国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第12卷第2期 1990年6月 Vol.12 No.2

顺序PROLOG机KD-PP的系统结构和硬件实现技术

刘滨海 闻 烽 王剑琪 张晨曦
(计算机系)

摘 要 本文提出的KD-PP系统是一种基于编译技术的顺序PROLOG推理系统，该系统的设计为逻辑型程序语言PROLOG的实现提供了硬件支持，因而能高效地执行PROLOG程序。本文从数据表示、存储系统、机器状态和指令系统等方面全面地介绍了顺序PROLOG机KD-PP的系统结构和硬件实现技术。

关键词 系统结构，指令系统，存储器，PROLOG系统，数据表示
分类号 TP303，TP302.2

顺序PROLOG已发展成为一种实用的逻辑型程序设计语言，其实现效率已成为当今一个主要的技术研究课题。我们认为提高实现效率应基于编译技术。文献[1]在研究WAM的基础上，提出了一个编译型顺序PROLOG推理系统。我们对其作了进一步修改与扩充，提出KD-PP系统。该系统不仅实现了WAM的功能，而且实现了扩充的非逻辑型的内部谓词，从而提高了KD-PP系统的可用性。

本文从数据表示、存储系统、机器状态和指令系统等方面介绍KD-PP系统的系统结构，然后介绍硬件实现技术。

1 KD-PP的系统结构

1.1 系统配置

KD-PP系统采用主辅机结构。这种结构可以利用主控机现有的软硬件资源，系统研制工作量较小，性能价格比较高。如果适当修改接口，辅处理机PP（PROLOG处理机）将可以联接到多种主控机，从而获得较广的应用。另外，PP的研制，能为以后并行PROLOG推理系统的研究工作提供基础。KD-PP系统配置如图1.

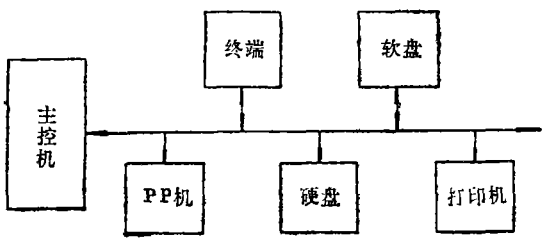

图 1 KD-PP的系统配置

主控机将用户程序编译为可由

1989年6月25日收稿

PP处理的目标代码，PP采用微程序技术解释目标代码。主控机执行非逻辑成份中需要传统执行机制支持的库管理等非逻辑成份；主控机还提供用户界面和程序设计环境、输入输出以及PP加载等控制。

KD-PP系统用户语言选用GKD-PROLOG文本，系统语言是WAM指令系统的扩充和修改。

1.2 系统结构

1.2.1 数据表示

采用带标志的数据表示形式，字长32位，其中标志域8位。KD-PP基本数据类型包括：常数、变量、结构和表。

1.2.2 存储空间

KD-PP采用分布式存储系统，由PROLOG数据库区（程序存储区）和PROLOG工作区（数据存储区）组成。

PROLOG数据库区放置在主控机的存储系统中，用于存储PROLOG源程序和编译后的目标代码，并存储各类表格。

PROLOG工作区设置在PP中，字长32位。PROLOG工作区是一种多栈结构，由全局工作区，全局栈HEAP，局部栈STACK，跟踪栈TRAIL构成。其中全局工作区用于支持算术表达式求解，为主控机和PP的同步通讯提供缓冲区。关于HEAP, STACK, TRAIL, PDL的功能见文献[1], [2]。

图 2　PROLOG工作区结构

1.2.3 机器状态

PP机器状态由一组寄存器定义，包括当前指令指针P，后继指令指针CP，最近选择点指针B，最近环境指针E，HEAP栈顶H，HEAP回溯点HB，结构指针S，TRAIL指针TR，PDL栈顶指针PDL，全局工作区指针T，STACK栈顶指针A以及支持CUT实现的后备B寄存器CT，另外还有若干变元寄存器XA.

PROLOG程序执行过程中，回溯是不可避免的。回溯要求机器能恢复到先前某个确定状态，该状态是通过在STACK中存放的选择点和环境来保留的。

1.2.4 指令系统

KD-PP指令系统分为两大类：一是支持PROLOG中逻辑成份执行的类WAM指令；二是支持PROLOG中内部谓词等非逻辑成分执行的非WAM指令。

1.2.4.1 类WAM指令

KD-PP的类WAM指令是WAM指令集[2]的扩充与修改[1]，主要变动有如下四方面。

(1) 常数索引指令和结构索引指令

switch_on_constant1 N　　switch_on_structure1 N

switch_on_constant2　　switch_on_structure2

suitch_on_constant1 (structure1)与switch_on_constant2 (structure2)的不同之处仅在于对应的HASH表结构不同，HASH表结构取决于它的项数。前者对应HASH表项

