WUTP

普通高等学校计算机科学与技术专业新编系列教材

Compiling Principle

编译原理

周经野　张继福　主编

```
#include <stdio.h>
void main()
{
   void swap(int *ptr1,int *ptr2);
  int x,y,*ptr1,*ptr2;
  printf("input x,y:");scanf("%d,%d",&x
  printf("%d\t%d\n",x,y);ptr1=&x;ptr2
  if(x<y)
  swap(ptr1,ptr2);
  printf("%d\t%d\n",x,y);
}
void swap(int *ptr1,int *ptr2)
```

武汉理工大学出版社
Wuhan University of Technology Press

普通高等学校计算机科学与技术专业新编系列教材

Compiling Principle

编译原理

周经野　张继福　主编

武汉理工大学出版社
Wuhan University of Technology Press

内 容 提 要

本书系统地介绍了程序设计语言编译系统的基本原理和方法，内容包括：词法分析、自顶向下与自底向上的语法分析、属性方法与语法制导翻译技术、语义分析和中间代码生成、目标程序运行时存储空间组织、代码优化、目标代码生成、并行编译技术基础，以及相关的形式语言和有限自动机的知识。在本书的编写中我们尽量做到深入浅出，便于理解，便于自学。本书可作为高等院校计算机科学与技术本科专业“编译原理”或“编译方法”课程的教材或参考书，也可供其他专业的学生或者从事计算机工作的有关人员阅读参考。

图书在版编目(CIP)数据

编译原理/周经野，张继福主编. —武汉：武汉理工大学出版社，2003.8
普通高等学校计算机科学与技术专业(本科)新编系列教材
ISBN 7-5629-1961-5

Ⅰ.编… Ⅱ.①周…②张… 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(2002)第 106882 号

出版发行：武汉理工大学出版社(武汉市武昌珞狮路 122 号 邮编：430070)
http://cbs.whut.edu.cn
E-mail:wutp02@163.com wutp@public.wh.hb.cn
经 销 者：各地新华书店
印 刷 者：武汉理工大印刷厂
开 本：787×960 1/16
印 张：23.5
字 数：460 千字
版 次：2003 年 8 月第 1 版
印 次：2003 年 8 月第 1 次印刷
印 数：1—5000 册
定 价：30.00 元

普 通 高 等 学 校
计算机科学与技术专业新编系列教材
编 审 委 员 会

出版说明

当今世界已经跨入了信息时代，计算机科学与技术正在迅猛发展。尤其是以计算机为核心的信息技术正在改变整个社会的生产方式、生活方式和学习方式，推动整个人类社会进入信息化社会。为了顺应时代潮流，适应计算机专业调整及深化教学改革的要求，充分考虑到不同层次高校的教学现状，满足广大高校的教学需求，武汉理工大学出版社经过广泛调研，与国内近30所高等院校的计算机专家进行探讨，决定组织编写“普通高等学校计算机科学与技术专业新编系列教材”。

我们在组织编写新编本套系列教材时，**以培养现代化高级人才为重任，以提高学生综合素质、培养学生应用能力和创新能力为目的，以面向现代化、面向世界、面向未来为准绳，注重系列教材的特色和实用性，反映最新的教学与科研成果，体现本专业的时代特征。**同时，面对教育改革的需要、人才的需要和社会的需要，在编写本教材时，借鉴、学习国外一流大学的先进教学体系，结合国内的实际需要，吸取具有先进性、实用性和权威性的国外教材的精华，以更好地促进国内教材改革顺利进行。从时代和国际竞争要求的高度来思考，为打造一套高起点、高水平、高质量的系列教材而努力。

本套教材具有以下特色：

与时俱进，内容科学先进——充分体现计算机学科知识更新快的特点，及时更新知识，确保教材处于学科前沿，以拓宽学生知识面，培养学生的创新能力。

紧跟教学改革步伐，体现教学改革的阶段性成果——符合全国高校计算机专业教学指导委员会、中国计算机学会教育委员会制订的“计算机学科教学计划2000”的内容要求。

实现立体化出版，适应教育方式的变革——本套教材努力使用和推广现代化的教学手段，凡有条件的课程都准备组织编写、制作和出版配合教材使用的实验、习题、课件、电子教案及相应的程序设计素材库。

