ECE 375 LAB 4

Introduction to AVR Development Tools

Lab Time: Wednesday 12-2

Bradley Martin

INTRODUCTION

The purpose of this lab is to start programming in assembly and combining all the skills learned in the previous labs to run an LCD display. A skeleton code was provided to help organize our code and give ideas for how to start coding. Using the AVR simulator and referencing the AVR instruction set manual for instructions we might not have encountered in class is pertinent to get our code to work.

PROGRAM OVERVIEW

This LCD program gives instructions on what to print if buttons 1,2 or 7 are pressed. If button 1 is pressed then it will print my name on the first line and “hello, world” on the second. If button 2 is pressed then it will reverse the order with “hello, world” on the first line and my name on the second. Pressing button 7 will clear the LCD Display. The program achieves this with the standard Initialization and Main routine along with three other routines called Button1, Button2, and Button3

INITIALIZATION ROUTINE

In the Initialization routine we first initialize the stack pointer. We then call a function provided in the LCDDriver.asm file called LCDInit that initializes the necessary registers for the LCD to proper work. Last PORT D is initialized to take input from the buttons.

MAIN ROUTINE

The Main routine executes a loop that checks if any of the three buttons have been pushed. The buttons are active low so the input is compared to a binary number to check if any of the 8 bits have hit 0. It first checks to see if button 1 has been pushed. If it has then it calls the corresponding function for button 1. If not then it branches to the next loop which checks if button 2 has been pushed. If button 2 has not been pushed it moves on to check if button 7 has been pushed and then loops back to the top of main.

BUTTON1

The Button1 routine first initializes the Z pointer with the addresses that string1 are stored in and the Y register with memory on the first line of the LCD screen and a temp register that hold how many bytes are left. The routine then enters a loop that takes the contents of the addresses the Z pointer is looking at and stores it in one past the Y register and decrements how many bytes are left. Once all bytes for the first line are finished, we then load the addresses of the second string into the Z registers and have memory in the Y register for the second line of the LCD. The second loop runs the same as the first decrementing through until finished. The LCD display then gets a call to LCDWrtite function which updates the screen.

BUTTON2

The Button2 routine first initializes the Z pointer with the addresses that string2 are stored in and the Y register with memory on the first line of the LCD screen and a temp register that hold how many bytes are left. The routine then enters a loop that takes the contents of the addresses the Z pointer is looking at and stores it in one past the Y register and decrements how many bytes are left. Once all bytes for the first line are finshed, we then load the addresses of the first string into the Z registers and have memory in the Y register for the second line of the LCD.

The second loop runs the same as the first decrementing through until finished. The LCD display then gets a call to LCDWrtite function which updates the screen.

BUTTON3

The Button3 routine calls a Function given by LCCDDriver.asm that clears the LCD display.

ADDITIONAL QUESTIONS

1) In this lab, you were required to move data between two memory types: program memory and data memory. Explain the intended uses and key differences of these two memory types

Program memory store our code and instructions, things that our program will need to run. Data memory will store the values that the program runs or changes.

2) You also learned how to make function calls. Explain how making a function call works (including its connection to the stack), and explain why a RET instruction must be used to return from a function.

To call a function we used the command "rcall" which takes the return address and pushes it on the stack. In order to get out of the function, "ret" must be called because it takes that return address pushed to the stack and pops it off so that we can get back to where we were.

3) To help you understand why the stack pointer is important, comment out the stack pointer initialization at the beginning of your program, and then try running the program on your mega128 board and also in the simulator. What behavior do you observe when the stack pointer is never initialized? In detail, explain what happens (or no longer happens) and why it happens.

With no stack pointer there is nothing to push or pop our addresses to/from when we do our calls, jumps, or ret's. So, our program will stop when it is trying to use an "rjmp" call in the initialization of the interrupt vector because there are nowhere to push or pop addresses to.

CONCLUSION

In this lab we were required to make an assembly program that would print to an LCD display depending on what button was pushed. This lab used all the skills we have learned in previous labs to get the code to work. I had to use the AVR simulator to see if my loops were running correctly or to see what certain registers stored values were. I also used the AVR instruction manual to look for different commands that we have not used in previous labs or classes.

