21世纪高等学校计算机专业规划教材

程序设计语言编译方法

第三版 （内附习题参考答案）

编著◎肖军模

大连理工大学出版社 Dalian University of Technology Press

21世纪高等学校计算机专业规划教材

程序设计语言编译方法

第三版(内附习题参考答案)

肖军模　编著

大连理工大学出版社

内 容 提 要

本书以类 PASCAL 的 SIMPLE 语言为背景，介绍程序设计语言的各种编译技术。本书前三章重点介绍编译技术所涉及的概念、形式语言理论和程序语言的结构与定义方法；第四章至第十章分别介绍各种实用的词法分析、语法分析和语义翻译、存贮管理、数据流分析技术、代码优化、目标代码生成等技术；第十一章介绍连接装配程序的设计方法。书中还按软件工程的观点介绍了 SIMPLE 语言编译器的设计方法。全书内容通俗易懂，理论性、系统性与实用性兼融。

本书可作为计算机专业的大专生、本科生及研究生的教材，也可以作为计算机科技工作者的参考书。本书学时：课内 60 学时，上机实验 20 学时。

图书在版编目(CIP)数据

程序设计语言编译方法/肖军模编著.—3 版.—大连：大连理工大学出版社，2000.8
(21 世纪高等学校计算机专业规划教材)
ISBN 7-5611-0099-X

Ⅰ.程 … Ⅱ.肖 … Ⅲ.编译程序-高等学校-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 23158 号

大连理工大学出版社出版发行
大连市凌水河　邮政编码　116024
电话：0411-4708842　传真：0411-4708898
E-mail：dutp@mail.dlptt.ln.cn
URL：http://www.dutp.com.cn
丹东日报印刷厂印装

开本：787×1092 毫米　1/16　字数：468 千字　印张：19.5
印数：30001－36000 册
1988 年 12 月第 1 版　2000 年 8 月第 3 版
2000 年 8 月第 7 次印刷

责任编辑：吕志军　责任校对：李　鸽
封面设计：孙宝福

定价：20.00 元

再版说明

1989年，根据国务院关于高等学校教材工作分工的规定，按照原电子工业部教材办公室的安排，我社承担了全国高等学校工科电子类专业教材的出版工作，出版了《计算机网络》、《数据结构》、《数字电路逻辑设计》和《程序设计语言编译方法》四本电子类专业教材。这些教材来自于教学实践，又经编审委员会小组择优评选出来，所以质量较高。十多年来，这些教材经过大范围的推广发行，得到了广大读者的认可，受到各个学校的广大师生的一致好评，其中《计算机网络》第一版获得第二届全国优秀教材奖，《数字电路逻辑设计》获得原机械电子工业部电子类优秀教材二等奖。

近年来，我社在计算机图书的运作方面投入很大，每年都有大批计算机方面的图书问世，其中不乏各个层次的教材，可以说，我们的计算机教材出版发行工作已有了一个非常厚实的积累。在此基础上，借世纪之交计算机专业教学计划调整的机会，我们组织编写了这套《21世纪高等学校计算机专业规划教材》。

本套教材的编写宗旨是不求大而全，而求简而精。由于教学时数的限制，教材要做到面面俱到不现实，也没有必要，我们主要从先进性、实用性和可操作性角度把握，使本套教材能简捷精炼，重点突出。

本套教材的使用对象是普通高等学校的本、专科生。我们针对这一定位，结合普通高等学校的具体情况，对每本教材的具体内容进行了认真的研究，使其能紧密结合各个学校的教学实际。

为了保证这套教材的质量，我们精心组织了国内多所大学的知名专家、学者作为我们的顾问和编者，他们是清华大学、解放军理工大学、北京联合大学、哈尔滨工业大学、东北大学、大连理工大学等学校的教授，其中有的在本专业有很深的造诣，是国内相应领域的知名专家，有的多年工作在教学和科研第一线，有着丰富的教学经验。强大的编审者队伍给本套教材提供了强有力的技术支持和保证。

