Invited Paper

Fast LISP Machine and List-Evaluation Processor EVAL II — Processor Architecture and Hardware Configuration*

HIROTOSHI MAEGAWA**, TOSHIFUMI SAITO**, TOSHIO DOI**, TAKESHI NISHIKAWA** and HIROSHI YASUI**

This paper describes the architecture and the hardware configuration of an experimental LISP machine developed with the primary emphasis on high-speed list processing. This machine consists of a processor EVAL II, an I/O processor and a common main memory; the EVAL II processes data of list structure, the I/O processor executes LISP input/output functions and manages external memories, such as files, and the main memory chiefly stores list cells. The EVAL II is a newly designed processor suitable for list processing. To develop such a processor, no existing microprocessor is used. The EVAL II has common address/dara buses because it processes mainly address pointers. The unique hardware features of the EVAL II include CARCDR-operation facility which executes list searching at high speed in parallel to ALU operations and/or branch operations, and the dispatch facility by which the program efficiently branches to a destination according to a list-cell type or a function type. Parallel processing (branch operation and other operations) and pipeline processing are also executed efficiently. The machine executes benchmark programs as fast as LISP systems on very large general-purpose computers. The dynamic measurements show that this architecture is suitable for LISP machines.

1. Introduction

LISP[1] is a list-processing language which is used for researching artificial intelligence, knowledge engineering, and other applications. Thus, it is at present one of the principal computer languages. Various high-speed LISP systems therefore have been developed for large general-purpose computers as well as many others [2]-[5]. The development of LISP machines [6]-[21] has also been promoted by the spread of microcomputers or processor IC's and by the technological progress of semi-conductor integrated circuits.

As a new approach to attain higher speed with LISP systems, the EVLIS machine [22], [23] has been proposed and constructed. It is a LISP machine of multiprocessor configuration.

Special attention is payed to the fact that the first argument of the function *evlis* in LISP interpreter [1] consists of comparatively independent elements. Each of these elements is defined as a processing unit called a process. The EVLIS machine executes these processes in parallel by multiple processors. Moreover, the EVLIS machine takes care of all the problems related with parallel execution, including the allocation of the process to processors, the communication among processors, and the correct propagation of side effects caused by the execution of list-altering functions such as *rplaca*. Therefore, the machine automatically executes parallel processing even if the program is written in the grammar of LISP 1.5, which does not support parallel processing.

The architecture of the EVAL II, which is a component of the EVLIS machine and is devoted to evaluating list data, is designed for list processing. As a result, LISP machine consisting of a single EVAL II implements higher performance than preceding LISP machines.

This paper describes the architecture of the EVAL II and the hardware design of LISP machine constructed with a single EVAL II.

2. Designing Policy

The LISP machine is an experimental and small-scaled machine which can be developed with the budgets and technology of one laboratory.

From the start of the designing, emphasis has been put on providing processor EVAL II with a LISP-oriented architecture. The characteristics are as follows:

(1) Most of the data in the memory are address pointers. Therefore, no distinction between address bus and data bus is made, and the connection between addresses and data is raised.
(2) A three-address ALU operation provides the high-speed processing and the operational flexibility.
(3) Maskers for the operational sources of the ALU

*This paper is a translation from Japanese of Transactions of IPSJ, Vol. 24, No. 5 (1983), which received an award as the 1983 winning paper of IPSJ.
**Department of Applied Physics, Faculty of Engineering, Osaka University, Suita, Osaka 565, Japan.

facilitate the extraction of the appropriate field from data; e.g., to extract the pointer from list data.

(4) Since branch operations are heavily used, an ALU operation and a branch operation are executed in parallel in one microinstruction. Moreover, the dispatch operation implements high-speed branches according to a function type, a list-cell type and some other codes.

(5) The CARCDR-operation facility is adopted. It implements parallel execution of a list-searching operation and ALU operations and/or branch operations.

(6) Pipeline processing is executed for fetching and executing instructions, and for writing in memory and other operations.

(7) A diagnostic facility for direct access to the internal status of the EVAL II is incorporated to facilitate hardware and software debugging and statistical data collection.

To implement such an architecture, a microprogram control is adopted for the EVAL II because of high flexibility of processing and easy construction of software.

