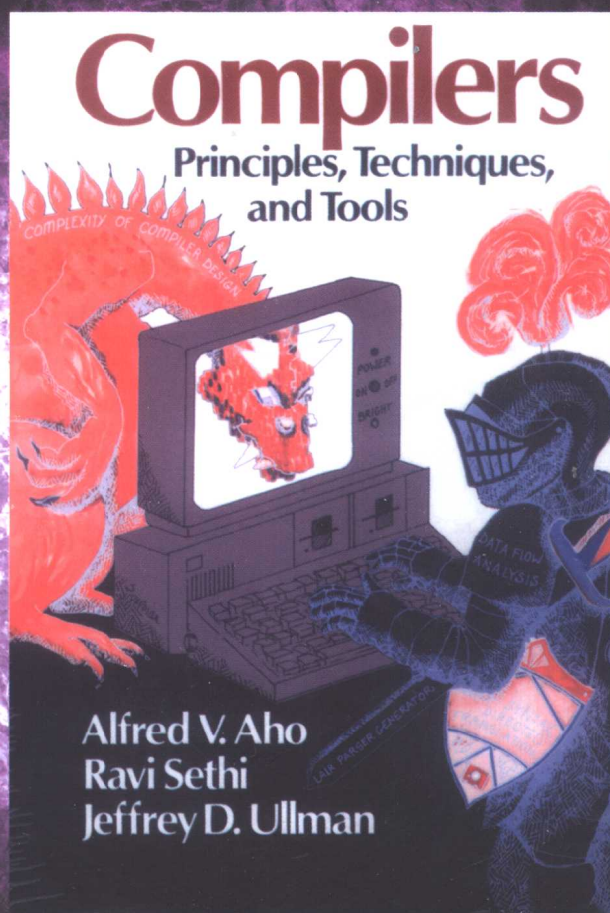


计 算 机 科 学 丛 书

编译原理

(美) Alfred V. Aho Ravi Sethi Jeffrey D. Ullman 著 李建中 姜守旭 译
贝尔实验室 Avaya实验室 斯坦福大学



Compilers
Principles, Techniques, and Tools



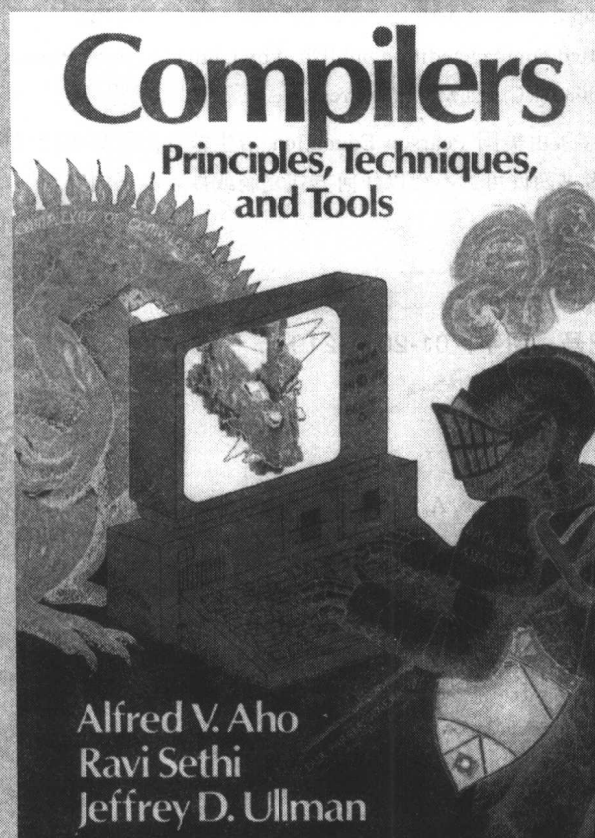
机械工业出版社
China Machine Press



计 算 机 科 学 丛 书

编译原理

(美) Alfred V. Aho Ravi Sethi Jeffrey D. Ullman 著 李建中 姜守旭 译
贝尔实验室 Avaya实验室 斯坦福大学



Compilers Principles, Techniques, and Tools



机械工业出版社
China Machine Press



0767737

~41

057
10
02

本书深入讨论了编译器设计的重要主题,包括词法分析、语法分析、语法制导分析、类型检查、运行环境、中间代码生成、代码生成、代码优化等,并在最后两章中讨论了实现编译器的一些编程问题和几个编译器实例,每章都提供了大量的练习和参考文献。本书从介绍编译的原理性概念开始,然后通过构建一个简单的遍编译器来逐一解释这些概念。

本书是编译原理课程的经典教材,作者曾多次使用本书的内容在贝尔实验室、哥伦比亚大学、普林斯顿大学和斯坦福大学向本科生和研究生讲授初等及高等编译课程。

Authorized translation from the English language edition entitled *Compilers: Principles, Techniques, and Tools* by Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, published by Pearson Education, Inc, publishing as Addison-Wesley, Copyright © 1986 by Bell Telephone Laboratories, Incorporated.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanic, including photocopying, recording, or by any information storage retrieval system, without permission of Pearson Education, Inc.

