ACE-A NEW MODULAR COMPUTER ARCHITECTURE

Hajime IIZUKA, Osamu ISHII, Kensuke FUJII,
Ryoichi OHOMOTE and Tatsumi FURUYA

Electrotechnical Laboratory, Tokyo, Japan

This paper describes the architecture of a new experimental modular computer, named ACE (Adaptive Computer Experiment), and its standard processor module. The prototype system is now under development at the Electrotechnical Laboratory. The basic design goal is to provide the system with an ability to adapt itself to the application environment. In order to achieve this goal, the following techniques are employed.

(1) Processor-Memory-Switch level modular organization with a very universal bus structure for inter-module connection.
(2) The nucleus of the standard processor module is a newly designed emulation-oriented microprocessing unit, which is planned to be put into LSI in a few years, after some possible modifications.
(3) The structure of the standard processor module includes various novel features such as dynamic microprogramming with a microcache, a data cache with 'cachability' control, automatic data length alignment and so on.

1. INTRODUCTION

Interest in computr architecture research has been inspired by the fascinating progress of semiconductor technology, especially in the area of multiprocessor computer organization. Although not a few experimental systems are now being constructed, none of them has been reported to be very successful yet. It is a obvious fact that computer architecture must be matched to the current technology and the demands from user applications. While most of the efforts on multiprocessor experiments are quite reasonable from the viewpoint of hardware development, it is also true that there are an increasing number of applications which require very complex control structures, and future computer architectures must be capable of executing this kind of application job effectively. To achieve this goal, computer architectures, control structures and the relations among resources must be able to be tailored to the needs of applications problems. Therefore, one of the most important features of new computer organizations is the ability to adapt itself to the application environment.

In view of the above considerations we have started the design of a new experimental modular multiprocessor system, which we named "ACE" (Adaptive Computer Experiment). In order to achieve adaptability to problem structures, ACE has adopted the following three design features as basic techniques.

(1) Modular structure at PMS* level and an inter-module communication system of high flexibility and generality.
(2) Extensive dynamic microprogramming capabilities and a new microprocessing unit architecture oriented to general-purpose emulation.
(3) Use of descriptors at microprogram level.

In this paper we will describe these primary architectural features of ACE whose first version (ACE0.1) is now under development.

* Processor-Memory-Switch (Bell's notation).

2. COMPUTER COMPLEX WITH MODULAR STRUCTURE

For a computer capable of adapting itself to the application structures, the organization which is constructed from various modules with both logically and physically flexible communication features is most suitable. We shall call these computers modular type computer complexes. Although this modular architecture is not a new concept, very little research on PMS level modules with general communication capability has been carried out. It now seems the PMS level modular structure is compatible with the recent trends in semiconductor technology.

It is true that minicomputers like PDP-11 or Lockheed SUE have such design features in a sense and using them a few experimental application oriented computer complexes(1) have been trial constructed, but they are very restricted, especially in inter-module communication generality. To the best of the author's knowledge, Fuller's computer module(2) is the only research on the general modular design at the PMS level, but this proposed architecture has not been implemented yet. It therefore seems that ACE is the first experimental attempt to design a computer with very general modular structure.

3. INTERCONNECTIONS AMONG MODULES

Flexibility and generality of communication among system modules are most important for the modular structure computers. In ACE, all connections are achieved in one uniform way, both logically and physically, by a bus called the 'C-bus'. However, this does not mean that the system has only one C-bus. A C-bus can be uniformly used for both processor-processor communication and processor-memory communication. Each C-bus is provided with a C-bus controller (CC) which performs all the communication control, such as synchronization, arbitration and request monitoring. Data lines are 32 bits width and are common to all the connected modules, while control lines are

connected star-wise to CC. Up to eight modules can be connected to each C-bus. Each standard processor module under development has two C-bus connection ports (expansible to four). Taking advantage of its flexibility, the ACE system can be configured into various structures, e.g., array, hierarchy (see fig. 1) and various hybrids.

As buses allow the most flexible and general communication, the bus structure is consistent with the ACE design philosophy. Although this uniform bus method has the disadvantage of increased hardware and cost for communication, this is compensated for by its merit. The ACE C-bus has the following several features for flexible communication.

