国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第12卷第4期 1990年12月 Vol.12 No.4

智能工具机的性能分析*

王 朴 张晨曦 朱海滨

(计算机系)

摘 要 本文论述了提高智能工具机性能所采取的技术措施,并对其性能进行了模拟。模拟结果表明,智能工具机的平均推理速度可达500KLIPS左右,比日本的PSI—Ⅱ约快一倍,比美国加州大学Berkeley分校研制的PLM约快两倍。

关键词 模拟系统,计算机系统,人工智能/智能工具机,顺序PROLOG, 模拟系统,推理速度

分类号 TP302.7, TP387

智能工具机是为支持智能计算机研究和人工智能应用而研制的一种面向扩充的PROLOG语言的计算机。设计的主要目标是既能支持高速的数值计算,又能支持高速逻辑推理。数值计算速度平均达17MIPS. 本文主要对智能工具机的逻辑推理速度进行分析。为此,选择了若干典型的测试程序,经过手工和利用顺序推理机模拟实验系统进行模拟,表明智能工具机平均具有500KLIPS左右的推理速度。

1 系统构成

智能工具机采用主——辅机结构,由前端机和后端机两大部分组成。前端机为一台通用的计算机,负责向用户提供方便、良好的智能软件开发环境,完成程序的修改与调试、文件的编辑、输入/输出以及对后端机的加载、控制等工作。后端机(称为PP)是一台面向扩充了的逻辑程序设计语言的高性能Prolog处理机。它由以下几部分构成:

(1) 通用的高性能RISC CPU芯片;

(2) 高速数据缓冲部件DBU,该部件不仅对数据进行缓冲,而且能直接进行Prolog中的dereference操作;

(3) 一致化部件UU,专门进行Prolog中的一致化操作;

(4) 指令Cache;

(5) 大容量主存;

(6) 通道控制器,用于实现与前端机的接口;

* 本项研究得到国家八六三计划的资助
1990年1月14日收稿

(7) 浮点处理机（任选）。

关于CPU，我们分析了多种实现途径，其特点如表1所示。

表1

所用器件	最高性能（估计）	规模	专用/通用性
1. F100K ECL器件	500～1000KLIPS	庞大	专用
2. 位片器件（例如：Am29300系列）	400KLIPS	中等	专用
3. Prolog芯片	1MLIPS	小	专用
4. 通用高性能RISC芯片	平均500KLIPS	小	通用

表中Prolog芯片这一途径因受国内VLSI技术的限制，无法实现，为了使工具机既能高速地进行推理，又能快速地执行传统语言程序，使之成为一个通用的智能工具机，而且规模较小，我们决定采用通用高性能RISC CPU芯片，并辅之以专用的支持Prolog硬件。