Chinese simplified language edition published by China Machine Press.

Copyright © 2003 by China Machine Press.

本书中文简体字版由美国 Pearson Education 培生教育出版集团授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号:图字:01-2001-2194

图书在版编目(CIP)数据

编译原理/(美)阿霍(Aho, A. V.)等著;李建中等译. —北京:机械工业出版社, 2003.8
(计算机科学丛书)

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

ISBN 7-111-12349-2

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

中国版本图书馆CIP数据核字(2003)第050118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:杨海玲

北京中加印刷有限公司印刷·新华书店北京发行所发行

2003年8月第1版第1次印刷

787mm × 1092mm 1/16 · 34印张

印数: 0 001-5 000册

定价: 55.00元

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

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中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，不仅擘划了研究的范畴，还揭橥了学术的源变，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、从业人员较少的现状下，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图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall, Addison-Wesley, McGraw-Hill, Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum, Stroustrup, Kernighan, Jim Gray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、研究及收藏。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，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，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，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步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步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总体规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业

的教学度身订造的。其中许多教材均已为M. I. T., Stanford, U.C. Berkeley, C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程,而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下,读者必将在计算机科学的宫殿中由登堂而入室。

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,但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。教材的出版只是我们的后续服务的起点。华章公司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:

电子邮件: hzedu@hzbook.com

联系电话: (010) 68995264

联系地址: 北京市西城区百万庄南街1号

邮政编码: 100037

专家指导委员会

(按姓氏笔画顺序)

尤晋元
石教英
张立昂
邵维忠
周克定
郑国梁
高传善
裘宗燕

王 珊
吕 建
李伟琴
陆丽娜
周傲英
施伯乐
梅 宏
戴 葵

冯博琴
孙玉芳
李师贤
陆鑫达
孟小峰
钟玉琢
程 旭

史忠植
吴世忠
李建中
陈向群
岳丽华
唐世渭
程时端

史美林
吴时霖
杨冬青
周伯生
范 明
袁崇义
谢希仁

译者序

编译器产生于20世纪60年代,在计算机科学技术的发展历史中发挥了巨大作用,是开发计算机应用系统不可缺少的重要工具。编译器的原理和技术具有十分普遍的意义。在每一个计算机科学技术工作者的职业生涯中,这些原理和技术都被反复用到。编译器的编写涉及到程序设计语言、计算机体系结构、语言理论、算法和软件工程等学科,是计算机科学技术的重要基础。作为计算机科学技术学科的专业基础课,编译器原理和技术是计算机科学技术专业学生的必修课程。本书是一部优秀的编译器原理和技术教材。

本书是 Alfred V. Aho 和 Jeffrey D. Ullman 所著的《*Principles of Compiler Design*》一书的后裔版。本书作者 Alfred V. Aho 是 AT&T 贝尔实验室计算机原理研究部负责人, Jeffrey D. Ullman 是斯坦福大学计算机科学系教授, Ravi Sethi 是 AT&T 贝尔实验室研究人员。Alfred V. Aho 和 Jeffrey D. Ullman 是世界著名的计算机科学家,他们在计算机科学理论、数据库等很多领域都做出了杰出贡献。他们的很多著作都被国际公认为是权威性著作,深受读者的喜爱。本书的英文版出版于20世纪80年代,是一部著名的编译器原理与技术方面的教材,一直被国际著名高等院校特别是美国著名大学作为编译器原理与技术的教科书。这部著作对我国计算机界也具有重大影响。机械出版社独具慧眼,决定将这部著作翻译成中文在国内出版,这必将对我国计算机科学技术的编译原理教学工作产生积极的推动作用。有幸承担该书的翻译工作,我们感到十分荣幸。

本书是编译器原理与技术的基本教程,旨在介绍编译器的一般原理,解决人们在编译器设计中遇到的普遍问题。本书在系统地介绍编译器的一般原理的同时,特别注重编译原理和技术的实际应用,给出了许多启示,并配置了大量的例题和习题。本书的内容适用于所有源语言和目标机器。本书介绍的概念和技术不仅适用于编译器的设计,也适用于一般的软件设计。显然,本书在目前只有少数人涉及编译器的构造和维护的情况下仍然具有重要的意义和价值。

