GEBZE TECHNICAL UNIVERSITY

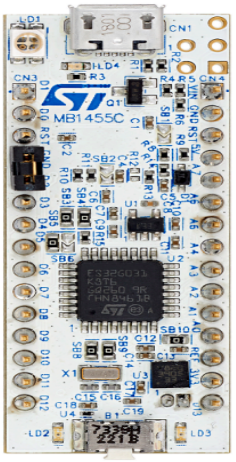
ELECTRONIC ENGINEERING

ELM335

MICROPROCESSORS LAB

LAB2 REPORT

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



❖ OBJECTIVE

In this lab we will work with assembly language, practice connecting basic components such as LEDs and buttons to the board, read from / write to them. In this way, we will develop our assembly coding knowledge and do more work on the STM32 card.

❖ PROBLEMS

☐ *Problem 1:*

Write assembly code that will toggle the on-board LED at a rate of 1 second.

☐ *Solution 1:*

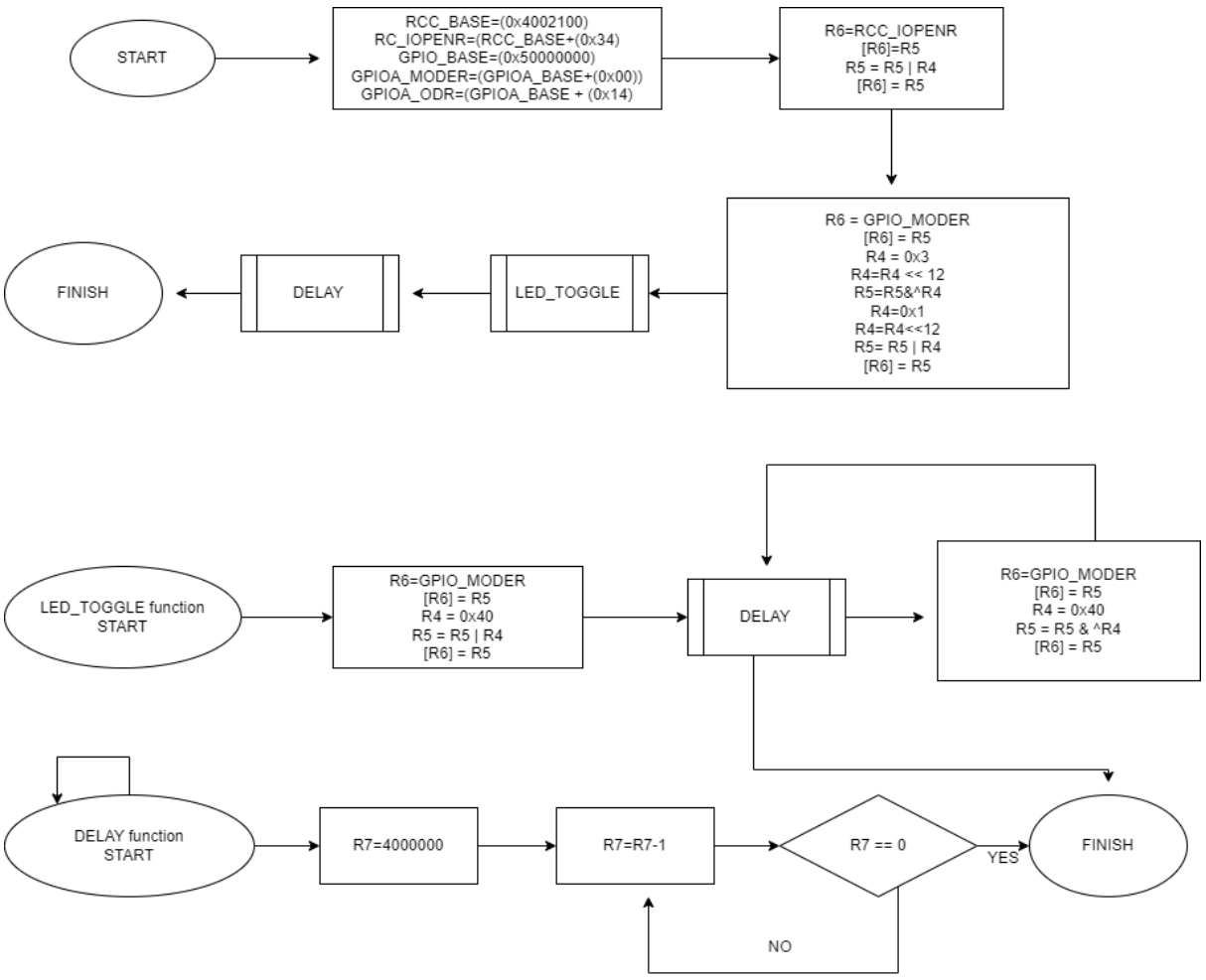


Figure1: *Problem1 Flowchart*

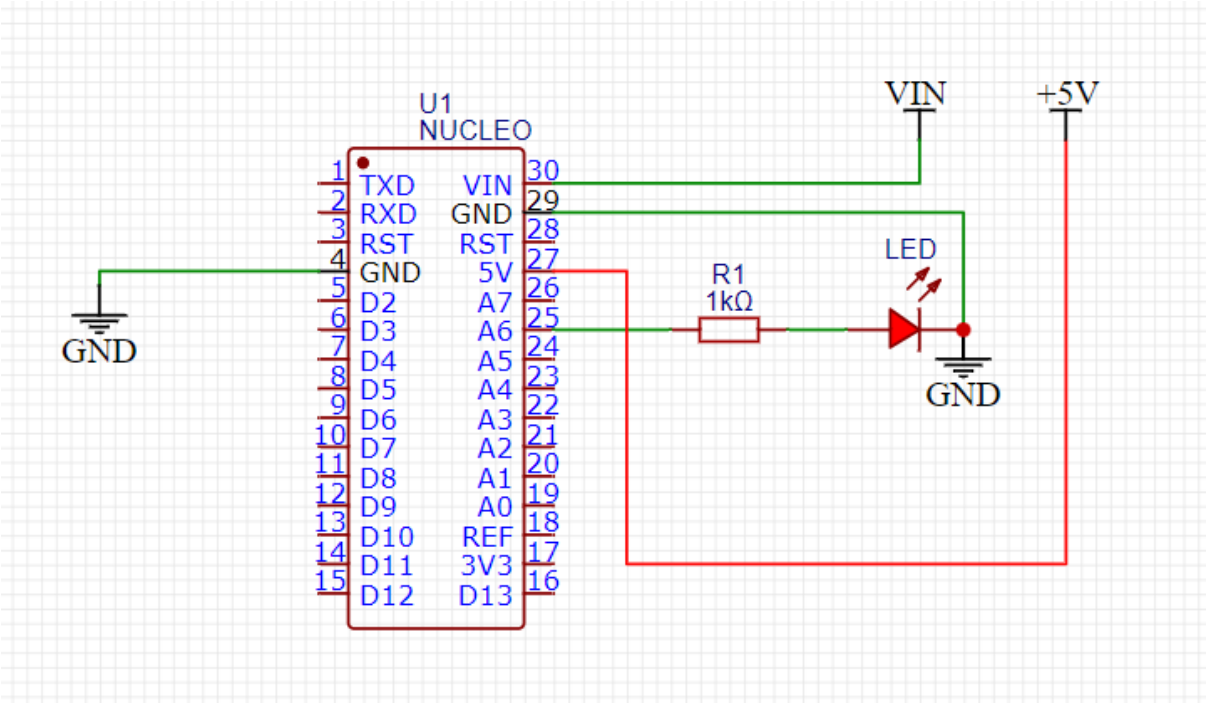


Figure2: *Problem1 Schematic*

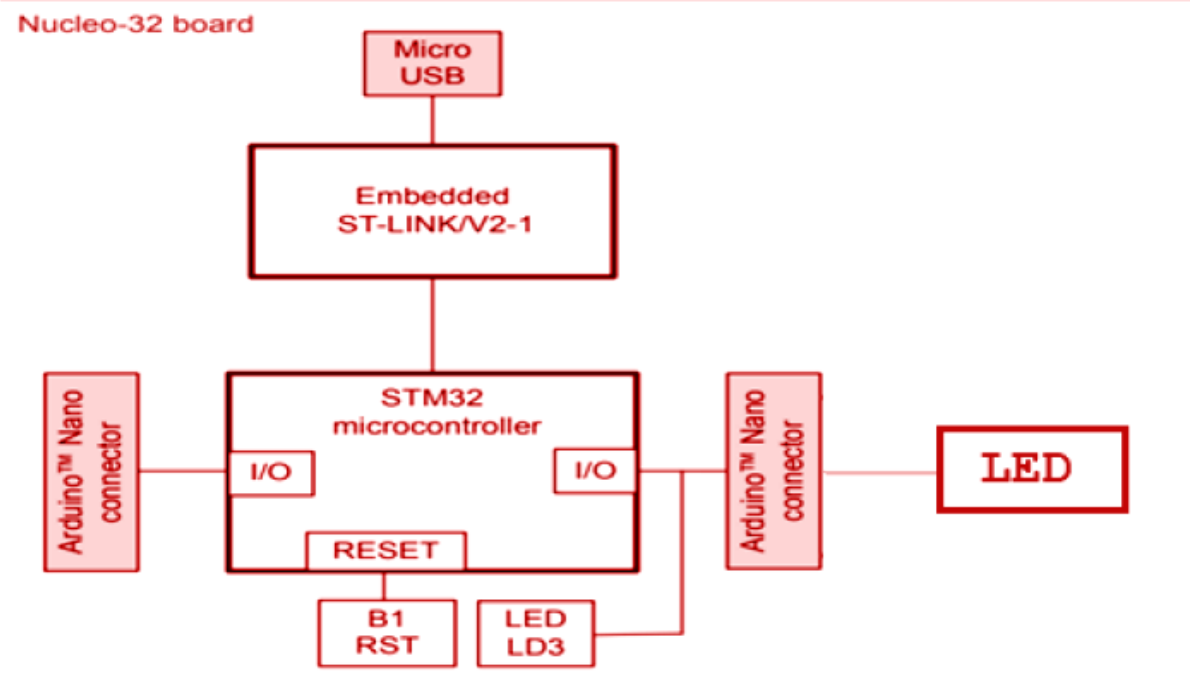


Figure3: *Block Diagram*

```
/*
 * asm.s
 *
* LAB2 - QUESTION 1
* Write assembly code that will toggle the on-board LED at a rate
of 1 second.
 *
 */


.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR
register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER
register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register
offset


/* vector table, +1 thumb mode */
.section .vectors
vector_table:
     .word _estack             /*     Stack pointer */
     .word Reset_Handler +1    /*     Reset handler */
     .word Default_Handler +1  /*       NMI handler */
     .word Default_Handler +1  /* HardFault handler */
```

```
      /* add rest of them here if needed */


/* reset handler */
.section .text
Reset_Handler:
      /* set stack pointer */
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
      /* call main */
      bl main
      /* trap if returned */
      b .


/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
      ldr r4, =_ebss
      movs r3, #0
```

```
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4

      LoopFillZerobss:
            cmp r2, r4
            bcc FillZerobss

      bx lr


/* default handler */
.section .text
Default_Handler:
