PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT.

SEQUENTIAL PROLOG MACHINE PEK

Naoyuki TAMURA*, Koichi WADA*, Hideo MATSUDA*, Yukio KANEDA**, Sadao MAEKAWA**

* The Graduate School of Science and Technology,
Kobe University, Rokkodai, Nada, Kobe 657 JAPAN.

** Department of Systems Engineering, Faculty of Engineering,
Kobe University, Rokkodai, Nada, Kobe 657 JAPAN.

ABSTRACT

This paper describes the sequential Prolog machine PEK currently under development.

PEK is an experimental machine especially designed for high speed execution of Prolog programs. Our objective is to research the hardware architecture available for a Prolog machine. PEK employs bit slice LSIs as the sequencer and ALU, and micro-program control. To enable the high speed execution of Prolog programs, PEK includes hardware circuits for unification and backtracking.

Simple Prolog interpreter has been developed to assess the performance of PEK. The performance is about to 40K LIPS (logical inference per second) that is equivalent to the performance of DEC-10 Prolog compiler on DEC-2060.

1 INTRODUCTION

Prolog is a logic programming language which has recently received considerable attention for its application in the field of artificial intelligence. As the execution mechanism of Prolog is quite different from the other conventional languages, the research of the hardware architecture for Prolog is important.

ICOT's PSI (Personal Sequential Inference machine), designed for research and evaluation purposes, employs a tagged architecture, micro-program control, hardware stack, and multi-processing support functions (Taki et al. 1984). PEK includes more ambitious hardware functions, such as automatic trailing circuit, automatic undoing circuit, etc. To attain high performance of more than 100K LIPS by one processor, these special hardware functions will be needed, because only 100 instructions can be executed per one logical inference if the cycle time is 100nsec.

Sequential Prolog machine designed by Evan Tick and David Warren employs a reduced instruction set, a pipelined execution, and an interleaved memory to improve execution speed of compiled Prolog programs (Tick and Warren 1984). Of course, compiling logic programs is necessary for high speed execution, but we also believe that a Prolog machine should interpret Prolog programs efficiently.

PEK is an experimental Prolog machine including some ambitious hardware functions, and is designed to investigate the architecture of Prolog machines which can execute Prolog programs efficiently. To enable these functions by low-cost hardware, we employed following design policies.

1.1 Sequential execution

It is true that the large-scale parallelism is necessary to attain high performance for logic programs, but, as David Warren pointed out in his paper (Tick and Warren 1984), it is important to investigate the maximum performance that can be achieved by a sequential Prolog machine because the performance will be bottlenecked by the speed of sequential inference. Therefore, small-scale parallelism is exploited in PEK instead of large-scale parallelism.

* Data transfer is performed in 3 fields, ie. frame field, tag field, and value field.
* Data areas are separately allocated to memory modules, such as common memory, process memory, global stack, trailing stack, etc.
* Horizontal micro-instruction format is employed to enable simultaneous

control of hardware modules.
* Undoing of the assignments for variables is performed by a sub-sequencer.
* Reading of structures is pipelined.

1.2 Micro-program control

PEK employs micro-program control, and its Prolog interpreter is written in micro-codes, and also its compiler will translate Prolog programs into micro-codes. Bit slice LSIs (Advanced Micro Devices, Am2909A and Am2903A) are used as the sequencer and ALU to reduce the size and cost of hardware.

1.3 Structure sharing method

The decision as to the use of the structure sharing or structure copying method is a subject of considerable discussion. Structure sharing was adopted with PEK, and processes for which overhead is predicted are improved by specialized hardware.

1.4 Other features

As data areas are separately allocated to memory modules, it become possible to access these areas with a complex addressing manner by simple hardware, that is

* index addressing for process memory,
* stack addressing for hardware stack,
* post-increment for address registers, and
* address calculation of variable cells for global stack.

Moreover, PEK includes special hardware for unification and backtracking, such as

* matching circuit,
* automatic trailing circuit, and
* automatic undoing circuit.

2 SYSTEM CONFIGURATION

System configuration is shown in Fig.1.

The system consists of an MC68000 host processor and PEK. All I/O devices are connected to the host processor via a Z-80. The host processor initializes PEK and supports I/O operation during Prolog program execution.

PEK is controlled (execution of halt and step etc.) from the host processor via the CMR register. Other communication registers used are ICR (Input Command Register, host to PEK) and OCR (Output Command Register, PEK to host).

