鲁斌 主编
李继荣 黄建才 闫蕾 岳燕 副主编

编译技术基础教程

清华大学出版社

编译技术基础教程

鲁斌 主编
李继荣 黄建才 闫蕾 岳燕 副主编

清华大学出版社
北京

内 容 简 介

编译的精髓在于做到原理、技术与实践方法的融会贯通，本书正是这样一部综合、全面、实用的编译技术教材。本着知识与能力相结合、理论与实用相并重的指导思想，以贯穿全书的样本语言编译器的开发为例，在简要介绍了编译技术所涉及的基本知识和高级语言的语法描述方法之后，按照编译程序的工作过程逐步介绍编译各个阶段的主要内容，具体包括词法分析、语法分析、语义分析与中间代码生成、符号表与运行时存储空间组织、代码优化以及目标代码生成等。通过本书的学习能够使读者系统而全面地掌握编译各个阶段的基本原理、技术和实践方法，并且运用所学技术进行编译程序的设计与开发。本书可用作高等学校计算机及其相关学科各专业本科生的教材或教学参考书，也可供其他技术开发人员参考。

图书在版编目（CIP）数据

编译技术基础教程 / 鲁斌主编. —北京：清华大学出版社，2011.10
ISBN 978-7-302-25197-2

Ⅰ. ①编… Ⅱ. ①鲁… Ⅲ. ①编译程序－程序设计－教材 Ⅳ. ①TP314

中国版本图书馆 CIP 数据核字(2011)第 060521 号

责任编辑：汪汉友　李玮琪
责任校对：梁　毅
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：13.5　　**字　数**：336 千字
版　　次：2011 年 10 月第 1 版　　**印　次**：2011 年 10 月第 1 次印刷
印　　数：1～3000
定　　价：23.00 元

产品编号：037582-01

前　言

编译技术作为计算机及其相关学科基础课程之一，经过多年的发展，现已成为一门理论体系完备、实践技术可行的专业核心课程。对于编译技术的学习，其价值不仅仅在于掌握了一门课程知识，更有助于加深对计算机软硬件系统的认识和理解，树立起大型系统软件开发的思路和体系，有助于学生基本计算机素养的培养和提高。

当前，已出版的编译教材不在少数，有些以原理介绍为主，有些以技术介绍为主，还有一些则以编程实践为主，当然，也存在着二者或三者相结合的情况，它们中不乏佼佼者。严格来说，把编译划分为原理型、技术型和实践型其实很难做到，因为它们之间是一种互为依托、相辅相成的关系，离开谁都不全面。但是，大家共有的认识是，理论必须与实践相结合，使学生既懂基本的原理，又能够掌握主要的技术，更为重要的是能够将所学知识应用于实践当中。重新组织和安排内容体系结构，按照认知的基本规律，有条理、有逻辑地逐步分析编译内涵，在有限的学时中达到上述目的，这是本书撰写的宗旨。

本书在内容的安排上，首先，简洁明了地介绍了编译技术所涉及的基本知识和高级语言的语法描述方法，然后，按照编译程序的工作过程逐步介绍编译各个阶段的主要内容，具体包括词法分析、语法分析、语义分析与中间代码生成、符号表与运行时存储空间组织、代码优化以及目标代码生成等。上述安排能够有效地帮助读者系统地理解和掌握编译的主要知识和技术。

本书是一部综合、全面、实用的编译技术教材，反映了作者多年来的教学思路和经验，经过反复推敲、几易其稿后得以成文。本书具有贴近教学、层次分明、体系完整等特点和教学参考价值，具体体现在：

(1) 贴近教学。编译的精髓在于做到原理、技术与实践方法的融会贯通，而非只知其一。若想做到这一点，就有限的学时而言，确实是一件很困难的事情。这就需要在充分理解编译内涵的基础上，对众多的编译技术知识重新梳理，归纳总结，详略得当、重点分明、通俗易懂地安排教学内容。这正是本书撰写的整体思路。