      b Default_Handler


/* main function */
.section .text
main:

      ldr r6, =RCC_IOPENR
      ldr r5, [r6]
      /* movs expects imm8, so this should be fine */
      movs r4, 0x1 // We will activate pins A
      orrs r5, r5, r4 // Or, the status of other pin ports is
preserved, but the desired A pin must be 1
      str r5, [r6]
      /* setup PA6 for led 01 for bits 12-13 in MODER */
      ldr r6, =GPIOA_MODER
      ldr r5, [r6]
      movs r4,0x3
      lsls r4, r4, #12// When r4 = 0x3, bits 12 and 13 will be set
and
      r4 = 0x3000
      bics r5, r5, r4 // ~r4 = FFFF CFFF Set bits cleared
      r5 = 0xFFFFCFFF
      movs r4, 0x1
      lsls r4, r4, #12// When r4 = 0x1, 12 bits will be shifted to
the left and the 12th bit will be set and r4 = 0x1000
      orrs r5, r5, r4 // When r5 = 0xFFFF CFFF or is set r5 =
0x1000
      str r5,[r6]
```

```
led_toggle:
/* turn on led connected to PA6 in ODR */
     ldr r6, =GPIOA_ODR //r6 output register
     ldr r5, [r6] //reset value r5=0x0000 0000
     movs r4, 0x40 // A value of 1 has been sent to the PA6 pin so
that the LED will light (r4 = 0x0000 0040)
     orrs r5, r5, r4 // The appropriate value has been assigned to
the register that holds the signal that the LED will light
     str r5, [r6] // to be transferred to the output port

     ldr r7, =#4000000 // 1 million very fast 4 million

     bl delay
     /* turn off led connected to PA6 in ODR */
     ldr r6, =GPIOA_ODR //r6 output register
     ldr r5, [r6] //reset value r5=0x0000 0000
     movs r4, 0x40 // A value of 1 has been sent to the PA6

     mvns r4,r4 //r4=0xFFFF FFCF
     ands r5, r5, r4 // The register holding the signal that

     str r5, [r6] // the led will be transferred to the

     ldr r7,= #4000000 // 1 million very fast 4 million

     bl delay
     b led_toggle
delay:
     subs r7,r7,#1 // usual delay function
     bne delay
     bx lr
/* for(;;); */
     b .

     /* this should never get executed */
     nop
```

