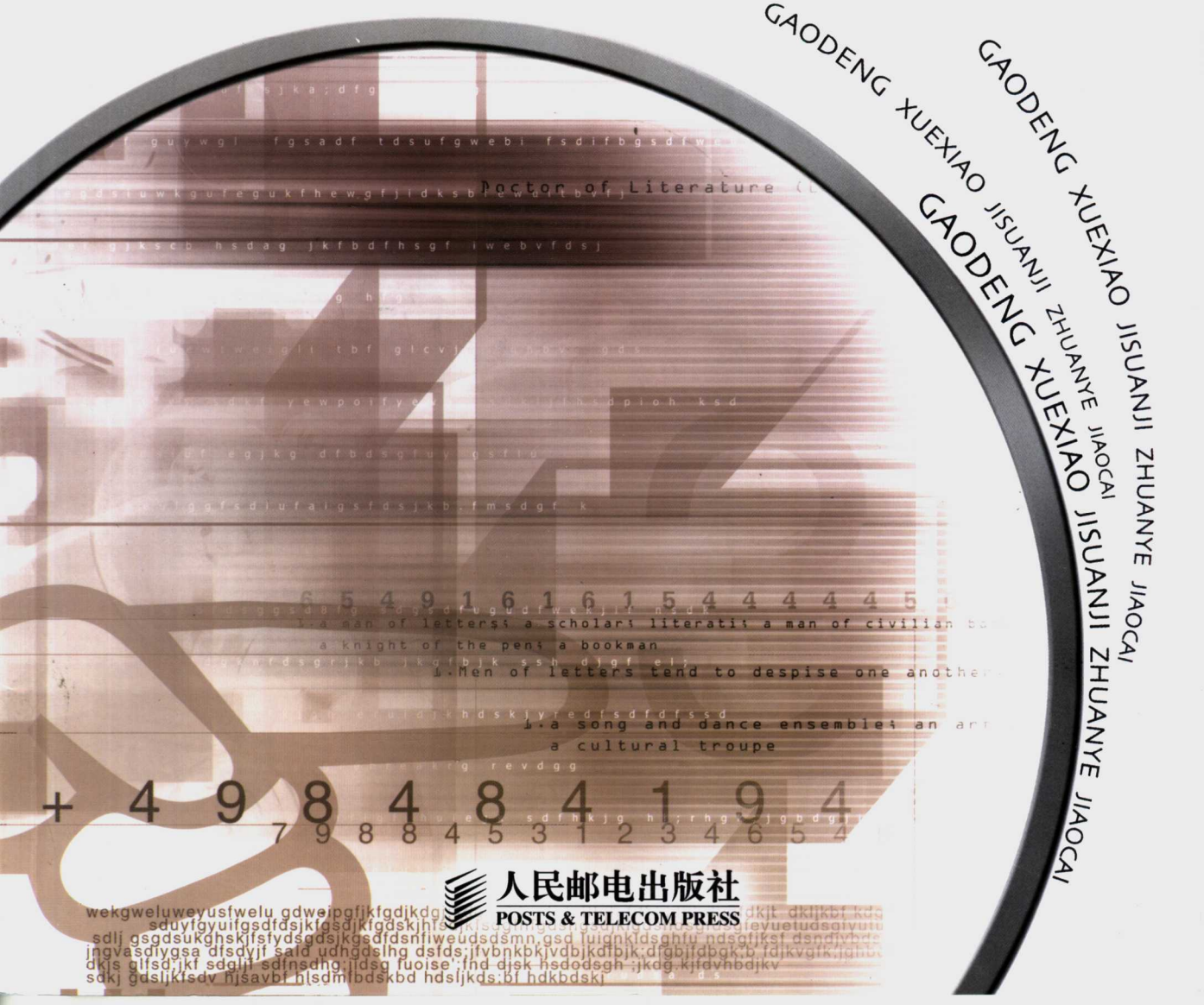
高等学校计算机专业教材

GAODENG XUEXIAO JISUANJI ZHUANYE JIAOCAI

编译原理

◉ 李冬梅 施海虎 编著



人民邮电出版社
POSTS & TELECOM PRESS

高等学校计算机专业教材

编译原理

李冬梅　施海虎　编著

人民邮电出版社

图书在版编目（CIP）数据

编译原理 / 李冬梅，施海虎编著. —北京：人民邮电出版社，2006.8
高等学校计算机专业教材

ISBN 7-115-14465-6

Ⅰ. 编… Ⅱ. ①李…②施… Ⅲ. 编译程序—程序设计—高等学校—教材 Ⅳ. TP314

中国版本图书馆 CIP 数据核字（2006）第 059403 号

内 容 提 要

本书系统全面地介绍编译程序的构造原理和实现技术，主要内容包括：形式语言的基本知识、词法分析、语法分析、语义分析与中间代码生成、符号表管理和错误处理、运行时的存储组织与分配、代码优化和目标代码生成等。在介绍编译原理和方法的同时，提供了一个小型编译程序——PL/0 编译程序的具体实现过程，并对常用分析器的自动生成工具（LEX 和 YACC）的功能和使用方法做了详细的介绍，在附录中分别给出了 PL/0 编译程序的 C 语言版本和 LEX 与 YACC 版本，从而使理论与实践紧密结合。

