MIMD FEATURES SUPPORTED IN NEC'S COMPUTER SYSTEMS

Akihiro IWAYA
EDP Planning Office
NEC Corporation
33-1, Shiba 5-chome
Minato-ku, Tokyo 108
JAPAN

Tadashi WATANABE
Computer Engineering Division
NEC Corporation
1-10 Nisshin-cho, Fuchu,
Tokyo 183
JAPAN

Abstract

In this paper, NEC's development approach to multiprocessing in scientific computer is introduced by showing three computers as examples, where inherent multiprocessing (or MIMD) features are incorporated. Computers introduced are; ACOS Mainframes, Supercomputer SX system and NEDIPS Dataflow computer. These computers were developed according to the NEC's approach and now commercially available.

1. INTRODUCTION

NEC's approach to develop scientific computer series is always to offer these computers based on market requirements. It is observed user want to have a cost effective high-speed and yet easy to use scientific processing environment and to utilize many application programs already developed. It is also found major application areas now in use require only homogeneous computational environment (the same computation is applied to every grid points of the model) such as aerodynamics, even though exceptionally some applications such as Monte Calro type simulation require non-homogeneous computation.

On the other hand, in the computer technology field, speed limit of devices is not yet reached, connection delay may be further minimized by using VLSI and high-density packaging technology. New high-speed device such as GaAs will be available in the near future.

Considering above market requirements, applications and high-speed technology our approach for scientific market was decided in the following way;

. Offer a cost effective high-speed vector (SIMD) machine with high-speed scalar processing.
. Offer multiprocessor version of SIMD machine to increase the total performance. This may include not only homogeneous multiprocessor but also heterogeneous one.

. Multiprocessing system with hundreds or thousands of processors, central high-speed scalar processor and linearly configured main memory may be feasible, but we have to resort to the appearance of micro-processor with tens of megaflops. Yet high-speed connection network between main memory and processors have to be developed. (Sometimes called Von-Neumann bottle-neck problem)

. Further advanced type MIMD machine based on dataflow concept which may give the solution for Von-Neumann bottleneck problem is to be developed. This machine achieves instruction level (function level) concurrency and proves to be effectively operable.

According to the observation just described, three types of scientific computer, all of them have some sort of MIMD features was successfully developed and now commercially available.

2. MULTIPROCESSING IN NEC'S ACOS MAINFRAMES

NEC's mainframe computers called ACOS series, entry model to high-end model, all have multiprocessing features. In particular, high-end mainframes called S-950 and S-1000 support both vector processing and multiprocessing up to 4 processors. Fig. 1 shows the high-end mainframe's central complex configuration.

2.1 Process structure of ACOS mainframes

Multiprocessing function in ACOS mainframes is controlled by a mechanism called process control feature. A Process is defined as a sequence of instructions which are always executed without concurrency (at least conceptually), it may be seen this corresponds to the programming notation of a task. The programmer thinks of the ACOS series environment as the parallel execution of his program with others (multiprogramming) or together with the parallel execution of tasks (multitasking) within the program. (Dijkstra (1968))

Processes have to be created by system software, and made known to the processors by a specific 'start process' instruction. Each process has a process control block to serve as an area for firmware dispatching mechanism.

Event controls between processes are made by a mechanism called semaphore. The semaphore is a description located in a special segment whose data contents are used by firmware to represent a counter and a queue pointer. These semaphores are used for the following purposes with associated P-operation (wait) and V-operation (Signal);

- Dispatching processors for competing processes in multi-tasking environment
- the passing of messages between processes
- the control of areas which are not reentrable (monitor)
- the interpretation of I/O interrupts.

Fig. 2 shows the schematic view of semaphore operation. Many semaphores are defined for operating system purposes in the operating system kernel. Others can be included in object programs by the compilers.

