李文忠 主编

软件技术基础

南京大学出

华东地区高校计算机基础教学研究会推荐教材

软件技术基础

李文忠　主编

南京大学出版社

1991·南京

内 容 提 要

本书是高校非计算机专业研究生、本科生的教学用书，亦可供开发计算机应用的工程技术人员参考。学习本书应具有一定的计算机硬件知识和程序设计语言基础。

本书分六章介绍程序设计的基本方法和技巧，数据结构的描述及其与程序设计的关系，操作系统的功能，数据库系统的建立和应用，计算机网络基础和软件工具等知识。有的章节配有必需的小结、习题和实验。

软件技术基础

李文忠 主编

*

南京大学出版社出版

（南京大学校内）

江苏省新华书店发行　　武进第三印刷厂印刷

*

开本 787×1092 1/16　　印张 23.75　字数 575 千

1991年3月第1版　1991年3月第1次印刷

印数 1—5000

ISBN 7-305-00718-8/TP·22

定价：9.20元

前　言

随着计算机科学的发展及其推广应用，我国高等院校的计算机教育事业得到了蓬勃发展，特别是高校的非计算机专业的计算机教学已由只学习一、二门程序设计语言扩展到学习微型计算机原理、单片机、数据库、操作系统、数据结构等多门计算机课程。这主要是因为计算机在各个学科领域中的推广应用，正在改变非计算机专业大学生的知识结构，要求他们掌握必需的软件基础知识，从而具有程序设计的基本能力，以便结合本专业的需要从事计算机的开发和应用。本书正是针对这一实际情况编写的。

本书作为华东地区高校计算机教学研究会的推荐教材，编写大纲经全国计算机基础教育研究会编委会审定，由东南大学李文忠主编，浙江大学俞瑞钊、华东师范大学王西靖、成都科技大学史济民、南开大学刘瑞挺等教授审稿。全书共分六章，第一章由上海第二工业大学倪祖彭编；第二章由杭州大学吴美朝编；第三章由江西工业大学沈树铭编；第四章由东南大学陈汉武编；第五章由浙江大学张德馨编；第六章由华东工学院陈次白编。东南大学胡显民、沈军参加了本书的校订工作。本书在编写过程中得到了有关高校计算机软件教师的大力支持和帮助，在此一并表示感谢。

由于本书内容繁多，涉及面广，同时没有前书可鉴，我们热切期望广大师生对本书提出宝贵意见。

编者

1989年11月

目　录

第一章　程序设计基础

第二章　数据结构

第三章　操作系统

第四章　数据库系统及应用

第五章　计算机网络基础

第六章　软件工具

第一章　程序设计基础

§1-1　程序设计的基本概念

在程序员看来，计算机是一台能可靠地、高速地执行指令序列的机器。一条指令对应于计算机执行的一项基本操作。为了使计算机处理某一个任务，就要编制一段对应的指令序列，此指令序列称之为程序。

一定型号的计算机，只能识别特定的指令系统，即机器指令。计算机的工作总是按机器指令所规定的次序一步步地处理输入数据，然后得到输出结果。其工作过程如图 1-1 所示。

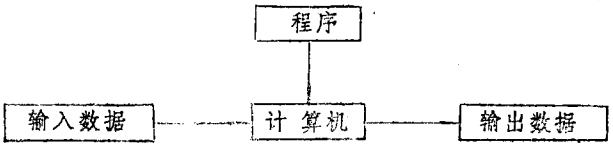

图 1-1　计算机工作过程

一、程序设计语言

计算机程序是用一定形式的语言编写的，一般可分为机器语言、汇编语言和高级语言。

（一）机器语言

机器语言是用二进制代码"0"和"1"的组合来表示操作码和操作数（即机器指令），是最原始的程序设计语言，完全面向计算机，也是计算机唯一能直接识别的程序设计语言。计算机指令系统本身就是这种用机器语言编写的程序。