本书系统性强，内容循序渐进，实例丰富。对算法的描述深入浅出，文字简练，通俗易懂。每章都配有各种类型的习题，便于教学也便于自学。

本书可作为高等院校计算机科学及相关专业的本科生教材，也可作为教师和计算机软件工程技术人员的参考书。

高等学校计算机专业教材
编 译 原 理

◆ 编　　著　李冬梅　施海虎
责任编辑　邹文波

◆ 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
北京艺辉印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：20.5
字数：495 千字　　2006 年 8 月第 1 版
印数：1 – 3 000 册　　2006 年 8 月北京第 1 次印刷

ISBN 7-115-14465-6/TP・5203

定价：27.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223

编 者 的 话

编译技术是计算机语言发展的支柱，也是计算机科学中发展最迅速、最成熟的一个分支。

通过学习编译程序的构造原理和技术，将有助于深刻理解和正确使用程序设计语言。编译内容涉及到计算机的组织结构、指令系统以及操作系统，掌握编译技术有助于加深对整个计算机系统的理解。另外，编译原理课程中所介绍的一些原理、方法和算法并不局限于编译，它们同样可用于其他软件的设计开发。

编译原理作为计算机专业的一门核心课程，在教学中占据十分重要的地位。由于其内容具有较强的理论性和实践性，学生在学习过程中普遍感到内容抽象、算法复杂、难于理解，在此，编者力图奉献一本可读性较强的教材，以便于编译原理的教学和自学。

和其他教材比较起来，本书具有以下特色。

（1）篇幅适中，本教材参考学时数为 60～80，非常适合作为高等院校计算机及相关专业的本科生教材使用。

（2）系统性强，条理清楚，文字简练，通俗易懂，便于自学。

（3）考虑到学生对 C 语言较为熟悉，书中主要算法、例题和习题均以 C 语言为背景。

（4）基本概念和算法是结合作者的理解与体会进行阐述的，易于理解和接受。对于较复杂的算法，深入浅出，先给出算法的分析和基本步骤，再给出相应的类 C 语言描述或流程图，最后用实例对算法进行验证。

（5）把编译原理和编译技术两方面有机地结合起来。本书在介绍编译程序的构造原理和方法的同时，还提供了一个 C 语言版本的小型编译程序的教学模型“PL/0 语言编译程序”，而此模型的实现过程贯穿于各个章节，它是使用每一章所讨论的技术进行开发的。这样使读者在掌握相关的编译原理之后，通过阅读理解和扩充此编译程序，加深对所学内容的理解，使所学内容融会贯通。

（6）对常用分析器的自动生成工具（LEX 和 YACC）的功能和使用方法做了详细的介绍，并给出了使用 LEX 和 YACC 实现的 PL/0 语言编译器的源程序，从而使读者真正掌握如何借助 LEX 和 YACC 实现一个编译器，这一点在国内的教材中还没有先例。

（7）为了帮助读者掌握每章的重点和难点，每章附有小结和习题。

全书共分 9 章。第 1 章编译概述，主要介绍编译程序的基本概念和编译程序的结构。第 2 章形式语言的基本知识，这是编译原理课程的理论基础，介绍形式语言的基本概念和理论。第 3 章词法分析，以正规文法、正规式和有限自动机为工具，讨论词法分析的设计原理和设计方法，并介绍词法分析程序的自动生成工具 LEX 的使用方法和实现原理。第 4 章语法分析，对常规的两大类分析方法，即自顶向下和自底向上的分析方法进行了较深入的讨论，并给出了语法分析程序的自动生成工具 YACC 的使用方法。第 5 章语义分析及中间代码生成，介绍语法制导翻译的方法和常见的几种中间代码形式。第 6 章符号表管理和错误处理，先阐述了符号表的作用和组织方式，然后分别分析了错误处理中的词法错误、语法错误和语义错误。第 7 章运行时的存储组织与分配，针对运行时的几种不同的分配策

略——静态分配、栈式动态分配和堆式动态分配做了详细介绍。第 8 章代码优化，讨论了优化技术的相关知识，主要介绍局部优化和循环优化的方法。第 9 章目标代码生成，简单介绍目标代码常见的几种形式，然后以某一假想的计算机模型为背景，给出了一简单代码生成器的实现方法。附录 A 提供了 PL/0 编译程序的 C 语言版本，附录 B 提供了与附录 A 功能类似的 LEX 与 YACC 版本。