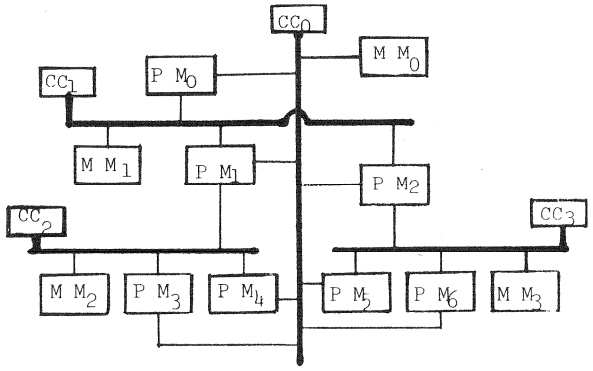


Fig. 1. An example of hierarchical connection of processor and memory modules.

3.1 Use of global address

For the purpose of inter-module communication, the system uses global logical addresses common only on the C-bus (Global Addresses; abbreviated as GA) instead of the physical module numbers and the internal Local Addresses (LA) in each module. The GA may be completely independent of the LA in each module, because each module can convert LA to GA before issuing communication requests on the C-bus.

Every module connected to a C-bus is assigned certain ranges of the GA space (possibly dynamically) as its Recognize Address (RA) space. Whenever the requesting module puts a GA on the C-bus, each module compares it with its RA, and all recognize the GA on the C-bus and are ready to perform the requested action respond 'Ready' and reconvert the GA to its own particular LA, thus establishing a transmission path. When some modules recognize the GA but find the some time is necessary to prepare the required block transmission, if the preparation time is short enough, they respond 'Wait'. If it happens to be rather long, they respond 'Hold' and put a temporary end to the communications establishment procedure. After a while when they become ready, 'Continue' requests are issued. On the other hand modules which find the GA on the C-bus is not in the ranges of their RA space simply respond 'No' and take no further action.

As easily seen from the above description, the GA may be cosidered as a kind of name given to the information. Accordingly, as long as the relation between GA and LA is fixed, the information assigned to this GA may reside in any module on the C-bus.

3.2 One to many communication

The communication method explained above easily facilitates one to many communication. If certain ranges of GA's are set to be recognized by all modules on a C-bus, information with these GA's are received by all the modules (broadcast). Needless to say, receiving modules can be any subset of modules on the C-bus. Fig. 2 gives an example of the address conversion process.

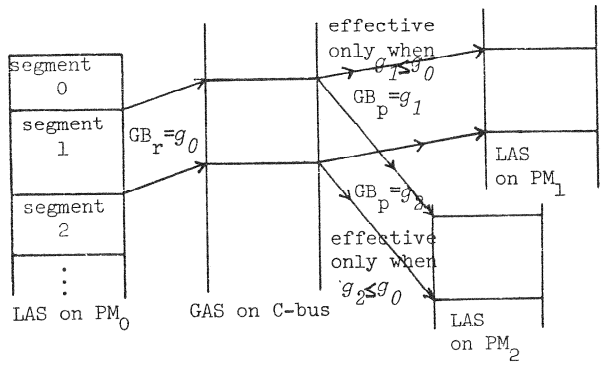


Fig. 2 An example of address mapping
(GB_r:request globality, GB_p:port globality)

3.3 Globality

In a modular system a hierarchy of modules is often wanted. In order to achieve a hierarchcal system among the modules connected to a C-bus, two-bit information which shows how global it is, is transmitted with each communication in ACE. This information is called 'Globality' (GB) and each module except the requesting one compares the transmitted GB with its own port GB, and after address recognition participates in communication only when the received GB is equal or greater than its port GB. This GB information augments communication flexibility.

3.4 Block transmission

Data transfer on a C-bus must be carried out at a very high speed. It is usual for a bus method to require a long time to establish transmission because of request arbitration etc. But once the path is organized, data transmission at a relatively high speed is possible. Therefore, increasing the number of transmission units is effective in augmenting the average transmission rate. However, as a bus is a common resource, too many continuous transmission units cause other requesting modules to have long waiting time. Considering this, we limited the nimber of transmission units of one request to sixteen, where the maximum waiting time is then about 4μs at a 5mhz transmission rate.

3.5 Multipriority request

The usual method of arbitrating simultaneous requests is to give a priority to each module and select the one with the highest priority. This common priority scheme assigns priorities to modules. But it seems desirable to assign priorities to transmission re-

quests, rather than the modules. Therefore we decided that ACE C-bus should allow each module to issue a request with one of two priorities. Consequently, the relative weight of priorities of these modules is dynamically changeable and therefore the bus can be used more effectively (3).

