第 32 卷 第 16 期 计 算 机 工 程 2006 年 8 月
Vol.32 № 16 Computer Engineering August 2006
·软件技术与数据库· 文章编号：1000—3428(2006)16—0083—03 文献标识码：A 中图分类号：TP311

一种新型计算机代数系统编译器的设计与实现

覃 安，符红光
（中国科学院成都计算机应用研究所，成都 610041）

摘 要：计算机代数系统（Computer Algebra System，CAS）是集大整数运算和符号运算于一体的计算平台，这方面的代表软件有 Maple 和 Mathematica。然而，在计算机代数系统的设计中，编译器的设计与实现始终是一个关键和难点。GiNaC 是 Linux 平台上开放源码的符号计算包，该文以它为基础提出了一种新型 CAS 编译器的设计方法，并在 Linux 平台上实现。新的 CAS 编译器兼容 Maple 编程语言，对比测试结果显示它的效率并不逊色于 Maple。

关键词：计算机代数系统；虚拟机；编译器

Design and Implementation of a Novel CAS Compiler

QIN An, FU Hongguang
(Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041)

【Abstract】A computer algebra system (CAS) is a software platform that facilitates the symbolic computation and the big number computation. The typical software includes Maple and Mathematica. However, in building CAS, one of difficult problems is the design and implementation of its compiler. GiNaC is a symbolic computation package running on the Linux platform. Based on the package, this paper presents a new design method to implement the CAS compiler, and implements a new CAS compiler on Linux platform. The novel CAS compiler is compatible with Maple's program language. The testing results show its efficiency is not inferior to Maple.

【Key words】 Computer algebra system (CAS); Virtual machine(VM); Compiler

1 概述

随着计算机的进一步普及和人工智能等新兴科研领域的发展，国外许多机构纷纷开始关注运用计算机进行代数运算，由此发展出一批功能强大的计算机代数系统，它们中的典型代表就是 Maple 和 Mathematica。在国内，对于计算机代数系统的研究还处于起步阶段。

计算机代数系统(Computer Algebra System, CAS)是集大整数运算和符号运算于一体的计算平台。实现一个功能强大的 CAS，关键和核心在于其编译器的设计和实现。

在现在的计算机代数系统软件中，对用户的命令处理，大多是采用如图 1 所示的流程。

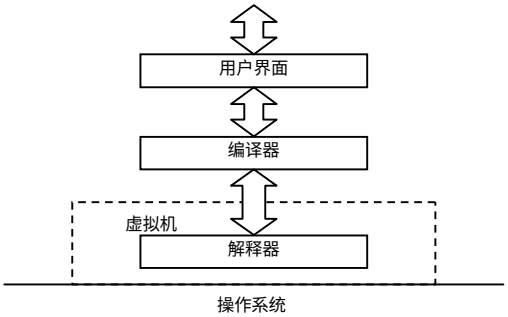


图 1 用户命令处理流程

在整个处理过程中，编译器构成了整个系统的核心，除了负责处理源语言的识别、翻译和相关的检查之外，还要完成大整数处理、符号处理以及内存管理等多项功能，并对每种功能生成对应的目标指令。而处于系统底层、对编译器起支撑作用的解释器和虚拟机往往只被设计为编译器的辅助设施，在系统功能级上处于次要地位。这种设计方式造成了 CAS 编译器规模庞大、结构复杂，使得 CAS 编译器的开发变得异常困难和艰巨，同时整个 CAS 的维护和扩展也受到了限制。

与上述介绍的CAS的编译器的设计相比，本文提出的设计思路在于：编译器并不是把所有的功能都集中在一起实现，它只是负责对源语言进行计算语言学方面的处理，而将符号计算和大整数运算的功能下放到虚拟机上去实现。其中，虚拟机是基于Linux下的GiNaC符号计算包[1] 来设计和实现的，把整个编译器称为新型编译器，以区别于普通的CAS编译器。GiNaC是由德国Christian Bauer等人于 2000 年开发出的一个基于Linux的开放源码的符号计算包，最初被用于进行量子物理计算。

