新一代信息科学与技术丛书

编译原理

包含代数方法的新编译方法

苏运霖　颜松远

高等教育出版社
HIGHER EDUCATION PRESS

新一代信息科学与技术丛书

编译原理

包含代数方法的新编译方法

BIANYI YUANLI

BAOHAN DAISHU FANGFA DE XIN BIANYI FANGFA

苏运霖　颜松远

高等教育出版社·北京
HIGHER EDUCATION PRESS BEIJING

图书在版编目(CIP)数据

编译原理：包含代数方法的新编译方法/苏运霖，颜松远著. —北京：高等教育出版社，2011.8

ISBN 978-7-04-033047-2

Ⅰ.①编… Ⅱ.①苏… ②颜… Ⅲ.①编译程序-程序设计 Ⅳ.①TP314

中国版本图书馆 CIP 数据核字(2011)第 129268 号

策划编辑 陈红英　**责任编辑** 陈红英　**封面设计** 李卫青　**版式设计** 余　杨

责任校对 金　辉　**责任印制** 毛斯璐

出版发行	高等教育出版社	**咨询电话**	400-810-0598
社　　址	北京市西城区德外大街 4 号	**网　　址**	http://www.hep.edu.cn
邮政编码	100120		http://www.hep.com.cn
印　　刷	北京市卫顺印刷厂	**网上订购**	http://www.landraco.com
开　　本	787×1092　1/16		http://www.landraco.com.cn
印　　张	24	**版　　次**	2011 年 8 月第 1 版
字　　数	500 000	**印　　次**	2011 年 8 月第 1 次印刷
购书热线	010-58581118	**定　　价**	49.00 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

物 料 号　33047-00

前　言

编译程序是最重要的系统软件之一。任何一个计算机用户，当他编写计算机程序时必然要用某种程序设计语言，而不是使用计算机的指令系统。这意味着，在他所用的这台计算机上，必须安装这种程序语言的编译程序，否则就不能运行所编的程序。

编译程序和程序设计语言不同，程序设计语言一旦被制定，就基本保持不变，而实现编译的技术却可能发生变化。而且人们总是在探索新的更有效的技术来提高编译程序的质量和效率。

类似“编译程序原理”的课程已成为高等学校计算机学科最重要的课程之一。编译技术的发展沿着两个方向，其一是对于现存程序设计语言编译技术的改进；其二是对于新的程序设计语言编译技术的研究和开发。这类新语言包括面向对象语言、分布式语言、并行程序设计语言，等等。本书介绍最新的知识，探索用于新的语言和计算的编译技术；把程序设计语言的编译和在人脑中自然语言的翻译相联系，使读者更易于理解编译原理；同时还介绍了编译的代数方法，它属于形式化技术。

本书共16章。第1章概论概述了编译的过程，并把程序设计语言的编译同人脑对自然语言的理解和处理关联起来。第2章介绍文法和语言。语言是基于文法产生的，是编译过程的基础。第3章介绍有限自动机和正则语言。第4章与第3章一同致力于词法分析，即编译的分析阶段的第一个任务。第3章可看作是词法分析的理论准备，而第4章是词法分析的具体实践。第5章、第6章和第7章共同致力于语法分析。第5章介绍下推自动机，它对应于上下文无关文法。第6章介绍上下文无关文法和它所生成的上下文无关语言。第7章探讨分析阶段的第二个任务——语法分析。接着是语义分析。在分析阶段完成之后，综合阶段才能开始，而综合阶段的主要任务是生成目标代码。第8章介绍属性文法及分析。第9章介绍一种新的编译方法——编译的形式化方法。第10章介绍中间代码的生成。第11章介绍用于编译程序中的调试和优化技术。第12章介绍与程序编译有关的存储管理。第13章是编译的终点站，即目标代码的生成。本章介绍了由唐·欧·克努特(D. E. Knuth)在他的著作《计算机程序设计艺术》中提出的虚拟机器MMIX。该虚拟机组合了目前市场上最流行的14种机器的特征，指令系统丰富，因此使目标程序更灵活。第14章和第15章介绍了面向对象语言和并行程序设计语言的编译技术。第16章讨论网格计算的问题。目前，网格计算虽已引起关注，但还没有专门用于网格计算的语言出现，

