编译原理Lab4

专业	姓名	学号
计算机科学与技术	吴浩岚	19335209

相关指令

生成优化后的IR

```
export PATH=~/sysu/bin:$PATH   CPATH=~/sysu/include:$CPATH
LIBRARY_PATH=~/sysu/lib:$LIBRARY_PATH
LD_LIBRARY_PATH=~/sysu/lib:$LD_LIBRARY_PATH &&  clang -E tester/wuhlan.c | clang
-cc1 -ast-dump=json | /home/wuhlan3/sysu/build/generator/sysu-generator |
/home/wuhlan3/sysu/build/optimizer/sysu-optimizer
```

相关函数

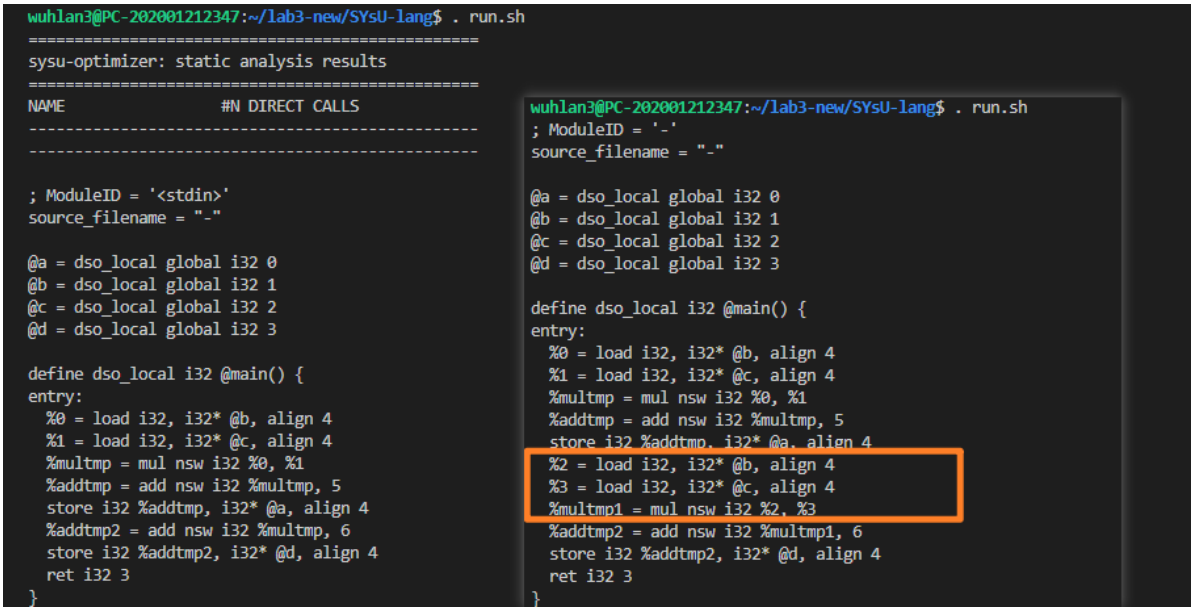
```
//获取操作符名称
getOpcodeName()
//获取操作符数量
getNumOperands()
//获取操作符
I->getOperand(i)
//替换
这个函数用一个值替换给定指令的所有使用，然后删除原始指令。
ReplaceInstWithValue(BasicBlock::InstListType &BIL,  BasicBlock::iterator &BI,
Value *V)
```

消除公共子表达式

针对下面的代码：

```
int a = 0;
int b = 1;
int c = 2;
int d = 3;
int main(){
    a = b * c + 5;
    d = b * c + 6;
    return 3;
}
```

对比优化后的IR，如下图所示：



左边为优化后的IR，可以看到，消除公共子表达式后，减少了b*c的取值和计算过程，代码变为这样的形式：

```
int a = 0;
int b = 1;
int c = 2;
int d = 3;
int main(){
    int temp = b * c;
    a = temp + 5;
    d = temp + 6;
    return 3;
}
```

实验结果

排行榜分数

performance 0.19701919712938304

score 422

提交排行

消除公共子表达式这一部分，在优化的过程中，耗时比较长。对于死代码消除等样例，可能会增加其生成IR的时间，所以最终导致相比于没有优化，又有两个样例超时了。后续，应该优先考虑一下，怎么进行死代码消除。