21世纪高等学校计算机规划教材

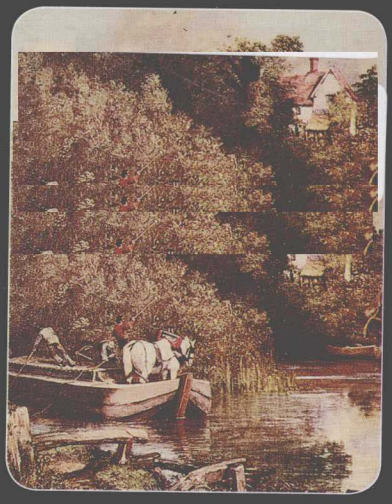
21st Century University Planned Textbooks of Computer Science

编译原理

Principles for Compiler Construction

王生原 董渊 杨萍 张素琴 编著

- 没有深奥理论，但求论述严谨
- 没有面面俱到，但求自成体系
- 立足原理学习，兼顾实际应用



名家系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

编译原理

Principles for Compiler Construction

王生原 董渊 杨萍 张素琴 编著

阜阳师范学院图书馆
藏书

名家系列

人民邮电出版社
北京

图书在版编目（CIP）数据

编译原理 / 王生原等编著. -- 北京 : 人民邮电出版社, 2010.8
（名师系列）
21世纪高等学校计算机规划教材
ISBN 978-7-115-21731-8

Ⅰ. ①编… Ⅱ. ①王… Ⅲ. ①编译程序－程序设计－高等学校－教材 Ⅳ. ①TP314

中国版本图书馆CIP数据核字(2010)第036974号

内 容 提 要

本书主要介绍编译系统的一般构造原理和基本实现技术。内容包括语言基础知识、词法分析、语法分析、中间代码生成、代码优化、目标代码生成、符号表的构造和运行时存储空间的组织等，同时将“PL/0语言编译程序”的设计作为实例贯穿于相关章节中。最后还通过一系列程序实例介绍了工业界广泛使用的开源工具 GCC 和 Binutils。

21 世纪高等学校计算机规划教材

编 译 原 理

◆ 编　　著　王生原　董　渊　杨　萍　张素琴
责任编辑　武恩玉

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787 × 1092　1/16
印张：19.25　　2010 年 8 月第 1 版
字数：449 千字　　2010 年 8 月北京第 1 次印刷

ISBN 978-7-115-21731-8

定价：35.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

出版者的话

计算机科学与技术日新月异的发展，对我国高校计算机人才的培养提出了更高的要求。许多高校主动研究和调整学科内部结构、人才培养目标，提高学科水平和教学质量，精炼教学内容，拓宽专业基础，优化课程结构，改进教学方法，逐步形成了“基础课程精深，专业课程宽新”的良性格局。作为大学计算机教材建设的生力军，人民邮电出版社始终坚持服务高校教学、致力教育资源建设的出版理念，在总结前期教材建设成功经验的同时，深入调研和分析课程体系，并充分结合我国高校计算机教育现状和改革成果，推出“推介名师好书，共享教育资源”的教材建设项目，出版了“21世纪高等学校计算机规划教材”名家系列。

本套教材的突出特点如下：

（1）作者权威 本套教材的作者均为国内计算机学科中的学术泰斗或高校教学一线的教学名师，他们有着深厚的科研功底和丰富的教学经验。可以说，这套教材汇聚了众师之精华，充分显示了这套教材的格调和品位。无论是刚入杏坛的年轻教师，还是象牙塔内的莘莘学子，细细品读其中的章节文字，定会受益匪浅。

（2）定位准确 本套教材是为普通高等院校的学生量身定做的精品教材。具体体现在：一是本套教材的作者长期从事一线科研和教学工作，对高校教学有着深刻而独到的见解；二是本套教材在选题策划阶段便多次召开调研会，对普通高校的教学需求和教材建设情况进行充分摸底，从而保证教材在内容组织和结构安排上更加贴近实际教学；三是组织有关作者到较为典型的普通高等院校讲授课程教学方法，深入了解教师的教学需求，充分把握学生的理解能力，以教材内容引导授课教师严格按照科学方法实施教学。

（3）教材内容与时俱进 本套教材在充分吸收国内外最新计算机教学理念和教育体系的同时，更加注重基础理论、基本知识和基本技能的培养，集思想性、科学性、启发性、先进性和适应性于一身。