本书第 1 章到第 7 章由李冬梅编写，第 8 章、第 9 章由施海虎编写，全书由李冬梅统稿。学生王潇爽编写和测试了附录 B 中的程序，段文芳、胡英飞、贺莉娜、强亮亮等同学参加了文字编辑和书稿校对方面的工作，在此对他们表示诚挚的感谢。

中国科学院软件研究所程虎研究员审阅了全稿，并提出了许多宝贵的意见，在此表示衷心的感谢。

由于作者水平有限，错误与不妥之处在所难免，敬请读者批评指正。

编　者

2006 年 4 月

目　录

第 1 章　编译概述 …… 1
1.1　程序的翻译 …… 1
1.2　编译程序的组成 …… 3
1.2.1　编译过程概述 …… 3
1.2.2　编译程序的逻辑结构 …… 6
1.2.3　编译阶段的前端和后端 …… 7
1.2.4　遍（pass） …… 8
1.3　编译程序的构造 …… 8
1.3.1　如何构造一个编译程序 …… 8
1.3.2　编译程序的生成方法 …… 9
1.4　编译技术的应用及发展 …… 10
1.4.1　编译技术在软件开发中的应用 …… 10
1.4.2　编译技术的发展 …… 11
1.5　小结 …… 12
习题 …… 12
第 2 章　形式语言的基本知识 …… 13
2.1　字母表和符号串的基本概念 …… 13
2.1.1　字母表和符号串 …… 13
2.1.2　符号串的运算 …… 14
2.2　文法和语言的形式定义 …… 15
2.2.1　文法的非形式定义 …… 15
2.2.2　文法的形式定义 …… 17
2.2.3　推导的形式定义 …… 18
2.2.4　语言的形式定义 …… 19
2.2.5　递归文法 …… 21
2.3　句型的分析 …… 22
2.3.1　规范推导和规范归约 …… 22
2.3.2　语法树与文法的二义性 …… 23
2.4　文法和语言的分类 …… 25
2.5　PL/0 编译程序概述 …… 27
2.5.1　PL/0 语言的功能 …… 28
2.5.2　一个 PL/0 程序实例 …… 28
2.5.3　PL/0 语言的文法 …… 29
2.5.4　PL/0 编译程序的结构 …… 32
2.6　小结 …… 35

习题 35
第 3 章 词法分析 37
3.1 词法分析的任务 37
3.2 词法分析程序的输出形式 38
3.2.1 单词符号 38
3.2.2 词法分析程序的输出形式 38
3.3 词法分析程序的设计与实现 39
3.3.1 正规文法及其状态图 40
3.3.2 词法分析程序的实现 41
3.4 正规式与有穷自动机 45
3.4.1 正规式与正规集 45
3.4.2 正规文法与正规式 47
3.4.3 有穷自动机 48
3.4.4 正规式与有穷自动机的等价性 54
3.4.5 正规文法与有穷自动机的等价性 56
3.5 词法分析程序的自动生成工具 LEX 57
3.5.1 LEX 的源程序 58
3.5.2 LEX 的正规式 61
3.5.3 LEX 的实现 63
3.5.4 Parser Generator 的使用 65
3.6 PL/0 编译程序的词法分析 66
3.6.1 PL/0 词法分析程序的任务 66
3.6.2 PL/0 词法分析程序的设计 67
3.7 小结 69
习题 70
第 4 章 语法分析 73
4.1 语法分析的任务 73
4.2 自顶向下分析法 74
4.2.1 自顶向下分析的一般过程 74
4.2.2 自顶向下分析存在的主要问题 75
4.2.3 LL(1)文法的判别 78
4.2.4 递归下降分析法 82
4.2.5 LL(1)分析法 84
4.3 自底向上分析法 89
4.3.1 自底向上分析的一般过程 89
4.3.2 自底向上分析存在的主要问题 90
4.4 算符优先分析法 93
4.4.1 方法概述 93
4.4.2 算符优先文法的定义 94

4.4.3　算符优先关系表的构造 …… 95
4.4.4　算符优先分析算法 …… 100
4.4.5　优先函数 …… 104
4.5　LR 分析法 …… 106
4.5.1　LR 分析器的逻辑结构和工作过程 …… 107
4.5.2　LR(0)分析法 …… 111
4.5.3　SLR(1)分析法 …… 120
4.5.4　LR(1)分析法 …… 124
4.5.5　LALR(1)分析法 …… 128
4.5.6　二义性文法的 LR 分析法 …… 130
4.6　语法分析程序的自动生成工具 YACC …… 133
4.6.1　YACC 的源程序组成 …… 134
4.6.2　用 LEX 建立 YACC 的词法分析器 …… 141
4.7　PL/0 编译程序的语法分析 …… 143
4.8　小结 …… 144
习题 …… 146
第 5 章　语义分析与中间代码生成 …… 149
5.1　语义分析的任务 …… 149
5.2　语法制导翻译 …… 150
5.2.1　属性文法 …… 150
5.2.2　语法制导翻译的过程 …… 151
5.2.3　语法制导定义 …… 152
5.2.4　抽象语法树的构造 …… 157
5.2.5　S 属性定义与自底向上翻译 …… 160
5.2.6　L 属性定义与自顶向下翻译 …… 161
5.3　中间代码 …… 166
5.3.1　后缀式 …… 167
5.3.2　三地址代码 …… 167
5.3.3　图形表示 …… 170
5.4　说明语句的翻译 …… 171
5.4.1　简单说明语句的翻译 …… 171
5.4.2　过程中说明语句的翻译 …… 171
5.5　赋值语句的翻译 …… 172
5.5.1　简单赋值语句的翻译 …… 172
5.5.2　含数组元素的赋值语句的翻译 …… 174
5.6　布尔表达式的翻译 …… 180
5.6.1　布尔表达式的两种计算方法 …… 180
5.6.2　数值表示法翻译方案 …… 180
5.6.3　控制流语句中布尔表达式的翻译 …… 181

