编译原理

陈 英 编著

北京理工大学出版社

编译原理

陈　英　编著

北京理工大学出版社

内 容 简 介

本书系统全面地介绍经典、广泛应用的高级程序设计语言编译程序的构造原理、实现技术、方法和工具。本书包含了现代编译程序设计的基础理论和技术，并在语义分析、代码优化、面向对象语言的编译等方面，反映了20世纪90年代后的一些重要研究成果。本书的组织，注重循序渐进，深入浅出，每章开头提炼了该章涉及的主要内容提要和要点，并提供了编译程序实现的具体实例，每章末尾给出了丰富的练习和思考题，辅助读者更好地掌握编译原理。

本书可以作为计算机学科类专业及相关专业的教材，也可以作为软件技术人员的参考用书。

图书在版编目(CIP)数据

编译原理/陈英编著 .—北京：北京理工大学出版社，2001.9
ISBN 7-81045-838-8

Ⅰ.编… Ⅱ.陈… Ⅲ.编译程序-程序设计 Ⅳ.TB314

中国版本图书馆 CIP 数据核字(2001)第 050020 号

责任印制：李绍英　　责任校对：郑兴玉

北京理工大学出版社出版发行
(北京市海淀区中关村南大街 5 号)
邮政编码 100081　电话(010)68912824
各地新华书店经售
北京市房山先锋印刷厂印刷
*
787 毫米×1092 毫米　16 开本　19.75 印张　466 千字
2001 年 9 月第 1 版　2001 年 9 月第 1 次印刷
印数：1—4000 册　　定价：30.00 元

前　言

编译原理作为计算机学科的一门重要专业基础课，该课程列入国际 ACM 教程和 IEEE 计算机学科的正式主干课程，并提高该课程内容的课时比重，这充分体现了其在计算机科学中的地位和作用。

编译程序是计算机系统软件的主要组成部分，是计算机科学中发展迅速、系统、成熟的一个分支，其基本原理和技术也适用于一般软件的设计和实现，而且在软件工程、软件自动化、逆向软件工程、再造软件工程等领域有着广泛的应用。

本书旨在介绍编译程序设计的基本原理、实现技术、方法和工具。全书分为 10 章，第 1 章作为全书的开场白，介绍了编译程序有关的概念，编译过程，编译程序的结构与组织等要点。第 2 章作为后续各章的基础知识，也是编译程序应该起码具备的理论基础，对形式语言与自动机理论作了基本的介绍。第 3 章以正则文法、正规式、有限自动机为工具，讨论了词法分析器的设计和实现。第 4 章对常规的语法分析方法，即自上而下和自下而上分析中的几种典型方法展开了较深入的讨论，并结合流行、实用、高效的 LR 分析方法，介绍了二义文法的分析应用，编译程序的出错处理。第 3 章和第 4 章的最后，进一步讨论了流行的词法分析和语法分析自动生成工具 LEX 及 FLEX，YACC 及 Bison 的构造原理与应用，并以 ANSI_C 语言为例，给出了其 LEX 和 YACC 的描述。第 5 章涉及语义分析方法和属性翻译文法，中间语言，符号表及类型检查技术，流行语言中的典型语句的翻译。第 6 章介绍编译程序运行时环境的有关概念和存储组织与分配技术。第 7 章介绍中间代码级上的优化，展开讨论了优化的基本概念，优化涉及的控制流分析和数据流分析技术，以及中间代码上的局部优化和循环优化技术及实现。第 8 章对面向对象语言的翻译要点进行了讨论。第 9 章简单介绍了代码生成的有关知识点及目标代码级可实施的窥孔优化技术。第 10 章以类 PASCAL 语言为源语言，提供了一个短小、精悍的编译程序实现的范例，以弥补编译程序从原理到工程实现的鸿沟。

纵观本书的组织，注重循序渐进，深入浅出，每章开头提炼了该章涉及的主要内容提要和要点，并提供了编译程序实现的具体实例，每章末尾给出了丰富的练习和思考题，辅助读者更好地掌握编译原理。本书包含了现代编译程序设计的基础理论和技术，并在语义分析，代码优化，面向对象语言的编译等方面，反映了 20 世纪 90 年代后的一些重要研究成果。