（4）一纲多本，合理配套 根据不同的教学法，同一门课程可以有多本不同的教材，教材内容各具特色，实现教材系列资源配套。

总之，本套教材中的每一本精品教材都切实体现了各位教学名师的教学水平，充分折射出名师的教学思想，淋漓尽致地表达着名师的教学风格。我们相信，这套教材的出版发行一定能够启发年轻教师们真正领悟教学精髓，教会学生科学地掌握计算机专业的基本理论和知识，并通过实践深化对理论的理解，学以致用。

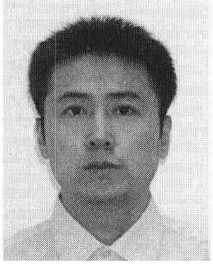
我们相信，这套教材的策划和出版，无论在形式上还是在内容上都能够显著地提高我国高校计算机专业教材的整体水平，为培养符合时代发展要求的具有较强国际竞争力的高素质创新型计算机人才，为我国普通高等教育的计算机教材建设工作做出新的贡献。欢迎各位老师和读者给我们的工作提出宝贵意见。

作者简介

张素琴 清华大学计算机科学与技术系教授，主要研究领域为语言与编译技术。具有三十余年“编译原理”课程教学经验。

王生原 清华大学计算机科学与技术系副教授，主要研究领域为程序设计语言与系统、Petri 网应用。讲授“形式语言与自动机”和“编译原理”课程。

杨萍 北京语言大学信息学院副教授，主要研究领域为人工智能、程序设计语言与系统。讲授“编译原理”和“数据结构”等课程。



董渊 清华大学计算机科学与技术系副教授，主要研究领域为操作系统、编译系统和基于语言的可信软件。讲授“编译原理专题训练”和“软件工程”课程。

试读结束：需要全本请在线购买： www.ertongbook.com

前 言

编译程序是重要的计算机系统软件。编译程序的设计和实现涉及程序设计语言、计算机体系结构、语言理论、算法和软件工程等学科知识，正如著名的计算机科学家 Alfred V. Aho 和 Jeffrey D.Ullman 在他们的编译教材中说得那样：这些原理和技术在每一个计算机学者的工作生涯中，都会反复用到。

本书根据作者多年的教学经验，参照国内外较具影响的专著和教材编写，主要介绍经典的编译原理和技术，内容包括词法分析程序和语法分析程序的设计及自动构造理论、语法制导翻译基础、语义分析、中间代码生成、目标代码运行时的存储组织策略，代码优化和目标代码生成等。这些内容是计算机科学理论应用于计算机系统设计的典范，是计算机学科体系中不可缺少的部分。本书选择 PL/0 编译程序作为实例，以利于原理部分相应知识点的讲解。此外，考虑到近年来嵌入式系统的迅速发展、体系结构的推陈出新对编译技术的迫切需求，本书安排了有关开源的 GCC 编译器和相关工具 Binutils 的章节，为学生将来可能从事相关领域工作构筑基础。

全书由 10 章和 2 个附录组成。第 1 章、第 7 章和第 8 章由张素琴编写，第 2 章、第 3 章、书中与 PL/0 编译程序实例相关的部分和附录 B 由杨萍编写，第 4 章、第 5 章和第 6 章由王生原编写，第 9 章和第 10 章由董渊编写。附录 A 提供 PL/0 源程序，其中的 C 版本由蔡锐、梁英毅 、龚珩、高崇南和王曦等多名助教书写并逐年维护。此外，李叠同学参与了部分章节的校对。

另外，作者要感谢北京工业大学蒋宗礼、北京林业大学李冬梅、北京信息科技大学李明新、中央民族大学林原、北京化工大学史晟辉、中国农业大学王莲芝、北京建筑工程学院张琳等对本书编写提出的宝贵意见和给予的热情帮助！

编译原理课程的选材非常广泛，不同的教学理念对教材内容的要求会有所不同，加之编者水平所限，书中难免存在不当和疏漏之处，诚请读者批评指正。

编 者

2009 年 9 月于清华园

目　录

第 1 章　编译程序概论 …… 1

1.1　什么是编译程序 …… 1
1.2　编译过程和编译程序的结构 …… 3
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 …… 6
1.2.5　代码优化 …… 6
1.2.6　目标代码生成 …… 6
1.2.7　符号表管理和出错处理 …… 7
1.2.8　编译阶段的组合和编译结构 …… 9
1.3　实例：PL/0 编译程序 …… 10
1.3.1　PL/0 语言简介 …… 10
1.3.2　PL/0 语言处理系统 …… 11
习题 …… 13

