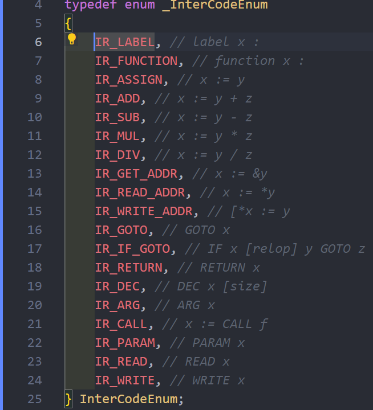
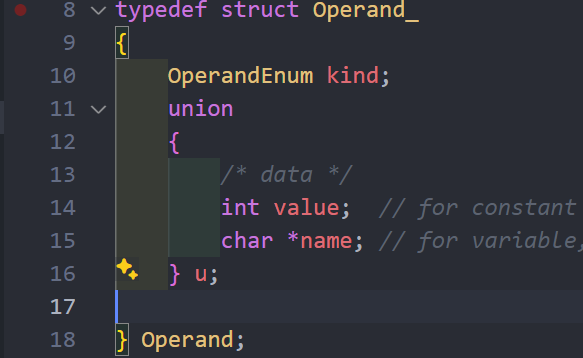
编译原理实验2报告

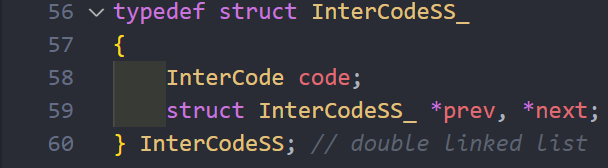
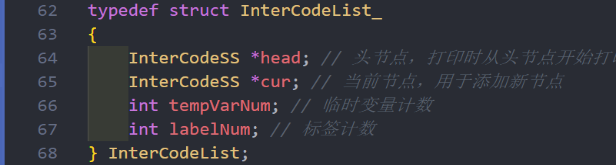
姓名：2朱宸慷 学号：2021110908

1. 程序实现的功能
2. 所使用的数据结构

电脑萤幕画面

描述已自动生成首先参照指导书，定义可能出现的操作数类型。在我们的中间代码生成中，操作数可以分为两类：与运算相关的常数、变量以及指针；与控制流相关的标签、函数以及操作符。同样参照指导书，可以定义翻译“一条”中间代码的模板形式，总共可以分为表达式、调用与函数、控制流与跳转三类共计19种种类，各自对应一个中间代码翻译模板。

在确定了一条中间代码的翻译模板之后，还需要确定填充的操作数，将操作数保存为操作数种类和操作数的字面值的结构体Operand，同时根据中间代码对于操作数个数需求的不同，构建中间代码类型和数个操作数的结构体InterCode。

在生成中间代码时，将InterCode连接成一个链表，保存头尾指针，同时保存过程中生成的临时变量和标签的数目，用于生成下一个变量或标签。

1. 中间代码生成的思路

基于实验一构建的语法树，从根节点开始前序遍历整棵语法树。由于实验指导书要求不能使用全局变量，因此我们直接从定义ExtDefList开始进入翻译过程。

在实验三中，总共需要完成的翻译模式有基本表达式、语句、函数调用、数组与结构体。其中结构体不做要求，笔者在实验中实现了没有明确给出的数组翻译模式。

**翻译表达式**分为翻译数值计算和翻译布尔表达式，本质上都是使用一个临时变量接收表达式计算的结果，只是数值计算的结果可以直接返回，而布尔表达式的结果需要经过条件跳转来决定最终返回将接收值的变量赋值为1或0。

**翻译语句**过程中，需要将以下五类语句——表达式语句、复合语句、返回语句、跳转语句和循环语句翻译为对应的中间代码。每种语句出现的位置比较固定，可以检查语法树节点的类型并决定使用何种翻译方法。参考实验指导书可以找到一部分语句对应的翻译模板。

**翻译函数调用**需要完成调用前参数压栈以及函数体内部的参数弹栈过程，这个过程可以借助符号表中存放的函数参数直接完成。

**翻译数组**需要获取数组首地址以及完成偏移地址的计算。在该计算过程中涉及到偏移量的获取以及单个元素size的预定义。

综上所述，实验三主要完成的就是：针对每种语法树节点的翻译函数，将该节点的子节点全部翻译成一条InterCode链表之后，将该链表插入到整体链表合适的位置，并在整体链表中维护临时变量与标签的计数器，用于在中间代码中表示对应的操作数。

1. 翻译过程

递归调用翻译函数，由于本实验不涉及全局变量的使用，所以第一个需要具体实现的产生式是ExtDef -> Specifier FunDec CompSt。其中Specifier节点由于不需要区分类型是浮点数还是整型数，所以没有必要分析。

在解析FunDec的过程中，需要完成参数弹栈中间代码的生成，该函数的参数可以在符号表中查询，按照顺序弹栈即可。在函数调用的翻译中，会有对应的参数压栈过程，此时同样可以查询符号表获得参数，但是需要注意压栈顺序与弹栈顺序是相反的，此处由于数据结构限制，笔者使用双指针链表来获得反向连接的参数链表。