本套教材首批25种预定在2003年秋季全部出齐。我们的编审者、出版者决不敢稍有懈怠，一定高度重视，兢兢业业，按最高的质量标准工作。教材建设是我们共同的事业和追求，也是我们共同的责任和义务，我们诚恳地希望大家积极选用本套教材，并在使用过程中给我们多提意见和建议，以便我们不断修订、完善全套教材。

武汉理工大学出版社

2002年10月

前　言

编译原理是计算机科学与技术专业的核心课程之一。为了促进编译原理课程的教学，我们编写了这本教材。

编译程序是重要的计算机系统软件之一。它对于计算机软件的开发起着举足轻重的作用。编译原理课程是比较抽象的，学生普遍反映学习起来较为困难。因此在编写的过程中，我们力图写得深入浅出，以便于读者的学习。

编译程序是一个复杂的系统，本书基本上依据编译程序的结构来编写。全书共分十二章。第 1 章：编译概论，介绍编译程序的结构以及它在软件技术中的应用。第 2 章：文法和语言，介绍形式文法和形式语言的一些基本概念。第 3 章：词法分析，介绍词法分析器的构造、单词的描述工具，正规表达式和正规文法，以及识别单词的机器——有限自动机。第 4 章：自顶向下的语法分析，介绍自顶向下分析的 LL(1)文法及其分析器的构造。第 5 章：自底向上的语法分析，介绍自底向上分析的文法，算符优先文法和 LR 文法，以及它们的分析器的构造。第 6 章：属性文法与语法制导翻译技术，介绍语义分析的工具——属性文法，以及语言翻译的技术——语法制导翻译。第 7 章：语义分析和中间代码生成，通过对程序设计语言的几个典型的语法单位、算术表达式和布尔表达式、条件语句、循环语句等等的介绍更进一步地讲解语法制导翻译技术的思想和实现。第 8 章：符号表，介绍编译程序中各种符号表的组织和管理。第 9 章：目标程序运行时存储空间组织，介绍编译程序对目标程序运行时存储空间组织和管理的技术。第 10 章：优化，介绍编译程序对中间代码的优化技术，主要是局部优化和循环优化。第 11 章：目标代码生成，在一个假想的计算机模型上介绍目标代码生成的技术。第 12 章：并行编译技术基础，介绍并行计算机以及并行编译器的构造和基础的思想方法。

本书可以作为大专院校计算机科学与技术专业的编译原理课程教材，也可以作为各类计算机软件教学的培训教材或者自学参考书。

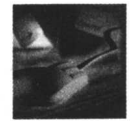
本书第 1、6、7、12 章由周经野编写，第 2、5 章由张继福编写，第 4 章由饶文碧编写，第 8、9 章由张陵山编写，第 3、11 章由曹江莲编写，第 10 章由何九周编

写。全书由周经野和张继福任主编。

由于编者水平有限，编写时间仓促，书中错误及不当之处在所难免，诚望读者批评指正。

编 者

2003 年 6 月



目　录

1　编译概论 …… (1)

1.1　程序设计语言和编译程序 …… (1)
1.2　编译的过程和编译程序结构 …… (3)
1.3　编译阶段的组合 …… (5)
1.4　编译技术在软件工具中的应用 …… (6)

2　文法和语言 …… (9)

2.1　文法的非形式化描述 …… (9)
2.2　符号、符号串及其运算 …… (11)
2.3　文法和语言的形式化定义 …… (13)
2.3.1　文法的形式化定义 …… (13)
2.3.2　推导的形式化定义 …… (14)
2.3.3　语言的形式化定义 …… (17)
2.3.4　语法树 …… (19)
2.4　文法和语言的类型 …… (21)
思考题与习题 …… (23)

3　词法分析 …… (25)

3.1　词法分析概述 …… (25)
3.2　单词的描述工具 …… (27)
3.2.1　正规文法 …… (28)
3.2.2　正规表达式 …… (29)
3.3　有限自动机(DFA) …… (32)
3.3.1　确定的有限自动机(DFA) …… (35)
3.3.2　不确定的有限自动机(NFA) …… (36)
3.3.3　NFA 的确定化 …… (38)
3.3.4　DFA 的最小化 …… (42)
3.4　正规文法、正规式和有限自动机之间的等价转换 …… (45)
3.4.1　正规文法与正规式 …… (45)

3.4.2 正规文法与有限自动机 …… (49)
3.4.3 正规式与有限自动机 …… (51)
3.5 词法分析程序的自动生成 …… (56)
3.5.1 LEX 源文件的结构 …… (59)
3.5.2 LEX 系统中的正规式 …… (62)
3.5.3 LEX 翻译程序简介 …… (66)
3.5.4 LEX 的使用方式 …… (67)
3.5.5 LEX 应用举例 …… (68)
思考题与习题 …… (72)

