高等学校教材

编译原理与实现

BIANYI YUANLI YU SHIXIAN

金成植 编著

高等教育出版社

高等学校教材

编译原理与实现

金成植　编著

高等教育出版社

图书在版编目（CIP）数据

编译原理与实现/金成植编著．—北京：高等教育出版社，1989.3（2001 重印）

ISBN 7-04-002083-1

Ⅰ．编… Ⅱ．金… Ⅲ．编译程序-高等学校-教材 Ⅳ．TP314

中国版本图书馆 CIP 数据核字（95）第 13550 号

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街 55 号	**邮政编码**	100009
电　　话	010—64054588	**传　　真**	010—64014048
网　　址	http://www.hep.edu.cn		
经　　销	新华书店北京发行所		
印　　刷	成都新华印刷厂		
开　　本	850×1168　1/32	**版　　次**	1989 年 3 月第 1 版
印　　张	7.625	**印　　次**	2001 年 5 月第 13 次印刷
字　　数	181 000	**定　　价**	8.10 元
插　　页	2		

凡购买高等教育出版社图书，如有缺页、倒页、脱页等质量问题，请在所购图书销售部门联系调换。

内 容 提 要

本书是继1984年出版的《编译方法》之后，作者以PASCAL语言为模型编写的教材。本书系统、完整地讲述了编译程序的构造原理及其实现方法。在书的取材及结构安排方面，注重了内容的易懂性和完整性。主要内容有：编译原理的基本概念、自动机与词法分析、形式语言与语法分析、中间代码生成与优化以及目标代码生成等。并附有一定数量的习题，帮助读者理解教材的内容。

本书可作为高等学校计算机软件专业及有关专业的教材，也可供有关科技人员阅读参考。

序　言

现代计算机系统都配有 FORTRAN, PASCAL 等语言的编译系统。只有高级语言而没有相应的编译系统，则高级语言就无法在计算机上使用。因此，编译系统是现代计算机系统的重要组成部分。

不同编译程序采用的技术不尽相同，但基本原理是一致的。本教材将以 PASCAL 语言的子语言为模型，系统而完整地介绍编译程序的构造原理及其实现方法。

本教材始终把易懂性和完整性放在第一位，为此特别注意到了各部分深度的一致性和各部分之间的衔接问题。有时为了增强易读性在算法的效率方面做了些让步。

本教材配有一定数量的习题。这些题对于教授内容的理解会有很大帮助，因此建议读者能完成大部分的习题。

由于篇幅所限，本教材舍去了部分内容，如输入输出语句的处理、数组和记录类型外的其它结构类型的处理、WHILE 循环外的其它循环语句以及语法和语义错误的处理等。

由于作者水平所限，加上绝大部分算法是新设计的，因而估计有不少缺点和错误，殷切希望广大读者批评指正。

本书由清华大学计算机系郑人杰副教授审阅，并提出许多宝贵意见。在此表示衷心的感谢。

作　者

1988.4

目　录

第一章　编译程序概述

1.1 什么叫编译程序

程序设计语言分为两大类：**低级语言**和**高级语言**．低级语言又可分为两类：**机器语言**和**汇编语言**．机器语言是计算机的指令系统．而汇编语言是符号化的指令系统。这两种语言都依赖于计算机，使用起来既烦琐又容易出错，易读性差、没有通用性，因此相继出现了很多种高级语言．常见的高级语言有 BASIC，FORTRAN，PASCAL，ALGOL，COBOL 等．

计算机硬件只懂自己的指令系统，因此想用汇编语言或高级语言算题，就必须有这样一个程序，它把用汇编语言或高级语言写成的程序转换成机器语言程序，我们把这种程序统称为**翻译程序**（Translator），把汇编语言的翻译程序称为**汇编程序**（Assembler），把高级语言的翻译程序称为**编译程序**（Compiler）．

编译程序的输入对象称为**源程序**（Source program），输出对象称为**目标程序**（Object program）．目标程序也可以是汇编语言程序．

在计算机上执行一个高级语言程序一般要分为两步：第一步是通过编译程序把源程序翻译成机器语言程序；第二步是执行目标程序．编译程序在进行翻译时，做语法检查和语义检查，凡是有错误的源程序均指出错误．

高级语言程序的处理也可采用另一种方式，即它并不把源程序翻译成机器语言程序然后再执行该机器语言程序，而是采用边翻译边执行的解释执行方法．这种处理程序称为**解释程序**（Interpreter）．解释程序的工作结果是源程序的执行结果而不是目标程

序.

