《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 4 班
学 生 姓 名 ：	杨志成
学　　号	16337281
时　　间 ：	2017　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。

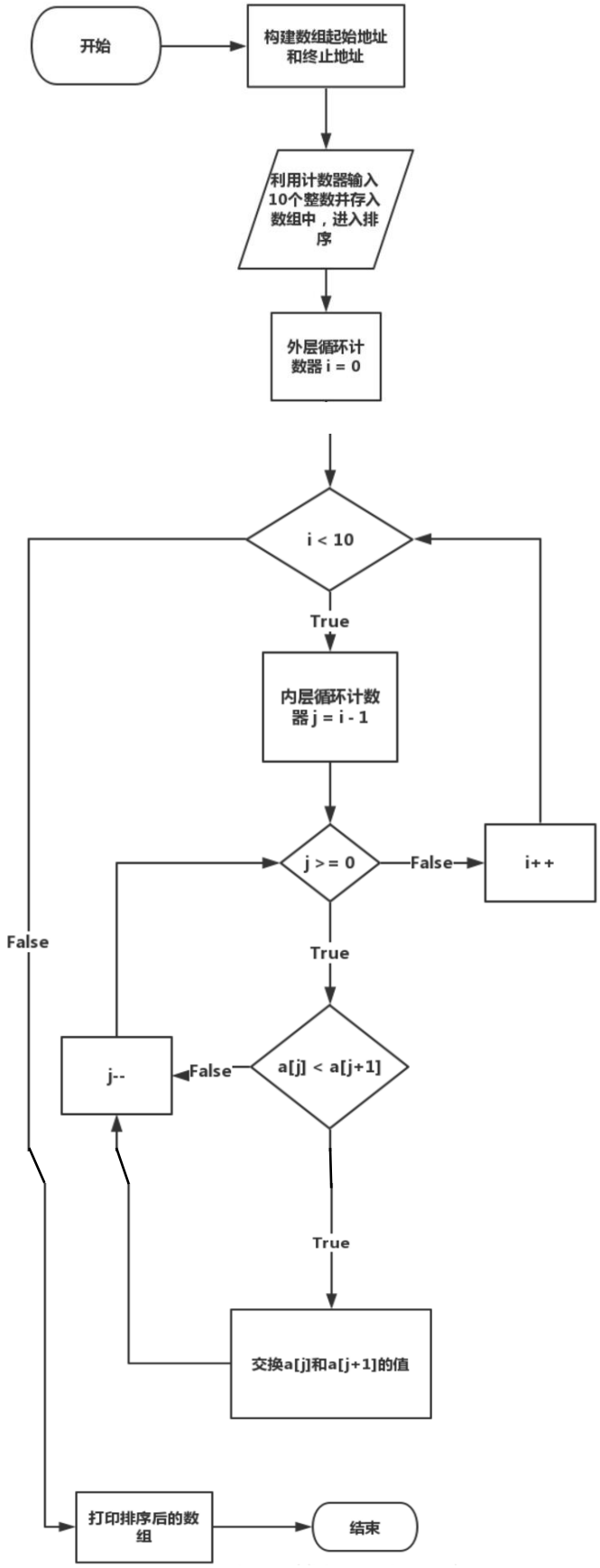
四. 实验过程与结果

1.设计思路与方法: 此题即为简单的排序题目，可将该问题分成三步: 1.输入 2.排序 3.输出

C++代码实现:

```
#include<iostream>
using namespace std;
int main()
{
    unsigned int num[10]; //申请一个长度为10的无符号整形变量空间
    int tmp;              //tmp用于交换两个数值，相当于临时变量
    //输入数组
    for(int i = 0 ; i < 10 ; i++)
        cin>>num[i];
    //接来下进行冒泡排序（从大到小）
    for(int i = 0 ; i < 9 ; i++)
        for(int j = 0 ; j < 10 - i - 1 ; j++)
            if(num[j] < num[j+1])
            {
                tmp = num[j];
                num[j] = num[j+1];
                num[j+1] = tmp;
            }
    //输出数组
    for(int i = 0 ; i < 10 ; i++)
        cout<<num[i]<<" ";
    return 0 ;
}
```

2.程序流程图：


开始
构建数组起始地址和终止地址
利用计数器输入10个整数并存入数组中，进入排序
外层循环计数器 i = 0
i < 10
True
内层循环计数器 j = i - 1
j >= 0
False
i++
False
True
a[j] < a[j+1]
False
j--
True
交换a[j]和a[j+1]的值
打印排序后的数组
结束

