21世纪高等学校规划教材 | 计算机科学与技术

重庆市精品课程配套教材

编译原理及实践教程
（第2版）

黄贤英 曹琼 王柯柯 编著

清华大学出版社

编译原理及实践教程
（第2版）

黄贤英 曹琼 王柯柯 编著

清华大学出版社
北京

内 容 简 介

本书系统地介绍了编译程序的设计与构造以及各组成部分的软件技术和实用方法。全书共 8 章，主要包括编译程序概述、高级语言设计基础、词法分析、语法分析、语义分析和中间代码生成、运行时存储空间的组织、代码优化以及目标代码生成。本书的目标是使学习者建立一个较为完整的编译系统的模型，掌握各个阶段的基本算法、常用的编译技术和方法，为今后从事系统软件和应用软件的开发打下理论和实践基础。为此，本书力求讲清基本概念、基本原理和实现方法；书中引入了丰富的典型例题，配以大量的习题；本书以 Sample 语言为例来贯穿各章内容，介绍了其编译程序的具体实现技术和构造方法。

本书可供高等学校计算机科学与技术及相关专业本科教学使用，也可供计算机系统软件和应用软件开发人员自学和参考。

图书在版编目(CIP)数据

编译原理及实践教程/黄贤英，曹琼，王柯柯编著. --2 版. --北京：清华大学出版社，2012.3
(21 世纪高等学校规划教材·计算机科学与技术)
ISBN 978-7-302-27743-9

Ⅰ. ①编… Ⅱ. ①黄… ②曹… ③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(2011)第 280164 号

责任编辑：付弘宇　战晓雷
封面设计：傅瑞学
责任校对：时翠兰
责任印制：王静怡
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**：15　　**字　　数**：364 千字
版　　次：2008 年 2 月第 1 版　　2012 年 3 月第 2 版　　**印　　次**：2012 年 3 月第 1 次印刷
印　　数：1～3000
定　　价：25.00 元

产品编号：044107-01

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程”（简称“质量工程”），通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的“高等学校教学质量与教学改革工程精品课程建设工作”，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了“清华大学出版社教材编审委员会”（以下简称“编委会”），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。“编委会”成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，“编委会”一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上。精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻性。教材由个人申报或各校推荐（通过所在高校的“编委会”成员推荐），经“编委会”认真评审，最后由清华大学出版

社审定出版。

目前,针对计算机类和电子信息类相关专业成立了两个“编委会”,即“清华大学出版社计算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。推出的特色精品教材包括:

(1) 21世纪高等学校规划教材·计算机应用——高等学校各类专业,特别是非计算机专业的计算机应用类教材。

(2) 21世纪高等学校规划教材·计算机科学与技术——高等学校计算机相关专业的教材。

(3) 21世纪高等学校规划教材·电子信息——高等学校电子信息相关专业的教材。

(4) 21世纪高等学校规划教材·软件工程——高等学校软件工程相关专业的教材。

(5) 21世纪高等学校规划教材·信息管理与信息系统。

(6) 21世纪高等学校规划教材·财经管理与应用。

(7) 21世纪高等学校规划教材·电子商务。

(8) 21世纪高等学校规划教材·物联网。

清华大学出版社经过三十多年的努力,在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌,为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格,这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会

联系人:魏江江

E-mail:weijj@tup.tsinghua.edu.cn

前言

“编译原理”是计算机及其相关专业的重要专业基础课，主要研究设计和构造编译程序的原理和方法。编译原理蕴涵着计算机学科中解决问题的思路、形式化问题和解决问题的方法，对应用软件和系统软件的设计与开发有一定的启发和指导作用，编译程序构造的原理和技术在软件工程和语言转换等许多领域中有着广泛的应用。

