CSC 230 Assignment 3
Spring 2017

This assignment has three parts:

1) **Programming Pre-submission**: A subset of the complete program needs to be submitted via the Assignments link on the Connex site on or before March 11 well before 11:55.
2) **Programming Submission**: The final program will be submitted via the Assignments link on the Connex site on or before March 18 well before 11:55. The main submission file should be in a file called display.asm; the extended work should be in a file called display_ext.asm.
3) **Demonstration**: You will be required to demonstrate the functioning of the programming part of your assignment to a member of the teaching team. **Demos will occur the week of March 20-24 in the labs.**

Objectives

- Gain confidence with assembly language programming.
- Create effective subroutines/functions.
- Use peripherals: the LCD
- Become familiar with the use of busy-waiting
- Become familiar with the use of interrupts

Academic Integrity

Prior to submitting your assignment, carefully read the University policy on Academic Integrity: http://web.uvic.ca/calendar2014/FACS/UnIn/UARe/PoAcI.html
If you do not understand the meaning of anything in the document, please ask. A plagiarism detection tool will be used on assignment submissions.

Assignment 3 Description

Please note for all programing assignments, ***you must include your name and student number*** and ***edit the documentation*** appropriately**.** Make sure the submitted work is yours and not someone else. The directions below describe

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. Review both files carefully until you are able to understand their functionality. While doing that, search for a way to set the number of rows on the LCD to 2 and the number of columns to 20.

LCD Moving Message Sign

LCD advertising signs often move written messages on a screen and flash. The movement and the flashing are very effective at attracting attention. The goal of this programming task will be 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.asm" for submission on connex.**

In particular, the program will need to:

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

```
msg1 = "YourName"
msg2 = "UVIC: Computer Sci"
```

b. Using the messages (`msg1` and `msg2`) create another two messages that contain the same characters, but in reverse order. Using the above example messages, the messages, called `msg3` and `msg4`, would be:

```
msg3 = "emaNruoY"
msg4 = "icS retupmoC :CIVU"
```

c. When the program starts, the LCD screen will contain both: `msg1` in the first row of the screen and `msg2` in the second row both of which start in the first column.

d. After approximately one second, the LCD screen will be cleared then will be set to contain: `msg2` on the first line of the screen, starting in the second column, and `msg3` on the second line of the screen, also starting in the second column.
e. After approximately one second, the LCD screen will be cleared then will be set to contain: `msg3` on the first line of the screen, starting in the first column, and `msg4` on the second line of the screen, also starting in the first column.
f. After approximately one second, the LCD screen will be cleared then will be set to contain: `msg4` on the first line of the screen, starting in the second column, and `msg1` on the second line of the screen, also starting in the second column.
g. <continue here>
h. After approximately two seconds, clear the screen, then repeat all of the above steps 2 more times.
i. Finally, clear the screen, then set the screen to flash (alternate between) the screen totally full of the character '*' and the screen totally full of the character '.'.

Implementation Help

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

```
; sample strings
; These are in program memory
msg1_p: .db "YourName", 0
msg2_p: .db "UVIC: Computer Sci", 0

.dseg
;

; The program copies the strings from program memory
; into data memory.
;
msg1: .byte 200
msg2: .byte 200
; These strings contain the 18 characters to be displayed on the LCD
; Each time through, the 18 characters are copied into these memory locations
line1: .byte 19
line2: .byte 19
line3: .byte 19
line4: .byte 19
```

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

```
initialize the lcd
clear the lcd
copy the strings from program to data memory
do 3 times:
  set row counter to 0 and column counter to 0
  display line1
  set row counter to 1
```

```
  display line2
  delay 1 second
  set row counter to 0 and column counter to 1
  display line2
  set row counter to 1
  display line3
  delay 1 second
  set row counter to 0 and column counter to 0
  display line3
  set row counter to 1
  display line4
  delay 1 second
  set row counter to 0 and column counter to 1
  display line4
  set row counter to 1
  display line1
  delay 2 seconds

  do many times:
    clear the lcd
    display all *
    delay 1 second
    display all .
    delay  1 second

clear the lcd
```

You should make extensive use of subroutines in your code. In fact, each line above is its own subroutine. Including the delay code (which you can borrow from your lab) the solution is ~300 lines of assembly language. Start now!

Extending the Moving Message Sign

Now, add some features. Here are some options. You can choose **any one** of them.

- Implement a scrolling display for longer messages: Scroll them across the screen, wrapping such that a character that falls off one end will appear on the other.
- Extend the application so that you can press the "UP" button will display only messages 1 and 2 while pressing the "DOWN" button will display only message 3 and 4.
- Add another improvement, here are some suggestions:
    - Allow button presses to increase and decrease the sign change speed
    - Allow the user to select different messages
- **Please name this file as "display_extended.asm" for submission on connex.**

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.