GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

ELM335

MICROPROCESSORS LAB

LAB 1 Report

Prepared By
1) 1901022038 – Selen Erdoğan
2) 200102002043 – Senanur Ağaç
3) 1901022050 - Merve Tutar

1- Introduction

The objective of this lab is to get familiarized with the development board, understand all the ICs and their connections with the microprocessor. Practice assembly language, and write basic programs. Work with the debugging tools for analyzing program flow.

2- Problem 1

In the Nucleo G031K8 board, identify all the ICs and explain their usage. Also explain all the connected peripherals and their pin connections with the microcontroller.

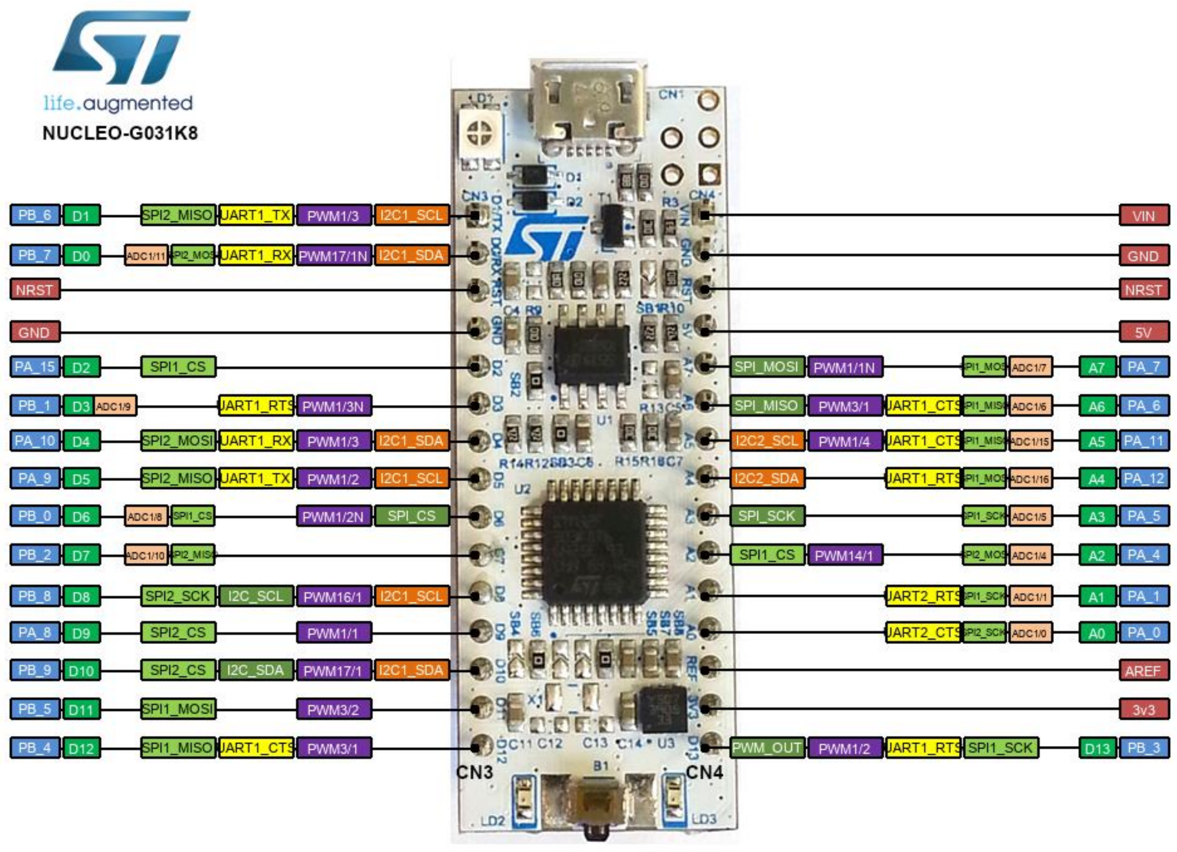

life.augmented
NUCLEO-G031K8
PB_6 D1 SPI2_MISO UART1_TX PWM1/3 I2C1_SCL
PB_7 D0 ADC1/11 SPI2_MOSI UART1_RX PWM17/1N I2C1_SDA
NRST
GND
PA_15 D2 SPI1_CS
PB_1 D3 ADC1/9 UART1_RTS PWM1/3N
PA_10 D4 SPI2_MOSI UART1_RX PWM1/3 I2C1_SDA
PA_9 D5 SPI2_MISO UART1_TX PWM1/2 I2C1_SCL
PB_0 D6 ADC1/8 SPI1_CS PWM1/2N SPI_CS
PB_2 D7 ADC1/10 SPI2_MISO
PB_8 D8 SPI2_SCK I2C_SCL PWM16/1 I2C1_SCL
PA_8 D9 SPI2_CS PWM1/1
PB_9 D10 SPI2_CS I2C_SDA PWM17/1 I2C1_SDA
PB_5 D11 SPI1_MOSI PWM3/2
PB_4 D12 SPI1_MISO UART1_CTS PWM3/1
CN3
CN4
VIN
GND
NRST
5V
SPI_MOSI PWM1/1N SPI1_MOSI ADC1/7 A7 PA_7
SPI_MISO PWM3/1 UART1_CTS SPI1_MISO ADC1/6 A6 PA_6
I2C2_SCL PWM1/4 UART1_CTS SPI1_MISO ADC1/15 A5 PA_11
I2C2_SDA UART1_RTS SPI1_MOSI ADC1/16 A4 PA_12
SPI_SCK SPI1_SCK ADC1/5 A3 PA_5
