



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

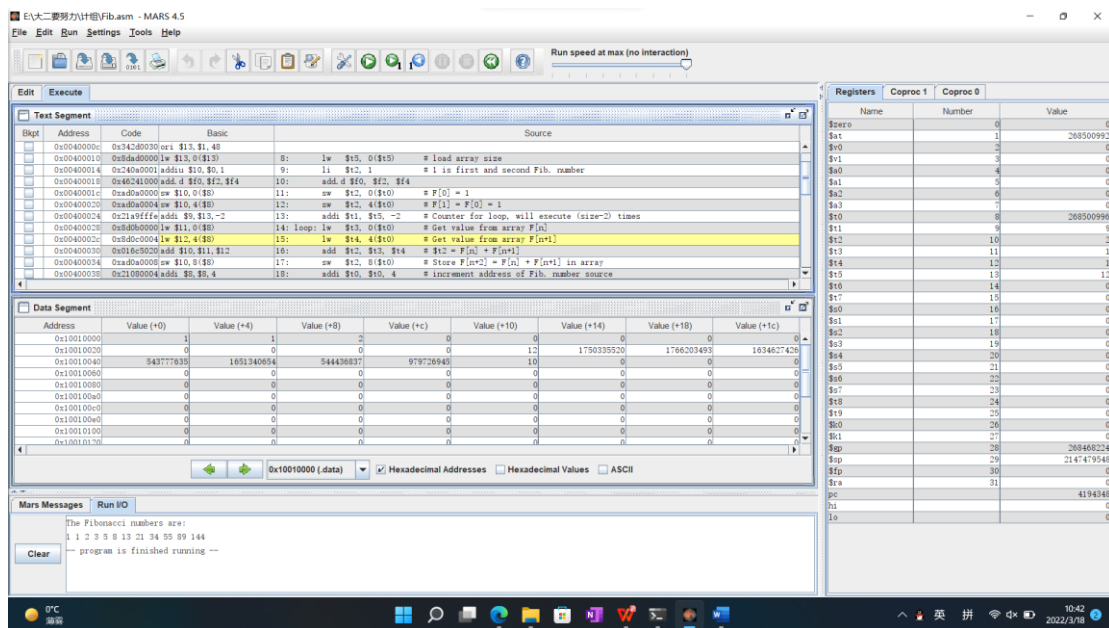
一、实验要求

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

.data Subsequent items stored in Data segment at next available address

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。

.text Subsequent items (instructions) stored in Text segment at next available address

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”。

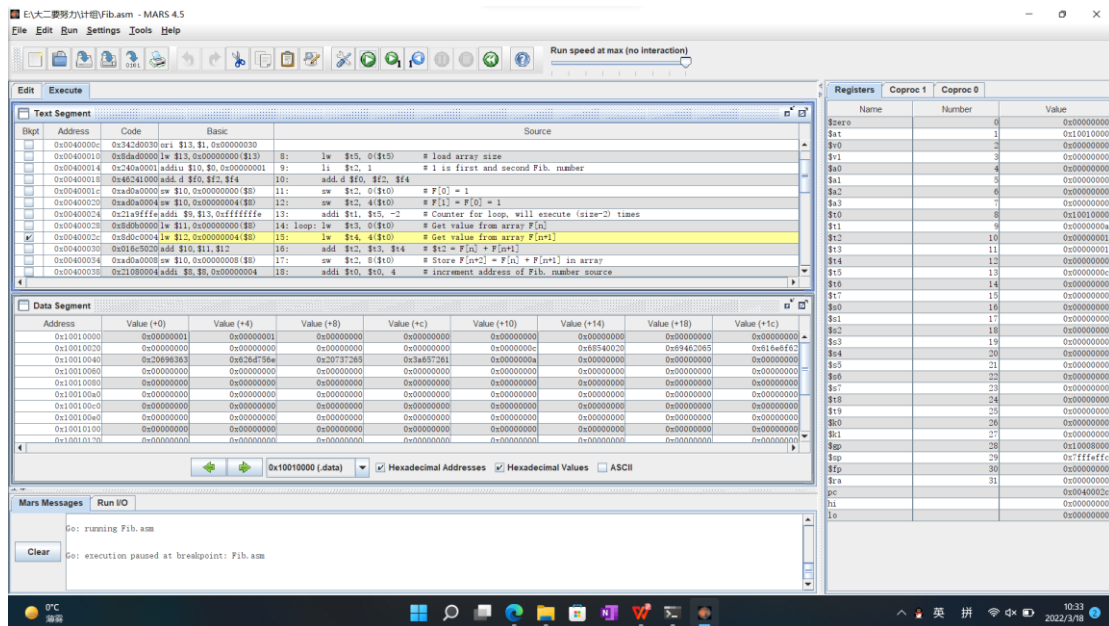
.word Store the listed value(s) as 32 bit words on word boundary