本书共有十二章和一个附录。第1章介绍编译器的基本结构;第2章描述了一个变中缀表达式为后缀表达式的翻译器;第3章介绍了词法分析器、正规表达式、有穷自动机以及词法分析器的自动生成工具;第4章详述常用的语法分析技术;第5章阐述了语法制导翻译的基本概念;第6章介绍了实现静态语义检查的基本思想;第7章讨论了程序运行环境的存储组织问题;第8章首先介绍了中间语言的概念,然后讨论如何把一般的程序设计语言翻译成中间代码的问题;第9章介绍目标代码生成技术和代码生成器的自动生成方法;第10章全面介绍了代码优化方法;第11章讨论了实现编译器的一些编程问题;第12章提供了几个编译器实例;附录A描述了一种简单的语言,学生可以把它作为源语言,构造一个编译器。

本书可以作为高等院校计算机专业本科生和研究生编译原理与技术课程的教材,也可以作为计算机软件工作者的技术参考书。

限于译者水平,译文中疏漏和错误难免,欢迎批评指正。

李建中, 姜守旭

2003年7月1日

译者简介



李建中，哈尔滨工业大学教授，博士生导师，国家杰出青年基金获得者，中国计算机学会理事，中国计算机学会数据库专业委员会副主任。从事计算机科学技术的教学、研究、开发工作二十余年。主要研究领域为数据库系统与并行计算，主持完成研究项目20余项，在统计与科学数据库、并行数据库、数据仓库、数据挖掘等方面取得了一系列研究成果，在IEEE Transactions on Knowledge and Data Engineering、VLDB、ACM SIGMOD等国内外重要学术刊物和学术会议发表学术论文180余篇，出版学术专著和教材4部，获得各类科学技术奖励多项。



姜守旭，哈尔滨工业大学副教授，硕士生导师。1990年毕业于哈尔滨工业大学计算机及应用专业并留校任教，1995年获得计算机软件硕士学位。多年来一直从事计算机科学技术的教学与科研工作，主要研究方向为操作系统与对等计算。曾获部科技进步三等奖一项，出版教材一部。

前言

本书是 Alfred V.Aho和Jeffrey D.Ullman 所著的《*Principles of Compiler Design*》一书的后续版本。与后者类似，本书也是编译器设计基础课程的教材。本书的重点是解决人们在设计语言翻译器时遇到的普遍问题，而不论源和目标机器是什么。

本书介绍的概念和技术不仅适用于编译器的设计，也适用于一般的软件设计。例如，建立词法分析器的串匹配技术已用于文本编辑器、信息检索系统和模式识别器；上下文无关文法和语法制导定义等概念已用于设计许多诸如本书产生的排版、绘图系统这样的小语言；代码优化技术已用于程序验证器和从非结构化程序产生结构化程序的程序检验器之中。显然，本书在目前只有少数人涉及编译器的构造和维护的情况下仍然具有重要的意义和价值。