SPI1_CS PWM14/1 SPI2_MOSI ADC1/4 A2 PA_4
UART2_RTS SPI1_SCK ADC1/1 A1 PA_1
UART2_CTS SPI2_SCK ADC1/0 A0 PA_0
AREF
3v3
PWM_OUT PWM1/2 UART1_RTS SPI1_SCK D13 PB_3

Figure 1 Stm32G031 PINOUT

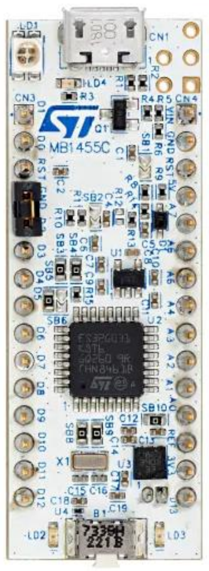
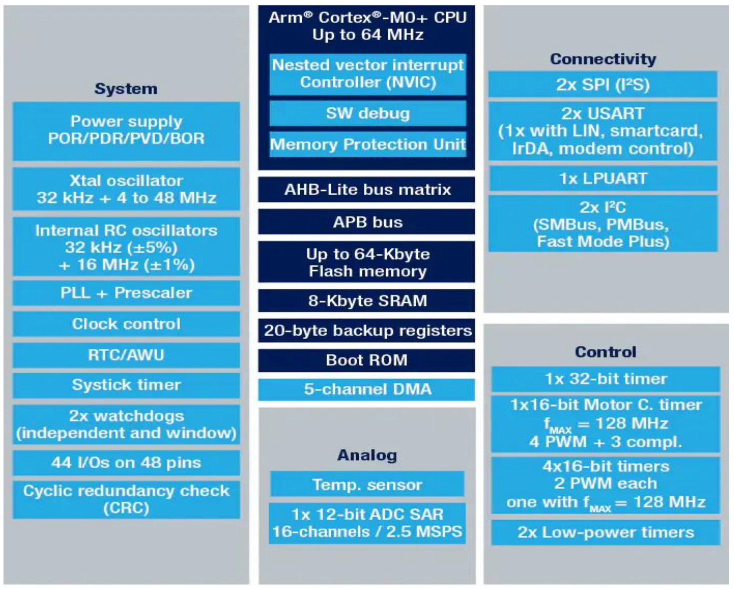
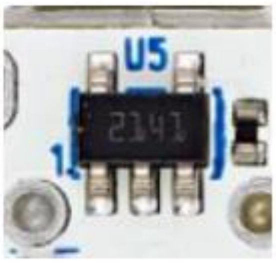


Figure 2

The STM32G031x4/x6/x8 mainstream microcontrollers are based on high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at up to 64 MHz frequency. Offering a high level of integration, they are suitable for a wide range of applications in consumer, industrial and appliance domains and ready for the Internet of Things (IoT) solutions. The devices incorporate a memory protection unit (MPU), high-speed embedded memories (8 Kbytes of SRAM and up to 64 Kbytes of Flash program memory with read protection, write protection, proprietary code protection, and securable area), DMA, an extensive range of system functions, enhanced I/Os, and peripherals. The devices offer standard communication interfaces (two I2Cs, two SPIs / one I2S, and two USARTs), one 12-bit ADC (2.5 MSps) with up to 19 channels, an internal voltage reference buffer, a low-power RTC, an advanced control PWM timer running at up to double the CPU frequency, four general-purpose 16-bit timers, a 32-bit general-purpose timer, two low-power 16-bit timers, two watchdog timers, and a SysTick timer.The devices come in packages with 8 to 48 pins.