(2) 层次分明。本书是对作者长期教学与实践经验的总结，在知识点的逻辑结构上按照层次化教学理念，围绕横纵两条主线将编译系统逐步解剖开来。横向上，按照编译程序的工作过程组织章节；纵向上按照先原理，再技术，然后实践方法的顺序安排内容，从而使读者学习思路清晰，不仅知其然，而且知其所以然，由浅入深，循序渐进，以适合教与学。

(3) 体系完整。本书特别强调知识与能力结合，理论与实用并重。在讲授基本原理和技术的同时，也讲授相关的实践知识，力图培养读者扎实的动手实践能力，使读者能够深入地运用所学技术进行编译程序的设计与开发。

认真学完本书之后，读者将能够做到：

(1) 系统而全面地掌握编译各个阶段的基本原理、技术和实践方法。

(2) 通过贯穿全书的样本语言编译器的设计和实现过程，读者运用编译技术时得心应手。

本书由鲁斌主编，并负责第1、2章的撰写工作，第3章由岳燕编写，第4、5章由李继荣编写，第6、7章由黄建才编写，第8、9章由闫蕾编写，鲁斌负责统稿。特别感谢黄志强教授审阅了全书并提出了宝贵建议。另外，陈娟、张强、王艳丽和刘换霞在本书创作过程中做了大量的素材搜集和整理工作，在此一并表示感谢。

由于编者水平有限，错误之处在所难免，恳请广大读者批评指正。

本书可用作高等学校计算机及其相关学科各专业本科生的教材或教学参考书，也可供其他技术开发人员参考。

鲁　斌

2010年12月

目　录

第1章 绪 论

高级语言编译程序是计算机最为重要的系统软件之一。任何一门高级语言若想真正在计算机上被执行，都必须通过相应的翻译程序将其翻译为机器语言才行。翻译程序的典型代表是编译程序，开发编译程序所涉及的有关原理、方法和技术具有普适性的特点，广泛应用于与计算机相关的各个领域之中，因此，学习它具有十分重要的意义。本章首先介绍编译程序的基本概念和应用，然后介绍编译的过程与结构，最后对编译程序的构造方法进行简单说明。

1.1 编译程序简介

世界上最早的编译程序是20世纪50年代中期研制成功的FORTRAN编译程序。那时，人们普遍认为设计和实现编译程序是一件十分困难的事情。后来，为了改善这种状况，人们花费了大量时间来研究编译器的自动构造技术，开发出了用于词法和语法分析的自动生成工具Lex与Yacc。在20世纪70年代后期和80年代早期，许多项目都致力于编译器其他部分的自动生成，其中也包括了代码生成的自动化。

经过半个多世纪的不懈努力，编译理论与技术得到迅速发展，现在已形成了一套较为成熟的、系统化的理论和方法，并且开发出了为数不少的、优秀的编译程序的实现语言、环境与工具。在此基础上设计并实现一个编译程序不再是高不可攀的事情。

目前，编译器的发展与复杂的程序设计语言的发展密切结合，编译器也已成为交互式开发环境(IDE)的一个核心组成部分。随着多处理机和并行技术、并行语言的发展，将串行程序转换成并行程序的自动并行编译技术正在深入研究之中。另外，嵌入式应用的快速增长也极大地推动了编译技术的飞速发展。同时，对系统芯片设计方法和关键EDA技术的研究也带动了VHDL等专用语言及其编译技术的不断深化。

1.1.1 编译概述

计算机系统中的语言可以分为3个层次：机器语言(Machine Language)、汇编语言(Assembly Language)以及高级语言(High-level Language)。计算机刚出现时，人们编写程序使用的是机器语言。用机器语言编程费时、费力、难度大，因此，机器语言很快就被汇编语言取而代之。汇编语言以助记符的形式表示指令和地址，与机器语言相比尽管程序开发的难度有所降低，然而仍与人类的思维方式相差甚远，不易阅读和理解，并且严格依赖于机器，在一种机器上编写的代码应用于另一种机器时必须完全重写。高级语言的出现拉近了人类思维和计算机语言之间的距离，使得编写高级语言程序就如同书写自然语言一般，并且与机器无关。

