



# 《计算机组成原理与接口技术实验》 实验报告

(实验一)

学 院 名 称 : 数据科学与计算机学院

学 生 姓 名 : 王迎旭

学 号 : 16340226

专业(班级) : 16 软件工程三(6)班

时 间 : 2018 年 4 月 12 日

成绩：

## 实验一：MIPS汇编语言程序设计

### 一. 实验目的

1. 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模拟器软件一套。

### 四. 实验分析与设计

（注：总流程图设计图在最后一页）

#### （1）确定实验需求

用户输入10个无符号数，进行排序，输出排序结果。

（在项目开始前，首先是查阅了网上关于MIPS实现排序算法的相关资料，发现基本都是实现的冒泡排序算法，为了更好理解MIPS的编程架构，自己选择去写一个选择排序，不受外界干扰的去完成本学期的第一次设计任务）

#### （2）抽象分块处理

I、将整个程序划分为：**【输入模块】**、**【排序模块】**、**【输出模块】**

II、将排序模块进一步划分为：**【循环查找最大数模块】**、**【交换数字模块】**

#### （3）数组元素存取的设计

本实验打算使用数组配合循环进行存取与排序，因此需要设计数组元素的存取。

I、首先是**【计算偏移量】**：计算数组元素的地址到数组首地址的偏移量。

II、随后计算【**计算元素地址**】：数组首地址与偏移量的和。

III、最后使用 load 或 store 指令进行存取。

如：

```
sll $t3, $t5, 2 # 求出 j 的偏移量
add $t3, $t1, $t3 # 找到 arr[j]
lw $t3, ($t3) # 把 t3 中存放的数字拿出来

# get arr[maxIndex]
sll $t6, $t4, 2 # 找到 max 的偏移量
add $t6, $t1, $t6 # 找到 arr[max]
lw $t6, ($t6) # 把 t6 中存放的数字拿出来
```

#### (4) 循环的设计

概述：

本实验所要完成设计的三个模块外面套的都是一个循环，所以确定如何用汇编语言写循环是本实验的关键所在。

根据以往使用高级语言写for循环的经验，步骤确定如下

I、先对循环条件进行【**初始化**】，然后进入循环。

II、在进入循环之前先要【**检查循环条件**】：若条件检验失败，则程序跳转到 break 标签，也即跳出循环。若条件检验成功，则继续执行。

III、【**执行循环体**】。

IV、执行后，【**跳转到 continue**】标签，进行自增等工作。然后再跳转到 II。

举例说明：

##### ① 输入模块

如：

```
inputLoop:
    slt $t3, $t2, $t0 # for 循环判断条件如果 i < 10 置
    t3 为 1 否则置 t3 为 0
    beq $t3, $0, inputLoopBreak

    li $v0, 5 # 读一个数字
    syscall

    sll $t3, $t2, 2 # 求出 i 的偏移量
    add $t3, $t1, $t3 # 找到 arr[i] 的地址
    sw $v0, ($t3) # arr[i] = 输入的数字
    j inputLoopCont
```

```
inputLoopCont:
    addi $t2, $t2, 1 # i ++
    j inputLoop
```

```
inputLoopBreak:
    add $t2, $0, $0 # i = 0
```

## ② 寻找最大数模块

如：

```
outerLoop:
    slt $t3, $t2, $t0 # for 循环判断条件如果 i < 10 置
    t3 为 1 否则置 t3 为 0
    beq $t3, $0, outerLoopBreak

    add $t4, $t2, $0 # max = i
    addi $t5, $t2, 1 # j = i + 1

innerLoop:
    slt $t3, $t5, $t0 # for 循环判断条件如果 j < 10 置
    t3 为 1 否则置 t3 为 0
    beq $t3, $0, innerLoopBreak

# get arr[j]
    sll $t3, $t5, 2 # 求出 j 的偏移量
    add $t3, $t1, $t3 # 找到 arr[j]
    lw $t3, ($t3) # 把 t3 中存放的数字拿出来
# get arr[maxIndex]
    sll $t6, $t4, 2 # 找到 max 的偏移量
    add $t6, $t1, $t6 # 找到 arr[max]
    lw $t6, ($t6) # 把 t6 中存放的数字拿出来

    slt $t3, $t3, $t6 # 判断 arr[j]与 arr[max]大小
    bne $t3, $0, afterReplace
```

