**计算机组成原理大作业**

学号： 姓名：

1. **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 的一组数字。

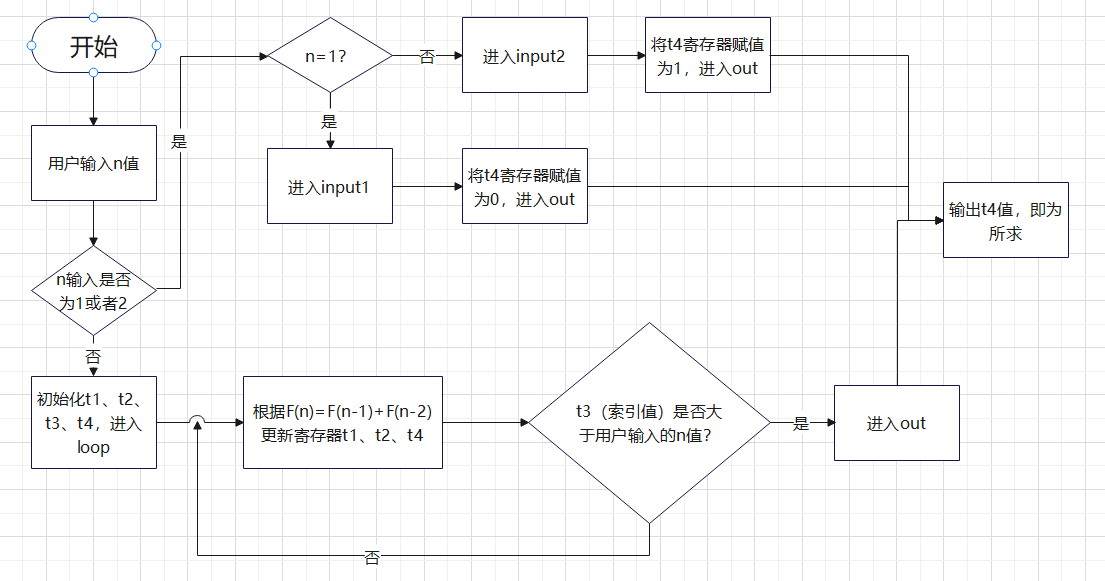
2.C语言代码实现

1. int Fib(int n)
2. {
3. if(n == 1) return 0;
4. if(n == 2) return 1;
6. return Fib(n-1) + Fib(n-2);
7. }

