ECE 375 LAB 4

Data Manipulation and the LCD Display

Lab Time: Tues 7-9

James Stallkamp

INTRODUCTION

The purpose of this lab is to use the LCD Display on the AVR microcontroller board by create an AVR assemble project in AVR Studio. In this lab we are provided with project skeleton file that is ready to compile and well commented so it helps us to create a project that are will structured, as well as, well support us to build our function, and then used the Universal Programmer to download the program onto the AVR board to get the LCD Display our project.

In the same way, the objective of this lab was to learn how to read and move data in memory by understanding the data manipulation, set up the program with initializing it, and uses indirect with the X, Y, Z-pointers. The project we have created makes our names on the first line of the LCD Display on the AVR microcontroller board and the second line a phrase.

PROGRAM OVERVIEW

The following program load a two-line ("james stallkamp","Hello World") to an LCD display connected to the Mega128 (Tekbot Board), using AVR assembly code. This process works as follows:

- Each line is saved into program memory as pairs of bytes (memory words) stored in order.
- The Z register is loaded with the high and low byre of the first line.
- The Y register loads the destination in memory of the beginning of the message to be written to.
- A counter with number of characters to read is set to the number of characters in the message.
- Each character of the message is copied from Z to Y , with Z and Y incrementing after storing the character.
- The process repeats for the second line.
- The program moves into a never-terminating loop after having written the byte.

INCLUDE

"m128def.inc" is included which defines commonly-used register on the AVR ATmega128 microprocessor.

DEFINITION

- Register 16 (r16) is defined as the multi-purpose register (mpr).
- Register 23 (r23) is used as the read-count (readCnt), loaded with the number of characters that need to be stored for each message.

PLACEHOLDING

.cseg marks where the code begins. The location of the interrupt vectors (not used in this lab) is set to $000,after which $0046 marks the beginning of the program.

INIT

This routine makes the appropriate first steps to load the messages stored in program memory into data memory so the LCD driver can read them.

INITIALIZING THE STACK POINTER

Initializing the stack pointer is important, as doing this allows routine calls to be made. This is done by loading the stack pointer's low byte with the low byte of the last address in RAM, and the stack's high byte with the high byte of the last address in RAM. Since we cannot load the last address in RAM to the stack pointer directly (as there is no immediate output command), both bytes must first be written to the MPR.

LCDINIT

The LCDinit routine from the LCD driver is called to properly initialize the LCD display.

LOADING THE FIRST MESSAGE

In order to write the message onto the first line, we must do the following:

LOADING Z AND Y REGISTER

The Z register is loaded with the address from program memory corresponding to the first letter in STRING_BEG.

To retrieve this, however, these bytes must be shifted left, to be compatible with the addressing scheme used by high () and low (). After doing this, the Y register is loaded with the bytes corresponding to the data memory location LCDLn1Addr.

WRITING TO MEMORY

Within a while loop (running for each character), the mpr is loaded with the next character of the message, stored in Z. following this, Z advances to the next address. The content of the mpr is then copied to Y, and Y is updated. The counter is then decremented, and if thee are no more byres to save to the LCD, the loop terminates.

LOADING THE SECOND MESSAGE

This process works exactly the same as loading the first message. However, STRING_END is used instead, LCDLn2Addr is used instead LCDLn1Addr, and ReadCnt is set to 12 (to correspond with the number of bytes in "GOBLIN POWER").

MAIN

Main loops forever without making any further changes.

STRING_BEG

This routine writes as a set of bytes in “james Stallkamp” to a set of address in program memory.

STRING_END

This routine writes a set of bytes in “Hello World” to a set of memory addresses in program memory.

INCLUDING LCD DRIVER

.include “LCDDriver.asm” includes the LCD driver needed for the project.

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.

Program Memory is a 16 bits, used to store the program on it without loss it after it shut down. Whereas, Data memory is an 8 bit, used to stored variables/registers so it is easy to retrieve, however it removed after we shut it down.

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.

The function call put the address of the next instruction to a location pointed to the stack, it then decrement the stack pointer. Return must be called so that the function call can read back the locations on the program that the stack pointer has stored.

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.

Without out the stack properly initialized the LCD panel does not get initialized.

CONCLUSION

This lab shows us how to use an LCD display, this lab's actual purpose seems to be storing and loading information from program memory to data memory, and setting up the stack pointer. Both of these operations are important to understand.