普通高等教育“十一五”国家级规划教材

清华大学名优教材立项资助

清華大學 计算机系列教材

王生原 董渊 张素琴 吕映芝 蒋维杜 编著

编译原理（第3版）

清华大学出版社

普通高等教育“十一五”国家级规划教材

清华大学名优教材立项资助

清華大學 计算机系列教材

王生原 董渊 张素琴 吕映芝 蒋维杜 编著

编译原理（第3版）

清华大学出版社
北京

内 容 提 要

本书介绍程序设计语言编译程序构造的一般原理、基本设计方法和主要实现技术，主要内容包括文法、自动机和语言的基础知识，词法分析，语法分析，语法制导的语义计算，语义分析，中间代码生成，运行时存储组织，代码优化和目标代码生成。

除了基本设计原理外，书中还包含两个小型编译程序的设计实例，可选作课程设计的素材。一个是PL/0语言编译程序，其设计和实现框架贯穿于本书相关章节中；另一个是简单面向对象语言Decaf的编译程序。本书最后还介绍了业界广泛使用的开源编译器GCC及和它紧密相关的Binutils工具链，通过一系列程序实例说明这些工具的作用和基本用法。

本书可作为高等院校计算机科学与技术相关专业的本科生教材，也可作为相关教师、研究生或工程技术人员的参考书。

图书在版编目（CIP）数据

编译原理/王生原等编著. --3版. —北京：清华大学出版社，2015
清华大学计算机系列教材
ISBN 978-7-302-38141-9

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(2014)第227991号

责任编辑：白立军　战晓雷
封面设计：常雪影
责任校对：白　蕾
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：25.75　　**字　　数**：626千字
版　　次：1998年1月第1版　2015年6月第3版　　**印　　次**：2015年6月第1次印刷
印　　数：1～3000
定　　价：49.00元

产品编号：026315-01

前　言

编译程序(或编译器、编译系统)在计算机科学与技术的发展历史中发挥了巨大作用,是计算机系统的核心支撑软件。“编译原理”一直以来是国内外大学计算机相关专业的重要课程,其知识结构贯穿程序设计语言、系统环境以及体系结构,能以相对独立的视角体现从软件到硬件以及软硬件协同的整机概念;同时,其理论基础又涉及形式语言与自动机、数据结构与算法等计算机学科的许多重要方面,不愧为联系计算机科学理论和计算机系统的典范。这一知识体系所涉及的原理和技术不仅用于编写编译程序,也适用于很多软件的设计。著名的计算机科学家 A. V. Aho 和 J. D. Ullman 在他们的著作中说:“在每一个计算机科学家的研究生涯中,这些原理和技术都会反复用到。”

本书介绍程序设计语言编译程序构造的一般原理、基本设计方法和主要实现技术,主要面向计算机科学与技术相关专业本科生的专业学习和素质培养,也可供从事系统软件和软件工具研究及开发的人员参考。

全书共 12 章。前面几章中有关词法分析和语法分析的部分,基本上延续了本书前两个版本的风格和内容,有利于之前阅读和使用过这套教材的教师和学生衔接。新版本重新组织了语法制导的方法、语义分析、中间代码生成、运行时存储组织、代码优化和目标代码生成等相关内容,进行了适当的充实与删减,力求在各主要知识点之间达到某种较合理的均衡,使学生在本科层次的学习中尽可能对编译程序的构造原理和实现技术从整体知识层面上有较好的掌握。

对于结合实例的讲解,本书沿用了前两个版本使用的 PL/0 编译程序。PL/0 编译程序比较简单,但不失代表性,在编译原理教学中具有广泛的使用基础。通常情况下,学生能够在很短的时间内掌握 PL/0 编译程序的实现脉络,对于快速了解一个具体编译程序的作用和设计思想有很好的帮助。和前面的版本不同,第 3 版中是将 PL/0 编译程序的介绍分散于不同章节中,不同学校或专业的课程可根据自身的情况选择集中学习和分阶段学习。

“编译原理”是一门对实践性要求较高的课程,通常应该设置专门的课程设计。书中涉及两个小型编译程序的设计实例,可选作课程设计的素材。一个是 PL/0 语言编译程序,其设计和实现框架贯穿于全书相关章节;另一个是简单面向对象语言 Decaf 的编译程序,参见第 11 章。不同学校或专业的课程可根据自身的情况制订适当的课程设计方案。

近年来,在许多专业应用场合,熟练使用与编译程序/系统相关的系统级软件工具已成为必须掌握的基本技能之一。为此,本书安排了有关开源的 GCC 编译器和相关工具链 Binutils 的章节(第 12 章),为学生将来有可能从事相关领域的工作进行基本和必要的准备。对于这部分内容,不同学校或专业的课程可根据自身情况引导或建议学生进行适当的训练。

