マルチプロセッサシステムMICS－II による並列処理

A VIRTUAL MACHINE COMPLEX MICS-II AND ITS APPLICATION FOR PARALLEL PROCESSINGS

大森健児　小池誠彦　山崎竹視　大宮哲夫

KENJI OHMORI, NOBUHIKO KOIKE, TAKEMI YAMAZAKI AND TETSUO OHMIYA

日本電気中央研究所

CENTRAL RESEARCH LABORATORIES, NIPPON ELECTRIC COMPANY

1. INTRODUCTION

In recent years, the expansion of mini-computer fields has brought about two kinds of applications with which a single processor cannot cope. One requires high speed processing capacity, the other various functions.

The Interface Message Processor (IMP) in the ARPANET is a typical example which requires high speed processing capacity. The Pluribus IMP has succeeded in obtaining enough processing speed by load-sharing among the processors.

On the other hand, a mini computer in a laboratory, which is used to solve differential equations, to store and analyze data collected from measuring instruments, or to control testing devices, is required to have various functions. User requirements always expand in such a way that several users, who have differing kinds of applications, want to use the laboratory computer simultaneously. The traditional way wherein an old mini computer is replaced with a high level computer, when user requirements exceed the mini computer capacity, does not hold true any more. Laboratory computer applications require delicate process controls, in which the high level computer is weak.

The virtual machine complex--MICS-11--has been developed to satisfy such demands. The MICS-11 implementation has been due to the development of the LSI technology, especially microprocessor and IC memory technology.

The main advantage of the MICS-11 is to allow several users to work simultaneously on various applications. The MICS-11 can be considered a various purpose multi-user service system in comparison to the traditional calculation oriented time sharing system.

The MICS-11 consists of 6 processing modules, 64Kw main memory and 12 input output devices. It has been operational since May, 1977.

2. SYSTEM ARCHITECTURE

The traditional idea, in which a user program is executed under the control of an operating system, cannot be accepted as the MICS-11 architecture. The MICS-11 allows bare-machine usage wherein a user can put all mini-computer level instructions to use without any restrictions. Therefore, a virtual processing system (VPS) and a distributed system control (DSC) have been developed.

A VPS, which is used by each subscriber, provides him with the same processing environment as a laboratory computer does. However, the VPS structure is not fixed, but is changed in the course of program execution. The VPS logical structure can be changed to meet user requirements, so that a user can construct his own instruction set. The hardware structure of the VPS consists of the nucleus and dynamically allocated system resources.

It is not performance effective to construct a fixed structure processing system, wherein any program can be executed. In such a system, most resources are wasted without achieving the desired effective utilization. Therefore, the MICS-11 has been so designed that expensive resources and less often utilized resources are pooled as system resources which are shared among several users. The VPS nucleus consists of a microprogram control processor and an 8 K word local memory. System resources include memories and I/O devices.

Full processing environment is not made available to a user immediately. Only the nucleus is furnished the user at the beginning of execution. This method is used because it is cumbersome for a user to declare his own processing environment in advance. The required resources are dynamically connected to this nucleus when the program cannot be processed in the current processing environment.

The main function of the DSC is to connect system resources to each nucleus in such a way that an effective processing environment is always furnished each user. The DSC consists of two management systems, event management and resource management. The event management extracts user requirements when user programs exceed its own current processing environment. The event management is controlled in the distributed form and the resource management is controlled in the centralized form. For each nucleus, several hardware modules for the event management have been implemented to observe execution status. Resource management is executed by the management oriented processing system, which has the same structure a VPS has.

3. SYSTEM CONFIGURATION

The MICS-11 takes the form of a homogeneous multi-processor system, as shown in Fig. 1. It consists of up to 8 processing modules (PMs), memory modules (MMs), disk units, I/O devices and three kinds of buses: memory buses, I/O buses, and a control bus.

As shown in Fig. 2, each PM is physically separated into two different modules: a control module (CM) and a user module (UM). The UM, where a VPS is generated, executes event management tasks for its own VPS. A PM, whose UM executes a user task, is called a user processing module (UPM). One whose UM executes system resource management tasks is called a system processing module (SPM). One SPM exists in the system, which is assigned dynamically at system generation time or system reconfiguration time for error recovery.

