

编译原理作业 (7)

姓名: _____ 学号: _____

2024 年 06 月 08 日

请独立完成作业, 不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料, 请给出引用。
鼓励讨论, 但需独立书写解题过程。
允许并鼓励使用 ChatGPT 等工具, 但需明确说明使用方式。

1 作业 (必做部分)

题目 1 (RISC-V; 2023 年期末测试题目)

考虑如下计算第 n 项 Fibonacci 数的 C 语言程序 fib-rec.c

```
5  #include <stdio.h>
6
7  int fib(int n) {
8      // fib(0) = 0, fib(1) = 1
9      if (n <= 1) {
10         return n;
11     }
12
13     return fib(n - 1) + fib(n - 2);
14 }
15
16 int n = 20;
17
18 int main(void) {
19     int result = fib(n);
20
21     printf("fib(%d) = %d\n", n, result);
22
23     return 0;
24 }
```

下面两页给出了 fib-rec.c 对应的 RISC-V 代码片段, 请填充缺失的代码行。

说明:

- 第 17、23、49 行代码是相同的, 仅计一次分数。
- 第 42 行处, 假设 fib-rec.c 中的 n 占 4 字节, 也就是一个 word 大小。
- 所有填充处仅允许使用如下指令 (包括伪指令; 汇编伪指令不限): li、add、addi、sub、ble、bge、lw、sw、jal、jalr、ecall、j、la。

```

1  .text
2  .global main
3
4  ##### fib #####
5  fib:
6  # base case: n <= 1
7  li t0, 1
8  
9
10 addi sp, sp, -16    # allocate stack
11       # store a0 on stack
12       # store ra on stack
13
14 # n > 1: fib(n - 1) + fib(n - 2)
15 lw a0, 12(sp)      # a0: n
16 addi a0, a0, -1     # a0: n - 1
17       # call fib on (n - 1)
18 mv t1, a0           # t1: fib(n - 1)
19 
20
21 lw a0, 12(sp)      # a0: n
22 addi a0, a0, -2     # a0: n - 2
23       # call fib on (n)
24 mv t2, a0           # t2: fib(n - 2)
25 

```



```

26
27 lw t1, 4(sp)
28 lw t2, 0(sp)
29 add a0, t1, t2      # a0: fib(n - 1) + fib(n - 2)
30
31       # restore ra
32       # clear stack
33
34 j end
35
36 base_case:
37
38 end:
39  # ret
40 ##### main #####
41 .data
42 n: 
43
44 # n = 10: 0 1 1 2 3 5 8 13 21 34 55
45 .text
46 main:
47  # a0: address of n
48  # a0: value of n
49  # call fib

```

解答:

```

1  .text
2  .global main
3
4  ##### fib #####
5  fib:
6  # base case: n <= 1
7  li t0, 1
8  ble a0, t0, base_case
9
10 addi sp, sp, -16    # allocate stack
11 sw a0, 12(sp)      # store a0 on stack
12 sw ra, 8(sp)       # store ra on stack
13
14 # n > 1: fib(n - 1) + fib(n - 2)
15 lw a0, 12(sp)      # a0: n
16 addi a0, a0, -1     # a0: n - 1
17 jal ra, fib        # call fib on (n - 1)
18 mv t1, a0          # t1: fib(n - 1)
19 sw t1, 4(sp)
20
21 lw a0, 12(sp)      # a0: n
22 addi a0, a0, -2     # a0: n - 2
23 jal ra, fib        # call fib on (n)
24 mv t2, a0          # t2: fib(n - 2)
25 sw t2, 0(sp)
26
27 lw t1, 4(sp)
28 lw t2, 0(sp)
29 add a0, t1, t2      # a0: fib(n - 1) + fib(n - 2)
30
31 lw ra, 8(sp)        # restore ra
32 addi sp, sp, 16     # clear stack
33
34 j end
35
36 base_case:
37
38 end:
39     jalr zero, 0(ra) # ret
40 ##### main #####
41 .data
42 n: .word 5
43
44 # n = 10: 0 1 1 2 3 5 8 13 21 34 55
45 .text
46 main:
47 la a0, n            # a0: address of n
48 lw a0, 0(a0)        # a0: value of n
49 jal ra, fib         # call fib

```

题目 2

编写两段 RISC-V 函数代码，实现控制流在这两个函数之间“跳来跳去”的效果。

解答：

先留给大家思考、尝试。

题目 3

阅读理解《The RISC-V Reader》^① ^② 图 2.8 中插入排序 C 语言代码 (图 2.5) 对应的 RISC-V 代码。

^① [The RISC-V Reader @ compilers-resources](#)

^② [MIT 6.191 \(6.004\) RISC-V ISA Reference Card @ compilers-resources](#)