

本书全面、深入地探讨了编译器设计方面的重要主题,包括词法分析、语法分析、语法制导定义和语法制导翻译、运行时刻环境、目标代码生成、代码优化技术、并行性检测以及过程间分析技术,并在相关章节中给出大量的实例。与上一版相比,本书进行了全面修订,涵盖了编译器开发方面最新进展。每章中都提供了大量的实例及参考文献。

本书是编译原理课程方面的经典教材,内容丰富,适合作为高等院校计算机及相关专业本科生及研究生的编译原理课程的教材,也是广大技术人员的极佳参考读物。

Simplified Chinese edition copyright © 2009 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Compilers: Principles, Techniques and Tools, Second Edition* (ISBN 0-321-48681-1) by Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffery D. Ullman, Copyright © 2007.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison Wesley.

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签,无标签者不得销售。

版权所有,侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号: 图字: 01-2006-6521

图书在版编目(CIP)数据

编译原理 第2版/(美)阿霍(Alfred, V. A.)等著;赵建华等译. —北京:机械工业出版社, 2009. 1

(计算机科学丛书)

书名原文: *Compilers: Principles, Techniques and Tools, Second Edition*

ISBN 978-7-111-25121-7

I. 编… II. ①阿… ②赵… III. 编译程序—程序设计 IV. TP314

中国版本图书馆 CIP 数据核字(2008)第 173435 号

机械工业出版社(北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 朱 劼

北京京北印刷有限公司印刷

2009 年 2 月第 2 版第 2 次印刷

184mm × 260mm · 40.5 印张

标准书号: ISBN 978-7-111-25121-7

定价: 89.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换
本社购书热线:(010) 68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅肇划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson, McGraw-Hill, Elsevier, MIT, John Wiley & Sons, Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum, Bjarne Stroustrup, Brian W. Kernighan, Dennis Ritchie, Jim Gray, Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, Abraham Silberschatz, William Stallings, Donald E. Knuth, John L. Hennessy, Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

华章网站：www.hzbook.com

电子邮件：hzsj@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037



华章教育

译者序

绝大部分软件是使用高级程序设计语言来编写的。用这些语言编写的软件必须经过编译器的编译,才能转换为可以在计算机上运行的机器代码。编译器所生成代码的正确性和质量会直接影响成千上万个软件。因此,编译器构造原理和技术是计算机科学技术领域中的一个非常重要的组成部分。不仅如此,编译技术在当前已经广泛应用于编译器构造之外的其他领域,比如程序分析/验证、模型转换、语言处理等领域。因此,虽然大部分读者不会参与设计商用编译器,但拥有编译的相关知识仍然会对他们的研究开发生涯产生有益的影响。

A. V. Aho 等人撰写的《Compilers: Principles, Techniques, and Tools》被誉为编译教科书中的“龙书”。这说明这本书具有很高的权威性。我们有幸受机械工业出版社的委托,翻译龙书的第2版。

本书不仅包含了词法分析、语法分析、语义分析、代码生成等传统、经典的编译知识,还详细介绍了一些最新研究成果,比如过程间指针分析的最新进展。这使得本书不仅适用于编译原理的初学者,还可以作为研究人员的参考书目。本书不仅介绍编译器构造的基本原理和技术,还详细介绍一些有用的编译器构造工具,比如对 Lex 和 Yacc 的介绍使得读者可以了解这些工具的工作原理和使用方法。除此之外,读者还可以看到很多其他领域的概念在编译器构造中的应用。比如在第9章,读者可以看到群论中的抽象概念“格”被完美地应用于数据流分析算法的设计。而在第11章,线性规划和整数规划技术被成功地应用于程序并行化技术。这些内容对拓展读者的视野和思路有很大的好处。

由于本书覆盖的范围非常广,不可能在一个学期内讲完本书的全部内容。因此我建议采用本书作为本科生教材的老师只选择讲授其中的基础部分,即第1章到第9章中的大部分内容。第2章是对后面各章内容的介绍,可以在讲授相应内容之前指导学生预习。

最后感谢机械工业出版社的温莉芳女士以及姚蕾和朱劼两位编辑在本书的翻译过程中给予我们的有力帮助,也感谢其他给予我们支持的同事。由于水平有限,翻译中的错漏之处在所难免,欢迎读者批评指正。

译者

2008年6月于南京

前言

从本书的 1986 版出版到现在,编译器设计领域已经发生了很大的改变。随着程序设计语言的发展,提出了新的编译问题。计算机体系结构提供了多种多样的资源,而编译器设计者必须能够充分利用这些资源。最有意思的事情可能是,古老的代码优化技术已经在编译器之外找到了新的应用。现在,有些工具利用这些技术来寻找软件中的缺陷,以及(最重要的是)寻找现有代码中的安全漏洞。而且,很多“前端”技术——文法、正则表达式、语法分析器以及语法制导翻译器等——仍然被广泛应用。