A memory module consists of a 32 K word RAM, whose memory cycle time is 700 nsec. Up to 8 memory modules can be supported by one memory bus. The MMs are used as an extension memory for the local memory in a UM or common area to enable sharing programs or data among tasks.

A memory bus, I/O bus and control bus have 50, 28 and 24 bi-directional signals, respectively. The dual-bus scheme is employed for memory access and I/O access, so as to reduce the bus traffic and to increase reliability.

(i) User Module

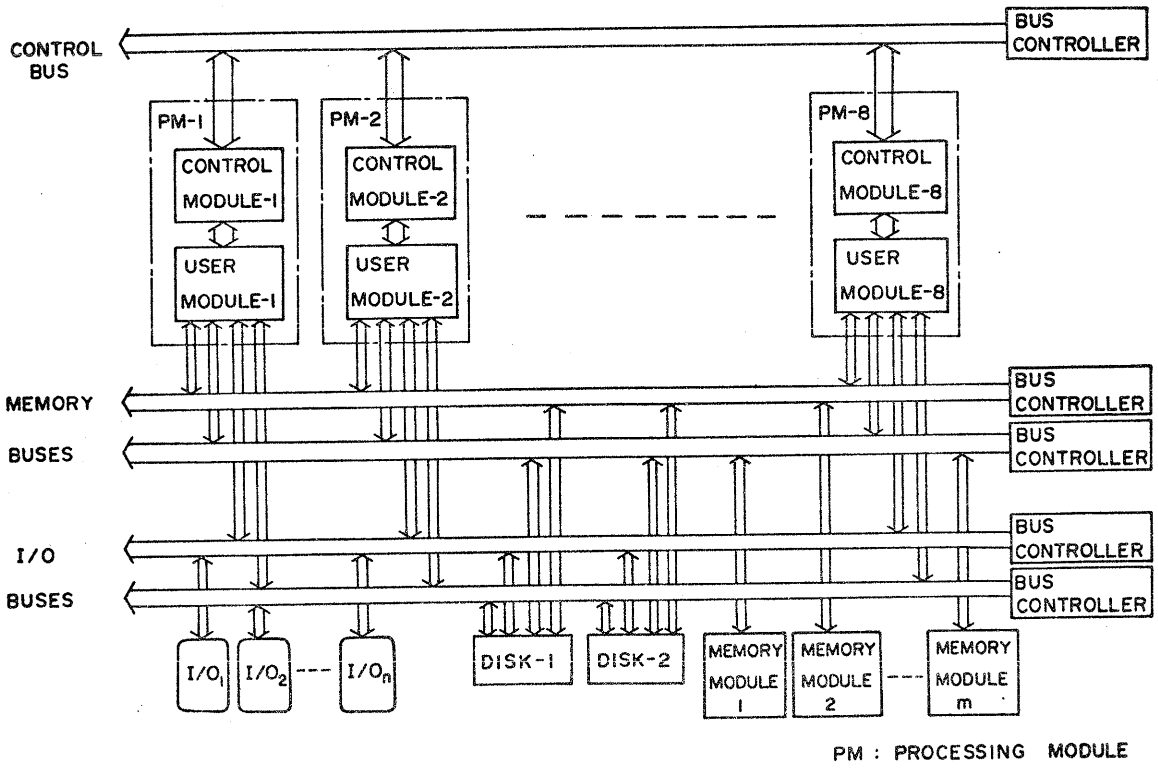


Fig. 1. MICS-II blockdiagram

A user module is composed of a user processor (UP), a local memory (LMM), address translators (ATs) and I/O ports (IOPs). The user processor is a single board, micro-programmable processor which has access asynchronously to memory and I/O devices. The memory space is divided into 32 pages. The first 8 pages are assigned to the local memory and others are left for assignment to the MMs. A UP which is included in another UM is not allowed to have access to the LMM, so that full security is given to programs and data in the LMM.

An AT translates a logical processor address into a pysical memory address. An illegal memory access for the MM is detected by checking two control bits when a logical address is translated into a physical address. One control bit shows whether the MM memory space is assigned for the logical address, the other shows whether memory write access is allowed. When a memory access fails to satisfy two conditions, page fault or write access error is detected.

An IOP is an interface module between the UM and the I/O devices. When the UP tries to attain access to an I/O device, which has not been assigned to its VPS, the IOP halts the UP and informs the CM of the illegal I/O access.

