ECE353 In Class Exercise

ARM Assembly – Function Calls

Problem 3B

You will implement a bubble sort algorithm to sort an array of unsigned **bytes**. You will be asked to complete the ARM assembly code in ece353_main.s and bubbleSort.s. Hand in a copy of bubbleSort.s. Read the requirements for each file below.

Requirements (bubbleSort.s)

1. Add the correct export statement for bubble_sort
2. **Do NOT add any PUSH or POP statements to this file!**
3. Complete the function swap_values. The description of the function can be found in the comments before the function
4. At the label for bubble_sort, write the necessary ARM assembly to implement the bubble sort algorithm. There is a C implementation of the bubble sort algorithm below. The parameters to bubble_sort are as follows
    a. R0 ← Address of array to be sorted.
    b. R1 ← Number of **unsigned bytes** in the unsorted array. Make sure to use LDRB.

A simple C version of Bubble sort is given below

```
//*****************************************************************************
// C Implementation of bubble sort
//*****************************************************************************
uint32_t bubble_sortC(uint8_t * address, uint16_t size)
{
  uint8_t temp;
  uint8_t j;

  size = size - 1;  // Only need N-1 passes through array

  while(size > 0)
  {

    j = 0;

    while(j < size)
    {
        swap_values(address[j]); // If address[j] > address[j+1], then swap
        j++;
    }

    size--;
  }

}
```

Requirements (ece353_main.s)

1. Add the correct import statement at the top of the file for the bubble_sort function
2. Initialize R0 and R1 to have the following values
    a. R0 ← Address of SORTED
    b. R1 ← Number of unsigned bytes in SORTED
3. Call the function bubble_sort.
4. A function called verifyArray has been provided for you that will verify the results of your bubble sort. Place a break point at the two infinite loops at the end of __main and observe if your bubble sort routine works correctly.