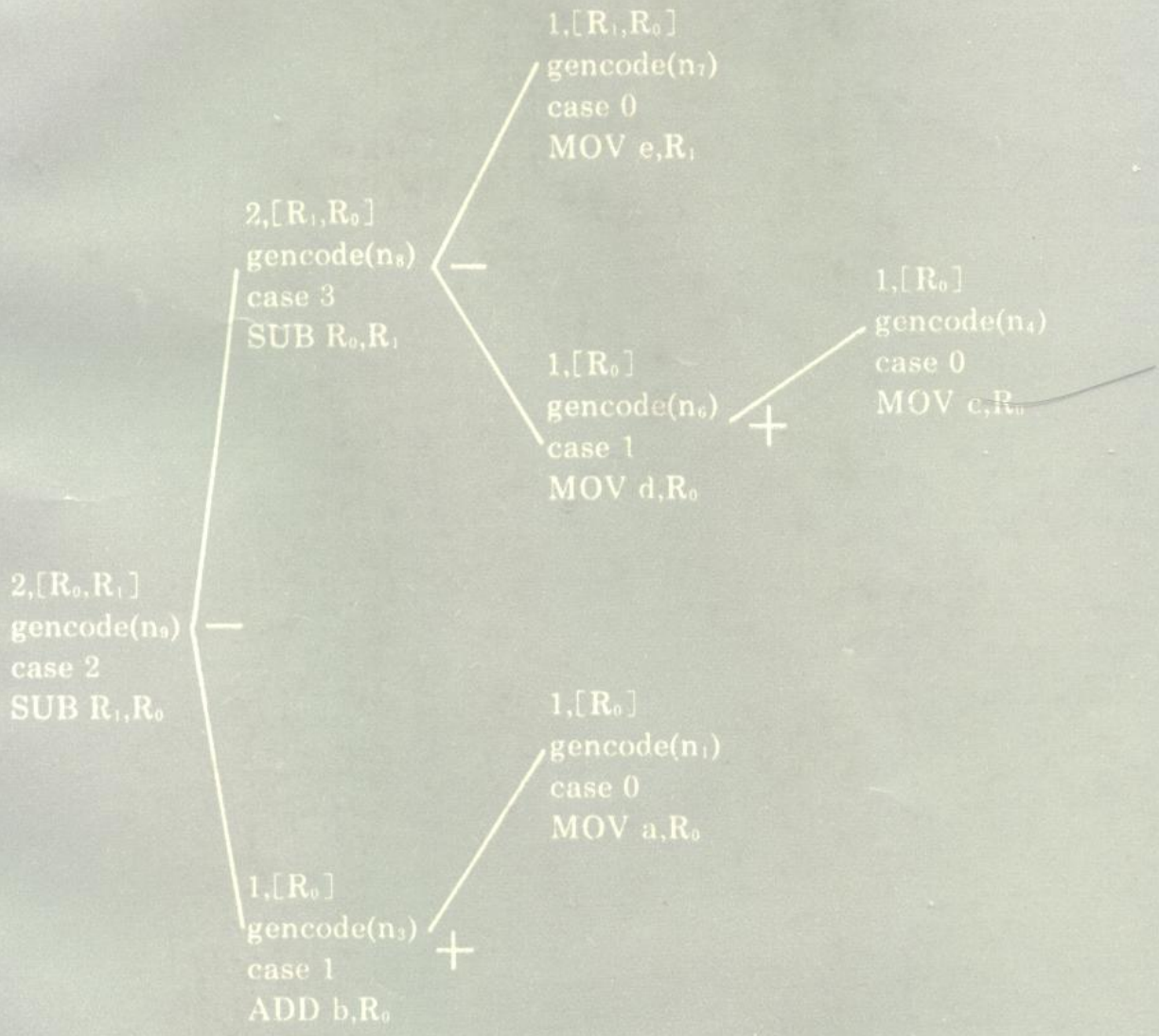
侯文永　编著

编译原理及其实现技术



上海交通大学出版社

编译原理及其实现技术

侯文永　编著

上海交通大学出版社

内容简介

本书系统介绍了编译程序设计的基础、原理和方法。在详细讨论各种语法分析方法的基础上，注重词法分析器、语法分析器的自动生成。在属性文法基础上讨论了语法制导翻译的形式，语义分析趋向形式化。控制流、数据流分析则是代码优化中的重要技术。本书力求易读，注重对能力的培养。本书可作为高等院校计算机系各专业编译原理或编译技术课程的教材或参考书，也可作其他专业师生及计算机工作者的参考书。

