Lab 1

Tuesday 14:15

Jack Landers

Henry Fang

Introduction

Try using the C in Keil to write the program that runs on STM32L476VGT6 Discovery board. The program should blink green or red LEDs on the board. Also using the structures and pointer to access the registers in the peripheral.

Procedure

We downloaded the zip pack Minimal C template to get a barebone project to work with.

Part 1

We write C programs that combine two strings to a new string. Then we write a C program to calculate the 25 elements of the Fibonacci sequence.

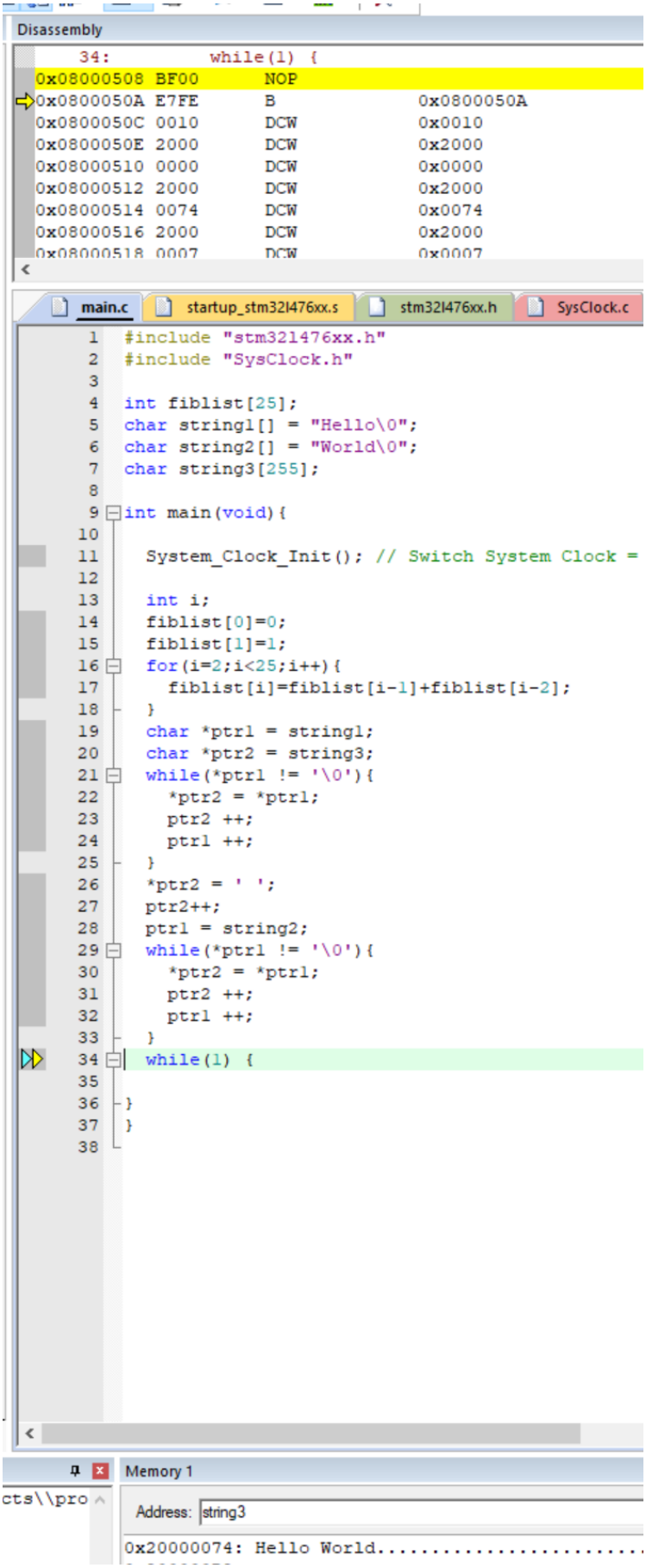
Part 2

We look up the stm32l476xx.h header file to find/understand the structure of the peripheral system. We find the typedef part of the header file, searching for the GPIO ports and pins we want to initialize, then we use those structures and their substructures to navigate to find the pointers to the peripheral registers. Then other things are easy C programming. We make subroutines that turn on and off red or green leds.

