<<编译器设计>>

图书基本信息

书名：<<编译器设计>>

13位ISBN编号：9787115301949

10位ISBN编号：7115301948

出版时间：2012-12

出版时间：人民邮电出版社

作者：Keith Cooper,Linda Torczon

页数：578

字数：990000

译者：郭旭

<<编译器设计>>

前言

构建编译器的实践方法一直在不断变化，部分是因为处理器和系统的设计会发生变化。
例如，当我们在1998年开始写作本书初版时，一些同事对书中指令调度方面的内容颇感疑惑，因为乱序执行威胁到了指令调度，很有可能会使其变得不再重要。
现在第2版已经付印，随着多核处理器的崛起和争取更多核心的推动，顺序执行流水线再次展现吸引力，因为这种流水线占地较少，设计者能够将更多核心放置在一块芯片上。
短期内，指令调度仍然很重要。

同时，编译器构建社区还将继续产生新的思路和算法，并重新发现原本有效但在很大程度上却被遗忘的旧技术。
围绕着寄存器分配中弦图（chordal graph）使用（参见13.5.2节）的最新研究颇为令人振奋。
该项工作承诺可以简化图着色分配器（graph-coloring allocator）的某些方面。
Brzozowski的算法是一种DFA最小化技术，可以追溯到20世纪60年代早期，但却已有数十年未在编译器课程中讲授了（参见2.6.2节）。
该算法提供了一种容易的路径，可以从子集构造（subset construction）的实现得到一个最小化DFA的实现。
编译器构建方面的现代课程本该同时包括这两种思想。

那么，为了让学习者准备好进入这个不断变化的领域，我们该如何设计编译器构建课程的结构呢？
我们相信，这门课应该使每个学生学会建立新编译器组件和修改现存编译器所需的各项基本技能。
学生既需要理解笼统的概念，如链接约定中隐含的编译器、链接器、装载器和操作系统之间的协作，也需要理解微小的细节，如编译器编写者如何减少每个过程调用时保存寄存器的代码总共所占的空间。

第2版中的改变　　本书提供了两种视角：编译器构建领域中各问题的整体图景，以及各种可选算法方案的详细讨论。
在构思本书的过程中，我们专注于该书的可用性，使其既可作为教科书，又可用做专业人士的参考书。
为此，我们特别进行了下述改动。

改进了阐述思想的流程，以帮助按顺序阅读本书的学生。
每章章首简介会解释该章的目的，列出主要的概念，并概述主题相关内容。
书中的示例已经重写过，使得章与章之间的内容具有连续性。
此外，每章都从摘要和一组关键词开始，以帮助那些会将本书用做参考书的读者。

在每节末尾都增加了本节回顾和复习题。
复习题用于快速检查读者是否理解了该节的要点。

关键术语的定义放在了它们被首次定义和讨论的段落之后。

大量修订了有关优化的内容，使其能够更广泛地涵盖优化编译器的各种可能性。

现在的编译器开发专注于优化和代码生成。
对于新雇用的编译器编写者来说，他们往往会被指派去将代码生成器移植到新处理器，或去修改优化趟，而不会去编写词法分析器或语法分析器。
成功的编译器编写者必须熟悉优化（如静态单赋值形式的构建）和代码生成领域当前最好的实践技术（如软件流水线）。
他们还必须拥有相关的背景和洞察力，能理解未来可能出现的新技术。
最后，他们必须深刻理解词法分析、语法分析和语义推敲（semantic elaboration）技术，能构建或修改编译器前端。

本书是一本教科书、一门教程，帮助学生接触到现代编译器领域中的各种关键问题，并向学生提供解决这些问题所需的背景知识。
从第1版开始，我们就维持了各主题之间的基本均衡。

