南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

计算机组成原理实验报告四

姓名：　卫焱滨　　学号：11710823

一、实验内容

1. 分别用循环与递归方式实现求解斐波那契数 F(n)，输出 F(5)并比较两者执行的指令条数 .

2. 使用 SPIM 系统调用读入整形数. 若整数非正，输出提示“Invalid Entry”并终止；否则打印输出各数位的名称，用空格分割.

3. 通过实现两个函数：

   1. test_prime (n) 若 n 为质数返回 1 否则返回 0
   2. main()　　遍历整数测试其是否为质数，并将前 100 个打印输出

二、实验步骤（阐述代码思路或操作步骤）

1.

题意分析：

分别用循环与递归方式实现求解斐波那契数 F(n)，输出 F(5)并比较两者执行的指令条数 .

操作思路和步骤 ：

1. 对于代码 1，利用过程的递归调用求解斐波那契数，设定 n 为 5 调用递归过程求解 F（5）并利用系统调用打印输出

2. 对于代码 2，循环求解斐波那契数，其 MIPS 代码操作与以下 C 代码相对应：

```
#MIPS 代码基本与以下 C 代码对应
#int fib(int n) {   //循环
#   int fn;
#   int f1 = 1;
#   int f2 = 1;
#   int i = 2;
#   if(n < 1) fn = 0, return fn;
#   if(n == 1) fn = 1, return fn;
#   if(n == 1) fn = 1, return fn;
#   else{
#       for(i ;i<n;i++) {
#           fn = f1 + f2;
#           f1 = f2;
#           f2 = fn;
#       }
#       return fn;
#   }
# }
```

具体 MIPS 代码分别对应哪行已经在代码内用红色注释标出

设置 n=5，利用以上代码对应的 MIPS 循环求解 F(5)，系统调用打印输出

3. 利用 Mars 自带的 Instruction Counter 分别统计两者的指令数目，结果截屏贴在报告的实验结果部分

代码 1 递归求 fib 数:

```
#使用 Lab 课件给定的递归求 fib 代码求解 fib (5)
.text
main:
li $a0 5    #n=5
jal fib #调用 fib 过程

#将答案按照 syscall 要求输出
move $a0 $v0
li $v0 1
syscall
# Exit
li $v0 10
syscall

#递归求 fibonacci
fib:    addi $sp $sp -12
    sw $ra 8($sp)
    sw $s0 4($sp)
    sw $a0 0($sp)
    bgt $a0 $0 test2
    add $v0 $0 $0
    j rtn
    test2: addi $t0 $0 1
     bne $t0 $a0 gen
     add $v0 $0 $t0
     j rtn
    gen:    subi $a0 $a0 1
    jal fib
    add $s0 $v0 $0
    subi $a0 $a0 1
    jal fib
    add $v0 $v0 $s0
    rtn:    lw $a0 0($sp)
    lw $s0 4($sp)
    lw $ra 8($sp)
    addi $sp $sp 12
    jr $ra
```

代码 2 循环求 fib 数:

```
#循环求解 fib (5)
```

```
.text
main:
    li $s0 5                #int n=5
    li $s1 1                #int f1=1
    li $s2 1                #int f2=1
    li $s3 2                #int i=2
    fib:
    blt $s0 1 less0         #if(n<1) jump to less0
    beq $s0 1 equ1          #if(n==1) jump to equ1
    beq $s0 2 equ2          #if(n==2) jump to equ2
    j loop                  #else n>=3 jump to loop to compute
    less0: li $s7 0             #if(n<1) fn=0 return fn
         j print
    equ1:    li $s7 1           #if(n==1) fn=1 return fn
         j print
    equ2:    li $s7 1           #if(n==2) fn=1 return fn
         j print
    loop:                       #if (n>=3) for(i ;i<n;i++) {
        add $s7 $s1 $s2         #           fn = f1 + f2;
        move $s1 $s2            #           f1 = f2
        move $s2 $s7            #           f2 = fn
        addi $s3 $s3 1
        blt $s3 $s0 loop        #}
        j print                 # return fn;

    # print answer
    print:
    li $v0 1
    move $a0 $s7
    Syscall

    # Exit
    li $v0 10
    syscall
```

2.