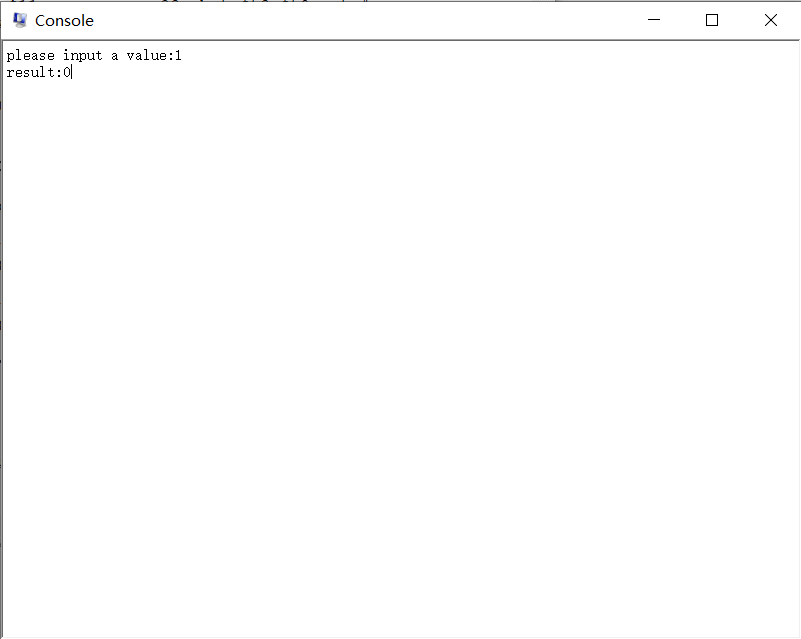
3.寄存器功能介绍

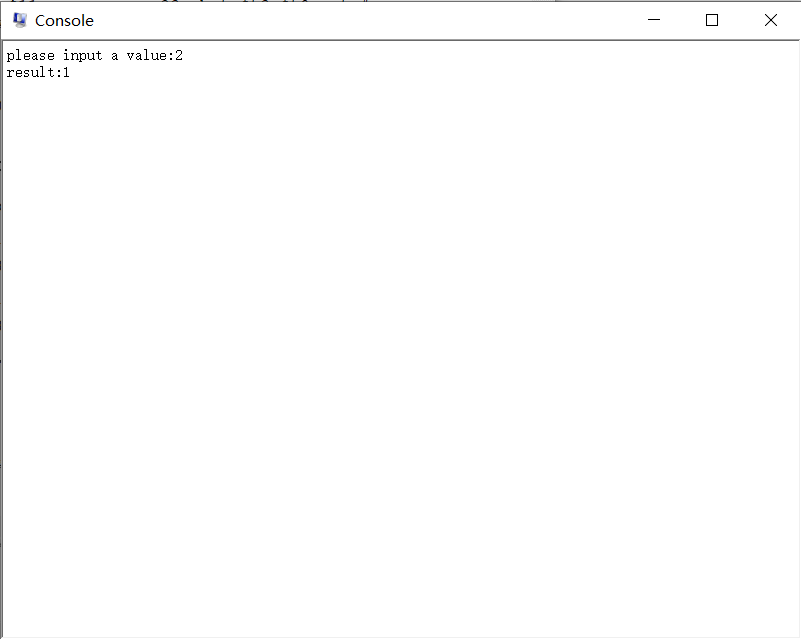
|  |  |
| --- | --- |
| 寄存器名称 | 功能 |
|  | 储存用户输入的n值，即输出斐波那契数列的第几项 |
|  | F[n-2]，初始值为0 |
|  | F[n-1]，初始值为1 |
|  | 当前项索引值 |
|  | F[n] |