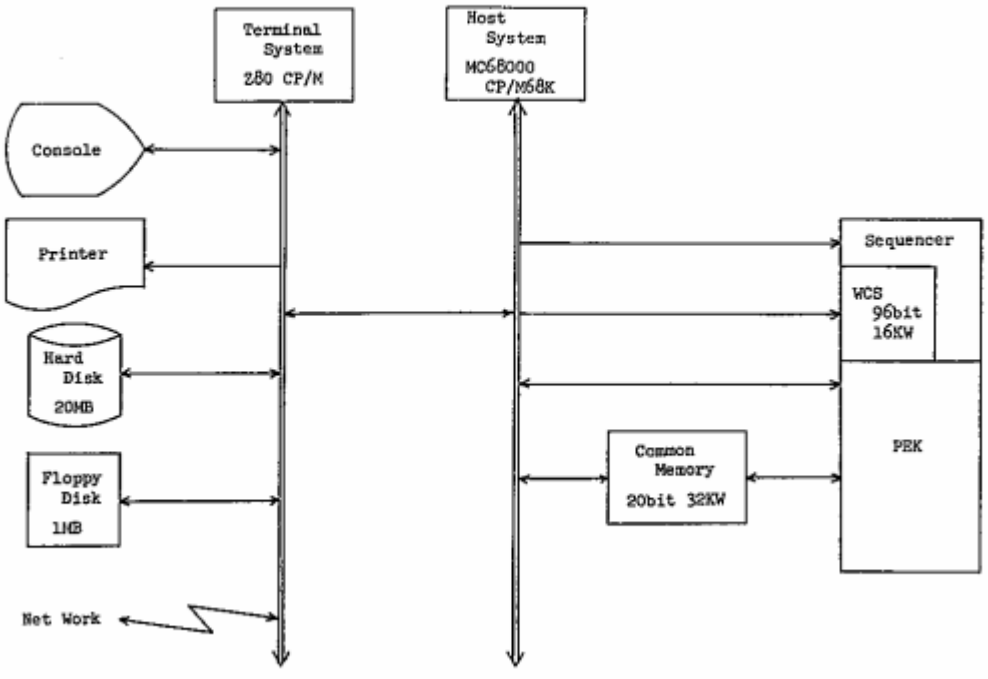


Fig.1 System Configuration

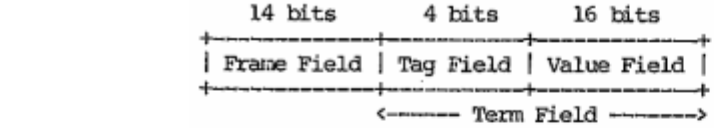
3 HARDWARE

Hardware configuration of PEK is shown in Fig.2. The hardware consists of approximately 600 ICs on the following five PCBs (45x28cm, 300pin).

[No.1] CCU board
Sequencer, WCS, and interface with MC68000 host processor.
[No.2] ALU board
ALU, bypass controllers, process memory, hardware stack, etc.
[No.3] Unification board
Global stack, trailing stack, matching circuit, undo circuit etc.
[No.4] Common memoryboard
Common memory and two address registers, two read registers, and two write registers.
[No.5] System evaluation board
Timer to measure execution time, counter to count number of micro-instructions executed.

3.1 Word format

A word used in PEK consists of 14-bit frame field and 20-bit term field to permit the transfer of molecules. The term field consists of a 4-bit tag field and a 16-bit value field.

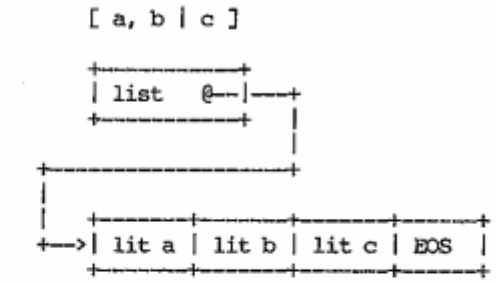


The following 11 tags are currently being considered for use.

integer
literal atom
undef
global variable
local variable
void variable
structure (term)
structure (list)
end of structure
clause
code

Lists are expressed with one-dimensional vectors in order to reduce memory size and the access times, ie. lists have a separate tag and list elements are stored in contiguous addresses. To determine the length of lists, a cell having a special tag EOS (End Of Structure) is added at the end of each list.

