21世纪应用型本科系列教材

编译原理

鱼滨 侯红 龚晓庆

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

应用型本科系列教材

编　译　原　理

西安交通大学出版社

内容简介

本书是按照国家教育部制定的计算机专业编译原理课程教学大纲，兼顾目前学时压缩的要求编写而成。

本书系统地介绍程序设计语言编译程序构造的一般原理和基本实现方法。主要讲授词法分析、正规式与有限自动机、语法分析、语义分析与中间代码生成、运行时存储器的组织、代码优化和生成等内容。通过本书的学习，使读者对编译的基本概念、原理和方法有完整而清晰的理解，并能正确地运用。

本书作为高等学校计算机类专业的本科生教材，也可作为教师、学生及软件相关技术人员的参考书籍。

图书在版编目(CIP)数据

编译原理/鱼滨，侯红，龚晓庆编. —西安：西安交通大学出版社，2007.8
(21世纪应用型本科系列教材)
ISBN 978-7-5605-2499-3

Ⅰ. 编… Ⅱ. ①鱼… ②侯… ③龚… 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(2007)第101851

书　　名	编译原理
编　　者	鱼　滨　侯　红　龚晓庆
出版发行	西安交通大学出版社
地　　址	西安市兴庆南路10号　(邮编：710049)
电　　话	(029)82668357　82667874(发行部) (029)82668315　82669096(总编办)
印　　刷	陕西元盛印务有限公司
字　　数	242千字
开　　本	787 mm×1 092 mm　1/16
印　　张	10.375
版　　次	2007年8月第1版　2007年8月第1次印刷
印　　数	0 001～3 000
书　　号	ISBN 978-7-5605-2499-3/TP·497
定　　价	13.80元

21 世纪应用型本科系列教材

计算机类教材编委会

前　言

编译原理是高校计算机科学与技术专业的一门专业必修课，课程从理论和实践两个方面来介绍编译程序涉及的计算机技术基础理论及程序设计技术。通过编译原理课程的学习，使学生掌握编译程序本身的实现技术，同时提高对程序设计语言的理解，培养学生的抽象思维能力和软件开发能力。

本课程涉及程序设计语言的相关基础理论，形式化成分较多，学生感觉不容易学，目前国内大多数编译教材内容偏多而深，而授课时数又压缩，从而造成教材内容不能完全使用，学生负担较重的现象。本书旨在以简练的语言、较少的篇幅讲述编译的原理和技术，并涵盖本课程所要求学生掌握的基本内容，本教材可以供二、三本的学生使用。

本书系统地介绍了编译程序构造的一般原理和基本实现方法，内容包括词法分析、语法分析、中间代码产生、优化和目标代码产生等五大部分。作为原理性教科书，着重介绍编译的基本理论和方法，对于实现的具体细节未予详述。本书注重最经典、最广泛应用的基本编译技术。

全书分为六章。第一章绪论，介绍程序设计语言和语言翻译的基本概念，内容包括：语言翻译与编译程序，编译器与解释器，编译程序的基本原理与结构，编译器的构造等。第二章词法分析，本章是整个编译内容的基础，除了介绍词法分析器的构造外，主要讲授正规式和有限状态自动机的基本知识，内容包括：词法分析器的工作方式，词法模式的形式化描述，有限自动机，正规式到有限状态自动机、词法分析器的自动生成等。第三章语法分析，讲授文法和基本语法分析器的构造，内容包括：上下文无关文法，自上而下的语法分析，自下而上的语法分析，语法分析器自动生成工具 YACC 简介等。第四章语法制导翻译生成中间代码，讲授语法制导翻译生成中间代码的方法，内容包括：语法制导翻译，中间代码，说明语句的翻译，赋值语句的翻译，布尔表达式的翻译，控制语句的翻译，过程调用及类型检查等。第五章符号表与运行时环境，内容包括：符号表，目标程序运行时活动，运行时存储器的划分，存储分配策略简介等。第六章代码优化与代码生成，讲授代码生成所需考虑的问题，内容包括：代码优化，代码生成的相关问题，简单的机器模型，简单的代码生成器，所有算法用 C 语言表达等。