Table1: *Problem 1 Code*

❖ **Problem 2:**

☐ *Connect a button to the board, and turn on the onboard LED when the button is pressed. When the button is released, the LED should turn off.*

❖ **Solution 2:**

The button is connected to the circuit using the information that the vertical lines of the button are short circuit and the horizontal line is open circuit. With the information about whether the button was pressed on the PA4 pin, functions were written on the code that tests this situation, and according to this test result, it was decided whether the LED should be turned on or not, whether or not R5 is equal to 0x1.
Peripheral addresses from RM0444 are found as follows;

RCC_BASE, (0x40021000)
RCC_IOPENR, (RCC_BASE + (0x34))
GPIOA_BASE, (0x50000000)
GPIOA_MODER, (GPIOA_BASE + (0x00))
GPIOA_ODR, (GPIOA_BASE + (0x14))
GPIOA_IDR, (GPIOA_BASE + (0x10))

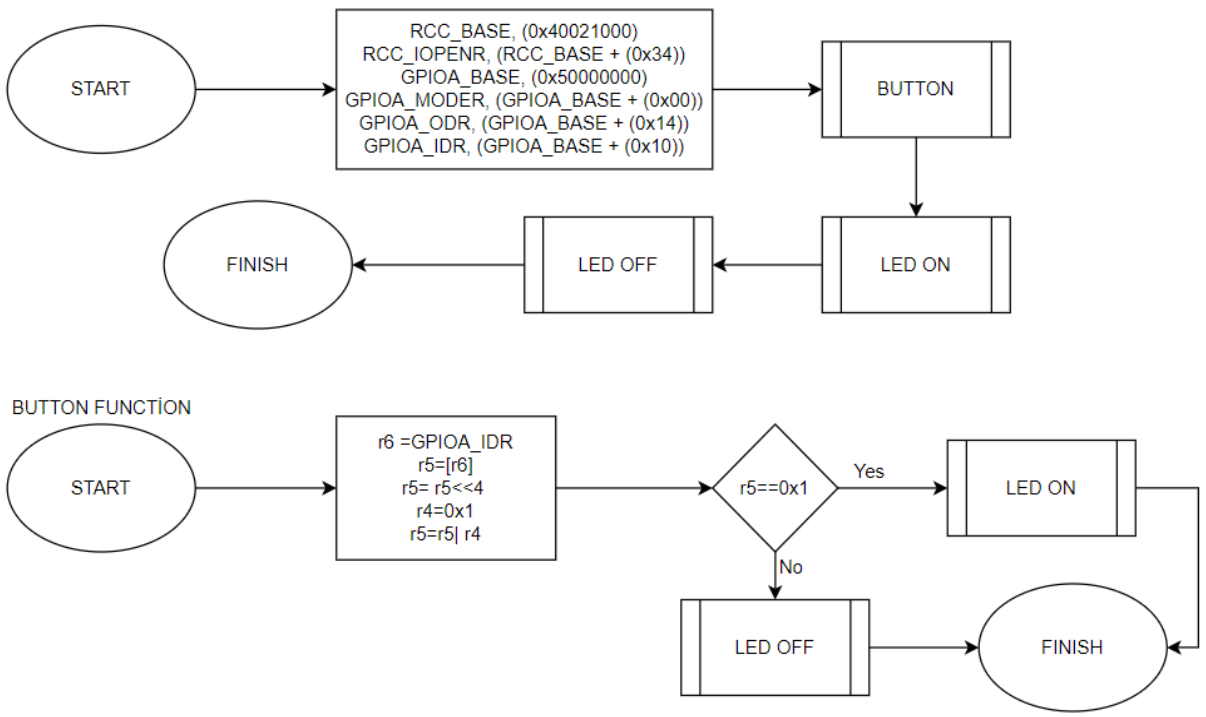


Figure 3. *Problem 2 Flowchart*

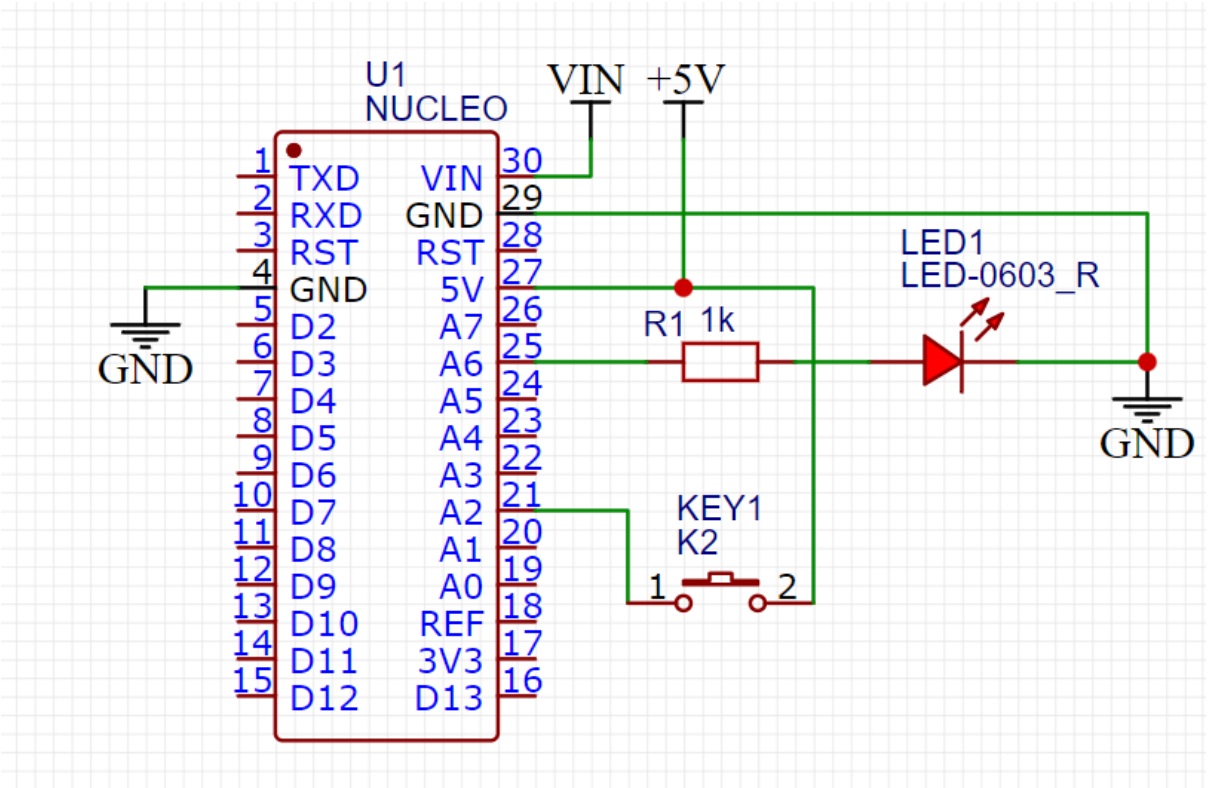


Figure 4. *Problem 2 Schematic*

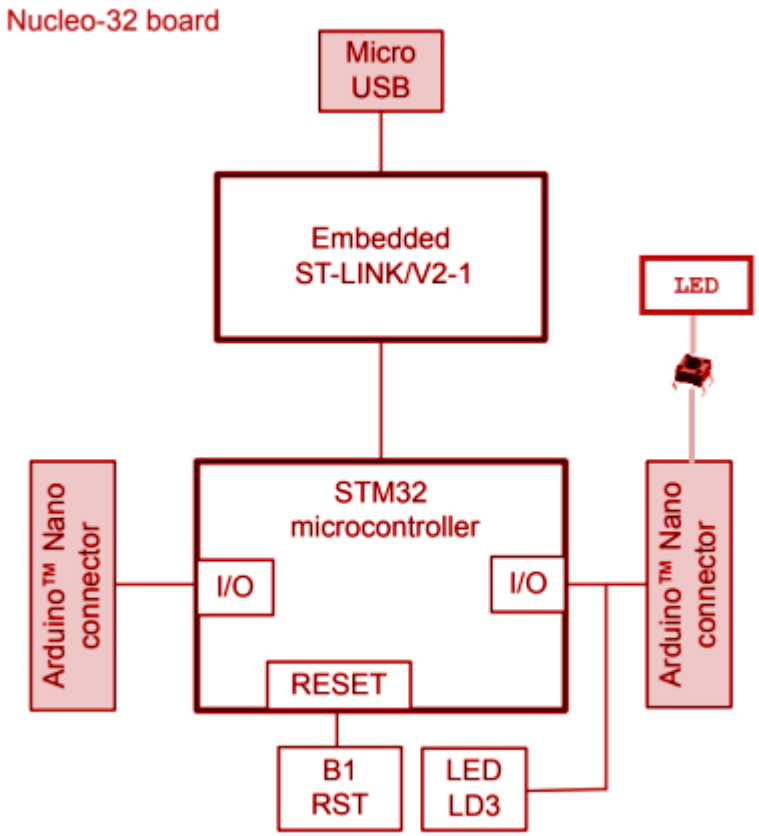


Figure 4. *Problem 2 Block Diagram*

```
/*
 * asm.s
 *
 * LAB2 - QUESTION 2
 *
 * description: Added the necessary stuff for turning on the green
LED on the
 *   G031K8 Nucleo board. Mostly for teaching.
 */


.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR
register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER
register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register
offset
.equ GPIOA_IDR,     (GPIOA_BASE + (0x10))

/* vector table, +1 thumb mode */
.section .vectors
vector_table:
     .word _estack             /*     Stack pointer */
     .word Reset_Handler +1    /*     Reset handler */
```

```
      .word Default_Handler +1  /*       NMI handler */
      .word Default_Handler +1  /* HardFault handler */
      /* add rest of them here if needed */


/* reset handler */
.section .text
Reset_Handler:
      /* set stack pointer */
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
      /* call main */
      bl main
      /* trap if returned */
      b .


/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
```

```
      ldr r4, =_ebss
      movs r3, #0
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4

      LoopFillZerobss:
            cmp r2, r4
            bcc FillZerobss

      bx lr
/* default handler */
.section .text