Composite terms are also represented in the similar structure as lists.



3.2 Memory modules

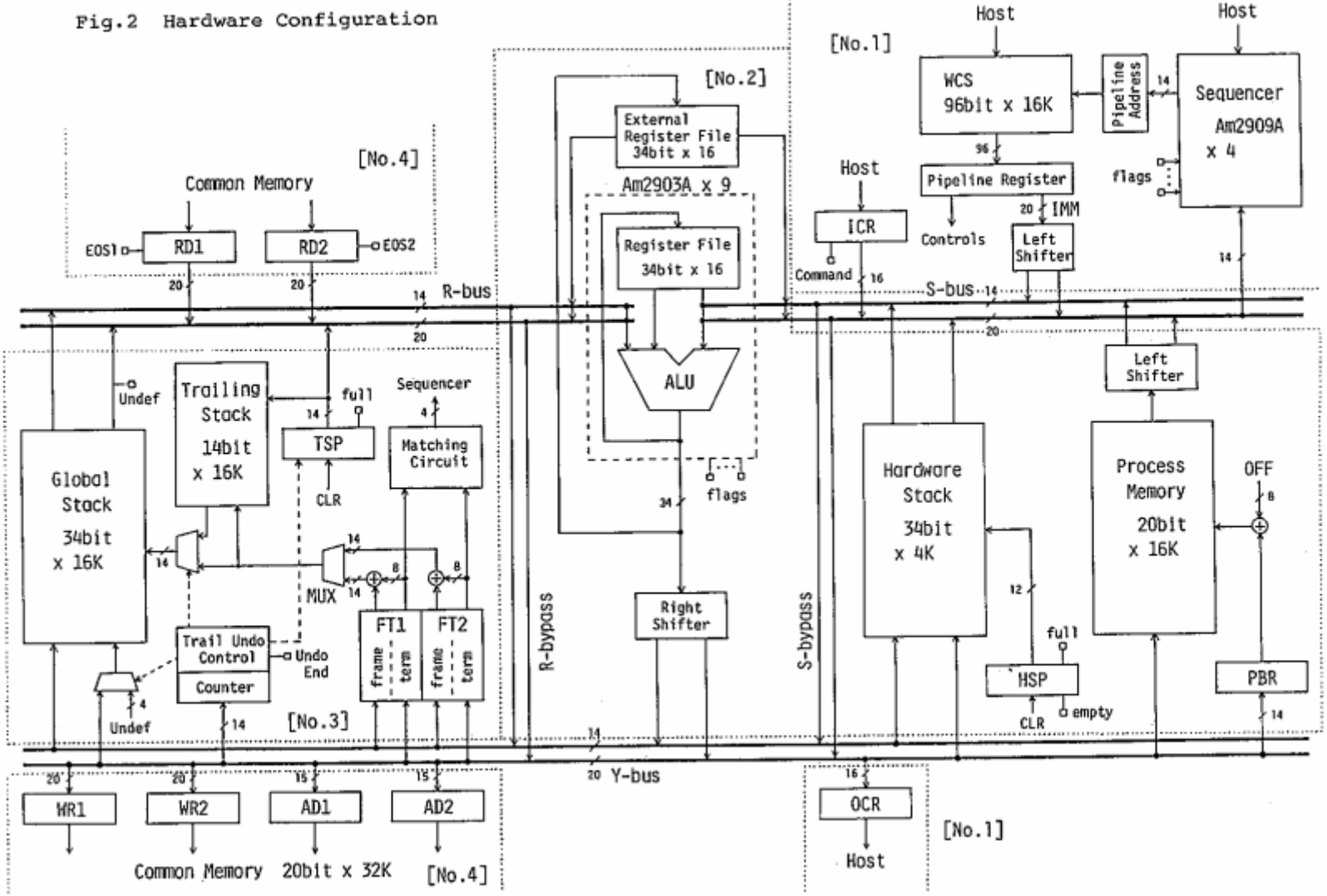
PEK memory modules are distributed according to their application and may be accessed in parallel. All memory ICs have an access time of 55nsec.