作者力图反映编译的基本理论和程序技术，并且力图用简洁的语言讲述抽象的原理。由于作者水平所限，难免有错误和不当之处，敬请批评指正。

编者

目　录

第1章 绪 论

1.1 语言翻译与编译程序

使用过现代计算机的人都知道，多数用户都应用高级语言来实现他们所需要的计算。而计算机上执行一个高级语言程序一般都分为两个步骤，首先，使用编译程序把高级语言翻译成机器语言程序；其次，运行机器语言程序求得计算结果。通常所说的编译程序是指一种能够把某一种语言程序（源语言程序）转换成另一种语言程序（目标语言程序）的程序，后者与前者在逻辑上是等价的。这里特指把高级语言程序转变为目标语言（如汇编语言或机器语言）程序的程序。编译程序根据不同的用途和侧重点可进一步分为诊断编译程序（Diagnostic Compiler）和优化编译程序（Optimizing Compiler）。诊断编译程序是专门用于帮助程序开发和调试的编译程序；优化编译程序是着重于提高目标代码效率的编译程序。现在很多编译程序同时提供了调试和优化等多种功能，用户可以通过“开关”进行选择。

我们将运行编译程序的计算机称为宿主机，运行编译程序产生目标代码的计算机称为目标机。如果一个编译程序产生不同于其宿主机的机器代码，则称它为交叉编译程序（Cross Compiler）。如果不需要重新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）。

现在计算机系统一般都含有不止一个的高级语言编译程序，对有些高级语言甚至配置了几个不同性能的编译程序，供用户按不同需要进行选择。高级语言编译程序是计算机系统软件最重要的组成部分之一，也是用户最直接关心的工具之一。

1.2 编译器与解释器

编译器是将一种语言翻译为另一种语言的计算机程序。编译器将源语言（source language）编写的程序作为输入，而产生用目标语言（target language）编写的等价程序。通常源程序为高级语言（high-level language ），如 JAVA、C++、FORTEAN 等，而目标语言则是目标机器的目标代码（object code），有时也称作机器代码（machine code）。也就是写在计算机机器指令中的用于运行的代码。这一过程表示为：

源程序→ 编译器→ 目标程序

编译器是一种相当复杂的程序,其代码的长度可从 10 000 行到 1 000 000 行不等。编写甚至读懂这样的一个程序并非易事,大多数计算机科学家和专业人员从来没有编写过一个完整的编译器。但是,几乎所有形式的计算均要用到编译器,而且任何一个与计算机打交道的专业人员都应掌握编译器的基本结构和操作。除此之外,计算机应用程序中经常遇到的一个任务就是开发命令解释程序和界面程序,这些程序比编译器要小,但使用的技术却是相同的。因此,掌握这一技术具有非常重要的实际意义。

高级语言程序除了像上面所说的先编译后执行外,有时候也可以用"解释"来执行。解释器是这样一种程序,它以高级语言写的源程序作为输入,但不产生目标程序,而是一边解释一边执行源程序本身。

从原理上讲,任何程序设计语言都可被解释或被编译,但是根据所使用的语言和翻译情况,很可能会选用解释程序而不用编译器。例如,我们经常解释 BASIC 语言而不是去编译它。但是,当执行的速度是最重要的因素时就要使用编译器,这是因为被编译的目标代码比被解释的源代码要快得多,有时要快 1 0 倍或更多。

1.3 编译程序的工作原理与基本结构

1.3.1 高级语言的主要成分

我们要构造编译程序,理解和定义高级程序语言是必不可少的。因为对于高级程序语言的定义和理解是我们进行编译的基础。人们使用高级程序语言来实现自己所需要的计算。而这些程序语言就构成了我们编译器的输入——源语言(source language)。

高级程序语言是用来描述算法和计算机实现这两个目的的。目前,世界上的高级语言有上千种,且较大范围内得到使用的语言也有几十种甚至上百种。从应用角度看,它们各有不同的侧重点。如 FORTRAN 宜于数值计算,COBOL 宜于事务处理,PROLOG 适用于人工智能,Ada 适合于大型嵌入式实时处理,SNOBOL 则更利于符号处理。从语言范畴来分,高级程序语言可分为强制式语言、作用式语言、基于规则的语言和面向对象语言等。