第 2 章　语言和文法 …… 14

2.1　语言的基本概念 …… 14
2.1.1　字母表和字 …… 14
2.1.2　关于字的运算和字母表上的运算 …… 14
2.1.3　语言 …… 15
2.1.4　关于语言的运算 …… 15
2.2　上下文无关文法 …… 16
2.2.1　上下文无关文法的基本概念 …… 16
2.2.2　归约与推导 …… 17
2.2.3　上下文无关语言 …… 18
2.2.4　句型、句子与分析树 …… 20
2.2.5　归约、推导与分析树之间关系 …… 20
2.2.6　文法的二义性 …… 21
2.3　PL/0 语言的语法 …… 25
2.3.1　PL/0 语言语法的上下文无关文法描述 …… 25
2.3.2　PL/0 语言语法的 EBNF 描述 …… 26
习题 …… 28

第 3 章　词法分析程序及其自动构造 …… 31

3.1　词法分析概述 …… 31
3.1.1　词法分析的任务 …… 31
3.1.2　词法分析在编译程序中的组织 …… 32
3.1.3　词法分析程序中如何识别单词 …… 33
3.2　实例：PL/0 编译程序中词法分析程序的设计和实现 …… 33
3.3　词法分析程序自动构造原理 …… 37
3.3.1　正规表达式与正规语言 …… 37
3.3.2　有限自动机 …… 40
3.3.3　词法分析程序构造的自动化 …… 52
3.4　LEX：一个词法分析程序的生成工具 …… 52
3.4.1　LEX 描述文件中使用的正规表达式 …… 53
3.4.2　LEX 描述文件的格式 …… 54
3.4.3　LEX 的使用 …… 56
3.4.4　与 YACC 的接口约定 …… 56
3.4.5　用 LEX 构造 PL/0 词法分析程序 …… 57
习题 …… 57

第 4 章　自顶向下语法分析 …… 60

4.1　自顶向下分析思想 …… 60
4.2　LL（1）分析方法 …… 63
4.2.1　First 集合和 Follow 集合 …… 63
4.2.2　LL（1）文法 …… 66
4.2.3　LL（1）分析的实现 …… 66
4.2.4　一些有用的文法变换 …… 72
4.3　实例：PL/0 编译程序中语法分析程序的设计和实现 …… 76
4.3.1　PL/0 语法分析程序的自顶向下预测分析思想 …… 76
4.3.2　PL/0 递归下降语法分析程序的设计 …… 78
4.3.3　PL/0 编译程序中的错误处理 …… 80
习题 …… 82

第 5 章　自底向上语法分析 …… 85

5.1　自底向上分析思想 …… 85
5.1.1　短语和直接短语 …… 86
5.1.2　句柄 …… 87
5.1.3　移进-归约分析 …… 89
5.2　LR 分析方法 …… 92

5.2.1 LR 分析基础 …… 92
5.2.2 LR（0）分析 …… 95
5.2.3 SLR（1）分析 …… 100
5.2.4 LR（1）分析 …… 102
5.2.5 LALR（1）分析 …… 107
5.2.6 某些非 LR 文法的强制 LR 分析 …… 109
5.3 LR 分析中的错误处理 …… 111
5.4 几类分析文法之间的关系 …… 113
习题 …… 113

第 6 章 语法制导的语义分析和中间代码生成 …… 118

6.1 语法制导的语义处理基础 …… 118
6.1.1 属性文法以及基于属性文法的语义处理 …… 118
6.1.2 翻译模式以及基于翻译模式的语义处理 …… 127
6.2 语法制导的语义分析 …… 133
6.2.1 语义分析的主要工作 …… 134
6.2.2 类型检查 …… 134
6.3 语法制导的中间代码生成 …… 137
6.3.1 常见的中间表示形式 …… 137
6.3.2 生成抽象语法树 …… 138
6.3.3 生成三地址码 …… 139
6.4 YACC：一个语法分析/语义处理程序的生成工具 …… 147
6.4.1 YACC 描述文件 …… 147
6.4.2 使用 YACC 的一个简单例子 …… 151
6.4.3 用 LEX 和 YACC 实现 PL/0 编译程序 …… 152
习题 …… 152