PP的字长为32位，采用带标志的数据表示。每个字包含6位标志域和26位值域。标志域表示数据类型，值域表示数据值或指针。PP中有11种数据类型（见图1），其

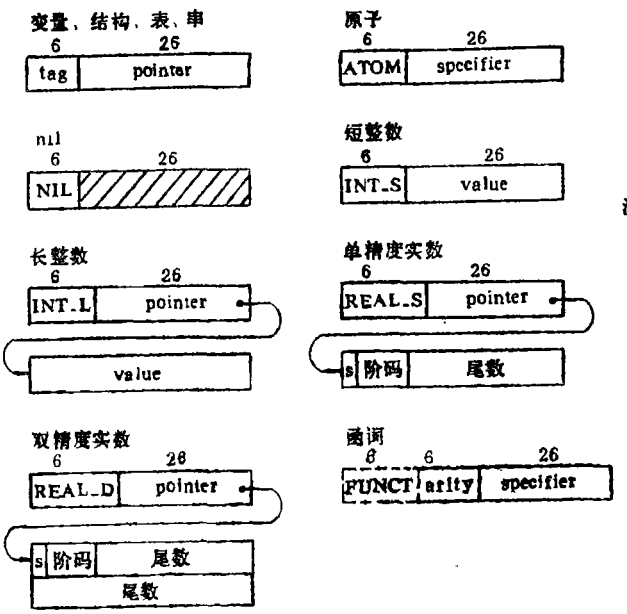


图1 基本数据类型的格式

注：S——符号位；
specifier——指向符号表的指针；
value——值；
arity——秩；
pointer——指向局部栈或全局栈的指针

中变量中的未约束变量与引用变量作为同一类型，其区别是未约束变量的指针指向自身，引用变量的指针指向被引用的量（图2）。

PP的指令字长为32位，8位操作符，3个8位操作数地址，或者1个8位操作数地址和1个16位直接数。操作数地址是面向通用寄存器组编址。

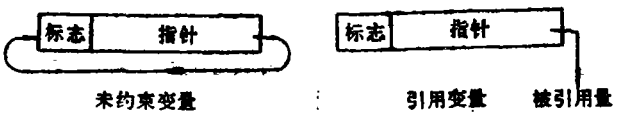


图 2　PP中的变量表示

2　性能模拟

2.1　WAM—PLUS—SES模拟系统简介

WAM—PLUS—SES是一个在VAX—11/780上用C语言开发的Prolog抽象机模拟系统。它是在扩充Warren抽象机(WAM)的基础上实现的。这是一个指令级模拟系统，能执行类Warren代码。该系统提供了良好的用户界面，具有功能强、使用方便的特点。由于我们的工具机将采用WAM作为基础（以类Warren代码为中间语言），所以可以利用该系统进行性能模拟。

WAM—PLUS—SES由以下几部分构成：

(1) 交互式命令系统；(2) 汇编/装入程序；(3) 反汇编程序；(4)输入输出程序；(5) 指令执行程序；(6) 内部谓词执行机制；(7) 代码库管理程序；(8) 源程序库管理程序；(9) 数据统计程序。

2.2　类Warren指令和基本操作在工具机中的执行时间

在工具机中，每一条类Warren指令翻译成一段CPU指令序列。为了进行性能模拟，我们根据各条指令和基本操作的CPU指令序列，用手工计算出它们在各种情况下的执行时间。结果如下（时间单位：节拍）：

2.2.1　unify指令

(1) unify_void N

“读”方式	“写”方式
1	2N

(2) unify_nil

“读”方式			“写”方式
约　束	成功	失败	
6.5+T(trailing)	7.5	9.5+T(fail)	2

表中　(i) “约束”指进行binding操作的情况；

(ii) T(trailing) 为进行trailing操作所需的时间，见2.2.7节。

(iii) T(fail) 为进行fail操作所需的时间，见2.2.7节。

(3) unify_constant C

"读"方式				
约 束	deref(*S)和C都为原子或短整数		其 它	"写"方式
	成功	失 败		
7.5+T(trailing)	7.5	8.5+T(fail)	9.5+T(unify)	4

表中 (i) *S表示指针S所指单元的内容;

(ii) deref(X)表示对X进行dereference操作的结果;

(iii) T(unify)为进行unify操作所需的时间。

(4) unify_variable Vi

	方 式	
Vi	"读"	"写"
Xi	2	3
Yi	4	4

(5) unify_Value Vi

	方 式	
Vi	"读"	"写"
Xi	7+T(unify)	2
Yi	7+T(unify)	4

(6) unify_unsafe_value Vi

方式	Vi	deref(Vi)		
		非未约束变量	未约束变量	
			在局部栈	不在局部栈
"写"	Xi	6.5	14.5	7.5
	Yi	7.5	14.5	8.5
"读"	Vi	7+T(unify)		

2.2.2 get指令

(1) get_nil Ai

(Ai). tag≠'var'		deref(Ai)为非未约束变量		deref(Vi)为未约束变量
成功	失 败	成功	失 败	
5.5	5.5+T(fail)	7.5	7.5+T(fail)	7.5+T(trailing)