一个程序语言是一个记号系统。如同自然语言一样,程序主要由语法和语义两个方面定义。有时候语言定义也包含语用信息,语用主要是有关程序设计技术和语言成分的使用方法,它使语言的基本概念与语言的外界(如数学或计算机的对象和操作)联系起来。下面我们就针对语法和语义做进一步的解释。

1. 语法

任何语言程序都可以看成是一定字符集(称为字母表)上的一字符串(有限序列)。并且一个程序语言只使用一个有限字符集作为字母表。所谓一个语言的语法是指这样的一组规则,用它可以形成和产生一个合式的程序。这些规则的一部分称为词法规则,另一部分称为语法规则(或产生规则)。

语言的语法规则规定了如何从单词符号形成更大的结构(即语法单位),换言之,语法规则是语法单位的形成规则。一般程序语言的语法单位有:表达式、语句、分程序、函数、过程和程序等等。要真正地描述语法规则一般是很不容易的。但就现今的多数程序语言来说,上下文无关文法仍是一种可取的有效工具。有限自动机和上下文无关文法是我们讨论词法分析和语法分析的主要理论基础。

词法规则是指单词符号的形成规则。单词符号是语言中具有独立意义的最基本结构。在现今多数程序语言中,单词符号一般包括:各类型的常数、标志符、基本字、算符和界符等。由于单词符号本身很简单,因此形成规则也不复杂。正规式和有限自动机理论就是描述词法结构和进行词法分析的有效工具。

语法单位比单词符号具有更丰富的意义。例如单词符号串"2+3*3.14"代表一个算术式,具有通常的算术意义。如何定义各种语法单位的含义属于语言的语义问题。

语言的词法规则和语法规则定义了程序的形式结构,是判断输入字符串是否构成一个形式上正确(即合式)的程序的依据。

一般而言,程序语言的词法、语法规则并不限定程序的书写格式。但是,某些程序语言要求程序的书写服从一定的格式,而这样的要求就增加了词法分析的复杂性。现在多数语言倾向于使用自由格式书写法,容许程序员随自己的意愿编写程序。这既便于阅读,又可以回避因书写格式不正确而造成的错误。

有些语言规定,空白字符除了在文字常数中出现之外,在其他任何地方出现都是没有意义的。在这种情况下空白字符可用于编排程序格式,但增加了词法分析的麻烦。在某些语言中,空白字符用作间隔符,它的出现决定了单词符号的划分。

2. 语义

对于一个语言来说,不仅要给出它的词法、语法规则,而且要定义它的单词符号和语法单位的意义。这就是语义问题。离开语义,语言只不过是一堆符号的集合。在许多语言中,有着形式上完全相同的语法单位,但含义却不尽相同。同样的符号串代表的"算术表达式",但是含义却不同了。对于编译来说,只有了解程序的语义,我们才知道应把它翻译成什么样的目标指令代码。

所谓一个语言的语义是指这样的一组规则,使用它可以定义一个程序的意义。这些规则称为语义规则。阐明语义要比阐明语法难得多。现在还没有一种公认的形式系统,借助于它可以自动地构造出实用的编译程序。目前大多数编译程序普遍采用的一种方法,即基于属性文法的语法制导翻译方法,虽然这还不是一种形式系统,但它还是比较接近形式化的。

一个程序的基本功能是描述数据和数据的运算。所谓一个程序,从本质上来说是描述一定数据的处理过程。在现今的程序语言中,一个程序的层次结构大体上如图 1-1 所示。

自上而下看上述层次结构:顶端是程序本身,它是一个完整的执行单位。一个程序通常是由若干个子程序或分程序组成的,它们常常含有自己的数据。子程序或分程序是由语句组成的。而组成语句的成分则是各种类型的表达式。表达式是表述数据运算的基本结构,它通常含有数据引用、算符和函数调用。