The existing bit-sliced microprocessor families have various problems for list processing; for example, the program stack includes only a few words, the processor architecture restricts bus configuration, and less important operations on list data waste many bits of microinstructions. Therefore, neither any existing microprocessor, nor its peripheral LSI is used for the EVAL II. Schottky TTL IC's are mainly used to construct the EVAL II.

3. Machine Configuration

The general machine configuration is shown in Fig. 1.

The EVAL II is a list-evaluation processor. It evaluates the list data which is given by way of a mailbox in the main memory from the I/O processor, and returns the value to a mailbox. The hardware of the EVAL II is described in detail in the next section.

The maximum capacity of the main memory is 32 K

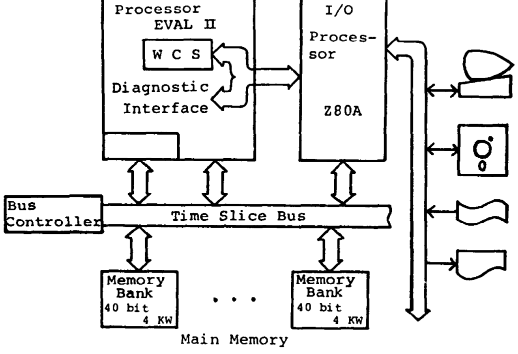


Fig. 1 LISP machine configuration

40-bit words; the read-access time and the write-access time are 350 nsec and 200nsec, respectively. The memory consists of up to eight memory banks (1 bank=4 K words), a common bus and a bus controller. Up to two main memories can be mounted. The two main memories can operate in parallel. Accessing between the processors and memory banks is multiplexed by time division (50 nsec) of the bus. The conflicts among banks for the bus are resolved by the priority sequence which changes every 50 nsec. An access is allowed to the bank of a higher priority level. The conflicts among processors for a bank are resolved by the fixed priority level of each processor in each bank. The main memory is used for storing list cells (car part, cdr part; 20 bits each) and as mailboxes for communication between processors.

The I/O processor has a Z80A CPU with a 4 MHz clock and a 64 K-byte local memory. This processor operates as a service processor of the machine; it manages files, supports programming by the microassembler, activates and stops the EVAL II, and executes LISP input/output functions. It also loads microprograms to the WCS by way of the bus between the I/O processor and the EVAL II, monitors the internal status of the EVAL II, supports debugging, and collects statistical data.

The numbers of circuit boards and IC's used for constructing the machine are as follows: 10 circuit boards with 620 IC's in the EVAL II; 4 circuit boards with 270 IC's in the I/O processor; 12 circuit boards with 690 IC's in the main memory.

4. Hardware of the EVAL II

The hardware configuration of the EVAL II and the microinstruction format are shown in Figs. 2 and 3, respectively.

4.1 Internal Buses and ALU

The internal buses consist of an A-source bus (ASB), a B-source bus (BSB) and a destination bus (DSB); each is 20 bits wide.

The ALU manipulates the three-address instructions and can execute logical AND, logical OR, exclusive OR, fixed-point addition/subtraction and right shift instructions. The two input operands are given by the ASB and the BSB, and the result is send to the DSB. All the registers and memories necessary for ALU operations are connected to the ASB and the DSB. On the other hand, eight frequently used general-purpose registers (R0-R7) and five specialized registers (holding NIL, *T*, all 1's, etc.) are connected to the BSB in order to simplify the hardware. The B-source field of the microinstruction format is therefore shortened.

The inputs of an ALU operation can be masked by the 20-bit pattern which is specified in the K-field of the microinstruction. A mask for the upper 4 bits and one for the lower 16 bits of input data are frequently used

