专升本

教育部师范教育司组织编写
中学教师进修高等师范本科(专科起点)教材

编译原理

徐国定　主编

高等教育出版社

编译原理

教育部师范教育司组织编写
中学教师进修高等师范本科（专科起点）教材

编 译 原 理

徐国定 主编

高等教育出版社

内容提要

本书根据教育部师范教育司制订的《中学教师进修高等师范本科(专科起点)教学计划》编写而成。编译原理是大学计算机系本科的一门专业基础课程。本书系统地介绍了编译程序的构造方法。内容涉及词法分析、句法分析、语义分析、目标代码生成、代码优化和出错处理。为了让读者深刻了解编译程序的工作过程，本书以自动机理论为模型叙述了经典的词法分析和句法语义分析的方法。本书还给出了代码生成的详细过程，每章后还对本章内容进行小结并配有习题，以帮助读者理解和掌握本章内容。通过本书的学习，可以使读者对编译程序的工作原理有一个完整的了解，并具有设计编译程序的基本能力。

本书可以作为中学教师进修本科(专科起点)编译原理课程的教材或参考书，也可作为普通本科、其他类“专升本”或成人本科教育的教材。

图书在版编目(CIP)数据

编译原理/徐国定主编. —北京：高等教育出版社，2001

ISBN 7-04-009247-6

I. 编… II. 徐… III. 编译程序-基本知识 IV. TP314

中国版本图书馆 CIP 数据核字（2001）第 10528 号

编译原理
徐国定 主编

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街 55 号	邮政编码	100009
电　　话	010-64054588	传　　真	010-64014048
网　　址	http:// www.hep.edu.cn http:// www.hep.com.cn		
经　　销	新华书店北京发行所		
印　　刷	河北新华印刷一厂		
开　　本	787×960　1/16	版　　次	2001 年 6 月第 1 版
印　　张	13.5	印　　次	2001 年 6 月第 1 次印刷
字　　数	240 000	定　　价	11.90 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

编译程序是计算机系统配置的基本系统软件。它在整个计算机的系统软件中占据十分重要的地位。编译程序的作用是把用户用高级语言所书写的源程序翻译成用低级语言表示的目标程序，从而为计算机系统执行用高级语言描述的源程序作必不可少的准备。

本书以作者近几年来在华东师范大学计算机科学技术系讲授此课程编写的教材为基础，几经修改整理而成。在编撰的过程中，由于编译程序所涉及的内容十分广泛，我们只能选择其中最基本的内容作必要的叙述，帮助读者掌握编译的基本原理和方法，同时我们力求理论联系实际，为学生日后在实际工作过程中进行程序设计及研读有关文献打下基础。

全书共分九章。第一章描述了编译程序的基本结构；第二章介绍形式语言的基本知识，该章是学习其他各章的基础；第三章讲解有限状态自动机、正则表达式和词法分析程序的设计；第四章和第五章详细介绍了句法分析的主要方法；第六章以句法制导翻译为工具，讨论了程序语言常见结构的翻译方法和符号表的构造；第七章详细讨论了运行时刻存储和环境的管理；第八章介绍了一个代码生成程序的构造，使学生对代码生成程序有一个完整的了解，该章还对代码优化所涉及的主要问题和方法作了概述；第九章讨论了出错恢复的常用方法。

为方便教师使用本书，对学时分配建议如下：

第一章	概论	2	学时
第二章	形式语言基础	6	学时
第三章	有限状态自动机和词法分析	6~8	学时
第四章	自顶向下句法分析	6~8	学时
第五章	自底向上句法分析	6	学时
第六章	中间代码生成和符号表	10	学时
第七章	运行时刻存储和环境管理	10	学时
第八章	代码生成	8~10	学时
第九章	出错恢复	2	学时

讲授本书大约需要 60 学时左右。在本书编写过程中所参考的文献资料，我们一并列于本书的末尾，书中不再一一列出。

本书由华东师范大学徐国定主编，由西南师范大学张为群主审。本书的

第二章和第三章由华东师范大学罗雪平、郑奕莉和徐国定共同编写，其余各章由徐国定编写。由于编者的水平有限，书中如有不当之处，恳请广大读者、教师指正。

编 者

2000 年 8 月于华东师范大学

目　录

第一章　概　　论

1.1　程序语言和语言处理程序

程序语言在计算机科学中占有特殊的地位，是遵守一定书写规范的语言，人们通过程序语言描述计算和解决实际问题的过程。程序语言的设计必须让人们易于读写，它也应该让计算机能够理解和处理。

