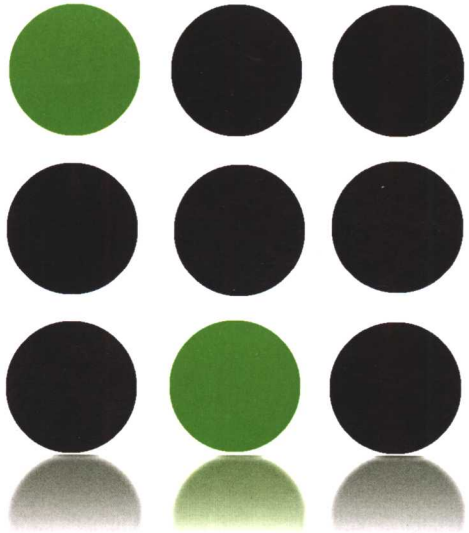
高等院校计算机专业课程综合实验系列规划教材



丛书主编　何钦铭　陈根才

冯　雁　鲁东明　李　莹
王　强　徐海燕　陈　华　著
陈志刚　主审

编译原理课程设计

ZHEJIANG UNIVERSITY PRESS
浙江大学出版社

浙江大学魏绍相计算机教材建设基金资助
高等院校计算机专业课程综合实验系列规划教材

编译原理课程设计

冯　雁　鲁东明　李　莹
王　强　徐海燕　陈　华　著

陈志刚　主审

浙江大學出版社

内容提要

本书围绕着编译技术的基本原理和方法，以模拟程序设计语言 SPL(Simple Pascal Language)的编译器的设计和实现为主线，结合词法分析、语法分析、语义分析、代码生成、代码优化、错误处理等各个基本模块，对原理和实现方法进行了详细分析。该编译器可接受 SPL 的程序，并将其翻译成汇编语言程序，最终实现汇编语言到 8086/8088 机器语言的翻译。本书为编译技术等相关课程的实验提供了参考。在附件中还提供了三类不同类型和难度的实验题，可供课程实验选择。本书所附光盘包含了 SPL 编译器的所有代码。

本教材适合作为编译技术课程的配套的实验教材，也可作为有关编译方面研究的参考资料。

图书在版编目（CIP）数据

编译原理课程设计 / 冯雁等著. —杭州：浙江大学出版社，2007.11

（高等院校计算机专业课程综合实验系列规划教材；11406）

ISBN 978-7-308-05633-5

Ⅰ.编… Ⅱ.冯… Ⅲ.编译程序－程序设计－高等学校－教材 Ⅳ.TP314

中国版本图书馆 CIP 数据核字（2007）第 167175 号

编译原理课程设计

冯雁 鲁东明 李莹 王强 徐海燕 陈华 著

陈志刚 主审

丛书主编 何钦铭 陈根才

策 划 黄娟琴 希 言

责任编辑 邹小宁 吴昌雷

封面设计 氧化光阴

出版发行 浙江大学出版社

（杭州天目山路 148 号 邮政编码 310028）

（E-mail：jsjsyb@zju.edu.cn）

（网址：http://www.zjupress.com

http://www.press.zju.edu.cn）

排 版 浙江大学出版社电脑排版中心

印 刷 德清县第二印刷厂

开 本 787mm×1092mm 1/16

印 张 14.5

字 数 362 千

版 印 次 2007 年 11 月第 1 版 2007 年 11 月第 1 次印刷

印 数 0001—3000

书 号 ISBN 978-7-308-05633-5

定 价 26.00 元（含光盘）

专家指导委员会

序　言

近10多年来，以计算机和通信技术为代表的信息技术迅猛发展，并已深入渗透到国民经济与社会发展的各个领域。信息技术成为国家产业结构调整和推动国民经济与社会快速发展的最重要的支撑技术。与此同时，深入掌握计算机专业知识、具有良好系统设计与分析能力的计算机高级专业人才在社会上深受欢迎。