(沪)新登字 205 号

编译原理及其实现技术

出版：上海交通大学出版社
（上海市华山路1954号·200030）
发行：新华书店上海发行所
印刷：立信常熟印刷联营厂
开本：797×1092（毫米） 1/16
印张：14.25

字数：348,000
版次：1993年11月 第 1 版
印次：1993年12月 第 1 次
印数：1—2,000
科目：308—244

ISBN 7-313-01280-2/TP·39

定 价：7.20 元

前　言

编译原理是计算机专业开设的一门重要的专业课程，虽然只有少数人从事编译方面的工作，但是这门课在理论、技术、方法上都对学生提供了系统而有力的训练，是提高软件人员的素质和能力的重要环节。这本书对编译的主要内容作了较深入的讨论。

本书共九章，第一章引论，概述了编译、编译结构及编译的构造方法。第二章文法和语言，简要介绍了必要的基础知识。第三章词法分析，既介绍了词法分析器设计的一些问题，也介绍了关于词法分析器自动生成的原理和方法，正规式和有限自动机及其等价变换的内容。第四章语法分析，以较大篇幅详细介绍了好几种典型的语法分析方法，特别是表驱动的分析方法，讨论了这些语法分析器自动生成的原理和方法。第五章语法制导翻译，引进了属性文法的概念，语义规则与产生式相联系的语法制导定义或翻译方案，讨论了计算属性值的一般方法。第六章运行时的环境，讨论了存贮组织，符号表等如何支持程序语言的作用或规划，静态或动态分配等要求。第七章中间代码生成，则对具体的文法，讨论生成一种中间代码的技术和方法。可以说是语法制导翻译的具体化。第八章代码优化，以不少的篇幅讨论了中间代码级上的局部优化和循环优化，较详细地介绍了控制流分析和数据流分析的方法。第九章代码生成，根据模型机讨论了由中间代码生成目标代码的方法，也涉及了目标代码级的优化。

这些年关于编译原理的教材陆续出版了一些，书后的参考书目列出了其中一部分，编者根据本校特点，并吸取了这些参考书的长处，为学生讲授编译原理课程多年，本书是在讲稿基础上形成的。高汉钦为本书的前第四章提出了修改意见，须继红打印了全部图稿，张琴芬誊写了全部手稿，作者深致谢意。由于编者水平有限，错误与不妥之处难免，敬请批评指正。

侯　文　永
上海交通大学计算机系
1993 年 4 月 30 日

目　录

第一章 引 论

当你拿到这本书的时候，你一定想问编译程序是什么？它与翻译程序、解释程序、汇编程序之间有什么区别？如何来设计、构造或了解一个编译程序？本章将就上述问题作出初步讨论。

§1.1 编译程序是一种特定的翻译程序

描述计算过程通常借助于一种程序设计语言，这种计算过程包含了从初始状态转变为终止状态的一系列的操作。描述算法的语言可有很多种，将一种语言写的程序转换成另一种语言写的程序，这就是翻译，实现这种功能的程序便是翻译程序，显然翻译前的程序与翻译后的程序两者应等价。翻译前的程序为源语言程序，简称源程序，翻译后的程序为目标语言程序，简称目标程序。如将PL/1或COBOL写的程序转换成C语言写的程序，便属翻译之列，只是构造这样的翻译程序十分困难。

事实上，人们几乎都用面向用户的高级语言来描述他们的计算，除了理论上的尝试而外，几乎所有的计算机都无法直接执行高级语言写的程序，因为计算机只能执行面向机器的机器语言程序。为了能执行，必须将高级语言程序转换成机器语言程序，这种转换程序，便是编译程序。显然编译程序是翻译程序中的一类。

如果我们把源程序记为S，目标程序记为T，编译程序记为C，那么可以将编译看成一个函数，一种映射：

$$T=C(S)$$

人们常将可直接执行的机器语言的指令系统符号化，这便是汇编语言，当然汇编语言程序也必须转换成机器语言程序才能执行，这种转换程序便是汇编程序。汇编程序也是一种翻译程序，由于符号化的指令系统与机器指令系统之间有比较明确的对应关系，因此作为它们之间变换的汇编程序就比较容易构造，而且由于汇编语言和机器语言一样都是面向机器的，故相对于面向用户的高级语言而言，我们便将它们都称为低级语言，所以编译程序是将高级语言写的源程序转换成低级语言写的目标程序的翻译程序。