前端是实用组件，可以从可靠的厂商购买或由某个开源系统改编而得。
但是，优化器和代码生成器通常是对特定处理器定制的，有时甚至针对单个处理器型号定制，因为性能严重依赖于所生成代码的底层细节。
这些事实影响到了当今构建编译器的方法，它们也应该影响我们讲授编译器构建课程的方法。
本书结构　　本书内容划分为篇幅大致相等的四个部分。
第一部分（第2章～第4章）涵盖编译器前端及建立前端所用工具的设计和构建。
第二部分（第5章～第7章）探讨从源代码到编译器的中间形式的映射，这些章考查前端为优化器和后端所生成代码的种类。
第三部分（第8章～第10章）介绍代码优化。
第8章提供对优化的概述。
第9章和第10章包含了对分析和转换的更深入的处理，本科课程通常略去这两章。
第四部分（第11章～第13章）专注于编译器的后端所使用的算法。
编译的艺术性与科学性　　编译器构建的内容有两部分，一是将理论应用到实践方面所取得的惊人成就，一是对我们能力受限之处的探讨。
这些成就包括：现代词法分析器是通过应用正则语言的理论自动构建识别器而建立的；LR语法分析器使用同样的技术执行句柄识别，进而驱动了一个移进归约语法分析器；数据流分析巧妙有效地将格理论应用到程序分析中；代码生成中使用的近似算法为许多真正困难的问题提供了较好的解。
另一方面，编译器构建也揭示了一些难以解决的复杂问题。
用于现代处理器的编译器后端对两个以上的NP完全问题（指令调度、寄存器分配，也许还包括指令和数据安排）采用了近似算法来获取答案。
这些NP完全问题，虽然看起来与诸如表达式的代数重新关联这种问题相近（示例见图7-1）。
但后者有着大量的解决方案，更糟的是，对于这些NP完全问题来说，所要的解往往取决于编译器和应用程序代码中的上下文信息。
在编译器对此类问题近似求解时，会面临编译时间和可用内存上的限制。
好的编译器会巧妙地混合理论、实践知识、技术和经验。
打开一个现代优化编译器，你会发现各式各样的技术。
编译器使用贪婪启发式搜索来探索很大的解空间，使用确定性有限自动机来识别输入中的单词。
不动点算法被用于推断程序行为，通过定理证明程序和代数化简器来预测表达式的值。
编译器利用快速模式匹配算法将抽象计算映射到机器层次上的操作。
它们使用线性丢番图方程和普瑞斯伯格算术（Pressburger arithmetic）来分析数组下标。
最后，编译器使用了大量经典的算法和数据结构，如散列表、图算法和稀疏集实现方法等。
本书尝试同时阐释编译器构建的艺术和科学这两方面内容。
通过选取足够广泛的题材，向读者表明，确实存在一些折中的解决方案，而设计决策的影响可能是微妙而深远的。
另一方面，本书也省去了某些长期以来都列入本科编译器构建课程的技术，随着市场、语言和编译器技术或工具可用性方面的改变，这些技术已变得不那么重要了。
讲述方法　　编译器构建是一种工程设计实践。
每个方案的成本、优点和复杂程度各异，编译器编写者必须在多种备选方案中做出抉择。
每个决策都会影响到最终的编译器。
最终产品的质量，取决于抉择过程中所做的每一个理性决断。
因而，对于编译器中的许多设计决策来说，并不存在唯一的正确答案。
即使在“理解透彻”和“已解决”的问题中，设计和实现中的细微差别都会影响到编译器的行为及其产生的代码的质量。
每个决策都涉及许多方面。
举例来说，中间表示的选择对于编译器中其余部分有着深刻的影响，无论是时间和空间需求，还是应用不同算法的难易程度。
但实际上确定该决策时，通常可供设计者考虑的时间并不多。