然而，只有机器语言程序才能直接在计算机上执行，因此，必须寻找一种方法，能够将高级语言和汇编语言程序转换为对应的机器语言程序，这种方法就是翻译。通常所说的**翻译**

程序(Translator)是指能够把某一种语言(称为**源语言**,Source Language)书写的程序转换成另一种语言(称为**目标语言**,Target Language)书写的程序的程序,而后者与前者在逻辑上是等价的。如果源语言是诸如 FORTRAN、Pascal、C 或 Java 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",则称这样的翻译程序为**编译程序**(Compiler)或**编译器**。

根据用途和功能的不同,编译程序可以分为以下几种类别。专门用于帮助程序开发和调试的编译程序称为**诊断编译程序**(Diagnosic Compiler)。着重于提高目标代码运行效率的编译程序叫做**优化编译程序**(Optimizing Compiler)。运行编译程序的计算机称为**宿主机**,而运行编译程序所产生目标代码的计算机则称为**目标机**。如果一个编译程序产生不同于其宿主机的目标代码,则称其为**交叉编译程序**(Cross Compiler)。如果不需要重写编译程序中与机器无关的部分就能改变目标机,则称该编译程序为**可变目标编译程序**(Retargetable Compiler)。目前,很多编译程序具备多种功能,往往是上述不同类型编译程序的集合体,用户只需简单地通过选项设置即可选择不同的功能。

高级语言易编程、易调试、效率高的特点使得程序员开发程序成为一件相对轻松的事情,然而高级语言源程序要想在计算机上最终被执行,除了需要编译程序的处理外,还需要预处理程序、汇编程序以及装配连接程序等的相互配合才能达到目的。一个典型的高级语言程序的处理过程如图 1-1 所示。

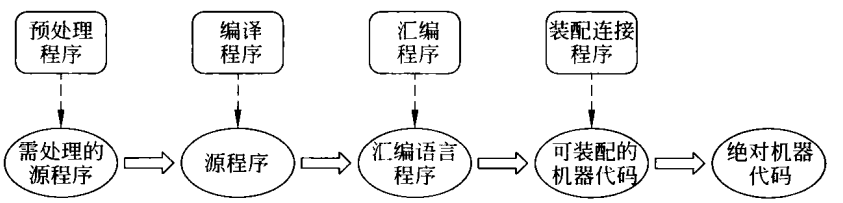

图 1-1 高级语言程序的处理过程

其中,**预处理程序**(Preprocessor)是指既能够将位于不同文件中的源程序模块汇集在一起,又能够将源程序中的宏语句展开为原始语句并加入到源程序中的程序。**汇编程序**(Assembler)是指汇编语言转换为机器语言的程序。**装配连接程序**(Loader-linker)是指将可装配的机器代码进行装配连接得到绝对机器代码的程序。预处理结束后,源程序经过编译过程生成目标程序,对于图 1-1 所示的汇编语言目标程序,需要经过汇编过程转换为可装配的机器代码,再经装配连接过程才能得到真正可执行的绝对机器代码。

高级语言除了可以"编译"执行外,也可以"解释"执行。**解释程序**(Interpreter)是指按高级语言程序的语句执行顺序边翻译边执行相应功能的程序。编译程序和解释程序的主要区别如下:

(1) 编译程序将整个源程序翻译为目标程序之后再执行,而且目标程序可以反复执行。

(2) 解释程序对源程序逐句地翻译执行,不产生目标程序。若需再次执行,则必须重新解释源程序。

本书重点介绍编译程序的基本原理、方法和技术,至于解释程序则不做专门探讨。但是,书中有关编译程序的构造与实现技术同样也适用于解释程序。

1.1.2 编译技术的重要性

高级语言的不断向前发展促使编译程序日益完善起来，现已形成了一套系统化的理论和方法。学习编译技术对于提高计算机软硬件素养均具有举足轻重的作用，主要体现在如下几个方面：

(1) 有助于深刻理解和正确使用程序设计语言。学习编译技术之前，读者在开发程序时仅仅能够按照语法规则来编写代码，缺乏对语言机理的深入理解，只知其“形”，不知其“神”。例如，全局变量和局部变量的本质区别是什么、函数的递归调用又是如何实现的？诸如此类的问题，通过学习编译技术都可以得到很好的解答，从而有助于读者深刻理解和正确使用程序设计语言。

(2) 有助于加深对整个计算机系统的理解。在目标代码的生成过程中，编译程序的内容涉及计算机内部的组织结构和指令系统，也涉及计算机的动态存储管理，还有可能涉及操作系统方面的知识。可以形象地说，编译程序如同一根红线，它把程序语言、计算机组织结构、指令系统以及操作系统等各方面的知识串联起来，使读者能够通过学习编译技术加深对整个计算机系统的理解。

(3) 编译程序的设计与开发技术具有普适性。本书在讲述编译技术时采用的是“自顶向下”、“逐步求精”的结构化程序设计思想，将整个编译程序划分为几个相对独立的功能模块分别予以介绍，从而将复杂的问题简单化。这样做既便于读者掌握各模块的构建方法和技术，又能够有效地保证程序的正确性、可靠性和可维护性。这种程序设计技术同样可用于其他软件的设计与开发之中。

(4) 编译技术的应用越来越广泛。编译技术不仅仅应用于编译器的开发当中，还广泛应用于文本编辑、程序调试、语言转换以及排版系统等领域内。近年来，随着微处理器技术的飞速发展，处理器性能的好坏在很大程度上取决于编译器的质量。因此，编译技术当之无愧地成为计算机的核心技术，其地位变得越来越重要。

1.1.3 编译技术的应用

编译技术的应用领域很广，最为典型的当属程序设计语言方面。为了提高编程的效率，缩短调试的时间，确保程序的可靠性，软件工作者研发了不少用于程序语言处理方面的工具。开发这些工具不同程度地应用了编译程序各个部分的方法和技术，具体表现在：

(1) 结构化的程序编辑器。这种编辑器除了具备基本的文本编辑功能外，还能够像编译程序那样对程序文本进行分析，从而使程序的书写规范化。代表性的软件工具有 Editplus 和 Ultraedit 等。

(2) 程序调试工具。结构化编辑器只能解决语法错误的问题，对于一个已经通过编译的程序而言，需要进一步了解程序的执行过程是否满足算法的设计要求，是否实现了预期的功能以及程序的运行结果是否正确等，而这些工作都可以利用调试工具来完成。调试工具的开发主要涉及程序的语法和语义分析技术，调试功能越强大，实现起来就越复杂。

(3) 程序测试工具。一般来说，程序的测试工具包括两种，一种是静态分析器，另一种是动态测试器。其中，静态分析器用来对程序进行静态的分析，主要工作包括对程序进行语

法分析并查填相关表格，检查变量定值与引用的关系等。例如，检查变量是否未定值就引用，或定值后未被引用，或源代码冗余等一些编译程序的语法分析所发现不了的错误。动态测试器通过在程序的合适位置插入一些信息，结合测试用例来记录程序的实际执行路径，并将运行结果与期望的结果进行比较分析，以此来帮助程序开发人员查找所存在的问题。C语言的测试工具就属于这种类型。