本套教材的特点是：

简洁——避开了高深的理论，简明扼要地介绍学生最需要的基础知识和技术；

通俗——通过通俗易懂的语言讲授计算机专业技术知识；

先进——在内容上吸收新技术、新动向，保持一定的前沿性。

实用——本套书能既适合于教，更适合于学，对普通高等学校计算机专业的教学具有较强的适用性。

真心希望本套教材能成为老师的助手，学生的良师益友。

大连理工大学出版社

2000年6月

再版前言

首先感谢广大读者对本书的厚爱，以及大连理工大学出版社的大力支持，使得本书有了一次再版的机会，有了一次增补与修改的机会。

本教材在我院教学实施过程中，同学们的反应是良好的。他们感到本书以形式语言原理为基础介绍各种编译方法与技术，学习起来逻辑性强，容易理解。同时，在教学过程中也发现了不少错误，其中有的是印刷错误、文字错误，有的则是内容错误。这次修订对已发现的错误做了相应的修改，但还不能说已全部修改完。希望广大读者发现问题后，给我来信，以便今后有机会再修改。

在上次修订中增加了词法分析器和语法分析器的自动生成原理。读者可以从这些内容中了解到任何问题只要形式化了，就可以由计算机自动生成。这一观念对目前计算机界正在深入研究的计算机辅助软件工程(CASE)工具有很重要的意义。

在本次再版中又增加了 LR(1)、LALR(1)语法分析器的原理和程序数据流分析技术与代码优化技术。增加这些内容的目的是让读者对编译器技术有全面的了解，有助于实现功能更强的编译器。

另外，这次修订还附上了部分较难习题的答案，为读者自学提供了方便。但一个习题往往可有多种解法，所附答案仅供参考。答案大部分是王智学副教授做的。他用本书讲授编译课多年，提出了不少修改意见。在本次出版前，大连理工大学计算机系林晓惠老师和姚卫红老师仔细地审阅了书稿，指出了许多不足之处。在这里向这三位老师特别表示诚挚的谢意。

最后，再次谢谢读者，谢谢所有关心这本书和为这本书出版做出贡献的人们。

编　者

于解放军理工大学

2000 年 4 月 12 日

前　言

本教材系按电子工业部制定的工科电子类专业教材1986年～1990年编审出版规划，由计算机和自动控制教材编审委员会，计算机教材编审小组组织征稿、评选、推荐出版的。

本教材由南京通信工程学院肖军模同志编著，哈尔滨工业大学郭福顺同志担任主审。

本课程的课内教学时数为60学时，主要介绍程序设计语言的各种编译技术。全书共分为十章：第一章（引言）主要介绍编译技术的基本概念术语；第二章（编译基础）主要介绍与编译技术有关的形式语言理论；第三章（程序语言的结构与定义）介绍语言的组成特点、定义方法；第四章（词法分析）介绍扫描器的各种设计方法；第五章（语法分析）介绍算符优先、递归下降、LL(1)及LR等常用的语法分析技术；第六章（语法制导翻译技术）重点介绍自底向上的语义翻译方法，还介绍递归下降、LL(1)和属性翻译等语法制导翻译技术；第七章（运行时的存贮分配）主要介绍栈式管理技术，对静态与堆式管理技术也做了适当介绍；第八章介绍程序中变量的定值引用的数据流分析技术；第九章（代码优化）介绍利用树结构和DAG对基本块进行局部优化的方法；第十章（目标代码生成）介绍目标代码的生成方法和寄存器分配技术；第十一章介绍各种运行准备工作。在四、六、九、十章中，分别介绍用软件工程观点设计SIMPLE编译器各部分的方法。考虑到不少院校都广泛采用陈火旺等同志主编的《程序设计语言编译原理》一书作为编译教材，所以本书尽量采用《程序设计语言编译原理》中的名词术语，以使得教员在使用本书时感到方便。

使用本教材时，应要求学员耐心阅读书中的程序或算法，并适当多做一些习题或补充题。学好本书最好的方法是结合教学进度完成书中的上机习题（上机实验20小时），编写并调试一个完整的SIMPLE语言的小编译程序，或至少完成其中的词法分析与语法制导翻译部分。书中带“*”的章节只供选用。

参加本教材审阅的同志还有蒋宗礼同志，他为本书提出了许多宝贵意见。另外庄雯同志几乎抄写了全部书稿并做了一些文字加工工作，加快了书稿的完成时间。在这里向他们表示诚挚的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