4. STANDARD PROCESSOR MODULE

As the Standard Processor Module (PR-S) is the basic processing component of ACE, it must have processing power and be easily adaptable to a wide range of problems. Therefore, it is constructed with a newly designed MicroProcessing Unit (μPU), four chunks of high speed memory (256 32 bit words/chunk) and a considerable amount of supporting control circuits which provides PR-S with various new general-purpose emulation oriented facilities and powerful communication capabilities.

Fig. 3 gives a basic block diagram of PR-S minus the μPU. Communication with the μPU is via the two ports shown at the lower left corner for microinstruction supply and those at the lower right for data. The two interfaces shown at the upper portion indicate the connection to C-buses. The major characteristics of the PR-S architecture are summarized below.

(1) Dynamic microprogramming by microcache.
(2) Cache for data buffer with 'cachability' control.
(3) Local memory.
(4) Separation of local address space from global address space.
(5) Segmentation at microprogram level.
(6) Variable length data unit.
(7) Selection facility for external requests.
(8) Two (expansible to four) C-bus interface ports.

We have already described items (4), (7) and (8). The remaining items are described in the rest of this paper.

4.1 Dynamic microprogramming

In order to give the PR-S its personality dynamically, it is provided with a dynamic microprogramming facility. From several possible methods of implementing it, we have chosen a way which gives a very large microprogram address space (8 segments, each can hold 8k steps) and allows storing microprograms in the same address space with data or macro level programs. This is not only flexible but allows microprogram sharing very easily. However, one of the problems we must cope with is caused by access time; if microinstructions must be fetched through the C-bus one by one, the system's operating speed would be too slow. We have solved this problem by using a chunk of high speed memory as PR-S's own microcache, which holds 256 microinstructions.

Microinstructions are fetched as sixteen instruction block through the C-bus when a microinstruction requested is not in the microcache. Table 1 summarizes the microcache parameters, which were determined by several preliminary simulation studies and the conditions unique to the microprogram. We have developed a new simple replacement algorithm, called Variable

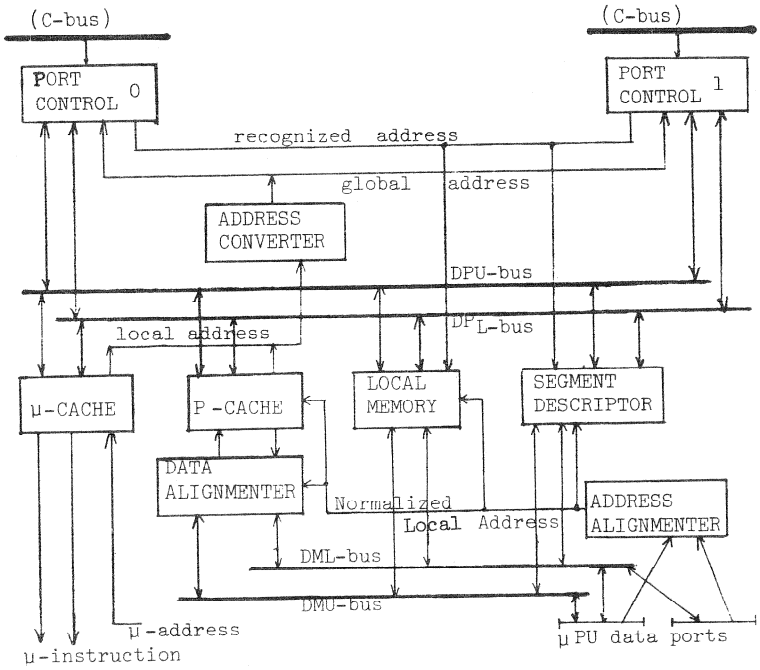


Fig. 3. A block diagram of the processor module.

Thresold Recently Used (VTRU), some simulation studies (4) of which have proved its effectiveness.

4.2 Data cache

Sharing of information in (main) storage is usual on multiprocessor systems, where the decrease in access time is main concern. For this purpose, ACE PR-S utilizes two chunks of high speed memory as a data cache, whose major parameters are shown in table 1.

Although the cache is a rather common technique to speed up the access time of uniprocessor systems, they have been rarely used in multiprocessor environment. The reason for this consists in the difficulty of solving the problem of altering shared information. When a processor issues a write command for some information, memory must check whether its copy exists in caches of all other processors and must simultaneoysly change the contents of all existing copies to keep data integrity. This is a very troublesome, hardware-and-time consuming procedure. Therefore, we determined to put a two bit cache control quantity, called 'cachability' in each segment descriptor, which the hardware references and processes in a manner explained in table 2. Although the control hardware is increased somewhat, the hardware can utilize data reference characteristics to avoid useless swapping and to improve efficiency.