Default_Handler:
      b Default_Handler

/* main function */
.section .text
main:
      ldr r6, =RCC_IOPENR
      ldr r5, [r6]
      /* movs expects imm8, so this should be fine */
      movs r4, 0x1
      orrs r5, r5, r4
      str r5, [r6]
      /* setup PA6 for led 01 for bits 12-13 in MODER */
      ldr r6, =GPIOA_MODER
      ldr r5, [r6]
      /* cannot do with movs, so use pc relative */
      movs r4,0x3
      lsls r4, r4, #12// When r4 = 0x3, bits 12 and 13 will be set
and r4 = 0x3000
      bics r5, r5, r4 // ~r4 = FFFF CFFF Set bits cleared r5 =
0xFFFF CFFF
      movs r4, 0x1
      lsls r4, r4, #12// When r4 = 0x1, 12 bits will be shifted to
the left and the 12th bit will be set and r4 = 0x1000
      orrs r5, r5, r4 // When r5 = 0xFFFF CFFF or is set r5 =
0x1000
      str r5,[r6]
      ldr r6, =GPIOA_MODER
      ldr r5, [r6]
      /* for button */
```

```
      movs r4, 0x3
      lsls r4, r4, #8 // 8. and 9th bits are set
      mvns r4, r4 // 8. and the 9th bits will now be 0.
      ands r5, r5, r4
      str r5, [r6]
button:
//button connected to PA4 in IDR./
      ldr r6, =GPIOA_IDR
      ldr r5, [r6]
      lsrs r5, r5, #4 // r5 register carries the button value
      movs r4, #0x1
      ands r5, r5, r4
      cmp r5, #0x1
      bne led_off // if r5 isn't 1, go to led_off function
      beq led_on
led_on:
/* turn on led connected to PA6 in ODR */
      ldr r6, =GPIOA_ODR
      ldr r5, [r6] //reset value r5=0x0000 0000
      movs r4, 0x40 // A value of 1 has been sent to the PA6 pin so
that the LED lights up (r4 = 0x0000 0040)
      orrs r5, r5, r4 // The appropriate value has been assigned to
the register that holds the signal that the LED will light
      str r5, [r6]
      b button
led_off:
/* turn off led connected to PA6 in ODR */
      ldr r6, =GPIOA_ODR
      ldr r5, [r6] //reset value r5=0x0000 0000
      movs r4, #0x40 // A value of 1 has been sent to the PA6 pin
so that the LED lights up (r4 = 0x0000 0040)
      mvns r4,r4 //r4=0xFFFF FFCF
      ands r5, r5, r4 // The register holding the signal that the
led will turn off has been assigned an appropriate value
      str r5, [r6]
      b button
      /* this should never get executed */
      nop
```

Table: Problem2 Code

❖ **Problem 3:**

☐ *Connect 8 external LEDs to the board, and toggle all LEDs at the same time at arate of 1 second.*