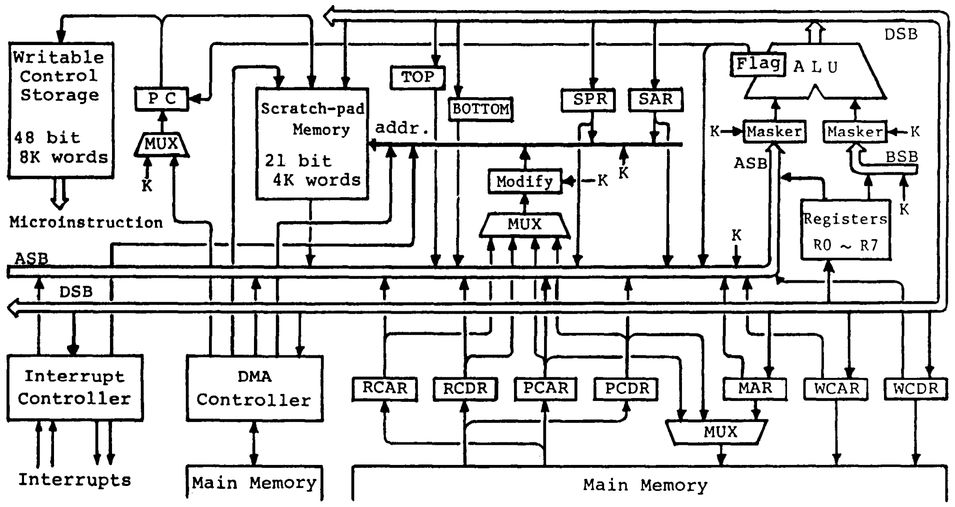
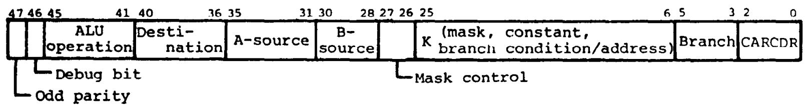


Fig. 2 Prosessor "EVAL II" block diagram



Debug bit	: used to set breakpoints, to measure execution frequencies, etc.
ALU operation	: specify ALU op. and flag control
Destination A-source B-source	: specify operands of ALU op.
Mask control	: specify masking for A- and/or B-source
K	: specify a mask pattern, branch condition and address, etc. (see Fig. 4)
Branch	: specify a kind of branches (see Fig. 4)
CARCDR	: specify a source and destinations of CARCDR-op.

Fig. 3 Microinstruction format

for extracting the tag part and the pointer part of a list cell. Therefore, two bits of the K-field are assigned for these mask specifications and the other bits are used for branch condition and branch address, etc. (See 4.4 and Fig. 4.) Any constant for input to the ALU can be constructed by specifying a mask from a specialized register containing of all 1's.

Status flags (zero, negative and carry) are set or reset according to the result of an ALU operation. Carry can be the third input of ALU operations.

4.2 Scratch-pad Memory

As a multipurpose storage in addition to the eight general-purpose registers, the EVAL II has a scratch-pad memory (SM) of 4 K 21-bit words capacity; one bit of each word is used for even parity. Static memory with 35-nsec access time is used for the scratch-pad memory. The SM is used to store 1) a stack which temporarily holds the return addresses of microprograms and the intermediate data, 2) dispatch tables used for multidirection-branch instructions, and 3) tables of destination addresses for interrupt caused by communications from other processors, a parity error, etc. A location address in the SM is specified by a Stack Pointer Register (SPR), an Address Register for the SM (SAR), a K-field of a microinstruction, the dispatch address of a multidirection-branch instruction, or the interrupt vector.

The SPR indicates the address of the topmost filled location of the stack. The value of the SPR is automatically incremented or decremented at the time of an accesss to the top of the stack; however it is possible to access the stack without changing the value of the SPR. A stack overflow or underflow is detected by comparing the boundary registers (TOP and BOTTOM) with the SPR and causes an interrupt.

The SAR and the K-field are used for accessing any location in the SM.

Pipeline processing is executed for data writing to the SM and an ALU operation; parity generation and preparation for data writing are performed in the first half of the next instruction cycle, and then data are written in the latter half of that cycle. When an instruction to read the SM is to be executed immediately after an in-