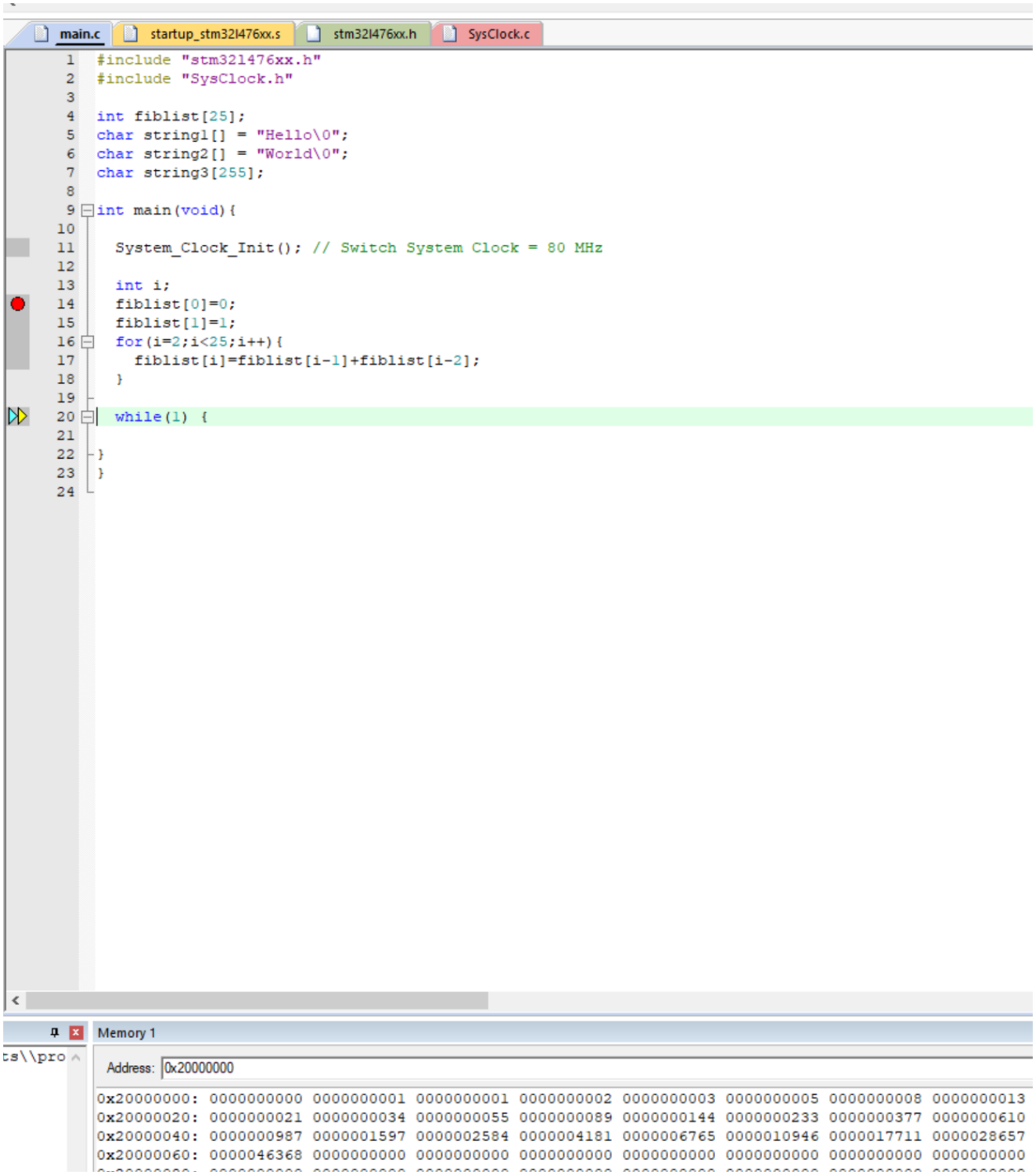
Results

Place the results of your experiments and tasks here. Tables, charts, screenshots, etc.. Answer any questions from the lab document here.

For the delaymicro(), we know the clock rate is 80MHz, so it can run 80M instructions per second. So one instruction takes 1.25*10^-8 seconds to run, if we want 1 microsecond, then we need 80 instructions. If the increase index in “for loop”takes one instruction, then we need 80 loops in the “for loop”.


