





无03_王治_homework_数逻_MIPS大作业_01

 无03 王治 2020010699

实验目的

 在 Mars 模拟器上编写 MIPS 汇编指令，包括指令的编写，编译，运行，调试

实验内容

1.Experiment_01_01_syscall–系统调用

1) 功能

- (1) 申请一个 8byte 整数的内存空间。
- (2) 从“a.in”读取两个整数。
- (3) 向”a.out”写入这两个整数。
- (4) 从键盘输入一个整数 i。
- (5) $i = i + 1$ 。
- (6) 向屏幕打印这个整数

2) 实现

syscall 指令：

```
li $v0, 1    # 1 代表打印整数
li $v0, 5    # 5 代表输入一个整数
li $v0, 13   # 13 为打开文件的 syscall 编号
li $v0, 14   # 14 为读取文件的 syscall 编号
li $v0, 15   # 15 是写入文件的 syscall 编号
li $v0, 16   # 16 是关闭文件的 syscall 编号
```

2.Experiment_01_02_loop–循环分支

1) 功能

- (1) 将输入值取绝对值，存在变量 i, j 中
- (2) 从变量 i 开始，循环 j 轮，每轮 $i = i+1$

2) 实现

syscall 指令：

```
li $v0, 1    # 1 代表打印整数
```

```
li $v0, 5    # 5 代表输入一个整数
```

mips 指令：

```
slt $t1, $t2, $t3    #if(t2<t3) t1=1
```

```
beq $a0, $a1, label  #jump to label, if a0==a1
```

```
j loop            #跳转至 loop
```

```
move $a0, $t0      #将 t0 寄存器的值放到 a0 寄存器中
```

```
blt $a0, $a1, label #jump to label, if a0<a1
```

3.Experiment_01_03_array–数组指针

1) 功能

- (1) 输入数组 a 的长度 n
- (2) 任意输入 n 个整数
- (3) 将数组 a 逆序，并且仍然存储在 a 中
- (4) 打印数组 a 的值

2) 实现

syscall 指令：

```
li $v0 1      # 1 代表打印整数
```

```
li $v0 4      # 4 代表打印字符串
```

```
li $v0 5      # 5 代表输入一个整数
```

```
li $v0 9      # 9 代表分配动态内存
```

mips 指令：

```
j loop            #跳转至 loop
```

```
move $a0, $t0      #将 t0 寄存器的值放到 a0 寄存器中
```

```
mul $s2,$s1,$s0    #换算地址
```

```
bge $t1, $t0, label # if t1>=t0,jump to label
```

4.Experiment_01_04_hanoi–函数调用

1) 功能

- (1) 汉诺塔问题：输入 n
- (2) 递推公式： $Hanoi(n) = 2 * Hanoi(n - 1) + 1$

2) 实现

syscall 指令：

```
li $v0 1      # 1 代表打印整数
li $v0 4      # 4 代表打印字符串
li $v0 5      # 5 代表输入一个整数
```

mips 指令：

```
jal product    #跳转到子过程 product
bne $a0, $a1 ,label #jump to label, if a0!=a1
sw $t3, 500($t4) # store word
lw $t1, 30($t2) # load word
jr $ra         # 跳回上一级程序(ra 为返回地址)
```

实验小结

我在此次大作业中学习到了利用 MARS 模拟器中编写 MIPS 汇编指令的过程，也进一步掌握了一些 MIPS 的指令，比如 LW SW BLT 等指令，而且对于工程的流程，包括编写，编译，运行，调试等都有了基础的了解。