本书主要面向普通本科院校，理论学时为 40～48 学时，压缩了编译课程中的理论部分，删除了实用意义不大的编译方法。以程序编译的 5 个主要阶段——词法分析、语法分析、中间代码生成、代码优化和目标代码生成为线索，重点放在设计与构造编译程序及各个组成部分的软件技术和实用方法上。通过本课程的教学，使学生建立一个较为完整的编译系统的模型，掌握各个阶段的基本算法以及常用的编译技术和方法，为今后从事系统软件和应用软件的开发打下一定的理论和实践基础。

本书的主要特色如下。

(1) 力求将基本概念、基本原理和实现方法的思路阐述清楚，条理清晰，通俗易懂。

(2) 为便于自学，书中引入典型例题，以实例形式讲解理论，加强学生对理论的理解，并配以大量习题，以巩固所学的知识，并提供了参考答案。

(3) 为切实做到理论联系实际，便于读者更深刻地理解编译程序的实现过程，以 Sample 语言为贯穿本书各个章节的语言实例，重点介绍 Sample 语言的编译程序在各个阶段的具体实现技术和构造方法，并给出了部分程序框架。

(4) 本书注重实际应用，配套软件实现了 Sample 语言的词法分析、语法分析、语法制导的翻译，以及本书涉及的各种核心算法的实现，形象生动地展示了编译程序的分析过程，教师可将该软件用作课堂教学演示，也可用作“编译原理”课程作业的参考实例和实训内容；学生也可以通过该软件进行自学，在课后反复观看揣摩，并参考该软件中的编译程序实现方法，自己动手实现编译器中的部分内容。加 * 的章节为可选内容，请教师根据具体情况选择。

本书在第 1 版的基础上，对很多章节进行了删改，第 2 章增加了高级语言的设计部分，以便读者在了解编译方法的基础上，从高级语言的使用者过渡到高级语言的实现者和设计者。本书的编写得到了重庆理工大学教材出版基金的资助，本书第 1 版的使用院校的教师和学生也为本书的修订提出了宝贵意见和建议，在此一并表示衷心的感谢。

编　者

2011 年 8 月

目 录

第 1 章 概述 …… 1

1.1 程序设计语言与编译程序 …… 1
1.1.1 程序设计语言 …… 1
1.1.2 翻译程序 …… 2
1.1.3 编译程序和解释程序 …… 3
1.1.4 编译程序的伙伴 …… 3
1.2 编译过程和编译程序的结构 …… 5
1.2.1 编译过程概述 …… 5
1.2.2 编译程序的结构 …… 8
1.2.3 编译阶段的组合 …… 9
1.3 编译程序的设计 …… 11
1.3.1 编译程序的构造方式 …… 11
1.3.2 Sample 语言编译程序的设计 …… 12
1.4 编译程序的发展及应用 …… 13
1.4.1 编译程序的发展 …… 13
1.4.2 为什么要学习编译原理及其构造技术 …… 14
1.4.3 编译技术的应用 …… 14
1.5 小结 …… 16
1.6 习题 …… 16

第 2 章 高级语言设计基础 …… 17

2.1 符号和符号串 …… 17
2.2 文法与语言 …… 18
2.2.1 文法的定义 …… 19
2.2.2 文法产生的语言 …… 21
2.2.3 文法的二义性 …… 23
2.2.4 文法的分类 …… 25
2.3 高级语言的设计 …… 26
2.3.1 程序语言的定义 …… 26
2.3.2 冯·诺依曼体系结构与高级语言 …… 27
2.3.3 数据类型 …… 29
2.3.4 语句和控制结构 …… 29

2.3.5 语言设计的步骤 …… 32
2.4 语言设计实例 …… 32
2.4.1 Sample 语言字符集的定义 …… 33
2.4.2 Sample 语言单词的定义 …… 33
2.4.3 Sample 语言数据类型的定义 …… 35
2.4.4 Sample 语言表达式的定义 …… 35
2.4.5 Sample 语言语句的定义 …… 35
2.4.6 Sample 语言程序体和程序的定义 …… 37
2.4.7 符合 Sample 语言定义的源程序举例 …… 37
2.5 小结 …… 38
2.6 习题 …… 38