Disassembly
34: while(1) {
0x08000508 BF00 NOP
0x0800050A E7FE B 0x0800050A
0x0800050C 0010 DCW 0x0010
0x0800050E 2000 DCW 0x2000
0x08000510 0000 DCW 0x0000
0x08000512 2000 DCW 0x2000
0x08000514 0074 DCW 0x0074
0x08000516 2000 DCW 0x2000
0x08000518 0007 DCW 0x0007
main.c
startup_stm32l476xx.s
stm32l476xx.h
SysClock.c
1 #include "stm32l476xx.h"
2 #include "SysClock.h"
3
4 int fiblist[25];
5 char string1[] = "Hello\0";
6 char string2[] = "World\0";
7 char string3[255];
8
9 int main(void){
10
11 System_Clock_Init(); // Switch System Clock =
12
13 int i;
14 fiblist[0]=0;
15 fiblist[1]=1;
16 for(i=2;i<25;i++){
17 fiblist[i]=fiblist[i-1]+fiblist[i-2];
18 }
19 char *ptr1 = string1;
20 char *ptr2 = string3;
21 while(*ptr1 != '\0'){
22 *ptr2 = *ptr1;
23 ptr2 ++;
24 ptr1 ++;
25 }
26 *ptr2 = ' ';
27 ptr2++;
28 ptr1 = string2;
29 while(*ptr1 != '\0'){
30 *ptr2 = *ptr1;
31 ptr2 ++;
32 ptr1 ++;
33 }
34 while(1) {
35
36 }
37 }
38
Memory 1
Address: string3
0x20000074: Hello World.....

Part 1



Part 2

Conclusion

Henry: In this lab, we learned how to write C code for embedded systems. Also learned how to use the structures and pointers to quickly find the peripheral register we need.

Appendix

Place code and other data here

```
#include "stm32l476xx.h"
#include "SysClock.h"

int fiblist[25];
char string1[] = "Hello\0";
char string2[] = "World\0";
char string3[255];

int main(void){

	System_Clock_Init(); // Switch System Clock = 80 MHz

	int i;
	fiblist[0]=0;
	fiblist[1]=1;
	for(i=2;i<25;i++){
		fiblist[i]=fiblist[i-1]+fiblist[i-2];
	}
	char *ptr1 = string1;
	char *ptr2 = string3;
	while(*ptr1 != '\0'){
		*ptr2 = *ptr1;
		ptr2 ++;
		ptr1 ++;
	}
	*ptr2 = ' ';
  ptr2++;
	ptr1 = string2;
	while(*ptr1 != '\0'){
		*ptr2 = *ptr1;
		ptr2 ++;
		ptr1 ++;
	}
	while(1) {

}
}
```

```
#include "stm32l476xx.h"
#include "SysClock.h"
#include "stdint.h"
void red_on(){
        GPIOB->ODR |= GPIO_ODR_ODR_2;
        return;
}

void red_off(){
        GPIOB->ODR &= ~GPIO_ODR_ODR_2;
        return;
}

void red_toggle(){
        GPIOB->ODR ^= GPIO_ODR_ODR_2;
        return;
}

void green_on(){
        GPIOE->ODR |= GPIO_ODR_ODR_8;
        return;
}

void green_off(){
        GPIOE->ODR &= ~GPIO_ODR_ODR_8;
        return;
}

void green_toggle(){
        GPIOE->ODR ^= GPIO_ODR_ODR_8;
        return;
}

void delaymicro(unsigned int x){
                x=1;
                while(x!=6500000){
                        x++;
                }
                return;
}
```

```
int main(void){

        System_Clock_Init(); // Switch System Clock = 80 MHz

        RCC->AHB2ENR |= 0x12;           //Port B and E Clock enabled

        GPIOB->MODER ^= GPIO_MODER_MODER2_1;            //Port B.2 set to push pull

        GPIOE->MODER ^= GPIO_MODER_MODER8_1;            //Port E.8 set to push pull

        //int i;
        //for(i=0;i<100;i++){
        //      red_toggle();
        //      delaymicro(500000);
        //}

        red_on();
        green_on();
        red_off();
        green_toggle();
        red_toggle();
        green_toggle();
        red_toggle();
        green_off();



        while(1) {
        red_toggle();
        delaymicro(500000);
        red_toggle();
        green_toggle();
        delaymicro(500000);
        green_toggle();
}
}
```