本书是作者多年教学实践和科研工作的汇集、提炼和整理，同时参考了国内外一些专著、论文和资料，参考了一些专家学者的研究成果，对所有这些前辈和同行的引导和帮助表示衷心的感谢。

本书完成过程中，得到了北京理工大学出版社的鼎力协助，尤其是总编辑林国璋教授、责任编辑李俨老师的热忱支持和具体指导，得到了我的家人和同事，以及教研室和课题组的研究生们的全力支持和帮助。是他们：刘建平、刘洁颖、黄湘武、施兴华、刘然、邹静、张荣妹、黄波、吴沁奕等承担了本书的全部编辑、排版和校斟工作，作者在此一并深为致谢。

鉴于作者水平有限，时间制约，书中难免有不妥之处，敬请读者批评指正。

编　者

2001.6

目　录

第1章 编译引论

【本章导读提要】

本章内容作为了解、学习和掌握编译程序原理与技术的基础，主要涉及的内容与要点是：

- 什么是编译程序；源程序的运行；编译程序的分类与表示。
- 术语：源语言、源程序、目标语言、目标程序、宿主语言、遍。
- 编译程序的组成结构，各部分间逻辑关系和主要功能。
- 编译程序的构造要素。

1.1 程序设计语言与编译程序

1.1.1 编译程序鸟瞰

学习、掌握编译程序的构造原理和技术，需搞清编译程序的由来及定义，即何为编译程序，这亦是本书的研究对象。

众所皆知，一个计算机程序总是基于某种程序设计语言。半个多世纪以来，程序设计语言经历了由低级向高级的发展，从最初的机器语言、汇编语言，发展到较高级的程序设计语言直至今天的第四代、第五代高级语言。高级程序设计语言的以人为本，面向自然表达，易学、易用、易理解、易修改等优势加速了程序设计语言的发展。计算机语言的发展和应用，自然展示了这样的问题，它大大提高了计算机的功能和效率，促进了计算机的普及使用，这在计算机科学发展史上是一个重要的里程碑。计算机的深入发展和应用普及除了计算机硬件本身发展迅速的因素外，与之相适应的更为重要的因素是计算机软件的飞速发展，多数计算机用户更为直接的是通过应用程序设计语言来实现使用计算机的意图和目的。

但是就目前而言，计算机硬件自身根本不懂 BASIC，PASCAL，C， C++， ADA 和 JAVA 等高级语言，用高级语言编写的程序机器不能直接执行，因为计算机能识别的是机器语言。高级程序设计语言只是人和计算机交互的媒介。那么，如何使一个高级语言编写的程序能够在只认得机器语言的计算机上执行呢？这就需要像人们为了通信、交流的方便， 建立各种语言的翻译一样，由从事计算机软件工作的人员搭一座桥梁，沟通计算机硬件与用户之间的渠道，这个桥梁即为“编译程序”，亦称“语言处理程序”。通过这样的程序的翻译处理工作，机器才能执行高级语言编写的程序。编译程序所起的桥梁作用，可类比为两个不同语言的人借助翻译进行交流，不同之处在于编译程序是一个单向的翻译。编译程序的功能如图 1.1 所示。

确切地讲，把用某一种程序设计语言写的源程序翻译成等价的另一种语言程序(目标程序)的程序，称之为编译程序(编译器 compiler)或翻译程序（翻译器 translator）。简单地说 ，编译程序是一个翻译程序，它是程序设计语言的支持工具或环境。术语“编译”的内涵是实现从源语言表示的算法向目标语言表示的算法的等价变换。

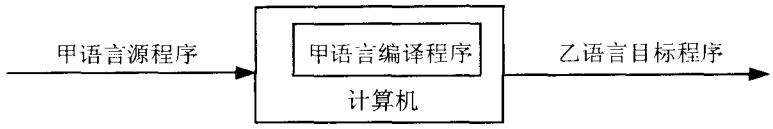


图 1.1 编译程序的功能

用源语言写的程序称为源程序(source program)。源语言是用来编写源程序的语言，一般是汇编语言或高级程序设计语言。源程序经过编译程序翻译后生成的程序称之为目标程序(target program)。目标程序可以用机器语言、汇编语言、甚至高级语言或用户自定义的某类中间语言来描述。例如，若将 A 语言的一个程序翻译成 B 语言的一个程序，翻译的实现语言是 Y 语言，称 A 语言是翻译的源语言，B 语言是目标语言，Y 语言是宿主语言。

