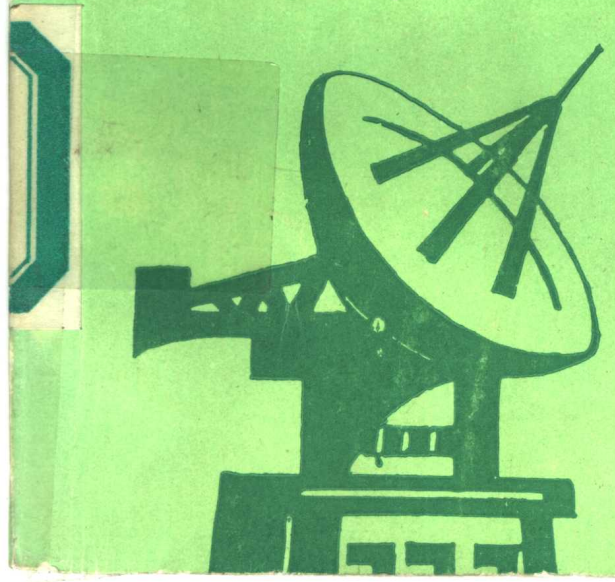
程序设计语言编译原理

国防科学技术大学　陈火旺
复　旦　大　学　钱家骅　编
上海交通大学　孙永强



国防工业出版社

程序设计语言

编 译 原 理

中国人民解放军
国防科学技术大学　陈火旺
复 旦 大 学　钱家骅　编
上海交通大学　孙永强

国防工业出版社

内 容 简 介

本书是在第一版的教学实践基础上，按高等学校工科电子类《计算机与自动控制》教材编审委员会计算机编审小组的修订意见修编而成。

本书旨在介绍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。作为一本原理性的书，重点在于介绍基本的理论和方法，不拘泥于具体的实现细节。本书既注意了最经典、最广泛应用的编译技术，又反映了七十年代以来的一些最重要的研究成果。在词法、语法分析方面特别注重分析器的自动产生；在翻译方面突出了语法制导翻译方法；在优化方面强调全局数据流分析。全书的组织注意了前后连贯，循序渐进，各章之后并附有习题。

本书可作为高等（理、工）院校计算机科学（或工程）专业的教材，或作为教师、研究生、高年级学生或软件工程技术人员的参考书。

程序设计语言
编译原理
中国人民解放军国防科学技术大学 陈火旺
复旦大学 钱家骅
上海交通大学 孙永强 编
国防工业出版社出版发行
（北京市海淀区紫竹院南路23号）
（邮政编码 100044）
新华书店经售
河北涿州中学印刷厂印刷
开本787×1092 1/16 印张20 458千字
1984年6月第2版 1996年5月第15次印刷 印数，128701－143700册
ISBN7-118-01256-4/TN·197 定价：18.00元

前　言

本教材系由《计算机与自动控制》教材编审委员会计算机编审小组评审选定，并推荐出版。

该教材由中国人民解放军国防科学技术大学陈火旺担任主编，华中工学院陶葆兰担任主审。编审者是依据计算机编审小组审定的编写大纲进行编写和审阅的。

本课程的参考教学时数，对硬件专业来说，删去所有打星号的章节后，为60学时，对软件专业而言，为80～100学时。但在学这门课之前，学生必须预修过计算引论(程序设计方法）和高级语言（FORTRAN、ALGOL或PASCAL)，并且最好具有数据结构和离散数学方面的基本知识。

本书旨在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，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。作为一本原理性教科书，我们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。本书既注意了最经典、最广泛应用的基本编译技术，如算符优先法和递归子程序法，又力求反映七十年代以来的一些最重要的新成果，如LR分析法和全局数据流分析。

