编译原理及编译程序构造

秦振松

东南大学出版社

编译原理
及编译程序构造

秦振松

东南大学出版社

内 容 提 要

本书介绍编译理论基础及其实现方法，强调语言的形式化定义、编译技术的各种 概念及实现过程的具体方法。介绍过程以算法为核心，力求简单明了地反映编译的基础知识。从形式语言理论角度讨论词法分析和语法分析技术，为计算机软件工作者开发大型软件打下良好基础。

本书以理论联系实际为宗旨，内容深入浅出，重点突出，并结合构造 EL 语言的编译程序介绍一种常用而又简单的编译方法。

本书可作为高等院校计算机专业的本科或专科教材，也可作为硕士研究生入学考试及计算机软件技术人员的参考书

责任编辑 张 克

编译原理及编译程序构造

秦振松

*

东南大学出版社出版发行

（南京四牌楼 2 号 邮编 210018）

江苏省新华书店经销 南京邮电学院印刷厂印刷

*

开本：787 × 1092 毫米 1/16 印张：18 字数：469 千字

1995 年 12 月第 1 版 2005 年 3 月第 9 次印刷

印数：14501—16500 册

ISBN 7—81050—086—4/TP·13

定价：22.00 元

前　言

编译原理及编译程序构造是计算机专业的一门很重要的专业基础课，它在计算机的系统软件中占有十分重要的地位，是计算机专业学生的一门主修课，也是硕士研究生入学考试的课程之一。通过30多年的研究与开发，该课程在理论上已臻成熟、技术上日趋完善。有关"编译"方面的书籍很多，内容也很丰富，然而，编者通过多年的教学实践，发现现有书籍存在一些不足之处：有的注重于理论完整性，概念多且抽象；有的偏向于具体语言的编译方法，缺乏普遍性；有的内容太多，重点不突出；有的是译文或原文。这些书籍对于初学者来说是比较难学的。其原因是这些书籍缺乏理论联系实际，而该课程又恰恰是理论基础坚实、形式化程度高，同时又是实践性很强的一门课程。因此，理论与实践相联系便是编者编撰这本书的宗旨。

由于这门课所涉及的内容相当广泛，我们只能从中选择最基本的内容介绍之。为了读者参阅其它书籍方便，文中尽量采用国内公认的有关这门课的术语与符号。本书既注重理论基础又注意理论联系实际，在讨论具体实现时都给出了简明算法并通过实例揭示算法的真正含义。为了加深读者对构造编译程序的理解，书末附有EL语言编译程序构造的实践指导供读者在阅读本书之余上机实践用。

全书共分10章：第1章对编译程序的几个主要组成部分作一概述，使读者对全书有一大致了解；第2章介绍编译的基础知识，包括对文法的形式化定义、文法与语言关系及语言的识别方法等；第3章介绍有限自动机、正规文法和正规式三者关系，并在此基础上介绍词法分析；第4～6章介绍预测分析、优先分析和规范分析等三种语法分析方法；第7章介绍翻译方法，以自下而上语法制导翻译为主兼顾介绍自上而下语法制导翻译，强调如何实现自动翻译过程；第8章讨论运行时的数据区存储管理；第9、10章介绍中间代码优化与目标代码生成。

本书拟讲授64～72学时，带"*"的章节只供选用。若本书用于专科教学，可放慢教学进度，语法分析选两种介绍，优化一章仅介绍优化概念一节。学习本书的读者须有程序设计语言（如Pascal或C等）、数据结构、离散数学方面的知识。

本书的编写曾得到复旦大学钱家骅老师的鼓励和支持，在此表示深深悼念。编者曾得到众多同行的支持与鼓励，并吸收到各方面的宝贵意见，在此向他们表示感谢。编者特别感谢孙志挥老师、周佩德老师以及本课程小组的各位老师对编写、出版本书所给予的支持，感谢张幸儿老师、夏德深老师对本书提出许多宝贵意见。

限于编者水平，错误与不妥之处欢迎读者批评与指正。

编　者

1995年11月

目　录

1 引 论

1.1 程序设计语言与编译 …… (1)
1.2 编译程序概述 …… (2)
1.2.1 词法分析 …… (3)
1.2.2 语法分析 …… (3)
1.2.3 中间代码生成 …… (5)
1.2.4 优化 …… (5)
1.2.5 目标代码生成 …… (6)
1.2.6 表格与表格管理 …… (6)
1.2.7 出错处理 …… (8)
1.2.8 遍 …… (8)
1.3 编译程序生成 …… (9)
1.4 编译程序构造 …… (10)