题意分析:

使用 SPIM 系统调用读入整形数. 若整数非正, 输出提示"Invalid Entry"并终止; 否则打印输出各数位的名称, 用空格分割.

操作步骤与思路 :

1. 在内存(.data) 中定义各数字对应数位的名称(.asciiz)、提示信息(.asciiz)和换行符(.asciiz)
2. 调用 syscall , 读入整形数 n, 并存于寄存器 v0 中

3. 若 n 非正，跳转到标签 Messege 处打印输出
4. 计算分割和储存数位:
   初始化$s0 寄存器的值为输入数 n
   利用循环，令$s0 除 10，其商存在寄存器$s0 内，作为下次循环的被处理数
   余数即为当前数位，按序存放于栈顶
   用$s2 作为计数器计循环次数
5. 输出数位名:
   利用循环，从栈顶一一取出数位值，参照内存中定义的对应名称进行输出
   输出使用分支结构（Branch）实现
6. 退出程序

代码:

```
    .data
        Invalid: .asciiz "Invalid Entry.\n"
        n1:.asciiz "One "
        n2:.asciiz "Two "
        n3:.asciiz "Three "
        n4:.asciiz "Four "
        n5:.asciiz "Five "
        n6:.asciiz "Six "
        n7:.asciiz "Seven "
        n8:.asciiz "Eight "
        n9:.asciiz "Nine "
        n0:.asciiz "Zero "
    .text

    #read the number n
     li $v0 5
     syscall

    #if n not positive ,print message
     blt $v0 1 Message

    #loop to figure out and store the names of the digits
    #method:
    #Initially,$s0 is the input number n
    #Every loop $s0/10 ,the reminder is digit number,push to stack, and the
quocient is assigned to update $s0
    #Use a counter $t2 record the times of loops
     move $s0 $v0
     li $t0 10
     li $t2 0
```

```
 loop:
    div $s0 $t0
    mflo $s0
    subi $sp $sp 1
    mfhi $t1
    sb $t1 0($sp)
    addi $t2 $t2 1
    bne $s0 0 loop

 #loop to print the names of the digits
 #method:
 #Fetch digits from the stack top one by one
 #For every digit : print corresponding name use branch
 #Until all the number of digits has been fetched and handled
 li $v0 4
 loop2:
    lb $t0 0($sp)
    beq $t0 0 print0
    beq $t0 1 print1
    beq $t0 2 print2
    beq $t0 3 print3
    beq $t0 4 print4
    beq $t0 5 print5
    beq $t0 6 print6
    beq $t0 7 print7
    beq $t0 8 print8
    beq $t0 9 print9
test:   addi $sp $sp 1
    subi $t2 $t2 1
    beq $t2 0 Exit
    bne $t2 0 loop2

# branch to print name of digits
print0:la $a0 n0
    syscall
    j test
print1:la $a0 n1
    syscall
    j test
print2:la $a0 n2
    syscall
    j test
```

```
print3:la $a0 n3
    syscall
    j test
print4:la $a0 n4
    syscall
    j test
print5:la $a0 n5
    syscall
    j test
print6:la $a0 n6
    syscall
    j test
print7:la $a0 n7
    syscall
    j test
print8:la $a0 n8
    syscall
    j test
print9:la $a0 n9
    syscall
    j test

#dismiss the condition (n is positive)
 Message:
 li $v0 4
 la $a0 Invalid
 syscall

 #exit safely
 Exit:
 li $v0 10
 syscall
```

3.