(ii) Control Module

A control module is composed of a control processor (CP), an 8K word control memory (CMM), an inter-processor port (IPP), a console module (CON), a communication module (COM) and a bus window module (BWM). The CP is a low level processor of a UP. The CMM is used to store control programs for the CP.

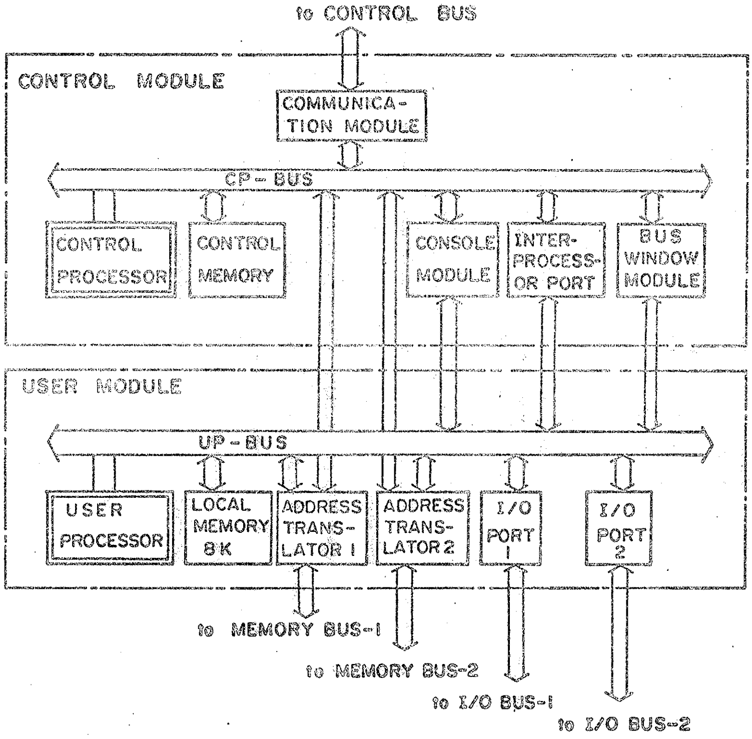


Fig. 2 . Processing module blockdiagram

The IPP, which is composed of dual 4*8 bit message files, is used to exchange messages between UM and CM. If a PM functions as a UM, the IPP is used to transfer data between user tasks in order to allow a user to accomplish parallel processings. If a PM functions as the SPM, it is used to transfer control messages for the system management.

The CON enables the CP to control the UM, to load a microprogram in the UP and to detect illegal UM conditions, such as UP halt, execution of an endless indirect instruction and UP faults. The COM enables the CM to transmit or receive messages to/from another CM. Important DSC activities are performed by this module.

The BWM enables the CP to have access directly to the local memory. However, the BWM facilities are allowed only for the SPM, so that the SPM can handle system resource management tasks effectively by load-sharing between the CM and the UM.

4. SYSTEM CONTROL STRUCTURE

Each user can use all the resources in the system and creat his own processing environment. The DSC is required not only to provide a user with a proper processing system, but also to resolve contentions regarding shared resource usage, and to prevent task interferences and deadlocks.

The facilities for the DSC are physically separated from those for user tasks. Their activities are