程序语言可以划分为低级语言和高级语言两大类。低级语言又称面向机器语言，它是特定的计算机系统所固有的语言。

世界上第一台计算机问世至今已有半个多世纪，现代计算机的运算速度已是人类第一台计算机所不可比拟的。但是，当前计算机硬件仍然只能理解机器自己的语言——机器指令。计算机的机器指令相当原始，它通过电子线路对取值为 0 和 1 的位进行操作。用机器语言进行程序设计，需要对机器结构有较多的了解。用机器指令编制出来的程序可读性很差，程序难以修改和维护。为了提高程序设计效率，人们考虑用有助记忆的符号来表示机器指令中操作码和操作数。例如用 ADD 表示加法操作码，SUB 表示减法操作码，等等，这就是汇编语言。使用汇编语言编制程序时，用户可以不必使用数字来表示机器指令的操作码和操作数地址，而采用符号的含义和功能十分接近，用户比较容易记住的助记符。这样，用户可以比较方便地表达自己的思想，编制程序的效率和程序的可读性都提高了。然而，汇编语言是一种和计算机的机器语言十分接近的语言，它的书写格式在很大程度上取决于特定计算机的机器指令，因此它是一种低级语言。

人们在使用汇编语言编写程序时，发觉它仍未能摆脱机器指令的束缚，这对于人们抽象思维和交流十分不便，由此高级语言的建立被提到日程上来。目前已有许多高级语言，如 FORTRAN，COBOL，Pascal，C，等等。这类语言与人们的自然语言比较接近，因而极大提高了人们进行程序设计的效率，同时也方便了人们用这类语言进行交流。

今天的计算机仍然只能理解和执行机器语言，而各种程序语言只能是人和机器之间进行信息交流的“媒介”。程序语言的引入意味着：必须有一个程序，其任务是使计算机能够理解用某一程序语言书写的用户程序，担负这一工作的程序称为“语言处理程序”，它可以分为两大类：解释程序和翻译程序。

解释程序是一种语言处理程序，它直接执行源程序或源程序的内部形式。因此，解释程序并不产生目标程序，这是它和编译程序的主要区别。图 1.1 显示了高级语言实现的 4 种可能情况。

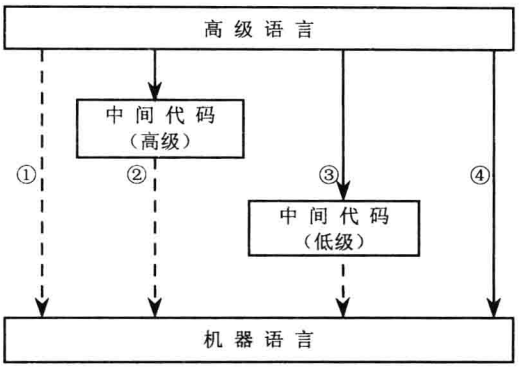

图 1.1 高级语言的 4 种实现方案

在图 1.1 的实现方案①中，源程序被直接解释执行。这种解释程序对源程序进行逐个字符检查，然后执行规定的动作。例如，扫描到字符串序列：

go to L

解释程序就开始搜索源程序中标号“L”后面紧跟冒号“：”的位置。这类解释系统在实现时需要反复扫描源程序，这使得按方案①实现的解释程序效率很低。

在实现方案②中，首先将源程序翻译成高级中间代码，然后再扫描高级中间代码，对高级中间代码进行解释执行。通常，在高级中间代码和高级语言的语句之间存在着一一对应关系。APL 和 SNOBOL 4 语言的实现很多都采用这种方法。

在实现方案③中又是一种解释程序的实现方法。实现方案③的解释程序和实现方案②的解释程序不同点在于，实现方案③的解释程序首先将源程序转化成和机器代码十分接近的低级中间代码，然后再解释执行这种低级中间代码。一般说来，在这种实现方案下，高级语言的语句和低级中间代码之间存在着 1—n 的对应关系。在编译程序基本原理有关章节中，将会讨论逆波兰表示。逆波兰表示常常被用来作为低级中间代码。Pascal 解释系统就是这类解释程序的一个实例，它在词法分析、句法分析和语义分析基础上，先将源程序翻译成 P-代码，一个非常简单的解释程序解释执行这种 P-代码。这类系统具有良好的可移植性。