2 编译基础知识

2.1 字母表与符号串 …… (11)
2.1.1 符号串集合的运算 …… (11)
2.1.2 字母表的闭包与正闭包 …… (12)
2.2 文法与语言的关系 …… (12)
2.2.1 文法的概念 …… (12)
2.2.2 文法与语言的形式定义 …… (15)
2.3 文法构造与文法简化 …… (18)
2.3.1 由语言构造文法的例子 …… (18)
2.3.2 文法的简化 …… (19)
2.3.3 构造无 ε 产生式的上下文无关文法 …… (20)
2.4 语法树与文法的二义性 …… (21)
2.4.1 语法树 …… (21)
2.4.2 文法的二义性 …… (22)
习 题 …… (24)

3 词法分析

3.1 正规文法和有限自动机 …… (26)
3.1.1 正规文法、正规集与正规式 …… (26)

3.1.2 有限自动机 …… (28)
3.1.3 正规式与有限自动机之间的关系 …… (33)
3.1.4 正规文法与有限自动机 …… (36)
3.2 词法分析程序 …… (38)
3.2.1 预处理与超前搜索 …… (38)
3.2.2 扫描器的输出格式 …… (39)
3.2.3 扫描器的设计 …… (41)
3.3 词法分析程序的自动生成 …… (47)
3.3.1 LEX 语言 …… (47)
3.3.2 LEX 编译程序的构造 …… (49)
习 题 …… (51)

4 自上而下语法分析

4.1 下推自动机 …… (54)
4.2 自上而下分析法的一般问题 …… (55)
4.2.1 消除左递归 …… (57)
4.2.2 消除回溯——预测与提左因子 …… (59)
4.3 预测分析程序与 LL(1)文法 …… (60)
4.3.1 求串 α 的终结首符集和非终结符 A 的随符集 …… (62)
4.3.2 构造预测分析表 …… (64)
4.3.3 状态表 …… (66)
4.4 递归下降分析法 …… (68)
习 题 …… (74)

5 优先分析法

*5.1 简单优先分析方法 …… (77)
5.1.1 基本思想 …… (77)
5.1.2 有关文法的一些关系 …… (78)
5.1.3 优先矩阵的构造算法 …… (82)
5.1.4 简单优先分析算法 …… (84)
5.2 算符优先分析法 …… (85)
5.2.1 算符优先分析技术的引进 …… (87)
5.2.2 算符优先文法及优先表的构造 …… (89)
5.2.3 算符优先分析的若干问题 …… (92)
5.3 优先函数 …… (96)
习 题 …… (98)

6 LR 分析法及分析程序自动构造

6.1 LR 分析器 …… (101)

6.2 LR(0)项目集族和 LR(0)分析表的构造 …… (104)
6.2.1 LR(0)项目集规范族的构造 …… (106)
6.2.2 LR(0)分析表的构造算法 …… (107)
6.3 SLR 分析表的构造 …… (108)
6.4 规范 LR 分析表的构造 …… (111)
6.4.1 构造 LR(1)项目集规范族的算法 …… (112)
6.4.2 构造 LR(1)分析表算法 …… (113)
6.5 LALR 分析表构造 …… (114)
6.5.1 基本思想 …… (114)
6.5.2 构造 LALR 分析表算法 …… (116)
6.6 二义文法的应用 …… (117)
6.7 分析表的自动生成 …… (121)
6.7.1 终结符和产生式的优先级 …… (122)
6.7.2 结合规则 …… (122)
6.7.3 LR 分析表的安排 …… (124)
习 题 …… (125)

7 语法制导翻译并产生中间代码

7.1 概述 …… (127)
7.2 简单算术表达式和赋值语句的翻译 …… (129)
7.2.1 四元式 …… (129)
7.2.2 赋值语句的翻译 …… (130)
7.2.3 类型转换 …… (132)
7.3 布尔表达式的翻译 …… (133)
7.3.1 布尔表达式在逻辑演算中的翻译 …… (133)
7.3.2 控制语句中布尔式的翻译 …… (134)
7.4 控制语句的翻译 …… (138)
7.4.1 标号和转移语句 …… (139)
7.4.2 IF 语句的翻译 …… (140)
7.4.3 WHILE 语句的翻译 …… (142)
7.4.4 REPEAT 语句的翻译 …… (143)
7.4.5 循环 FOR 语句的翻译 …… (144)
*7.4.6 分情语句的翻译 …… (147)
7.4.7 复合语句的翻译 …… (151)
7.5 数组元素及其在赋值语句中的翻译 …… (152)
7.5.1 数组及下标变量地址的计算 …… (152)
7.5.2 数组元素引用的中间代码形式 …… (154)
7.5.3 按行存放赋值语句中数组元素的翻译 …… (155)
*7.5.4 按列存放赋值语句中数组元素的翻译 …… (158)