1.1.2 源程序的执行

需要编译程序支持的源程序的执行分为两个阶段：编译阶段和运行阶段，如图 1.2 所示。编译阶段对整个源程序进行分析，翻译成等价的目标程序，然后在运行子程序的支持下运行。运行子程序是为了支持目标的运行而开发的程序，例如有系统提供的标准函数及其它目标程序所调用的程序等。

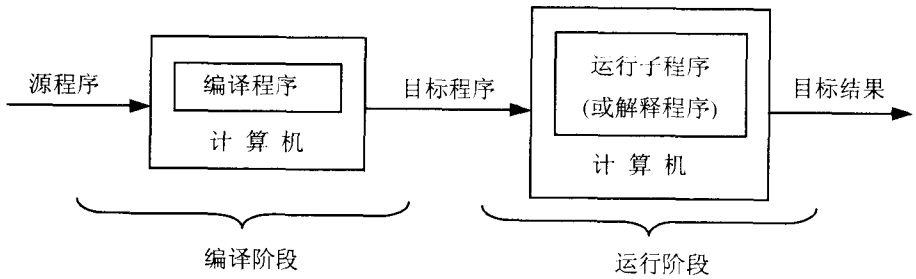


图 1.2 源程序的编译执行

1.2 编译程序的表示与分类

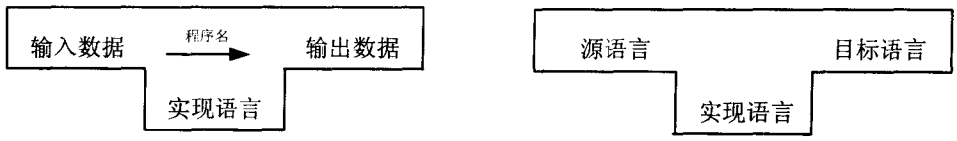
1.2.1 T 型图

T 型图是表示一个程序，特别是编译程序的一种有效的表示方法。图 1.3 给出了这种表示方法。

T 型图可以方便地表示编译程序的三个方面，即源语言、目标语言和编译程序的实现语言 (或宿主语言)。其中，T 型图的左上角表示源语言，右上角表示目标语言，而其底部表示实现语言。

例如，对一个用 Z 语言实现的，从源语言 X 到目标语言 Y 的编译程序，可用图 1.4 表

示，此编译程序也可记作 C_Z^{XY}。



(a) 一般程序的 T 型图　　(b) 编译程序的 T 型图

图 1.3　T 型图表示

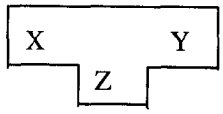


图 1.4　C_Z^{XY}的 T 型图

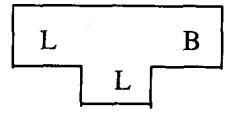


图 1.5　C_L^{LB}的 T 型图

用 T 型图表示交叉编译和编译程序的移植非常方便、清晰。例如，将 A 机器上已经存在的 L 语言的编译程序移植到 B 机器上，按下列步骤实现。

第一步，用 L 语言书写 L 语言的编译程序，该编译程序产生的目标程序是 B 机器上的汇编语言或机器语言，可表示为 C_L^{LB}，其 T 型图如图 1.5 所示。

第二步，把 C_L^{LB}经过 C_L^{LA}编译得到 C_A^{LB}，其 T 形图如图 1.6 所示。

T 型图的结合规则很简单，只要满足以下条件：即中间那个 T 型图的两臂上的语言分别与左右两个 T 型图脚上的语言相同，且对于左右两个 T 型图而言，其两个左端的语言必须相同，两个右端的语言必须相同。

