outputs instead of overwriting all bits.
   - **c)** See Flow Chart
   - **d)** See Pseudocode

e) Modify example project to meet specs for this project

Conclusion:

I considered the most important part of this lab with familiarizing ourselves with the MSP432 board, and the syntax of ARM assembly code. We were able to look through the demo projects in order to analyze some code and gain some clarity on how to interface with the board. Once I had a good understanding of how the example code works, I combined the two projects and used the strengths of each in order to implement a design that would meet the criteria for the design given.

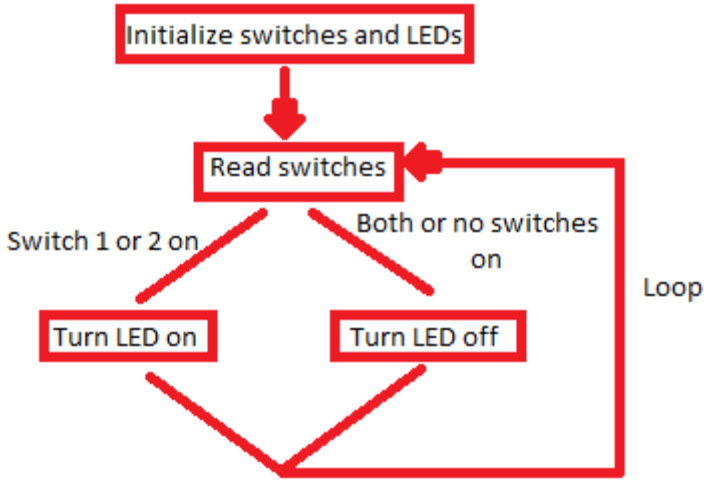
Overall, I feel this was a good first lab in order to familiarize ourselves with this assembly language and this board.

References:

For this project I used the lecture slides, the instruction set manual for this board, and talked with a few other students about design specifications.

Appendix:

Flowchart:



Pseudocode:

Initialize switches for input, and LED1 for output
Turn LED1 on for default

Start Loop
    Read input from switches
    Test which switch is pressed
        Switch 1
        Switch 2
        Both
        None
        Catch Error if unexpected output