在本书的引论中，我们概要地介绍编译程序的功能和结构。第一章的内容对于读者应是熟知的，这部分材料主要是对高级语言若干基本概念的回顾和复习。第二章是全书的基础，学生除了掌握词法分析器的基本构造方法外，还应该了解正规式和有限自动机的基本理论，它们和二、三、四各章的关系都很密切。第三章讲文法和基本语法分析器构造，这是现今所有讲授编译方法的教科书大多具备的部分。第四章介绍LR分析法和语法分析器的自动构造。第五章介绍各种中间代码及产生方法，着重讨论语法制导翻译产生四元式的过程。第六章讨论符号表组织。第七章讲存储分配问题。第八章介绍出错诊察处理方法。第九、十两章讨论局部优化、循环优化和全局数据流分析问题。第十一章介绍目标代码产生过程。

本书的体系是参照引论后的参考文献〔1〕和〔2〕的结构建立的，有些章节的基本内容也是从〔1〕移植过来的。大多数章节之后附有习题和参考文献。

本书的前七章由陈火旺编写，第八章由孙永强编写，九至十一章由钱家骅编写。陈火旺统编全稿。参加审阅工作的还有肖泽志、王春森和钱乐秋诸同志，他们对本书提出了许多宝贵意见，在此谨表示诚挚的感谢。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1983.7

目　录

引　论

0.1　什么叫编译程序

使用过现代计算机的人都知道，多数用户是应用高级语言来实现他们所需要的计算的。现代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的最重要组成部分之一，也是用户最直接关心的工具之一。

在计算机上执行一个高级语言程序一般要分为两步：第一步，用一个编译程序把高级语言翻译成机器语言程序；第二步，运行所得的机器语言程序求得计算结果。

通常所说的翻译程序是指这样的一个程序，它能够把某一种语言程序（称为源语言程序）改造成另一种语言程序（称为目标语言程序），而后者与前者在逻辑上是等价的。如果源语言是诸如 FORTRAN、PASCAL、ALGOL 或 COBOL 这样的“高级语言”，而目标语言是诸如汇编语言或机器语言之类的“低级语言”，这样的一个翻译程序就称为编译程序。

本书是介绍设计和构造编译程序的原理和方法的。编译理论与技术是三十年来计算机科学中发展得最迅速、最成熟的一个分支。现在已形成了一套比较系统化的理论与方法。本书是引论性的，我们不想罗列很多具体材料，只讲一些原理性的东西，但将反映一些较新的进展。

0.2　编译过程概述

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。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。

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，对于 FORTRAN 的循环语句

```
DO  150  I = 1, 100
```

词法分析的结果是识别出如下的单词符号：

基本字	DO
标　号	150
标识符	I
等　号	=
整常数	1
逗　点	,

整常数　　100

这些单词是组成上述 FORTRAN 语句的基本符号。单词符号是语言的基本组成成份，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的基础。如同将英文翻成中文的情形一样，如果你对英语单词不理解或对构词法不熟悉，那就谈不上进行正确的翻译。在词法分析这阶段的工作中所依循的是语言的**构词规则**。

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**（文法规则），把单词符号串分解成各类**语法单位**（**语法范畴**），如“**短语**”、“**子句**”、“**句子**”（“**语句**”）、“**程序段**”和“**程序**”。通过语法分解，确定整个输入串是否构成一个语法上正确的“**程序**”。语法分析所依循的是语言的语法规则。例如，在一般的面向科学、工程计算的语言中，符号串

X + 0.618乘Y

代表一个“**算术表达式**”。因而，语法分析的任务就是识别这个符号串属于“**算术表达式**”这个范畴。

第三阶段，**中间代码产生**。这一阶段的任务是对各类不同语法范畴按语言的语义进行初步翻译的工作。“翻译”仅仅在这里才开始涉及到。所谓“中间代码”是一种结构简单、含义明确的记号系统。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。例如，许多编译程序采用了一种与“三地址指令”非常近似的“四元式”作为中间代码。这种四元式的形式是：

