GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

ELM335

MICROPROCESSORS LAB

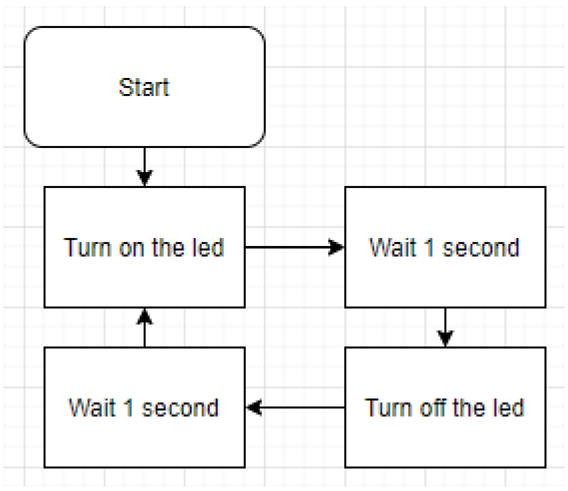
# LAB 2 Report

| |
|---|
| Prepared by |
| 171024086 Berat KIZILARMUT |

## 1. Introduction

In this lab new practices will be made in assembly language. We will experiment with using on-board LEDs, connecting external LEDs, toggling them and shifting them in various patterns. Whole lab is completed **WITHOUT** the test board since it has not yet arrived. Everything is done in keil uVision. Apologies for the inconvenience.

## 2. Problem 1



Chose the onboard led as PC6. Unlike the problem 4 on lab1, clocking the peripherals are required. Assembly code starts with that, enabling the GPIOC clock, since the led is in C peripherals, by setting bit2, 3rd bit, as 1 on IOPENR. To achieve that, address (0x40021000 + 0x34) is set as 0x4.

After clocking GPIOC it's time to set PC6 as output, bits 13 and 12 are set to 0 and 1 on GPIOC modder. To achieve that, on address (0x50000800 + 0x00) firstly bit 13 is set as zero, nextly bit 12 is set as one.

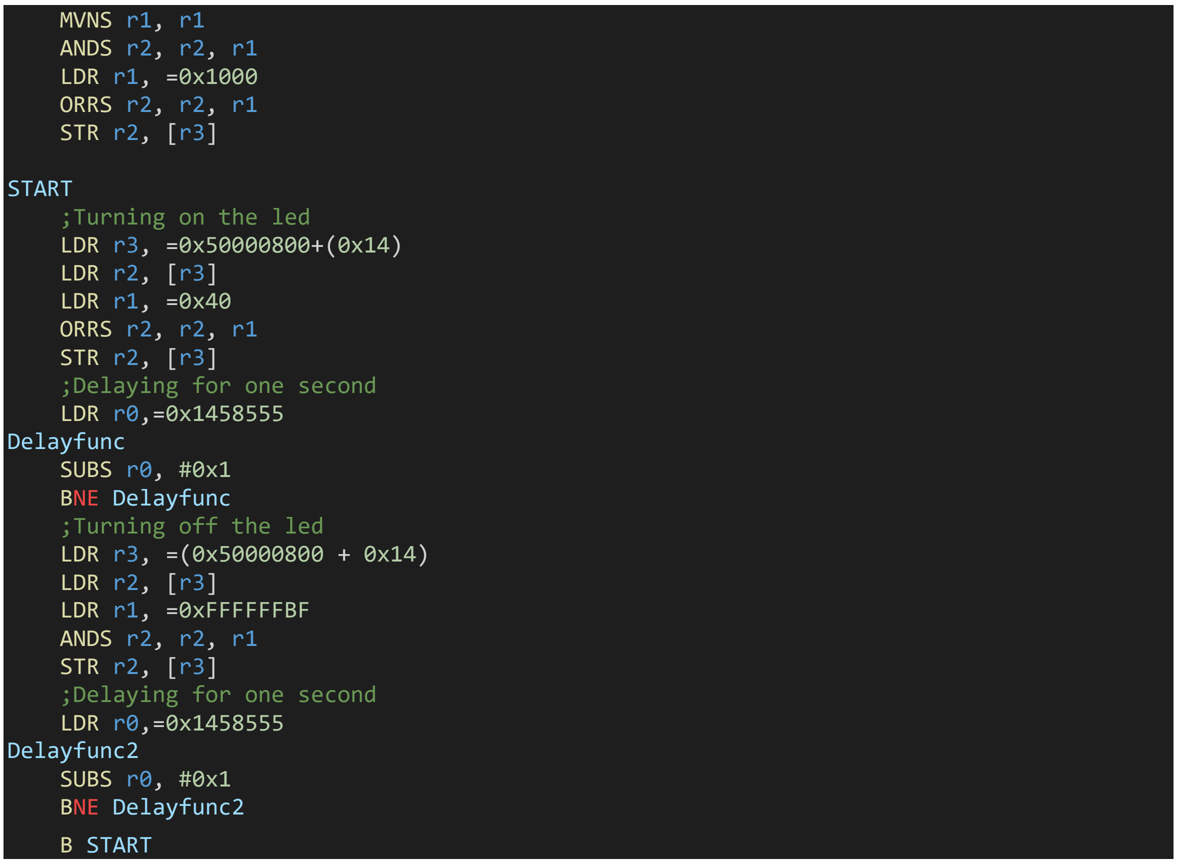
After the prep phase is done and peripherals at the ready, it's time to start turning on the led. Firstly setting the address of the led 0x50000800+(0x14) which is GPIOC address + ODR offset. Then setting the value as 0x40 for pin 6. After this, led is lit by using or operation with the value of the set address and the newly set pin value.

Led should be turned on now, it's time to wait one second. 1 Second delay is achieved by giving the microprocessor task to waste its time. I've set the XTAL value as 64 MHz on keil since our board's max clock speed is this.. Since branch not equal and subtract operations take 3 cycles total, I've set a register value as 64 Million / 3 which is 21.3 million decimal, 1458555 hexadecimal. Exactly 1 second time is wasted by making the microprocessor do 64 million cycles.