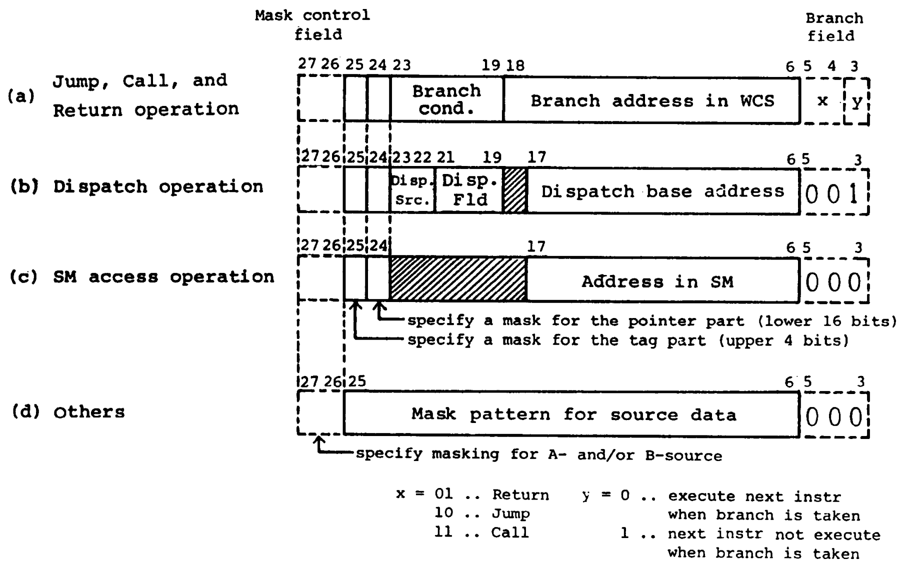


Fig. 4 K-field construction

struction to write to the SM, the two access addresses are compared. If they are the same, the correct data is read by passing around the SM.

A DMA feature is provided between the SM and the main memory for efficiently saving a part of the contents of the stack at the time of a stack overflow and for efficiently storing the status of the EVAL II, though DMA controller is not provided at present. The unit of transfer is two words, and the SM is accessed by cycle stealing. An independent path connected to the time sharing bus is used for transfer between the SM and the main memory.

4.3 Main Memory Access and CARCDR-operation Facility

The main memory is accessed at the address specified by the register MAR. The contents of registers WCAR and/or WCDR are written to the main memory. The data from the main memory are loaded into the first two or all of the registers RCAR, RCDR, PCAR and PCDR. Accessing can be activated by an instruction which loads the result of an ALU operation into MAR, WCAR, or WCDR. The EVAL II is able to execute other operations during access to the main memory.

The CARCDR-operation facility is provided to raise the efficiency of list-searching operation; the content of the main memory can be directly read at the address specified by PCAR or PCDR independent of MAR. Accessing is activated by specifying the CARCDR-field of a microinstruction. It enables accessing to the main memory in parallel and simultaneously with other operations.

4.4 Microinstruction and Sequence Control

The writable control storage (WCS) which stores microinstructions has the capacity of 8 K 48-bit words; one bit of each word is an odd parity. Static memory with access time of 35 nsec is used for the WCS. The basic cycle time of an instruction is 100 nsec, but is extended by 50 nsec when the ALU operation is addition/subtraction, and when the data of the SM is to be read out. Accessing for the main memory is executed by pipeline processing. The cycle time of an instruction, however, will be extended if the main memory is busy at the time of accessing.

The instruction sequence branches to the location specified by the Branch-field and the K-field of a microinstruction. Fig. 4 shows the functions of these fields. In the case of three types of branch operations (Jump, Call and Return) given in Fig. 4(a), branching is determined by one of five conditions (positive, zero, negative, carry and no-carry) in the K-field for the status flags of the ALU. Since fetching and executing of instructions are overlaid by pipeline processing, the *y* bit in the Branch-field specifies whether to execute the next instruction already fetched or to suppress its execution when a branch occurs.

Fig. 4(b) shows the case of multidirection-branch operation (dispatch operation) in 16 or 256 directions. The destination address is generated in one instruction cycle as shown in Fig. 5.

4.5 Diagnostic Interface

The diagnostic interface enables the I/O processor to directly access the internal status of the EVAL II. This facilitates hardware and software debugging,

