ECE 375 LAB 4

Data Manipulation & the LCD Display

Lab Time: Wednesday 5-7

Zheng Zheng 932101499 Abhishek Raol 932065306

INTRODUCTION

The purpose of this lab is to display a string using the LCD display on the Mega128 board. To do this requires manipulation of data and the moving of data from program memory to data memory, as well as the use of basic assembly commands. This lab will introduce students to AVR assembly.

PROGRAM OVERVIEW

This program starts off by moving strings from the program memory to the data memory. This sets the string at register Z, and the address of the line that it displays the string at register Y. Then it loads Z into a multi purpose register at R16 and then stores it at Y. Then it loops through the string and outputs it.

INITIALIZATION ROUTINE

The initialization routine initializes the stack pointer to always point to the highest address in data memory. Then it initializes Z and Y to be the string and the address of the output line. It always initializes the LCD display and R16 as a multi purpose register. There is a while loop that loops through the string and outputs them to the given line.

MAIN ROUTINE

The main routine simply outputs the contents of the lines by doing rcall LCDWrite.

ADDITIONAL QUESTIONS

1) In this lab, you were asked to manipulate data by moving it around in memory. In fact, the AVR architecture has two different memories, a program memory and data memory. Briefly explain the differences and purposes of these memories within your write-up.

The program memory is used for saving the code being executed. The data memory consists of Rx space, I/O memory/ and internal SRAM. Program memory is non volatile while data memory is volatile.

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

Function calls work by using the command rcall followed by the name of the function. An example is given as: rcall LCDWrite. A RET instruction must be called to return from a function since there is no brackets or parenthesis to signal end of a function. The memory is important to a function call since it must remember the location it was at before it entered the function.

3) To help you understand the importance of initializing the stack pointer correctly, comment out the stack pointer initialization in the beginning of your code and try running the program (on your mega128 board and also in the simulator). Give some comments about the behavior you observe when the stack pointer is never initialized.

Without initializing the stack pointer, nothing will show up on the LCD display. This is because without the stack pointer, there is no way to call functions since after you call a function the program will have no idea where to go next since it does not remember its previous location.

CONCLUSION

The purpose of this lab is to display two strings to an LCD display. The learning outcomes of this lab was to be introduced to AVR assembly programming and to learn string and data manipulation. One of the main activity of this lab was to move data from program memory to data memory. This lab will help make students familiar with the X, Y and Z registers.

SOURCE CODE

Provide a copy of the source code. Here you should use a mono-spaced font and can go down to 8-pt in order to make it fit. Sometimes the conversion from standard ASCII to a word document may mess up the formatting. Make sure to reformate the code so it looks nice and is readable.

```
;
; AssemblerApplication1.asm
;
; Created: 2/3/2016 4:03:52 AM
; Author : Zheng Zheng
;


.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 R16, LOW(RAMEND) ; Low Byte of End SRAM Address
               OUT SPL, R16 ; Write byte to SPL
               LDI R16, HIGH(RAMEND) ; High Byte of End SRAM Address
               OUT SPH, R16 ; Write byte to SPH

               ; Initialize LCD Display
               rcall LCDInit ;Call LCD Init Subroutine

               ; Move strings from Program Memory to Data Memory
                      LDI ZL, low(STRING_BEG<<1)    ;Moving strings according to psuedocode in
manual
                      LDI ZH, high(STRING_BEG<<1)
                      LDI YL, low(LCDLn1Addr)
                      LDI YH, high(LCDLn1Addr)

               ;While loop to put Z into mpr and store in Y
               loop:
                      LPM     mpr, Z+
```

```
                        st Y+, mpr
                        CPI ZL, low(STRING_END<<1)
                        brne loop
                        cpi ZH, high(STRING_END<<1)


                        ;Repeat step of moving string from program memory to data memory for line
2
                        LDI ZL, low(HELLOSTRING_BEG<<1)
                        LDI ZH, high(HELLOSTRING_END<<1)
                        LDI YL, low(LCDLn2Addr)
                        LDI YH, high(LCDLn2Addr)


                        ;Same while loop to put Z into mpr and store in Y
                loop2:
                        LPM     mpr, Z+
                        st Y+, mpr
                        CPI ZL, low(HELLOSTRING_END<<1)
                        brne loop2
                        cpi ZH, high(HELLOSTRING_END<<1)


                ; 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

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

;-----------------------------------------------------------
; Func: Template function header
; Desc: Cut and paste this and fill in the info at the
;               beginning of your functions
;-----------------------------------------------------------
FUNC:                                                   ; Begin a function with a label
                ; Save variables by pushing them to the stack

                ; Execute the function here

                ; Restore variables by popping them from the stack,
                ; in reverse order

                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
;-----------------------------------------------------------
STRING_BEG:
.DB                     "Zheng Zheng "          ; Declaring data in ProgMem
```

```
STRING_END:
HELLOSTRING_BEG:
.DB                     "Hello World "
HELLOSTRING_END:

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