中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

ARCSYSU

Refines Computing SYStem Uses

编译系统设计赛作品展示

中山大学 Yat-CC

队伍介绍

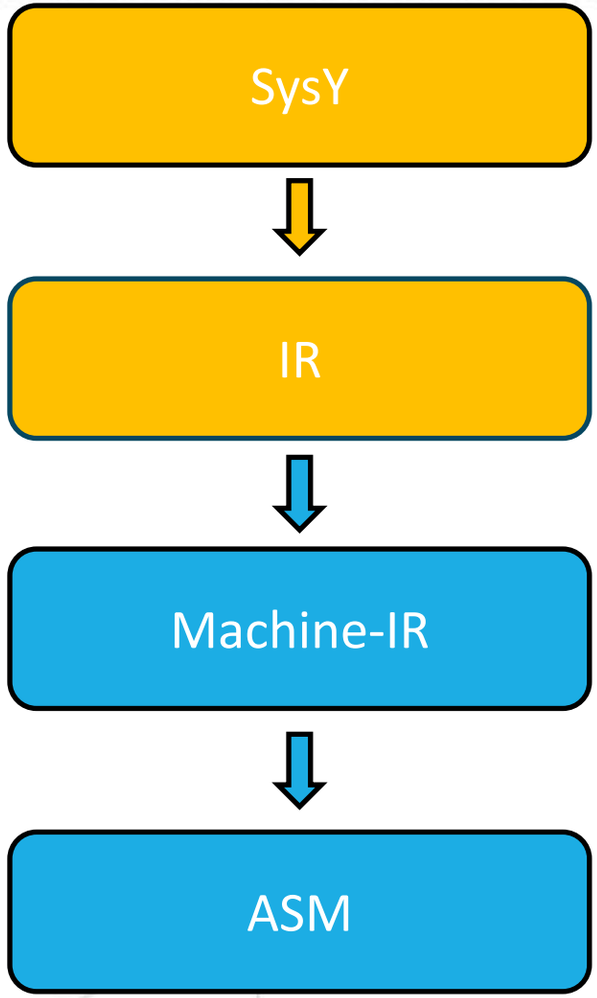
队伍成员

- 指导老师：张献伟副教授
- 队长：郑中淳 2019级本科生
- 队员：黄瀚 2019级本科生
- 队员：潘文轩 2019级本科生
- 队员：陈冠一 2020级本科生

总体优化介绍

前端

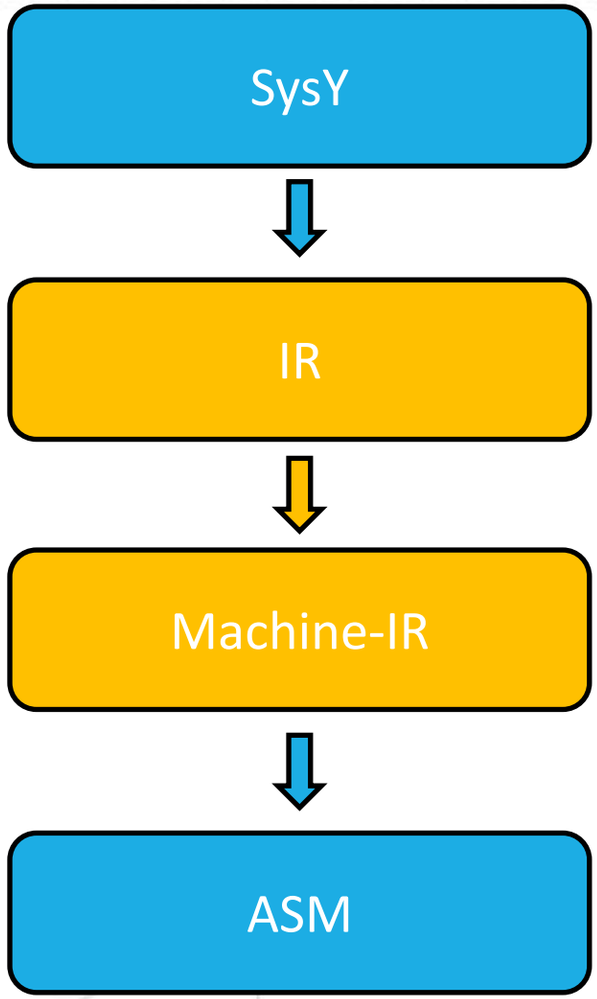
- 使用ANTLR，将SysY语言分析成IR
- 使用的IR与LLVM等价，可以直接使用llvm后端执行