(4) 语言转换工具。计算机硬件的逐步更新换代推动着程序设计语言顺应时代潮流，与时俱进，不断向着更新、更好的方向迈进，同时，程序设计语言的推陈出新也为提高计算机的使用效率提供了良好的条件。然而，如何让一些常用软件或重要软件在无须重新编程的前提下就能在新的机器和语言环境下使用起来，成为一个十分关键的问题。为了避免重新编制程序所带来的人力和时间耗费，一种可行的解决途径就是将一种高级语言程序自动转换成另一种高级语言程序。当然，这种方法也适用于汇编语言程序向高级语言程序的转换。转换工具的开发要用到程序的词法和语法分析技术，最终生成的目标语言是另一种高级语言，而非编译程序所生成的汇编语言或机器语言。

1.2 编译程序的结构及编译过程

编译程序的工作，从输入源程序开始到输出目标代码为止的整个过程，是非常复杂的。图 1-2 给出了一个典型的编译程序的结构，从图 1-2 中可以看出，一个完整的编译程序主要包含 7 个功能模块，分别是词法分析器、语法分析器、语义分析与中间代码生成器、代码优化器、目标代码生成器，以及表格管理模块和错误处理模块。

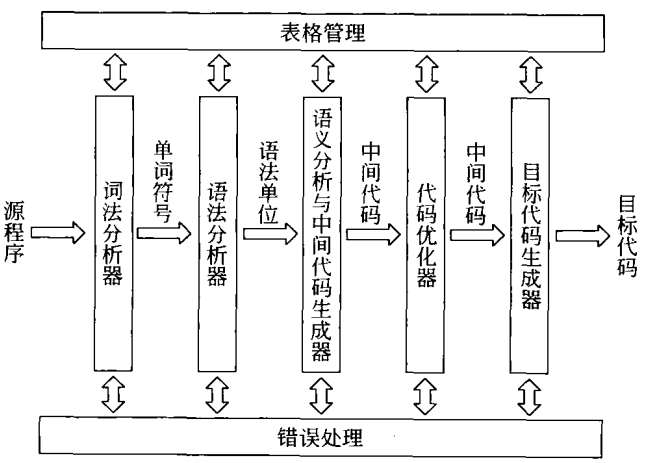

图 1-2 编译程序的结构

从工作过程的角度来分析，可将编译程序的工作过程划分为 5 个阶段：词法分析阶段、语法分析阶段、语义分析与中间代码生成阶段、代码优化阶段以及目标代码生成阶段。每个阶段分别与除表格管理和错误处理之外的编译模块相对应，完成不同的任务，并且各个阶段所做的操作在逻辑上是紧密相关的。编译过程尽管复杂，但与人们进行自然语言之间的翻译有许多相似之处，可以将二者进行比较，进而加深对编译过程的理解和认识。例如，把英文句子翻译为中文时，其过程与编译过程的比较如表 1-1 所示。

表 1-1　英文翻译与编译过程的比较

序号	英文翻译	编译过程
1	识别句子中的单词	词法分析
2	分析句子的语法结构	语法分析
3	初步翻译句子的含义	语义分析与中间代码生成
4	译文修饰	代码优化
5	写出最终译文	目标代码生成

词法分析是实现编译器的基础，语法分析是实现编译器的关键，只有语法结构正确，才能进行正确的翻译。然而，上述编译过程的 5 个阶段只是一种典型的分法，实际上，并非所有的编译程序都分成这 5 个阶段。有些编译程序不提供优化功能，那么优化阶段就可以省去；有时为了提高编译速度，中间代码生成阶段也可以省去；甚至有些编译程序简化到在语法分析的同时直接产生目称代码。尽管如此，多数实用的编译程序其工作过程大都与上面所说的那 5 个阶段相一致，本书将按照这个顺序来讲述编译程序各个阶段所涉及的基本理论、方法和技术。