第三步，把 C_L^{LB}在 A 机器上经过 C_A^{LB}编译得到 C_B^{LB}，其 T 形图如图 1.7 所示。

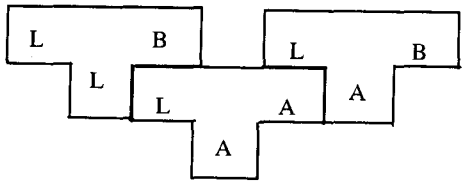


图 1.6　C_A^{LB}的双层 T 型图

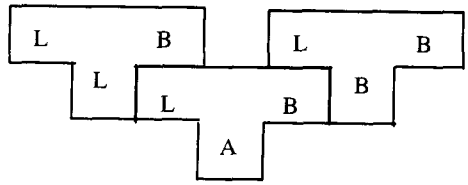


图 1.7　C_B^{LB}的双层 T 型图

经过以上的三步最终在 B 机器上实现了 L 语言的编译程序 C_B^{LB}。

1.2.2　编译程序的分类

如同各种事物的分类，基于不同的角度可以对编译程序进行分类。

编译程序从实现角度一般可以分为汇编程序、解释程序和编译程序，甚至把宏汇编器、预处理器等也认为是编译程序的一类。

- 汇编程序(assembler)：汇编语言是计算机语言的符号形式。若源程序用汇编语言编写经翻译生成的是机器语言表示的目标程序，该翻译程序称为“汇编程序”。通常，编译程序会

生成汇编语言程序作为其目标语言，然后再由汇编程序将它翻译成目标代码。

- 编译程序：若源程序用高级语言编写，经翻译加工生成某种形式的目标程序，该翻译程序称为“编译程序”。
- 解释程序(interpreter)：接收某语言的源程序(或经翻译生成的中间代码程序)直接在机器上解释执行的一类翻译程序。

编译程序从对源程序执行途径的角度不同，可分为解释执行和编译执行的翻译程序。

所谓解释执行是借助于解释程序完成，即按源程序语句运行时的动态结构，直接逐句地分析、翻译并执行。像自然语言翻译中的口译，随时进行翻译。解释执行的过程如图 1.8 所示。解释执行的优点是易于查错，在程序执行过程中可以修改程序。

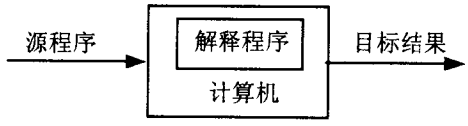


图 1.8 源程序的解释执行

所谓编译执行，是将源程序先翻译成一个等价的目标程序，然后再运行此目标程序，故编译执行分为编译阶段和运行阶段，如图 1.2 所示。

要注意源程序两种执行方式的区别，编译执行是由编译程序生成一个与源程序等价的目标程序，它可以完全取代源程序，目标程序可运行任意多次，不必依赖编译程序。正像自然语言翻译中的笔译，一次翻译可多次阅读。而解释执行不生成目标程序，对源程序的每次执行都伴随着重新翻译的工作，而且不能摆脱翻译程序。有些编译程序既支持解释执行又支持编译执行，在程序的开发和调试阶段用解释执行，一旦程序调试完毕，便采用编译执行。

按照编译程序的用途或侧重面，可以把编译程序分为：

优化型编译程序：这类编译程序针对为了产生高效率的机器代码，设计成多级别、多层次的优化供用户选择。优化型编译程序的代价是增加了编译程序的复杂性和编译时间。

交叉型编译程序：当一个编译程序在某种型号的目标机上运行，而生成另一种型号目标机上运行的目标程序时，称该编译程序为交叉型编译程序。

诊断型编译程序：此类编译程序是为了帮助用户开发和调试程序，它能检查、发现源程序中的错误，并能在一定程度上校正一些错误，它适合于在程序开发的初始阶段使用。

可重定位型编译程序：这里可重定位是指可重新定位目标代码的地址。通常的编译程序都是为某个特定的程序设计语言和特定的目标机设计的，编译生成的目标程序只能在特定的目标机上运行。当对于同一种程序设计语言为另一种不同型号的目标机配置编译程序时，原有的编译程序不再适用而需重新设计。当然仅重写与机器有关的部分而与机器无关部分不必重写。可重定位型编译程序是不重写此编译程序中机器无关部分就可以改变目标机的编译程序，它的可移植性好。