（二）汇编语言

汇编语言是一种用助记符表示的面向机器的程序设计语言。这种语言比较直观、易懂、易记忆。对指令中的操作码和操作数也容易区分清楚。例如：要求计算机执行两个数32(20H)和18(12H)相加，则用Z80汇编语言编写的源程序如下：

```
LD    A, 20H    ; 立即数20H送累加器A
ADD   A, 12H    ; 立即数12H和累加器A相加
HALT            ; 暂停
```

显然，汇编语言与特定计算机的结构和其指令系统密切相关，汇编语言的符号操作码与机器语言的操作码一一对应，不同类型的计算机有不同的汇编语言。程序员必须先熟悉机器的硬件结构、指令系统以及寻址方式，然后方能进行程序设计。汇编语言虽比机器语言有进步，但仍比较繁琐、费时。但由于用汇编语言编写程序时，可以直接操作到机器内

部的寄存器，能把计算过程刻划得很具体，因而可把程序编制得很紧凑(既节约内存容量又提高程序运行速度)，在时间和空间上充分发挥计算机的效益，很适合于实时控制的场合。

(三) 高级语言

高级语言是一种面向用户的计算机通用语言。高级语言接近于自然语言，语法规则简单、清晰，易为各类专业人员掌握和使用。利用高级语言编写程序，不必了解计算机的内部逻辑，允许软件开发者和程序员独立于机器，集中精力去研究解题、算法和过程的描述。由于高级语言对解题过程的描述比较接近人们的习惯，因而易学易懂。据统计，程序员用高级语言编写程序要比用汇编语言编写快十倍左右。目前，已有200多种高级语言在使用，但广泛使用的不超过10种。这些语言可分为三类：

基础语言　这类语言是50年代后期60年代开发的。它们包括科学和商业上通用的语言，如BASIC，FORTRAN和COBOL等。

结构化语言　这类语言容纳了复杂的数据结构、子程序定义、分程序结构和逻辑构造。ALGOL，PASCAL和C语言是这类语言的代表。

专用语言　这类语言提供某些特殊功能的语句以适应特定的软件应用。如用于正文编辑系统模拟的APL语言，用于商业生成报表的RPG语言，用于人工智能研究的PROLOG语言等。

二、语言处理程序

人们往往用高级语言或汇编语言来编写计算机程序，然而计算机只能识别机器语言。因此用高级语言或汇编语言编写成的源程序必须通过编译、解释或汇编等方法将它们转化成机器语言表示的程序(称为目标程序)，才能在特定的机器上执行。语言处理程序提供这种手段，通过对源程序一系列的加工、优化和查错，最后产生出结构紧凑的目标程序。

(一) 汇编程序

1. 汇编程序的功能

汇编程序的主要功能是把汇编语言程序翻译成机器能识别的目标程序。如图1-2所示。

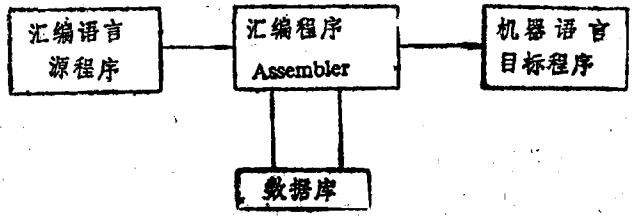

图 1-2　汇编过程意示图

具体地说，汇编程序必须完成几方面的任务：(1)产生机器指令，包括处理语句中的操作码，生成与之对立的机器代码；(2)处理语句中的标号和名字，并代之以具体的单元地址；(3)处理伪指令。这些任务可用一次扫描或多次扫描来完成。汇编程序从源程序的第一个字符读起，逐个向后读，一直读到源程序的末尾。一边读，一边完成一定的任务。这一全过程称为扫描。

2. 汇编程序的手编模拟

为了深入了解汇编程序的结构和汇编过程，我们先做一下汇编过程的手编模拟。

例 1-1　若要汇编的源程序如表1-1所示。

表 1-1　汇 编 的 源 程 序

语句号	源程序			
	标　号	操作码	操作数	注　解
1		ORG	100H	
2	START:	LD	HL, DATA	
3		XOR	A	; 清 0
4		LD	B, COUNT	
5	LOOP:	ADD	A, (HL)	
6		INC	HL	
7		DJNZ	LOOP	
8		LD	(RESULT), A	
9		NOP		
10		HALT		
11	DATA:	DB	1, 2, 3, 4, 5	
12	COUNT:	EQU	$-DATA	
13	RESULT:	DS	1	
14		END		

