计 算 机 组 成 原 理 实 验 报 告

题目：MIPS 指令系统

姓　　名：　段欣然

专　　业：　计算机科学与技术

年　　级：　2020 级

学　　号：　202011081033

任课教师：　王志春

完成日期：　2021 年 3 月 18 日

人工智能学院

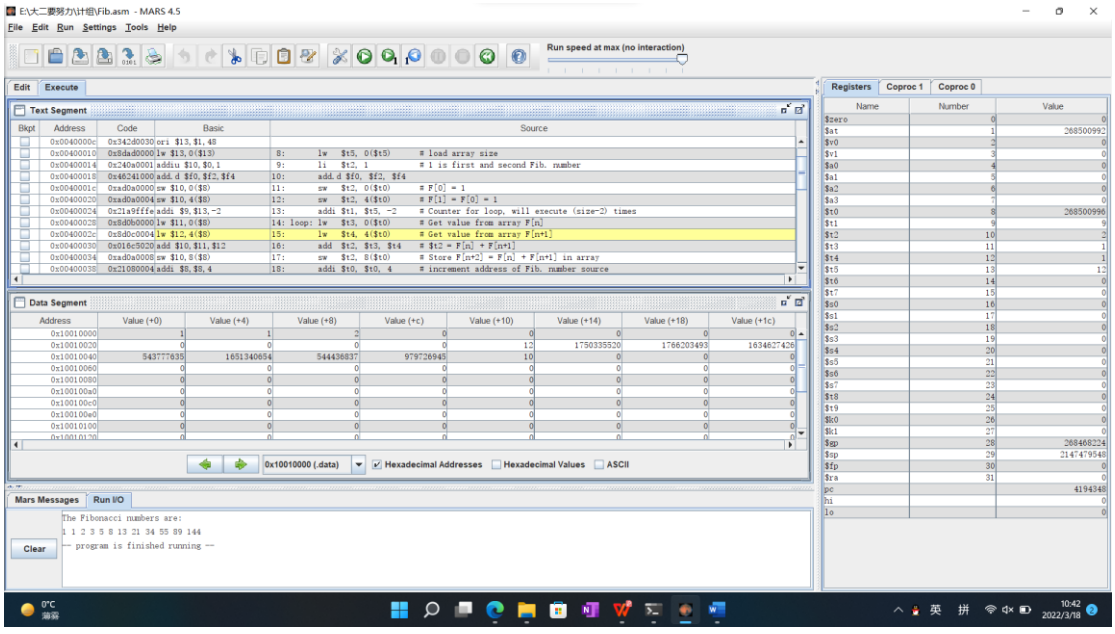
一、 实验要求

通过在 MARS 模拟器上运行和调试汇编程序，掌握 MIPS 指令系统计算机运行原理。

1. 熟悉并掌握 MARS 软件
2. 编写并执行 MIPS 程序
3. 了解 MIPS 对过程调用的支持，掌握指令的寻址方式

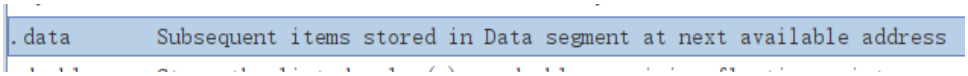
二、 实验结果与分析

1. 熟悉并站务 MARS 软件



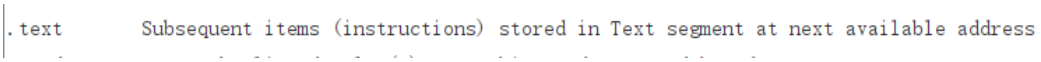
图表 1 运行完成截图

a) 程序代码中“.data”“.word”“.text”关键字的含义



图表 2MARS 软件 help 中指令描述

“.data”表示以下是初始化数据段，变量声明后，即在主存中分配空间。



图表 3MARS 软件 help 中指令描述

“.text”表示接下来是代码段，即各项指令操作，程序入口为“main”。

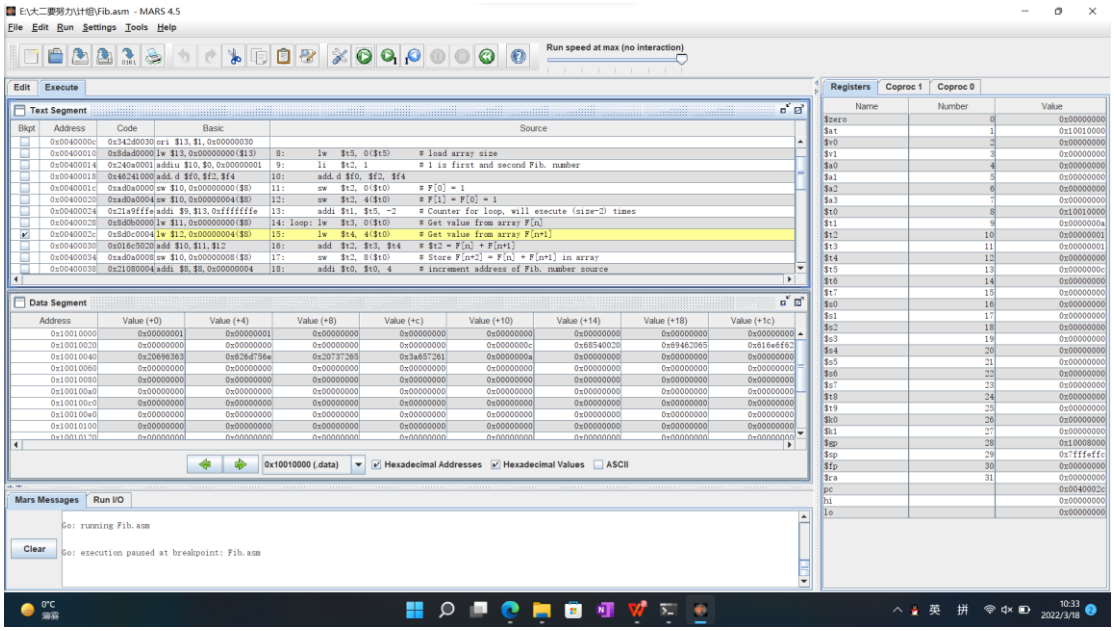
.word	Store the listed value(s) as 32 bit words on word boundary

图表 4MARS 软件 help 中指令描述

“.word”定义一个字，并为它分配空间，用于数据声明。例如，“var1: .word 3”声明一个 $word$ 类型的变量 $var1$, 同时给其赋值为 3。