Table 1. Parameters of microcache and data cache.

	microcache	data cache
capacity	256 w(32bits/w)	512 w(32bits/w)
mapping	set assoiative	set associative
no. of sets	2	32
blocks/set	8	2
block size	16 w	8 w
replacement algorithm	VTRU(7)	LRU

cachability		READ		WRITE	
		exist in cache	not in cache	exist in cache	not in cache
00	do not use cache		fetch from M_p		store into M_p fetch from M_p, then store into M_p, if data length is not integral value of 16
01	store immediately	fetch from C_d	allocate C_d block fetch from M_p	store into C_d and M_p	↑
10	store at swap	↑	↑	store into C_d set modify bit on	allocate C_d block fetch from M_p and store into C_d set modify bit on
11	never store	↑	↑	store into C_d	allocate C_d block fetch from M_p and store into C_d

Table 2. Cachability and read/write treatment. (M_p:primary memory, C_d:data cache)

4.3 Local memory

The last chunk of high speed memory is used as temporary storage and inter-module communication area. This local memory is subdivided into eight 32 word blocks and each of these is controlled by a 4 bit block descriptor.

4.4 Segmentation

The Local Address Space (LAS) used by each module and the Global Address Space (GAS) used for inter-module communication should be separated to allow a high degree of expansibility, flexibility and adaptability to the required structures. For the purpose of address translation from LAS to GAS, we have adopted segmentation because the basic philosophy of ACE insists on utilizing logical characteristics of stored information for hardware control, and, from such a viewpoint, segmentation is clearly more convenient than the physically simpler paging system.

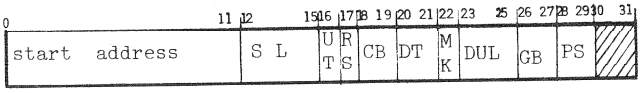
4.4.1 Two-unit segmentation

Since the Normalized Local Address (NLA, see 4.5.2) is divided into a 4 bit segment number field and a 20 bit word number field, sixteen segments are available to each PR-S. However, segment #0 is used for the local memory and registers, and is never converted into GAS.

The size of a segment must be variable according to the requirements of the stored information, but at the same time ease and effectiveness of allocation and address translation are also necessary. Therefore, we have developed a new two-unit allocation method, which requires a simple addition at address translation in contrast with the 2's power method of Fuller's computer module (2), but still can decrease the average worst case storage loss to 15% instead of 50%. This method uses two kinds of unit, called 'L' (2^{16}bits for PR-S) and 'S' (2^{12}bits), as basic sizes and allows 1 to 2^4 multiples of 'L' or 'S' as segment length. This means that any segment of PR-S takes one of the following 31 lengthes.

$S(2^{12})$, 2S, , 15S, $16S(2^{16})$
||
L, 2L, , 15L, $16L(2^{20})$

Accordingly, when all segments have the largest size of 16L, an LAS becomes the maximum size of about 2 mbytes.

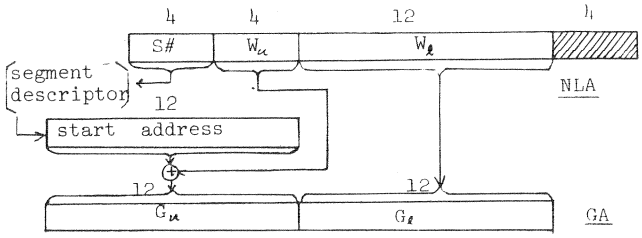


SL:segment length, UT:unit type, RS:reserved, CB:cachability, DT:data type, MK:mask, DUL:data unit length, GB:globality, PS:port selection.

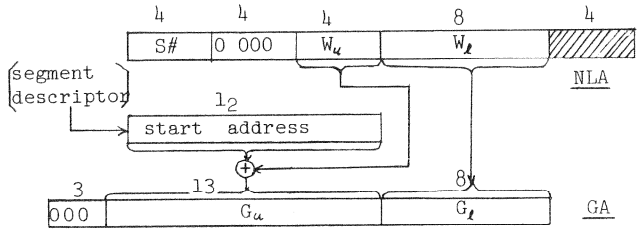
Fig. 4. Segment descriptor format.