为此，我们设计两张表格：一张是标准编码单，见表 1-2；另一张为标准符号表，见表1-3。

表 1-2　标 准 编 码 单

程序名称＿＿＿＿＿＿＿＿＿＿＿＿　　页　号＿＿＿＿＿＿＿＿

标 准 编 码 单　　计算机＿＿＿＿＿＿＿＿

行　数	地　址	代　码	源程序语句

表 1-3　标 准 符 号 表

程序名称＿＿＿＿＿＿＿＿＿＿＿＿

符　号　表

符　号	值	定义行数	访问行数

首先我们把源程序抄在标准编码单中，如表1-4所示。要把源程序翻译成目标程序，还需要几种数据库：(1) 地址计数器(Location Counter)，用以追踪和确定指令的地址；(2) 机器操作码表MOT(Machine Operation Table)，用以确定指令的长度，并把助记符转换成机器码；(3) 符号表(Symbol Table)，在第一次扫描时，把各个标号的值列入表格，便于第二次扫描时，把标号的值代入到相应的符号地址中去；(4)伪指令操

表 1-4 程序的标准编码单

程序名称 _____ 页 号 _____

标准编码单 计算机 _____

行数	地址	代码			源程序		
1						ORG	100H
2	100H	21	0F	01	START:	LD	HL, DATA
3	103H	AF				XOR	A
4	104H	06	05			LD	B, COUNT
5	106H	86			LOOP:	ADD	A, (HL)
6	107H	23				INC	HL
7	108H	10	FC			DJNZ	LOOP
8	10AH	32	14	01		LD	(RESLUT), A
9	10DH	00				NOP	
10	10EH	76				HALT	
11	10FH				DATA:	DB	1, 2, 3, 4, 5
12		0005H			COUNT:	EQU	$-DATA
13	114H	1			RESULT:	DS	1
14						END	

作表POT(Pseudo Operation Table)，便于在汇编中遇到伪指令时在该表中寻找相应操作指令；(5) 输入的汇编语言源程序；(6) 输出的机器码目标程序。

同时还需要一些简单算法和操作：(1) 追踪地址计数器；(2) 逐条地读源程序；(3) 查机器操作码表MOT或伪指令表POT，以得到相应的操作码；(4) 计算地址；(5) 建立符号表；(6) 检查源程序的语法，寻找是否有语法错误；(7) 逐条地写出目标程序。

针对上述源程序：

第一步，ORG 100H 指令，指出起始单元地址为100H，即START的值为100H；查MOT表可知：第二条指令是立即数送HL寄存器，它是三字节指令；操作码为21，第二、三字节是由二个立即数组成的地址。但在指令中它是用符号地址DATA代替，至此还不知道DATA的值，故留出二个空格(用符号␣表示)。同时在符号表中填写DATA。从第二条指令的字节数，即能计算出第三条指令的首地址(100 + 3 = 103)，填入相应的栏内。再查MOT表，找到第三条的操作码代入，……按同样的方法一直进行下去。但在操作数部，第四行中的立即数COUNT；第八行中的符号地址RESULT都还未知，故先用空格代替。同时在符号表中填写相应的符号。在第七行中的相对转移指令的偏移量需进行计算才能得到(偏移量 e = LOOP − PC = 106 − 10A = −4，其补码为FC)。当汇编到第11行，它是一条伪指令，由上一条指令可知，DATA的首地址为010FH，伪操作DB的功能从DATA单元开始，定义为五个数：1，2，3，4，5，并依次存放在010FH，0110H，0111 H，0112H以及0113H 地址单元中，第12条伪指令定义标号 COUNT 本身不占用内存单元，但 EQU 右边的表达式$-DATA需要计算。$是程序计数器的当前值，本例中为0114H，DATA 为010FH，所以 COUNT = 0114H − 010FH = 0005H。第13行伪指令定义存贮空间，为