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。



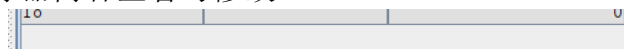
图表 5 程序添加断点后运行截图

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

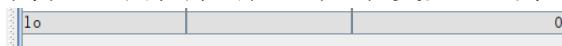
d) 寄存器内容查看与修改



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

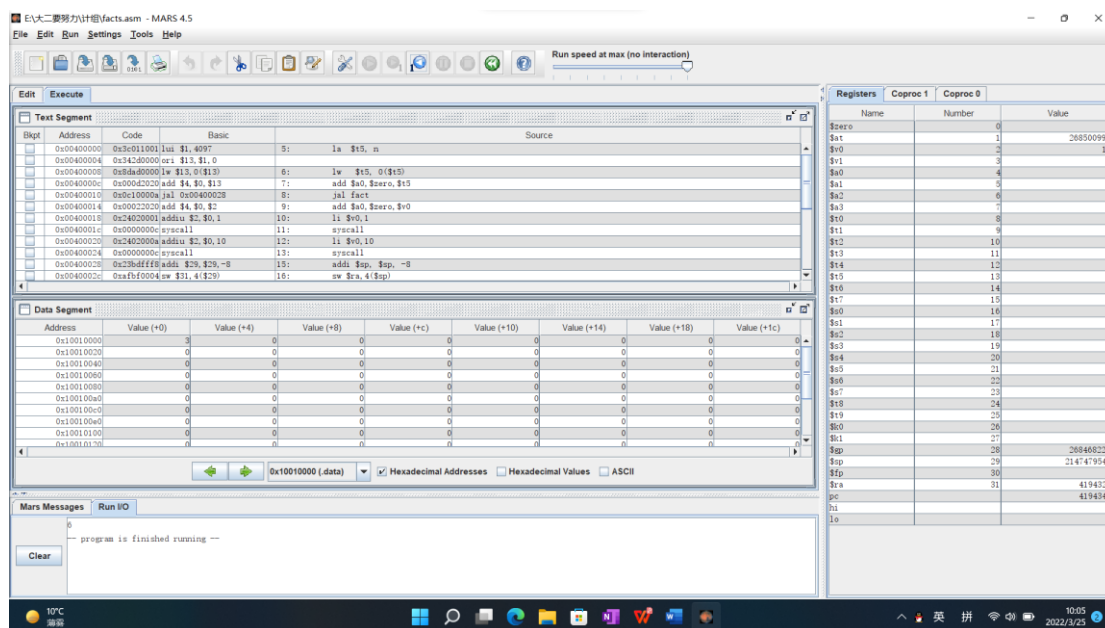
```
.data
num1: .word 1
num2: .word 0

.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

Text Segment					Name			Number	Value
Offset	Address	Code	Basic	Source					
4194356	0x1000001	17:1	la \$t5, 3	17: la \$t5, 3	\$zero		0		0
4194360	0x1000003	18:3	lw \$t5, 0(\$t5)	18: lw \$t5, 0(\$t5)	\$a0		1		268500992
4194364	0x2002001	19:3	add \$a0, \$zero, \$t5	19: add \$a0, \$zero, \$t5	\$v0		1		1
4194368	0x23b4000	20:3	jal fact	20: jal fact	\$t0		1		0
4194372	0x23b4000	21:3	add \$a0, \$zero, \$v0	21: add \$a0, \$zero, \$v0	\$a1		5		0
4194376	0x2084fff	22:3	li \$v0, 1	22: li \$v0, 1	\$a2		6		0
4194380	0x0e1000a	23:3	syscall	23: syscall	\$a3		7		0
4194384	0x0e1000a	24:3	li \$v0, 10	24: li \$v0, 10	\$a4		8		1
4194388	0x0e1000a	25:3	syscall	25: syscall	\$a5		9		0
4194392	0x23b4000	26:3	addi \$sp, \$sp, -8	26: addi \$sp, \$sp, -8	\$a6		11		0
4194396	0x23b4000	27:3	sw \$ra, 4(\$sp)	27: sw \$ra, 4(\$sp)	\$a7		23		0
4194400	0x23b4000	28:3	sw \$a0, 0(\$sp)	28: sw \$a0, 0(\$sp)	\$a8		24		0
4194404	0x23b4000	29:3	addi \$sp, \$sp, 8	29: addi \$sp, \$sp, 8	\$a9		25		0
4194408	0x23b4000	30:3	mul \$v0, \$a0, \$v0	30: mul \$v0, \$a0, \$v0	\$a10		26		0
4194412	0x23b4000	31:3	jr \$ra	31: jr \$ra	\$a11		27		0
4194416	0x23b4000	32:3	jr \$ra	32: jr \$ra	\$a12		28		268500992
4194420	0x23b4000	33:3	jr \$ra	33: jr \$ra	\$a13		29		2147479524
4194424	0x23b4000	34:3	jr \$ra	34: jr \$ra	\$a14		30		4194384
4194428	0x23b4000	35:3	jr \$ra	35: jr \$ra	\$a15		31		4194388
4194432	0x23b4000	36:3	jr \$ra	36: jr \$ra	\$a16				0
4194436	0x23b4000	37:3	jr \$ra	37: jr \$ra	\$a17				0
4194440	0x23b4000	38:3	jr \$ra	38: jr \$ra	\$a18				0
4194444	0x23b4000	39:3	jr \$ra	39: jr \$ra	\$a19				0
4194448	0x23b4000	40:3	jr \$ra	40: jr \$ra	\$a20				0
