

计算机系统结构 作业2 - MIPS 汇编

UNikeEN 2023/3/17

1.熟悉MARS

1. .data, .word, .text 指令的含义是什么？（即：它们的用途是什么？）

`.data` 指令标记以下为数据段，即存放程序中以及初始化的全局变量的一块内存区域。

`.word` 指令定义数据变量

`.text` 指令标记以下为代码段

2. 如何在 MARS 中设置断点 在第 14 行设置断点并运行至此。指令的地址是什么？ 第 14 行是否执行？

编译完程序，跳转至程序运行界面后，代码段窗口的左侧Bkpt列的复选框即用来设置断点。选中需要打上断点的行前的复选框，即为该行设置上了断点（如下图即在14行设置断点）。第14行指令的地址是 `0x00400020`，此时第14行代码未执行。

Text Segment					
Bkpt	Address	Code	Basic		
<input type="checkbox"/>	0x00400000	0x00004020	add \$8,\$0,\$0	7: main:	add \$t0,\$0,\$zero
<input type="checkbox"/>	0x00400004	0x20090001	addi \$9,\$0,0x00000001	8:	addi \$t1,\$zero,1
<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,0x00001001	9:	la \$t3,n
<input type="checkbox"/>	0x0040000c	0x342b0010	ori \$11,\$1,0x00000010		
<input type="checkbox"/>	0x00400010	0x8d6b0000	lw \$11,0x00000000(\$11)	10:	lw \$t3,0(\$t3)
<input type="checkbox"/>	0x00400014	0x11600006	beq \$11,\$0,0x00000006	11: fib:	beq \$t3,\$0,finish
<input type="checkbox"/>	0x00400018	0x01285020	add \$10,\$9,\$8	12:	add \$t2,\$t1,\$t0
<input type="checkbox"/>	0x0040001c	0x00094021	addu \$8,\$0,\$9	13:	move \$t0,\$t1
<input checked="" type="checkbox"/>	0x00400020	0x000a4821	addu \$9,\$0,\$10	14:	move \$t1,\$t2
<input type="checkbox"/>	0x00400024	0x20010001	addi \$1,\$0,0x00000001	15:	subi \$t3,\$t3,1
<input type="checkbox"/>	0x00400028	0x01615822	sub \$11,\$11,\$1		
<input type="checkbox"/>	0x0040002c	0x08100005	j 0x00400014	16:	j fib
<input type="checkbox"/>	0x00400030	0x21040000	addi \$4,\$8,0x00000000	17: finish:	addi \$a0,\$t0,0
<input type="checkbox"/>	0x00400034	0x24020001	addiu \$2,\$0,0x00000001	18:	li \$v0,1 #
<input type="checkbox"/>	0x00400038	0x0000000c	syscall	19:	syscall
<input type="checkbox"/>	0x0040003c	0x2402000a	addiu \$2,\$0,0x0000000a	20:	li \$v0,10

3. 如果在断点处，如何继续运行你的代码？如何单步调试你的代码？将代码运行至结束。

取消断点后点击“运行”（或F5）按钮以继续运行，或点击单步运行按钮（或F7）运行下一句指令

单步调试即使用单步运行按钮。

4. 找到 Run I/O “窗口” 程序输出的数字是什么？如果 0 是第 0 个斐波那契数，那么这是第几个斐波那契数？

程序输出34，这是第9个斐波那契数

5. 在内存中，n 存储在哪个地址？尝试通过（1）查看 Data Segment，以及 2）查看 机器代码（Text Segment 中的 Code 列）理解，如何从存储器中读取 n。

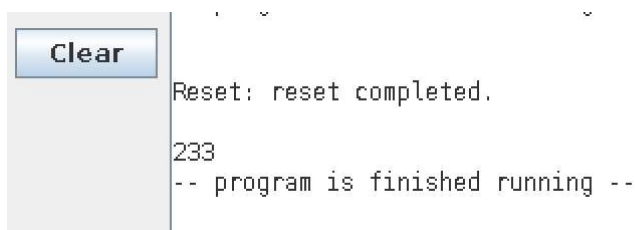
n 存储在地址 0x10010010

先通过 lui 在 \$at 中写入地址的高16位，再通过 ori 将低16位写入\$t3，然后通过 lw 将 \$t3 中存储的内存地址所对应的值写入 \$t3

6. 如何在不改变“Edit 栏”下的代码的条件下，通过在执行前手动修改存储位置的值，让程序计算第 13 个斐波那契数（索引从 0 开始）？

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	2	4	6	8	13
0x10010020	0	0	0	0	0
0x10010040	0	0	0	0	0
0x10010060	0	0	0	0	0
0x10010080	0	0	0	0	0
0x100100a0	0	0	0	0	0