附：源代码&效果

1.Experiment_01_01_syscall

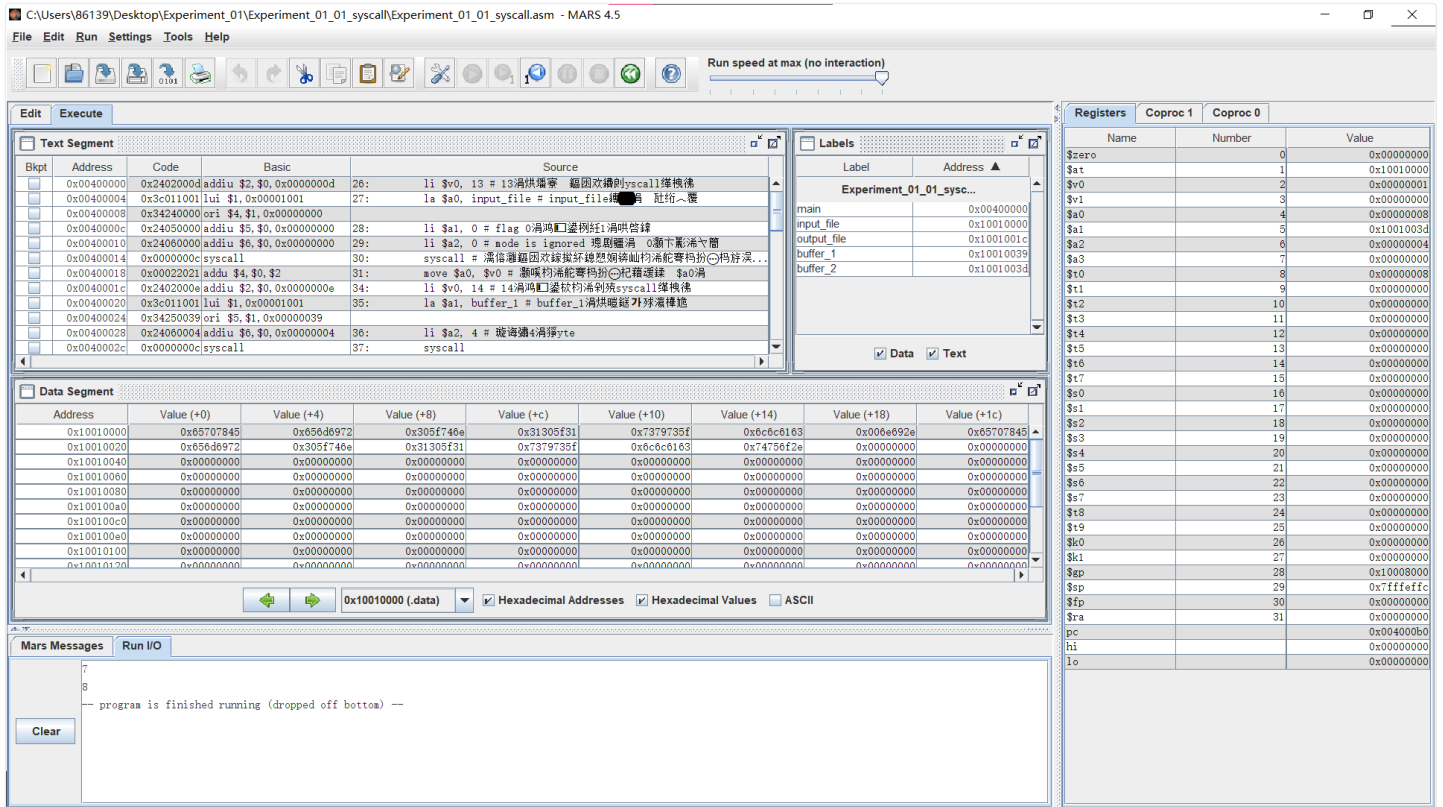
```
1  # #####
2  # Author:TX-Leo
3  # Tool Versions: Mars4_5 for Mips
4  # Create Date: 2022/05/01
5  # Project Name: Experiment_01_01_syscall.asm
6  # Source File: Experiment_01_01_syscall.cpp
7  # Description: It's TX-Leo's Experiment_01_01
8  # Function:
9  # (1)  申请一个8byte整数的内存空间。
10 # (2)  从"a.in"读取两个整数。
11 # (3)  向"a.out"写入这两个整数。
12 # (4)  从键盘输入一个整数i。
13 # (5)  i = i +1。
14 # (6)  向屏幕打印这个整数
15 # #####
16
17 .data # 数据段
18     input_file: .asciiz "Experiment_01_01_syscall.in" # 表示字符串，其中asciiz会自动在最后补上null字符
19     output_file: .asciiz "Experiment_01_01_syscall.out"
20     buffer_1: .space 4 # .space表示一个以byte计长度的数组
21     buffer_2: .space 4
22
23 .text # 代码段
24     open_file_for_reading:
25         # Open "Experiment_01_01_syscall.in" for reading
26         li $v0, 13 # 13为打开文件的syscall编号
27         la $a0, input_file # input_file是一个字符串
28         li $a1, 0 # flag 0为读取，1为写入
29         li $a2, 0 # mode is ignored 设置为0就可以了
30         syscall # 如果文件打开成功，文件描述符返回到v0
31         move $a0, $v0 # 将文件描述符载入到$a0中
```

