Project 4

Bubble Sort Redux

Due: Friday the 8th December 2017

Overview

In this project, you will revisit the first project of the semester: bubble sort. Only this time, you'll be writing it in assembly.

In order to run your assembly program, you will need the qtSPIM simulator for MIPS assembly. You may download this program at the following address:
`http://spimsimulator.sourceforge.net/`

Additionally, the tutorial at the following address will likely prove helpful:
`http://chortle.ccsu.edu/AssemblyTutorial/index.html`
Keep in mind though that they only use the basic assembly instructions rather than the extended ones in most places. You don't have to do that though!

Submission Instructions

Submit only your .s file for this project. The name of the file is unimportant, but something sensible like `bubble_sort.s` would be appreciated.

Technical Description and Instructions

To lessen the burden of this project, you will only be required to sort a static list of numbers hardcoded into your program. Thus, no user input will be involved.

Your program needs to do only three things:

1. Print a list of 10 integers
2. Sort the list
3. Print the sorted list

Print a list of 10 integers

When your program prints the unsorted and sorted lists, it can do so in a variety of ways. However, I suggest that you make things easy for yourself by simply printing out

Sort the list

Your program will use bubble sort to sort the integers the user provides as the algorithm is simple and easy to implement. If you need a refresher on how bubble sort works, consider consulting the Wikipedia page for it.

Things to Remember and Helpful Hints:

- To store a static sequence of ten numbers, you should consider using the `.word` directive.
- There should always be a `nop` (do nothing) instruction after every jump or branch instruction. This is explained here:
  `http://chortle.ccsu.edu/AssemblyTutorial/Chapter-17/ass17_2.html`
- When iterating through the number sequence, you will need to track both the number of completed iterations **and** the index of the next number. They are not the same in this case! Links for looping and indexing:
  `http://chortle.ccsu.edu/AssemblyTutorial/Chapter-20/ass20_1.html`
  `http://chortle.ccsu.edu/AssemblyTutorial/Chapter-24/ass24_11.html`
- Don't be clever with the registers. Just assign each non-temporary variable to a particular regester and then stick with that scheme. Likewise, don't leave important values in registers that are frequently overwritten like `$v0` or `$a0`.

Grading Specification

For your submission to receive a grade of 'pass', it must fulfill the following criteria:

- It must be submitted correctly
- I must be able to load your program into qtSPIM and run it with the default settings
- The program should run with no errors and should produce a correct result without crashing