第5章考察了中间表示的空间需求，以及其他一些应该在选择中间表示时考虑的问题。
在本书中其他地方，我们会再次提出该问题，既会在正文中直接提出，也会在习题里间接提出。
本书探索了编译器的设计空间，既从深度上阐释问题，也从广度上探讨可能的答案。
它给出了这些问题的某些解决方法，并说明了使用这些方案的约束条件。
编译器编写者需要理解这些问题及其答案，以及所作决策对编译器设计的其他方面的影响。
只有这样，编写者才能作出理性和明智的选择。
思想观念　　本书阐释了我们在构建编译器方面的思想观念，这是在各自超过25年的研究、授课和实践过程中发展起来的。
例如，中间表示应该展示最终代码所关注的那些细节，这种理念导致了对底层表示的偏爱。
又比如，值应该驻留在寄存器中，直至分配器发现无法继续保留它为止，这种做法产生了使用虚拟寄存器的例子，以及仅在无可避免时才将值存储到内存的例子。
每个编译器都应该包括优化，它简化了编译器其余的部分。
多年以来的经验，使得我们能够理性地选择书的主题和展现方式。
关于编程习题的选择　　编译器构建方面的课程，提供了在一个具体应用程序（编译器）环境中探索软件设计问题的机会，任何具备编译器构建课程背景的学生，都已经透彻理解了该应用程序的基本功能。
在多数编译器设计课程中，编程习题发挥了很大的作用。
我们以这样的方式讲授过这门课：学生从头到尾构建一个简单的编译器，从生成的词法分析器和语法分析器开始，结束于针对某个简化的RISC指令集的代码生成器。
我们也以另一种方式讲授过这门课程，学生编写程序来解决各个良好自包含的问题，诸如寄存器分配或指令调度。
编程习题的选择实际上非常依赖于本课程在相关课程中所扮演的角色。
在某些学校，编译器课程充当高年级的顶级课程，将来自许多其他课程的概念汇集到一个大型的实际设计和实现项目中。
在这样的课程中，学生应该为一门简单的语言编写一个完整的编译器，或者修改一个开源的编译器，以支持新的语言或体系结构特性。
这门课程可以按本书的内容组织，从头到尾讲授本书的内容。
在另外一些学校，编译器设计出现在其他课程中，或以其他方式呈现在教学中。
此时，编译器设计教师应该专注于算法及其实现，比如局部寄存器分配器或树高重新平衡趟这样的编程习题。
在这种情况下，可以选择性讲解本书中的内容，也可以调整讲述的顺序，以满足编程习题的需求。
例如，在莱斯大学，我们通常使用简单的局部寄存器分配器作为第一个编程习题，任何具有汇编语言编程经验的学生，都可以理解该问题的基本要素。
但这种策略，需要让学生在学习第2章之前，首先接触第13章的内容。
不管采用哪种方案，本课程都应该从其他课程取材。
在计算机组织结构、汇编语言编程、操作系统、计算机体系结构、算法和形式语言之间，存在着明显的关联。
尽管编译器构建与其他课程的关联不那么明显，但这种关联同等重要。
第7章中讨论的字符复制，对于网络协议、文件服务器和Web服务器等应用程序的性能而言，都发挥着关键的作用。
第2章中用于词法分析的技术可以应用到文本编辑和URL过滤等领域。
第13章中自底向上的局部寄存器分配器是最优离线页面替换算法MIN的“近亲”。
补充材料　　还有一些补充的资源可用，可帮助读者改编本书的内容，使之适用于自己的课程。
这包括作者在莱斯大学讲授这门课程的一套完整的讲义以及习题答案。
读者可以联系本地的Elsevier业务代表，询问如何获取这些补充材料。
致谢　　许多人参与了第1版的出版工作，他们的贡献也体现在第2版中。
许多人指出了第1版中的问题，包括Amit Saha、Andrew Waters、Anna Youssefi、Ayal Zachs、Daniel Salce

、David Peixotto、Fengmei Zhao、Greg Malecha、Hwansoo Han、Jason Eckhardt、Jeffrey Sandoval、John Elliot、Kamal Sharma、Kim Hazelwood、Max Hailperin、Peter Froehlich、Ryan Stinnett、Sachin Rehki、Sa ·nak Ta··rlar、Timothy Harvey和Xipeng Shen，在此向他们致谢。
我们还要感谢第2版的审阅者，包括Jeffery von Ronne、Carl Offner、David Orleans、K. Stuart Smith、John Mallozzi、Elizabeth White和Paul C. Anagnostopoulos。
Elsevier的产品团队，特别是Alisa Andreola、Andre Cuello和Megan Guiney，在将草稿转换成书的过程中发挥了关键作用。
所有这些人都以其深刻的洞察力和无私的帮助，从各个重要方面提升了本书的质量。
最后，在过去5年中，无论是从精神方面，还是从知识方面，许多人都为我们提供了莫大的支持。
首先，我们的家庭和莱斯大学的同事都在不断地鼓励我们。
特别感谢小女Christine和Carolyn，她们耐心容忍了无数次关于编译器构建方面各种主题的长时间讨论。
Nate McFadden以其耐心和出色的幽默感，从开始到出版，一直指导着本书的工作。
Penny Anderson对于日常行政事务管理方面的帮助对于本书的完成至关重要。
对所有这些人，我们表示衷心的感谢。

