计算机组成原理大作业

学号：　　姓名：

一、 斐波那契数列

1. 背景介绍

斐波那契数列，即

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, , 377……

满足 $F(n) = F(n-1) + F(n-2)$ 的一组数字。

2. C 语言代码实现

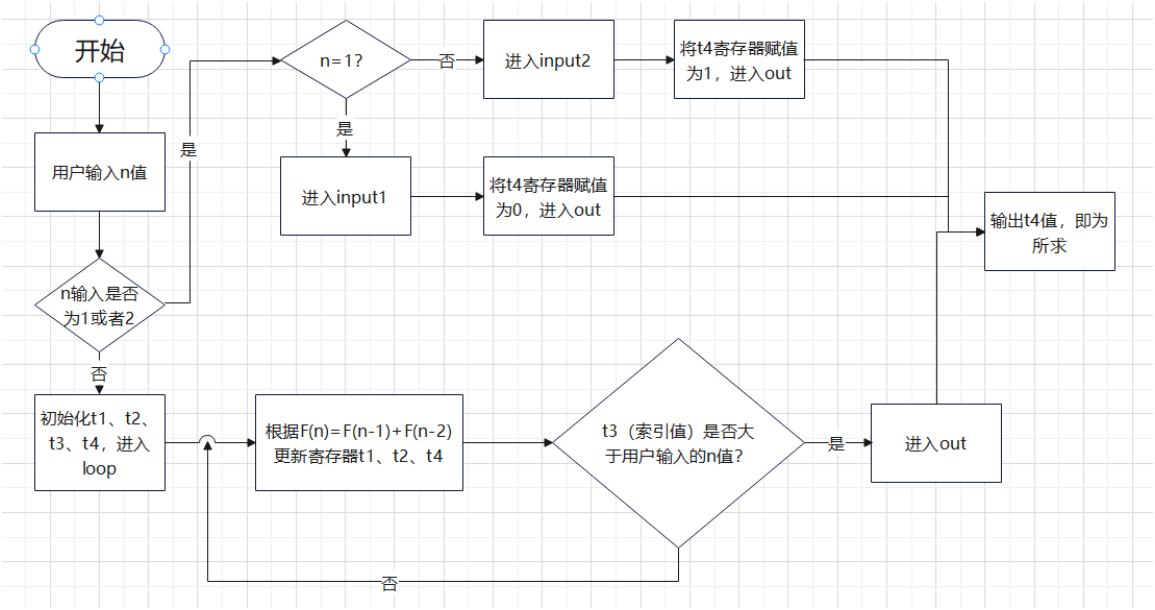
```c
int Fib(int n)
{
 if(n == 1) return 0;
 if(n == 2) return 1;

 return Fib(n-1) + Fib(n-2);
}
```

