《编译器构造实验》lab3报告

20331041　徐锶达　xusd3@mail2.sysu.edu.cn

1 实现功能

1.1 完成内容

- **所有必做内容**：在假设1-7下，将C--源代码翻译为符合规范的中间代码
- **所有选做内容**：满足要求3.1和3.2

1.2 中间代码

1 中间代码的表示 - 线形

采用**双向链表**存储中间代码，则lab3就是通过抽象语法树和符号表的信息，逐步构建中间代码的过程。

- 中间代码的结构：操作符+类型+辅助信息（如二元运算符）
- 操作符的结构：
  - 类型+ifaddress（判断是否需要*或者&）
  - 辅助信息（变量名、函数名、序号、深度）
    - 深度是数组的中间代码生成部分的关键信息

2 中间代码生成与插入过程

- 通过new_op构造新的操作符，通过new_intercode构造新的中间代码，再通过add_inter往中间代码双向链表插入节点
- 在new_op中：
  - 对于临时变量OP_TEMPVAR：如果ifaddress是OP_ADDRESS那么就在前面加上*，表示读取内存单元的内容操作
  - 对于变量OP_VARIABLE：如果ifaddress是OP_ADDRESS那么就在前面加上&，表示取地址操作
  - 这样处理可以减少用于表示地址的重复指令

1.3 翻译模式与设计

1 Stmt的cond部分优化

- GOTO指令部分：通过`!cond`，也即对条件取反，减少一次GOTO操作，比如：

原版：

```
label4:
if cond goto label5(true label)
goto label6(false label)
label5:
...
goto label4
label6:
...
```

改进：

```
label4:
if !cond goto label6(false label)
....(true content)
goto label4
label6:
...
```

- IFGOTO指令部分：通过一些判断简化成GOTO操作，比如条件中的INT部分

2 数组处理

- 通过gettypesize确定一个类型type的大小，该函数是递归的
  - 对于数组是相乘的过程，对于结构体是相加的过程
- 数组处理有两个部分：VarDec_g部分和Exp_g部分
  - VarDec_g部分：通过ID确定数组的type，通过gettypesize确定要申请的空间的大小
  - Exp_g部分：递归方案，计算偏移量

3 结构体处理

- 通过对应的ID查询到对应ID相对结构体首部的偏移
- 使用一个TEMPVAR记录再加上前面的exp的偏移

1.4 细节

- 在`lexical.l`中，对于二元运算符RELOP，要把运算符名存入节点信息中
- 在`symbol_table.c`中，enter_scope和exit_scope都直接返回作用域链表头（即不再删除作用域变量）
- 在`semantic.c`中，符号表中预先添加read和write这两个预定义的函数
- 结构体内的field不仅会插入结构体表，还会插入符号表，方便查询
- 结构体Symbol_node添加了一些新的域：
  - belongtostructname：方便查阅结构体的域所属于的结构体
  - offset：仅用于结构体内元素，用于指示元素相对于结构体首部的偏移量，插入时更新

2 Makefile编译

- 进入`Code`文件夹：运行：`make`
- 清理编译生成文件，运行：`make clean`

3 测试

1️⃣ **单个文件手动测试**，运行：`./parser test.cmm out1.ir`

若`test.cmm`和`out1.ir`与`parser`不在同一文件夹下，需自行添加路径

比如：`./parser ../Test/test.cmm ../result/out1.ir`

2️⃣ **多个文件脚本测试**，运行：`./test.sh`

将多个测试文件（**只分析后缀为.cmm的文件**）统一放在`Test`文件夹下，运行**测试脚本**`test.sh`。

结果输出文件放在`result`文件夹下，文件名同测试文件名，后缀为.ir。

注意默认情况是`parser`在`Code`文件夹下，`Code`文件夹与`Test`和`result`文件夹同级。不同则自行修改脚本中的`path1`和`path2`。

3️⃣ **虚拟机小程序测试IR**，运行：`python3.8 irsim.pyc`