❖ **Solution 3:**

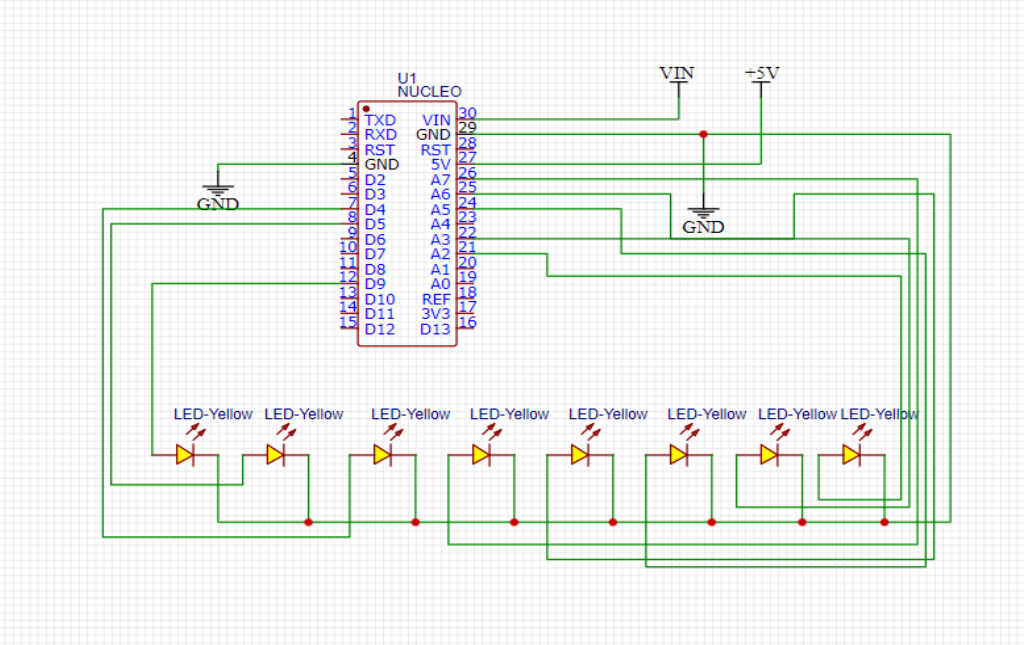


Figure: *Problem 3 Schmeatic*

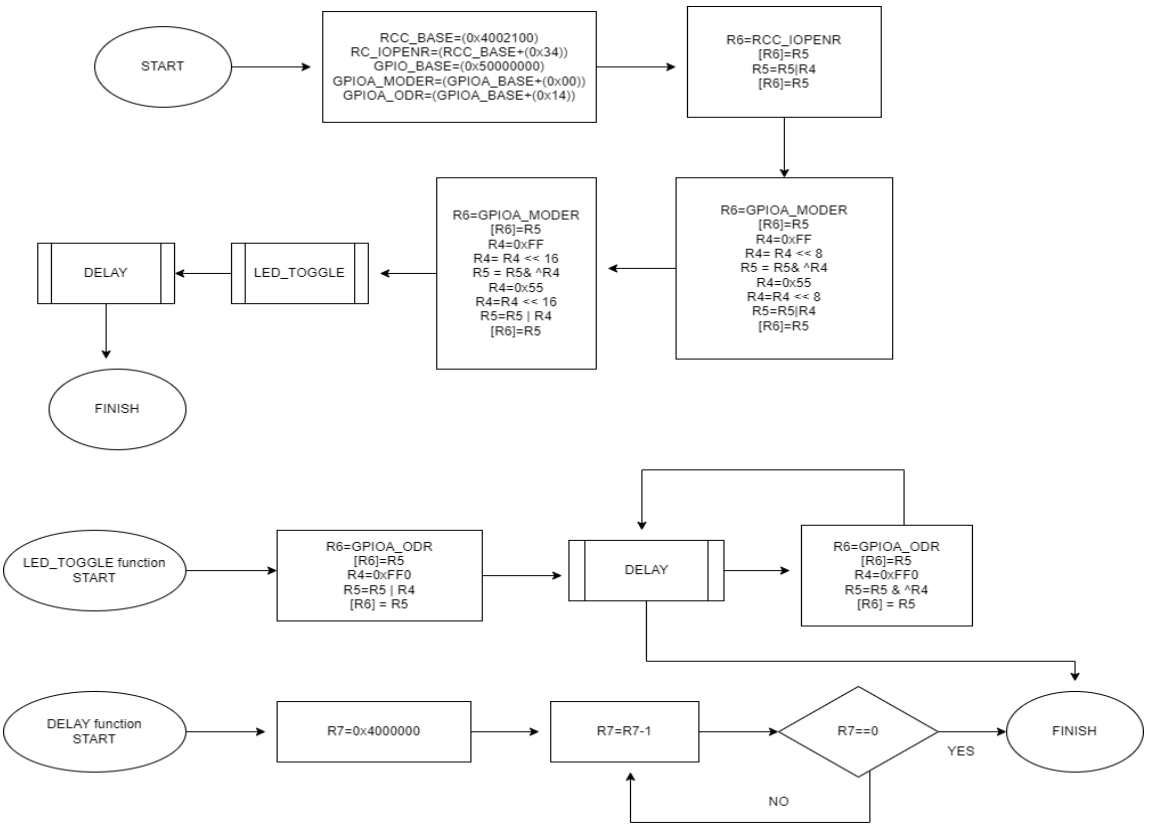


Figure: *Problem 3 Flowchart*

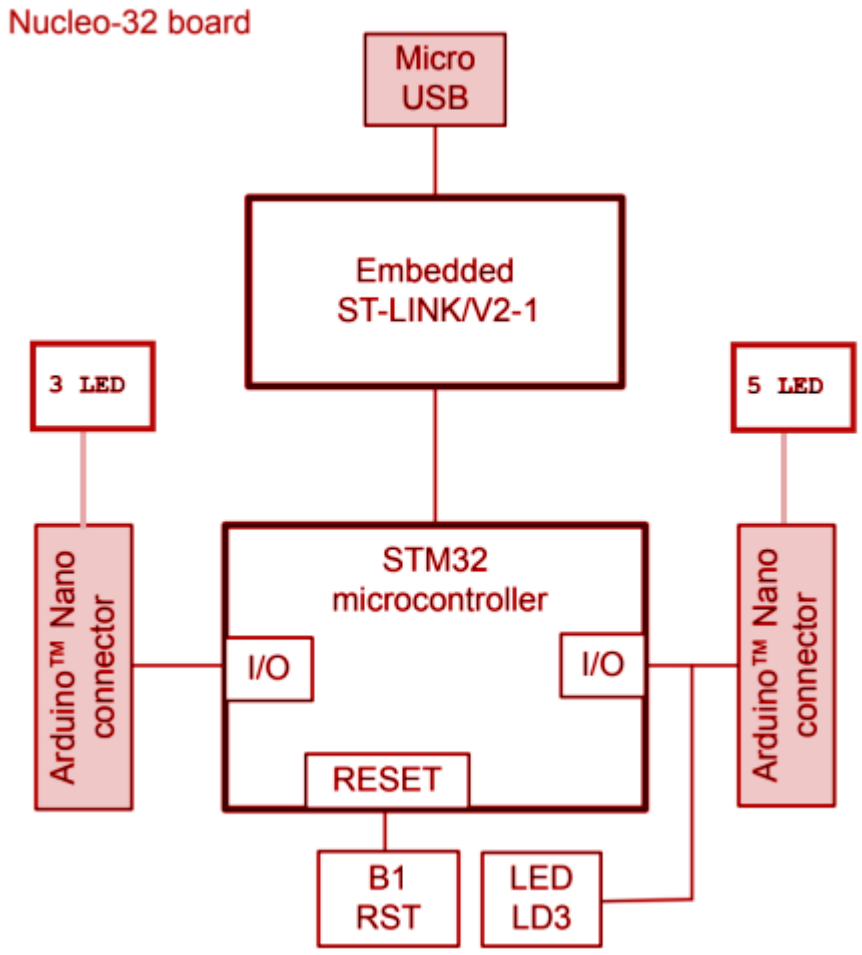


Figure: *Problem 3 Block Diagram*

```
/*
 * asm.s
 *
 *
 *
 * description: Added the necessary stuff for turning on the green
LED on the
 *   G031K8 Nucleo board. Mostly for teaching.
 */


.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb
```

```
/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR
register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER
register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register
offset


/* vector table, +1 thumb mode */
.section .vectors
vector_table:
      .word _estack             /*     Stack pointer */
      .word Reset_Handler +1    /*     Reset handler */
      .word Default_Handler +1  /*       NMI handler */
      .word Default_Handler +1  /* HardFault handler */
      /* add rest of them here if needed */


/* reset handler */
.section .text
Reset_Handler:
      /* set stack pointer */
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
```

```
      /* call main */
      bl main
      /* trap if returned */
      b .


/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
      ldr r4, =_ebss
      movs r3, #0
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4

      LoopFillZerobss:
            cmp r2, r4
            bcc FillZerobss

      bx lr


/* default handler */
```

```
.section .text
Default_Handler:
      b Default_Handler


/* main function */
.section .text
main:

      ldr r6, =RCC_IOPENR
      ldr r5, [r6]
      /* movs expects imm8, so this should be fine */
      movs r4, 0x1 // We will activate pins A