总体优化介绍

中端

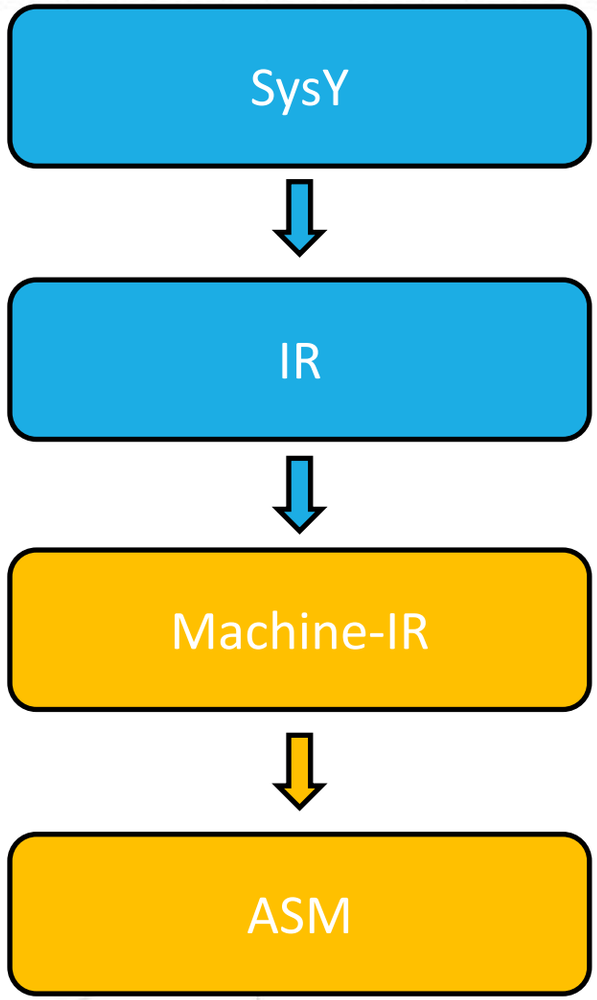
- SSA转换 (Mem2Reg)
- 死代码消除（Dead Code Elimination）
- 公共子表达式消除（Common Subexpression Elimination）
- 常量传播（Constant Propagation）
- 常量折叠 (Constant Folding)
- 函数内联（Function Inline）
- 指令重排（Reassociation）
- 循环优化（Loop Optimization）
- 控制流简化（CFG Simplification）



总体优化介绍

后端

- 死代码消除（Dead Code Elimination）
- 强度削减（Strength Reduction）
- 地址生成（Address Generation）
- 窥孔优化（Peephole）
- 寄存器分配（Register Allocation）