2.2 Integrated array processor (IAP)

Vector processing feature called Integrated array Processor (IAP) is incorporated in high-end ACOS mainframes. This feature, which accelerates vector processing speed, is implemented by adding relatively small amount of hardware. Around 60 vector instructions such as vector Add, Subtract, Multiply, Divide, Inner-product, Sum and Find-max are supported. Programs written in FORTRAN are compiled to the vectorized object code by FORTRAN 77 compiler.

However in order to utilize multiprocessing feature described above, user have to partition his program into individual FORTRAN program which correspond to the process and final load module which may contain vector instructions and executable in parallel is generated at linking time.

3. HETEROGENEOUS MULTIPROCESSING IN NEC SUPERCOMPUTER SX SYSTEM

NEC Supercomputer SX System which offer up to 1.3 gigaflops processing power employs heterogeneous multiprocessor architecture in order to offer increased total system throughput. The functions of the SX System are allocated to two processors called arithmetic processor (AP) and control processor (CP). These two processors operate in parallel. (Watanabe (1984))

3.1 SX System Configuration

Fig. 3 shows the configuration of SX System central processing complex. Processor part called Scientific Processing Unit (SPU) consists of two processors; AP and CP. AP is a kind of high-speed FORTRAN engine dedicated to execute user program and CP is a supervisory processor to control the entire system.

The objectives to employ such heterogeneous multiprocessor configuration are described below.

Generally speaking, the internal processing part by processor (central processing unit) is divided into two types of operation;

. System control such as resource management and scheduling

. User program execution

In large scientific computation, significant reduction in user programs execution time may be expected by the use of high-speed vector and scalar processing power. This results in an increase of relative percentage of processor time for system control. Even worse, expensive vector processing function is not effectively utilized in this type of system control processing because most of the system control is processed as scalar operation. Fig. 4 (a) illustrates the internal processing time of conventional mainframe which exemplifies the above situation.

The SX system employs multiprocessing system and functions required for scientific processing are distributed to AP and CP. AP and CP share the Main Memory Unit and in each processor, process control primitives described in section 2 are implemented to offer multiprocess structures. System control process and user program process run in parallel in CP and AP respectively, turn-around time as well as total system throughput is considerably improved in SX system. Fig 4 (b) illustrates this situation.

3.2 Arithmatic Processor (AP)

AP is a heart of SX system and dedicated to run user program. AP is divided into the scaler unit and vector unit, each can operate in parallel.

The scaler unit includes 64 k bytes cache and 128 scaler registers and employ scaler arithmetic pipelines. The scaler unit interprets the instructions and controls their execution, and high-speed scaler operation is performed on it.

The vector unit includes 16 multi-parallel pipelines and 80 k bytes vector registers and executes vector arithmetic operation, which include extensive vector operations such as vector mask operation, scatter/gather operation in order to increase vectorizing ratio.

AP may be consided to be a FORTRAN engine optimized to run FORTRAN program. To maintain high-speed operation, AP employs so called 'RISC' (Reduced instruction set computer) architecture.

3.3 Control processor (CP)

CP performs operating system function. It controls AP, IOP (Input output processor), XMU (Extended Memory Unit).

The system control functions included are; resource management, scheduling, compilation and linking. Time sharing system called ATSS is also supported to use SX system from remote site.

CP employs the similar architecture to ACOS mainframes. It can be considered to be powerful main-frame processor. Fig. 5 shows the typical job flow and functional distribution between CP and AP.

Not only system programs but user program execution is also possible on CP. (FORTRAN 77 is supported for both AP and CP)

3.4 AP and CP overlapped operation

Although parallel operation of AP and CP is obvious under multi-programming environment, such operation is also possible within the operation of one user program, reducing the turn-around time. One such example is asynchronous input/output function which enables input/output process and user program process for a program run in parallel. Input/output process including format editing run on CP and simultaneously user program process on AP.

Fig. 6 shows asynchrous input/output operation where I/O operation (READ) is specified with identification (ID) and user program process proceeds until it has to wait the completion of data transfer with the identification. While user program process is running on AP, entire I/O processing is performed on CP until the I/O termination. This function may sometimes be very effective when user want to output intermediate results to the file system in order to reduce the turn-around time of user program.

