



# 《计算机组成原理实验》 实验报告

(实验一)

学院名称：数据科学与计算机学院

专业（班级）：17 计教学 3 班

学生姓名：郑康泽

学号：17341213

时间：2018 年 10 月 15 日

**成 绩：**

---

## **实验一：MIPS汇编语言程序设计实验**

---

### **一. 实验目的**

- (1) 初步认识和掌握MIPS汇编语言程序设计的基本方法；
- (2) 熟悉PCSpim模拟器的使用。

### **二. 实验内容**

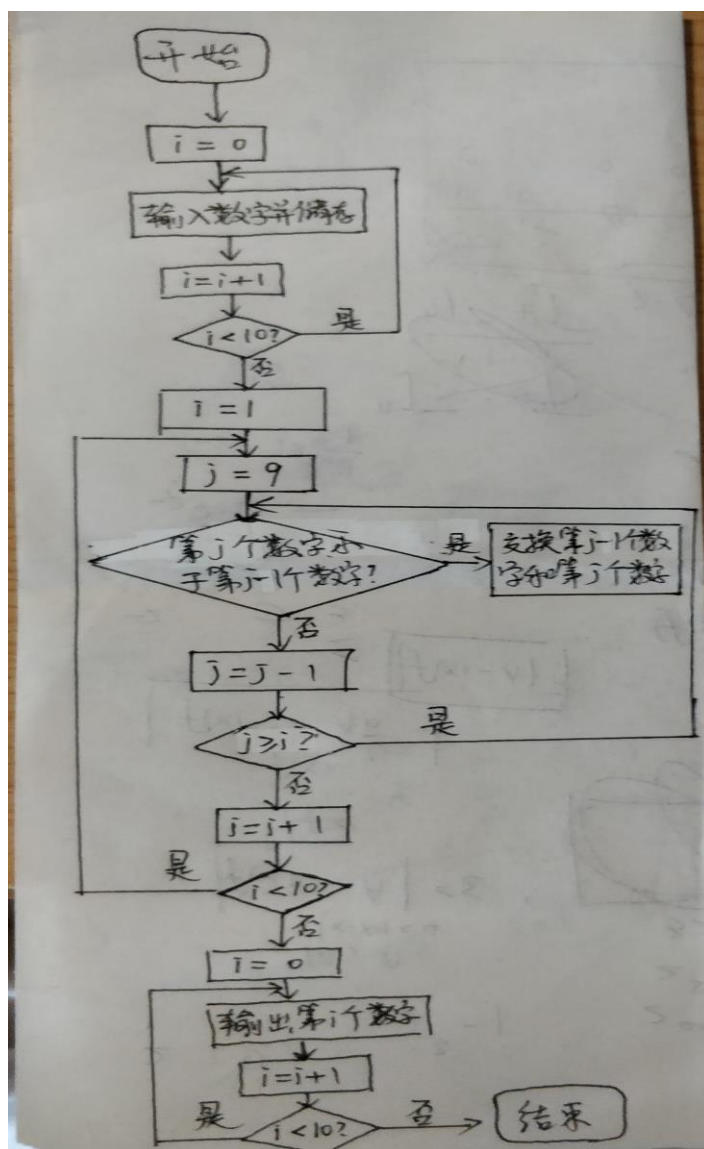
从键盘输入 10 个无符号字数或从内存中读取 10 个无符号字数并从大到小进行排序，排序结果在屏幕上显示出来。

### **三. 实验器材**

电脑一台，PCSpim仿真器软件一套。

### **四. 实验过程与结果**

- (1) 程序流程图：



## (2) 设计的思想与方法:

设计思想相对简单，只需完成输入数字并储存，然后冒泡排序，最后输出结果。

设计方法可以根据C语言来转换成MIPS语言，我自己就是照着C打的MIPS。

```
for (int i=0; i<10; ++i)
```

```
    scanf("%d", &a[i]);
```

```
// 输入10个无符号数
```

```
for (int i=1; i<10; ++i)
```

```
// 开始冒泡排序
```

```
    for (int j=9; j>=i; --j)
```

```
        if (a[j] > a[j-1]) swap(a[j], a[j-1])
```

```
for (int i=0; i<10; ++i)
```

```
// 输出结果
```

```
    printf("%d ", a[i]);
```