翻译程序是另一种语言处理程序。翻译程序接受所输入的用程序语言书

写的程序，然后将它翻译成用另一种语言书写的与源程序等价的程序(称为目标程序)。如果源语言是汇编语言，而目标语言是机器语言，则一般将这种翻译程序称为汇编程序。如果源语言是高级语言，而目标语言是低级语言(汇编语言或机器语言)，则这种翻译程序一般称为编译程序。实现方案④就是普通的编译程序。在编译程序方案下，用高级语言书写的源程序最终被翻译成用机器语言(或汇编语言)表示的目标程序，其间还可能进行代码优化工作。因此，这类系统目标程序的执行效率最高。

在编译程序实现方案下，用高级语言书写的程序的执行一般分几步完成：源程序首先被翻译成目标程序，如果目标程序是用汇编语言表示的，则还需经过汇编程序进一步转换成用机器语言表示的目标程序，才能最后将目标程序装入并执行。

1.2 解释程序概况

一般说来，建立在翻译基础上的系统在执行速度上都优于建立在解释执行基础上的系统。翻译程序的缺点是复杂，这使得它的开发和维护费用都较大。相反，解释系统比较简单，可移植性较好，适合于以交互方式执行的程序，其缺点是执行速度慢。

下面对解释系统的结构作一扼要的描述。解释系统通常可以分成两部分，第一部分包括通常的词法分析程序以及句法和语义分析程序，它的作用仍是把源程序翻译成中间代码，中间代码的设计常常采用逆波兰表示形式。第二部分是解释部分，用来对第一部分所产生的中间代码进行解释。由于真正的解释工作在解释部分完成，下面介绍仅涉及第二部分。

这里用数组 MEM 来模拟计算机内存，(源程序的)中间代码程序和解释部分的各个子程序都存放在数组 MEM 中。全局变量 PC 是一个程序计数器，它记录了当前正在执行的中间代码位置。这种解释部分的常见总体结构可以由下面两部分组成：

I1： PC:＝PC+1

I2： 执行位于 opcode_table［MEM［PC］］的子程序(子程序执行后返回前面 I1)。

用一个简单例子来说明其工作情况。设有两个实型变量 A 和 B，A 和 B 相加的中间代码如下：

```
Start: Ipush
       A
       Ipush
```

```
        B
        Iaddreal
```

其中，中间代码 Ipush 和 Iaddreal 实际上都是 opcode_table 表的索引值(即位移)，而该表单元中存放的值为：

```
opcode_table［Ipush］=push
opcode_table［Iaddreal］=addreal
```

就是对应的解释子程序的起始地址，A 和 B 都是 MEM 中的索引值，解释部分开始执行时，PC 的值为 Start–1，解释部分可表示如下：

```
interpreter_loop:
        PC:=PC+1;
        goto opcode_table［MEM［PC］］;
push:
        PC:=PC+1;
        stackreal(MEM［MEM［PC］］);
        goto_interpreter_loop;
addreal:
        stackreal(popreal( ) + popreal( ));
        goto interpreter_loop;
        … (其余各解释子程序)
```

其中，stackreal( )表示把相应值压入下推栈，而 popreal( )表示把下推栈栈顶值取出，然后弹出栈顶元素。这里所说的下推栈是解释部分工作时的一个工作栈。

1.3 编译程序概况

编译程序的职能是将用某程序设计语言书写的程序(源程序)翻译成与之等价的机器语言或汇编语言程序(目标程序)，这里所说的等价是指目标程序能完成源程序预定的任务。编译程序的工作实际上相当复杂，一般可把编译程序分成下面几部分：① 词法分析；② 句法和语义分析；③ 代码优化；④ 代码生成；⑤ 符号表管理。各部分之间关系，如图 1.2 所示。

① 词法分析

词法分析程序是编译程序的第一部分，它的输入就是源程序中由字符所组成的一串符号。词法分析程序将源程序的这种外部形式转换成更适合编译程序其余几个部分处理的内部形式，词法分析程序有如下功能：

