CS231 Computer Systems and Organization
Department of Computer Science
SEDS

Lab 5 (Lab Practice)

NOTE: Students don't have to submit your work (i.e., will not be graded), but will be included in the midterm based on the Ch2 presentation slides P83 - P87 (the C Sort example).

1. **As a simple variation of the presentation slides (Pages 83 ~ 87)**, implement a procedure **string_bubble_sort** in MIPS assembly language that, given a string ***S*** and its ***length***, sort ***S***. You should print out the original string and the sorted string respectively.

For example, if ***S*** ="HelloWorld" and ***length*** = 10, then after calling your procedure ***S*** becomes "HWdellloor", and this reversed ***S*** should be printed out. (**NOTE**: **S** = "H ello" and ***length*** = 6, ***S*** becomes " Hello", assuming each space will be calculated as an each length with the corresponding ASCII code).

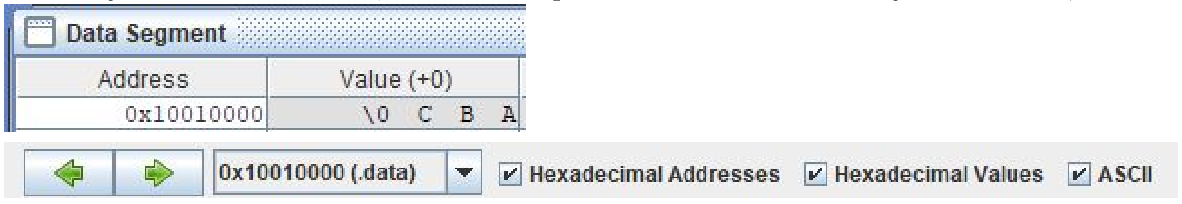
In the program, we assume the variables (e.g., ***S*** and ***length***) should be declared and initialized manually in the **.data** section. (Need to be tested by changing the ***S*** and ***length*** manually.)

The signature of this procedure in a high level language would look like this:
void string_bubble_sort(char String[], int length);

Output: for ***S*** =**"CAB"** and ***length*** = 3

```
CAB
ABC
```

With the printed **ABC**
The string ***S*** MUST have **ABC** (,with ASCII representation; the address might be different)


Data Segment

Address	Value (+0)
0x10010000	\0 C B A

NOTES: How to print Integers and Strings/space/newline etc using 'syscall'
https://courses.missouristate.edu/KenVollmar/mars/Help/SyscallHelp.html

```
        .data
x:              .word    5
msg1:           .asciiz  "x="
nl:             .asciiz  "\n"
space:          .asciiz  " "

        .text
main:
        # Register assignments
        # $s0 = x

        # Initialize registers
        lw      $s0, x          # Reg $s0 = x

        # Print msg1
        li      $v0, 4          # print_string syscall code = 4
```

```
la      $a0, msg1
syscall

# Print result (x)
li      $v0,1           # print_int syscall code = 1
move    $a0, $s0        # Load integer to print in $a0
syscall

# Print newline
li      $v0,4           # print_string syscall code = 4
la      $a0, nl
syscall

# Exit
li      $v0,10          # exit
syscall
```