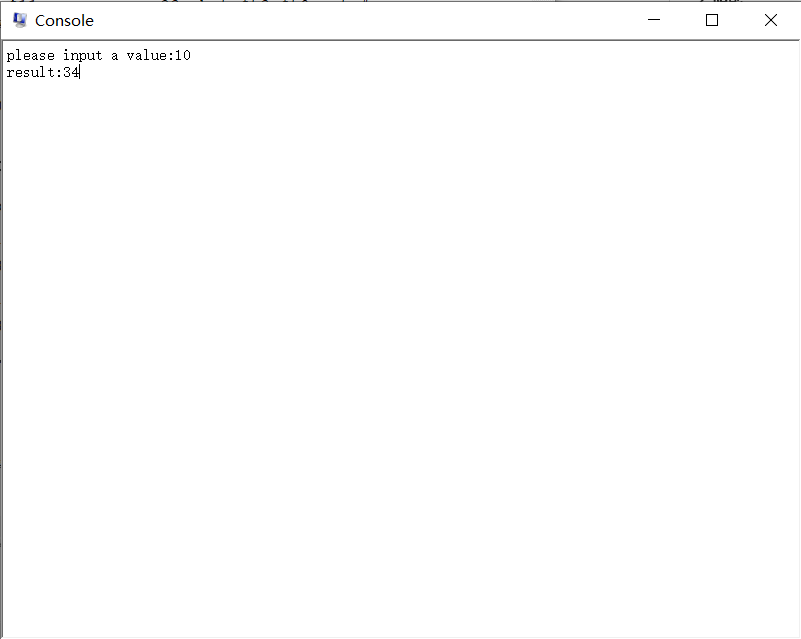
4.代码结构图



5.调试图







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. li $v0,5  *# 系统调用把控制台中的数据读入寄存器*
10. syscall
11. move $t0,$v0 *# 把斐波那契数列的第几项存入t0*
12. beq $t0,1,input\_1 *# 如果输入为1或者2，则直接跳转到输出部分*
13. beq $t0,2,input\_2
15. li $t1,0  *# 寄存器t1，放置f[n-2]，初始置为0*
16. li $t2,1  *# 寄存器t2，放置f[n-1]，初始置为1*
17. li $t4,1  *# 寄存器t4，放置f[n]，即当前项，初始置为1*
18. 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*
19. loop:
20. 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*
21. move $t1,$t2 *# 将寄存器t2的值赋值给t1，更新f[n-2]*
22. move $t2,$t4 *# 将寄存器t4的值赋值给t2，更新f[n-1]*
23. add $t4,$t1,$t2 *# 求和，更新f[n]*
24. addi $t3,$t3,1 *# 当前索引项加一更新*
25. j loop  *# loop循环*
26. input\_1:
27. li $t4,0  *# 第1项，直接将当前项置为0即可跳转到out*
28. j out
29. input\_2:
30. li $t4,1  *# 第2项，直接将当前项置为1即可跳转到out*
31. j out
32. out:
33. li $v0,4
34. la $a0,result *# 给用户输出提示*
35. syscall
36. li $v0,1
37. move $a0,$t4 *# 将当前项，即所求项赋值给a0，并输出*
38. syscall
40. li $v0,10
41. 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。