1.2.1　词法分析器

词法分析器，又称扫描器(Scanner)，其功能是输入源程序，进行词法分析，输出单词符号。

每种高级语言都规定了允许使用的字符集，常见的字符有大写字母 A～Z、小写字母 a～z、数字0～9、＋、－、＊、/等。高级语言的单词符号(简称为单词)都由定义在字符集上的符号构成，它们都是语言中有意义的最小单位。单词一般分为 5 种，分别是保留字、标识符、常数、运算符和界符。

例如，对于如下的 Pascal 程序段：

```
var x,y,z : real;
x:=y+z * 60;
```

词法分析器将从左到右扫描输入的字符流，识别出一个个单词符号，并输出其内部表示形式，如表 1-2 所示。

表 1-2　程序中的单词符号

序号	类型	单词	内部表示	序号	类型	单词	内部表示
1	保留字	var	$ var	10	标识符	x	id
2	标识符	x	id	11	运算符	:=	:=
3	界符	,	,	12	标识符	y	id
4	标识符	y	id	13	运算符	+	+
5	界符	,	,	14	标识符	z	id
6	标识符	z	id	15	运算符	*	*
7	界符	:	:	16	常数	60	int
8	保留字	real	$ real	17	界符	;	;
9	界符	;	;				

为便于区分，x、y、z 的内部形式分别用 id_1、id_2 和 id_3 来表示，常数 60 的内部形式则用 int_1 来表示，于是上述程序片段经过词法分析后的输出为：

```
$var id1,id2,id3: $real;
id1:=id2+id3 * int1;
```

一般来说，词法分析器还要求将识别出来的名字填入符号表，以备后续阶段使用。单词符号是语言的基本组成单位，是用户理解和编写程序的基本要素。识别和理解这些要素同样也是自然语言翻译的基础，正如将英文翻译成中文一样，如果对英语单词不理解，那就谈不上正确的翻译。词法分析器主要依据语言的词法规则进行工作，描述词法规则的有效工具主要是正规式和有限自动机，这些内容将在第 3 章中详细介绍。

1.2.2 语法分析器

语法分析器，简称分析器(Parser)，其功能是依据语法规则对单词符号串进行语法分析(推导或归约)，从而识别出各类语法单位，最终判断输入串是否构成语法上正确的“程序”。

语法分析是在词法分析的基础上，将单词符号组成各类语法单位(又叫做语法范畴)，如“表达式”、“语句”和“程序”等，并通过语法分析来确定整个输入串是否构成语法上正确的“程序”。语法分析过程可以形象地用语法树表示出来，例如，单词符号串 $id_1:=id_2+id_3*int_1$ 经过语法分析后可以确定是一个赋值语句，该赋值语句可以表示为如图 1-3 所示的语法树。

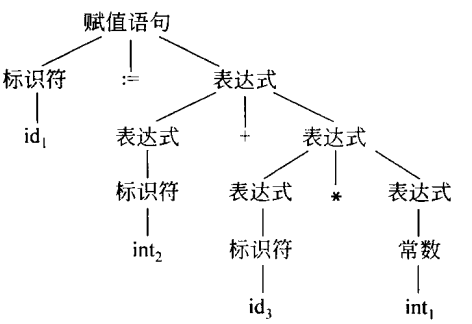

图 1-3 语句 $id_1:=id_2+id_3*int_1$ 的语法树

语法分析所依据的是语言的语法规则，而赋值语句和表达式的语法规则可定义如下：

(1) 赋值语句的语法规则：标识符：= 表达式。

(2) 表达式的语法规则：

① 任何标识符都是表达式；

② 任何常数都是表达式；

③ 若表达式 1 和表达式 2 都是表达式，则表达式 1+表达式 2，以及表达式 1 * 表达式 2 都是表达式。

图 1-3 中的语法树就是依据上述规则生成的。语言语法的描述一般采用的是上下文无关文法(参见第 2 章)，分析过程则采用下推自动机实现，具体内容将在第 4 章和第 5 章中介绍。