3. 问题分析

此题本质主要考察的应该是 MIPS 汇编实现排序算法。

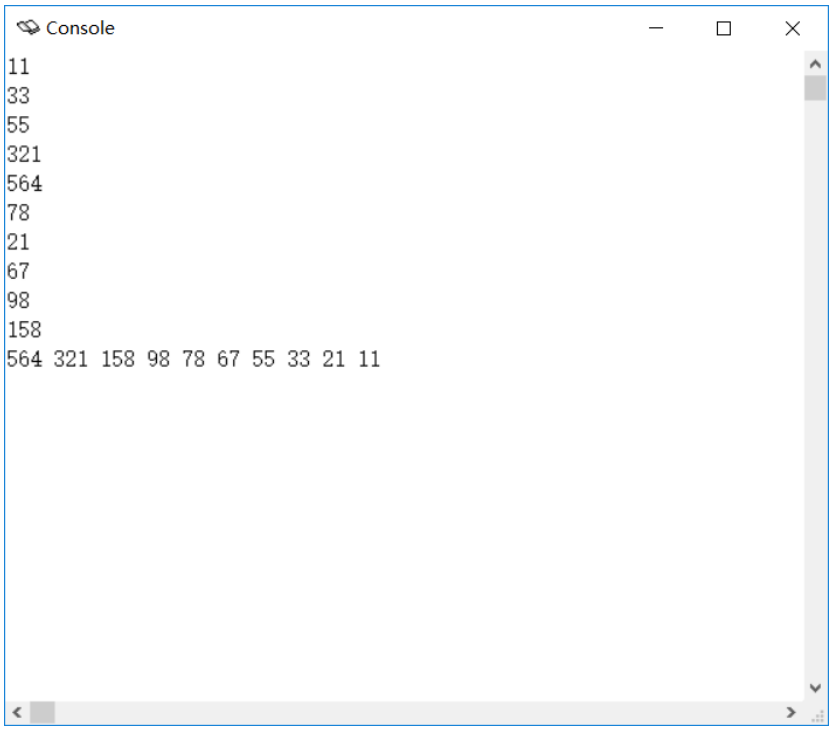
但此题还有一个需要注意的地方，即输入、排序、输出的都是无符号整形，虽然 c 语言和 c++代码在输入无符号类型时会给以区分，但是在计算机中，输入无符号整形和有符号整形是没有区别的，即存数在计算机中没有区别

但是取数时，取有符号数会将符号位填充所有剩余位。取无符号数时，需用 0 填充内存中的所有剩余位。

4. 实验步骤

将汇编代码放入 PCSPIM 中运行

随意输入 10 个无符号数字，代码会将其从大到小排序：



由图可知，实验结果正确。

5. 实验结果

五. 实验心得

这是我第一次尝试使用汇编语言解决问题，因此过程中遇到了很多问题以及自己所思考的问题。

问题1：用非叶过程实现是什么样子的?

解决这个问题时，我一开始并没有运用调用函数函数的方法，是直接在main函数中进行输入，排序，输出操作。那么如果运用了非叶过程，mips汇编代码长什么样呢?

我们知道调用非叶过程的一个解决方法就是将所有需要保留的寄存器压栈，指针sp随着栈中寄存器个数进行调整。返回时，寄存器从栈中恢复，栈指针也随着重新调整。

非叶过程的mips汇编代码放在附录当中。

非叶过程c++代码：

```cpp
#include <iostream>
using namespace std;
void swap( int * a , int * b )
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
void sort( int num[] , int n)
{
    int tmp = 0;
    for(int i = 0 ; i < n-1 ; i++)
    {
        for( int j = 0 ; j < n -1 -i ; j++ )
           if( num[j] < num[j+1] )
           {
              tmp = num[j];
              num[j] = num[j+1];
              num[j+1] = tmp;
           }
    }
}
int main()
{
    int num[10];
    for( int i = 0 ; i < 10 ; i++ )
       cin>>num[i];
    sort(num,10);
    for( int i = 0 ; i < 10 ; i++ )
       cout<<num[i]<<" ";
}
```

问题2：如何给数组分配一个地址?

一开始以为，随便用一个地址当做数组的基址，便可以顺利进行各种操作，然而后来发现，如果不在.data中用.space分配空间，那么我们是不可以在改地址上进行读写操作的。

问题3：计算机中如何存取unsigned值?

计算机中取数时，取有符号数会将符号位填充所有剩余位。取无符号数时，需用0填充内存中的所有剩余位，这种操作被称为符号位扩展。从而在取操作中，用lhu，lbu进行取半字节符号位补充。这便实现了计算机中的无符号值存取

本次实验感想：

这是我第一次计组实验，实验过程中遇到了些许困难，但这些问题最终都被解决了。在汇编语言方面，我了解的不多，这也是今后需要提高的方面之一。此外通过这次机组实验，我发现自己对计算机底层原理的了解也不充分，比如计算机是如何实现和打印unsigned类型的，代码的内存是怎么分配的，以及代码之间的调用在计算机中如何实现等等。没有接触过计算机或者对计算机不是特别了解的人可能觉得计算机特别神秘而且不知道为什么它可