编译后在 Data Segment 窗口中双击需修改值的地方，以在执行前手动修改存储位置的值。将 0x10010010 修改为13以计算第13个斐波那契数，结果为233。



7. 如何观察和修改一个寄存器中的值？重置模拟（Run→Reset）并通过（1）在一个设置好的断点停下，（2）只修改一个寄存器，（3）解除断点，来计算第 13 个斐波那契数。

在右侧 Registers 窗口中观察和修改寄存器的值。

在11行设置断点，将寄存器 \$t3 的值由9修改为13，运行结果为233。

Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$zero	0	0		
\$at	1	268500992		
\$v0	2	0		
\$v1	3	0		
\$a0	4	0		
\$a1	5	0		
\$a2	6	0		
\$a3	7	0		
\$t0	8	0		
\$t1	9	1		
\$t2	10	0		
\$t3	11	13		
\$t4	12	0		
\$t5	13	0		
\$t6	14	0		

8. **第 19 行和第 21 行用到了 syscall 指令。它是什么？如何使用它？**

`syscall` 指令通过读 `$v0` 中的值调用特定的系统函数。本例中分别赋值 1 和 10, `$v0 = 1` 表示打印整数, `$v0 = 10` 表示退出程序。

首先通过 `li` 指令给 `$v0` 赋值立即数, 然后使用 `syscall` 指令。

Issue a system call : Execute the system call specified by value in `$v0`

2.代码改错

错误在于, 内外层循环的“循环变量”(记录数组长度的变量) 合用了 `$s1` 。

新增 `$s2` 用于内层循环, 修改后的代码如下:

```
.data
arr: .word 4,2,1,5,3      # 待排序的数组
.text
.globl main
main:
    la $s0, arr           # 把数组地址存入 $s0
    li $s1, 5             # 数组长度为 5
    li $s2, 5
    li $t0, 0             # $t0 用来记录是否有交换发生
    j loop1               # 跳转到外层循环
loop1:
    addi $s1, $s1, -1     # 每次循环结束后, 最后一个元素已经排好序, 所以数组长度减 1
    li $t0, 0             # 每次开始循环时, $t0 初始化为 0
    la $s0, arr           # 每次开始循环时, $s0 指向数组开头
    addi $s2, $s1, 0
loop2:
    lw $t1, 0($s0)        # 加载当前元素
    lw $t2, 4($s0)        # 加载下一个元素
    ble $t1, $t2, skip     # 如果当前元素小于等于下一个元素, 跳过交换
    sw $t2, 0($s0)        # 交换当前元素和下一个元素
    sw $t1, 4($s0)
    li $t0, 1             # 标记有交换发生
skip:
    addi $s0, $s0, 4      # $s0 指向下一个元素
    addi $s2, $s2, -1     # 数组长度减 1
    bgtz $s2, loop2       # 如果数组长度大于 0, 继续内层循环
    beq $t0, 1, loop1     # 如果有交换发生, 继续外层循环
    j end                 # 如果没有交换发生, 排序完成, 跳转到结束
end:
    li $v0, 10            # 结束程序
    syscall
```

可以看到运行后排序结果正确

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	1	2	3	4	5
0x10010020	0	0	0	0	0
0x10010040	0	0	0	0	0
0x10010060	0	0	0	0	0
0x10010080	0	0	0	0	0
0x100100a0	0	0	0	0	0
0x100100c0	0	0	0	0	0
0x100100e0	0	0	0	0	0
0x10010100	0	0	0	0	0
0x10010120	0	0	0	0	0
0x10010140	0	0	0	0	0
0x10010160	0	0	0	0	0
0x10010180	0	0	0	0	0
0x100101a0	0	0	0	0	0
0x100101c0	0	0	0	0	0

3.函数调用的过程

添加的代码部分为

```
# prologue
### YOUR CODE HERE ###
addi    $sp, $sp, -16
sw      $ra, 12($sp)
sw      $s0, 8($sp)
sw      $s1, 4($sp)
sw      $s2, 0($sp)
```

```
# epilogue
### YOUR CODE HERE ###
lw      $ra, 12($sp)
lw      $s0, 8($sp)
lw      $s1, 4($sp)
lw      $s2, 0($sp)
addi    $sp, $sp, 16
jr      $ra
```

观察运行结果，正确：

Mars MessagesRun I/O

Clear

Should be 1, and it is:
Reset: reset completed.

Should be 1, and it is: 1
Should be 4, and it is: 4
Should be 6, and it is: 6
Should be 4, and it is: 4
Should be 1, and it is: 1
Should be 0, and it is: 0

-- program is finished running --