1988年1月25日

目　录

再版说明
再版前言
前　言
第一章　引　言……(1)
1.1　计算机的语言层次与翻译 ……(1)
1.2　编译的阶段(phase)……(2)
1.3　编译的遍(pass) ……(4)
1.4　SIMPLE 编译的结构设计 ……(5)
第二章　编译基础……(6)
2.1　字母表、串和语言……(6)
2.2　文法的引例 ……(7)
2.3　文法的形式定义 ……(8)
2.4　文法的分类 ……(9)
2.5　正规文法和有穷自动机……(10)
2.5.1　正规集与正规文法……(10)
2.5.2　有穷自动机……(11)
2.5.3　正规文法与有穷自动机……(14)
2.5.4　正规文法和正规式……(15)
2.6*　正规式、NFA 和 DFA 之间的等价变换 ……(16)
2.6.1　从正规式构造 NFA ……(16)
2.6.2　NFA 的确定化 ……(18)
2.6.3　DFA 的最小化 ……(19)
2.7　上下文无关文法与下推自动机……(21)
2.7.1　上下文无关文法……(21)
2.7.2　上下文无关文法的变换……(23)
2.7.3　语法树与文法的二义性……(24)
2.7.4　下推自动机……(26)
习　题 ……(28)
第三章　程序设计语言的构造基础与定义 ……(32)
3.1　程序设计语言的构造基础……(32)
3.1.1　对象的名字、属性与汇聚 ……(32)

3.1.2 数据型对象及其存贮表示…… (33)
3.1.3 运行时的存贮管理…… (36)
3.1.4 运算及运算的复合…… (36)
3.1.5 运算的顺序控制…… (37)
3.1.6 参数传递…… (37)
3.2 程序设计语言的定义…… (40)
3.2.1 程序语言的语法定义…… (40)
3.2.2 程序语言的语义定义…… (45)
3.3 SIMPLE 语言的主要成分 …… (47)
习 题 …… (47)
第四章 词法分析 …… (51)
4.1 单词的种类与机内表示法…… (51)
4.1.1 单词的分类…… (51)
4.1.2 单词的机内表示法…… (51)
4.2 扫描器的任务与设计考虑…… (53)
4.3 扫描器的设计…… (54)
4.3.1 扫描器的界面与数据结构…… (54)
4.3.2 扫描器的数据流图…… (55)
4.3.3 扫描器的详细设计…… (57)
4.4 扫描器的其他设计技术…… (62)
4.4.1 有穷自动机技术…… (62)
4.4.2 正规式技术…… (64)
4.5 某些高级语言的词法分析问题…… (65)
4.6 词法分析阶段的错误处理…… (66)
4.7 扫描器的自动生成原理…… (67)
4.7.1 LEX 语言的基本语句与功能 …… (67)
4.7.2 LEX 编译器原理 …… (69)
习 题 …… (70)
第五章 语法分析 …… (72)
5.1 自顶向下分析法…… (72)
5.1.1 自顶向下分析法中的问题研究…… (72)
5.1.2 左递归与回溯问题…… (74)
5.1.3 LL(1)文法的定义 …… (75)
5.1.4 LL(1)文法的判别 …… (78)
5.1.5 LL(1)文法的讨论 …… (79)
5.2 递归下降法…… (80)
5.3 LL(1)分析法 …… (83)
5.4 自底向上分析法…… (86)