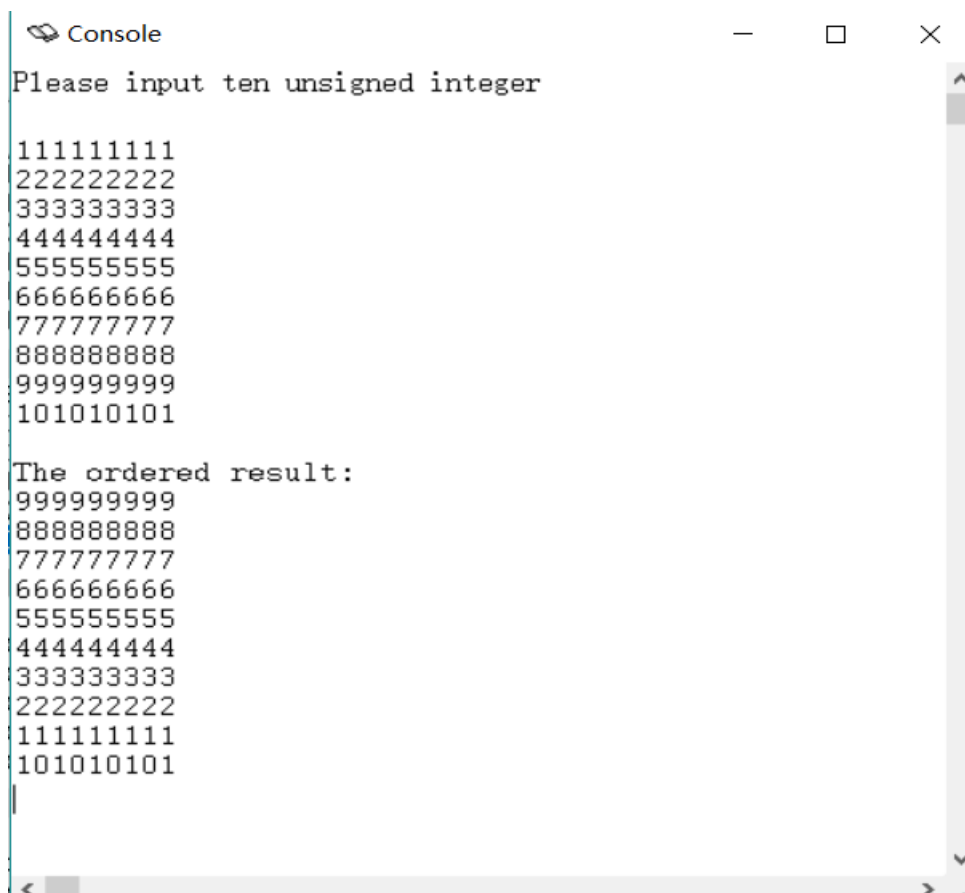
(3) 分析：数组的地址是需要有寄存器记录的，10也需要寄存器记录，其他变量可以

用temporary寄存器记录，因为随时会变。对于不变的地址值和个数10，用\$s寄存器记录。对于loop内的跳转要小心，条件一定要正确。

#### (4) 实验步骤:

输入第一个无符号数—>将数字放进内存—>输入第二个无符号数—>将数字放进内存……重复10次，就把10个数字放进了内存—>第一次循环，把最大的数放在数组的首位—>第二次循环，把第二大的数放在数组的第二位……重复9次，就把数字按从大到小排好了一—>按内存地址从小到大的顺序输出数字

#### (5) 实验结果及分析:



```
Console
Please input ten unsigned integer

111111111
222222222
333333333
444444444
555555555
666666666
777777777
888888888
999999999
101010101

The ordered result:
999999999
888888888
777777777
666666666
555555555
444444444
333333333
222222222
111111111
101010101
```

