《计算机组成原理实验》

实验报告

（实验二）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	18 计教学 3 班
学 生 姓 名 ：	谢俊杰
学　　号	18340181
时　　间 ：	2019　年　10　月　19　日

成 绩 ：

实验二 ： 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 QEMU 模拟器和GNU/Linux 下 x86-MIPS 交叉编译环境的使用。

二. 实验内容

实验的具体内容与要求。

从4道题目中根据学号末2位对4求余后加一的规则确定必做的题目，用MIPS汇编语言进行程序设计，使输入输出符合题目要求，而我所必做的题号为2。同时要求不允许使用高级语言（如C/C++）编译出汇编语言代码，不允许调用 libc 库函数（如 stdio.h，string.h 中的函数）。

三. 实验器材

PC机一台，装有Linux操作系统的虚拟机一套，QEMU模拟器软件一套，GNU跨平台交叉编译环境一套。

四. 实验过程与结果

题二（必做）

1、确定题目需求

第一行输入商店个数n，第二行输入n个数为菜价，要求输出该商店与相邻商店的菜价平均值，即在一个循环中遍历求相邻的3或2（两端）个数的平均值。但是，由于该实验中模拟环境仅仅提供MIPS的读取和输出字符流的的函数，并没有明确的读取和输出一个int类型数的函数，故在这无疑中有一个隐含的需求——以字符数组的形式读入两行，并将两字符数组中的值转化为熟悉的十进制进行题目中要求的求相邻数平均值的运

算，更有甚者，输出时也需转化为字符数组形式输出。

2、分模块处理

在用汇编语言编写程序前，我先是用高级语言写了关于这个程序的大概思路，并检验各字符转化十进制数、十进制数转化为字符输出的函数的正确性，然后按照【输入模块】、【转化为十进制数模块】、【计算模块】、【转化为字符输出模块】这四个模块进行MIPS汇编程序的编写。

（1）输入模块

对于两行的输入，由于长度都是未知的，所以先开辟两个分别为4和1024字节字符数组num和array对输入的内容进行储存。在读入时，将其读入函数的返回值即实际输入字符的个数存在寄存器$v0中，但输入的回车也会被读入，故需将该长度均需减一以便后续处理。

```
li $v0, 4003    #read length
li $a0, 0
la $a1, num
li $a2, 4
syscall
```

```
li $v0, 4003
li $a0, 0
la $a1, array
li $a2, 1024
syscall
```

（2）转化为十进制数模块

由于输入时都是以十进制的格式进行输入，我们需要重新将其字符类型存储的数字转化为十进制形式，对于第一行读入的仅有的一个数，在这我使用寄存器$t2来存储，所以只需遍历该字符数组并不断将$t2中的数乘10和加上将每个字符减去48后的值，而遍历数组的每个元素时，先是计算偏移量，由于是以字节为单位，即偏移量为i，而元素地址则为数组首地址与偏移量的和。最后我将该数存在寄存器$s5中。

```
.global input
input:
    addu $t6, $t7, $t1  #t6 = t7 + i
    mul $t2, $t2, 10
    lb $t4, 0($t6)
    sub $t3, $t4, 48    #t3 = a[i] - 48
    addu $t2, $t2, $t3  #t2 += t3
    addi $t1, $t1, 1
    blt $t1, $t0, input
    nop
    jr $ra
    nop
```

同理对于第二行的输入，同样对于其中每一个数可以运用上述方法，但需要注意的

是，第二行输入的是n个数且n个数间有空格间隔，并且需要将这n个数存起来，在这，我事先开辟一个1024字节的字符数组str，用ASCII码的值来存放这n个数的值，这样也就用一个字符数组来模拟一个int类型的数组了，往后的取数存数都可以按字节来存取，虽然值的范围（0-127）小了很多，但考虑到题目的具体要求和现实情况，这是符合需求的。

```
move $t7, $zero
la $t7, array    #address->t7
la $s1, str
move $t0, $zero
move $t0, $v0    #temp-> t0 = v0
sub $t0, $t0, 1
move $t2, $zero #index-> t2 = 0
move $t1, $zero #t1 = i = 0
```

而这样的处理需要使用嵌套循环，外层循环是数组str的对应每一个数，内层循环是上述对第一行输入的处理，条件则为该字符不为空格，且若遇到空格则跳过忽略，继续遍历下一个字符；在内层循环转换一个十进制数时，需要将其以一个字节的方式store进str字符数组中。