算　符	左操作数	右操作数	结　果

它的意义是：对“左、右操作数”进行某种运算（由“算符”指明），把运算所得的值作为“结果”保留下来。在采用四元式作为中间代码的情形下，中间代码产生阶段的任务就是按语言的**语义规则**把各类语法范畴翻译成四元式序列。例如，下面的 FORTRAN 赋值句

Z = (X + 0.418)乘Y/W

可被翻译成如下的四元式序列：

序　号	算　符	左操作数	右操作数	结　果
(1)	+	X	0.418	T_1
(2)	*	T_1	Y	T_2
(3)	/	T_2	W	Z

其中 T_1 和 T_2 是编译期间引进的临时工作变量；第一个四元式意味着把 X 的值加上 0.418存放于 T_1 中；第二个四元式指将 T_1 的值和Y的值相乘存于 T_2 中；第三个四元式指将 T_2 的值除以Y的值留结果于Z中。

把语法范畴翻译成中间代码所依循的是语言的语义规则。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，除四元式之外，还有三元式、间接三元式和逆波兰记号等等。但是，对于某一个具体的编译程序来说，为了后续的代码优化阶段和目标代码生成阶段的方便，中间代码的选择往往与所采用的优化技术和目标机器

的体系结构有关。

第四阶段，**优化**。优化的任务在于对前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（省时间和省空间）的目标代码。优化的主要方面有：**公共子表达式的提取、循环优化、算符归约**等等。优化所依循的原则是程序的**等价变换规则**。例如，如果我们能把程序片断

```
for  K := 1  to  100  do
  begin
      M := I + 10*K;
      N := J + 10*K
  end
```

的中间代码

序　号	OPR●	OPN1	OPN2	RESULT	注
(1)	:=	1		K	K := 1
(2)	j<	100	K	(9)	若100<K转至第(9)个四元式
(3)	*	10	K	T_1	T_1 := 10 * K; T_1为临时变量
(4)	+	I	T_1	M	M := I + T_1
(5)	*	10	K	T_2	T_2 := 10 * K; T_2为临时变量
(6)	+	J	T_2	N	N := J + T_2
(7)	+	K	1	K	K := K + 1
(8)	j			(2)	转至第(2)个四元式
(9)					

转换成如下的等价代码：

序　号	OPR	OPN1	OPN2	RESULT	注　　解
(1)	:=	I		M	M := I
(2)	:=	J		N	N := J
(3)	:=	1		K	K := 1
(4)	j<	100	K	(9)	if(100<K)goto(9)
(5)	+	M	10	M	M := M + 10
(6)	+	N	10	N	N := N + 10
(7)	+	K	1	K	K := K + 1
(8)	j			(4)	goto(4)
(9)					

那么，最终所得的目标程序的执行效率就肯定会提高很多。因为，对于前者，在循环中需做300次加法和200次乘法；对于后者，在循环中只需做300次加法。尤其是，在多数硬件中，乘法的时间比加法的时间要费得多。

● OPR、OPN1、OPN2、RESULT分别表示算符、第一操作数、第二操作数、结果。

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）变换成特定机器上的**绝对指令代码**或**可重新定位的指令代码**或**汇编指令代码**。这阶段实现了最后的翻译，它的工作有赖于硬件系统结构和机器指令含义。这阶段的工作也是最复杂的，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，以及寄存器和后缓寄存器的调度，等等。如何产生出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。

如果最终所得到的目标代码是绝对指令代码，则这种目标代码可立即运行。如果目标代码是汇编指令代码，则这种目标代码需经汇编器汇编之后才可运行。必须指出，现代多数实用编译程序所产生的目标代码都是一种可重新定位的指令代码。这种目标代码在运行前必须借助于一个**连接装配程序**把各个目标模块连接在一起（包括I/O或数学子程序之类的系统模块），确定程序变量（或常数）在主存中的位置，对指令地址部分进行代真，使之成为一个可以独立运行的绝对指令代码程序。

