



《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）：16 计算机类 3 班

学生姓名：王凯祺

学号：16337233

时间：2017 年 10 月 12 日

成绩：

实验一：MIPS汇编语言程序设计实验

一. 实验目的

1. 掌握 MIPS 汇编语言（包括数据类型、寄存器、算术运算指令、程序控制指令、系统中断等）。
2. 能使用 MIPS 设计出一个简单的程序。

二. 实验内容

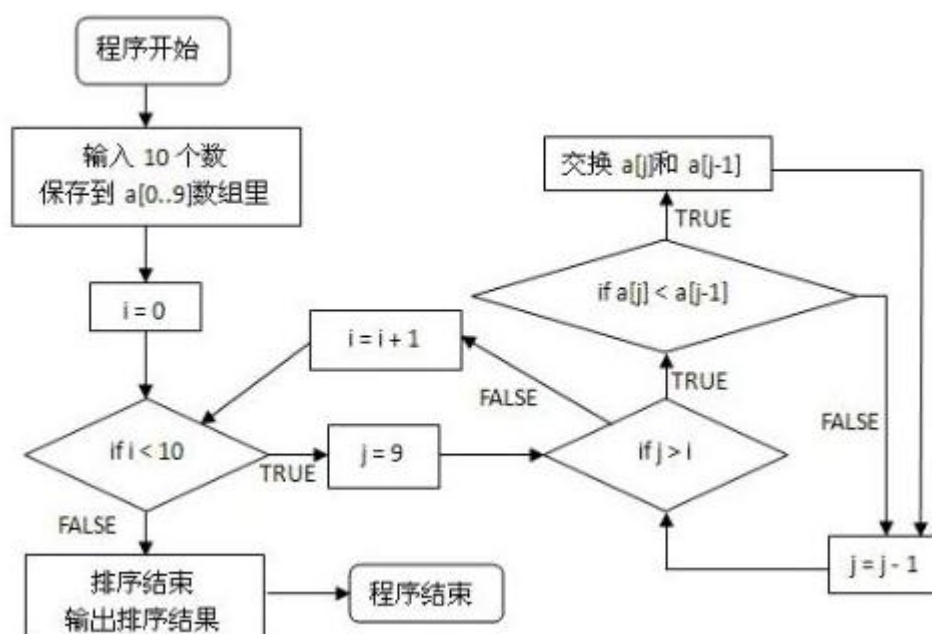
从键盘输入10个无符号字数或从内存中读取10个无符号字数并**从大到小**进行排序，排序结果在屏幕上显示出来。

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



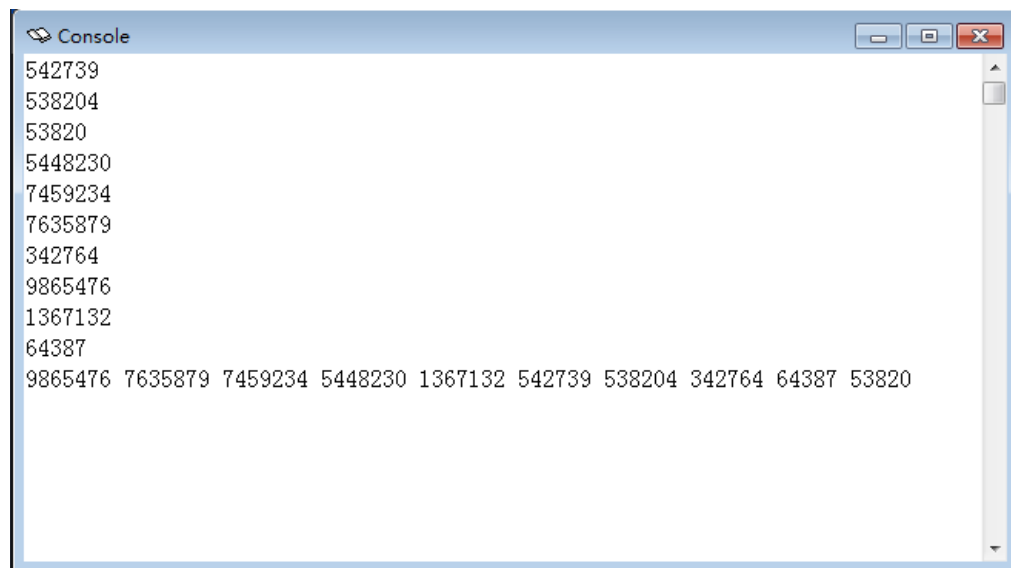
2. 设计的思想与方法

用冒泡排序来实现，每次从左到右扫描，若当前数小于右边的数，则交换，重复10次。这样第k次移动后，移到第11-k位置的数的都是前面11-k个数中最小的数。

3. 实验步骤

编写代码，编写一部分就调试一部分，查看寄存器的值是否正确，若正确则继续编写剩余的代码，否则查错直到程序正确运行。

4. 实验结果



```
Console
542739
538204
53820
5448230
7459234
7635879
342764
9865476
1367132
64387
9865476 7635879 7459234 5448230 1367132 542739 538204 342764 64387 53820
```

程序能正确运行，正确输出结果。

五. 实验心得

在整个实验中，最耗时间的是查指令（包括指令的符号以及用法）。查清楚指令之后，编写代码是相当简单的。在写了MIPS和x86的代码之后，我发现它们的区别太大了：

1. MIPS仅支持寄存器与寄存器进行运算，而x86允许寄存器与存储器进行运算。
2. MIPS只有代码段，而x86有四个段（数据段、堆栈段、代码段、扩展段）。
3. MIPS使用大数端存储方式，x86使用小数端存储方式。
4. MIPS没有标志位，而x86有标志位。相比之下，MIPS更加灵活，比如做一个小于的判断，MIPS可以指定任意寄存器存放返回值，而x86指定了CF，ZF等标志位存放返回值。
5. MIPS的输入输出接口更加丰富。MIPS支持输入/输出整数、浮点数、字符、字符串，而x86仅支持输入/输出字符。相比之下，使用MIPS编写汇编程序，实在方便太多。

【程序代码】

```
.text

.globl main

main:

    li $t0, 10

    la $t1, data

read:

    li $v0, 5                                # 输入整数
    syscall

    sw $v0, 0($t1)

    addi $t1, $t1, 4

    addi $t0, $t0, -1

    bne $t0, $zero, read

    li $t0, 10                                # $t0 为外循环计数器

l1:

    addi $t0, $t0, -1

    la $t1, data

    li $t2, 9                                # $t2 为内循环计数器

l2:

    addi $t3, $t1, 4

    addi $t2, $t2, -1

    lw $t4, 0($t1)                            # 取数

    lw $t5, 0($t3)

    slt $t6, $t4, $t5                        # 比较data[k]和data[k+1]

    beq $t6, $zero, dnswap                   # 若data[k]<data[k+1]交换

    move $t6, $t4                            # 交换寄存器中的数

    move $t4, $t5

    move $t5, $t6

dnswap:
```

```
sw $t4, 0($t1)                # 存数
sw $t5, 0($t3)
move $t1, $t3
bne $t2, $zero, 12
bne $t0, $zero, 11

li $t0, 10
la $t1, data
write:
lw $a0, 0($t1)                # 取数, 输出
li $v0, 1
syscall
li $a0, 32
li $v0, 11
syscall
addi $t1, $t1, 4
addi $t0, $t0, -1
bne $t0, $zero, write

li $a0, 0x0a
li $v0, 11
syscall
li $a0, 0x0d
li $v0, 11
syscall
li $v0, 10
syscall

.data
data: .word 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
```

0x00000000