基于 MIPS 汇编的冒泡排序

胡毅翔 PB18000290

摘要

本次课后实验的内容为掌握 MIPS 的基本指令，并基于 MIPS 编写冒泡排序程序，最后测量程序运行时间。

引言

本次实验的实验环境为 MARS 模拟器，在该环境下完成程序的编写及调试工作。

1 程序代码讲述

本次实验所需完成的功能为冒泡排序，程序设计思路为，通过系统调用获得数组长度 n 及数组元素 v[0...n-1]的值。随后进入 sort 模块，并提供系统调用获得排序正式开始前的时间。而后通过交换，每次把一个元素调整至有序（从大到小依次处理，故排序结束后的数组为单调递增的有序数组）。

在退出排序前，再次进行系统调用，获取时间，进而得到排序程序运行时间，最后依次输出数组元素，排序程序运行时间及其单位（ms）。完整代码如下：

```
# BubbleSort
.data

.text
.globl main
main:
        li $v0,5            # 系统调用得到 n
        syscall             # get n
        move $t7,$v0        # $t7 = n
        move $s0,$zero      # i = 0
        move $s2,$zero
        li   $s3,0x10010000 # 数组 v[]的地址
                            # for(i=0;i<n;i++) scanf("%d",&v[i])
for1:                       # get v[0 ... n-1]
        slt  $s1,$s0,$t7    # if(i < n) $s1 = 1 else $s1 = 0
        beq  $s1,$zero,sort
        addi $s0,$s0,1      # i++
        li   $v0,5
        syscall             # get v[i]
        sw   $v0,0($s3)     # Memory[0+$s3] = v[i]
        addi $s3,$s3,4      # $s3 = $s3 + 4
```

```
        j    for1
sort:
        li   $v0,30          # 系统调用获得排序开始时的时间
        syscall              # get start time $a1,$a0
        move $t6,$a0         # $t6 <- $a0
        move $t5,$a1         # $t5 <- $a1
        li   $a0,0x10010000
        move $a1,$t7         #  $a1 = n
        move $s2,$a0         #  $a0 = v
        move $s3,$a1         #  $s3 = n
        move $s0,$s3         #  $s0 = i = n
        addi $s0,$s0,-1      #  i = n - 1
                             #  for(i=n-1;i>0;i--)
forlist1:
        slt  $t0,$zero,$s0   # if(0 < i) $t0 = 1 else $t0 = 0
        beq  $t0,$zero,exit0
        li   $s1,0           # $s1 = j = 0
        move $t2,$a0         # $t2 = v
                              # for(j=0;j<i;j++)
forlist2:
        slt  $t0,$s1,$s0     # if(j < i) $t0 = 1 else $t0 = 0
        beq  $t0,$zero,exit1
        lw   $t3,0($t2)      # $t3 = v[j]
        lw   $t4,4($t2)      # $t4 = v[j+1]
        slt  $t0,$t4,$t3     # if(v[j+1] < v[j]) $t0 = 1 else $t0 = 0
        beq  $t0,$zero,exit2
        sw   $t3,4($t2)      # swap
        sw   $t4,0($t2)

exit2:
        addi $s1,$s1,1       # j++
        addi $t2,$t2,4       # v += 4
        j    forlist2
exit1:
        addi $s0,$s0,-1      # i--
        j    forlist1
                             # 系统调用获得排序结束时的时间
exit0:
        move $s0,$t7         # i=n
        li   $v0,30          # get end time $a1,$a0
        syscall
        move $t9,$a0         # $t9 <- $a0
        move $t8,$a1         # $t8 <- $a1
        li   $a0,0x10010000
```

```
        move $t2,$a0          # $t2 = v
for2:
        slt  $t0,$zero,$s0    # if(0 < i) $t0 = 1 else $t0 = 0
        beq  $t0,$zero,end    # if(0 = i) jump to end
        li   $v0,1            # print v[n-i-1]
        lw   $a0,0($t2)
        syscall
        li   $v0,11
        li   $a0,0x00000020   # print ' '
        syscall
        addi $t2,$t2,4        # v += 4
        addi $s0,$s0,-1       # i--
        j    for2
end:    sub  $a0,$t9,$t6      # end time - start time
        li   $v0,1            # print (end time - start time)
        syscall
        li   $a0,0x0000006d   # print 'm'
        li   $v0,11
        syscall
        li   $a0,0x00000073   # print 's'
        li   $v0,11
        syscall
```

2 程序运行

2.1 汇编

编写完成 Bubblesort.asm 的代码后，点击 Aessmble 开始调试，如图 1.