5.7 控制流语句的翻译 …… 186
5.7.1 基本控制流语句的翻译 …… 186
5.7.2 for 循环语句的翻译 …… 190
5.7.3 case 语句的翻译 …… 191
5.8 过程调用语句的翻译 …… 193
5.9 PL/0 编译程序的语义分析 …… 194
5.9.1 说明部分的分析 …… 195
5.9.2 过程体部分的分析 …… 197
5.9.3 PL/0 编译程序目标代码的结构 …… 197
5.9.4 PL/0 编译程序目标代码的生成 …… 198
5.10 小结 …… 203
习题 …… 203
第 6 章 符号表管理和错误处理 …… 206
6.1 符号表管理 …… 206
6.1.1 符号表的作用和内容 …… 206
6.1.2 符号表的组织 …… 207
6.1.3 分程序结构语言符号表的管理 …… 209
6.1.4 非分程序结构语言符号表的管理 …… 210
6.2 错误处理 …… 211
6.2.1 错误处理概述 …… 211
6.2.2 词法分析阶段的错误处理 …… 212
6.2.3 语法分析阶段的错误处理 …… 212
6.2.4 语义分析阶段的错误处理 …… 216
6.3 PL/0 编译程序的错误处理 …… 218
6.3.1 PL/0 编译程序的语法错误处理方法 …… 218
6.3.2 PL/0 编译程序的错误局部化处理原则 …… 218
6.3.3 PL/0 编译程序的错误局部化处理方法 …… 219
6.3.4 PL/0 语言的出错信息表 …… 220
6.4 小结 …… 221
习题 …… 222
第 7 章 运行时的存储组织与分配 …… 223
7.1 存储组织概述 …… 223
7.1.1 存储空间的一般组织 …… 223
7.1.2 过程的活动和活动记录 …… 224
7.2 静态存储分配 …… 225
7.3 栈式动态存储分配 …… 225
7.3.1 简单语言的栈式存储分配 …… 225
7.3.2 嵌套过程语言的栈式存储分配 …… 228
7.4 堆式动态存储分配 …… 231

7.5 PL/0 编译程序目标代码解释执行时的存储分配 …… 233
7.5.1 PL/0 目标计算机的组织结构 …… 233
7.5.2 PL/0 语言的栈式动态存储分配 …… 234
7.6 小结 …… 237
习题 …… 238
第 8 章 代码优化 …… 240
8.1 局部优化 …… 240
8.1.1 基本块的定义及其划分 …… 240
8.1.2 基本块内的优化 …… 241
8.1.3 基本块的 DAG 表示 …… 243
8.1.4 利用 DAG 进行基本块的优化处理 …… 248
8.2 循环优化 …… 250
8.2.1 程序流图 …… 250
8.2.2 循环的定义和查找 …… 251
8.2.3 循环优化 …… 253
8.3 小结 …… 258
习题 …… 259
第 9 章 目标代码生成 …… 261
9.1 目标代码的形式 …… 261
9.2 假想的计算机模型 …… 262
9.3 一个简单的代码生成程序 …… 263
9.3.1 待用信息与活跃信息 …… 263
9.3.2 寄存器分配 …… 265
9.3.3 简单的代码生成算法 …… 266
9.4 小结 …… 268
习题 …… 269
附录 A 使用 C 语言实现的 PL/0 程序 …… 270
A.1 程序简介 …… 270
A.2 程序文本 …… 270
附录 B 使用 LEX 和 YACC 语言实现的 PL/0 程序 …… 297
B.1 程序简介 …… 297
B.2 程序文本 …… 298
参考文献 …… 317

第 1 章　编 译 概 述

计算机语言是人机交流的工具，在当今世界上，现有的高级语言已超过千种。计算机无法直接执行高级语言，因此必须为计算机构造编译程序，将高级语言翻译成计算机能够识别的机器语言，然后由计算机去执行。本章内容作为学习编译原理的基础，给出编译程序中所涉及的一些基本概念，阐述编译过程和编译程序的基本结构，最后介绍编译技术的主要应用及发展趋势。