编译时间加目标程序的执行时间比解释执行源程序的时间要短得多，因此通常都采用编译方法而不采用解释方法.

采用编译方式和解释方式的解题过程如图 1.1 所示。

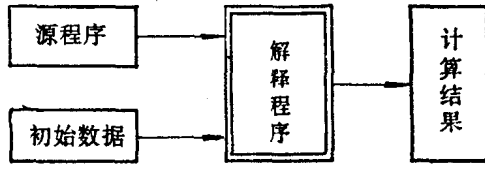


图 1.1 (a) 解释方式

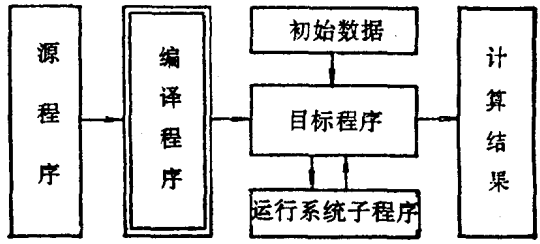


图 1.1 (b) 编译方式

在目标程序运行时还要用到一些子程序，如 sin，cos 及其它共用的子程序，称这些程序为**运行系统子程序**.

1.2 编译程序的组成部分

不同的编译程序都有自己的组织和工作方式，它们都是根据源语言的具体持点和对目标程序的具体要求设计出来的，因此很难给出编译程序的标准结构，也不好说哪种结构的好，哪种结构的不好。但就编译程序所做的工作内容来看是基本相同的.

比较大的编译程序要完成以下工作：词法分析，语法分析，

中间代码生成，中间代码优化，目标代码生成，表格管理，错误处理.它们之间的关系如图 1.2 所示．其中除了中间代码部分外，其它所有工作都是任何编译程序所要完成的工作．但完成的方式可能不同，有的可能把几个工作合起来交错完成的．不过从原理的角度来说，图 1.2 所示的结构是比较理想的逻辑结构，它具有一般性

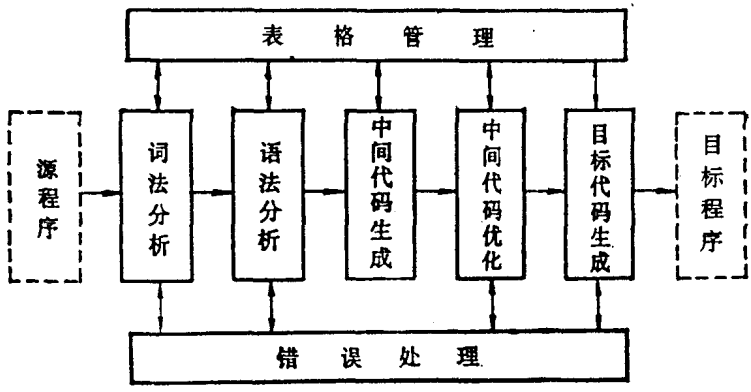


图 1.2 编译程序逻辑图

各部分的主要任务：

• 词法分析：扫描源程序的 ASCII 码序列，拼出每一个单词．并把每个单词的 ASCII 码序列替换为所谓的机内表示 TOKEN形式.这时还检查词法错误．词法分析阶段不依靠语法关系．

• 语法分析：扫描对象可能是源程序的 ASCII 码序列，也可能是词法分析后的 TOKEN 序列．前者情形，词法分析程序作为语法分析程序的子程序．语法分析的主要任务是检查源程序的形式语法错误.每当发现错误时将输出有关信息.很多编译程序在进行语法分析时同时完成其它工作．但要注意到如果语法有错误那么其它工作也就白做.

