第 1 章　编译程序概论

本章学习目标

编译程序是高级语言的支撑基础，是计算机系统中重要的系统软件之一。本章主要内容:

- 编译程序的功能
- 编译程序体系结构
- 编译程序工作过程
- 编译程序设计

1.1　程序设计语言

现代计算机系统之所以功能强大，除了具有先进的硬件功能之外，当归功于计算机软件系统。软件系统的设计离不开程序设计语言。程序设计语言分成两大类：高级语言和低级语言。低级语言又包括机器语言和汇编语言，主要是面向机器的。高级语言则是面向应用的，分成很多种，如 FORTRAN、Pascal、C、ADA、Java 等。

机器语言本身是由 0 和 1 组成的，符合计算机的硬件特性，因此能够直接执行。但用机器语言编写程序很不方便且容易出错，因此就用助记符代替机器语言，这就是汇编语言。汇编语言比较直观，但不能直接运行，必须翻译成机器语言才能执行，翻译过程是由汇编程序实现的。

汇编语言比机器语言在可读性方面有了进步，但是其依赖具体机器的特性无法改变，给程序设计语言增添了难度。随着计算机应用的不断普及，出现了更加接近人类自然语言的功能更强、抽象级别更高的面向应用的各种高级语言。用高级语言编写的程序可以在不同种类的计算机上运行且不出错，这是汇编语言难以做到的。

高级语言不能直接在机器上运行，它不是面向机器而是面向应用的，因此，要想让高级语言运行必须有编译程序。编译程序就是这样的一种程序，它能将高级语言编写的源程序转换成与之在逻辑上等价的低级语言形式的目标程序。

高级语言程序的执行通常分为两个阶段，即编译阶段和运行阶段，如图 1-1 所示。编译阶段将源程序变换成目标程序；运行阶段则由所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，运行后输出计算结果。如果目标程序是汇编语言的形式，则需要在编译阶段和运行阶段之间加一个汇编阶段。

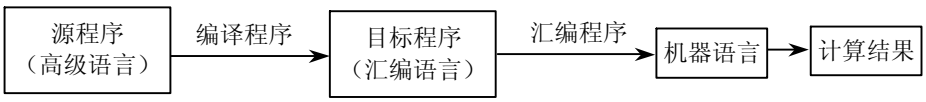

图 1-1　源程序的编译、汇编和运行阶段

高级语言编写的程序除了可以通过编译方式外，还可以通过解释程序执行。所谓解释程序是一种语言翻译程序，读入一条语句，解释一条语句，执行一条语句，边读入边执行。解释程序与编译程序的主要区别是：编译程序将源程序翻译成目标程序后再执行目标程序，而解释程序是逐条读出源程序中的语句并执行，即在解释程序的执行过程中并不产生目标程序。例如BASIC 语言和 PROLOG 语言就是解释执行的典型语言。采用编译方式运行的程序运行效率较高，采用解释方式运行的程序便于调试和改错，为了兼顾这两个方面，有时将两种方式结合使用。高级语言的解释方式如图 1-2 所示。

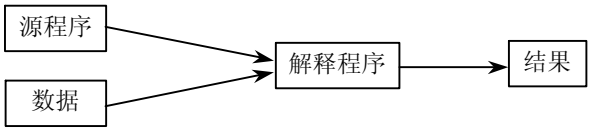

图 1-2　高级语言的解释方式

本书主要介绍编译技术及其实现，但同样的技术也适合于解释程序。掌握了编译技术，就不难设计和实现解释程序，而汇编语言的设计与实现和编译程序相比则更为简单。

1.2　编译程序的编译过程和结构

编译程序的功能是把高级语言源程序翻译成等价的低级语言目标程序。源程序是由一些基本符号构成的，我们在运行这个程序时，先编译，若某处有错误，就报错，无错误就运行。编译程序在编译时，先将程序中的单词一个个分离出来，登记在一个表中，这个过程叫做词法分析。接着检查语句格式，叫做语法分析。然后检查类型是否一致，计算表达式的值叫语义分析。这些功能都是由编译程序的相应程序完成的。一般来说，整个编译过程可以划分成五个阶段：词法分析阶段、语法分析阶段、语义分析和中间代码生成阶段、中间代码的优化和目标代码的生成。