1.2.3 语义分析与中间代码生成器

语义分析与中间代码生成器的功能是按照语义规则对语法分析器识别出的语法单位进行语义分析，并把它们翻译成一定形式的中间代码。

这一阶段通常包括两方面的工作。一方面，对语法分析所识别出的各类语法单位进行静态语义检查，例如，变量是否定义，类型是否匹配等。另一方面，如果语义正确，则根据语法单位的类型分类进行处理。若为说明语句，则将变量的类型等属性填入符号表；若为可执行语句，则将其翻译为中间代码。

所谓**中间代码**(Intermediate Code),是指一种含义明确、便于处理的记号系统,通常具有硬件无关性,但与现代计算机的指令形式非常相似,很容易转换成特定计算机的机器指令。例如,许多编译程序采用了一种与"三地址指令"非常相似的"四元式"作为中间代码,其形式是:

(序号)(op,arg_1,arg_2,result)

其中,op 表示运算符,arg_1 表示左操作数,arg_2 表示右操作数,result 表示运算结果。与之等价的三地址指令表示如下:

result:=arg_1 op arg_2

语义分析和中间代码生成阶段的工作通常穿插在语法分析过程中来完成,因此,语义分析程序通常可定义为一组语义子程序,以便语法分析器进行调用。每当语法分析器分析出一个完整的语法单位,就会调用相应的语义子程序执行分析和翻译任务。例如,当语法分析器分析完语句 $ var id_1,id_2,id_3: $ real 后,就会把变量 id_1、id_2 和 id_3 的类型 real(实型)填入符号表中;当对赋值语句 $id_1:=id_2+id_3*int_1$ 进行分析时,则会一边检查 id_1、id_2、id_3 和 int_1 是否已经定义,类型是否一致,一边生成四元式形式的中间代码序列,如表 1-3 所示。

表 1-3 语句 $id_1:=id_2+id_3*int_1$ 的中间代码

序号	四元式	序号	四元式
(1)	(itr,int,,T_1)	(3)	(+,id_2,T_2,T_3)
(2)	(*,id_3,T_1,T_2)	(4)	(:=,T_3,,id_1)

其中,T_1、T_2 和 T_3 是编译期间引进的临时变量;第 1 个四元式表示将整型数 int_1(60)转换为实型数(60.0)并存放于 T_1 中;第 2 个四元式将 id_3 和 T_1 相乘后存放结果于 T_2 中;第 3 个四元式将 id_2 和 T_2 相加后存放结果于 T_3 中;最后 1 个四元式用来将 T_3 的值赋给 id_1。

除了四元式之外,常见的中间代码形式还有后缀式、三元式、间接三元式以及图形表示等。语义分析依据的是语言的语义规则,通常采用属性文法予以表示,而分析方法则采用语法制导翻译方法,这些内容将在第 6 章中介绍。

1.2.4 代码优化器

代码优化器的功能是对中间代码进行优化处理。

优化的任务在于对产生的中间代码进行等价变换,以期最终能够生成时间和空间方面更为高效的目标代码。其目的主要是提高运行效率,节省存储空间。可以将优化分为两类:一类是与机器无关的优化,主要是针对中间代码进行的,包括局部优化、循环优化和全局优化等;另一类是与机器有关的优化,主要是在生成目标代码时进行的,涉及如何分配寄存器以及如何选择指令等。

对于表 1-3 的中间代码,优化前和优化后的差异如图 1-4 所示。

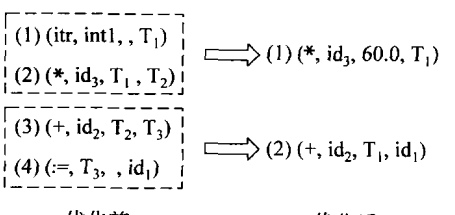

图 1-4 语句 $id_1:=id_2+id_3*int_1$ 的中间代码的优化