金成植 编著

ALGOL60

编译方法

上册

科学出版社

ALGOL60 编译方法

上 册

金成植 编著

科 学 出 版 社

1983

内 容 简 介

本书较系统地介绍了 ALGOL 60 语言的基本编译技术、各种优化技术以及编译中的一些基本理论。

本书分上、下册。上册共十章，前五章介绍编译程序的基础知识，主要描述数据结构、查表技术、形式语言的语法分析理论和中间语言；随后五章通过一个教学用通用编译程序，具体地讨论了实现编译程序的全过程。下册共六章，讲述了几种局部优化和全局优化的基础理论。

本书可供大专院校软件专业的师生、从事软件工作的科技人员及自学编译方法的人员参考。

ALGOL60 编译方法

上 册

金成植 编著

责任编辑 杨家福 邢莉莉

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1983 年 5 月第 一 版 开本：787×1092 1/32

1983 年 5 月第一次印刷 印张：13 1/4

印数：0001—12,500 字数：301,000

统一书号：15031·488

本社书号：3049·15—8

定价：2.05 元

序　言

编译程序是计算机系统的重要组成部分，因此，编译理论和编译技术是计算机科学中的一个重要领域．本书是在吉林大学计算机科学系用过多年的《编译方法讲义》的基础上，经过比较全面的修改和补充而写成的．内容力求系统、完整，可帮助初学者了解并掌握编译程序的整个过程．

全书内容可分成三个部分．第一至五章为第一部分，介绍编译程序的基础知识，主要是数据结构、查表技术、形式语言的语法分析理论和中间语言．作者认为，形式语言的语法分析理论是编译程序的基础，因此用了较多的篇幅描述了几种比较实用的语法分析方法．

第六至十章为第二部分，是本书的核心，通过一个教学用通用编译程序，系统地介绍了编译原理和编译程序的全过程．要掌握编译技术必须精读一个完整的编译程序．但是，因为编译程序通常都很庞大，一般资料中又不介绍编译原理，所以任何一个具体机器上的编译程序都很难阅读．鉴于这种情况，我们设计了上述供教学(自学)用通用编译程序．

第十一至十六章为第三部分，主要介绍几种局部优化和全局优化的基础理论．局部优化部分主要包括基本块上的优化(常表达式节省和公共子表达式节省)、循环上的优化(不变表达式外提和运算强度削减）及下标变量的单独优化．全局优化的基础理论部分主要包括程序的控制流分析和数据流分析．

本书不要求读者熟悉具体的机器．书中所给出的教学用

编译程序经过几年的教学实践，已比较完善和稳定.

徐立本同志审阅了本书的全部手稿，并提出了宝贵意见，特此致谢.

目　　录

序言 …… v

第一章　翻译程序 …… 1

1.1　汇编语言和汇编程序 …… 2

1.2　高级语言和编译程序 …… 5

1.3　解释程序 …… 9

第二章　数据结构 …… 13

2.1　数组（ARRAY） …… 13

2.2　栈（STACK） …… 18

2.3　排队（QUEUE） …… 20

2.4　表（LIST） …… 22

第三章　查表技术 …… 26

3.1　引言 …… 26

3.2　顺序查表法 …… 26

3.3　平分查表法 …… 27

3.4　散列查表法 …… 29

3.5　查表速度 …… 42

第四章　形式语言及其语法分析 …… 45

4.1　形式语言 …… 45

4.2　语法分析 …… 61

第五章　波兰式和四元式 …… 135

5.1　波兰式 …… 135

5.2　四元式 …… 142

第六章　ALGOL 单词翻译（I） …… 151

6.1　引言 …… 151

6.2 单词机内符 …… 153
6.3 单词的翻译方法 …… 159
6.4 SCANER-1 …… 167
6.5 单词翻译举例 …… 173
第七章 ALGOL语法分析 …… 176
7.1 引言 …… 176
7.2 语法图 …… 178
7.3 SCANER-2 …… 186
7.4 语法分析例 …… 197
7.5 错误局部化 …… 202
7.6 状态转换式的存放与实现方法 …… 206
第八章 ALGOL单词翻译(II) …… 212
8.1 引言 …… 212
8.2 标识符的局部化 …… 213
8.3 单元分配 …… 219
8.4 使用在前说明在后的处理 …… 223
8.5 标号的处理 …… 231
8.6 信息表 …… 232
8.7 SCANER-3 …… 239
8.8 单词翻译举例 …… 264
第九章 ALGOL句子翻译 …… 269
9.1 引言 …… 269
9.2 模型机 …… 277
9.3 表达式的翻译 …… 283
9.4 赋值语句的翻译 …… 317
9.5 转向语句的翻译 …… 324
9.6 条件语句的翻译 …… 330
9.7 循环语句的翻译 …… 335
9.8 过程语句的翻译 …… 348
9.9 过程说明的翻译 …… 379