## (5) 条件语句的设计

使用标签来执行条件验证不成功时的跳转

如：

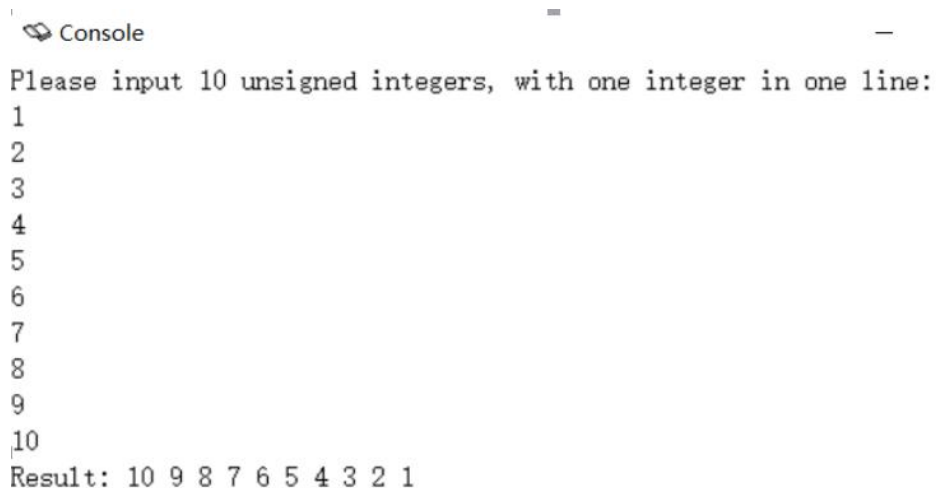
```
outerLoop:
    slt $t3, $t2, $t0 # for 循环判断条件如果 i < 10 置 t3
    为 1 否则置 t3 为 0
```

```
beq $t3, $0, outerLoopBreak
```

(6) 结合流程图，整合思路先打出对应的C语言框架，然后转化为MIPS汇编语言

(完整C代码与MIPS代码放置于实验报告末尾)

(7) 实验结果



```

Console
Please input 10 unsigned integers, with one integer in one line:
1
2
3
4
5
6
7
8
9
10
Result: 10 9 8 7 6 5 4 3 2 1

```

(8) 实验结果分析

I、Console提示框中首先输出一行提示语，提示用户输入。输入的内容存在内存里。用户输入10个数字后，开始排序。排序完从内存读取并输出提示语和排序结果，用空格分隔每个数。

II、数据段分析

①内存地址0x10040000-0x10400020这一块内存里存储是排序后的数据。

[0x10040000]	0x0000000a	0x00000009	0x00000008	0x00000007
[0x10040010]	0x00000006	0x00000005	0x00000004	0x00000003
[0x10040020]	0x00000002	0x00000001	0x00000000	0x00000000

②内存地址0x10010000-0x10010050这块内存中存储的是给定的字符串比如

“Please input 10 unsigned number , with one integer in one line” 等

[0x10000000]...[0x10010000]	0x00000000
[0x10010000]	0x61656c50 0x69206573 0x7475706e 0x75200020
[0x10010010]	0x6769736e 0x2064656e 0x65746e69 0x73726567
[0x10010020]	0x6977720c 0x6f206874 0x6920656e 0x6765746e
[0x10010030]	0x69207265 0x6e6f206e 0x696c2065 0x0a3a656e
[0x10010040]	0x73655200 0x3a746c75 0x0a002000 0x00000000
[0x10010050]	0x0000000a 0x00000000 0x00000000 0x00000000

### III、寄存器分析

本次实验自己管理的部分总共用了 \$zero、\$a0、\$v0、\$t0 到 \$t7 等寄存器，所以在实验过程中对其中存储的值进行了改变来完成程序的功能，如图：

General Registers			
R0 (r0) = 00000000	R8 (t0) = 0000000a	R16 (s0) = 00000000	R24 (t8) = 00000000
R1 (a0) = 10010000	R9 (t1) = 10040000	R17 (s1) = 00000000	R25 (t9) = 00000000
R2 (v0) = 0000000a	R10 (t2) = 0000000a	R18 (s2) = 00000000	R26 (k0) = 00000000
R3 (v1) = 00000000	R11 (t3) = 00000000	R19 (s3) = 00000000	R27 (k1) = 00000000
R4 (a0) = 1001004b	R12 (t4) = 00000009	R20 (s4) = 00000000	R28 (gp) = 10008000
R5 (a1) = 7fff578	R13 (t5) = 0000000a	R21 (s5) = 00000000	R29 (sp) = 7fff574
R6 (a2) = 7fff57c	R14 (t6) = 00000002	R22 (s6) = 00000000	R30 (s8) = 00000000
R7 (a3) = 00000000	R15 (t7) = 00000006	R23 (s7) = 00000000	R31 (ra) = 00400018

