实验三：LLVM IR循环不变量外提优化

* 1. 实验目的

1.实现LLVM IR的循环不变量外提优化，减少代码运行时执行的代码条数。

2.保持程序语义不变，确保优化后代码的正确性。

1.2 实验环境

1. 硬件环境：鲲鹏开发板。

2. 软件环境：openEuler操作系统。

3. 测试环境：提供一组Cminus语言的测试代码。

1.3 实验内容

1. 实现循环不变量外提优化，包括四则运算结果、赋值load、分配空间、类型转化、比较表达式、零拓展指令和取数组值的循环不变量外提。

1.4 实验过程

1.4.1 循环不变量外提

先获取所有函数，通过遍历每个函数来获取每个函数中的所有循环，遍历所有获得的循环，找到每个循环的最内层循环，将最内层的循环入口放入inner\_loops中，可以避免重复查找。然后，对于每一个最内层循环，由最内层循环访问到最外层循环，然后再访问下一个最内层循环，可以保证对所有循环不变量均进行了代码外提。

实现代码外提

1. **if**(moves.size() != 0){
2. **for**(**auto** move: moves){
3. **auto** prebbs = loop\_searcher.get\_loop\_base(in2out\_loop)->get\_pre\_basic\_blocks();
4. *//找到前继中在循环外部的基本块*
5. **for** (**auto** prebb : prebbs ){
6. **if**(in2out\_loop->find(prebb) == in2out\_loop->end()){
7. destina = prebb;
8. }
9. }
10. *//先删除最后一个终止指令，再将需要外提的代码插入，在最后插入原来被删除的终止指令*
11. **auto** term = destina->get\_terminator();
12. **if**(term != nullptr){
13. destina->delete\_instr(term);
14. move->get\_parent()->delete\_instr(move);
15. move->set\_parent(destina);
16. destina->add\_instruction(move);
17. destina->add\_instruction(term);
18. }
19. }
20. }

优化前后的IR对比（举一个例子）并辅以简单说明：

test.cminus:

**void** **main**(**void**){

**int** i;

**int** j;

**int** ret;

i = 1;

**while**(i<10000)

{

j = 0;

**while**(j<10000)

{

ret = (i\*i\*i\*i\*i\*i\*i\*i\*i\*i)/i/i/i/i/i/i/i/i/i/i;

j=j+1;

}

i=i+1;

}

output(ret);

**return** ;

}

优化前test.ll:

; ModuleID = 'cminus'

source\_filename = "../tests/4-ir-opt/testcases/LoopInvHoist/testcase-1.cminus"

declare i32 @input()

declare void @output(i32)

declare void @outputFloat(float)

declare void @neg\_idx\_except()

define void @main() {

label\_entry:

%op0 = alloca i32

%op1 = alloca i32

%op2 = alloca i32

store i32 1, i32\* %op0

br label %label3

label3: ; preds = %label\_entry, %label58

%op4 = load i32, i32\* %op0

%op5 = icmp slt i32 %op4, 10000

%op6 = zext i1 %op5 to i32

%op7 = icmp ne i32 %op6, 0

br i1 %op7, label %label8, label %label9

label8: ; preds = %label3

store i32 0, i32\* %op1

br label %label11

label9: ; preds = %label3

%op10 = load i32, i32\* %op2

call void @output(i32 %op10)

ret void

label11: ; preds = %label8, %label16

%op12 = load i32, i32\* %op1

%op13 = icmp slt i32 %op12, 10000

%op14 = zext i1 %op13 to i32

%op15 = icmp ne i32 %op14, 0

br i1 %op15, label %label16, label %label58

label16: ; preds = %label11

%op17 = load i32, i32\* %op0

%op18 = load i32, i32\* %op0

%op19 = mul i32 %op17, %op18

%op20 = load i32, i32\* %op0

%op21 = mul i32 %op19, %op20

%op22 = load i32, i32\* %op0

%op23 = mul i32 %op21, %op22

%op24 = load i32, i32\* %op0

%op25 = mul i32 %op23, %op24

%op26 = load i32, i32\* %op0

%op27 = mul i32 %op25, %op26

%op28 = load i32, i32\* %op0

%op29 = mul i32 %op27, %op28

%op30 = load i32, i32\* %op0

%op31 = mul i32 %op29, %op30

%op32 = load i32, i32\* %op0

%op33 = mul i32 %op31, %op32

%op34 = load i32, i32\* %op0

%op35 = mul i32 %op33, %op34

%op36 = load i32, i32\* %op0

%op37 = sdiv i32 %op35, %op36

%op38 = load i32, i32\* %op0

%op39 = sdiv i32 %op37, %op38

%op40 = load i32, i32\* %op0

%op41 = sdiv i32 %op39, %op40

%op42 = load i32, i32\* %op0

%op43 = sdiv i32 %op41, %op42

%op44 = load i32, i32\* %op0

%op45 = sdiv i32 %op43, %op44

%op46 = load i32, i32\* %op0

%op47 = sdiv i32 %op45, %op46

%op48 = load i32, i32\* %op0

%op49 = sdiv i32 %op47, %op48

%op50 = load i32, i32\* %op0

%op51 = sdiv i32 %op49, %op50

%op52 = load i32, i32\* %op0

%op53 = sdiv i32 %op51, %op52

%op54 = load i32, i32\* %op0

%op55 = sdiv i32 %op53, %op54

store i32 %op55, i32\* %op2

%op56 = load i32, i32\* %op1

%op57 = add i32 %op56, 1

store i32 %op57, i32\* %op1

br label %label11

label58: ; preds = %label11

%op59 = load i32, i32\* %op0

%op60 = add i32 %op59, 1

store i32 %op60, i32\* %op0

br label %label3

}

优化后test.ll:

declare i32 @input()

declare **void** @output(i32)

declare **void** @outputFloat(**float**)

declare **void** @neg\_idx\_except()

define **void** @main() {

label\_entry:

