EE 337: Interfacing to LCD Display
Lab 4

August 17, 2016

In this set of experiments, we develop display and other utilities which will be useful to us for the later experiments. For these experiments, you will have to attach the LCD unit to the Pt-51 board. Please remember that the display has to be plugged in such a way that it extends *outside* the board and *not* over it. Plugging in the display in the wrong orientation may damage it ! Please refer to the attached tutorial on Liquid Crystal Display Control.

For this lab, you are provided with a subroutine (`lcd.asm`) which writes characters to the LCD display. Study the program to see how it works. You can use it or write your own code to achieve the same functionality.

Note: For doing these exercises you can make use of `bin2ascii` subroutine developed in previous lab to display ascii characters to LCD. Also, you can use any other subroutine developed as part of previous labs.

1 Homework

1. Using the supplied routine for writing characters to the LCD, Write a program that will display "EE 337 - Lab 2" on the first line and your name on the second line. Pad the display lines with spaces such that these are centered on the LCD when displayed. Your name should not be hard coded in the program, but stored as a 16 byte array of characters in the upper RAM. The program should display whatever is stored in this array. You should assemble, debug, download and run this program on the Pt-51 kit before coming to the lab.

2. This set of subroutines developed are used to display the element of an array, whose index is specified using switches.

   (a) Write a subroutine `packNibbles`. Two successive 4 bit values read using `readNibble` (developed in previous lab) should be combined to form a byte with most signif-

icant nibble being read first followed by least significant nibble, which should be stored at location $4FH$.

(b) Write a subroutine `readValues` that will call `packNibbles` to read in K bytes ($0 < K < 10$, with K stored in 50H) this way, storing them in an array in memory starting at location P. Location $51H$ holds the value of P. Remember that `packNibbles` will always place the byte at $4FH$ but `readValues` should copy the values to different locations.

(c) Write another subroutine `displayValues` that should read a pointer to an array from $51H$ and value of K from 50H. It should then read a 4 bit value from the port. If this value is greater than or equal to K (invalid input), the subroutine should clear the LCD and stop. Otherwise, this value should be used as an index in the stored array. The corresponding byte should be displayed on LCD every 2 seconds. The subroutine must be able to read the switches and update continuously.

2 Lab Work

1. Using the subroutines developed in homework problem-2, store K elements of an array by reading the switches of port P1. Use location $60H$ to store the values of array A.

   Now, write another subroutine `shuffleBits` which does the following task: if the original array is $A[0]$, $A[1]$, ..., $A[K-1]$, generate the array B, such that it contains

   $$A[0]\ XOR\ A[1], A[1]\ XOR\ A[2], \ldots, A[K-1]\ XOR\ A[0].$$

   Location for array B is $70H$. Now, call `displayValues` from homework 2 to display the elements of B by reading the index from the port. Use the template code provided below.

```
;======================MAIN====================
ORG 0XXXH
MAIN:

MOV SP,#0CFH;-----------------------Initialize STACK POINTER
MOV 50H,#XX;------------------------Set value of K
MOV 51H,#XX;------------------------Array A start location
MOV 4FH,#XX;------------------------Clear location 4FH
LCALL readValues

MOV 50H,#XX;------------------------Value of K
MOV 51H,#XX;------------------------Array A start location
```

```
MOV 52H,#XX;-----------------------Array B start location
LCALL shuffleBits

MOV 50H,#XX;-----------------------Value of K
MOV 51H,#XX;-----------------------Array B start Location
LCALL displayValues;----------------Display the last four bits of elements on LEDs

here:SJMP here;---------------------WHILE loop(Infinite Loop)
END
; -----------------------------------END MAIN-----------------------------------
```

2. Write a program which will display the contents of 16 locations in the on-chip RAM. The location will be specified by setting switches on the board. These 4 bits will be interpreted as the more significant nibble of the RAM address, the less significant nibble will be taken as 0. You have to read the switches twice with a delay and only when the two values agree that you will proceed to display the memory contents.

   The contents should be displayed 4 bytes per line. Thus 8 bytes will be displayed at a time. The next 8 bytes should be displayed after a pause of 5 seconds.

   Notice that you have to check whether the address range in question is in the directly addressable memory (00-7FH) or in the indirectly addressable memory (80-FFH). The contents shown should have been fetched using the correct addressing mode.

   The whole sequence should repeat endlessly. That is, read switches, display 8 bytes, wait for 5 seconds, display the next 8 bytes, wait for 5 seconds, read the switches again and so on.