一个高级语言程序的执行通常分成两个阶段，即编译阶段和运行阶段，编译阶段将源程序变换成目标程序，运行阶段，是由编译阶段生成的目标程序连同运行系统（数据空间分配子程序，标准函数程序等）接受程序的初始数据作为输入，输出计算结果，见图1.1。

如果编译生成的目标程序是汇编语言写的程序T′，那么在编译与运行阶段中间还得增加一个汇编阶段，它将编译生成的汇编语言程序T′经汇编程序A变换成以机器语言写的目标程序T，见图1.2。

程序除了编译执行外，还可以有另一种执行方式即解释执行。解释执行是将源程序中的语句按动态顺序，逐句逐段翻译成可执行代码，一旦具备执行条件(获得必要的初始数据等)立即将这一段代码执行得到部分结果。即解释执行是按动态顺序步进式推进着翻译和执行。

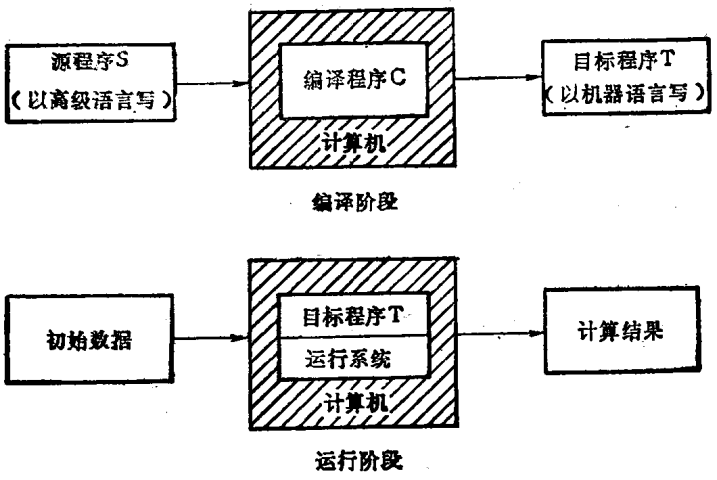


图1.1 程序的编译执行

图1.2 汇编阶段

图1.3 程序的解释执行

完成这样功能的程序称为解释程序，可以用I来记它，程序的解释执行与如图1.3所示。

解释执行特别适合于交互式语言，也用于抽象机语言，但通常情况下执行效率不如编译执行高，因为编译产生整个程序的目标代码，就可以在全局范围做压缩空间缩短运行时间的优化工作；同时编译是按程序的静态顺序线性扫描源程序的，源程序中重复执行的成分，那只是运行阶段的重复，编译阶段并不因此重复，而解释执行则是按动态顺序重复翻译，重复执行的，速度自然会慢得多。

§1.2 编译程序的结构

编译程序将源程序变换成目标程序，这是个非常复杂的过程，通常分成五个阶段：

一、词法分析阶段

对组成源程序的字符串进行扫描和识别，识别出一个个具独立意义的单词(或称符号)，如基本字(if，for，begin等)、标识符、常数、运算符、界符(括号，=，；等)，将识别出的单词用统一长度的标准形式(也可称为内部码)来表示，对以后的变换来说，这种标准形式对区分单词和单词的属性应是十分方便的。因此词法分析是将字符串变换成单词的符号流。

二、语法分析阶段

在词法分析输出的单词流基础上，根据源语言的语法规则分析这种单词流是否正确地组成了各类语法单位，如短语、子句、程序段和程序等。例如 2＊3。1416＊R＊R 是表达式，单词符号:=后应跟着一个表达式等。

三、语义分析、中间代码生成阶段

经过语法分析的单词流若在语法结构上是正确的，就可以在这个基础上做实质性的翻译工作，虽然可以直接翻译成可执行代码(机器语言或汇编语言程序)，但通常是将单词流翻译成在语义上是等价的中间语言程序。中间语言是介于高级语言和低级语言之间、但并不面向具体机器、语言结构又十分接近低级语言的一种中间形式。中间语言相对于低级语言而言，既有一定程度的超脱，又有一定程度的对应。中间语言程序到目标语言的转换是容易的，因此语义上的等价变换主要在语义分析阶段进行。其任务是根据语言的语义规则，将语法单位逐一翻译成中间语言代码，这通常称为是语法制导翻译。