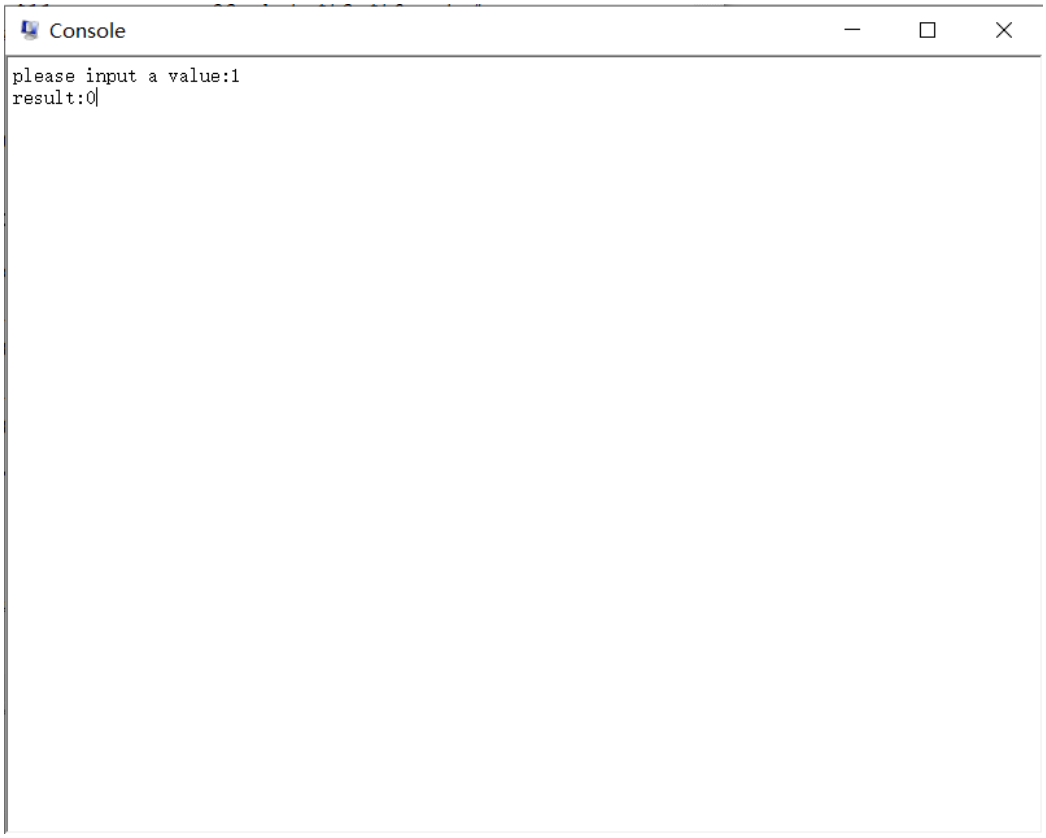
3. 寄存器功能介绍

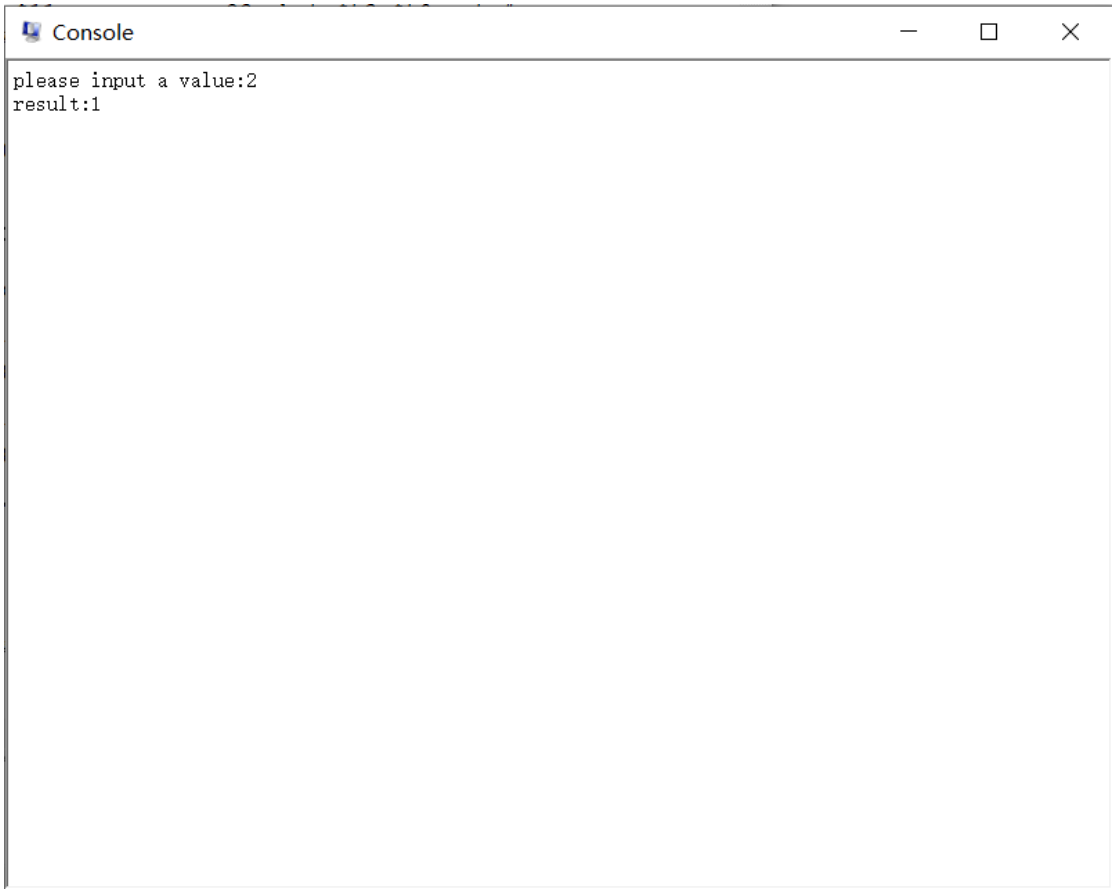
寄存器名称	功能
$\$t0$	储存用户输入的 n 值，即输出斐波那契数列的第几项
$\$t1$	F[n-2]，初始值为 0
$\$t2$	F[n-1]，初始值为 1
$\$t3$	当前项索引值
$\$t4$	F[n]