第十章　数组的动态分配及其它…… 387
10.1　数组的动态分配…… 387
10.2　标准过程及整型量的处理…… 401
10.3　块地址方法…… 405
10.4　调试语句…… 408
参考文献…… 412

第一章 翻译程序

在计算机出现的初期，用户只能用机器指令编写程序．凡是用机器指令编过程序的人都知道，这项工作是极为繁重的，效率也很低．于是人们设计了各种高级语言．据一些资料介绍，目前已有数百种语言，其中用得最多的是 ALGOL60、FORTRAN 和 COBOL 等几种语言．ALGOL60 和 FORTRAN 语言主要用于科技计算，COBOL 则用于商业计算．用于表处理的 LISP 语言也出现得较早．以后又出现了更高级的大型语言，如 PL/I和ALGOL68；人们把这两种语言形象地称为“百货公司”语言，意思是说这种语言包罗万象，大到什么都有．后来，为了软件自动化，人们又设计了一些可用于描述编译等系统程序的高级语言，其代表之一便是 PASCAL 语言．使用高级语言必须有相应的翻译程序，否则只能是纸上谈兵．因此，在设计语言时，必须考虑到实现问题．如果语言中的一些成分是无法实现的，那末这些成分也就失去了意义．比如，在 ALGOL60 语言中，一个形式参数可以是赋值型数组标识符，但从翻译的角度来看，这是难于实现且效率极低的成分．因此，目前绝大部分编译程序都不允许用形式参数作为赋值型数组．又如表达式换名，这也是不易实现且效率极低的成分，因此很多编译程序也不允许用任意表达式作为换名实在参数．

不同语言的翻译方法都有自己的特点，但只要掌握了某一种语言的翻译方法，就不难掌握其它同类语言的翻译方法．本书将介绍 ALGOL 60 语言的翻译方法．本章先简单

讨论汇编程序、编译程序和解释程序的基本概念。

1.1 汇编语言和汇编程序

通常，我们把计算机的指令系统称为**机器语言**. 机器指令是一种代码，机器程序是指令的序列. 大型题目的机器语言程序可多达上万条指令.

机器语言有以下缺点：

1. 没有通用性. 不同机器的机器语言是不相同的，因此甲机上的程序可能无法用到乙机上，于是人们不得不熟悉乙机的机器语言，并且用它重写上千甚至上万条指令的程序. 这显然是一件相当麻烦的事情.

2. 容易出错. 指令是一种数码，因此容易记错，也容易写错，而且找错也很难.

3. 不容易修改. 因为程序中的指令直接用到内存地址，因此，从内存中移动一个数据或一条指令，会影响到很多条指令. 比如，想在机器语言程序中插入一条指令，就必须把该插入指令的下面程序部分统统往下移动一个单元，同时要修改有关转移指令的地址. 这显然是一件十分烦琐的事情.

4. 很费时间. 一条机器指令只能完成加减等很简单的操作，因此实际问题的机器程序经常有几百条，甚至可达上万条指令(在编指令时还要考虑很多细节). 要编这样一个程序有时需要几个月甚至更多的时间.

为了克服机器语言所固有的这些缺点，人们提出了**汇编语言**的概念. 汇编语言的基点是：

- 指令的地址部分直接写变量名或标号名，
- 操作码部分用符号来表示，
- 在指令前面可加标号.

这样，在汇编程序中就见不到数码形式的操作码了，也见不到机器的具体地址了，转移地址用的是标号。

例如，假定有函数

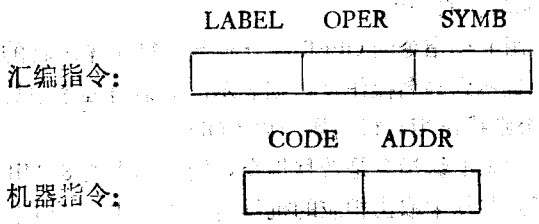
$$y = 2*a + b*(d + e)$$

则其汇编语言程序便为

CLA	2	$2 \Longrightarrow$ ACC
MUL	a	(ACC)$*a \Longrightarrow$ ACC
STO	h	(ACC)$\Longrightarrow h$
CLA	d	$d \Longrightarrow$ ACC
ADD	e	(ACC)$+e \Longrightarrow$ ACC
MUL	b	(ACC)$*b \Longrightarrow$ ACC
ADD	h	(ACC)$+h \Longrightarrow$ ACC
STO	y	(ACC)$\Longrightarrow y$

其中 ACC 表示累加器．由此可见，汇编语言比机器语言方便得多．但机器本身并不懂汇编语言，因此必须有一个翻译程序把汇编语言程序翻译成相应的机器语言程序．我们把这个翻译程序称为**汇编程序**．