System
Power supply POR/PDR/PVD/BOR
Xtal oscillator 32 kHz + 4 to 48 MHz
Internal RC oscillators 32 kHz (±5%) + 16 MHz (±1%)
PLL + Prescaler
Clock control
RTC/AWU
Systick timer
2x watchdogs (independent and window)
44 I/Os on 48 pins
Cyclic redundancy check (CRC)
Arm® Cortex®-M0+ CPU Up to 64 MHz
Nested vector interrupt Controller (NVIC)
SW debug
Memory Protection Unit
AHB-Lite bus matrix
APB bus
Up to 64-Kbyte Flash memory
8-Kbyte SRAM
20-byte backup registers
Boot ROM
5-channel DMA
Analog
Temp. sensor
1x 12-bit ADC SAR 16-channels / 2.5 MSPS
Connectivity
2x SPI (I²S)
2x USART (1x with LIN, smartcard, IrDA, modem control)
1x LPUART
2x I²C (SMBus, PMBus, Fast Mode Plus)
Control
1x 32-bit timer
1x16-bit Motor C. timer f_{MAX} = 128 MHz 4 PWM + 3 compl.
4x16-bit timers 2 PWM each one with f_{MAX} = 128 MHz
2x Low-power timers

"STM32 F103CBT6 GQ25Q1293" this is the ST F103CBT6 microcontroller, used as the STLINK/V2-1. This microcontroller is used as a debugger/programmer



There is a IC marked “2141” which is a component named STMPS2141STR, this component is a enhanced single channel power switch with active low enable pin.

Figure 3

There is a IC named “USBLC6-2P6”, which is an USB OTG FS Electro Static Decharge protection controller.

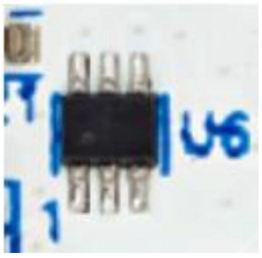
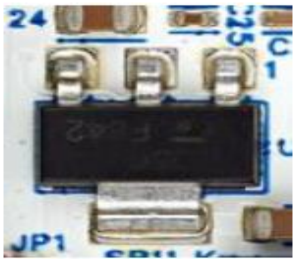


Figure 4



We have an IC numbered U8, named LD1117S59TR. This is a REG Low-dropout regulator, working on 5V, 0.8 amps

Figure 5

2- Problem 2

Implement assembly code that will light up 1 LED connected to pin PA8. For this problem (and lab) you do not need to worry about clocking the peripherals (and that is fine if you don't even know what that means).

● Just assume an LED is connected to the given port/pin. A represents the port GPIOA, and 8 represents the pin 8.

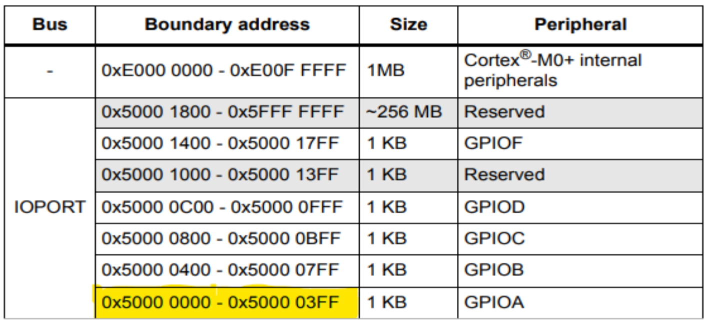
● Get the port address from RM0444. Look for GPIOA base address from the memory map table (Section 2) Then find the offset for ODR register in GPIO peripheral (Section 6).

● To turn on an LED, just write a 1 to the corresponding bit location in the ODR register memory address. ○ For example, to turn on an LED on PE5, you need to write 1 to the GPIOE ODR register bit 5.

● Remember in Keil, defining a word is different from GNU assembly. Check the lecture slides (or book).

