高等学校21世纪计算机教材

编译原理

陈应明　马俊杰　张怀庆　编著

冶金工业出版社

高等学校21世纪计算机教材

编译原理

陈应明　马俊杰　张怀庆　编著

北　京

冶金工业出版社

2004

内 容 简 介

本书全面地介绍了编译程序的基本结构，系统地阐述了编译原理的一般理论和常用的方法和技术。全书共分为9章，包括编译简介、简单的一遍编译器、词法分析、语法分析、类型检查、语法制导翻译与中间代码生成、运行时的环境、代码生成、代码优化等相关内容。在内容的组织上，本书将编译原理的基本理论和具体的实现技术有机地结合起来，既准确清楚地阐述了相关的概念和原理，又给出了典型的实现程序流程图。

本书理论和实践并重，叙述严谨、简明，富有启发性，且内容深入浅出，便于自学。本书不仅可以作为高等院校相关专业的教材，也可以作为计算机专业人员的参考用书。

图书在版编目（CIP）数据

编译原理 / 陈应明等编著. —北京：冶金工业出版社，2004.9
ISBN 7-5024-3612-X

I. 编… II. 陈… III. 编译程序—程序设计 IV. TP314

中国版本图书馆 CIP 数据核字（2004）第 086877 号

出版人　曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）
责任编辑　程志宏
湛江蓝星南华印务公司印刷；冶金工业出版社发行；各地新华书店经销
2004 年 9 月第 1 版，2004 年 9 月第 1 次印刷
787mm × 1092mm　1/16；　20 印张；　459 千字；　310 页；　1-3500 册
30.00 元
冶金工业出版社发行部　**电话**：（010）64044283　**传真**：（010）64027893
冶金书店　**地址**：**北京东四西大街 46 号**（100711）　**电话**：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于编译原理

编译原理是计算机专业一门重要的专业课程，在计算机专业的学科课程中占有非常重要的地位，它是每个优秀的计算机专业人员必修的一门课程。设置本课程的目的在于系统地向学生讲述编译程序设计的基本理论、编译系统的结构及编译程序各部分的设计原理和实现技术。通过对这些知识的学习，使学生既能掌握编译理论和编译方法等方面的基本知识，又具有设计、实现、分析和维护编译程序等方面的能力。

编译原理的学习可以划分侧重点，比如编译原理理论、编译原理技术及编译原理工具等，读者在学习的过程中要确定自己学习的重点。对于高等学校大专学生来说，掌握编译原理技术和掌握编译原理的工具要比研究编译原理的理论有用得多，当然，如果真的对编译原理理论很感兴趣，也可以深入研究该理论。

掌握编译原理的技术，关键在于把学到的知识用于实践之中。本书是作者根据《编译原理大纲》的要求，结合多年的教学经验编写而成的。本书部分章节配有丰富的代码，读者在学习的过程中可以结合里面的代码进行相关操作，以巩固所学知识。

二、本书内容结构

全书分为 9 章，各章内容如下：

第 1 章：编译简介。主要介绍了编译器及其各个阶段和伙伴、各阶段的分组及构造工具等内容。

第 2 章：简单的一遍编译器。主要介绍了简单的一遍编译器的相关知识、语法定义、语法制导翻译、语法分析、简单表达式的翻译器、词法分析、符号表、抽象堆栈机等内容。

第 3 章：词法分析。主要介绍了词法分析器的作用、记号的描述、单词符号的识别、有穷自动机及词法分析器描述语言等内容。

第 4 章：语法分析。主要介绍了语法分析器的作用、上下文无关文法、文法的编写、自顶向下语法分析、自底向上语法分析、算符优先分析法、LR 语法分析器、二义文法的应用以及语法分析器的生成器等内容。

第 5 章：类型检查。主要介绍了类型系统、一个简单的类型检查器的说明、类型表达式的等价、类型转换、函数和运算符的重载、多态函数以及合一算法等内容。

第 6 章：语法制导翻译与中间代码生成。主要介绍了语法制导定义、S 属性的自下而上计算、L 属性定义、自上而下翻译、继承属性的自下而上计算、递归计算、语法制导定义的分析、中间语言、声明语句、赋值语句、布尔表达式以及 case 语句等内容。