四、优化阶段

前一阶段产生的中间代码是以语法单位这样的局部区间为单位，不能保证效率是高的，有必要进行等价变换，使程序占用空间少，运行时间短。常用的优化措施有删除冗余运算，删除无用赋值，合并已知量，循环优化等，有些优化措施效果很明显，例如循环中参于运算的运算量，如其值并不随着循环而发生变化，这类运算称为循环不变运算，它完全不必每次循环都计算一次，将它们提到进入循环前计算一次即可。

五、目标代码生成阶段

将中间代码转换成机器语言程序或汇编语言程序，最后完成了翻译，这阶段的工作因为目标语言的关系而十分依赖于硬件系统，如何充分利用寄存器、合理选择指令、生成尽可能短而有效的目标代码，都与目标机的结构有关。

生成的目标代码如果是汇编语言程言，则需经由汇编程序汇编后才能执行。生成的目标代码如果是绝对指令代码，则已可直接投入运行。如果是可重定位的指令代码，那么目标代码只是一个代码模块，必须由连接装配程序，将输入／输出模块，标准函数等系统模块与目标代码模块连接在一起，确定数据对象和各程序点的位置(即代真)才可能形成一个绝对指令代码程序供运行。

以上五个阶段反映了编译程序的动态特征，但编译的实现还有赖于符号表管理和出错管理。

六、符号表管理

源程序中的有关数据对象的信息和程序信息是编译各阶段要经常使用的，因此编译程序中还应包括一个符号表管理程序，它涉及到符号表的构造，查询和更新等各种操作，提供用户命名的标识符的各种属性、存贮位置、类型、作用域等信息，对这些信息的操作，是频繁的，占了编译时间的很大一个比例，因此符号表的结构，符号表上各种操作的算法必须精心设计。

七、出错管理程序

编译的各个阶段都可能遇到错误。出错管理程序应在发现错误后，将错误的有关信息如错误类型、错误地点向用户报告。为了尽可能多地发现错误，在发现了错误后还应继续编译下去，因此要设法将错误造成的影响限制在尽可能小的范围中，发现错误后能自动校正错误当然最好，只是在大部分情况下，校正的代价太大，而且也未必尽如人意。

因此编译程序的总框图可用图1.4来表示。

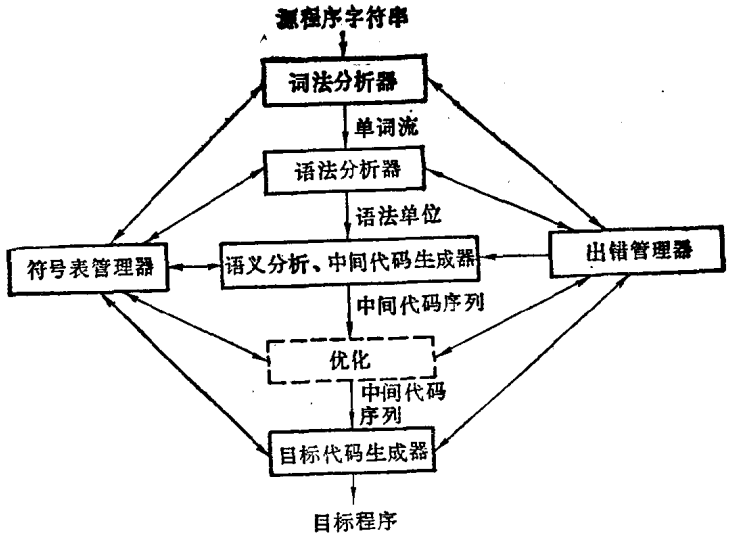


图 1.4 编译程序总框图

其中优化是可选部件，对于像学生的作业这类量不大，不会反复运行的程序来说，可以不经过优化阶段，也即应该在优化所需花的代价和优化能取得的效益之间作出选择。

八、编译阶段的前端和后端

编译的前三个阶段(词法分析、语法分析、中间代码生成)称为编译的前端。它其实还可分成分析(词法分析、语法分析、语义分析)和综合(代码生成)两个过程。而目标代码生成则是编译的后端。中间代码的优化可归入前端。很明显编译后端是面向目标语言的，而编译前端则不是，它几乎独立于目标语言。