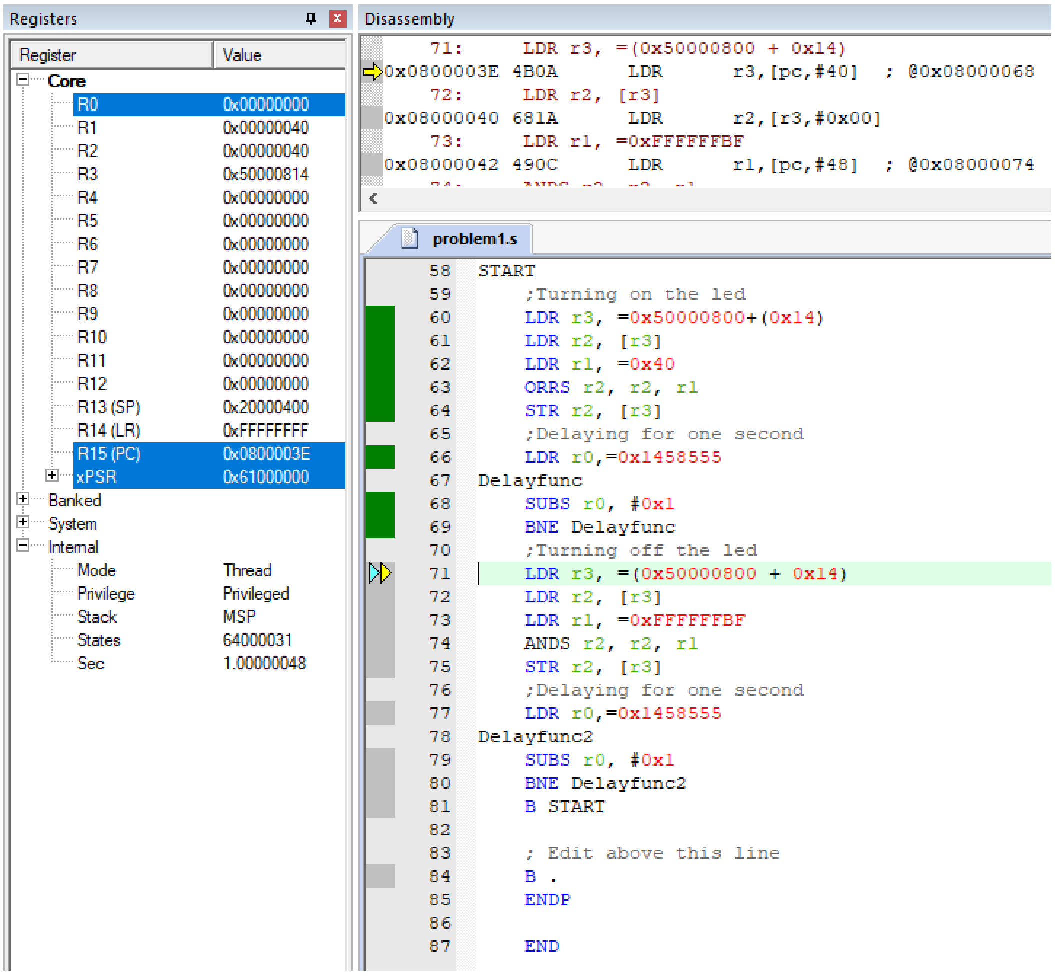
After the waiting period is over almost the exact almost the exact procedure is followed except setting the value as 0x40, this time value is set as 0xFFFFFFBF. Then the led turns off and another second is wasted on cycles. After the wait, it's all looped back together.

Code is given without the provided assembly templates to not cause a mess.

```
;Clocking the peripherals
;Enabling GPIOC clock, Setting bit2 on IOPENR 1
LDR r3, =(0x40021000 + 0x34)
LDR r2, [r3]
LDR r1, =0x4
ORRS r2, r2, r1
STR r2, [r3]

;Setting PC6 for output by setting bits 13 and 12 to 01
LDR r3, =(0x50000800 + 0x00)
LDR r2, [r3]
LDR r1, =0x2000
```



```
    MVNS r1, r1
    ANDS r2, r2, r1
    LDR r1, =0x1000
    ORRS r2, r2, r1
    STR r2, [r3]

START
    ;Turning on the led
    LDR r3, =0x50000800+(0x14)
    LDR r2, [r3]
    LDR r1, =0x40
    ORRS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc
    SUBS r0, #0x1
    BNE Delayfunc
    ;Turning off the led
    LDR r3, =(0x50000800 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFFBF
    ANDS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc2
    SUBS r0, #0x1
    BNE Delayfunc2

    B START
```



Registers

| Register | Value |
|---|---|
| Core | |
| R0 | 0x00000000 |
| R1 | 0x00000040 |
| R2 | 0x00000040 |
| R3 | 0x50000814 |
| 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) | 0x0800003E |
| xPSR | 0x61000000 |
| Banked | |
| System | |
| Internal | |
| Mode | Thread |
| Privilege | Privileged |
| Stack | MSP |
| States | 64000031 |
| Sec | 1.00000048 |