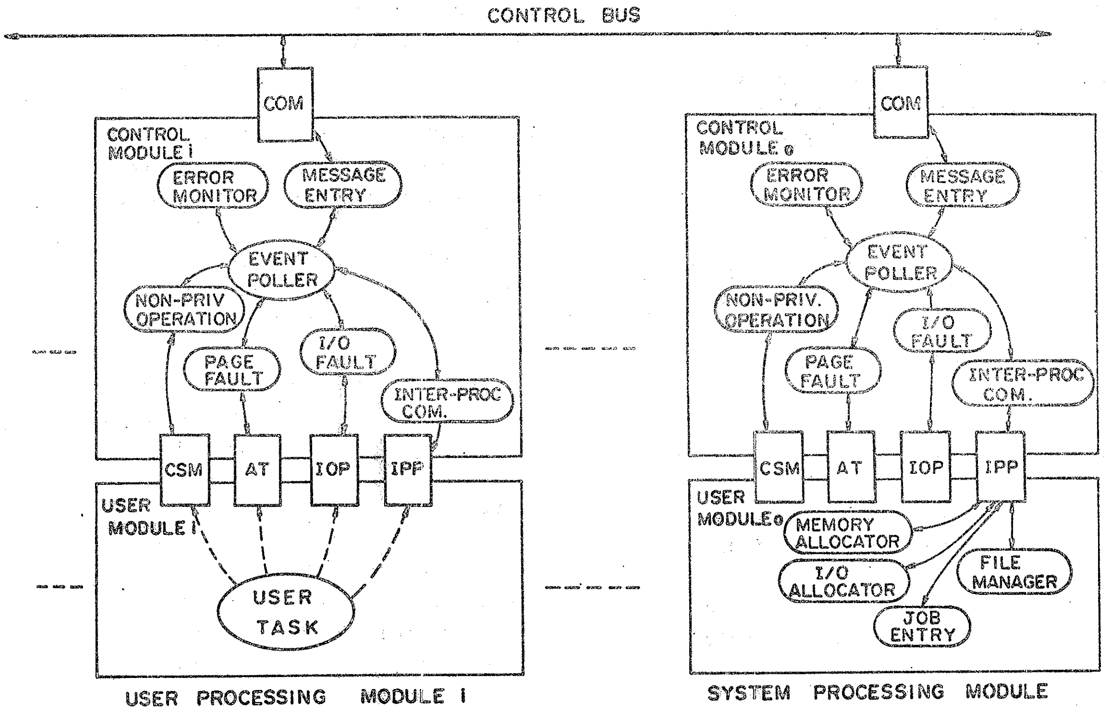


Fig. 3. System control structure in MICS-II

performed by the CMs and the SPM. Thus, task oriented VPS can be created to meet user requirements and hiqh reliability is achieved.

The DSC initializes sub-modules in the UMs, assiqns a UM for a user task as the nucleus of a VPS, allocates resources for user task as VPS components, provides an inter-processor communication means, detects erroneous modules and isolates them. The DSC structure is shown in Fiq. 3. System resources are manaqed in the centralized form at the SPM and events are manaqed in the distributed form at each CM.

A manaqement oriented processinq system is created in the UM of the SPM. It performs job entry service, VPS creation, memory allocation, I/O allocation and file management.

In each CM, sub-modules, which have been implemented by specially desiqned hardware, are used to detect events when the UP is attemptinq access out of its own processinq environment. An event poller and correspondinq event handlers, which reside in the CM's control memory, are:

(i) Paqe Fault Event. Occurs when a UP has attempted access to a paqe for which the MM memory space is not assiqned.

(ii) I/O Fault Event. Occurs when a UP has attempted access to an I/O devices which is not furnished for its current VPS. It is detected by the I/O port. The CM requests the SPM to assiqn the device, or informs the user of an I/O fault and stops the UP.

(iii) Non-priviledqed Instruction Event. Occurs when a UP has tried to execute a non-priviledqed instruction which is not allowed for its VPS. It is detected by the console module. The CM informs the user of a non-priviledqed operation and stops the UP.

(iv) Inter-processor Communication Event. Occurs when a task requests communication with another task, which may be executed in another VPS. It is detected by the inter-processor port. The CM forms a communication messaqe and transfers it to the desiqnated CM via the control bus.

(v) Messaqe Entry Event. Occurs when a messaqe has arrived from some other UPM. It is detected at the communication module. The CM receives the messaqe and outputs to the UM via the IPP.

Once a VPS is created for a task, almost all processinqs are allowed in the VPS. Therefore, several events may occur in the course of execution. In such a case, the CM detects them and informs the SPM via inter-CM communication. Then, the SPM performs adequate resource allocation, avoidinq resource contentions, task interferences and deadlocks.