5.4.1 自底向上分析法中的问题研究……(86)
5.4.2 句柄的定义与识别方法……(88)
5.5 算符优先分析技术……(90)
5.5.1 算符优先文法……(91)
5.5.2 算符优先矩阵……(93)
5.5.3 算符优先分析法的实现问题……(96)
5.5.4 算符优先分析的数据流图与模块结构图……(98)
5.5.5 算符优先分析技术的改进……(101)
5.6 LR分析技术……(102)
5.6.1 概 述……(102)
5.6.2 LR(0)分析表的构造方法……(103)
5.6.3 SLR(1)分析表的构造方法……(107)
5.6.4 LR(1)分析表的构造方法……(109)
5.6.5 LALR分析表的构造方法……(114)
5.6.6 二义文法的LR分析法……(120)
5.6.7 LR分析器总控程序……(122)
5.7 语法分析中的错误处理……(123)
5.8 各种语法分析技术的比较……(124)
5.9 语法分析器自动生成的原理……(126)
5.9.1 文法定义语言的功能……(126)
5.9.2 其他文法类分析表的自动生成问题……(128)
习 题……(129)
第六章 语法制导翻译技术……(133)
6.1 语法制导翻译概述……(133)
6.2 中间代码……(134)
6.3 自底向上的语法制导翻译……(135)
6.3.1 自底向上翻译的特点……(135)
6.3.2 说明语句的翻译……(136)
6.3.3 简单赋值句的翻译……(139)
6.3.4 布尔表达式的翻译……(140)
6.3.5 IF语句的翻译……(146)
6.3.6 REPEAT语句的翻译……(148)
6.3.7 FOR循环语句的翻译……(148)
6.4 自顶向下的语法制导翻译……(149)
6.4.1 递归下降的语法制导翻译……(150)
6.4.2 LL(1)语法制导翻译……(153)
6.5 属性文法与属性翻译……(156)
6.5.1 属性文法与L_属性文法……(156)

6.5.2 属性计值规则的改造与属性翻译 …… (158)
6.6 SIMPLE 语言的语法制导翻译程序的设计 …… (163)
6.7 某些复杂语言成分的翻译 …… (172)
6.7.1 分程序结构的符号表 …… (172)
6.7.2 数组变量说明的翻译 …… (174)
6.7.3 过程语句的翻译 …… (176)
6.7.4 过程调用语句和返回语句的翻译 …… (176)
6.7.5 复杂赋值句的翻译 …… (178)
6.7.6 CASE 语句的翻译 …… (181)
6.7.7 输入输出语句的翻译 …… (182)
6.8 P-代码 …… (183)
习 题 …… (186)
第七章 运行时的存贮分配 …… (190)
7.1 数据区的内容 …… (191)
7.2 静态存贮分配 …… (191)
7.3 栈式存贮分配 …… (194)
7.3.1 栈式存贮管理和过程数据区格式 …… (194)
7.3.2 运行时地址的计算 …… (198)
7.3.3 过程调用与返回语句的目标代码 …… (199)
7.3.4 过程的目标代码结构和运行栈的管理 …… (201)
7.3.5 递归过程的调用 …… (204)
7.4 堆式存贮分配 …… (206)
7.4.1 堆空间的分配管理 …… (206)
7.4.2 堆空间的释放与收集 …… (208)
习 题 …… (210)
第八章 程序控制流与数据流分析 …… (213)
8.1 程序控制流图与循环分析 …… (213)
8.1.1 基本块与程序控制流图的构造 …… (213)
8.1.2 查找循环结构 …… (215)
8.2 到达-定值数据流分析 …… (217)
8.2.1 引用与定值(ud)链 …… (217)
8.2.2 到达-定值数据流方程 …… (218)
8.3 活跃变量分析 …… (219)
8.3.1 活跃变量的数据流方程 …… (219)
8.3.2 定值-引用(du)链及其数据流方程 …… (220)
8.4 可用表达式分析 …… (220)
8.5 非常忙表达式分析 …… (222)
8.6 模块间数据流分析 …… (222)

第九章　代码优化…… (224)
9.1　局部优化 …… (224)
9.1.1　局部优化的种类 …… (224)
9.1.2　适合于优化的中间代码 …… (225)
9.2　合并已知量 …… (226)
9.3　利用公共子表达式和消除无用赋值 …… (232)
9.4　循环优化技术 …… (238)
9.4.1　代码外提 …… (239)
9.4.2　强度削弱和变换循环变量 …… (240)
9.4.3　循环展开和循环合并 …… (242)
习　　题…… (243)
第十章　目标代码生成…… (245)
10.1　目标机…… (245)
10.2　各类四元式的翻译方法…… (246)
10.3　寄存器分配与基本块代码的生成…… (248)
10.3.1　引用信息与活跃信息…… (248)
10.3.2　寄存器的分配问题…… (249)
10.3.3　基本块的代码生成算法…… (250)
10.4　DAG 的目标代码生成 …… (252)
10.5　SIMPLE 目标代码生成器的设计 …… (256)
10.6　目标代码的其他优化技术…… (259)
10.6.1　窥孔优化…… (259)
10.6.2　区域内寄存器的分配…… (259)
习　　题…… (260)
第十一章　运行准备…… (262)
11.1　装配程序…… (262)
11.1.1　绝对装配程序…… (262)
11.1.2　重定位装配程序…… (263)
11.2　外部访问的辨认…… (265)
11.3　连接程序(LINKER)…… (266)
11.4　库管理…… (270)
11.5　存贮分配…… (271)
习　　题…… (272)
部分习题参考答案…… (274)
参考文献…… (298)

