Joshua Pollock

CS 200 Project 8

14 April 2017

Joshua Pollock

CS 200

Project 8 – Bubble Sort

14 April 2017

**Purpose:**

The goal of this project is to create a program that will prompt a user to enter an amount of numbers they wish to sort (one to ten integers). The program will then prompt the user to enter an integer until they entered the amount that they wished to sort. This program will then sort the inputted numbers and print out the sorted numbers.

The program should work in four procedures as described by the project guidelines. These procedures are: ReadNums, BSort, PrintNums and a main. The program should not be written in one large routine. Besides these few requirements, there were not many guidelines for project 8.

**Research:**

When I first started this project, I had no idea what bubble sorting was. A quick search revealed a Wikipedia page (<https://en.wikipedia.org/wiki/Bubble_sort>) with a small GIF image that showed how bubble sorting worked. This Wikipedia page was quite helpful in seeing what exactly needed to be done, and I began brainstorming ideas on how I would attempt this project.

The next thing I searched was for existing bubble sort solutions. I did not find much in the way of existing solutions in MIPS that fit the project requirements. There were several posts on StackOverflow about bubble sorting in MIPS, but I found little to no use in these solutions. However, I searched for a program in C and found a quite useful framework for creating MIPS code. This code was located at <http://www.programmingsimplified.com/c/source-code/c-program-bubble-sort>.

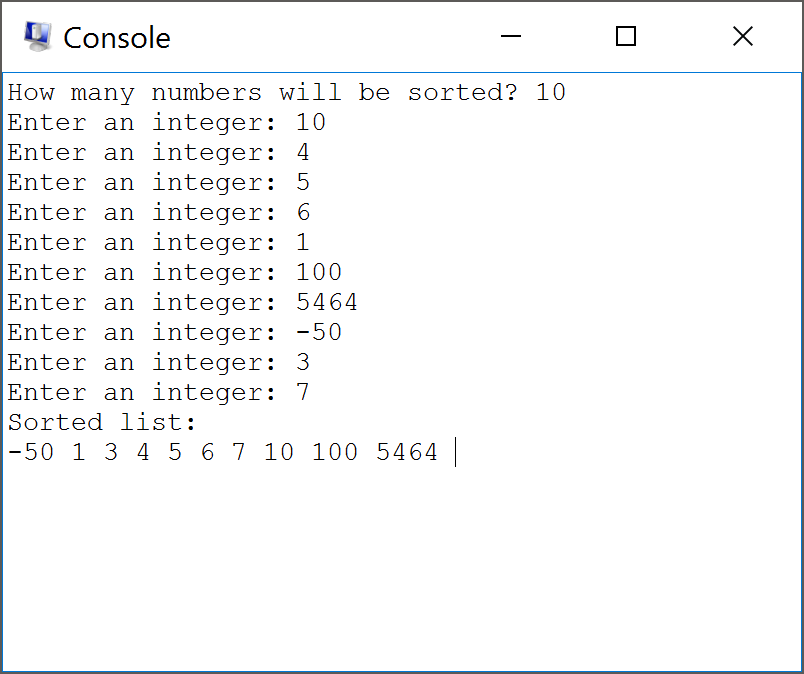
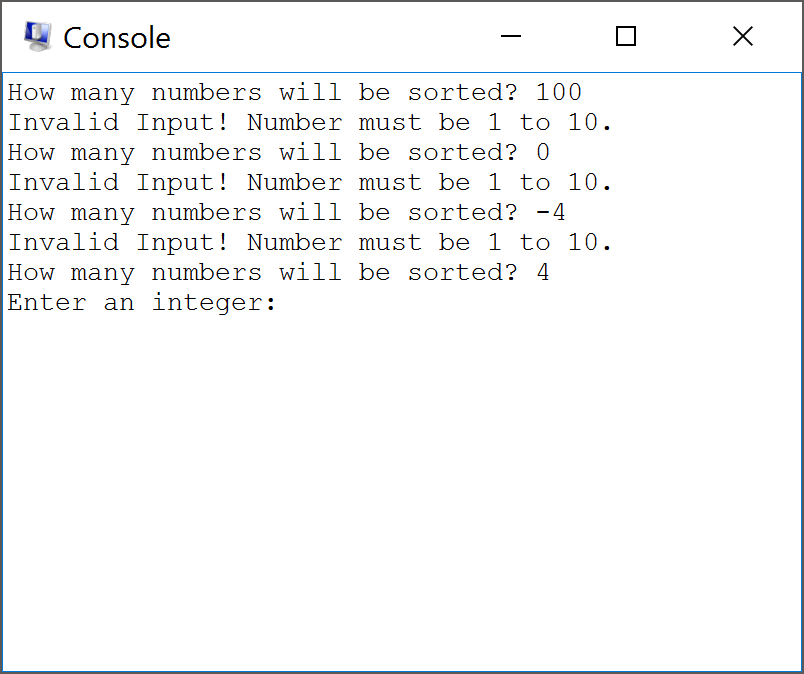
I do not know much in the way of arrays in MIPS, but I had begun experimenting with the stack. While the MIPS reference sheet on BBLearn does have some material about the stack, I looked into more information about the MIPS stack. This led me to the website <https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/stack.html> which does a great job at explaining the stack. I did not know whether or not I would be marked down for using the stack instead of an array, but found the stack to be much easier to implement than an array.

**Program:**

After many trials and errors, I had finally created working code. Getting the code to ask the user for input was quite easy to implement. I used a lot of my techniques from project 7 for this to work. The BSort function was by far the most difficult to implement.

The BSort function contains two loops, one being “nested” within the other. It took a while to figure out how to move and edit around the stack. The way I have it set up currently, the program creates a counter and a stop condition, and then begins the “nested” loop. The nested loop loads two adjacent numbers from the stack, and tests if the first numbers is greater than the second. From here the program jumps to swap or noSwap. If no swap occurs the program simply increases the stack pointer and repeats the loop if there are numbers remaining in the stack. If a swap does occur, all that is different is that the program swaps the first number with the second number. After this swap occurs, the program then continues to follow the noSwap function.

Once BSort runs through the entire stack, it will then jump to the PrintNums function. The function starts off by printing outStr, and then creates a counter based of the stack pointer. The program then creates a stop condition based off the $s0 and $t5 registers. Next the program starts a loop that prints items from the stack until there are no more items to print. To finish it off the program jumps to end which jumps registers.



**Conclusion:**

Overall, I enjoyed this project just as much as project 7. The project guidelines were a bit confusing and I did not understand some of the requirements that the project was giving. I did not understand what I was supposed to store in the $s0 and $s1 registers. I decided instead of using an array, I would use the stack. While at first I bit off more than I could chew, after a little bit of research into the stack, things became much clearer. Unlike projects in the past, there was not a lot of useful code already on the internet.

The key to this project for me was pushing through. Experimentation with the stack and loops proved to be quite useful in implementing a final version of the code. I am now quite comfortable in using the stack compared to last week where I did not know how to use the stack at all. I am a bit concerned that I did not do this project correctly as I did not store anything in the $s1 register. I did end up storing a value in the $s0 register, but I do not know if this was the correct thing to store in that register. To run my program, I used QTSpim and to create my program I used the Atom text editor.