Above is an example where system process (I/O operation) and user process run in parallel. User may also want to run one user process on CP in parallel with another user process on AP within one program. This is also possible. Each user's CP task and AP task have to be complied individually, then at link time they can be combined to form a load module with multi-process structure. Load-balancing between CP and AP is user's responsibility.

Heterogeneous tightly coupled multi-processor configuration of SX System with mainframe compatible processor (CP) and FORTRAN vector processing engine (AP) is, we believe, the one of the best choice considering the present state of the art. In SX System, actually no front-end system is required.

4. DATA-FLOW COMPUTER NEDIPS

Multi-processing system presented in section 2,3 exemplifie MIMD (Multi instruction Multi Data) in process level. However there exists MIMD or parallel operation possibility in instruction level even when the program is executed is scalar mode. Parallelism inherently found between instructions (or data) can be found by using data-flow concepts.

Data-flow computer called NEDIPS (NEC Data Flow Image Processing System) which employs so called non-Von-Neumann architecture was developed in NEC and proved to be very effective for scientific processing.

Data-flow computer where each instruction (operation) is processed in parallel, we believe, is the ultimate goal of MIMD processing (Temma (1984)).

4.1 Non-Von-Neumann type computer

Every computer commercially available today except NEDIPS uses Von-Neumann architecture. In this architecture, Principal operation is as follows; programs and data are stored in central memory and program execution is sequencially executed according to program counter. This principal operation often limits the system performance. One is the bottle-neck between memory and processor. Another is the sequential processing even though there exists inherent instruction level parallelism.

4.2 Data-flow control

In contrast to the Von-Neumann computer, the NEDIPS architecture is designed to execute the processing as soon as the necessary data is available. In other words, processing is controlled by the flow of data, not by the flow of control.

Fig. 7 shows the models of data-flow operation.

In this model, the operation modules are connected in a ring. In each module, a program is stored as a 'template'. A template is used in the data-flow computer in place of the program used in a conventional Von-Neumann computer.

The template in the data-flow computer specifies how data read by the processing module will be processed and where the results will be sent. No sequence of processing is specified. Further, in Von-Neumann computers, data is identified by its memory location. In non-Von-Neumann computers, however, data is identified only by the data name attached to it, regardless of sequence and storage location. Therefore, processing can be executed even though data arrives at the operation module and processing starts whenever sufficient data is available.

In Fig. 7 the reader module fetches data from memory and attached ID and destination tags in accordance with the reader template. Then, data is placed on the data bus. Each module continuously checks the data flowing on the bus, whenever a data item destined for that module is found, the module captures the data. Then, the module checks the tag related to the template to determine whether processing can proceed. If data is incomplete, it is temporarily stored in the queue area and processing starts on whatever data has accumulated. Processed data is again supplied with ID and destination tags in accordance with the template. When processing has been completed on a data item, it is given new tags in accordance with the template of the final module and again placed on the data bus. When data is returned to the writer module, its tag is removed and it is stored at the specified memory address.

4.3 NEDIPS system configuration

NEDIPS consists of a data-flow processor, memory unit and software as shown in Fig. 8. By virtue of the data-flow architecture, the Von-Neumann bottle-neck has been solved and the efficiency of the operation module has been greatly increased.

Memory unit includes a mass storage system, a host computer interface, and functions as an I/O buffer memory for the processor unit.

The hardware that makes up NEDIPS is connected to a host computer by a control channel and a data channel, and processing is executed under the control of the host computer.

The NEDIPS data-flow processor is connected to memory unit by means of a high-speed data bus and several control signal lines.

4.4 Data-flow processor configuration

The NEDIPS data-flow processor consists of an addressing unit, an arithmetic unit and a control unit as shown in Fig. 9. The arithmetic unit and the addressing unit are connected in a ring of operation modules by a data bus. The data flows between operation modules. Each operation module stores programs (templetes) and takes necessary data into the module. The module processes whichever data is ready first. The results are sent through the data bus.