引进中间语言生成中间代码，不仅使编译程序的逻辑结构更合理，有利开展中间代码上的优化，而且对编译的生成和移植更有利。将源语言S的程序转为目标语言T的编译程序C，如果想移植到目标语言为T′的系统上去，那么只要重写编译C的后端，使之生成T′的目标代码即可，同样如果源语言为S′目标语言仍为T，则它们间编译程序只要改写C的前端使之面向S′即可。

因此从形式上说，如果不设中间代码，不将编译分成前端F和后端B，那么m个高级语言和n种目标语言之间需要m×n个编译程序C。反之我们只需要m个前端和n个后端即可；故工作量为m×F+n×B。但是要选择适合m个高级语言和n个目标语言的中间语言并不容易。因为一种通用语言是很难表示一种特定语言的微妙特性的。

九、遍

在编译过程中，从头至尾地扫描一个输入文件，经变换形成一个输出文件，这个过程称为编译的一遍。

除了符号表管理程序和出错管理程序外，编译的其他五个阶段都可以单独组成一遍。如果将词法分析作为一个子程序，语法分析器每当需要时调用词法分析程序获得一个单词，每当按语法规则发现形成一个短语(句柄)时，便调用相应的语义子程序，做翻译工作，生成中间代码，那么将词法分析、语法分析、语义分析及中间代码生成组成一遍是十分自然的。因此编译过程一般可用两遍或三遍完成。

编译程序任务繁重，结构复杂，占用空间也很大，因此将编译过程分割成几遍来完成，不仅使编译程序结构清晰，而且每遍只完成一部分工作。对资源需求也可降低要求。

将编译分割成几遍，除了上述原因外，当然有逻辑上的需要，也有优化上的需要，但最初的根本原因是内存的不敷使用，曾有一内存仅4K的机器，将近4万条指令的编译程序分成19遍，每遍占用内存仅2K上下，这在内存代价昂贵的当时，可以说是编译分遍的一个杰作。

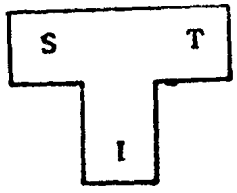
并不是说编译程序扫描遍数越多越好，因为从头到尾的每遍扫描使编译开销上升很多，有一些技术就是为了减少遍数的，如中间代码生成中的回填技术。

§1.3 编译程序的生成

编译程序是一个足够复杂的程序，何况语言功能的完善，硬件结构的发展，环境的友好要求，都对编译程序提出了更多更高的要求。因此一个编译系统的构造并非易事，对完全想用手工方法来构造编译器来说更是如此。好在现在有各个层次的生成工具，如lex，yacc，GAG，CGSS等，而环境也提供开发工具来帮助产生高效的编译器，如make等，只要熟练运用这些工具，构成编译器的难度与强度都可大为降低。在以后章节中将把lex，yacc等工具的自动生成原理作为重要内容予以介绍。其实，现在要构造一个完全独立的全新的编译器可能性很小，大部分可在现有编译器的基础上扩展，有的采用自展的方式，有的则用移植的方式。不管是用手工方法、自动生成方法、自展和移植方法、或者兼而有之的综合方法，构造者对源语言的语法结构和语义，对目标语言及其面向的目标机的系统结构，操作系统功能应有准确的理解。

一、自展

一个编译程序一般涉及三种语言，因为一个编译程序本身也是一个程序（这个程序的功能是将S语言的程序等价变换成T语言的程序），它也必须以一种语言来书写，它可以是既不同于S，又不同于T的语言I，称它为编译的实现语言，因此一个编译程序C的完整表示为 $C_I^{S,T}$，或者用图形来表示，因为它的形状像字母T，故称为T图。请注意T图上方S，T是这个程序具有的功能(将S语言的程序等价变换T语言的程序)，T图下方的I是书写这个程序所用的语言。只是除了构造编译程序外，通常使用的编译程序都是 $C_A^{S,A}$ 形式的，即A机器上的，高级语言S的编译程序。

