ICS Lab4

PB24061302 | 赵国华 | 2025.12.16

1.实验目的

- 熟练掌握递归思想
- 熟练掌握汇编语言中递归的实现
- 优化代码风格，提高代码鲁棒性

2.实验过程：

2.1 实验思路：

- **递归结构**：二叉树，其中叶子节点是**i = 0**或者**j = 0**的坐标
- **计算思路**：求二叉树所有叶子节点的个数
- **具体实现**：用**栈**存储待定坐标，遇到叶子节点则计数加一
  - **数据结构设计***：栈中存储的数据设计成flag+N+M的二进制形式，其中flag为0表示下一步递归左子树，flag为1表示下一步递归右子树。利用掩码获取对应数据。
  - **递归结束条件**：栈为空

2.2 算法步骤：

1. **初始化数据**： R0, R1分别存储N,M
2. **计算Routes**：递归结构
   - **当前节点访问**：N=0或者M=0，则叶子节点数加1，同时处理栈顶元素，标记所在子树完成访问；否则构造flag+N+M的二进制数据，入栈。
   - **访问节点更新**：栈为空则所有节点访问完毕，结束，输出答案。否则访问栈顶元素，根据flag选择递归左子树/递归右子树。其中递归左右子树通过RO,R1的更新并再次执行上述步骤来实现。
3. **计算Steps**: R2 = R0 + R1
4. **计算最终结果并输出**：Routes*5 - steps;

2.3 代码难点：

1. **寄存器的合理使用**：因为寄存器个数有限，所以将steps的计算放在最后，而且尽量不选用全局变量。同时要注意寄存器的初始化。
2. **算法逻辑的合理实现**：在初版代码中，没有考虑到栈顶元素POP之后也算是递归结束从而出现逻辑错误。后来修复了该错误，也即遇到叶子节点或者栈元素POP都算是分支的结束，要更新上一级元素的状态。

2.3 待优化代码：

- **输出显示**：初版代码尝试使用数字+48的方式转化为ASCLL码从而输出在屏幕上，但是该方法只适用于0~9，而结果有可能出现更大的数据。目前还没有找到更好的方式来显示。
- **代码复用**：目前有部分功能相同的代码重复使用，在思考能否像高级语言的函数一样，抽出特定的代码块实现代码复用。
- **掩码逻辑**：为了存入同一个栈槽，构造了新的数据类型，使用了掩码逻辑。实则尝试多个栈槽存储应该会效率更高。

3.实验结果(使用线上评测工具)