本文以现今流行的计算机代数系统 Maple 的编程语言(简称 Maple 编程语言)为背景，逐步说明新型编译器的设计和实现过程。

2 新型编译器的基本结构

新型编译器可以由 8 个模块组成，分别是：词法分析器，语法分析器，语义分析器，代码生成器，解析器，表格管理，差错管理，主控器。各模块之间的关系如图 2 所示。

各模块的功能如下：

(1)词法分析器：编译器首先要能读入字符，并能产生源语言的最小单位——词，因此需要一个词法分析器。它的主要功能是扫描程序字符，按照语言的词法规则，识别出各类

基金项目：国家“973”计划基金资助项目（2004CB318003）；中国科学院计算技术研究所“百人计划”资助项目
作者简介：覃 安(1980—)，男，硕士生，主研方向：符号计算；符红光，研究员、博导
收稿日期：2005-10-02 **E-mail：**Qin.an@163.com

单词符号并输出，同时进行词法检查。

(2)语法分析器：在词法分析的基础上，按照语法规则，从单词串中识别出各类语法成分，同时进行语法检查。语法分析的目的是为语义分析和代码生成做准备的。

(3)语义分析器：源语言经过语法分析后只能得到一棵符合语法要求的语法树，它所包含的运算信息、类型信息和定义信息只是局部的。而对于一种语言的处理，更重要的是知道它是否符合该语言定义的语义规则。尤其对于类型采用隐式定义和传递的 CAS 编程语言，更需要进行语义上的分析。这个工作是在语义分析器中完成的。语义分析器主要完成类型检查、控制流检查、唯一性检查和定义处理。

(4)代码生成器：如一段源代码在经过上述 3 个模块分析后，都没有发现任何错误，即被认为是有效的。那么就可由代码生成器将它转换成能被虚拟机理解和执行的目标指令。

(5)解析器：解释执行目标指令。

(6)表格管理：提供编译和解释执行过程中存储各种信息的场所，并具有对信息进行修改，检索和查询的管理功能。

(7)出错管理：用户输入的源语言并不能保证每次都是完全正确的。诊查出各类错误并准确定位出错的位置，同时报告错误的性质是出错管理模块的主要功能。它通过一次编译尽可能地找出所有的错误，并具备一定的错误纠正能力。

(8)主控器：主要是为了协助各个处理模块之间的信息交流和控制；同时也作为与外界交互的接口，接收用户信息和反馈处理结果。

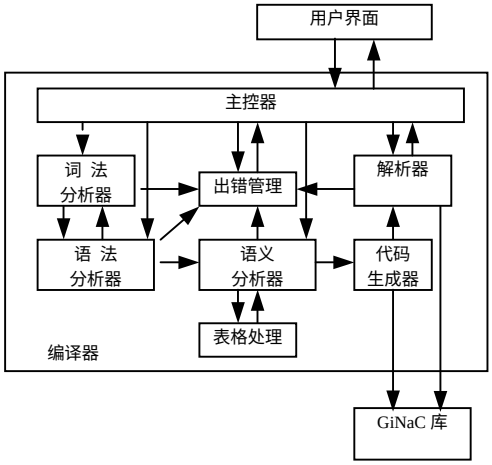


图 2 新型编译器的结构

3 虚拟机的设计

编译器是直接和硬件打交道的，每个编译器都对应一个目标机，而目标机就是编译器所针对的具体机器。

由于不可能为每台具体的机器都设计一个专用的编译器，因此引入了虚拟机的概念。虚拟机是指：通过对具体机器进行一定的模拟和抽象而产生虚拟的目标机。在通常的编译器设计中，采用虚拟机主要是为了提高程序的可移植性；但在新型编译器设计中，主要是为了实现特定的功能，如大整数的表示和符号运算的实现等。

3.1 基本思想

虚拟机的设计是基于以下两个基本思想：

(1)为了源程序的可移植性

编译器依功能可以分为前端和后端。前端主要进行源语言文法处理，后端可以看作目标机的运行。从前端到后端的信息流是源语言结构的处理形式，而从后端到前端则为目标机运行的信息。二者的接口是通过虚拟机来实现的，它能够将源语言中的各种结构映射到虚拟机的伪操作上。使用虚拟机可以使前端与后端相对独立，这意味着如果需要将一个编译器移植到另一台新的机器上，只需根据具体的硬件情况重新开发该编译器的后端，就可针对不同的后端产生不同形式的目标代码。而前端只需作少量的修改甚至不作修改。这样就最大可能地增强了源程序的可移植性。