      orrs r5, r5, r4 // Or, the status of other pin ports is
preserved, but the desired A pin must be 1
      str r5, [r6]
      /* setup PA6 for led 01 for bits 12-13 in MODER */
      ldr r6, =GPIOA_MODER
      ldr r5, [r6]
      movs r4,0xFF
      lsls r4, r4, #8
      bics r5, r5, r4
      movs r4, 0x55 // 01010101 so that only one bit is 1 now
      lsls r4, r4, #8
      orrs r5, r5, r4
      str r5,[r6]
      //8 9 10 11 PİNS LED CONNECTED
      ldr r6, =GPIOA_MODER
      ldr r5, [r6]
      movs r4, 0xFF
      lsls r4, r4, #16
      bics r5, r5, r4
      movs r4, 0x55
      lsls r4, r4, #16
      orrs r5, r5, r4
      str r5, [r6]

loop_toggle:

 //turn on led connected
      ldr r6, =GPIOA_ODR
      ldr r5, [r6]
      ldr r4, =#0x0FF0
      movs r4,r4
      orrs r5, r5, r4
```

```
      str r5, [r6]
      ldr r7,= #4000000 // 1 million very fast 4 million
corresponds to an average of 1second
      bl delay

       //turn off led connected
      ldr r6, =GPIOA_ODR //r6 output register
      ldr r5, [r6]
      ldr r4, =#0x0FF0 // 4 5 6 7 8 9 10 11 PİNS
      bics r5, r5, r4 // The register holding the signal that the
led will turn off has been assigned an appropriate value
      str r5, [r6]
      ldr r7,= #4000000 // 1 million very fast 5 million
corresponds to an average of 1second
      bl delay
      b loop_toggle
delay:
      subs r7,r7,#1 // usual delay function
      bne delay
      bx lr
      /* for(;;); */
      b .
      /* this should never get executed */
      nop

      /* this should never get executed */
      nop
```

Table. *Problem 3 Code*

- **Problem 4:**
  - ☐ *Connect 8 LEDs and 1 button to the board, and implement a shift pattern.*

