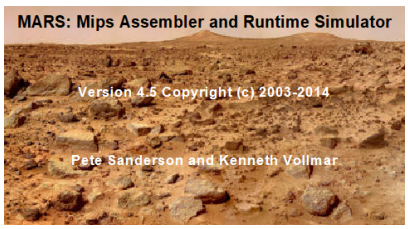
基于MIPS汇编设计冒泡程序

学号：PB17111623

姓名：范睿

模拟机：MARS 4.5



冒泡排序数据类型：有符号整型

代码解释

数据段

```
#######data segment#######
    .data
start_time:
    .asciiz "start time:"
input_num:
    .asciiz "\nplease input the number of digits in the sequence:"
input:
    .asciiz "please input the sequence(seperated by enter):\n"
read_finished:
    .asciiz "read finished!\n"
sort_start:
    .asciiz "sort start time:"
sort_finish:
    .asciiz "\nsort result:\n"
space:
    .asciiz " "
finish_time:
    .asciiz "\nfinish time:"
```

数据段保存要打印的字符串

代码段

1. 读取数据总数

```
main:
    li $v0, 4           #print “start time: ”
    la $a0, start_time
    syscall

    jal get_time        #call get_time function to print current time

    li $v0, 4           #print
    la $a0, input_num
    syscall
```

```
11
12      li $v0, 5           #get the total number of the digits in the sequence
13      syscall             #the number is stored in $v0
14
15      sw $v0, 0($gp)      #Mem[gp] stores the total number of the digits
16      sw $sp, 4($gp)      #Mem[gp+4] stores the base address of the satck
```

- get_time函数打印当前时间（具体实现在后面）
- 打印字符串的系统调用号为4（45-47）
- 读取整型的系统调用号为5，结果存入$v0中（12-13）
- 52行将读取到的（放在$v0）中的数字总数存放进$gp的内存地址
- 53行将栈底地址存放在$gp+4的内存地址

2. 读取被排序的序列

```
1       li $v0, 4           #print a string
2       la $a0, input
3       syscall
4       lw $t1, 0($gp)      #the total number of digits is loaded to t1
5   loop_read:
6       li $v0, 5
7       syscall
8       sw $v0, 0($sp)      #push the data onto the stack
9       addi $sp, $sp, 4
10      addi $t1, $t1, -1
11      bne $t1, $zero, loop_read   #if read is not finished, go to loop_read
```

- 先将数字总个数载入$t1中（4）
- 调用读取整型的系统调用，将结果入栈（8-9）
- $t1减一，若此时$t1中为0，说明读取完毕，退出循环，否则回到loop_read

3. 冒泡排序

```
1       li $v0, 4           #print a string
2       la $a0, sort_start
3       syscall
4       jal get_time        #call get_time function to print current time
5
6       addi $t1, $sp, 0    #t1 = i, loop from sp-1 -> [gp+4] (iterate the stack)
7   BiggerLoop:
8       lw $t2, 4($gp)      #t2 = j, loop from [gp+4] -> i-1
9       addi $t1, $t1, -4
10      beq $t1, $t2, done  #done when #t1==the stack's base address
11  SmallLoop:
12      lw $t3, 0($t2)
13      lw $t4, 4($t2)
14      slt $t5, $t4, $t3
15      beq $t5, $0, NoChange   #if(Mem[$t2]<= Mem[$t2+4]) NoChange
16      sw $t3, 4($t2)          #else exchange
17      sw $t4, 0($t2)
18  NoChange:
19      addi, $t2, $t2, 4
20      beq $t1, $t2, BiggerLoop
21      j SmallLoop
```

- 冒泡排序开始前先打印当前时间（4）
- $t1为外层循环的标志位，从栈顶（$sp）循环至栈底（Mem[$gp+4]）
- $t2为内层循环的标志位,从栈底（Mem[$gp+4]）循环至$t2-4
- 每次$t2更新结束后，若没有退出本次循环，对比Mem[$t2]和Mem[$t2+4]位置的数字，若前者大于后者，则交换次序
- 每次$t2更新

4. 打印排序结果

```
done:
    li $v0, 4           #print a string
    la $a0, sort_finish
    syscall

print_loop:
    li $v0, 1
    lw $a0, 0($t2)
    syscall             #read the result from the stack and print

    li $v0, 4
    la $a0, space
    syscall

    addi $t2, $t2, 4
    beq $t2, $sp, finish
    j print_loop

finish:
    la $a0, finish_time
    syscall
    jal get_time        #call get_time function to print current time
```

