基于 MIPS 汇编的冒泡排序程序

姓名：廖洲洲　学号：PB17081504　实验日期：2019-4-19

一、实验题目：

基于 x86 或 MIPS 汇编，设计一个冒泡排序程序

二、实验目的：

使用 MIPS 汇编语言编写一个冒泡排序程序，并对其进行调试

三、实验平台：

MARS 4.5

四、实验过程：

1. 写出冒泡排序的 c 语言代码

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
int main(){
            int num;
            printf("Please input the number of Numbers to sort:\n");
            scanf("%d",&num);
            int *v;
            int i,j,temp;
            v=(int *)malloc(num*sizeof(int));
            printf("Please input the Numbers:\n");
            for(i=0;i<num;i++)
                    scanf("%d",&v[i]);

            for(i=0;i<num;i++)
                    for(j=0;j<num-i-1;j++)
                            if(v[j]<v[j+1]){
                                    temp=v[j];
                                    v[j]=v[j+1];
                                    v[j+1]=temp;
                            }
            printf("After sorted:\n");
            for(i=0;i<num;i++)
                    printf("%d ",v[i]);
}
```

2. 编写汇编指令（关键代码）

- 输出调试

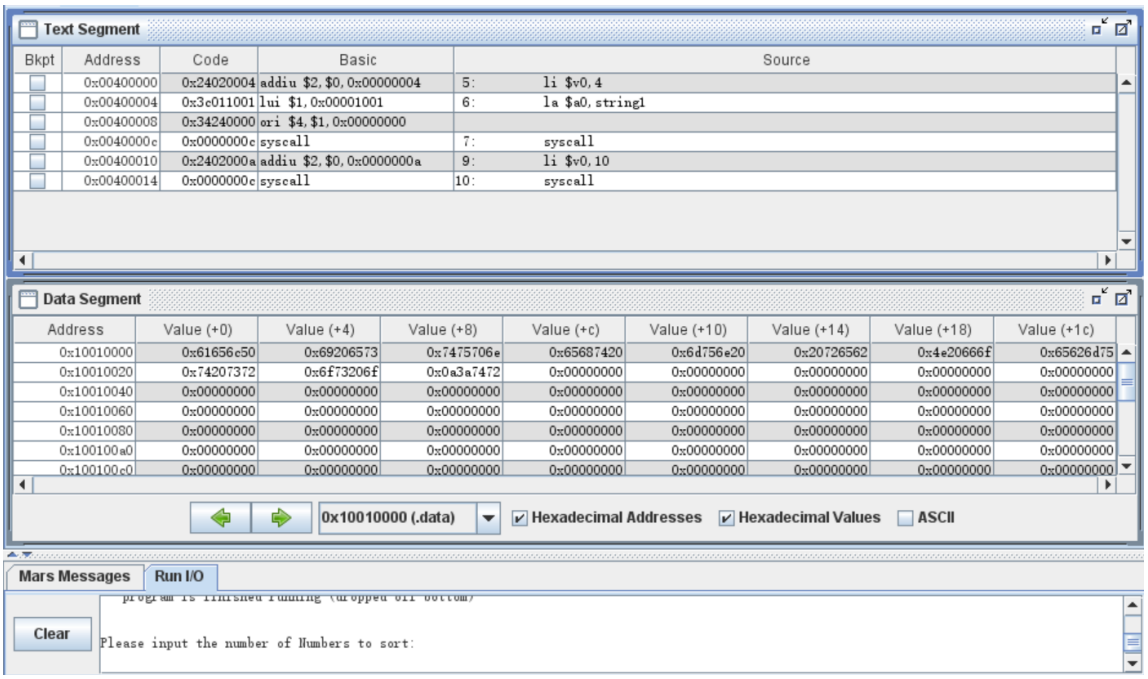
```
.data
string1 : .asciiz "Please input the number of Numbers to sort: \n"
.text
.globl main
main:
la $a0 , string1
li $v0 , 4
syscall  #系统调用输出提示
```

刚开始时未将 string1 放入.data 段，结果报错，明白了数据段不能放入代码段，修正后成功输出。

报错

