<<编译原理及实现技术>>

图书基本信息

书名：<<编译原理及实现技术>>

13位ISBN编号：9787111312611

10位ISBN编号：7111312619

出版时间：2010-8

出版时间：机械工业出版社

作者：刘磊

页数：184

前言

编译原理是计算机学科的一门重要专业基础课。
学习编译课程，不仅可以掌握编译程序本身的实现技术，而且能够提高对程序设计语言的理解，提高开发大型软件的能力，提高软件程序的设计能力，提高抽象思维能力。

编译程序是计算机系统软件的重要组成部分，其基本原理和实现技术也适用于一般软件的设计和实现，而且在软件工程、软件自动化、程序分析等领域有着广泛的应用。
通常把编译程序视为高级语言到机器语言的转换程序，而这种转换不是结构上的变换，而是基于语言语义的等价变换，因此，编译程序设计的难度和复杂性是很高的。
同时，编译原理也是一门对实践性要求较高的课程。
本书充分考虑了便于教师教学，便于学生自学的问题，循序渐进地介绍了编译程序设计的基本原理、主要实现技术、基本设计方法和一些自动构造工具，深入浅出地介绍了完整的编译程序构造和实现过程，使学生能够掌握编译的整体结构。

本书共分10章。
第l章介绍了编译程序的基础知识。
第2章作为编译程序的理论基础，简单介绍了形式语言、有限自动机理论和正则表达式等基础知识。
第3章以正则表达式、有限自动机为工具，讨论了词法分析程序的设计与实现，并简要介绍了词法分析器生成器LEX的基本原理和使用方法。
第4章介绍了自顶向下的语法分析方法的基本思想，并讨论了递归下降语法分析方法和LL（1）语法分析方法的实现技术。
第5章介绍了自底向上语法分析方法的基本思想，并详细讨论了LR类语法分析的基本原理和实现方法，同时简单介绍了流行的语法分析器生成器YACC、Bison等工具。
第6章专门介绍语义分析，包括标识符、类型、值的内部表示及其构造，符号表的构造及其管理。
第7章介绍了中间代码生成，包括常用中间代码结构、表达式的中间代码、下标变量的中间代码以及语句的中间代码。
第8章介绍了中间代码优化的基本方法，重点讨论了常量表达式优化、公共表达式优化和循环不变式优化。
第9章介绍编译程序运行时的存储空间组织与存储分配技术，重点讨论了运行时的存储结构、存储分配、过程活动记录以及变量访问环境等。
第10章介绍了目标代码生成的基本技术，重点讨论了中间代码到目标代码的翻译。

<<编译原理及实现技术>>

内容概要

编译原理是计算机学科的一门重要专业基础课。
本书旨在介绍编译程序设计的基本原理、实现技术、方法和工具，充分考虑了教师便于教学，学生便于自学的问题。
在介绍基本原理和实现技术中，注重循序渐进、深入浅出，每一章节都提供了编译程序实现的具体实例，每章末尾给出了丰富的习题以辅助学生更好地掌握编译过程。
本书包含了编译程序设计的基础理论和具体实现技术，主要内容有：形式语言和自动机理论、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等编译过程。
本书可作为大专院校计算机专业本科生教材，也可作为计算机工程技术人员的参考书。