第 7 章 符号表 …… 158

7.1 名字的属性和说明 …… 158
7.2 符号表的组织 …… 159
7.2.1 符号表的总体组织 …… 159
7.2.2 关键字域的组织 …… 160
7.2.3 符号表的基本实现技术 …… 160
7.3 分程序结构的符号表 …… 161
7.4 PL/0 编译程序中符号表的设计与实现 …… 164
7.4.1 PL/0 符号表的设计 …… 164
7.4.2 作用域与可见性 …… 166
7.4.3 符号表的操作 …… 168
习题 …… 169

第 8 章　目标程序运行时的存储组织 …… 171

8.1　数据空间的使用和管理方法 …… 171
8.1.1　静态存储分配 …… 172
8.1.2　动态存储分配 …… 172
8.2　栈式存储分配的实现 …… 173
8.2.1　动态地分配和释放一个过程的数据空间 …… 174
8.2.2　对非局部变量的引用 …… 175
8.2.3　分程序共享过程的活动记录 …… 179
8.3　参数传递 …… 180
8.3.1　形实参对应的方法 …… 181
8.3.2　传值的实现 …… 181
8.3.3　传地址的实现 …… 182
8.4　PL/0 程序运行时的存储组织 …… 183
8.4.1　PL/0 程序运行栈中的过程活动记录 …… 183
8.4.2　实现过程调用和返回的类 P-code 指令 …… 185
习题 …… 186

第 9 章　代码优化和代码生成 …… 189

9.1　基本块和流图 …… 191
9.1.1　基本块 …… 191
9.1.2　控制流图 …… 192
9.1.3　循环的判定 …… 193
9.1.4　跟踪基本块内部变量使用信息 …… 194
9.1.5　跟踪基本块之间变量使用信息 …… 196
9.2　中间代码优化 …… 200
9.2.1　局部优化 …… 200
9.2.2　循环优化 …… 202
9.2.3　全局优化 …… 205
9.3　目标代码的生成和优化 …… 207
9.3.1　设计代码生成程序的基本考虑 …… 207
9.3.2　一个简单的代码生成算法 …… 209
9.3.3　目标代码优化 …… 211
9.4　PL/0 编译程序中的目标代码生成 …… 212
习题 …… 214

第 10 章　编译器和相关工具实例——GCC/Binutils …… 217

10.1　开源编译器 GCC …… 217

10.1.1 GCC 介绍 …… 218
10.1.2 GCC 总体结构 …… 219
10.1.3 GCC 编译流程 …… 220
10.1.4 GCC 代码组织 …… 221
10.2 开源工具 Binutils …… 222
10.2.1 目标文件 …… 222
10.2.2 汇编器和链接器 …… 223
10.2.3 其他工具 …… 224
10.3 编译器和工具使用实例 …… 224
10.3.1 编译特定版本的编译器 …… 224
10.3.2 查看目标文件 …… 227
10.3.3 程序代码优化 …… 229
习题 …… 232

附录 A PL/0 编译程序文本 …… 233

附录 A-1 PL/0 编译程序文本（Pascal） …… 233
附录 A-2 PL/0 编译程序文本（C） …… 250

附录 B 用于生成某个 PL/0 编译程序的 LEX 描述文件和 YACC 描述文件 … 279

附录 B-1 LEX 描述文件 pl0.l …… 279
附录 B-2 YACC 描述文件 pl0.y …… 280
附录 B-3 头文件 define.h …… 292

参考文献 …… 293

试读结束：需要全本请在线购买： www.ertongbook.com

第 1 章 编译程序概论

程序设计语言是人们和计算机用于描述计算任务的记号和工具。计算机软件无一例外地都是由程序设计语言编写的，要使得这些软件程序能够工作，必须先将其变换为可以在计算机上执行的形式，完成这个变换任务的软件系统称为语言处理系统，编译程序是语言处理系统的一种。本章介绍什么是编译程序，描述编译的一般过程和编译程序的结构。

1.1 什么是编译程序

一个编译程序就是一个语言处理程序，更简单地讲，它是一个语言翻译程序，其功能是把一种语言书写的程序翻译成另一种语言的等价程序。语言翻译程序有很多种。有把一种高级语言程序翻译成另一种高级语言程序的，如把 C++语言程序翻译成 C 语言程序；有把一种高级语言程序翻译成一种低级语言程序的，如把 C 语言程序翻译为某个汇编程序；也有在低级语言程序之间进行翻译的，如把汇编语言程序翻译成机器语言程序。通常把被翻译的语言称作源语言，翻译后的语言称作目标语言，源语言和目标语言书写的程序分别称作源程序和目标程序。如果源语言是像 FORTRAN、Pascal 或 C 那样的高级语言，目标语言是像汇编语言或机器语言那样的低级语言，则这种翻译程序称作编译程序。编译程序的一个重要任务是向用户报告翻译过程中所发现的源程序错误。如果把编译程序看成一个“黑盒子”，它所执行的工作可以用图 1-1 来说明。