第 3 章 词法分析 …… 40

3.1 词法分析的任务和功能 …… 40
3.1.1 词法分析的功能 …… 40
3.1.2 单词的类型和种别码 …… 41
3.2 词法分析器的设计 …… 42
3.2.1 词法分析程序的接口 …… 43
3.2.2 词法分析程序的总体设计 …… 43
3.2.3 词法分析程序的详细设计 …… 45
3.2.4 单词的识别和状态转换图 …… 46
3.2.5 符号表及其操作 …… 48
3.2.6 词法分析阶段的错误处理 …… 49
3.3 正规文法、正规式与有穷自动机 …… 50
3.3.1 正规文法 …… 50
3.3.2 正规式 …… 51
3.3.3 有穷自动机 …… 52
3.3.4 正规文法与有穷自动机的等价性 …… 59
3.3.5 正规式与有穷自动机的等价性 …… 60
3.4 词法分析程序的自动生成 …… 63
3.4.1 LEX 的概述 …… 63
3.4.2 LEX 源文件的书写 …… 64
3.4.3 LEX 的工作原理 …… 69
3.4.4 LEX 使用中的一些注意事项 …… 71
3.4.5 使用 LEX 自动生成 Sample 语言的词法分析程序 …… 72
3.5 小结 …… 73
3.6 习题 …… 73

第 4 章 语法分析 …… 77

4.1 语法分析概述 …… 77
4.2 自上而下的语法分析 …… 78
4.2.1 自上而下分析方法中的问题探究 …… 78
4.2.2 递归下降分析方法 …… 86
4.2.3 预测分析方法 …… 90
4.2.4 Sample 语言自上而下语法分析程序的设计 …… 95
4.3 自下而上的语法分析 …… 97
4.3.1 自下而上分析方法概述 …… 98
4.3.2 算符优先分析法 …… 101
4.3.3 LR 分析法 …… 110
4.4 语法分析器的自动生成工具 YACC …… 131
4.4.1 YACC 概述 …… 131
4.4.2 YACC 源文件的格式 …… 132
4.4.3 YACC 的翻译规则 …… 134
4.4.4 YACC 的辅助程序 …… 135
4.5 语法分析程序中的错误处理 …… 135
4.5.1 语法分析中的错误处理的一般原则 …… 135
4.5.2 自上而下语法分析的错误处理 …… 136
4.5.3 自下而上语法分析的错误处理 …… 139
4.6 小结 …… 143
4.7 习题 …… 143

第 5 章 语义分析和中间代码生成 …… 147

5.1 概述 …… 147
5.1.1 语义分析和中间代码生成的功能和任务 …… 147
5.1.2 静态语义检查 …… 148
5.1.3 语义处理 …… 148
5.2 属性文法和语法制导的翻译 …… 151
5.2.1 属性文法的定义 …… 151
5.2.2 综合属性的计算 …… 153
5.2.3 继承属性的计算 …… 153
5.2.4 语法制导的翻译方法 …… 154
5.3 常见语句的语法制导的翻译 …… 156
5.3.1 语义变量和语义函数 …… 156
5.3.2 常量说明语句的语义处理 …… 157
5.3.3 变量说明语句的语义处理 …… 157
5.3.4 算术表达式和简单赋值语句的翻译 …… 159

5.3.5 布尔表达式的翻译 …… 161
5.3.6 if 语句的翻译 …… 167
5.3.7 do…while 语句的翻译 …… 169
5.3.8 for 语句的翻译 …… 170
5.4 Sample 语言语法制导的翻译程序的设计 …… 172
5.5 小结 …… 173
5.6 习题 …… 173