上述编译过程的五阶段是一种典型的分法。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什么要求，优化阶段就可以省去。在某些情况下，为了加快编译的速度，中间代码生成阶段也可去掉。有些最简单的编译程序是在语法分析的同时产生目标指令代码的。但是，多数实用编译程序的工作过程大致都象上面所说的那五个阶段。

0.3 编译程序的结构

0.3.1 编译程序总框

上述编译过程的五阶段是编译程序工作时的动态特征。编译程序的结构可以按照这五阶段的任务分模块进行设计。编译程序的结构设计可从图0.1所示的总框开始，这个总框是一个数据转换图，由它能够比较容易地设计出程序结构的框图。

图0.1中的词法分析器、语法分析器、中间代码生成器、优化段和目标代码生成器将分别完成上述五阶段的编译任务。每阶段的输出为下一阶段的输入，第一阶段的输入是源程序，最后阶段的输出为目标代码程序。每阶段的工作都和“表格管理”和“出错处理”这两部分功能模块相关。

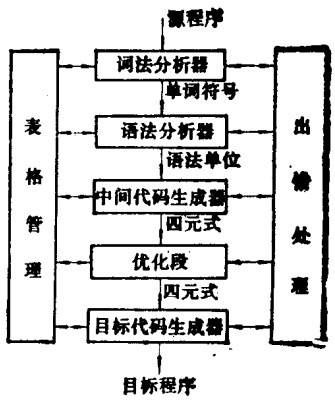


图0.1 编译程序总框

词法分析器，又称**扫描器**，输入源程序，进行词法分析，输出单词符号。

分析器，全称**语法分析器**，对单词符号串进行语法分析（根据语法规则进行推导或归约），输出由语法单位构成的语法树，判断输入串是否构成语法上正确的“**程序**”。

中间代码生成器，按照语义规则把语法分析器归约出（或推导出）的语法单位翻译成一定形式的中间代码，譬如说四元式。

在许多编译程序中，扫描器、分析器和中间代码生成器三者并非是截然分开的，而

是相互穿插的。这样可以大大提高编译程序自身的工作效率。

优化段，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。

编译过程中源程序的各种信息被保留在种种不同的表格里，编译各阶段的工作都涉及到构造、查找、或更新有关的表格。因此，在编译程序中必含有一组管理各种表格的程序。

如果源程序有错误，编译程序应设法发现错误，把有关出错信息报告给用户。这部分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地点，并且能将错误所造成的影响限制在尽可能小的范围内，使得源程序的其余部分能继续被编译下去，以便进一步发现其它可能的错误。如果不仅能够发现错误而且还能自动校正错误，那当然就更好了。但是，自动校正错误的代价是非常高的。

出错处理程序与编译各阶段都有联系，与前三阶段的联系尤为密切。

0.3.2 表格与表格管理

编译程序在工作过程中需要保持一系列的表格，以登记源程序的各类信息和编译各阶段的进展状况。合理的设计和使用表格是编译程序构造的一个重要问题。正是在这些方面我们需要数据结构的一些基本技巧和方法。表格的分类、表格的结构和表格的处理是与所编译的源语言有关的，同时也与所采用的优化措施和目标机器系统有关。一般而言，与编译的头三个阶段有关的重要表格有：

符号名表：登记源程序中的常数名、变量名、数组名、过程名等等的性质和定义、引用状况；

常数表：登记源程序中出现的各种类型常数的值；

标号表：登记源程序中出现的标号的定义和引用情况（此表视情况可合并于符号名表之中）；

入口名表：登记每个过程的入口名和入口位置；

过程引用表：登记所引用的外部过程（包括系统过程）的名字和引用位置。

这些表格的结构大体上都象下面所示的格式：

NAME （名字）	INFORMATION（信息）

其中信息栏通常还分成若干子栏分别登记名字的各种属性和状态（如赋值、引用情形）。

在编译的头三阶段中所用的表格还有：循环特征表、等价名表、公用链表、格式表、中间代码表（如下面即将看到的记录四元式序列的四元式表），等等。

作为一个例子，让我们来看一看 FORTRAN 编译程序常用的几种表格的结构。例如，对于下面的程序段

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

```
      RETURN
      END
```