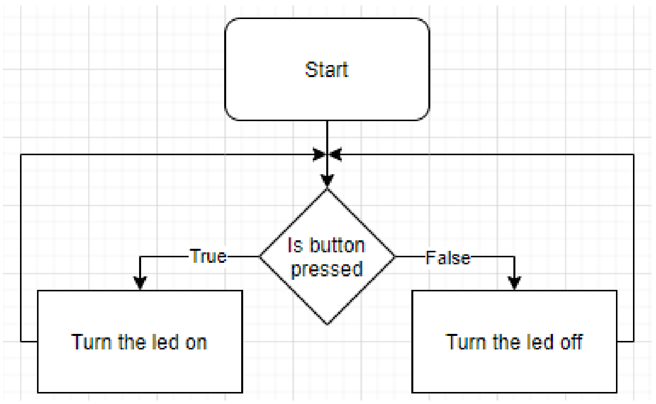
Disassembly

```
    71:     LDR r3, =(0x50000800 + 0x14)
0x0800003E 4B0A      LDR      r3,[pc,#40]  ; @0x08000068
    72:     LDR r2, [r3]
0x08000040 681A      LDR      r2,[r3,#0x00]
    73:     LDR r1, =0xFFFFFFBF
0x08000042 490C      LDR      r1,[pc,#48]  ; @0x08000074
```

problem1.s

```
START
    ;Turning on the led
    LDR r3, =0x50000800+(0x14)
    LDR r2, [r3]
    LDR r1, =0x40
    ORRS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc
    SUBS r0, #0x1
    BNE Delayfunc
    ;Turning off the led
    LDR r3, =(0x50000800 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFFBF
    ANDS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc2
    SUBS r0, #0x1
    BNE Delayfunc2
    B START

    ; Edit above this line
    B .
    ENDP

    END
```

## 3. Problem 2



Chose the onboard led PC6 to light up and chose the PB6 to connect the button. First order of business is to clock the peripherals. Clocked both GPIOB and GPIOC at the same time by setting IOPENR 0x6. After clocking the peripherals, set the PB6 for input using GPIOB modder and setting the bits 13 and 12 as zero. Then setting the PC6 for output using GPIOC modder and setting the bits 13 and 12 as one and zero.

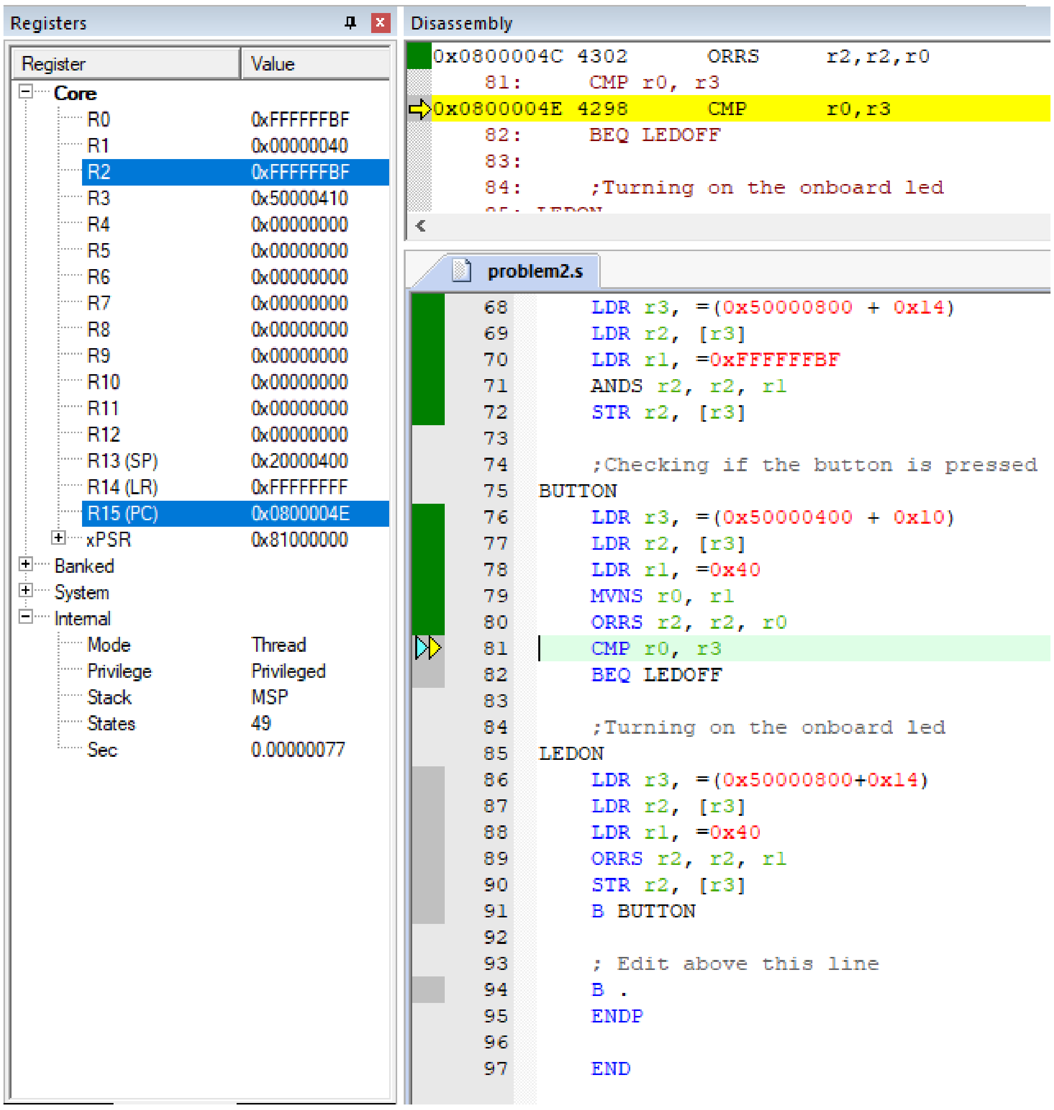
Getting to the actual turning the led off or on part, firstly starting with the led turning off code. Using my usual technique to turn off the led PC6 on the address (0x50000800 + 0x14) setting the value 0xFFFFFFBF.

After the led turning off part is done, we get to the button checking part. Crucial part of using an input is I got used to using ODR offset previously, but it is required to use the IDR offset now, so the address loaded to the register is (0x50000400 + 0x10), rather than (0x50000400 + 0x14). After setting the address, I load the pin 6 value, then invert it, then or operation it with the read input value to compare it. If the r0 and r3 is equal, then this means led is turned off, so it loops back to ledoff. If they are not equal the code goes forward to the led on part.