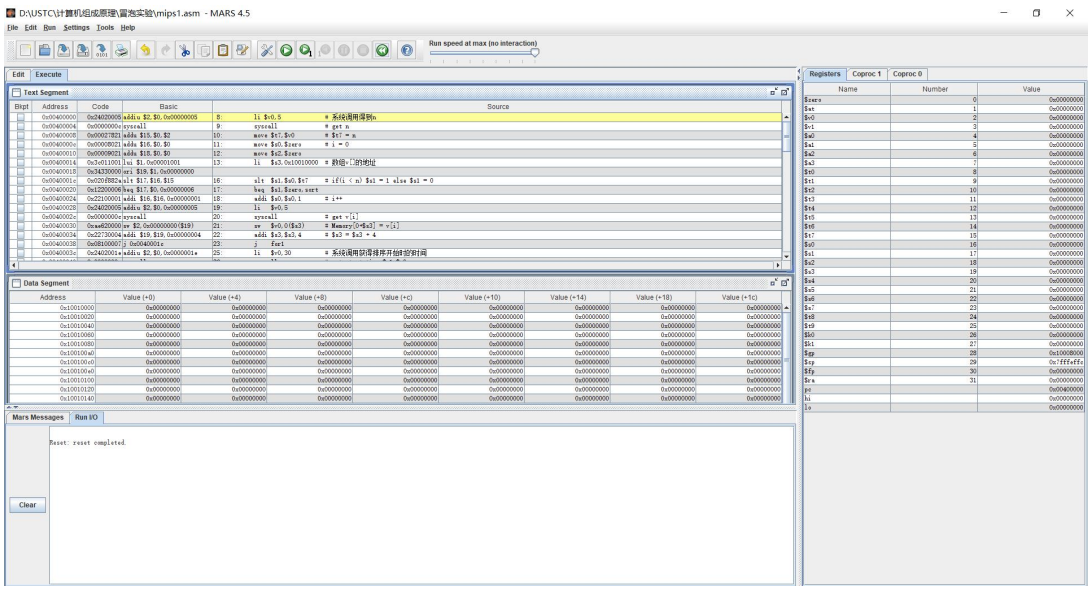

图 1

2.2 输入

系统系统调用时，在 I/O 界面输入，数组长度，及数组元素的值，如图 2.

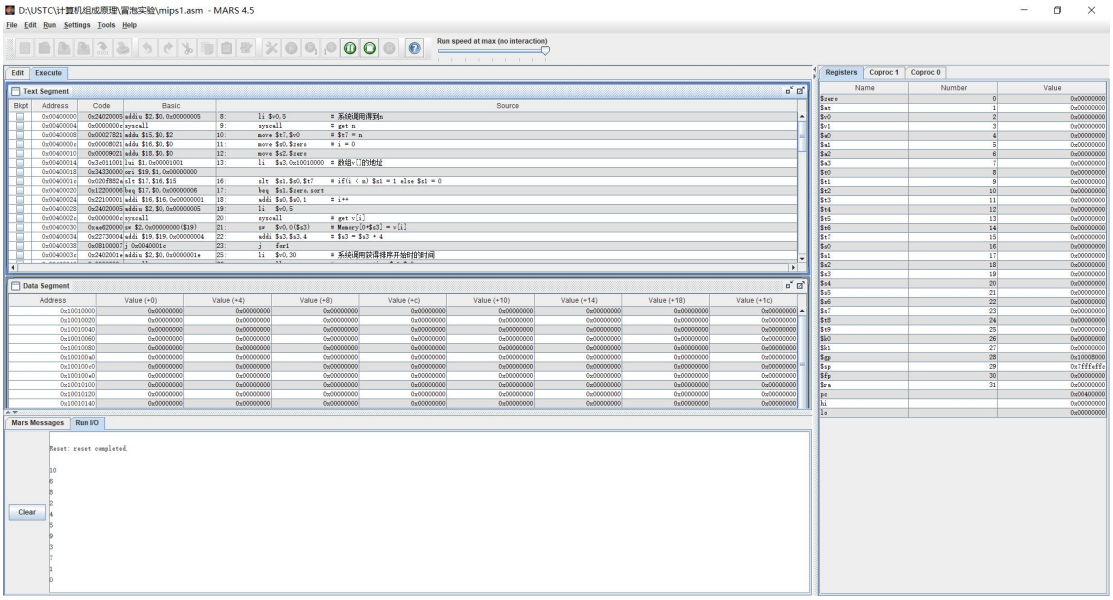

图 2

2.3 输出

冒泡排序结束后，在 I/O 界面输出，排序后的数组及排序部分的运行时间，如图 3.