因此,本书先前的版本所体现的我们的价值观一直没有改变。我们知道,只有很少的读者将会去构建甚至维护一个主流程序设计语言的编译器。但是,和编译器相关的模型、理论和算法可以被应用到软件设计和开发中出现的各种各样的问题上。因此,我们会关注那些在设计一个语言处理器时常常会碰到的问题,而不考虑具体的源语言和目标机器究竟是什么。

使用本书

要学完本书的全部或大部分内容至少需要两个学季(quarter),甚至两个学期(semester)[⊖]。通常会在在一门本科课程中讲授本书的前半部分内容;而本书的后半部分(强调代码优化)会在研究生层面或另一门小范围的课程中讲授。下面是各章的概要介绍:

- 第 1 章给出一些关于学习动机的资料,同时也将给出一些关于计算机体系结构和程序设计语言原则的背景知识。
- 第 2 章会开发一个小型的编译器,并介绍很多重要概念。这些概念将在后面的各章中深入介绍。这个编译器本身将在附录中给出。
- 第 3 章将讨论词法分析、正则表达式、有穷状态自动机和词法分析器的生成器工具。这些内容是各种文本处理的基础。
- 第 4 章将讨论主流的语法分析方法,包括自顶向下方法(递归下降法、LL 技术)和自底向上方法(LR 技术和它的变体)。
- 第 5 章将介绍语法制导定义和语法制导翻译的基本思想。
- 第 6 章将使用第 5 章中的理论,并说明如何使用这些理论为一个典型的程序设计语言生成中间代码。
- 第 7 章将讨论运行时刻环境,特别是运行时刻栈的管理和垃圾回收机制。
- 第 8 章将主要讨论目标代码生成技术。该章会讨论基本块的构造,从表达式和基本块生成代码的方法,以及寄存器分配技术。
- 第 9 章将介绍代码优化技术,包括流图、数据流分析框架以及求解这些框架的迭代算法。

⊖ 美国大学的学制大致可以分为两种: quarter(学季)和 semester(学期)。前者是把一年分为 4 个 quarter,每个 quarter 3 个月,原则上是上 3 个 quarter 修一个 quarter 的假;而后者则类似我国国内的寒暑假制的大学学制,大概 4 个月一个 semester。——编辑注

- 第 10 章将讨论指令级优化。该章的重点是从小段指令代码中抽取并行性，并在那些可以同时做多件事情的单处理器上调度这些指令。
- 第 11 章将介绍大规模并行性的检测和利用。这里的重点是数值计算代码。这些代码具有对多维数组进行遍历的紧致循环。
- 第 12 章将介绍过程间分析技术。它将讨论指针分析、别名和数据流分析。这些分析都考虑了到达代码中某个给定点时的过程调用序列。

哥伦比亚大学、哈佛大学、斯坦福大学已经开设了讲授本书内容的课程。哥伦比亚大学定期开设一门关于程序设计语言和翻译器的课程，使用了本书前 8 章的内容。该课程常年面向高年级本科生/一年级研究生讲授，这门课程的亮点是一个长达一个学期的课程实践项目。在该项目中，学生分成小组，创建并实现一个他们自己设计的小型语言。学生创建的语言涉及多个应用领域，包括量子计算、音乐合成、计算机图形学、游戏、矩阵运算和很多其他领域。在构建他们自己的编译器时，学生们使用了很多种可以生成编译器组件的工具，比如 ANTLR、Lex 和 Yacc；他们还使用了第 2 章和第 5 章中讨论的语法制导翻译技术。后续的研究生课程的重点是本书第 9 章到第 12 章的内容，着重强调适用于当代计算机（包括网络处理器和多处理器体系结构）的代码生成和优化技术。

斯坦福大学开设了一门历时一个学季的课程，大致涵盖了本书第 1 章到第 8 章的内容，同时还会简介本书第 9 章中全局代码优化的相关内容。第二门编译器课程包括本书第 9 章到第 12 章的内容，另外还包括第 7 章中更为深入的有关垃圾收集的内容。学生使用一个该校开发的、基于 Java 的系统 Joeq 来实现数据流分析算法。

预备知识

学习本书的读者应该拥有一些“计算机科学的综合知识”，至少学过两门程序设计课程，以及数据结构和离散数学的课程。具备多种程序设计语言的知识对学习本书会有所帮助。

练习

本书包含内容广泛的练习，几乎每一节都有一些练习。我们用感叹号来表示较难的练习或练习中的一部分。难度最大的练习有两个感叹号。

万维网上的支持

在本书的主页 (<http://dragonbook.stanford.edu>)[⊖] 上可以找到本书已知错误的勘误表以及一些支持性资料。我们希望将我们讲授的每一门与编译器相关的课程的可用讲义（包括家庭作业、答案和练习等）都提供出来。我们也计划公布由一些重要编译器的作者撰写的关于这些编译器的描述。