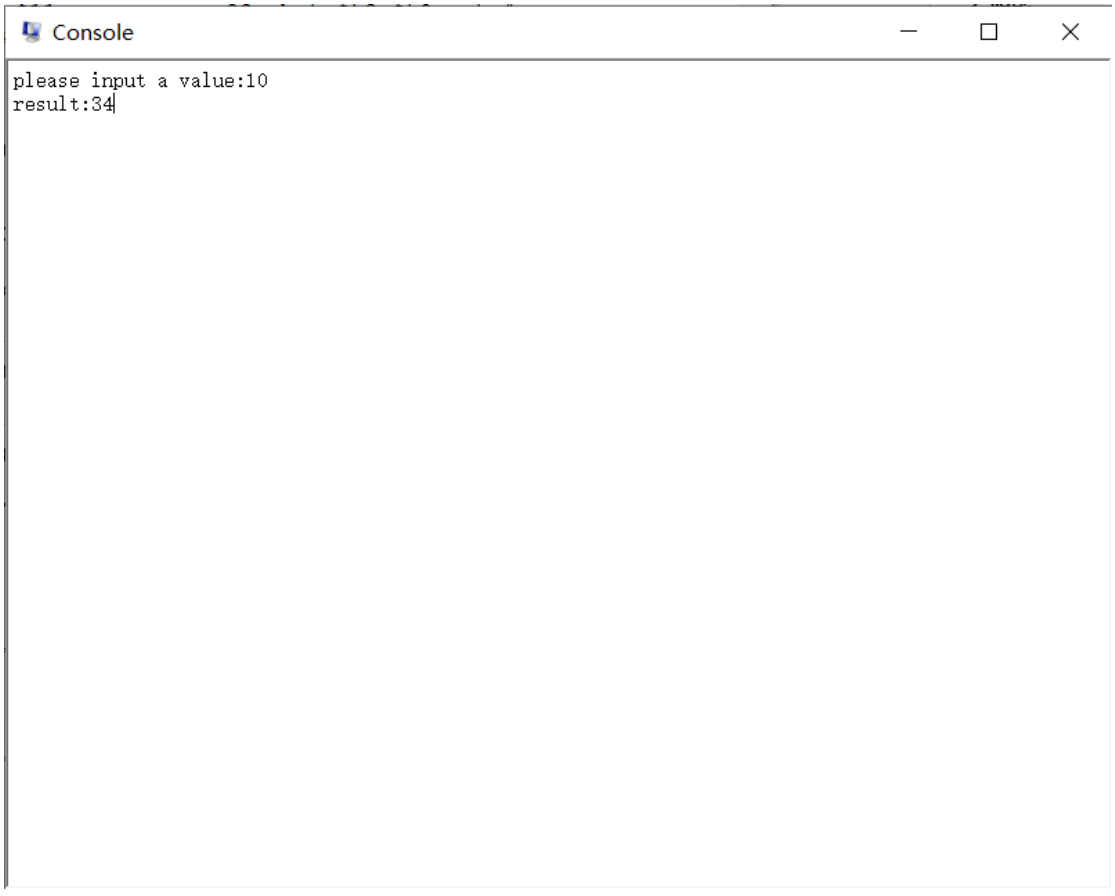
4. 代码结构图


开始
n=1?
否
进入input2
将t4寄存器赋值为1，进入out
是
用户输入n值
是
进入input1
将t4寄存器赋值为0，进入out
输出t4值，即为所求
n输入是否为1或者2
否
初始化t1、t2、t3、t4，进入loop
根据F(n)=F(n-1)+F(n-2)更新寄存器t1、t2、t4
t3（索引值）是否大于用户输入的n值?
是
进入out
否

