

预计完成时间: 12 月 12 日, 星期一

在项目的最后阶段, 学生要实现一些结构级优化。学生要实现寄存器分配。任何剩余的优化都是可选的。

• **寄存器分配**—编译器必须要能够实现基于寄存器分配的图着色。“Always or never”的分配算法事不够的, 但是我们也不要要求实现网格分割和合并。见鲸书的 16 章和龙书的 § 9.7。学生的分配算法应该充分利用 x86 的一般目的寄存器, 而不仅仅事 r10 和 r11。同时需要考虑寄存器的调用者保护和被调用者保护属性。

除了给程序变量和临时变量分配寄存器以外, 学生还必须使用这些寄存器来传递在完整调用惯例中指定的函数实参。

• **指令执行时序**—指令执行时序使长延迟操作, 如载入, 乘和除, 的管道阻塞最小化。见鲸书的 17 章。

• **指令选择**—到目前为止, 我们只是使用了 x86-64 很有限的指令子集。和窥视孔优化一样, 学生可以用更高效的指令序列来代替老的指令序列。举例来说, 直接从存储器中压入和弹出值使可能的, 这就意味着不需要中间临时寄存器。

关于结构方面的细节, 见完备的 AMD64 参考资料 (在删减版本上有链接)。为了在 chocura 上加快代码执行速度, 可以使用 -gcc 的 pg 选项, 以生成分析信息, 使用 gprof 来做真正的分析工作。

提交

编译器必须提供如下的命令行标志:

- -opt regalloc: 打开寄存器分配
- -opt instsel 打开指令选择窥视孔优化 (可选)
- -opt sched: 打开指令执行时序 (可选)

在最后的作业中, 学生必须提供 -opt all 标志来打开所有优化功能。如果学生选择要实现指令执行时序, 学生必须仔细思考编译器执行的顺序。学生可能甚至会希望对一些优化执行两次。

和以前一样, 所生成的代码要能够执行列示在语言规格书中的运行时间检测。只要这些代码能在未优化程序报告错误的时候报告运行时间错误, 那么他们就是可以被优化的 (或者在一些情况下使可以移出的)。

在提交方面请遵从分发资料 3 中的要求。对于电子部分, 文件的命名为 leNN-instruction.jar 和 leNN-instruction.tar.gz。书写文档部分要呈现你们如何决定优化执行顺序的讨论。请给出表明优化功能清晰的 MIPS 示例摘要, 并且描述对于一些棘手问题的任何解决方法或程序。

测试

这部分没有新的测试 case。但是, 学生应该要确认任何合法的程序都能够成功执行。我们会

在一些隐藏测试中测试编译器。基于优化的有效性和正确性，我们会给出相应分数。