4194452	0x23b4000	41:3	jr \$ra	41: jr \$ra	\$a21				0
4194456	0x23b4000	42:3	jr \$ra	42: jr \$ra	\$a22				0
4194460	0x23b4000	43:3	jr \$ra	43: jr \$ra	\$a23				0
4194464	0x23b4000	44:3	jr \$ra	44: jr \$ra	\$a24				0
4194468	0x23b4000	45:3	jr \$ra	45: jr \$ra	\$a25				0
4194472	0x23b4000	46:3	jr \$ra	46: jr \$ra	\$a26				0
4194476	0x23b4000	47:3	jr \$ra	47: jr \$ra	\$a27				0
4194480	0x23b4000	48:3	jr \$ra	48: jr \$ra	\$a28				0
4194484	0x23b4000	49:3	jr \$ra	49: jr \$ra	\$a29				0
4194488	0x23b4000	50:3	jr \$ra	50: jr \$ra	\$a30				0
4194492	0x23b4000	51:3	jr \$ra	51: jr \$ra	\$a31				0
4194496	0x23b4000	52:3	jr \$ra	52: jr \$ra	\$a32				0
4194500	0x23b4000	53:3	jr \$ra	53: jr \$ra	\$a33				0
4194504	0x23b4000	54:3	jr \$ra	54: jr \$ra	\$a34				0
4194508	0x23b4000	55:3	jr \$ra	55: jr \$ra	\$a35				0
4194512	0x23b4000	56:3	jr \$ra	56: jr \$ra	\$a36				0
4194516	0x23b4000	57:3	jr \$ra	57: jr \$ra	\$a37				0
4194520	0x23b4000	58:3	jr \$ra	58: jr \$ra	\$a38				0
4194524	0x23b4000	59:3	jr \$ra	59: jr \$ra	\$a39				0
4194528	0x23b4000	60:3	jr \$ra	60: jr \$ra	\$a40				0
4194532	0x23b4000	61:3	jr \$ra	61: jr \$ra	\$a41				0
4194536	0x23b4000	62:3	jr \$ra	62: jr \$ra	\$a42				0
4194540	0x23b4000	63:3	jr \$ra	63: jr \$ra	\$a43				0
4194544	0x23b4000	64:3	jr \$ra	64: jr \$ra	\$a44				0
4194548	0x23b4000	65:3	jr \$ra	65: jr \$ra	\$a45				0
4194552	0x23b4000	66:3	jr \$ra	66: jr \$ra	\$a46				0
4194556	0x23b4000	67:3	jr \$ra	67: jr \$ra	\$a47				0
4194560	0x23b4000	68:3	jr \$ra	68: jr \$ra	\$a48				0
4194564	0x23b4000	69:3	jr \$ra	69: jr \$ra	\$a49				0
4194568	0x23b4000	70:3	jr \$ra	70: jr \$ra	\$a50				0
4194572	0x23b4000	71:3	jr \$ra	71: jr \$ra	\$a51				0
4194576	0x23b4000	72:3	jr \$ra	72: jr \$ra	\$a52				0
4194580	0x23b4000	73:3	jr \$ra	73: jr \$ra	\$a53				0
4194584	0x23b4000	74:3	jr \$ra	74: jr \$ra	\$a54				0
4194588	0x23b4000	75:3	jr \$ra	75: jr \$ra	\$a55				0
4194592	0x23b4000	76:3	jr \$ra	76: jr \$ra	\$a56				0
4194596	0x23b4000	77:3	jr \$ra	77: jr \$ra	\$a57				0
4194600	0x23b4000	78:3	jr \$ra	78: jr \$ra	\$a58				0