题意分析:

通过实现两个函数:

test_prime (n) 若 n 为质数返回 1 否则返回 0

main() 遍历整数测试其是否为质数，并将前 100 个打印输出

操作步骤与思路 ：

1. 在内存中定义空格和换行（.asciiz）
2. test_prime (n)函数 ： 循环遍历 2 到 n-1，检验是否整除 n，若整除，函数返回 0,

否则返回 1

3. main() ： .globa 函数 从 2 开始循环枚举正整数，调用 test_prime (n)函数检验其是否为质数，若为质数，将其打印输出（相邻两质数间用空格分隔，10 个为一组换行）

4. 安全退出程序

代码:

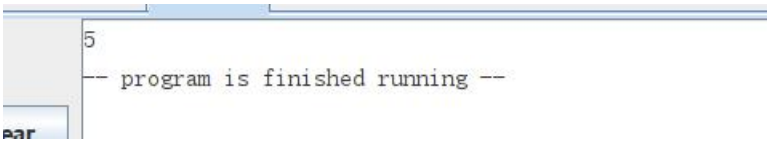
```
.data
    spac: .asciiz " "
    n: .asciiz"\n"
.text
.globl main
main:
    li $s0 1
    li $s1 1
    li $s2 10
    loop2:
         addi $s1 $s1 1
         move $a0 $s1
         jal test_prime
         beq $v0 1 print
    loopp: blt $s0 101 loop2
         li $v0 10
         syscall
    print:
        li $v0 1
        syscall
        la $a0 spac
        li $v0 4
        syscall
        div $s0 $s2
        mfhi $t3
        bne $t3 0 noline
        li $v0 4
        la $a0 n
        syscall
    noline:
        addi $s0 $s0 1
        j loopp



test_prime:
```

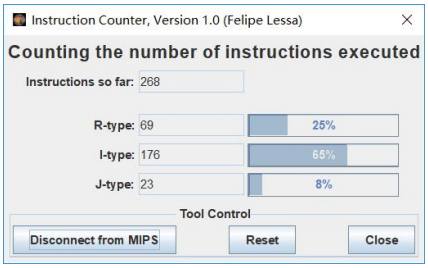
```
li $t0 2
  li $v0 1
loop: div $a0 $t0
    mfhi $t1
    beq $t1 0 rtn
    addi $t0 $t0 1
    blt $t0 $a0 loop
    jr $ra
rtn:  li $v0 0
    jr $ra
```

三、实验结果（截图并配以适当的文字说明）

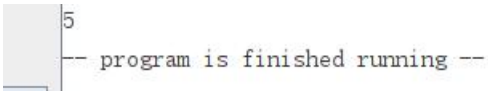
1.

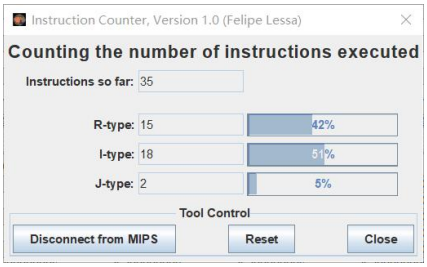
代码 1（递归）计算 F(5)运行结果和指令条数统计截图如下:





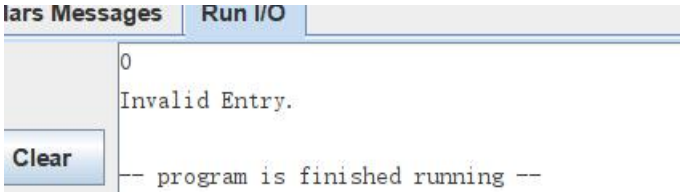
代码 2（循环）计算 F(5)运行结果和指令条数统计截图如下:





比较得到，循环实现求解斐波那契数指令数少于递归实现

2. 对于非正的整数输入，输出如下:



对于正整数输入，举例输出如下：

```
12345
One Two Three Four Five
-- program is finished running --
```

3. 输出前 100 个质数如下：

```
3 5 7 11 13 17 19 23 29 31
37 41 43 47 53 59 61 67 71 73
79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179
181 191 193 197 199 211 223 227 229 233
239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353
359 367 373 379 383 389 397 401 409 419
421 431 433 439 443 449 457 461 463 467
479 487 491 499 503 509 521 523 541 547

-- program is finished running --
```

四、实验分析（遇到的问题以及解决方案）

问题 1：Lab4_3 输出结果过长，屏幕无法装下

解决：引入 counter，运用分支结构，在输出时进行判断。每 10 个数字进行换行操作