Homework #9 - Due 11/21/08 (Friday)

Write a MIPS assembly language program to perform bubble sort on an array containing "length" elements.

```
lastUnsorted = length - 1
 sortedFlag = 0
 while (lastUnsorted >= 1 and sortedFlag == 0) do
    sortedFlag = 1
    for test = 0 to lastUnsorted-1 do
        if (numbers[test] > numbers[test+1]) then
           temp = numbers[test]
           numbers[test] = numbers[test+1]
           numbers[test+1] = temp
           sortedFlag = 0
        end if
    end for
    lastUnsorted = lastUnsorted - 1
 end while
```

Use the data below when you run your program.

```
            .data
numbers:    .word 20, 30, 10, 40, 50, 60, 30, 25, 10, 5
length:     .word 10

            .text
             .globl main
main:
      ...

      li      $v0, 10              # system code for exit
      syscall
```

Directions: (See hw #8 description for detailed directions: http://www.cs.uni.edu/~fienup/cs041f08/homework/hw8.pdf)

1) Write your assembly language program on paper first! I will not help anyone debug their program without your handwritten program.
2) Type in your program using WordPad. Remember to save it as a **Text Document** on a USB flash memory stick.
3) Debug your MIPS assembly language program.
4) When it is correct, **run it to completion** and copy to the Window's clipboard a snapshot of the PCSpim window by using the <Alt> and <Print Screen> keys together.
5) Open up new Word document and set its page layout to Landscape by File | Page Setup | Paper Size and then select Landscape.
6) Paste the snapshot of the PCSpim Debugger window into the Word document. Resize the snapshot to the margins and print a copy to turn in.
7) Print a copy of the assembly language program to turn in too.
8) Hand in a copy of your assembly language program **and** the snapshot of the PCSpim window showing the resulting sorted memory.

EXTRA CREDIT: Rewrite your bubble sort program using the technique of "walking pointers" to reduce the amount of array address calculations, i.e., use the regular access pattern in the array to update the register containing the address of numbers[test] to the correct spot in memory.