4194604	0x23b4000	79:3	jr \$ra	79: jr \$ra	\$a59				0
4194608	0x23b4000	80:3	jr \$ra	80: jr \$ra	\$a60				0
4194612	0x23b4000	81:3	jr \$ra	81: jr \$ra	\$a61				0
4194616	0x23b4000	82:3	jr \$ra	82: jr \$ra	\$a62				0
4194620	0x23b4000	83:3	jr \$ra	83: jr \$ra	\$a63				0
4194624	0x23b4000	84:3	jr \$ra	84: jr \$ra	\$a64				0
4194628	0x23b4000	85:3	jr \$ra	85: jr \$ra	\$a65				0
4194632	0x23b4000	86:3	jr \$ra	86: jr \$ra	\$a66				0
4194636	0x23b4000	87:3	jr \$ra	87: jr \$ra	\$a67				0
4194640	0x23b4000	88:3	jr \$ra	88: jr \$ra	\$a68				0
4194644	0x23b4000	89:3	jr \$ra	89: jr \$ra	\$a69				0
4194648	0x23b4000	90:3	jr \$ra	90: jr \$ra	\$a70				0
4194652	0x23b4000	91:3	jr \$ra	91: jr \$ra	\$a71				0
4194656	0x23b4000	92:3	jr \$ra	92: jr \$ra	\$a72				0
4194660	0x23b4000	93:3	jr \$ra	93: jr \$ra	\$a73				0
4194664	0x23b4000	94:3	jr \$ra	94: jr \$ra	\$a74				0
4194668	0x23b4000	95:3	jr \$ra	95: jr \$ra	\$a75				0
4194672	0x23b4000	96:3	jr \$ra	96: jr \$ra	\$a76				0
4194676	0x23b4000	97:3	jr \$ra	97: jr \$ra	\$a77				0
4194680	0x23b4000	98:3	jr \$ra	98: jr \$ra	\$a78				0
4194684	0x23b4000	99:3	jr \$ra	99: jr \$ra	\$a79				0
4194688	0x23b4000	100:3	jr \$ra	100: jr \$ra	\$a80				0
4194692	0x23b4000	101:3	jr \$ra	101: jr \$ra	\$a81				0
4194696	0x23b4000	102:3	jr \$ra	102: jr \$ra	\$a82				0
4194700	0x23b4000	103:3	jr \$ra	103: jr \$ra	\$a83				0
4194704	0x23b4000	104:3	jr \$ra	104: jr \$ra	\$a84				0
4194708	0x23b4000	105:3	jr \$ra	105: jr \$ra	\$a85				0
4194712	0x23b4000	106:3	jr \$ra	106: jr \$ra	\$a86				0
4194716	0x23b4000	107:3	jr \$ra	107: jr \$ra	\$a87				0
4194720	0x23b4000	108:3	jr \$ra	108: jr \$ra	\$a88				0
4194724	0x23b4000	109:3	jr \$ra	109: jr \$ra	\$a89				0
4194728	0x23b4000	110:3	jr \$ra	110: jr \$ra	\$a90				0
4194732	0x23b4000	111:3	jr \$ra	111: jr \$ra	\$a91				0
4194736	0x23b4000	112:3	jr \$ra	112: jr \$ra	\$a92				0
4194740	0x23b4000	113:3	jr \$ra	113: jr \$ra	\$a93				0
4194744	0x23b4000	114:3	jr \$ra	114: jr \$ra	\$a94				0
4194748	0x23b4000	115:3	jr \$ra	115: jr \$ra	\$a95				0
4194752	0x23b4000	116:3	jr \$ra	116: jr \$ra	\$a96				0