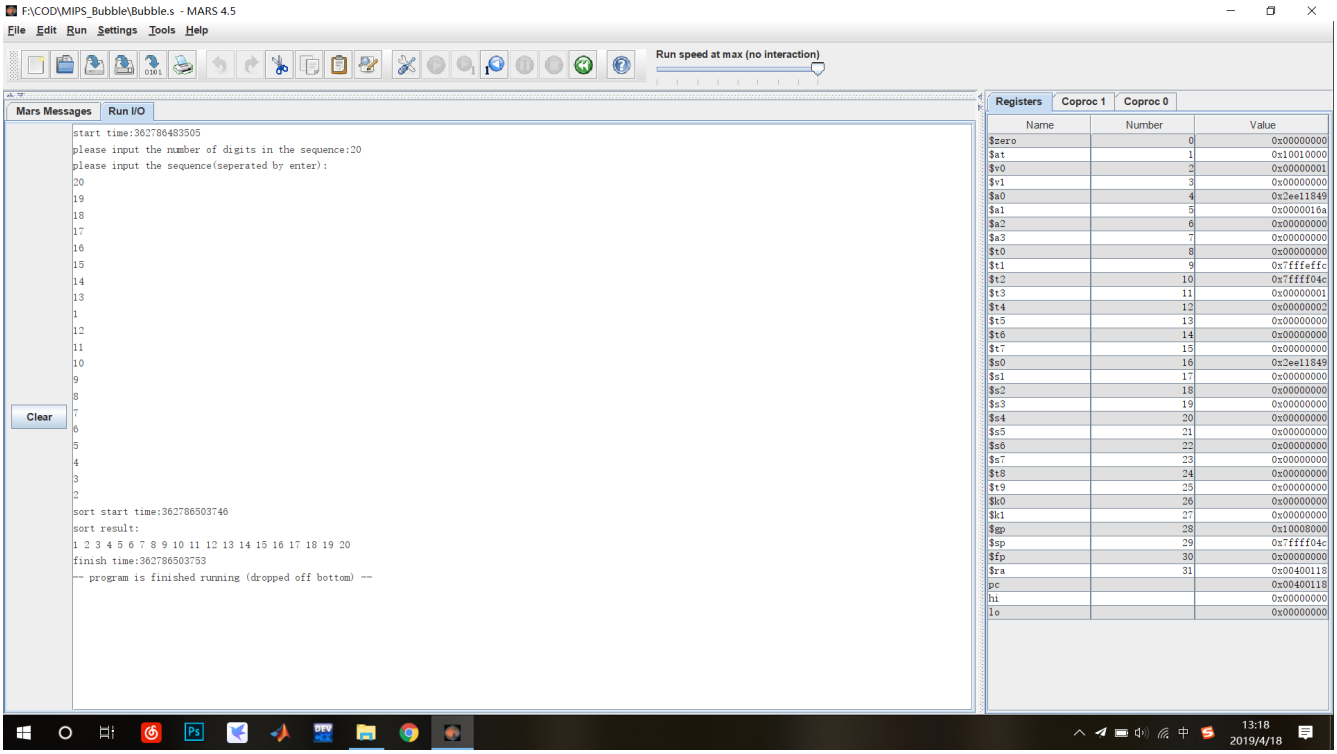
- 退出循环时，$t2一定指向栈底，将$t2从栈底循环至栈顶，打印结果

5. get_time函数

```
get_time:
    li $v0, 30          #get time
    syscall
    li $v0, 1
    add $s0, $a0, $0
    add $a0, $a1, $0
    syscall
    add $a0, $s0, $0
    syscall
    jr $ra
```