1．词法分析阶段

词法分析是编译过程的基础，其任务是扫描源程序，根据语言的词法规则，分解和识别出每个单词，并把单词翻译成相应的机内表示。当然，词法分析在识别单词的过程中，同时也做了词法检查。

在高级语言中，所谓单词，就是指逻辑上紧密相连的一组字符，这些字符具有“集体”含义。单词是语言中最小的语义单位，如语言中的关键字、标识符、运算符和界限符等。词法分析的依据是词的构造。单词的构造规则在高级语言中有明确的规定，比如哪些为保留字、变量如何定义、常量如何构造、分界符有哪些等。

例如，用 C 语言编写的程序段如下：

```
main()
{
    float x=2,y=3,s;
        s=x+y*5;
}
```

识别出的单词序列如表 1-1 所示。

表 1-1　词法分析结果

类型名	单词	类型名	单词
保留字	main	左括号	(
右括号	)	花括号	{
保留字	float	标识符	x
等号	=	常量	2
逗号	,	标识符	y
等号	=	常量	3
逗号	,	标识符	s
分号	;	标识符	s
等号	=	标识符	x
运算符	+	标识符	y
运算符	*	常量	5
分号	;	花括号	}

2．语法分析

语法分析是在词法分析的基础上进行的，是编译过程的第二个阶段。语法分析的任务是根据语言的语法规则，把单词符号串分解成各类语法单位，如表达式、语句等。通过语法分析，可以确定整个输入符号串是否构成一个正确的程序。例如上例的 s=x+y*5，倘若写成 s+x=y*5 就是错误的，不符合语法错误，必然报错。编译程序的语法检查在程序的调试过程中就能看到。例如在 C 语言中，当我们编辑好一个源程序后，我们必须对程序进行调试，其中就包括语法检查。语法检查的依据是语言的语法规则，包括高级语言源程序的结构，表达式的结构，顺序结构、分支、循环以及数组的结构等。关于语言语法规则的描述，在后面的章节再做详细的介绍。

3．语义分析和中间代码的生成

语义分析的任务是对各种不同语句进行翻译，包括两方面的工作：一是对语法范畴进行语义检查，如变量是否定义、类型是否正确等；二是在语义检查正确的情况下，进行中间代码的翻译。中间代码是介于高级语言语句和低级语言语句之间的一种独立于具体硬件的记号系统，它既有一定程度的抽象，又和低级语言十分接近，因此，转换成目标代码很容易。中间代码的表示形式有很多种，常见的有四元式、三元式、间接三元式和逆波兰式。其中四元式的形式为（运算符，运算对象 1，运算对象 2，结果），例如上例中的语句 s=x+y*5;，翻译成四元式形式的中间代码为：

(inttofloat ,5, -,T_1)

(*,y,5,T_1,T_2)

(+,x,T_2,T_3)

(=, T_3 ,- , s)

其中 T_1、T_2、T_3 为临时工作变量。

4. 中间代码优化

中间代码优化是调整和改变中间代码中某些操作次序，最终产生更加高效的目标代码。优化的主要手段有删除冗余运算、删除无用赋值、合并已知量、循环优化等。语句 s=x+y*5 的中间代码可以优化为：

(*,y,5.0,T_1)

(+,x,T_1,s)

很显然，中间代码只有两条，占据的内存空间比四条要少。

5. 目标代码的生成

这一阶段的任务是将前一阶段产生的中间代码变换成特定机器上的机器语言或汇编语言程序，实现机器的最终翻译。最后阶段的工作因为目标语言的关系而十分依赖机器的硬件系统，即如何充分利用机器现存的寄存器合理地选择指令，生成尽可能短的目标代码，这与机器的硬件有关。例如机器有两个寄存器，则上述生成的汇编代码为：