自下而上看上述层次结构:我们希望通过对下层成分的理解掌握上层成分,从而掌握整个程序。

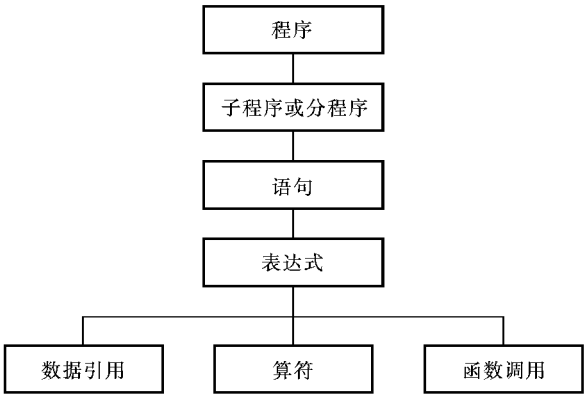

图 1-1 一般程序的层次结构

1.3.2 编译的基本过程

编译程序的工作,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。但就其过程而言,它与人们进行自然语言之间的翻译有许多相近之处,就像我们把一种文字翻译成另外一种文字一样。如果我们将一段英文翻译成中文时,经常需要以下五个步骤,而编译程序的工作也可以分为五个阶段。如图 1-2 所示。

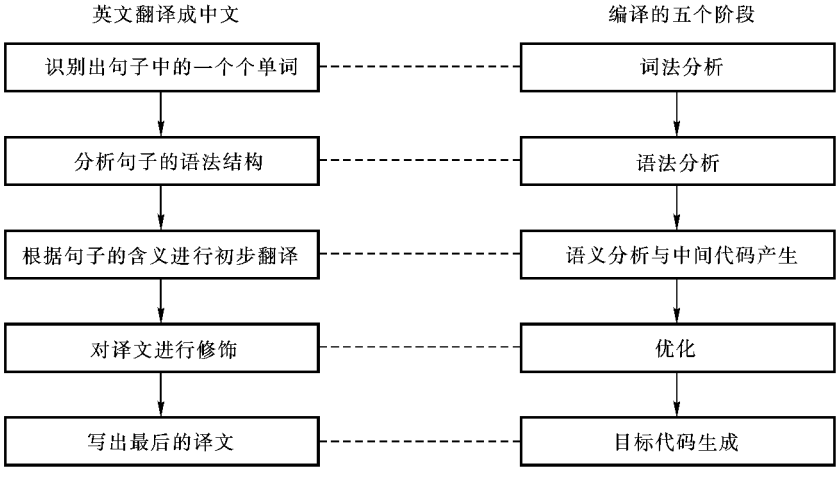

图 1-2 编译程序的主要工作过程

1.3.3 编译各阶段的工作

编译程序有五个阶段:词法分析、语法分析、语义分析与中间代码产生、优化、目标代码生成。

第一阶段:词法分析。

词法分析的任务是:输入源程序,对构成源程序的字符串进行扫描和分解,识别出一个个

的单词(亦称单词符号或简称符号),如基本字(begin、end、if、for、while 等)、标识符、常数、算符和界符(标点符号、左右括号等等)。例如在下面的代码行(它可以是 C 程序的一部分)中:

```
a [index] = 4 + 2
a ----------标识符
[ ----------左括号
index-------标识符
] ----------右括号
= ----------赋值号
4 ----------常数
+ ----------加号
2 ----------常数
```

这些单词是组成上述 C 语句的基本符号。单词符号是源程序的基本组成成分,是人们理解和编写程序的基本要素。识别和理解这些要素无疑也是翻译基础。与把英语翻译成中文的情形一样,如果你对英语单词不理解,那就谈不上进行正确的翻译。在词法分析阶段的工作中所依循的是语言的词法规则(或称构词规则)。描述词法规则的有效工具是正规式和有限自动机。

第二阶段:语法分析。

语法分析的任务是:在词法分析的基础上,根据语言的语法规则,把单词符号串分解成各类语法单位(语法范畴),如"短语"、"字句"、"句子"("语句")、"程序段"和"程序"等。通过语法分析,确定整个输入串是否构成语法上正确的"程序"。语法分析所依循的是语法规则。语法规则通常用上下文无关文法描述。词法分析是一种线型分析,而语法分析是一种层次结构分析。例如,符号串

$$c = a + 3 * b$$

代表一个赋值语句,而其中的"a + 3 * b"代表一个算术表达式。因而语法分析的任务是识别"a + 3 * b"为算术表达式,同时还识别上述整个符号串属于赋值语句的范畴。