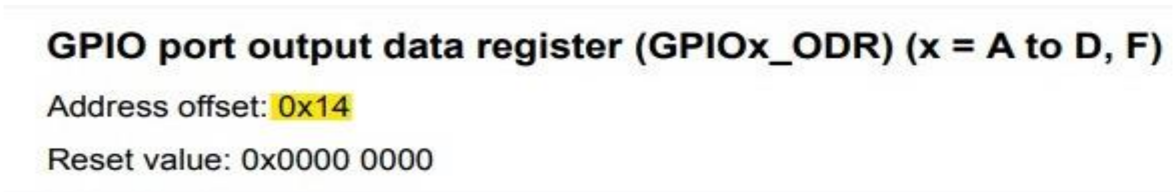
Solution :

It is requested that code will light up a LED connected to pin PA8. From the datasheet, as seen below, the GPIOA limit adress is “0x5000 0000 – 0x5000 03FF” learned. Offset for the ODR register is also required. Therefore, the offset for the ODR register was found in the GPIO peripheral. Offset value is “0x14”.



Bus	Boundary address	Size	Peripheral
-	0xE000 0000 - 0xE00F FFFF	1MB	Cortex®-M0+ internal peripherals
IOPORT	0x5000 1800 - 0x5FFF FFFF	~256 MB	Reserved
	0x5000 1400 - 0x5000 17FF	1 KB	GPIOF
	0x5000 1000 - 0x5000 13FF	1 KB	Reserved
	0x5000 0C00 - 0x5000 0FFF	1 KB	GPIOD
	0x5000 0800 - 0x5000 0BFF	1 KB	GPIOC
	0x5000 0400 - 0x5000 07FF	1 KB	GPIOB
	0x5000 0000 - 0x5000 03FF	1 KB	GPIOA

Figure 6 Boundary adress for GPIOA



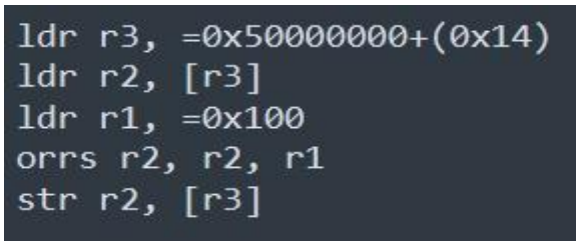
GPIO port output data register (GPIOx_ODR) (x = A to D, F)

Address offset: 0x14

Reset value: 0x0000 0000

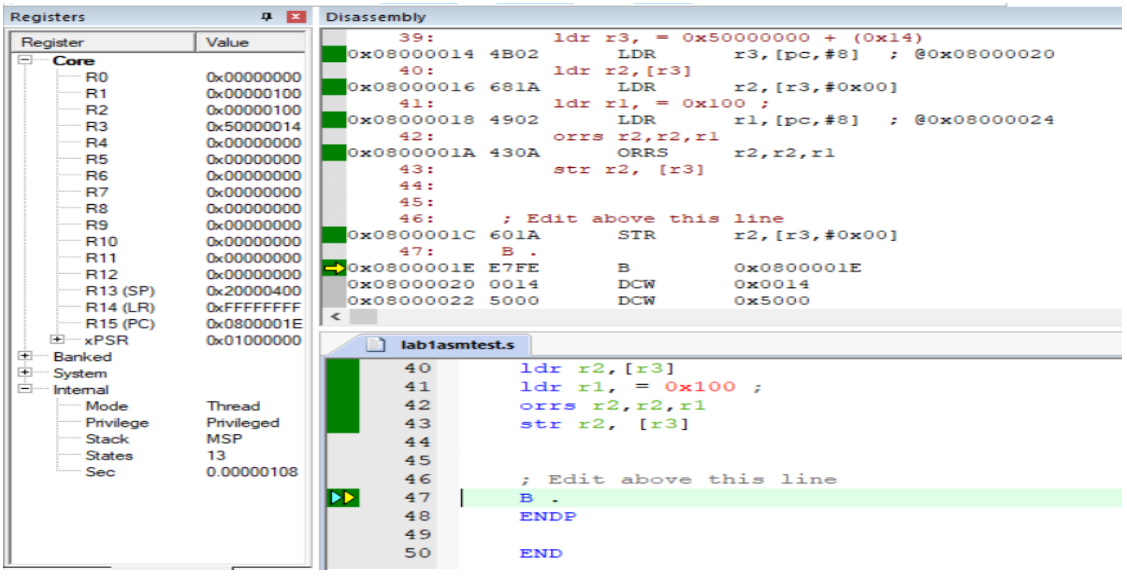
Figure 7 Adress Ofsett

To turn on a led, the corresponding bit in the register must be set to 1 . Since we are asked to light the LED connected to the PA8 pin, this corresponds to 0x100 in hexadecimal.
0x100 -> binary 0001 0000 0000. Assembly code is as shown below.



```
ldr r3, =0x50000000+(0x14)
ldr r2, [r3]
ldr r1, =0x100
orrs r2, r2, r1
str r2, [r3]
```