Turning the led on is almost the same as the ledoff part, except setting the value as 0x40, rather than 0xFFFFFFBF.

Code is given without the provided assembly templates to not cause a mess.

```
    ;Clocking the peripherals
    ;Enabling GPIOB and GPIOC clock, Setting bit1 and bit2 on IOPENR 1
    LDR r3, =(0x40021000 + 0x34)
    LDR r2, [r3]
    LDR r1, =0x6
    ORRS r2, r2, r1
    STR r2, [r3]

    ;Setting PB6 for input by setting bits 13 and 12 to 00 using modder
    LDR r3, =(0x50000400 + 0x00)
    LDR r2, [r3]
    LDR r1, =0x3000
    MVNS r1, r1
    ANDS r2, r2, r1
    STR r2, [r3]

    ;Setting PC6 for output by setting bits 13 and 12 to 01 using modder
    LDR r3, =(0x50000800 + 0x00)
    LDR r2, [r3]
    LDR r1, =0x2000
    MVNS r1, r1
    ANDS r2, r2, r1
    LDR r1, =0x1000
    ORRS r2, r2, r1
    STR r2, [r3]
    ;Turning off the onboard led
LEDOFF
    LDR r3, =(0x50000800 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFFBF
    ANDS r2, r2, r1
    STR r2, [r3]

    ;Checking if the button is pressed
BUTTON
    LDR r3, =(0x50000400 + 0x10)
    LDR r2, [r3]
    LDR r1, =0x40
    MVNS r0, r1
    ORRS r2, r2, r0
    CMP r0, r3
    BEQ LEDOFF

    ;Turning on the onboard led
LEDON
    LDR r3, =(0x50000800+0x14)
    LDR r2, [r3]
    LDR r1, =0x40
    ORRS r2, r2, r1
    STR r2, [r3]
    B BUTTON
```



Registers

| Register | Value |
|---|---|
| Core | |
| R0 | 0xFFFFFFBF |
| R1 | 0x00000040 |
| R2 | 0xFFFFFFBF |
| R3 | 0x50000410 |
| 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) | 0x0800004E |
| xPSR | 0x81000000 |
| Banked | |
| System | |
| Internal | |
| Mode | Thread |
| Privilege | Privileged |
| Stack | MSP |
| States | 49 |
| Sec | 0.00000077 |

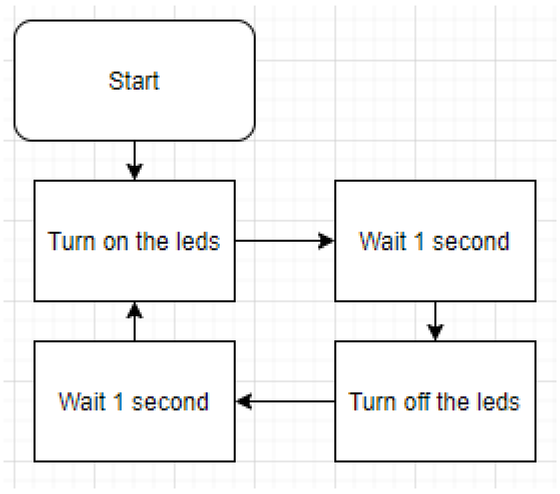
Disassembly

```
0x0800004C 4302      ORRS     r2,r2,r0
    81:     CMP r0, r3
0x0800004E 4298      CMP      r0,r3
    82:     BEQ LEDOFF
    83:
    84:     ;Turning on the onboard led
```

problem2.s

```
    LDR r3, =(0x50000800 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFFBF
    ANDS r2, r2, r1
    STR r2, [r3]

    ;Checking if the button is pressed
BUTTON
    LDR r3, =(0x50000400 + 0x10)
    LDR r2, [r3]
    LDR r1, =0x40
    MVNS r0, r1
    ORRS r2, r2, r0
    CMP r0, r3
    BEQ LEDOFF

    ;Turning on the onboard led
LEDON
    LDR r3, =(0x50000800+0x14)
    LDR r2, [r3]
    LDR r1, =0x40
    ORRS r2, r2, r1
    STR r2, [r3]
    B BUTTON

    ; Edit above this line
    B .
    ENDP

    END
```

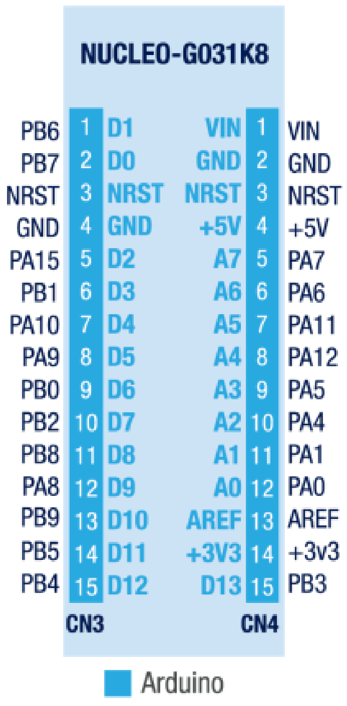
## 4. Problem 3



Utilizing the code from problem1 and upscaling it. Since we are gonna use the external pins, I looked up to pinout of the board and chose pins PB0 to PB7 to connect the leds.

First thing to do is clock the GPIOB, using IOPENR, setting bit1 as one. After this set the pins as output for all 8 leds at once by setting the GPIOB Modder, (0x50000400 + 0x00) as 0xAAAA and inverting it, and operation with the existing value thus setting the first 16 bits zero, then setting the even bits as one by setting the value 0x5555.