(1) WCS
WCS contains micro-programs. It has a capacity of 96bit x 16KW and may be accessed from the host processor.
(2) Common memory
This 20bit x 32KW memory is shared with the MC68000 host processor and used to contain atom headers, structures, etc. Access from PEK is via two address registers (AD1, AD2), two read (RD1, RD2) and two write registers (WR1, WR2).
(3) Process memory
This 20bit x 16KW memory is used to store the management information required for execution of Prolog programs. Address is the sum of the base register PBR and the value (+0~+255) of the OFF field in the micro-instruction.
(4) Global stack
Used to store the value cells of global and local variables. Addresses for the value cells are calculated using registers FT1 or FT2 (described later).
(5) Trailing stack
This 14bit x 16KW stack memory is used to store the addresses for variable cells to be reset to "undef" at backtracking. Uses the stack pointer TSP.
(6) Hardware stack
A small 34bit x 4KW stack used during unification. Uses the stack pointer HSP.
(7) Register files
Both the internal and external register files have a capacity of 34bit x 16words.

3.3 Internal buses

The system employs two source buses (R-bus and S-bus) and one destination bus (Y-bus). All have 34-

Fig.2 Hardware Configuration



bit width to enable the transfer of molecules. Transfer from a source bus to the destination bus is via the ALU or bypasses.

3.4 Sequencer

The sequencer consists of four Am2909A (Advanced Micro Devices) 4-bit slice LSIs. Am2925 clock generator generates eight kinds of micro-cycles from 120nsec. to 400nsec. in 40nsec. increments.

Micro-instruction fetch and execution of instructions are overlapped using the pipeline address register and pipeline instruction register. In case of a branch instruction, the branch destination address is generated during the first half of the cycle and the instruction of the destination is fetched in the latter half. In case of a non-branch instruction (continuous execution), the next instruction is fetched during the first half of the cycle. This pipeline processing enables a minimum execution time of 120nsec. for continuous execution, and execution time of 160nsec. for most branch instructions.

3.5 ALU

The ALU consists of nine Am2903A 4-bit slice LSIs which are divided into three blocks of 4, 1, and 4 corresponding to the three fields of the word format. ALU operations are performed independently for each field, and also the operation result flags are independent. Nine Am29705A LSIs are added as an external register file to store a total of 32 registers.

3.6 Bypass controllers

Two bypasses are employed, the R-bypass used to transfer data from the R-bus to the Y-bus, and the S-bypass used to transfer data from the S-bus to the Y-bus. The benefits of the bypass have already been shown in the previous paper on a high level language machine (Wada et al. 1983). The use of bypasses permit, for example, data transfer from a source bus to the destination bus and simultaneous ALU operation.

3.7 Shifters

Two shifters are used, a left shifter used when shifting the lower 14 bits of the term field to the frame field, and a right shifter used when shifting the frame field or the tag field to the value field.

3.8 Matching circuit

During unification, different processings are required according to the two term types to be matched. The FT1 and FT2 special 34-bit registers are therefore provided to PEK to permit 16-way jumps using the 9-bit value consists of the two 4-bit tags and the 1-bit comparison result of the values of FT1 and FT2. For example, whether two atoms are the same or not can be determined in a single instruction. If multi-way jump is desired with the value from only one of the two registers, a term having a special tag should be written into the other register.

3.9 Automatic address calculation for variable cells

Variable cell addresses are calculated automatically using the registers FT1 and FT2. The sum of the FT1 frame field and the lower 8 bits of the value field (index value of the variable, 0~255), or the same value for FT2 may be used as the global stack address. Selection of two address sources is specified with the MUX field in the micro-instruction.

Determination of whether the variable is bound or unbound is by checking the flag (UNDEF1 for FT1, UNDEF2 for FT2) without actually reading the value from the global stack.

The frame fields of the FT1 and FT2 normally contain the global frame addresses. Therefore, for local variables, the frame fields of the FT1 or FT2 must be rewritten.

3.10 Automatic trailing

Backtracking requires the undoing of assignments, ie. resetting variable values to "undef". Thus, at the assignment to a variable, the variable cell address must be pushed onto the trailing stack. PEK performs this operation by hardware, ie. when the write operation to the global stack is executed, the address is automatically pushed onto the trailing stack. Automatic push operation is specified with the TSC field in the micro-instruction.

3.11 Automatic undoing

Undoing is also performed automatically in PEK by a sub-sequencer. As no bus is used, undoing may be performed in parallel with main sequencer operations. Operation of the undo sequencer is begun by writing the number of undo operations into the undo counter. Popping from the trailing stack and writing into the

global stack are overlapped to permit high speed undo operation.