计算机科学与技术是一门实践性很强的学科。良好的系统设计和分析能力的培养需要通过长期、系统的训练(包括理论和实践两方面)才能获得。高等学校的实践教学一般包括课程实验、综合性设计(课程设计)、课外科技活动、社会实践、毕业设计等，基本上可以分为三个层次：第一，是紧扣课堂教学内容，以掌握和巩固课程教学内容为主的课程实验和综合性设计；第二，是以社会体验和科学研究体验为主的社会实践和课外科技活动；第三，是以综合应用专业知识和全面检验专业知识应用能力的毕业设计。课程实践(含课程实验和课程设计)是大学教育中最重要也最基础的实践环节，直接影响后继课程的学习以及后继实践的质量。由于课程设计是以培养学生的系统设计与分析能力为目标，通过团队式合作、研究式分析、工程化设计完成较大型系统或软件的设计题目的，因此课程设计不仅有利于学生巩固、提高和融合所学的专业课程知识，更重要的是能够培养学生多方面的能力，如综合设计能力、动手能力、文献检索能力、团队合作能力、工程化能力、研究性学习能力、创新能力等。

浙江大学计算机学院在专业课程中实施课程设计(project)已有10多年的历史，积累了丰富的经验和资料。为全面总结专业课程设计建设的经验，推广建设成果，我们特别组织相关课程的骨干任课教师编写了这套综合实验系列教材。本系列教材的作者们不仅具有丰富的教学和科研经验，而且是浙江大学计算机学院和软件学院的教学核心力量。这支队伍目前已经获得了两门国家精品课程以及四门省部级精品课程，出版了几十部教材。

本套教材由《C程序设计基础课程设计》、《软件工程课程设计》、《数据结构课程设计》、《数值分析课程设计》、《编译原理课程设计》、《逻辑与计算机设计基础实验与课程设计》、《操作系统课程设计》、《数据库课程设计》、《Java程序设计课程设计》、《面向对象程序设计课程设计》、《计算机组成课程设计》、《计算机体系结构课程设计》和《计算机图形学课程设计》等十三门课程的综合实验教材所组成。该系列教材构思新颖、案例丰富，许多案例直接取材于作者多年教学、科研以及企业工程经验的积累，适用于作为计算机以及相关专业课程设计的实验教材；也适用于对计算机有浓厚兴趣的专业人士进一步提升计算机的系统设计与分析

能力。从实践的角度出发，大部分教材配备了随书光盘，以方便读者练习。

可以说，本套教材涵盖了计算机专业绝大部分必修课程和部分选修课程，是一套比较完整的专业课程设计系列教材，也是国内第一套由研究型大学计算机学院独立组织编写的专业课程设计系列教材。鉴于书中难免存在的谬误之处，敬请读者指正，以便不断完善。

主编　何钦铭、陈根才

2007 年 6 月于求是园

前　言

编译原理是计算机科学与技术专业最重要的专业课之一，在计算机本科教学中占有十分重要的地位。编译原理课程具有很强的理论性与实践性，但是在教学过程中容易偏重于理论的介绍而忽视了实验环节，因此学生很难真正掌握这门学科的精髓。学生在学习时普遍感到内容抽象，不易理解，掌握起来难度较大。本书正是为弥补这一缺陷而编写的，可以和讲授编译原理理论的教材配合使用，对学生的实验有指导和帮助作用。

本书设计了模拟程序设计语言 SPL(Simple Pascal Language)及其编译器。该语言具有标准的数据类型和结构数据类型，以及基本的函数调用和过程调用，并包括各种控制语句。该语言的编译器涵盖了编译原理的词法分析、语法分析、中间代码生成、代码优化和目标代码生成等各阶段的内容，可接受 SPL 的程序，并将其翻译成我们所熟悉的 PC 机汇编语言程序，最终实现汇编语言到 8086/8088 机器语言的翻译。通过对该语言的编译器的分析，可使读者对编译原理有一个形象、直观和透彻的认识和感受，更深入地了解和掌握编译原理的内容和实现方法，进而提高分析问题与解决问题的能力。