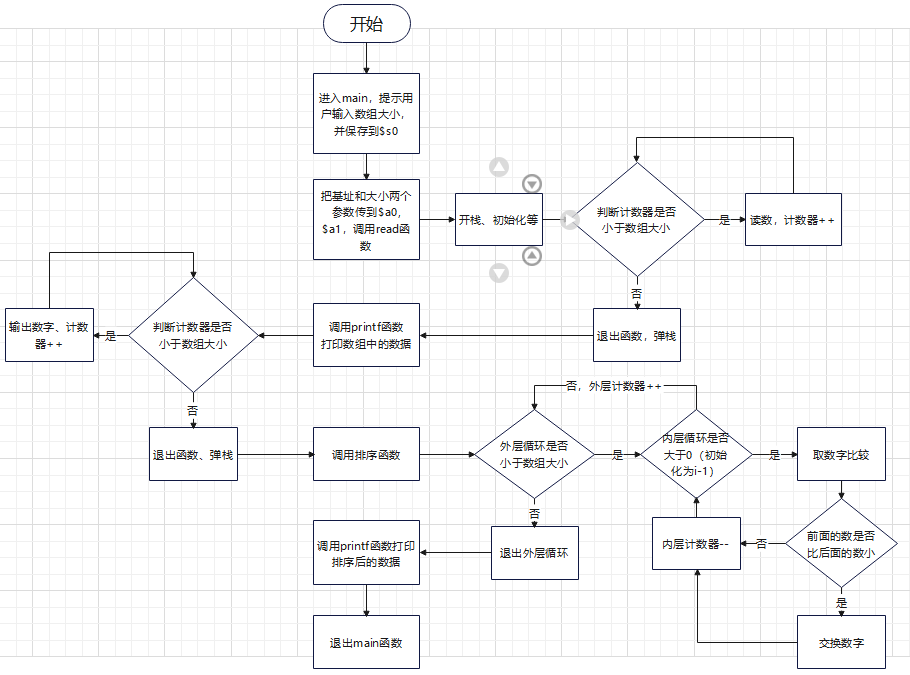
2. C语言代码实现

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

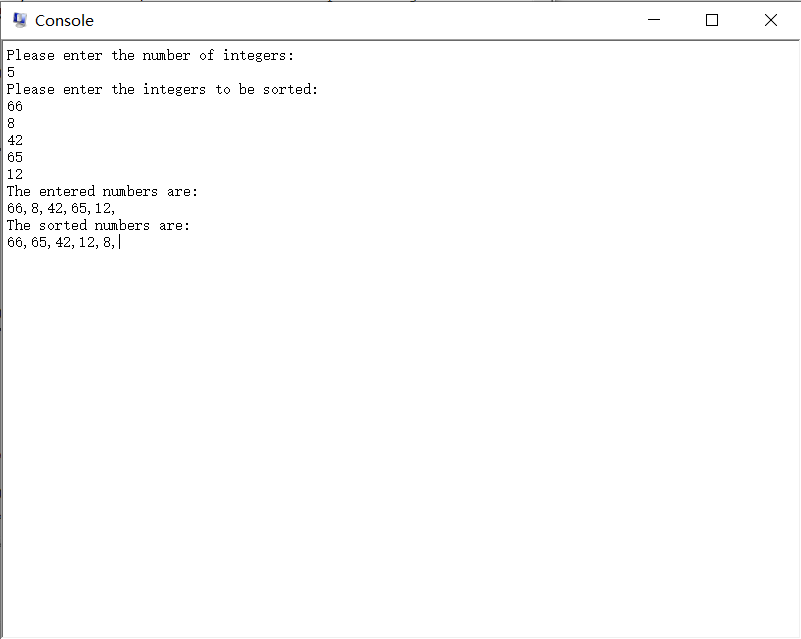
3. 寄存器功能介绍

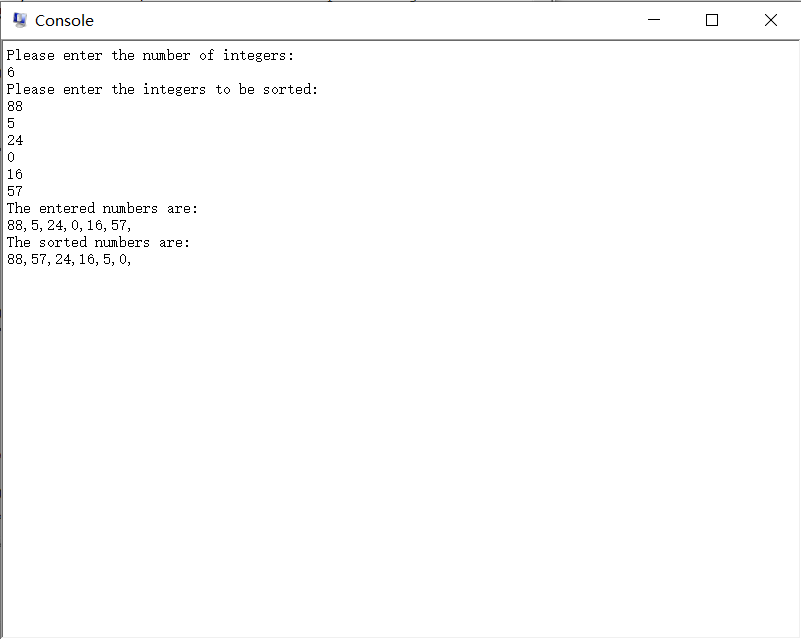
|  |  |
| --- | --- |
| 寄存器名称 | 寄存器功能 |
|  | 栈地址 |
|  | 代表冒泡算法中的i |
|  | 代表冒泡算法中的j |
|  | 储存用户输入数字的基址 |
|  | 用户输入的数字个数 |
|  | 返回址 |
|  | 参数 |