7.6 过程调用语句 …… (159)
7.6.1 参数传递 …… (160)
7.6.2 过程调用语句的翻译 …… (161)
7.6.3 过程调用和数组元素相混淆的处理 …… (162)
7.7 说明语句的翻译 …… (163)
7.7.1 分程序结构的符号表 …… (163)
7.7.2 整型、实型说明语句的翻译 …… (166)
7.7.3 常量定义语句的翻译 …… (167)
7.7.4 数组说明语句的翻译 …… (168)
7.7.5 过程说明语句的翻译 …… (169)
7.8 输入/输出语句的翻译 …… (169)
7.9 自上而下分析制导的翻译 …… (171)
7.9.1 算术表达式的翻译 …… (171)
7.9.2 布尔表达式的翻译 …… (173)
7.9.3 简单语句的翻译 …… (175)
7.9.4 LL(1)语法制导翻译 …… (178)
*7.10 属性文法与属性翻译 …… (180)
7.10.1 属性文法与L属性文法 …… (180)
7.10.2 属性翻译 …… (182)
7.11 中间代码的其它形式 …… (184)
7.11.1 后缀式表示法 …… (184)
7.11.2 三元式 …… (187)
7.11.3 间接三元式 …… (188)
7.11.4 树 …… (189)
习 题 …… (190)

8 运行时数据区的管理

8.1 静态存储管理 …… (194)
8.1.1 数据区 …… (194)
8.1.2 公用语句处理 …… (196)
8.1.3 等价语句处理 …… (197)
8.1.4 地址分配 …… (199)
8.1.5 临时变量地址分配 …… (201)
8.2 栈式存储管理 …… (203)
8.2.1 允许过程(函数)递归调用的数据存储管理 …… (203)
8.2.2 嵌套过程语言的栈式存储管理 …… (206)
8.3 堆式存储管理 …… (211)
8.3.1 堆式存储管理技术 …… (211)
8.3.2 堆空间的释放与无用单元收集 …… (214)

习　题 …… (215)

9　中间代码优化

9.1　优化概述 …… (218)
9.1.1　局部优化简介 …… (218)
9.1.2　循环优化简介 …… (219)
9.1.3　全局优化简介 …… (221)
9.2　局部优化 …… (222)
9.2.1　基本块 …… (222)
9.2.2　基本块的 DAG 表示 …… (224)
9.2.3　DAG 在基本块优化中的作用 …… (228)
9.2.4　DAG 构造算法讨论 …… (230)
9.3　控制流程分析和循环查找算法 …… (232)
9.3.1　程序流图与必经结点集 …… (232)
9.3.2　深度为主排序 …… (234)
9.3.3　查找循环算法 …… (234)
9.4　数据流分析 …… (235)
9.4.1　到达 - 定值数据流方程 …… (235)
9.4.2　引用 - 定值链(ud 链) …… (238)
9.4.3　活跃变量及数据流方程 …… (238)
9.5　循环优化 …… (240)
9.5.1　代码外提 …… (240)
9.5.2　强度减弱与归纳变量删除 …… (242)
习　题 …… (243)

10　目标代码生成

10.1　模型计算机的指令系统 …… (248)
10.2　一种简单代码生成算法 …… (249)
10.2.1　活跃信息与待用信息 …… (249)
10.2.2　寄存器和变量地址描述 …… (250)
10.2.3　代码生成算法 …… (251)
10.3　循环中寄存器分配 …… (254)
10.4　DAG 结点的一种启发式排序 …… (256)
习　题 …… (259)

附录　EL 语言编译程序 …… (261)

A　EL 语言文法的扩充 Backus 表示法 …… (261)
B　EL 语言编译程序构造的实践指导 …… (262)
C　扩充的 EL 语言文法与中间代码的解释执行程序 …… (274)

参考文献 …… (278)