<<编译器设计>>

内容概要

《编译器设计(第2版)》是编译器设计领域的经典著作，主要从以下四部分详解了编译器的设计过程。
第一部分涵盖编译器前端设计和建立前端所用工具的设计和构建；第二部分探讨从源代码到编译器中间形式的映射，考察前端为优化器和后端所生成代码的种类；第三部分介绍代码优化，同时包含对分析和转换的进一步处理；第四部分专门讲解编译器后端使用的算法。

《编译器设计(第2版)》适合作为高等院校计算机专业本科生和研究生编译课程的教材和参考书，也可供相关技术人员参考。

作者简介

Keith D.
Cooper是莱斯大学的计算工程Doerr讲席教授。
他研究过编译后代码优化领域的大量问题，包括过程间数据流分析及其应用、值编号、代数重新关联、寄存器分配和指令调度等。
他近期的工作专注于从根本上重新考察传统编译器的结构和行为。
他讲授过各种本科生水平的课程，从程序设计入门到研究生水平的代码优化等。
他还是ACM院士。

Linda
Torczon是莱斯大学计算机科学系的高级研究科学家，她是PACE（平台可感知编译环境）项目的首席研究员，该项目由DARPA（国防高级研究计划局）赞助，意在开发一种优化编译器环境，能够针对新平台自动地调整其优化机制和策略。
从1990年到2000年，Torczon担任并行计算研究中心的执行总监，该中心是美国国家科学基金会下属的一个科技中心。
她还担任过HiPerSoft、洛斯阿拉莫斯计算机科学研究所和虚拟网格应用开发软件项目的执行总监。

<<编译器设计>>

书籍目录

媒体关注与评论

“编译器是一个内容丰富的研究领域，将整个计算机科学融汇在一个优雅的构造中。
Cooper和Torczon的这本书很受欢迎，可以指导读者轻松学习编译器这种软件系统，新版增加了两位作者的一些设计经验，并明确指出了许多必须注意的细节，同时又不忘强调设计的大局观。
对任何不熟悉编译器的人来说，本书都是不可多得的参考手册。
”——Michael D. Smith，哈佛大学文理学院院长，工程与应用科学John H. Finley Jr.讲席教授
“本书是构建现代优化编译器的最佳指南。
作者汲取了编译器构建领域大量的经验，以帮助学生掌握整体设计思路，同时引导学生了解构建有效的优化编译器所必需的许多重要而微妙的细节。
尤其值得一提的是，在我读过的书中，本书对静态单赋值形式的阐述最为清晰。
”——Jeffery von Ronne，得克萨斯大学圣安东尼奥分校计算机科学系助理教授“本书采用了更常规且一致的结构，还包含大量辅助教学的内容，如复习题、附加示例、术语解释和文本框说明等，这提升了它作为教科书的价值。
本书还包括大量技术上的更新，包括非传统语言、实际编译器和编译器技术非传统用途方面的更多内容。
优化方面的内容是第1版的特色，这一版中变得更为清晰易读。
”——Michael L. Sccot，罗彻斯特大学计算机科学系教授，Programming Language Pragmatics的作者
“Keith Cooper和Linda Torczon不仅很好地讲述了编译器的历史，也从实践者的角度阐述了如何开发编译器。
书中包括了大量颇具实用价值的讨论和说明，既涉及理论，也涉及众多现存编译器的实例（如Lisp、FORTRAN等）。
对入门和高级“分配”与“优化”概念的全面讨论，实际上涵盖了编译器设计的整个生命周期。
对于计算机科学专业的学生以及编译器设计和开发人员来说，本书都是必备参考书。
”——David Orleans，诺瓦东南大学“这本书写得实在是棒极了，内容翔实，辅以大量图表和示例说明，作为大学编译器课程的教科书和从业人员的参考书再合适不过了。
代码优化是其重点。
”——Reviews.com

<<编译器设计>>

编辑推荐

深入剖析现代编译器运用的算法和技术。
强调代码优化和代码生成。
体现编译原理教学的最新理念

版权说明