1.1　程序的翻译

语言是人与人之间传递信息的媒体和手段，在计算机的应用领域，程序设计语言充当了人与计算机之间的通信工具。程序设计语言是用来编写程序的工具，可分为两大类：第一类称为低级语言，包括机器语言和汇编语言，这类语言与特定的机器有关，功效高，但使用复杂、烦琐，容易出错；第二类称为高级语言，这类语言不依赖具体的机器，移植性好，易使用和维护，比如 BASIC、PASCAL、C 等。高级语言和低级语言相比，不论在算法描述的能力上，还是在编写和调试程序的效率上，都有很大的优越性。

然而，就目前的情况而言，计算机硬件只能识别机器语言，而不能直接执行用高级语言或汇编语言编写的程序。因此，除机器语言外，用其他语言编写的程序都必须经过翻译才能被计算机识别，这一过程由翻译程序来完成。

定义 1.1　所谓**翻译程序**是指这样一种程序，它能将甲语言编写的源程序翻译成与之等价的乙语言程序（称为目标程序）。甲语言称为该翻译程序的源语言，乙语言称为该翻译程序的目标语言。

翻译程序的功能如图 1.1 所示。

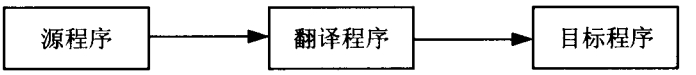


图 1.1　翻译程序的功能

程序的翻译通常有两种方式，一种是编译方式，另一种是解释方式。

（1）编译方式

定义 1.2　如果一个翻译程序的源语言是某种高级语言，其目标语言是某一计算机的汇编语言或机器语言，则称这种翻译程序为**编译程序**。

编译方式是一种分阶段进行的方式，一般需经过两个阶段：第一阶段称为编译阶段，

其任务是由编译程序将源程序编译为目标程序；第二阶段称为运行阶段，其任务是在目标计算机上执行编译阶段所得到的目标程序。在执行目标程序时，一般还应有一些子程序配合进行工作，如数据空间分配子程序、标准函数子程序等，这些子程序组成一个子程序库，称为运行系统。通常所说的编译系统实际上包含编译程序和相应的运行系统，在执行时，程序的初始数据作为目标程序和运行系统的输入，处理后输出相应的“计算”结果。

整个编译执行的过程如图 1.2 所示。

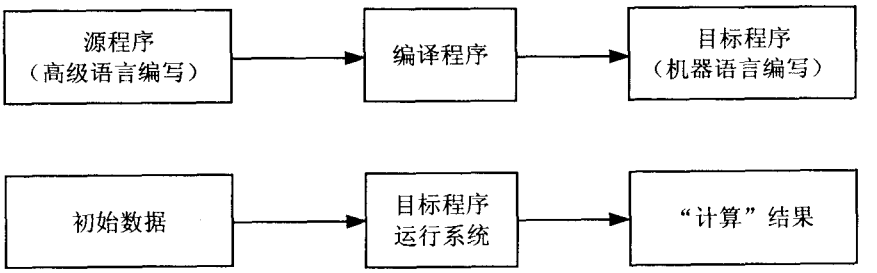


图 1.2　程序的编译执行

若编译生成的目标程序不是机器代码，而是汇编语言程序，则还要增加一个汇编程序将其汇编为机器代码。

定义 1.3　如果一个翻译程序的源语言是某种汇编语言，其目标语言是某一计算机的机器语言，则称这种翻译程序为**汇编程序**。

汇编程序的功能如图 1.3 所示。

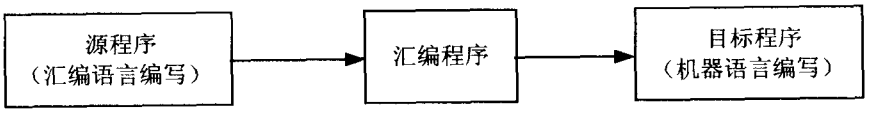


图 1.3　汇编程序的功能

（2）解释方式

用高级语言编写的程序也可以通过解释程序来执行。

定义 1.4　**解释程序**是将源程序中的语句按动态顺序，逐句翻译成可执行代码，一旦具备执行条件（如获得必要的初始数据等），则立即执行这一段代码得到部分结果。

解释执行的过程如图 1.4 所示。

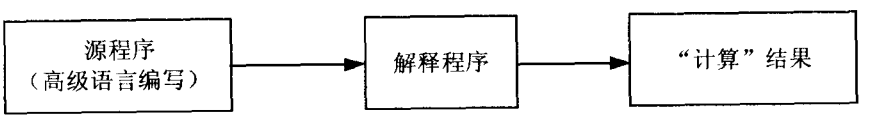


图 1.4　程序的解释执行

解释执行的工作效率很低，但由于解释程序的结构比编译程序简单，占用内存较少，在执行过程中也易于对源程序进行修改，因此一些规模较小的语言常采用解释执行，比如

BASIC语言。然而就目前的情况来看，纯粹的解释程序并不多见，通常是将编译程序和解释程序结合起来。例如，有的先将源程序翻译成某种易于进行解释执行的内部中间语言形式，然后对此中间语言进行解释执行。