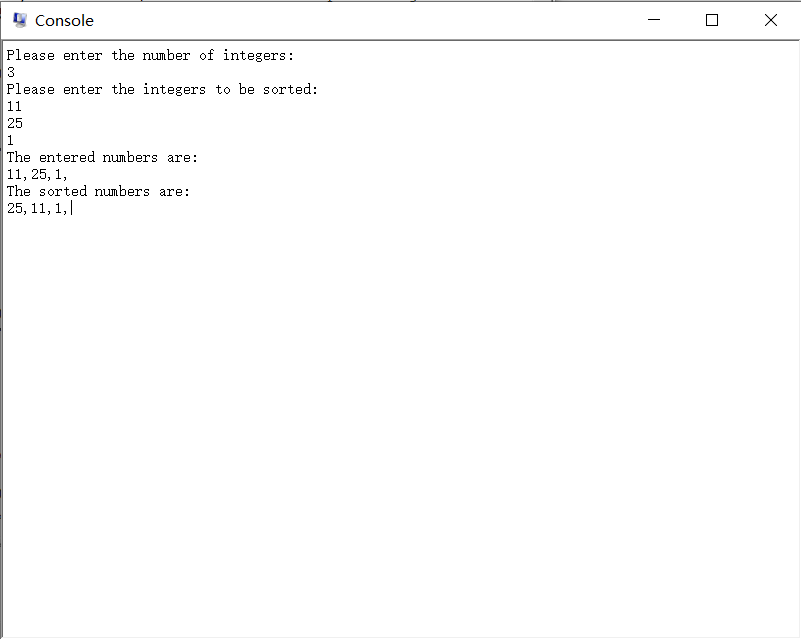
4. 代码结构图



5. 调试图







以上测试结果均正确

6. 源代码

1. .text
2. .globl  main
3. *# $gp存数组基址*
4. *# $s0存数组大小*
5. *# 函数调用的时候分别传给a0和a1*
6. main:
7. la $a0, str\_1             *# 输出提示用户输入数组大小*
8. li $v0, 4
9. syscall
10. li $v0, 5              *# 系统调用把控制台中的数据读入寄存器*
11. syscall
12. move $s0, $v0       *# 把数组大小保存到s0*
13. la $a0, str\_2             *# 输出提示用户开始输入数据*
14. li $v0, 4
15. syscall
16. *#调用read函数*
17. 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*
18. move $a1, $s0
19. jal read                *# 跳转到read函数的同时保存主函数地址到$ra*
20. *#调用打印函数打印刚才用户的输入结果*
21. li $v0, 4
22. la $a0, str\_5
23. syscall
24. move $a0, $gp
25. move $a1, $s0
26. jal prinf
27. *#调用排序函数*
28. move $a0, $gp
29. move $a1, $s0
30. jal sort
31. *#调用输出函数*
32. li $v0, 4
33. la $a0, str\_3
34. syscall
35. move $a0, $gp
36. move $a1, $s0
37. jal prinf
38. *#从控制台读数据的read函数*
39. read:
40. addi $sp, $sp, -4        *# 栈中开辟1个新地址保存数组元素个数*
41. sw $s0, 0($sp)
42. li $s0, 0                   *# 把s0寄存器置零，作为读数个数的计数器*
43. *#下面是利用跳转语句和条件控制的读数循环*
44. *#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*
45. read\_1:
46. 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*
47. beq $t0, $zero, exit\_1  *# t0=zero 则跳转到exit\_1*
48. sll $t0, $s0, 2           *# s0左移两位*
49. add $t1, $a0, $t0        *# a0加上t0生成新地址*
50. move $t2, $a0
51. li $v0, 5                 *# 读数*
52. syscall
53. sw $v0, 0($t1)              *# 保存读入的数据到主存*
54. move $a0, $t2
55. addi $s0, $s0, 1          *# s0++*
56. j read\_1
57. exit\_1:
58. lw $s0, 0($sp)              *# 把栈里面的东西（数组大小）写回寄存器*
59. addi $sp, $sp, 4          *# 弹栈，就是堆栈指针归位*
60. jr $ra
61. *#排序函数*
62. sort:
63. addi $sp, $sp, -20          *# 在栈中开辟5个新地址*
64. *#依次把要用变量的位置定出来并压栈*
65. sw $ra, 16($sp)            *# 返回地址*
66. sw $s3, 12($sp)            *# 数组大小*
67. sw $s2, 8($sp)           *# 数组基址*
68. sw $s1, 4($sp)            *# j*
69. sw $s0, 0($sp)            *# i*
70. move $s2, $a0
71. move $s3, $a1
72. move $s0, $zero
73. forOut:
74. slt $t0, $s0, $s3        *# 如果i<n，则t0=1*
75. beq $t0, $zero, exit1     *# 如果t0=0，跳转到exit1退出外层循环*