• 中间代码生成：扫描对象通常是语法分析后的结果．这一

部分把源程序的 TOKEN 序列转换成更接近于目标代码的中间代码三元式或四元式的序列．这时要完成一些语义检查工作．如果不搞优化，那么这部分的工作是可不要的．

·中间代码优化：扫描对象是中间代码，任务是把原中间代码转换成可产生高质量目标代码的中间代码．其中的优化工作包括常表达式优化、公共子表达式优化、不变表达式外提和削减运算强度等．

·目标代码生成：扫描对象是中间代码，任务是从中间代码产生目标代码．这一部分的工作与目标机紧密相关，其它部分的工作几乎与目标机无关．

·错误处理：错误包括词法错误、语法错误、静态语义错误、动态语义错误．其中动态语义错误只能在运行目标程序时才能发现．在编译程序的各个阶段都要有错误处理部分．词法错误和语法错误都集中一次完成检查，而语义检查则分散在以后的各个阶段在完成别的工作时顺便完成．

·表格管理：较大的编译程序用到很多表格，甚至可达几十种表．这些表将会占用大量存储区．因此，合理设计和使用表格是编译程序的一个重要问题．表格的分类、表格的结构、表格的处理都是所要考虑的基本问题．有些表格以后可能无用，这时应及时删除它们以扩大空区．如果处理不好表格的管理工作，就可能出现因表区溢出而不得不停止编译的现象，但这时可能还有空区．为了合理的管理表格(构造、查找、更新)和表区，不少编译程序都设立一些专门子程序(称为**表格管理程序**)，它们专门负责管理表格．

1.3 编译程序的分遍

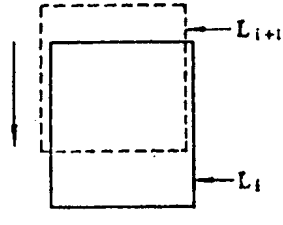
编译程序按其扫描遍数分为**一遍扫描**和**多遍扫描**的．如果通过对源程序的扫描直接生成目标代码，则说编译程序是单遍的．

多遍扫描的好处是算法清晰，便于分工，便于优化．但在前后扫描之间难免做一些重复性工作．在小机器上搞较大编译程序时最好采用多遍扫描法．对多遍编译程序来说，每遍的输出结果为下一遍的输入对象．

目前大部分编译程序都是多遍的．设 $COMP_i$ 是第 i 遍扫描程序，L_{i-1}是 $COMP_i$ 的扫描对象，L_i 是 $COMP_i$的扫描结果，则我们有

$$COMP_1(L_0)\rightarrow COMP_2(L_1)\rightarrow\cdots\cdots\rightarrow COMP_n(L_{n-1})=L_n$$

其中 L_0 是源程序，L_n 是目标程序，$L_1,L_2,\cdots,L_{n-1}$是中间语言代码．中间代码在被加工后不再需要保留，因此后一中间代码 L_{i+1}可以覆盖 L_i 的被加工部分．具体图示如下：



如果内存小需要用外存时可组织如下．首先在内存中开辟一个编译程序区 COMP．其大小可取 $COMP_i$(i=1,2,…,n)的最大值，即 COMP 区中可存放每遍的扫描程序．另外再开辟扫描对象区 L 和部分结果区 L′，最后在外存中开辟扫描结果区 L″，每当 L′区满时，将 L′的内容送到 L″区中．这时用到总控程序，它负责完成各种调度．可图示如下(图 1.3)．

本教材将按比较标准的分遍结构介绍，具体分遍情况如下：

第一遍：词法分析．

第二遍：语法分析．

第三遍：标识符和符号表处理．

第四遍：中间代码生成．

第五遍：中间代码优化．

第六遍：目标代码生成．

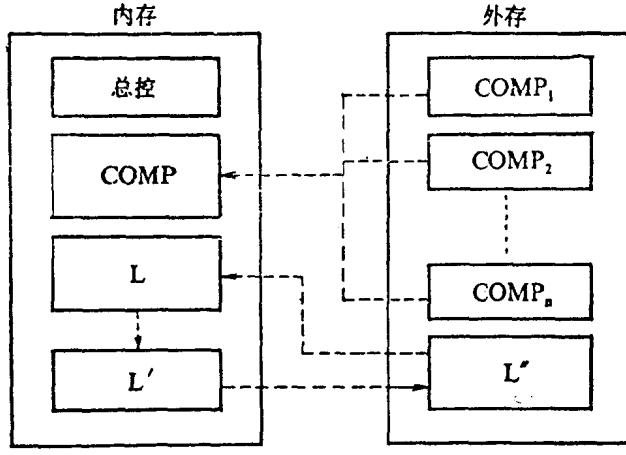


图 1.3

1.4 编译程序的开发

编译程序是一个相当复杂的系统程序，通常有上万甚至几万条指令．随着编译技术的发展，编译程序的生成周期也在逐渐缩短．但仍然需要很多人年，而且工作很艰巨，正确性也不易保证．