4.4.2 Segment descriptor and address translation

The attributes of each segment such as the unit type, segment length, start address etc. are kept in a 32 bit segment descriptor shown in fig.4. The start address is twelve bits long, and its address unit is 2^{12} bits for 'S' and 2^{16}bits for 'L' respectively. Therefore, each GA can be expressed in twenty-eight bits. Actually, 24 bit length addresses are used on a C-bus because the minimum unit length is sixteen bits there. According to this rule, address translation is carried out as shown in fig. 5. Though only the



(a) L-unit



(b) S-unit

Fig. 5. Primary memory address translation.

case of primary memory addresses is shown, the case of microinstruction addresses is just the same except that local addresses are instruction addresses in unit of 32 bits instead of NLA.

4.5 Data alignment

One of the trends of new computer applications (languages) is proliferation of data types, and emulation must treat data of various sizes. Under these circumstances, it is absolutely clear that byte and word addressing alone is unsatisfactory, and the treatment of variable length fields must be attached more importance. Therefore, PR-S is provided with a hardware facility for aligning the accessed data automatically.

4.5.1 Data alignment specification

(1) Data unit size set-up
Data unit size is indicated in each segment descriptor. As a consequence, each segment can have only one data unit length, but a direct length specification is not necessary for each access.

(2) Selection of data unit sizes
As data lengthes, only 1,2,4,8,16 and 32 bits are allowed. Although other sizes may be supported by microprograms, our decision to support only these data lengthes in hardware is a practical cost/performance tradeoff rather than allowing all possible lengthes. A program can address a memory location in terms of data unit size chosen for the corresponding segment. Thus, address updating can be carried out in the same manner regardless of the data unit size (see 4.5.2).

(3) Effective data unit length
A user is allowed to select one of two modes of effective data unit size. This is controlled by the mask bit in the segment descriptor. If the mask bit is on, then only the specified number of bits are effective and the remaining part (data transmission through μPU port is either 16 or 32 bits) is masked out to zero. If, on the other hand, the mask bit is off, bits between 16 and 32 (the exact number depends on the actual address and the data unit length) from a designated bit location towards lower address are transfered. Accordingly, in the case where the data unit length is one (i.e., bit addressing) and the mask bit is off, sixteen bits starting at the specified location are given, thus giving a microprogram ability of accessing any number of bits starting at an arbitrary bit location by repeatedly using this mode of addressing.

(4) Limitation
Except for segments whose cachability equals '00', data alignment crossing a 256 bit cache block boundary is not allowed.

4.5.2 Address alignment

As described above, confining the data unit length to a 2's power permits a user to describe a data location in terms of the data unit of the corresponding segment. The PR-S address alignment hardware converts the user supplied LA to a length-independent bit address, called Normalized Local Address (NLA), by shifting the given address to the left by an appropriate amount and inserting 1's into vacated position(s). This procedure is shown in fig. 6.

4.6 Port control

PR-S has four C-bus ports with the same specification (only two were actually implemented on the first model), but only one of them can be active at a time since there is no buffering facility at the ports. As we have already described the C-bus communication, only a short discussion of the mechanism of GA recognition at the ports is given below.

GA recognition at each PR-S port is designed to be simple and fast. Actually, only equal comparison of the upper twelve bits of the GA issued on the C-bus with the information stored in port registers (see fig. 7) is executed. There are two such port registers at each port and if one or both of the active registers recognizes an address match, the match signal is returned to the C-bus controller and the lower twelve bits of the GA are used to address the PR-S local high speed memory and registers. The status information concerning an attempted access through the C-bus is recorded in a hardware register and the microprogram can be interrupted by an access occurrence.

The address recognition process described above implies that ACE basically uses a mail-box inter-module communication method. It is true that this process requires some time for communication, however, the flexibility of communication obtained is big. Moreover, this communication method is well-suited for modular multiprocessor systems, in general.

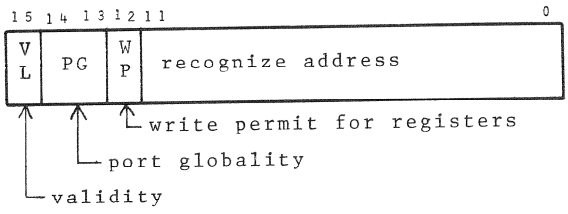
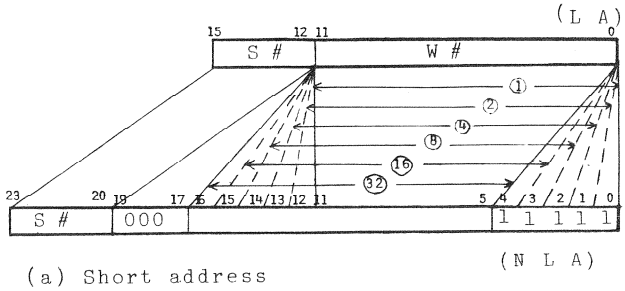


Fig. 7. Recognize address register format.