（1）MOVF y,R_2

（2）MULF #5.0,R_2

（3）MOVF x,R_1

（4）ADDF R_1,R_2

（5）STR R_2,s

含义：

（1）将 y 送到寄存器 R_2 中。

（2）将 R_2 中的数据乘上 5，结果存储在 R_2 中。

（3）将 x 的数据送到寄存器 R_1 中。

（4）将寄存器 R_1 和 R_2 相加，结果存储在 R_2 中。

（5）将相加后的结果送到存储单元 s 中。

根据编译程序的工作过程，可以看出编译程序的结构是按照五个阶段的任务分模块设计的。

在编译过程中，源程序的各种信息需要保留在各种表格中，在编译的各个阶段都需要查找或更新有关的表格。编译程序的绝大部分时间都用在造表、查表和更新表格的事务上，这个任务由编译程序中的表格编译处理程序来完成。

在编译过程中，不管在词法分析、语法分析还是语义分析过程中都会发现错误，编译程序中需要有出错处理程序发现源程序中可能出现的错误，并把错误报告给用户，指出错误的性质和发生错误的位置。因此编译程序的结构示意如图 1-3 所示。

编译过程可以划分成五个阶段，这种划分是编译程序的逻辑组织形式。实际上，编译过程往往分成前端和后端。前端包括词法分析、语法分析、语义分析、中间代码生成和中间代码优化，主要依赖于源程序；后端主要包括目标代码生成，依赖于计算机硬件系统和机器指令系统。这种组织方式，便于编译程序的移植，若要将编译程序移植到不同类型的机器，只需要修改编译程序的后端即可。

编译程序还采用“分遍”的形式，即编译过程可以由一遍或多遍编译来完成。对于源程序或中间代码，从头到尾扫描一次并完成所规定的工作称为一遍。在一遍中，可以完成一个或相连几个逻辑步骤的工作。例如可以把词法分析作为第一遍；语法分析和语义分析作为第二遍；

代码优化和存储分配作为第三遍；代码生成作为第四遍；从而构成一个四遍扫描的编译程序。

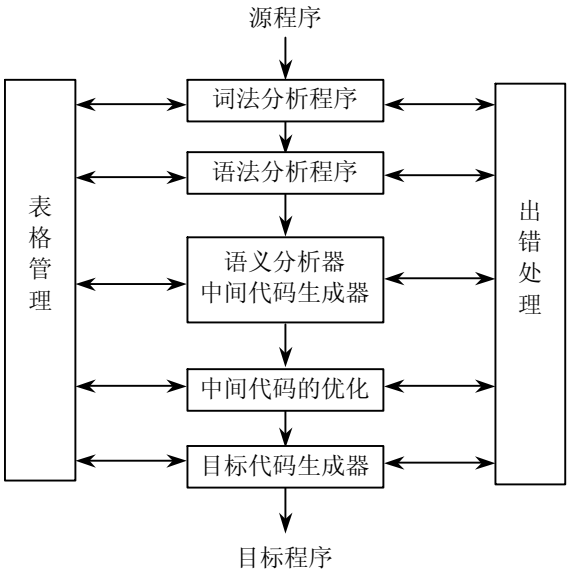

图 1-3　编译程序结构示意图

一个编译程序是否需要分遍，如何分遍，通常根据下面的因素决定：宿主机的存储容量的大小；编译程序功能的强弱；源语言的繁简；目标程序优化的程度；设计和实现编译程序时使用工具的先进程度以及参加人员的多少和素质等。一般，当源程序较繁、编译程序功能很强、目标程序优化程度较高且宿主机的存储容量较小时，采用多遍扫描方式。分遍的好处是：多遍的功能独立、单纯；相互联系简单；逻辑结构清晰；优化准备工作充分并利于多人分工合作。不足之处是不可避免地做些重复性的工作，且多遍之间有一定的交接工作，因而增加了编译程序的长度和编译时间。

1.3　编译程序的设计技术

由于计算机语言功能的完善、硬件结构的发展、环境界面的要求等对编译程序提出了更高、更多的要求，因而构造一个编译系统并非易事。虽然编译理论和编译技术的不断发展已使编译程序的生产周期不断缩短，但要完成一个编译程序仍需相当长的时间。如何高效、高质量地完成一个编译程序一直是计算机系统设计人员追求的目标。