Figure 8. Arduino™ connectors pinout

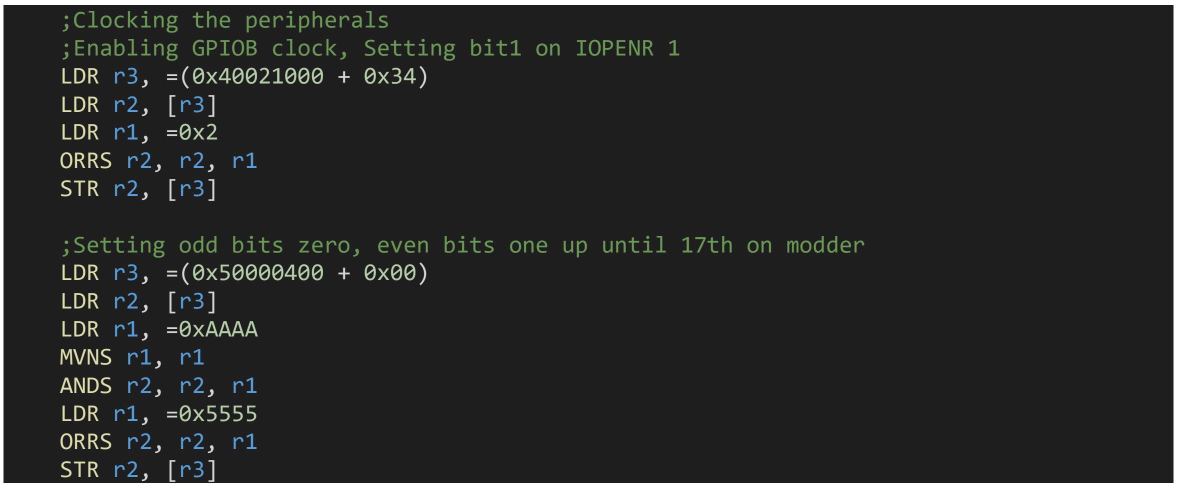


Turning the leds on is an easy task after that. Just like turning on a single led, set the values of pins on the "GPIOB base address + ODR Offset" to one. In this case, setting the value as 0xFF to set 8 bits as one.

Since the processor is still set to 64 Mhz, and the delayfunction cycle amount is still 3, set the subtraction value as 0x1458555 again.

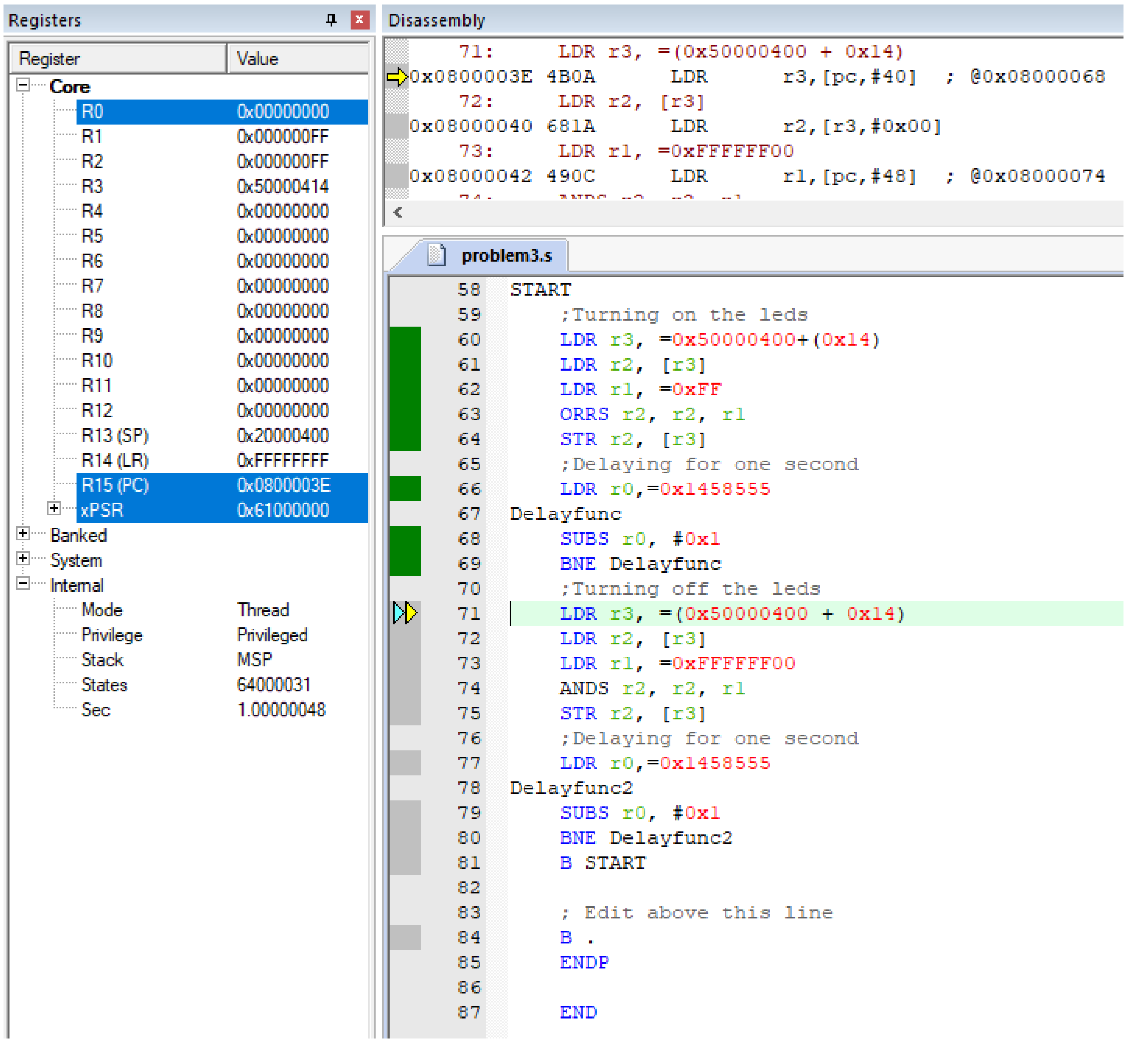
After the waiting period is over almost the exact almost the exact procedure is followed except setting the value as 0xFF, this time value is set as 0xFFFFFF00. Then the leds turn off and another second is wasted on cycles. After the wait, it's all looped back together.