<<编译原理及实现技术>>

书籍目录

第1章 编译引论 1.1 程序设计语言和编译程 1.2 编译程序的结构 1.2.1 编译程序的构成 1.2.2 遍 1.2.3 编译程序的前端和后端 1.3 编译程序和程序设计环境 1.4 编译程序的实现 习题第2章 形式语言与自动机理论基础 2.1 基本概念 2.2 文法 2.2.1 文法的定义 2.2.2 文法分类 2.2.3 推导和归约 2.2.4 语法树与文法二义性 2.2.5 文法等价变换 2.3 有限自动机(FA) 2.3.1 确定有限自动机 2.3.2 非确定有限自动机 2.3.3 DFA与NFA的等价 2.3.4 DFA的化简 2.4 正则表达式 2.4.1 正则表达式与正则集 2.4.2 正则表达式与有限自的相互转换 习题第3章 词法分析 3.1 词法分析介绍 3.1.1 词法分析程序的功能 3.1.2 词法分析程序的接口 3.2 词法分析程序设计 3.2.1 单词分类 3.2.2 单词的内部表示 3.2.3 单词的形式描述 3.2.4 自动机的实现 3.3 词法分析程序的实现 3.3.1 实现词法分析程序应注意的问题 3.3.2 单词结 3.3.3 实现算法 3.4 词法分析程序自动生成 3.4.1 LEX简介 3.4.2 LEX工作原理 3.4.3 LEX源文件结构 3.4.4 LEX系统中的正则式 3.4.5 LEX的使用方式 3.4.6 应用实例 习题第4章 语法分析——自顶向下分析方法 4.1 语法分析程序介绍 4.1.1 语法分析程序的功能 4.1.2 语法错误类别及错误处理 4.1.3 自顶向下语法分析基本思想 4.1.4 3个重要的集合 4.1.5 自顶向下语法分析条件 4.2 递归下降法 4.2.1 递归下降法语法分析原理 4.2.2 递归下降法语法分析程序的构造 4.3 LL(1)分析方法 4.3.1 LL(1)分析法原理 4.3.2 LL(1)分析表的构造 4.3.3 LL(1)驱动程序的构造 4.4 自顶向下分析程序的自动生成 习题第5章 语法分析——自底向上分析方法 5.1 自底向上语法分析方法介绍 5.2 简单优先分析 5.2.1 简单优先文法及其优先关系矩阵的构造 5.2.2 简单优先分析算法 5.3 LR分析法 5.3.1 LR类分析法的工作过程 5.3.2 LR(0)分析方法 5.3.3 SLR(1)分析方法 5.3.4 LR(1)分析方法 5.3.5 LALR(1)分析方法 5.3.6 LR方法小结 5.4 自底向上分析程序的自动生成 习题第6章 语义分析和符号表 6.1 语义分析概述 6.1.1 语义 6.1.2 语义分析的功能 6.1.3 语义分析的一般过程 6.2 符号表的数据结构 6.2.1 标识符的属性 6.2.2 标识符的内部表示 6.2.3 类型的内部表示 6.2.4 值的内部表示 6.3 符号表的管理 6.3.1 符号表的建立与访问 6.3.2 符号表的组织 6.3.3 符号表的局部化处理 6.4 程序设计语言符号表的实例 6.4.1 Pascal的符号表 6.4.2 C的符号表 习题第7章 中间代码生成 7.1 常用的中间代码结构 7.1.1 后缀式 7.1.2 抽象语法树和DAG 7.1.3 三地址中间代码 7.2 语法制导方法概论 7.3 类型检查和类型转换 7.4 中间代码生成中的几个问题 7.4.1 语义信息的获取和保存 7.4.2 语义栈Sem及其操作 7.4.3 常用的语义子程序 7.5 表达式的中间代码生成 7.6 下标变量的中间代码生成 7.6.1 下标变量的地址 7.6.2 下标变量的四元式结构 7.6.3 下标变量的中间代码生成过程 7.6.4 下标变量中间代码生成实例 7.7 赋值语句的中间代码 7.8 过程调用和函数调用的中间代码 7.9 控制语句的中间代码生成 7.9.1 goto语句和标号定位的中间代码 7.9.2 条件语句的中间代码 7.9.3 while语句的中间代码 7.10 过程／函数声明的中间代码生成 习题第8章 中间代码优化 8.1 优化方法概述 8.2 基本块划分 8.3 常量表达式局部优化 8.4 公共表达式局部优化 8.5 循环不变式外提 8.5.1 循环不变式外提概述 8.5.2 循环不变式外提原理 8.6 其他各类优化介绍 习题第9章 运行时存储空间的组织与管理 9.1 目标程序运行时的存储结构 9.1.1 目标程序运行时内存的划分 9.1.2 目标程序运行时的存储分配策略 9.2 过程活动记录和运行时栈 9.2.1 过程活动记录 9.2.2 过程活动记录的申请和释放 9.3 变量访问环境 9.3.1 变量访问环境概述 9.3.2 Display表方法 9.3.3 静态链方法 习题第10章 目标代码生成 10.1 目标代码生成介绍 10.1.1 代码生成器的输入和输出 10.1.2 指令选择 10.2 虚拟机 10.3 寄存器的分配 10.3.1 单寄存器机器的寄存器分配 10.3.2 多寄存器机器的寄存器分配 10.4 四元式到目标代码的翻译 10.4.1 表达式四元式的翻译 10.4.2 赋值语句四元式的翻译 10.4.3 输入输出语句四元式的翻译 10.4.4 条件语句四元式的翻译 10.4.5 循环语句四元式的翻译 10.4.6 标号语句四元式和goto语句四元式的翻译 10.4.7 过程、函数说明语句四元式的翻译 10.4.8 过程和函数调用语句四元式的翻译 习题参考文献