当目标程序是一个可以执行的机器语言程序时，它就能由用户调用，接受输入数据而生成输出结果，如图 1-2 所示。

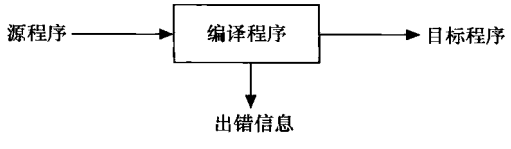


图 1-1 编译程序的功能

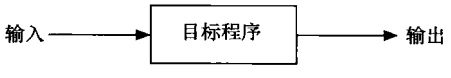


图 1-2 运行目标程序

编译程序是现代计算机系统的基本组成部分之一，而且多数计算机系统都配有不止一个高级语言的编译程序，对有些高级语言甚至配置了几个不同性能的编译程序。

另一种常用的语言处理程序是解释程序。解释程序和编译程序的相同之处是它们所处理的源程序通常都是高级语言程序；但解释程序又不同于编译程序，它不是将高级语言的源程

序翻译之后产生一个目标程序，而是在用户提供的输入之上直接执行源程序所说明的操作，如图 1-3 所示。

通常，编译程序产生目标程序比解释程序在映射输入和输出上快得多，但是解释程序给出的错误诊断信息通常比编译程序好，因为它是一句一句地执行源程序的。

很多语言处理系统组合了编译和解释两个程序，源程序先被翻译为一种中间表示形式的程序，再接受输入数据并对源程序进行解释而生成输出结果，如图 1-4 所示。

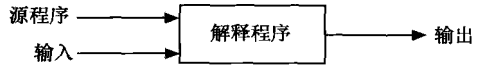


图 1-3　解释程序

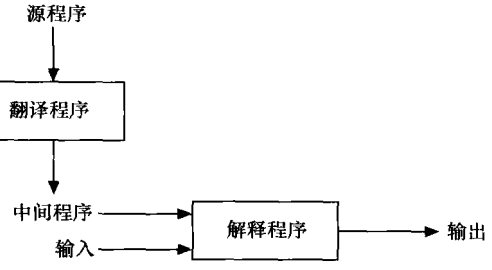


图 1-4　一个混合的语言处理系统

尽管这样的语言处理系统由编译程序和解释程序两部分构成，通常我们还是将其称为一个语言编译系统。比如 Java 语言处理系统包括编译和解释程序，但我们还是称其为 Java 编译系统。图 1-5 是一个典型的 Java 语言环境。Java 语言程序首先被编译为 bytecodes 代码，然后再通过一个称为虚拟机的解释程序对这个 bytecodes 代码进行解释，这样的优点是：在一台机器上编译生成的 bytecodes 可以在另外的机器上，或许跨过网络被解释。为了避免解释 bytecodes 耗时的问题，一般使用 Java 即时编译程序（just-in-time compiler），先把 bytecodes 立即转换为机器语言，再处理输入数据，如图 1-5 所示。

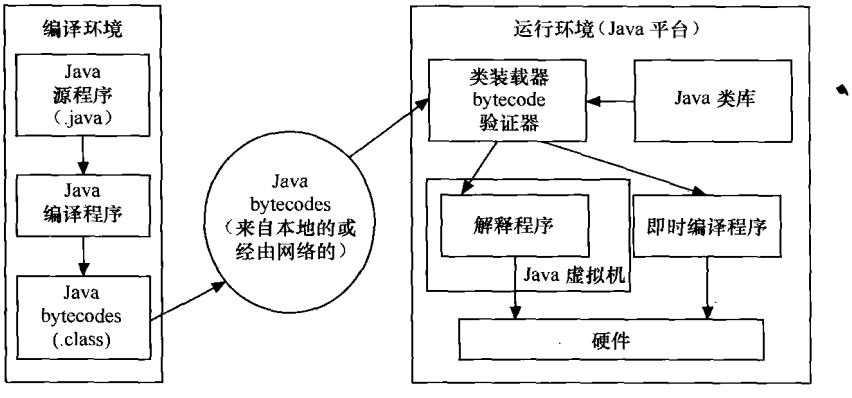


图 1-5　Java 语言环境

在本章 1.3 节中将介绍的 PL/0 编译系统也是由编译程序和解释程序组成的。

除了编译程序外，还需要一些其他的程序才能生成一个可以在计算机上执行的目标程序。图 1-6 所示为一个程序设计语言的典型的处理过程。

一个源程序有时可能分成几个模块存放在不同的文件里，将这些源程序汇集在一起的任务，由一个叫做预处理程序的程序来完成，有些预处理程序也负责宏展开，像 C 语言的预处理程序要完成文件合并、宏展开等任务。经过预处理的程序交由编译程序翻译生成的目标程