汇编代码如下：

```
    bge $t1, $t0, loop3
    nop
    addu $t6, $t7, $t1  #t6 = t7 + i
    move $t4, $zero
    la $t4, space_str
    lb $s4, 0($t4)
    move $t3, $zero
    lb $t3, 0($t6)
    move $s0, $zero

loop2:  #若不为空格则处理为十进制数并以字节存入char[]
    bge $t1, $t0, loop3
    nop
    mul $s0, $s0, 10
    sub $t5, $t3, 48
    addu $s0, $s0, $t5
    addu $s2, $s1, $t2
    sb $s0, 0($s2)
    addi $t1, $t1, 1
    bge $t1, $t0, loop3
    addu $t6, $t7, $t1
    lb $t3, 0($t6)
    bne $t3, $s4, loop2
    nop
```

```
loop:
    sub $t1, $t1, 1

loop1:  #遇上空格则跳过
    addi $t1, $t1, 1
    addu $t6, $t7, $t1  #t6 = t7 + i
    move $t4, $zero
    la $t4, space_str
    lb $s4, 0($t4)
    move $t3, $zero
    lb $t3, 0($t6)
    beq $t3, $s4, loop1
    nop
```

（3）计算模块

计算平均值首先分为两种情况，若商店个数为1时，则直接输出输入的字符数组；若商店个数大于1时，则求自己和相邻商店菜价的平均值，即上述得到的str数组的遍历

到的该数和相邻数的平均值，但需注意的是在数组str两端的数的相邻数仅有一个，故需要分类计算即进行条件判断下标i是否是0或n-1，以避免越界进行计算。每个平均值计算公式如下：

i = 0, ave = (str[0] + str[1]) / 2

i = n - 1, ave = (str[n-1] + str[n -2]) / 2

else, ave = (str[i-1] + str[i] + str[i+1]) / 3

```
li $t1, 1
bne $s5, $t1, cal    #s5 ?= 1
nop

li $v0, 4004         #s5 == 1
li $a0, 1
la $a1, array
li $a2, 1024
syscall
b end
```

同时这个计算过程也是一个遍历过程即一个下标i从0到n-1的循环，其中str[i]的地址也是数组首地址与偏移量i相加得到。

```
cal:
    move $t1, $zero     #t1 = i = 0
    la $t7, str
    move $t3, $zero

output:          #s5 !=1
    beq $t1, $zero, output1 #t1 = 0
    nop
    sub $t2, $s5, 1
    beq $t1, $t2, output2   #t1 = s5 - 1
    nop

    addu $t6, $t7, $t1      #output3
    sub $t2, $t1, 1
    addu $t5, $t7, $t2
    lb $s3, 0($t6)
    lb $s4, 0($t5)
    move $s0, $zero
    add $s0, $s3, $s4
    addi $t2, $t1, 1
    addu $t5, $t7, $t2
    lb $s3, 0($t5)
    add $s0, $s0, $s3   #3_num_add
    div $s0, $s0, 3         #s0->average
    jal tra
    nop
    addi $t1, $t1, 1
    blt $t1, $s5, output
    nop
    b end
    nop
```

```
output1:
    lb $s3, 0($t7)
    lb $s4, 1($t7)
    move $s0, $zero
    addu $s0, $s3, $s4
    div $s0, $s0, 2
    jal tra
    nop
    addi $t1, $t1, 1
    blt $t1, $s5, output
    nop
    b end
    nop

output2:
    move $t2, $s5
    sub $t2, $t2, 1
    addu $t6, $t7, $t2
    sub $t2, $t2, 1
    addu $t5, $t7, $t2
    lb $s3, 0($t6)
    lb $s4, 0($t5)
    move $s0, $zero
    addu $s0, $s3, $s4
    div $s0, $s0, 2
    jal tra
    nop
    addi $t1, $t1, 1
    blt $t1, $s5, output
    nop
    b end
    nop
```

（4）转化为字符输出模块

由于输出函数也是以字符数组的形式输出，故事先开辟一个1024字节的字符数组str_out，用来存储经过计算后的n个平均值的对应字符。而这个从十进制数转化为字符的过程我将其写为一个函数tra，以寄存器$s0为十进制数的传递容器。而这个函数主要是将$s0中的数逐位取出并+48字符化，然后再逐位以字节方式store进str_out这个字符数组中，每store完一个字符后下标加一，通过数组首地址加偏移量使地址指向下一个字节，直至将该十进制数的每位字符化完毕，并且在最后加上空格以符合输出格式。