第三阶段:语义分析与中间代码产生。

这一阶段的任务是:对语法分析所识别出的各类语法范畴,分析其含义,并进行初步翻译(产生中间代码)。这一阶段通常包括两个方面的工作:首先,对于每种语法范畴进行静态语义检查;如果语义正确了,则进行下一步的工作,即进行中间代码的翻译。这一阶段所依循的是语言的语义规则。通常使用属性文法描述语义规则。

所谓"中间代码"是一种含义明确、便于处理的记号系统,它通常独立于具体的硬件。这种记号系统或者与现代计算机的指令形式有某种程度的接近,或者能够比较容易地把它变换成现代计算机的机器指令。如采用四元式作为中间代码,"中间代码产生"的任务就是按语言的语义规则把各类语法范畴译成四元式序列。如

算符	左操作数	右操作数	结果

它的意义是:对"左、右操作数"进行"算符"规定的运算,把运算所得的值作为"结果"保留下来。例如

$$z=(a+b) * c/d$$

的四元式可为

(+ a b T1)
(* T1 c T2)
(/ T2 d z)

其中 T1、T2 都是编译期间引进的临时工作变量。

一般而言,中间代码是一种独立于具体硬件的记号系统。常用的中间代码,除了四元式外,还有三元式、逆波兰记号和树形表示等等。

第四阶段:优化。

优化的任务在于对前段产生的中间代码进行加工变换,以求在最后阶段能产生出更为高效(省时间和空间)的目标代码。它包括以下几个方面:公共子表达式的提取、循环优化、删除无用代码等等。有时,为了便于"并行运算",还可以对代码进行并行化处理。优化所依循的原则是程序的等价变换规则。

第五阶段,目标代码生成。

这一阶段的任务是:把中间代码(或经优化处理之后的代码)变换成特定机器上的低级语言代码。这个阶段实现了最后的翻译,它的工作依赖于硬件系统结构和机器指令含义。这一阶段工作非常复杂,涉及到硬件系统功能部件的运用,机器指令的选择,各种数据类型变量的存储空间分配,以及寄存器和后援寄存器的调度等等。如何产生足以充分发挥硬件效率的目标代码是一件非常不容易的事情。

目标代码可以是绝对指令代码、可重定位的指令代码或汇编指令代码。如目标代码是绝对指令代码。则这种目标代码可立即执行。如果目标代码是汇编指令代码,则需要汇编器汇编之后才能运行。必须指出,现在多数使用编译程序所产生的目标代码都是一种可重定位的指令代码。这种目标代码在运行前必须借助于连接装配程序把各个目标模块连接在一起,确定程序变量在主存中的位置,装入内存中指定的真实地址,使之成为一个可以运行的绝对指令代码程序。

上述编译程序的五个阶段是一种典型的逻辑模型。事实上,并非所有编译程序都分成这五阶段。有些编译程序对优化没有要求,优化阶段就可以省去;在某些情况下,为了加快编译速度,中间代码产生阶段也可以去掉;有些最简单的编译程序是在语法分析的同时产生目标代码。但是,大多数使用编译程序的工作过程均由上述五个阶段构成。