输入的十个无符号数是：111111111，222222222，333333333，444444444，555555555，666666666，777777777，888888888，999999999，101010101。按照从大到小的顺序，应该是：999999999，888888888，777777777，666666666，555555555，444444444，333333333，222222222，111111111，101010101。显然，输出结果正确。其他测试结果也是正确。

## 五. 实验心得

(1) 我遇到的第一个问题就是PCSpim模拟器给数组分配地址的值不满足字对齐。一开始我没有发现这个问题，一直在检查我的代码有没有错误，最后发现地址值有问题，只好百度怎么解决，查到的解决方法是在 `array: .space 40` 中间插入 `.align 2`，即变成 `array: .align 2 .space 40`。这样就解决了字对齐的问题，但是这个 `.align 2` 怎么使得字对齐，并不很懂。

(2) 我遇到的第二个问题是排序排错了，但最后看代码看出错误了，原来是跳转去 `swap` 代码段后忘了跳回 `loop` 代码段，所以跳转要小心，有始有终，跳出去也要跳回来。

(3) 本次实验，实现了冒泡排序，感觉比高级语言难写的多，并且对内存地址也要做到心中有数，不像其他高级语言自动帮你分配地址，也不会出现字对不齐这个问题（或许）。同时，也训练了跳转指令的使用以及 `syscall` 系统功能调用，对MIPS指令更熟悉了。不过PCSpim模拟器有个好处，就是可以直接看到内存和寄存器里存储的值，`debug` 更简单了。（不然我永远不会知道分配的地址不满足字对齐）

#### 【程序代码】

```
.text
.globl main
main:
    la $a0, str1
    li $v0, 4
    syscall

    la $a0, nline
    li $v0, 4
    syscall

    add $t0, $zero, $zero    # int i
    addi $s0, $zero, 10     # int n = 10
    la $s1, array           # the address of data
```

loop1:

    sll \$t1, \$t0, 2

    add \$t2, \$t1, \$s1

    li \$v0, 5

    syscall

    sw \$v0, 0(\$t2)

    addi \$t0, \$t0, 1

    slt \$t3, \$t0, \$s0

    beq \$t3, \$zero, next1

    j loop1

next1:

    la \$a0, nline

    li \$v0, 4

    syscall

    la \$a0, str2

    li \$v0, 4

    syscall

    la \$a0, nline

    li \$v0, 4

    syscall

    addi \$t0, \$zero, 1     # int i = 1

loop2:

```
addi $t1, $zero, 9    # int j = 9
```

loop3:

```
sll $t2, $t1, 2        # the offset of array[j]
addi $t3, $t2, -4      # the offset of array[j-1]
add $t4, $t2, $s1      # the address of array[j]
add $t5, $t3, $s1      # the address of array[j-1]
lw $s2, 0($t4)
lw $s3, 0($t5)
sltu $t6, $s2, $s3
beq $t6, $zero, swap
```

next2:

```
addi $t1, $t1, -1
slt $t6, $t1, $t0
beq $t6, $zero, loop3
```

```
addi $t0, $t0, 1
slt $t6, $t0, $s0
beq $t6, $zero, next3
j loop2
```

swap:

```
sw $s3, 0($t4)
sw $s2, 0($t5)
j next2
```

next3:

```
add $t0, $zero, $zero
```

loop4:

    sll \$t1, \$t0, 2

    add \$t2, \$t1, \$s1

    lw \$a0, 0(\$t2)

    li \$v0, 1

    syscall

    li \$v0, 4

    la \$a0, nline

    syscall

    addi \$t0, \$t0, 1

    slt \$t3, \$t0, \$s0

    beq \$t3, \$zero exit

    j loop4

exit:

    li \$v0, 10

    syscall

.data

str1:

    .asciiz "Please input ten unsigned integers\n"

str2:

    .asciiz "The ordered result: "

nline:

    .asciiz "\n"



**array:**

**.align 2**

**.space 40**