MATLAB

```

30      # Read integer_01 from "Experiment_01_01_syscall.in"
31      li $v0, 14          # 14为读取文件的syscall编号
32      la $a1, buffer_1    # buffer_1为数据暂存区
33      li $a2, 4           # 读取4个byte
34      syscall
35
36      # Read integer_02 from "Experiment_01_01_syscall.in"
37      li $v0, 14          # 14为读取文件的syscall编号
38      la $a1, buffer_2    # buffer_2为数据暂存区
39      li $a2, 4           # 读取4个byte
40      syscall
41
42      # Close "Experiment_01_01_syscall.in"
43      li $v0, 16          # 16是关闭文件的syscall编号
44      syscall
45
46  open_file_for_writing:
47      # Open "Experiment_01_01_syscall.out" for writing
48      li $v0, 13          # 13为打开文件的syscall编号
49      la $a0, output_file # output_file是一个字符串
50      li $a1, 1           # flag 0为读取, 1为写入
51      li $a2, 0           # mode is ignored 设置为0就可以了
52      syscall
53      move $a0, $v0        # 将文件描述符载入到$a0中
54
55      # Write interger_01 in "Experiment_01_01_syscall.out"
56      li $v0, 15          # 15是写入文件的syscall编号
57      la $a1, buffer_1    # buffer_1为数据暂存区
58      li $a2, 4           # 写入4个byte
59      syscall
60
61      # Write interger_02 in "Experiment_01_01_syscall.out"
62      li $v0, 15          # 15是写入文件的syscall编号
63      la $a1, buffer_2    # buffer_2为数据暂存区
64      li $a2, 4           # 写入4个byte
65      syscall
66
67      # Close "Experiment_01_01_syscall.out"
68      li $v0, 16          # 16是关闭文件的syscall编号
69      syscall
70
71  input:
72      # Input an integer by keyboard
73      li $v0, 5           # 5代表输入一个整数
74      syscall            # 等待输入一个整数
75
76  add:
77      # i = i + 1
78      addi $t0, $v0, 1
79
80  print:
81      # Print the integer
82      li $v0, 1           # 1代表打印整数
83      move $a0, $t0       # 将整数存到a0中
84      syscall
85
86
87

```



2.Experiment_01_02_loop

MATLAB

```

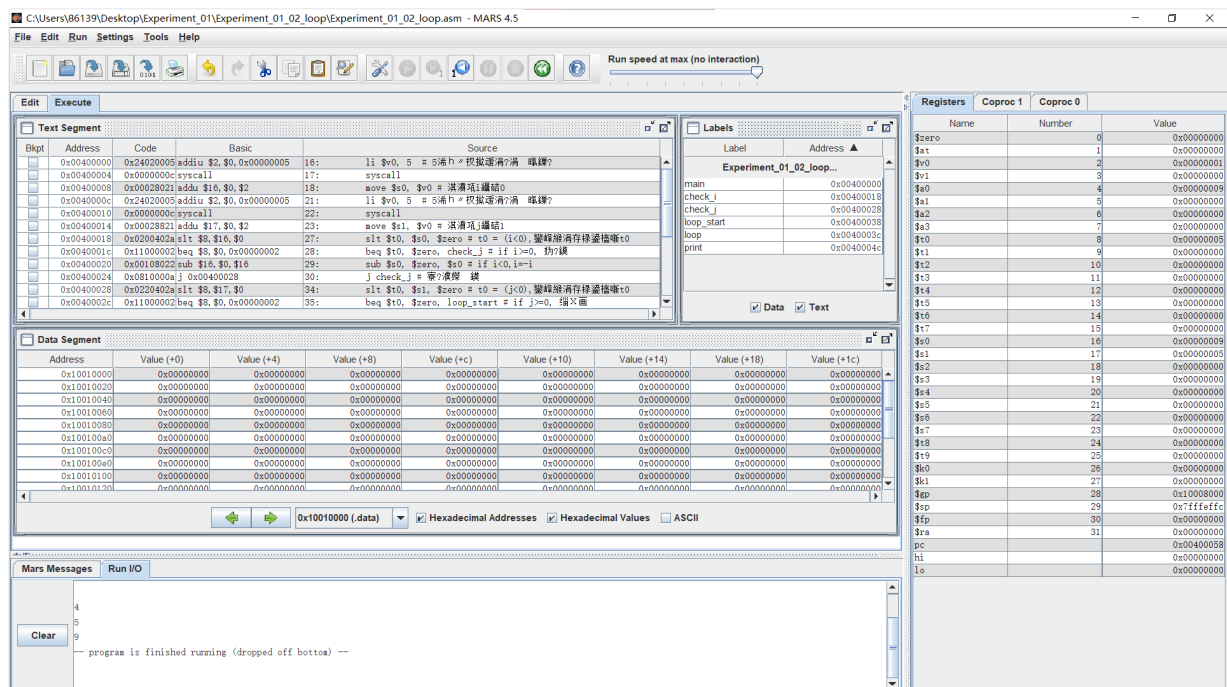
1  # #####
2  # Author:TX-Leo
3  # Tool Versions: Mars4_5 for Mips
4  # Create Date: 2022/05/01
5  # Project Name: Experiment_01_02_loop.asm
6  # Source File: Experiment_01_01_loop.cpp
7  # Description: It's TX-Leo's Experiment_01_02
8  # Function:
9  # (1) 将输入值取绝对值, 存在变量i, j中
10 # (2) 从变量i开始, 循环j轮, 每轮i = i+1
11 # #####
12
13 .text # 代码段
14     main:
15     # 主函数
16
17     # 输入i并且保存至s0
18     li $v0, 5 # 5代表输入一个整数
19     syscall
20     move $s0, $v0 # 保存i至s0
21
22     # 输入j并且保存至s1
23     li $v0, 5 # 5代表输入一个整数
24     syscall
25     move $s1, $v0 # 保存j至s1
26
27     check_i:
28     # 检查i
29     # 获得i的绝对值
30     slt $t0, $s0, $zero # t0 = (i<0), 获得临时变量t0
31     beq $t0, $zero, check_j # if i>=0, 检查j
32     sub $s0, $s0, $zero, $s0 # if i<0, i=-i
33     j check_j # 开始检查j
34
35     check_j:
36     # 检查j
37     # 获得j的绝对值