RESULT留取一个存贮单元，按上式计算，其地址为0114H。

这样，程序中的所有标号都确定下来了，将其回填到上面的空格里，并填补到符号表中。如第二条，DATA用010F代入。要注意，指令中的立即数，低位字节在前，高位字节在后。至此手编过程就完成了。

由此例可见，在源程序中采用了符号地址之后，第一次汇编扫描时，往往指令中的符号地址值还不能确定下来。只有在第一次扫描完了之后，才能把程序中的所有标号的值确定下来。第二次再进行扫描，把确定的符号地址值代到相应的标号中去，即需要采用二次扫描的办法来完成整个源程序的汇编。

3．两次扫描的汇编程序

同手编模拟，为了在汇编语言中提前使用符号地址，通用的是二次扫描的汇编程序。它通过对源程序从头到尾二次扫描来完成汇编过程。

在第一次扫描过程中，要完成的任务是：(1)确定每条指令的长度(由查MOT表得到)；(2)追踪地址计数器LC；(3)建立符号表(由LC值确定每一个标号的值，把它们引入到一个表格文件中)；(4)处理伪指令操作码，如EQU，DB，DS等。

第二次扫描过程建立在第一次的基础上，最后产生机器语言。第二次扫描具体完成的任务是：(1)查符号的值；(2)产生机器指令；(3)确定伪指令DB定义的数据；(4)处理伪指令操作码DB。

上述过程可用图1-3形象地表示(圆柱体表示磁盘文件)。其中：

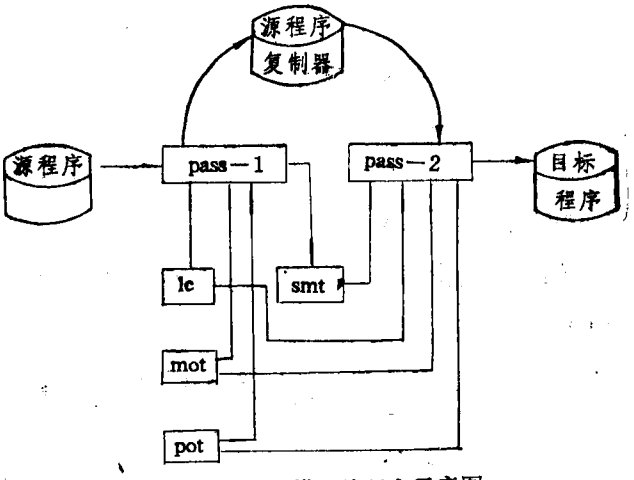

图 1-3　两次扫描汇编程序示意图

(1) MOT是机器操作码表，其形式如表1-5所示。此表在第一次扫描(PASS-1)中，用以确定指令的长度，从而能得到LC；在第二次扫描(PASS-2)时，用以产生机器指令。

表 1-5　MOT

助记符操作码	机器操作码	指令字节数
LD　BC, nn	01	3
LD　DE, nn	11	3
…	…	…
LD　BC, (nn)	ED　4B	4
…	…	…

(2) POT是伪指令操作表，其格式如表1-6所示。表中的右面部分是汇编程序的一些标号，即相应的伪指令处理程序的入口地址。在实际处理时，以绝对地址的形式出现。在PASS-1以及PASS-2中都要用到POT表。

表 1-6 POT

伪指令码	处理伪操作的程序入口地址
EQU	PI EQU (EQU 处理程序入口地址)
DB	PI DB (DB 处理程序入口地址)
DS	PI DS (DS 处理程序入口地址)
ORG	PI ORG (ORG 处理程序入口地址)
...	..... (.....)

(3) SYMT符号表，其格式如表1-7所示。

表 1-7 SYMT

符 号	值	注 释
START	0	"R"
DATA	F	"R"
COUNT	5	"A"
...	...	...

