report

学号	姓名
20337091	马佳欣

Deadcode

不使用的局部变量、函数参数、函数

在generator,使用is_promotable判断。

• 优点:在generator阶段就可以实现,不需要生成了ir再删除ir。

• 注意: main不会在其他地方被调用, 所以判断函数是否可优化时要判断是否为main函数。

强度削减

在generator,在BinaryExprToIR处判断。

- 优点:在generator阶段就可以简单实现,在optimizer阶段实现代码更复杂
- 优化方向:
 - 1. Algebraic Identity $x+0=0+x, x\times 1=1\times x\Rightarrow x$ // 只能消除整型,不能消除浮点类型
 - 2. Strength Reduction $2^n \times x = x \times 2^n \Rightarrow x \ll n$

Mem2Reg

优化流程

• 在代码中已有详细的注释。

注意事项

• 值栈 SmallDenseMap<AllocaInst *, std::stack<Value *>>value_stack_of_alloca 需要 undef 兜底值:

phi 节点对应的值可能是 undef 的, 例如:

对于while.cond中的a变量的phi节点,从entry进入的值是未知的。

```
1 define void @func() {
 2
   entry:
 3
     br label %while.cond
 4
 5 while.cond:
                                                       ; preds =
    %while.body, %entry
 6
     %a0 = phi i32 [ undef, %entry ], [ %1, %while.body ]
 7
     %0 = icmp \ sgt \ i32 \ %a0, 0
     br i1 %0, label %while.body, label %while.end
 8
 9
10 while.body:
                                                       ; preds = %while.cond
11
     %1 = \text{sub i} 32 \% a0, 1
12
     br label %while.cond
13
14 while.end:
                                                       ; preds = %while.cond
    ret void
15
16 }
```

• to_remove_loads和to_remove_stores:

不要在遍历 inst 的时候删除,会导致迭代器失效,所以需要先暂存起来。

问题和解决方法

1. o tester/function_test2020/02_arr_defn4.sysu.c:

- o tester/function_test2020/35_array_test.sysu.c: 数组取地址也是 <badref>
- 。 解决方法: 不优化数组。
- 2. tester/function_test2020/10_break.sysu.c:

循环中 a 值会发生变化,且根据 a 值判断是否继续执行循环,但是上述 phi 节点被放在了block的最后面,导致出错。

- o 解决方法:将 phi 节点插在 block 的最前面。
- 3. 使用了 Mem2Reg 优化会出错 (段错误) 的测例:
 - 1. tester/function_test2022/73_int_io.sysu.c
 - 2. tester/h_functional/090_int_io.sysu.c (与1是一样的)

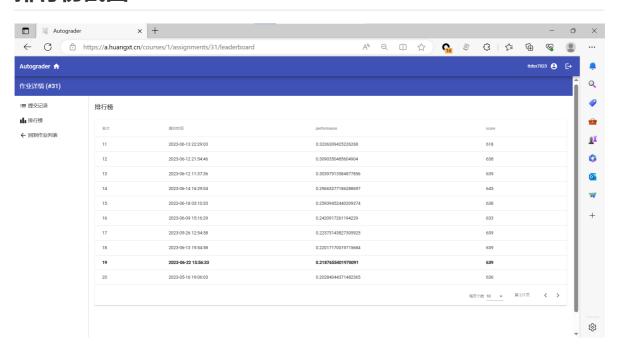
- 3. third_party/SYsU-lang-tester-perfermance/performance_test2021-public/median2.sysu.c
- 4. /workspace/SYsU-lang/long_tester/third_party/SYsU-lang-tester-perfermance/performance_test2022-private/vector_mul2.sysu.c

解决方法:在 main 函数判断即将优化的函数是否为测例里面的函数,若是则不进行优化。

4. 优化除法

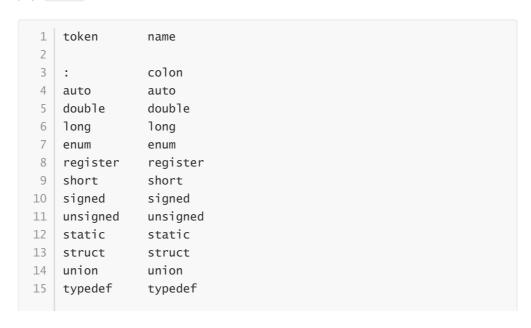
强度削减中一开始包括除法优化,但是到了某个测例就出错了,因此放弃除法优化。

排行榜截图



挑战工作

- lexer:
 - 1. 扩展更多 C 语言的 token
 - 踩过的坑:
 - 不支持数据类型 bool , Clang 将其识别为 identifier ;
 - 不需要写 long long 类型对应的规则,Clang 将其识别为两个 long。
 - 更多 token:



```
16 do
              do
17
   switch
             switch
18 case
             case
19 default
             default
20 for
             for
21 goto
             goto
22 sizeof
             sizeof
            extern
23 extern
24 ++
             plusplus
25 +=
             plusequal
26 --
             minusminus
27 -=
             minusequal
28 *=
             starequal
             slashequal
29 /=
30 %=
              percentequal
```

2. 根据新扩展的 token ,编写新的测例 tester/mytest.c。