1 引 论

本章扼要叙述全书的主要内容,编译程序与高级程序设计语言的关系,简单介绍编译程序几个阶段所完成的任务及编写编译程序的主要方法,最后阐述理论联系实际的重要性。

1.1 程序设计语言与编译

计算机是人们用来进行信息处理的工具。要让计算机进行信息处理,就得把问题告诉给计算机,让计算机按照人的命令进行信息处理。如何把信息传给计算机? 这就要用程序设计语言来实现。

1) 计算机程序设计语言

在计算机领域里,通常人们把计算机可以直接接受的语言(代码)称作机器语言。机器语言是由二进制数(0, 1 序列)组成的,是唯一可以在机器上执行的语言。由于它难读、难写,又与硬件环境关系太密切,现在几乎不为用户直接使用,但作为硬件设计、工业控制场合还是需要的。

汇编语言是对机器语言的一大改进。它用记忆符表示指令的操作码,用标识符表示操作数的地址,这就大大方便了书写与阅读。汇编语言仍然与硬件关系太密切,每一种机器有一套甚至几套汇编语言,给使用者带来困难。汇编语言,机器不能接受它,必须通过汇编程序转换成机器语言之后,才为计算机所接受。

高级程序设计语言是当今使用者普遍采用的一种语言,它彻底摆脱了对硬件的依赖性,它是易读、易写、不易出错的一种语言,而且便于算法交流。但是,这种语言也不为计算机所接受,必须通过编译程序变换成机器语言才能为计算机所接受。现代计算机系统一般含有不止一种高级语言的编译程序,供用户按不同用途进行选择。高级语言编译程序是计算机系统软件的最重要组成部分之一,也是用户最直接关心的工具之一。

根据不同用途产生了各种不同特点的高级程序设计语言,其中具有代表性的计算机语言有 20 多种。如:

面向算法的 ALGOL、Pascal、Modula 语言;

面向系统的 C 语言;

面向数值计算的 FORTRAN、BASIC 语言;

面向数据处理的 COBOL、DBASE、FOXBASE、FOXPRO 语言;

面向对象的程序设计语言 C++;

面向字符串处理的 SNOBOL 语言;

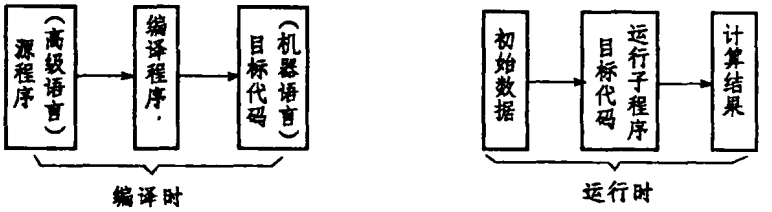
面向人工智能的 PROLOG、LOGLISP、LISP 语言;

以及功能较强、较为通用的 Ada 语言和 PL/1 语言等。

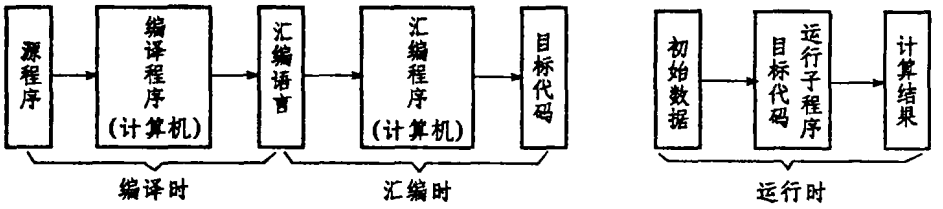
2) 程序设计语言的转换(CONVERSION)

翻译(TRANSLATION)是指在不改变语言的语义条件下,由一种语言翻译成另一种语言,它们在逻辑上是等价的,如 Pascal→C、中文→英文、高级语言→汇编语言……等。

编译是专指从高级语言转换为低级语言(如从高级语言→汇编语言或高级语言→机器语言，也可以从高级语言→汇编语言→机器语言)，然后对编译出来的目标程序进行运行计算。通常编译过程分两个阶段或三个阶段：前一阶段由编译程序(或包含汇编程序)完成，后一阶段由运行子程序配合完成。编译过程可用图1.1表示，它分为编译时与运行时两个阶段的编译过程(图(a))；或者分为编译时、汇编时和运行时三个阶段的编译过程(图(b))。



