New Paltz

STATE UNIVERSITY OF NEW YORK

EGC331-01 Microcontrollers System Design

Professor Michael Otis

Midterm

10/16/2020

Frank Seelmann

Introduction

This program is a 4 bit ALU on the NUCLEO-F446RE board. The arithmetic functions it can perform are addition and subtraction. The logical functions it can perform are bitwise INV, AND, OR, NAND, NOR, and XOR.

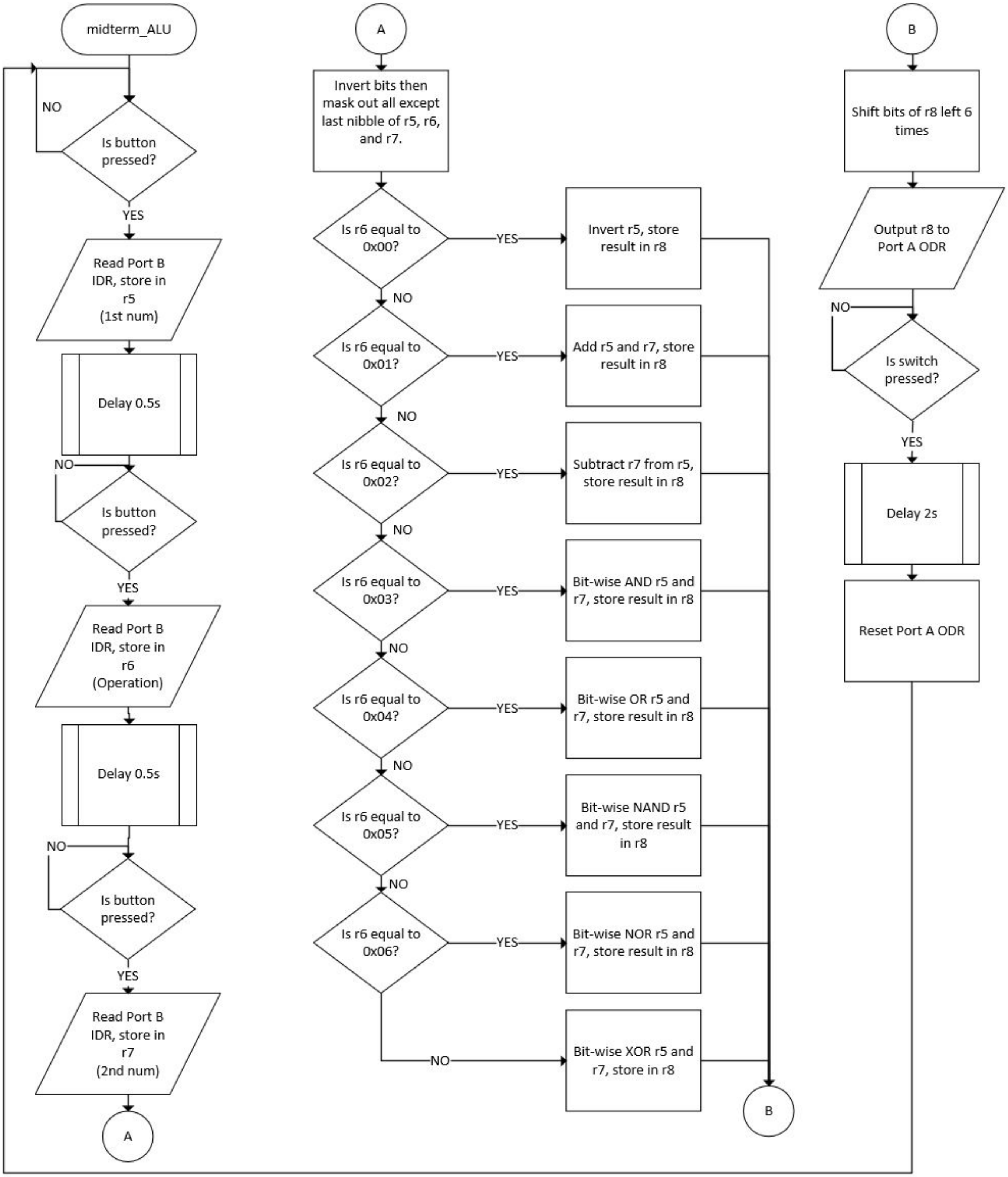
Video: https://www.youtube.com/watch?v=tDFA8UCyC84&t=3s
Github: https://github.com/Frank-Seelmann/Microntrollers-Midterm-4-Bit-ALU

