编译原理实验 3

孙伟杰 191250127

2022 年 5 月 11 日

1 项目进度

完成了所有文档中的必做和选做要求

2 使用方法

使用方式即在 Code/路径下在 terminal 中输入 make parser，然后./parser + 需要运行的文件名 + 输出中间代码的文件名即可

3 实现方法

3.1 对语句的组织

中间代码的组织形式采用双向链表形式来进行组织

3.2 对语句的翻译

对于可能有结果的产生式，我都选择传进一个 operand 的指针，operand 指针中包含有 type 变量并且可以指示是地址还是形参，如此调用函数的内部结果（包括内容和类型）可以通过此 operand 指针传给调用者。当可以进行常数折叠时可以直接不生成 code 而直接将 place 指针设成结果来简化代码的实现，如此就不一定要对 place 变量进行赋值而可以减少部分代码

3.3 代码生成的位置

因为给出了限定没有全局变量，所以选择在进行语义分析的过程中，在每个 function 定义的时候，在对每个 compst 进行语义分析后，在对符号表中 compst 中的元素删除前，进行每个 compst 的中间代码的生成，这样不需要重新在符号表中进行统计。
但如此操作引起了一个问题，因为 function 中的 stmt 中可能会出现 compst，而 compst 里可能会出现变量的定义，如果在 function 的 compst 后再进行中间代码生成，stmt 中的 compst 中定义的变量已经从符号表中删除。
而如果在 compst 中先进行语法分析的话，可能又会出现非基本类型变量不知道 dec 出来还是 param 出来的情况
为了解决这个问题，我选择在进行 stmt 中的 compst 的生成时，再进行一次当前 compst 的语义分析再进行中间间代码分析

4 参考资料

感谢 wzj 同学提供的测试代码