CMU-CS-72-112

C.mmp: THE CMU MULTIMINIPROCESSOR COMPUTER
Requirements and Overview of the Initial Design

C. G. Bell, W. Broadley, W. Wulf, A. Newell

and

C. Pierson
R. Reddy
S. Rege

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania
August 24, 1971

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C-0107) and is monitored by the Air Force Office of Scientific Research.

ABSTRACT

This document describes a proposed CMU multiprocessor system to be constructed around a set of PDP-11 computers connected through a cross-point switch to a large sharable primary memory. The present design constitutes a solution to a specific set of needs existing in our environment. The system has research consequences that reach well beyond the particular demands it was designed to satisfy. For although multiprocessors have been much talked about and advocated, there are remarkably few operational systems more complex than dual-processor systems, and even fewer documented scientific investigations into their performance and operating structure.

This document is limited to a presentation and analysis of the (hardware) system. It gives enough description of the usage requirements, software, and the research potentials and problems to make clear why we believe the effort to be a sound one. It does not attempt a systematic discussion of the field of multiprocessor research, nor of alternative systems that might be of interest, either to meet our computing demands or as research directions.

Section II discusses the requirements and research potential. Section III lists the design constraints adopted. Section IV lays out the PMS structure of the system. Section V describes the main specifications of the operating system. Section VI provides some details on a performance analysis.

I. INTRODUCTION

This document describes a proposed CMU multiprocessor system to be constructed around a set of PDP-11 computers connected through a crosspoint switch to a large sharable primary memory. The system is to operate as a third node in the existing Computer Science computer system, along with two existing PDP-10's.

The present design constitutes a solution to a specific set of needs existing in our environment. It replaces a prior planned solution, which consisted of a collection of individual stand-alone PDP-11 systems and PDP-11 I/O processor systems situated in front of the PDP-10's. It is not the only system that satisfies our requirements, as the existence of the prior plan indicates. However, it provides a highly effective solution and does so within the cost framework with which we have been working.

The system has research consequences that reach well beyond the particular demands it was designed to satisfy. For although multiprocessors have been much talked about and advocated, there are remarkably few operational systems more complex than dual-processor systems, and even fewer documented scientific investigations into their performance and operating structure. Thus, the multiprocessor system offers substantial opportunities for significant research that is consonant with our research interests and capabilities.

This document is limited to a presentation and analysis of the (hardware) system. It gives enough description of the usage requirements, software, and the research potentials and problems to make clear why we believe the effort to be a sound one. It does not attempt a systematic discussion of the field of multiprocessor research, nor of alternative systems that might be of

interest, either to meet our computing demands or as research directions.

Section II discusses the requirements and research potential. Section III lists the design constraints adopted. Section IV lays out the PMS structure of the system, giving the various components which have to be fabricated, their gross specification and the design decisions remaining. Section V describes the main specifications of the operating system. Section VI provides some details on a performance analysis.

II. BACKGROUND ON REQUIREMENTS AND RESEARCH POTENTIAL

The CMU multiprocessor project is directly responsive to two requirements:

1. the need for particular computational facilities
2. a research interest in computer structures.

The design itself may be viewed as attempting to satisfy the computational needs with a system that is conservative enough to ensure successful construction in an immediate time frame. Within this constraint, the system appears to be a research vehicle of considerable potential, both directly in terms of research on multiprocessor systems and in its ability to support a wide range of important investigations in computer design and systems programming. Beyond this, the system will produce a significant amount of information processing capability, thus satisfying an important side condition that the funds we have available to us for computing facility provide enough general power for the continued growth of the computer science community at CMU.

The co-existence of usage and direct research interests raises the (not unknown) spectre of conflicts between these interests in terms of stability and availability. Consistent with the view that this system is to provide computing facility, such conflicts, should they arise, will be resolved in favor of the users of the system.

All of our interests in the system are computer research interests. The users of the system are engaged in varieties of computer science research not significantly different from the direct research into multiprocessor hardware and software. (Indeed, they are often the same people, wearing different hats).

The existence of the system will generate new research not now being done in the environment, some of which we can quite clearly see and only part of which will be direct research on the system. When these additional efforts come into existence they will have as great a claim as present projects to the use of the system and will make as great contributions to justifying the system as satisfying their computation requirements. The temptation is to lump all these research efforts, direct and indirect, pre-existing and potential, into a single collection for the purposes of justifying the design. Instead we will preserve some fidelity to history, describing the requirements in terms of (1) usage by those research efforts who had already laid a claim for PDP-11-style resources; (2) potential research efforts and (3) research into multiprocessor systems.

USAGE REQUIREMENTS

The pre-existing research plans that already involved using small computers (PDP-11's in fact), and which define the core requirements for the present multiprocessor system, cover almost the full range of computer science research at CMU: artificial intelligence, systems programming, and computer structures. We take up each in turn.