b) MARS 程序中的断点

在 Execute 界面左侧$Bkpt$属性下可勾选以添加断点，在程序15行添加断点并运行，如下图所示。



图表 5 程序添加断点后运行截图

c) 再次点击运行即可继续执行后续代码

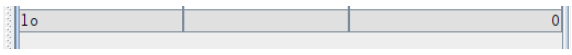
图表 6 继续执行后续代码

d) 寄存器内容查看与修改

图表 7 在界面右端可查看与修改寄存器内容

e) 变量n保存位置，计算第13个$Fibonacci$数

变量n保存位置为$\$t5$，由程序可知，$\$t5$从变量$size$的地址访问了$size$值。



图表 8 把n设置为13后寄存器数值截图

图表 9 运行结束后输出 13 个$Fibonacci$数

f) 代码中$syscall$指令的作用

图表 10*syscall*（1）

此处*syscall*的位置在输出运算结果之后，作用是声明程序结束。

```
syscall                 # print heading
```

图表 11*syscall*（2）

此处位于*print*函数的末尾，作用是通过系统调用实现终端输出字符串*head*

```
jr    $ra               # return
```

图表 12*syscall*（3）和*syscall*（4）

这两处的*syscall*用于循环输出*Fibonacci*数列，第一个*syscall*通过系统调用实现终端输出*Fibonacci*数，之后一个*syscall*实现终端输出空格分隔每个*Fibonacci*数，最终实现如图表1（或9）的结果输出。