本书的使用方法

本书深入地讨论了编译器设计的重要主题。

第1章介绍编译器的基本结构，是阅读本书其余部分的基础。

第2章描述了一个变中缀表达式为后缀表达式的翻译器。这个翻译器使用本书介绍的一些基本技术构建。后面的一些章节逐渐地扩展了第2章介绍的内容。

第3章介绍了词法分析器、正规表达式、有穷状态机以及词法分析器的生成器工具。本章的内容已经被广泛地用于文本处理。

第4章深入介绍了常用的语法分析技术。本书讨论的语法分析技术比较广泛，从适用于手工实现的递归下降技术到用于语法分析器生成器的计算更密集的LR技术。

第5章介绍了语法制导翻译的主要概念。本章的内容将被用于本书中说明和实现翻译的其余各章。

第6章介绍了实现静态语义检查的主要思想，详尽讨论了类型检查与合一问题。

第7章讨论了用于支持程序运行环境的存储组织问题。

第8章首先介绍了中间语言的概念，然后讨论如何把一般的程序设计语言结构翻译成中间代码的问题。

第9章介绍目标代码生成技术，包括简单的代码生成方法以及产生表达式代码的优化方法。本章也讨论了窥孔优化方法和代码生成器的生成器。

第10章全面介绍了代码优化，详细讨论了各种数据流分析方法和几种主要的全局优化方法。

第11章讨论实现编译器的一些编程问题。软件工程和软件测试在构造编译器的过程中是非常重要的。

第12章提供几个编译器实例。这些编译器都使用了本书介绍的技术。

附录A描述了一种简单的语言。这种语言是Pascal语言的“子集”，它可以用做实现项目的基础。

本书作者曾使用本书的内容多次在贝尔实验室、哥伦比亚大学、普林斯顿大学和斯坦福

大学为本科生和研究生讲授初等和高等编译课程。

初等编译课程可以由本书以下章节构成：

简介	第1章、2.1~2.5节
词法分析	2.6节、3.1~3.4节
符号表	2.7节、7.6节
语法分析	2.4节、4.1~4.4节
语法制导翻译	2.5节、5.1~5.5节
类型检查	6.1~6.2节
运行环境的组织	7.1~7.3节
中间代码的生成	8.1~8.3节
代码生成	9.1~9.4节
代码优化	10.1~10.2节

第2章描述了编程项目所需的信息。

以编译器构造工具为核心的课程可以包括3.5节的词法分析器的生成器、4.8节和4.9节的语法分析器的生成器、9.12节的代码生成器的生成器以及第11章中有关编译器构造技术的内容。

高等编译课程可以由本书的以下内容组成：第3章和第4章介绍的词法分析器的生成器和语法分析器的生成器中使用的算法、第6章介绍的有关类型等价、重载、多态和合一的内容、第7章介绍的程序运行环境的存储组织、第9章介绍的模式制导的代码生成方法、第10章介绍的代码优化。

习题

我们用星号来标记习题的难度。没有星号的习题检验学生对基本定义的理解；具有一个星号的习题用于高等编译课程；具有两个星号的习题是值得深思熟虑的难题。

致谢

在本书的编写过程中，很多人都对手稿给予了有价值的建议。在此，我们对以下人员表示衷心的感谢，他们是：Bill Appelbe、Nelson Beebe、Jon Bentley、Lois Bogess、Rodney Farrow、Stu Feldman、Charles Fischer、Chris Fraser、Art Gittelman、Eric Grosse、Dave Hanson、Fritz Henglein、Robert Henry、Gerard Holzmann、Steve Johnson、Brian Kernighan、Ken Kubota、Daniel Lehmann、Dave MacQueen、Dianne Maki、Alan Martin、Doug McIlroy、Charles McLaughlin、John Mitchell、Elliott Organick、Robert Paige、Phil Pfeiffer、Rob Pike、Kari-Jouko Rähkä、Dennis Ritchie、Sriram Sankar、Paul Stoecker、Bjarne Stroustrup、Tom Szymanski、Kim Track、Peter Weinberger、Jennifer Widom 和 Reinhard Wilhelm。

作者特别感谢 Patricia Solomon 对本书图片的加工制作。她的工作热情和专业水准是非常令人钦佩的。在本书的编写过程中，J.D.Ullman 得到了以色列艺术和科学学会的爱因斯坦奖学金的资助。最后，我们要特别感谢贝尔实验室在本书准备过程中对我们的支持。

A. V. A., R. S., J. D. U.

目 录

出版者的话	
专家指导委员会	
译者序	
前言	
第1章 编译简介	1
1.1 编译器	1
1.1.1 编译的分析-综合模型	1
1.1.2 编译器的先驱与后继	3
1.2 源程序分析	3
1.2.1 词法分析	3
1.2.2 语法分析	3
1.2.3 语义分析	5
1.2.4 文本格式器中的分析	5
1.3 编译器的各阶段	6
1.3.1 符号表管理	7
1.3.2 错误检测与报告	7
1.3.3 各分析阶段	7
1.3.4 中间代码生成	9
1.3.5 代码优化	9
1.3.6 代码生成	10
1.4 编译器的伙伴	10
1.4.1 预处理器	10
1.4.2 汇编器	11
1.4.3 两遍汇编	12
1.4.4 装配器和连接编辑器	12
1.5 编译器各阶段的分组	13
1.5.1 前端与后端	13
1.5.2 编译器的遍	13
1.5.3 减少编译的遍数	14
1.6 编译器的构造工具	14
参考文献注释	15
第2章 简单的一遍编译器	17
2.1 概述	17
2.2 语法定义	17
2.2.1 分析树	19
2.2.2 二义性	20
2.2.3 操作符的结合规则	20
2.2.4 操作符的优先级	21
2.3 语法制导翻译	22
2.3.1 后缀表示	22
2.3.2 语法制导定义	22
2.3.3 综合属性	23
2.3.4 深度优先遍历	24
2.3.5 翻译模式	25
2.3.6 翻译的输出	25
2.4 语法分析	26
2.4.1 自顶向下语法分析	27
2.4.2 预测分析法	29
2.4.3 何时使用 ϵ 产生式	30
2.4.4 设计一个预测语法分析器	30
2.4.5 左递归	31
2.5 简单表达式的翻译器	32
2.5.1 抽象语法和具体语法	32
2.5.2 调整翻译模式	33
2.5.3 非终结符 $expr$ 、 $term$ 和 $rest$ 的过程	33
2.5.4 翻译器的优化	35
2.5.5 完整程序	35
2.6 词法分析	37
2.6.1 剔除空白符和注释	37
2.6.2 常数	37
2.6.3 识别标识符和关键字	37
2.6.4 词法分析器的接口	38
2.6.5 词法分析器	38
2.7 符号表	40
2.7.1 符号表接口	40
2.7.2 处理保留的关键字	41
2.7.3 符号表的实现方法	41
2.8 抽象堆栈机	42
2.8.1 算术指令	42
2.8.2 左值和右值	43