致谢

本书封面由 Strange Tonic Productions 的 S. D. Ullman 设计。

Jon Bentley 针对本书的初稿中的多章内容与我们进行了广泛深入的讨论。我们收到了来自下列人员的有帮助的评价和勘误：Domenico Bianculli、Peter Bosch、Marcio Buss、Marc Eaddy、

⊖ 读者可以从 <http://infolab.stanford.edu/~ullman/dragon/errata.html> 找到本书的勘误表，并可以从 <http://infolab.stanford.edu/~ullman/dragon.html> 处找到本书的一些支持资料。

Stephen Edwards、Vibhav Garg、Kim Hazelwood、Gaurav Kc、Wei Li、Mike Smith、Art Stanness、Kryta Svore、Olivier Tardieu 和 Jia Zeng。我们衷心感谢这些人的帮助。当然，书中还可能有错漏之处，希望得到指正和反馈。

另外，Monica 希望能够向她在 SUIF 编译器团队的同事表示感谢，感谢他们在 18 年的时间里给予她的支持和帮助，他们是：Gerald Aigner、Dzintars Avots、Saman Amarasinghe、Jennifer Anderson、Michael Carbin、Gerald Cheong、Amer Diwan、Robert French、Anwar Ghuloum、Mary Hall、John Hennessy、David Heine、Shih-Wei Liao、Amy Lim、Benjamin Livshits、Michael Martin、Dror Maydan、Todd Mowry、Brian Murphy、Jeffrey Oplinger、Karen Pieper、Martin Rinard、Olatunji Ruwase、Constantine Sapuntzakis、Patrick Sathyanathan、Michael Smith、Steven Tjiang、Chau-Wen Tseng、Christopher Unkel、John Whaley、Robert Wilson、Christopher Wilson 和 Michael Wolf。