3.12 Pipelined reading of structures

As Prolog unification processing is performed for two structure data elements, PEK contains two address registers (AD1 and AD2), two read registers (RD1 and RD2), and two write registers (WR1 an WR2). Moreover, in order to enable high speed read of data in contiguous addresses, when data is read from RD1 or RD2, AD1 or AD2 are automatically incremented and the read operation of the next data is started. The data in the next address will be placed in the read register approximately 250nsec. later. Determination of whether or not the tag in the read register (RD1 or RD2) is EOS is by checking a flag (EOS1 or EOS2).

The frame field of the read register has the same value with the frame field stored in the address register.

4 MICRO-INSTRUCTION FORMAT

Micro-instruction of PEK is horizontal and 96-bit in width, and contain 24 fields (Fig.3).

* The 4-bit DEB field is used for debugging and system evaluation and comprises a halt bit, timer/counter start and stop bits, and an instruction counter bit.
* The CYC field is used for control of the Am2925 clock generator. One of the eight clocks (120~400nsec.) may be selected.
* The FMX field is for control of the flag multiplexer. It is used for selection of branch conditions.
* The SEQ field specifies the instruction for the Am2909A sequencer. It has a 4-level micro-stack and permits subroutine calls and loops etc. Branching using data on the S-bus as a destination address is possible.
* The ORE field is for multi-way jump. When this bit is set to "1" multi-way jump is executed according to the output value from the matching circuit.
* The XWE and RWE fields are used to specify the write operation to external and internal registers. Frame, tag, and value fields may be specified independently. The number of the register to be written is specified with the RB field.
* The SC and CC fields are for control of shift and carry input to the ALU.
* The CEM field is used to set the ALU operation result flag in the status register.
* The EA field is used to select the ALU R source.
* The RB and RA fields are used to specify the S and R source register (internal and external) number. The RB field is also used for specification of the destination register number.
* The ALS and ALF fields specify the ALU shift operation and arithmetic and logic operation. Operation is performed independently for 3 fields in the word format.
* The MUX field is used for selection of the global stack address source.
* The TSC and HSC field control the trailing and hardware stacks respectively.

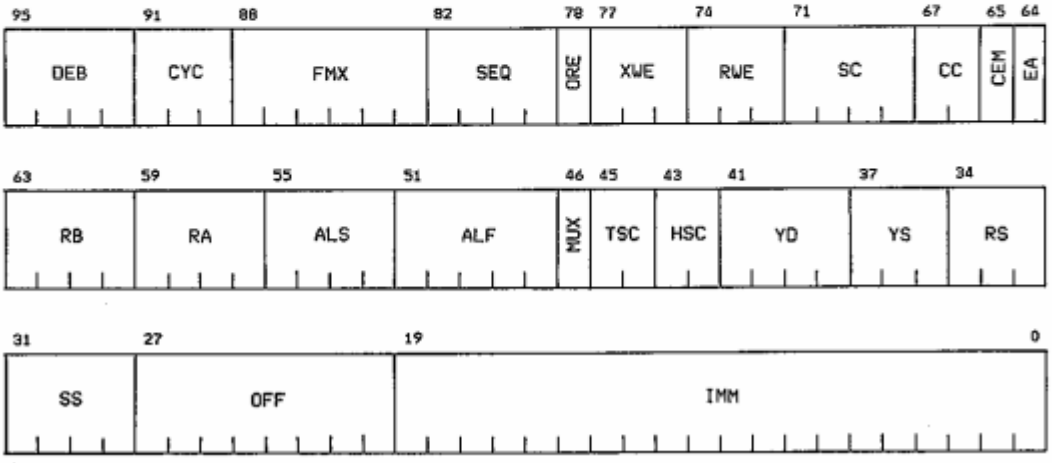


Fig.3 Micro-instruction format

* The YD field is used to specify the Y-bus destination.
* The YS field is used for control of Y-bus source and right shifter.
* The RS field is used to specify the R-bus source.
* The SS field is used for control of S-bus source and left shifter.
* The OFF field is used to specify the offset (+0~+255) to the process memory base register PBR.
* The IMM field is a 20-bit field used to set the constant output to the S-bus. Normally output to the S-bus term field, and output to the S-bus frame field when the left shifter is used.