第 6 章 运行时存储空间的组织 …… 176

6.1 程序执行时的活动 …… 176
6.1.1 源程序中的过程 …… 176
6.1.2 过程执行时的活动 …… 177
6.1.3 名字的作用域 …… 178
6.1.4 参数的传递 …… 178
6.1.5 名字的绑定 …… 179
6.2 程序执行时的存储器组织 …… 179
6.2.1 程序执行时存储器的划分 …… 180
6.2.2 活动记录 …… 181
6.2.3 存储分配策略 …… 182
6.3 静态存储分配 …… 183
6.3.1 静态存储分配的性质 …… 183
6.3.2 静态存储分配的实现 …… 184
6.3.3 临时变量的地址分配 …… 185
6.4 栈式存储分配策略 …… 185
6.5 堆式存储分配 …… 187
6.5.1 堆式存储分配的主要问题 …… 188
6.5.2 堆式动态存储分配的实现 …… 189
6.5.3 存储回收 …… 190
6.6 小结 …… 191
6.7 习题 …… 191

第 7 章 代码优化 …… 192

7.1 概述 …… 192
7.1.1 代码优化的地位 …… 192
7.1.2 基本块的概念及流图 …… 193
7.2 局部优化 …… 195
7.2.1 删除公共子表达式 …… 196
7.2.2 复写传播 …… 196
7.2.3 删除无用代码 …… 197

7.2.4 对程序进行代数恒等变换 …… 197
7.2.5 基本块的 DAG 表示及优化 …… 198
7.3 循环优化 …… 202
7.3.1 循环的定义 …… 202
7.3.2 代码外提 …… 203
7.3.3 强度削弱 …… 204
7.3.4 删除归纳变量 …… 205
7.4 小结 …… 206
7.5 习题 …… 206

第 8 章 目标代码生成 …… 209

8.1 概述 …… 209
8.2 目标机器 …… 211
8.3 简单的代码生成算法 …… 212
8.3.1 中间代码的简单翻译方法 …… 212
8.3.2 引用信息和活跃信息 …… 214
8.3.3 寄存器描述和地址描述 …… 216
8.3.4 基本块的代码生成算法 …… 217
8.4 从 DAG 生成目标代码 …… 219
8.5 Sample 代码优化及目标代码生成器的设计 …… 221
8.6 小结 …… 223
8.7 习题 …… 223

参考文献 …… 225

第1章 概述

计算机只能执行用机器语言编写的程序，用高级语言编写的程序不能直接在计算机上执行，要想执行它，需要通过相应的翻译程序将其翻译为机器语言程序。编译程序就是这样一种翻译程序，它是现代计算机系统的基本组成部分。编写编译程序所涉及的一些原理、技术和方法是计算机工作者所必须具备的基本知识，在计算机相关的各个领域中都有广泛的应用，学习它具有非常重要的意义。

本章首先介绍编译程序的基本概念及与之相关的一些工具，然后介绍编译的过程以及编译程序的结构、组成以及构造方法，最后简单介绍编译技术的发展及其应用。

1.1 程序设计语言与编译程序

除了用机器语言书写的程序可以直接在计算机上执行外，其他所有语言编写的程序都需要先翻译为机器语言程序后才能执行。本节主要介绍程序设计语言及其翻译程序、高级语言程序的执行过程，以及编译器的伙伴工具。

1.1.1 程序设计语言

计算机是处理信息的工具。针对预定的任务，首先需要告诉计算机“做什么”、“怎么做”，计算机就可以自动处理，对给定的问题进行求解。为此，人们需要将有关信息告诉计算机，同时也需要计算机将计算的结果告诉人们。这样，人与计算机之间就要进行交流。正如人与人之间用语言进行交流一样，人们设计出词汇量少、语法简单、语义明确的程序设计语言(Programming Language)来实现人和计算机之间的交流。同时程序设计语言也是人与人之间的技术交流工具，在许多大型软件开发及软件维护中，程序员也需要读懂他人写的程序代码。

每当出现一种新的计算机，就随之产生一种该机器能理解并能直接执行的程序设计语言，这就是机器语言(Machine Language)。起初，人们直接用机器语言编写程序，如“将2放到一个存储单元”的机器代码为“C7 06 00 00 00 02”。机器语言很不直观，易出错，对硬件的依赖性大，可移植性差。用机器语言编写程序费时、乏味，开发难度大。程序员必须受过一定的训练，且熟悉计算机硬件，这在很大程度上限制了计算机的推广应用。