本书的主要内容如下：第 1 章简单介绍了编译程序的基本结构、设计平台和 SPL 语言。第 2 章简单介绍了词法分析的原理，详细分析了词法分析程序的基本数据结构、预处理程序、词法分析程序的实现。第 3 章简单介绍了语法分析的原理和具体方法，详细分析了语法分析程序的基本数据结构及其语法分析程序的具体实现。第 4 章简单介绍了符号表的管理，并对符号表的数据结构和语义分析的实现进行了详细的说明。第 5 章简单介绍了编译原理的错误处理技术及错误分类，并对错误处理的实现机制进行了探讨，介绍了对错误处理程序的实现。第 6 章介绍了代码生成涉及的指令编码、生成代码的管理，存储管理和寄存器管理等；详细分析了各种控制结构生成代码的方法。第 7 章介绍了中间代码的优化技术，并对相应的实现进行了详细的说明。第 8 章对 SPL 语言编译器进行了详细剖析，包括各模块的构成、具体的数据结构、各个模块之间的接口等。附件根据不同层次的读者给出了三类实验题目，有针对性地提出了实验环节的学习目标。书后光盘给出了基于 SPL 的语言编译器源代码，供读者参考使用。

本书由冯雁撰写第 1 章和第 4 章，徐海燕撰写第 2 章，鲁东明撰写第 3 章，，王强撰写第 5 章和第 7 章，李莹撰写第 6 章，陈华撰写第 8 章。另外，在本书的写作过程中也得到了研究生占志峰、王小燕等的帮助，在此表示深切的谢意。

由于水平所限，对书中存在的谬误之处，敬请读者指正。

作　者

2007 年 10 月

目　录

第1章 引 论

编译器的原理与技术具有十分普遍的意义，在计算机科学中也是比较成熟的学科，其发展历史虽然不长，但其内容却十分丰富，用途也十分广泛。它是计算机专业，特别是软件专业的主要基础课。编译器的编写涉及程序设计语言、计算机体系结构、语言理论、算法和软件工程等内容。

编译原理是一门实践性较强的课程，仅仅是介绍原理和算法，很难真正学好这门课程，在以后的实践中也很难得到较好的应用。本书的目的就是配合编译原理课程的教学，能促使学生善于综合运用所学理论，融会贯通地掌握所学知识，锻炼实际开发设计项目的能力。

1.1 本书介绍

学习编译原理最好的方法莫过于自己动手设计实现一个编译器。本书以设计、实现一个类 PASCAL 编译程序(SPL)为主题，以编译器的各个模块：词法分析、语法分析、语义分析、代码生成、代码优化等为主线，分析介绍在具体设计和开发过程中所需要解决的一些问题，综合介绍编译原理、数据结构、C 语言、汇编语言多方面的内容。重点介绍了 SPL 编译器。

1.2 SPL 语言的特点及实验安排

SPL 语言是一种类 PASCAL 语言，后者由 N. Wirth 于 1971 年提出来的，它是系统地体现由 E. W. Dijkstra 和 C. A. R. Hoare 定义的结构程序设计概念的第一种语言。PASCAL 语言具有结构清晰、便于学习和有丰富的数据类型和语句的特点，因而选择它作为编译器的源语言。SPL 语言具有标准的数据类型和结构数据类型，包括数组和记录结构，具有基本的函数调用和过程调用，语句包括赋值语句、if 语句、repeat 语句、while 语句、for 语句、case 语句等，表达式包括逻辑表达式、数学表达式。

Intel 80X86 体系有较为丰富的技术资料，所以本书选择 X86 汇编语言作为目标语言，这也便于在开发中使用现有的汇编器(Assembler)和调试器(Debugger)。

本书中实现的编译器采用了模拟的 SPL 语言(Simple Pascal Language)。SPL 具备了过程式语言的基本特征，该语言的编译器涵盖了编译原理的词法分析、语法分析、中间代码生成、代码优化和目标代码生成等各阶段的内容，该编译器可接受 SPL 的程序，并将其翻译成

我们所熟悉的 PC 机汇编语言程序，最终实现从汇编语言到 X86 机器语言的翻译。通过对该语言的编译器的分析，使学生对编译原理有一个形象、直观和透彻的认识和感受，以便更深入了解和掌握编译原理的内容和实现方法，进而提高分析问题与解决问题的能力。

SPL 编译器的运行环境如下：

对于类 PASCAL 程序 EXAMPLE.PAS(在 DOS 环境下)

(编译) SPLC EXAMPLE.PAS

(汇编)MASM EXAMPLE.ASM

(拷贝操作系统相关库函数的汇编文件，符合 DOS 8.3 文件命名规则)