Design

The ALU takes input in three stages, the first being the first four-bit number, the second being the selector for the arithmetic/logic operation being performed, and the final being the second four-bit number. All of these inputs are taken sequentially using the same 4 switch dip-switch, where a button on the NUCLEO board is used to confirm the input and move on to the next stage. Since internal pull-up resistors are used with the switches, the inputs are then inverted to make them easier to work with, then have everything except the least significant nibble masked out for all three. At this stage a conditional ladder is used to determine which operation to perform based on the operation input.

To avoid the program running through immediately after the first button press, each instance of the button being pressed is followed by a half second delay to give time to prevent bouncing and to let the user have to let go of the button.

Since Ports A2 and A3 are reserved, and A5 is an on-board LED, the final output is shifted left 6 times to align with Port A10 through Port A6 (MSB to LSB) to get a usable output.



midterm_ALU
NO
Is button pressed?
YES
Read Port B IDR, store in r5 (1st num)
Delay 0.5s
NO
Is button pressed?
YES
Read Port B IDR, store in r6 (Operation)
Delay 0.5s
NO
Is button pressed?
YES
Read Port B IDR, store in r7 (2nd num)
A
A
Invert bits then mask out all except last nibble of r5, r6, and r7.
Is r6 equal to 0x00?
YES
Invert r5, store result in r8
NO
Is r6 equal to 0x01?
YES
Add r5 and r7, store result in r8
NO
Is r6 equal to 0x02?
YES
Subtract r7 from r5, store result in r8
NO
Is r6 equal to 0x03?
YES
Bit-wise AND r5 and r7, store result in r8
NO
Is r6 equal to 0x04?
YES
Bit-wise OR r5 and r7, store result in r8
NO
Is r6 equal to 0x05?
YES
Bit-wise NAND r5 and r7, store result in r8
NO
Is r6 equal to 0x06?
YES
Bit-wise NOR r5 and r7, store result in r8
NO
Bit-wise XOR r5 and r7, store in r8
B
B
Shift bits of r8 left 6 times
Output r8 to Port A ODR
NO
Is switch pressed?
YES
Delay 2s
Reset Port A ODR

