**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：汪庭葳、刘洋旗

|  |  |  |  |
| --- | --- | --- | --- |
| 年级&班级 | **2018级1班** | 专业(方向) | **计算机类** |
| 学号 | **18308045** | 姓名 | **谷正阳** |
| 电话 | **13355426001** | Email | [**Guzy0324@163.com**](mailto:Guzy0324@163.com) |
| 开始日期 | **2019.10.1** | 完成日期 | **2019.10.8** |

**一、实验题目**

计算机结构与组成 实验1

1. **实验目的**

本实验的目的是让你熟悉MARS仿真器的使用，学习用它来运行和调试程序。如果你需要在其他地方使用MARS，可以直接下载mars.jar。MARS程序由密苏里州立大学开发。UCB的教师消除了一些BUG，但还没被主要开发者接受. MARS程序是用JAVA写的，因此使用它之前，需要在你的机器上安装Java J2SE 1.5.0 SDK或者更高版本,该程序可以从Sun公司下载。

**三、实验内容**

**1.** 实验步骤

练习 2: 一个简短的MIPS程序

编写MIPS代码完成：在给定$s0 和 $s1的值的前提下，将下列值放到 $t? 寄存器中（其中？表示任意0-7之间的数）:

$t0 = $s0

$t1 = $s1

$t2 = $t0 + $t1

$t3 = $t1 + $t2

...

$t7 = $t5 + $t6

换言之, 对$t2 到 $t7的每个寄存器，都存储其前两个$t? 寄存器的值. 寄存器$s0 和 $s1 中包含初始值.

不要在代码中设置$s0 和 $s1 的值. 取而代之, 学会如何在MARS中手动设置它们的值.

将你的代码存储到文件lab4\_ex2.s 中，然后给老师检查.

练习 3: 调试（Debugging）MIPS程序

调试程序lab4\_ex3.s中的循环. 该程序将从$a0所指示的内存地址中复制一个整数到$a1所指示的内存地址, 起到读入一个zero值时结束. 复制的整数的个数(不含zero值)应存储在中$v0.

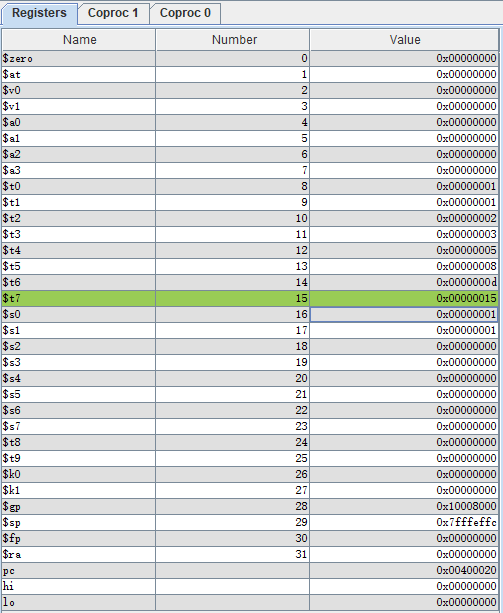
请在文件lab4\_ex3.txt 中描述代码的错误bug(s). 新建一个文件lab4\_ex3\_ok.s ，其中放的是没有bug的代码lab4\_ex3.s.把程序给老师看。