在解释执行方式下，并不生成目标代码，而是直接执行源程序本身，这是编译方式与解释方式的根本区别。解释过程类似于自然语言中的口译，随时进行翻译。而编译过程类似于自然语言中的笔译，一次翻译可多次阅读。

编译程序已成为现今计算机系统最重要的系统程序之一。本课程的目的，在于向读者介绍编译的基本理论、常用的编译技术及编译系统的构造原理。其中的很多方法都适用于构造解释程序或汇编程序。事实上，任何一个熟悉编译程序构造原理的人，都会很容易掌握解释程序或汇编程序的实现方法。因此，本书将着重介绍编译程序实现的原理与方法。

1.2 编译程序的组成

1.2.1 编译过程概述

编译程序的功能是将用高级语言编写的源程序翻译成等价的用机器语言或汇编语言表示的目标程序。既然编译过程是一种语言的翻译过程，那么它的工作过程就类似于外文的翻译过程。例如，我们要将英文句子“I wish you success”翻译成中文，大致过程如下。

（1）词法分析。根据英语的词法规则，识别出四个英语单词，并检查是否有单词的拼写错误。

（2）语法分析。根据英语的语法规则，对词法分析后的单词串进行分析、识别，并做出语法正确性检查，看其是否能组成一个符合英语语法的句子。在此句中，“I”是代词，可以作主语；“wish”是动词，可以作谓语；“you”是代词，可以作间接宾语；“success”是名词，可以作直接宾语。此句符合英语语法。

（3）语义分析。对语法正确的英文句子分析其含义，并用汉语表示出来。“I”的含义是“我”，“wish”的含义是“希望”、“祝愿”等，“you”的含义是“你”、“你们”等，“success”的含义是“成功”、“成就”等。这样就可将此句初步翻译成“我希望你（们）成功”。

（4）对译文进行修饰。根据上下文的关系及汉语语法的相关规则进行综合考虑，对初步翻译后的句子进行必要的修饰。

（5）最后翻译成中文，“祝你成功”。

与上述的翻译过程类似，编译程序是将一种语言形式翻译成另一种语言形式，因此其工作过程可划分为五个阶段：词法分析、语法分析、语义分析及中间代码生成、代码优化和目标代码生成。

假设有以下C语言的程序段：

```
float x1,x2;
x1=5*x1+x2;
```

下面以上述程序段为例，概要介绍编译程序五个阶段的任务。

第一阶段　词法分析（lexical analysis）

词法分析的任务是分析和识别单词。源程序是由字符序列构成的，词法分析程序依次扫描源程序中的每个字符，根据语言的词法规则识别出一个一个具有独立意义的最小语法单位，即单词（TOKEN）。例如，关键字、标识符、运算符等。上述源程序通过词法分析识别出如下单词符号：

（1）关键字　float

（2）标识符　x1　x2

（3）赋值符　=

（4）常数　　5

（5）运算符　*　+

（6）界符　　，　；

除了识别单词之外，词法分析程序还要完成在语法分析之前需做的工作，如删除无用的空白符、回车符，删除注释，进行词法检查，报告所发现的错误等。

第二阶段　语法分析（syntax analysis）

语法分析的任务是分析和识别各种语法成分。在词法分析的基础上，根据语言的语法规则从单词符号串中识别出各种语法单位，如表达式、各种说明、语句、过程和函数等，并检查各种语法单位在语法结构上的正确性。通常将语法分析的结果表示为语法树，例如，依据赋值语句和表达式的定义规则，x1=5*x1+x2;经过语法分析后可以表示成如图 1.5 所示的语法树。图 1.6 给出了这种语法结构更简单的一种语法表示，运算符以内部节点的形式出现，而不是以叶节点出现。

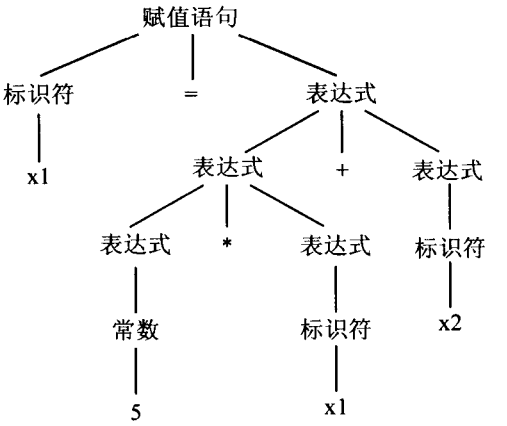


图 1.5　语句 x1=5*x1+x2;的语法树

第三阶段　语义分析（semantic analysis）及中间代码生成

语义分析是整个编译程序完成的最实质性的翻译任务。语义是指对语言的各种语法单位赋予具体的意义，语义分析的任务是对识别出的各种语法成分分析其含义，分析语义上的正确性，并进行初步翻译，生成中间代码或目标代码。