A. V. A. , Chatham NJ
M. S. L. , Menlo Park CA
R. S. , Far Hills NJ
J. D. U. , Stanford CA
2006 年 6 月

目 录

出版者的话

译者序

前言

第1章 引论 1

1.1 语言处理器 1

1.2 一个编译器的结构 2

1.2.1 词法分析 3

1.2.2 语法分析 4

1.2.3 语义分析 5

1.2.4 中间代码生成 5

1.2.5 代码优化 5

1.2.6 代码生成 6

1.2.7 符号表管理 6

1.2.8 将多个步骤组合成趟 6

1.2.9 编译器构造工具 7

1.3 程序设计语言的发展历程 7

1.3.1 走向高级程序设计语言 7

1.3.2 对编译器的影响 8

1.3.3 1.3节的练习 8

1.4 构建一个编译器的相关科学 8

1.4.1 编译器设计和实现中的建模 9

1.4.2 代码优化的科学 9

1.5 编译技术的应用 10

1.5.1 高级程序设计语言的实现 10

1.5.2 针对计算机体系结构的优化 11

1.5.3 新计算机体系结构的设计 12

1.5.4 程序翻译 13

1.5.5 软件生产率工具 14

1.6 程序设计语言基础 15

1.6.1 静态和动态的区别 15

1.6.2 环境与状态 15

1.6.3 静态作用域和块结构 17

1.6.4 显式访问控制 18

1.6.5 动态作用域 19

1.6.6 参数传递机制 20

1.6.7 别名 21

1.6.8 1.6节的练习 22

1.7 第1章总结 22

1.8 第1章参考文献 23

第2章 一个简单的语法制导翻译器 24

2.1 引言 24

2.2 语法定义 25

2.2.1 文法定义 26

2.2.2 推导 27

2.2.3 语法分析树 28

2.2.4 二义性 29

2.2.5 运算符的结合性 29

2.2.6 运算符的优先级 30

2.2.7 2.2节的练习 31

2.3 语法制导翻译 32

2.3.1 后缀表示 33

2.3.2 综合属性 33

2.3.3 简单语法制导定义 35

2.3.4 树的遍历 35

2.3.5 翻译方案 35

2.3.6 2.3节的练习 37

2.4 语法分析 37

2.4.1 自顶向下分析方法 38

2.4.2 预测分析法 39

2.4.3 何时使用 ϵ 产生式 41

2.4.4 设计一个预测分析器 41

2.4.5 左递归 42

2.4.6 2.4节的练习 42

2.5 简单表达式的翻译器 43

2.5.1 抽象语法和具体语法 43

2.5.2 调整翻译方案 43

2.5.3 非终结符号的过程 44

2.5.4 翻译器的简化 45

2.5.5 完整的程序 46

2.6 词法分析	47	3.5 词法分析器生成工具 Lex	89
2.6.1 剔除空白和注释	48	3.5.1 Lex 的使用	89
2.6.2 预读	48	3.5.2 Lex 程序的结构	89
2.6.3 常量	49	3.5.3 Lex 中的冲突解决	91
2.6.4 识别关键字和标识符	49	3.5.4 向前看运算符	92
2.6.5 词法分析器	50	3.5.5 3.5 节的练习	92
2.6.6 2.6 节的练习	53	3.6 有穷自动机	93
2.7 符号表	53	3.6.1 不确定的有穷自动机	93
2.7.1 为每个作用域设置一个符号表	54	3.6.2 转换表	94
2.7.2 符号表的使用	56	3.6.3 自动机中输入字符串的接受	94
2.8 生成中间代码	57	3.6.4 确定的有穷自动机	95
2.8.1 两种中间表示形式	57	3.6.5 3.6 节的练习	96
2.8.2 语法树的构造	58	3.7 从正则表达式到自动机	96
2.8.3 静态检查	61	3.7.1 从 NFA 到 DFA 的转换	96
2.8.4 三地址码	62	3.7.2 NFA 的模拟	99
2.8.5 2.8 节的练习	66	3.7.3 NFA 模拟的效率	99
2.9 第 2 章总结	66	3.7.4 从正则表达式构造 NFA	100
第 3 章 词法分析	68	3.7.5 字符串处理算法的效率	103
3.1 词法分析器的作用	68	3.7.6 3.7 节的练习	105
3.1.1 词法分析及语法分析	69	3.8 词法分析器生成工具的设计	105
3.1.2 词法单元、模式和词素	69	3.8.1 生成的词法分析器的结构	105
3.1.3 词法单元的属性	70	3.8.2 基于 NFA 的模式匹配	106
3.1.4 词法错误	71	3.8.3 词法分析器使用的 DFA	107
3.1.5 3.1 节的练习	71	3.8.4 实现向前看运算符	108
3.2 输入缓冲	71	3.8.5 3.8 节的练习	109
3.2.1 缓冲区对	72	3.9 基于 DFA 的模式匹配器的优化	109
3.2.2 哨兵标记	72	3.9.1 NFA 的重要状态	109
3.3 词法单元的规约	73	3.9.2 根据抽象语法树计算得到的 函数	110
3.3.1 串和语言	74	3.9.3 计算 nullable、firstpos 及 lastpos	111
3.3.2 语言上的运算	75	3.9.4 计算 followpos	112
3.3.3 正则表达式	75	3.9.5 根据正则表达式构建 DFA	113
3.3.4 正则定义	77	3.9.6 最小化一个 DFA 的状态数	114
3.3.5 正则表达式的扩展	78	3.9.7 词法分析器的状态最小化	116
3.3.6 3.3 节的练习	78	3.9.8 DFA 模拟中的时间和空间权衡	116
3.4 词法单元的识别	80	3.9.9 3.9 节的练习	117
3.4.1 状态转换图	82	3.10 第 3 章总结	118
3.4.2 保留字和标识符的识别	83	3.11 第 3 章参考文献	119
3.4.3 完成我们的例子	84	第 4 章 语法分析	121
3.4.4 基于状态转换图的词法分析器的 体系结构	84	4.1 引论	121
3.4.5 3.4 节的练习	86		

4.1.1 语法分析器的作用	121	4.6.5 可行前缀	163
4.1.2 代表性的文法	122	4.6.6 4.6 节的练习	164
4.1.3 语法错误的处理	123	4.7 更强大的 LR 语法分析器	165
4.1.4 错误恢复策略	123	4.7.1 规范 LR(1) 项	165
4.2 上下文无关文法	124	4.7.2 构造 LR(1) 项集	166
4.2.1 上下文无关文法的正式定义	125	4.7.3 规范 LR(1) 语法分析表	169
4.2.2 符号表示的约定	125	4.7.4 构造 LALR 语法分析表	170
4.2.3 推导	126	4.7.5 高效构造 LALR 语法分析表 的方法	173
4.2.4 语法分析树和推导	127	4.7.6 LR 语法分析表的压缩	176
4.2.5 二义性	128	4.7.7 4.7 节的练习	177
4.2.6 验证文法生成的语言	129	4.8 使用二义性文法	178
4.2.7 上下文无关文法和正则 表达式	130	4.8.1 用优先级和结合性解决冲突	178
4.2.8 4.2 节的练习	130	4.8.2 “悬空-else” 的二义性	179
4.3 设计文法	132	4.8.3 LR 语法分析中的错误恢复	181
4.3.1 词法分析和语法分析	132	4.8.4 4.8 节的练习	182
4.3.2 消除二义性	133	4.9 语法分析器生成工具	183
4.3.3 左递归的消除	134	4.9.1 语法分析器生成工具 Yacc	183
4.3.4 提取左公因子	135	4.9.2 使用带有二义性文法的 Yacc 规约	185
4.3.5 非上下文无关语言的构造	136	4.9.3 用 Lex 创建 Yacc 的词法 分析器	187
4.3.6 4.3 节的练习	137	4.9.4 Yacc 中的错误恢复	188
4.4 自顶向下的语法分析	137	4.9.5 4.9 节的练习	189
4.4.1 递归下降的语法分析	139	4.10 第 4 章总结	189
4.4.2 FIRST 和 FOLLOW	140	4.11 第 4 章参考文献	191
4.4.3 LL(1) 文法	141	第 5 章 语法制导的翻译	194
4.4.4 非递归的预测分析	144	5.1 语法制导定义	194
4.4.5 预测分析中的错误恢复	145	5.1.1 继承属性和综合属性	195
4.4.6 4.4 节的练习	147	5.1.2 在语法分析树的结点上对 SDD 求值	196
4.5 自底向上的语法分析	148	5.1.3 5.1 节的练习	198
4.5.1 归约	149	5.2 SDD 的求值顺序	198
4.5.2 句柄剪枝	149	5.2.1 依赖图	198
4.5.3 移入-归约语法分析技术	150	5.2.2 属性求值的顺序	199
4.5.4 移入-归约语法分析中的 冲突	151	5.2.3 S 属性的定义	200
4.5.5 4.5 节的练习	152	5.2.4 L 属性的定义	200
4.6 LR 语法分析技术介绍: 简单 LR 技术	153	5.2.5 具有受控副作用的语义规则	201
4.6.1 为什么使用 LR 语法分析器	153	5.2.6 5.2 节的练习	202
4.6.2 项和 LR(0) 自动机	154	5.3 语法制导翻译的应用	203
4.6.3 LR 语法分析算法	158	5.3.1 抽象语法树的构造	203
4.6.4 构造 SLR 语法分析表	161		