所以该转化函数主要由两个循环构成，一是通过除以10判断十进制数的位数，二是通过逐次除以位基数（如4位数除以1000）得出的高位加上48字符化后以字节方式store进数组并取余使去除取出的高位，直至该数变为0。

在这个程序最后直接调用write函数将得到的str_out字符数组输出。

汇编代码如下：

```
.global tra
tra:
    # move $t3, $zero   #t3 = i = 0
    la $s7, str_out
    la $t5, space_str
    move $t2, $s0
    li $t8, 1
    bne $t2, $zero, loop4
    nop
    div $t8, $t8, 10
    bne $t8, $zero, loop5
    nop
    jr $ra
    nop


loop4:  #判断位数
    div $t2, $t2, 10
    mul $t8, $t8, 10
    bne $t2, $zero, loop4
    nop
    div $t8, $t8, 10
    bne $t8, $zero, loop5
    nop
```

```
loop5:  #逐位取出字符化
    div $s1, $s0, $t8
    addi $s2, $s1, 48
    addu $s6, $s7, $t3
    sb $s2, 0($s6)
    mul $s1, $s1, $t8
    sub $s0, $s0, $s1
    div $t8, $t8, 10
    addi $t3, $t3, 1
    bne $t8, $zero, loop5
    nop
    lb $t4, 0($t5)
    addu $s6, $s7, $t3
    sb $t4, 0($s6)
    addi $t3, $t3, 1
    # li $v0, 4004
    # li $a0, 1
    # la $a1, str_out
    # li $a2, 4
    # syscall
    jr $ra
    nop
```

3、实验结果及分析

经交叉编译后，运行程序并输入样例，结果如下：

```
eagle@ubuntu:~/Desktop$ qemu-mips ./ex
8
4 1 3 1 6 5 17 9
2 2 1 3 4 9 10 13 eagle@ubuntu:~/Desktop$
```

而输入3个数时，提示错误输入：

```
eagle@ubuntu:~/Desktop$ qemu-mips ./ex
8
1 2 3
Wrong input!
```

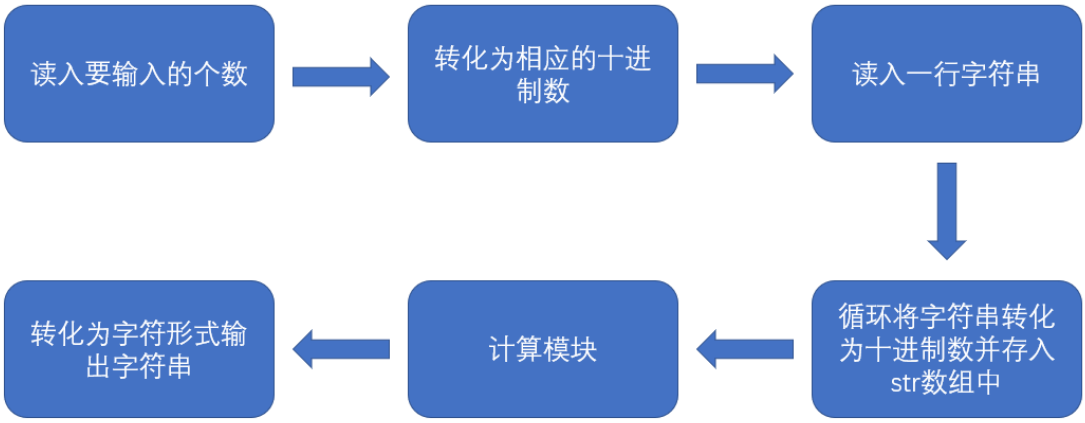
在【转化为十进制数模块】中：

经过输入转化为十进制数处理后，以ASCII码存储这些十进制数的数组str的内存0x00411404-0x00411404b这一块内存存储的数据如下：

```
(gdb) x/8u 0x00411404
0x411404:       4       1       3       1       6       5       17      9
```