中间代码是一种介于源语言和目标语言之间的中间语言形式。生成中间代码的目的是为了便于代码的优化处理，便于目标代码的移植。对于中间代码的形式，编译程序设计者可以自己设计，常用的有四元式、三元式和逆波兰表示等。

在前述的赋值语句中，由语法分析识别出赋值语句后，语义分析首先要分析语义上的正确性，例如，要检查表达式中和赋值号两边的数据类型是否一致，对于语句 x1=5*x1+x2;中的整数 5，在语义分析阶段应该将其转换成实数，再与 x1 进行乘法运算，如图 1.7 所示，在语法树中增加了一个 inttoreal 的内部节点，然后根据赋值语句的语义，计算赋值号右边表达式的值，将其值送到赋值号左边的变量所确定的内存单元中。

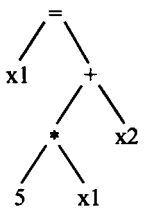


图 1.6　语句 x1=5*x1+x2;的语法树的另一种形式

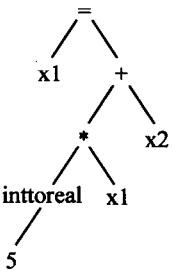


图 1.7　插入语义处理结点的语法树

下面以四元式为例，来说明语句 x1=5*x1+x2;所产生的中间代码，如表 1.1 所示。

表 1.1　　语句 x1=5*x1+x2;的四元式

序　号	运 算 符	左运算对象	右运算对象	结　果
(1)	inttoreal	5	—	T1
(2)	*	T1	x1	T2
(3)	+	T2	x2	T3
(4)	=	T3	—	x1

第四阶段　代码优化（optimization）

代码优化的任务是对中间代码进行等价的加工变换，以得到质量较高的目标代码。用来衡量目标代码质量的标准有两个：一个是空间指标，即目标代码所占用的存储空间的大小；另一个是时间指标，即运行时所需的时间。

在表 1.1 中，四元式(1)可以直接放在编译时做，将 5 变成 5.0，节省了运行时间。四元式(4)中的 T3 只用一次，可以用 x1 代替 T3，这样优化后得到表 1.2 所示的四元式中间代码。

表 1.2　　语句 x1=5*x1+x2;优化后的四元式

序　号	运 算 符	左运算对象	右运算对象	结　果
(1)	*	5.0	x1	T1
(2)	+	x2	T1	x1

第五阶段　目标代码生成（code generator）

目标代码生成的任务是把中间代码翻译成所期望的目标代码，一般为特定机器的机器语言代码或汇编语言代码。翻译过程中涉及具体机器的硬件以及指令和寄存器的选择。

例如，以假设的汇编语言为例，利用寄存器 R1 和寄存器 R2，由表 1.2 中的中间代码可生成如下目标代码：

```
MOV x1，R1
MUL 5.0，R1
MOV x2，R2
ADD R1，R2
```

MOV R2，x1

上述赋值语句编译的整个过程如图 1.8 所示。

读者应该注意的一点是，尽管编译过程与外文翻译的工作比较类似，但由于编译程序所翻译的毕竟不是自然语言，因此必然有其自身特有的一些工作，如中间代码的产生，运行时存储空间的分配等。在此将程序的编译和外文的翻译作比较只是让读者对编译过程有一个大致的理解。

1.2.2 编译程序的逻辑结构

按逻辑功能不同，可将编译过程划分为五个基本阶段，与此相对应，我们将实现整个编译过程的编译程序划分为五个逻辑阶段（即五个逻辑子过程）：词法分析程序、语法分析程序、语义分析及中间代码生成程序、代码优化程序和目标代码生成程序。

在编译程序的各个阶段中，都要涉及到符号表管理和错误处理。

1. 符号表管理

编译程序在整个编译过程中需要建立一些表格，以登记源程序中所提供的或在编译过程中产生的信息，在随后的编译过程中同时又要不断地查找这些表格中的信息，因此编译程序需要一个符号表管理程序。

例如，对于 float x1,x2;这条说明语句，词法分析程序识别出标识符 x1 和 x2 后，就将其填入符号表。但这时尚不能确定标识符的各种属性，它的各种属性要在后续的各阶段才能填入。例如，标识符的数据类型要在语义分析阶段才能确定，而标识符的地址可能要到目标代码生成阶段才能确定。

2. 错误处理

规模较大的源程序难免有多种错误，编译程序必须要有错误处理的功能，即能诊察出错误，并能报告用户错误的性质和位置，以便用户修改源程序。错误处理能力的优劣是衡量编译程序质量好坏的一个重要指标。