第一章 引 言

1.1 计算机的语言层次与翻译

在计算机系统中，语言的层次可用图 1-1 来描述。计算机使用的语言可以分为 3 个层次，高级语言层、汇编语言层和机器语言层。将源语言程序转换为另一种语言程序的程序

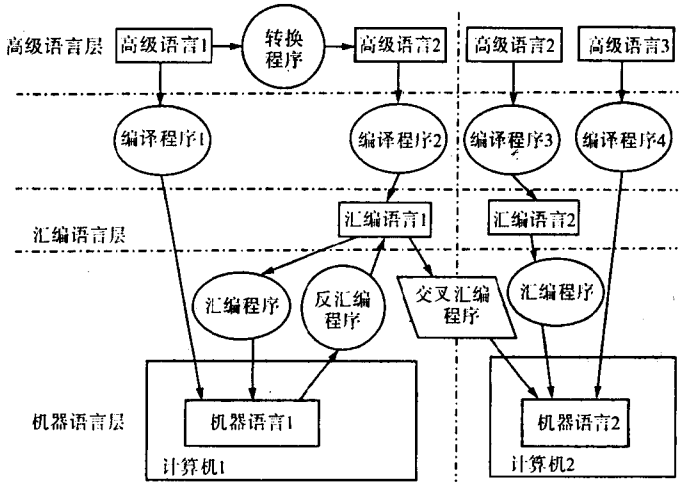


图 1-1 计算机系统中的语言层次和转换关系

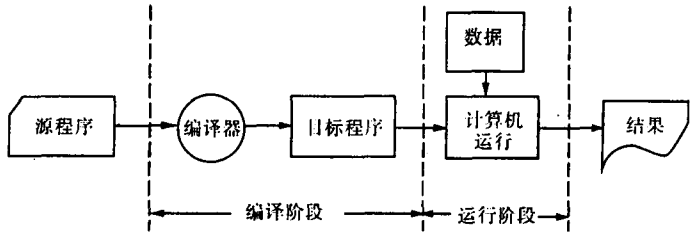
通称为**翻译程序**(translator)。从汇编语言到机器语言的翻译程序称为**汇编程序**(assembler)，而逆向的转换称为**反汇编程序**(disassembler)，在甲机上把乙机的汇编语言程序转换成乙种机器上的机器码的程序称为**交叉汇编程序**(cross assembler)。从高级语言到汇编语言或机器语言的翻译程序称为**编译程序**(compiler)或**解释程序**(interpreter)。两种高级语言之间也可以互相转换，目前尚无通用名称，暂且把它称之为**转换程序**(converter)。机器语言也需要翻译成为有序的逻辑操作，这个工作是靠硬件译码器完成的。

编译程序与解释程序都把高级语言翻译为汇编或机器语言，它们之间的主要差别是：

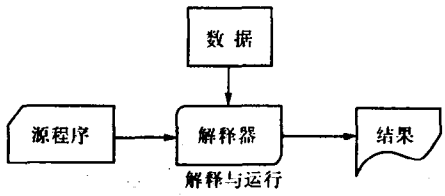
● 编译程序先把全部源程序翻译为目标程序，然后再执行，而且目标程序可以反复执行。

● 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。

编译过程类似笔译，笔译后的结果可以反复阅读。解释过程类似于即席翻译，别人说一句，他就译一句。图 1-2 表示了编译和解释过程。



(a) 编译过程



(b) 解释过程

图 1-2 编译与解释过程

本书将着重介绍编译原理与方法，关于解释程序请参阅本书参考文献 1，2。

1.2 编译的阶段(phase)