5 DEVELOPMENT SOFTWARE

5.1 Micro-assembler

Micro-instruction of PEK is horizontal and 96-bit in width, and and contain 24 fields. An assembler specifying the mnemonic for each field was written initially, however it was found that programs were difficult to read. Therefore, a preprocessor including a macro call function with pattern matching was developed using Prolog system in the host processor.

The Am2925 clock generator used in PEK can generate eight kinds of 4-phase (C1~C4) clock, and the clock to be used is selected by the 3-bit CYC field in the micro-instruction. Therefore, the micro-assembler should choose the most appropriate cycle at assembly time.

Two programs to determine cycle length were written and evaluated. The first program determines the cycle length by delimiting operation time for each device into 40nsec. units. In this case, most conditions need not be considered and cycle length may therefore be determined simply. Determination of cycle length with hardware would employ this method.

The second program determines the cycle length by finding the longest data path and improved cycle time by 15% in comparison with the first program. The cycle time can be improved by up to a further 40% under some special conditions. For example, if the global stack address source has been selected in the immediately previous micro-instruction, a speed of the reading from the global stack would increase 96nsec.

5.2 Monitor/debugger

A monitor has been developed on the MC68000 host processor for debugging the hardware modules. This monitor includes a screen editor, a micro-assembler, a disassembler, and some debugging aids such as setting of break points, dumping of register contents, displaying flag conditions, etc. to simplify development of test programs.

6 INTERPRETER

The specifications of Prolog language is scheduled to be compatible to the DEC-10 Prolog (Warren 1977) and no expansion is currently being considered, and detailed interpreter design is proceeding.

Simple Prolog interpreter have been coded to evaluate the system performance. In this version, all variables are treated as global, and dispensable functions, such as clause indexing, tail recursion optimization, are not implemented.

Two Prolog programs were executed on this interpreter.

One is a program for appending a list of length n to an empty list.

```
(1) append([],Z,Z).
(2) append([W|X],Y,[W|Z]) :- append(X,Y,Z).
(3) ?- append([1,2,...,n],[],Z).
```

The other is for reversing a list of length n.

```
(4) reverse([P|Q],S) :-
        !, reverse(Q,R), append(R,[P],S).
(5) reverse([],[]).
(6) append([W|X],Y,[W|Z]) :-
        !, append(X,Y,Z).
(7) append([],Z,Z).
(8) ?- reverse([1,2,...,n],S).
```

Table 1 shows the number of executed instructions at the interpretation of these programs. and Table 2 shows the case of n=30. The average cycle length is estimated to be 170nsec. Therefore, the speed is about 40K LIPS.

To evaluate the contribution of the specialized hardware for unification, an unification process of the append program was traced.

The number of executed instructions was counted at the unification of the body of (2) with the head of (2) when goal (3) was matched once with the head of (2). All variables are considered as global variables, and frames on the global stack are assumed to have already been reset to "undef". Fig.4 shows the status immediately prior to the