Artificial Intelligence

The work in artificial intelligence relevant to C.mmp is in speech and vision; the development of a speech-understanding system being a major research venture at CMU, research on vision being a minor one. The requirements are of two kinds. The first, common to speech and vision, is that special high data rate, real time interfaces are required to acquire speech and vision data from

the external environment. For reasons of total system reliability these devices should be acquired by an input-output processor (Pio) and then transferred to Mp under standard internal conditions. A major cause of instability at a PDP-10 research installation is the specialized hardware added by that installation.* Use of a Pio also permits some pre-processing of the incoming high rate data, thus lightening the load on the main system (the PDP-10's) which is tuned to a slower interactive rate.

The second requirement, and by far the more stringent one, is to obtain real time processing for the speech-understanding system. The class of schemes under active pursuit involve multiprocessing with several PDP-11's plus a PDP-10, organized to work on all the levels of speech representation in parallel and with continuous intercommunication (i.e., the acoustic, parametric, phonemic, lexical, syntactic and semantic levels). The extent and shape of the parallel computation and intercommunication is a matter for intensive investigation, but some such scheme seems a fruitful approach to achieve real time speech processing. The total requirements of the system involve very large amounts of Mp, shared between several processors, backed up by extensive secondary storage. In short it is a computational problem that puts demands on the system along most dimensions.

Systems Programming and Operating Systems

At CMU we are intensively involved in research on operating systems and on understanding how software systems are to be constructed. Three faculty members independently (Habermann, Parnas and Wulf) and many graduate students are interested in these jroblems. Research in these areas has a strong

*This is extremely difficult to prove, of course. In our own system the addition of any hardware causes some instability for a period of up to several weeks.

empirical and experimental component, requiring the design and construction of many systems. We have discarded generally the notion of using virtual or simulated machines for reasons of performance, credibility and realistic users. Given the unavailability of the PDP-10 system, which operates in a user oriented time-sharing mode around the clock, a uniform solution has been to move to small stand-alone computers as the object machines for such experimental systems work.

The primary requirement of these systems is isolation, so they can be used in a completely idiosyncratic way and be restructured in terms of software from the ground up. On the other hand, the exact configurations needed varies from time to time. More important, work on the various experimental systems does not go on continuously 24 hours a day, but exhibits a variable pattern of usage, depending on how many people are working on the system and how intensively. These systems also require in an essential way access by multiple users and varying amounts of secondary memory.

Computer Structures

There is currently intensive activity at CMU in working with register transfer modules, using a system of modules (RTM's) developed here and at DEC (Bell, Grason et al, 1970) and now in production as the PDP-16. A dedicated facility is needed for the design testing of experimental systems constructed of these modules. A small computer is appropriate, not only to avoid building specialized test equipment, but to permit the design testing to become fully automatic. When operating in test mode the computer must be directly coupled to the system being tested and isolated from all other systems.

A similar need for a dedicated system arises from an interest in fault tolerant systems. Here, it is necessary that the various processors can have access to one another so that checking can take place. For example, a console of a faulty computer would be attached to a checking computer so that all tests could be carried out under control of the checking computer.

General Requirements

All of these research efforts require the same type of general purpose service as is now provided by our PDP-10 system: files, editing, higher languages, interactive access, etc. That, under our original plans, some of the stand-alone systems might have had to give up some of these facilities only expresses the price that appeared necessary to obtain an isolatable system that could be restructured from the hardware up. Access to these facilities simultaneously with the capability for isolation on demand is certainly to be preferred.

The reasons for requiring general facilities for the speech and vision research appear somewhat different, since these programs are already large integrated software systems with an essential component running on the PDP-10.

POTENTIAL RESEARCH EFFORTS

We include in this category those researches that have not guided the design of the multiprocessor sytem, in the sense of providing constraints it had to satisfy, but which appear to capitalize on the system so strongly that they have a probability of occurring that ranges from high to certain.

Network Research

Our original plans in obtaining the PDP-10 system involved moving towards a network configuration of several PDP-10's. Our motives for so doing were in large part conditioned on our wanting to do research on networks, as an appropriate structure for a within-institution computer system (Bell, Habermann, McCredie, Rutledge, Wulf, 1969). Currently we have two PDP-10's, the second one just achieving full system status. The growth of an actual network has been slow, being affected by funding limitations and by other developments in the environment.

The multiprocessor affects the network plans in two ways. The multiprocessor itself provides a field within which one can set up experimental network configurations of assorted sizes and shapes. There are limits to what can be achieved by simulation and artificially constructed abstract networks, just as there are limits to what can be achieved by simulated and virtual operating systems. But they provide an essential tool. The proposed multiprocessor will permit systems with up to perhaps a dozen real nodes to operate in real time; and we expect a substantial amount of network research to evolve in this direction.