5. INTER-CM COMMUNICATION

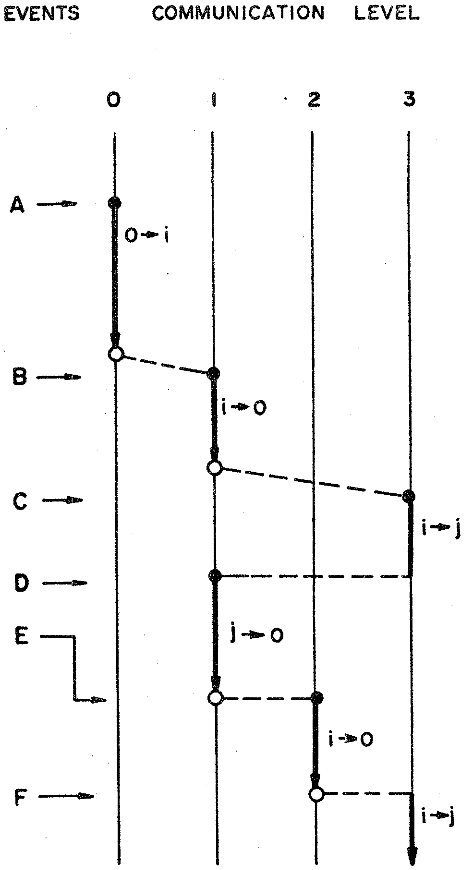
A communication module supports messaqe communication amonq CMs. An element switchinq transmission system (ESTs) has been developed so as to transmit various kinds of messaqes by means of low level protocols.

The ESTs enables a CM to transmit one element at a time. An element is composed of 4 bit functon code, 4 bit destination name, 4 bit source name and 8 bit data.

The low level protocols are composed of communication link establishment, messaqe transfers and communication link release. The element type is specified by the function code. At first, a COM starts to send a communication link establishment element, specifyinq a communication priority level and a destination COM name. The COM whose name coincides with the destination name becomes the desianated one. If the desianated COM is readv for communicaton, a S ACK is returned to the sender and a communication link is established. If the desianated COM reauires a hiaher level communication link, an NS ACK is returned to the sender and the communication link is established. If the desianated COM reauires a hiaher level communication link, an NS ACK is returned to the sender and the communication link reauirement is reiected. If the desianated COM is establishina a communication link with another COM bv means of a lower prioritv level, the desiqnated COM is interrupted for hiqher level communication, releasina the current communication link.

When the communication link is established, the source name and the priority level are stored in the command reaister (CMD) of the desianated COM.

After establishina a communication link, the sender COM sends a messaqe transfer element to the desiqnated COM. To protect aqainst violation from other COMs, the source name is checked before receivina data.



(A) Job entry for task i occurs.

(B) Page fault occurs in VPSi.

(C) Inter-process communication is started between VPSi and VPSj.

(D) Page fault occurs in VPSj.

(E) I/O fault occurs at VPSi.

(F) Inter-process communication is resumed between VPSi and VPSj.

Fig. 4. Inter·CM communication

If the source name does not coincide with the name stored in the CMD, an NS ACK signal is returned to the sender.

The communication link is released in a similar way. If the communication link is released, the designated COM sets up an adequate priority level to prepare for another communication link. A low level protocol is constructed for system initialization message, control message, error recovery message, page fault message, illegal I/O access message or inter-user process communication message.

Each element is transfered at an 800 nsec speed, so that task execution is not degraded by the communication control.

An example of communication is shown in Fig. 4. At time A, VPSi is generated for newly arriving task i. Control messages are transfered according to priority 0 from CMo, which is a part of SPM, to CMi so that CMi supports VPSi. At time B, a page fault occurs in VPSi. Page fault messages are transfered with priority 1 from CMi to CMo, so that memory module area is assigned to VPSi. At time C, task i requires communication with task j on VPSj, which is supported by CMj. Inter-user process communication messages are transfered with priority 3 from CMi to CMj. However, a page fault occurs at time D in VPSj during the task communication. The task communication is interrupted, and page fault messages are transfered from CMj to CMo. At time E, an illegal I/O access occurs in VPSi. However, the illegal I/O access message transfer is held up until the page fault process finishes, because page fault messages have a higher priority level than illegal I/O access messages have. After finishing the transfer of illegal I/O access messages, user process communication is resumed.

6. ERROR RECOVERY

The MICS-11 is physically organized as a very closely-coupled multi-processor system, where memory and I/O devices are shared with processors. Special consideration has been given to fault-tolerance, so that the system can continue its processing with a minimum of degradation without causing total system down, even when faults occur in several modules.

(i) UM faults are strongly protected against by the CM. By separating system control from UMs, UMs cannot change memory address table or I/O address table so that faults in a UM do not affect other modules. Each CM has facilities to monitor the UM, to check its errors, and to isolate it.

(ii) Though system resources are managed at the SPM in the centralized form, the SPM is diagnosed by other CMs and could be replaced by another PM in case of faults.