5.3.2 类型的结构	206	6.4.1 表达式中的运算	243
5.3.3 5.3 节的练习	207	6.4.2 增量翻译	244
5.4 语法制导的翻译方案	207	6.4.3 数组元素的寻址	245
5.4.1 后缀翻译方案	207	6.4.4 数组引用的翻译	246
5.4.2 后缀 SDT 的语法分析栈实现	208	6.4.5 6.4 节的练习	247
5.4.3 产生式内部带有语义动作 的 SDT	209	6.5 类型检查	248
5.4.4 从 SDT 中消除左递归	210	6.5.1 类型检查规则	248
5.4.5 L 属性定义的 SDT	212	6.5.2 类型转换	249
5.4.6 5.4 节的练习	216	6.5.3 函数和运算符的重载	250
5.5 实现 L 属性的 SDD	216	6.5.4 类型推导和多态函数	251
5.5.1 在递归下降语法分析过程中 进行翻译	217	6.5.5 一个合一算法	254
5.5.2 边扫描边生成代码	219	6.5.6 6.5 节的练习	256
5.5.3 L 属性的 SDD 和 LL 语法 分析	220	6.6 控制流	256
5.5.4 L 属性的 SDD 的自底向上语法 分析	224	6.6.1 布尔表达式	257
5.5.5 5.5 节的练习	226	6.6.2 短路代码	257
5.6 第 5 章总结	227	6.6.3 控制流语句	257
5.7 第 5 章参考文献	228	6.6.4 布尔表达式的控制流翻译	259
第 6 章 中间代码生成	229	6.6.5 避免生成冗余的 goto 指令	261
6.1 语法树的变体	230	6.6.6 布尔值和跳转代码	262
6.1.1 表达式的有向无环图	230	6.6.7 6.6 节的练习	263
6.1.2 构造 DAG 的值编码方法	231	6.7 回填	263
6.1.3 6.1 节的练习	232	6.7.1 使用回填技术的一趟式目标 代码生成	263
6.2 三地址代码	233	6.7.2 布尔表达式的回填	264
6.2.1 地址和指令	233	6.7.3 控制转移语句	266
6.2.2 四元式表示	235	6.7.4 break 语句、continue 语句和 goto 语句	267
6.2.3 三元式表示	235	6.7.5 6.7 节的练习	268
6.2.4 静态单赋值形式	237	6.8 switch 语句	269
6.2.5 6.2 节的练习	237	6.8.1 switch 语句的翻译	269
6.3 类型和声明	237	6.8.2 switch 语句的语法制导翻译	270
6.3.1 类型表达式	238	6.8.3 6.8 节的练习	271
6.3.2 类型等价	239	6.9 过程的中间代码	271
6.3.3 声明	239	6.10 第 6 章总结	272
6.3.4 局部变量名的存储布局	239	6.11 第 6 章参考文献	273
6.3.5 声明的序列	241	第 7 章 运行时刻环境	275
6.3.6 记录和类中的字段	242	7.1 存储组织	275
6.3.7 6.3 节的练习	242	7.2 空间的栈式分配	276
6.4 表达式的翻译	243	7.2.1 活动树	277
		7.2.2 活动记录	279
		7.2.3 调用代码序列	280