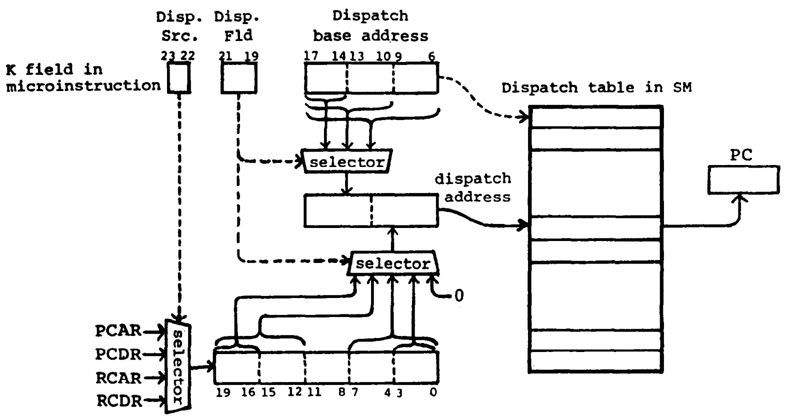
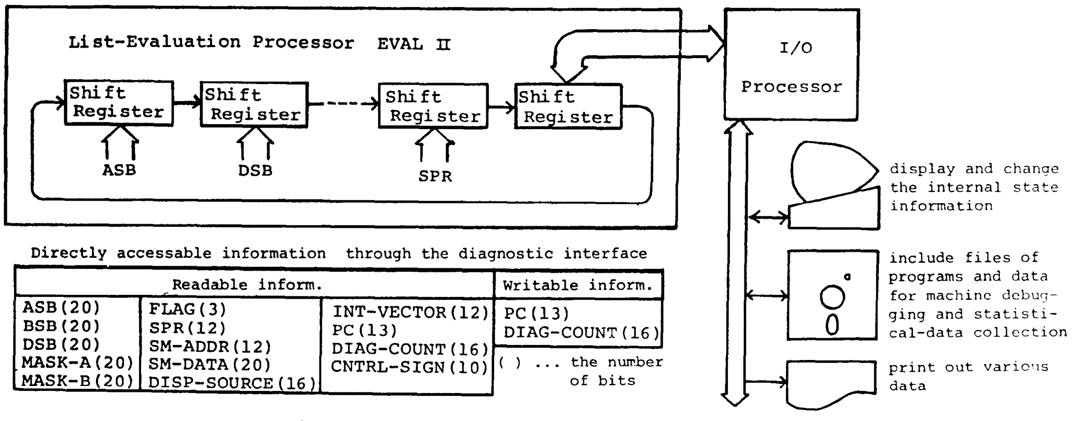


Fig. 5 Branch address generation on dispatch operation


List-Evaluation Processor EVAL II
Shift Register
Shift Register
Shift Register
Shift Register
ASB
DSB
SPR
I/O Processor
display and change the internal state information
include files of programs and data for machine debugging and statistical-data collection
print out various data
Directly accessable information through the diagnostic interface

Readable inform.			Writable inform.
ASB(20)	FLAG(3)	INT-VECTOR(12)	PC(13)
BSB(20)	SPR(12)	PC(13)	DIAG-COUNT(16)
DSB(20)	SM-ADDR(12)	DIAG-COUNT(16)	
MASK-A(20)	SM-DATA(20)	CNTRL-SIGN(10)	
MASK-B(20)	DISP-SOURCE(16)		

( ) ... the number of bits

Fig. 6 EVAL II diagnostic interface and monitoring function

maintenance and statistical data collection. To implement the diagnostic interface with less wire, shift registers are buried in the EVAL II to transfer information (Fig. 6). The diagnosis counter of a 16-bit register is connected directly to the diagnostic interface. This counter can be set to an appropriate value. It is incremented whenever a microinstruction whose debug bit is 1 is executed. The counter overflow causes the clock of the EVAL II to stop. The break points of the program can be set and dynamic characteristics can be measured by combining this counter with debug bits.

The scratch-pad memory and the registers not connected directly to the diagnostic interface are accessed by executing the instructions on the WCS (activated by the I/O processor). The data are read out through the diagnostic interface by taking them out to the ASB or BSB. The data to be set are written by using the K-field of a microinstruction.

The information accessed directly or indirectly can be monitored by a CRT terminal. This facilitated the check of machine status for hardware adjustments and reduced the debugging time. Moreover, the interpreter was produced and debugged only with the CRT terminal and a printer.

5. Conclusion

The developed LISP machine can execute list processing at high speed. High efficiency has been attained with parallel processing such as the CARCDR-operation, simultaneous execution of branch operation and some other operations, and pipeline processing. The EVAL II, a list-evaluation processor, is constructed mainly with logical IC's to implement the LISP-oriented architecture and high-speed operations.

It is proved that the interpreter of this machine processes benchmark programs as fast as LISP interpreter systems implemented on very large computers such as ACOS-1000 and M200-H. Thus, the initial design requirements are met almost completely. The adequacy of this architecture for a LISP machine is confirmed by the measurements of dynamic characteristics. These are discussed in detail in a related paper [24].

Acknowledgement

The authors would like to thank Mr. Nobuyuki Otera and other members of our laboratory for the cooperation for machine configuration. This research was supported in part by the Science Research Fund of the Ministry of Education.

