ECE 375 LAB 4

Introduction to AVR Development Tools

Lab Time: Mon 12-2

James Stallkamp

INTRODUCTION

This lab is focused on driving a LCD panel to display two strings. The strings are stored in program memory and must be moved into data memory in order to be written to the LCD display. This lab will cover how to properly initialize an assembly program and how to move data from program memory to data memory. A skeleton code as well as a driver for the LCD panel have been provided.

PROGRAM OVERVIEW

This LCD program writes two lines of characters to a LCD panel. Two strings are defined in program memory using the .db instruction. This program consists of a main and an init, once the initialization has been completed LCDWrite is invoked and if the strings are loaded into data memory correctly then the strings will be written to the LCD panel. The initialization portion of the program is when the stack is initialized and when the strings are moved from program memory into data memory. The MAIN routine consists of a call to LCDwrite and a jump back to main in order to run an infinite loop.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of important registers and declarations. First the Stack Pointer is initialized. Next the Z pointer is pointed at the beginning of the first string, and the X pointer is pointed to the destination addresses in data memory. Then characters are loaded one at a time into data memory using the lpm and st instructions. Characters are loaded one at a time in a loop until the address of the z pointer matches the address of the last character in the string.

MAIN ROUTINE

The main routine consists of a call to LCDwrite and a jump back to main in order to create an infinite loop. The infinite loop is for the challenge.

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.

Data memory is 8 bit and is used to store variables and registers in a way so that it is easily available. Program memory is 16 bit and is where the actual program declaration is located, it is where any declarations of strings are stored.

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 puts the address of the instruction to be jumped to onto the top of the stack so that it becomes the next instruction to be executed. RET is invoked in order to push the address of where to return to onto the top of stack.

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 LCD panel requires the stack to be initialized in order to initialize itself. Without the stack initialized the LCD panel doesn't initialize and nothing is written.

CONCLUSION

This lab is focused on driving a LCD panel. Writing the code to drive the LCD panel was actually quite simple and consisted of just two function calls. The main difficulty in this lab was moving data from program memory to data memory. This is not as easy as it seems because program memory is 16 bits wide and the data memory is 8 bits wide. This means that the Z pointer had to be used and was pointed to the high and low bytes of a string at the same time. Being able to move data from program memory to data memory is essential because often starting strings or inputs can be hard coded in program memory and must be moved into data memory in order to be used.

SOURCE CODE

```
;***********************************************************
;*
;*     Lab4
;*
;*     LCD manipulation lab
;*
;*     This is the skeleton file for Lab 4 of ECE 375
;*
;***********************************************************
;*
;*      Author: James Stallkamp
;*        Date: 1/3/19
;*
;***********************************************************

.include "m128def.inc"                  ; Include definition file

;***********************************************************
;*     Internal Register Definitions and Constants
;***********************************************************
.def   mpr = r16                        ; Multipurpose register is
                                                ; required for LCD Driver
.equ top = $0100
.equ bottom = $0110

;***********************************************************
;*     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
              ; Top Line Load
              ldi zl, low(STRING_BEG<<1)
              ldi zh, high(STRING_BEG<<1)

              ldi yl, low(top)
              ldi yh, high(top)


topLineLoop:
              lpm mpr, z+
              st y+, mpr
              cpi zl, low(STRING_END<<1)
              brne topLineLoop

              cpi zh, high(STRING_END<<1)
              brne topLineLoop

botLineLoad:
              ldi zl, low(STRING2_BEG<<1)
              ldi zh, high(STRING2_BEG<<1)

              ldi xl, low(bottom)
              ldi xh, high(bottom)
botLineLoop:
              lpm mpr, z+
              st x+, mpr

              cpi zl, low(STRING2_END<<1)
              brne botLineLoop

              cpi zh, high(STRING2_END<<1)
              brne botLineLoop

              ; 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           "Hello World!"             ; Declaring data in ProgMem
STRING_END:
STRING2_BEG:
.DB           "James Stallkamp "
STRING2_END:

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