2. 编写并执行 MIPS 程序

代码如下

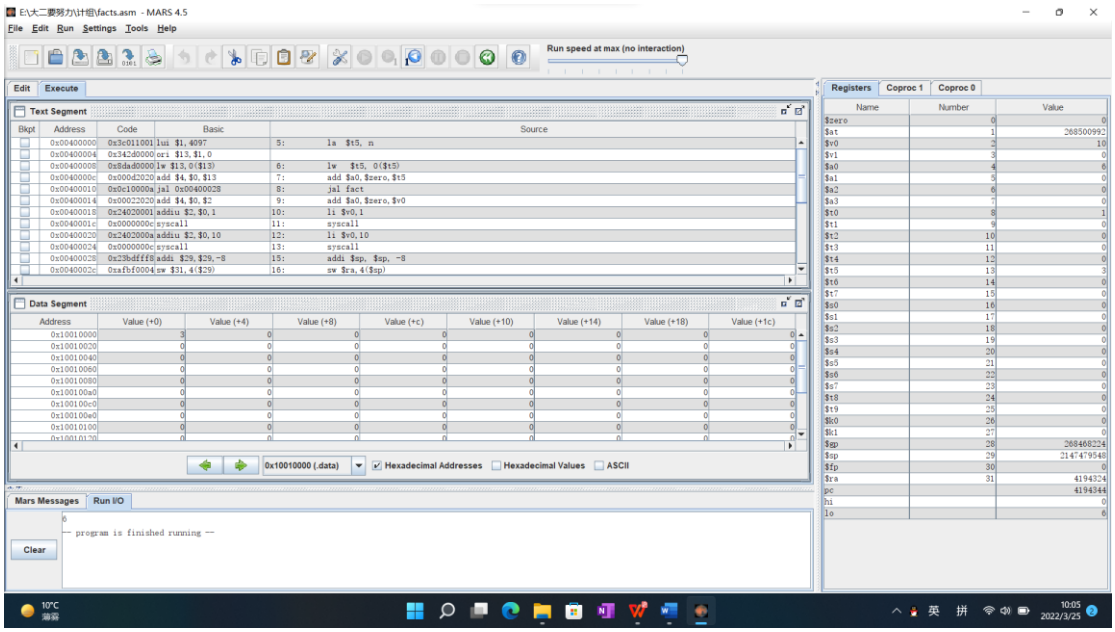
```
    .data
num1: .word 1
num2: .word 0
    .text
    la $s0, num1
    la $s1, num2
    lw $s0, 0($s0)
    lw $s1, 0($s1)
    move $t0, $s0
    move $t1, $s1
    xor $t2, $t1, $t0
    xor $t3, $t2, $t1
    xor $t4, $t3, $t2
    xor $t5, $t4, $t3
    xor $t6, $t5, $t4
    xor $t7, $t6, $t5
    move $a0, $t7
    li $v0, 1
    syscall
    li     $v0, 10
    syscall
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	0
$t2	10	1
$t3	11	1
$t4	12	0
$t5	13	1
$t6	14	1
$t7	15	0
$s0	16	1
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194380
hi		0
lo		0

图表 13 程序运行结束（寄存器内容）截图

3. 了解 MIPS 对过程调用的支持，掌握指令的寻址方式



图表 14 运行结果截图

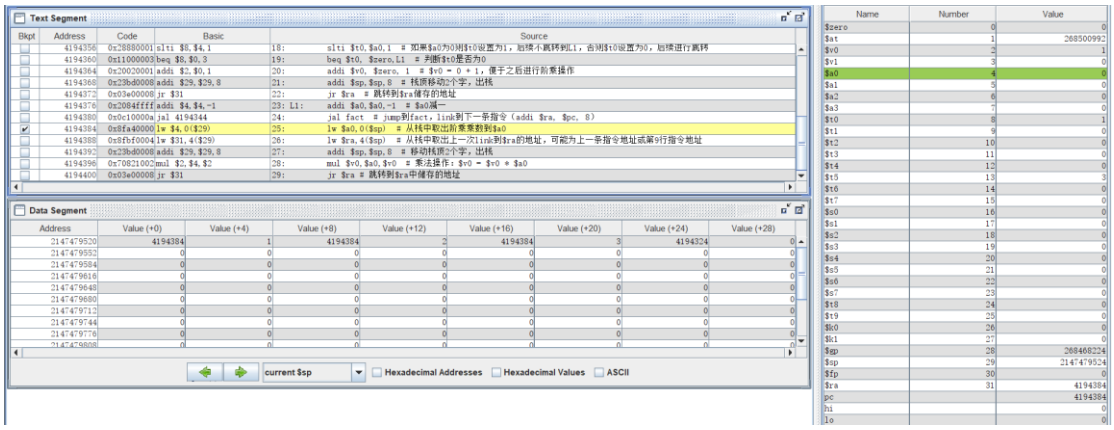
a) 每行汇编指令添加注释，解释其在求解阶乘中的作用

如下图所示