(iii) Memory buses and I/O buses are diagnosed by the SPM periodically. When any faults are detected, the system is reconfigured and can survive using remaining buses in a slightly degraded form.

(iv) For the control bus, a parity bit is provided in the function field of a message. Also, a hardware watchdog timer and a software watchdog timer are used to detect any faults. When a parity error is detected, whole messages are re-transmitted. However, when the hardware or software watchdog timer has detected time out, the DSC reconfigures the whole system.

(v) For user program errors, the

MICS-11 provides several facilities to help user debugging. Among them, a control panel, which is provided for a user by each CM, is very useful for laboratory control usages. It is also possible for the CM to detect an error and inform the user when a task has run out of the user defined memory boundaries, when it could not get out of infinite indirect loops or when it caused a write error.

7. PARALLEL PROCESSING

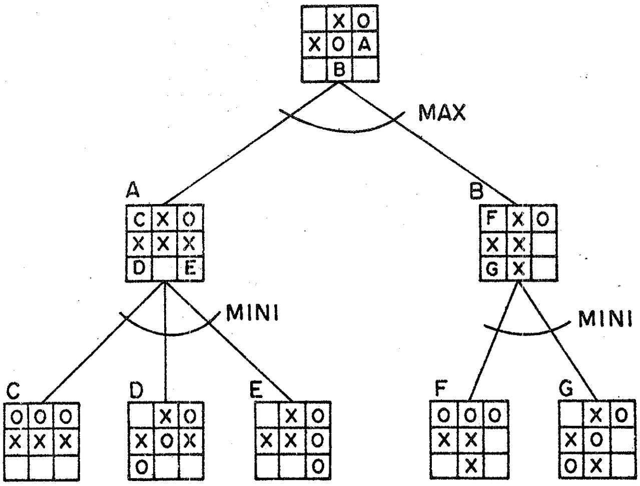
The advantage of the MICS-11 lies on the application fields for parallel processings. The MICS-11 is a closely-connected multi-processor system. The multi-processor system has such characteristics that several processors run simultaneously on a common data base. Especially, in a closely-connected system, parallel processings are realized by means of load sharing or function distribution. However, the parallel processings on the multi-processor system differ from ones on a parallel processing machine, such as ILLIAC-IV. In case of the parallel processing machine, the parallel processings are realized by an instruction level. On the other hand, in case of the multi-processor system, the parallel processings are realized by a task level.

Therefore, the application fields for the multi-processor system lie on artifical intelligence, pattern recognition etc.

A reverse game is selected as one of the examples which demonstrate efficacy of the multi-processor system MICS-11 on the parallel processing fields. The minimax procedure is applied to obtain a solution for the reverse game.

7.1 Minimax Procedure

As an example, shown in Fig. 5, assume that a minimax procedure is to generate the top two levels of the game playing tree starting with the original position and to get an optimal move from the game playing tree. The procedure starts by generating successors A and B of the original position. It then generates successors C,D and E of successor A and obtains values for successors C,D and E by using the evaluation function. The procedure



Fiq.5. Minimax procedure

assigns the minimum value among the values for successors C,D and E to the value for successor A. In the same way, the proceddure obtains a value for successor B. Comparing the values for successors A and B, the procedure determines the maximum valued successor to the next move.

7.2 Processing Structure

The minimax procedure is prohibited from being executed in parallel, because the subprocedures, which the minimax procedure consists of, are recursively called.

Therefore, a process generation/elimination method has been developed for the reverse game play. There are three different processes in the reverse game program.

A. Search next moves for a given position.
B. Obtain the value for a termination position.
C. Obtain the value for a minimax position.