为了提高程序的可读性和可写性，人们将机器语言符号化，以助记符的形式表示指令和地址，这就产生了汇编语言(Assembly Language)，如“将2放到一个存储单元”的汇编代码

为“MOV X，2”。汇编语言仍然是依赖于机器的，且与人类的思维相差甚远，不易阅读和理解，程序设计的效率也很低。

为了解决这些问题，1954—1957 年 John Backus 等人参照数学语言设计了第一个描述算法的语言(即 FORTRAN 语言)，随后相继出现了许多语言，如 ALGOL 60、C 语言和 Pascal 等面向过程的语言，以及后来出现的面向问题的 SQL 语言与面向对象的语言，如 C++和 Java 等。

机器语言和汇编语言都是与机器有关的语言，通常称为低级语言(Low-Level Language)，其他与机器无关的程序设计语言通常称为高级语言(High-Level Language)。高级语言的出现缩短了人类思维和计算机语言之间的差距，“将 2 放到一个存储单元”用 C 语言书写就是 x=2。编写高级语言程序类似于定义数学公式或书写自然语言，与机器无关，便于理解和学习。

1.1.2 翻译程序

人类社会存在多种语言，为了相互交流，各种语言之间需要进行翻译。同样，人与计算机之间的信息交流也需要进行翻译。由于计算机只能理解机器语言，可直接执行用机器语言编写的程序，而不能直接执行用汇编语言和高级语言编写的程序，必须将其翻译成完全等价的机器语言程序才能执行。图 1.1 表示了计算机系统中语言的 3 个层次及其翻译。我们把能将一种语言(源语言，Source Language)书写的程序(源程序，Source Program)翻译成另一种语言(目标语言，Target Language)书写的程序(目标程序，Target Program)的程序统称为翻译程序(Translator)，翻译前后的程序在逻辑上是等价的。

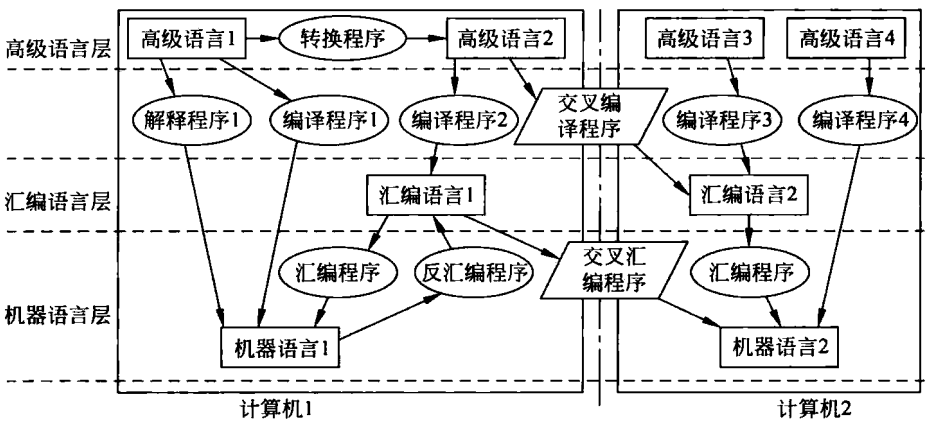

图 1.1 语言层次和转换关系

将汇编语言程序翻译为机器语言程序的程序称为汇编程序(Assembler)，又称汇编器。将高级语言(如 C 语言或 Java、Pascal 等)程序翻译为对应的低级语言(如汇编语言或机器语言)程序的程序称为编译程序(Compiler)，又称编译器。

实际上，除了高级语言程序和低级语言程序之间的翻译之外，同一种机器上的不同语言和不同种机器上的相同或不同语言书写的程序之间都可以进行翻译。高级语言之间可以相互转换，把一种高级语言程序转换为另一种高级语言程序的程序统称为转换程序