5. 调试图


Console
please input a value:1
result:0


Console
please input a value:2
result:1


Console
please input a value:10
result:34

可以验证，以上结果均正确

6. 源代码

```
1. .data
2.  prompt:.asciiz "please input a value:"
3.  result:.asciiz "result:"
4. .text
5. main:
6.  li $v0,4
7.  la $a0,prompt # 提示用户输入
8.  syscall
9.
10. li $v0,5  # 系统调用把控制台中的数据读入寄存器
11. syscall
12.
13. move $t0,$v0 # 把斐波那契数列的第几项存入 t0
14.
15. beq $t0,1,input_1 # 如果输入为 1 或者 2，则直接跳转到输出部分
16. beq $t0,2,input_2
17.
18. li $t1,0  # 寄存器 t1，放置 f[n-2]，初始置为 0
19. li $t2,1  # 寄存器 t2，放置 f[n-1]，初始置为 1
20. li $t4,1  # 寄存器 t4，放置 f[n]，即当前项，初始置为 1
21. li $t3,4   # 寄存器 t3，当前项的索引，初始为 4，因为前 3 项已经被初始化
22.loop:
23. bgt $t3,$t0,out # 如果 t3 的值大于 t0 的值，即如果当前索引值大于用户输入索引值，则循环结束跳转到 out
24. move $t1,$t2 # 将寄存器 t2 的值赋值给 t1，更新 f[n-2]
25. move $t2,$t4 # 将寄存器 t4 的值赋值给 t2，更新 f[n-1]
26. add $t4,$t1,$t2 # 求和，更新 f[n]
27. addi $t3,$t3,1 # 当前索引项加一更新
28. j loop  # loop 循环
29.input_1:
30. li $t4,0  # 第 1 项，直接将当前项置为 0 即可跳转到 out
31. j out
32.input_2:
33. li $t4,1  # 第 2 项，直接将当前项置为 1 即可跳转到 out
34. j out
35.out:
36. li $v0,4
```

```
la $a0,result # 给用户输出提示
syscall

li $v0,1
move $a0,$t4 # 将当前项，即所求项赋值给a0，并输出
syscall

li $v0,10
syscall
```

二、冒泡排序

1. 背景介绍

常用的排序方式，它的基本思想是对所有相邻记录的关键字值进行比效，如果是逆顺（a[j]>a[j+1]），则将其交换，最终达到有序化。

2. C 语言代码实现

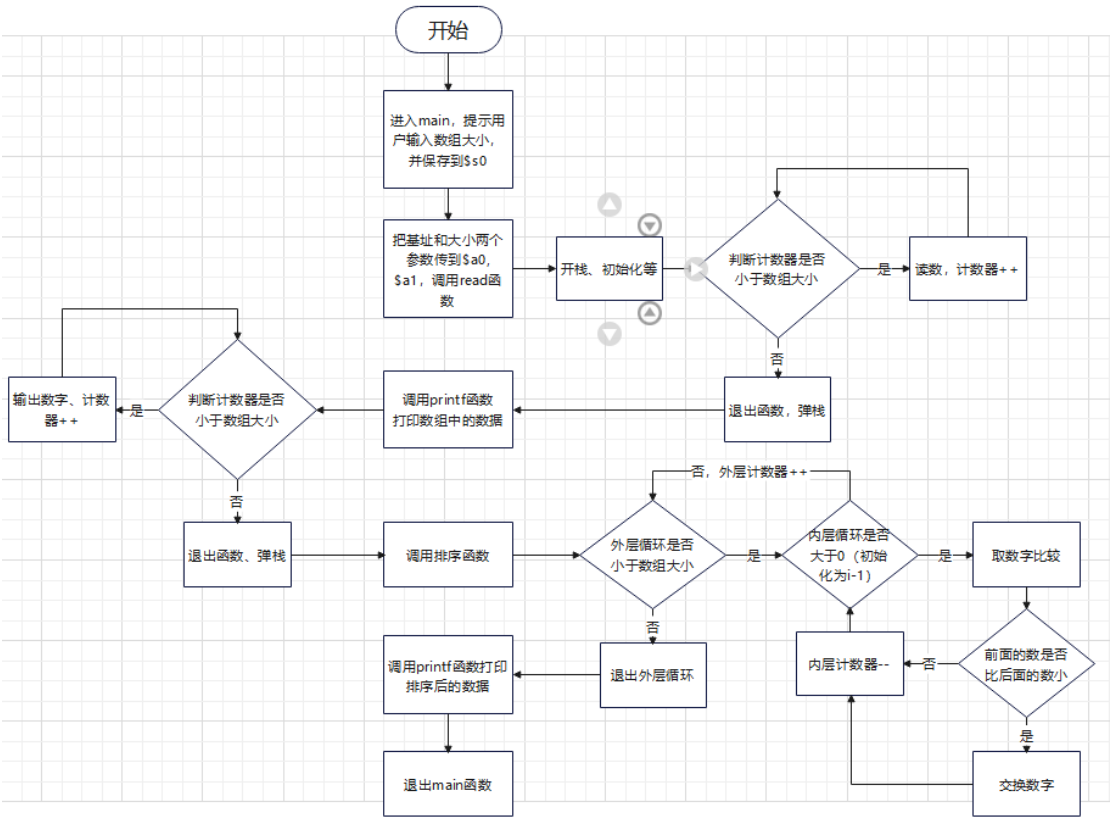
```
for(int i = 0; i < 10; ++i) // 从大到小排的冒泡
 for(int j = i; j < 10; ++j)
  if(a[j] < a[j + 1]) swap(a[j], a[j + 1]);
```

