

实验三 中间代码生成

实验目的

1. 巩固对中间代码生成的基本功能和原理的认识。
2. 能够基于语法指导翻译的知识进行中间代码生成。
3. 掌握类高级语言中基本语句所对应的语义动作。

实验环境

Ubuntu 20.10 64 位
gcc version 9.4.0
bison (GNU Bison) 3.5.1
flex 2.6.4

中间代码生成设计

首先，该数据结构使用多重链表中的 InterCode 连接方法。并酌情展开，在这里，我们只需要在自己的讲义和理解中根据表 11 创建一个替换顺序。并对实验的三名学生进行调查生成的 InterCode 绑定列表，按 interCode 类型和特定操作数。选择要导出到的特定说明。out.s 你可以选择订单来完成。

堆栈管理，这时候我们考虑 ARG、PARAM、RETURN、CALL 等命令。其余的用于选择命令。我们将这些消息分为两类。一是调用者命令。另一个是调用者的命令。调用者需要保存注册。对应的变量分别存放在寄存器或堆栈中，ARG 和接收方应保留寄存器或堆栈，反之亦然。返回时读取变量（PARAM）（RETURN）调用者应依次返回之前保存的寄存器（restore）值得注意的是，调用者和调用者对变量的处理必须一一对应。

主要任务是设计数据结构。相关接口的具体实现而且进入 ExtDefList 后第二次测试生成中间代码也很重要，这里也有很多关于我们的假设，我们可以在语义分析的时候选择不生成 IR。但在语义上分析后不要删除符号表并再次跳过语法结构来生成 IR，因此再次探索语法结构以找到 ExtDefList 节点并开始处理。

实现方法及结果

1. Makefile 用于编译用于测试输入文件的词法和语法分析的解析器程序

```
(kali@kali)-[~/Desktop/done_compiler3a/code]
$ make
flex -o ./lex.yy.c ./lexical.l
bison -o ./syntax.tab.c -d -v ./syntax.y
gcc -c ./syntax.tab.c -o ./syntax.tab.o
gcc -c -o semantic.o semantic.c
gcc -c -o inter.o inter.c
gcc -c -o main.o main.c
gcc -o parser ./semantic.o ./inter.o ./main.o ./syntax.tab.o -lfl
```

2. 只需调用以下命令即可调用所有命令制作
3. 可以使用以下任何命令使用测试文件测试解析器
 - a) 使用解析器 `./parser ../Test/test1.c ./output.s`

```
(kali㉿kali)-[~/Desktop/done complier3a/code]
$ ./parser ../tests/test1.txt output.s

(kali㉿kali)-[~/Desktop/done complier3a/code]
$ cat output.s
FUNCTION main :
READ t2
vn := t2
IF vn > #0 GOTO label1
GOTO label2
LABEL label1 :
WRITE #1
GOTO label3
LABEL label2 :
IF vn < #0 GOTO label4
GOTO label5
LABEL label4 :
WRITE #-1
GOTO label6
LABEL label5 :
WRITE #0
LABEL label6 :
LABEL label3 :
RETURN #0
```