76. addi $s1, $s0, -1        *# j = i - 1*
77. forIn:
78. slti $t0, $s1, 0          *# 如果s1（j）<0，则t0=1*
79. bne $t0, $zero, exit2     *# 如果t0!=0，跳转到exit2*
80. sll $t1, $s1, 2           *# $t1 = j\*4*
81. add $t2, $s2, $t1   *# $t2存了arr[j]的地址*
82. lw $t3, 0($t2)              *# 取出arr[j]的数据到$t3*
83. lw $t4, 4($t2)              *# 取出arr[j+1]的数据到$t4*
84. slt $t0, $t3, $t4        *# 如果arr[j]<arr[j+1], 则t0=1*
85. beq $t0, $zero, exit2     *# 不满足上面条件，跳转到exit2退出内层循环*
86. move $a0, $s2             *# 把数组地址这个参数传给swap函数*
87. move $a1, $s1             *# 另一个参数j也传过去*
88. jal swap
89. addi $s1, $s1, -1         *# j--*
90. j forIn
91. exit2:
92. addi $s0, $s0, 1          *# i++*
93. j forOut                  *# 跳至外层循环*
94. exit1:
95. lw $s0, 0($sp)
96. lw $s1, 4($sp)
97. lw $s2, 8($sp)
98. lw $s3, 12($sp)
99. lw $ra, 16($sp)
100. addi $sp, $sp, 20
101. jr $ra
102. swap:
103. sll $t0, $a1, 2       *# j左移两位放到t0中*
104. add $t0, $a0, $t0    *# 基址值加上偏移量arr[j]的地址*
105. lw $t1, 0($t0)          *# 把arr[j]的值放入t1中*
106. lw $t2, 4($t0)          *# 把arr[j+1]的值放入t2中*
107. sw $t1, 4($t0)          *# arr[j]=arr[j+1]*
108. sw $t2, 0($t0)          *# arr[j+1]=arr[j]*
109. jr $ra                *# 返回调用前的地址处*
110. *#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*
111. prinf:
112. addi $sp, $sp, -4
113. sw $s0, 0($sp)         *#保存寄存器s0 s1*
114. li $s0, 0                   *#将s0置零*
115. prinf\_1:
116. *# 输出原数组元素*
117. sltu $t0, $s0, $a1
118. beq $t0, $zero, exit\_2
119. sll $t0, $s0, 2
120. add $t1, $a0, $t0
121. move $t2, $a0
122. lw $a0, 0($t1)
123. li $v0, 1
124. syscall
125. *# 输出‘，’*
126. li $a0, ','
127. li $v0, 11
128. syscall
129. *# s0 += 1*
130. move $a0, $t2
131. addi $s0, $s0, 1
132. j prinf\_1
133. exit\_2:
134. lw $s0, 0($sp)
135. addi $sp, $sp, 4
136. jr $ra
137. .data
138. str\_1:
139. .asciiz "Please enter the number of integers:\n"
140. str\_2:
141. .asciiz "Please enter the integers to be sorted:\n"
142. str\_3:
143. .asciiz "\nThe sorted numbers are:\n"
144. str\_5:
145. .asciiz "The entered numbers are:\n"