(a) 分为两个阶段的编译过程



(b) 分为三个阶段的编译过程

图 1.1 编译过程

解释(INTERPRETION)是指接受某高级语言的一个语句输入，进行解释并控制计算机执行，而且马上得到结果，然后再接受一个语句，重复上述过程直至源程序处理结束。也就是它把解释与运行融为一体，不再分解释时与运行时。解释程序执行过程的示意图如图 1.2 所示。解释好比口头翻译，输入一句翻译一句，输入完翻译也完成，并不把源程序转换成目标代码，所以第二次翻译时又得按此过程重新做一遍，因此它是低效的一种方法。对于要反复执行的程序不宜采用解释办法，但用解释办法也有不少优点：直观易懂、解释程序结构简单易于实现，易于实现人机会话等。例如 BASIC 语言是一种结构简单易学易用的会话型语言，一般采用解释办法实现。

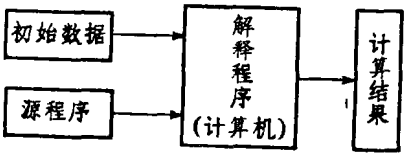


图 1.2 解释执行过程

解释程序与编译程序的结构大同小异，本书主要介绍编译程序构造的理论基础及其实现方法，这些内容绝大部分也适合于后者。对详细的解释程序构造方法，读者可参看有关书籍。

1.2 编译程序概述

编译程序的工作即从输入源程序开始到输出目标程序为止的整个过程，是非常复杂的。一般来说，这整个过程可以划分成五个阶段：词法分析、语法分析、中间代码生成、优化和目标代码生成(有时也分成六个阶段，在语法分析之后加上一个语义分析，不过这个阶段也可归入中间代码生成阶段来完成)，见图 1.3 所示。

在一些微、小型机中，在对编译质量要求不高的场合，可跳过中间代码生成优化两个阶段，由语法分析后直接生成目标代码。这时的语义分析可归入目标代码生成阶段来完成。

下面试以一个包含有错误语句的 Pascal 程序段为例，来看看这五个阶段的工作过程。

假定程序写为：

```
1  program EXAMPLE;
2    var y,c, d: integer;
3        x,a,b: real;
4    begin
5      x: = a + b * 50;
6      y: = c + ) d * (x + b;
7    end.
```

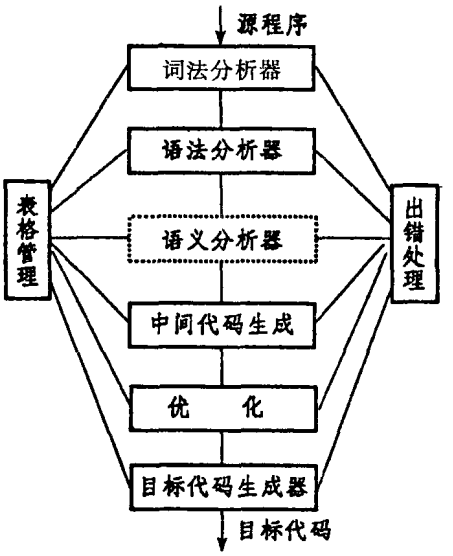


图 1.3　编译阶段

1.2.1　词法分析

单词是高级语言中有实在意义的最小语法单位，而单词又由字符组成。每一种高级语言都定义一组字符集。单词有的由单字符组成，如 +，-，*，/等；有的由两个字符组成，如：=，>=等；有的由一个或多个字符组成，如常数、标识符、基本字或标准标识符等。从输入的源程序字符串中逐个地把这些单词识别出来，并转化成机器比较容易使用的内码形式，这是词法分析的主要任务。一般内码可以用二元式(类号，内码)表示。如上面一段程序可以通过词法分析识别出以下五类单词：

基本字　program，var，integer，real，begin，end

标识符　a，b，c，d，x，y，EXAMPLE

整常数　50

运算符　+，*，:=

界限符　；，：，(，)，.

其中，基本字、运算符、界限符的数目对于一种语言而言是一定的，每一个赋予它一个类号，可以做成一一对应。而标识符与常数是由用户任意使用的，其数目无限。解决办法是给标识符分配一个类号，不同的标识符用它的符号表入口地址(或变量地址)来区分，将这些地址当作内码给出。同样，将常数分为若干类：整型常数、实型常数、字符常数、布尔常数，然后也用它们的常数表入口地址作为内码给出。在这个过程中需要查造符号表和常量表并进行出错处理。