```

```

37      slt $t0, $s1, $zero          # t0 = (j<0), 获得临时变量t0
38      beq $t0, $zero, loop_start  # if j>=0, 继续
39      sub $s1, $zero, $s1         # if j<0, j=-j
40      j loop_start                # 开始循环
41
42  loop_start:
43  # loop_start
44      li $t0 0                    # 临时变量temp = 0
45
46  loop:
47  # loop
48      addi $s0, $s0, 1            # i++
49      addi $t0, $t0, 1            # temp++
50      blt $t0, $s1, loop          # if temp < j, 继续循环
51
52  print:
53  # print
54      li $v0, 1                  # 1是打印整数的标识
55      move $a0, $s0              # 将整数存到a0中
56      syscall

```



3.Experiment_01_03_array

```

1  # #####
2  # Author:TX-Leo
3  # Tool Versions: Mars4_5 for Mips
4  # Create Date: 2022/05/01
5  # Project Name: Experiment_01_03_array.asm
6  # Source File: Experiment_01_03_array.cpp
7  # Description: It's TX-Leo's Experiment_01_03
8  # Function:
9  # (1) 输入数组a的长度n
10 # (2) 任意输入n个整数
11 # (3) 将数组a逆序, 并且仍然存储在a中
12 # (4) 打印数组a的值
13 # #####
14
15 .data # 数据段
16     string_1: .asciiz "please input the length of the array:\n"
17     string_2: .asciiz "please input a["

