Computer Science 230
Computer Architecture and Assembly Language
Summer 2023

Assignment 3

Due: Monday July 17, 11:55 pm by Brightspace submission
(Late submissions are **not** accepted)

Programming environment

For this assignment you must ensure your work executes correctly on Arduino boards in ECS 249. If you have installed Microchip Studio on your own computer then you are welcome to do some of the programming work on your machine. However, please plan to spend a significant amount of time in the lab. ***Please note that Lab 7 and Lab 8 material is very useful to learn more about timers and LCD display.***

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor.** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact me as soon as possible. (Code-similarity analysis tools will be used to examine submitted work.) The URLs of significant code fragments you have found and used in your solution must be cited in comments just before where such code has been used.

Objectives of this assignment

- Use AVR timers.
- Write interrupt handlers for timers.
- Output a representation of program state onto the Arduino mega2560 board's 2x16 LCD display.
- Gain confidence with assembly language programming.
- Create effective subroutines/functions.
- Use peripherals: the LCD

Interrupts, LCD panel

In this assignment you will use two features of the Arduino boards that have been introduced recently in lab: interrupts and the use of the 2x16 LCD display. The handlers required for the assignment are not particularly complex, but they must work precisely. The LCD panel will finally allow you to create board behavior that is

richer than simply turning LEDs on and off; the labs earlier this semester introduced you to the LCD panel and interrupts to produce timing (delay) information.

The LCD

The boards we are using have a Hitachi HD44780 compatible LCD display that has its own (simple) processor. To communicate with that processor a specific protocol is used that initializes and controls the LCD screen. Learn more about how the LCD controller works by looking at the data sheet:
https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

A previous CSC 230 student has written a library of subroutines for this assignment. 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.

There is an example program that shows how to use the LCD functions in the file: **`lcd_example.asm`** which displays two lines. Review the files, searching for a way to set the number of rows on the LCD to 2 and the number of columns to 16. You can use the lcd_example.asm as a starter file and expand on this. Please note that the lcd_example.asm file is a fully working example. It can be assembled and uploaded to the board.

Part A: LCD Moving Message Sign

Advertising signs often move or flash written messages on a screen. The movement and the flashing are very effective in attracting attention. The goal of this programming task is to display alternately two written messages, gradually moving

them down and across the screen, and to display and flash both messages. **Please name this file as "display_partA.asm" for submission on Brightspace.**

In particular, the program will need to:

a. Create two messages that will be displayed on the screen. For example, assume:

```
msg1 = "YourName"
msg2 = "CSC 230: Summer 2023"
```

b. When the program starts, the LCD screen will contain: YourName in the first row of the screen and `CSC 230: Summer 2023` in the second row.

c. After approximately one second, the LCD screen will be cleared then will be set to contain only YourName as the first line of the screen. After approximately one second, the LCD screen will be updated and the first message cleared with only the "**`CSC 230: Summer 2023`**" displayed on second line **.**

d. After another one second, repeat all of the above from (a) in a continuous loop.

e. Use a timer to generate 1 second delay in the above. Clearly mention how the main program detects the 1 second timing information,

Part B: LCD Scrolling Message Sign

The LCD screen is limited to 16 characters at a time per line. One way to display longer messages is to implement scrolling. Extend Part A so that the text now scrolls left. Test it also with message length > 16 characters. Use another timer to generate delay to scroll text which controls the scroll speed. **Please name this file as "display_partB.asm" for submission on Brightspace**

Part C: LCD Scrolling Message Sign with keypad interaction

Extend Part B to show on the LCD panel message scroll to left if left button is pressed and to the right if right button is pressed and ignore all other button presses. **Please name this file as "display_partC.asm" for submission on Brightspace**

A side note: "What button was pressed?"

In lab 4 you may have done a little bit more with buttons – in fact, you may have even had an opportunity to write code to determine precisely which button is pressed. The ADC obtains a value from 0 to 1023 from that represent button states – if the value is greater than 900, then definitely no button is being pushed. The ranges for all buttons on our LCD/button shields are maddingly inexact and may require a bit of tweaking and tuning, but they can be described roughly as follows:

- from 0 to around 50: "right" button[12]
- from around 50 to around 176: "up" button
- from around 176 to around 352: "down" button
- from around 352 to around 555: "left" button
- from around 555 to around 800: "select" button
- above 900: *no button is pressed*

For this assignment we will only use the "left" and "right" buttons.

What you must submit

- Your three completed parts: `display_partA.asm`, `display_partB.asm` and `display_partC.asm`. **Do not change the name of these files!** Do not submit the provided LCD files. **Submit only the `.asm` files listed above!**

Evaluation

- 5 marks: Solution for part A
- 5 marks: solution for part B
- 5 marks: solution for part C

Therefore the total mark for this assignment is 15.

Some of the evaluation above will also take into account whether or not submitted code is properly formatted (i.e., indenting and commenting are suitably used), and the file correctly named. Please appropriately comment your code so that the evaluator can understand your design.

Unlike other assignments, A#3 will be evaluated via a code demo. That is, each student will meet with a member of the teaching team, who will have access to the student's submitted code and will ask that student questions about their code. Instructions on how to sign up for a demo will be given sometime during the week of July 24th.