The multiprocessor system, taken as a single computer system, provides a third node in a network consisting of itself and the other two PDP-10's. This is a real network with real problems of exploitation. The multiprocessor will serve as a highly specialized node with unique functional capabilities. It will be in no way a minor node in the system in terms of processing power, though this power will be realized on relative short word lengths (16 b/w).

Design of Processors

The multiprocessor is designed around a set of identical Pc's, namely PDP-11's. However, as long as certain interface design philosophies are maintained, the design permits processors of quite different designs to operate in the system, and to obtain from the rest of the system support with respect to files, input/output, initialization, etc. Various systems constructed with RTM's can take advantage of this.

The recent design exercise on an ARPA list processing system led to a local design (C.ai, see Bell, Freeman, et al, 1971; Barbacci, et al, 1971; and McCracken Robertson, 1971) that stressed, among other things, the design of processors that went a long ways in specialization to a given list processing system (e.g., a LISP system, an L* system, etc.). The multiprocessor offers an opportunity to design and realize some experimental processors. Considerable investigation is required before one can be sure such a venture is worth the costs involved, but with the lowering cost of processors it appears on first analysis to be quite feasible. On the positive side is the principle, now stated several times, that only a small fraction of the necessary research on computer systems can be done via simulations and virtual machines. An actual unconventional processor (say for L*), which offered substantial processing advantages over programmed versions of the same system, would call forth the programming efforts to explore the system's real worth and determine its ultimate defects as a user system.

MULTIPROCESSOR RESEARCH

It is a fact that one cannot put together a multiprocessor system of any complexity today without becoming involved in research on multiprocessors.

Multiprocessor systems (other than dual processor structures and nPio-Pc structures) have not become current art. Some of the reasons for this state of affairs appear to be:

1. The absolutely high cost of processors and primary memories so that a complex multiprocessor system was simply beyond the computational realm of all but a few extraordinary users, no matter what the advantages.

2. The relatively high cost of processors in the total system, so that an additional processor did not increase the performance/cost ratio.

3. The unreliability and performance degradation of operating system software, which made going to more complex system structures a venture in futility.

4. The inability of technology to construct the central switches required for such structure, due in part to low component density, in part to high cost.

5. The loss of performance in multiprocessors due to memory access conflicts and switching delays.

6. The unknown problems of dividing tasks into subtasks that can be executed in some parallel, though possibly interactive way.

Thus, the expense was prohibitive, even for discovering what advantages of organization might overcome the obvious decrements of performance. And in any event, it seemed mostly to complicate matters in a world that was already

too complicated to function properly.

We appear to have entered into a technological domain when many of the difficulties listed above no longer stand so strongly:

1'. Providing we limit ourselves to multiprocessors of minicomputers, the total system costs of Pc and Mp are now within the price range of a reasonable facility. (E.g., the CMU system with 16 Pc, suitable Mp, Ms and T will still be around.5M$, yet execute approximately 3 ~ 5 million instructions/sec.).

2'. The Pc is now generally a smaller part of the total system cost. Hence, a small incremental cost that provides a substantial increase instruction/sec. will be cost effective.

3'. Software reliability is now somewhat improved, primarily because a large number of operating systems have been constructed. Something is understood about the overhead costs imposed by various forms of organization.

4'. Current medium and large scale integrated circuit technology enables the construction of switches that do not have the large losses of the older decentralized switches. Centralization of the switch permits fewer and shorter cables.

5'. Memory conflict is not high for the right balance of processors, memories and switching system. The trade-offs possible, of course, are characteristic of a given domain of technology. Appropriate balances appear to be attainable currently.

6'. There has been work on the problem of task parallelism, not only general studies, but highly specific studies, e.g., around ILLIAC IV and CDC STAR. Other work on modular programming (Krutar, 1971; Wulf, 1971) at CMU suggests how subtasks can be executed in a pipeline.

In short, the price of experimentation appears eminently reasonable, given that there are requirements that appear to be satisfied in a sufficiently direct and obvious way by a proposed multiprocessor structure (i.e., for us, the usage requirements listed at the beginning of the section).

However, the state indicated above does not settle many issues about multiprocessors, nor does it make the development of one routine. We list below the main areas of research interest in the multiprocessor system itself. All of these call for some attention, although the extent of our effort in each is unclear.

1. The multiprocessor design itself, i.e., its PMS structure. Few enough multiprocessors have been built, so each one represents an important point in design space.

2. The processor-memory switch design, especially with respect to reliability.

3. The configuration of computations on the multiprocessor. There are many processing structures and little is known about when they are appropriate and how to exploit them, especially when not treated in the abstract but in the context of an actual processing system:

Parallel processing: a task is broken into a number of subtasks and assigned to separate processors.

Pipeline processing: various independent stages of the task are executed in parallel (e.g., as in a co-routine structure).