2.8.3 堆栈操作	43	3.6.1 不确定的有穷自动机	77
2.8.4 表达式的翻译	43	3.6.2 确定的有穷自动机	78
2.8.5 控制流	44	3.6.3 从NFA到DFA的变换	79
2.8.6 语句的翻译	44	3.7 从正规表达式到NFA	81
2.8.7 输出一个翻译	45	3.7.1 从正规表达式构造NFA	81
2.9 技术的综合	46	3.7.2 NFA的双堆栈模拟	84
2.9.1 翻译器的描述	46	3.7.3 时间空间的权衡	85
2.9.2 词法分析器模块lexer.c	47	3.8 设计词法分析器的生成器	85
2.9.3 语法分析器模块parser.c	48	3.8.1 基于NFA的模式匹配	86
2.9.4 输出模块emitter.c	48	3.8.2 词法分析器的DFA	88
2.9.5 符号表模块symbol.c和init.c	48	3.8.3 实现超前扫描操作	88
2.9.6 错误处理模块error.c	48	3.9 基于DFA的模式匹配器的优化	89
2.9.7 编译器的建立	48	3.9.1 NFA的重要状态	89
2.9.8 程序清单	49	3.9.2 从正规表达式到DFA	89
练习	53	3.9.3 最小化DFA的状态数	93
编程练习	54	3.9.4 词法分析器的状态最小化	95
参考文献注释	55	3.9.5 表压缩方法	95
第3章 词法分析	57	练习	97
3.1 词法分析器的作用	57	编程练习	103
3.1.1 词法分析中的问题	58	参考文献注释	103
3.1.2 记号、模式、词素	58	第4章 语法分析	105
3.1.3 记号的属性	59	4.1 语法分析器的作用	105
3.1.4 词法错误	60	4.1.1 语法错误的处理	106
3.2 输入缓冲	60	4.1.2 错误恢复策略	108
3.2.1 双缓冲区	61	4.2 上下文无关文法	109
3.2.2 标志	62	4.2.1 符号的使用约定	110
3.3 记号的描述	62	4.2.2 推导	110
3.3.1 串和语言	62	4.2.3 分析树和推导	112
3.3.2 语言上的运算	63	4.2.4 二义性	113
3.3.3 正规表达式	64	4.3 文法的编写	113
3.3.4 正规定义	65	4.3.1 正规表达式和上下文无关文法的 比较	114
3.3.5 缩写表示法	66	4.3.2 验证文法所产生的语言	114
3.3.6 非正规集	66	4.3.3 消除二义性	115
3.4 记号的识别	67	4.3.4 消除左递归	116
3.4.1 状态转换图	68	4.3.5 提取左因子	117
3.4.2 状态转换图的实现	70	4.3.6 非上下文无关语言的结构	118
3.5 词法分析器描述语言	72	4.4 自顶向下语法分析	120
3.5.1 Lex说明	72	4.4.1 递归下降语法分析法	120
3.5.2 超前扫描操作	75	4.4.2 预测语法分析器	121
3.6 有穷自动机	76		