1.3　编译程序的结构与组织

1.3.1　编译程序的结构

编译程序是比较复杂、庞大的系统软件。它所涉及的处理对象—源语言程序，从通用语言到计算机应用的各个领域的专用语言有成百上千种，它所涉及的处理结果—目标程序，其形式既可以是另一种程序设计语言或特定目标表示，又可以是从微型计算机到超大型计算机的某种机器语言，其编译程序物理构造也可为一遍乃至多遍编译，甚至包括程序优化、连接、装配、调试等功能。现在比较流行，使用比较广泛的一些编译器，如 TURBO 系列，已不仅仅是一个语言翻译工具，更是一个庞大的集成开发环境。尽管编译程序的处理过程复杂，且不同的编译程序实现方法千差万别，构造原理各异，但任何编译程序的基本逻辑功能及必须完成的处理任务具有共同点。图 1.9 给出了基本的编译程序结构的经典表示，以说明编译程序的概貌与组成。

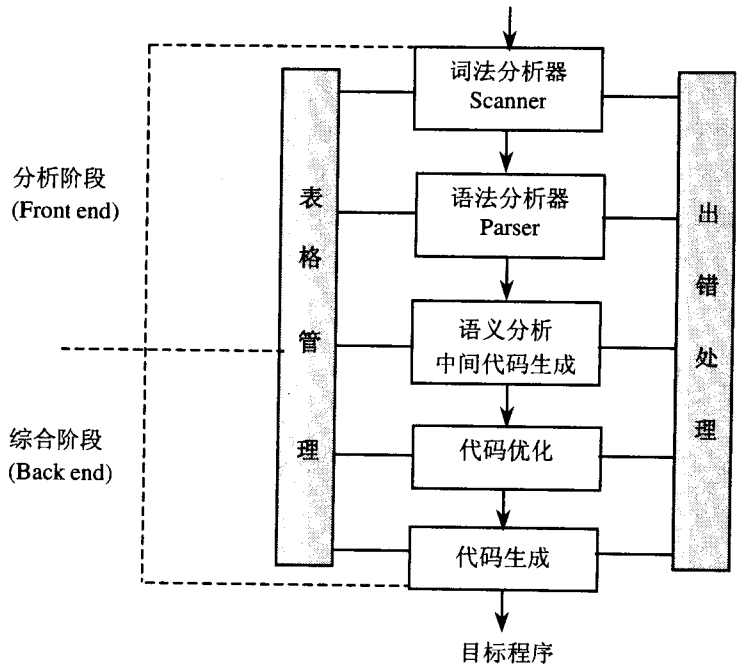


图 1.9　编译程序的逻辑结构

图 1.9 所示编译程序的逻辑结构图中间位置的纵向五个矩形框表示完成编译程序某种特定功能的模块,各模块间有密切的逻辑联系，显示了基本编译过程的五个阶段。图中两边的灰色矩形框是编译程序的辅助模块，可在编译的任何阶段被调用，辅助完成编译功能。

1. 词法分析 (lexical analysis)

词法分析阶段的任务是对输入的符号串形式的源程序进行最初的加工处理。它依次扫描读入的源程序中的每个字符，识别出源程序中有独立意义的源语言单词，用某种特定的数据结构对它的属性予以表示和标注。词法分析实际上是一种线性分析。例如，有如下 C 语句代

码行：

a[index] = 12 * 3 ；

经词法分析后将产生对 9 个单词识别的如下属性标注：

(1)	a	标识符
(2)	[	左方括号
(3)	index	标识符
(4)	]	右方括号
(5)	=	赋值
(6)	12	整常数
(7)	*	乘号
(8)	3	整常数
(9)	;	分号

当然，属性字的数据结构可据不同语言及编译程序实现方案来设计，但一般设计成单词属性标识及单词内码两个数据项。

2. 语法分析 (syntax analysis)

语法分析的任务是依据语言文本所限定的语法规则，对词法分析的结果进行语法检查，并识别出单词序列所对应的语法范畴，类似于自然语言中句子的语法分析。通常将语法分析的结果表示为分析树(paser tree)或语法树(syntax tree)，是一种层次结构的形式。例如，前面所述的 C 语言的赋值表达式语句经过语法分析，确认合乎 C 语言的语法规则且语法范畴为表达式语句，产生的语法树形式的语法分析结果如图 1.10 所示。