Figure 8 Assembly Code



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000100
R2	0x00000100
R3	0x50000014
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800001E
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	13
Sec	0.00000108

Disassembly

```
    39:           ldr r3, = 0x50000000 + (0x14)
0x08000014 4B02       LDR       r3,[pc,#8]  ; @0x08000020
    40:           ldr r2,[r3]
0x08000016 681A       LDR       r2,[r3,#0x00]
    41:           ldr r1, = 0x100 ;
0x08000018 4902       LDR       r1,[pc,#8]  ; @0x08000024
    42:           orrs r2,r2,r1
0x0800001A 430A       ORRS      r2,r2,r1
    43:           str r2, [r3]
    44:
    45:
    46:       ; Edit above this line
0x0800001C 601A       STR       r2,[r3,#0x00]
    47:       B .
0x0800001E E7FE       B         0x0800001E
0x08000020 0014       DCW       0x0014
0x08000022 5000       DCW       0x5000
```

lab1asmtest.s

```
        ldr r2,[r3]
        ldr r1, = 0x100 ;
        orrs r2,r2,r1
        str r2, [r3]


        ; Edit above this line
        B .
        ENDP

        END
```

Figure 9 Assembly Code in Keil

2- Problem 3

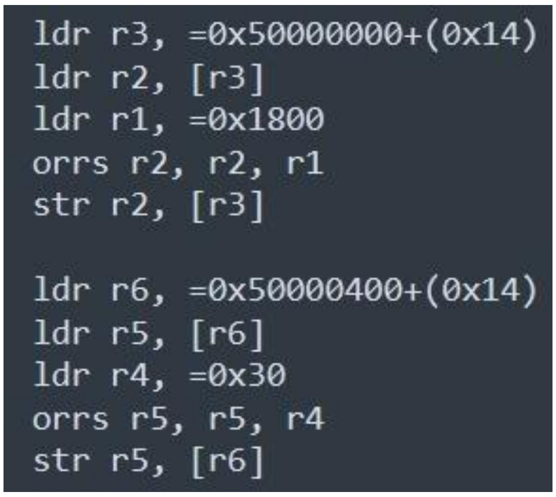
Implement assembly code that will light up 4 LEDs connected to pins PA11, PA12, PB4, PB5.

Solution :

In this question, the LEDs connected to the PA11, P12, PB4 and PB5 pins are requested to light up. First of all, the GPIOA base addresses are 0x5000 0000 as in the previous question. The ODR address is 0x14, as in the first question. The 12th bit must be 1 to turn on the led of the 11th pin, and the 13th bit must be 1 to turn the led of the 12th pin on. This would be equal to 1100000000000. Heximal counterpart is 0x1800.