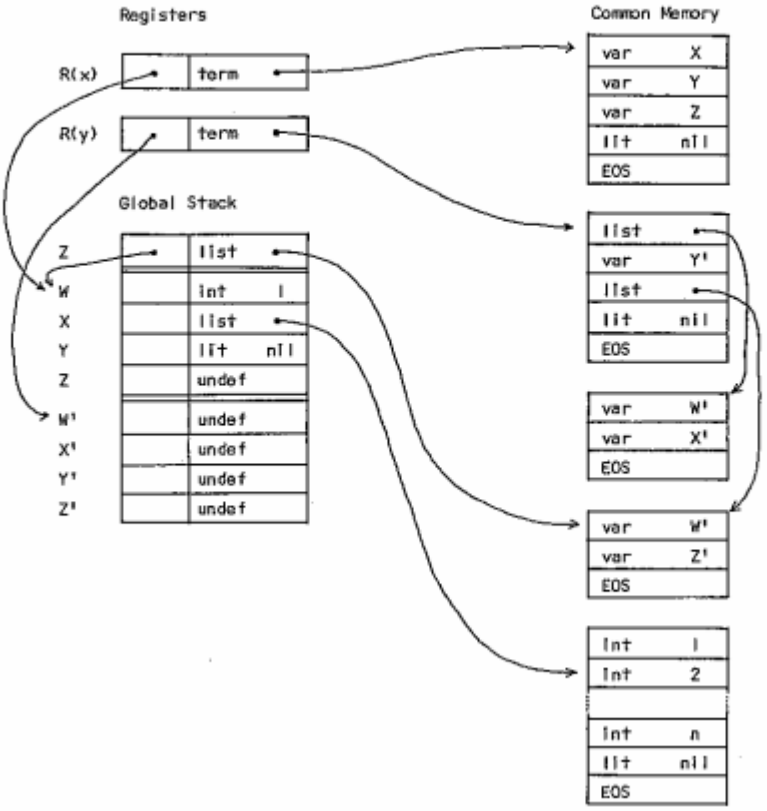


Fig.4 Status before the unification

Table 1 The number of executed instructions

	# of logical inference	# of executed instructions	# of instructions for control	# of instructions for unification
append([1,2,..,n],[],Z)	n + 1	145n + 149	49n + 94	96n + 55
reverse([1,2,..,n],S)	$\frac{1}{2}n^2 + \frac{3}{2}n + 1$	$\frac{145}{2}n^2 + \frac{357}{2}n + 178$	$33n^2 + 85n + 101$	$\frac{79}{2}n^2 + \frac{187}{2}n + 77$

Table 2 The number of executed instructions (n=30)

	# of logical inference	# of executed instructions	# of instructions for control	# of instructions for unification	logical inference per second (LIPS)
append([1,2,..,30],[],Z)	31	4499	1564	2935	40.5 K
reverse([1,2,..,30],S)	496	70783	32351	38432	41.2 K

unification. Register R(x) contains the caller argument list, and register R(y), the source argument list. Source variables are identified by addition of '.

The unification program was traced under these conditions. Table 3 shows the details. "Fetch" includes write instructions to registers FT1 and FT2, and detection of EOS flags. "Match", "assign", "dereference", and "push" include a multi-way jump instruction using the matching circuit.

7 SUMMARY

This paper has described the architecture and software of the sequential Prolog machine PEK. This machine includes specialized hardware for unification and backtracking essential to high speed execution of Prolog programs.

Simple Prolog interpreter has been developed to assess the performance of PEK. The performance is about 40K LIPS that is equivalent to the performance of DEC-10 Prolog compiler on DEC-2060.

Table 3 Trace of the unification

(1) Call		1 instruction
(2) Initialize		10 instructions
(3) Set start addresses of (X,Y,Z) and ([W'|X'],Y',[W'|Z'])		4 instructions
(4) Unify X with [W'|X']		10 instructions
* fetch	4	
* dereference	3	
* push	3	
(5) Unify Y with Y'		11 instructions
* fetch	4	
* dereference	4	
* assign	3	
(6) Unify Z with [W'|Z']		7 instructions
* fetch	4	
* assign	3	
(7) Unify nil with nil		7 instructions
* fetch	6	
* match	1	
(8) Prepare for unification of [2,3,..,n] with [W'|X']		4 instructions
* pop & set start addresses	4	
(9) Unify 2 with W'		7 instructions
* fetch	4	
* assign	3	
(10) Unify [3,..,n] with X'		9 instructions
* fetch	6	
* assign	3	
(11) Return		2 instructions
	Total	72 instructions

REFERENCES

Naoyuki TAMURA, Koichi WADA, Hideo MATSUDA, Yukio KANEDA, and Sadao MAEKAWA: Prolog Machine PEK (in Japanese), Proc. of the Logic Programming Conference '84, 8-2, ICOT, 1984.

Yukio KANEDA, Naoyuki TAMURA, Koichi WADA, and Hideo MATSUDA: Sequential PROLOG Machine PEK Architecture and Software System, International Workshop On High-Level Computer Architecture, 1984.

Koichi WADA, Yukio KANEDA, and Sadao MAEKAWA: High-Speed Execution of FORTH and PASCAL programs on a High-Level Language Machine, 9-th EUROMICRO, 1983.

Kazuo TAKI, Hiroshi NISHIKAWA, Akira YAMAMOTO, and Minoru YOKOTA: Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI), Proc. of the International Conf. on Fifth Generation Computer System, 1984. (to appear)

Evan TICK and David H.D.WARREN: Towards A Pipelined PROLOG Processor, Proc. of the International Symposium on Logic Programming, pp.29-40, 1984.

David H.D.WARREN: Implementing Prolog -- Compiling Predicate Logic Programs Vol.1,2, D.A.I.Research Report No.39,40, University of Edinburgh, 1977.

Advanced Micro Devices Inc.: The Am2900 Family Data Book.