总之，词法分析是扫描源程序字符串，按词法规则识别出正确的单词，并转换成统一规格(类号、内码)交语法分析使用。

1.2.2　语法分析

语法分析阶段的任务是组词成句。每一种语言都有一组规则，称之为语法规则或文法。按照这些文法，可以由单词组成语法单位，如短语、语句、过程和程序。语法分析就是通过语法分解，确定整个输入串是否能构成语法上正确的句子和程序等。

语法规则写成 Backus-Naur-Form 式，简称 BNF。其形式是 A::=B | C，读做“A 定义为 B 或 C”。

下面是赋值语句的语法规则，A 是 Assignment 的缩写，意思是赋值语句，称 A 为规则的开

始符号,而 V, E, T, F, C 分别是变量、表达式、项、因子、常数的意思。每一行称作一个规则。

A:: = V: = E

E:: = T | E + T

T:: = F|T * F

F:: = V|(E)|C

V:: = 标识符

C:: = 常数

当然,还有其它语句的语法规则,如条件语句、循环语句、过程调用语句和说明语句等规则。上面的 EXAMPLE 程序的第 2、3 行是说明语句。语法分析对说明语句的处理主要是填写符号表,而对一般语句处理则是构造语法树。

语法分析有两种方法:推导(derive)和归约(reduce)。以推导为例,推导是从文法的开始符号开始,按照语法规则,每次选择某规则右部的一个候选式取代左部直至识别了句子或者找出错误为止。

下面就以上述程序的第 5、6 行语句为例,看看它们的分析过程。采用最右推导,语句 x: = a + b * 50的分析如下:

$$A \Rightarrow V := E \Rightarrow V := E + T \Rightarrow V := E + T * F \Rightarrow V := E + T * C \Rightarrow V := E + T * 50$$

$$\Rightarrow V := E + F * 50 \Rightarrow V := E + V * 50 \Rightarrow V := E + b * 50 \Rightarrow V := T + b * 50$$

$$\Rightarrow V := F + b * 50 \Rightarrow V := V + b * 50 \Rightarrow V := a + b * 50 \Rightarrow x := a + b * 50$$

这个过程可以用一棵倒立的语法树来描述(见图 1.4(a))。把语法树末端符从左到右连结起来,正是要求识别的语句,所以该语句为正确的语句。

再看另一语句 y: = c +)d * (x + b 的分析过程:

$$A \Rightarrow V := E \Rightarrow V := E + T \Rightarrow V := E + F \Rightarrow V := E + V \Rightarrow V := E + b \Rightarrow V := T + b$$

$$\Rightarrow V := T * F + b \Rightarrow V := T * V + b \Rightarrow V := T * x + b$$

再也无法继续往下推导了,这表明这时的')'号是错的,即输入串有错。如果也画成语法树(见图 1.4(b)),按其从左到右的末端符连结再也不是被识别的句子。

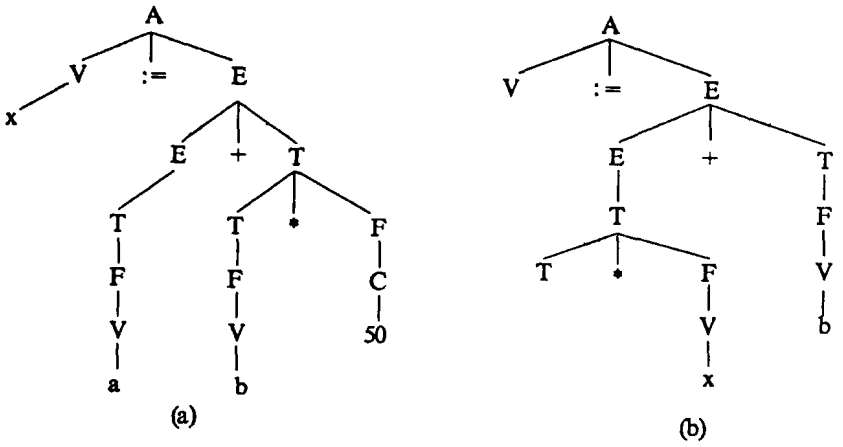


图 1.4 语法树

1.2.3 中间代码生成

中间代码是在语法分析正确的基础上，按照相应语义规则产生的一种介于源语言与目标代码之间的一种代码，这种代码不依赖于机器但又便于产生依赖于机器的目标代码。中间代码形式有多种：四元式、三元式和逆波兰形式等。其中用得最广的是四元式。源语句 x：= a + b * 50 转换成中间代码形式是：

```
T1: = inttoreal (50)        /* 整常数转换成实常数的四元式表示法 */
T2: = b * T1                /* 或写作( * , b, T1, T2) */
T3: = a + T2                /* 或写作( + , a, T2, T3) */
x: = T3                     /* 或写作(: = , T3,_, x) */
```