4.4.3 预测语法分析器的状态转换图	121	参考文献注释	182
4.4.4 非递归的预测分析	123	第5章 语法制导翻译	185
4.4.5 FIRST和FOLLOW	124	5.1 语法制导定义	185
4.4.6 预测分析表的构造	125	5.1.1 语法制导定义的形式	186
4.4.7 LL(1)文法	126	5.1.2 综合属性	186
4.4.8 预测分析的错误恢复	127	5.1.3 继承属性	187
4.5 自底向上语法分析	128	5.1.4 依赖图	187
4.5.1 句柄	129	5.1.5 计算顺序	189
4.5.2 句柄裁剪	130	5.2 语法树的构造	189
4.5.3 用栈实现移动归约分析	131	5.2.1 语法树	190
4.5.4 活前缀	133	5.2.2 构造表达式的语法树	190
4.5.5 移动归约分析过程中的冲突	133	5.2.3 构造语法树的语法制导定义	191
4.6 算符优先分析法	134	5.2.4 表达式的无环有向图	192
4.6.1 使用算符优先关系	135	5.3 自底向上计算S属性定义	194
4.6.2 从结合律和优先级获得算符优先关系	136	5.4 L属性定义	195
4.6.3 处理一元操作符	137	5.4.1 L属性定义	196
4.6.4 优先函数	137	5.4.2 翻译模式	196
4.6.5 算符优先分析中的错误恢复	139	5.5 自顶向下翻译	198
4.7 LR语法分析器	142	5.5.1 从翻译模式中消除左递归	198
4.7.1 LR语法分析算法	142	5.5.2 预测翻译器的设计	201
4.7.2 LR文法	145	5.6 自底向上计算继承属性	202
4.7.3 构造SLR语法分析表	146	5.6.1 删除嵌入在翻译模式中的动作	202
4.7.4 构造规范LR语法分析表	151	5.6.2 分析栈中的继承属性	203
4.7.5 构造LALR语法分析表	155	5.6.3 模拟继承属性的计算	204
4.7.6 LALR语法分析表的有效构造方法	158	5.6.4 用综合属性代替继承属性	206
4.7.7 LR语法分析表的压缩	161	5.6.5 一个难计算的语法制导定义	207
4.8 二义文法的应用	163	5.7 递归计算	207
4.8.1 使用优先级和结合规则来解决分析动作的冲突	163	5.7.1 从左到右遍历	207
4.8.2 悬空else的二义性	164	5.7.2 其他遍历方法	208
4.8.3 特例产生式引起的二义性	165	5.8 编译时属性值的空间分配	209
4.8.4 LR语法分析中的错误恢复	167	5.8.1 在编译时为属性分配空间	209
4.9 语法分析器的生成器	168	5.8.2 避免复制	211
4.9.1 语法分析器的生成器Yacc	169	5.9 编译器构造时的空间分配	211
4.9.2 用Yacc处理二义文法	171	5.9.1 从文法中预知生存期	212
4.9.3 用Lex建立Yacc的词法分析器	173	5.9.2 不相重叠的生存期	214
4.9.4 Yacc的错误恢复	174	5.10 语法制导定义的分析	215
练习	174	5.10.1 属性的递归计算	216
		5.10.2 强无环的语法制导定义	216
		5.10.3 环形检测	217
		练习	219

参考文献注释	221	7.2.3 编译时的局部数据布局	259
第6章 类型检查	223	7.3 存储分配策略	260
6.1 类型系统	224	7.3.1 静态存储分配	260
6.1.1 类型表达式	224	7.3.2 栈式存储分配	262
6.1.2 类型系统	225	7.3.3 悬空引用	265
6.1.3 静态和动态类型检查	226	7.3.4 堆式存储分配	265
6.1.4 错误恢复	226	7.4 对非局部名字的访问	266
6.2 一个简单的类型检查器的说明	226	7.4.1 程序块	267
6.2.1 一种简单语言	226	7.4.2 无嵌套过程的词法作用域	268
6.2.2 表达式的类型检查	227	7.4.3 包含嵌套过程的词法作用域	269
6.2.3 语句的类型检查	228	7.4.4 动态作用域	274
6.2.4 函数的类型检查	228	7.5 参数传递	275
6.3 类型表达式的等价	229	7.5.1 传值调用	275
6.3.1 类型表达式的结构等价	229	7.5.2 引用调用	276
6.3.2 类型表达式的名字	231	7.5.3 复制-恢复	277
6.3.3 类型表示中的环	232	7.5.4 传名调用	277
6.4 类型转换	233	7.6 符号表	278
6.5 函数和运算符的重载	234	7.6.1 符号表表项	278
6.5.1 子表达式的可能类型的集合	235	7.6.2 名字中的字符	279
6.5.2 缩小可能类型的集合	236	7.6.3 存储分配信息	280
6.6 多态函数	237	7.6.4 符号表的线性表数据结构	280
6.6.1 为什么要使用多态函数	237	7.6.5 散列表	281
6.6.2 类型变量	238	7.6.6 表示作用域的信息	283
6.6.3 包含多态函数的语言	239	7.7 支持动态存储分配的语言措施	285
6.6.4 代换、实例和合一	240	7.7.1 垃圾单元	285
6.6.5 多态函数的检查	241	7.7.2 悬空引用	286
6.7 合一算法	244	7.8 动态存储分配技术	287
练习	247	7.8.1 固定块的显式分配	287
参考文献注释	251	7.8.2 变长块的显式分配	287
第7章 运行时环境	253	7.8.3 隐式存储释放	288
7.1 源语言问题	253	7.9 Fortran语言的存储分配	288
7.1.1 过程	253	7.9.1 COMMON区域中的数据	289
7.1.2 活动树	253	7.9.2 一个简单的等价算法	290
7.1.3 控制栈	255	7.9.3 Fortran语言的等价算法	292
7.1.4 声明的作用域	256	7.9.4 映射数据区	294
7.1.5 名字的绑定	256	练习	294
7.1.6 一些问题	257	参考文献注释	298
7.2 存储组织	257	第8章 中间代码生成	299
7.2.1 运行时内存的划分	257	8.1 中间语言	299
7.2.2 活动记录	258	8.1.1 图表示	299