(Converter)。图中的交叉编译程序(Cross Compiler)是指能够将甲计算机上的高级语言程序翻译为乙计算机上的机器语言或汇编语言程序的程序。反汇编程序(Disassembler)是把机器语言程序逆向翻译为汇编语言程序的程序。交叉汇编程序(Cross Assembler)是把甲计算机上的汇编语言程序翻译为乙计算机上的机器语言程序的程序。

1.1.3 编译程序和解释程序

用高级语言编写程序简单方便,多数程序都可用高级语言编写,这就需要配置相应的编译程序将高级语言程序翻译为对应的低级语言程序。在一台计算机上要运行某一种高级语言程序,至少要为该语言配备一个编译程序。对有些高级语言甚至配置了几个不同性能的编译程序,以实现用户的不同需求。

根据用途和侧重点的不同,编译程序可进一步分类。专门用于帮助程序开发和调试的编译程序称为诊断编译程序(Diagnostic Compiler);着重于提高目标代码效率的编译程序称为优化编译程序(Optimizing Compiler)。运行编译程序的计算机称为宿主机(Host),运行编译程序所产生的目标代码的计算机称为目标机(Target)。如果不需要重写编译程序中与机器无关的部分就能改变目标机,则称该编译程序为可变目标编译程序(Retargetable Compiler)。

在实际使用中,高级语言除了通过编译程序将其翻译为机器语言执行外,也可以通过解释程序(Interpreter)把高级语言翻译为机器语言。解释程序是指按高级语言程序的语句执行的顺序边翻译边执行相应功能的程序,又称解释器。编译程序和解释程序的主要区别如下。

(1) 编译程序是源程序的一个转换系统,解释程序是源程序的一个执行系统。也就是说,解释器的工作结果是源程序的执行结果,而编译器的工作结果是等价于源程序的某种目标机程序。

(2) 编译程序先把全部源程序翻译为目标程序再执行,该目标程序可以反复执行;解释程序对源程序逐句地翻译执行,目标代码只能执行一次,若需重新执行,则必须重新解释源程序。编译过程类似于笔译,笔译后的结果可以反复阅读,而解释过程则类似于口译,别人说一句,就译一句,翻译的结果没有保存下来。

(3) 解释程序比编译程序更加通用。解释程序一般是用高级语言写的,能够在绝大多数类型的计算机上运行,而编译程序生产的目标代码只能在特定类型的计算机上运行。

(4) 通过编译运行,源程序和数据是在不同的时间进行处理的;通过解释运行,源程序和数据是同时处理的。图 1.2 是两个过程的比较。

本书重点介绍编译程序及相关原理、方法,关于解释程序不作深入探讨。许多编译程序的构造与实现技术同样适用于解释程序。

1.1.4 编译程序的伙伴

编译程序的重要性在于它使得多数计算机用户不必考虑与机器有关的烦琐细节,使程序员独立于机器硬件。除编译程序外,还需要其他一些程序相互配合才能使高级语言程序转换为能在计算机上执行的目标程序。图 1.3 给出了一个典型的高级语言程序的处理