中间代码是为后续的优化和目标代码生成提供方便，因此中间代码的选择往往与所采用的优化技术和计算机硬件结构有关。

1.2.4 优化

优化的任务在于对前一阶段产生的中间代码进行加工变换，以期在最后阶段能产生出更为高效(省时间、省空间)的目标代码。譬如上面的中间代码可变换成如下两句四元式代码：

```
T1: = b * 50.0
x: = a + T1
```

优化主要包括：删除公共子表达式、合并已知量、删除无用赋值、循环优化等。优化所依据的原则是程序的等价变换规则。例如，把程序段

```
FOR K: = 1 TO 100 DO
    BEGIN
      M: = I + 10 * K
      N: = J + 10 * K
    END;
```

转换成的四元式和经优化后产生的四元式用下表表示：

转换成四元式	经优化后产生四元式
(1) K: = 1	(1) K: = 1
(2) if K > 100 goto (9)	(2) T_1: = 10 * K
(3) T_1: = 10 * K	(3) R: = 10 * 100
(4) M: = I + T_1	(4) if T_1 > R goto (9)
(5) T_2: = 10 * K	(5) M: = I + T_1
(6) N: = J + T_2	(6) N: = J + T_1
(7) K: = K + 1	(7) T_1: = T_1 + 10
(8) goto (2)	(8) goto (4)
(9) …	(9) …

显然，经优化后循环体缩小了，原来要做701条指令现在只做503条指令；原来要做300次加法，200次乘法，现在只做300次加法2次乘法。

编译程序所产生的目标代码质量的高低，主要取决于代码优化程序功能的强弱。当然，若要求优化结果的质量越高，所付出的代价也就越大，因此只能根据具体情况，适可而止。

1.2.5 目标代码生成

这一阶段主要任务是把中间代码程序转换为具体机器的指令序列。转换过程需涉及具体机器的指令系统以及寄存器分配等硬件功能。例如，上述经优化后的中间代码，可生成如下用汇编语言表示的目标代码：

```
LOAD    R2, b
MUL     R2, 50.0
LOAD    R1, a
ADD     R1, R2
STORE   R1, x
```

1.2.6 表格与表格管理

编译的五个阶段都需要与表格打交道，用以记录源程序的各种信息以及编译过程中各种状况，以便后继阶段使用。也即在编译过程的各个阶段都有查造表、填表等功能。一般而言，与编译的头三个阶段有关的表格有：

符号表：登记源程序中的常量名、变量名、数组名、过程名等的性质、定义和引用状况；

常数表：登记源程序中出现的各种类型字面常数(直接量)的值；

标号表：登记源程序中出现的标号的定义和引用情况(此表可与符号表合并)；

分程序入口表：登记过程的层号、分程序符号表的入口(指分程序结构的语言)等；

中间代码表：记录四元式序列的表。

这些表的格式一般分为两栏。

NAME(名 字)	INFORMATION (信息)

例如，对于FORTRAN程序段：

```
SUBROUTINE INCWAP(M,N)
10    K = M + 1
      M = N + 4
      N = K
      RETURN
      END
```

能构造如下表格：符号表(表1.1)，常数表(表1.2)，入口名表(表1.3)，标号表(表1.4)，以及四元式序列表(表1.5)。有时入口名表也可以并入符号表。

表 1.1 符号表

	NAME	INFORMATION
1	M	哑元、整形、变量地址
2	N	哑元、整型、变量地址
3	K	整型、变量地址

表 1.2 常数表(CT)

	值
1	1
2	4

表 1 3 入口名表

NAME	INFORMATION
…	…
INCWAP	二目子程序、四元式序号 1

表 1 4 标号表

NAME	INFORMATION
…	…
10	四元式序号 4

表 1.5 四元式序列表

序号	OP	ARG_1	ARG_2	RESULT
1	link	–	–	–
2	actpar	INCWAP	1	M
3	actpar	INCWAP	2	N
4	+	M	1	K
5	+	N	4	M
6	:=	K	–	N
7	paract	INCWAP	1	M
8	paract	INCWAP	2	N
9	return	–	–	–