8.1.2 三地址码	300	9.1.5 寄存器分配	335
8.1.3 三地址语句的类型	301	9.1.6 计算次序的选择	336
8.1.4 语法制导翻译生成三地址码	302	9.1.7 代码生成方法	336
8.1.5 三地址语句的实现	303	9.2 目标机器	336
8.1.6 表示方法比较: 间址的使用	305	9.3 运行时存储管理	338
8.2 声明语句	305	9.3.1 静态分配	339
8.2.1 过程中的声明语句	305	9.3.2 栈式分配	340
8.2.2 跟踪作用域信息	306	9.3.3 名字的运行地址	342
8.2.3 记录中的域名	308	9.4 基本块和流图	343
8.3 赋值语句	309	9.4.1 基本块	343
8.3.1 符号表中的名字	309	9.4.2 基本块的变换	344
8.3.2 临时名字的重用	310	9.4.3 保结构变换	344
8.3.3 寻址数组元素	311	9.4.4 代数变换	345
8.3.4 数组元素寻址的翻译模式	312	9.4.5 流图	345
8.3.5 赋值语句中的类型转换	314	9.4.6 基本块的表示	345
8.3.6 记录域的访问	315	9.4.7 循环	346
8.4 布尔表达式	315	9.5 下次引用信息	346
8.4.1 翻译布尔表达式的方法	316	9.5.1 计算下次引用信息	346
8.4.2 数值表示	316	9.5.2 临时名字的存储分配	347
8.4.3 短路代码	317	9.6 一个简单的代码生成器	347
8.4.4 控制流语句	317	9.6.1 寄存器描述符和地址描述符	348
8.4.5 布尔表达式的控制流翻译	319	9.6.2 代码生成算法	348
8.4.6 混合模式的布尔表达式	321	9.6.3 函数getreg	349
8.5 case语句	321	9.6.4 为其他类型的语句生成代码	350
8.6 回填	323	9.6.5 条件语句	351
8.6.1 布尔表达式	323	9.7 寄存器分配与指派	351
8.6.2 控制流语句	326	9.7.1 全局寄存器分配	352
8.6.3 翻译的实现方案	326	9.7.2 引用计数	352
8.6.4 标号和goto	327	9.7.3 外层循环的寄存器指派	353
8.7 过程调用	328	9.7.4 图染色法寄存器分配	354
8.7.1 调用序列	328	9.8 基本块的dag表示法	354
8.7.2 一个简单的例子	328	9.8.1 dag的构造	355
练习	329	9.8.2 dag的应用	357
参考文献注释	331	9.8.3 数组、指针和过程调用	358
第9章 代码生成	333	9.9 窥孔优化	359
9.1 代码生成器设计中的问题	333	9.9.1 冗余加载与保存	360
9.1.1 代码生成器的输入	333	9.9.2 不可达代码	360
9.1.2 目标程序	334	9.9.3 控制流优化	361
9.1.3 存储管理	334	9.9.4 代数化简	361
9.1.4 指令选择	334	9.9.5 强度削弱	361