Network processing: the computers operate quasi-independently with intercommunication (with various data rates and delay times).

Functional specialization: the processors have either special capabilities or access to special devices; the tasks must be shunted to processors as in a job shop.

Multiprogramming: a task is only executed by a single processor at a given time.

Independent processing: a configurational separation is achieved for varying amounts of time, such that interaction is not possible and thus doesn't have to be processed.

4. The decomposition of tasks for appropriate computation. Detailed analysis and restructuring of the algorithm appear to be required, as in the work with ILLIAC IV. The speech-understanding system is the one major example we know we will undertake. It has research interest from the multiprocessor viewpoint, as well as from the speech recognition viewpoint.

5. The operating system design and performance. While the basic operating system design must be conservative, since the multiprocessor will run as a computation facility, it will have substantial research interest. Variations and alternative operating systems will also be constructable.

6. The measurement and analysis of performance of the total system. One of our complaints about the current field is the high ratio of design studies to performance studies.

7. The achievement of reliable computation by organizational schemes at higher levels, such as redundant computation.

III. DESIGN CONSTRAINTS

The following constraints have been set up as a policy to guide our research and development. While we consider many of the constraints to be absolute at this time, we may on occasion have to exceed them. The user and research requirements of the previous section are essentially the objective function.

PDP-11 PRODUCTION COMPONENTS

We would like to use PDP-11 components as they are produced, without modification. These components include Mp (primary memory modules), Ks (slow device controllers for non-direct memory transfers, such as teletypes), Kf (fast device controllers for direct memory transfers, such as disks), and Pc (central processors, various models present and future). Modifications to the central parts of the processors or to the device controllers are no doubt the most serious, because there are so many of the devices, and a modification may preclude future device connections. Modification to the primary memory seems somewhat less serious. There is only one type of component, we will have many copies of it, the cost is relatively high for this portion of the machine (hence we can amortize any change over a larger number of units). In addition, total system performance is strongly affected by the Mp characteristics. Hence there may be a large payoff for changing it.

For the above reasons we have decided to modify the memory system in order to get the resulting increase in performance and lower cost. Accompanying these modifications we must modify each processor interface slightly

to get a faster memory bus. However, since there are other minor modifications to the processor for mapping and protection, we can incorporate the modifications into a single interface that is added on to the standard DEC processors. This is discussed in a later section.

We have not adopted the constraint that devices need to be addressed anonomously by all processors at any time. It will be acceptable to set switches that establish a processor to be in control of a particular device for a long time (e.g., perhaps seconds or minutes, but possibly even days). Requests for device data transfers are handled by the particular processor in control of the device. A particular device must be assignable to more than one processor in order to meet reliability requirements. In addition, many devices must be assignable to a single processor to permit specialization. This constraint on device accessibility is established in the clear knowledge that in general the processors in the system will be identical (i.e., PDP-11's Pc's), hence anonymous from a user-program view, and capable of being assigned device duties according to the needs of the system.

Although we believe it would be desirable to keep the device addresses identical to those of DEC, in order to run software without reassembly, we must modify these numbers. The change is required to avoid putting path-finding memories (content addressable) in the switches for each of the devices.

LOGIC TECHNOLOGY

Initially, we have decided to use only standard TTL integrated circuitry, which can be obtained via standard procurement channels. The TTL constraint is imposed both because our facilities to fabricate multiple layered boards are limited and because the interface to the PDP-11 is via TTL.

Eventually we would like to explore the use of LSI technology in the design of the switch sections. We have currently rejected the use of this technology because it requires s close coupling with an integrated circuit facility, hence putting the scheduling, outcome, etc. beyond our control. We would like to engage in such a design, when the present system is more operational.

EXTENDABILITY

We are constraining the design to be relatively fixed, but large, so that it works with a certain maximum configuration size. Any subsequent extensions would require a redesign along the same lines, but with different parameters. Another approach would be to have a design which would let the configuration grow in a more indefinite fashion. However, we are not willing to undertake such a design, because of what appear to be high time penalties for the smaller systems and the uncertainty of being able to build a system of indefinite capabilities.*

CONFIGURABILITY AND INDEPENDENCE

In order that individual research projects not be coercively constrained by running within a large system, the design permits partitioning into smaller, independent computers in a reliable fashion. In this mode the subset of equipment constitutes an independent computer, with appropriate secondary memory and input-output terminals.

---

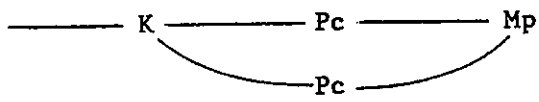
*Possibly such a system might reach a point of diminishing returns at a smaller size than with the fixed, maximum size approach.