表中 (Ai). tag表示Ai中数据字的tag

(2) get_constant C, Ai

(Ai).tag≠'var'			(Ai).tag='var'			
Ai和C都为短整数或原子		其　它	deref(Ai)	deref(Ai)和C都是原子或短整数		其　它
成功	失败		为未约束变量	成功	失败	
6	7	8+T(unify)	6.5+T(trailing)	6.5	7.5+T(fail)	8.5+T(unify)

(3) get_structure F, Ai

(Ai).tag≠'var'			(Ai).tag='var'			
成功	失　败		deref(Ai)	deref(Ai)不是未约束变量		
	tag≠'str'	函词不同	为未约束变量	成功	失　败	
					tag≠'str'	函词不同
11	4.5+T(fail)	10.5+T(fail)	11.5+T(trailing)	11.5	8.5+T(fail)	14.5+T(fail)

(4) get_list Ai

(Ai)tag≠'var'		(Ai).tag='var'		
成功	失败	deref(Ai)为未约束变量	deref(Ai)为表(成功)	deref(Ai)不为表(失败)
5	4	7.5+T(trailing)	10.5	9.5+T (fail)

(5) get_value Vi, Aj　6+T(unify)

(6) get_variable Vi, Aj

Xi	Yi
1	2

2.2.3 put指令

(1) put_nil Aj 1拍；(2) put_constant C, Aj 2拍；(3) put_list Aj 1拍；(4) put_structure F, Aj 5拍；(5) put_unsafe_value Yi, Aj.

deref(Yi)是未约束变量			deref(Yi)不是未约束变量
在当前环境中		不在当前环境中	
有trailing	无trailing		
13.5	11.5	6.5	4.5

(6) put_value Vi, Aj

Xi	Yi
1	2.5

(7) put_variable Vi, Aj

Xi	Yi
2	4

2.2.4 控制指令

(1) allocate

局部栈栈顶为选择点	栈顶为环境
11	10

(2) deallocate 6拍；(3) call P/a, n 7拍；(4) excute P/a 4拍；(5) proceed 1拍。

2.2.5 选择点指令

(1) try_me_else L

局部栈栈顶为选择点	栈顶为环境
22+NA	20+NA

表中：NA为调用目标的变元个数

(2) retry_me_else L 4拍；(3) trust_me_else fail 5拍；(4) try L.

局部栈栈顶为选择点	栈顶为环境
25+NA	23+NA

(5) retry L 8拍；(6) trust L 9拍。

2.2.6 多路转移指令

(1) switch_on_termerm Lv, Lc, Ll, Ls

(Ai).tag	变量	常数	结构	表
时间	6	9	8	10

(2) switch_on_constant和switch_on_structure

不查子表（无冲突）		查子表（冲突）	
成 功	失 败	成 功	失 败
15+T(hash)	12+T(hash)+T(fail)	15+T(hash)+8(k-1)	13+T(hash)+8(K-1)

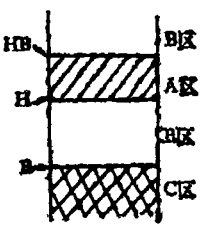
表中：(i) T(hash)为计算hash函数的时间；(ii) K为在子表中查找的项数。

2.2.7 基本操作

(1) fail：17+NA+6K，其中NA为变元数，K为需恢复为未约束变量的单元数

(2) trailing(X)

X所在的区	A	B	C
时间	6	7	5



(3) unify(T1, T2)

T1和T2至少有一个是未约束变量	T1和T2都不是未约束变量								
	都是原子或短整数		都是结构			都是表			
	成功	失败	函词相同		函词不同	成功	失败	失败	其它
			成功	失败					
10+ T(trailing)	9	6+ T(fail)	31+ 9N+ tu1+ …+ tuN	18+ 9i+ tu1+ …+ tui+ T(fail)	13+ T(fail)	25+ 9N+ tu1+ +… tuN	12+ 9i+ tu1+ …+ tui+ T(fail)	6+ T(fail)	未考虑