与输入的字符数组相应顺序一样，达到与先前用字符数组模拟int数组的预期目的。

4、程序流程图



题一（选做）

1、确定题目需求

该题要求将10个无符号数从大到小进行排序，同样，输入输出均和题二一样，都是将其以字符数组形式读入，而在此我选择用冒泡排序算法对这些数进行排序。

2、分模块处理

在用汇编语言编写程序前，我先是用高级语言写了关于这个程序的大概思路，而输入输出的转换方法基本上和题二一样，然后按照【输入模块】、【转化为十进制数模块】、【计算模块】、【转化为字符输出模块】这四个模块进行MIPS汇编程序的编写。

由于我是先做了题二再做题一的，所以在题二的模块基础上（也就是输入的数的范围为0-127），题一主要就是【计算模块】与题二有所不同，转换模块在这就不再详细叙述了，具体实现可往上翻看题二的实验报告，而计算模块主要就是冒泡排序算法的编写，而这样的MIPS程序设计就显得简单很多。

冒泡排序算法主要就是就是两个for循环和交换操作。

关于两个for循环，主要就是遍历该数组，将前面的数与后面的逐个相比较，再由大小关系条件判断是否进行交换操作。

对于交换操作，若前者的值比后者的值小便两者交换，而在汇编层面的交换操作，就是两个地址所对应的值的相互store到相应的地址上。

汇编代码如下：

```
sort1:
    bge $t0, $t3, out1  #i < 9
    nop
    addi $t2, $t0, 1
    move $t1, $t2   #t1 = j = i + 1
    li $t4, 10

sort2:
    bge $t1, $t4, out2  #j < 10
    nop
    addu $t6, $t7, $t0
    addu $t5, $t7, $t1
    lb $s0, 0($t6)
    lb $s1, 0($t5)
    bge $s0, $s1, out3  #[i] >= [j]
    nop
    sb $s0, 0($t5)  #交换两个
    sb $s1, 0($t6)

out3:
    addi $t1, $t1, 1
    b sort2
    nop

out2:
    addi $t0, $t0, 1
    b sort1
    nop

out1:   #退出循环
```

3、实验结果及分析

经交叉编译后，运行程序并输入样例，结果如下：

```
eagle@ubuntu:~/Desktop$ qemu-mips ./ex
4 1 3 1 6 5 17 9 8 6
17 9 8 6 6 5 4 3 1 1 eagle@ubuntu:~/Desktop$
```

五. 实验心得

在这个实验中，虽然我是先写出该程序设计的高级语言版本，再由此用MIPS去编写得出的，可能也算不上完全汇编程序设计，但是在真正实现题目输入输出功能时，还是感觉有点小成就的。用MIPS编写一路上并不顺利，但也是这些不顺利让我学到了很多东西，比如，关于寄存器的管理和使用方面，由于管理不当，在编写循环和函数的过程中，遇到了寄存器间混淆使用的情况，这是我用gdb debug了很久，然后又通过在纸上的不断演算才发现的，虽然有一些都是临时变量的使用，但是由于函数和main间的寄存器都是互通的，所以在函数中更要注意自己在主函数已经在用哪些寄存器了，最好能够记下常用作下标的是哪些寄存器和记下自己的使用习惯。

还有数组元素的遍历是先要通过数组首地址和偏移量计算得到的，在处理读入的字符数组中与空格字符相比较时，比较的是在其地址值的元素的值，而非单纯与地址相比，需要通过上述地址进行取出后比较，存数时亦一样。这也C语言中用方括号简单存取相差甚大。

在汇编语言的世界里，很多操作都变得很底层，需要自己去管理相应寄存器和地址值，还有各种各样的跳转指令，与高级语言很不同，简单的语句能实现的功能，但在用MIPS编写时往往都很复杂，比如第2题简单的几个循环，代码量都几乎是翻倍的。而且在debug过程中，发现每次翻看前面的代码时，往往都需要花费时间重新思考其功能和操作，特别是跳转指令，可能是我的注释不够的缘故吧，下次在编写汇编程序的时候更应关注注释，最好把寄存器的含义和具体操作的高级语言形式写在一旁的注释。其实，感觉还是要多看看网上或同学优秀的代码，多学习他们管理寄存器的方法和注释风格。

在实验检查时，在输入8个数的前提下输入5个数，发现输出为6个数，最后两个的输出是错的，也就是在模拟数组中，第5个数后依然有储存第6个数并且数值为0，这确实是我在写汇编程序时没有注意到的问题——忽略程序的鲁棒性，但也在指出该问题后对该问题进行了修改。在今后也应注意在满足程序基本需求的情况下，程序的鲁棒性也是程序设计的一个重要部分。