RELIABILITY

We are not designing the circuitry of the system to meet extremely high reliability constraints, mainly to avoid several lengthy iterations of design and testing. Nevertheless, we expect to be much concerned with the ultimate reliability of the system, and will attempt to achieve it by various techniques at higher levels. From an operational point of view, we believe the following constraints are the appropriate ones:

1. There must be at least n+1 components of the particular types necessary to constitute a minimum usable of n components. (Components for special functions are not considered necessary.)

2. Considering each Mp|Pc|K to have roughly the same reliability, there must be multiple paths for all component information flow paths, e.g.,



3. All busses shall function properly when a component connected to it has power removed.

4. A component can be connected or disconnected when system power is on.

SCHEDULE

The development time constraint (schedule) determines to a large degree the methods we use in the design because we feel it is important to have an

operational computer in a short time. The short schedule is based on the need of our users for the facility and our desire to provide a research facility. Contributions of this system to such efforts as the ARPA list processing machine or to improved versions of multiminiprocessor systems all require rapid completion. Much of the contribution to computer science of constructing a multiprocessor system comes not from proposing a design and not even from putting together the physical system, but from exploring the effect of its structure on computational tasks and constructing operating systems that exploit its capacity. To ever get to the latter requires setting a short schedule for producing the former. Thus, time is of essence for the construction phases of the project.

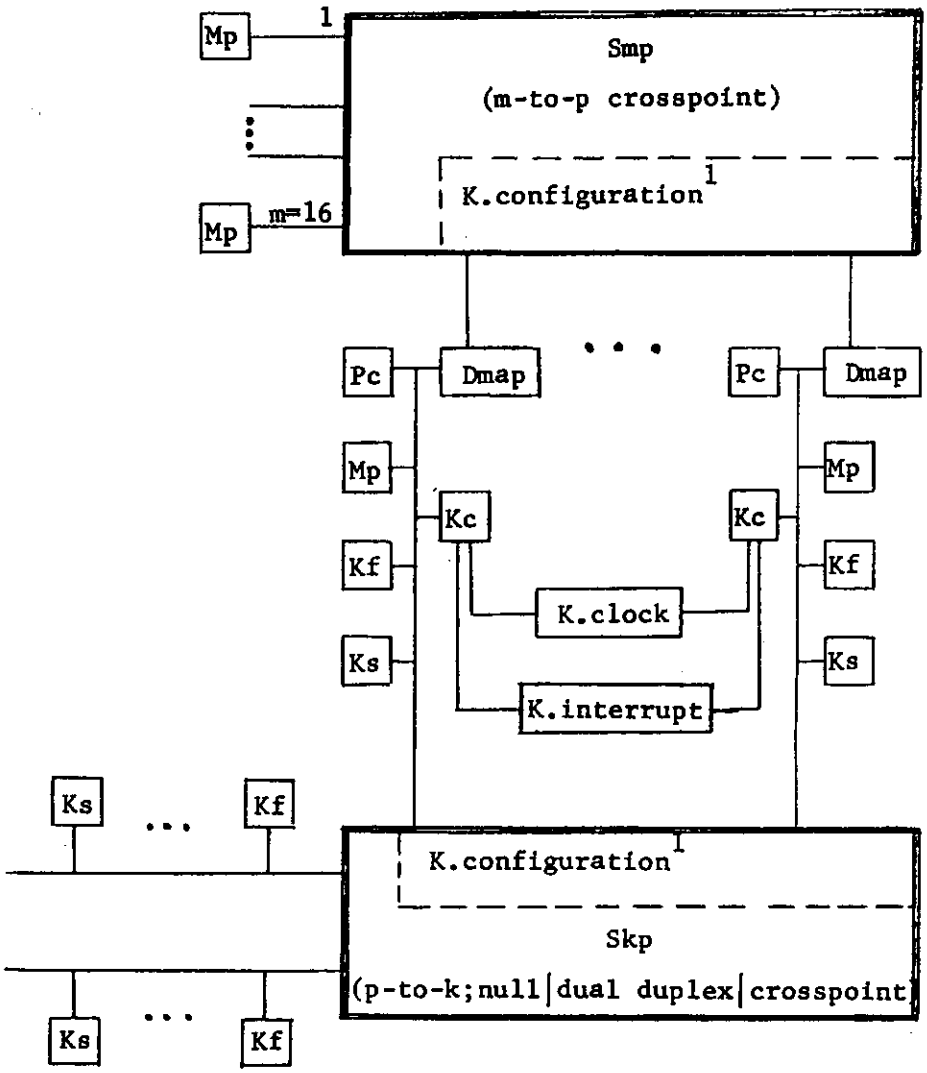
IV. MULTIPROCESSOR PMS STRUCTURE, IMPLEMENTATION AND ISP

In this section the design will be described briefly and directly, without explicitly relating each part to the design constraints. We assume the reader is familiar with the DEC PDP-11 structure. The configuration is basically a multiprocessor system of conventional type that has been proposed and has existed in smaller-sizes (usually two processors) or different form, e.g., the Burroughs B5000 and D825 computers, IBM's 360/65 and various proposed aerospace computers (see Bell and Newell, 1971a). The structure of the system is given in Figure 1.

