

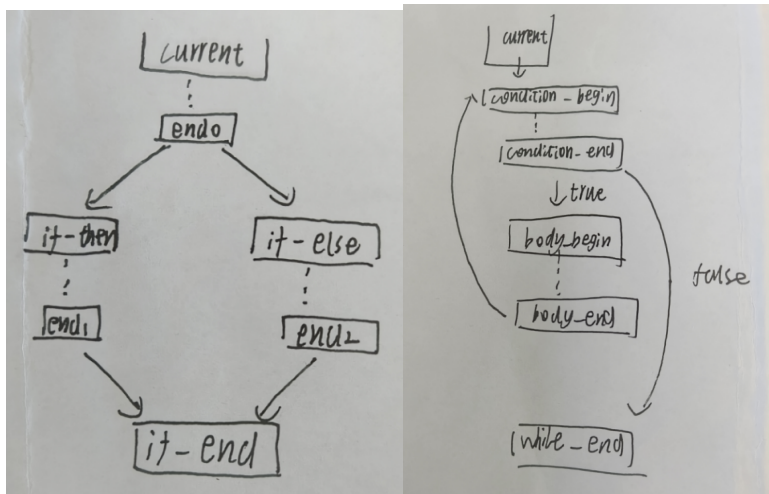
lab3: 中间代码生成

21307140 李明俊

一. 实验过程简述

这次实验主要完成了一个中间代码生成器，程序的输入是实验二的答案，也就是JSON格式的抽象语法树，程序的输出是LLVM IR。在实验结束后，我总结出了本次实验中完成的几部分工作以及这个过程中自己遇到的错误和问题：

- 局部变量和全局变量的声明和初始化：**这一部分是实验的第一个小门槛，其中一个难点是全局变量和局部变量的区分，这是通过判断当前是否在写一个block来决定的，如果没有在写block则说明是全局变量。需要注意的是，全局变量的声明初始化过程其实就是在执行一个函数，只不过这个函数中只做了声明初始化全局变量的工作，因此这个过程就要涉及新函数和基本块的创建，以及最后记得要将mCurrIrb置为空表示当前没写基本块，防止对后续全局变量的影响。
- 数组的声明，初始化和访问：**我认为这一部分是整个实验中最难的部分，我在这个部分上卡了两整天。首先由于多维数组要求各部分通过递归来实现，而且这个过程中还需要思考怎么获得需要的相关变量（比如数组类型，数组指针，索引下标等等），因此实现的过程比较难想，我感觉自己写的数组初始化部分的代码比较丑陋，就是把初始化元素一个个塞到临时数组里，然后再把这个临时数组的值一个个取出来赋值给数组变量，但是我认为真正的实现应该会比这个优美很多。
- 函数的定义和调用：**这一部分难度不是很高，只需要处理好函数类型，参数的处理，函数调用部分即可。我自己是在参数处理这一部分卡了一天，因为发现函数参数不管怎么样都传不到函数里面。于是我在这一部分给每一个参数又单独申请一个变量，再把参数传给这个变量使用，因此实现的也不够优美，有一定的改进空间。
- 与，或的短路实现：**与和或的正确实现是if，while语句正确实现的基石。短路实现的方式在文档里已经写得非常详细了，但是有很多潜在的bug文档里并没有提及，是需要自己发现的：首先最重要的就是要考虑到基本块的递归，例如当你调用self()处理右边子式结束的时候，这里irb中的当前基本块就可能不是执行self前的基本块了，因此这里可能需要涉及到一些额外的基本块的记录。另一点就是额外考虑i32类型到i1类型的转换。
- if, while语句：**这一部分的难点主要就是要和基本块打交道，这个过程中一定要清楚你当前在写哪个基本块，你想要写哪个基本块，基本块间的跳转是怎么样子的，还要考虑刚刚提到的递归出现的基本块变换问题。例如下面是我在实现if，while时在纸上画的简单的控制流程图，里面的虚线就是可能潜在的递归造成的基本块变换。



6. **其它类型，语句，表达式的添加**：包括各种二元表达式，一元表达式，隐式类型转换，空表达式等等的处理，这一部分比较零散且实现难度不大，根据样例查看自己缺哪一部分即可。

二. 实验运行结果

编译原理实验三（提供七天补测）

提交要求



提交项目里打包好的压缩包，点击提交即可。

须知

- 0. 评测机负责人为郑腾扬，联系方式：zhengty26@mail2.sysu.edu.cn 或QQ群内@我
- 1. 本次不要求提交到**排行榜**，同学们可以自行决定是否要提交到排行榜
- 2. 有能力本地测评的同学请在本地测试后再提交到评测机，评测机的资源请留给有需要的同学
- 3. 我们将会在每次实验结束后进行代码审查，一经审定为抄袭者，本次实验分数无效

不做才以骄人，不以宠而作威。——诸葛亮

提交记录

+ 上传提交			
提交 ID	提交时间	得分	操作
1012	2024-05-30 17:13:42	7300 / 7300	 

三. 感想与建议

虽然老师和助教都说lab3会比lab2更简单，但是做完之后还是感觉lab3会难一些，不过做下来的体验lab3确实会比lab2好一些。主要是因为lab2没有很多有效的输出文件来让我们调试，所以很多时候感觉是在摸石头过河，但是lab3有 `output.ll` `output.compile` 会给实验的调试减轻很大的负担。以及lab3的文档写的真的很棒！几乎所有的llvm接口都条理清晰地列在了里面，并且提供的上手教程也非常的细致，负责lab3的几位助教也非常的热心负责，所以这些其它因素有效的降低了更有技术难度的lab3的门槛。

有关lab3的建议的话就是希望可以加大数组测例的比重，以及添加一些多维数组（二维以上）的样例。因为整个lab3数组花费的时间是最多，也是我们觉得最难的一部分，但是整个实验下来发现包含数组的样例可能只有十分左右，因此我觉得可以适当减少前期加减乘除二元表达式分数的占比，加大数组的分数占比，会让整个实验的分数难度曲线更加平衡一些。