("R代表地址为浮动，"A"代表地址不随程序的起始地址而改变)

SYMT表由PASS-1建立，在PASS-2中用以确定符号数值。如上例，两次汇编列表如下：

表 1-8 两次汇编程序

	源 程 序			PASS-1		PASS-2	
				LC	L	LC	操作码
1		ORG	100H				
2	START,	LD	HL, DATA	100H	3	100H	21 0F 01
3		XOR	A	103	1	103	AF
4		LD	B, COUNT	104	2	104	0605
5	LOOP,	ADD	A, (HL)	106	1	106	86
6		INC	HL	107	1	107	23
7		DJNZ	LOOP	108	2	108	10FC
8		LD	(RESULT), A	10A	3	10A	321401
9		NOP		10D	1	10D	00
10		HALT		10E	1	10E	76
11	DATA,	DB	1, 2, 3, 4, 5	10E	5	10E	01, 02, 03, 04, 05
12	COUNT,	EQU	$-DATA	114			
13	RESULT,	DS	1	114	1		
14		END					

表 1-9 SYMT

符　号	值	注　释
START	0	R
LOOP	6	R
DATA	F	R
COUNT	5	A
RESULT		

（二）编译程序

编译程序是一个十分复杂的加工处理程序。计算机不能直接用高级语言（如FORTRAN,PASCAL，COBOL 等）编写的源程序运行。首先要把源程序通过编译程序翻译成语义上等价，且可在计算机上执行的目标程序；第二步才能在目标程序控制下，对输入数据进行处理，得到所需要的输出数据，其过程如图1-4所示。编译程序与目标程序密切联系，所以编译程序决定于不同类型机器，是有针对性的。

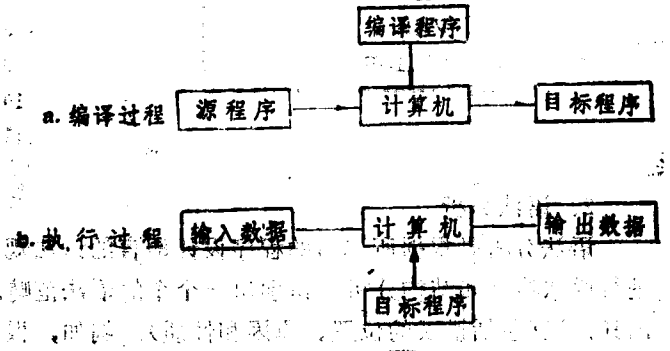

图 1-4　编译和执行过程

一般将编译过程划分成五个处理阶段，即词法分析和造表；语法分析；语义分析(中间代码生成)；代码优化；目标代码生成，如图1-5所示。

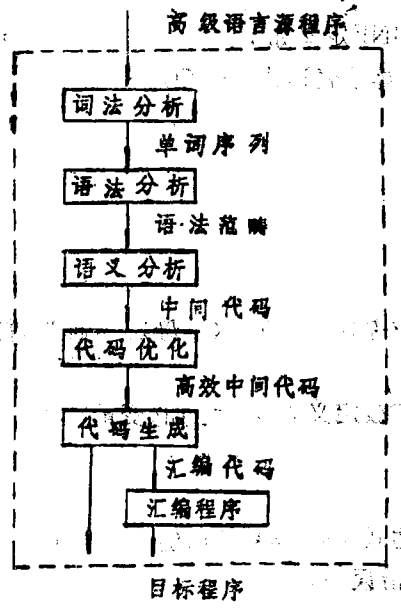

图 1-5　一般编译过程

1. 词法分析

词法分析阶段，对源程序进行自左向右扫描，对输入的源程序的每一个符号进行辨认，识别出哪些是运算符，哪些是常数，哪些是标识符或拼写的定义符等。即在源程序中分解出一个个有独立意义的语法单位(称之为单词)，同时还要识别出与其相关的属性，然后立即将其改造成为长度统一的最小语法单位。同时查填各单词表格，并作一些语法检查，为以后语法分析提供方便。