Code is given without the provided assembly templates to not cause a mess.



```
    ;Clocking the peripherals
    ;Enabling GPIOB clock, Setting bit1 on IOPENR 1
    LDR r3, =(0x40021000 + 0x34)
    LDR r2, [r3]
    LDR r1, =0x2
    ORRS r2, r2, r1
    STR r2, [r3]

    ;Setting odd bits zero, even bits one up until 17th on modder
    LDR r3, =(0x50000400 + 0x00)
    LDR r2, [r3]
    LDR r1, =0xAAAA
    MVNS r1, r1
    ANDS r2, r2, r1
    LDR r1, =0x5555
    ORRS r2, r2, r1
    STR r2, [r3]
```

```
START
    ;Turning on the leds
    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0xFF
    ORRS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc
    SUBS r0, #0x1
    BNE Delayfunc
    ;Turning off the leds
    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFF00
    ANDS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc2
    SUBS r0, #0x1
    BNE Delayfunc2
    B START
```

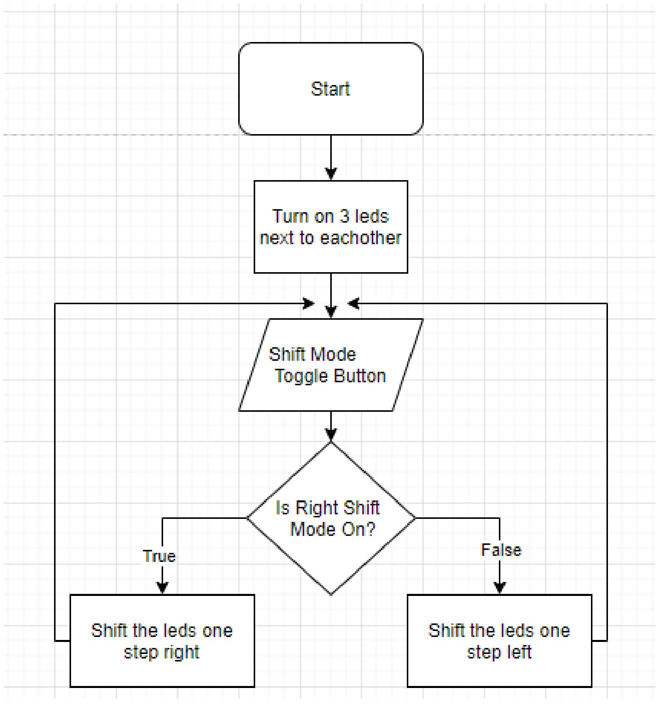


| Register | Value |
|---|---|
| Core | |
| R0 | 0x00000000 |
| R1 | 0x000000FF |
| R2 | 0x000000FF |
| R3 | 0x50000414 |
| R4 | 0x00000000 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x20000400 |
| R14 (LR) | 0xFFFFFFFF |
| R15 (PC) | 0x0800003E |
| xPSR | 0x61000000 |
| Banked | |
| System | |
| Internal | |
| Mode | Thread |
| Privilege | Privileged |
| Stack | MSP |
| States | 64000031 |
| Sec | 1.00000048 |

```
    71:      LDR r3, =(0x50000400 + 0x14)
0x0800003E 4B0A      LDR      r3,[pc,#40]  ; @0x08000068
    72:      LDR r2, [r3]
0x08000040 681A      LDR      r2,[r3,#0x00]
    73:      LDR r1, =0xFFFFFF00
0x08000042 490C      LDR      r1,[pc,#48]  ; @0x08000074
```

```
START
    ;Turning on the leds
    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0xFF
    ORRS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc
    SUBS r0, #0x1
    BNE Delayfunc
    ;Turning off the leds
    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0xFFFFFF00
    ANDS r2, r2, r1
    STR r2, [r3]
    ;Delaying for one second
    LDR r0,=0x1458555
Delayfunc2
    SUBS r0, #0x1
    BNE Delayfunc2
    B START

    ; Edit above this line
    B .
    ENDP

    END
```

## 5. Problem 4


Start
Turn on 3 leds next to eachother
Shift Mode Toggle Button
Is Right Shift Mode On?
True
False
Shift the leds one step right
Shift the leds one step left

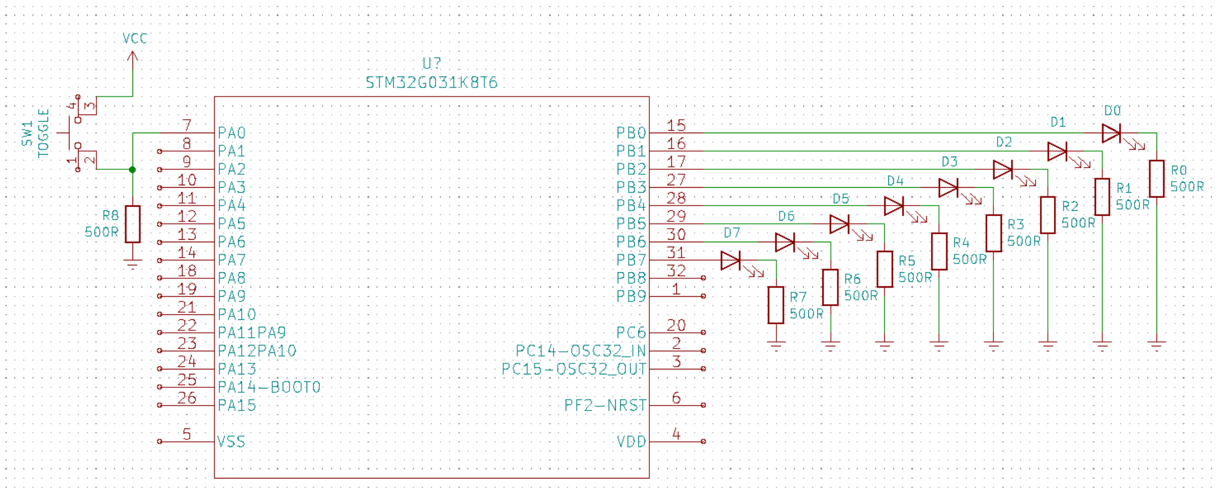
Just like before, first order of business is to enable the clocks on peripherals, in this case GPIOA and GPIOB. Set the bits 0 and 1 on IOPENR as 1. After enabling their clocks, set the all of the 8 leds are output on GPIOB modder. Then set the PA0 as input on GPIOA modder. Set the r6, which is the toggle switch to change shift modes, to zero.