The addressing unit is composed of operation modules which perform fixed-point operations, memory interface and so forth. The arithmetic unit consists of floating-point operation modules. The addressing unit and the arithmetic unit are connected by a ring interface. The data input from memory to the addressing unit is sent through the ring interface to the arithmetic unit and the ring of operation modules.

While the data is passing through the operation modules, complex formula are executed. Notice that no additional memory I/Os are required during a specific operation. Therefore, the speed of Data-flow Processor is little affected by the speed of memory Input/Output. Furthermore, because each of the operation modules operates efficiently in parallel, high-speed processing can be expected.

4.5 NEDIPS system application field

At present NEDIPS is in actual operation for Remote Sensing Image processing. However, NEDIPS can be applied in every field of scientific processing where high-speed computation is required. Language supported in NEDIPS is a data-flow type language called templete assembler language. Furthermore, full FORTRAN support is planned, then it is expected to be used extensively as an alternatives of entry level supercomputer.

5. CONCLUSION

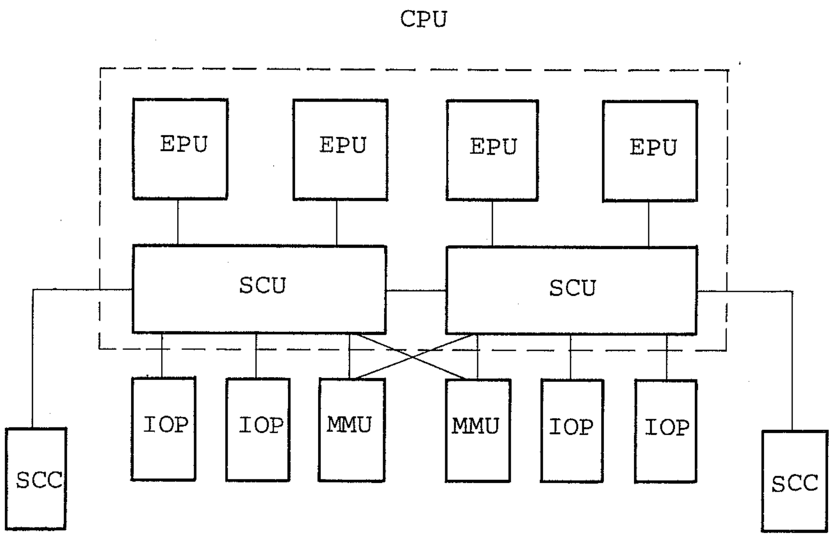
NEC's development approach for scientific computer, particularly those of multiprocessing (or MIDM machine) is introduced as three processors as examples. It seems SIMD architecture (or multiprocessor SIMD) with powerful scalar power and wide memory band width like SX system has still have advantages over true MIMD architecture which may require thousands of micro-processors considering the present state of arts. One candidate of next generation supercomputer may be one based on data-flow concepts like NEDIPS.

Reference

Dijkstra, E.W. 1968: "Cooperating sequential processes" in programming Languages. New York: Academic Press, Inc.

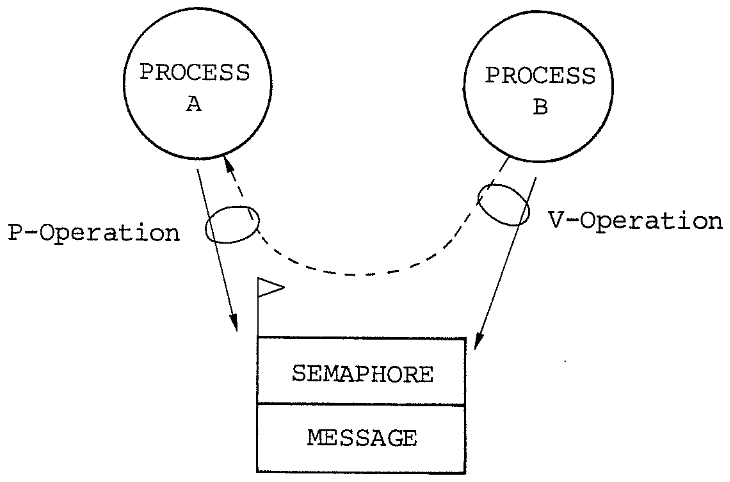
Temma, T. and Mizoguchi, M. and Hanaki, S. 1984: Template-controlled image processor TIP-1. NEC Res. & Develop., No. 73, pp. 33-41

Watanabe, T. 1984: Architecture of supercomputes - NEC Supercomputer SX system. NEC Res. & Develop., No. 73, pp. 1-6.