序可能是某种汇编代码形式，需要经由汇编程序翻译成可再装配的机器代码，再经由装配/连接编辑程序与某些库程序连接成真正能在机器上运行的代码。也就是说，一个编译程序的输入可能要由一个或多个预处理程序来产生，另外，为得到能运行的机器代码，编译程序的输出可能仍需要进一步的处理。

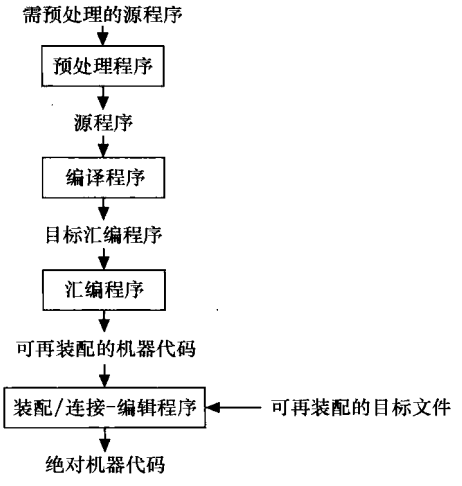


图 1-6　高级语言程序的处理过程

编译程序的基本任务是将源语言程序翻译成等价的目标语言程序。源语言的种类成千上万，从常用的诸如 FORTRAN、Pascal、Java、C++和 C 语言，到各种各样的计算机应用领域的专用语言；而目标语言也是成千上万的，加上编译程序根据它们的构造不同，所执行的具体功能的差异又分成了各种类型，比如，一趟编译、多趟编译的、具有调试或优化功能的等。尽管存在这些明显的复杂因素，但是任何编译程序所必须执行的主要任务基本是一样的，通过理解这些任务，使用同样的基本技术，我们可以为各种各样的源语言和目标语言设计和构造编译程序。

20 世纪 50 年代中期出现了 FORTRAN 等一批高级语言，相应的一批编译系统开发成功。随着编译技术的发展和社会对编译程序需求的不断增长，20 世纪 50 年代末有人开始研究编译程序的自动生成工具，提出并研制编译程序的编译程序。它的功能是以任一语言的词法规则、语法规则和语义解释出发，自动产生该语言的编译程序。目前很多自动生成工具已广泛使用，如词法分析程序的生成系统 LEX，语法分析程序的生成系统 YACC 等。20 世纪 60 年代起，不断有人使用自展技术来构造编译程序。自展的主要特征是用被编译的语言来书写该语言自身的编译程序。1971 年，Pascal 的编译程序用自展技术生成后，其影响就越来越大。

随着并行技术和并行语言的发展，处理并行语言的并行编译技术及将串行程序转换成并行程序的自动并行编译技术也出现了很多研究成果。近年来，支持嵌入式系统和高性能体系结构的编译技术正在深入研究之中。

1.2　编译过程和编译程序的结构

编译程序完成从源程序到目标程序的翻译工作，是一个复杂的整体的过程。从概念上来讲，一个编译程序的整个工作过程是划分成几个阶段进行的，每个阶段将源程序的一种表示形式转换成另一种表示形式，各个阶段进行的操作在逻辑上是紧密连接在一起的，图 1-7 所示为一个编译过程的各个阶段，这是一种典型的划分方法。将编译过程划分成了词法分析、语法分析、语义分析、中间代码生成、中间代码优化、目标代码生成等阶段，我们通过观察源程序在不同阶段所被转换成的表示形式来理解编译各个阶段的任务。

1.2.1　词法分析

词法分析是编译过程的第 1 个阶段。这个阶段的任务是从左到右一个字符一个字符地读

入源程序，对构成源程序的字符流进行扫描和分解，从而识别出一个个单词（也称单词符号或符号），因此这个阶段也称为扫描源程序。这里所谓的单词是指逻辑上紧密相连的一组字符，这些字符具有集体含义。比如标识符是一种单词，保留字（关键字或基本字）是一种单词，此外还有算符、界符等。例如某源程序片段如下：

```
sum:=first+count*10
```

