编译原理实验 Lab3-1190201303-王艺丹

实验完成情况综述

完成所有必做内容以及选做3.1内容：测试用例1-3均通过

完成C--语言文法的中间代码生成
选做了 3.1 结构体相关的中间代码生成
使用了小程序进行测试

实验内容及编译方法

文件组成结构

```
└── Lab3-1190201303-王艺丹
    ├── code
    │   ├── lexical.l
    │   ├── main.c
    │   ├── makefile
    │   ├── syntax.y
    │   ├── enum.h
    │   ├── TreeNode.c
    │   ├── TreeNode.h
    │   ├── semantic.h
    │   ├── semantic.c
    │   ├── inter.h
    │   └── inter.c
    ├── test
    │   ├── test1.txt
    │   ├── ......
    │   └── test4.txt
    └── Lab3-Report.pdf
    └── README.md
```

编译方法

在code目录下使用make命令编译所有文件
在code目录下使用make test命令进行中间代码生成，对测试样例进行测试

实验完成功能及亮点

基本数据结构分析

在*inter.h*文件中:

- Operand 和interCode 的结构体定义参考实验指导书，不做赘述，另外按照实验要求，采用链表式的 IR
- 可以看到，c--中结构体作为参数传递时是通过传址完成，所以可将其看作类似数组的数据结构，循环遍历获得偏移量offset查表得到其中各变量；于是加入一个地址指针，实现低维数组与结构体相关的中间代码生成

相关代码如下：

```c
typedef struct _interCodeList {
    pInterCodes head;
    pInterCodes cur;
    // 实现低维数组与结构体，c--结构体作为参数传递时是通过传址完成，看作数组与偏移量查表得到
    char* lastArrayName;
    int tempVarNum;
    int labelNum;
} InterCodeList;
```

低维数组访问

在*inter.c*文件中：

- ✧ 根据假设，Exp1只会展开为 Exp DOT ID 或 ID，所以让前一种情况将ID作为name，回填至place，返回到至base处，同时在语义分析时将结构体变量也添加至表中（因为假设无重名）这使得两种情况都可以查表得到
- ✧ 如果是第二种情况ID[Exp],则需要对ID取址，如果前面是结构体内访问，则会返回一个地址类型，不需要再取址

  相关代码较长，不足赘述

结构体访问

在*inter.c*文件中：

- ✧ 两种情况，如果将Exp直接为一个变量，则需要先取址；若Exp为数组，或多层结构体访问，或结构体作为形参，则直接将target填成地址，直接调用
- ✧ 为了访问结构体内部定义的数组，需要把id名通过place回传给上层

  相关代码较长，不足赘述

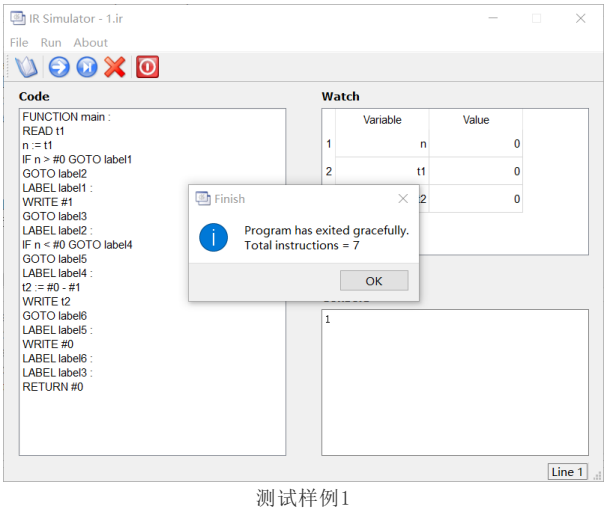
代码优化及分析过程

在*inter.c*文件中：

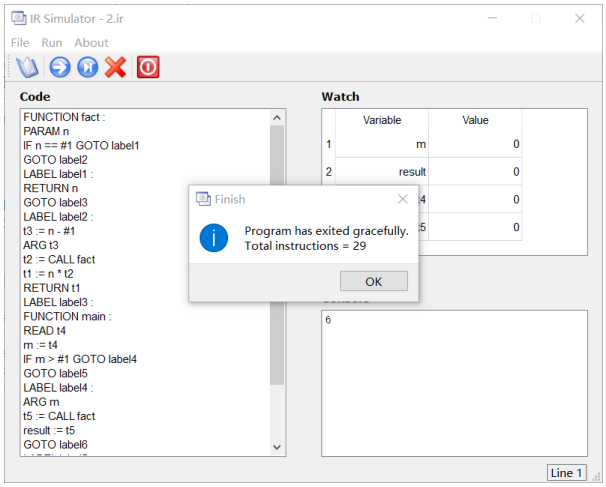
- ✧ 并未做过多优化，仅针对直接使用的符号与立即数，不创建新临时变量。
- ✧ 先进行语义分析，若无语法错误，再遍历语法树生成 IR

实验结果

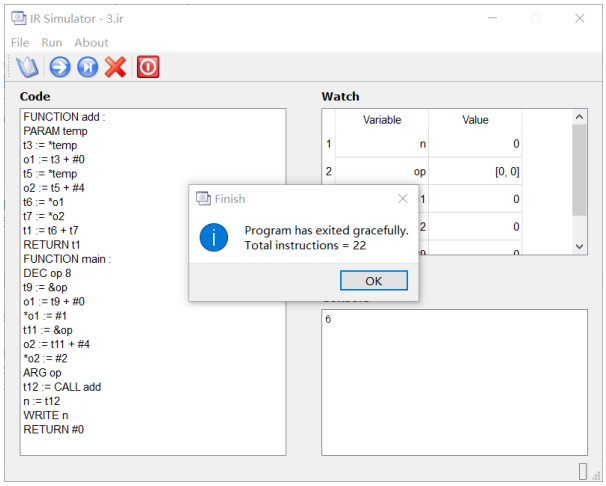
```
sxj19910136@ubuntu-hitics:~/Desktop/wyd-lab3/code$ make test
./parser ../test/test1.txt 1.ir
./parser ../test/test2.txt 2.ir
./parser ../test/test3.txt 3.ir
./parser ../test/test4.txt 4.ir
Cannot translate: Code containsvariables of multi-dimensional array type or para
meters of array type.
```



测试样例1


IR Simulator - 2.ir
File Run About
Code
FUNCTION fact :
PARAM n
IF n == #1 GOTO label1
GOTO label2
LABEL label1 :
RETURN n
GOTO label3
LABEL label2 :
t3 := n - #1
ARG t3
t2 := CALL fact
t1 := n * t2
RETURN t1
LABEL label3 :
FUNCTION main :
READ t4
m := t4
IF m > #1 GOTO label4
GOTO label5
LABEL label4 :
ARG m
t5 := CALL fact
result := t5
GOTO label6
Watch
Variable
Value
1
m
0
2
result
0
Finish
Program has exited gracefully.
Total instructions = 29
OK
6
Line 1

测试样例2


IR Simulator - 3.ir
File Run About
Code
FUNCTION add :
PARAM temp
t3 := *temp
o1 := t3 + #0
t5 := *temp
o2 := t5 + #4
t6 := *o1
t7 := *o2
t1 := t6 + t7
RETURN t1
FUNCTION main :
DEC op 8
t9 := &op
o1 := t9 + #0
*o1 := #1
t11 := &op
o2 := t11 + #4
*o2 := #2
ARG op
t12 := CALL add
n := t12
WRITE n
RETURN #0
Watch
Variable
Value
1
n
0
2
op
[0, 0]
Finish
Program has exited gracefully.
Total instructions = 22
OK
6

测试样例3