编译原理第七章第一次作业

李昊宸　2017K8009929044

7.2.4. 下面是两个 C 语言函数 f 和 g 的概述：

int f（int x）｛int i；…return i+1；…｝

int g（int y）｛int j；…f（j+1）；…｝

函数 g 调用函数 f。画出在 g 调用 f 而 f 即将返回时，运行时刻栈中从 g 的活动记录开始的顶端部分。你可以只考虑返回值、参数、控制链以及存放局部数据的空间。你不用考虑存放的机器状态，也不用考虑没有在代码中显示的局部之和临时值。但你应该指出：

1） 哪个函数在栈中为各个元素创建了所使用的空间？

2） 哪个函数写入了各个元素的值

3） 这些元素属于哪个活动记录

答：

<table>
<tr><th>record stack</th><th colspan="3">description</th></tr>
<tr><td>int y</td><td></td><td rowspan="3">activation record for g</td><td rowspan="2"></td></tr>
<tr><td>int g</td><td>written by g</td></tr>
<tr><td>g control and saved status</td><td></td><td rowspan="3">created by g</td></tr>
<tr><td>int j</td><td>written by g</td><td rowspan="4">activation record for f</td></tr>
<tr><td>int x</td><td>written by g</td></tr>
<tr><td>int f</td><td>written by f</td></tr>
<tr><td>f control and saved status</td><td>written by g</td></tr>
<tr><td>int i</td><td>written by f</td><td rowspan="2">created by f</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

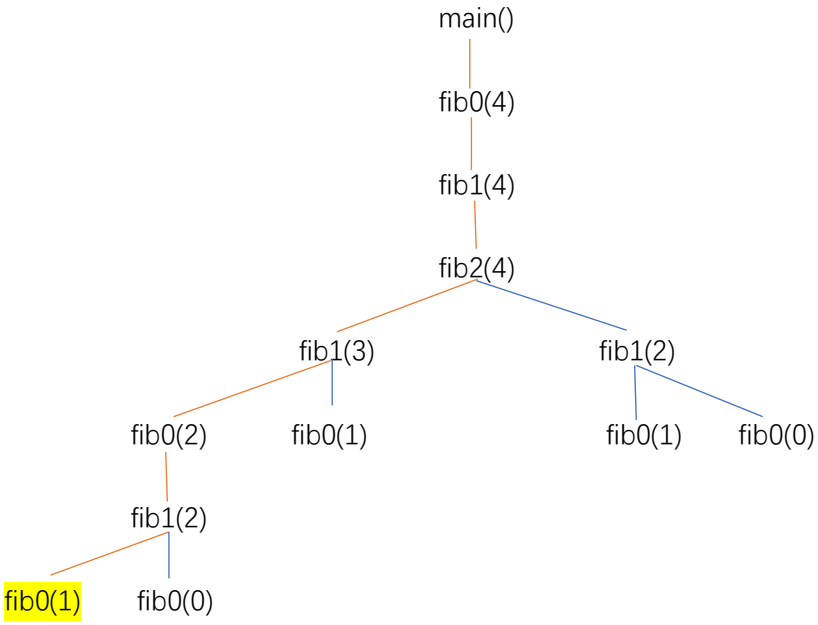
7.3.1. 图 7-15 中给出了一个按照非标准方式计算 Fibonacci 数的 ML 语言的函数 main。函数 fib0 将计算第 n 个 Fibonacci 数（n>=0）。嵌套在 fib0 中的是 fib1，它假设 n>=2 并计算第 n 个 Fibonacci 数。嵌套在 fib1 中的是 fib2，它假设 n>=4.请注意，fib1 和 fib2 都不需要检查基本情况。我们考虑从对 main 的调用开始，直到（对 fib0（1）的）第一次调用即将返回的时段，请描述当时的活动记录栈，并给出栈中的各个活动记录的访问链。

```
fun main () {
    let
        fun fib0(n) =
            let
                fun fib1(n) =
                    let
                        fun fib2(n) = fib1(n-1) + fib1(n-2)
                    in
                        if n >= 4 then fib2(n)
                        else fib0(n-1) + fib0(n-2)
                    end
            in
                if n >= 2 then fib1(n)
                else 1
            end
    in
        fib0(4)
    end;
```

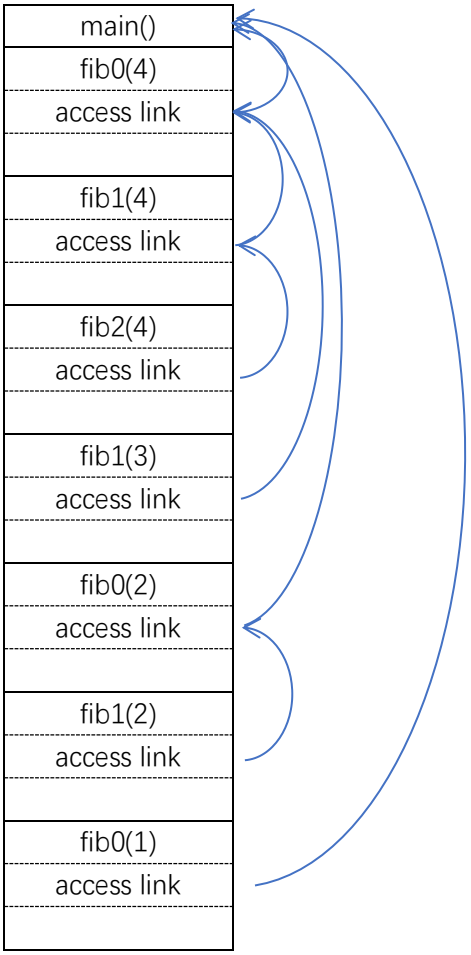
图 7-15　计算 Fibonacci 数的嵌套函数

答:

先列出活动树:



黄色标注为第一次调用 fib0（1）返回的时刻。橙色路线为当前栈中的活动记录。



7.3.2. 假设我们使用 display 表来实现下图中的函数。请给出 fib0（1）的第一次调用即将返回时的 display 表。同时指明那时在栈中各个活动记录中保存的 display 表条目。