COPY X86RTLDOS.ASM X86DOS.ASM

(汇编)MASM X86DOS.ASM

(连接) LINK EXAMPLE.OBJ X86DOS.OBJ

(运行)EXAMPLE

在后面每一章节中，有关编译器的主要理论以及实现技术，将围绕 SPL 语言的编译器加以展开。同时，结合每部分的具体内容，也都布置了若干个实验程序可以练习，并作为附件 1 放在本书最后。

1.2.1 SPL 语言的特点

SPL 语言文法类似 PASCAL 语言文法。

数据类型：整型、布尔型、字符型、字符串、枚举、子界、数组、记录。

数据操作：加、减、乘、除、整除、取模、布尔代数。

I/O 操作：read、write、readln、writeln。

标准函数：abs、odd、sqr、sqrt、succ、pred。

语句：if-then-else 语句、repeat-until 语句、while 语句、for 语句、case 语句。

函数和过程：具有分程序规则，可嵌套、递归调用。

1.2.2 SPL 语言编译器的主要结构

SPL 编译器是由词法分析、语法分析、代码生成、符号表管理和错误处理等部分组成，如图 1.1 所示。

在结构上，SPL 编译器由词法分析、语法分析、语义分析、中间代码生成、中间代码优化和目标代码生成、符号表管理、错误处理等模块组成。其中词法分析、语法分析、语义分析、中间代码生成和部分的中间代码优化(目标机器无关优化)是语言相关目标机器无关的前端，而剩余部分的中间代码优化(目标机器相关优化)和目标代码生成是语言无关目标机器相关的后端，符号表和错误处理是公共部分。前端和后端之间的联系是 DAG 表示的中间代码。将编译器的结构分为前端和后端的原因是为了提高对 SPL 进行移植或扩展时的代码重用性。在现有的 SPL 编译器的基础上，如开发另外一种语言的编译器只需要扩充前端，而如果需要让 SPL 编译器支持另外的目标机器和操作系统只需要扩充后端。

SPL 编译器的词法分析采用了 LEX 工具，代码位于 spl.l 文件。语法分析采用 YACC 工具，代码位于 spl.y 文件，另外有辅助生成语法树的 tree.c 文件。出于效率和 YACC 工具的特性，语义分析的代码也存在于 spl.y 文件中，由 YACC 在识别语法单位时调用。中间代

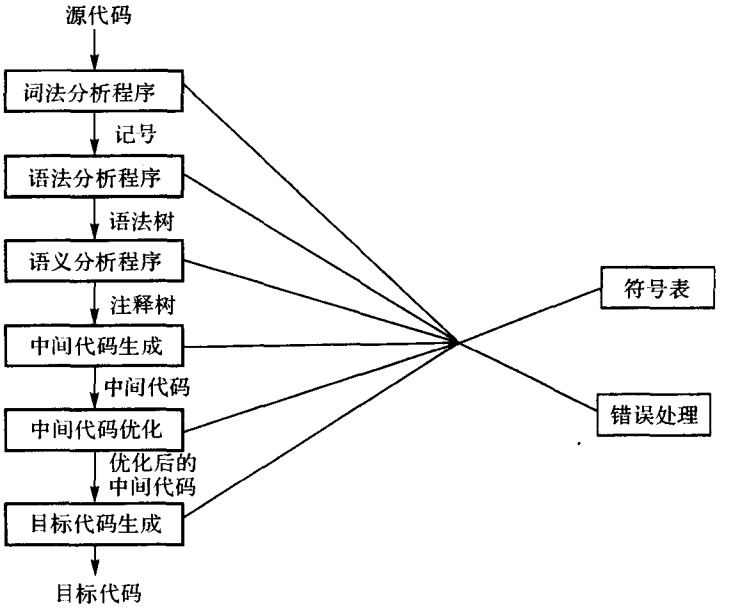


图 1.1 编译器的主要结构示意

码生成位于 dag.c 文件,将语法树转换为 DAG 图。部分的中间代码优化在中间代码生成阶段完成,其他的优化部分存在于文件 opti.c 中。代码生成部分包括了 X86linux.c,X86dos.c,分别对应于生成 Linux 操作系统下目标代码和 DOS 操作系统下目标代码,另外还有两个汇编语言编写的文件 X86rtllinux.asm 和 X86rtldos.asm,分别实现了 Linux 和 DOS 操作系统下的 SPL 系统函数(如 read,write,sqrt 等)。符号表操作部分包括 symtab.c 和 type.c。错误处理包含在各个部分中,error.c 处理错误的输出格式。