经编译的头三阶段工作后所产生的主要表格有：符号名表 SNT（表 0.1）、常数表 CT（表 0.2）、入口名表 ENT（表 0.3）、标号表 LT（表 0.4）和四元式表 QT（表 0.5）。

表0.1 符号名表 SNT

	NAME	INFORMATION
(1)	M	哑元。整型。变量
(2)	N	哑元。整型。变量
(3)	K	整型。变量

表0.2 常数表CT

	值(VALUE)
(1)	1
(2)	4

表0.3 入口名表ENT

	NAME	INFORMATION
(1)	INCWAP	二目子程序。入口四元式:1

表0.4 标号表LT

	LABEL	INFORMATION
(1)	10	四元式:(4)

表0.5 四元式表QT

	OPR	OPN1	OPN2	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 SNT 表记录了源程序段里出现的三个符号名M、N和K的有关性质；CT 表记录了常数 1 和 4 的值（已被翻成内部表示）；ENT 表记录了入口名 INCWAP 的入口地址：四元式序号；LT 表记录了标号 10 的对应四元式序号；QT 表记录了源程序段所对应的全部四元式：

link 保护返回地址和各有关寄存器；

actpar, INCWAP, 1, M 把 INCWAP 的第一个实在参数的值传送到 M 中；

actpar, INCWAP, 2, N 把 INCWAP 的第二个实在参数的值传送到 N 中，

+，M，1，K　指K:=M+1；

+，N，4，M　指M:=N+4；

:=，K，—，N　指N:=K；

paract，INCWAP，1，M　把M的值回送到INCWAP的第一个实在参数地址中；

paract，INCWAP，2，N　把N的值回送到INCWAP的第二个实在参数地址中；

return　恢复寄存器，按返回地址把控制返回到调用段。

注意：在四元式表中实际上不是直接写上操作数（或结果数）的名字而是填上它们在有关表格中的入口位置（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（中间代码的设计也是一件很不容易的事情），周密地考虑各种全局性名表（即各阶段都要用到的表格）的信息安排。这些事情出了差错势必导致后来的大返工，甚至招致无可挽回的失败。

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异。由于各种信息是被保留在种种不同的表格中，因此，编译过程的绝大部分时间是花在造表、查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为重要的。

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。它的工作就象一位秘书一样，按照上司的指令，抄抄写写，专做簿记工作。“表格管理”和编译的各阶段程序之间是相互作用的。

0.3.3 遍

编译程序的结构是十分复杂的，而且体积也很大。由于受到具体机器主存容量的限制，往往把编译的几个不同阶段的工作组合成**遍**（pass）。每遍的工作由从外存（如磁盘、磁鼓或磁带等）上获得前一遍的工作结果开始（对于第一遍而言，从外存上获得源程序）。完成它所含的有关阶段程序的工作之后，再把结果记录于外存之中。每遍所含的诸阶段工作往往是穿插进行的，施行控制的是每遍有关控制程序。当一遍工作完之后，它所占用的存储空间大部分被释放。下一遍进入后，即可使用几乎全部的存储空间。至于一个编译程序究竟应分成几遍，如何划分，是和所面临的具体机器的内、外存设备有关的，因此难于统一划定。遍数多一点有个好处，即整个编译程序的逻辑结构可能清晰一点。但遍数多势必增加输入/输出所消耗的时间。因此，在主存可能的前提下，一般还是遍数尽可能少一点为好。