(a) Short address

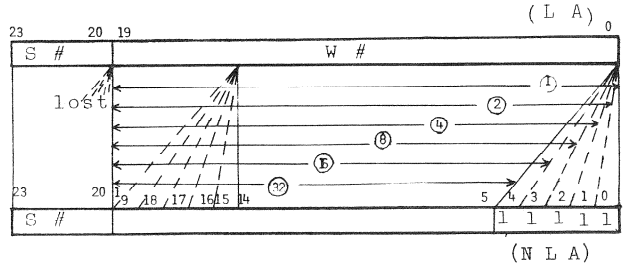


Fig. 6. Address alignment.

5. CONCLUSION

We have described a new modular computer system and the architecture of its basic processing module. They are designed to have a high degree of adaptability to application problems. The first version of ACE (ACE 0.1) which is currently under construction will be configured as shown in fig. 8. Its hardware is supposed to become operational by June, 1975. The system has only one PR-S, but its I/O processor with adapter behaves like a processor module as far as C-bus communication is concerned. In this sense, ACE 0.1 may be considered as a two processor system.

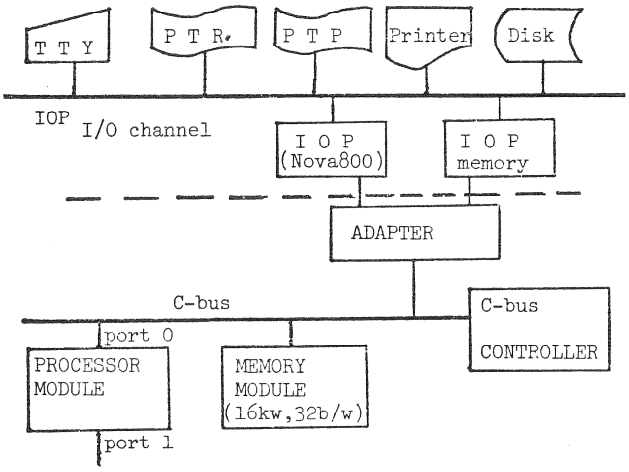


Fig. 8. The configuration of ACE 0.1.

ACE is an experimental system which was designed with power and flexibility rather than the cost as the primary considerations. Consequently, we do not claim that the system is optimized with respect to cost/performance, in particular as a commercial computer must be. However, we believe the directions and the new facilities tried out in ACE may prove to be important and may appear commonly in the near future.

Most of the work already done for ACE was related to its architecture and hardware. Unfortunately, we cannot yet describe the software of ACE in this paper. The study of ACE has just begun and much more work on hardware (e.g., more PR-S's in a system), software (e.g., a multiprocessor operating system, high-level language oriented machine, etc.) and evaluation (assesing the design parameters by simulation, especially those of the system with more than 10 PR-S's) must be continued.

ACKNOWLEDGMENT

The research on ACE has been supported by the direct or indirect aid of many persons. The authors gratefully acknowledge the specific design contribution of Mr. Sakamura. Helpful ideas and criticisms were provided by Profs. H.Aiso and T.Motooka. The authors also wish to express appreciation for the support and encouragement provided by Drs. H.Nishino and K.Kurokawa.

REFERENCES

[1] F.E.Heart, S.M.Ornstein, W.R.Crowther, and W.B. Barker, A new minicomputer/multiprocessor for the ARPA network, AFIPS Proc. NCC, vol. 42, 1973, 529-537.

[2] S.H.Fuller, D.P.Siewiorek, and R.J.Swan, Computer modules: an architecture for large digital modules, Proc. of the first annual symposium on computer architecture, December, 1973, 231-237.

[3] M.Pirtle, Intercommunication of processors and memory, AFIPS Proc. FJCC, vol. 31, 1967, 621-633.

[4] H.Iizuka, and K.Sakamura, Variable threshold recently used replacement algorithm and its performance simulation, Johoshori, vol. 15, no. 11, November, 1974, 900-902, (in Japanese).