CST250 Final Project: Sorting Optimization

Learning Objectives:

- Compare different approaches/solutions to a specific assembly programming situation/problem and evaluate which one is better
- Create and deliver technical presentation that is a well designed, easy to follow and clearly conveys the work that was done

The Task

The goal of this project is to test your ability to write and optimize PLP programs. You will be given skeleton code, which has a call to a sorting function, which you will need to implement. This sorting function will be an implementation of the gnome sort algorithm and needs to perform an in-place sort on an array in memory (only memory within the array is modified while sorting). This project will end in a competition. To the victors go the spoils, extra credit points (each competition is worth 1 point). Your algorithm implementation will compete for two separate prizes. One for the implementation with the fewest number of instruction (comments, labels, and white space don't count as instructions) and one for the implementation that takes the fewest cycles to execute. The top 10 implementations in each competition will receive a extra credit (in the case of a tie for 10^{th} place, all students that tie will receive extra credit).

Your implementation needs to be written in the ASM file, *gnome_sort.asm*, where a stub has already been added (functions should use the return instruction that is already located in the stub). The source files, *main.asm*, *input_and_verification.asm*, and *output.asm* **should not be modified**.

When the program is run it waits for a list input from the UART. The format for input is a list of positive decimal values separated by a single space and ending with a semicolon (;). Multiple inputs can be concatenated together and will be run by your sorting function sequentially (note that there is not space between the semicolon and the next number). For example, the following are both valid input:

```
250 10 2015 4 1;
```

And:

```
1 2 3 4;78 54 2891 75;
```

Your function will be passed an array starting address and the number of elements in the array using the registers `$a0` and `$a1`, respectively. It should sort the array in-place (i.e. it should not allocate additional memory or write outside the designate array in memory) from smallest to largest. The array will then be validated to ensure it is

sorted and then an output will appear on the UART. The output for will either be a running sum of the cycles used by the algorithm **since the program started running**, or the number of times the algorithm has failed validation since the program started.

PowerPoint Presentation and Extra Credit Video

Make a PowerPoint presentation document that cover your approach to optimizing your sorting algorithm implementations. Keep in mind that discussing an approach that didn't work is just as valuable as one that did so your presentation is not dependent on a working implementation. It should include visual aids, such as flow charts or block diagrams, to illustrate your approach. You are allowed to create a video of this presentation for extra credit. Because it is for extra credit, we expect high quality sound (limited background noise, such as fans) and clear video. It should be no longer than 8 minutes (it will not receive credit if it is over 8 minutes) and it should be uploaded to the video sharing website of your choice (YouTube, Vimeo, etc.). If the video is hosted on YouTube then setting it as **unlisted** works well, but if you would prefer to make the video private, please grant viewing permission to *cmar2@asu.edu*.

Deliverables:

1. Submit a PowerPoint presentation document detailing your approach and lessons learned (5 points)
2. Submit your PLP program on Blackboard (15 points)
3. **Extra Credit (2 points):** Submit a link to your video along with your presentation slides