词法分析阶段将构成这段程序的字符组成了如下单词序列：

① 标识符：sum
② 赋值号：:=
③ 标识符：first
④ 加号：+
⑤ 标识符：count
⑥ 乘号：*
⑦ 整数：10

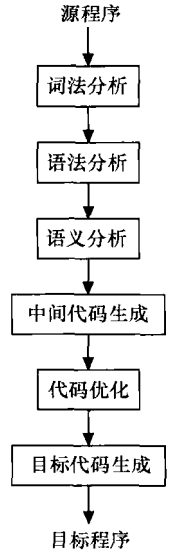


图 1-7　编译的各个阶段

可以看出，5 个字符即 f、i、r、s 和 t 构成了一个分类为标识符的单词 first，一个字符即+和*分别构成了表示加法和乘法的符号，两个字符:和=构成了表示赋值运算的符号。这些单词间的空格在词法分析阶段都被滤掉了。词法分析依据的是构词规则，比如，标识符是由字母字符开头，后跟字母、数字字符的字符序列组成的一种单词，保留字（关键字或基本字）是由字母字符组成的等。第 2 章和第 3 章将分别介绍描述单词构成规则的工具：上下文无关文法以及正则文法和正则表达式。

我们使用 id_1、id_2 和 id_3 分别表示 sum、first 和 count 这 3 个标识符的内部形式，那么经过词法分析后上述程序片段中的赋值语句 sum:=first+count*10 则表示为 id_1:=id_2+id_3*10。

1.2.2　语法分析

语法分析是编译过程的第 2 个阶段。语法分析的任务是在词法分析的基础上将单词序列分解成各类语法短语，如“程序”，“语句”，“表达式”等。一般这种语法短语，也称语法单位，可表示成语法树，比如上述程序段中的单词序列：

id1:=id2+id3*10 经语法分析得知其是 Pascal 语言的“赋值语句”，表示成如图 1-8 所示的语法树或是如图 1-9 所示的语法树。

语法分析所依据的是语言的语法规则，即描述程序结构的规则。通过语法分析确定整个输入串是否构成一个语法上正确的程序。

程序的结构通常是由递归规则表示的，如我们可以用下面的规则来定义表达式。

（1）任何标识符都是表达式。

（2）任何常数（整常数、实常数）都是表达式。

（3）若表达式 1 和表达式 2 都是表达式，那么：

表达式 1 + 表达式 2

表达式 1 * 表达式 2

（表达式 1）

都是表达式。

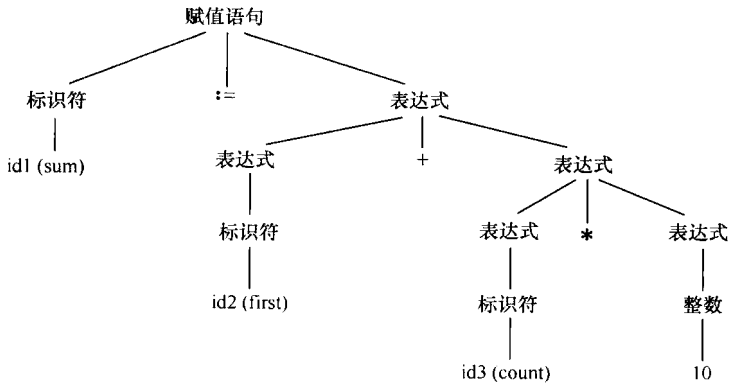


图 1-8　语句 id_1:= id_2 + id_3 * 10 的语法树

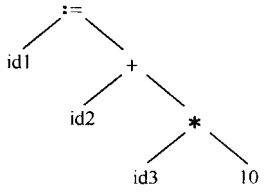


图 1-9　语句 id_1:= id_2 + id_3 * 10 的语法树的另一种形式

类似地，语句也可以递归地定义，例如：

（1）标识符:=表达式　是语句；

（2）while　（表达式）　do　语句和

if　（表达式）　then　语句　else　语句

都是语句。

上述赋值语句 id_1:=id_2+id_3*10 之所以能表示成图 1-8 所示的语法树，依据的是赋值语句和表达式的定义规则。

词法分析和语法分析本质上都是对源程序的结构进行分析。但词法分析的任务仅对源程序进行线性扫描即可完成，比如识别标识符，因为标识符的结构是以字母打头的字母和数字串，只要顺序扫描输入流，遇到的既不是字母又不是数字字符时，将前面所发现的所有字母和数字组合在一起即可构成单词标识符。但这种线性扫描不能用于识别递归定义的语法成分，比如就不能用此办法去匹配表达式中的括号。第 2 章将介绍描述程序结构的工具——上下文无关文法。