与一般的编译器相比,SPL 编译器的结构要紧凑得多,这主要是出于高效率和便于实现的要求。其核心部分是语法分析和代码生成部分,即由语法定义配上语义子程序的 YACC 程序,它把语法分析和代码生成合成为一个部分。其他的词法分析、符号表管理、代码优化和错误处理部分则采用通常的方式来构造。

具体实现上,对 SPL 编译器的主要功能模块有一些基本考虑点。

1.对词法分析器的基本考虑

(1) 正确识别每一个单词,为语法分析提供正确的单词,这主要依靠正确设计正规定义和辅助动作。

(2) 保持与标准 PASCAL 程序的一致性,由于 SPL 编译器只能处理 PASCAL 语言的子集,对标准 PASCAL 程序必然有无法处理的情况,因此要在词法处理过程中报告出现了这种情况,以避免在语法分析中产生不正确的错误信息。

(3) 保持词法分析器与语法分析器通信的正确性和动作的同步性。

(4) 在 I/O 操作和单词识别两方面具有较高的效率,可采取设置缓冲区,增加每次读写量从而减少总的 I/O 操作次数来满足 I/O 的要求。通过改进对关键字表的查找方法来提高单词识别的效率。

(5) 词法分析器的设计具有一定的通用性,只需很少的改动就可以移植到其他系统。

这一要求主要通过提高词法分析模块相对整个系统的独立性，提高与词法有关的过程和通用过程的相对独立性来实现。

2. 对符号表的基本考虑

(1) 完整性。符号表项应包含编译各阶段所需的信息，能完整地体现符号的特征。

(2) 灵活性。能将不同性质、类型的符号以尽可能统一的形式表达，在提高符号表应用范围的同时降低其复杂度。

(3) 动态性。由于标识符的数量随程序的不同而不同，也随同程序内模块的不同而有较大的不同，因此要求符号表中表项的数量能随符号数量的增长而动态增长。这不仅有利于提高内存的利用率，而且有利于提高编译程序对不同程序的适应能力。

(4) 可维护性。符号表体系应该层次化、模块化，便于维护和改进。

3. 对代码生成的基本考虑

由于代码生成模块是整个编译器的核心模块，因此在设计时不仅要考虑到模块自身的组织方式、数据结构和算法，还要考虑与其他所有模块之间的接口问题。在设计中的主要应考虑的问题有：

(1) 语法制导的定义。要尽量保证描述语言的文法正确性，还有考虑解决文法二义性的具体方法。

(2) 编译模式。编译模式也称为内存模式，它是指如何在内存为程序、数据、堆栈分配空间并存取它们。通常编译器提供微模式、小模式、紧凑模式、中模式、大模式和巨模式 6 种编译模式。由于程序规模的限制，SPL 编译器只提供小模式，适于代码规模小、数据量小的程序。在小模式中，代码段与数据段分离，即最大可用空间为 128K，寻址方式都是 16 位。所有调用都是近调用，但同时允许对个别不在代码段内的函数用 FAR 关键字来调用。

(3) 调用帧的设计。调用帧是支持子程序结构的重要部分。子程序调用及返回包含着数据传送和控制转移两方面的内容，要保证调用的正确执行，必须遵循一定的规则，调用帧要有较固定的格式。

(4) 运行库的设计。将经常用到的 abs，sqr，pred 等标准函数和 read，write 等标准过程集中在一起，建立运行库(Run-Time Library)，在用户代码中仅生成一条过程调用的指令，由连接程序将有关代码装入可执行程序。

(5) 错误检测和恢复。主要考虑如何限制错误的重复报告，如何保证提供的错误信息的准确性，并使得错误检测和恢复的实施不会在语法分析等模块的效率上造成太大的影响。

1.2.3 实验安排

在编译原理的学习过程中，实验非常重要，只有通过上机实验，才能对比较抽象的课程内容产生一个具体的感性认识，对这些工作原理有一个详细的了解，达到“知其然，且知其所以然”的目的。建议实验环境主要为 C 或 C++ 环境及一个词法分析器自动生成工具 LEX 和一个语法分析器自动生成工具 YACC。在下一节中，将对这两个工具做初步的介绍。