- **Solution 4:**

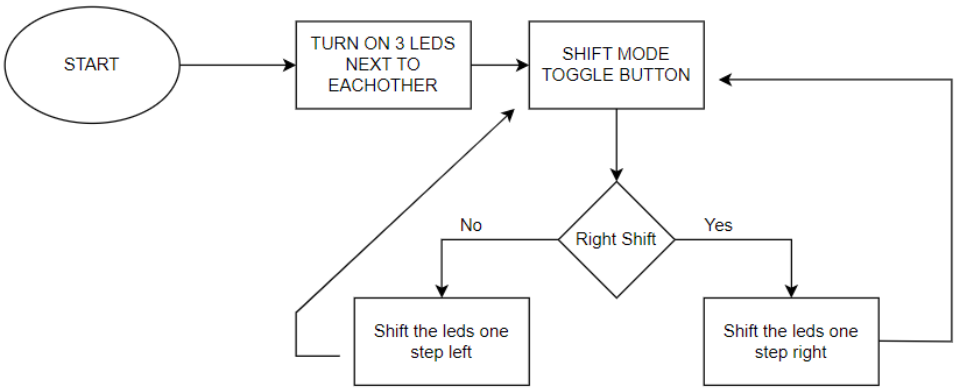


Figure . *Problem 4 Flowchart*

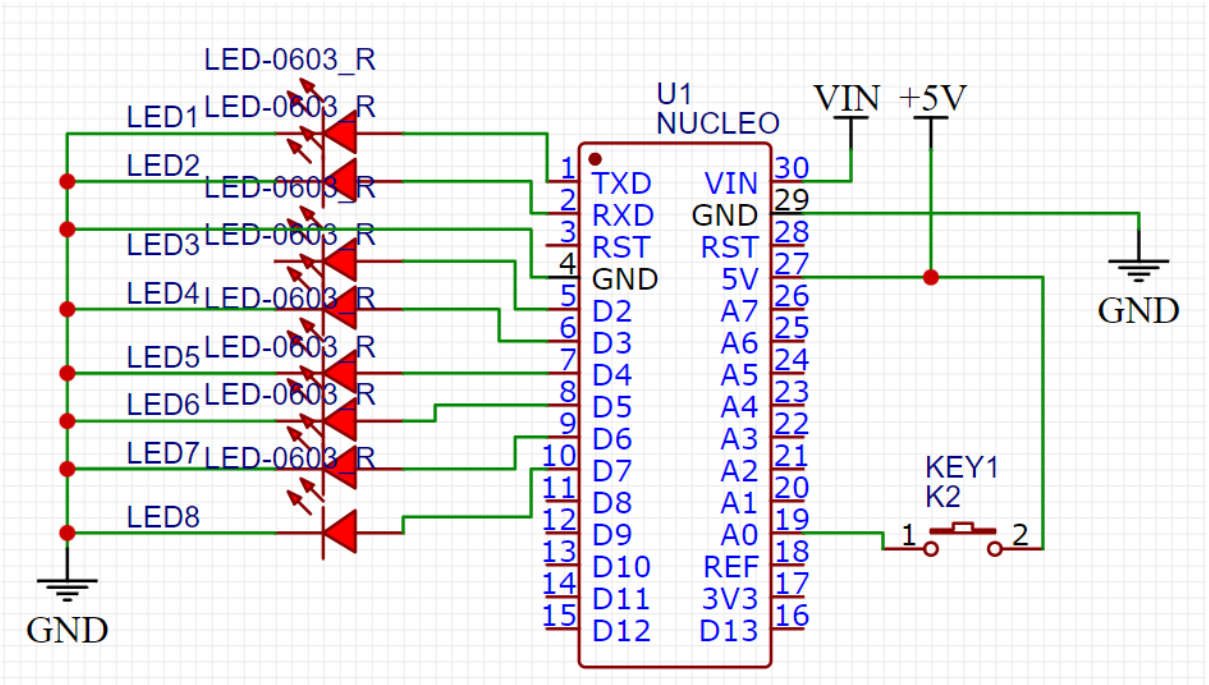


Figure: *Problem 4 Schmeatic*

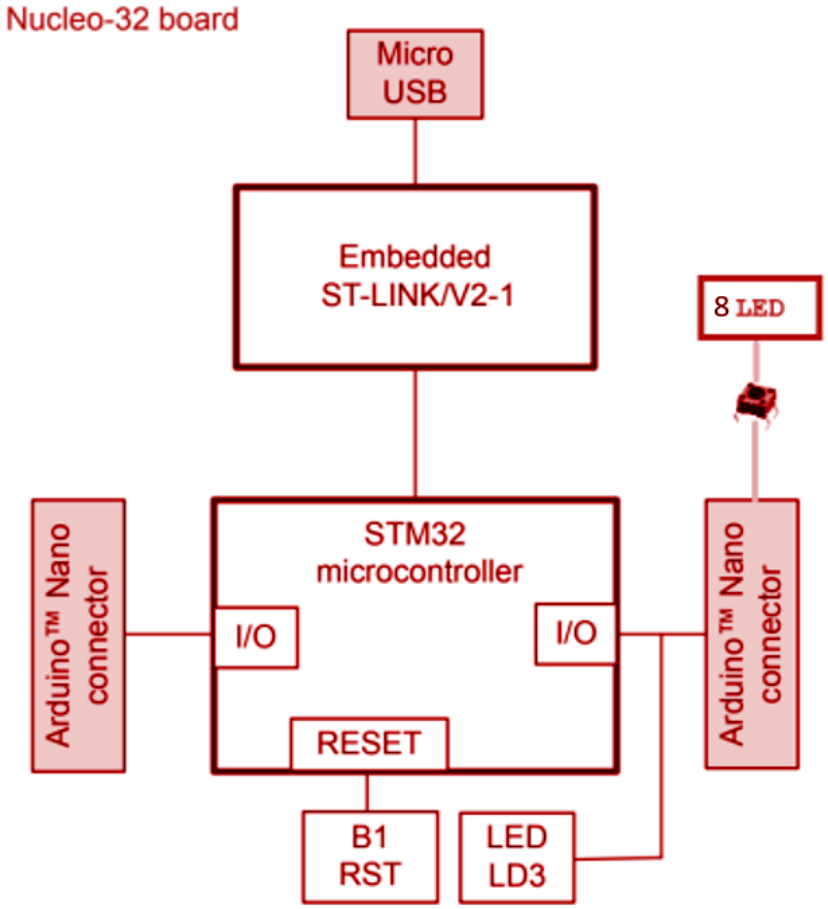


Figure: *Problem 4 Block Diagram*

```
/*
 * asm.s
 *
 */

.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR
register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER
register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register
offset


/* vector table, +1 thumb mode */
.section .vectors
vector_table:
     .word _estack             /*     Stack pointer */
     .word Reset_Handler +1    /*     Reset handler */
     .word Default_Handler +1  /*       NMI handler */
     .word Default_Handler +1  /* HardFault handler */
     /* add rest of them here if needed */


/* reset handler */
.section .text
```

```
Reset_Handler:
      /* set stack pointer */
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
      /* call main */
      bl main
      /* trap if returned */
      b .


/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
      ldr r4, =_ebss
      movs r3, #0
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4
```

```
      LoopFillZerobss:
            cmp r2, r4
            bcc FillZerobss

      bx lr


/* default handler */
.section .text
Default_Handler:
      b Default_Handler


/* main function */
.section .text
main:

      ;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 G
PIOB 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
```