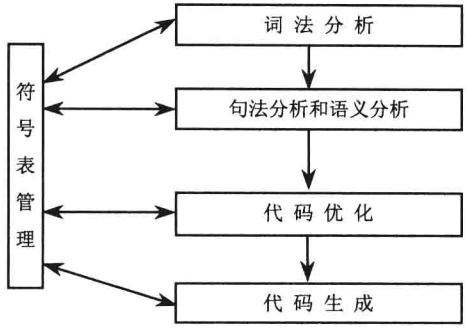

图 1.2 编译程序的 4 个阶段

(a) 识别出源程序中意义独立的最小词法单位——单词；

(b) 删除无用的空格、回车和其他与输入介质有关的符号；

(c) 删除程序员为了提高程序可读性所加的注解；

(d) 如果发现错误则报告出错。

作为例子，观察下面输入符号串：

if B1=24 then goto L2

在程序中，用作变量、过程和标号名的标识符是一个整体，代表(数值或非数值)常量的符号串也是一个整体。程序语言中规定的其他符号，如 if，while，=，；，等等，都各自有独立的含义。一般经过词法分析程序处理，每个单词包含 2 个基本信息：

(类别，值)

单词的类别指出该单词属于哪一类，而单词的值则把这一个单词和同类中其他单词区分开来。一个程序语言的单词究竟分成几类，由编译程序设计者决定。一个常见的单词类别划分方法是：标识符自成一类，常量归为一类，而所有其他符号都各自成为一类。当一个单词是标识符时，其值可能是该标识符在符号表登记项的地址。完全类似，类别为常量的单词的值是该常量在常量表登记项的地址。这样，上述语句应该被分解成 7 个单词：(if，)，(标识符，B1 在符号表地址)，(=，)，(常量，24 在常量表地址)，(then，)，(goto，)，(标识符，L2 在符号表地址)。

② 句法和语义分析

经过词法分析程序处理后，源程序就转化为单词串。每个单词都是一个意义独立的单位，它所包含的信息量个数固定。这种大小固定、分类明确的单词串对于下面句法和语义分析程序的处理十分有益。程序语言是结构严格的语言，它有一组文法规则。句法和语义分析程序接受词法分析程序的输出(即单词串)，然后检查其是否符合该语言的文法规则。一旦句法分析程序分解出其

中一个文法结构，该结构的语义分析程序就进行相应的语义检查。如果需要，就输出相应的内部代码(称为中间代码)，这种中间代码可以理解成假想计算机的指令，其执行次序反映了用户程序的原始含义。例如，当句法和语义分析程序处理赋值语句：

A:=B+3*6

的相应单词串时，由于每一个运算符只能以特定的数据类型中的元素作为其运算对象，它就依次检查各运算符两边的运算对象类型是否相同，如果运算对象不是所预期的类型，它就会报告出错。否则，它输出相应的中间代码。下面是此赋值语句的一种中间代码的表示形式：

$(*, C_3, C_6, T_1)$

$(+, i_B, T_1, T_2)$

$(:=, T_2, i_A)$

其中，T_1，T_2是编译程序所生成的临时变量，C_3表示单词“常量3”，i_A表示单词“标识符 A”，等等。这样，词法分析程序输出的 7 个单词经过句法和语义分析程序的加工变成 3 条意义明确的中间代码。这 3 条中间代码分别表示“将常量 3 和常量 6 的乘积送入临时变量 T_1”，“将变量 B 和临时变量 T_1 的和送入临时变量 T_2”，以及“将临时变量 T_2 的值送入变量 A”，这一次序恰好反映原先输入的赋值语句的含义。句法和语义分析程序是编译程序中的关键部分。

③ 代码优化

中间代码可以直接由代码生成程序翻译成对应的机器指令(或汇编指令)。然而，亦可在句法语义分析程序和代码生成程序之间插入代码优化程序，代码优化程序对所输入的中间代码进行改动，将它修改成一种更有效的形式。常量合并就是代码优化程序所作的工作之一，即在编译时刻计算原先在运行时刻所要进行的常量运算。例如，上述赋值语句 A:=B+3*6 中，子表达式 3*6 可以由 18 来代替，得到 A:=B+18，这样做完全不改变原赋值语句的含义。下面是代码优化程序所输出的中间代码：