9.9.6 机器语言的使用	362	10.4.5 可约流图	394
9.10 从dag生成代码	362	10.5 全局数据流分析介绍	395
9.10.1 重排序	362	10.5.1 点和路径	396
9.10.2 对dag的启发式排序	362	10.5.2 到达定义	396
9.10.3 树的最优排序	363	10.5.3 结构化程序的数据流分析	397
9.10.4 标记算法	364	10.5.4 对数据流信息的保守估计	399
9.10.5 从标记树中产生代码	364	10.5.5 in 和 out 的计算	400
9.10.6 多寄存器操作	367	10.5.6 处理循环	401
9.10.7 代数性质	367	10.5.7 集合的表示	402
9.10.8 公共子表达式	368	10.5.8 局部到达定义	403
9.11 动态规划代码生成算法	368	10.5.9 引用-定义链	404
9.11.1 一种寄存器计算机	368	10.5.10 计算顺序	404
9.11.2 动态规划的原理	369	10.5.11 一般控制流	404
9.11.3 邻近计算	369	10.6 数据流方程的迭代解	405
9.11.4 动态规划算法	369	10.6.1 到达定义的迭代算法	406
9.12 代码生成器的生成器	371	10.6.2 可用表达式	408
9.12.1 采用重写树技术的代码生成	371	10.6.3 活跃变量分析	410
9.12.2 借助语法分析的模式匹配	375	10.6.4 定义-引用链	411
9.12.3 用于语义检查的例程	376	10.7 代码改进变换	412
练习	376	10.7.1 全局公共子表达式删除	412
参考文献注释	378	10.7.2 复制传播	413
第10章 代码优化	381	10.7.3 循环不变计算的检测	415
10.1 引言	381	10.7.4 代码外提	415
10.1.1 代码改进变换的准则	381	10.7.5 可选的代码外提方案	417
10.1.2 性能的提高	382	10.7.6 代码外提后对数据流信息的 维护	418
10.1.3 优化编译器的组织	383	10.7.7 归纳变量删除	418
10.2 优化的主要种类	384	10.7.8 带有循环不变表达式的归纳 变量	421
10.2.1 保持功能变换	385	10.8 处理别名	422
10.2.2 公共子表达式	386	10.8.1 一种简单的指针语言	422
10.2.3 复制传播	387	10.8.2 指针赋值的作用	422
10.2.4 无用代码删除	387	10.8.3 利用指针信息	424
10.2.5 循环优化	388	10.8.4 过程间的数据流分析	425
10.2.6 代码外提	388	10.8.5 带有过程调用的代码模型	425
10.2.7 归纳变量和强度削弱	388	10.8.6 别名的计算	426
10.3 基本块的优化	390	10.8.7 存在过程调用时的数据流分析	427
10.4 流图中的循环	392	10.8.8 change信息的用途	428
10.4.1 支配节点	392	10.9 结构化流图的数据流分析	429
10.4.2 自然循环	393		
10.4.3 内循环	393		
10.4.4 前置首节点	394		

10.9.1 深度优先搜索	429	10.12.3 前向方案	452
10.9.2 流图的深度优先表示中的边	431	10.12.4 后向方案	453
10.9.3 流图的深度	431	10.13 优化代码的符号调试	455
10.9.4 区间	432	10.13.1 基本块中变量值的推断	456
10.9.5 区间划分	432	10.13.2 全局优化的影响	459
10.9.6 区间图	433	10.13.3 归纳变量删除	459
10.9.7 节点分裂	433	10.13.4 全局公共子表达式删除	459
10.9.8 T_1 - T_2 分析	434	10.13.5 代码外提	459
10.9.9 区域	434	练习	460
10.9.10 寻找支配节点	435	参考文献注释	465
10.10 高效数据流算法	436	第11章 编写一个编译器	469
10.10.1 迭代算法中的深度优先顺序	436	11.1 编译器设计	469
10.10.2 基于结构的数据流分析	437	11.1.1 源语言问题	469
10.10.3 对基于结构的算法的一些速度上的改进	440	11.1.2 目标语言问题	469
10.10.4 处理不可约流图	441	11.1.3 性能标准	469
10.11 一个数据流分析工具	441	11.2 编译器开发方法	470
10.11.1 数据流分析框架	442	11.3 编译器开发环境	472
10.11.2 数据流分析框架的公理	443	11.4 测试与维护	474
10.11.3 单调性和分配性	444	第12章 编译器实例	475
10.11.4 数据流问题的聚合路径解	447	12.1 数学排版预处理器EQN	475
10.11.5 流问题的保守解	447	12.2 Pascal编译器	475
10.11.6 通用框架的迭代算法	448	12.3 C编译器	476
10.11.7 一个数据流分析工具	448	12.4 Fortran H编译器	477
10.11.8 算法10.18的性质	449	12.4.1 Fortran H中的代码优化	478
10.11.9 算法10.18的收敛性	449	12.4.2 代数优化	478
10.11.10 初始化的修正	450	12.4.3 寄存器优化	478
10.12 类型估计	450	12.5 BLISS/11编译器	479
10.12.1 处理无穷类型集	451	12.6 Modula-2优化编译器	480
10.12.2 一个简单的类型系统	452	附录 一个程序设计项目	483
		参考文献	489
		索引	511