1.2.3　语义分析

语义分析的主要功能是审查源程序有无语义错误，为代码生成阶段收集信息。比如语义分析的一个工作是进行类型审查，审查每个算符是否具有语言规范允许的运算对象，当不符合语言规范时，编译程序应该报告错误。比如有的编译程序要对实数用作数组下标的情况报告错误；又如某些语言规定运算对象类型可以被强制，那么当两目运算符施于一个整型运算对象和一个实型运算对象时，编译程序应将整型对象转换成实型对象而不能认为是源程序的错误，假如在语句 sum:=first+count*10 中，*的两个运算对象分别是 count 和 10，count 不是整型，10 是整型，则语义分析阶段进行类型审查之后，在语法分析所得到的分析树上增加一个语义处理结点，表示整型变成实型的一目运算符 inttoreal，则图 1-9 所示的树变成图 1-10 所示的那样。

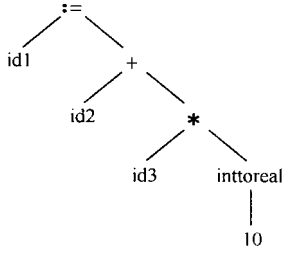


图 1-10　插入语义处理结点的树

1.2.4 中间代码生成

在进行了上述的语法分析和语义分析阶段的工作之后，有的编译程序将源程序变成一种内部表示形式，这种内部表示形式叫做中间语言或中间代码。“中间代码”不同于源程序表示，也不同于目标程序表示，是一种结构简单、含义明确的记号系统，这种记号系统可以设计为多种多样的形式，重要的设计原则为两点：一是容易生成；二是容易将它翻译成目标代码。很多编译程序采用了一种近似“三地址指令”的“四元式”中间代码，这种四元式的形式为：（运算符，运算对象 1，运算对象 2，结果）。

比如源程序 sum:= first + count * 10 可生成四元式序列，如图 1-11 所示，其中 t_i（i=1，2，3）是编译程序生成的临时名字，用于存放运算的中间结果。

1.2.5 代码优化

代码优化的目的是使生成的目标代码更为高效，即省时间和省空间。这个工作可以在两个阶段进行，一个是在中间代码生成后，一个是在目标代码生成后。

在中间代码一级上进行的优化是与机器无关的，在目标代码生成之后，进行的目标代码优化常常根据机器本身的特点来考虑，比如利用特殊指令，像加一、减一指令来优化一些运算等。目标代码优化也称为与机器有关的优化。

优化任务是对代码进行变换或进行改造，以求编译程序生成的目标代码运行的时效更优。比如图 1-11 所示的代码可变换为图 1-12 所示的代码，仅剩余两个四元式而执行同样的计算。也就是编译程序的这个阶段已经把将 10 转换成实型数的代码化简掉了，同时因为 t_3 仅仅用来将其值传递给 id1，也可以被化简掉。编译优化的工作基础和各种不同类型的优化方法将在第 9 章详细介绍。

1.2.6 目标代码生成

目标代码生成的任务是把中间代码变换成特定机器上的绝对指令代码或可重定位的指令代码或汇编指令代码。假如采用本书第 9 章引入的计算机模型 MM 的汇编指令（指令语义参见图 9.27），则图 1-12 所示的中间代码生成如图 1-13 所示的汇编代码。

(1)(inttoreal 10 – t_1)
(2)(* id_3 t_1 t_2)
(3)(+ id_2 t_2 t_3)
(4)(:= t_3 – id_1)

图 1-11 中间代码

(1)(* id_3 10.0 t_1)
(2)(+ id_2 t_1 id_1)

图 1-12 优化后的中间代码

(1) li r8, id3
(2) li r9, 10.0
(3) mul r10, r8, r9
(4) li r11, id2
(5) add r12, r10, r11
(6) sw r12, id1

图 1-13 目标代码

目标代码生成是编译的最后阶段，它的工作与硬件系统结构和指令含义有关，这个阶段的工作很复杂，涉及硬件系统功能部件的运用、机器指令的选择、各种数据类型变量的存储空间分配及寄存器和后缓寄存器的调度等。将在第 9 章进行讨论。

试读结束：需要全本请在线购买： www.ertongbook.com