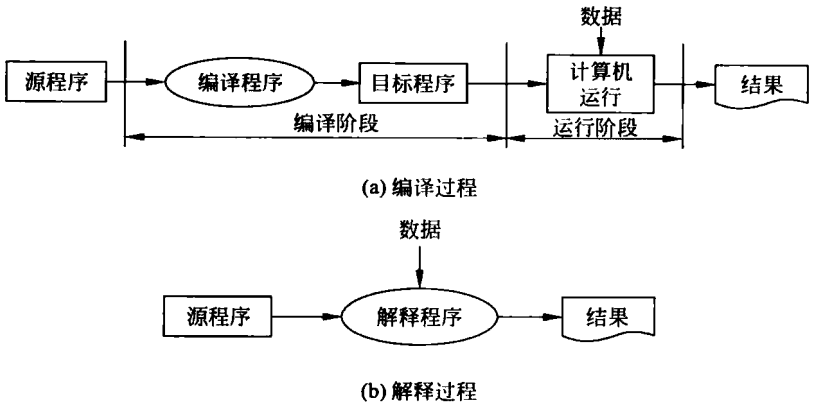

图 1.2 编译与解释过程对比

过程。

如果用户源程序分成多个模块并存储在不同的文件中,这就需要将源程序的各个模块汇集并连接在一起,这个任务是由预处理程序(Preprocessor)来完成的;预处理程序还要完成的主要功能包括宏定义、文件包含和语言扩充等功能。如在用户编写 C 语言程序时使用宏定义 # define max (a>b)?a:b,在预处理时由预处理程序将宏展开,凡是在源程序中遇到 max 的地方,全部用(a>b)?a:b 去代替;在 C 语言中使用文件包含 # include <stdio. h>语句,在预处理时,就用文件 stdio. h 中的内容来替换此语句。有些预处理程序还能够处理语言功能的扩充,用更先进的控制结构和数据结构来增强原来语言的功能。当源程序中使用了该结构,预处理器就把它转换为原编译器能够识别的形式加入到标准源程序中。

标准源程序由编译程序编译生成目标程序,图 1.3 中的目标程序是汇编代码,需要经过汇编程序(Assembler)将它翻译为可装配的机器代码。当然,有些编译程序不生成汇编代码,直接生成可装配的机器代码,直接传给装配连接程序。也有编译程序直接生成可执行代码的。

装配连接程序(Loader-Linker)通常又称为连接程序,或者连接装入程序,它完成连接和装入两个任务。装入是指读入可重定位的机器代码,修改需要重定位的地址,把修改后的指令和数据放在内存中适当的地方或形成可执行文件。连接是指把几个可重定位的机器代码文件连接成一个可执行的程序,这些文件可以是分别编译或汇编得到的,也可以是系统提供的库文件。

在对源程序进行处理的过程中,还有两个非常有用的工具程序:调试器(Debugger)和优化器(Optimizer)。调试器是自从计算机诞生伊始就始终伴随着程序员的一个挚友。起初的调试器都是基于硬件直接实现的,直到计算机行业有了比较大的发展之后,商业化的软件调试器才与程序员见面。调试是软件维护与错误修正的一个最重要、最直接,也是必不可少的一种机制。优化器主要是在编译过程中使程序在时间和空间方面得到最优的性能。

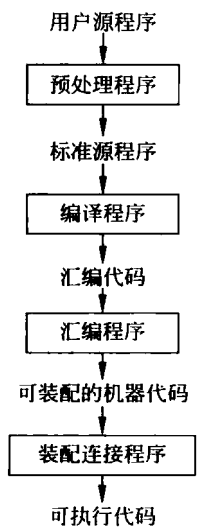

图 1.3 高级语言程序处理过程

现在,编译程序和它的伙伴工具以及其他一些工具,如编辑

器，往往都集成在一个集成开发环境中。

1.2 编译过程和编译程序的结构

编译程序完成从源程序到目标程序的翻译工作，这是一个复杂的过程。整个工作过程需要分阶段完成，每个阶段完成不同的任务，各个阶段进行的操作在逻辑上是紧密相关的，每个阶段的工作都通过相应的程序来完成。编译程序就是由完成这些功能的全部程序组成的。

1.2.1 编译过程概述

为了便于研究和学习，根据各个阶段的复杂程度、理论基础和实现方法的不同，通常将编译程序的工作过程划分为词法分析、语法分析、语义分析与中间代码生成、代码优化和目标代码生成5个阶段(如图1.4所示)，这是一种普遍的划分方法。