4194756	0x23b4000	117:3	jr \$ra	117: jr \$ra	\$a97				0
4194760	0x23b4000	118:3	jr \$ra	118: jr \$ra	\$a98				0
4194764	0x23b4000	119:3	jr \$ra	119: jr \$ra	\$a99				0
4194768	0x23b4000	120:3	jr \$ra	120: jr \$ra	\$a100				0
4194772	0x23b4000	121:3	jr \$ra	121: jr \$ra	\$a101				0
4194776	0x23b4000	122:3	jr \$ra	122: jr \$ra	\$a102				0
4194780	0x23b4000	123:3	jr \$ra	123: jr \$ra	\$a103				0
4194784	0x23b4000	124:3	jr \$ra	124: jr \$ra	\$a104				0
4194788	0x23b4000	125:3	jr \$ra	125: jr \$ra	\$a105				0
4194792	0x23b4000	126:3	jr \$ra	126: jr \$ra	\$a106				0
4194796	0x23b4000	127:3	jr \$ra	127: jr \$ra	\$a107				0
4194800	0x23b4000	128:3	jr \$ra	128: jr \$ra	\$a108				0
4194804	0x23b4000	129:3	jr \$ra	129: jr \$ra	\$a109				0
4194808	0x23b4000	130:3	jr \$ra	130: jr \$ra	\$a110				0
4194812	0x23b4000	131:3	jr \$ra	131: jr \$ra	\$a111				0
4194816	0x23b4000	132:3	jr \$ra	132: jr \$ra	\$a112				0
4194820	0x23b4000	133:3	jr \$ra	133: jr \$ra	\$a113				0
4194824	0x23b4000	134:3	jr \$ra	134: jr \$ra	\$a114				0
4194828	0x23b4000	135:3	jr \$ra	135: jr \$ra	\$a115				0
4194832	0x23b4000	136:3	jr \$ra	136: jr \$ra	\$a116				0
4194836	0x23b4000	137:3	jr \$ra	137: jr \$ra	\$a117				0
4194840	0x23b4000	138:3	jr \$ra	138: jr \$ra	\$a118				0
4194844	0x23b4000	139:3	jr \$ra	139: jr \$ra	\$a119				0
4194848	0x23b4000	140:3	jr \$ra	140: jr \$ra	\$a120				0
4194852	0x23b4000	141:3	jr \$ra	141: jr \$ra	\$a121				0
4194856	0x23b4000	142:3	jr \$ra	142: jr \$ra	\$a122				0
4194860	0x23b4000	143:3	jr \$ra	143: jr \$ra	\$a123				0
4194864	0x23b4000	144:3	jr \$ra	144: jr \$ra	\$a124				0
4194868	0x23b4000	145:3	jr \$ra	145: jr \$ra	\$a125				0
4194872	0x23b4000	146:3	jr \$ra	146: jr \$ra	\$a126				

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

<input checked="" type="checkbox"/>	0x00400038	0x11000003	beq \$8, \$0, 0x00000003	19: beq \$t0, \$zero, L1
-------------------------------------	------------	------------	--------------------------	--------------------------

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。