所以只能针对它的特点,指出如果有了这种语言对它的编译需要处理的问题。

最后,我们感谢高等教育出版社编辑陈红英,没有她的鼓励、帮助和耐心,也许我们都不能完成此书的写作。我们也要感谢在本书中所引用成果的作者们。他们的知识组成了本书的大部分内容。我们也感谢我们的家庭和学生对我们工作的支持。

本书疏漏或错误难免,敬请读者不吝指正。在此对所有指正表示诚挚的谢意。

苏运霖 颜松远

2011 年 5 月

目　录

第1章 概 论

语言使你懂得，在诸多事物当中，章鱼是如何做爱的，樱桃如何去掉菌株；为什么塔德(Tad)伤心至极；在世界职业棒球锦标赛中，红袜子队是否会获胜，而无需大换投球手；如何在地下室制作原子弹；沙皇卡特琳娜又是如何去世的。

——斯蒂夫·平克尔

1.1 语言和人类

如果你读了上面的文字，你一定会被那些描述所感动——这就是通过语言来影响别人的途径。当今世界上有如此众多的语言，以至于几乎没有人能准确地说出究竟有多少种。因此在不同的语言使用者之间需要架起一座桥梁，使得他们之间可以彼此交流，这座桥梁就是翻译。本书的主旨就是讨论在形式语言和机器之间的翻译，即编译。什么是编译程序？简单地说，编译程序是将以程序设计语言写成的程序翻译成以某种机器语言写成的在该机器上能运行的程序。为了说明问题，我们需要多费些笔墨。

语言是人类交流和沟通的主要工具。通过语言，人们建立起联系，表达各自的意图和感情，描述所见景物和对于事物的理解[1]。语言是一种智能，或者说是智能的产物。但是在人类进化之初，还没有语言，在经历了漫长的时间后才产生了口头语言。这是人类在语言方面的第一个突破，也是人类文明方面的一个突破。从口头语言到书面语言，人类经历了更漫长的时间。书面语言的出现是人类在语言方面又一个重大的突破。人们的思维和解决问题的过程可以概念化为设计语言的过程。许多解决问题的形式是在大脑内部，即不存在外部刺激下进行的。例如，把难题抽象化为文字符号就提供了思考问题答案的一个途径。不难想象，如果没有语言，思维的过程就不可能完整、继续和深化。没有语言，人们无法表达自己的见解；当他/她要回忆往事时却不能谈及涉及许多对象或复杂景物的过程，因为他/她将无从描述它们。书面语言比口头语言更强有力。书面语言不仅能把现代人联系起来，还能把现代人和古代人联系起来，使得现代人知道在古代发生的事情。特别是通过现代化的通信工具，例如计算机网络、电视以及电话，人们彼此的联系可以更加快捷、方便和安全。这也意味着，书面语言改变了人们时空范围。过去说“秀才不出门，全知

天下事”,在当时多少有些夸大其辞,但今天确是一语中的。

由于人类生活地域的差别,每个民族或种族形成了自己的语言。历史上曾经存在数千种语言。随着时间的推移,一些很少人使用的口头语言消失了,但仍然存在一些只有口头语言而无书面语言的语言。如果两个人要交谈,就需要架起桥梁,这座桥梁就是翻译。翻译的任务就是把 A 所使用的甲语言翻译成 B 所使用的乙语言,以及把 B 所使用的乙语言翻译成 A 所使用的甲语言。

今天,随着科学技术的迅猛发展以及经济全球化的趋势,语言翻译,包括口头和书面翻译,都已成为热门的职业。例如,口头翻译涉及三个人,即为了某个目的,A 和 B 两个人要交谈,但他们彼此语言不同,需要 C 来帮助。假设 A 使用的是甲语言,而 B 使用的是乙语言,很显然,为使 A 和 B 能彼此了解,C 要把 A 讲的甲语言翻译成乙语言;反过来,为使 B 讲的话能为 A 所理解,C 又要把乙语言翻译成甲语言。所以,在这种情况下 C 必须是双语使用者,即他既通晓甲语言,也通晓乙语言。这种情况如图 1.1 所示。

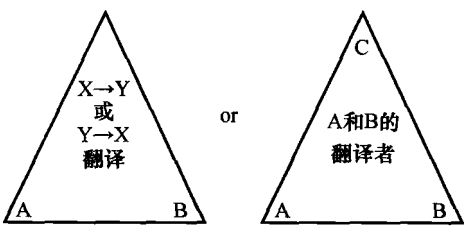

图 1.1 口头翻译过程

有时,C 的作用要由 C 和 D 两个人完成,比如 C 只负责把甲语言译成乙语言,而 D 则负责把乙语言翻译成甲语言。在正式的场合,这种情况相当普遍。一般都是翻译者把外国语翻译成本国语言。不论如何做,目的都是使交谈双方彼此了解。