4 自顶向下的语法分析 …… (75)

4.1 自顶向下的语法分析思想 …… (76)
4.1.1 确定的自顶向下分析 …… (77)
4.1.2 不确定的自顶向下分析 …… (83)
4.2 LL(1)文法 …… (86)
4.2.1 LL(1)文法的判别 …… (86)
4.2.2 某些非 LL(1)到 LL(1)文法的等价转换 …… (93)
4.3 预测分析方法 …… (100)
4.3.1 预测分析程序的工作流程 …… (101)
4.3.2 预测分析表的构造 …… (104)
4.3.3 预测分析方法举例 …… (104)
思考题与习题 …… (109)

5 自底向上的语法分析 …… (113)

5.1 自底向上的分析方法简介 …… (113)
5.1.1 算符优先分析法 …… (114)
5.1.2 LR 分析法 …… (116)
5.2 算符优先分析方法 …… (119)
5.2.1 算符优先文法的定义 …… (119)
5.2.2 构造算符优先关系表 …… (122)
5.2.3 算符优先分析算法 …… (125)
5.2.4 优先函数 …… (127)
5.2.5 算符优先分析方法的讨论 …… (129)
5.3 LR 分析方法 …… (130)
5.3.1 LR(0)分析器 …… (130)

5.3.2 SLR(1)分析器 …… (138)
5.3.3 LR(1)分析器 …… (140)
5.3.4 LALR(1)分析器 …… (144)
5.3.5 LR 分析方法的讨论 …… (147)
思考题与习题 …… (149)

6 属性文法与语法制导翻译技术 …… (151)

6.1 属性文法 …… (152)
6.1.1 继承属性和综合属性 …… (154)
6.1.2 属性翻译文法 …… (158)
6.2 语法制导翻译技术 …… (161)
6.2.1 自顶向下的语法制导翻译 …… (162)
6.2.2 自底向上的语法制导翻译 …… (167)
思考题与习题 …… (174)

7 语义分析和中间代码生成 …… (176)

7.1 中间代码 …… (176)
7.1.1 逆波兰记号 …… (176)
7.1.2 三元式 …… (178)
7.1.3 四元式 …… (179)
7.2 赋值语句与算术表达式的翻译 …… (180)
7.3 布尔表达式的翻译 …… (184)
7.4 控制结构的翻译 …… (188)
7.4.1 条件语句 …… (188)
7.4.2 循环语句 …… (190)
7.4.3 过程调用 …… (193)
7.5 说明语句的翻译 …… (194)
7.5.1 数组说明的翻译 …… (195)
7.5.2 结构说明的翻译 …… (197)
7.6 数组元素访问的翻译 …… (199)
思考题与习题 …… (203)

8 符号表 …… (207)

8.1 符号表概述 …… (207)
8.2 符号表的组织 …… (208)

8.2.1 符号表的一般形式 …… (208)
8.2.2 符号表的名字栏 …… (209)
8.2.3 符号表的信息栏 …… (210)
8.2.4 符号表中的数据 …… (211)
8.3 符号表的管理 …… (212)
8.3.1 符号表的初始化 …… (212)
8.3.2 符号表的编制与查找 …… (212)
8.3.3 名字作用域信息的表示 …… (219)
思考题与习题 …… (221)

9 目标程序运行时存储空间组织 …… (223)

9.1 概述 …… (223)
9.1.1 存储空间的一般组织 …… (223)
9.1.2 过程和过程的活动记录 …… (225)
9.1.3 名字和名字的作用域 …… (227)
9.2 数据对象的存储分配 …… (228)
9.2.1 基本数据对象的存储分配 …… (229)
9.2.2 数组的存储分配 …… (229)
9.2.3 记录结构的存储分配 …… (232)
9.3 参数传递 …… (233)
9.3.1 名字调用 …… (233)
9.3.2 引用调用 …… (234)
9.3.3 值调用 …… (235)
9.3.4 结果调用 …… (235)
9.4 静态存储分配 …… (235)
9.5 栈式存储分配 …… (237)
9.5.1 简单栈式存储分配 …… (238)
9.5.2 嵌套过程的栈式存储分配 …… (240)
9.5.3 分程序结构的栈式存储分配 …… (243)
9.6 堆式存储分配 …… (247)
思考题与习题 …… (250)