7.2.4 栈中的变长数据	282	7.8.1 并行和并发垃圾回收	319
7.2.5 7.2 节的练习	283	7.8.2 部分对象重新定位	321
7.3 栈中非局部数据的访问	284	7.8.3 类型不安全的语言的保守垃圾回收	321
7.3.1 没有嵌套过程时的数据访问	284	7.8.4 弱引用	322
7.3.2 和嵌套过程相关的问题	284	7.8.5 7.8 节的练习	322
7.3.3 一个支持嵌套过程声明的语言	285	7.9 第7章总结	322
7.3.4 嵌套深度	285	7.10 第7章参考文献	324
7.3.5 访问链	286	第8章 代码生成	326
7.3.6 处理访问链	287	8.1 代码生成器设计中的问题	327
7.3.7 过程型参数的访问链	288	8.1.1 代码生成器的输入	327
7.3.8 显示表	289	8.1.2 目标程序	327
7.3.9 7.3 节的练习	290	8.1.3 指令选择	328
7.4 堆管理	291	8.1.4 寄存器分配	329
7.4.1 存储管理器	291	8.1.5 求值顺序	330
7.4.2 一台计算机的存储层次结构	292	8.2 目标语言	330
7.4.3 程序中的局部性	293	8.2.1 一个简单的目标机模型	330
7.4.4 碎片整理	295	8.2.2 程序和指令的代价	332
7.4.5 人工回收请求	297	8.2.3 8.2 节的练习	332
7.4.6 7.4 节的练习	299	8.3 目标代码中的地址	334
7.5 垃圾回收概述	299	8.3.1 静态分配	334
7.5.1 垃圾回收器的设计目标	299	8.3.2 栈分配	335
7.5.2 可达性	301	8.3.3 名字的运行时刻地址	337
7.5.3 引用计数垃圾回收器	302	8.3.4 8.3 节的练习	337
7.5.4 7.5 节的练习	303	8.4 基本块和流图	338
7.6 基于跟踪的回收的介绍	304	8.4.1 基本块	339
7.6.1 基本的标记-清扫式回收器	304	8.4.2 后续使用信息	340
7.6.2 基本抽象	305	8.4.3 流图	340
7.6.3 标记-清扫式算法的优化	306	8.4.4 流图的表示方式	341
7.6.4 标记并压缩的垃圾回收器	307	8.4.5 循环	341
7.6.5 拷贝回收器	309	8.4.6 8.4 节的练习	342
7.6.6 开销的比较	311	8.5 基本块的优化	342
7.6.7 7.6 节的练习	311	8.5.1 基本块的 DAG 表示	342
7.7 短停顿垃圾回收	311	8.5.2 寻找局部公共子表达式	343
7.7.1 增量式垃圾回收	312	8.5.3 消除死代码	344
7.7.2 增量式可达性分析	313	8.5.4 代数恒等式的使用	344
7.7.3 部分回收概述	314	8.5.5 数组引用的表示	345
7.7.4 世代垃圾回收	315	8.5.6 指针赋值和过程调用	346
7.7.5 列车算法	316	8.5.7 从 DAG 到基本块的重组	347
7.7.6 7.7 节的练习	318	8.5.8 8.5 节的练习	348
7.8 垃圾回收中的高级论题	319	8.6 一个简单的代码生成器	348

8.6.1 寄存器和地址描述符	349	9.1.1 冗余的原因	374
8.6.2 代码生成算法	349	9.1.2 一个贯穿本章的例子:快速 排序	375
8.6.3 函数 getReg 的设计	352	9.1.3 保持语义不变的转换	377
8.6.4 8.6 节的练习	352	9.1.4 全局公共子表达式	377
8.7 窥孔优化	353	9.1.5 复制传播	378
8.7.1 消除冗余的加载和保存指令	353	9.1.6 死代码消除	379
8.7.2 消除不可达代码	354	9.1.7 代码移动	379
8.7.3 控制流优化	354	9.1.8 归纳变量和强度消减	380
8.7.4 代数化简和强度消减	355	9.1.9 9.1 节的练习	381
8.7.5 使用机器特有的指令	355	9.2 数据流分析简介	382
8.7.6 8.7 节的练习	355	9.2.1 数据流抽象	382
8.8 寄存器分配和指派	355	9.2.2 数据流分析模式	383
8.8.1 全局寄存器分配	356	9.2.3 基本块上的数据流模式	384
8.8.2 使用计数	356	9.2.4 到达定值	385
8.8.3 外层循环的寄存器指派	358	9.2.5 活跃变量分析	390
8.8.4 通过图着色方法进行寄存器 分配	358	9.2.6 可用表达式	391
8.8.5 8.8 节的练习	359	9.2.7 小结	393
8.9 通过树重写来选择指令	359	9.2.8 9.2 节的练习	394
8.9.1 树翻译方案	359	9.3 数据流分析基础	395
8.9.2 通过覆盖一个输入树来生成 代码	361	9.3.1 半格	396
8.9.3 通过扫描进行模式匹配	362	9.3.2 传递函数	399
8.9.4 用于语义检查的例程	363	9.3.3 通用框架的迭代算法	400
8.9.5 通用的树匹配方法	363	9.3.4 数据流解的含义	402
8.9.6 8.9 节的练习	364	9.3.5 9.3 节的练习	404
8.10 表达式的优化代码的生成	365	9.4 常量传播	404
8.10.1 Ershov 数	365	9.4.1 常量传播框架的数据流值	404
8.10.2 从带标号的表达式树生成 代码	365	9.4.2 常量传播框架的交汇运算	405
8.10.3 寄存器数量不足时的表达式 求值	366	9.4.3 常量传播框架的传递函数	405
8.10.4 8.10 节的练习	368	9.4.4 常量传递框架的单调性	406
8.11 使用动态规划的代码生成	368	9.4.5 常量传播框架的不可分配性	406
8.11.1 连续求值	368	9.4.6 对算法结果的解释	407
8.11.2 动态规划的算法	369	9.4.7 9.4 节的练习	408
8.11.3 8.11 节的练习	371	9.5 部分冗余消除	408
8.12 第8章总结	371	9.5.1 冗余的来源	408
8.13 第8章参考文献	372	9.5.2 可能消除所有冗余吗	410
第9章 机器无关优化	374	9.5.3 懒惰代码移动问题	411
9.1 优化的主要来源	374	9.5.4 表达式的预期执行	412
		9.5.5 懒惰代码移动算法	413
		9.5.6 9.5 节的练习	418
		9.6 流图中的循环	419