- 获取时间的系统调用号为30， 时间结果的第32位存放在$a0，高32位存放在$a1
- 打印结束后返回主函数

调试执行过程及程序执行时间



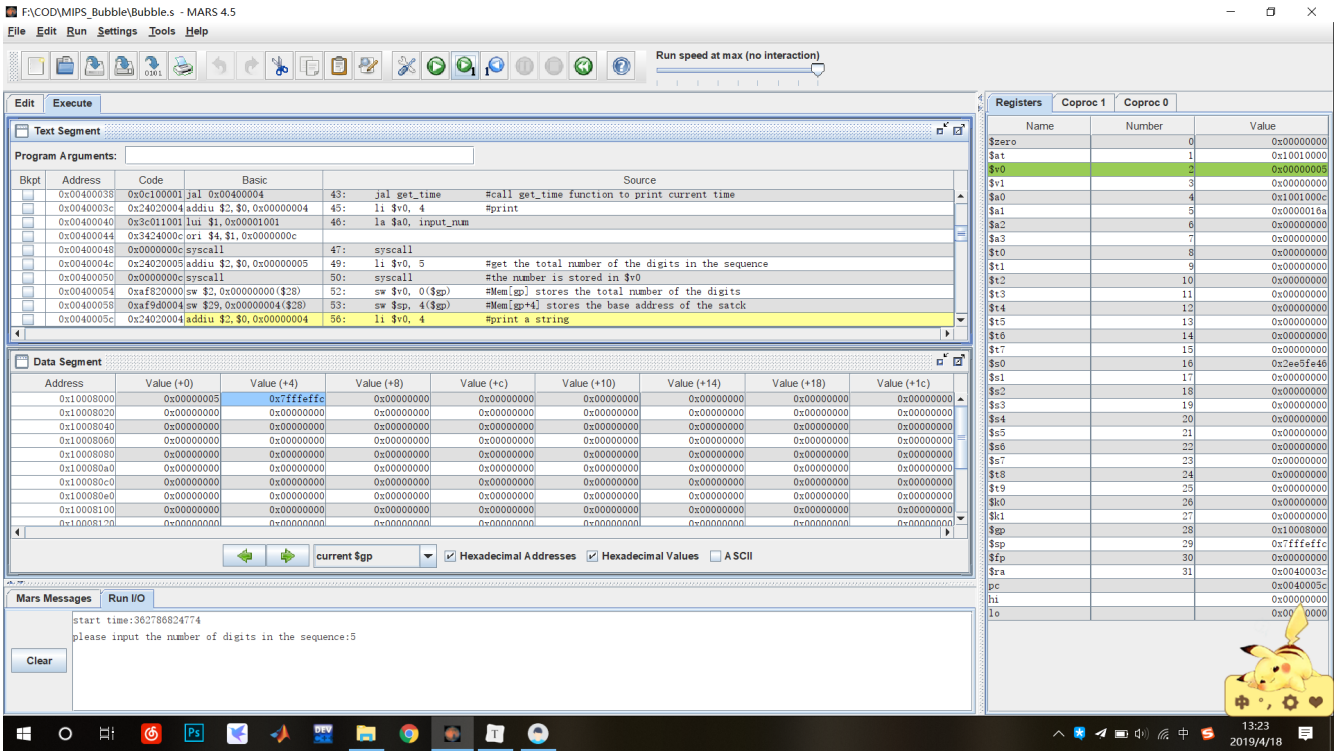
对20个数据进行排序：

程序执行总时间 =（362786503753-362786483505）ms = 20248 ms

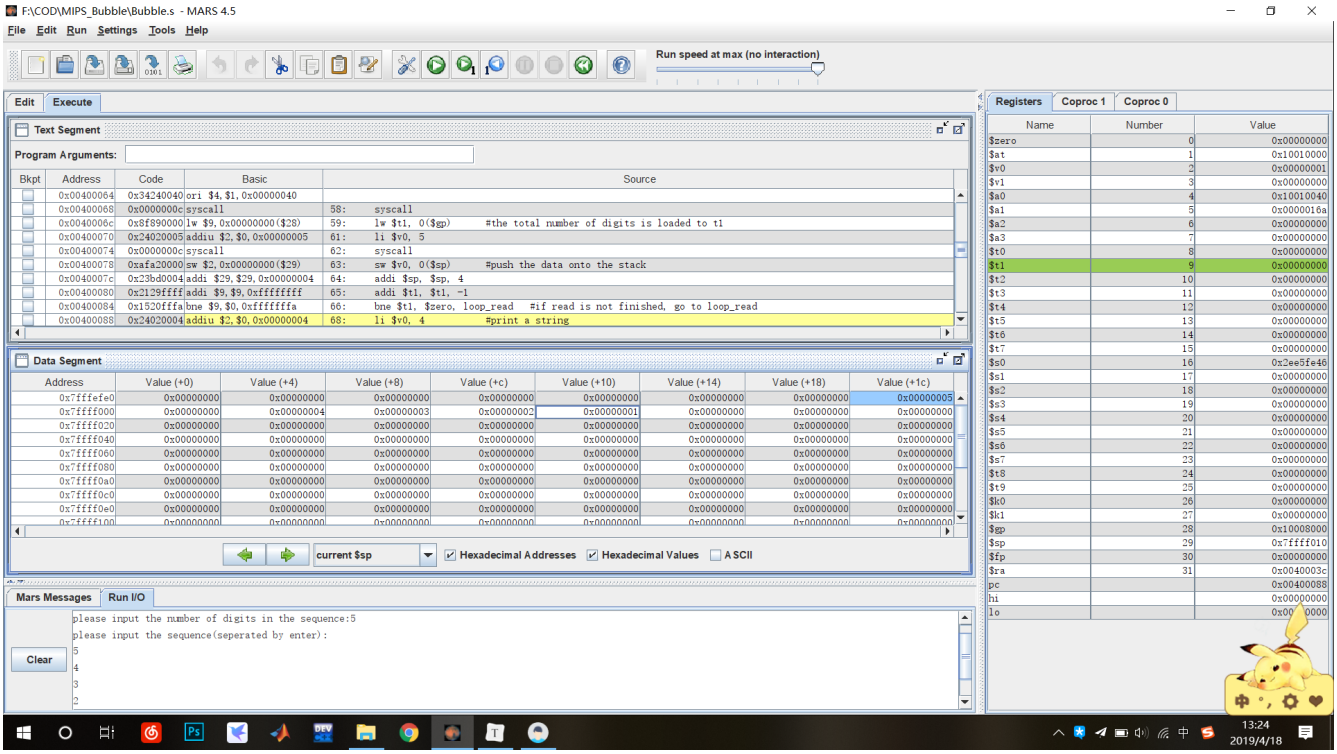
排序执行总时间 =（362786503753-362786503746）ms = 7 ms

调试过程截图

- 读入数字总数后，将总数及栈底存入$gp



- 读取全部数字序列，将其压栈（数据段中为栈中值）



- 排序完成后，栈中数据从小到大排列