实验一：LLVM IR常量传播优化

* 1. 实验目的

1.实现LLVM IR的常量传播优化，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. 实验要求

1.四则运算表达式的常量折叠

2.比较表达式的常量折叠

3.零拓展指令常量折叠

4.类型转化指令常量折叠

5.load和store联合使用时的常量传播

6.全局变量的常量传播

* 1. 实验内容

1.实现常量传播优化，包括四则运算表达式、比较表达式、零拓展指令、类型转化指令的常量折叠。

2.实现load和store联合使用时的常量传播。

3.实现全局变量的常量传播。

1.5 实验过程

1.5.1 常量折叠

对可能产生常量结果的指令，检查指令的操作数是否均为常量，进而确定指令结果是否为常量，如果是则直接生成常量代替原有的值。

1.5.2 常量传播

在进行常量传播时需要考虑当前变量名对应的“状态”，是属于常量状态还是处于变量状态。对常量状态的变量的使用应该向前找到其常量值，直接使用。

1.5.3具体实现

1. 常量折叠

这一块比较简单，主要思路是对可能产生常量结果的指令，检查指令的操作数是否均为常量，进而确定指令结果是否为常量，如果是则直接调用ConstFolder::computeBinary()或者ConstantFolder::computeCMP（）直接生成常量代替原有的值。

1. 常量传播

这一块比较困难，主要思路是在进行常量传播时需要考虑当前变量名对应的“状态”，是属于常量状态，还是处于变量状态。我们对常量状态的变量的使用应该向前找到其常量值，直接使用。

相应代码：(节选部分重要代码)

1. 文件：

ConstantPropagation.hpp: 头文件

ConstantPropagation.cpp: C++文件

1. 函数：

ConstantInt \*ConstFolder::computeBinary() : 两个Int常量的常量折叠

ConstantFP \*ConstFolder::computeBinary(): 两个FLOAT常量的常量折叠

ConstantInt \*ConstFolder::computeCMP(): 两个常量比较的常量折叠

ConstantFP \*cast\_constantfp(): 判断是否为FLOAT常量

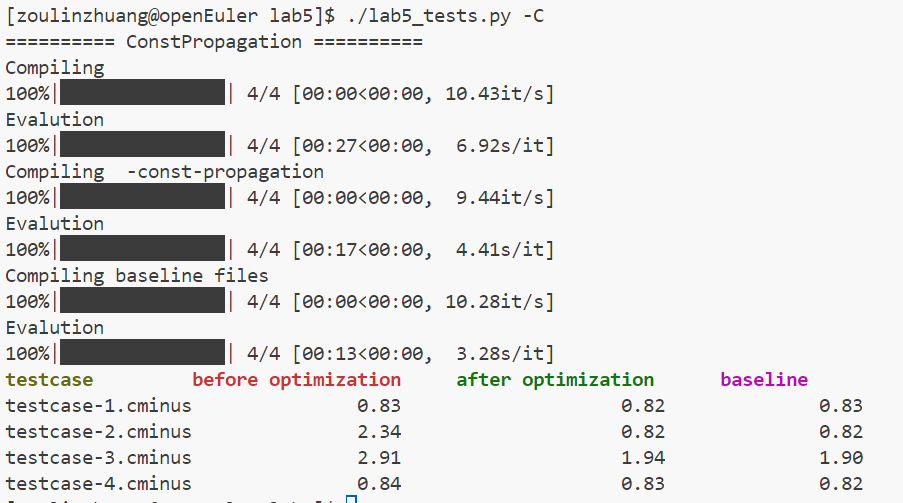
ConstantInt \*cast\_constantint(): 判断是否为Int常量

void ConstPropagation::run()：常量折叠与传播

|  |
| --- |
| *// store指令时记录时候对某个变量有过常数的赋值*  *// 右值非常量*  *// 擦除对应的表项*  **if** (r\_val\_int\_constantptr)  {  *// 保存了一个int常量*  *// printf("SAVE A INT!\n");*  std::cout << r\_val->get\_name();  ConstantInt\_var\_list.insert({l\_val, r\_val\_int\_constantptr});  }  **else** **if** (r\_val\_fp\_constantptr)  {  ConstantFP\_var\_list.insert({l\_val, r\_val\_fp\_constantptr});  }  **else**  {  *// 根本不是常量，擦除*  **if** (ConstantInt\_var\_list.find(l\_val) != ConstantInt\_var\_list.end())  {  ConstantInt\_var\_list.erase(l\_val);  }  **if** (ConstantFP\_var\_list.find(l\_val) != ConstantFP\_var\_list.end())  {  ConstantFP\_var\_list.erase(l\_val);  }  }  *// store指令中将常数变量值直接插入*  **if** (ConstantInt\_var\_list.find(r\_val) != ConstantInt\_var\_list.end())  {  *// 表中显示这是int常量*  *// printf("USE A INT!\n");*  **auto** key = ConstantInt\_var\_list.find(r\_val);  instr->replace\_all\_use\_with(key->second);  wait\_delete.push\_back(instr);  }  **else** **if** (ConstantFP\_var\_list.find(r\_val) != ConstantFP\_var\_list.end())  {  **auto** key = ConstantFP\_var\_list.find(r\_val);  instr->replace\_all\_use\_with(key->second);  wait\_delete.push\_back(instr);  }  **continue**; |

1.6 实验结果

通过常量传播优化，成功减少了LLVM IR中的冗余指令，提高了代码运行效率。



1.7 实验总结

本次实验加深了对LLVM IR优化技术的理解，掌握了常量传播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