数小于某个定值(如20)的情况,HASH表结构为<key, pointer>, pointer指向对应子句;后者对应HASH表项数大于等于该定值的情况,HASH表构造为层次结构,第一层是<jump, pointer>结构,pointer指向第二层HASH表,第二层为<key, pointer>结构。这种修改既避免了建立不必要的选择点,又避免了处理过程中的断流现象。

(2) 动态过程选择点的建立、修改和撤销指令

ctry-me-else La, L　　trust-fail

为了支持实现一致的动态代码语义,增设动态过程的选择点建立和修改指令ctry-me-else及撤销指令trust-fail. ctry-me-else总是作为动态过程中各子句的第一条指令出现,但只有出现在第一个子句之前的ctry-me-else是建立选择点,其余是修改选择点。trust-fail是动态过程目标代码的最后一条指令,负责撤销选择点并引起回溯。

(3) 非直接过程调用指令

call proc/arity, N　　execute proc/arity

ncall proc, N　　nexecute proc

KD-PP系统动态过程的编译采用执行驱动编译策略,因而过程调用之前,首先要判断该过程是否已被编译,而不能直接转移到该过程,这就需要由非直接过程调用命令实现。

(4) 动态过程执行驱动编译指令

compile

如果调用到一个尚未被编译的过程,则执行compile指令。compile启动编译器,对被调用过程进行编译,编译结束后方可转入过程继续执行。

1.2.4.2 非WAM指令

(1) 内部谓词指令

KD-PP系统设置了丰富的内部谓词,实现这些内部谓词有三种方法:

a 专用内部谓词指令。为使用频度较高且容易实现的内部谓词设置了专用指令,支持其快速实现。例如,参数测试类、项的比较类等内部谓词均采用这种方法。

b 通用接口指令。

escape pred　　escape-wait pred

pred是内部谓词表征参数。对于一些需要借助于主控机才能实现的内部谓词,例如数据库操作类、输入输出类等,编译为通用接口指令,PP执行通用接口指令时,请求主控机提供服务。

escape与escape-wait指令的区别在于前者的执行不会影响其后的目标代码,而后者则不然。

c 还有一类内部谓词被编译为其它指令。例如为CUT谓词的实现设置了如下指令:

save-CUTB Yi　　save-B Yi

restore-B-CUTB　　restore-B-from Yi

这四条指令能为CUT谓词(包括折取目标CUT)提供完备的支持。

(2) 面向数值计算的指令

KD-PP系统设置了实现整数、实数运算指令：

add	subtract	multiply
devide	eval(Vi)	result(Vi)

KD-PP 系统中结构型算术表达式求解采用部分编译/部分解释策略。编译器将算术表达式编译为一串数值计算指令，其中的结构型算术子表达式被编译为eval(Vi)指令，该指令调用微程序解释求解Vi，result(Vi)实现Vi的传送与比较。

2 硬件实现技术

后端顺序PROLOG处理机 PP，负责高效执行由主控机提供的目标代码。PP 机的结构组织如图 3 所示。

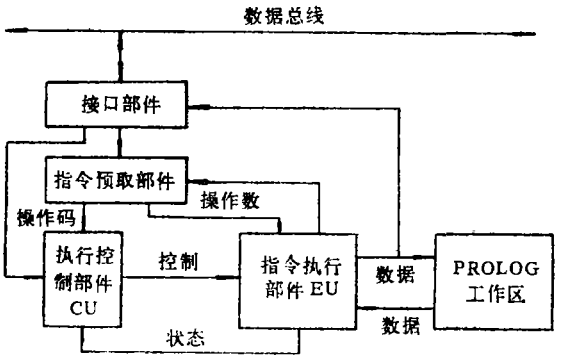

图 3　PP的组织结构

2.1 接口和指令预取部件(IFIPU)

IFIPU通过主控机外总线从目标代码库中取出指令进行组装，并对某些指令进行预处理或部分预处理，并将组装好的指令装入指令缓冲栈。IFIPU框图见图 4 .