There are two switches, Smp and Skp. Smp allows the processor to communicate with primary memories. Skp allows the processor to communicate with the various controllers (K), which in turn manage the secondary memories (Ms and T) and terminals (T), respectively. The switches are under both computer and manual control.

Each processor system is actually a complete computer with its own local primary memory and controllers for secondary memories and terminals (e.g., Teletypes). Each processor has a Data operations component, Dmap, for translating addresses at the processor into physical memory addresses. The local memory serves both to reduce the bandwidth requirements to the central memory, since PDP-11 makes many references into a stack area of memory, and to allow completely independent operation and off-line maintenance.

A central clock, K.clock, allows precise time to be measured. This is necessary to allow the software to log message transfers and to perform measurement. Each processor can interrupt the others for message transmission. A central time base is broadcast to all processors so that each can compute independent elapsed times.



where: Pc/central processor; Mp/primary memory; T/terminals;
Ks/slow device control (e.g., for Teletype);
Kf/fast device control (e.g., for disk);
Kc/control for clock, timer, interprocessor communication

[1]Both switches have static configuration control by manual and program control

Fig. 1. Proposed CMU multiminiprocessor computer/C.mmp.

Smp - THE SWITCH BETWEEN PROCESSORS AND PRIMARY MEMORIES

This switch handles information transfers between primary memory and the processors (requestors). Transfers from the fast controls, Kf, for primary memory via the processor, also go through Smp. The switch has ports (i.e., connections) for m busses for primary memories and p busses for processors. Up to min(m,p) simultaneous conversations are possible via the cross-point arrangement. There is no memory in Smp for data port buffering, and a single processor has only one memory request pending at a time. A more complete discussion of cross-point switch structures is given in Bell and Newell (1971a). Unlike most cross-point switches, this one is located centrally (as opposed to being distributed in the memory as is usual, e.g., in the PDP-10, 360/65 $\sim$ 67, or 1108). While this requires a larger initial configuration and implies non-modularity, the cost of an average system should be less and we are interested in a rather large configuration. A large cost component of a switch system (together with the associated mechanical and circuitry problems) is the cables. This structure requires only p+m cables, as opposed to p×m cables in the case of a distributed switch. This structure also has less cable delays than the associated distributed switch.

Another aspect of Smp is the control required for providing different configurations. Smp can be set under programmed control (i.e., a Unibus carried configuration parameters). In addition, these parameters can also be set via manual switches on an override basis. The control of Smp can be by any of the processors, but one processor is assigned the control. In this respect, switch control is like any other T or Ms control:

Smp is not Unibus compatible, although the signals are essentially the same. We have decided against compatibility in order to improve the speed through the switch and to achieve better signal-to-noise characteristics. Since we are modifying the processor, the links: P——Smp and Smp——Mp can be changed rather easily.

Switch Details

The CMUnibus enters the cross-point and the lines on this bus serve the following functions:

1. From P: request that a selection be made (terminates in Smp).
2. From P: request a particular memory module, Mp (terminates in Smp).
3. M-P dialogue: control the flow of data between M and P on an interlocked basis (uni-directional, switched through Smp).
4. From P: select the word within a module, Mp (uni-directional, switched through Smp).
5. Between P and M: pass information between the selected P and Mp (bi-directional, switched through Smp).

These lines fall into two categories: those that terminate in Smp and control the selection process to a selected m-bus, and those that are switched to a selected m-bus. For those of the second category, which we will consider to be data, the basic structure of the switching is shown in Figure 2. Here, we show the components involved with the switching of a bi-directional bus. If data is only transmitted in one direction (as in the case of addresses that arrive on p-busses and must be transmitted to the appropriate m-bus) only half of the circuits are installed in the printed circuit cards (cases

3 and 4 above). There are p+m receiving circuits per bit that connect to the entering cables from the processors and memories; also there are p+m transmitting circuits at these cable entries which transmit the switched results to the processors and memories. Since this is a bi-directional bus, the Smp must know the direction of data flow and set the switches accordingly. There are pXm multiplex circuits per cross-point bit that operate as follows.

An incoming bit from P to M goes through the p-bus receiver, the appropriate outgoing m-bus line is known (by the m-bus control) and this control selects the appropriate multiplexor position. The bit passes through the multiplexor to the transmitter where it is sent on to the m-bus.

A bit from M to P arrives on an m-bus, goes through the m-bus receiver, and the m-bus control selects the multiplexor corresponding to the p-bus which is to receive the bit. The multiplexor is selected and the bit is transmitted on to the processor via a p-bus. This type of switching is used for case 3, 4 (uni-directional) and 5 (bi-directional).