本测试点的结果与预期相符。

输入

```
2 2
```

预期输出

```
x001a (26)
```

实际输出

```
x001a
```

#2 通过 / Accepted

本测试点的结果与预期相符。

输入

```
3 4
```

预期输出

```
x00a8 (168)
```

实际输出

```
x00a8
```

#3 通过 / Accepted

本测试点的结果与预期相符。

输入

```
3 4
```

预期输出

```
x00a8 (168)
```

实际输出

```
x00a8
```

#4 通过 / Accepted

本测试点的结果与预期相符。

输入

```
5 4
```

预期输出

```
x026d (621)
```

实际输出

```
x026d
```

#5 通过 / Accepted

本测试点的结果与预期相符。

输入

```
1 1 4
```

预期输出

```
1 x0014 (20)
```

实际输出

```
1 x0014
```

#6 通过 / Accepted

本测试点的结果与预期相符。

输入

```
1 5 3
```

预期输出

```
1 x0110 (272)
```

实际输出

```
1 x0110
```

#7 通过 / Accepted

本测试点的结果与预期相符。

输入

```
3 1
```

预期输出

```
x0010 (16)
```

实际输出

```
x0010
```

#8 通过 / Accepted

本测试点的结果与预期相符。

输入

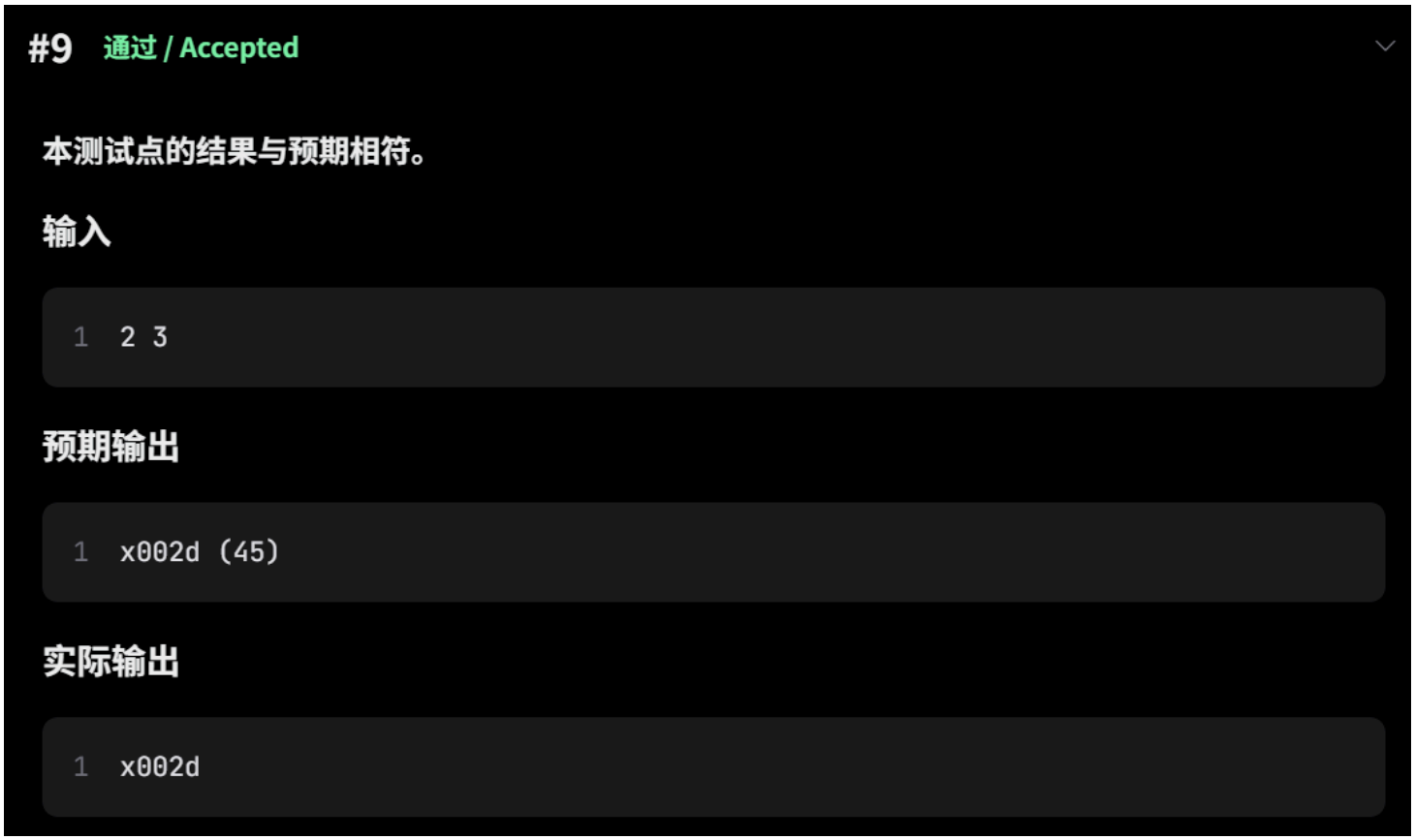
```
2 1
```

预期输出

```
x000c (12)
```

实际输出

```
x000c
```



4.问题讨论与思路优化：

1. Q: It is easy to see that this program can be rewritten as a simple iterative program, and theiterative method seems like more efficient compared to the basic recursive method. Why?
   A: 因为递归时会有很多重复计算的步骤，但是迭代可以避免这一问题从而提升效率。
2. Q: Based on the reasons you mentioned, how can you improve the efficiency of this program (nocode required)?
   A: 设计一个哈希表类似的结构存储已经计算过的值，下次遇到相同的值不必重复计算，只需直接查找，从而提升效率。