9.6.1 支配结点	419	10.3.1 数据依赖图	459
9.6.2 深度优先排序	421	10.3.2 基本块的列表调度方法	460
9.6.3 深度优先生成树中的边	423	10.3.3 带优先级的拓扑排序	461
9.6.4 回边和可归约性	423	10.3.4 10.3 节的练习	461
9.6.5 流图的深度	424	10.4 全局代码调度	462
9.6.6 自然循环	424	10.4.1 基本的代码移动	462
9.6.7 迭代数据流算法的收敛速度	425	10.4.2 向上的代码移动	463
9.6.8 9.6 节的练习	427	10.4.3 向下的代码移动	464
9.7 基于区域的分析	428	10.4.4 更新数据依赖关系	465
9.7.1 区域	429	10.4.5 全局调度算法	465
9.7.2 可归约流图的区域层次结构	429	10.4.6 高级代码移动技术	467
9.7.3 基于区域的分析技术概述	431	10.4.7 和动态调度器的交互	468
9.7.4 有关传递函数的必要假设	431	10.4.8 10.4 节的练习	468
9.7.5 一个基于区域的分析算法	433	10.5 软件流水线化	468
9.7.6 处理不可归约流图	436	10.5.1 引言	468
9.7.7 9.7 节的练习	437	10.5.2 循环的软件流水线化	470
9.8 符号分析	437	10.5.3 寄存器分配和代码生成	471
9.8.1 参考变量的仿射表达式	438	10.5.4 Do-Across 循环	472
9.8.2 数据流问题的公式化	440	10.5.5 软件流水线化的目标和约束	472
9.8.3 基于区域的符号化分析	442	10.5.6 一个软件流水线化算法	474
9.8.4 9.8 节的练习	445	10.5.7 对无环数据依赖图进行调度	475
9.9 第9章总结	445	10.5.8 对有环数据依赖图进行调度	476
9.10 第9章参考文献	448	10.5.9 对流水线化算法的改进	480
第10章 指令级并行性	450	10.5.10 模数变量扩展	480
10.1 处理器体系结构	450	10.5.11 条件语句	482
10.1.1 指令流水线和分支延时	451	10.5.12 软件流水线化的硬件支持	483
10.1.2 流水线执行	451	10.5.13 10.5 节的练习	483
10.1.3 多指令发送	451	10.6 第10章总结	484
10.2 代码调度约束	452	10.7 第10章参考文献	485
10.2.1 数据依赖	452	第11章 并行性和局部性优化	487
10.2.2 寻找内存访问之间的依赖关系	453	11.1 基本概念	488
10.2.3 寄存器使用和并行性之间的折衷	454	11.1.1 多处理器	488
10.2.4 寄存器分配阶段和代码调度阶段之间的顺序	455	11.1.2 应用中的并行性	490
10.2.5 控制依赖	455	11.1.3 循环层次上的并行性	491
10.2.6 对投机执行的支持	456	11.1.4 数据局部性	492
10.2.7 一个基本的机器模型	457	11.1.5 仿射变换理论概述	493
10.2.8 10.2 节的练习	458	11.2 矩阵乘法：一个深入的例子	495
10.3 基本块调度	459	11.2.1 矩阵相乘算法	495
		11.2.2 优化	497
		11.2.3 高速缓存干扰	499
		11.2.4 11.2 节的练习	499

