

中山大学数据科学与计算机学院本科生实验报告

(2019 学年秋季学期)

课程名称: 计算机组成原理实验

任课教师: 郭雪梅

助教: 丁文、汪庭葳

年级&班级	2019 级 04 班	专业(方向)	计算机科学与技术 (超算方向)
学号	19335112	姓名	李钰
电话	19847352856	Email	1643589912@qq.com
开始日期	2020.10.9	完成日期	2020.10.16

一、实验题目

编写简短的 MIPS 程序

二、实验目的

掌握 MIPS 语言, 熟悉 syscall 的用法, 掌握输入输出数据、循环操作

三、实验内容

练习 1

假定你想编写一个 MIPS 程序 foo, 该程序使用 5 个字的数组, 数组元素初始化为整数 1, ..., 5.

```
.data
```

```
foo:    .word 1,2,3,4,5
```

你用程序来把数组 foo 中的每个数加 2 再写回数组 foo

代码:

```
.data
foo:    .word 1, 2, 3, 4, 5
.text
main:
    la $t0, foo           #取地址, 将foo的地址赋给$t0
    lw $t1, 0($t0)        #将$t0中第一个数读入到$t1
    addiu $t1, $t1, 2      # $t1中的数自加2  下面以此类推
    sw $t1, 0($t0)
    lw $t1, 4($t0)
    addiu $t1, $t1, 2
    sw $t1, 4($t0)
    lw $t1, 8($t0)
    addiu $t1, $t1, 2
    sw $t1, 8($t0)
    lw $t1, 12($t0)
    addiu $t1, $t1, 2
    sw $t1, 12($t0)
    lw $t1, 16($t0)
    addiu $t1, $t1, 2
    sw $t1, 16($t0)
```

编译运行:

一开始,foo 里的数分别为:1,2,3,4,5,如图

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005

运行后,foo 中的数值都比原来增加了 2

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007

练习 2

从键盘输入两个数, 计算并输出这两个数的和

代码:

```

.data
    str1: .ascii "Enter 2 numbers:"
    str2: .ascii "The sum is "
.text
main:
    ori $v0, $0, 4          #System call code 4 for printing a string
    la $a0, str1            #address of Str1 is in $a0
    syscall                #print the string

    ori $v0, $0, 5          #System call code 5 for read integer, $v0 contains integer read
    syscall
    add $t0, $v0, $zero     #将$v0里的数移到$t0中

    ori $v0, $0, 5          #System call code 5 for read integer, $v0 contains integer read
    syscall
    add $t1, $v0, $zero     # 将$v0里的数移到$t1中

    ori $v0, $0, 4          #System call code 4 for printing a string
    la $a0, str2            #address of Str2 is in $a0
    syscall                #print the string

    ori $v0, $0, 1          # service 1 is print integer
    add $a0, $t0, $t1        # load desired value into argument register $a0, using pseudo-op
    syscall

```

```

exit:  ori $v0, $0, 10      #System call code 10 for exit
        syscall            #print the sum

```

实验结果:

```

Enter 2 numbers:1
2
The sum is 3
— program is finished running —

```

练习 3

计算 $1^2+2^2+...+100^2$ ，参照练习 2 输出

思路： 利用循环，将变量 i 从 1 递加到 100，每加一次，做平方然后增添到最终输出的 sum 变量上，判断循环是否继续进行的条件是变量 i 是否等于 101

```

.data
    .align 2
    Str: .ascii "The sum of square from 1 to 100 is "

.text

main:
    add $t0, $zero, $zero
    addi $t1, $zero, 1
    addi $t3, $zero, 101

loop:
    mul $t2, $t1, $t1
    add $t0, $t2, $t0
    addi $t1, $t1, 1
    bne $t1, $t3, loop

print:
    ori $v0, $0, 4           #System call code 4 for printing a string
    la $a0, Str              #address of Str1 is in $a0
    syscall                  #print the string

    ori $v0, $0, 1           # service 1 is print integer
    add $a0, $t0, $zero       # load desired value into argument register $a0, using pseudo-op
    syscall

exit:
    ori $v0, $0, 10          #System call code 10 for exit
    syscall                  #print the sum

```

实验结果:

```

The sum of square from 1 to 100 is 338350
— program is finished running —

```

练习 4

编写两个版本的 `firstlpos` (starting from [firstlpos.s](#)) 函数, 在 `$a0` 中给定一个数, 而在 `$v0` 中返回 `$a0` 字中最左边的非零位的位置. 如果 `$a0` 的值是 0, 在 `$v0` 中存 -1. 在查找此位置的过程中, 允许你修改 `$a0` 值. 位置从 0 (最右位) 到 31 (符号位). 其中一种解应该重复移位 `$a0`, 每次移位后, 检查符号位. 另一种方法是初始时使用 `0x80000000` 作为掩码, 并不断右移该掩码来检查 `$a0` 的每一位.

思路:

采用不断左移 `$a0`, 并和 `0x800` 相与的方法, 找出最左端的 1. 如果相与之后为 `0x8000`, 则说明此时 `$a0` 的最左端为一, 输出当时的位置.

设置一个值\$t0 初始为 0，每循环一次自加一（最后代表从左数第一个非零位的位置），从原数开始和 0x8000 相与，如果结果是 0，即代表着当前\$a0 的最高位是 0，则\$a0 左移一位，\$t0 自加一；如果相与结果不是 0，则输出\$t0 自加一后的值。当循环直 32 位数全部比较完之后，即\$t0z 最终值为 33 时，跳转至 end 循环，输出-1。

代码如下

```
firstlpos:
    addi $t0, $zero, 0
    addi $sp, $sp, -4
    sw $ra, 0($sp)
    lui $s5, 0x8000
loop:    and $t1, $a0, $s5
        sll $a0, $a0, 1
        addi $t0, $t0, 1
        beq $t0, 33, end
        beq $t1, $zero, loop

        add $a0, $t0, $0
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra
end:    addi $a0, $0, -1
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra
```

结果：

1
16
32
-1

五、实验感想

实验过程遇到的问题及想法：

练习一：写代码的时候，一开始在读 foo 中第二个数值时出现错误，一开始的错误代码是 “lw \$t1, 1(\$t0)” ,后来仔细看了数据段每个数据的地址都是相隔 4 的，所以正确的应该是 “lw \$t1, 4(\$t0)” 往后的依次加 4;

练习二：“syscall” 写在 “ori \$v0, \$0, 5” 等调用语句后面，中间不能插入 add 等语句，通过该练习我掌握了基本输入输出数字，以及输出字符串的方法

练习三：就是通过循环，自加来计算结果，注意最终终止条件

练习四：联系到了条件分支语句，以及函数的调用，其中第一次用到了移位比较的解题方法，在后续的学习过程中应该熟悉掌握其用法。