第 7 章：运行时的环境。主要介绍了源语言问题、存储组织、存储分配策略、访问非局部名字以及参数传递等内容。

第 8 章：代码生成。主要介绍了代码生成器设计中的问题、目标机器、基本块和流图、下次引用信息以及一个简单的代码生成器等内容。

第 9 章：代码优化。主要包括了优化技术的相关知识、局部优化、循环优化、数据流分析与全局优化等内容。

三、本书特点

本书简明易懂、条理清晰，用简练的语言结合大量的图、表详细介绍了编译原理的基本原理和相关知识。在编写过程中，作者力图将其中的基本概念、基本编译技术和实现方法的思路阐述清楚，同时书中还穿插了大量的实例。本书各章都配有一定数量的习题，以帮助读者巩固所学内容。此外，本书最后还附有参考答案，以方便读者对照课后的习题进行练习。

四、本书适用对象

本书适用范围较广，既可作为高等院校相关专业的教材，也可作为计算机专业人员的参考用书。

本书由陈应明、马俊杰、张怀庆编写。

由于时间仓促，水平有限，书中难免会出现一些错误和纰漏，欢迎广大读者批评指正。

虽然经过严格的审核、精细的编辑，本书在质量上有了一定的保障，但我们的目标是力求尽善尽美，欢迎广大读者和专家对我们的工作提出宝贵建议，联系方法如下：

电子邮件：service@cnbook.net

网址：www.cnbook.net

此外，**本书所附送的电子教案也可从该网站免费下载**，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2004 年 7 月

目　录

第 1 章　编译简介

编译程序是程序设计语言的支撑环境，是计算机系统的重要系统软件之一，在编译程序的支持下，高级语言程序才能运行。编译原理的发展相当迅速，现已基本形成了一套比较完整、系统的理论和方法。编写编译器的原理和技术具有非常普遍的意义，其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，其中有几种基本编译器的编写技术已经被用于构造许多计算机的多种语言翻译器。本章主要介绍编译器的体系结构、各个阶段、组织方式和构造工具。这些知识对学习本书其他章节的内容是必不可少的。

1.1　编译器

编译器就是一个程序，它读入用源语言编写的程序，然后将其翻译成一个与之等价的以目标语言编写的程序，如图 1-1 所示，并且在这个翻译过程中，编译器还能向用户报告被编译的源程序中出现的错误。

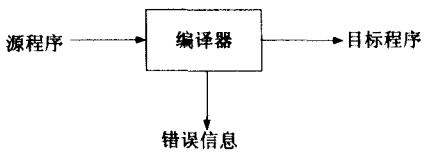

图 1-1　编译器

在 20 世纪 40 年代，由于冯·诺伊曼在存储—程序计算机方面的先锋作用，编写一串代码或程序已成为必要，这样计算机就可以执行所需的计算。开始时，这些程序都是用机器语言（machine language）编写的，机器语言就是表示机器实际操作的数字代码。由于编写这样的代码十分费时和乏味，这种代码形式很快就被汇编语言（assembly language）代替了。在汇编语言中，是以符号形式给出指令和存储地址的。汇编语言大大提高了编程的速度和准确度，人们至今仍在使用着它，在编码需要极快地速度和极高地简洁程度时尤为如此。但是，汇编语言也有许多缺点：编写起来不容易，阅读和理解很难；汇编语言的编写严格依赖于特定的机器，所以为一台计算机编写的代码在应用于另一台计算机时必须完全重写。很明显，发展编程技术的下一个重要步骤就是以一个更类似于数学定义或自然语言的简洁形式来编写程序的操作，它应与任何机器都无关，而且也可由一个程序翻译为可执行的代码。

在 1954 年至 1957 年期间，IBM 的 John Backus 带领的一个研究小组对 FORTRAN 语言及其编译器的开发，使得汇编语言代码可以写成一个简洁的与机器无关的形式。但是，由于当时处理中所涉及到的大多数程序设计语言的翻译并不为人所掌握，所以这个项目的成功来之不易。