1.2 语言和计算机

众所周知,计算机是人类最伟大的发明之一。它体现了人类科学技术的最新发展。计算机依靠它自己的运行来解决人类提出的问题,而其运行则依靠事先编好的一组指令组成的程序来完成的。用来编写程序的指令系统可看作是机器的语言,计算机执行指令就如同按由序列组成的语言来说话一样,而这组序列是由 0 和 1 组成的。

为使计算机为人工作和运行,人们必须编出为求解既定问题的程序,为此,就需要掌握指令系统。然而,对于计算机的普通用户而言,要求他们掌握机器指令系统这种繁琐、乏味而又不直观的语言无疑是非常困难的,这就好像谁想看电视就要掌握电视的原理并自己安装电视机一样。

在计算机问世初期,没有可以用来对计算机进行操作的高级语言,计算机的指令系统

是用来对计算机进行操作的唯一语言，这个时期被称为手编程序的时期。机器指令通常是由操作码和操作数地址组成的。操作码是对操作数执行的操作的表示，而操作数就由其地址给出。指令中有时包含控制码，用于表示处理特定的情况。因此，要使用指令系统编写程序，无疑变成了人们使用计算机的绊脚石[2]。为摆脱机器语言的限制，人们开始探索这个问题的解决方法。最初，人们使用易于记忆的符号来代替指令的操作码，例如用 ADD 表示加法操作，用 SUB 表示减法，MUL 表示乘法，DIV 表示除法等，或者更简单地用 +、-、×和/(或÷)来表示算术运算，然后，用符号代替实际的二进制地址等。进行变换后的语言就不是计算机语言或原来的计算机指令系统了，尽管它基本上还是对应于指令系统，而且它们完全等价。这是人们摆脱机器语言的第一步，尽管还只是一小步，但它是关键的一步，这表明人们可以不受机器指令系统的限制，可以使用更方便的语言来为计算机编程序。这种语言称为汇编语言。在这里，图 1.1 给出的模式仍然适用。图 1.2 左下部表示以汇编语言写成的程序，称为源程序。三角形的右边是由机器指令系统写成的，与源程序完全等价的程序，称为目标程序，它是由三角形顶部的汇编程序从源程序翻译而成的。这里，汇编程序起了编译程序的作用，负责把源程序翻译成可在机器上执行的由机器语言写成的目标程序，因此这个汇编程序也必须是可执行的，而且通过它产生目标程序。汇编程序是编译程序的早期版本。由于源语言是汇编语言，是机器指令系统的简单改编(例如，操作码只是原来的操作码的记忆性代码)，因此也称为低级语言。这里，“低级”指的是面向机器(低级)，而不是面向人类(高级)的程序设计语言。

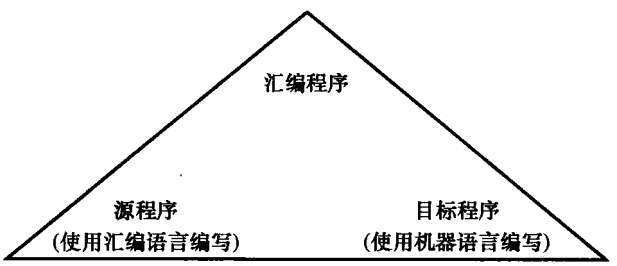

图 1.2 汇编语言翻译

在汇编语言取得成功后，人们开始设计面向人类的高级程序设计语言。这种语言的特点是摆脱了机器指令系统的限制，采用自然语言(一般是英语)的子集，并为它建立文法来描述用来编写程序的语句或其他成分。这些语言称为面向过程的语言，或者简单地称为过程语言或命令式语言。早期的语言包括 FORTRAN(Formula Translation，公式翻译，1954 年开始设计)[3]，ALGOL (ALGOrithmic Language，算法语言，1958 年开始设计)，COBOL(Common Business Oriented Language，面向普通商业的语言，1959 年开始设计，其成功来自于美国国防部的强力支持)[4]。就程序设计的发展而言，20 世纪 60 年代是程序设计语言分枝繁衍的年代，这个时期出现的各种语言数以千计，但经过大浪淘沙般的考验，