表中：(i) N为结构的变元数或表中元素的个数（这里采用cdr编码法表示表）；

(ii) tu_j为第j个项的一致化时间，其计算方法同上（递归定义）；

(iii) 在一致化失败后，假设失败发生在第i个变元的一致化过程中。

2.2.8 部分内部谓词

(1) var(X), nonvar (X)　3.5拍；(2) atom(X), list(X), structure(X) 4.5拍；(3) integer(X), real(X), string(X)　4.5或6.5拍；(平均：5.5)。

(4) 简单计算类

A0为未约束变量：9.5+T (trailing)

A0不是未约变量：成功：9.5

失败：9.5+T(fail)

(5) 比较数值类　7拍

2.3 模拟结果

我们在WAM—PLUS—SES上运行了表连接(append 30)、简单求反表(nrev 30)、快速分类(qsort 50)、八皇后(8 queens)、梵塔(8 hanoi)、排序(ser)、查询(query)等典型程序。运行结果及其与日本的顺序推理机PSI—Ⅱ、美国加州大学Berkeley分校的prolog处理机PLM和GKD—PROLOG解释系统的比较如下表所示。

(1) 与PSI—Ⅱ比较

典型程序	PSI—II		工具机			(1)/(2)
	时间(1) (msec)	推理速度 (KLIPS)	时钟周期数 (Cycle)	时间(2) (msec)	推理速度 (KLIPS)	
append30	0.075	400.0	53520	0.0535	560.7	1.40
nrev 30	1.53	324.8	21792	0.872	570.0	1.75
qsort 50	3.86	158.0	37118	1.485	410.7	2.60
8 queens	29.6	194.2	309407	12.376	464.5	2.39
平 均		269.3			501.5	1.9

工具机比PSI—Ⅱ平均快约1倍。

(2) 与Berkeley PLM的比较

典型程序	Berkeley PLM时间(1) (msec)	工具机 时间周期数	工具机 时间 (2) (msec)	(1)/(2)
nrev 30	2.66	21792	0.872	3.05
qsort 50	5.50	37118	1.485	3.70
ser	3.19	24375	0.975	3.27
query	17.57	167790	6.712	2.62
平 均	7.23		2.51	2.9

工具机比Berkeley PLM平均快2倍。

(3) 与GKD—PROLOG比较

典型程序	GKD—PROLOG执行时间(1) (msec)	工具机的执行时间(2) (msec)	(1)/(2)
append 30	20	0.054	370
8 queens	7420	12.376	600
nrev 30	280	0.872	321
qsort 50	560	1.485	377
ser	340	0.975	349
query	4560	6.712	679
8 hanoi	700	2.64	265
平 均	1983	3.588	553

3 提高逻辑推理速度的措施

同传统的程序设计语言一样，编译型PROLOG程序的执行时间是下列三部分之乘积：

(1) 执行的指令条数；

(2) 每条指令执行所需要的时钟周期数；

(3) 时钟周期。

因此，要减少编译型PROLOG程序的执行时间，必须从这三个方面着手。

为了减少程序中执行的指令条数，我们一方面对PROLOG的编译采取许多优化技术，如采用部分计算技术对程序进行预处理；采用任一变元索引技术，减少执行中的不确定性；尽量减少不必要的进栈和退栈操作等。另一方面，我们在系统设计中也采用了如下措施：

(1) 增加dereference指令。dereference操作是编译型PROLOG程序执行的基本操作，每条get和unify抽象指令，都要执行dereference操作，一般都采取循环执行的办法。即：

```
L: Load ra, ra, rb      读取数据
   Nop                  空操作
```

```
Jmpt rc. L          若是引用变量转L
Nop                 空操作
 ⋮
```

其中Nop表示空操作，说明Load和Jmpt指令都需要两个时钟周期执行时间。设 dereference的深度为N，则一般执行一次 dereference 操作需要 4 （N+1）条指令，而增加dereference指令以后，只需一条指令，执行速度提高一倍。