10 代码优化 …… (253)

10.1 代码优化技术概述 …… (254)
10.1.1 删除公共子表达式(删除多余运算) …… (255)

10.1.2 代码外提 …… (255)
10.1.3 强度削弱 …… (256)
10.1.4 变换循环控制条件(删除归纳变量) …… (257)
10.1.5 合并已知变量 …… (257)
10.1.6 复写传播 …… (257)
10.1.7 删除无用赋值 …… (258)
10.2 局部优化 …… (259)
10.2.1 基本块的划分和变换 …… (259)
10.2.2 基本块的 DAG 表示和应用 …… (261)
10.3 循环优化 …… (270)
10.3.1 程序流图 …… (270)
10.3.2 循环 …… (272)
10.3.3 可归约流图 …… (279)
10.3.4 循环优化方法 …… (280)
10.4 全局优化概述 …… (284)
10.4.1 “到达-定值”数据流方程 …… (285)
10.4.2 活跃变量数据流方程的推导 …… (290)
思考题与习题 …… (293)

11 目标代码生成 …… (298)

11.1 概述 …… (298)
11.1.1 代码生成器的输入和输出 …… (299)
11.1.2 指令的选择 …… (300)
11.1.3 寄存器分配 …… (301)
11.1.4 计算顺序的选择 …… (301)
11.2 计算机模型 …… (301)
11.3 代码生成器 …… (304)
11.3.1 待用信息 …… (305)
11.3.2 寄存器分配 …… (307)
11.3.3 代码生成算法 …… (311)
思考题与习题 …… (321)

12 并行编译技术基础 …… (324)

12.1 并行计算机 …… (325)
12.1.1 向量计算机 …… (325)

12.1.2 共享存储器多处理机 …… (326)
12.1.3 分布式存储器大规模并行计算机 …… (328)
12.2 并行编译器的结构 …… (329)
12.2.1 程序分析 …… (330)
12.2.2 程序优化 …… (330)
12.2.3 并行代码生成 …… (331)
12.3 依赖关系 …… (331)
12.3.1 依赖关系的定义 …… (332)
12.3.2 语句依赖图 …… (336)
12.3.3 循环的迭代依赖图 …… (337)
12.4 循环的向量化和并行化 …… (344)
12.4.1 可向量化循环 …… (344)
12.4.2 可并行化循环 …… (347)
12.5 循环的变换技术 …… (348)
12.5.1 语句重排(Statement Reordering) …… (349)
12.5.2 循环置换(Loop Permutation) …… (350)
12.5.3 循环逆转(Loop Reversal) …… (352)
12.5.4 圈收缩(Cycle Shrinking) …… (353)
12.5.5 循环分布(Loop Distribution) …… (354)
思考题与习题 …… (358)

参考文献 …… (363)

1 编译概论

1.1 程序设计语言和编译程序

计算机必须要能够与人交流，能够明白人的命令，才能为人服务。怎样才能让计算机懂得人的命令呢？因此，就需要一种能够沟通人和机器的语言。

我们知道，计算机硬件能够直接识别并执行的命令是它的指令系统。指令系统称为机器语言，或者说，计算机的母语是它的机器语言。可以选择机器语言作为人-机沟通的语言。也就是说人迁就机器，用机器语言和二进制指令码来编写程序。这样当然可以，并且机器收到命令后就可以直接识别立即执行，效率很高。计算机刚诞生的最初一段时间里，人们也的确是这样做的。我们把机器语言称为第一代程序设计语言。

但是这种迁就让人感到太为难。二进制形式的指令代码中除了“1”就是“0”，人们很难记住它们，也很难读懂它们，或者说，用机器语言编写程序难度大，调试困难。并且不同种类的计算机之间的机器语言也是不同的。这种状况阻碍了计算机的应用和发展。为此，人们便采用了一些符号来代表那些二进制形式的指令码，并且还把数据部分也用符号来代替。这种符号形式的指令系统称为汇编语言，并被称为第二代程序设计语言。

尽管汇编语言最接近机器语言，汇编语句基本上与指令是一一对应的，但是汇编语言已经不能被机器直接识别，需要经过一个翻译程序将其翻译成机器语言，机器才能够识别并执行。这个翻译过程称为“汇编”。汇编语言实际上是机器语言的符号化，所以汇编语言程序和机器语言程序的运行效率几乎是一样的。直到今天，汇编语言仍然是专业计算机工作者进行开发程序时的重要工具，特别是在要求高效率运行的程序开发中更是如此。