具体的处理过程是：在扫描过程中，一旦识别出关键字、标识符、常数和界符中之一者，即以单词形式(剔除其中多余的空格符)输出。然后再继续扫描以形成下一个单词，直至整个源程序扫描完毕，形成相应的单词串。

经过词法分析之后的各类单词均有相同的结构和长度，每一个单词，均由两部分组成(t, i)。其中t表示单词的类别，共分四类：K(关键词)类，I(标识符)类，C(常数)类和P

(界符)类。每类对应一种表格，每种表格存放对应类的单词；i为指向该类表格中的一个特定项目的指针。因此(t，i)唯一地确定了一个单词。K，P两种表格的内容取决于所选用语言的子集(见表1-10和1-11)，而I和C两种表格是根据临时输入的字符串形成的。

表 1-10 K表

1	BEGIN
2	DO
3	ELSE
4	END
5	IF
6	PROCEDURE
7	THEN
8	VAR
9	WHILE

表 1-11 P表

1	<	=
2	<	>
3	<	
4	[	
5	*	
6	:	=
7	+	
8	(	
9	]	
10	)	
11	,	

2. 语法分析

语法分析是根据所选用的程序设计语言的语法规则，对经过词法分析所产生的单词串进行语法检查和结构分析，识别出一个个的语法范畴。凡不符合语法者，尽可能确切地指出其错误(包括错误的位置、原因和性质)。例如，根据形式文法的规定，对简单变量的类型说明：首先写出类型说明符(如TYPE)然后紧接着写出同类型变量的名词，两个变量之间用逗号隔开，最后一个变量后面用冒号，最后写上变量类型。如在语法分析中发现一个简单变量不符合这种规范，就可认为它出错。

又如，分析一个算术表达式时，规定分析对象的BNF定义如下：

〈算术表达式〉::=〈项〉|〈算术表达式〉+〈项〉|〈算术表达式〉-〈项〉

〈项〉::=〈因式〉|〈项〉*〈因式〉|〈项〉/〈因式〉

〈因式〉::=〈变量〉|〈算术表达式〉

〈变量〉::=〈字母〉

〈字母〉::= A|B|C|…X|Y|Z

在BNF定义中，我们使用了三个符号，解释如下：

(1) 尖括号“〈 〉”，其内是语法的结构成分。如〈基本符号〉、〈字母〉、〈标识符〉、〈数组〉等，表示一种成分，不能写入到源程序中。

(2) 符号“::=”是BNF定义符，它表示的意思是“被定义为”，例如〈变量〉::=〈字母〉，即变量被定义为字母。

(3) 符号“|”表示“或”的意思。

根据上述定义，进行语法分析时，认定，算术表达式A+B*C是正确的；(((C+H)*I)/J)也是正确的；而(A+G是错误的；A|+B也是错误的。

3. 语义分析

语义分析阶段，是根据语法分析所识别出来的语法范畴，在语义分析的基础上，产生中间代码，以便优化目标程序。在编译过程中，词法分析、语法分析和语义分析三个阶

段，是根据形式文法和源程序进行分析的阶段，与具体机器无关，故可称为不依赖机器的过程。但在程序优化和目标程序产生阶段，必须依据所使用机器的指令系统，是依赖机器的过程。

以中间代码表示的内部程序，是介于源程序和机器语言程序之间的程序，它接近于机器语言，便于机械地转换成机器语言程序。中间语言形式的内部程序，可以用波兰表示法、三元组法或四元组法表示。现用例子分别作一简单说明：

(1) 波兰表示法

波兰表示法是由一个波兰人在1929年提出的一种表示算术公式的方法。一般也称后缀法。就是将算术运算符放在运算对象的后面。例如，

算术表达式A+B-C和A*(B+C)-(F-D)的波兰表示法，分别为：AB+C-和ABC+*FD--。

扫描用波兰表示法书写的算式，都是从左到右顺序扫描。当碰到运算符时，就把该算符左边的两个符号所对应的数值进行运算，并将运算结果作一些适当处理或保存，以便进行后面的运算。如表达式ABC+*FD--，从运算对象A开始象右扫描。