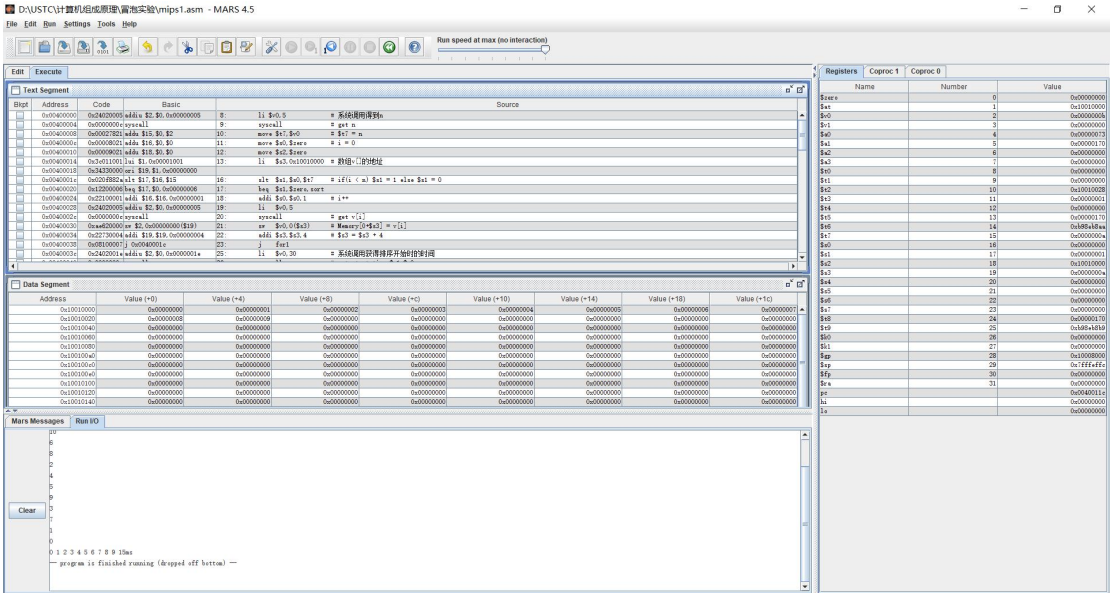

图 3

3 时间测量及结果分析

本次实验利用系统调用的时间进行程序运行时间分析，每次调用得到的时间为自 GMT 1970-01-01 00:00:00 起到此时的毫秒数，分别保存在$a1，$a0（高 32 位，低 32 位）。

3.1 排序部分时间测量

本次实验的时间测量，在数组长度为 10，20，30，及数组元素单调递增，单调递减，无序，3×3 共 9 种情况下完成，每种情况进行 3 次试验去平均值。结果如下：

Δt(ms)	N=10	N=20	N=30
递增	<1	31	77.67
递减	15.67	52	109.67
其他	15.67	41.67	93.67

表 1

3.2 时间测量中遇到的问题

在时间测量过程，发现时间测量的结果具有如下特征：

1. 同一数据的测量的结果并不稳定，有的数据存在 15-16ms 的误差。

2. 得到的测量结果之间的间隔有一定规律性，约为 15-16ms。

3.3 结果分析

实验结果基本符合预期，单调递增的数据排序所需时间最短，单调递减的数据排序顺序时间最长，其他情况则介于两者之间。对于时间间隔有一定的规律性，可能原因为系统调用获取时间有窗口期，若第一次调用后，第二次调用介于窗口期之间，则需等待约 15ms，导致实验结果的规律性。

4 结论

本次实验通过基于 MIPS 的汇编，完成了冒泡排序程序，并提供系统调用获取时间的方式，对程序运行时间进行了测量和分析，得到的结果与预期基本一致。

5 参考文献

[1]A MIPS Assembly Language Simulator Designed for Education. Ken Vollmar and Pete Sanderson. Journal of Computing Sciences in Colleges, 21:1, October 2005. Pages: 95 - 101.

[2]MARS: An Education-Oriented MIPS Assembly Language Simulator, Kenneth Vollmar and Pete Sanderson. ACM SIGCSE Bulletin, 38:1 (March 2006), 239-243.

[3]Tutorial on MARS at CCSC-CP, Drury University, Apr. 13-14, 2007, by Pete Sanderson and Ken Vollmar.