ECE253 – Digital and Computer Systems

University of Toronto

Lab 9: Subroutines, Stacks, I/O devices

The goal of this lab exercise is to continue to gain familiarity with assembly language programming, to learn more about subroutines, and to communicate with I/O devices by using memory-mapped I/O.

Preparation and In-Lab Marks

Preparation marks: Write the required pseudo code and assembly language program for Part I. **Note:** your code be commented well enough to enable another person to understand them.
In-lab marks: You are required to demonstrate to a TA *both* parts of the lab exercise. For Part I you should use the Monitor Program to illustrate your sorting algorithm—place a breakpoint after each call to the SWAP subroutine, and display the Memory tab so that you can observe the sorting of the list. For Part II, you need to be able to demonstrate the sweeping pattern of the LEDs and show that you can start and stop it using KEY_3.

Part I

Write an assembly language program to sort a list of 32-bit unsigned numbers into ascending order. The first 32-bit number in the list gives the number of items in the list and the remainder of the entries are the numbers to be sorted. The list of data must be sorted "in place", meaning that you are not allowed to create a copy in memory of the list to do the sorting. The list you should sort, including the number count, can be defined using the **.word** directive at the end of your program as follows:

```
List:    .word    10, 1400, 45, 23, 5, 3, 8, 17, 4, 20, 33
```

Note that the first number in the list is 10, indicating that there are 10 numbers following it. In your code you should have a main program that loops through the list of numbers as needed to perform the sorting operation. Use **bubble sort** to perform the sorting. Provide pseudo code, similar to C code, that illustrates how your sorting algorithm works. Then, write an assembly language program.

As your main program loops through the list of numbers it will be necessary to compare the values of pairs of list elements to see if they need to be swapped. You are to use a subroutine, called SWAP, to compare such list elements. The SWAP subroutine is passed the address of a list element, compares it to the following element in the list, and swaps the two elements in memory if necessary. The SWAP subroutine should return 1 if a swap is performed, and 0 if not. Pass the address of the first list element to SWAP in register R0, and also pass the return value back to the main program in register R0.

Part II

Write an assembly-language program that turns on two LEDR lights at a time on the DE1-SoC board. First, the lights $LEDR_9$ and $LEDR_0$ should be on, then $LEDR_8$ and $LEDR_1$, then $LEDR_7$ and $LEDR_2$, and so on. When you get to $LEDR_5$ and $LEDR_4$ being on, the direction should be reversed. Only two LEDR lights are ever on at one time. The effect should be a two lights sweeping first towards the centre, then outwards towards the edges, and so on.

Use a delay so that lights move every 0.25 seconds. To implement the delay, use the MPCORE Private Timer described in the lectures (and also in Section 2.4.1 of the Intel documentation on the Piazza website.) When KEY_3 is pressed, the sweeping motion should stop. Pressing KEY_3 again should restart it.

To synchronize with the timer device in your main program, use polled I/O by repeatedly reading the F bit in the timer's status register. Also use polled I/O to deal with the pushbutton KEY_3.