

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 级计科一班	专业(方向)	计算机类
学号	18340013	姓名	陈琮昊
电话	15734867907	Email	1062080021@qq.com
开始日期	2019. 9. 25	完成日期	2019. 10. 7

一、实验题目：计算机结构与组成 MIPS 汇编程序

二、实验目的：熟悉 MARS 仿真器的使用，学习用它来运行和调试程序

三、实验内容

1. 实验步骤：

(i) 熟悉MARS：将一段代码载入MARS，并汇编代码，回答一些基本问题

(ii) 汇编一个简短的 MIPS 程序：学会如何在 MARS 中手动设置寄存器的值

(iii) 练习调试 MIPS 程序：给出一段代码进行 Debug

2. 实验原理：汇编语言的语法

四、实验结果：

(1).data, .word, .text 指示器 (directives) 的含义是什么(即在每段中放入什么内容)?

.data 的含义:后续项存储在下一个可用地址的数据段中。

.word 的含义:存储列出的值为 32 位在字边界上。

.text 的含义:后续项存储在下一个可用地址的文本段中。

(2) 在 MARS 中如何设置断点 breakpoint? 请在第 15 行设置断点, 并在所有问题解答完后, 将此结果给老师检查。

如何设置断点: 在 Execute 一栏中最左侧有一个 “Bkpt”, 在需要设置断点的那一行勾选即可。在第 15 行设置断点则如图所示:

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x0040000c	0x342b0000	ori \$11,\$1,0x00000000	
<input type="checkbox"/>	0x00400010	0x8d6b0000	lw \$11,0x00000000(\$11)	9: lw \$t3, 0(\$t3)
<input type="checkbox"/>	0x00400014	0x11600006	beq \$11,\$0,0x00000006	10: fib: beq \$t3, \$0, finish
<input type="checkbox"/>	0x00400018	0x01285020	add \$10,\$9,\$8	11: add \$t2,\$t1,\$t0
<input type="checkbox"/>	0x0040001c	0x00094021	addu \$8,\$0,\$9	12: move \$t0, \$t1
<input type="checkbox"/>	0x00400020	0x000a4821	addu \$9,\$0,\$10	13: move \$t1, \$t2
<input type="checkbox"/>	0x00400024	0x20010001	addi \$1,\$0,0x00000001	14: subi \$t3, \$t3, 1
<input type="checkbox"/>	0x00400028	0x01615822	sub \$11,\$11,\$1	
<input checked="" type="checkbox"/>	0x0040002c	0x08100005	j 0x00400014	15: j fib

(3) 在程序运行到断点处停止时, 如何继续执行? 如何单步调试代码?

找到该图标即可继续执行:  找到该图标即可单步调试代码: 

(4) 如何知道某个寄存器 register 的值是多少? 如何修改寄存器的值。

MARS 页面右侧一栏 Registers 中的 Value 一栏查看即可。在 Value 栏双击即可修改。

(5) n 存储在内存中的哪个地址? 通过修改此内存处的值来计算第 13 个 fib 数。

n 的存储地址如图所示:

<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,0x00001001	8: la \$t3, n
--------------------------	------------	------------	--------------------	---------------

\$t2 为得到的 fib 数, \$t3 为 n 存储的位置, n 从 13 开始, 递减至 1 (0x00000001) 时为第 13 项, 可以看到此时 \$t2 为 0x000000e9 即 233, 故第 13 个 fib 数为 233。

\$t2	10	0x000000e9
\$t3	11	0x00000001

(6)16 和 18 行使用了 `syscall` 指令。其功能是什么，如何使用它？(提示：`syscall` 在 Help 中有说明!如何英文不是太好，可以一边运行，一边看效果，来体会其用途)

`syscall` 的功能：系统调用。其中第 16 行的 `syscall` 是打印一个整型，第 18 行的 `syscall` 是退出。

如何使用 `syscall`：在寄存器 `$v0` 中加载服务号；加载指定的 `$a0`、`$a1`、`$a2` 或 `$f12` 中的参数值（如果有的话）；发出 `syscall` 指令；从指定的结果寄存器中检索返回值（如果有的话）。

练习 2 和练习 3 内容详见附录。

五、实验感想：本次实验主要是练习使用 MARS 来编译 MIPS 程序，我本人也是第一次使用汇编语言来写程序。首先要做的就是之前习惯的 C 语言程序转为汇编程序，这一点在牢记指令之后问题不大。个人认为问题比较大的就是 MARS 这个软件的使用，需要学习的地方有很多：在 MARS 下 Debug；学会去看“Execute”那一栏，这些是难点。在经过大量时间的练习后，基本上已经可以使用 MARS。

附录（流程图，注释过的代码）：

(1)练习 2 的代码如下，该代码为计算斐波那契数列，修改 `$s0`, `$s1` 的值为 1 后运行得到如下结果。

代码：

```

        .data

        .text

main:
add $t0,$s0,$zero
add $t1,$s1,$zero
add $t2,$t0,$t1
add $t3,$t1,$t2
add $t4,$t2,$t3
add $t5,$t3,$t4
add $t6,$t4,$t5
add $t7,$t5,$t6
j main
syscall

```

结果：可以看到的的确是斐波那契数列，故编译成功。

\$t0	8	0x00000001
\$t1	9	0x00000001
\$t2	10	0x00000002
\$t3	11	0x00000003
\$t4	12	0x00000005
\$t5	13	0x00000008
\$t6	14	0x0000000d
\$t7	15	0x00000015
\$s0	16	0x00000001
\$s1	17	0x00000001

(2) 练习 3 修改后的代码：

```

.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
      beq    $v1,$zero,loopend
      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    finish         # print it

      la     $a0,countmsg   # $a0 <- countmsg
      jal    finish         # print it

      li     $a0,0x0A       # $a0 <- '\n'
      jal    finish         # print it

finish:
      li     $v0, 10        # Exit the program
      syscall

```

原代码错误在于：打印时显然应该为 finish 而不是 put；循环时指针地址应该加 4 而不是 1；

loop 循环中 0 被复制了多次，更改后 0 不会被复制。

结果：如图所示，可以看出实验目的已经达到。

\$v0	2	0x00000001
\$v1	3	0x00000003
\$a0	4	0x10010004
\$a1	5	0x10010020

\$v0	2	0x00000002
\$v1	3	0x00000001
\$a0	4	0x10010008
\$a1	5	0x10010024

\$v0	2	0x00000003
\$v1	3	0x00000004
\$a0	4	0x1001000c
\$a1	5	0x10010028

\$v0	2	0x00000004
\$v1	3	0x00000001
\$a0	4	0x10010010
\$a1	5	0x1001002c

\$v0	2	0x00000005
\$v1	3	0x00000005
\$a0	4	0x10010014
\$a1	5	0x10010030

\$v0	2	0x00000006
\$v1	3	0x00000009
\$a0	4	0x10010018
\$a1	5	0x10010034