ECE 375 LAB 4

Data manipulation & the LCD

Lab Time: Tuesday 6-7:50

Adit Kotharii(Aditya Kothari)

Mclntyre Santa Cruz

INTRODUCTION

The goal of the third lab was to display was to display our names and “Hello world” string on the LCD display, while making use of the pre-written LCD driver functions. In the process of achieving our goal, we learned about how to initializing the stack pointer and move data from program memory to the data memory.

PROGRAM OVERVIEW

In this program we display two strings “Adit Kotharii” and “Hello world!” from the program memory on to the LCD display of the AVR board with the help of pre-written functions.

INITIALIZATION ROUTINE

We first initialize the stack pointer by making it point to the ramend, so that we can use the stack. Next we call the LCD display initialization subroutine (LCDInit). After initializing the LCD screen, we transfer our two strings from the program memory to data memory, using loops. The 2 loops are used to transfer the data from where Z points to where X and Y point to using an intermediate register and post-incrementing capabilities of the X, Y and Z registers.

MAIN ROUTINE

Once the data is placed in the right locations, “LCDWrite” subroutine is called that reads strings from data memory and displays them on the LCD. After that the program enters into an infinite loop.

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 type?

The program memory is 128 Kb in size, it is mostly read only memory (unless flashing the AVR). Each memory word in the program memory is 16 bits long which hold the contents of the program. While the Data Memory is 4 kb in size and includes GPRS and IO registers. The data memory word is 8 bit in size and is used to hold temporary data.

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.

Stack is initialized in the beginning of the program. When a function call is made using (RCALL, ICALL, CALL), return address of these functions is pushed on to the stack and jump is made to the label. RET is used to pop the return address to return back to where we stopped before jumping to the label.

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

Once the stack pointer is removed, nothing can be seen on the LCD. Probably because we are moving data from unknown memory location.

CONCLUSION

In this lab, we coded in assembly for the first time. We learned about basics of assembly programing such as stack pointer initializations, make function calls, write loops and move data from program memory to data memory.

SOURCE CODE

```
;***********************************************************
;*
;*      Enter name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the skeleton file for Lab 4 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter date
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multipurpose register is
                                                        ; required for LCD Driver
.def    countName = r17
.def    countHello = r18

;***********************************************************
;*      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) ; Lower byte of ramend address 0xFF loaded in r16
                OUT SPL, R16 ; write byte to SPL
                LDI R16, HIGH(RAMEND) ; higer byte of ramend address 0x10 loaded in r16
                OUT SPH, R16 ; write byte to SPL
                ;SP = 0x10FF

                ; Initialize LCD Display
                rcall LCDInit; call to subroutine
```

```
                ; Move strings from Program Memory to Data Memory
                LDI ZL, LOW(STRING_NAME) << 1; intilizing Z pointer
                LDI ZH, HIGH(STRING_HELLO) << 1
                LDI YL, $00 ; y pointer points to data in address space $0100-$010F
                LDI YH, $01
                LDI XL, $10 ; x pointer points to data in address space $0110-$010F
                LDI XH, $01
                LDI countName, 13 ; counter = string length
                LDI countHello, 12

NameLoop:
                LPM R16, Z+ ; loaded constant from program memory
                ST Y+, R16
                DEC countName ; deceremented the counter
                BRNE NameLoop
                LPM R16, Z+
HelloLoop:
                LPM R16, Z+ ; load constant from program memory
                ST      X+, R16
                DEC countHello ; decrementing the counter
                BRNE HelloLoop

                ; 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 ; subroutien writes string to lcd
                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_NAME:
.DB             "Adit Kotharii"               ; Declaring data in ProgMem
STRING_HELLO:
.DB             "Hello World!"
```

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