**2.** 实验原理

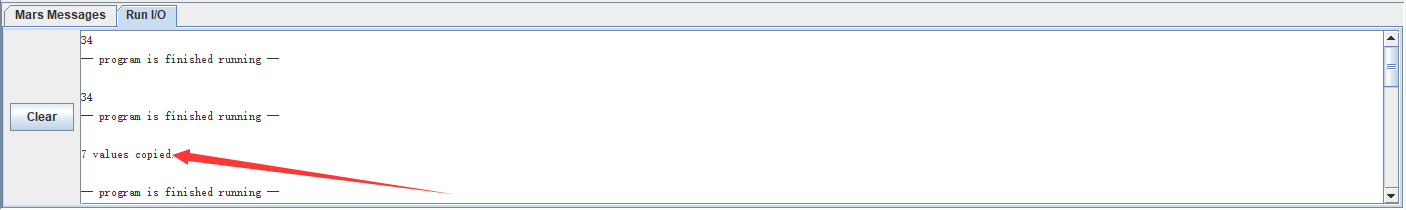
见上

**四、实验结果**

练习 2：（设置$s0，$s1为1，结果满足斐波那契数列



练习 3：

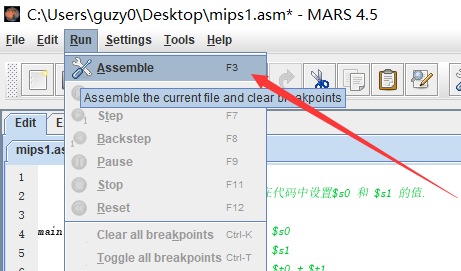


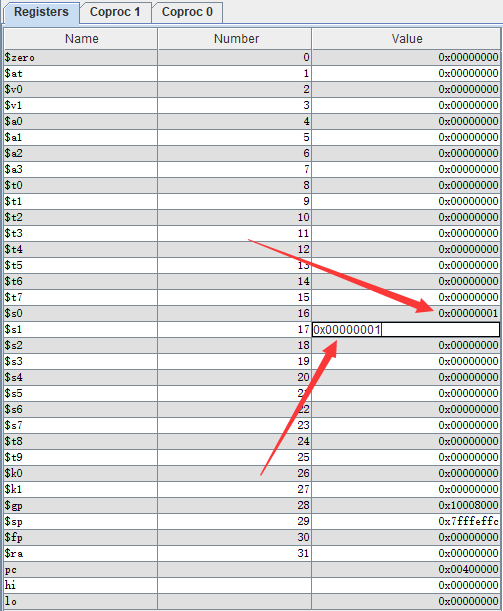
**五、实验感想**

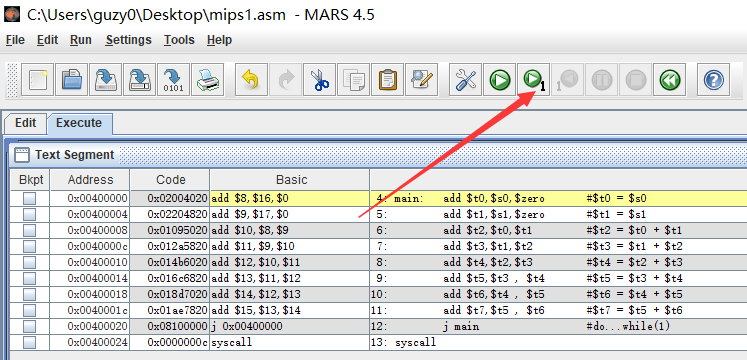
MIPS指令语法简单易懂，真不愧为RISC

**附录（流程图，注释过的代码）：**

练习2：







.data

#不要在代码中设置$s0 和 $s1 的值.

.text

main: add $t0,$s0,$zero #$t0 = $s0

add $t1,$s1,$zero #$t1 = $s1

add $t2,$t0,$t1 #$t2 = $t0 + $t1

add $t3,$t1,$t2 #$t3 = $t1 + $t2

add $t4,$t2,$t3 #$t4 = $t2 + $t3

add $t5,$t3 , $t4 #$t5 = $t3 + $t4

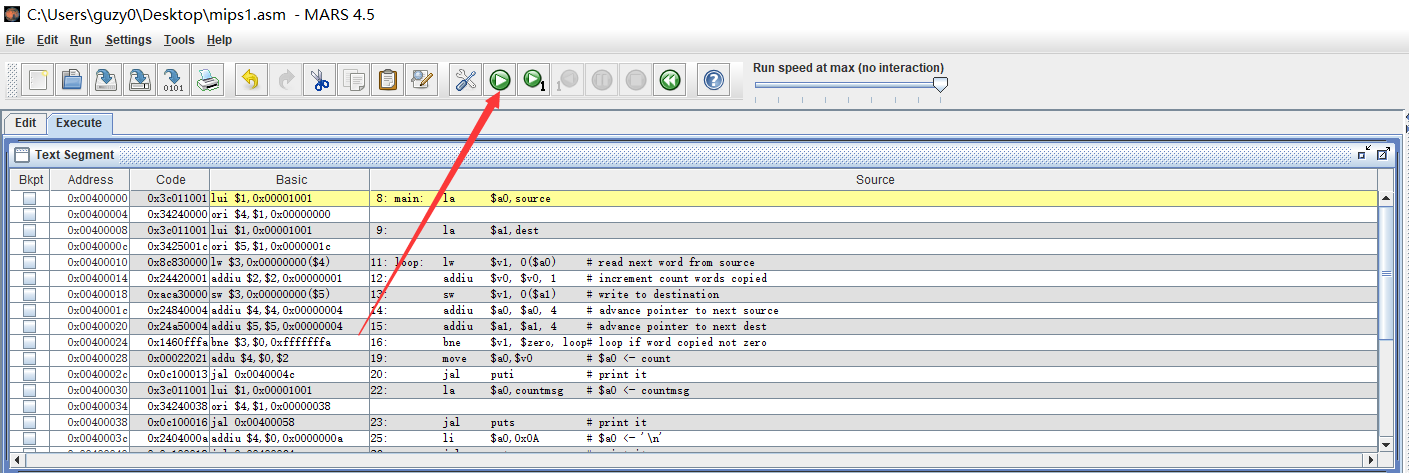
add $t6,$t4 , $t5 #$t6 = $t4 + $t5

add $t7,$t5 , $t6 #$t7 = $t5 + $t6

j main #do...while(1)

syscall

练习 3：



.data

source: .word 3, 1, 4, 1, 5, 9, 0

dest: .word 0, 0, 0, 0, 0, 0, 0

countmsg: .asciiz " values copied. "

.text

main: la $a0,source

la $a1,dest

loop: lw $v1, 0($a0) # read next word from source

addiu $v0, $v0, 1 # increment count words copied

sw $v1, 0($a1) # write to destination

addiu $a0, $a0, 4 # advance pointer to next source

addiu $a1, $a1, 4 # advance pointer to next dest

bne $v1, $zero, loop# loop if word copied not zero

loopend:

move $a0,$v0 # $a0 <- count

jal puti # print it

la $a0,countmsg # $a0 <- countmsg

jal puts # print it

li $a0,0x0A # $a0 <- '\n'

jal putc # print it

finish:

li $v0, 10 # Exit the program

syscall

### The following functions do syscalls in order to print data (integer, string, character)

#Note: argument $a0 to syscall has already been set by the CALLEE

puti:

addi $sp, $sp, -4

sw $v0, 0($sp) # push $v0 into stack

li $v0, 1 # service 1 is print integer

syscall

lw $v0, 0($sp) # change $v0 back to the original value

addi $sp, $sp, 4

jr $ra # return to main function

puts:

addi $sp, $sp, -4

sw $v0, 0($sp) # push $v0 into stack

li $v0, 4 # service 4 is print string

syscall

lw $v0, 0($sp) # change $v0 back to the original value

addi $sp, $sp, 4

jr $ra # return to main function

putc:

addi $sp, $sp, -4

sw $v0, 0($sp) # push $v0 into stack

li $v0, 11 # service 11 is print character

syscall

lw $v0, 0($sp) # change $v0 back to the original value

addi $sp, $sp, 4

jr $ra # return to main function