Lab 1 Report

In this lab the goal was to run assembly program on the microcontroller and de-bug another program that copies data in a couple ways. Find the correct ways to write down code and run programs in right ways.

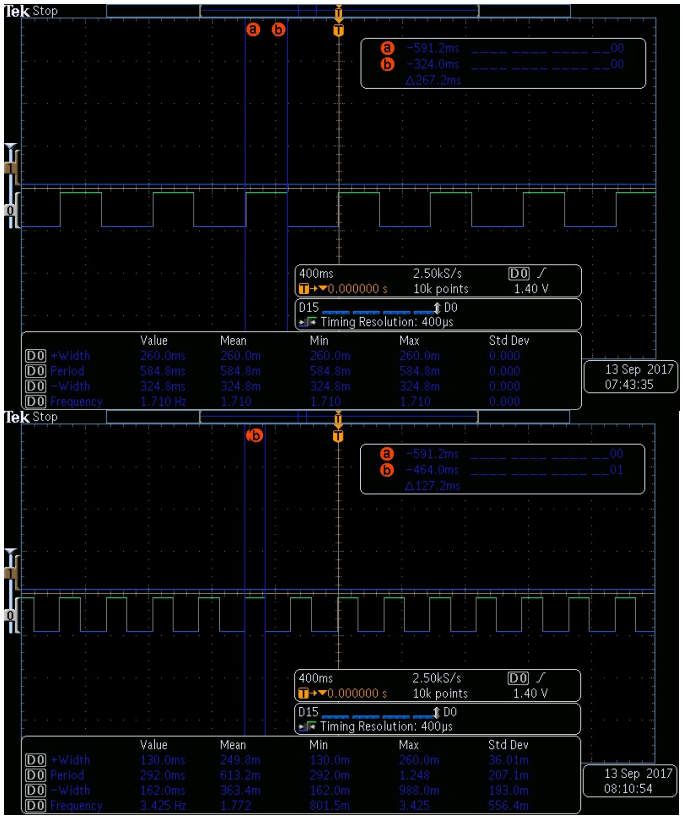
Section 1: Setup Blinky

In this lab section, I downloaded the files from course wiki. Then I followed the steps from lab instruction to set up the program to use the files from wiki. I made sure the program run without error. From this section, we can learn how to set up program on computer and how to fix errors when we run it.

Section 2: Running Blinky on Your Board

In this lab section, first we connect the physical board to computer. Then we used the downloaded files from last section to debug and run the program and made sure it had no error. After that, we connected the physical board to logic analyzer to measured the frequency of the LED blinking, and modify ***blinky.asm***(downloaded from wiki).

To modify ***blinky.asm*** we can change `MOV32 R5, #0xFFFFF` to `MOV32 R5, #0x7FFFF` . Following are the screen captures for two frequencies.



From this section, we can learn how to connect physical board to logic analyzer and how to use the logic analyzer. The hardest part for this section is to make sure you connect the right pin.

Section 3: Commenting Code

In this lab section, we need describe the function by filling in in the blanks in the comments. Following is the screen capture for comments.

```
loop
    MOV32 R1, #0x02 ;load value for turning on LED color red.
    STR R1, [R0,#0x38]  ;write the above value to GPIOF ODR register.

    MOV R4, #0          ;initial value for iteration loop
    MOV32 R5, #0x7FFFF  ;number of iterations for delay loops
delay1
    ADD R4, #1          ; Add 1 to the value in register 4.
    CMP R4, R5          ;check number of iterations
    BLE delay1          ;continue if iterated less than 0xFFFFF + 1 times, otherwise repeat delay loop

    MOV32 R1, #0x04 ;load value for turning on LED color blue.
    STR R1, [R0, #0x38] ;write the above value to GPIOF ODR

    MOV R4, #0      ;initial value for iteration loop
    ;           ; **** the tm4c123gh6pm has 16 MHz clock.
    ;;           how long should the loop take with that clock?

delay2
    ADD R4, #1          ;Add 1 to the value stored in register 4
    CMP R4, R5          ;check number of iterations
    BLE delay2          ;continue if iterated less than 0xFFFFF + 1 times, otherwise repeat delay loop

    MOV32 R1, #0x08 ;load value for turning on LED green.
    STR R1, [R0, #0x38] ;write the above value to GPIOF ODR
```

From this section, we can better understand what is going on for the code.

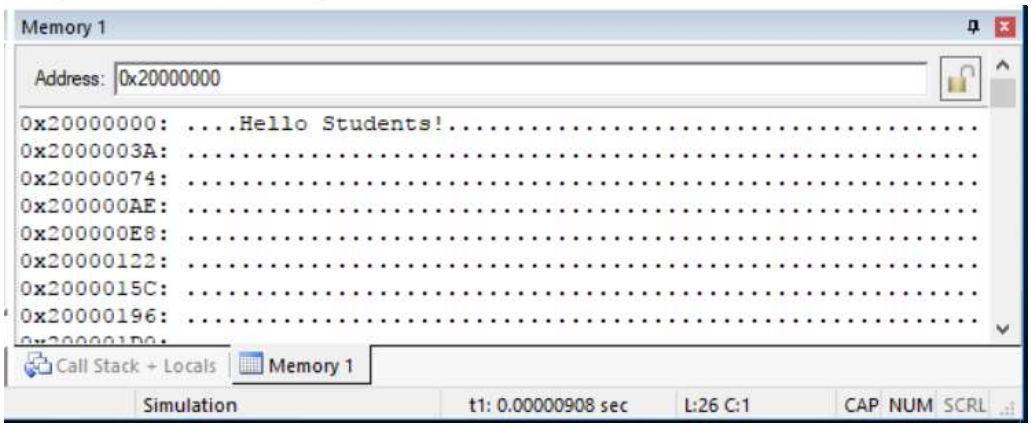
Section 4: Running Blinky Using the Simulator

In this lab section, ***blinky.asm*** was run using a simulator. From this we can learn how to simulate the code, and how to step through the program and examine registers/memory. This section helped us prepare for section 5.