```
.data
n: .word  3  # 声明一个word类型变量，给它分配空间并赋值3
.text
main:
        la  $t5, n  # 加载n的地址
        lw  $t5, 0($t5)  # 加载n
        add $a0,$zero,$t5  # 让$a0=n
        jal fact  # 跳转到求阶乘函数fact，link到下一条指令（addi $ra, $pc, 8）
        add $a0,$zero,$v0  # 函数运行结果移到$a0中，准备调用系统输出
        li $v0,1  # 输出整数功能调用码
        syscall  # 系统调用，在终端输出
        li $v0,10  # 退出程序调用码
        syscall  # 系统调用，退出程序
fact:
        addi $sp, $sp, -8  # 栈顶移动两个字，压栈
        sw $ra,4($sp)  # 将寄存器$ra内容写入栈
        sw $a0,0($sp)  # 将寄存器$a0内容写入栈
        slti $t0,$a0,1  # 如果$a0为0则$t0设置为1，后续不跳转到L1，否则$t0设置为0，后续进行跳转
        beq $t0, $zero,L1  # 判断$t0是否为0
        addi $v0, $zero, 1  # $v0 = 0 + 1，便于之后进行阶乘操作
        addi $sp,$sp,8  # 栈顶移动2个字，出栈
        jr $ra  # 跳转到$ra储存的地址
L1:     addi $a0,$a0,-1  # $a0减一
        jal fact  # jump到fact，link到下一条指令（addi $ra, $pc, 8）
        lw $a0,0($sp)  # 从栈中取出阶乘乘数到$a0
        lw $ra,4($sp)  # 从栈中取出上一次link到$ra的地址，可能为上一条指令地址或第9行指令地址
        addi $sp,$sp,8  # 移动栈顶2个字，出栈
        mul $v0,$a0,$v0  # 乘法操作：$v0 = $v0 * $a0
        jr $ra # 跳转到$ra中储存的地址
```

图表 15 源代码及注释

b) 给出第一次运行到第 25 行代码时，栈顶位置和栈内存储的内容，并对每项内容给出解释

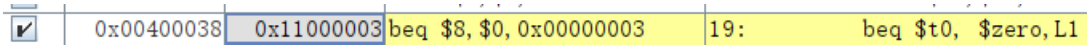


图表 16 第一次运行到 25 行代码时栈顶位置和栈内存储的内容

其中栈顶位置在 2147479524，指向数值1，在此位置的原因是最后一次跳转

到 fact 时栈顶阶乘乘数为 0，判断后进行了出栈操作。栈内存储内容分别为在第三次跳转并链接到 fact 时入栈的阶乘乘数1，第 25 行指令地址 4194384，第二次跳转并链接到 fact 时入栈的阶乘乘数2，第 25 行指令地址 4194384，第一次跳转并链接到 fact 时入栈的阶乘乘数3，第 9 行指令地址 4194324。

c) 在第 19 行代码第一次跳转时，给出该行代码对应的机器语言（即二进制代码）；给出该机器语言中对应 L1 的立即数，并解释为什么是该立即数



图表 17 第 19 行代码第一次跳转

机器语言：0x11000003，对应二进制代码 0b 0001 0001 0000 0000 0000 0000 0000 0011

对应 L1 地址的立即数为 0x00400048，因为 beq 指令为 pc 相对寻址方式，机器语言中跳转指令的相对位置为 3 条指令，且当前地址为 0x00400038 故对应 L1 地址的立即数为 0x00400038+（1+3）*4 = 0x00400048。

三、 实验小结

MIPS 指令直接对计算机硬件操作，在使用时需要考虑直接作用于寄存器的操作，编写一个 MIPS 程序要比直接用高级语言编写更加困难。经过本次实验，拓宽了我对计算机知识的认知，我觉得计算机理所应当的具备功能，却很难知晓相应的汇编指令是什么。我想学习汇编语言的作用就是让我知其然所以然，能够更加透彻地认知信息世界。