CPU:	CENTRAL PROCESSING UNIT
EPU:	EXECUTION PROCESSING UNIT (PROCESSOR)
SCU:	SYSTEM CONTROL UNIT
MMU:	MAIN MEMORY UNIT
IOP:	INPUT/OUTPUT PROCESSOR
SCC:	SYSTEM CONTROL CONSOLE

Fig. 1 ACOS Mainframe Central Complex Configuration



(1) Semaphore
(2) P-Operation
(3) V-Operation
(4) Wait Queue/Ready Queue

Fig. 2 Semaphore Operation

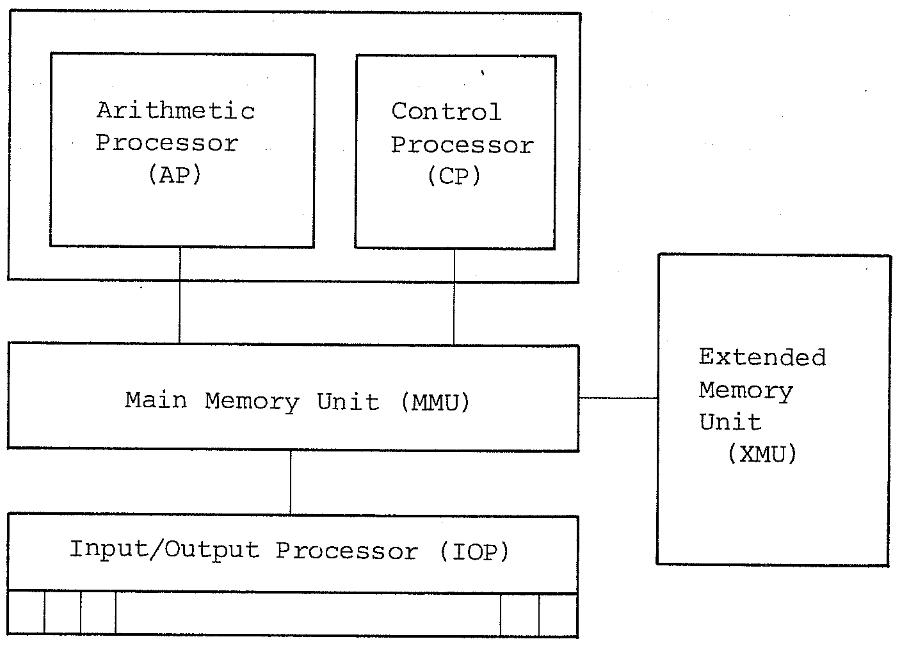
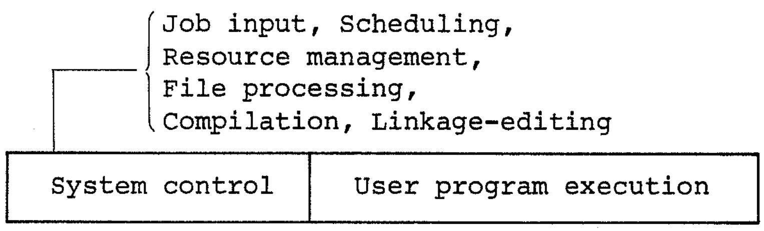
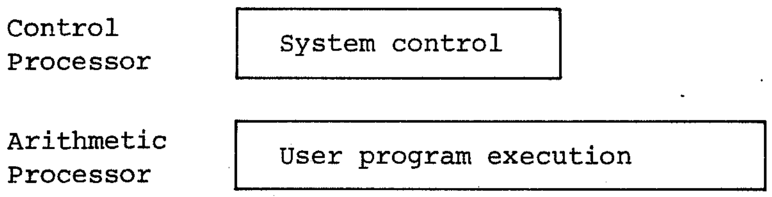