其中符号表记录了源程序中出现的三个变量名 M、N 和 K 的有关性质;CT 表记录了常数 1 和 4 的值(已经是内部二进制代码);入口名表记录了子程序名 INCWAP 的入口地址,即为四元式表的序号 1;标号表记录了标号 10 对应的四元式序号 4;四元式序列表记录了源程序翻译成的四元式序列。其中:

(link,_,_,_)　　表示保护返回地址和有关寄存器内容;它相当于宏
(actpar, INCWAP, 1, M)　表示传递第一个实变元到 M 单元
(actpar, INCWAP, 2, N)　表示传递第二个实变元到 N 单元
(+,M,1,K)　　表示 K:=M+1
(+,N,4,M)　　表示 M:=N+4
(:=, K, _, N)　　表示 N:=K
(paract, INCWAP, 1, M)　表示把 M 送回到第一实变元所指地址单元
(paract, INCWAP, 2, N)　表示把 N 送回到第二实变元所指地址单元
(return,_,_,_)　　表示恢复寄存器内容,并把控制返回到调用程序

注意:在四元式表中实际上不是直接写上操作数(或结果数)的名字而是填上有关表格的入口地址或序号。

当着手为某种语言在某个机器上设计编译程序时,首先必须根据用户的整体要求(例如对于优化方面的要求),审慎地选择中间代码,周密地考虑各种全局性名表(即各个阶段都要用到的表格)的信息安排。这些事情出了差错必导致后来的大返工,甚至招致失败。

在编译过程中,随着源程序的不断被改造,编译的各阶段常常需要不同的表格。例如,语

法分析阶段和目标代码生成阶段所需要的表格就有很大差别。由于各种信息是被保存在各种不同表格中，因此编译过程的绝大部分时间是花在查表、造表和更新表格的事务上。所以，选择一种好的表格结构和查找算法对于构造编译程序来说是至关重要的。

在大多数的编译程序中，表格的构造、查找和更新通常是由一组专门的程序来完成的，这组程序称为表格管理程序。对编译程序而言它们是工具，事先编制好、供需要时调用。这部分内容在数据结构课中学过，本课程不再详细介绍。

1.2.7 出错处理

如果源程序有错误，编译程序应设法发现错误，并把有关的出错信息报告给用户，这部分的工作是由专门的一组程序（叫做出错处理程序）完成的。一个好的编译程序应该能最大限度地发现源程序中的各种错误，指出错误的性质和发生错误的位置（用源程序的行号、列号定位），并且能将错误所造成的影响限制在尽可能小的范围内，使得源程序剩余部分能继续被编译下去，通常是跳过所在语句，接着分析后继语句，以便进一步发现其它可能的错误。如果能让机器自动地校正错误，那当然最好，迄今为止许多人花了很大力气做这件事，但收效甚微。因为，有的错误甚至是无法自动改正的。

查错也是不容易的，往往一个错误掩盖另一错误或者一个错误诱发多个错误，所以查错也没有形式化的办法解决，本书也不打算深入探讨查错与纠错问题。

1.2.8 遍

遍是编译程序从外部介质（磁盘或其它外部存储器）读取源件（源程序或中间代码）经过加工获得某种结果件（中间代码或目标代码）并将其送回外部介质的过程。所以，遍与阶段的含义毫无关系。

有的编译程序把编译的五个阶段工作结合在一起，通过对源程序的从头到尾扫描完成编译的各项工作，把源程序翻译成可在机器上运行的目标程序。这种编译程序称作一遍扫描。它的内部结构不再分成五个阶段，而是互相穿插进行，如图1.5所示。它是以语法分析为核心，当分析需要源程序的单词时，向扫描器发出取单词的调用命令，扫描器送回单词的信息（类号，内码）；当分析到进行归约时，调用语义子程序，产生目标代码，并返回有关信息，让语法分析继续进行。

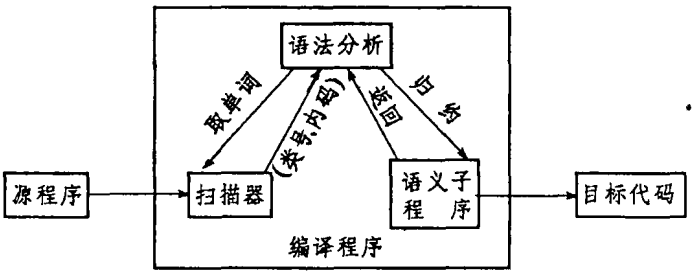


图 1 5 一遍扫描编译过程

有的编译程序需要多遍扫描才能完成，至于采用何种方式是根据具体情形决定的，一般是