仅有13种被认为在概念和应用上有重要性。其中包括:APL(A Programming Language,一种程序设计语言),是由IBM公司的肯尼思·艾弗森(Kenneth Iverson)设计的会话式语言(Iverson,1962)[5];PL/1(Programming Language/1,程序设计语言1),是适合于科学计算的语言;LISP(List Processing,列表处理),是由斯坦福大学的麦卡锡(McCarthy)及其合作者设计的,用于处理符号表达式的概念简单的语言,其应用领域是人工智能[6];PROLOG(Programming for Logic,逻辑程序设计),是另一个用于人工智能的语言;SNOBOL(20世纪60年代中期在贝尔电话实验室设计的语言[7]),是以符号串作为处理对象的语言;SIMULA67是一个以模拟作为主要应用领域的语言,后来在CLU、Euclid和MODULA中被改进[8]。GPSS或SIMSCRIPT[9]则提供了可以改进或推广传统程序设计语言,以使模拟可以容易地描述的范例。其后又出现了通用程序设计语言ADA[10],这是为了纪念著名诗人拜伦的女儿艾达·奥古斯塔(Ada Augusta)而命名的。她曾同计算机的先驱查尔斯·巴贝奇(Charles Babbage,1792—1871)合作,为查尔斯·巴贝奇所研制的机器编写程序。巴贝奇曾于1820年至1850年间设计了两台机器,一台以有限差分为基础,称为差分机,另一台体现了现代计算机的许多原理,称为分析机。艾达·奥古斯塔就是为分析机编写程序的,因此,她被公认为世界上第一个程序设计员。起初是为编写操作系统UNIX的核心而被设计的C语言,后来被广泛应用于程序设计。

上面提到的语言基本上都是面向过程的语言。在20世纪60年代的软件危机之后,结构化程序设计方法被提出,催生了Pascal语言(以纪念法国数学家巴斯卡尔),它由瑞士的计算机科学家尼克劳斯·沃思(Niklaus Wirth)设计,作为解决软件危机的另一个方法学——面向对象的软件设计方法。它引起了面向对象语言的产生,例如,基于C语言,产生了C++语言。不久,基于C语言[11]又产生了JAVA语言。此外,SMALLTALK也属于面向对象的语言。

由于硬件的不断发展,对软件也提出了新的要求。新的计算机结构,如分布式系统、并行计算机系统计算机网络等对计算机程序设计提出了新的要求。因此,为满足这些要求又相继出现了新的程序设计语言[12,13]。

但是不管程序设计语言如何变化,有一件事是不变的,这就是以这些语言写成的源程序必须先由编译程序的翻译才能变成在计算机上可执行的目标程序。也就是说,它们遵循图1.3所示的模式。

这里的编译程序必须用机器语言编写,只有这样,它才能在计算机上执行,并且把源程序翻译成目标程序。然而,直接用机器语言写编译程序并非易事,工作量是巨大的。因此,人们就想用高级语言来编写它,然后使用以机器语言写成的简单的编译程序来编译它。这样通过两次编译过程,源程序就可实现实际可执行的目标程序了。这个过程如图1.4所示。图中有两个三角形。首先第二个即右边的三角形执行,产生目标程序。这个目标程序代替可执行的第一个即左边的三角形顶部编译程序,这样第一个三角形才可运行。通过它的运行,原来的源程序才可以由这个编译程序翻译成可执行的目标程序,这也就是