本书的第 1 章、第 2 章和第 3 章由张素琴和王生原共同编写,第 4 章和第 6 章由吕映芝、张素琴和王生原共同编写,第 5 章由吕映芝编写,第 7 章和第 11 章由王生原编写,第 8 章和第 9 章由王生原和蒋维杜共同编写,第 10 章由董渊和王生原共同编写,第 12 章由董渊编写。

附录中包含 PL/0 源程序的 Pascal 版本和 C 版本的代码，Java 版本的代码可从清华大学出版社网站上获取。另外，若相关课程需要用到 Decaf 编译实验框架的代码，任课教师可与清华大学出版社或编者联系（仅限于用作教学资源的共享与交流）。

适合在“编译原理”课程中讲授的内容非常广泛，从国际上的著名教材来看，在侧重点、内容和风格上都有相当大的差异。由于编者水平所限，书中必然存在不当和疏漏之处，诚请广大读者批评指正。

编　者

2015 年 5 月

目　录

第1章 引 论

1.1 什么是编译程序

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种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。从功能上看，一个编译程序就是一个语言翻译程序。语言翻译程序把一种语言（称作源语言）书写的程序翻译成另一种语言（称作目标语言）的等价程序。比如，汇编程序是一个翻译程序，它把汇编语言程序翻译成机器语言程序。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。把编译程序看成一个“黑盒子”，它所执行的转换工作可用图 1.1 来说明。

一个编译程序的重要性体现在它使得多数计算机用户不必考虑与机器有关的烦琐细节，使程序员和程序设计专家独立于机器，这对于当今机器的数量和种类持续不断地增长的年代尤为重要。

使用过计算机的人都知道，除了编译程序外，还需要一些其他程序才能生成一个可在计算机上执行的目标程序。下面分析一个程序设计语言程序的典型的处理过程（见图 1.2），可以从中进一步了解编译程序的作用。

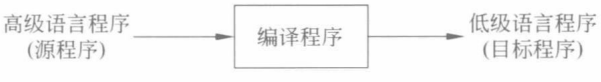

图 1.1 编译程序的功能

图 1.2 高级语言程序的处理过程

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。图 1.2 中的编译程序生成的目标程序是汇编代

码形式，需要经由汇编程序翻译成可再装配(或可重定位)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地处理。

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、C、Java 和 C++ 等语言，到各种各样的计算机应用领域的专用语言；而目标语言也是种类繁多的，加上编译程序由于构造不同，所执行的具体功能有差异，又分成了各种类型，如一趟编译、多趟编译、具有调试或优化功能的编译等。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，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。

据说第一个编译程序出现在 20 世纪 50 年代早期，很难讲出确切的时间，因为当初大量的实验和实现工作是由不同的小组独立完成的，多数早期的编译工作是将算术公式翻译成机器代码。用现在的标准来衡量，当时的编译程序能完成的工作十分初步，如只允许简单的单目运算，数据元素的命名方式有很多限制，然而它们奠定了对高级语言编译程序的研究和开发的基础。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。

1.2 编译过程和编译程序的结构

1.2.1 编译过程概述

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 1.3 给出了一个编译过程的各个阶段，这是一种典型的划分方法，将编译过程划分成词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成 6 个阶段。

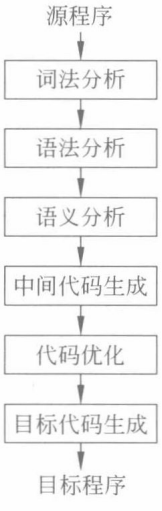

图 1.3 编译的各个阶段

下面通过源程序在不同阶段所被转换成的表示形式来介绍各个阶段的任务。

1. 词法分析

词法分析是编译过程的第一个阶段。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(一些场合下也称单词符号或符号)。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。例如，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）是一种单词，此外还有算符、界符等。例如，某源程序片段如下：

```
begin var sum,first,count:real;sum :=first+count * 10 end.
```

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) 保留字 begin　(2) 保留字 var　(3) 标识符 sum
(4) 逗号，　(5) 标识符 first　(6) 逗号，
(7) 标识符 count　(8) 冒号：　(9) 保留字 real
(10) 分号；　(11) 标识符 sum　(12) 赋值号 :=
(13) 标识符 first　(14) 加号+　(15) 标识符 count
(16) 乘号 *　(17) 整数 10　(18) 保留字 end
(19) 界符 .