实验题目主要分为三大类：

第一类实验题主要以单独的一些词法分析和语法分析小程序为主，无须与本书提供的 SPL 编译器相关，相对难度较低，适合一般性的学习要求。

第二类实验题与SPL编译器有较紧密的联系，主要在SPL编译器的基础上，对SPL的功能进行一些替换或增加。有一定的难度，适合要求较高的学习。

第三类实验题是在课程学习的同时，参考SPL编译器，自行设计和实现某类语言的小型编译器。有一定的难度，适合要求较高的学习。

具体关于实验学时和安排，可根据实际情况和难度要求，选做其中的一部分。有以下一些建议：仅考虑第一类实验题，可根据具体实验学时数选择若干题，如1个学分的实验学时，建议安排第一类题3～5题。如希望有一定难度要求，可根据实验学时数，在第一类题目中各选一个词法分析和语法分析的小程序作为准备工作，再选择第二类中的1～2个题或者第三类题。

有关具体实验题目详见最后附件1。

1.3 平台的选择和介绍

在编译过程中，词法分析和语法分析是两个重要阶段。上节也提到了SPL编译器中采用了辅助工具LEX和YACC。它们是Unix环境下非常著名的两个工具，可以生成分别完成词法分析和语法分析功能的C代码。在学习编译原理过程中，可以利用这两个工具，加深对两个阶段的理解。选择该工具作为开发编译器的平台也是由于该工具是一个开放的源码，并且随着计算机技术的发展，也诞生了很多不同环境下的版本，比较方便在这些版本上开发编译器。如熟知的有Unix环境下的LEX和YACC，有英国Bumble-Bee Software公司生产的Windows环境下的YACC和LEX集成环境Parser Generator，它包括一个图形用户界面，同时包括YACC和LEX两个版本，分别叫做AYACC和ALEX。

在这一节中对LEX和YACC的一些主要功能进行介绍。

1.3.1 LEX简介

LEX是LEXical compiler的缩写，主要功能是生成一个词法分析器(Scanner)的C源码。描述词法分析器的文件，经过LEX编译后，生成一个lex.yy.c的文件，然后由C编译器编译生成一个词法分析器。词法分析器，简单地说，其任务就是将输入的各种符号，转化成相应的标识符(token)，转化后的标识符很容易被后续阶段处理。

LEX源程序是用一种面向问题的语言写成的，核心是正规表达式，描述输入串的词法结构。在这个识别语言中还可以描述当一个单词被识别时要完成的动作。LEX并不是一个完整的语言，只是某种高级语言(称为LEX的宿主语言)的扩充，因此LEX没有为描述动作设计新的语言，而是借助其宿主语言来描述动作。在此以C语言为例。图1.2给出LEX的工作示意图。LEX源程序经过LEX编译器的处理，自动把表示输入词法结构的正规式及相应的动作转换成一个宿主语言的程序，在Unix环境中，lex.l为LEX的源程序，Lex.yy.c为LEX的目标程序，它是一个C程序，包括以正规式构造的表格形式表示的状态转换图，以及使用该表格识别单词的标准子程序。在lex.l中包含一些C代码段，是词法分析器需要执行的动作。Lex.yy.c程序经由C编译生成目标文件a.out，该目标文件可以将所编译的高级程序设计语言的输入字符流变换成标记流。

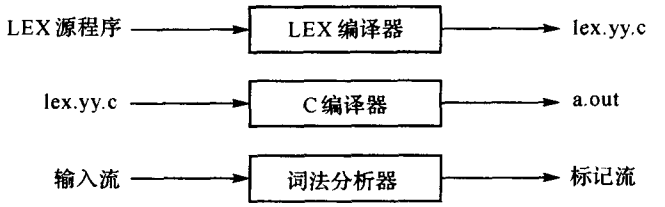


图 1.2 LEX 工作示意

LEX 编译器的工作原理是:LEX 编译器将 LEX 源程序中的正规式经过若干步骤的转换(正规式→NFA→DFA→最小状态数的 DFA),最终转换成相应的等价确定有限状态自动机,并将其动作插入到 lex.yy.c 中适当的地方。控制流是由确定的有限状态自动机的解释器掌握,解释器是 LEX 的构成部分,对不同的输入源程序来说解释器是相同的。