综上所述，一个典型编译程序的逻辑结构包括七部分，如图 1.9 所示。这上述的七个逻辑部分，是典型的编译程序在逻辑功能上共性的提炼，而实际上对于具体的编译程序，逻辑关系是多种多样的，有些阶段的工作是可以组合和交叉进行的，甚至可以缺省。比如对于小型的简单编译程序，可以直接在语义分析之后生成目标代码，而省略了中间代码生成和代码优化阶段，但是其他部分的基本功能是不能缺省的。

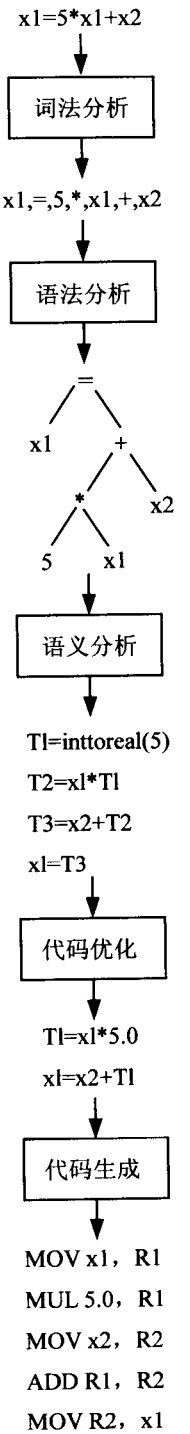


图 1.8 一个语句的编译过程

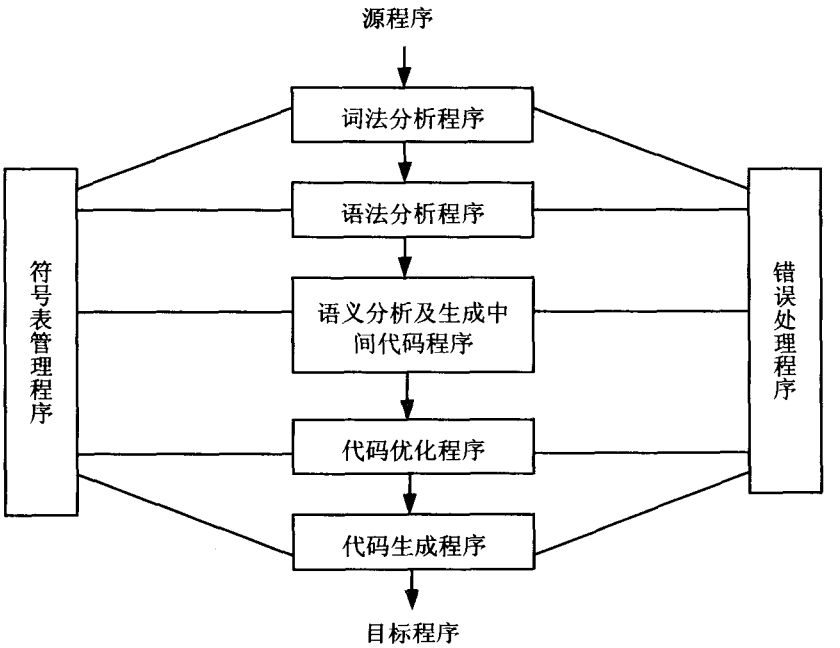


图 1.9 编译程序的逻辑结构

1.2.3 编译阶段的前端和后端

从概念上讲，可以把编译程序划分为编译前端和编译后端。

定义 1.5 编译的前三个阶段，即词法分析、语法分析、语义分析和中间代码生成是对源程序进行结构分析和语义分析，通常称作分析部分，也称为编译的**前端**。

也可将与机器无关的代码优化工作看作为编译的前端，前端的工作与源语言有关，而与目标机无关。

定义 1.6 编译的后两个阶段，即代码优化和目标代码生成称作综合部分，也称为编译的**后端**。

编译的后端工作与目标机有关。

通过划分编译的前端和后端，生成中间代码，不仅使编译程序的逻辑结构更合理，利于代码优化，而且对目标代码的生成和移植更有利，可以从以下两方面来说明。

一方面可以使同一前端配以不同的后端，便能在不同的机器构造同一语言的编译程序。比如，当需要变更一种语言的目标机时（即移植编译程序），则只需对新的目标机重新开发相应的后端即可。例如 Java 语言就是采用这种思想实现的（解释执行），即对于不同的计算机，解释器的前端都相同，所产生的也都是同一种中间代码——Byte code，若在不同的计算机上配置了能解释执行这种中间代码的后端（通常称为 Java 虚拟机），便能在这些计算机上解释执行 Java 程序，从而实现了 Java 语言的跨平台特性。目前流行的.NET 框架也是这个实现思想。

另一方面也可以使不同的前端配以同一后端，便能在同一机器上生成几个语言的编译程序，如源语言版本升级，只需对编译程序的前端进行相应的修改即可。另外，目前在全世界范围内得到广泛应用的编译程序集合 GCC（GNU Compiler Collection）的前端是一组程序设