注：

其中，有一些固定功能的寄存器，如

- ①\$zero 存零值，不能变
- ②\$v0 用来储存系统调用号和输入的结果
- ③\$a0 作为系统调用的参数

也有一些临时变量寄存器，如

- ①\$t0 固定存排序的个数 10
- ②\$t1 固定存数组的首地址
- ③\$t2 是外循环的计数器 i，这里最终的结果等于 \$t0进而判断循环结束
- ④\$t3 临时存各种表达式的运算结果。例如最终的 0 是最后一次条件判断外循环计数器  $i < 10$  的结果
- ⑤\$t4 存的是每轮循环最大值的下标。最后一次循环，最大值的下标被设为了最后一个外循环计数器 i 的值
- ⑥\$t5 存的是内循环计数器 j，这里最终的结果等于 \$t0进而判断内层循环结束
- ⑦\$t6 也是临时存各种结果，如数组元素的偏移量、地址、值
- ⑧\$t7 也是临时存各种结果，如数组元素的值

⑨当然，还有例如 `$gp`、`$sp` 等本实验没有自己管理的寄存器也在其他部分起关键作用

## 五. 实验心得

从这次实验中，我学到了很多东西。

首先，这是我第一次写较为复杂的排序算法的汇编代码并让它成功运行，而不是简单入门级别的 `hello world`，也算是小有成就。学习任何编程语言，如果能顺利的写出排序算法，那么就差不多算得上掌握了这门语言的精髓所在，同时也是对逻辑思维能力的一种肯定。与高级语言相比，由于汇编语言更为底层，所以就多了许多需要考虑的内容，同时自己在编写时候也遇到了不少问题。例如，高级语言替我们封装了数组元素存取、循环流程、内存空间分配的问题，在高级语言里只需要简单的语法，便可以方便地实现许多目的。

比如，一个 `for` 循环，原来是通过合适地条件判断和跳转形成的。寄存器的管理，实际上十分复杂，自己要去寻找某一阶段哪些寄存器能用，哪些已被占用。在调用寄存器这个坑上更是栽了好几次，同时由于对 `PCspim` 软件使用不熟练，先是需要去研究了软件的功能，随后再导入自己的代码一点一点调试程序。再比如，数组的存取，在 `C` 语言里方便地用方括号语法，在汇编的世界里，是要先计算偏移量，加上首地址，最后再通过地址读取或存取数值。

另外一个比较长知识的东西就是 `MIPS` 的 读取/储存架构。`MIPS` 汇编语言中的操作被分为两类：内存和寄存器间的内存访问，以及寄存器间 `ALU` 计算。一个简单的赋值语句 `arr[max] = arr[j]`，需要将 `arr[max]` 和 `arr[j]` 的地址算出，然后通过其地址读取 `arr[j]` 的值到寄存器，再将寄存器里的值存入 `arr[max]` 的地址。无法从 `arr[j]` 直接到 `arr[max]`。

在做完完整的实验之后，仍然有两个问题不太清楚。第一个是**申请的堆空间如何释放，是否需要“释放”**。上网查阅资料，有的说可以通过传入申请参数的相反数，来让栈指针移动回原来的地方。然而在模拟器上，这被视为非法操作。我查了查相关资料堆空间是否需要汇编程序员释放，然而并没有获得满意的解答。另一问题是，**关于汇编代码的维护**。代码是写出来了，但现在看来，却并不好维护。我觉得应该要多研究研究别人优秀无误的代码，总结如何写出更加易于维护的代码，编写时有没有什么技巧。

## 【程序代码】

I、C代码:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
int main()
{
    uint32_t num = 10 ;
    uint32_t *arr = malloc(sizeof(uint32_t)*num);
    printf("Please input:");
    printf("%u",num);
    printf(" unsigned integer, with one integer in one
line:\n");
    for(uint32_t i = 0 ; i < num ; i ++ )
    {
        scanf("%u",&arr[i] );
    }//输入
    for(uint32_t i = 0 ; i < num ; i ++ )
    {
        uint32_t max = i ;
        for( uint32_t j = i + 1 ; j < num ; j ++ )
        {
            if(arr[j] > arr[max])
            {
                max = j ;
            }
        }
        uint32_t temp = arr[i] ;
        arr[i] = arr[max];
        arr[max] = temp ;
    }//排序
    printf("Result: ");
    for(uint32_t i = 0 ; i < num ; i ++ )
    {
        printf(" ");
        printf("%u",arr[i]);
    }//输出
    printf("\n");
    free(arr);
}
```



## II、MIPS代码:

