# 实验 1: MIPS 程序设计

实验时间: 2025/2/19 - 2025/2/22 实验人员: 李维杰, 23307140052

指导老师: 王雪平

## 1 实验目的

### 实验目的:

- · 熟悉 QtSPIM 模拟器:
- · 熟悉编译器、汇编程序和链接器;
- · 熟悉 MIPS 体系结构的计算。包括
  - a. MIPS 的数据表示;
  - b. 熟悉 MIPS 指令格式和寻址方式;
  - c. 熟悉 MIPS 汇编语言;
  - d. 熟悉 MIPS 的各种机器代码表示,包括
    - 1) 选择结构;
    - 2) 循环结构;
    - 3) 过程调用:调用与返回、栈、调用约定等;
    - 4) 系统调用。

其中加粗的部分要特别注意掌握。

## 2 实验过程

## 2.1 调试

P1. asm 运行结果:

```
= 40003c
EPC
Cause
BadVAddr = 0
Status = 3000ff10
ΗI
         = 0
LO
         = 0
R0 [r0] = 0
    [at] = 0
R1
    [v0] = a
R2
R3
    [v1] = 0
    [a0] = 1
R4
    [a1] = 7ffff480
    [a2] = 7ffff488
    [a3] = 0
R8
    [t0] = 0
R9
    [t1] = 0
R10 [t2] = 28
R11 [t3] = 39
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

1 / 7

#### P2. asm 运行结果:

```
= 40003c
EPC
Cause
            = 0
BadVAddr = 0
           = 3000ff10
Status
ΗI
            = 0
            = 0
LO
R0 [r0] = 0
    [at] = 12340000
     [v1] = 0
     [a0] = 1
    [a0] = 1

[a1] = 7fffff480

[a2] = 7fffff488

[a3] = 0

[t0] = 0
R5
R6
R7
R8
R9 [t1] = 0
R10 [t2] = 12340028
R11 [t3] = 12340028
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

### P3. asm 运行结果:

```
PC
          = 400044
         = 0
EPC
Cause
BadVAddr = 0
Status = 3000ff10
LO
         = 0
R0 [r0] = 0
    [at] = 10010000
R1
R2
    [v0] = a
    [v1] = 0
R3
    [a0] = 1
R4
R5
    [a1] = 7ffff480
    [a2] = 7ffff488
R6
    [a3] = 0
R8 [t0] = 10010040
R9 [t1] = 10010000
R10 [t2] = 28
R11 [t3] = 3b
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

## 2.2 改写程序

```
翻译代码(源码见 p1-new. asm)
            .data
       .asciiz "Please enter 1st number: "
msg1:
       .asciiz "Please enter 2nd number: "
msg2:
msg3:
       .asciiz
                "The result of "
                " & "
msg4:
       .asciiz
       .asciiz "is: "
msg5:
msg6:
        .asciiz "\nDo you want to try another(0-continue/1-
exit): "
       .text
```

```
.globl main
main:
loop:
   li $v0, 4
   la $a0, msg1
   syscall
   li $v0,5
   syscall
   move $t1, $v0
   li $v0, 4
   la $a0, msg2
   syscall
   li $v0,5
   syscall
   move $t2, $v0
   add $t0, $t1, $t2
   li $v0, 4
   la $a0, msg3
   syscall
   li $v0, 1
   move $a0, $t1
   syscall
   li $v0, 4
   la $a0, msg4
   syscall
   li $v0, 1
   move $a0, $t2
   syscall
   li $v0, 4
   la $a0, msg5
   syscall
   li $v0, 1
   move $a0, $t0
```

```
li $v0, 4
la $a0, msg6
syscall

li $v0, 5
syscall

move $t3, $v0

bne $t3, $0, done
j loop

done:
li $v0, 10
syscall
```

#### 运行结果如图:

```
Very Console

Please enter 1st number: 20
Please enter 2nd number: 50
The result of 20 & 50 is: 70
Do you want to try another (0âcontinue/1âexit):
```

## 2.3 把 C 代码翻译成 MIPS 代码

```
翻译代码(源码见 p4. asm)
     .data
       .word 9, 7, 15, 19, 20, 30, 11, 18 # 数组 arrs
arrs:
                                     # 数组长度 N
       .word 8
N:
                                       # 输出的字符串
       .asciiz "The result is: "
msg:
       .text
       .globl main
sumn:
       # 参数: $a0 -> arr, $a1 -> n
       # 返回值: $v0 -> sum
              $t0, $0
                           \# sum = 0
       move
              $t1, $0
                           # idx = 0
       move
loop:
       slt
              $t2, $t1, $a1
```

```
$t2, 0, done
       beq
               $t3, $t1, 2 # 计算 arr[idx] 的地址偏移 (idx * 4)
       sll
               $t3, $t3, $a0
                                # arr + (idx * 4)
       add
       lw
               $t4, 0($t3)
       add
               $t0, $t0, $t4
       addi
               $t1, $t1, 1
       j
               loop
done:
               $v1, $t0
       move
       jr
               $ra
main:
       la
               $a0, arrs
               $a1, N
       lw
       jal
               sumn
       li
               $v0, 4
       la
               $a0, msg
       syscall
       li
               $v0, 1
               $a0, $v1
       move
       syscall
       li
               $v0, 10
       syscall
```

### 2.4 优化代码

```
优化代码(源码见 fib-op. asm)

## Daniel J. Ellard -- 02/27/94

## fib-o.asm-- A program to compute Fibonacci numbers.

## An optimized version of fib-t.asm.

## main--

## Registers used:

## $v0 - syscall parameter and return value.

## $a0 - syscall parameter-- the string to print.

.text

main:

subu $sp, $sp, 32 # Set up main's stack frame:
```

```
sw $ra, 28($sp)
    sw $fp, 24($sp)
    addu $fp, $sp, 32
   ## Get n from the user, put into $a0.
   li $v0, 5
                            # load syscall read int into $v0.
    syscall
                            # make the syscall.
   move $a0, $v0
                           # move the number read into $a0.
    jal fib
                            # call fib.
   move $a0, $v0
    li $v0, 1
                               # load syscall print_int into
$v0.
                            # make the syscall.
   syscall
   la $a0, newline
   li $v0, 4
   syscall
                           # make the syscall.
   li $v0, 10
                           # 10 is the exit syscall.
   syscall
                            # do the syscall.
## fib-- (hacked-up caller-save method)
## Registers used:
## $a0 - initially n.
## $t0 - parameter n.
## $t1 - fib (n - 1).
## $t2 - fib (n - 2).
.text
fib:
   bgt $a0, 1, fib_recurse # if n < 2, then just return a 1,
                           # don't build a stack frame.
   li $v0, 1
   jr $ra
# otherwise, set things up to handle
fib recurse:
                            # the recursive case:
    li $t0,0
   li $t3, 0
   li $t4, 1
   li $t5, 1
loop:
    slt $t1, $t0, $a0
   beq $t1, 0, done
```

```
add $t5, $t3, $t4

move $t3, $t4

move $t4, $t5

addi $t0, $t0, 1

j loop

done:

move $v0, $t5

jr $ra

## data for fib-o.asm:
.data

newline: .asciiz "\n"
```

说明: 优化部分在于把递归改写为循环, 使得求解的时间复杂度降为 0(n)。

## 3 实验结论

无

# 4 实验感想

学到了很多 MIPS 语言及应用,在学习过程中常与 C 语言作比较,对照着学可以轻松不少。