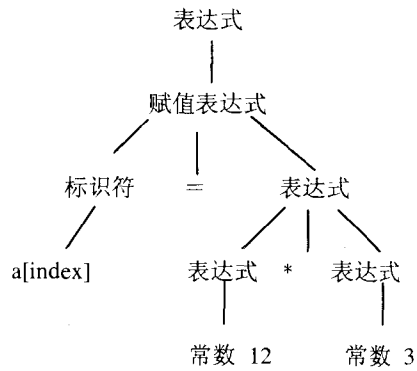


图 1.10 C 语句 a[index] = 12 * 3 ； 的语法树

3. 语义分析 (semantic analysis)与中间代码生成

语义分析阶段的任务是依据语言文本限定的语义规则对语法分析所识别的语法范畴进行语义检查和处理，并为生成目标代码收集类型等有关的信息，直至最后翻译成与其等价的某种中间代码或目标代码。语义分析是整个编译程序完成的最实质性的翻译任务。

4. 代码优化 (optimization)

代码优化是为了改进目标代码的质量而在编译过程中进行的工作。代码优化可以在中间

代码或目标代码级上进行，其实质是在不改变源程序语义的基础上对其进行加工变换，以期获得更高效的目标代码。而高效一般是对所产生的目标程序在运行时间的缩短和存贮空间的节省而言。

在前述的例子中，C 语句的中间代码经实现常量合并这类优化后，结果如下：

```
a[index] = 6
```

5. 目标代码生成 (code generator)

这个阶段的任务是，根据中间代码及编译过程中产生的各种表格的有关信息，最终生成所期望的目标代码程序。一般为特定机器的机器语言代码或汇编语言代码，这个阶段实现了最后的翻译工作，处理是繁琐的，需要充分考虑计算机硬件和软件所提供的资源，以生成较高质量的目标程序。

例如，对上面的示例在代码生成时，要考虑怎样存储整型数来为数组元素的引用生成目标代码，以假设的汇编语言描述如下：

```
MOV     R0,    index      ;; 索引值赋给寄存器 R0
MUL     R0,    2          ;; 存储按字节编址，整型数占 2 个字节
MOV     R1,    &a         ;; &a 表示数组 a 的基地址
ADD     R1,    R0         ;; 计算 a[index]的地址
MOV     *R1,   6          ;; a[index] = 6
```

上述编译过程的五个阶段，仅是对典型的编译程序在逻辑功能上共性的提炼，而实际上，对具体的编译程序，逻辑关系是多种多样的，有些阶段的工作可以组合、交叉和分割，甚至缺省，因此构成具有完全不同逻辑结构的各类编译程序。由于编译器的结构对其可靠性、有效性、可用性以及可维护性都有较大的影响，因此有必要更多地了解有关编译器结构的各种观点。图 1.9 中还从其它不同角度来观察编译器的结构。

6. 分析和综合

将编译过程分为分析和综合两个阶段的观点认为，对源程序仅进行结构分析和语义分析的操作看做是分析部分，而将生成翻译代码的操作看做是综合部分。

7. 前端和后端

这种观点按照编译器是依赖于对源语言的操作还是依赖于对目标语言的操作，将其分为前端和后端两部分。这与将编译器分为分析和综合两部分是一致的。前端进行分析，完成词法分析、语法分析与语义分析，一般与机器无关，因此适用于自动生成。后端进行综合，完成目标代码的生成与优化，一般是与机器相关的。如果在理想情况下，将编译器严格分为这两个部分，则中间表示就是其间的接口。这样结构的编译器对编译器的可移植性很重要。

1.3.2　编译程序结构的公共功能与编译程序的组织

编译程序将源程序翻译为目标程序的基本过程，是对源程序进行分析和加工的过程，这个过程除了如前所述的五个基本阶段以外，任何规模不同、结构各异的编译程序都还有两部分与编译各阶段都有密切联系的功能，即表格管理和出错处理。另外，编译的遍(pass)也是编译程序组成中的一个重要概念。

1. 表格与表格管理