(2)为了支持特殊的数据结构

新型编译器处理符号计算和大整数计算的能力很大程度上来自虚拟机的支持。在虚拟机中，采用特殊的数据结构来存储和表示那种突破常规计算机字长的整数，并赋予它们加、减、乘、模等基本运算；同时通过重新定义基本数据类型，使它具有封装符号计算的功能。这样，使最小的数据类型都支持符号计算，然后在此基础上实现更复杂的数据类型，显然这些复杂的数据类型也同样支持符号运算。用这样的方式进行设计，使得编译器的前端可以把“符号”作为一种基本数据类型来操作，就像整型、浮点类型一样，而无须考虑它的存储、表示和运算。

上述的第(1)个思想的实现使得相同的 Maple 源代码，即可以在 Windows 下的计算机代数系统 Maple 中运行，也可以在 Linux 下的新型编译器中解释执行。第(2)个思想的实现可以确保编译器前端在整个设计和实现的过程中只关注与语言文法有关的功能，而无须考虑符号和大整数的存储和表示，这样就大大降低了编译器设计的规模和复杂度。同时编译器功能的下放也有利于系统的维护和扩展，使得整个后端更趋向于结构化和模块化。

图 3 展示了新型编译器的层次图，以及和一般 CAS 编译器层次图的对比，从中可以看出二者的编译器和虚拟机在功能设计上的区别。

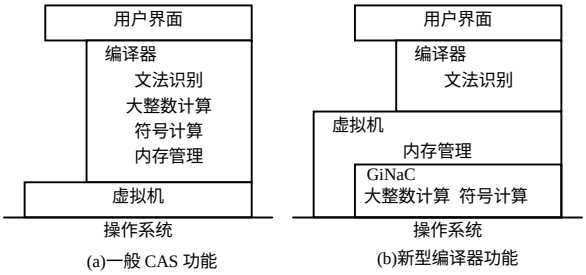


图 3 编译器功能的对比

3.2 基本数据结构

根据上述的设计思想，新型编译器的虚拟机在具体实现上定义了 6 个寄存器和两个存储器，以及 3 种指令格式的指令系统，如图 4 所示。PC 记录的是当前执行的指令序号；IR 存放的是当前执行的指令；T 存放着数据存储器当前工作栈的栈顶位置；BX 存放着当前工作栈的栈底位置(基地址)。指令存储器用来存放生成的虚拟机指令，它体现了源程序对数据的处理过程。数据存储器存放着程序运行过程中的所有数据，包括全局数据、局部数据和临时数据。3 种指令格式是自定义的，分别是二操作数指令、一操作数指令和无操作数指令，采用直接方式和间接寻址两种寻址方式。

新型编译器的虚拟机采用的是栈式管理，以更好地支持程序的递归调用和多层定义。每当进入一个函数块(外层的主程序也可看作一个函数)，它就在数据存储器中建立一块栈单元作为当前工作栈，并将有关的信息填入，这些信息包括：

静态链，动态链，返址，局部参数等。同时初始化局部变量单元和形式参数单元，并修改寄存器 T 与 BX 的值以指向正确的单元位置。

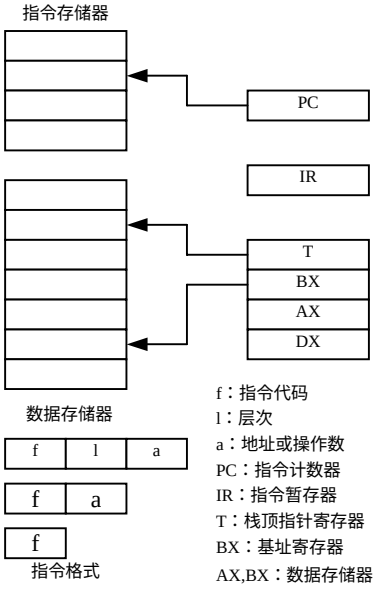


图 4 虚拟机结构

在实现过程中，虚拟机的所有数据单元全部采用经过封装后的类型。把封装大整数表示和运算后的类型定义为 numeric 类型，然后在此 numeric 类型的基础上封装符号计算能力进一步得到 ex 类型，再用此 ex 类型去定义向量、矩阵、列表等比较复杂的类型。这种方法把大整数计算能力和符号计算能力扩展到了各个运算类型，使得各运算类型都支持大整数计算和符号计算。同时，在整个虚拟机运转过程中，可以根据需要实时地判断和转换类型，以保证参与运算的类型相互匹配。