The selection of a particular m-bus is accomplished by a control associated with each m-bus that observes all incoming requests from all processors (cases 1 and 2 above) and decides to close the data port switches and establish direction for the particular m-bus it controls. The m-bus control unit (Figure 3) addresses the particular multiplexors for transmitting and receiving data shown in Figure 2. The inputs to this control are on the basis of the particular processor requesting the control, m-bus, and the inputs from manual switches that change the configuration (shown at the right-hand side of the control box). These inputs allow the control to rename

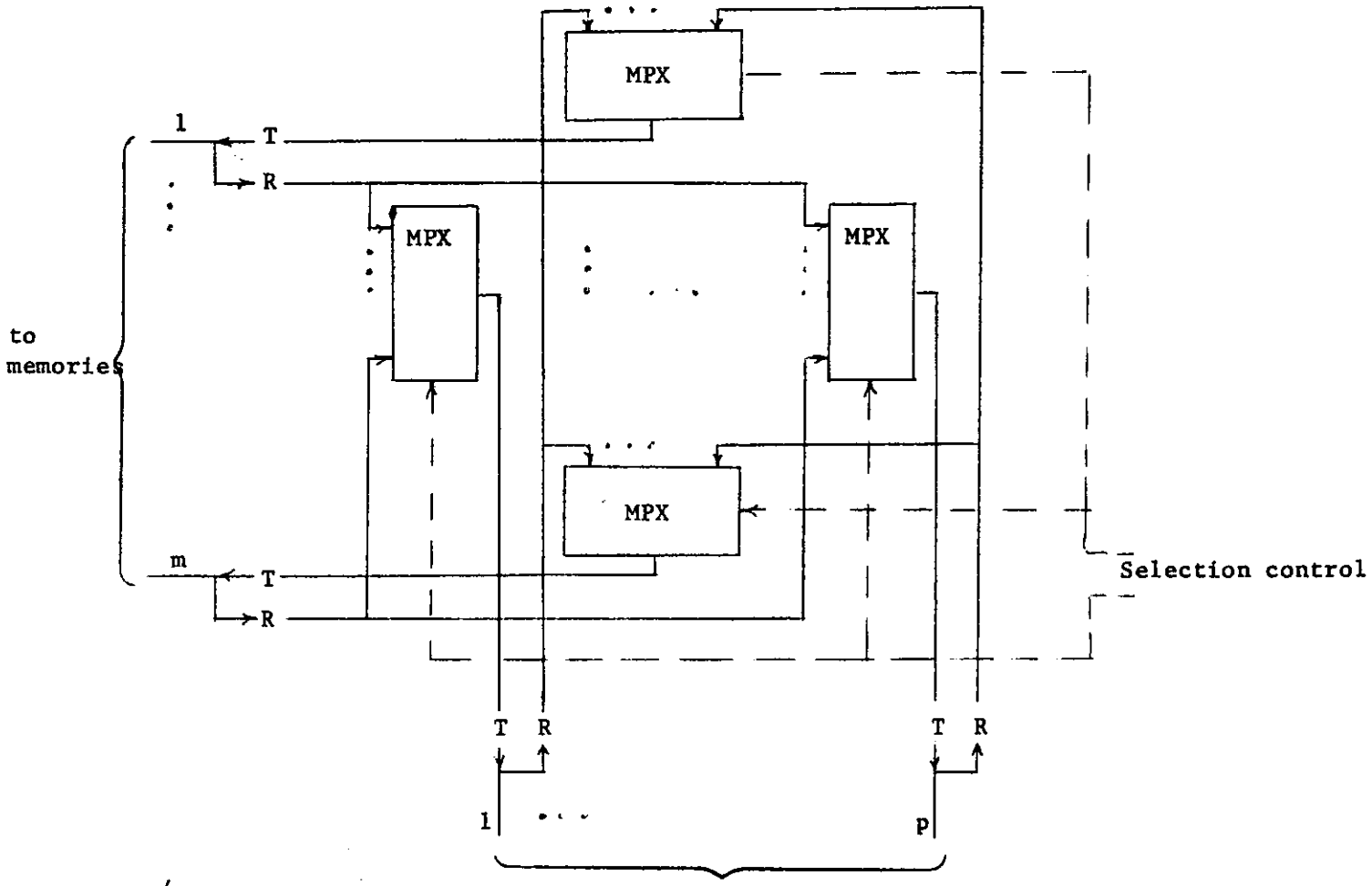


Fig. 2. Bi-directional, cross-point switch of 1 bit for Smp.