Set output
Switch 1 or 2
LED1 on
Both or None
LED1 off
Restart Loop

Code:

```
; InputOutput.s
; Runs on MSP432
; Test the GPIO initialization functions by setting the LED
; color according to the status of the switches.
; Daniel Valvano
; June 20, 2015

;  This example accompanies the book
;  "Embedded Systems: Introduction to the MSP432 Microcontroller",
;  ISBN: 978-1512185676, Jonathan Valvano, copyright (c) 2015
;  Section 4.2    Program 4.1
;
;Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
;   You may use, edit, run or distribute this file
;   as long as the above copyright notice remains
;THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
;OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
;MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
;VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
;OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
;For more information about my classes, my research, and my books, see
;http://users.ece.utexas.edu/~valvano/

; built-in LED1 connected to P1.0
; negative logic built-in Button 1 connected to P1.1
; negative logic built-in Button 2 connected to P1.4
; built-in red LED connected to P2.0
; built-in green LED connected to P2.1
; built-in blue LED connected to P2.2
     .thumb

     .text
     .align  2
P1IN    .field 0x40004C00,32  ; Port 1 Input
P1OUT   .field 0x40004C02,32  ; Port 1 Output
P1DIR   .field 0x40004C04,32  ; Port 1 Direction
```

```
P1REN   .field 0x40004C06,32  ; Port 1 Resistor Enable
P1DS    .field 0x40004C08,32  ; Port 1 Drive Strength
P1SEL0  .field 0x40004C0A,32  ; Port 1 Select 0
P1SEL1  .field 0x40004C0C,32  ; Port 1 Select 1

SW1     .equ 0x02             ; on the left side of the LaunchPad board
SW2     .equ 0x10             ; on the right side of the LaunchPad board

     .global main
     .thumbfunc main

main: .asmfunc
   BL  Port1_Init             ; initialize P1.1 and P1.4 and make them inputs (P1.1 and P1.4 built-in
buttons)
loop
   BL  Port1_Input            ; read both of the switches on Port 1
   CMP R0, #0x10              ; R0 == 0x10?
   BEQ sw1pressed             ; if so, switch 1 pressed
   CMP R0, #0x02              ; R0 == 0x02?
   BEQ sw2pressed             ; if so, switch 2 pressed
   CMP R0, #0x00              ; R0 == 0x00?
   BEQ bothpressed            ; if so, both switches pressed
   CMP R0, #0x12              ; R0 == 0x12?
   BEQ nopressed              ; if so, neither switch pressed
                         ; if none of the above, unexpected return value
   AND R0, #0x01                        ; R0 = (Off) (no LEDs on)
   BL  Port1_Output_Off       ; turn all of the LEDs on
   B   loop
sw1pressed
   BL  Port1_Output_On        ; turn the red LED on
   B   loop
sw2pressed
         BL  Port1_Output_On          ; turn the red LED on
   B   loop
bothpressed
         BL  Port1_Output_Off         ; turn the red LED off
   B   loop
nopressed
   BL  Port1_Output_Off       ; turn the red LED off
   B   loop
   .endasmfunc
;------------Port1_Init------------
; Initialize GPIO Port 1 for negative logic switches on P1.1 and
; P1.4 as the LaunchPad is wired.  Weak internal pull-up
```

```
; resistors are enabled.
; Input: none
; Output: none
; Modifies: R0, R1
Port1_Init: .asmfunc
   ; configure P1.4 and P1.1 as GPIO
   LDR  R1, P1SEL0
   LDRB R0, [R1]
   BIC  R0, R0, #0x12          ; configure P1.4 and P1.1 as GPIO
   STRB R0, [R1]
   LDR  R1, P1SEL1
   LDRB R0, [R1]
   BIC  R0, R0, #0x12          ; configure P1.4 and P1.1 as GPIO
   STRB R0, [R1]
   ; make P1.4 and P1.1 in
   LDR  R1, P1DIR
   LDRB R0, [R1]
   BIC  R0, R0, #0x12          ; input direction
   STRB R0, [R1]
   ; enable pull resistors on P1.4 and P1.1
   LDR  R1, P1REN
   LDRB R0, [R1]
   ORR  R0, R0, #0x12           ; enable pull resistors
   STRB R0, [R1]
   ; P1.4 and P1.1 are pull-up
   LDR  R1, P1OUT
   LDRB R0, [R1]
   ORR  R0, R0, #0x12           ; pull-up resistors
   STRB R0, [R1]
   BX  LR
   .endasmfunc
;------------Port1_Input------------
; Read and return the status of the switches.
; Input: none
; Output: R0  0x10 if only Switch 1 is pressed
;       R0  0x02 if only Switch 2 is pressed
;       R0  0x00 if both switches are pressed
;       R0  0x12 if no switches are pressed
; Modifies: R1
Port1_Input: .asmfunc
   LDR  R1, P1IN
   LDRB R0, [R1]              ; read all 8 bits of Port 1
   AND  R0, R0, #0x12           ; select the input pins P1.1 and P1.4
   BX   LR
```

```
   .endasmfunc
;------------Port1_Output_On------------
; Read inputs and return the output to turn on red led.
; Input: P1DIR
; Output: R3  0x00 if both switches are pressed
;       R3  0x12 if no switches are pressed
; Modifies: R1
Port1_Output_On: .asmfunc
          LDR  R1, P1DIR
   LDRB R0, [R1]               ; read all 8 bits of Port 1
   ORR  R0, R0, #0x01            ; select the input pins P1.1 and P1.4
   STRB R0, [R1]
   BX   LR
   .endasmfunc
;------------Port1_Output_Off------------
; Read inputs and return the output to turn on red led.
; Input: P1DIR
; Output: R3  0x00 if both switches are pressed
;       R3  0x12 if no switches are pressed
; Modifies: R1
Port1_Output_Off: .asmfunc
          LDR  R1, P1DIR
   LDRB R0, [R1]               ; read all 8 bits of Port 1
   BIC  R0, R0, #0x01            ; select the input pins P1.1 and P1.4
   STRB R0, [R1]
   BX   LR
   .endasmfunc
   .end
```