虽然汇编语言比机器语言容易记忆，也容易理解，但仍然是面向机器的语言，故称之为低级语言，以区别后来人们发明的模仿人类语言的程序设计语言，即所谓的高级语言。汇编语言仍然是依赖于具体机型、难以阅读、难于移植，所

以依然难以被人们所掌握。如果能用自然语言，即人类在社会发展的历史过程中所自然形成的语言，比如，汉语、英语等来和机器沟通，那当然是最好的、最方便的。但是，这需要有一个翻译程序能把某种自然语言翻译成机器语言。人们已经为此目标努力了半个多世纪，虽然取得了一些成绩，但至今还远远没有取得成功，仍然在继续地努力着。

西方哲学界将20世纪称为“语言哲学”的世纪。其实不仅是哲学，近百年来科学文化的众多领域都以“语言”为突破口取得了长足的发展。语言学终于摆脱了工具论的传统框架，作为社会与人的存在的重要依据而向纵深拓展着自己的研究空间。其中最为引人注目的是语言学理论的突破带来了计算机技术的突破。这种突破对整个世界的经济、科技、社会发展乃至人自身所产生的影响是无法估量的。

第二次世界大战中，出于对日本作战的需要，美军需要研究如何能尽快地了解和掌握南太平洋诸岛上众多的土著人的语言。这一背景直接刺激了语言学的研究。语言学家们开始研究如何用一些形式的规则来描述自然语言的语法。这样，在努力研究自然语言的法则的过程中，人们发明了一类新的语言——形式语言，即采用一套形式规则来描述的语言。形式语言与自然语言还有很大的差距，但是却成为一类新的语言，即高级程序设计语言，简称高级语言。

所谓高级语言就是人们模仿自然语言，采用一套形式规则来设计的人工语言。高级语言与汇编语言相比，更接近人类自然语言的语法和词汇，编写出来的程序更容易阅读和理解，对问题的表达也更容易。这样就大大地简化了程序的编制和调试，极大地提高了编程的效率。另外，高级语言是依据设计者所制定的形式规则来设计的，所以不依赖于具体的计算机机型。因此，高级语言编写的程序的通用性和可移植性好。

高级语言的诞生及其发展促进了软件产业的诞生和发展，也促进了计算机的推广应用。到目前为止，人们已经设计了很多种高级语言，并还在设计着新的高级语言。例如，世界上最早的而且至今仍被广泛使用的FORTRAN，第一个用于商务事务处理的COBOL，在人工智能领域中最早使用的函数型语言LISP，第一个在微机上使用并易于初学者学习的BASIC，第一个引入数据抽象概念而被认为是面向对象技术的先驱的SIMULA，第一个体现结构化程序设计思想的PASCAL，为编写操作系统UNIX而设计的C，逻辑型语言的典型代表PROLOG，目前在网络上最为流行的JAVA等等。正是这些高级语言支撑着数不清的各种各样的计算机软件的设计开发，使得计算机的应用越来越广泛、越来越深入。

但是，高级语言同样要翻译成机器语言后才能被机器识别和执行。由于高级语言的语法是一套人为制定的形式规则，人们就可以依据这套规则来为它设

计相应的翻译程序。我们说在一个计算机系统中配置了某种高级语言，就是指安装了这种语言的翻译程序。可以说，如果没有这些翻译程序，也不可能有高级语言的应用与发展。本书就是专门介绍这种翻译程序的工作原理和构造方法的。

高级语言的翻译程序有两种形式，一种叫“编译程序”，它好像“书面翻译者”，把一篇文章全部翻译完了并作一番总体的润色后才交给读者；另一种叫“解释程序”，它好像“口头翻译者”，讲一句就立即翻译一句给听众。前者翻译得慢些，但是翻译后的程序经过了优化，质量较好；后者翻译得快些，但翻译后的程序没有进行优化，总体质量逊色于前者。这两种翻译过程的工作原理和构造方法是一样的。因此，本书只介绍编译程序的工作原理，不对解释程序作专门的讨论。

1.2 编译的过程和编译程序结构

总的来说，编译程序是将用户输入的、用某种语言书写的程序（称为源程序）转换为可以在某个计算机上执行的程序（称之为目标程序）。这个将源程序转换为目标程序的过程是非常复杂的，一般来说，它经历了词法分析、语法分析、中间代码生成、优化和目标代码生成这样五个阶段（Phrase）。

