编译原理 Lab-3

姓名：郑奘巍　　学号：171860658　　日期：2020 年 5 月 7 日

1　基本功能

本次实验完成了基本要求和要求 3.1，要求 3.2。程序将遍历语法树生成中间代码（IR），具体实现如下

- **intercode.c/h**：定义了中间代码（IR）和操作符（Operand）的结构，以链表保存中间代码，并定义了生成操作符和中间代码的常用函数（如 *new_label,new_temp*）以及操作符和中间代码的输出。
- **translate.c/h**：遍历 AST 生成中间代码。支持数组和结构体的任意嵌套，以及高维数组作为参数传递。代码框架与语义分析（**semantic.c/h**）的框架一致，具有可读性。
- **optimize.c/h**：生成中间代码后重新扫描中间代码进行优化，详见中间代码优化。
- **cfg.c/h**：生成 CFG 流图并进行常量优化和无用代码删除。详见中间代码优化

2　环境与集成测试

本次实验的实验环境如下：

- 开发环境：macOS 10.15.4, flex 2.5.35 Apple(flex-32), bison 2.3, clang 11.0.3
- 集成环境：Ubuntu 18.04, flex 2.6.4, bison 3.0.4, gcc 7.5.0

本次实验通过自己搭建的集成测试 **CI4C–_Compiler**，确认了 Lab-1 到 Lab-3 的正确性。其中 Lab-3 的数据由同学共同共献，详见 **compilers-tests**。

3　中间代码优化

> 讲义上对于 *reference* 和 *dereference* 只提到了 *x := &y, x := *y, *x = y*，而虚拟机支持 *x := *t1 + *t2* 等操作。个人认为不存在的特性是虚拟机的 *UB*，不应该用虚拟机的 *UB* 优化。故以下优化结果没有使用该优化。

中间代码优化大体分为三个部分如下，优化等级可以在 **common.h** 中调整。

- OP 1: 在 IR 生成阶段直接优化
- OP 2: 线性扫描生成的 IR 进行优化
- OP 3: 构建 CFG 进行静态分析

以 **compilers-tests** commit 966a071 中测试文件作测试集，各优化性能如下：

OP	翻译出的指令条数	执行的指令条数
0	49013	134355618
1	33604	67415622
2	31287	61844725
3	30819	61800913

表 1　优化性能评估

各优化具体如下，可以通过定义或取消定义对应宏名开闭优化选项。

- OP 1
  - OP_INT: 当临时变量被赋值为整数时，取消该临时变量，直接使用整数
  - OP_ID: 当临时变量被赋值为变量时，取消该临时变量，直接使用变量
  - OP_TEMP_REPLACE: 当临时变量被赋值为另一个临时变量时，只保留一个临时变量
  - OP_ARITH_CONST: 当四则运算操作数均为整数时，直接计算
  - OP_ARR_CONST: 当计算数组/结构体偏移时有常量运算的，直接计算
  - OP_ASSIGN_TO_VAR: 当赋值语句左侧为变量时，直接赋值给变量
- OP 2
  - OP_LINEAR_USELESS_LABEL: 消除无用 label
  - OP_LINEAR_REPLICATE_LABEL: 当 label 连续出现时，仅保留一个
  - OP_LINEAR_DIRECT_GOTO: 当条件运算为常量结果时，直接跳转或删去
  - OP_LINEAR_REDUCE_RELOP: 减少 GOTO, RELOP 和 LABEL 语句（如将 if 语句翻译出的头三句缩减为一句）
- OP 3
  - OP_CFG_CONST: 常量分析，替换程序中所有常量
  - OP_CFG_DEAD_CODE: 遍历流图，去除无法访问的基本块

其中常量分析伪代码如下，**go-through** 函数遍历区块内的语句，判断变量是常量或 UNDEF(undefine) 或 NAC(not a number)：

Algorithm 1 常量分析

```
function CONSTANT-ANALYSIS(B)
    OUT[entry] = { }
    for each basic block B-entry do
        OUT[B] = { }
    while changes to any OUT occur do
        for each basic block B-entry do
            IN[B] = ⋂_{P a predecessor of B} OUT[P]
            OUT[B] = go-through(IN[B])
```