```
Assemble: assembling C:\Users\廖洲洲\Desktop\mips2.asm

Error in C:\Users\廖洲洲\Desktop\mips2.asm line 3 column 10: ".asciiz" directive cannot appear in text segment
Assemble: operation completed with errors.
```

成功



- 分配内存

  要通过

  array:.space 128

  和 la $t6 , array　　# $t6 是数组首地址

  来获得相应内存

- 输入调试

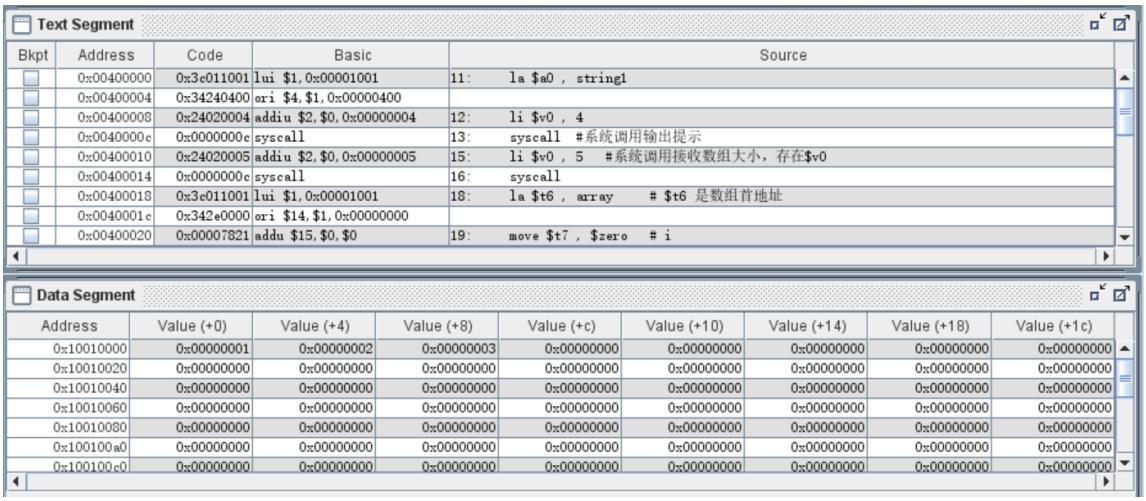
  在这里，要通过系统调用来接收数据，然后再将数据存在相应的寄存器，

  分别用了$t7,$t8,$t9 存储了 i，num，j，其中$t6 存储了数组首地址

```
    li $v0 , 5   #系统调用接收数组大小，存在$v0
    syscall

    la $t6 , array     # $t6 是数组首地址
    move $t7 , $zero   # i
    move $t8 , $v0       # num
    move $t9 , $zero   # j

input:                # 数组元素的输入
    la $a0 , string2
    li $v0 , 4
    syscall   # 打印字符串，提示用户输入数组的元素

    li $v0 , 5
    syscall

    move $t0 , $t7
```

```
    mul $t0 , $t0 , 4     # $to=i*4
    addu $t1 , $t0 , $t6 #得到新地址
    sw $v0 , 0($t1)

    addi $t7,$t7,1
    blt $t7,$t8,input    #i 和 num 不相等，继续输入
     move $t7,$zero  # 输入结束将循环变量置为 0
```

调试输入 1 2 3 后可以看到内存中存储了 1 2 3



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	11: la $a0 , string1
	0x00400004	0x34240400	ori $4,$1,0x00000400	
	0x00400008	0x24020004	addiu $2,$0,0x00000004	12: li $v0 , 4
	0x0040000c	0x0000000c	syscall	13: syscall #系统调用输出提示
	0x00400010	0x24020005	addiu $2,$0,0x00000005	15: li $v0 , 5 #系统调用接收数组大小，存在$v0
	0x00400014	0x0000000c	syscall	16: syscall
	0x00400018	0x3c011001	lui $1,0x00001001	18: la $t6 , array # $t6 是数组首地址
	0x0040001c	0x342e0000	ori $14,$1,0x00000000	
	0x00400020	0x00007821	addu $15,$0,$0	19: move $t7 , $zero # i

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

- 排序

通过内外两层对内存中的数据进行冒泡排序，本排序将大的排在前