编译器设计最近的发展包括：首先，编译器包括了更为复杂的算法的应用程序，它用于推断和简化程序中的信息；这又与更为复杂的程序设计语言的发展结合在一起。其次，编译器已越来越成为基于窗口的交互开发环境（interactive development environment，IDE）

的一部分，它包括了编辑器、链接程序、调试程序以及项目管理程序。这样的 IDE 的标准并没有多少，但是已沿着这一方向对标准的窗口环境进行开发了。尽管近年来对此进行了大量的研究，但是基本的编译器设计在近 20 年中都没有多大的改变，而且它们正迅速地成为计算机科学课程中的中心一环。

1.1.1 编译的分析—综合模型

为一个语言编制编译程序是一个相当复杂的任务，编译过程的复杂性在很大程度上取决于源语言，编译程序必须完成两个主要任务：

（1）分析。分析部分将源程序分解为源程序的各个基本组成部分，揭示源程序的结构和基本数据类型、数据结构，决定它们的语义，并建立源程序的中间形式表示。

（2）综合。综合部分接收分析部分产生的中间形式表示，并根据具体的目标计算机系统生成目标代码并对代码进行优化处理。这部分需要大量的专门化技术。

在分析期间，源程序所蕴含的操作将被确定下来并可以用一个称为语法树的分层结构来表示。语法树的每个结点表示一个操作，该结点的子结点表示这个操作的参数。一个赋值语句的语法树如图 1-2 所示。

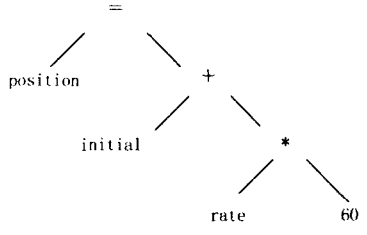

图 1-2 position = initial + rate*60 的语法树

许多操纵源程序的软件工具都首先完成某种类型的分析。下面是这类软件工具的示例：

（1）结构编辑器。结构编辑器将一个命令序列作为输入来构造一个源程序，不仅实现普通的文本编辑器的文本创建和修改功能，而且还对程序文本进行分析，为源程序构造恰当的层次结构。结构编辑器是引导用户在语言的语法制导下编制程序，能自动地提供与其相匹配的关键字。如 if 后必须有 then、begin 和 end 的配对，左括号和右括号的配对等，这样可以减少语法上的错误，可加快对源程序的调试，提高效率和质量。这类结构编辑器的输出常常类似于一个编译器的分析阶段的输出。

（2）智能打印机。智能打印机能够对程序进行分析，打印出结构清晰的程序。如：关键字、注释以一种特殊的字体打印；根据各个语句在程序的层次结构中的嵌套深度来缩排这些语句。

（3）静态检查器。静态检查器读入一个程序，分析这个程序，并在不运行这个程序的条件下试图发现程序的潜在错误。静态检查器可以查出变量在被定义以前使用，也可以查出源程序中永远不能被执行的语句，更进一步还能捕获诸如将实变量用作指针这种类型的逻辑错误。

（4）解释器。解释器不是通过翻译来产生目标程序，而是直接执行源程序中蕴含的操作。例如，对于一个赋值语句，解释器为之建立一个类似于如图 1-2 所示的树，然后通

过遍历这棵树来执行结点上的操作。在根结点，解释器会发现它有一个赋值操作要完成，它便调用表达式计算例程去计算赋值操作右端的表达式，然后将结果存放到与标识符 position 相关联的地址。在根结点的右子结点处，表达式计算例程将发现它要计算两个表达式的和。表达式的例程递归地调用它自身来计算表达式 rate*60 的值，然后将这个值加到变量 initial 的值上。

由于命令语言中执行的每个操作通常都是对编辑器或编译器一类复杂例程的调用，解释器经常用于执行命令语言。类似地，一些高级语言是解释执行的，因为有许多关于数据的信息（如数组的大小和形状）不能在编译时得到。

编译器一般被看成是使用 FORTRAN 等高级语言编写的源程序翻译成汇编语言或者某种计算机的机器语言的程序。但是在许多与语言翻译毫不相关的场合，编译技术常常被使用。下面各个例子中的分析部分都与传统观念中的编译器的分析部分相似。