```

MATLAB

```

18     string_3: .asciiz "]" : "
19     string_space: " "
20
21 .text # 代码段
22     input_demonstration:
23     # print input demonstration
24         li $v0, 4          # 4代表打印字符串
25         la $a0, string_1    # "please input the length of the array:\n"
26         syscall
27     input_array_length:
28     # input the length of the array
29         li $v0, 5          # 5代表输入一个整数 (数组的长度)
30         syscall
31         move $s0, $v0      # 保存整数至s0
32     create_array:
33     # get dynamic memory allocation
34         mul $t0, $v0, 4    # get max_offset  t0=v0*4是max_offset
35         move $a0, $t0      # 保存数组的大小size
36         li $v0, 9          #
37         syscall
38
39         move $s1, $v0      # 将动态内存的地址放在s1
40         li $t1, 0          # 将寄存器t1的值为0 (t1是现在的offset)
41
42         li $t8, 0          # temp
43     input_array:
44     # input and save n integers
45         bge $t1, $t0, end_input    # if offset>=max_offset, 输入结束
46         # input demonstration
47         li $v0, 4          # 4代表打印字符串
48         la $a0, string_2    # "please input a["
49         syscall
50
51         li $v0, 1          # 1是打印整数的标识
52         move $a0, $t8      # 将整数存到a0中
53         syscall
54
55         li $v0, 4          # 4代表打印字符串
56         la $a0, string_3    # "]"
57         syscall
58
59         li $v0, 5          # 5代表输入一个整数
60         syscall
61
62         add $t3, $t1, $s1    # a[i]的地址 t3=t1+s1
63         sw $v0, ($t3)        # 给a[i]赋值 {v0: (输入的整数), t3:(a[i]的地址)}
64
65         addi $t1, $t1, 4     # 更新offset
66         addi $t8, $t8, 1     # temp++
67         j input_array        # 循环input
68
69     end_input:
70     # end input
71         mul $t4, $s0, 2      #
72         li $t1, 0           # 更新现在的offset为0
73
74     reverse_array:
75     # reverse the array
76         bge $t1, $t4, end_reverse    # offset >= 2 * n, jump to end (equal to "i < n / 2")
77         add $t3, $t1, $s1    # addr of a[i]
78         lw $t5, ($t3)        # value of a[i]
79         sub $t6, $t0, $t1    # offset of a[n - i - 1]
80         add $t6, $t6, $s1    # addr of a[n - i - 1]
81         lw $t7, ($t6)        # value of a[n - i - 1]

```

The screenshot displays the Mars IDE interface. At the top, the title bar indicates the file path: C:\Users\B6139\Desktop\Experiment_01\Experiment_01_03_array\Experiment_01_03_array.asm - MARS 4.5. The menu bar includes File, Edit, Run, Settings, Tools, and Help. Below the menu is a toolbar with icons for file operations and execution. The main window is divided into several panes. The left pane shows the assembly code with columns for Bkpt, Address, Code, Basic, and Source. The code includes instructions like addiu, lui, ori, syscall, and mul. The right pane is split into two sections: 'Labels' and 'Registers'. The 'Labels' section shows a list of labels and their addresses, with 'Experiment_01_03_array...' highlighted. The 'Registers' section shows a table with columns for Name, Number, and Value, listing registers like \$8, \$12, \$13, and \$14. Below the assembly code is a 'Data Segment' pane showing a memory dump with addresses and hexadecimal values. At the bottom, the 'Mars Messages' pane shows a list of messages, including 'please input the length of the array:' and 'please input a[0]:1'. A 'Clear' button is visible next to the messages.