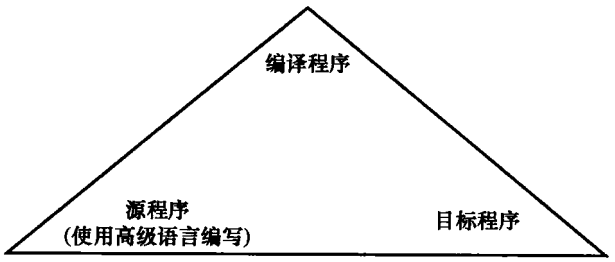

图 1.3 编译的过程

我们真正需要的。

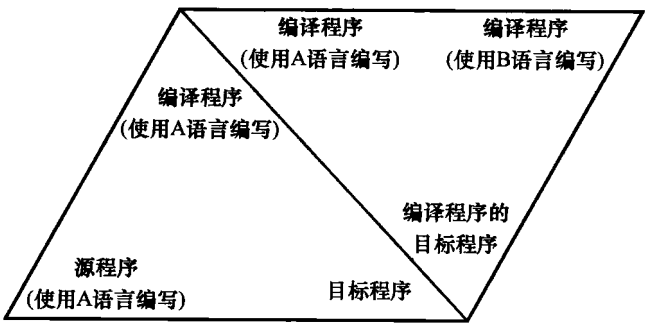

图 1.4 顺序的编译过程

这个模式可以进一步推广。例如,某人使用 A 语言来写源程序,而编译使用 B 语言写成。显然,在被编译成机器语言之前,这个编译程序不能运行。假设 B 的编译程序是用 C 语言写成的,以 B 语言写成的编译程序可以看作是一个源程序,而 C 语言写成的编译程序又被看作源程序,它被以机器语言写成的编译程序所编译,因此它可以编译用 B 语言写成的编译程序。这个用 B 语言写成的编译程序被编译为机器语言程序后再来编译以 A 语言写成的源程序。这样源程序也就变成可执行的目标程序了。

实际上,这个过程可以推广到任意级,只要最后一级可以执行,则从右到左,就可把最左边的源程序编译为目标程序。整个编译过程也就结束了。

在开发程序设计语言时,为使其具有生命力,必须遵循如下原则:

1) 具有明确定义的语法和语义描述

不论使用何种描述工具,目标都是,当用户设计程序时,语言的描述(包括语法和语义的描述)都应提供明确的信息,使用户可正确地工作。因此,程序设计语言要有明确的、易于理解的语法定义和语义描述。

就自然语言的使用而言,包括英文和中文在内,并非完美无缺,都有二义性和含糊的

问题。尽管这样,由于人类相比于计算机有较高的理解能力,因此人类面对的问题相对较少。但对于人们用来与计算机进行交互的程序设计语言而言,情况就不同了。即使在程序中仅出点小错或二义性,程序就不能正确运行或直接报告出错。程序设计语言的错误有时很难发现,比如:当唐·欧·克努特(D. E. Knuth)1967 年报告在 ALGOL60[14] 中仍余留的错误时,已是在 ALGOL60 发表和使用若干年之后了。很明显,如果人们正好使用到这些,那就会出错。这个例子也说明,语言的明确描述是不容易的。这也是为什么程序设计语言在 20 世纪 60 年代后不久便纷纷被淘汰了[15]。

2) 程序结构的确定性

程序结构的确定性同语言描述的明确性和易于理解性紧密相关。所谓程序结构的确定性,指的是语言中的任何程序都有明确的层次,易于理解,因而使它也易于设计、调试和修改。因此,结构的确定性不同于描述的明确性,但两者是相互补充的。

3) 快捷的翻译

这也可以说是翻译的高效性。当程序员设计程序时,必然希望这些程序能很快地被翻译。正如尼克劳斯·沃思所说,"对于编译程序来说分析简单的语言,人类程序员分析也简单,而这就是财富。"

高效无疑是评价程序语言质量的一个重要的准则。从人们开始使用程序设计语言的早期开始,就关注所设计程序的效率,因此高效比快速翻译有更宽的含义。高效的含义包括以下几层:

(1) 程序执行的高效率。开始时,当谈到高效率时几乎毫无例外地指程序执行的高效率。这涉及编译程序的质量,即以高效率的执行来把源程序编译成目标程序。因此,优化编译程序的设计、有效的寄存器分配以及支持程序运行的机制设计是非常重要的。尽管程序执行的效率和语言的设计密切相关,但编译程序编译的质量对程序执行的效率具有决定性的影响。

(2) 编译的效率。大型生产性程序是指经常运行的大型程序。因此,它的编译效率成为人们关心的问题。此外,其他类的程序也同编译程序的质量有关,如负责对学生所编程序编译或对用于教学程序编译的编译程序。典型地说,由学生编写的程序不用于生产,它们只是学生学习的成果,因此需要使之有效工作来编译程序,指出程序中的错误。在这种情况下重要的是编译程序的快速编译,而并不要求编译程序编译出有高的执行效率和优化的目标程序。

(3) 编写、调试和运行程序的效率。在程序设计语言的效率方面,其第三个含义是程序编写、调试和运行的效率。比如,如果人们使用一种语言来解决问题,结果是程序员花费在设计、编码、调试、修改和运行程序的时间和精力最少。因此,从实际意义上说,这是用计算机解决问题时某个语言效率的实际标准和测量。当使用一种语言来解决问题时,人们追求的效率正是这一点而不是传统的只考虑编译的效率。