11.3 迭代空间	499	11.7.9 11.7 节的练习	539
11.3.1 从循环嵌套结构中构建迭代空间	499	11.8 并行循环之间的同步	541
11.3.2 循环嵌套结构的执行顺序	501	11.8.1 固定多个同步运算	541
11.3.3 不等式组的矩阵表示方法	501	11.8.2 程序依赖图	542
11.3.4 混合使用符号常量	502	11.8.3 层次结构化的时间	543
11.3.5 控制执行的顺序	502	11.8.4 并行化算法	544
11.3.6 坐标轴的变换	505	11.8.5 11.8 节的练习	545
11.3.7 11.3 节的练习	506	11.9 流水线化技术	545
11.4 仿射的数组下标	507	11.9.1 什么是流水线化	545
11.4.1 仿射访问	507	11.9.2 连续过松弛方法: 一个例子	546
11.4.2 实践中的仿射访问和非仿射访问	508	11.9.3 完全可交换循环	547
11.4.3 11.4 节的练习	508	11.9.4 把完全可交换循环流水线化	548
11.5 数据复用	509	11.9.5 一般性的理论	549
11.5.1 数据复用的类型	509	11.9.6 时间分划约束	549
11.5.2 自复用	510	11.9.7 用 Farkas 引理解时间分划约束	552
11.5.3 自空间复用	513	11.9.8 代码转换	554
11.5.4 组复用	514	11.9.9 具有最小同步量的并行性	557
11.5.5 11.5 节的练习	515	11.9.10 11.9 节的练习	559
11.6 数组数据依赖关系分析	516	11.10 局部性优化	560
11.6.1 数组访问的数据依赖关系的定义	517	11.10.1 计算结果数据的时间局部性	560
11.6.2 整数线性规划	518	11.10.2 数组收缩	560
11.6.3 GCD 测试	518	11.10.3 分划单元的交织	562
11.6.4 解决整数线性规划的启发式规则	520	11.10.4 合成	565
11.6.5 解决一般性的整数线性规划问题	522	11.10.5 11.10 节的练习	566
11.6.6 小结	523	11.11 仿射转换的其他用途	566
11.6.7 11.6 节的练习	523	11.11.1 分布式内存计算机	566
11.7 寻找无同步的并行性	524	11.11.2 多指令发送处理器	567
11.7.1 一个介绍性的例子	525	11.11.3 向量和 SIMD 指令	567
11.7.2 仿射空间分划	526	11.11.4 数据预取	567
11.7.3 空间分划约束	527	11.12 第 11 章总结	568
11.7.4 求解空间分划约束	529	11.13 第 11 章参考文献	570
11.7.5 一个简单的代码生成算法	531	第 12 章 过程间分析	573
11.7.6 消除空迭代	533	12.1 基本概念	573
11.7.7 从最内层循环中消除条件测试	535	12.1.1 调用图	573
11.7.8 源代码转换	537	12.1.2 上下文相关	574
		12.1.3 调用串	576
		12.1.4 基于克隆的上下文相关分析	577
		12.1.5 基于摘要的上下文相关分析	578
		12.1.6 12.1 节的练习	579

12.2 为什么需要过程间分析	580	12.5.1 一个方法调用的效果	595
12.2.1 虚方法调用	580	12.5.2 在 Datalog 中发现调用图	596
12.2.2 指针别名分析	581	12.5.3 动态加载和反射	597
12.2.3 并行化	581	12.5.4 12.5 节的练习	597
12.2.4 软件错误和漏洞的检测	581	12.6 上下文相关指针分析	598
12.2.5 SQL 注入	581	12.6.1 上下文和调用串	598
12.2.6 缓冲区溢出	582	12.6.2 在 Datalog 规则中加入上下文 信息	600
12.3 数据流的一种逻辑表示方式	583	12.6.3 关于相关性的更多讨论	600
12.3.1 Datalog 简介	583	12.6.4 12.6 节的练习	600
12.3.2 Datalog 规则	584	12.7 使用 BDD 的 Datalog 的实现	601
12.3.3 内涵断言和外延断言	585	12.7.1 二分决策图	601
12.3.4 Datalog 程序的执行	587	12.7.2 对 BDD 的转换	602
12.3.5 Datalog 程序的增量计算	588	12.7.3 用 BDD 表示关系	603
12.3.6 有问题的 Datalog 规则	589	12.7.4 用 BDD 操作实现关系运算	603
12.3.7 12.3 节的练习	590	12.7.5 在指针指向分析中使用 BDD	605
12.4 一个简单的指针分析算法	591	12.7.6 12.7 节的练习	605
12.4.1 为什么指针分析有难度	591	12.8 第 12 章总结	606
12.4.2 一个指针和引用的模型	592	12.9 第 12 章参考文献	607
12.4.3 控制流无关性	592	附录 A 一个完整的编译器前端	611
12.4.4 在 Datalog 中的表示方法	593	附录 B 寻找线性独立解	630
12.4.5 使用类型信息	594		
12.4.6 12.4 节的练习	595		
12.5 上下文无关的过程间分析	595		