当碰到"+"时，就把"+"左边最邻近的二个运算对象B、C相加，产生(B+C)，再向右扫，碰到"*"时，就把左边最邻近的二个对象A和(B+C)相乘，得到A*(B+C)，并把它作为一个运算对象；再扫到"-"，即把它左边的二个运算对象F和D相减，得到(F-D)；再扫到"-"时，即把A*(B+C)和(F-D)这两个运算对象相减，从而得到最后的结果，A*(B+C)-(F-D)。

由此可见，用波兰表示法写出的算术表达式，可不考虑括号和运算优先关系，可机械地对该表达式进行处理。波兰表示法不但适用于算术表达式，也适用于其他语句。如赋值语句：A:=E可写成：AE:=

(2) 四元组法

四元组的形式包括四个项：运算符、运算对象1、运算对象2和运算结果。运算对象1和运算对象2分别指出了参加运算的二个对象所存放的内存地址，运算符指出了对它们应实施的操作，而运算结果表示运算结果存放的地址。如一个赋值语句

Y:=A+(A*B-(C+D/T))

用四元组形式写成内部程序，如表1-12所示。

表 1-12 四元组形式的内部程序

运算符	运算对象 1	运算对象 2	运算结果
*	A	B	T00
/	D	T	T01
+	C	T01	T02
-	T00	T02	T03
+	A	T03	T04
:=	T04		Y

表的最后一项是 :=，T_{04}，，Y，这里只有一个运算符和两个地址。运算对象2的地址是空的，这是因为运算符":="只要求一个运算对象(我们把这种运算称作一目运算。

相对应，前面几种是双目运算）。

（3）三元组法

三元组表示形式和四元组极为类似，可用文字表示为：

（组号）运算符，运算对象1，运算对象2

例如，计算一个圆周差的运算表达式Y:＝2＊3.1416＊(R1－R2)

用三元组形式写成的内部程序，如表1-13所示。

表 1-13　三元组形式的内部程序

（组号）运算符	运算对象 1	运算对象 2
(1)　＊	2	3.1416
(2)　－	R1	R2
(3)　＊	(1)	(2)
(4)　:＝	(3)	Y

在三元组中，运算对象如果是(1)，它表示第一个三元组的运算结果，是(2)，它表示第二个三元组的运算结果，其余类推。

从上面可见，把用波兰表示法，三元组法或四元组法表示的内部程序译成目标程序是比较容易的。

4．代码优化

代码优化阶段，对语义分析所产生的中间代码进行改造，以获得语义等价但效率更高的中间代码，从而节约时间和空间。一般将优化分为四类，一类是依赖于机器的优化，这种优化与具体计算机密切相关，大多是在代码生成(目标程序产生)阶段进行的。如寄存器、变址器和内存的最佳分配，发挥多处理机、多运算部件、先行控制、高速缓存等的作用。另一类优化是独立于机器的优化，根据所涉及的范围，又可分为局部优化，循环优化和全局优化。如节省共用的子表达式：当一个语句或一个表达式中，某一个子表达式多次出现时，只需要对该子表达式在第一次出现时进行计算，此后用到该子表达式时，只要应用前面已经计算出的结果即可，不必再作重新计算。又如对常数的处理：当一些运算对象的值在编译阶段已经是已知数时，就可在编译过程中进行运算，不必等到执行目标程序时才进行计算。这样不但可以节省存贮单元，而且缩短了执行时间。又如在循环体内的有些计算，其运算对象是固定不变的，则可以把这部分提到循环体之外。假如在循环体内有一个不受循环变量影响的乘法运算，而该循环要进行500次，把它提到循环体外之后，就减少了499次该项运算，大大加快了执行的速度。

实际上，不同优化对象的优化时机是各不相同的。有的可在源程序一级进行，如强度减弱；有的可在中间代码一级进行，如删除公共子表达式；还有的可在目标代码一级进行，如最佳寄存器分配。换句话说，优化工作往往分散到各遍扫描中去处理。