上述虚拟机的构建方式使得从上层编译器的角度来看，它底层是一台可以进行大整数运算和符号运算的目标机，从而实现了 CAS 两大功能的下放和编译器设计复杂性的降低。如图 3 所说明的那样，以 GiNaC 作为基础，在此基础上来建造虚拟机；在虚拟机内部实现系统对大整数和符号计算的处理以及必要的内存管理；然后在此基础上架设用于识别语言文法的编译器和提供友好交互的用户界面。这样就构建了一个完整的计算机代数系统。

4 实现和对比测试

应用上述的方法，我们在 RedHat9 Linux 平台下实现了一个计算机代数系统，它以新型编译器为内核，拥有计算机代数系统的两大特点：大整数运算和符号运算。这里给出两个例子给予说明。

例 1 用 for 循环计算 200 000！的阶乘值。

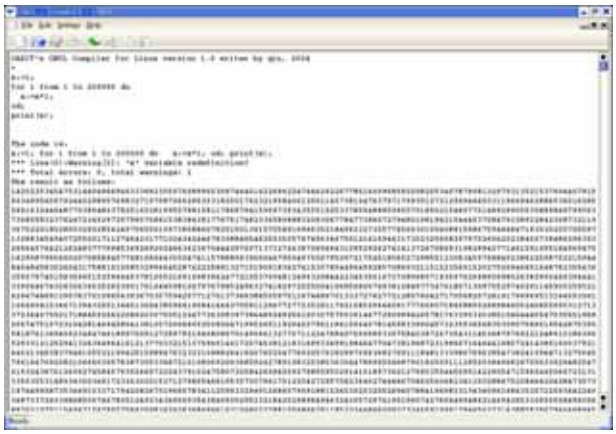
用 Maple 编程语言实现该运算的源代码为

```
x:=1;
for i from 1 to 200000 do
   x:= x*i;
od;
print(x);
```

在 PIII 556 MHz, 128MB RAM 的 IBM PC 机上运行上述代码。200 000！的浮点运算值为

1.420 225 345 470 314 405 1E973 350

经过运算，得到了它 97 多万位整数的精确解，如图 5(a) 所示。整个代码运行所有的时间为 1 127.914s，而 Maple 却需要 1 217.148s，从性能和效率上看已达到了 Maple 的水平，这也显示了新型编译器大整数运算的能力。



(a) for 循环计算 200 000！的值

(b)求两单变元多项式的 GCD

图 5 运行演示

例 2 求两单变元多项式 f 和 g 的最大公因子(GCD)。

假设所输入f和g的各系数是互素的。为了方便实现，用Maple编程语言定义一个求GCD的函数mygcd。它的功能是通过辗转相除法求两个单变元多项式f和g关于变元x的GCD，源代码如下：

```
mygcd := proc (f,g,x)
   local r0,r1,r2;
   r0 := f;
   r1 := g;
   while r1 <> 0 do
      r2 := rem(r0,r1,x);
      r0 := r1;
      r1 := expand(r2);
   od;
   return r0;
end;
```

这里，用两个多项式：

```
p:=  expand((24*x^12+2333*x^7+247)^17*((7*x^70+2547*x^19-325)*(15489*x-1)^20)^3*(2391*x^19-676*x^17+654)^8)
q:=  expand((24*x^12+2333*x^7+247)^11*((7*x^70+2547*x^19-325)^2*(15489*x-1))*(2391*x^19-676*x^17+654)^5)
```

为例子，调用上面的 mygcd 函数来求它们的 GCD：

```
r:= mygcd (p,q,x);
print(r);
```

(下转第 91 页)