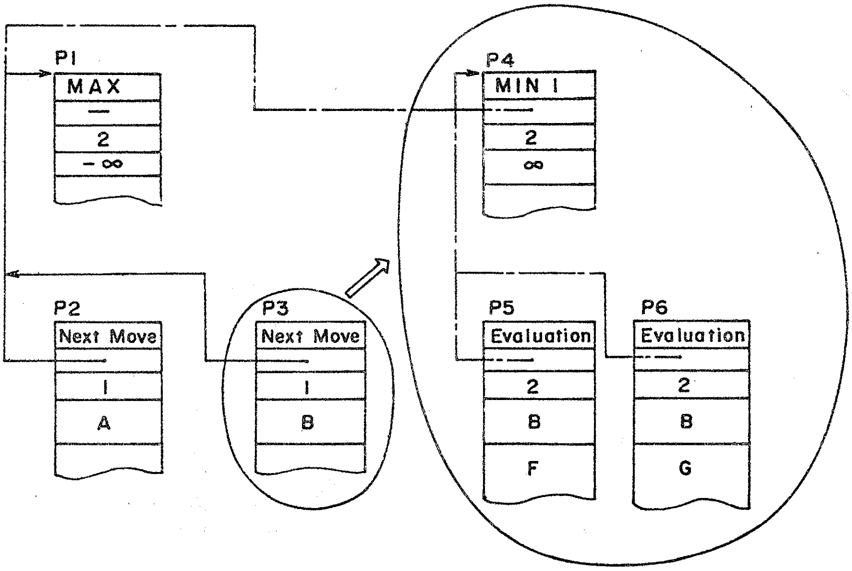
These processes have the following characteristics.

Process A starts to search the successors. It generates process C to assign the value of the highest valued successor to this position. It then generate process A OR C for each successor and eliminate itself.

Process B starts to obtain the value from the evaluation function. It hands over the value to its corresspondinq process C and eliminates itself.

Process C searches the minimum or maximum value among the values which have alreadv been handed over. When all values are compared, it hands over the minimum or maximum value to the correspondinq process C.

To clarifv these characteristics, the example in Fiq.5 is aqain described. Assume that successors A and B are qenerated and process A (P3) is now readv to start. The transition for the process block is shown in Fiq.6. P3 qets positions F and G for the successors to successor A. It qenerates process C (P4) to obtain the value for successor A. P4 is connected to P3 by the pointer so that the value of P4 is handed over to P3. For positions F and G, the process Bs



Fiq.6. Process control block transition

(P5,P6) are qenerated to obtain the values from the evaluation function. P5 (P6) is connected to P4 so that the value for P5 (P6) is handed over to P4. Then, P4 eliminates itself.

7.3 Exclusion Control

In the actual proqram, the process control block is stored in the main memory. Each UP takes an executable process from the main memory and executes it on the local memory. In this method, the process control block is destroyed, if UPs simultaneously have access to the block. Therefore, the process block has to be used exclusively among UPs. For this purpose, an exclusion control system, as shown in Fiq.7, is connected to the I/O bus. Each UP has access to the execlusion control system prior to havinq access to the process control block. The exclusion control system returns "permission" to the UP if no UP has access to the exclusion control system. Once the exclusion control system returns "permission", it never returns "permission until the UP, which qot the "permission", resets the execution control system.

In the reverse qame play proqram, the efficacy of the MICS-11 is confirmed. That is, the processinq speed rises in proportion to the number of UPs.

8. CONCLUSION

The hardware development phase for a prototype of the MICS-11 has been finished. The first version of the control proqram has been operational since May, 1977. The prototype consists of 6 processinq modules and a 64K word main memory.

The MICS-11 is characterized by its extensive use of inexpensive processors and memories, as well as extensive sharinq of system resources and facilities. A powerful system control support has made it possible to provide a user-oriented and completly secure processinq environment for each user.

The success of the MICS-11 lies in the pursuits of modularities, both in hardware and in software, and the physical separation of user tasks and the system control. The MICS-11 has been realized to construct a hiqhly reliable and easy-to-use computer system for a laboratory.

ACKNOWLEDGEMENT

The authors wish to thank Mr. K. Kiji for his kind interest in this work and Mr. K. Nezu for his encouraqement and support durinq the period of this research. They also thank Mr. H. Minowa and Mr. T. Deki for their efforts in developinq the MICS-11.

This work was supported as a part of the research project of pattern information processinq system by the Aqency of Industrial Science and Technoloqy of MITI.

REFERENCES

(1) K. OHMORI, N. KOIKE, K. NEZU and S.SUZUKI, "MICS-A MULTI-MICROPROCESSOR SYSTEM" INFORMATION PROCESSING 74, pp98-102.

(2) K.OHMORI, N.KOIKE, T.YAMAZAKI, T.OHMIYA and K.NEZU, "MICS-11 --VIRTUAL MACHINE COMPLEX" COMPCON SPRING 1978 to appear.