Table. *Problem 4 Code*

- **Problem 5:**

☐ *Connect 8 LEDs to the board, and implement a “Knight Rider.*

- *Solution 5:*

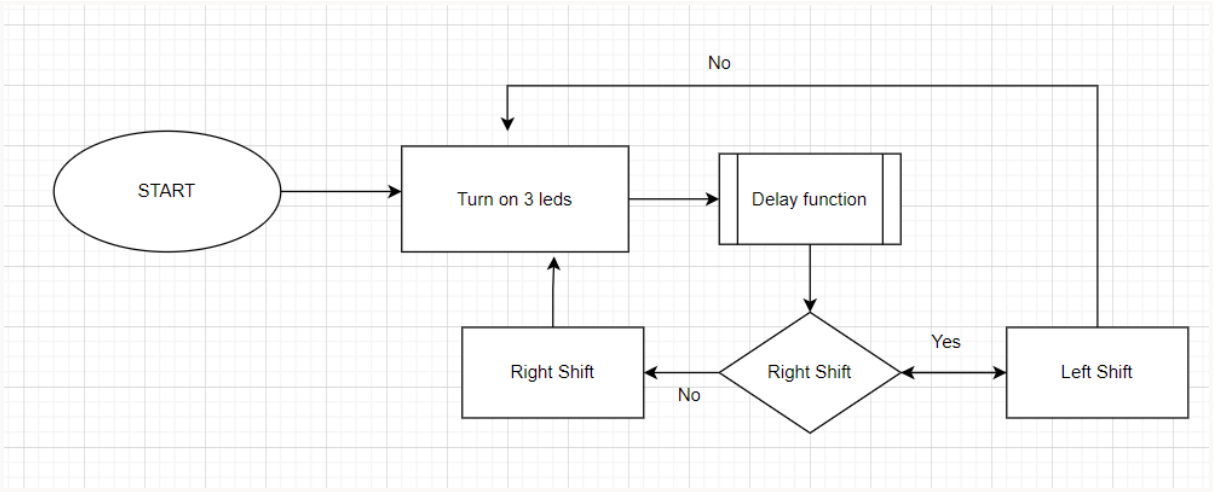


Figure. *Problem 5 Flowchart*

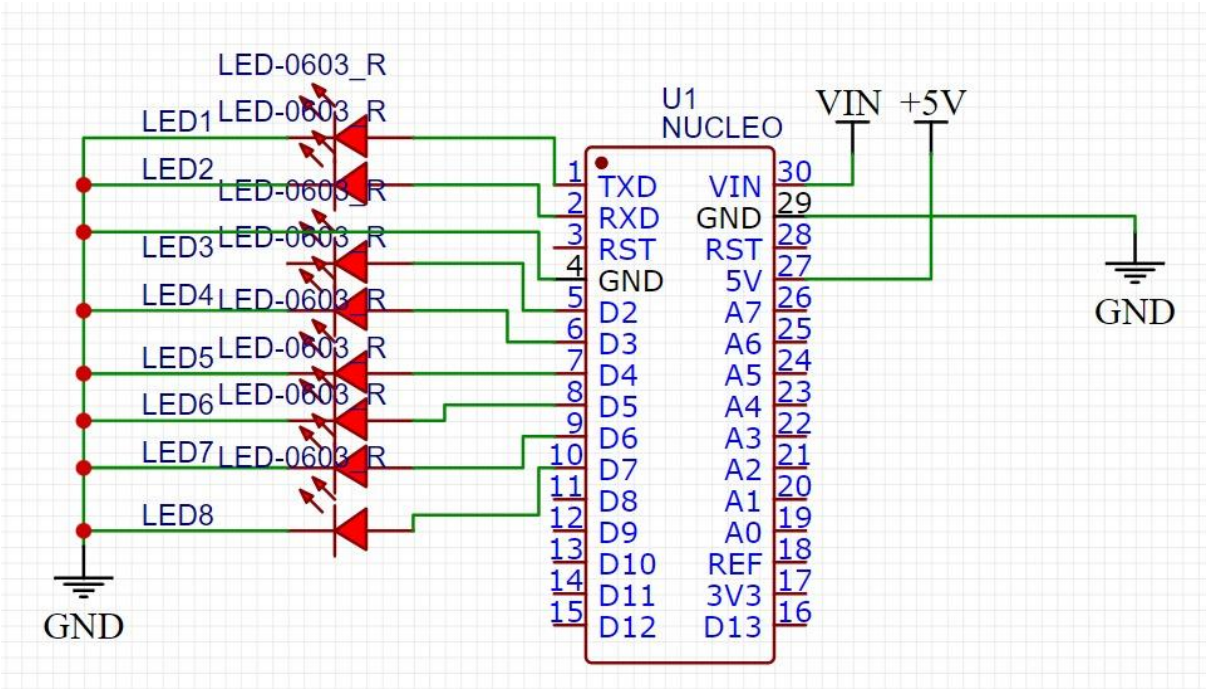


Figure. *Problem 5 Schematic*

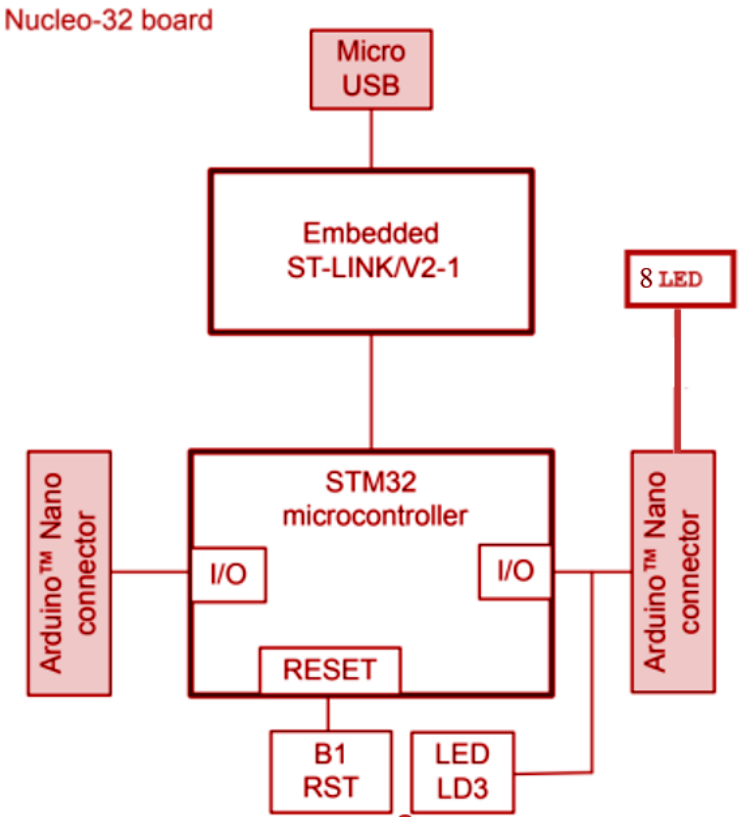


Figure. *Problem 5 Block Diagram*

```
/*
 * asm.s
 * description: Added the necessary stuff for turning on the green
LED on the
 *   G031K8 Nucleo board. Mostly for teaching.
 */
.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR
register offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER
register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register
offset

/* vector table, +1 thumb mode */
.section .vectors
vector_table:
      .word _estack             /*     Stack pointer */
      .word Reset_Handler +1    /*     Reset handler */
      .word Default_Handler +1  /*       NMI handler */
      .word Default_Handler +1  /* HardFault handler */
      /* add rest of them here if needed */
/* reset handler */
.section .text
Reset_Handler:
      /* set stack pointer */
```

```
      ldr r0, =_estack
      mov sp, r0

      /* initialize data and bss
       * not necessary for rom only code
       * */
      bl init_data
      /* call main */
      bl main
      /* trap if returned */
      b .

/* initialize data and bss sections */
.section .text
init_data:

      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit

      CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4

      LoopCopyDataInit:
            adds r4, r0, r3
            cmp r4, r1
            bcc CopyDataInit

      /* zero bss */
      ldr r2, =_sbss
      ldr r4, =_ebss
      movs r3, #0
      b LoopFillZerobss

      FillZerobss:
            str  r3, [r2]
            adds r2, r2, #4

      LoopFillZerobss:
            cmp r2, r4
```

```
          bcc FillZerobss

     bx lr

/* default handler */
.section .text
Default_Handler:
     b Default_Handler


/* main function */
.section .text
main:


     ;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 G
PIOB 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

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
```

Table. *Problem 5 Code*