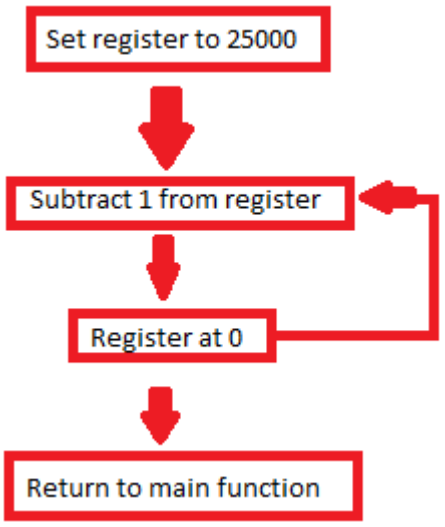
Introduction:

During this lab, we will be working with Code Composer Studio and will be looking at interfacing with the MSP432 board to control the red LED. We will be running a loop to check for a button press and if pressed toggle the red LED using a delayed loop otherwise leave the LED on.

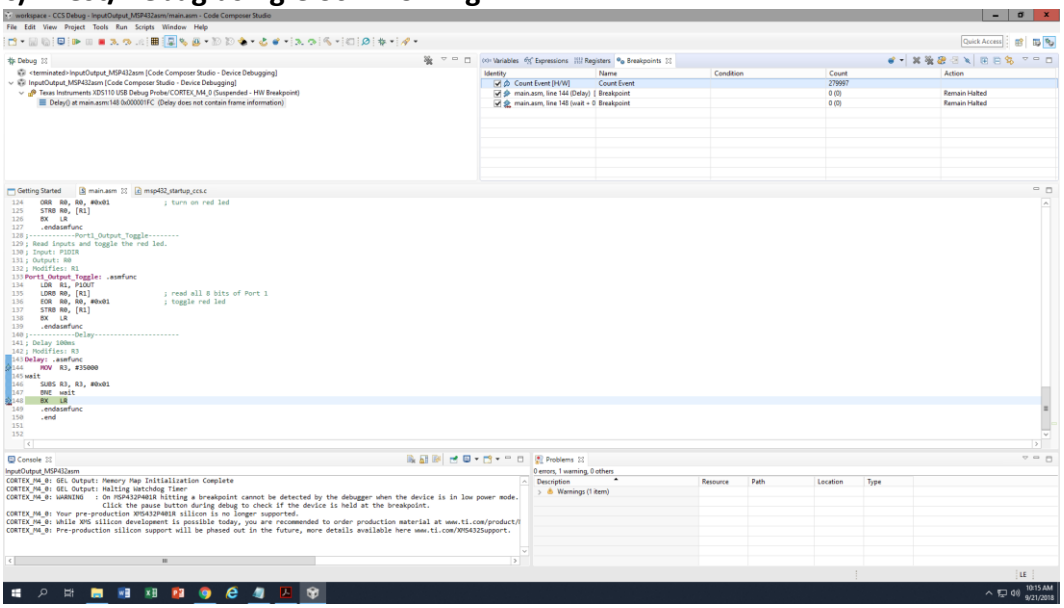
Procedure:

1. **Create a Time Delay Subroutine**
   a) **100 ms Time Delay**
      Bus Clock 3 MHz, 3000 bus cycles in 1 ms
      => 300,000 cycles in 100 ms

   b) **Flowchart**

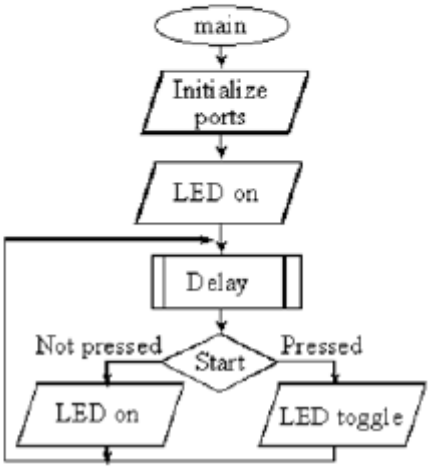


   c) **Test/Debug using Clock Profiling**



2. **Write ARM Assembly Program to Control the red LED**
   a) **Pseudocode & Flowchart**

main *Initialize Port 1:*
Set the Port 1 direction register so P1.4 is an input and P1.0 is an output
Set bit 4 of P1REN to enable the internal register for P1.4
Specify this resistor as a pull up by setting bit 4 of P1OUT
Set P1.0 so the LED is ON
loop *Delay about 100 ms*
Read the switch and test if the switch is pressed
If P1.4=0 (the switch is pressed), toggle P1.0 (flip bit from 0 to 1, or from 1 to 0)
If P1.4=1 (the switch is not pressed), set P1.0, so LED is ON
Go to loop



   b) Load software onto board and test it

Conclusion:

The result of this lab was determining how to create delays, how to use delays to make the LED blink, and to compartmentalize the code used. We made the code into snippets so that we may be able to reuse it when needed instead of having to re-write the code every time. This is beneficial as it reinforces the standard of Object Oriented Programming which is used in high level languages as well. Figuring out the delay was the hardest part of this lab for me. The math for the clock cycles seemed to not add up until I used the CCS Clock Profiling, this enabled me to visualize how the delay was being propagated and to figure out where my error was in the code.