Figure 1 - Midterm_ALU Flowchart

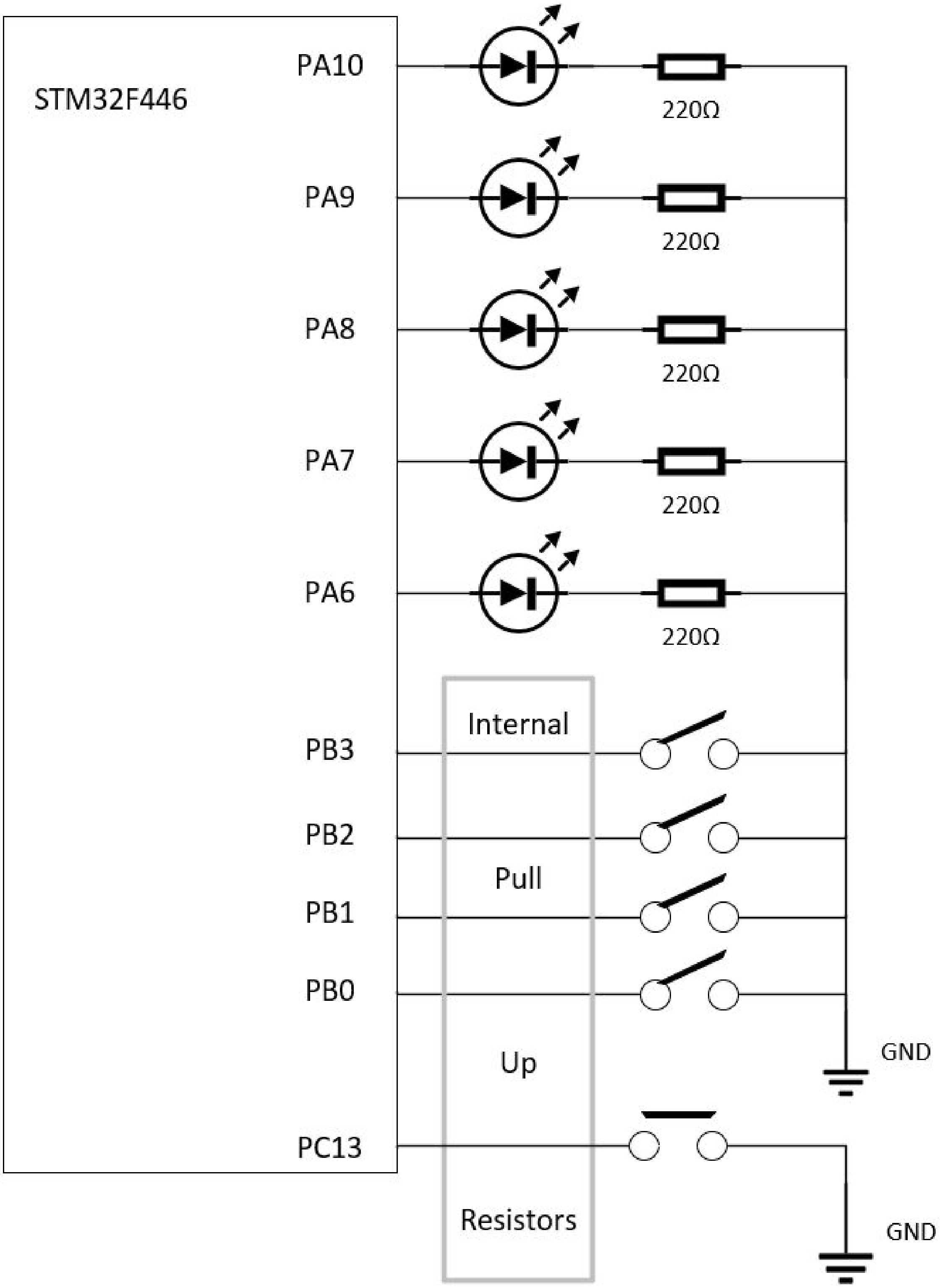


Figure 2 - Midterm_ALU Block Diagram

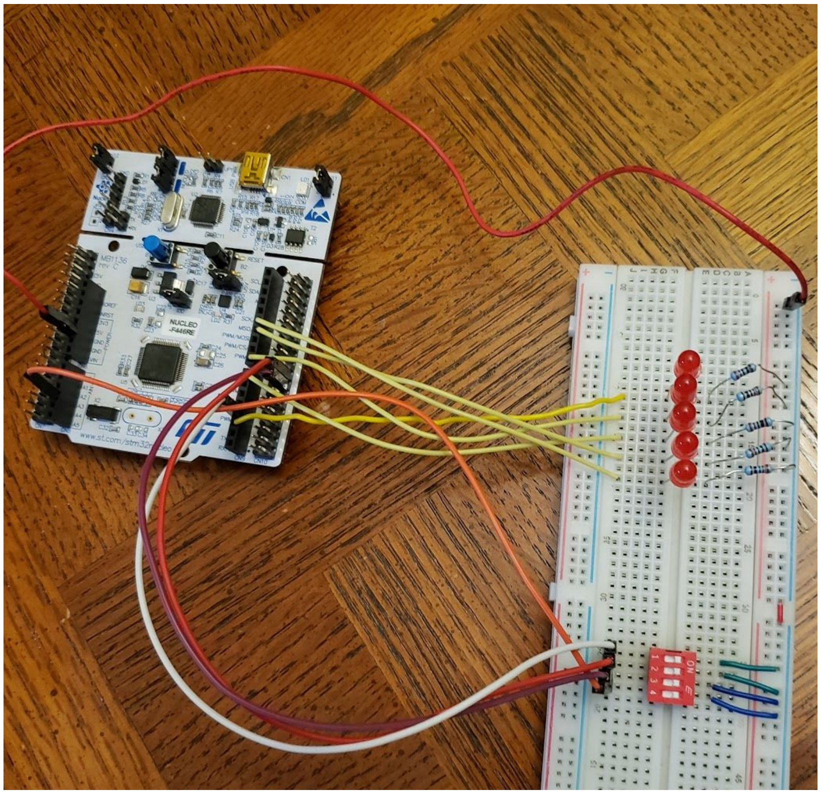


Figure 3 - Physical Setup

Results & Conclusion

The system works as anticipated, and even has more functions available to it than was required. This project allows students to use what they learned from the first 3 labs, such as how to handle input and output on the NUCLEO board and how to effect registers, specifically involving internal pull-up resistors. The particular methodology used in this solution also made use of the delay function used in previous labs as well as the on-board push button.