（1）文本格式器（text formatter）。文本格式器的输入是一个字符流。输入字符流中的多数字符串是需要排版输出的字符串，同时字符流中也包含一些用来说明字符流的段落、图表或者上标和下标等数学结构的命令。

（2）硅编辑器（silicon compiler）。硅编辑器的输入是一个源程序，这个源程序的程序设计语言类似于传统的程序设计语言。但是该语言中的变量不是内存中的地址，而是开关电路中的逻辑符号（0 或 1）或符号组。硅编辑器的输出是一个以适当语言书写的电路设计。

（3）查询解释器（query interpreter）。查询解释器把含有关系和布尔运算的谓词翻译成数据库命令，在数据库查询满足该谓词的记录。

1.1.2 编译器的前驱与后继

为了产生可执行的目标程序，除了编译器是必需的以外，还需要几个其他程序配合使用。一个源程序可以分成几个模块并存储在不同的文件中。有时候将这些源程序连接在一起的任务交给称为预处理器的程序来完成，该预处理器还可将宏扩展为源程序的语句。

如图 1-3 所示是典型的对源程序进行处理的全过程。

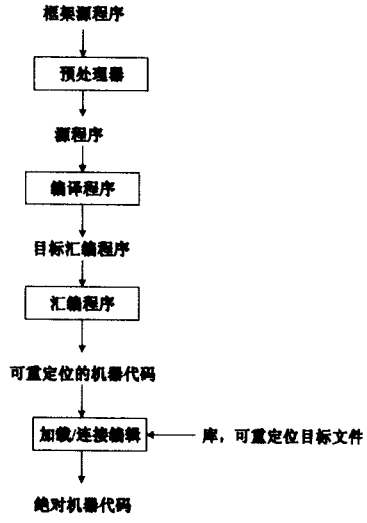

图 1-3 语言处理系统

另外，由编译程序生成的目标程序在其运行以前还必须做进一步的处理。图 1-3 中的编译程序生成汇编代码，该代码由汇编程序翻译成机器代码，然后与库程序连接一起形成可以在机器上运行的代码。

1.2 编译器的各阶段

编译器内部包括了许多步骤或阶段（phase），它们执行不同的逻辑操作。将这些阶段设想为编译器中一个个单独的片断是很有用的，尽管在应用中它们是经常组合在一起的，但它们确实是作为单独的代码操作来编写的。编译器的一个典型的阶段划分如图 1-4 所示。

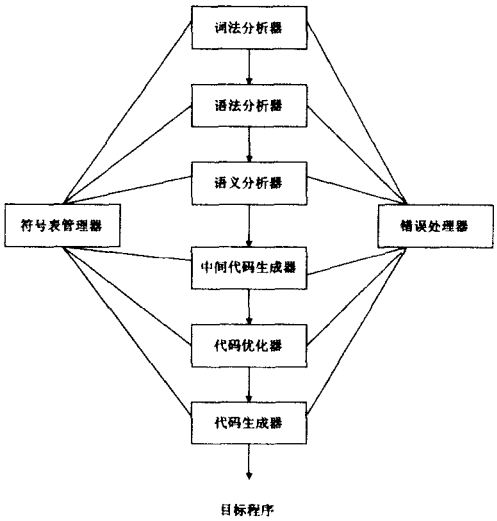

图 1-4 编译器的各阶段

图 1-4 中前三个阶段构成了编译器的分析部分。图中的符号表管理和错误处理器是编译器的六个阶段（词法分析、语法分析、语义分析、中间代码生成、代码优化和代码生成）都要涉及的两项活动。

1.2.1 词法分析

一个源程序实际上是由字母、数字、运算符、分隔符和某些专用符号组成的字符串。一个语言的源程序包含了许多该语言的基本的语言结构成份，如关键字、变量名、标号名、常数、运算符和分隔符等等，所以编译程序首先要识别出这些不同类型的基本语言结构成份。词法分析就是从左到右地读组成源程序的字符流，并分离、识别出一个个记号（Token），Token 是有确定含义的字符序列。词法分析也叫做线性分析。例如：

```
a [index] = 4 + 2
```