Section 5: Code Debugging

In this lab section, we fixed code from "***Hello Students!"*** file just like ***Blinky*** in the last few sections. But in this time we can use memory reserved for the stack to test what is the error when we print "Hello Students!" For this section, I think the hardest part is how to add correct code for the program. Following screen captures is the code we changed and the final print we got.

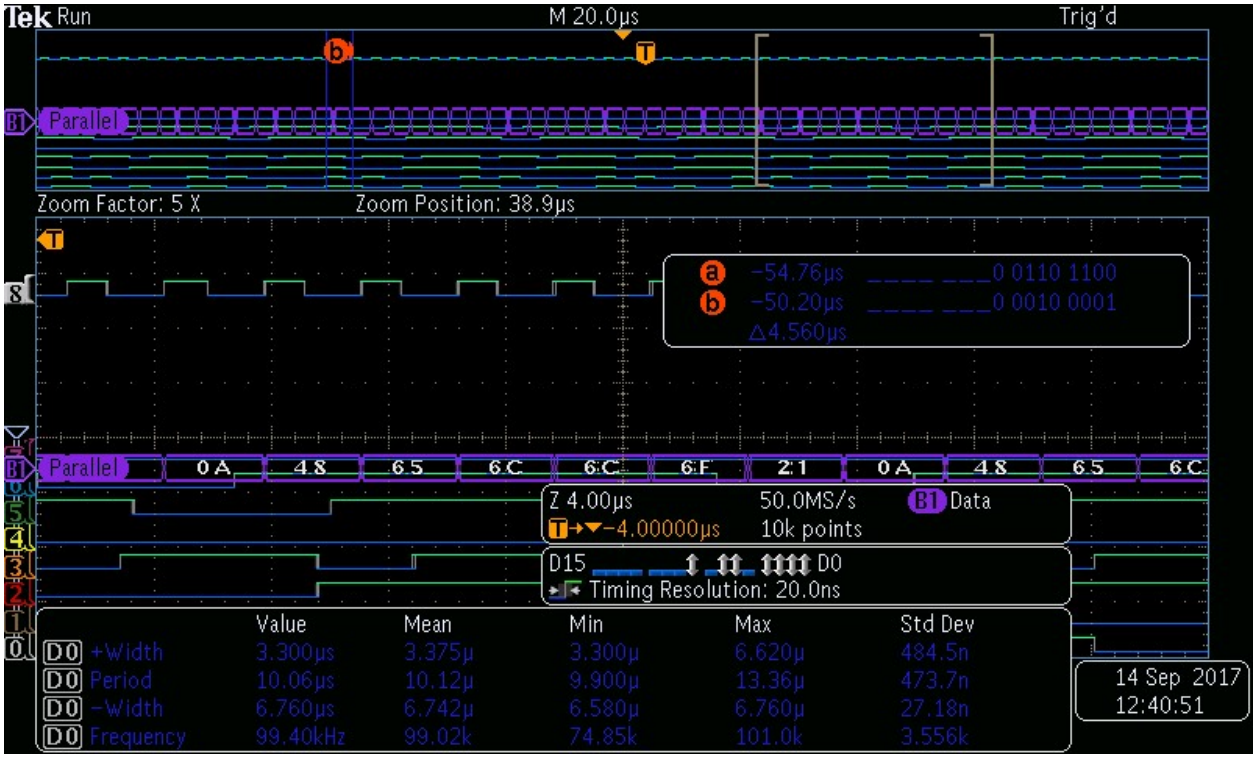
```
message         DCB     "Hello Students!",0         ; message stored readonly

                ALIGN ;pg149
Start           LDR     R0, =message                ; load address of the message

                MOV     R1, #0x20000000                     ; load memory location to store

                MOV     R3, #4                      ; used as a counter

load            LDR     R2,[R0]                     ; load a word of the message
                ADD     R0, R0, #4                  ; adds 4 to the pointer


                SUB     R3, R3, #1                  ; decrements counter
                STR     R2,[R1,#4]                  ; store the word to memory
                ADD     R1, R1, #4                  ; INCREMENT R1 BY 4

                CMP     R3, #0                      ; Check for null
                BNE     load                        ; repeat if not null terminated

loop            B       loop
```



From this section, we learned how to add correct code and how to find the bug.

Section 6: Logic Analyzer

In this lab section, we downloaded ***Logic_analyzer.c*** from wiki and replaced ***main.c*** to this file. Then we modified bus1 to match this configuration. Lastly, we found the data in bus1. I think the hardest part for this section is to know how to find bus1 on the logic analyzer and how to show it. Make sure you put in the correct pin in the physical board for the logic analyzer. The following screen capture is the data for bus1.



From this section, I learned how to find bus date.

In this lab I learned how to fix code using the simulator and how to find bus in the logic analyzer.