ECE 375 LAB 4

Introduction to AVR Development Tools

Lab Time: Tuesday 7-9

Aaron Rito

Introduction

Lab 4 uses Atmel Studio and assembly to control the LCD screen on the development board. The LCD will display a name and “hello world”, in a revolving marquee fashion.

Program Overview

The program uses a pre-exisiting driver to run the LCD. Strings are stored in program memory and then displayed to the LCD. There are two control functions, one to rotate the characters in the LCD, another to delay the time before updating the screen.

Initialization Routine

The program initializes the at128 chip , and then runs a routine to initialize the stack pointer and direct access registers. Then a short loop is executed to place the strings stored in program memory, to reserved space in data memory, where the LCD functions can access it directly.

Main Routine

The main routine simply runs the functions in order: Update LCD, roll the data in the LCD, delay, and then repeat the loop

LCDWrite Function

The LCDWrite function is part of the given LCD driver. It reads from the data memory starting at address 0x0100, and ending 32 addresses later at 0x011F, and pushes the data to the display. Each address holds one character for the LCD in order.

CYCLE_MEM Function (challenge questions)

The CYCLE_MEM function uses the Y pointer to load indirectly the data from last address of line one of the LCD data (0x010F) and stores the value in a temporary register. Then it loads the preceding value into another temporary register using the load indirect Y with pre-decrement. Then, it stores the data in the next address by incrementing the Y pointer and the indirectly storing the value. This continues in a loop backwards through the addresses until the Y pointer has reached the beginning of the data for the LCD line 1 (0x0100). When it reaches the start, it places the original value from the last position, into the first position. This allows the data to “scroll” across the LCD like a stock ticker or rotating marquee.

MY_Delay function (challenge questions)

The MY_DELAY function uses 3 registers to create a time delay loop. By loading a value into a register and decrementing it, then checking if it is 0, then decrementing another register, and checking if that is 0, and so on, we can “delay” the micro-controller by keeping it busy for the amount of time we are looking to delay. For 16Mhz the period of a clock cycle is $T = \frac{1}{16\text{Mhz}} = 62.5\,ns$. To get delay for .25 seconds,

$N_{cycles} = \frac{.25}{62.5\text{n}} = 4{,}000{,}000\, cycles$. To code the delay, we factor in the decrement cycles, and the 2 cycles it takes to compare and jump. It will take 3 registers to hold the values. With two holding 255 $255 * 2 = 65{,}025$ and then $R_3 = \frac{4{,}000{,}000}{65{,}025} = 65{,}087$. The delay can be easily adjusted by changing the values in the registers, and will need to consider the number of loops used to create the function.

Study 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:

The program memory is non-volatile, and is 16 bits wide. It's used to store constant data needed to be preserved in case of power failure. The data memory is volatile and only 8 bits wide. When transferring data from program memory to data memory, a high/low bit must be used when addressing the data memory registers.

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

When a function is called the address that the program was at before the function executed is save don the stack. The ret instruction pops the address back into the PC so the program can pick up where it left off.

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.

If the stack pointer is not initialized the program will not be able to store addresses for returning from function calls, as described in the last question.

Difficulties

This lab is straight forward.

Conclusion

Program memory should be used to store constants to preserve the data in case of power cycle. The data can be moved to sram during initialization for future manipulation. AVR delays are very efficient and accurate. This lab is straight forward.

Source Code

```
;***********************************************************

;*      Lab 4 ECE375

;*      LCD usage

;***********************************************************
```

```
;*       Author: Aaron Rito
;*         Date: 2/2/17
;***********************************************************
.include "m128def.inc"               ; Include definition file
;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                            ; Multipurpose register is required for LCD Driver
.equ RAM_START = 0x0100                      ; Marking the start of sram, where the LCD will read
from
;***********************************************************
;*      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
                ldi             mpr, low(RAMEND)       ; initialize Stack Pointer
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr
                rcall LCDInit  ; Initialize LCD Display
                ; Move strings from Program Memory to Data Memory
                ldi ZL, low(STRING_BEG<<1) ;setting Z pointer to the strings
                ldi ZH, high(STRING_BEG<<1)
```

```
                ldi YL, low(RAM_START)  ;setting Y pointer to the start of SRAM
                ldi YH, high(RAM_START)
                ldi XL, low(STRING_END<<1) ; setting X pointer to the end of the strings
                ldi XH, high(STRING_END<<1)
                LOOP:
                        LPM R17, Z+
                        ST Y+, R17
                        cp r31, r27    ;check if the high pointer registers are equal
                        BREQ LOW_CHECK
                        rjmp LOOP
                LOW_CHECK:
                        cp r30, r26 ;check if the low pointer registers are equal
                        BRNE LOOP ; if not return to loop to next space in mem
                ldi YL, low(RAM_START) ; set the y pointer back to the start of the sram
                ldi YH, high(RAM_START)


;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
                rcall LCDWrite ;push the data to the LCD
                call CYCLE_MEM ;rotate the letters in a moving marquee
                rcall MY_DELAY  ;a simple delay
                rjmp    MAIN    ;repeat
;***********************************************************
;*      Functions and Subroutines
;***********************************************************
; A standard delay function, this one is approx 1/3 second.
MY_DELAY:
        ldi  r18, 20
    ldi  r19, 8
    ldi  r20, 150
```

```
L1: dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
        ret
;This function cycles through the memory spaces used by the LCD in a moving marquee fashion
;It takes the last value of the line word, and stores in a temp register until all the other
;values have been moved one spaces to the right. The temp value then goes into the first position.
CYCLE_MEM:ldi YL, 0x0f  ;set the Y pointer to the last value of line 1
        ldi YH, 0x01
        LD r17, Y                    ; save the last value of line 1 into temp
        CYC:    LD r18, -Y     ; get the preceding value
                INC r28                ; return to next space in mem
                ST Y, r18              ; place preceding value in next space
                DEC r28                ; return to prev space
                cpi r28, 0x00  ; check if end of line
                BRNE CYC               ; if not, repeat
                ST Y, r17              ; place the saved last value from the line to the first position
        ldi YL, 0x1F           ; same process for line 2, starting at the last location
        ldi YH, 0x01
        LD r17, Y
        CYC1: LD r18, -Y
                INC r28
                ST Y, r18
                DEC r28
                cpi r28, 0x10
                BRNE CYC1
                ST Y, r17
        ret
```

```
;*************************************************************
;*      Stored Program Data
;*************************************************************
STRING_BEG:
.DB             "Aaron Rito      ", "Hello World!    "              ; Declaring data in ProgMem
STRING_END:
;*************************************************************
;*      Additional Program Includes
;*************************************************************
.include "LCDDriver.asm"              ; Include the LCD Driver
```