LEX 源程序由三部分构成:说明部分、转换规则以及辅助过程,各个过程之间用%%做间隔符,格式为:

```
说明部分
%%
转换规则
%%
辅助过程
```

这三个部分不是都必须具备的,当没有辅助过程时,第二个%%也可以省略。第一个%%是必须有的,标志着转换规则部分的开始。最短的合法 LEX 程序是:

```
%%
```

它的作用是将输入串原样抄到输出文件中去。

说明部分包括变量的说明、常量说明及正规定义。正规定义是形式如下的一系列定义:

$d_1 \rightarrow r_1$

$d_2 \rightarrow r_2$

...

$d_n \rightarrow r_n$

设 Σ 是基本字母表,每个 d_i 是不同的名字,每个 r_i 是在 $\sum \cup \{d_1, d_2, \cdots, d_{i-1}\}$ 上的正规式,即由基本字母表和前面定义的名字组成。正规定义的 d_i 可以用做转换规则中出现的正规表达式的成分使用。有些 LEX 版本不用"→",正规定义形式直接为:

$d_1 \quad r_1$

$d_2 \quad r_2$

...

$d_n \quad r_n$

如 IDENT [a-zA-Z] [a-zA-Z0-9]*。

转换规则是 LEX 的核心,它是一张表,左边一列是正规式,右边一列是相应的动作,形

式如下所示：

p_1　$\{action_1\}$

p_2　$\{action_2\}$

...

其中，每个 p_i 是一个正规式，每个 $action_i$ 是一段 C 代码，指明识别 p_i 后相应的动作。正规式与动作之间必须有空格分开，如果动作占两行或以上，必须用花括号括起来。正规定义在识别规则中必须用{}括起来使用，系统自动进行替换。在转换规则中可递归使用已定义的常量，但正规定义不能循环定义。输入串中不与任何正规式匹配的字符串被原样抄到输出文件中去。若希望滤掉输入中的某些字符串，可以用空语句实现。

识别规则的二义性：当多于一条规则与同一个字符串匹配时，LEX 的处理原则是匹配最多字符的规则优先；并且在能匹配相同数目的字符的规则中，先给出的规则优先。

辅助过程是 action 中所需要的一些辅助例程，这些例程可以分别编译并且放置于词法分析器中。

由 LEX 生成的词法分析器与语法分析器协同工作方式如下：词法分析器被语法分析器调用后，从尚未扫描的输入字符串中读字符，每次读入一个字符，直到发现能与某个正规表达式 p 匹配的最长前缀。然后，词法分析器执行相应的 action，通常 action 会把控制返回给语法分析器。如若不然，则词法分析器继续发现更多的标记，直到某个操作把控制返回给语法分析器。

通常，词法分析器只返回标记给语法分析器，而与标记相关的属性值是通过全局变量 yylval 传递的。

1.3.2　YACC 简介

YACC 代表 Yet Another Compiler Compiler。YACC 的 GNU 版叫做 Bison。它是一种工具，将任何一种编程语言的所有语法翻译成针对此种语言的 YACC 语法解析器。它用巴科斯范式(Backus Naur Form，BNF)来书写。按照惯例，YACC 文件有 .y 后缀。

从字面上理解，YACC 是一个编译程序的编译程序，但严格说它还不是一个编译程序自动产生器，因为它不能产生完整的编译程序。YACC 输入用户提供的语言的语法描述规格说明，基于 LALR 语法分析的原理，自动构造一个该语言的语法分析器，如图 1.3 所示，同时，它还能根据规格说明中给出的语义子程序建立规定的翻译。

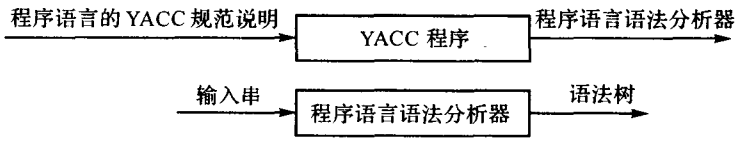


图 1.3　YACC 程序的作用

YACC 规格说明(或称 YACC 源程序)由说明部分、翻译规则和辅助过程三部分组成，其形式如下：