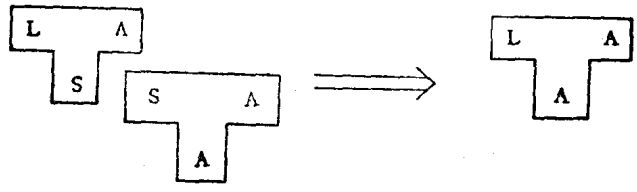


不言而喻，对一个相当规模的语言L，想用目标语言在目标机A上构造一个编译程序 $C_A^{L,A}$ 是十分困难的。为此，我们首先可以选择L的一个子集S，子集的规模只要达到有足够的描述能力即可。相对于L而言，S的编译程序 $C_A^{S,A}$ 就容易写得多，不妨认为它可以用手工完成。

第二步，我们可以用S语言来写一个L→A的编译程序 $C_S^{L,A}$，相对于A语言来说，用S语言写这样的编译程序也会容易得多。然后将这个S语言的程序 $C_S^{L,A}$ 作为S→A的编译程序的输入。其输出当然是A语言的程序，并且由于 $C_A^{S,A}$ 保证这种变换的等价性，因此输出程序将保持输入程序 $C_S^{L,A}$ 的功能，即将L语言程序等价变换成A语言程序，故输出程序是一个A语言书写的程序，其功能为将L语言程序等价变换成A语言的程序，即L→A的编译程序 $C_A^{L,A}$ 这便是我们所要求的：

$$C_S^{L,A} \longrightarrow \boxed{C_A^{S,A}} \longrightarrow C_A^{L,A}$$

用T图来表示，有：

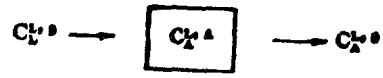


上述构造 $C_A^{L,A}$ 的过程便是编译的自展，或称为自编译。S为核。如果L相当大，则可以设 $S \subset S_1 \subset S_2 \cdots \subset L$，通过多次自展逐步滚大。

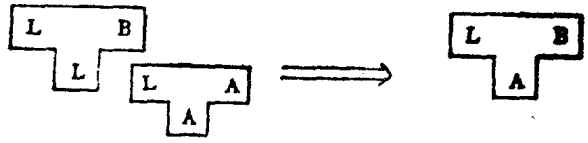
二、移植

已有 $C_A^{L,A}$，想借助于 $C_A^{L,A}$，得到 $C_B^{L,B}$，这便是移植。也即将A机器上L语言的编译移植到B机器上。

做法是：首先用L语言写一个L→B的编译程序 $C_L^{L,B}$，将这个L语言程序作为 $C_A^{L,A}$ 的输入，输出为 $C_A^{L,B}$

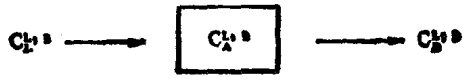


用T图表示有：

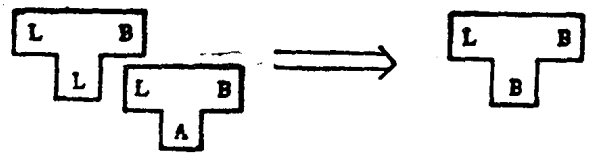


显然 $C_A^{L,B}$ 也是一个编译程序，只是用A语言写的将L语言程序编译为B语言的程序。这种在A 机器上生成B机器的目标代码的编译称为交叉编译。

第二步将 $C_L^{L,B}$ 作为交叉编译 $C_A^{L,B}$ 的输入，其输出 $C_B^{L,B}$ 即为所求，即：



用T图表示有：



三、对编译程序的评价

不管用什么方法生成的编译程序，对它的质量评价首先当然是正确性，即源程序到目标程序变换的等价性。在正确性的前提下，对编译程序的质量有二个层次的评价：

（1）生成的目标质量。这主要用目标代码运行时间的长短，占用空间的大小来评定。

（2）编译程序本身的质量。这主要用编译速度的快慢，编译占用空间的大小，以及编译程序本身的结构是否良好，可读性、可维护性如何来评定。

要提高目标程序的质量，所选择的编译的方法，算法当然很重要，特别优化是不可忽视的，但是提高目标程序质量的种种措施都是需要代价的，它们会影响编译程序本身的质量，这两方面的要求应有所权衡。需要指出的是片面追求目标质量而忽视编译本身的质量，以致要花很长时间才能将一个源程序编译成目标程序，这样会使用户失去信心，甚至不可容忍。另外编译程序的可移植性与可维护性也应重视。

§1.4 编译程序的学习

很明显，极少有人会有构造一个编译程序的机会，但是编译课程中所介绍的一些原理和方法、算法并不局限于编译。有限自动机的原理，形式描述的方法，自动生成的方法，数据