EEE351A Lab 05 A Sorted Affair

# LAB OBJECTIVE

The objective of this lab is to continue with the process of writing code for simple tasks. Use of the various addressing modes covered in class is emphasized.

# Introduction

Sorting algorithms are a common tool in many applications. The ability to rapidly and correctly sort any alpha and/or numeric data is a key component of applications such as spreadsheets and word processors. Use the [bubble sort](http://en.wikipedia.org/wiki/Bubble_sort) algorithm (see code below).

# Lab Challenge

## Goal

Your task is to write an algorithm (using subroutines) to sort a given series of numeric data stored in memory from low values to high values. The values to be sorted will be 1 byte (8 bits) values to get practice using data other than characters.

## Constraints

* Use at least one subroutine in your solution.
* Use ROM to hold your data and/or program as required.
* The list is given on the moodle website <http://moodle.rmc.ca>
* The list is terminated by word $FF.
* You may be given a zero-size list.
* The original list may not be altered. Place the sorted list at a memory location of your choosing.
* Use the bubble sort algorithm (or any of its [variations](http://en.wikipedia.org/wiki/Bubble_sort)) to sort the list.
* I recommend using indexed- addressing (of some form) in the algorithm.

# Questions

There are no questions for this report.

Hint: For the TESTING section, you must test a **variety** of data. The final data set I provided is NOT considered a variety. Add at least two other sets of data to sort (one should be an empty set).

# Report

# A SHORT lab report is required for this lab.

# Summary

In this lab you tackled a more difficult programming challenge. Sure, it may have been a little tricky, but you should be beginning to realize that there isn't too much you can't program in assembly.

### Pseudocode implementation (from wikipedia)

procedure bubbleSort(A : list of sortable items)

n := length(A)

repeat

swapped := false

for i := 1 to n-1 inclusive do

/\* if this pair is out of order \*/

if A[i-1] > A[i] then

/\* swap them and remember something changed \*/

swap(A[i-1], A[i])

swapped := true

end if

end for

until not swapped

end procedure