Appendix

```
; midterm_ALU.s
;
; Frank Seelmann              EGC331-01 - Microcontrollers System Design
; Prof Michael Otis           10/14/20
;
; The ALU takes input in three stages, the first being the first four-bit number,
; the second being the selector for the arithmetic/logic operation being performed,
; and the final being the second four-bit number. All of these inputs are taken
; sequentially using the same 4 switch dip-switch, where a button on the NUCLEO
; board is used to confirm the input and move on to the next stage. Since internal
; pull-up resistors are used with the switches, the inputs are then inverted to make
; them easier to work with, then have everything except the least significant nibble
; masked out for all three. At this stage a conditional ladder is used to determine
; which operation to perform based on the operation input.

        EXPORT  __Vectors
        EXPORT  Reset_Handler
        AREA    vectors, CODE, READONLY

__Vectors   DCD     0x10010000  ; 0x20008000    ; Top of Stack
        DCD     Reset_Handler               ; Reset Handler

RCC_AHB1ENR equ 0x40023830
GPIOA_MODER equ 0x40020000
GPIOB_MODER     equ     0x40020400
GPIOA_ODR  equ 0x40020014
GPIOB_ODR  equ 0x40020400
GPIOA_IDR    equ     0x40020010
GPIOB_IDR    equ     0x40020410
GPIOC_IDR    equ 0x40020810
GPIOB_PUPDR equ 0x4002040C
GPIOC_MODER equ 0x40020800

        AREA    PROG, CODE, READONLY
Reset_Handler
        ldr     r1, =RCC_AHB1ENR    ; enable GPIOA, GPIOB, and GPIOC clocks
        ldr     r2, [r1]
        orr     r2, #7
        str     r2, [r1]

        ldr     r1, =GPIOA_MODER    ; set A pins to output mode
        ldr     r2, =0x55555555
        str    r2, [r1]

        ldr    r1, =GPIOB_MODER    ; set B pins to input mode
        ldr    r2, =0x00000000
        str    r2, [r1]

        ldr            r1, =GPIOB_PUPDR  ; setup pull-up resistors for B pins
        ldr            r2, =0x55555555
        str            r2, [r1]

        ldr            r1, =GPIOC_MODER ; set C pins to input mode
        ldr            r2, =0x00000000
        str            r2, [r1]


; --------------------------Read first number-----------------------;
L1              ldr             r1, =0x00002000
again_n1        ldr             r2, =GPIOC_IDR        ; keep waiting until blue button is pressed
                ldr             r3, [r2]
                tst             r1, r3
                bne             again_n1

                ldr             r1, =GPIOB_IDR
                ldr             r5, [r1]              ; read first number
                rsb             r5, r5, #0xFFFFFFFF ; invert bits

                mov             r0, #500              ; wait - gives time to let go of button
                bl              delay


; --------------------------Read operation number-----------------------;
                ldr             r1, =0x00002000
again_op        ldr             r2, =GPIOC_IDR        ; keep waiting until blue button is pressed
                ldr             r3, [r2]
                tst             r1, r3
                bne             again_op

                ldr             r1,       =GPIOB_IDR
                ldr             r6, [r1]              ; read operator
                rsb             r6, r6, #0xFFFFFFFF  ; invert bits

                mov             r0, #500              ; wait - gives time to let go of button
                bl              delay
```

```
; -------------------------Read second number----------------------;
                ldr             r1, =0x00002000
again_n2        ldr             r2, =GPIOC_IDR          ; keep waiting until blue button is pressed
                ldr             r3, [r2]
                tst             r1, r3
                bne             again_n2

                ldr             r1,     =GPIOB_IDR
                ldr             r7, [r1]                ; read second number
                rsb             r7, r7, #0xFFFFFFFF     ; invert bits


; -------------------------Choose-operation ladder----------------------;
                and             r6,     r6,     #0x0000000F ; mask out bits we dont care about
                and             r5,     r5,     #0x0000000F ; mask out bits we dont care about
                and             r7,     r7,     #0x0000000F ; mask out bits we dont care about

                teq             r6, #0x00000000         ; 0000  =       INV
                beq             inversion
                teq             r6, #0x00000001         ; 0001  =       ADD
                beq             addition
                teq             r6, #0x00000002         ; 0010  =       SUB
                beq             subtraction
                teq             r6, #0x00000003         ; 0011  =       AND
                beq             anding
                teq             r6, #0x00000004         ; 0100  =       OR
                beq             orring
                teq             r6, #0x00000005         ; 0101  =       NAND
                beq             nanding
                teq             r6, #0x00000006         ; 0110  =       NOR
                beq             norring
                eor             r8, r5, r7              ; anything else = XOR
                b               output


; -------------------------Perform operation----------------------;
inversion       rsb             r8, r5, #0x0000000F
                b               output
addition        add             r8, r5, r7
                b               output
subtraction     sub             r8, r5, r7
                b               output
anding          and             r8, r5, r7
```

```
                b               output
orring          orr             r8, r5, r7
                b               output
nanding         and             r8, r5, r7
                rsb             r8, r8, #0x0000000F
                b               output
norring         orr             r8, r5, r7
                rsb             r8, r8, #0x0000000F
                b               output

; -------------------------Output & Reset----------------------;
output          ldr             r1, =GPIOA_ODR
                lsl             r8, #6                  ; shift to output to ports A6-A10
                str             r8, [r1]                ; output

                mov             r0, #500                ; wait - gives time to let go of button
                bl              delay

                ldr             r1, =0x00002000
again_reset     ldr             r2, =GPIOC_IDR          ; keep waiting until blue button is pressed
                ldr             r3, [r2]
                tst             r1, r3
                bne             again_reset

                mov             r0, #2000               ; wait - gives time to let go of button
                bl              delay

                ldr             r1, =GPIOA_ODR
                ldr             r8, =0x00000000
                str             r8, [r1]

                b               L1                      ; loop back up to the top

; delay milliseconds in R0
delay   ldr     r1, =5325
DL1     subs    r1, r1, #1
        bne     DL1
        subs    r0, r0, #1
        bne     delay
        bx      lr

        end
```