Arithmetic Processor (AP)
Control Processor (CP)
Main Memory Unit (MMU)
Extended Memory Unit (XMU)
Input/Output Processor (IOP)

Fig. 3 SX System Configuration



(a) Internal processing time of the central processing unit in a general-purpose computer.



(b) Internal processing time when functions are distributed.

Fig. 4 Internal Processing Time of SX System

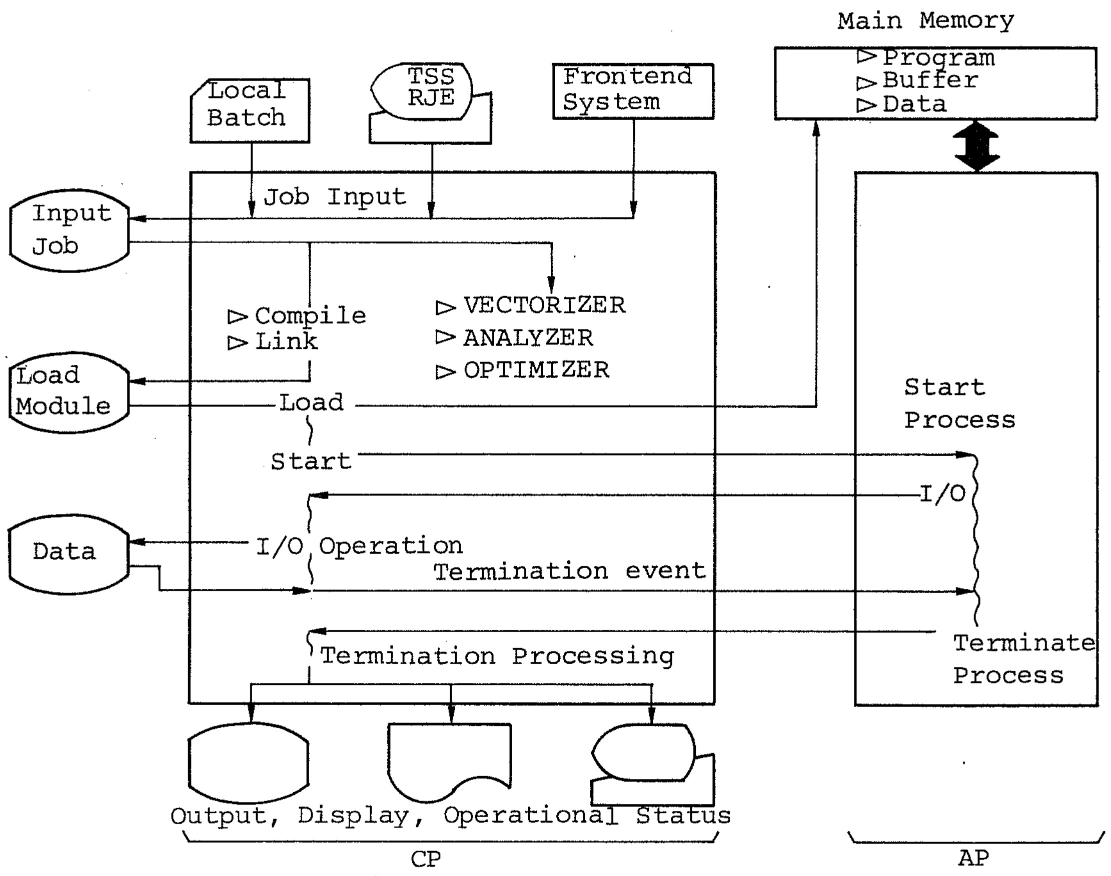


Fig. 5 Typical Job Flow and Function Distribution in CP and AP of SX System

- I/O Processing in CP, User Program in AP Simultaneously

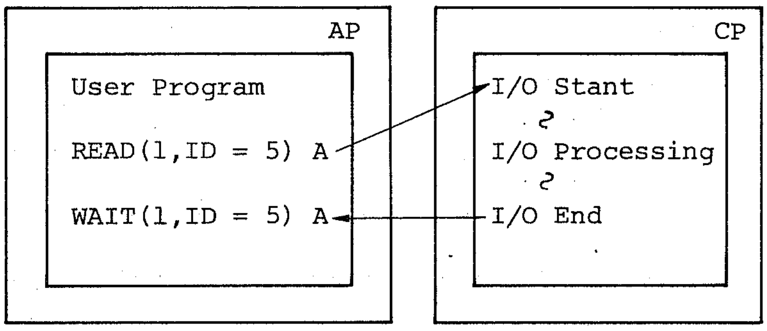


