Mental 特性和优化汇总

柯嵩宇

2016 年 5 月 10 日

目录

1 visitor 模式：from AST to linear IR

以 visitor 模式完全手写了 AST 到 IR 的 visitor。（ANTLR 生成 CST，用 ANTLR 的 CST 生成我的 AST，然后用我写的 visitor 生成 IR）

2 逻辑表达式短路求值

2.1 naive 的短路求值

对于二元运算的逻辑与或，可以先计算左边的元素，如果左侧表达式的结果可以决定整个表达式的值，那么就保存结果同时跳过右侧表达式的计算。

2.2 修改逻辑与或运算的结合性

在 superloop 中，出现了一个巨大的逻辑与运算，如果按照上面的方法短路求值，那么在执行过程中会出现大量的跳转。一个比较机智的优化就是修改逻辑与或运算的结合性。如果确定表达式结果的子表达式出现在整个表达式的前面，那么就可以用比较少的跳转离开这个表达式的计算。

2.3 Super Expression

修改逻辑与或运算结合性的优化并不是那么完美，有一个更厉害的优化：定义广义逻辑与或运算，允许逻辑与或运算有 2 个以上的操作数，这样在处理的时候就可以看到所有的操作数，然后就可以用 1 次的跳转离开表达式的求值。

3 Linear Scan 的局部寄存器分配

线性扫描每一个 Basic Block，然后以贪心的方式分配寄存器

4 伪活性分析：数据引用计数

对于非变量的数据计算引用次数，如果引用次数到 0，就不需要在保存到寄存器中。

5 超越语义的 print 语句

在 Mx 语言中，print 和 println 的函数只能是一个字符串，对于输出语句中的字符串加法（作为参数传给 print 函数）而言，这个加法的结果在以后完全不会被用到，所以在这个字符串加法其实是没有必要进行的，可以通过拆分 print(a+b) 为 print(a) 和 print(b) 来优化输出语句的效率。当然由于 MIPS 本身存在输出整数的 syscall，所以如果要完全优化的话，可以把 print(toString(x)) 直接优化掉（我并没有写到这一步，我就写了加法的拆分）。