Prep phase is completed. Created a state template and copied it to create 8 total states. This state template starts with reading the button input, from (0x50000000 + 0x10) which is GPIOA base address + IDR Offset. Compares the value, if the value is zero it skips, if it's one it toggles the internal switch which defines the shifting order. If r6 value is one, lights shift left, if it's zero they shift right by default.

After the button checking part, every state always turns off all the leds momentarily to cleanup any mess that can happen with previous states. Then the leds turn back on depending on the state number, for example on state T2, leds 1 2 and 3 are turned on.

After leds are turned on we get to the delaying part, it is required to delay the states 100 ms each time. To achieve that I made the microprocessor waste time on a useless task which is subtraction in this case. Since I've set the processor to run at 64 Mhz, I gave it 2.1 million subtraction operations which takes 3 cycles each to do including the branch operations.

100 ms later the delaying is done and we get to the next state deciding phase. Checking the switch value, if the switch is zero we go to state T(x+1). If the switch is one, we go to state T(x-1). States have all the same functions we explained above.

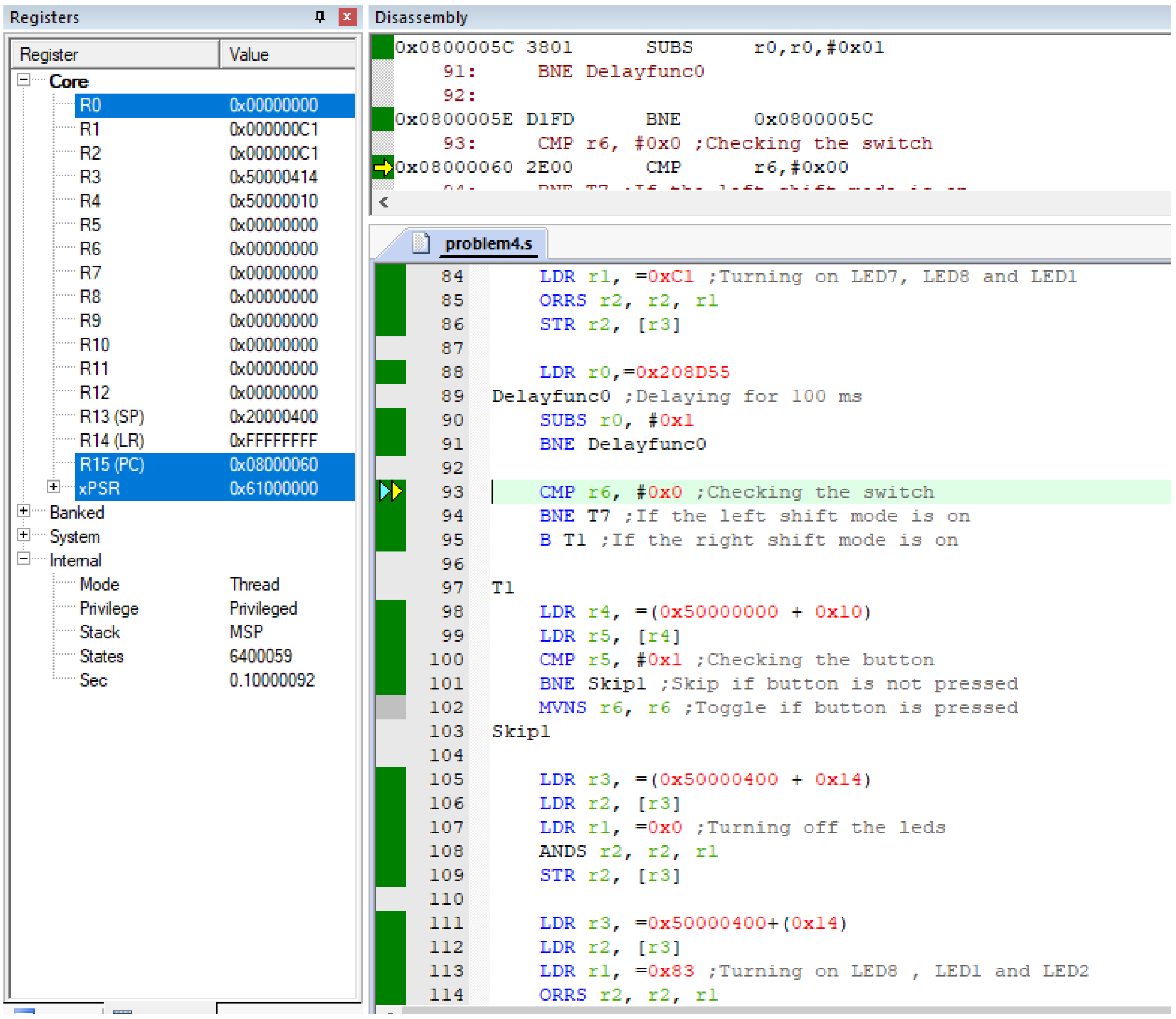


Code is given without the provided assembly templates to not cause a mess.

```
    ;Clocking the peripherals
    ;Enabling GPIOA and GPIOB clock, Setting bit0 and 1 on IOPENR as 1
    LDR r3, =(0x40021000 + 0x34)
    LDR r2, [r3]
    LDR r1, =0x3
    ORRS r2, r2, r1
    STR r2, [r3]

    ;Setting odd bits zero, even bits one up until 17th on modder to set the 8 leds on GPIOB modder
    LDR r3, =(0x50000400 + 0x00)
    LDR r2, [r3]
    LDR r1, =0xAAAA
    MVNS r1, r1
    ANDS r2, r2, r1
    LDR r1, =0x5555
    ORRS r2, r2, r1
    STR r2, [r3]

    ;Setting PA0 for input by setting bits 0 and 1 to 00 using modder
    LDR r3, =(0x50000000 + 0x00)
    LDR r2, [r3]
    LDR r1, =0x3
    MVNS r1, r1
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r6, =0x0 ;Zeroing the Switch value
T0
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip0 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip0

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0    ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0xC1 ;Turning on LED7, LED8 and LED1
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc0 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc0

    CMP r6, #0x0 ;Checking the switch
    BNE T7 ;If the left shift mode is on
    B T1 ;If the right shift mode is on

T1
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip1 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip1
```

```
    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x83 ;Turning on LED8 , LED1 and LED2
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc1 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc1

    CMP r6, #0x0 ;Checking the switch
    BNE T0 ;If the left shift mode is on
    B T2 ;If the right shift mode is on

T2
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    ;LDR r5, =0x1 ;Test fake button press, Delete the comment to test !!!!!!!!!!!!!!!!!!!!!!
    CMP r5, #0x1 ;Checking the button
    BNE Skip2 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip2

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x7 ;Turning on LED1 , LED2 and LED3
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc2 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc2

    CMP r6, #0x0 ;Checking the switch
    BNE T1 ;If the left shift mode is on
    B T3 ;If the right shift mode is on

T3
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip3 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip3

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]