1.3.4 编译程序的基本结构

编译过程被分为了五个阶段,编译程序的结构也可以按照这五个阶段的任务分模块进行设计。如图 1-3 所示。

词法分析器:输入源程序,进行词法分析,输出单词符号。

语法分析器:简称分析器,对单词符号串进行语法分析,识别出各类语法单位,最终判断输入串是否构成语法上正确的"程序"。

语义分析与中间代码产生器:按照语意规则对语法分析器归约出(或推导出)的语法单位进行语义分析并把它们翻译成一定形式的中间代码。

优化器:对中间代码进行优化处理。

目标代码生成器:把中间代码翻译成目标程序。

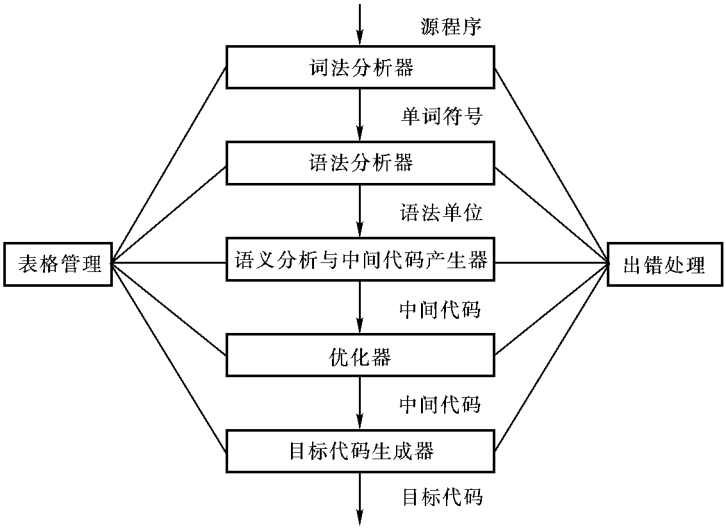

图 1-3 编译程序的总框架

在上述框架中还有两个重要的概念就是表格管理和出错管理,下面就来研究一下为什么需要表格管理和出错管理,以及它们的作用。

表格与表格管理:编译程序在工作中需要保存一系列的表格,以登记源程序中各类信息的编译中各阶段的进展状况。合理地设计和使用表格是编译程序构造的一个重要问题。在编译程序使用的表格中,最重要的是符号表。它用来登记源程序中出现的每个名字以及名字的各种属性。例如,一个名字是常量名、变量名,还是过程名等等,以及它的类型、内存地址等信息。

编译的各个阶段都涉及到了对这些表格的修改、查找或更新等工作。所以对于这些表格的管理也是编译程序的主要工作之一。

出错处理:一个编译程序不但要能对书写正确的程序进行翻译,而且应该能对出现在源程序中的错误进行处理。如果源程序中发现错误,编译器就应该将信息发送给用户。这部分的工作是由专门的一组程序(出错处理程序)完成。如果这部分程序不但能够发现错误,而且还能自动校正错误那就更好了。但是,自动校正的代价是非常高的。

源程序中的错误通常分为语法错误和语义错误两大类。语法错误是指源程序中不符合语法或词法规则的错误,它们可在词法分析或语法分析时检测出来。语义错误是指源程序中不符合规则的错误。语义错误通常包括:说明错误、作用域错误、类型不一致等等。

1.3.5 编译的前端和后端

我们把编译程序划分为编译前端和编译后端。前端主要由与源语言有关但与目标机无关的那些部分组成,通常包括词法分析、语法分析、语义分析与中间代码产生,有的代码优化工作也可包括在前端。后端包括编译程序中与目标机有关的那些部分,如与目标机有关的代码优化和目标代码生成等。

可以取编译程序的前端,改写其后端以生成不同目标机的相同语言的编译程序。如果后端的设计是经过精心考虑的,那么后端的改写将用不了太大工作量,这样就可实现编译程序的目标机改变。也可以设想将机中的各种语言都采用相同的中间语言,虽然这些语言的编译程序有不

同的前端,但它们都可以配上相同的后端,这样就可以高效地为同一台机器生成不同语言的编译程序。然而,由于不同语言存在某些微妙的区别,因此在这方面所取得的成果非常有限。

1.3.6 编译的遍数

前面所讲的编译过程的五个阶段仅仅是逻辑功能上的一种划分。具体实现时,受不同源语言、设计要求、使用对象和计算机条件(如主存容量)等限制,往往将编译程序组织为若干遍(pass)。所谓遍就是对源程序或源程序的中间结果从头到尾扫描一次,并作有关的加工处理,生成新的中间结果或目标程序。通常,每遍的工作由从外存上获得的前一遍的中间结果开始(对于第一遍而言,从外存上获得源程序),完成它所含的有关工作之后,再把结果记录于外存。既可以将几个不同阶段合为一遍,也可以把一个阶段的工作分为若干遍。为了便于处理,语法分析和语义分析与中间代码产生又常常合为一遍。在优化要求很高时,往往还可以把优化阶段分为若干遍来实现。当一遍中包含若干阶段时,各阶段的工作是穿插进行的。