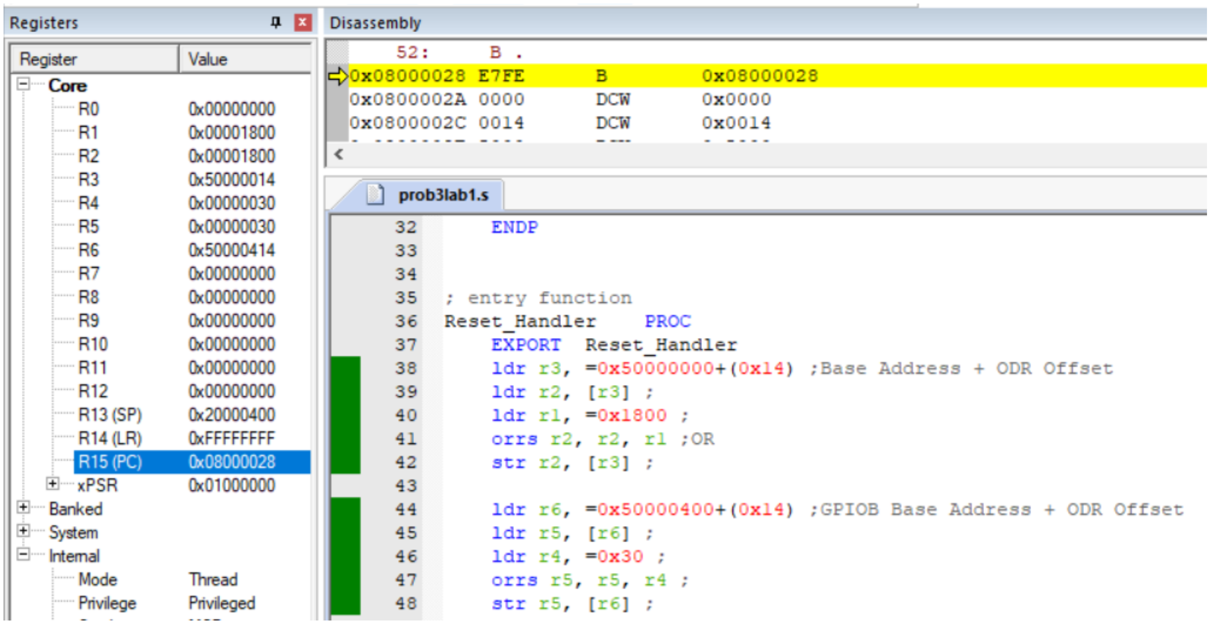
The GPIOB base address for PB4 and PB5 is given in section 2 as 0x5000 0400 - 0x5000 07FF. The ODR offset address is 0x14.The 5th bit must be 1 for the 4th pin's led to light, and the 6th bit must be 1 for the 5th pin's led to be lit. This would equal 110000. Its hexadecimal equivalent is 0x30.With these data, the code related to keil is written.

Bit set operation is done with the orrs command. The reason is that the or command is used because the leds are on in the 1 state. The or command will output as 1 (except 0-0).



```
ldr r3, =0x50000000+(0x14)
ldr r2, [r3]
ldr r1, =0x1800
orrs r2, r2, r1
str r2, [r3]

ldr r6, =0x50000400+(0x14)
ldr r5, [r6]
ldr r4, =0x30
orrs r5, r5, r4
str r5, [r6]
```

Figure 10 Assembly



Register	Value
Core	
R0	0x00000000
R1	0x00001800
R2	0x00001800
R3	0x50000014
R4	0x00000030
R5	0x00000030
R6	0x50000414
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000028
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged

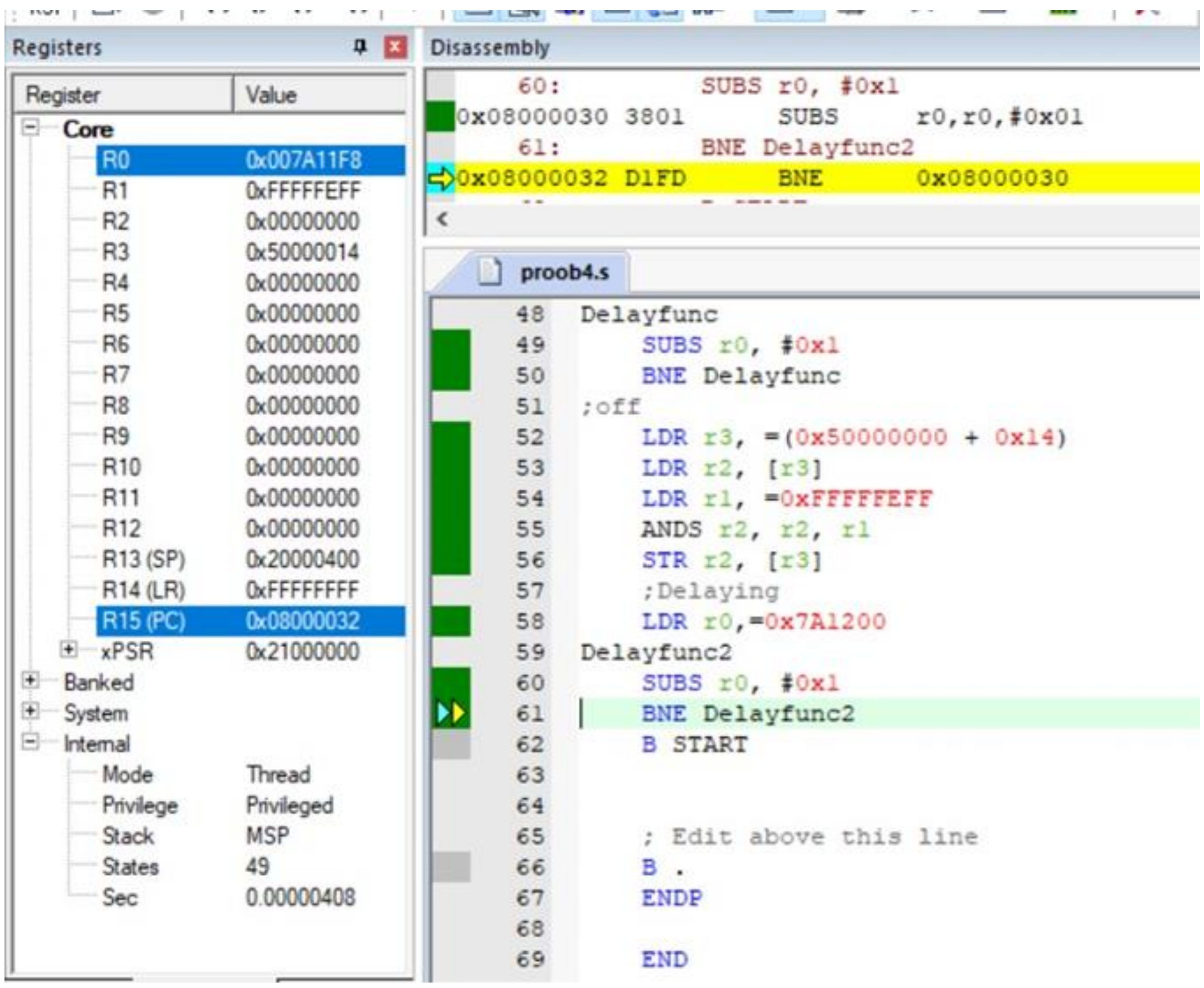
```
    52:      B .
0x08000028 E7FE      B         0x08000028
0x0800002A 0000      DCW       0x0000
0x0800002C 0014      DCW       0x0014
```

```
    ENDP


; entry function
Reset_Handler    PROC
    EXPORT  Reset_Handler
    ldr r3, =0x50000000+(0x14) ;Base Address + ODR Offset
    ldr r2, [r3] ;
    ldr r1, =0x1800 ;
    orrs r2, r2, r1 ;OR
    str r2, [r3] ;

    ldr r6, =0x50000400+(0x14) ;GPIOB Base Address + ODR Offset
    ldr r5, [r6] ;
    ldr r4, =0x30 ;
    orrs r5, r5, r4 ;
    str r5, [r6] ;
```

Figure 11 Assembly Code in Keil

Problem 4

Implement a delay routine and have 1 LED described on Problem 2 toggle in roughly 1 second intervals. For this assume the clock is running at 16 Mhz. (It will be impossible to single step through 1 second delay routine, so utilize breakpoints).



Registers

Register	Value
Core	
R0	0x007A11F8
R1	0xFFFFFEFF
R2	0x00000000
R3	0x50000014
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000032
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	49
Sec	0.00000408

Disassembly

```
    60:          SUBS r0, #0x1
0x08000030 3801       SUBS      r0,r0,#0x01
    61:          BNE Delayfunc2
0x08000032 D1FD       BNE       0x08000030
```

proob4.s

```
Delayfunc
    SUBS r0, #0x1
    BNE Delayfunc
;off
    LDR r3, =(0x50000000 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFEFF
    ANDS r2, r2, r1
    STR r2, [r3]
    ;Delaying
    LDR r0,=0x7A1200
Delayfunc2
    SUBS r0, #0x1
    BNE Delayfunc2
    B START


    ; Edit above this line
    B .
    ENDP

    END
```