（2）约束变量和引用变量采用同一标志，省去了某些修改标志和拼接操作指令。

（3）采用专用的一致化部件 UU，使绝大多数一致化操作只需四条指令和六个时钟周期就可完成。对于少数复合项的一致化操作，也大大减少了进行多路转移需要的指令条数。

（4）采用优化编码，使常用的标志码能迅速地被检测，如变量标志采用'000000'编码后，就可以直接判变量实现转移，而不需“取标志”、“比较”等指令。

在第二部分的模拟中，未考虑编译优化技术所带来的加速。

为了减少每条指令所需的时间周期数，我们采用RISC技术实现Warren抽象指令，同时，充分利用流水线技术，保证对于绝大多数指令，每一个时钟周期能执行一条指令。为了保证CPU的最大指令流和数据流，设计了指令 Cache 和数据 Cache，它们在流水线方式下访问时，分别能每拍提供一条指令或数据。在随机访问时，能每两拍提供一条指令或一个数据。

为使每条指令所需的时钟周期数最少，又要有较短的时钟周期，我们经过多方调研，决定选用其工作频率为25MHZ的通用CPU芯片。从而为高速逻辑推理和数值计算奠定了基础。

4 结　论

通过手工和模拟系统对智能工具机的逻辑推理模拟表明，智能工具机的逻辑推理速度平均为500KLIPS左右，是PSI—Ⅱ的两倍，是PLM的三倍。若考虑编译器的优化技术所带来的效益，其推理速度将更高。由此，我们认为：

（1）智能工具机采用现有的RISC芯片。利用简单指令等实现Warren抽象指令，能够达到较高的速度，而设计却被大大的简化。

（2）智能工具机的结构设计合理。在通用CPU芯片的基础上，增加少量专用逻辑，使系统既能支持高速数值计算（平均17MIPS），又能支持高速逻辑推理。

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，1987

[2] 王朴，张晨曦，胡守仁.智能工具机系统结构.国防科技大学学报，1990，12(4)

[3] 张晨曦.Prolog抽象机模拟系统WAM—PLUS—SES.小型微型计算机系统，1988，11(9)

[4] Nakashima H and Nakajima K. Hardware Architecture of the Sequential Inference Machine: PSI—II. IEEE, 1987: 104～113

[5] Dobry T P, et al. Performance Studies of a PROLOG Machine Architecture. Proc. of the 12th Int'l Symp.on Computer Architecture, 1985

国防科技大学学报
JOURNAL OF NATIONAL UNIFERSITY OF DEFENSE TECHNOLOGY
第12卷第4期 1990年12月 Vol.12 No.4

管理信息系统通用工具软件MT-1

秦 晓

(计算机系)

摘 要 MT-1是在汉字dBASE-Ⅱ的基础上开发的数据库工具软件，它具有图形化的数据描述和数据操纵语言，是一个实用的微机管理信息系统开发工具。它在图形化用户界面、数据字典、视图、数据完整性和安全性等方面优越于dBASE-Ⅱ。本文描述了MT-1的总体结构和主要功能，讨论了主要设计和实现问题。

关键词 管理系统，数据库/数据管理，信息

分类号 TP315, TP311.56

数据管理是管理信息系统(MIS)的基本组成部分，是MIS开发者面临的共性问题。为了提高数据管理能力，国内引进和开发了一些支持MIS的微型计算机关系数据库系

1989年7月15日收稿

[6] Yamaguchi S, et al. Architecture of High Performance Integrated Prolog Processor IPP. Proc. of Fall Joint Computer Conf., Dallas, IEEE Society Press, 1987

Performance Analysis of an Intelligent Machine

Wang Pu Zhang Chenxi Zhu Haibin

(Department of Computer Science)

Abstract

Some techniques used to enhance the performance of an intelligent machine is described. The performance of the machine is simulated. Simulation results show that the average inference speed of an intelligent machine is about 500 KLIPS, which is twice as fast as Japan's PSI—Ⅱ and 2 times faster than PLM developed by Berkeleg College of California Univ. in U.S.A

Key words analog system, computer system, artificial intelligence/intelligent machine, sequential PROLOG, inference speed