可以看出，5 个字符 b、e、g、i 和 n 构成了一个称为保留字的单词 begin，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。

用 id1、id2 和 id3 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 id1 :=id2+id3 * 10。

2. 语法分析

语法分析是编译过程的第二个阶段。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。这种语法短语也称为语法单位，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 1.4 所示的语法树或者如图 1.5 所示的简捷形式的语法树。

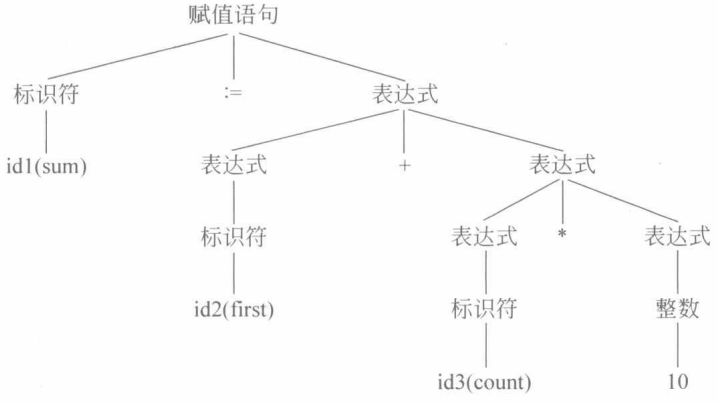

图 1.4　语句 id1 :=id2+id3 * 10 的语法树

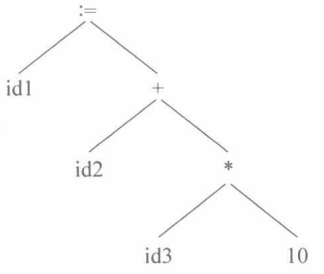

图 1.5　语句 id1 :=id2+id3 * 10 的语法树的简捷形式

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，例如，可以用下面的规则来定义表达式：

(1) 任何标识符是表达式。

(2) 任何常数(整常数、实常数)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。

类似地，语句也可以递归地定义，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 id1 :＝id2＋id3 * 10 之所以能表示成图 1.4 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字母打头的字母和数字串，这只要顺序扫描输入流，遇到既不是字母又不是数字的字符时，将前面所发现的所有字母和数字组合在一起构成标识符单词即可。但这种线性扫描不能用于识别递归定义的语法成分，比如不能用此办法去匹配表达式中的括号。

3. 语义分析

语义分析是审查源程序有无语义错误，为代码生成阶段收集类型信息。例如，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，编译程序应报告错误。有的编译程序要对实数用作数组下标的情况报告错误。某些语言规定运算对象可被强制转换数据类型，那么当二目运算施于一个整型对象和一个实型对象时，编译程序应将整型对象转换成实型对象进行处理而不能认为是源程序的错误，假如在语句 sum :＝first＋count * 10 中，count 是实型，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 inttoreal，表示整型 10 变成实型 10.0 的一目算符，则图 1.5 的树变成图 1.6 所示的树。

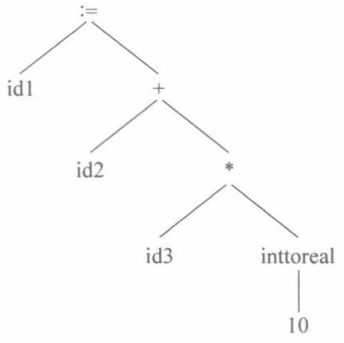

图 1.6　插入语义处理结点的树

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。所谓“中间代码”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)

例如，源程序 sum :=first+count * 10 可生成如图 1.7 所示的四元式序列，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。

5. 代码优化

这一阶段的任务是对前一阶段产生的中间代码进行变换或进行改造，目的是使生成的目标代码更为高效，即省时间和省空间。例如，图 1.7 的代码可变换为图 1.8 的代码，仅剩两个四元式而执行同样的计算。也就是说，在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了，同时因为 t_3 仅仅用来将其值传递给 id1，也可以被化简，这只是优化工作的两个方面；此外，诸如公共子表达式的删除、强度削弱、循环优化等优化工作将在第 10 章详细介绍。

(1)	(inttoreal	10	—	t_1)
(2)	(*	id3	t_1	t_2)
(3)	(+	id2	t_2	t_3)
(4)	(:=	t_3	—	id1)

图 1.7 中间代码

(*)	id3	10.0	t_1)	
(+	id2	t_1	id1)	

图 1.8 优化后的中间代码

(1)	MOV	id3,	R2
(2)	MUL	#10.0,	R2
(3)	MOV	id2	R1
(4)	ADD	R2,	R1
(5)	MOV	R1,	id1

图 1.9 目标代码

6. 目标代码生成

这一阶段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。这是编译的最后阶段，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以及寄存器和后缓寄存器的调度等。