汇编语言实际上是机器语言的符号化．两种语言的指令基本上一一对应，结构相似．因此汇编语言程序的翻译工作比较简单．假设汇编语言和机器语言指令的结构分别为



其中，

LABEL　表示汇编指令的标号部分

OPER　　表示汇编指令的操作符部分

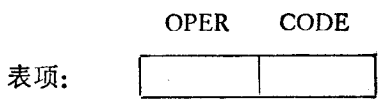
SYMB　　表示汇编指令中的变量符号部分

CODE　　表示机器指令的操作码部分

ADDR　　表示机器指令中的地址部分

那末，汇编程序的翻译有如下要点：

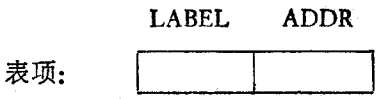
1. 事先造好汇编操作符和机器操作码之间的对应关系表，即



其中OPER是汇编操作符部分，CODE是机器操作码部分.

2. 每条汇编指令按下法处理.

(1) 如果汇编指令的 LABEL 部分为非空，则查标号表. 标号表结构为



其中 LABEL 是标号名部分，ADDR 是相应地址部分. 如果在标号表中已有同名标号，则表示标号名相重，于是产生错误，否则填写新的一项.

(2) 用汇编指令的 OPER 部分查操作符和操作码的对应表. 如果查不到同名操作符，则表示汇编语言程序中的操作符有错；否则找出相应机器操作码 CODE.

(3) 如果汇编指令是转移指令，则用其中的 SYMB 部分查标号表，并从中取出相应的机器地址 ADDR 部分（若查不到，怎么办？请读者思考.）；若不是转移指令，则用 SYMB 部分查单元分配表. 该表结构为

	SYMB	ADDR
表项：		

其中 SYMB 表示变量名部分，而 ADDR 表示相应分配单元地址部分. 如果在单元分配表中没有同名变量，则在该表中填写新的一项，否则不填表. 最后从单元分配表取出相应的 ADDR 部分.

(4) 用所取的CODE和ADDR构造机器指令，即

机器指令：	CODE	ADDR

较大的汇编语言还可包含宏指令和说明等部分，这时，翻译算法就更复杂了.

1.2 高级语言和编译程序

汇编语言虽然摆脱了对机器特别是单元地址的一些依赖性，但其指令仍与机器语言的指令一一对应，指令的结构仍依赖于机器语言指令的结构. 因此，不同机器的汇编语言彼此是不同的，所以汇编语言没有通用性，用起来仍然感到烦琐.

ALGOL、FORTRAN 和 COBOL 等**高级语言**完全摆脱了对机器的依赖，而且书写简便，容易修改. 为了在机器上使用高级语言，必须有一个翻译程序把高级语言程序翻译成相应的机器语言程序. 我们把这种翻译程序称为**编译程序**. 显然，只有高级语言而没有相应的编译程序，是不能真正进行计算的. 编译程序的加工对象称为**源程序**，而其加工结果则称为**结果程序**或**目标程序**. 编译程序的结果程序可以是下面三种形式之一.

1. 绝对指令. 它们放在固定单元中，编译后能直接执行这些指令.

2. 汇编语言程序. 在执行前必须由汇编程序将它们翻译成机器语言程序.

3. 模块型程序. 为了执行这个程序，它们必须与另外一些子程序联结起来，然后装配.

从时间的角度看，第一种形式是最有效的，第三种情形(即模块形式)则具有较多的灵活性. 但联结和装配本身是一种耗费时间的过程，在国内用得比较多的是第一种形式.

源程序的翻译和计算过程如图 1.1，1.2 和 1.3 所示.

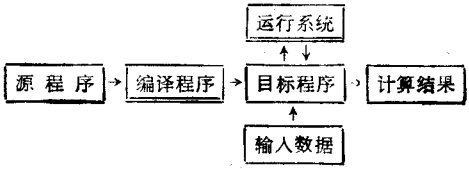


图 1.1 第一种形式

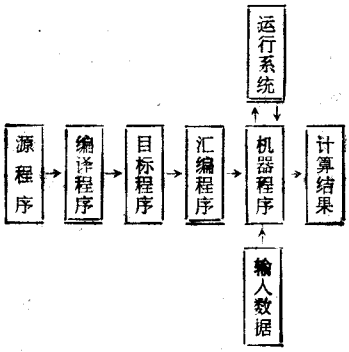


图 1.2 第二种形式

在目标程序运行时，还要一些子程序配合工作. 这种子程序称为**运行系统子程序**. 当目标指令为绝对指令时，编译和计算的步骤如下:

- 输入源程序，

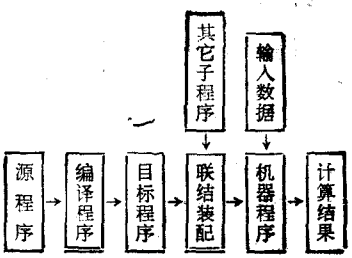


图 1.3 第三种形式

- 启动编译程序，
- 启动目标程序.