5．代码生成

目标代码生成阶段，是根据经过优化的中间代码和表格信息，进行存贮器分配和代码选择，最后形成可在计算机上执行的目标程序的阶段。如果在目标代码生成阶段，产生的代码是汇编语言程序，那么还应经过汇编程序才能最后产生目标程序(如图1-5所示)。

（三）解释程序

为了使高级语言程序能在机器上运行，除了编译程序之外还有一种办法也能完成源程序的翻译工作，这就是解释程序。解释程序的工作方式是：逐条地取出源程序中的语句，然后对它进行解释并翻译成机器语言，并立即执行。编译程序不同于解释程序，编译程序首先要将整个源程序翻译成目标程序，再通过执行目标程序间接地实行对源程序的执行。而解释程序却不需要产生完整的目标程序就能直接执行源程序本身。一般说来，解释执行的处理方式比整个地将源程序翻译成目标程序然后整体执行的方式要化费更多的机器执行时间，但却能节约内存空间，更适用于分时系统。因为在解释执行时，万一指令出现语法错误，解释程序会自动发出诊断信息，自动停止程序执行，等待程序员来纠错。程序员可以观察程序的效果，任何错误都可以在终端上立即得到纠正并让程序继续执行。而在等待阶段，分时系统可以继续执行其他用户程序。所以，这种方式对分时系统更为适用。

显然，解释程序的缺点是：程序循环(或转移)到前面某一条指令时，例如读入一组数值(用READ语句)，这条指令每次必须重新翻译。这就是说，读入1000个值的程序，要翻译1000遍指令。正因为如此，一般很少单独使用解释程序。

为了提高解释程序的效率，通常把解释程序分为翻译和执行两个阶段。

图1-6表示解释程序的执行过程。解释程序的第一阶段(即翻译阶段)是词法分析和语法分析。其任务为：(1)对源程序进行词法检查和部分语法检查；(2)把外部形式的源程序翻译成内部形式的源程序；(3)建立一些表格，如标号表，标识符表等，为执行阶段提供方便。

解释程序的第二个阶段(即执行阶段)由图1-6中的解释执行程序来完成。该段程序的任务是：(1)运用翻译阶段提供的一些表格，对内部源程序进行解释并执行；(2)在解释执行过程中，对内部源程序进行全部语法检查。

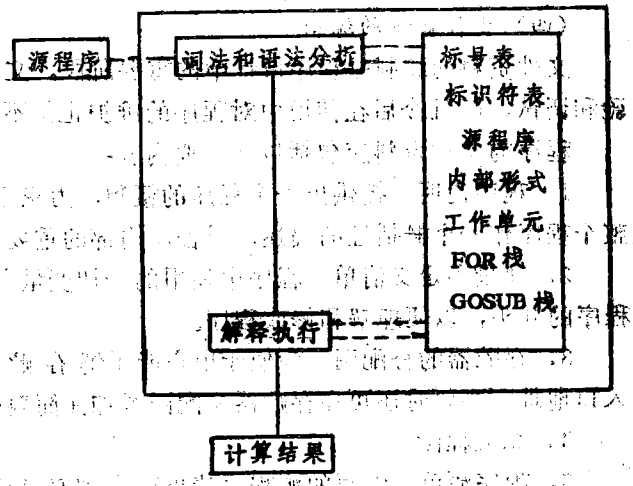

图 1-6 解释程序的执行过程图

三、程序设计的步骤

程序设计往往被人们仅理解为一种编码活动。对于一个简单的数值计算的程序设计，只要熟悉程序设计语言，经过构思并借助于简单的流程图就能实现。然而对于复杂的程序，例如要解决一个较大的过程控制问题，有几K甚至几十K的程序，编写起来就不那么容易了。在设计这种程序时，既要涉及计算机的系统组成，又要涉及计算方法，接口设备以及控制对象等。因此不仅要有软件的知识，还要熟悉硬件组成。概括起来，程序设计一般要按下述五个步骤进行。

（一）分析课题

首先对课题要解决的问题进行全面、详细、慎重的分析，并列出要解决问题的清单。