章节摘录

2.语法分析阶段　　语法分析的任务是根据程序设计语言的语法规则，把词法分析的结果分解成各种语法单位，同时检查程序中的语法错误。
语法分析的扫描对象有两种可能：一种是将词法分析程序作为独立的一遍运行，扫描整个源程序的ASCII码序列，将之转换为TOKEN序列，输出到一个中间文件，该文件作为语法分析程序的扫描对象继续编译的过程；更一般的情况是将词法分析程序设计成一个子程序，每当语法分析程序需要读取单词时，则调用该子程序。
这种设计方案中，词法分析程序和语法分析程序处于同一遍，可以省去中间文件。
3.语义分析阶段　　这一阶段的任务是对语法分析所识别出的各类语法范畴，分析其含义，并进行静态语义检查。
例如，变量是否定义、类型是否匹配等。
这一阶段所依循的是语言的语义规则。
通常使用属性文法描述语义规则。
4.中间代码生成　　在进行了上述的语法分析和语义分析阶段的工作后，有些编译程序将源程序变成一种内部表示形式，这种内部表示形式叫做中间代码。
使用中间代码的主要好处是便于移植、便于修改、便于优化。
这种中间代码的形式有很多种，常见的有后缀式（栈式）中间代码、三地址中间代码（三元式和四元式）、图结构中间代码（树，DAG）。
其中，后缀式中间代码是最早使用的一种中间代码，现在很少使用，目前使用的主要是后两种。
5.中间代码优化　　此阶段的任务是对前阶段产生的中间代码在不改变源程序语义的前提下进行加工变换，使生成的代码更为高效，缩短运行时间或节省存储空间。
主要的优化方式包括常量表达式优化、公共子表达式优化、不变表达式的循环外提和削减运算强度等。
6.目标代码生成　　这一阶段的任务是把中间代码变换成特定机器上的机器指令代码或汇编指令代码。
这是编译的最后阶段，因为目标语言的关系而十分依赖于硬件系统。
如何充分利用寄存器、合理选择指令、生成尽可能短而有效的目标代码，都与目标机的结构有关。
生成的目标代码如果是汇编指令代码，则需经由汇编程序处理后才能执行；生成的目标代码如果是绝对指令代码，则可直接投入运行；如果是可重定位的指令代码，那么目标代码只是一个代码模块，必须由连接装配程序将输入／输出模块、标准函数等系统模块与目标代码模块连接在一起，才能形成一个绝对指令代码程序以供执行。
大多数现代实用的编译程序生成的目标代码都是这种可重定位的指令代码。

版权说明