%op0 = alloca i32

%op1 = alloca i32

%op2 = alloca i32

store i32 1, i32\* %op0

%op12 = load i32, i32\* %op1

%op13 = icmp slt i32 %op12, 10000

%op14 = zext i1 %op13 to i32

%op15 = icmp ne i32 %op14, 0

%op17 = load i32, i32\* %op0

%op18 = load i32, i32\* %op0

%op19 = mul i32 %op17, %op18

%op20 = load i32, i32\* %op0

%op21 = mul i32 %op19, %op20

%op22 = load i32, i32\* %op0

%op23 = mul i32 %op21, %op22

%op24 = load i32, i32\* %op0

%op25 = mul i32 %op23, %op24

%op26 = load i32, i32\* %op0

%op27 = mul i32 %op25, %op26

%op28 = load i32, i32\* %op0

%op29 = mul i32 %op27, %op28

%op30 = load i32, i32\* %op0

%op31 = mul i32 %op29, %op30

%op32 = load i32, i32\* %op0

%op33 = mul i32 %op31, %op32

%op34 = load i32, i32\* %op0

%op35 = mul i32 %op33, %op34

%op36 = load i32, i32\* %op0

%op37 = sdiv i32 %op35, %op36

%op38 = load i32, i32\* %op0

%op39 = sdiv i32 %op37, %op38

%op40 = load i32, i32\* %op0

%op41 = sdiv i32 %op39, %op40

%op42 = load i32, i32\* %op0

%op43 = sdiv i32 %op41, %op42

%op44 = load i32, i32\* %op0

%op45 = sdiv i32 %op43, %op44

%op46 = load i32, i32\* %op0

%op47 = sdiv i32 %op45, %op46

%op48 = load i32, i32\* %op0

%op49 = sdiv i32 %op47, %op48

%op50 = load i32, i32\* %op0

%op51 = sdiv i32 %op49, %op50

%op52 = load i32, i32\* %op0

%op53 = sdiv i32 %op51, %op52

%op54 = load i32, i32\* %op0

%op55 = sdiv i32 %op53, %op54

%op56 = load i32, i32\* %op1

%op57 = add i32 %op56, 1

%op4 = load i32, i32\* %op0

%op5 = icmp slt i32 %op4, 10000

%op6 = zext i1 %op5 to i32

%op7 = icmp ne i32 %op6, 0

%op59 = load i32, i32\* %op0

%op60 = add i32 %op59, 1 *//以上新增的指令均为循环不变量外提的结果*

br label %label3 *//br指令在基本块最后位置不变*

label3: ; preds = %label\_entry, %label58

br i1 %op7, label %label8, label %label9

label8: ; preds = %label3

store i32 0, i32\* %op1

br label %label11

label9: ; preds = %label3

%op10 = load i32, i32\* %op2

call **void** @output(i32 %op10)

ret **void**

label11: ; preds = %label8, %label16

br i1 %op15, label %label16, label %label58

label16: ; preds = %label11

store i32 %op55, i32\* %op2

store i32 %op57, i32\* %op1

br label %label11

label58: ; preds = %label11

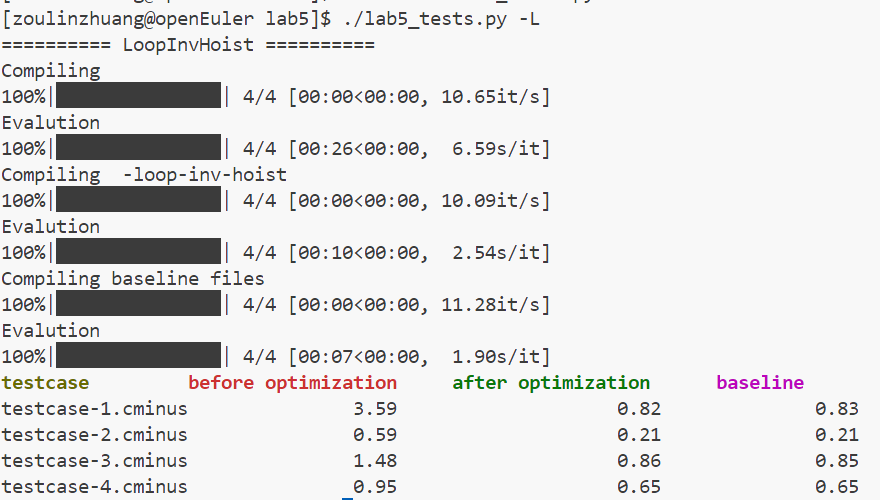
store i32 %op60, i32\* %op0

br label %label3

}

1.5 实验结果

通过循环不变量外提优化，成功减少了LLVM IR中的循环冗余指令，提高了代码运行效率。



1.6 实验总结

本次实验加深了对LLVM IR优化技术的理解，掌握了循环不变量外提优化的具体实现方法，并通过实验验证了优化效果。

姓名：邹林壮 学号：202208040412