接口部件由接口硬件逻辑与软件驱动器组成，主控机通过接口部件实施对 PP 的加载、信息通讯等

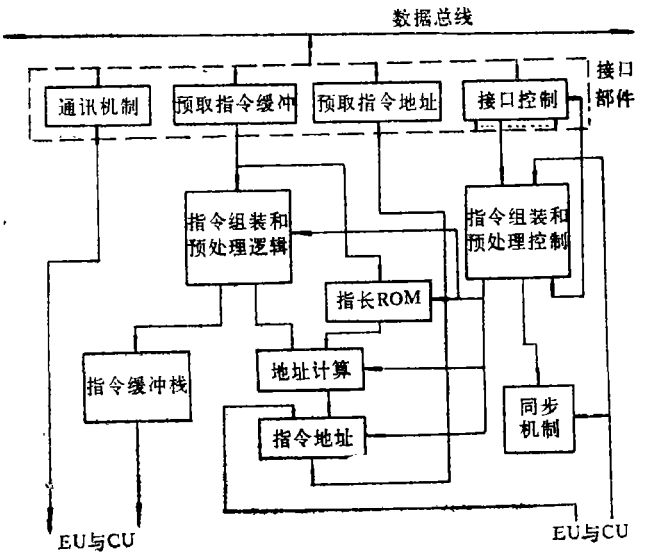

图 4　IFIPU框图

控制；此外，它还为联机调试提供硬件和软件支持。

IFIPU能对某些指令进行部分或完全预处理。完全预处理的有 switch-on-term 指令等，部分预处理的有call等非直接过程调用指令。

指令组装采用移位网络方案，用一个计数器记录组装寄存器中的剩余字节数，该字节数还控制新来指令字的写入位置。每取走一条指令就把组装寄存器左移相应的字节数，指令在进入缓冲器之前，在组装寄存器中总是左对齐的。组装控制及预处理控制均采用微程序技术；为了实现IFIPU和EU对指令缓冲栈的正确访问，并保证预处理的正确性，在IFIPU与EU之间设置同步机制。

2.2 指令执行部件(EU)和执行控制部件(CU)

CU采用微程序技术，实现当前微指令的执行与下条微指令地址计算重迭进行。CU从指令缓冲栈中读取指令操作码，译码之后得到微程序入口地址，然后从微程序控存中读取微指令，提供EU的全部控制信号，完成相应功能。指令译码逻辑是一个快速映象表，其输入是指令码，输出是微程序入口地址。在指令处理过程中，往往会出现根据某些条件进行转移的微指令。这时，状态选择测试逻辑从若干条件中选择某一条件进行测试，并将测试结果送到后继微地址生成逻辑，从而实现条件转移等改变微指令流程的控制功能。后继微地址生成逻辑负责从多个微地址源中选择生成正确后继微地址，该逻辑提供顺序执行，条件或无条件子程序调用、条件或无条件转移及递归调用等功能。

EU采用微程序技术解释KD-PP的指令系统。

EU的数据通路采用三总线结构，可以实现源操作数和结果的并行传输，以便在不采用流水线技术的情况下，获得尽可能高的速度。EU设置标志处理逻辑，包括标志修正器，标志比较器。标志修正器在将UNB型数据写进变元寄存器时，自动将UNB标志修改为REF标志；标志比较器用于判别某数据类型或是比较两个PROLOG数据项类型是否相等。

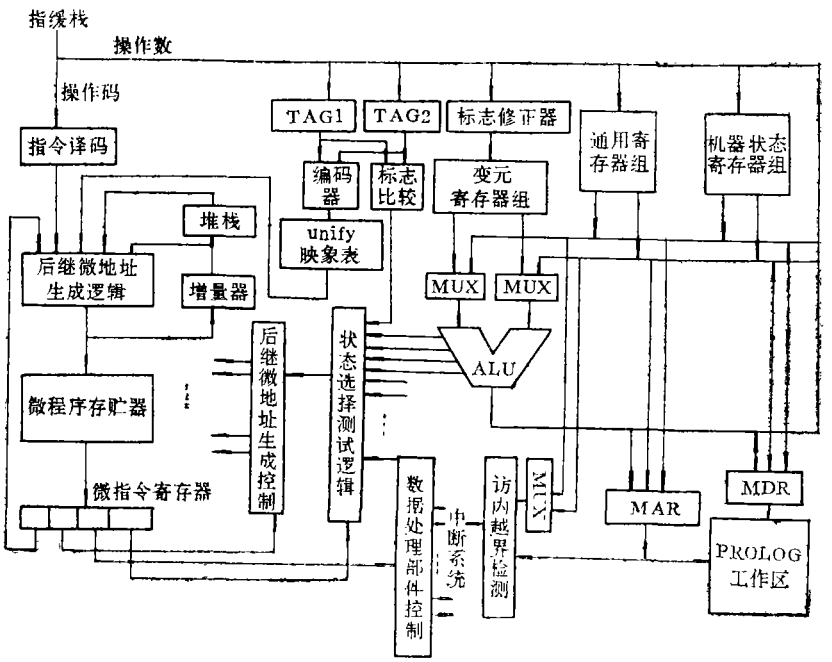

图 5 指令处理部件（CU和EU）逻辑框图

unify部件将参与unify的两个数据项进行判断、比较并编码，输出相应处理的微程序始地址。这些操作由组合逻辑实现，在一个微周期内完成，从而快速实现unify操作。

dereference部件支持dereference的快速实现。类型判别逻辑识别数据项类型是否为REF，根据识别结果，控制逻辑实施对dereference操作的控制。在一个微周期内完成REF链中的一次dereference操作。EU与CU逻辑框图见图5，dereference硬件控制逻辑框图见图6.

