Computer Architecture Assignment #4

2016320198, 백진헌

1. **Explanation of the assembly code**

   Using bubble sort, I sort 32 numbers in word format, natural unit of data.

   1) **Main**

```
la    $t0, Input_data    # Set Input_data address to $t0 register
la    $t1, Ouput_data    # Set Output_data address to $t1 register
subu $t2, $t1, $t0       # to compute how many numbers, first subtract two addresses
sra   $t2, $t2, 2        # how many numbers? shift right by 2
addu $t3, $t2, 0         # temp parameter for move input data to output (how many
                           times the loop should iterate)s
```

   2) **Move_input_to_output**

   ➢ Make descending order of the numbers at Output_data address, first move all numbers in Input_data to Output_data area

```
lw    $s0, 0($t0)        # load word at Input_data address
sw    $s0, 0($t1)        # save word at Output_data address


subu $t3, $t3, 1         # $t3 holds how many times the loop should iterate
beq   $t3, $zero, bubble_sort   # if $t3 value is equal to zero, all Input_data values are
                                  moved into Ouput_data address
addu $t0, $t0, 4         # if not, increase Input_data address by adding 4 (word size)
addu $t1, $t1, 4         # if not, increase Output_data address by adding 4 (word size)


j     move_input_to_output      # just for iterate loop, jump to begin area
```

3) **Bubble_sort**

- Initialize bubble sort (setting variables)

```
la    $t0, Ouput_data   # t0 : Output_data address storage

addu $t1, $t2, 0        # t1 : how many times, bubble sort loop 1 iterate

addu $t2, $t1, 0        # t2 : how many times, bubble sort loop 2 iterate

addu $t8, $t2, 0        # t8 : how many numbers, I have to sort
```

4) **Bubble_sort_loop_1:**

- First loop of bubble sort that repeat n times, call bubble_sort_loop_2 which sort numbers in descending order

```
la    $t0, Ouput_data   # t0 : Output_data address storage

addu $t2, $t8, 0        # t2 : how many times, bubble sort loop 2 iterate (for initialize
                        that value to size n before going to the loop 2, that says loop 2
                        iterate n times)


subu $t1, $t1, 1        # first loop decrement by 1 (control iterate count)

beq   $t1, $zero, done           # if $t1 value is equal to zero, bubble sort is finished


j      bubble_sort_loop_2        # if not, go to bubble sort loop 2 to sort numbers
```

5) Bubble_sort_loop_2

- Second loop of bubble sort that repeats n times, compares two numbers which stick together

```
subu $t2, $t2, 1          # second loop decrement by 1 (control iterate time)

beq  $t2, $zero, bubble_sort_loop_1    # if $t2 value is equal to zero, second loop of
                                         bubble sort is finished


lw   $s0, 0($t0)          # load two numbers to compare, $s0 is first number

lw   $s1, 4($t0)          # load two numbers to compare, $s1 is second number


addu $t0, $t0, 4          # change Ouput_data index


slt  $s2, $s0, $s1        # Compare two numbers

bne  $s2, $zero, swap          # if $s0, $s1 are not descending order, swap $s0, $s1 in
                                 Output_data


j    bubble_sort_loop_2        # if not, go to bubble sort loop 2 to sort numbers
```

6) Swap

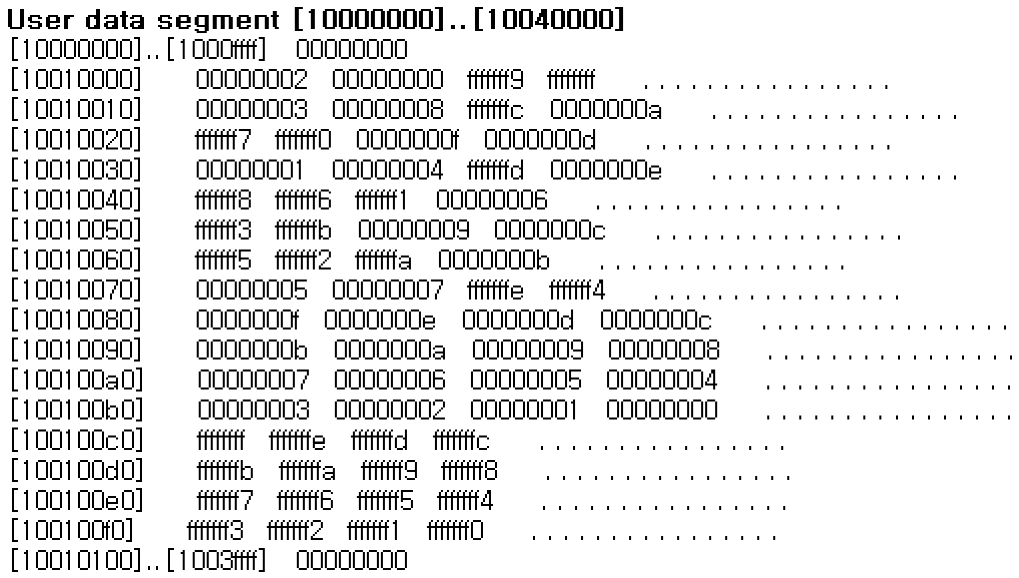
- Exchange two numbers that are not in descending order

```
addu $s3, $s0, 0          # temp number storage to swap

sw   $s3, 0($t0)          # swap two numbers in Output_data

sw   $s1, -4($t0)         # swap two numbers in Output_data

j    bubble_sort_loop_2        # after swap two numbers, go to bubble sort loop 2

                                 to sort numbers
```

7) Done

- Finishing bubble sort

2. Output screen-capture after the program execution



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000002  00000000  fffffff9  ffffffff    . . . . . . . . . . . . . . . .
[10010010]    00000003  00000008  fffffffc  0000000a    . . . . . . . . . . . . . . . .
[10010020]    fffffff7  fffffff0  0000000f  0000000d    . . . . . . . . . . . . . . . .
[10010030]    00000001  00000004  fffffffd  0000000e    . . . . . . . . . . . . . . . .
[10010040]    fffffff8  fffffff6  fffffff1  00000006    . . . . . . . . . . . . . . . .
[10010050]    fffffff3  fffffffb  00000009  0000000c    . . . . . . . . . . . . . . . .
[10010060]    fffffff5  fffffff2  fffffffa  0000000b    . . . . . . . . . . . . . . . .
[10010070]    00000005  00000007  fffffffe  fffffff4    . . . . . . . . . . . . . . . .
[10010080]    0000000f  0000000e  0000000d  0000000c    . . . . . . . . . . . . . . . .
[10010090]    0000000b  0000000a  00000009  00000008    . . . . . . . . . . . . . . . .
[100100a0]    00000007  00000006  00000005  00000004    . . . . . . . . . . . . . . . .
[100100b0]    00000003  00000002  00000001  00000000    . . . . . . . . . . . . . . . .
[100100c0]    ffffffff  fffffffe  fffffffd  fffffffc    . . . . . . . . . . . . . . . .
[100100d0]    fffffffb  fffffffa  fffffff9  fffffff8    . . . . . . . . . . . . . . . .
[100100e0]    fffffff7  fffffff6  fffffff5  fffffff4    . . . . . . . . . . . . . . . .
[100100f0]    fffffff3  fffffff2  fffffff1  fffffff0    . . . . . . . . . . . . . . . .
[10010100]..[1003ffff]  00000000
```