编译程序的任务是把源程序翻译成某台计算机上的目标程序。因此编程人员首先要熟悉源程序语言，对源程序语言的语法和语义要准确无误地理解，同时要确定开发方案。同时还要选择合适的语言编写编译程序，目前一般采用 Pascal、C、ADA 等高级语言编写，减少了开发的工作量，缩短了开发周期。最后，开发人员还要熟悉目标机的特点，产生质量较高的目标代码。

编译程序的开发通常采用自编译、交叉编译、自展、移植和自动化技术。

1. 高级语言的自编译技术

用某种高级语言书写自己的编译程序称为自编译。例如，假设在某机器上已有一个 Pascal

语言已经运行，然后用已有的 Pascal 语言编写出一个功能更强的 Pascal 语言编译程序，然后借助于原有的 Pascal 编译程序对新编写的 Pascal 编译程序进行编译，从而在编译后即得到一个能在机器上运行的功能更强的 Pascal 编译程序。

2. 交叉编译

交叉编译是指用 A 机器上的编译程序来产生可在 B 机器上运行的目标代码。例如，若 A 机器上已有 C 程序可以运行，则可以用 A 机器上的 C 程序书写一个编译程序，它的源程序是 C 语言程序，而产生的目标程序则是基于 B 机器的，即能够在 B 机器上执行的低级语言程序。

以上两种方法是假定已有了一个系统程序设计语言可以使用，若没有任何程序设计语言，则可采用自展或移植的办法来开发编译程序。

3. 编译程序的自展技术

自展的方法是：首先确定一个非常简单的核心语言 L_0，然后用机器语言或汇编语言书写它的编译程序 T_0；再把语言 L_0 扩充到 L_1，此时有 L_0 属于 L_1，并用 L_0 编写出 L_1 的编译程序 T_1，然后再把语言 L_1 扩充为 L_2，此时，L_1 属于 L_2，并用 L_1 编写 L_2 的编译程序 T_2……这样不断扩展下去，直到完成所要求的编译程序为止。自展技术的使用如图 1-4 所示。

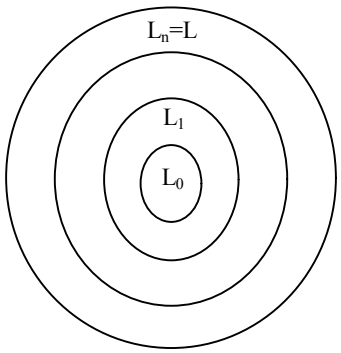

图 1-4 编译系统的自展过程

4. 编译程序的移植技术

编译程序可以通过移植得到，即可以将一个机器（宿主机）上的一个具有自编译性的高级语言编译程序移植到另一个机器（目标机）上。

5. 编译程序的自动化

在编译程序的自动化开发过程中，开发较早的是词法分析程序生成器和语法分析程序生成器，使用的工具是在 UNIX 操作系统下的软件工具 LEX 和 YACC。

LEX 是一个具有代表性的词法分析程序，它的输入是正规式，输出是词法分析程序（又称扫描器）。LEX 的基本思想是由正规式构造有穷自动机，功能结构如图 1-5 所示。

图 1-5 LEX 生成器

YACC（Yet Another Compiler Compiler）是一种基于 LALR(1)文法的语法分析程序生成器。

它接受 LALR(1)文法，生成一种相应的 LALR(1)分析式以及一个 LALR(1)分析器，而且 YACC 生成的语法分析程序可以和扫描器连接，生成器如图 1-6 所示。