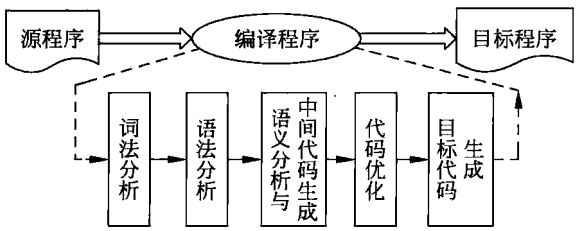

图1.4 编译的各个阶段

下面用例1.1的Pascal程序来扼要介绍编译程序如何将其翻译为目标程序，以及源程序在各个不同阶段被转换后的表示形式及各个阶段的任务。其中源程序是文本形式，即以字符串形式存在的高级语言源程序，目标程序是以文本形式存在的汇编代码。

例1.1 一个Pascal语言的源程序。

```
program test; /* this is an example,computing an area */
  var area, length, width: integer;
  begin
    length := 5;width := 5;
    area  :=  5 + length * width + length * width
end.
```

1. 词法分析

词法分析的任务是：从左到右扫描输入的源程序，检查词法错误，识别出一个个单词(或称为单词符号)，并输出单词的内部表示形式。

每种高级语言都规定了允许使用的字符集，如字母A～Z、a～z，数字0～9，以及符号+、-、*、/等。高级语言的单词都是由定义在该语言的字符集上的符号构成的，单词是语言中有意义的最小单位，有的单词由一个符号组成，如+、-、*、/等；有的单词由两个或多个符号组成，如<=、>=、end等。

在多数程序设计语言中,单词一般分为5类:保留字(如begin、end、if、for、while等)、标识符、常数、运算符(如+、-、*、/等)和界符(如标点符号、括号、注释符号等)。

例如,对例1.1所示的程序,词法分析首先去掉源程序中的空格和注释,识别出程序中出现的各个单词及其类型,如表1.1所示。

表1.1 例1.1的程序中的单词

序号	类型	单词	内部表示	序号	类型	单词	内部表示
1	保留字	program	$program	14	标识符	area	id
2	标识符	test	id	15	运算符	:=	:=
3	界符	;	;	16	常数	5	int
4	保留字	var	$var	17	运算符	+	+
5	标识符	area	id	18	标识符	length	id
6	界符	,	,	19	运算符	*	*
7	标识符	length	id	20	标识符	width	id
8	界符	,	,	21	运算符	+	+
9	标识符	width	id	22	标识符	length	id
10	界符	:	:	23	运算符	*	*
11	保留字	integer	$integer	24	标识符	width	id
12	界符	;	;	25	保留字	end	$end
13	保留字	begin	$begin	26	界符	.	.

其次将源程序转换为单词的内部形式输出。为便于区分,用id_1、id_2和id_3来表示area、length和width三个标识符的内部形式,用int_1、int_2表示常数5、3的内部形式,则上述输入串area := 5+length * width+length * width经过词法分析后输出为id_1 :=int_1+id_2 * id_3 +id_2 * id_3。

2. 语法分析

语法分析是在词法分析的基础上将单词组成各类语法短语(又称为语法单位或语法范畴,如表达式、语句、程序等),通过分析确定整个输入串是否具有语法上正确的程序结构,如果不能,则给出语法错误,并尽可能地继续检查。

语法分析依据语言的语法规则进行层次结构的分析,把token串按层次分组,以形成短语。语言的语法规则通常由递归规则来定义。如赋值语句和表达式可由下述递归规则来定义:

(1) 标识符 := 表达式(赋值语句的定义规则)。

(2) 任何标识符是表达式。

(3) 任何常数是表达式。

(4) 若表达式1和表达式2都是表达式,则表达式1+表达式2、表达式1 * 表达式2都是表达式,即表达式的运算也是表达式。

这里规则(2)和(3)是非递归的基本规则,规则(4)是把运算符+和*作用于其他表达式来定义表达式的规则。规则(1)是用表达式来定义赋值语句的规则。语法分析过程可以用一个语法树(通常也称为**分析树**)来表示。如上述输入串中的单词序列id_1 :=int_1 + id_2 *