3. 寄存器功能介绍

寄存器名称	寄存器功能
$sp	栈地址

$s0	代表冒泡算法中的 i
$s1	代表冒泡算法中的 j
$s2	储存用户输入数字的基址
$s3	用户输入的数字个数
$ra	返回址
$a0、$a1	参数

4. 代码结构图



5. 调试图


Console
Please enter the number of integers:
5
Please enter the integers to be sorted:
66
8
42
65
12
The entered numbers are:
66, 8, 42, 65, 12,
The sorted numbers are:
66, 65, 42, 12, 8,


Console
Please enter the number of integers:
6
Please enter the integers to be sorted:
88
5
24
0
16
57
The entered numbers are:
88, 5, 24, 0, 16, 57,
The sorted numbers are:
88, 57, 24, 16, 5, 0,

以上测试结果均正确

6. 源代码

```
.text
.globl  main
# $gp 存数组基址
# $s0 存数组大小
# 函数调用的时候分别传给 a0 和 a1

main:
   la $a0, str_1               # 输出提示用户输入数组大小
   li $v0, 4
   syscall

   li $v0, 5                 # 系统调用把控制台中的数据读入寄存器
   syscall

   move $s0, $v0          # 把数组大小保存到 s0
   la $a0, str_2               # 输出提示用户开始输入数据
   li $v0, 4
   syscall