开发一个编译程序，首先对语言的语法与语义要有准确无误的理解，否则编译程序中将反映出一些错误．其次要确定对编译程序的要求，如搞不搞优化？搞优化搞到哪一级？等等．同时对目标机也要有较深入的研究，否则将会生成质量不高的目标代码．再就是根据编译程序的规模要确定编译程序的具体分遍及每遍的具体任务．之后，设计各遍扫描程序的算法并加以实现．在完成编码之后，首先要分别调试各次的扫描程序，然后再连起来调试．编译程序的调试是比较头痛的事情，只能一个例子一个例子地进

行调试．调试工作将会占去很多时间，因为要改动一个地方可能牵涉到各部分，甚至可能由于原框架不合适要推倒重来．由于编译程序的复杂性，最好采用模块化方法，以保证编译程序的良好结构．在调试当中随着编译程序的修改，也要进行对其它资料的修改，以随时保证各种资料间的一致性．

我们的愿望是尽可能多地把编译程序的生成工作交给计算机去完成．这就是编译程序自动化或自动生成编译程序的问题．目前的主要方法是在实现各遍算法时，采用高级语言．用高级语言可以避免一些细节，从而可减少错误并能加快速度．但各遍的算法毕竟还是要人来设计的，而它是整个编译程序的最难的部分．

计算机科学家们，为了实现编译程序的自动生成做了大量的工作．早有词法分析程序的自动生成系统和语法分析程序的自动生成系统，其中最出名的是 YACC 系统．它只要给出 LALR 文法的有关信息，就自动生成 LALR 分析表．它能自动处理二义性问题．编译程序自动生成的最关键地方是语义处理问题．语义处理的自动化与语义描述的形式化工作直接有关．近十年来形式语义学的研究取得了巨大的进展，它大大推动了编译程序的自动生成研究工作．已出现了一些编译程序的自动生成系统，其中比较著名的系统是：

GAG(Generator based on Attributed Grammars)

HLP (Helsinki Language Processor)

SIS (Semantics Implementation System)

CGSG (Compiler Generator for Semantics Grammars)

这里 GAG 和 HLP 是基于属性文法 (AG) 的系统，而 SIS 是基于指称语义的系统．CGSG 则基于语义文法 (SG) 的系统．形式语义学和编译技术的发展已能实现编译程序的自动生成．目前的主要问题是时间效率、空间节省问题．一是由自动生成系统产生的编译程序，比“人工”产生的编译程序长，并且占用更多的空间．在

这一方面应该承认还有一个数量级的差距.比如，用人工方法生成的PASCAL编译程序大约占100KB，而由HLP系统生成的PASCAL编译程序则占310KB. 可见差距还是很大的.

第二章　自动机与词法分析

2.1 基 本 概 念

自动机理论是编译程序词法分析的理论基础．每个程序设计语言都有它的字符集．字符用来构造单词（如 begin，end，x，100，+），单词则用来构造〈表达式〉、〈说明〉、〈语句〉等复合对象．其中单词是最小的语义单位，它不包含任何子结构，因此每个单词只是简单的字符序列．

任何程序都是某一基本符号集上的字符序列．换言之，我们的处理对象是字符序列．自动机、正则表达式正是描述程序设计语言中单词结构的强有力工具，因此本章将介绍有关自动机和正则表达式的基础知识．我们并不需要很深的自动机理论，但掌握其基础知识，对于编译程序的构造来说是极为有用的．

字母表(alphabet)：它是非空有穷集．如

$$\sum=\{a,b,c\}.$$

符　号(symbol)：字母表中的元素称为符号．

符号串(string)：符号的有穷序列，如 a，ab，cba 等都是上述字母表 $\sum$ 上的符号串．符号串也称为**字**．特别用 ε 来表示空符号串(什么符号也没有)．

长度 (length)：符号串的长度是指该串所包含的符号个数．用 $|x|$ 表示符号串 x 的长度，如

$$|a|=1,\ |ab|=2,\ |\varepsilon|=0$$

连结 (catenation)：设 x 和 y 为符号串，则称 xy 为它们的连结．例如，假设有 x=aa，y=bb，则 xy=aabb．特别，对任一符号串 β 有