References

1. McCarthy, J. *et al. LISP 1.5 Programmer's Manual*, MIT Press, Cambridge, Mass., 1966.
2. Programming Symposium Committee. On the LISP Contest, *Proc. of Symbol Manipulation Symposium of IPSJ* (July 1974), 176–219, (in Japanese).
3. Takeuchi, I. On the Second Lisp Contest, *Joho-Shori* **20**, 3 (1979), 192–199, (in Japanese).
4. Kurokawa, T. Fast Interpreter of LISP 1.9, *Journal of Information Processing* **2**, 2 (1979), 81–88.
5. Chikayama, T. Lisp System as a Programming Tool, *Proc. of 23rd Annual Convention IPSJ* (Oct. 1981), 225–226, (in Japanese).
6. Greenblatt, R. The LISP Machine, MIT AI Lab., Working Paper, 79, 1974.
7. Knight, T. CONS, MIT AI Lab., Working Paper, 80, 1974.
8. Shimada, T., Yamaguchi, Y. and Sakamura, K. A LISP Machine and Its Evaluation, *Trans. of IECEJ* **J59D**, 6 (1976), 406–413, (in Japanese).
9. Goto, E. *et al.* Design of a Lisp Machine—FLATS, *Proc. of 1982 ACM Symposium on LISP and Functional Programming* (Aug. 1982), 208–215.
10. Steele, G. L. Jr. and Moon, D. A. CADR, MIT AI Lab., Working Paper, 1978.
11. Usuki, T., Tamaru, K. and Tokoro, M. LISP Machine Implementation on Multi-Microprocessor System, *Proc. of IECEJ Technical Group Meeting*, EC78-31, 1978, (in Japanese).
12. Williams, R. A Multiprocessing System for the Direct Execution of LISP, *Proc. of 4th Work Shop on Computer Architecture* (1978), 35–41.
13. Ida, M. and Mano, K. Lisp Machine Based on a Micro-processor: ALPS/I, *Trans. of IPSJ* **20**, 2 (1979), 113–121, (in Japanese).
14. Nagao, M., Nakajima, K., Mitamura, K., Itoh, H. and Kawaguchi, T. LISP Machine NK3 and Its Performance Evaluation, *Proc. of IPSJ WG Meeting*, SYM7-4, 1979, (in Japanese).
15. Taki, K., Kaneda, Y. and Maekawa, S. Experimental LISP Machine, *Trans. of IPSJ* **20**, 6 (1979), 481–493, (in Japanese).
16. Hibino, Y., Watanabe, K. and Ohsato, N. Architecture of a Lisp Machine: ELIS, *Proc. of IPSJ WG Meeting* SYM12-15, 1980, (in Japanese).
17. Burton, R. R. *et al.* Overview and Status of Dolado LISP, *Conf. Record of 1980 LISP Conference*, Stanford (1980), 243–247.
18. Hibino, Y. A Practical Parallel Garbage Collection Algorithm and Its Implementation, *Proc. of 7th Symposium on Computer Architecture* (1980), 113–120.
19. Sussman, G. J. *et al.* Scheme-79-Lisp on a Chip, *IEEE Comput.* **14**, 7 (1981), 10–21.
20. Guzḿan, A. A Heterarchical Multi-Micro Processor LISP Machine, *Proc. Conf. on Computer Architecture for Pattern Analysis and Image Database Management* (1981), 309–317.
21. Yasui, H. A survey of LISP Machine, *Joho-Shori* **23**, 8 (1982), 757–772, (in Japanese).
22. Yasui, H., Saito, T., Mitsuishi, A. and Miyazaki, Y. Architecture of EVLIS Machine and Dynamic Measurements of Parallel Processing in LISP, *Proc. of IPSJ WG Meeting*, SYM10-4, 1979, (in Japanese).
23. Maegawa, H., Doi, T., Nishikawa, T., Saito, T. and Yasui, H. A Constitution of List-Evaluation Processor on EVLIS Machine, *Proc. of IPSJ WG Meeting*, SYM17-1, 1982, (in Japanese).
24. Saito, T., Doi, T., Nishikawa, T., Maegawa, H. and Yasui, H. Fast LISP Manchine and List-Evaluation Processor EVAL II—Machine Evaluation on Interpreter, *Trans. of IPSJ* **24**, 5 (1983), 689–695, (in Japanese).

(Received December 7, 1984; revised February 20, 1985)