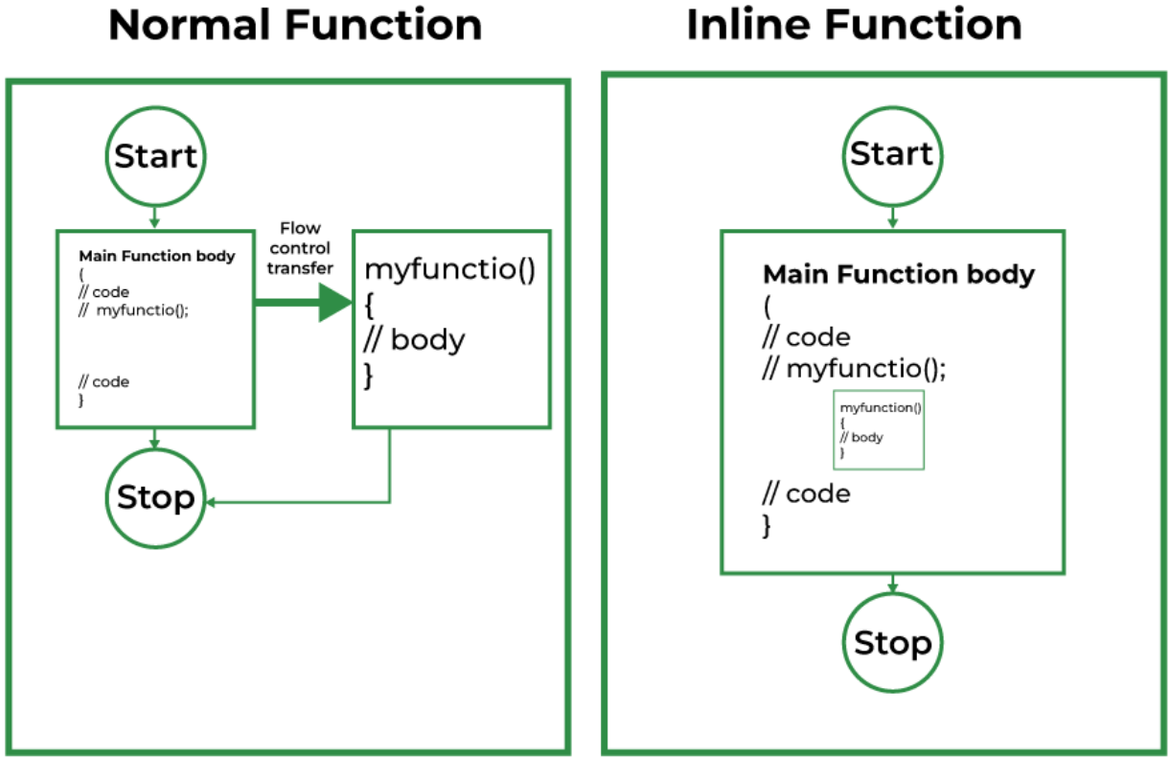


函数内联

优化简介

– Function Inline

- 如果在函数调用图中，一个函数不在任何环中，则将该函数进行内联
- 对内联函数的每个调用，复制一次函数，将函数参数替换为call指令的参数，使用复制的函数替换call指令
- 尽可能地进行函数内联，以获得更多的优化机会，但可能导致代码体积变大，损失指令缓存的时间



指令重排

优化简介

– Reassociation

- 对所有满足交换律和结合律的二元运算指令进行重排和组合
- IR中的二元运算可以视作一棵树，首先找出所有的叶子节点即该式子的所有运算符，例如

  a+b+c*d的叶子节点有三个为a;b;c*d
- 可以将整条算式中的所有常数预计算得到结果
- 可以将多个相同数字的加法改为乘法，如a+a+a->3*a
- 可以提取公因数，如a*b+c*b->b*(a+c)，并将出现次数最频繁的项放在树的最底层，方便后续进行CSE

循环优化

优化简介

- 从IR中重新构建循环
- 循环分析
  - SCEV分析循环代价
  - 分析GEP指令，为后端地址生成提供必要信息
- 循环不变量提取
- 循环展开
  - 当循环代价为常数时才展开（静态循环展开）
  - 展开配合常量传播和死代码消除，可以将多条指令合并

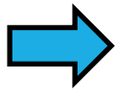
地址生成

优化简介

– LoopGEPCombine

- 中端与后端相结合
- 将GEP计算转化为指针的移动（如将数组寻址替换成ptr = ptr + 4）
- 中端插入GEP伪指令，并由后端生成基址计算指令和指针移动指令

```
entry:
  %a = alloca [10 x i32]
  %call0 = call i32 @getint()
  br label %while.cond0

while.cond0:                ; preds = %while.body0, %entry
  %phi0 = phi i32 [ 0, %entry], [ %binary0, %while.body0]
  %cmp0 = icmp slt i32 %phi0, %call0
  br i1 %cmp0, label %while.body0, label %while.end0

while.body0:                ; preds = %while.cond0
  %ext0 = sext i32 %phi0 to i64
  %arrayidx0 = getelementptr inbounds [10 x i32], [10 x i32]* %a, i64 0, i64 %ext0
  store i32 %phi0, i32* %arrayidx0
  %binary0 = add nsw i32 %phi0, 1
  br label %while.cond0
```



```
entry:
  %a = alloca [10 x i32]
  %call0 = call i32 @getint()
  %loopGEP0 = getelementptr inbounds [10 x i32], [10 x i32]* %a, i64 0, i64 0
  br label %while.cond0

while.cond0:                ; preds = %while.body0, %entry
  %phi0 = phi i32 [ 0, %entry], [ %binary0, %while.body0]
  %loopGEP.phi0 = phi i32* [ %loopGEP0, %entry], [ %arrayidx0, %while.body0]
  %cmp0 = icmp slt i32 %phi0, %call0
  br i1 %cmp0, label %while.body0, label %while.end0

while.body0:                ; preds = %while.cond0
  %ext0 = sext i32 %phi0 to i64
  %arrayidx0 = loopgep %loopGEP.phi0, i64 1
  store i32 %phi0, i32* %arrayidx0
  %binary0 = add nsw i32 %phi0, 1
  br label %while.cond0
```