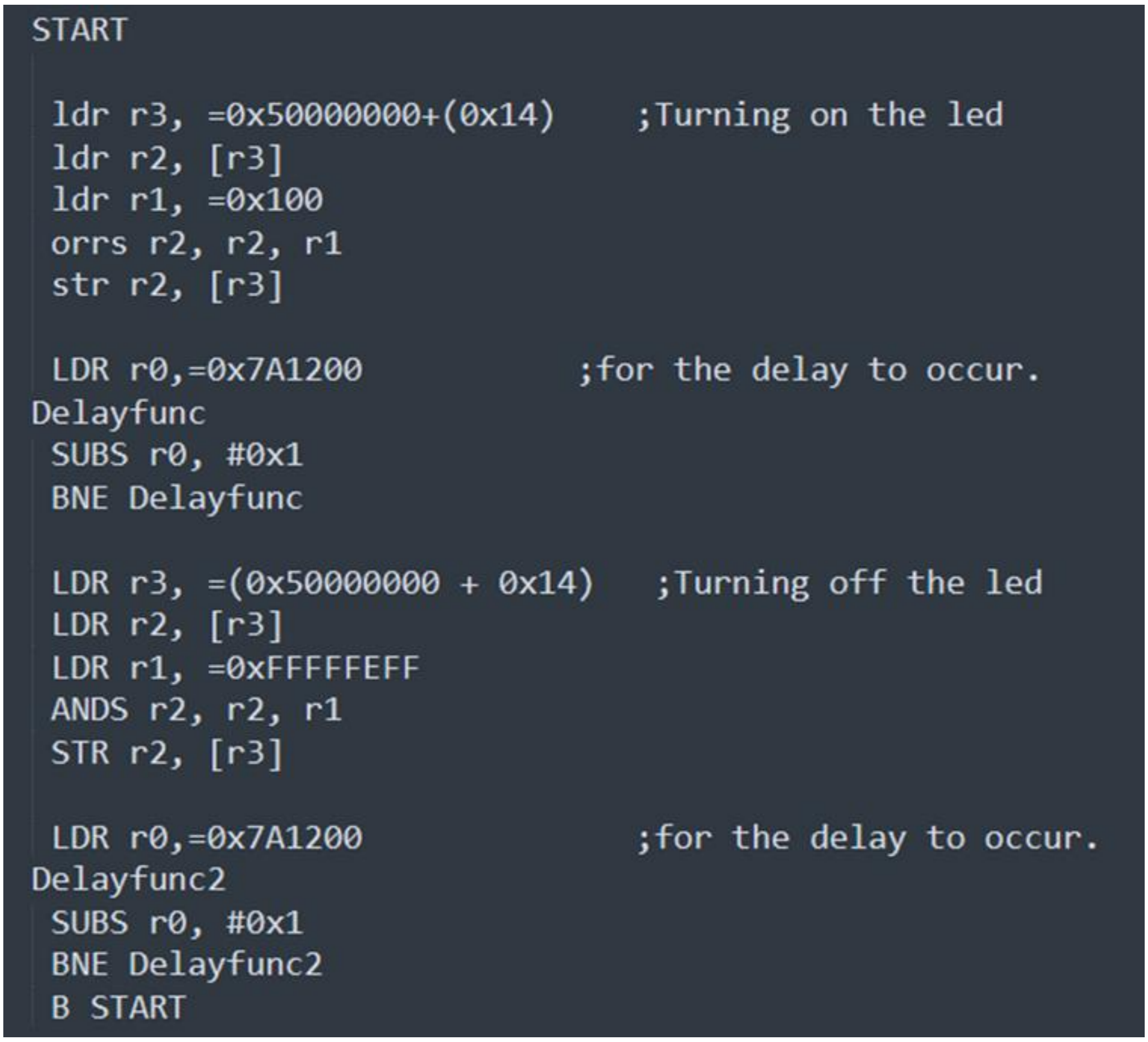
Figure 12 Assembly Code in Keil

Subtraction and branch not equal operations take 2 cycles. Divided 16 million by 2 for 2 cycles per whole operation, gathered 8 million per second. So set r0 to “0x7A1200” to get approximately 1 second delay.

We need to perform the LED burning process we did in Problem 2 in this question as well. Unlike Problem 2, we need to define a delay time and perform the LED off operation. We set the appropriate GPIO address and the appropriate pin. Thanks to the function structures we used, we applied the delay times on the relevant register.

0xFFFFFEFFF= 11111111111111111111111011111111

As can be seen from its expansion, we obtained the relevant hexadecimal number so that the bit to be turned off is zero. The LED off event was made by ANDS processing of the binary number given above with the relevant address. We made the LED on and off operations in the set reset logic



```
START

 ldr r3, =0x50000000+(0x14)    ;Turning on the led
 ldr r2, [r3]
 ldr r1, =0x100
 orrs r2, r2, r1
 str r2, [r3]

 LDR r0,=0x7A1200           ;for the delay to occur.
Delayfunc
 SUBS r0, #0x1
 BNE Delayfunc

 LDR r3, =(0x50000000 + 0x14)   ;Turning off the led
 LDR r2, [r3]
 LDR r1, =0xFFFFFEFF
 ANDS r2, r2, r1
 STR r2, [r3]

 LDR r0,=0x7A1200              ;for the delay to occur.
Delayfunc2
 SUBS r0, #0x1
 BNE Delayfunc2
 B START
```

Figure 13 Assembly Code