0.4 编译程序的生成

以前人们构造编译程序大多是用机器语言或汇编语言作工具的。为了充分发挥各种不同硬件系统的效率，为了满足各种不同的具体要求，现在许多人仍然采用这种工具来构造编译程序。但是，越来越多的人倾向于使用高级语言作工具来构造编译程序。因为，

这样可以节省大量的程序设计时间，而且所构造出来的编译程序也易于阅读、修改和移植。

现在人们已经建立了多种编制部分编译程序或整个编译程序的有效工具。有些能用于自动产生扫描器，有些可用于自动产生语法分析器，有些甚至可用来自动产生整个的编译程序。这些构造编译程序的工具有：**编译程序-编译程序**、**编译程序产生器**、**翻译程序书写系统**等，它们是按照对源语言和目标语言（或机器）的形式描述(作为输入数据)而自动产生编译程序的。

近年来，有些人主张采用“自编译方式”产生编译程序。意思是，先对语言的核心部分构造一个小小的编译程序（可用手编实现），再以它为工具构造一个能够编译更多语言成分的较大编译程序。如此扩展下去，就象滚雪球一样，越滚越大，最后形成人们所期望的整个编译程序。这种通过一系列自展途径而形成编译程序的过程叫做**自编译过程**。

现在，有些编译程序是通过“移植”而得的。即把某一机器上的编译程序移植到另一机器上。这需要寻找某种适当的“中间语言”。但是，由于建立通用中间语言实际上办不到，因此，移植也只能在几种语言和几种机种之间进行。

对于自编译和移植有兴趣的读者，请参阅有关专门论著。但对于自动产生器，本课程将把它作为一个重要课题来讨论。

0.5 学习构造编译程序

要在某一台机器上为某种语言构造一个编译程序，必须掌握下述三方面的内容：

1. 源语言　对被编译的源语言（如 ALGOL 或 FORTRAN），要深刻理解其结构（**语法**）和含义（**语义**）。

2. 目标语言　假定目标语言是机器指令，那么，就必须搞清楚硬件的系统结构和操作系统的功能。

3. 编译方法　把一种语言程序翻译成另一种语言程序的方法很多，但必须准确地掌握一二。

本课程是讲编译方法的，并且主要是讨论 ALGOL 和 FORTRAN 之类语言的翻译技术。尽管假定读者对这些语言已有一定的基本知识，但为了有所衔接，在第一章，我们仍将复习一下这些语言的基本概念。

在本门课中，我们并不假定以某一特定机器作为目标机器。当需要涉及目标机器指令时，将采用一些人所共知的假想指令。因此，在学习这门课之前，读者必须具有计算机基础程序设计的知识。

由于编译程序是一个极其复杂的系统，故在讨论时，只好把它肢解开来，一部分一部分地进行研究。因此，在学习过程中应注意前后联系，切忌用静止的、孤立的观点看待问题。作为一门技术课程，学习时务必注意理论联系实际。多做练习，多多实践。

本书中所引用的具体算法大多是用文字描述的，有些是用类似 PASCAL 的语言表示的。所有这些算法都是原理性和解释性的，而且大多是不完备的（忽略某些次要因素或尚未学到的成分）。因此，并不意味着这些算法可以直接照抄使用。

在着手构造一个编译程序时，需要预先考虑种种具体因素〔诸如、系统功能要求(这种要求常常是多方面的)、硬件设备、软件工具等等〕，特别是，必须估量所有这些因素对编译程序构造的影响。虽然这些都是工程实现时应予考虑的细节，但因篇幅所限，不可能涉及太多。

后面，在复习高级语言的基本概念之后，我们将按照 0.2 节中所说的编译过程的各基本阶段，逐步介绍编译程序的构造方法和技术。

参 考 文 献

〔1〕 Aho, A. V. and Ullman, J. D., Principles of Compiler Design, Addison-Wesley, 1977.
〔2〕 格里斯，数字计算机的编译程序构造，科学出版社，1976。