本节将以例 1.1 中的小 PASCAL 源程序为例，扼要地介绍编译程序是如何把它翻译为目标程序的。为便于研究和学习，通常把整个编译过程分为若干个阶段，每个阶段完成不同的任务。这些阶段包括：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等阶段。下面分别简要介绍各阶段中的主要任务。

例 1.1 一个简要的 PASCAL 程序

```
1 program examplel;
2 {used for illustrating compiling process}
3 var a,b,c:integer;
4         x: integer;
5 begin
6         a:=a+b*c+b*c;
7         x:=a*3-b*c;
8         b:=a+)b*(x-
9 end.
```

1. 词法分析阶段

每一种高级语言都规定了允许使用的字符集，如字母A～Z，数字 0～9 以及＋、－、*，/、．、，、、(、)等。高级语言的单词是语言中有实在意义的最小语法单位，它们都定义在各语言的字符集上。有的单词仅由一个符号组成，如＋、－、：、；等；有的单词则由两

个或更多的符号组成，如：=、(=、标识符、各类常数、关键字(**begin**,**end**,**program**,**procedure**,…)等。从输入的源程序字符流中逐个地把这些单词识别出来，并把它们表示成机内单词形式(称为 TOKEN 字)，这就是词法分析阶段的主要任务。完成这一任务的程序称为扫描器(SCANNER)。每个 TOKEN 字通常用两部分表示：一是种别码，另一是单词的内部形式。种别码用于区别各单词的种类，如标识符、常数、关键字、界限符等。为简单起见，这里不给出种别码，用符号

%单词符号%

表示某单词的 TOKEN 字。例 1.1 中的源程序经扫描器加工后，输出以下 TOKEN 串：

%**program**%example1%;%**var**%a%,%b%,%c%:%**integer**%;

%x%:%**integer**%;%**begin**%a%:=%a%+%b%*%c%+%b%*

%c%;%x%:=%a%*3%-%b%*%c%;%b%:=%a%+%)%b%

*%(%x%-%**end**%·%#

最后的符号"#"表示源程序结束。可以看出，经扫描器处理后，已经滤除了源程序中不必要的符号，如注解、多余空格等。此外，扫描器还需要指出源程序中的单词错误，但不管句法错误。有时还可能要求扫描器把识别出来的各种名字，如 example1,a,b,c,x 等填入符号表，以备后续阶段查用。

2. 语法分析阶段

语法分析阶段的任务是"组词成句"，把 TOKEN 串按语法规则构成更大的语法单位，如表达式，各类语句乃至程序，并指出其中的语法错误。完成语法分析的程序称为语法分析程序(PARSER)。语法分析程序在分析上述 TOKEN 串之后，应能把各种语句都识别出来，并能指出语句 b:=a+)b*(x--中的错误。

3. 语义分析与中间代码生成

语义分析阶段的任务是翻译由语法分析程序分析出来的各种语句或语法单位的含义。若是说明语句，则把变量的类型等属性填入符号表中；若是表达式，或其他可执行语句，则要把它们翻译为统一格式的中间指令形式(称为中间代码)。中间代码有许多形式，如三元式、四元式、逆波兰式等。

语义翻译工作通常穿插在语法分析过程中，因而语义翻译程序是由一组语义子程序构成的。每当语法分析程序分析出一个完整的语法单位时(如表达式、语句等)，就调用相应的语义子程序执行相应的翻译任务。例如，当语法分析程序分析完说明语句 **var** a,b,c:**integer**;后，应把变量 a,b,c 的类型 **integer** 填入符号表中。当分析到 end 后，将产生以下四元式序列：

(1) (*,b,c,T_1)　　(6)(*,a,3,T_5)

(2) (+,a,T_1,T_2)　　(7) (*,b,c,T_6)

(3) (*,b,c,T_3)　　(8) (-,T_5,T_6,T_7)

(4) (+,T_2,T_3,T_4)　　(9) (:=,T_7,-,x)

(5) (:=,T_4,-,a)