编译程序按其扫描次数可分为**一次扫描的**和**多次扫描的**. 从源程序开始通过一次扫描过程直接产生目标程序的称为一次扫描的编译程序，而通过多次扫描过程逐步生成目标程序的称为多次扫描编译程序. 多次扫描的优点是层次分明，算法简单，便于分工，便于优化. 特别是在小型机上搞比较大的编译系统时，必须采用多次扫描法. 多次扫描法的缺点是重复性工作多一些，因此会降低编译速度.

编译程序包括词法分析、语法分析、语义分析及代码生成等部分. 各部分的功能如下.

1. 词法分析程序的主要任务是：依次从源程序识别出单词，并确定其属性. 其属性通常用**语义字**或**机内符**来表示(ALGOL 60 语言的单词可分为标识符、常数和界限符三类).

2. 语法分析程序的主要任务是：在词法分析的基础上，识别出语法单位，同时检查语法错误(每当识别出语法单位时调用相应语义程序).

3. 语义分析程序的主要任务是：加工相应语法单位的有关信息，并检查语义的正确性，包括造表、单元分配及生成中间语言等各种工作.

4. 代码生成程序的主要任务是：在语义分析的基础上最后生成目标程序（汇编语言程序或机器语言程序或模块程序）.

有关编译的主要研究课题有以下几个：

- 如何提高目标程序的质量?
- 如何实现编译程序自动化?
- 如何保证编译程序的正确性?

为了提高目标程序的质量（主要指运行速度），人们提出了各种优化技术，其中包括局部优化和全局优化方法. 优化内容主要包括常表达式的节省、公共子表达式的节省、不变表达式提到循环外边、削减运算强度、下标变量优化等. 优化后目标程序的运行速度可提高好几倍，因此，目前的编译程序都做一定程度的优化工作. 优化搞得比较好的编译程序能产生较好的目标程序，其长度和速度可与手编程序相媲美.

实现编译自动化是人们长远而重要的研究课题. 形式语言的语法分析理论为编译程序的语法分析提供了各种技术，以至能够实现语法分析的自动化，即只要给出语言的形式文法，即可自动生成相应语法分析程序. 语法分析方法有递归子程序方法、优先矩阵方法、状态矩阵方法和 LR (k) 及 SLR (k) 方法等. 描述语法分析算法的典型语言是 Floyd-Evens 生成式语言，它在语法分析方面已有相当丰富的理论成果. 编译自动化的关键之一是语言的形式化问题. 语言包括语法和语义，因此语言的形式化包括语法形式化和语义形式化两方面的内容. 目前各种程序语言的语义都是用自然语言来描述的，因此还没有语义形式化，语义形式化一直是人们在程序理论中长期讨论的问题，至今还没得到令人满意的结果. 这个问题已不仅是为了使语言含义精确化，而且是与编译自动化以及自动查错等问题密切相关的. 语义形式化分两类，一类

是源语言的语义形式化，另一类是编译算法中语义部分的形式化．目前的自动化办法是用高级语言来描述编译算法，而该语言的编译程序是已经存在的．这个编译程序的目标程序不是解题程序，而是一个编译程序，因此称该编译程序为**编译程序的编译程序**，并记为 CC（Compiler-Compiler 的缩写）．ALGOL 60 和 FORTRAN 等语言主要用于描述数值计算过程，它们不太适用于描述编译算法．于是人们又研究出了适合于描述更广泛的系统软件算法的高级语言，通常把这种高级语言称为**系统设计语言**．系统设计语言的代表之一是 PASCAL 语言．最低级的自动化方法是用汇编语言描述编译算法．据了解，国外不少编译程序是用 PASCAL、BLISS 和扩充的 ALGOL 60 或 FORTRAN 语言写成的．

如何保证编译程序的正确性，这是另一个十分重要的问题．目前还没有证明或保证编译程序正确性的一种方法，只有初步的理论探讨．目前采用的主要办法是通过大量的例程序检查编译程序中的错误．这种方法只能检查错误，不能保证没有错误，因此在实用中发现一些错误是不奇怪的．当然，已交付使用的编译程序不应该有很多错误．编译程序的结构对调试来说是非常关键的，因此在设计编译程序时要把注意力放在结构上，而不应该把注意力放在节省几条指令上．总之，编译程序的正确性问题是人们十分关心的长远的研究课题．

1.3 解 释 程 序

前面介绍的编译程序，不管其目标程序采用哪种形式，最终都要在执行前把目标程序翻成机器语言程序，然后再执行之．**解释程序**是另一种翻译程序，它不产生目标程序，它把源程序作为输入接收并解释执行．解释程序的工作过程如图