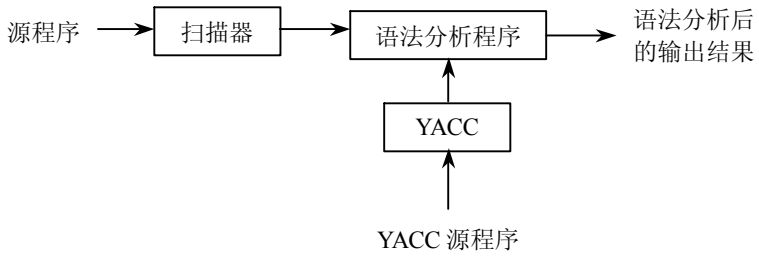

图 1-6　YACC 语法分析程序生成器

LEX 和 YACC 是关于编译程序前端的生成器，关于编译程序的后端（代码生成、代码优化）的生成器，直到最近才出现。

1.4　形式语言理论和编译实现技术

形式语言是一种不考虑含义的符号语言，形式语言理论研究的是组成这种符号语言的符号串集合、它们的表示法、结构和特性。按 Chomsky 文法分类法，文法可以分成四大类，即短语结构文法、上下文有关文法、上下文无关文法和正规文法。通常的程序设计语言，其符号的定义和正规文法相关联，语法定义和上下文无关文法相关联，而语义一般需要上下文有关文法来定义。总之，程序设计语言与形式语言理论紧密结合，依据形式语言理论，编译程序的词法分析与语法分析等不同的阶段可以采用最合适的分析技术。

形式语言理论采用类似数学那样的符号形式表示，类似数学那样的严格推理。采用形式语言方式讨论程序设计语言及其编译程序的构造，可以使我们不仅了解一些编译实现技术如何应用，还可以理解如此做的原因。

对编译程序的设计中所采用的技术，不但要知其然，而且要知其所以然。从理论高度上去掌握编译技术。可以用下列公式来表示一种关系，即：

编译原理=形式语言理论+编译技术

因此，在学习过程中，要重点学习三个方面的内容：

（1）高级语言程序设计。

（2）与实现编译有关的形式语言理论的基本概念。

（3）构造编译程序的基本概念、原理和技术。

小　结

自 FORTRAN 语言产生后，计算机高级语言得到了迅速发展。高级语言的种类越来越多，但计算机只能执行机器语言，不能直接执行高级语言编写的程序。要执行高级语言程序，必须提供该高级语言的翻译程序。翻译程序有编译程序和解释程序两种方式。编译程序将源程序翻译成目标程序，然后再执行目标程序。解释方式则是边翻译边执行。

编译程序一般包括词法分析程序、语法分析程序、语义分析程序、中间代码生成程序、代码优化程序、目标代码生成程序和出错处理程序等。

编译过程可以分遍编译。编译程序的构造有多种技术，主要有自编译、移植、交叉编译、自展和自动化技术。

LEX 是一个具有代表性的词法分析程序生成器。YACC 是一个基于 LALR(1)分析法的词法分析程序生成器。

习题一

一、选择题

1．构造编译程序应掌握（　）。

A．源程序　　B．目标语言

C．编译方法　　D．以上三项都是

2．编译程序绝大多数时间花在（　）。

A．出错处理　　B．词法分析

C．目标代码生成　　D．表格管理

3．编译程序是对（　）。

A．汇编语言的翻译　　B．高级语言程序的解释执行

C．机器语言的执行　　D．高级语言的翻译

4．高级语言程序设计是根据（　）定义的。

A．词法规则　　B．语法规则

C．语义规则　　D．以上三项都不是。

5．编译程序的各阶段都涉及到（　）。

A．词法分析　　B．表格管理

C．语法分析　　D．语义分析

6．编译程序将源程序加工成目标程序是（　）之间的转换。

A．词法　　B．语义

C．语法　　D．规则

7．解释程序和编译程序的区别是（　）之间的转换。

A．是否生成中间代码　　B．加工的对象不同

C．使用的实现技术不同　　D．是否生成目标代码

8．编译程序不能检查、处理的错误是程序中的（　）。

A．静态语义错误　　B．动态语义错误

C．语法错误　　D．词法错误

9．中间代码生成所依据的是语言的（　）。

A．词法规则　　B．语法规则

C．语义规则　　D．产生规则

二、判断题

（　）1．用高级语言编写的源程序都必须通过编译，产生目标程序后才能运行。
（　）2．源程序和目标程序在功能上具有等价关系。
（　）3．高级语言程序到低级语言程序的转换是结构上的变换。
（　）4．解释程序虽然不产生目标代码，但是它可能产生中间代码。
（　）5．目标程序一定是机器语言程序。

三、问答题

1．计算机执行用高级语言编写的程序有哪些途径？它们之间的区别是什么？
2．画出编译程序的构造图。
3．什么是编译程序的前端？什么是编译程序的后端？
4．简述自展技术、自编译技术、自动化和移植技术。