

# **CS 200: Computer Organization**

## **Project 8: Bubble Sort**

Shariq M. Jamil

Due: Friday, April 11, 2014

## **Overview**

### **Purpose**

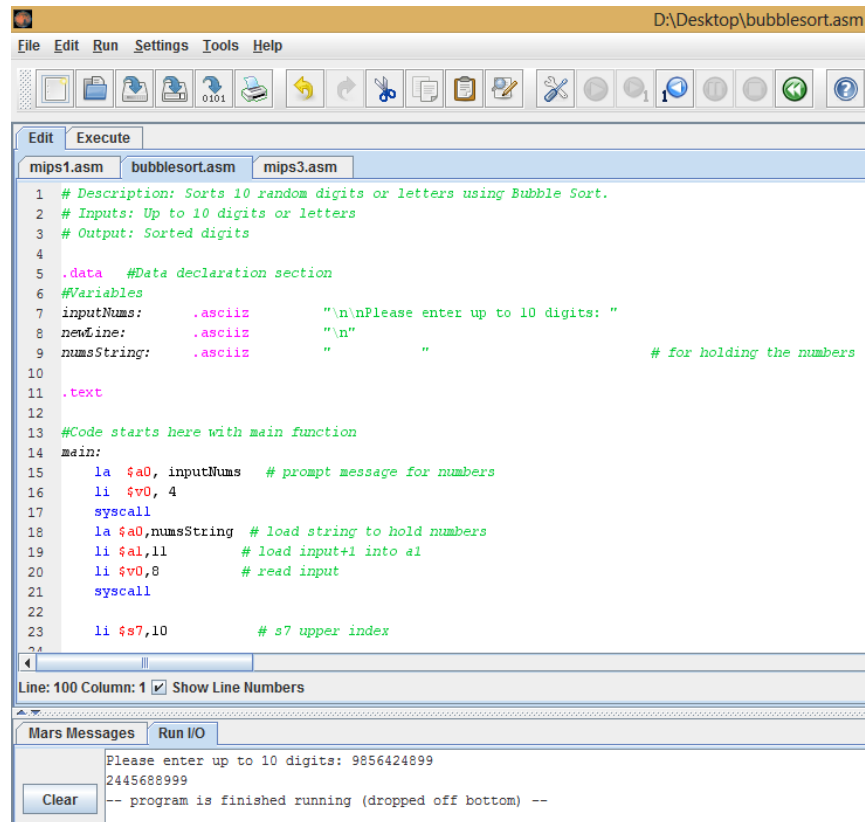
This project required us to write a program in MIPS assembly language that works as a random number generator. The program accepts a set of numbers from the user, sorts them in order with the bubble sort algorithm and displays the sorted set of numbers.

### **Approach**

I had to restudy the Bubble Sort algorithm for this project (embarrassing because I have had to implement Bubble Sort in Java and C++ in the past). This helped me develop pseudocode and consequentially assembly code for the program. I continued to use the MARS IDE for this assignment. Researching for this assignment paid off really well as I learned to use ASCII codes to sort numbers. If the current element was larger than the next, the elements got swapped in the array. Using this I could not only sort numbers but also letters. The caveat was that I had to convert lowercase numbers to uppercase and then sort them. The MIPS Instruction Reference came in very handy for this project in terms of finding functions to use to convert pseudocode to assembly code.

## Solution

## Sample Output



```
1  # Description: Sorts 10 random digits or letters using Bubble Sort.
2  # Inputs: Up to 10 digits or letters
3  # Output: Sorted digits
4
5  .data  #Data declaration section
6  #Variables
7  inputNums:  .asciiz      "\n\nPlease enter up to 10 digits: "
8  newLine:    .asciiz      "\n"
9  numsString: .asciiz      "                "      # for holding the numbers
10
11 .text
12
13 #Code starts here with main function
14 main:
15     la $a0, inputNums  # prompt message for numbers
16     li $v0, 4
17     syscall
18     la $a0,numsString  # load string to hold numbers
19     li $a1,11          # load input+1 into a1
20     li $v0,8           # read input
21     syscall
22
23     li $s7,10          # s7 upper index
24
```

Line: 100 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

Please enter up to 10 digits: 9856424899  
2445688999  
-- program is finished running (dropped off bottom) --

Clear

Figure 1: Sample Output

## Conclusion

This was a good exercise in working in assembly. I liked this project because it made me want to solve this problem in a way that I had not before. Using ASCII to do this was a good exercise in thinking out of the box and bringing letters into the game. Learning branch instructions was crucial for this project as they allowed me to change control flow and write conditional statements. I found a course lecture on MIPS Assembly Language that helped solidify branching and jump concepts and the MIPS Instruction

Reference was great for finding new functions to answer this problem. Both references are linked below.

## **References**

### MIPS Assembly Language

<http://www.inf.uni-konstanz.de/dbis/teaching/ws0304/computing-systems/download/rs-05.pdf>

### MIPS Instruction Reference

<http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html>