其中，四元式(*,b,c,T_1)表示把 b 与 c 相乘的结果存放在临时单元 T_1 中，其余类同。假定语法分析程序删去了有错误的赋值句。

经语义分析阶段后，源程序被加工为整齐和标准形式的中间代码程序，这一转化是本

质性的。

4. 代码优化阶段

代码优化是对代码的等价变换，优化的目的是提高运行效率，节省存贮空间，有时需在二者之间作出均衡。优化分为两类，一是与机器有关的优化，主要涉及如何分配寄存器，这种优化是在生成目标代码时进行的。另一类是与机器无关的优化，这类优化就是对上述中间代码程序的优化，这些优化包括：局部优化、循环优化等。例如上述四元式序列中，子表达式 b * c 多次重复计算，经局部优化后，得到以下四元式序列，

(1) (* ,b,c,T_1)　　(5) (* ,a,3,T_5)

(2) (+,a,T_1,T_2)　　(6) (−,T_5,T_1,T_7)

(3) (+,T_1,T_2,T_4)　　(7) (: =,T_7,−,x)

(4) (: =,T_4,−,a)

其中 b * c 只计算一次。

编译程序所产生的目标代码质量的高低，主要取决于这一阶段里代码优化程序功能的强弱。

5. 目标代码生成阶段

这一阶段的主要任务是把中间代码程序翻译为具体机器的指令程序。翻译过程中需要涉及具体机器的硬件以及指令和寄存器的选择。

在以上各个阶段中，都要涉及到表格管理和错误处理。一个编译程序在编译过程中，应尽量找出源程序中的错误，向用户提供更多更准确的有关错误信息，以便用户查找和纠正。

图 1-3 给出了一个编译程序的模型。词法、语法和语义分析是对源程序的分析，而代码生成与优化，则是对它的综合。

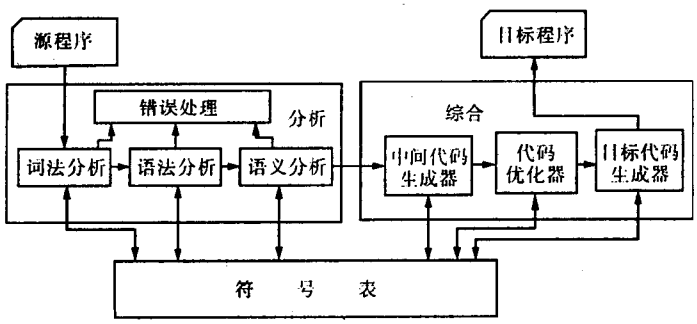


图 1-3　编译模型

1.3　编译的遍(pass)

由于内存的限制和编译程序的庞大等原因，常常把源程序分为几遍来编译，每遍可以完成上述某个阶段的一部分、全部或几个阶段的工作。每遍的结果存入外存贮器里，作为下一遍的输入。遍数的多少是一个技术问题，是和具体机器以及设计者的意愿有关的问题。例如 Gier ALGOL 编译使用的机器字长为 42 位，内存 1024 字，鼓 128 000 字。为能

基本完整翻译 ALGOL 语言，采用九遍的编译程序，参见参考文献 3。另一个例子是 IBM 360 FORTRAN Ⅳ H 编译，这是一个四遍编译。第一遍完成词法和语法分析工作；第二遍完成对公用语句、等价语句的加工、四元式生成以及存贮分配等工作；第三遍代码优化；第四遍目标代码生成。

在内存允许的情况下，减少编译的遍数可以加快编译速度。本书中将逐步引导读者实现一个类 PASCAL 的 SIMPLE 语言的编译程序，可以根据上机次数把它分成三遍或四遍编译。

1.4 SIMPLE 编译的结构设计

编译课程既是一门理论性强又是一门实践性强的课程。如果你想学好编译课，最好能随着课堂教学的深入亲手编制并调试一个完整的小编译程序。为此，在本书各章节中将逐步介绍 SIMPLE 编译的实现方法。由于编译程序是一个复杂的大型软件，应该遵守软件工程的原则去开发。本书将采用自顶向下、逐步细分的技术对 SIMPLE 编译进行模块设计，对编译程序结构进行划分。这里先给出 SIMPLE 编译的初级划分，参见图 1-4。

