Computer Science 230
Assignment 3
Due: Monday July 11th before 11:59PM

Objectives

- Introduction to peripherals: LCD
- Practice with C strings and indirect addressing
- Exposure to the issues around busy-waiting

Demonstrations

You will be required to demonstrate your solution to at least one part of the assignment. You will be required to explain how your code works and make a small change to your code. You must demonstrate competence with AVR Studio 4 and uploading the code to the board.

Demonstrations will take place outside lecture and lab times. A sign up sheet will be posted on the door to the lab. If you do not sign up for a demonstration, you will receive 0 for the assignment.

AVR Studio Project

As in assignment 2, you should create a new assembly language project in AVR Studio for this assignment. Download the sample code and add the files to the project.

The LCD

The boards we are using have a Hitachi HD44780 compatible LCD display attached. This LCD display has its own (simple) processor that you must communicate with using a specific protocol in order to initialize and control the LCD screen.

You can learn more about how the LCD controller works by looking at the data sheet:

https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Luckily for you, a former student has written a library which does all the hard work communicating with the LCD controller and all you have to do is use the subroutines they have written.

The table below lists the subroutines, their parameters and what they do.

Subroutine	Parameters (passed on the stack)	Description
lcd_init	None	Initializes the LCD screen. This subroutine must be called before any other subroutine in the LCD library is used.
lcd_gotoxy	X – 1 byte Y – 1 byte	Move the LCD cursor to position (x,y). The first line on the LCD is line 0, the second line is line 1.
lcd_puts	Address of C-String in data memory – 2 bytes	Display the null terminated string at the current cursor position. This routine does no length checking. It is up to you to make sure the string isn't longer than the available space on the LCD.
lcd_clr	None	Clear the LCD screen.
str_init	Address of source string in Program Memory – 2 bytes Address of destination string in Data Memory – 2 bytes	Copy a C string from program memory into data memory.

You've been given a sample program that shows you how to use these LCD functions in the file: lcd_example.asm.

Part I – Simple LCD scrolling

You've seen LCD screens in various devices where the message scrolls across the screen. In this assignment you will create your own scrolling message device.

In this assignment we will have two messages that are scrolled across the LCD, one message on the first line, and another message on the second line. For discussion purposes, assume that:

```
msg1 = "This is the first message displayed on the first line of the LCD."
msg2 = "On the second line of the LCD there is another message that is scrolled."
```

When the program first starts, the LCD screen will contain:

This is the firs
And on the secon

After approximately ½ a second, the LCD screen will be updated to contain:

his is the first
nd on the second

And so on, so that approximately every ½ second the message will scroll one character to the left.

One important point is what happens when either of the messages near the end. The messages are to be treated as if they wrap-around. For example, at some point, the LCD will contain:

of the LCD.This
that is scrolle

Which shows that the first message has wrapped around to the start, while the second hasn't yet reached the end.

Implementation Hints

Data Definitions

WARNING:

DO NOT include a .org directive in your data segment. The lcd library has already used the .org directive and if you use it again, you will overwrite internal storage for the lcd library.

Your instructor's solution has the following in the data segment:

```
; sample strings
; These are in program memory
msg1_p: .db "This is the message on the first line. Here it goes.", 0
msg2_p: .db "--- buy --- more --- pop --- buy ", 0

.dseg
;
; The program copies the strings from program memory
; into data memory.
; l1ptr and l2ptr index into these strings
;
msg1: .byte 200
msg2: .byte 200

; These strings contain the 16 characters to be displayed on the LCD
; Each time through the loop, the pointers l1ptr and l2ptr are incremented
; and then 16 characters are copied into these memory locations
line1:      .byte 17
line2:      .byte 17

; These keep track of where in the string each line currently is
l1ptr:      .byte 2
l2ptr:      .byte 2
```

Suggested Code Structure

With these data definitions shown above, the main loop of the scrolling application looks like:

initialize the lcd

clear the lcd

copy the strings from program to data memory

set l1ptr and l2ptr to point at the start of the display strings

do forever:

    clear line1 and line2

    display line1 and line2

    copy from the pointers in msg1 and msg2 to line1 and line2

    display line1 and line2

    move the pointers forward (wrap around when appropriate)

    delay

You should make extensive use of subroutines in your code. In your instructor's solution, each line above is its own subroutine.

Including the delay code (which you can borrow from your lab) your instructor's solution is ~300 lines of assembly language.

Start early!

Part II – Extend the scrolling application

Part II of the assignment allows you to get creative and extend the scrolling application to add some additional features.

At a minimum, you must extend the application so that you can press the “UP” button to stop the display scrolling. If the scrolling is stopped, pressing the “DOWN” button will restart the scrolling. Completing only this modification will score 50% of the Part II grade.

Some other suggested improvements:

1. Allow button presses to increase and decrease the scrolling speed
2. Allow the user to select different messages
3. Do something with the LEDs while scrolling

In order to handle button presses gracefully you will have to do something other than busy loop waiting – either by checking the buttons in your delay subroutine or by using timer interrupts.

Your instructor will be covering timer interrupts in the next week or so.

Submission

Submit a3.asm file using connex. Do NOT submit your project file – just the .asm files.

Grading

If you submit a program that does not assemble you will receive 0 for that part of the assignment.

If you do not complete a demonstration of your code during the posted demonstration times, you will not receive a grade for this assignment

Indirect addressing exercise	2 marks
Part I	10 marks
Part II	8 marks

Total 20 marks