MATLAB

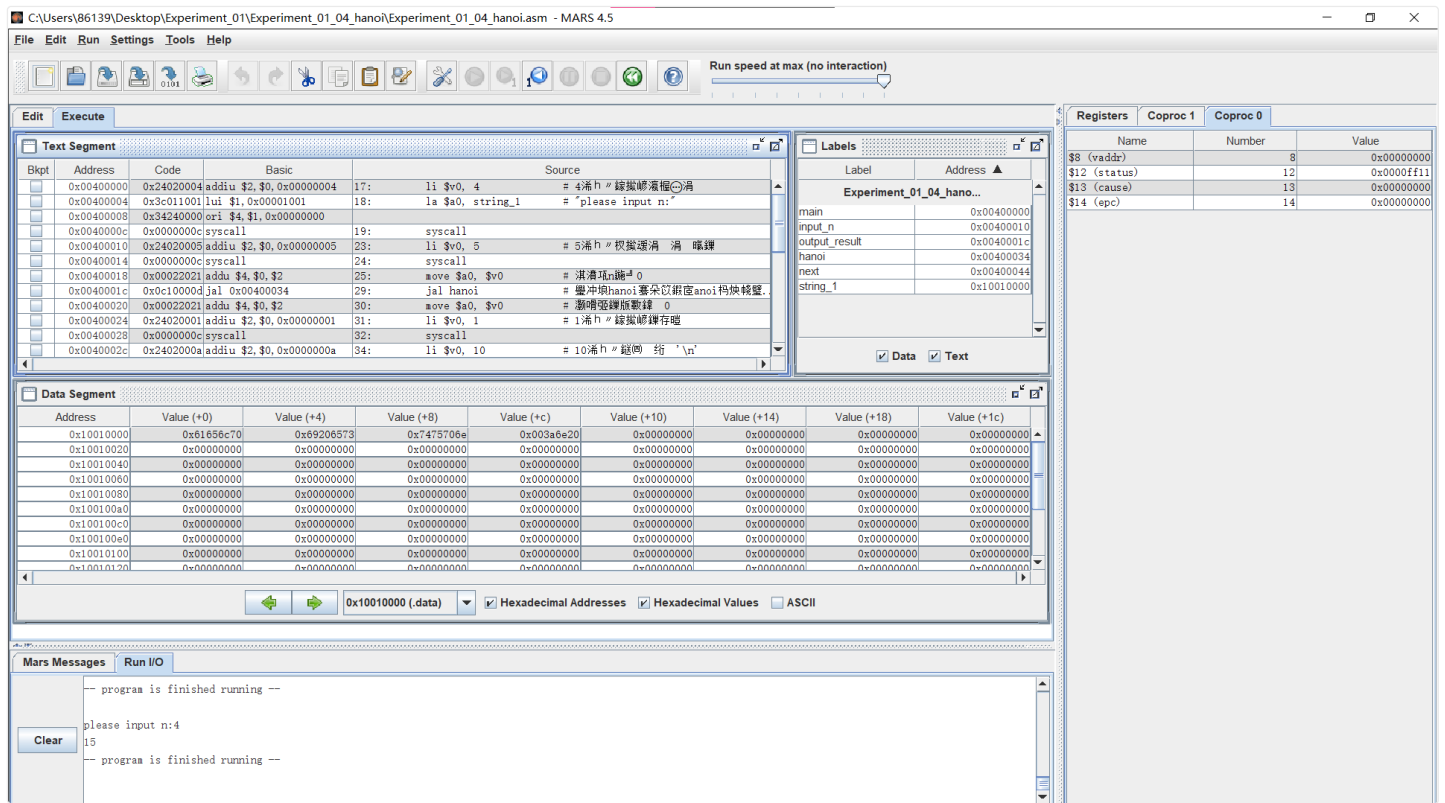
```
1 # #####
2 # Author:TX-Leo
3 # Tool Versions: Mars4_5 for Mips
4 # Create Date: 2022/05/01
5 # Project Name: Experiment_01_04_hanoi.asm
6 # Source File: Experiment_01_04_hanoi.cpp
7 # Description: It's TX-Leo's Experiment_01_04
```



```

8 # Function:
9 # (1) 汉诺塔问题: 输入n
10 # (2) 递推公式:  $Hanoi(n) = 2 * Hanoi(n - 1) + 1$ 
11 # #####
12 .data
13     string_1: .asciiz "please input n:"
14 .text
15     main:
16         # input demonstration
17         li $v0, 4           # 4代表打印字符串
18         la $a0, string_1    # "please input n:"
19         syscall
20
21     input_n:
22     # input n
23         li $v0, 5           # 5代表输入一个整数
24         syscall
25         move $a0, $v0       # 保存n在a0
26
27     output_result:
28     # output result from hanoi
29         jal hanoi           # 跳到hanoi并且和hanoi连接起来
30         move $a0, $v0       # 将参数放入a0
31         li $v0, 1           # 1代表打印整数
32         syscall
33
34         li $v0, 10          # 10代表换行符'\n'
35         syscall
36     hanoi:
37     # hanoi
38         li $t0, 1           # t0置为1
39         bne $t0, $a0, next   # if a0(输入的n)!=1, 进入循环next
40         li $v0, 1           # 1代表打印整数
41         jr $ra              # 跳回上一级程序(ra 为返回地址)
42     next:
43     # next
44         addi $sp, $sp, -8    # 移动堆栈指针
45         sw $a0, 0($sp)       # 保存 basic param
46         sw $ra, 4($sp)       # 保存寄存器地址
47
48         addi $a0, $a0, -1    # 更新参数 (n-1)
49         jal hanoi           # 跳到hanoi并且和hanoi连接起来( $Hanoi(n - 1)$ )
50
51         li $s1, 1
52
53         add $s1, $v0, $s1
54         add $s1, $v0, $s1
55         move $v0, $s1       # return  $2 * Hanoi(n - 1) + 1$ 
56
57         lw $a0, 0($sp)       # 更新参数
58         lw $ra, 4($sp)       # 更新寄存器地址
59         addi $sp, $sp, 8     # 更新堆栈指针
60
61         jr $ra              # 跳回上一级程序(ra 为返回地址)
62
63

```



文件清单：

Experiment_01

- Experiment_01_01_syscall
 - Experiment_01_01_syscall.asm
 - Experiment_01_01_syscall.cpp
- Experiment_01_02_loop
 - Experiment_01_02_loop.asm
 - Experiment_01_02_loop.cpp
- Experiment_01_03_array
 - Experiment_01_03_array.asm