$(+, i_B, C_{18}, T_1)$

$(:=, T_1, i_A)$

④ 代码生成

代码生成程序负责将所输入的中间代码串翻译成对应的机器指令。例如，对于上述未经优化的中间代码，代码生成程序可能生成下面目标代码：

```
LOAD   0, d3(3)
LOAD   1, d6(3)
MPY    0,1
```

```
LOAD   2,dB(3)
ADD    2,1
STORE 1,dA(3)
```

这里假定变量 A，B 都是整型变量，上述指令都是定点指令，上述指令组中 d3,d6,dA 和 dB 都是 3，6，A 和 B 的存储单元地址相对于某一地址(假定该地址已在 3 号基地址寄存器 R3 中)的位移。而对于上述经过优化处理的中间代码，代码生成程序只需生成：

```
LOAD   0,d18(3)
LOAD   1,dB(3)
ADD    0,1
STORE 1,dA(3)
```

很显然，从所占用的存储空间和执行时间上来看，后者的目标程序都要比前者好。

⑤ 符号表管理

编译程序在完成其任务过程中，还有一些其他工作要做，包括符号表管理和出错恢复。编译程序中符号表登录了源程序中出现的每一个标识符及其属性。这样，在整个编译阶段都可以了解某标识符的属性，如标识符是如何说明的(如实型量，数组等)；如果为数组，数组维数是多少；所属过程的静态嵌套层号、所需存储单元数、所分配的内存单元的地址(或形式地址)是多少。这些都是标识符的属性，出错恢复程序一般由句法和语义分析程序调用。一旦句法分析程序发觉源程序有错误，无法继续其正常工作时，出错恢复程序就开始工作，其任务是诊断源程序错误性质，并做某种恢复工作。例如，删去或插入某些单词，以使句法分析程序可以继续工作。

小 结

在本章结束时，学员应该了解：

1. 什么是语言处理程序、解释程序、翻译程序、汇编程序和编译程序。
2. 翻译程序的基本工作原理。
3. 编译程序的组成。

习 题 一

1. 试给出下述名词的含义：语言处理程序、解释程序、翻译程序、汇编程序和编译程序。
2. 试述编译程序的各个组成部分，每一部分的功能。

第二章　形式语言基础

2.1　上下文无关文法

众所周知，英文中句子是在字符集基础上构造而成的，该字符集的元素包括字母、数字、空格和一些标点符号。这些字符首先组成英文单词，然后根据英文文法规则再组成英文句子。完全类似，计算机程序也是从程序语言的字符集构造而成。这些字符先组成单词，然后再根据程序语言的文法规则组成程序。与英文不同的是，程序语言的文法规则相当严格。人们对此作了深入的研究，已经得到许多实用的结果。

用程序语言书写的程序一般由一些基本符号组成。下面是一些常见的基本符号：

字母：a, b, c,⋯ , x, y, z

数字：0, 1,⋯ , 9

其他符号：+，–，*，⋯, =, ：

在这些符号的基础上，组成如 if, then, else, for 等关键字，程序的标识符和常量，并进一步组成用户程序。

字母表是一个由字符所组成的有限集合。每一个程序语言都有自己的字母表，在程序语言定义时确定。程序语言的字母表来自键盘字符集，例如 FORTRAN 语言的字母表由 26 个英文字母、10 个数字和 12 个特殊符号如+ – * / ( ) = ' $ ： 等组成。FORTRAN 程序可以看成是一串由这些字符所组成的字符串。然而，一般说来，程序语言的字母表中有些元素由若干个字符组成。例如，许多程序语言中的赋值号“: =”由字符“：”和“=” 组成。字母表一般用 V 和 Σ 来表示。

定义 2.1　字母表 V 上的符号串的递归定义如下：

① 空串（一个不含任何元素的符号串）是字母表 V 上的符号串。空串用希腊字母 ε 表示；

② 字母表 V 中任何一个元素 a, 是字母表 V 上的符号串；

③ 若 x, y 是字母表 V 上的符号串, 则它们的并置—— 将符号串 y 的符号依次写在符号串 x 后面所得的符号串，也是字母表 V 上的符号串，记以 xy。

符号串 x 中的元素的个数称为符号串 x 的长度，记以 |x| 。设 z = xy 是字