```

```
20.
21.   #调用read 函数
22.   move $a0, $gp             # 把$gp 作为参数传递给read 函数拿到
    数组基址，不要认为现在gp 里面是空
23.   move $a1, $s0
24.   jal read                  # 跳转到read 函数的同时保存主函数地
    址到$ra
25.
26.
27.   #调用打印函数打印刚才用户的输入结果
28.   li $v0, 4
29.   la $a0, str_5
30.   syscall
31.
32.   move $a0, $gp
33.   move $a1, $s0
34.   jal prinf
35.
36.
37.   #调用排序函数
38.   move $a0, $gp
39.   move $a1, $s0
40.   jal sort
41.
42.
43.   #调用输出函数
44.   li $v0, 4
45.   la $a0, str_3
46.   syscall
47.
48.   move $a0, $gp
49.   move $a1, $s0
50.   jal prinf
51.
52.
53.#从控制台读数据的read 函数
54.read:
55.   addi $sp, $sp, -4         # 栈中开辟1 个新地址保存数组元素个
    数
56.   sw $s0, 0($sp)
57.   li $s0, 0                    # 把s0 寄存器置零，作为读数个数
    的计数器
58.
59.   #下面是利用跳转语句和条件控制的读数循环
```

```
60.  #t0 做判断标志位，t1 做存储地址储存输入的数字，s0 用于计数（i），a0 为基址，a1 保存了总共的数字个数
61.  read_1:
62.    sltu $t0, $s0, $a1       # s0<a1 则t0=1（否则t0=0），即已读入的数的个数小于用户输入的为真
63.    beq $t0, $zero, exit_1  # t0=zero 则跳转到exit_1
64.    sll $t0, $s0, 2           # s0 左移两位
65.    add $t1, $a0, $t0        # a0 加上 t0 生成新地址
66.    move $t2, $a0
67.    li $v0, 5                 # 读数
68.    syscall
69.
70.    sw $v0, 0($t1)              # 保存读入的数据到主存
71.    move $a0, $t2
72.    addi $s0, $s0, 1          # s0++
73.    j read_1
74.
75.  exit_1:
76.    lw $s0, 0($sp)              # 把栈里面的东西（数组大小）写回寄存器
77.    addi $sp, $sp, 4          # 弹栈，就是堆栈指针归位
78.    jr $ra
79.
80.
81.#排序函数
82.sort:
83.  addi $sp, $sp, -20          # 在栈中开辟5个新地址
84.  #依次把要用变量的位置定出来并压栈
85.  sw $ra, 16($sp)            # 返回地址
86.  sw $s3, 12($sp)            # 数组大小
87.  sw $s2, 8($sp)           # 数组基址
88.  sw $s1, 4($sp)            # j
89.  sw $s0, 0($sp)            # i
90.  move $s2, $a0
91.  move $s3, $a1
92.  move $s0, $zero
93.
94.  forOut:
95.    slt $t0, $s0, $s3        # 如果i<n，则t0=1
96.    beq $t0, $zero, exit1     # 如果t0=0，跳转到exit1 退出外层循环
97.    addi $s1, $s0, -1        # j = i - 1
98.
99.  forIn:
```

```
100.      slti $t0, $s1, 0          # 如果s1（j）<0，则t0=1
101.      bne $t0, $zero, exit2     # 如果t0!=0，跳转到exit2
102.      sll $t1, $s1, 2           # $t1 = j*4
103.      add $t2, $s2, $t1   # $t2存了arr[j]的地址
104.      lw $t3, 0($t2)              # 取出arr[j]的数据到$t3
105.      lw $t4, 4($t2)              # 取出arr[j+1]的数据到$t4
106.      slt $t0, $t3, $t4        # 如果arr[j]<arr[j+1]，则
    t0=1
107.      beq $t0, $zero, exit2     # 不满足上面条件，跳转到
    exit2退出内层循环
108.      move $a0, $s2             # 把数组地址这个参数传给swap
    函数
109.      move $a1, $s1             # 另一个参数j也传过去
110.      jal swap
111.      addi $s1, $s1, -1         # j--
112.      j forIn
113.
114.    exit2:
115.      addi $s0, $s0, 1          # i++
116.      j forOut                  # 跳至外层循环
117.
118.    exit1:
119.      lw $s0, 0($sp)
120.      lw $s1, 4($sp)
121.      lw $s2, 8($sp)
122.      lw $s3, 12($sp)
123.      lw $ra, 16($sp)
124.      addi $sp, $sp, 20
125.      jr $ra
126.
127.    swap:
128.      sll $t0, $a1, 2       # j左移两位放到t0中
129.      add $t0, $a0, $t0    # 基址值加上偏移量arr[j]的地址
130.      lw $t1, 0($t0)          # 把arr[j]的值放入t1中
131.      lw $t2, 4($t0)          # 把arr[j+1]的值放入t2中
132.      sw $t1, 4($t0)          # arr[j]=arr[j+1]
133.      sw $t2, 0($t0)          # arr[j+1]=arr[j]
134.      jr $ra                 # 返回调用前的地址处
135.
136.
137.  #输出函数prinf部分，功能是打印一个数组的所有元素，实现和读入
    差不多
138.  prinf:
139.    addi $sp, $sp, -4
```

```
140.    sw $s0, 0($sp)          #保存寄存器s0 s1
141.    li $s0, 0                       #将s0置零
142.
143.    prinf_1:
144.      # 输出原数组元素
145.      sltu $t0, $s0, $a1
146.      beq $t0, $zero, exit_2
147.      sll $t0, $s0, 2
148.      add $t1, $a0, $t0
149.      move $t2, $a0
150.      lw $a0, 0($t1)
151.      li $v0, 1
152.      syscall
153.
154.      # 输出‘，’
155.      li $a0, ','
156.      li $v0, 11
157.      syscall
158.
159.      # s0 += 1
160.      move $a0, $t2
161.      addi $s0, $s0, 1
162.      j prinf_1
163.
164.    exit_2:
165.      lw $s0, 0($sp)
166.      addi $sp, $sp, 4
167.      jr $ra
168.
169.
170.  .data
171.  str_1:
172.    .asciiz "Please enter the number of integers:\n"
173.  str_2:
174.    .asciiz "Please enter the integers to be sorted:\n"
175.  str_3:
176.    .asciiz "\nThe sorted numbers are:\n"
177.  str_5:
178.    .asciiz "The entered numbers are:\n"
```