例如，使用两个寄存器(R1 和 R2)，可能将图 1.8 所示的中间代码生成如图 1.9 所示的某种汇编代码。第 1 条指令将 id3 的内容送至寄存器 R2，第 2 条指令将其与实常数 10.0 相乘，这里用 # 表明 10.0 处理为常数，第 3 条指令将 id2 移至寄存器 R1，第 4 条指令将 R1 和 R2 中的值相加，第 5 条指令将寄存器 R1 的值移到 id1 的地址中。

上述编译过程的阶段划分是一个典型处理模式，事实上并非所有的编译程序都分成这样几个阶段，有些编译程序并不需要生成中间代码，有些编译程序不进行优化，即优化阶段可省去，有些最简单的编译程序在语法分析的同时产生目标指令代码，如 1.4 节介绍的 PL/0 语言编译程序。不过多数实用的编译程序都包含上述几个阶段的工作过程。

1.2.2 编译程序的结构

上述编译过程的 6 个阶段的任务可以分别由 6 个模块完成，分别称作词法分析程序、语法分析程序、语义分析程序、中间代码生成程序、代码优化程序和目标代码生成程序。此外，一个完整的编译程序还必须包括表格管理程序和出错处理程序。图 1.10 给出了一个典型的编译程序结构框图。

表格管理和出错处理与上述 6 个阶段都有联系。编译过程中源程序的各种信息被保留在种种不同的表格里，编译各阶段的工作都涉及构造、查找或更新有关的表格，因此需要有

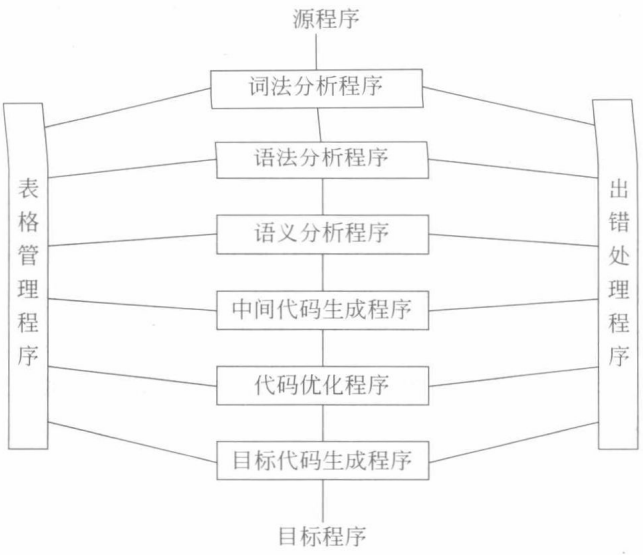

图 1.10　编译程序结构框图

表格管理的工作。如果编译过程中发现源程序有错误，编译程序应报告错误的性质和错误发生的地点，并且将错误所造成的影响限制在尽可能小的范围内，使得源程序的其余部分能继续被编译下去，有些编译程序还能自动校正错误，这些工作由出错处理程序完成。

1.2.3　编译阶段的组合

1.2.1 节所讨论的编译阶段的划分是编译程序的逻辑组织。有时把编译的过程分为前端(front end)和后端(back end)，前端的工作主要依赖于源语言而与目标机无关。前端通常包括词法分析、语法分析、语义分析和中间代码生成这些阶段，某些优化工作也可在前端做，还包括与前端每个阶段相关的出错处理工作和符号表管理工作。后端指的是那些依赖于目标机而一般不依赖于源语言，只与中间代码有关的那些阶段的工作，即目标代码生成，以及相关出错处理和符号表操作。

若按照这种组合方式实现编译程序，可以设想，某一编译程序的前端加上相应的后端则可以为不同的机器构成同一个源语言的编译程序。也可以设想，不同语言编译的前端生成同一种中间语言，再使用一个共同的后端，则可为同一机器生成几个语言的编译程序。

一个编译过程可由一遍、两遍或多遍完成。所谓“遍”，也称作“趟”，是对源程序或其等价的中间语言程序从头到尾扫描并完成规定任务的过程。每一遍扫描可完成上述一个阶段或多个阶段的工作。例如，一遍可以只完成词法分析工作，一遍完成词法分析和语法分析工作，甚至一遍完成整个编译工作。对于多遍的编译程序，第一遍的输入是用户书写的源程序，最后一遍的输出是目标程序，其余是上一遍的输出为下一遍的输入。在实际的编译系统的设计中，编译的几个阶段的工作究竟应该怎样组合，即编译程序究竟分成几遍，参考的因素主要是源语言和机器(目标机)的特征。例如，源语言的结构直接影响编译的遍的划分；像 PL/1 或 ALGOL 68 那样的语言，允许名字的说明出现在名字的使用之后，那么在看到名字之前是不便为包含该名字的表达式生成代码的，这种语言的编译程序至少分成两遍才容易生成代码。另外，机器的情况，即编译程序工作的环境也影响编译程序的遍数的划分。一个