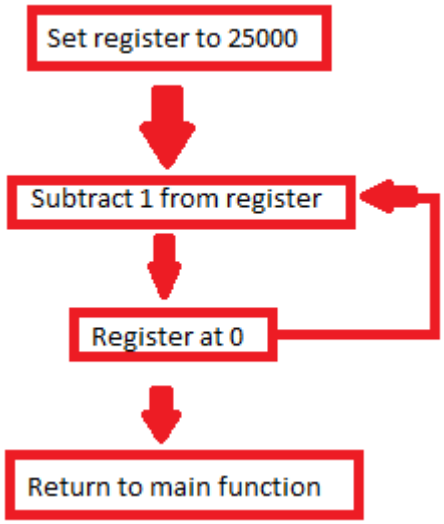
Overall, this lab helped a lot with being able to split up larger segments of code to understand them better and explore the concept behind delays.

References:

For this project I used the lecture slides, the CCS tutorial, and the instruction set manual for this board.

Appendix:

Flowchart:



Pseudocode:

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

Start Loop
    Read input from switches
    Test is switch is pressed
        Switch 2
        None
        Catch Error if unexpected output
    Set output
        Switch 2
            LED1 toggle
        None
            LED1 on
    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)
   ;BL  Port1_Output_On
loop
   BL  Port1_Input            ; read switch P1.4
   CMP R0, #0x00              ; R0 == 0x10?
```

```
   BEQ sw1pressed              ; if so, switch 1 pressed
        CMP R0, #0x10
        BEQ nopressed
   B   loop
sw1pressed
        BL  Delay                                          ; delay 100ms
   BL  Port1_Output_Toggle       ; Toggle red led
   B   loop
nopressed
        BL  Port1_Output_On
        B   loop
   .endasmfunc
;------------Port1_Init------------
; Initialize GPIO Port 1 for negative logic switches on
; P1.4 as the LaunchPad is wired.  Weak internal pull-up
; resistor are enabled.
; Input: none
; Output: none
; Modifies: R0, R1
Port1_Init: .asmfunc
   ; configure P1.4 as GPIO
   LDR  R1, P1SEL0
   LDRB R0, [R1]
   BIC  R0, R0, #0x10          ; configure P1.4 as GPIO
   STRB R0, [R1]
   LDR  R1, P1SEL1
   LDRB R0, [R1]
   BIC  R0, R0, #0x10          ; configure P1.4 as GPIO
   STRB R0, [R1]
   ; make P1.4 in
   LDR  R1, P1DIR
   LDRB R0, [R1]
   BIC  R0, R0, #0x10          ; input direction switch
   STRB R0, [R1]
   ; make P1.0 led1
   LDR  R1, P1DIR
   LDRB R0, [R1]
   ORR  R0, R0, #0x01           ; output direction led1
   STRB R0, [R1]
   ; enable pull resistors on P1.4
   LDR  R1, P1REN
   LDRB R0, [R1]
   ORR  R0, R0, #0x10           ; enable pull resistors
   STRB R0, [R1]
```

```
   ; P1.4 are pull-up
   LDR  R1, P1OUT
   LDRB R0, [R1]
   ORR  R0, R0, #0x10          ; pull-up resistors
   STRB R0, [R1]
   BX  LR
   .endasmfunc
;------------Port1_Input---------------
; Read and return the status of the switches.
; Input: P1IN
; Output: R0  0x02 if Switch 2 is pressed
; Modifies: R1
Port1_Input: .asmfunc
   LDR  R1, P1IN
   LDRB R0, [R1]               ; read all 8 bits of Port 1
   AND  R0, R0, #0x10          ; select the input pins P1.4
   BX   LR
   .endasmfunc
;------------Port1_Output_On------------
; Read input and turn on the red led.
; Input: P1DIR
; Output: R0
; Modifies: R1
Port1_Output_On: .asmfunc
         LDR  R1, P1OUT
   LDRB R0, [R1]               ; read all 8 bits of Port 1
   ORR  R0, R0, #0x01          ; turn on red led
   STRB R0, [R1]
   BX   LR
   .endasmfunc
;------------Port1_Output_Toggle--------
; Read inputs and toggle the red led.
; Input: P1DIR
; Output: R0
; Modifies: R1
Port1_Output_Toggle: .asmfunc
         LDR  R1, P1OUT
   LDRB R0, [R1]               ; read all 8 bits of Port 1
   EOR  R0, R0, #0x01          ; toggle red led
   STRB R0, [R1]
   BX   LR
   .endasmfunc
;------------Delay----------------------
; Delay 100ms
```

```
; Modifies: R3
Delay: .asmfunc
        MOV  R3, #25000
wait
        SUBS R3, R3, #0x01
        BNE  wait
        BX   LR
   .endasmfunc
   .end
```