```
.text
.globl main

main:
    la $a0, prompt1 # 打印 prompt1
    li $v0, 4
    syscall

    la $t0, num # 把 10 这个数字存到寄存器 t0 中
    lw $t0, ($t0)

    add $a0, $t0, $0 # 打印数字 10
    li $v0, 1
    syscall

    la $a0, prompt2 # 打印 prompt2
    li $v0, 4
    syscall

    sll $a0, $t0, 2 # 为数组申请空间
    li $v0, 9
    syscall

    add $t1, $v0, $0 # t1 寄存器中存数组首地址

    add $t2, $0, $0 # 把临时变量 i 存到寄存器 t2 中

inputLoop:
    slt $t3, $t2, $t0 # for 循环判断条件如果 i < 10 置 t3
    为 1 否则置 t3 为 0
    beq $t3, $0, inputLoopBreak

    li $v0, 5 # 读一个数字
    syscall

    sll $t3, $t2, 2 # 求出 i 的偏移量
    add $t3, $t1, $t3 # 找到 arr[i] 的地址
    sw $v0, ($t3) # arr[i] = 输入的数字
    j inputLoopCont

inputLoopCont:
    addi $t2, $t2, 1 # i ++
    j inputLoop
```

```
inputLoopBreak:
    add $t2, $0, $0 # i = 0

outerLoop:
    slt $t3, $t2, $t0 # for 循环判断条件如果 i < 10 置 t3
    为 1 否则置 t3 为 0
    beq $t3, $0, outerLoopBreak

    add $t4, $t2, $0 # max = i
    addi $t5, $t2, 1 # j = i + 1

innerLoop:
    slt $t3, $t5, $t0 # for 循环判断条件如果 j < 10 置 t3
    为 1 否则置 t3 为 0
    beq $t3, $0, innerLoopBreak

    # get arr[j]
    sll $t3, $t5, 2 # 求出 j 的偏移量
    add $t3, $t1, $t3 # 找到 arr[j]
    lw $t3, ($t3) # 把 t3 中存放的数字拿出来

    # get arr[maxIndex]
    sll $t6, $t4, 2 # 找到 max 的偏移量
    add $t6, $t1, $t6 # 找到 arr[max]
    lw $t6, ($t6) # 把 t6 中存放的数字拿出来

    slt $t3, $t3, $t6 # 判断 arr[j]与 arr[max]大小
    bne $t3, $0, afterReplace

replace:
    add $t4, $t5, $0 # max = j
    j afterReplace

afterReplace:
    j innerLoopCont # continue

innerLoopCont:
    addi $t5, $t5, 1 # j++
    j innerLoop

innerLoopBreak:
    beq $t2, $t4, afterSwap # 如果 i = max 不交换
```

swap:

```
sll $t3, $t2, 2 # 计算 i 偏移量
add $t3, $t1, $t3 # 求出 arr[i]

sll $t5, $t4, 2 # 计算 max 偏移量
add $t5, $t1, $t5 # 求出 arr[max]

lw $t6, ($t3) # 取出 arr[i]

lw $t7, ($t5) # 取出 arr[max]
sw $t7, ($t3) # 交换

sw $t6, ($t5)

j afterSwap
```

afterSwap:

```
j outerLoopCont # 继续循环
```

outerLoopCont:

```
addi $t2, $t2, 1 # i++
j outerLoop # 进入外层循环
```

outerLoopBreak:

```
la $a0, result
li $v0, 4
syscall
add $t2, $0, $0 # i = 0
```

outputLoop:

```
slt $t3, $t2, $t0 # 判断 i < 10 是否成立
beq $t3, $0, outputLoopBreak
```

```
la $a0, space # 打印空格
li $v0, 4
syscall
```

```
sll $t3, $t2, 2 # 求出 i 的偏移量
add $t3, $t1, $t3
lw $a0, ($t3) # 把 t3 中存放的数字拿出来存入 a0
```

```
li $v0, 1 # 打印数字
syscall
```

```
j outputLoopCont

outputLoopCont:
    addi $t2, $t2, 1 # i++
    j outputLoop

outputLoopBreak:
    la $a0, newline # 打印换行
    li $v0, 4
    syscall

    li $v0, 10 # 退出
    syscall

.data # 数据声明模块
prompt1:
    .asciiz "Please input "

    prompt2:
    .asciiz " unsigned integers, with one integer in
one line:\n"

    result:
    .asciiz "Result:"

    space:
    .asciiz " "

    newline:
    .asciiz "\n"

    num:
    .word 0x0000000a
```

