编译原理实验5实验报告：

功能及实现方式

核心逻辑：数据流分析

- 我们使用了平台提供的框架代码，在完成全局常量传播、活跃表达式分析、复制传播、活跃变量分析、死代码消除等数据流分析的填空中，同时借此完成了必做的局部优化
- 核心逻辑伪代码如下：

```
// 遍历所有函数，对函数使用数据流分析进行全局优化
for_vec(IR_function_ptr, i, ir_program_global->functions) {
    IR_function *func = *i;
    {
        // 常量传播，使用c语言面向对象方式完成
        constantPropagation = NEW(ConstantPropagation);
        worklist_solver((DataflowAnalysis*)constantPropagation, func);
        ConstantPropagation_constant_folding(constantPropagation, func);
        DELETE(constantPropagation);
        // 可用表达式分析
        availableExpressionsAnalysis = NEW(AvailableExpressionsAnalysis);

AvailableExpressionsAnalysis_merge_common_expr(availableExpressionsAnalysis,
func);
        worklist_solver((DataflowAnalysis*)availableExpressionsAnalysis, func);

AvailableExpressionsAnalysis_remove_available_expr_def(availableExpressionsAnalysi
s, func);
        DELETE(availableExpressionsAnalysis);
        // 复制传播
        copyPropagation = NEW(CopyPropagation);
        worklist_solver((DataflowAnalysis*)copyPropagation, func);
        CopyPropagation_replace_available_use_copy(copyPropagation, func);
        DELETE(copyPropagation);
    }

    // 常量传播，在消除部分表达式后可能会有新的常量传播机会
    constantPropagation = NEW(ConstantPropagation);
    worklist_solver((DataflowAnalysis*)constantPropagation, func);
    ConstantPropagation_constant_folding(constantPropagation, func);
    DELETE(constantPropagation);
    // 基于活跃变量分析的死代码消除，一直到收敛
    while(true) {
        liveVariableAnalysis = NEW(LiveVariableAnalysis);
        worklist_solver((DataflowAnalysis*)liveVariableAnalysis, func);
        bool updated = LiveVariableAnalysis_remove_dead_def(liveVariableAnalysis,
func);
        DELETE(liveVariableAnalysis);
        if(!updated) break;
```

```
    }
}
```

基于数据流分析的常量传播

- 框架中主要填写的内容如下

```
// 计算不同数据流汇入后的meet值
CPValue meetValue(CPValue v1, CPValue v2)
// 计算二元运算结果的CPValue值
CPValue calculateValue(IR_OP_TYPE IR_op_type, CPValue v1, CPValue v2)
// 接下来为常量传播的数据流分析实现
bool isForward(ConstantPropagation *t) {
    return true; // 前向数据流分析
}
/*
需要初始化Out[Exit] = NAC
并初始化所有blk有In[blk] = NAC; Out[blk] = NAC;
同时有transfer函数
In[blk] = ∪(all Out[pred]) // pred为blk的前驱
Out[blk] = gen[blk] ∪ (In[blk] - kill[blk])
*/
```

- 这样我们就完成了常量传播的实现，实现了全局和局部的常量折叠

基于数据流分析的可用表达式分析

- 可用表达式分析填写思路如下
    - 正向分析
    - May Analysis：只要表达式可能可用都不能够删除
    - 边界情况：Out[Entry] = empty
    - 基本块初始化：In[blk] = Uni(全集)，Out[blk] = Uni(全集)
    - 不同数据流汇入meet值：In[blk] = ∩(all Out[pred])
- 根据思路填空，就完成了可用表达式分析的实现，实现了全局和局部的可用表达式分析

基于数据流分析的复制传播

- 复制传播填写思路如下
    - 正向分析
    - Must Analysis
    - 边界情况：Out[Entry] = Uni
    - 基本块初始化：In[blk] = empty, Out[blk] = empty
    - meet值：取交集
    - transferStmt：

```
    // 非ASSIGN语句但是语句中存在def的情况
    if(VCALL(fact->def_to_use, exist, new_def)) {
```

```
        IR_var use = VCALL(fact->def_to_use, get, new_def);
        VCALL(fact->def_to_use, delete, new_def);
        VCALL(fact->use_to_def, delete, use);
    }
    if(VCALL(fact->use_to_def, exist, new_def)) {
        IR_var def = VCALL(fact->use_to_def, get, new_def);
        VCALL(fact->use_to_def, delete, new_def);
        VCALL(fact->def_to_use, delete, def);
    }
    // ASSIGN语句
    VCALL(fact->def_to_use, set, def, use);
    VCALL(fact->use_to_def, set, use, def);
```

- 根据思路填空即可

基于数据流分析的活跃变量分析

- 活跃变量分析填写思路如下
  - 后向分析（Backward）
  - May Analysis
  - In[Exit] = empty
  - In[blk] = empty; Out[blk] = empty
  - meet: Out[blk] = ∪(all Out[pred])
  - transferStmt: In[blk] = gen ∪ (Out[blk] - kill)
- 根据思路填写即可