存储边界检查器的设置，实现了存储边界检查与指令操作的并发执行。如果访问越界，则发出中新信号。

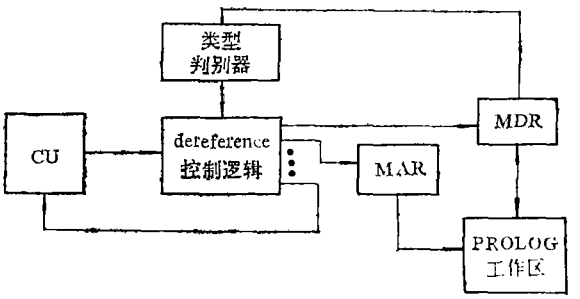

图6　dereference硬件控制逻辑框图

2.3　中断系统

向量优先中断控制器，对多个中断请求进行识别判优，发出服务请求，并将优先级最高的中断的中断向量送入中断向量寄存器。

中断有两种处理方法：一是中断处理需要借助主控机，主控机响应中断请求之后，从中断向量寄存器取出中断向量，进行识别，然后进行中断处理；二是由PP处理的中断，中断控制器在接收到某部件的中断请求后，从中断服务程序入口地址对照表中获得地址，转入服务程序进行处理。

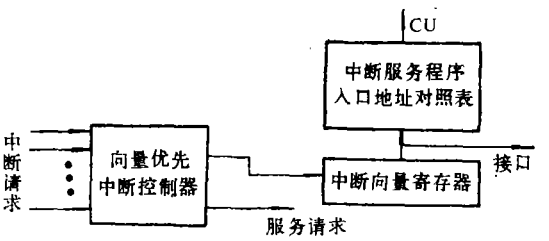

图7　中断系统逻辑框图

2.4　微程序设计

PP中的微指令采用水平型设计，主要控制字段有：

- EU与IFIPU同步控制
- 数据处理部件控制
- unify硬件控制
- 状态选择测试逻辑控制
- 数据通路接口控制
- 中断系统控制
- 微周期控制
- CU控制
- 标志处理部件控制
- dereference硬件控制
- 机器状态寄存器控制
- 后继微地址
- 存储访问控制

微指令提供的控制信号，实现对整个系统的控制。值得注意的是其中的微周期控制字段，该字段实现在确定的主频时钟的情况下，根据具体的微操作，控制产生所需要的微周期，从而避免需要较长时间的微操作对整个系统微周期的影响。

3　结　束　语

KD-PP具有如下特点：(1) 采用主辅机结构；(2) 具有丰富的指令系统；(3) 采用带标志的数据表示；(4) 分布式存储系统；(5) 采用微程序控制技术；(6) 采用多方面

的优化技术；(7) 设置指令预取部件，实现取指和执行指令重迭进行，并预处理某些指令。

作为研制推理机的一个开端，该系统的速度不是很高。按目前PP机微指令周期为200ns计算，确定性append程序推理速度约为150KLIPS～200KLIPS。虽然因条件限制该系统尚未能实现，但其设计方案对于进步的PROLOG机研究和实现有很大的参考价值。

鸣 谢

在此课题研究中，得到胡守仁教授的指导和帮助，谨致谢意。

参 考 文 献

[1] 张晨曦.基于WARREN抽象机的PROLOG实现技术的研究. 博士学位论文，国防科技大学计算机系，1987

[2] Warren, D H D. An Abstract Prolog Instruction Set. Technical Note 309, SRI International Oct.,1983

[3] Tick,E. An Overlapped Prolog Processor.Technical Note 308, SRI International Oct.,1983

[4] Tick E, Warren D H D. Towards a Pipelined Prolog Processor. New Generation Computer, 1984, (2)

[5] D.P.西维奥雷克，C.G.贝尔，A.纽厄尔著，陈炳从等译. 计算机系统结构：原理与实例（上册）. 科学出版社，1988

[6] John Mick, James Brick. Bit-slice Microprocessor Design. New York, McGraw-Hill Book Company,1980

Architecture and Hardware Implementation Techniques of the Sequential PROLOG Machine: KD-PP

Liu Binhai Wen Feng Wang Jianqi Zhang Chenxi
(Department of Computer science)

Abstract

This paper describes a sequential PROLOG inference processor: KD-PP which is based on compilation techniques. The processor has incorporated hardware mechanisms in it for logic programming language PROLOG execution. So it can execute PROLOG programs at high speed. The architecture of the processor, including data format, memory, machine states, instruction set, and the hardware implementing techniques are described in detal.

Key words: system architecture, instruction set, memory, PROLOG system, data format