以实现那么复杂的功能，而就我们而言越是深入学习越是渴望了解其工作原理。

希望自己能够在计算机组成原理实验今后的学习中，能够充分掌握计算机简易cpu的实现，对计算机的工作原理能够有进一步的认识

【程序代码】

叶子程序代码：

```
.data
    array: .space 40
    nullspace: .asciiz " "
.text
.globl main
main:
    la $a0,array          # a0储存数组的起始地址
    addi $t1,10           # t1储存数组长度
    move $t0,$zero        # t0计数，初始化为0
    move $s0,$a0          # s0记录当前输入的地址（以a0为起始地址）
    cin:
    li $v0,5              #  循环开始
    syscall
    sll $s1,$t0,2         # s1记录偏移量，s1 = t0 * 4
    add $s0,$s1,$a0       # s0 = a0 + 4*i (s0为当前输入地址)
    sw $v0,0($s0)         #  将v0中的数字存入s0表示的地址
    addi $t0,$t0,1        # t0 = t0 + 1
    bne $t0,$t1,cin       # if(t0 != t1) 跳转到cin

    addi $t2,$t1,-1       # t2 = 9
    move $t0,$zero        # t0计数(i)，初始化为0
    Loop1:

    move $t3,$zero        # for(j=0;j<9-i;j++)
    Loop2:
    sub $t4,$t2,$t0       # t4=t2-t0，储存9-i

    sll $t5,$t3,2         # t5记录偏移量
    add $t6,$t5,$a0       # t6记录当前地址
    lw $s4,0($t6)         # s4 = num[j]
    lw $s5,4($t6)         # s5 = num[j+1]
    slt $s6,$s4,$s5       # if s4<s5 ,s6=1
    beq $zero,$s6,ELSE # if s4>=s5 ,go to else
    move $t7,$s4
    move $s4,$s5
```

```
    move $s5,$t7
    sw $s4,0($t6)
    sw $s5,4($t6)
    ELSE:

    addi $t3,$t3,1        # t3++
    bne $t3,$t4,Loop2

    addi $t0,$t0,1        # t0 = t0 + 1
    bne $t0,$t2,Loop1   # if(t0 != t1) 跳转到Loop1

    move $a1,$a0          # 将a0记录的地址转移到a1
    move $t0,$zero        # t0计数，初始化为0
    cout:sll $s1,$t0,2 # s1记录偏移量，s1 = t0 * 4
    add $s0,$a1,$s1
    lw $a0,0($s0)
    li $v0,1
    syscall
    la $a0,nullspace     # a0储存空格地址
    li $v0,4
    syscall
    addi $t0,$t0,1        # t0 = t0 + 1
    bne $t0,$t1,cout     # if t0！=10 跳转到cout
```

非叶子程序代码：

```
.data
    sortarray:
        .space 40
    separate:
        .asciiz " "
    line:
        .asciiz "\n"

.text
.globl main

main:
    la $t0, sortarray              #数组起始地址
    add $t1, $zero, $t0            #指向数组起始地址
    addi $t8, $t0, 40              #数组终止地址

    addi $t3, $zero, 0             #输入计数器
    inputData:
```

```
    li $v0, 5                  #输入整型数据/read_int
    syscall
    sw $v0, 0($t1)             #存入数组

    addi $t1, $t1, 4       #指向数组下一个地址
    addi $t3, $t3, 1       #输入计数器加1
    slti $s0, $t3, 10          #计数器小于10，继续输入
    bnez $s0, inputData

    addi $t3, $zero, 0         #外层循环计数器i = 0
outLoop:
    add $t1, $zero, $t0        #每次进入排序循环，让$t1指向数组起始地址
    slti $s0, $t3, 10          #i < 10，进入内层循环
    beqz $s0, print                #i > 10, 退出循环，打印排序后的数组

    addi $t4, $t3, -1          #j = i - 1
inLoop:
    slti $s0, $t4, 0           #j < 0，退出内层循环
    bnez $s0, exitInLoop

    sll $t5, $t4, 2            #$t5 = j * 4
    add $t5, $t1, $t5          #$t5 =  数组起始地址  + j * 4
    lw $t6, 0($t5)               #$t6 = a[j]
    lw $t7, 4($t5)               #$t7 = a[j + 1]
    slt $s0, $t6, $t7        #a[j] < a[j + 1]，交换
    bnez $s0, swap
    addi $t4, $t4, -1          #j--
    j inLoop                    #继续内层循环

swap:
    sw $t6, 4($t5)               #$t6 = a[j + 1]
    sw $t7, 0($t5)               #$t7 = a[j]
    addi $t4, $t4, -1          #j--
    j inLoop                    #继续内层循环

exitInLoop:
    addi $t3, $t3, 1           #i++
    j outLoop                   #进入外层循环

print:
    lw $a0, 0($t0)               #要打印的数据存到$a0
    li $v0, 1                  #系统调用/print_int
    syscall
```

```
    la $a0, separate        #打印空格
    li $v0, 4               #系统调用/print_string
    syscall

    addi $t0, $t0, 4        #数组的下一个地址
    bne $t0, $t8, print     #在数组终止地址前继续打印

    la $a0, line            #数组打印完后换行
    li $v0, 4               #系统调用/print_string
    syscall

    j exit                  #退出程序

exit:
    li $v0, 10              #系统调用/退出程序
    syscall
```