```
loop1:
    move $t9,$zero    # 第一层循环，$t9 为 j，执行完后赋值为 0
loop2:
    move $t0,$t9      # a[j]
    mul $t0,$t0,4
    addu $t1,$t0,$t6
    lw $t2,0($t1)

    addi $t0,$t9,1    # a[j+1]
    mul $t0,$t0,4
    addu $t4,$t0,$t6
    lw $t3,0($t4)

    bge $t2,$t3,skip  # 如果 a[j] > a[j+1],不交换,否则交换两者的值
    sw $t3,0($t1)
    sw $t2,0($t4)

skip:
   addi $t9,$t9,1   # j=j+1
   addi $t0,$t9,1
   sub $t1,$t8,$t7
   blt $t0,$t1,loop2  # 如果满足，则跳转到 loop1
   addi $t7,$t7,1     # 如果不满足，则不跳转，继续执行下面的代码
   sub $t2,$t8,1
   blt $t7,$t2,loop1
```

- 排序后输出

仍需要一个循环，然后系统调用将内存中的数依次输出

```
  li $v0,4
   la $a0,string3  # After sort
```

```
    syscall

    move $t7,$zero   # i=0

print:          # 输出
    move $t0,$t7
    mul $t0,$t0,4
    addu $t1,$t0,$t6
    lw $a0,0($t1) #v[i]
    li $v0,1
    syscall

    la $a0,string4
    li $v0,4
    syscall

    addi $t7,$t7,1
    blt $t7,$t8,print   # 如果满足循环条件，跳转到 print 继续执行循环
```

- 加入时间测试

  刚开始把初始时间记录设置程序的开头，发现时间过长，可能输入时耗费的时间也记录在了其中，故把初始时间的记录设置在实现冒泡的双重循环之前。

  记录初始时间

```
    li $v0,30
    add $a2,$a1,$zero
    syscall

    li     $v0,    30
    syscall                              #record system time
    add    $s2,    $a0,    $zero  #$s0=$a0=low 32 bit of system time
    add     $s3,     $a1,  $zero
```

  记录结束时间

```
    li $v0,30
    syscall                      #record system time
    add $s4,$a0,$zero     #$s0=$a0=low 32 bit of system time
    add $s5,$a1,$zero
```

五、实验结果：

```
Please input the number of Numbers to sort:
5
Please input the numbers:
3
Please input the numbers:
1
Please input the numbers:
2
Please input the numbers:
3
Please input the numbers:
4
After sort :
4 3 3 2 1
— program is finished running (dropped off bottom) —
```

输入 3 1 2 3 4 输出 4 3 3 2 1（未加时间）

加时间后：

```
Please input the number of Numbers to sort:
5
Please input the numbers:
1 2 3 4 5
Reset: reset completed.

Please input the number of Numbers to sort:
5
Please input the numbers:
1
Please input the numbers:
2
Please input the numbers:
3
Please input the numbers:
4
Please input the numbers:
5
After sort :
5 4 3 2 1
starttime:362910028780
finishtime:362910028781
duration:1
— program is finished running (dropped off bottom) —
```

输入 1 2 3 4 5 输出 5 4 3 2 1 时间 1

```
Please input the numbers:
1
Please input the numbers:
2
Please input the numbers:
3
Please input the numbers:
4
Please input the numbers:
5
Please input the numbers:
6
Please input the numbers:
7
Please input the numbers:
8
Please input the numbers:
9
Please input the numbers:
10
After sort :
10 9 8 7 6 5 4 3 2 1
starttime:362910121968
finishtime:362910121971
duration:3
— program is finished running (dropped off bottom) —
```

输入 1 2 3 4 5 6 7 8 9 10 输出 10 9 8 7 6 5 4 3 2 1

时间 3

六、心得体会：

通过本次实验，我发现汇编代码的编写实在是复杂，它需要你自己去精确控制每个寄存器，因此用汇编来写程序相对于 c 语言需要更多的耐心和细心。但是这次作业确实让我了解了更多汇编指令，初步熟悉了汇编程序的编写。