这个代码包括了 12 个非空字符，但只有 8 个记号：

（1）标识符 a。

（2）左括号[。

（3）标识符 index。

（4）右括号]。

（5）赋值=。

（6）数字 4。

（7）加号+。

（8）数字 2。

每一个记号均由一个或多个字符组成，在进一步处理之前它已被收集在一个单元中。

扫描程序还可完成与识别记号一起执行的其他操作。例如，它可将标识符输入到符号表中，将文字（literal）输入到文字表中（文字包括诸如 3.1415926535 的数字常量以及诸如“Hello world!”的引用字符串）。

1.2.2 语法分析

语法分析程序从词法分析程序取得源程序（记号流，即单词串），并将源程序的记号分组形成语法短语。语法分析所遵循的是语言的语法规则。语法分析也叫做层次分析。源程序的语法短语常用分析树（Parse tree）或语法树（Syntax tree）表示。

程序的层次结构通常使用递归的语法规则表示。例如，表达式的语法规则可以定义为：

（1）标识符是一个表达式。

（2）数是一个表达式。

（3）如果 e1 和 e2 都是一个表达式，则：e1+e2、e1*e2、e1/e2、(e1)也都是表达式。

类似地，赋值语句可定义为：

```
ID = e;
```

注：分号（;）是语句的结束标记。其中，ID 是标识符，e 是表达式。

这样，根据上述定义的语法规则，语法分析器为赋值语句：

```
p = x0+v*60;
```

产生如图 1-5 所示的分析树。

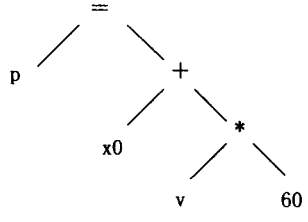

图 1-5 分析树

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和数字串，这只要按顺序扫描输入流，遇到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而构成单词标识符。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。

1.2.3 语义分析

语义分析是使用语法分析器产生的分析树／语法树来确定源程序的意义（语义），同时审查源程序有无语义错误，为代码生成阶段收集类型信息。

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。例如检查每个运算符的运算对象的类型，

判断其是否合法。例如，对于表达式 v*60，语义分析器检查符号表中变量 v 的类型，若 v 是整型变量，则表达式 v*60 仍有效。倘若 v 是实型，则根据语言是否允许混合类型运算，如果允许，应将整型数 60 转换为实数（60.0），然后参加运算；否则报告类型不一致的错误信息。

分析树描述了输入的语法结构，为了节省存储空间，大多编译器使用语法树（Syntax tree）作为其内部表示。语法树是分析树的压缩形式表示，其中算符作为内部结点（非叶子结点）出现，它的运算对象作为它的子结点。如图 1-6 所示显示了语义分析插入类型转换的 p = x0+v*60 的语法树。

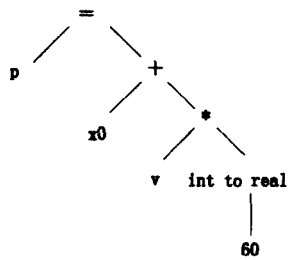

图 1-6 语义分析中插入了一个整数到实数的转换

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。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，C++语句 p = x + v*60 可以翻译成下列四元式序列或三地址语句序列：

```
(*,   v,   60,   temp1)          ;kmp1 = v*60
(+,   x,   temp1,   temp2)       ;kmp2 = x+temp1
(=,   temp2,   0,   p)           ;p = temp2
```

将各类语法范畴翻译成中间代码所依据的是语言的语义规则。一般而言，中间代码是一种独立于具体硬件的符号系统。除四元式外，常用的中间代码形式还有三元式、间接三元式和逆波兰表达式等。

1.2.5 代码优化

此阶段的任务是对前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。有些编译器几乎没有进行代码优化。

例如，下列赋值语句：

```
ix = j*k+16
iy = j*k+26*l-m*l
```

j*k 是公共子表达式，可以只计算一次。经过优化以后产生下列三地址代码：

```
temp1 = j*k
ix = temp1+16
```