SOURCE CODE

```
;***********************************************************
;*
;*      Bradley_Martin_Lab4_sourcecode
;*
;*      This program with take input from buttons 1,2 and 7 and depending on
;*      which button is pushed will display a preset string on the LCD display.
;*
;*
;***********************************************************
;*
;*       Author: Bradley Martin
```

```
;*         Date: 10/28/2020
;*
;***********************************************************

.include "m128def.inc"                  ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multipurpose register is
                                                ; required for LCD Driver
.def    input = r23
.def    temp = r25

.equ    B1 = 0
.equ    B2 = 1
.equ    B3 = 7

;***********************************************************
;*      Start of Code Segment
;***********************************************************
.cseg                                                   ; Beginning of code segment

;***********************************************************
;*      Interrupt Vectors
;***********************************************************
.org    $0000                                   ; Beginning of IVs
                rjmp INIT                               ; Reset interrupt

.org    $0046                                   ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:                                                   ; The initialization routine
                ; Initialize Stack Pointer
                ldi             mpr, low(RAMEND)
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr

                ; Initialize LCD Display
                rcall LCDInit

                ;Initialize Port D for input
                ldi             mpr, $00
                out             DDRD, mpr
                ldi             mpr, $FF
                out             PORTD, mpr

                ; NOTE that there is no RET or RJMP from INIT, this
                ; is because the next instruction executed is the
                ; first instruction of the main program

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:                                                   ; The Main program
                ; Display the strings on the LCD Display
                in              input, PIND                     ; Get input from the buttons
                mov             mpr, input                      ; Send the input to mpr
                cpi             mpr, 0b11111110         ; Check to see if button 1 has been pressed
                brne    NEXT                            ; Branch to NEXT if button 1 is not pressed
                rcall   Button1                         ; Run the Button1 function if it has been pressed
                rjmp    MAIN                            ; Loop back to MAIN after Button1 has run

NEXT:   cpi             mpr, 0b11111101         ; Check to see if button 2 has been pressed
                brne    NEXT2                           ; Branch to NEXT2 if button 2 is not pressed
                rcall   Button2                         ; Run Button2 function if it has been pressed
```

```
                rjmp    MAIN                            ; Loop back to MAIN after Button2 has run

NEXT2:  cpi             mpr, 0b01111111         ; Check to see if button 7 has been pressed
                brne    MAIN                            ; Branch to Main if button 3 is not pressed
                rcall   Button3                         ; Run Button3 if it has been pressed
                rjmp    MAIN                    ; jump back to main and create an infinite
                                                                ; while loop.  Generally, every main
program is an
                                                                ; infinite while loop, never let the
main program
                                                                ; just run off

;***********************************************************
;*      Functions and Subroutines
;***********************************************************

;-----------------------------------------------------------
; Func: Button1
; Desc: Loads string 1 to the top line of the LCD display and
;               loads string 2 to the bottom line of the LCD display.
;-----------------------------------------------------------
Button1:

                ldi ZL,         low(STRING1_BEG<<1)             ; Initialize the Z pointer to the
addresses of String 1
                ldi ZH,         high(STRING1_BEG<<1)
                ldi YL,         low(LCDLn1Addr)                 ; Initialize the Y pointer to hold our
string on line 1
                ldi YH,         high(LCDLn1Addr)
                ldi temp,       16                              ; Counter register that will decrement
through our bytes

L1:             lpm mpr, Z+                                     ; Load the strings from program memory
                st Y+, mpr                                      ; store the contents of the string to
the Y register shift by one
                dec     temp                                    ; Decrease the counter for the loop to
tell how many bits are left
                brne L1                                         ; If L1 has not decremented the counter
to 0 then keep looping

                ldi ZL,         low(STRING2_BEG<<1)             ; Initialize the Z pointer to the
addresses of string 2
                ldi ZH,         high(STRING2_BEG<<1)
                ldi YL,         low(LCDLn2Addr)                 ; Initialize the Y pointer to hold our
string on line 2
                ldi YH,         high(LCDLn2Addr);
                ldi temp,       16                              ; Reset the counter back to 16 from
previous loop

L2:             lpm mpr, Z+
                st Y+, mpr
                dec temp
                brne L2

                rcall LCDWrite                                  ; Update the LCD display

                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: Template function header
; Desc: Cut and paste this and fill in the info at the
;               beginning of your functions
;-----------------------------------------------------------
Button2:

                ldi ZL,         low(STRING2_BEG<<1)             ; Initialize the Z pointer to the
addresses of string 2
                ldi ZH,         high(STRING2_BEG<<1)
```

```
                ldi YL,         low(LCDLn1Addr)                 ; Initialize the Y pointer to hold our
string on line 1
                ldi YH,         high(LCDLn1Addr)
                ldi temp,       16                              ; Counter register that will decrement
through our bytes

upper:  lpm mpr, Z+                                             ; Load the String from program memory
                st Y+, mpr                                      ; Store the contents of the String into
the Y register shift by one
                dec temp                                        ; Decrease the counter
                brne L2                                         ; If upper is not finished not continue
to loop through

                ldi ZL,         low(STRING1_BEG<<1)             ; Initialize the Z pointer to the
addresses of string 1
                ldi ZH,         high(STRING1_BEG<<1)
                ldi YL,         low(LCDLn2Addr)                 ; Initialize the Y pointer to hold our
string on line 2
                ldi YH,         high(LCDLn2Addr)
                ldi temp,       16                              ; Reset the counter back to 16 from the
previous loop

lower:  lpm mpr, Z+
                st Y+, mpr
                dec temp
                brne L2

                rcall LCDWrite                                  ; Update the LCD Dispaly

                ret                                             ; End a function with RET
;-----------------------------------------------------------
; Func: Template function header
; Desc: Cut and paste this and fill in the info at the
;               beginning of your functions
;-----------------------------------------------------------
Button3:

                rcall LCDClr                    ; Clear LCD display

                ret                                             ; End a function with RET

;***********************************************************
;*      Stored Program Data
;***********************************************************

;-----------------------------------------------------------
; An example of storing a string. Note the labels before and
; after the .DB directive; these can help to access the data
;-----------------------------------------------------------
STRING1_BEG:
.DB             "Bradley Martin  "              ; Declaring data in ProgMem
STRING1_END:

STRING2_BEG:
.DB             "Hello, World    "
STRING2_END:

;***********************************************************
;*      Additional Program Includes
;***********************************************************
.include "LCDDriver.asm"        ; Include the LCD Driver
```