ECE 375 LAB 4

Data Manipulation & the LCD

Lab Time: Tuesday 5-7

Drake Vidkjer

INTRODUCTION

The purpose of this lab is to understand the basics of data manipulation and transfer using AVR assembly as well as learning how to initialize a basic program including the stack pointer, registers, peripherals and other components of the Atmega128 board. During the lab we use the X, Y, and Z pointers to perform indirect addressing to point to constant data that is declared and stored in the program memory, and transferred to the data memory. We also implement the LCD driver to allow us to display to the LCD on the Atmega128 development board.

PROGRAM OVERVIEW

The program, whose source is below, reads an array of characters from program memory and places the information into data memory. It then displays the characters to the LCD screen using the LCD driver.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the program to execute correctly. First it initializes the stack pointer based on the RAMEND value found in the Atmega128 include file. Next it calls the subroutine that initializes the LCD. It then loads the address of the start and the end of the array of data into the Z and X registers. Finally, iterates from the start of the array to the end and stores each element at the address located in the Z register.

MAIN ROUTINE

The Main routine is an infinite loop that calls the LCDWrite subroutine which writes the data that was placed in the addresses pointed to by the Z register to the LCD. After this the main function loops back to the beginning of main.

STORED DATA

Stored Data is the initial location of the character string in the instruction memory. It has two address labels at the beginning and end which help when loading the characters into data memory.

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 is flash based while data memory is based on static RAM. The most important difference is that program memory is not reset with power loss while data memory is. In addition, program memory and flash memory ore on different buses and program memory is 16 bits wide while data memory is only 8 bits.

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.

A function call is very similar to a jump, with the exception that with a function call the program returns to where it was called from once it reaches a RET instruction. This is why RET is important, as it tells the program when to return from the function call. To return, the program stores the address it is at when the call is made on the stack and then performs the function. When the function is done, the program retrieves that address and goes to that address again.

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

The program no longer runs correctly and does not return from function calls. This is because the stack pointer is required in order to be able to store the address of the call and return to that location. Without it, the program no longer properly runs.

DIFFICULTIES

Some difficulties were in creating the loop to load the data from program memory. I was getting errors about using invalid registers and with the TA's help realized that I had not initialized some of the registers I was trying to use. Once I resolved this, creating the program was not too difficult.

CONCLUSION

In this lab we wrote an assembly language program that outputted text to the LCD that had been stored in program memory. It loaded the data from program into data memory and implemented the LCD driver to then write this information to the screen.

SOURCE CODE

```
;***********************************************************
;*
;*        ECE 375 Lab 04
;*
;*        Enter the description of the program here
;*
;*        This is the skeleton file for Lab 4 of ECE 375
;*
;***********************************************************
;*
;*         Author: Drake Vidkjer
;*           Date: 01/31/17
;*
;***********************************************************

.include "m128def.inc"                                  ; Include definition file

;***********************************************************
;*        Internal Register Definitions and Constants
;***********************************************************
.def      mpr = r16                              ; Multipurpose register is
                                                 ; required for LCD Driver
;***********************************************************
;*        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 r17, HIGH(RAMEND)
                    out SPH, r17
                    ldi r17, LOW(RAMEND)
                    out SPL, r17

                    ; Initialize LCD Display
                    rcall    LCDInit

                    ; Move strings from Program Memory to Data Memory
                    ldi ZH, high(STRING_BEG<<1)
                    ldi ZL, low(STRING_BEG<<1)
                    ldi XH, high(STRING_END<<1)
                    ldi XL, low(STRING END<<1)
                    ldi YL, $00
                    ldi YH, $01

LOOP:
                    lpm mpr, Z+
                    st Y+, mpr
                    cp ZL, XL
                    brne LOOP

                    ; 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
                    rcall    LCDWrite

                    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

;***********************************************************
;*        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
;-----------------------------------------------------------
STRING_BEG:
.DB                 "Drake Vidkjer    Hello World"                ; Declaring data in ProgMem
STRING END:

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