一个编译程序究竟应分成几遍,如何划分,是与源语言、设计要求、硬件设备等诸因素有关的,因此难于统一划定,遍数多一些是有好处的,即整个编译程序的逻辑结构会更清晰。但是遍数多肯定要增加输入/输出所消耗的时间。因此,一般的编译程序遍数少一些比较好。应当注意,并不是每种语言都可以用单遍编译程序实现。

1.4 编译器的编写

人们早期使用汇编语言来编写编译器,但是由于编译器本身是一个十分复杂的系统,用汇编语言编写编译器的效率很低,往往给实现带来很大的困难,因此,除了特别需求,人们已经不再使用汇编语言编写编译器了。现在常用高级程序设计语言编写编译器,它的效率比汇编语言要高得多,不过用单纯的程序设计方法来对付编译器这样的庞大工程显然是不够的,因此需要一些专门的编译器编写工具来支持编译器某些部分的自动生成。

比较成熟和通用的工具有词法分析生成器和语法分析生成器,如被广泛应用的 YACC 和 LEX。另外还有一些工具如语法制导翻译工具(用于语义分析)、自动代码生成器(用于中间代码和目标代码生成)、数据流工具(用于优化)等。这些工具有一些共同的特点,就是仅需要对语言相应部分的特征进行描述,而把生成算法的过程隐蔽起来,同时所生成的部分可以很统一地并入到编译器的其他部分中。因此,这些工具往往与某程序设计语言联系在一起,如与 YACC 联系的程序设计语言就是 C。

1.5 本章小结

本章介绍了有关编译原理的一些基本概念和知识,例如:编译程序、高级程序语言、编译

器和解释器、编译器的前端和后端、编译器的遍数等。我们需要掌握的重点以及难点包括以下一些内容：

首先，我们需要了解什么是程序设计语言以及高级语言的翻译。

程序设计语言是用来编写程序的工具，分为低级和高级程序设计语言。低级语言包括机器语言及其面向机器的程序设计语言。而高级语言有上千种之多，常用的有 JAVA、C++、C、PASCAL 等。高级语言比起低级语言在算法描述能力、编程和调试效率上都有较高的优越性。

如果在计算机上要使用高级语言，需要该语言能够被计算机所理解。我们使用的方法就是对程序进行翻译或进行解释。编译程序就是将高级语言编写的程序（源程序）翻译为汇编语言或者机器语言形式（目标语言）的一种翻译程序。

其次，我们需要知道一个编译程序需要做的工作和它的逻辑结构。

编译程序的主要工作有五个：词法分析、语法分析、语义分析，中间代码产生、优化及目标代码生成。

一般编译程序的逻辑结构包含以下几个部分：

(1)词法分析程序；

(2)语法分析程序；

(3)语义分析程序；

(4)中间代码生成程序以及优化程序；

(5)目标代码生成程序；

(6)错误检查和处理程序；

(7)各种信息表的管理程序。

最后，编译程序的组织方式。

也就是说我们可采用一遍或者多遍扫描源程序或其中间代码的处理方法来实现编译程序。如何划分遍数，与源语言、设计要求、硬件设备等诸因素有关，因此遍数的划定是由具体情况来决定的，难以统一划定。

习题

一、选择题

1. 一个编译程序中，不仅包含词法分析，语法分析，中间代码生成，代码优化，目标代码生成五个部分，还应包括_____。其中，_____和代码优化部分不是每个编译程序都必需的。词法分析器用于识别_____，语法分析器则可以发现源程序中的_____。

(1) a. 模拟执行器　b. 解释器　c. 表格处理和出错处理　d. 符号执行器

(2) a. 语法分析　b. 中间代码生成　c. 词法分析　d. 目标代码生成

(3) a. 字符串　b. 语句　c. 单词　d. 标识符

(4) a. 语义错误　b. 语法和语义错误　c. 错误并校正　d. 语法错误