强度削弱

优化简介

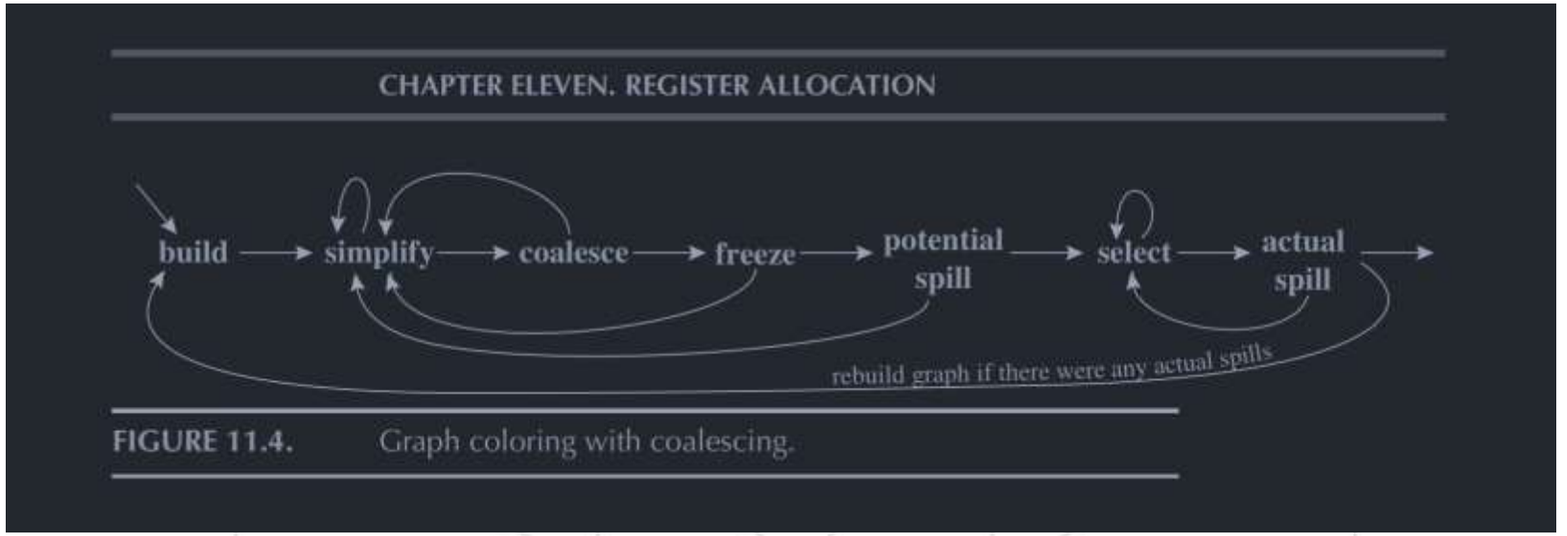
- 对于乘法/除法/取模进行常量优化
- 乘一个常量
  - 当常量为2的幂次时，将乘法优化成移位指令
- 除一个常量
  - 当常量为2的幂次时，优化成64位移位与加法指令
  - 当常量不为2的幂次时，使用magic number简化除法
- 模一个常量
  - A % B = A – (A / B) * B
  - 可规约为除以一个常量

```
struct ms {int M;           // Magic number
          int s;};          // and shift amount.

struct ms magic(int d) {    // Must have 2 <= d <= 2**31-1
                            // or   -2**31 <= d <= -2.
   int p;
   unsigned ad, anc, delta, q1, r1, q2, r2, t;
   const unsigned two31 = 0x80000000;      // 2**31.
   struct ms mag;

   ad = abs(d);
   t = two31 + ((unsigned)d >> 31);
   anc = t - 1 - t%ad;      // Absolute value of nc.
   p = 31;                  // Init. p.
   q1 = two31/anc;          // Init. q1 = 2**p/|nc|.
   r1 = two31 - q1*anc;     // Init. r1 = rem(2**p, |nc|).
   q2 = two31/ad;           // Init. q2 = 2**p/|d|.
   r2 = two31 - q2*ad;      // Init. r2 = rem(2**p, |d|).
   do {
      p = p + 1;
      q1 = 2*q1;            // Update q1 = 2**p/|nc|.
      r1 = 2*r1;            // Update r1 = rem(2**p, |nc|).
      if (r1 >= anc) {      // (Must be an unsigned
         q1 = q1 + 1;       // comparison here.)
         r1 = r1 - anc;}
      q2 = 2*q2;            // Update q2 = 2**p/|d|.
      r2 = 2*r2;            // Update r2 = rem(2**p, |d|).
      if (r2 >= ad) {       // (Must be an unsigned
         q2 = q2 + 1;       // comparison here.)
         r2 = r2 - ad;}
      delta = ad - r2;
   } while (q1 < delta || (q1 == delta && r1 == 0));

   mag.M = q2 + 1;
   if (d < 0) mag.M = -mag.M; // Magic number and
   mag.s = p - 32;            // shift amount to return.
   return mag;
}
```

寄存器分配

图着色算法

- 分析寄存器生存期，建立冲突图
- 对每个虚拟寄存器执行简化、合并、冻结与选择溢出中的一个或多个操作
- 对虚拟寄存器进行着色或溢出



问答环节

谢谢老师