    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0xE ;Turning on LED2 , LED3 and LED4
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc3 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc3

    CMP r6, #0x0 ;Checking the switch
    BNE T2 ;If the left shift mode is on
    B T4 ;If the right shift mode is on

T4
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip4 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip4

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x1C ;Turning on LED3 , LED4 and LED5
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc4 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc4

    CMP r6, #0x0 ;Checking the switch
    BNE T3 ;If the left shift mode is on
    B T5 ;If the right shift mode is on

T5
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip5 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip5

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x38 ;Turning on LED4 , LED5 and LED6
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
```

```
Delayfunc5 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc5

    CMP r6, #0x0 ;Checking the switch
    BNE T4 ;If the left shift mode is on
    B T6 ;If the right shift mode is on

T6
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip6 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip6

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x70 ;Turning on LED5 , LED6 and LED7
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc6 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc6


    CMP r6, #0x0 ;Checking the switch
    BNE T5 ;If the left shift mode is on
    B T7 ;If the right shift mode is on

T7
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip7 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip7

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0xE0 ;Turning on LED6 , LED7 and LED8
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc7 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc7

    CMP r6, #0x0 ;Checking the switch
    BNE T6 ;If the left shift mode is on
    B T0 ;If the right shift mode is on
```


Registers

| Register | Value |
|---|---|
| Core | |
| R0 | 0x00000000 |
| R1 | 0x000000C1 |
| R2 | 0x000000C1 |
| R3 | 0x50000414 |
| R4 | 0x50000010 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x20000400 |
| R14 (LR) | 0xFFFFFFFF |
| R15 (PC) | 0x08000060 |
| xPSR | 0x61000000 |
| Banked | |
| System | |
| Internal | |
| Mode | Thread |
| Privilege | Privileged |
| Stack | MSP |
| States | 6400059 |
| Sec | 0.10000092 |

Disassembly

```
0x0800005C 3801      SUBS     r0,r0,#0x01
    91:       BNE Delayfunc0
    92:
0x0800005E D1FD      BNE      0x0800005C
    93:       CMP r6, #0x0 ;Checking the switch
0x08000060 2E00      CMP      r6,#0x00
```

problem4.s

```
    LDR r1, =0xC1 ;Turning on LED7, LED8 and LED1
    ORRS r2, r2, r1
    STR r2, [r3]

    LDR r0,=0x208D55
Delayfunc0 ;Delaying for 100 ms
    SUBS r0, #0x1
    BNE Delayfunc0

    CMP r6, #0x0 ;Checking the switch
    BNE T7 ;If the left shift mode is on
    B T1 ;If the right shift mode is on

T1
    LDR r4, =(0x50000000 + 0x10)
    LDR r5, [r4]
    CMP r5, #0x1 ;Checking the button
    BNE Skip1 ;Skip if button is not pressed
    MVNS r6, r6 ;Toggle if button is pressed
Skip1

    LDR r3, =(0x50000400 + 0x14)
    LDR r2, [r3]
    LDR r1, =0x0 ;Turning off the leds
    ANDS r2, r2, r1
    STR r2, [r3]

    LDR r3, =0x50000400+(0x14)
    LDR r2, [r3]
    LDR r1, =0x83 ;Turning on LED8 , LED1 and LED2
    ORRS r2, r2, r1
```

## 6. References

1. ARM v6-M Architecture Reference Manual, ST Microelectronics
2. UM2591 User Manual STM32G0 Nucleo-32 board (MB1455), ST Microelectronics
3. Using a button with Arduino, programming gelectronics, https://www.programmingelectronics.com/tutorial-17-using-a-button-old-version/