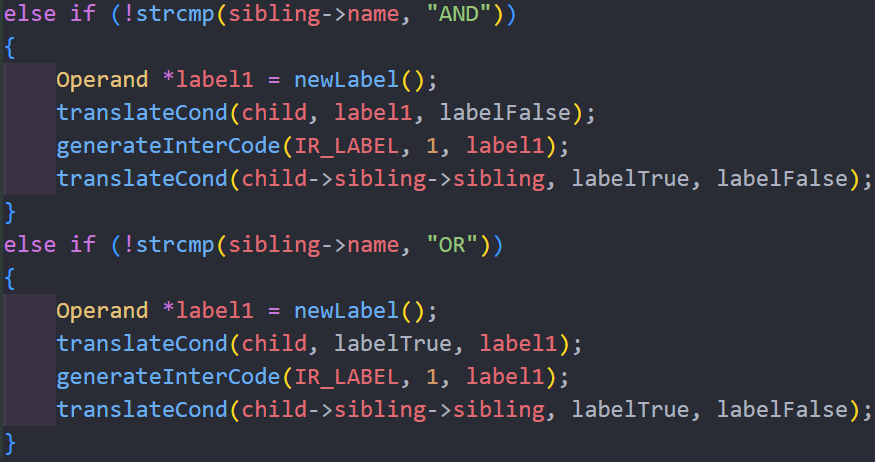
文本

描述已自动生成

在解析CompStmt时，涉及到产生式CompSt -> LC DefList StmtList RC，其中DefList可能为空，需要特别判断。非终结符StmtList是右递归的，最终需要解析节点Stmt，语句可能是表达式，也可能是控制流。其中最重要的是对if-else控制流的解析

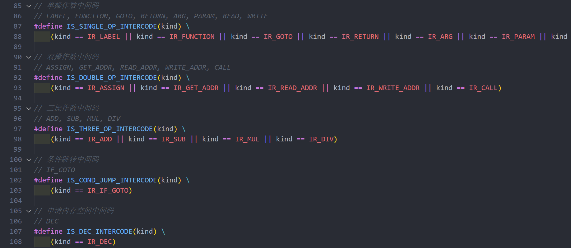
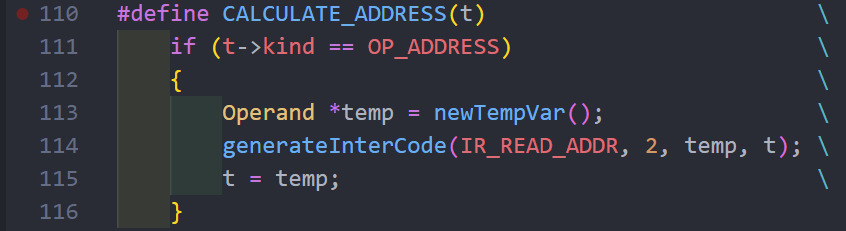
在if-else中，主要是基于检查if条件中表达式的值来完成cond-jump，若表达式为真，则跳往对应执行的语句块，否则就跳出该控制块，即：t = Exp()；if t != 0 goto label1；goto label 2；labe1:{ 语句块 }; label2: {语句块}，其中label1指向翻译出的条件为真时应执行的代码块，label2可能是else对应的代码，也可能是原代码中逻辑上的下一行代码。

在翻译上述语句中的表达式时，涉及到对布尔表达式的翻译。由于布尔表达式不能直接返回计算的结果，因此需要额外进行一次cond-jump，根据逻辑操作符的类型，来决定本表达式结果对应的布尔值。此时还会涉及到逻辑运算符的短路操作，即当a || b时，若a为真，则可以直接返回表达式为真



1. 个性化内容

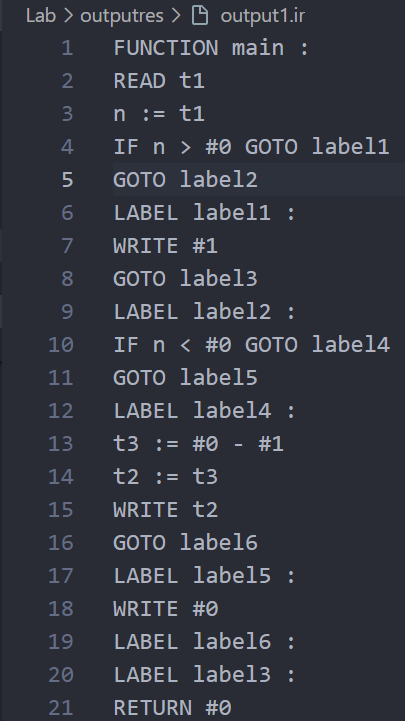
在处理中间表达式翻译时，将中间代码按照参数类型进行了包装，使用了宏定义来处理中间代码类型的判断

除此之外，部分重复操作也使用了宏进行包装，例如获取变量地址的操作

1. 程序的编译方法
2. 自动化执行makefile

通过使用makefile自动执行。在code文件夹下先make clean再运行make即可编译本实验的可执行文件。在code文件夹下使用命令“./test.sh <测试用例编号>”运行测试脚本test.sh即可自动编译并输出结果（注：要求存在对应路径）。

1. 文本

   描述已自动生成结果展示

如图所示是实验测试用例1、2的中间代码，其中左图实现了检查输入t1，根据t1的值进行cond-jump并打印不同的值；右图实现了阶乘函数fact，并在main函数中完成了fact函数的调用。