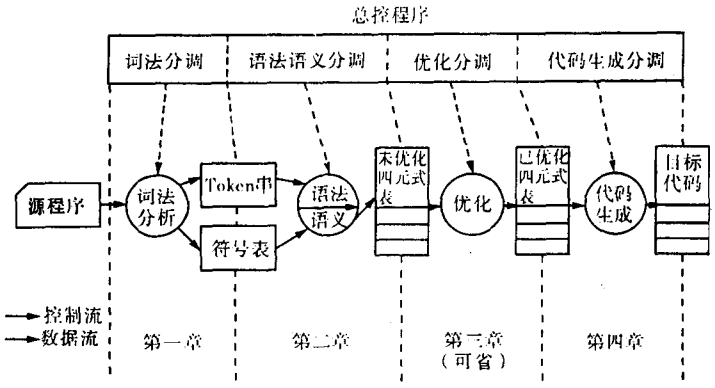


图 1-4 SIMPLE 编译的粗结构

我们将采用三遍编译，目的是为适合教学需要。把词法分析、语法和语义分析与中间代码生成各作为单独的一遍有利于这些程序的编写与调试。每遍的结果保留在磁盘文件上，作为下一阶段程序的输入。

SIMPLE 编译计划用 3 次完成，每次完成上述 1 个程序，总控程序负责调用并打印或显示每遍的输出。如果时间允许的话，还可以编调中间代码优化程序。优化工作也可以作为独立的一遍来实现，这样，SIMPLE 编译就成为四遍编译。

本书有关章节在描述 SIMPLE 编译的各个模块功能时将使用一种结构化语言。这种语言没有严格的定义，选用各种常用的程序语句，如赋值语句、**if** 语句、**while** 语句、**repeat** 语句、**for** 循环语句等，描述中还可以夹杂一些简洁的自然语言。因此，用这种语言描述的功能模块不仅很容易阅读而且很容易被转换成 PASCAL 程序。

第二章　编译基础

自 1956 年 Chomsky 提出形式语言理论后，大大促进了人们对程序设计语言定义方法的研究，并且也促进了编译理论的发展。下面简要地介绍形式语言与文法的基本知识，其中着重介绍正则文法和上下文无关文法及其对应的自动机，并说明它们在编译技术中的应用。

2.1　字母表、串和语言

字母表是一有穷非空字符集。习惯上用 V、Σ 或其它大写字母表示。例如 V＝{a，b，c}，Σ＝{0，1}等都是字母表。

字符串是字母表中符号组成的任何有穷序列。例如 0、1、00、10、100…等都是 Σ 上的字符串。字符串简称为**串**，行。不含任何符号的串称为**空串**，习惯上用 ε 表示。

字符串的运算　设 A、B 为字母表 V 上的字符串集合，x、y 是字符串，在字符串集合上的运算主要有：

- **连接**　串 xy 称为串 x 和 y 的连接。
- **积**　A 与 B 的积 $A \cdot B=\{xy \mid x \in A \text{ 且 } y \in B\}$
- **闭包**　A 的闭包 $A^*=A^0 \cup A^1 \cup A^2 \cup \cdots \cup A^n \cdots$

其中　　$A^0=\{\varepsilon\}, A^i=A^{i-1} \cdot A$

若把 A^* 中的 ε 去掉，就成为 A 的正闭包，用 A^+ 表示，即

$$A^+=A^*-\{\varepsilon\}$$

(形式)语言是一字母表上按某种规则构成的所有串的集合。在语言中，这些串常被称为**句子**或**字**。如果一个语言中含有无穷个句子，则称该语言是无穷的，否则是有穷的。有穷语言可以用穷举法枚举全部句子，但对无穷语言来说，需要解决该语言的有穷表示问题。

有两种表示语言的方法，一种方法用识别的观点表示语言，这种方法是给出一种算法由它判定一个句子是否在语言中。更一般的做法是给出一个过程，它对语言中句子回答“是”而停止，对不是语言中的句子或者不终止，或者回答“否”而停止。过程是一个能够被机械执行的有穷指令序列，例如一个计算机程序。过程的执行未必会终止。如果一个过程的执行总是要终止，那么这种过程就叫**算法**。