Fig. 6 Asynchronous I/O Operation in SX System

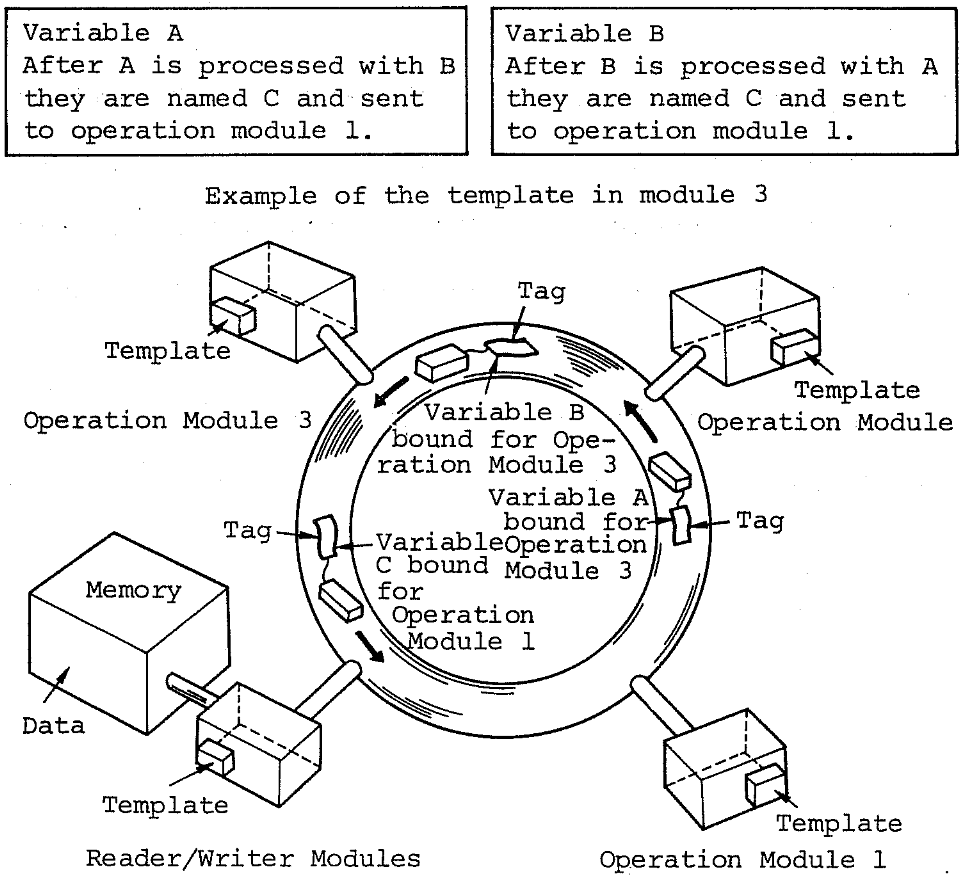

Variable A
After A is processed with B they are named C and sent to operation module 1.
Variable B
After B is processed with A they are named C and sent to operation module 1.
Example of the template in module 3
Tag
Template
Operation Module 3
Variable B bound for Operation Module 3
Template
Operation Module 2
Variable A bound for Operation Module 3
Tag
Tag
Variable C bound for Operation Module 1
Memory
Data
Template
Reader/Writer Modules
Template
Operation Module 1