源程序是一个不间断的符号流，即便是人们视为间断符号的空格、标点符号等，对机器同样是一种符号。但是，任何一种语言中表达最小独立意义的单位是单词，因此机器首先需要对构成源程序的符号流进行扫描和分解，识别成一个一个的单词，并且知道这是一个什么类型的单词，以便后续翻译工作的进行。这个识别单词的阶段称为词法分析。

单词构成了不同的语法单位。任何一种程序设计语言都定义了一些不同类型的语法单位，例如：算术/逻辑表达式、赋值语句、条件语句、循环语句、说明语句、程序段、程序，等等。各种类型的语法单位都有着其自身的结构和相对应的意义。识别了单词之后，编译器就需要识别这些单词构成了哪些语法单位以及它们是怎样构成的，这项工作称为语法分析。

语法分析是翻译必不可少的一部分。实际上翻译工作就是在语法分析的过程中同步完成的，或者说是在语法分析的制导下进行的。这种翻译的方式称为语法制导翻译。严格地说，语法分析应该与翻译工作同时进行，而不需要单独的语法分析阶段。但是，翻译除了语法分析外还有很多的附加的工作量，而在人们编写的源程序中又很难保证没有语法上的错误。因此，如果让语法分析与翻译同步进行，一旦发现源程序在语法上出现错误，则整个翻译工作要推倒重来。所以，人们在翻译之前加上了一个单纯进行语法分析的阶段，先来检查一下源程序

在语法上是否完全正确。只有当源程序在语法上完全正确之后，编译器才开始进行翻译工作。

编译器并不是直接将源程序翻译成目标代码，而是先翻译成某一种中间代码。因此，翻译工作被分成了两个阶段：一个是中间代码生成阶段，另一个是目标代码的生成阶段。这样做的目的主要有两个：第一，因为中间代码是与机器的型号无关的表示方式，而目标代码是历来与机器型号有关的，这样，当把这个语言的编译器用于不同型号的机器时，只需要修改编译器中生成目标代码的这一部分，编译器的其他部分都无需改动，这样增加了这个编译器的可移植性。其次，人们可以通过对中间代码优化的研究来找到一般的优化方法，即不依赖于具体机型的优化方法，这样可以把代码优化做得更好。

机器直接翻译出来的代码，如果不经过优化，其质量是不高的。因此还需要对中间代码进行优化处理，来获得较高质量的翻译代码。这个阶段就是优化阶段。

最后一个阶段就是目标代码生成，即编译器将经过优化处理后的中间代码序列转换为在机器上可以执行的目标代码的序列。在编译器的几个工作阶段中，只有这个阶段是依赖于机器的。在这个阶段中，编译器除了生成目标代码外，还进行适当的依赖于机器的优化工作。

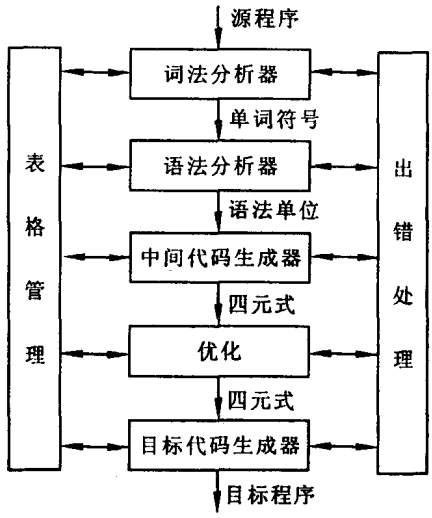


图 1.1 编译器的构造

整个编译器的结构如图 1.1 所示。

在图 1.1 中，除了承担上述五个阶段的工作模块之外，还有两个附加的模块：表格管理模块和出错处理模块。

在编译过程中编译器要将源程序中的各种信息保留在各种不同的表格中，例如，符号名表（登记源程序中的常量名、变量名、数组名、过程名等等的性质和定义等信息）、常数表（登记源程序中各种类型的常数的值）以及其他的表格。并且，在编译过程中，对这些表格的访问也是非常频繁的。因此合理地设计和使用这些表格是编译程序构造的一个重要问题。为此，在编译器中专门安排了一个模块来进行表格的管理。

如果源程序中有错误，编译器应该设法发现错误，并将相关的错误信息告诉用户，以便用户改正。这部分工作是由一个专门的出错处理模块来承担的。源