Fig. 7 Data-flow Model

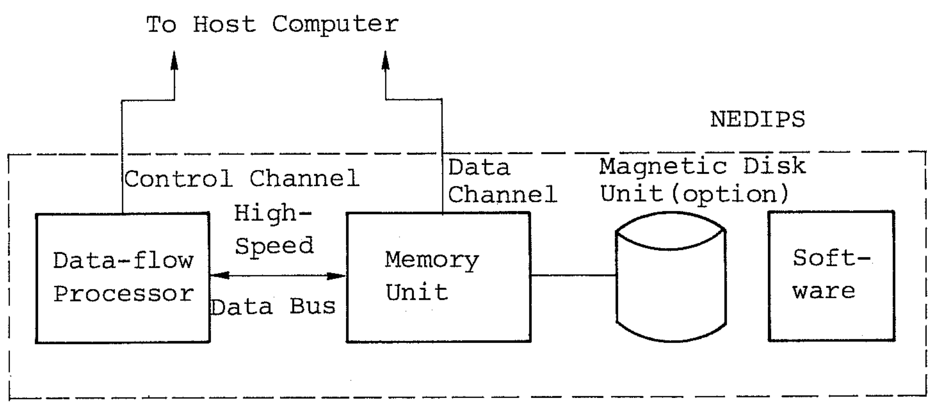


Fig. 8 NEDIPS System

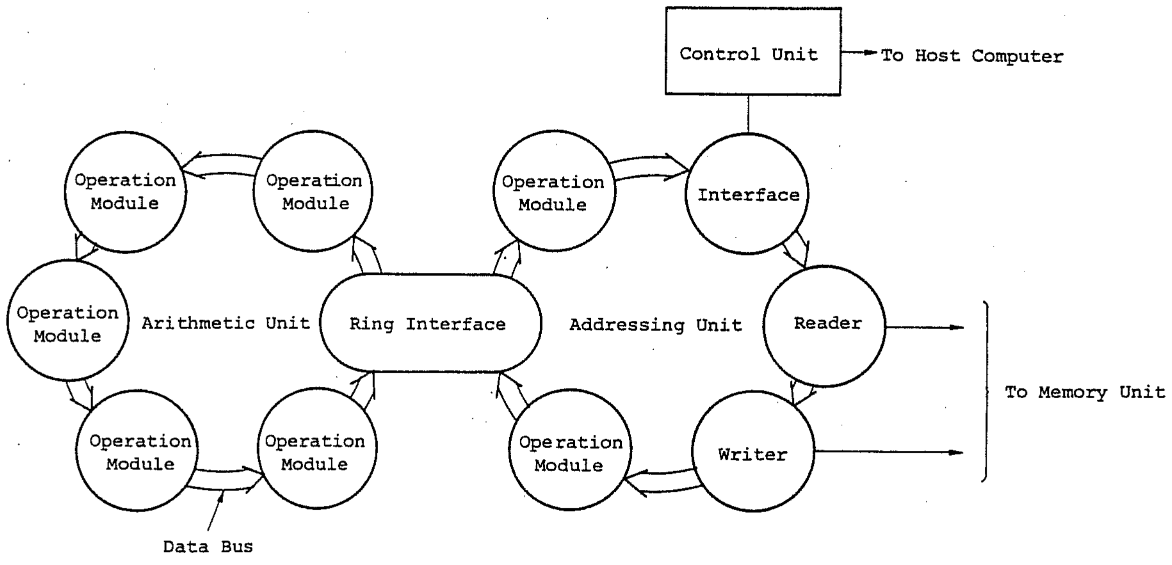


Fig. 9 Conceptual Illustration of NEDIPS Data-flow Processor