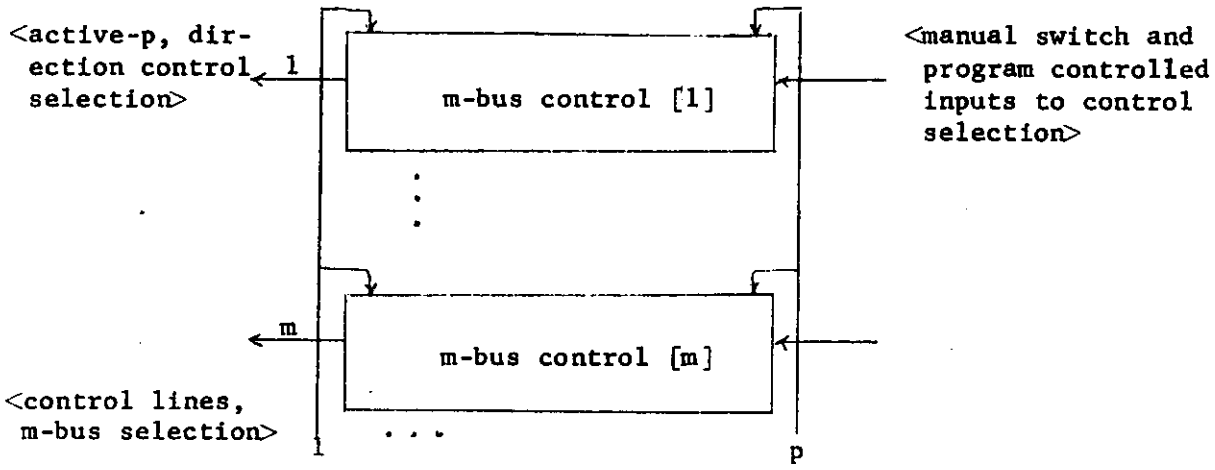


Fig. 3. Control part of Smp.

particular m-bus lines to correspond to different physical addresses and to ignore requests from specific processors.

Mp IMPLEMENTATION

A number of memory manufacturers were approached for implementation of Mp. At the final selections, not only was a cost/performance ratio important but also the vendor's ability and willingness to undertake construction of the switch. Two manufacturers met these requirements. One was proposing a fast cory memory system (650/250 ns cycle/access) at 1.5¢/bit; the other a 150/180 ns bipolar memory at 2.5¢/bit. Both memories were modular at the 8k (word) level. The core memory system was chosen.

Because of the large number of modules, the cycle time was less important than the access time. One of the major processors on this system, the PDP-11/20, only needs a word approximately every 1.5 μsec. and, in addition, never does a WRITE into memory, but always does a READ-MODIFY-WRITE (RMW). A RMW cycle in semiconductor memory takes two cycles whereas in core memory it takes only 1.2 cycle times. This, plus the better cost/performance ratio and the difference in access time of only 100 ηsec., led us to choose the core system.

Alternatives for the Switching Structure

Many alternatives were considered for carrying out the switching between memory, processors and the other components (primary memories, slow controllers and fast controllers). Only one competitive scheme we know of seems worth mentioning -- a switch design that was based on this design, undertaken

at the University of Newcastle in May, 1971. (See two memos by Bell, Lauer and Randall, 1971.) Most switches, while giving more flexibility for configuration, and hence reliability (e.g., like the S.trunk, Bell and Newell, 1971a), require more equipment and increase switching time.

Reliability

The reliability of the switch can be significantly increased by providing some form of a duplicated switch. We are not looking into a switch of this type currently, because we feel the number of parts in the switch is sufficiently small. The reliability will be comparable to that of a single 11/20 processor. The design of the switch is such that failures will appear either as a failure in a memory or a processor port. Since the system is designed to tolerate either of these types of failures, a switch failure can also be tolerated.

Skp - THE SWITCH BETWEEN PROCESSORS AND THE CONTROLS FOR SECONDARY MEMORIES AND TERMINALS

Skp allows one or more of k Unibusses which have several slow or fast controllers (Ks or Kf) to be connected to one of p central processors. The k Unibusses with the controllers are connected to the p processor Unibusses on a fairly long term basis (i.e., minutes to days), since a processor will manage a control completely as a resource, independent of the location of the actual user process requesting the physical resource. This management consists of initiating data transmission, processing interrupts, transmitting data for it, examining its status, and finally turning it off. One of the main reasons for only allowing a long term, but switchable, connection between the K Unibusses and the processor on a one at a time basis is to

avoid the problem of having to decide dynamically which of the p processors should manage a particular device. Information is not available in each device that could be used for this purpose. We have decided to provide only static switching because about the same amount of processing is involved, independent of which of the processors does it. Furthermore, this processing is usually small,* hence it does not generally matter which processor handles it.

Like Smp, Skp has a connection (via Unibus) which is used to control the configuration of the switch, i.e., determine which of the k busses is connected to the processors. There is both program (via the Unibus) and manual control of the configuration. Note that since the control of the configuration is via one of the controlled Unibusses, the switch must first be set to a known position to allow subsequent control to be made. Connecting the configuration control bus to a specific computer would violate the reliability goals.

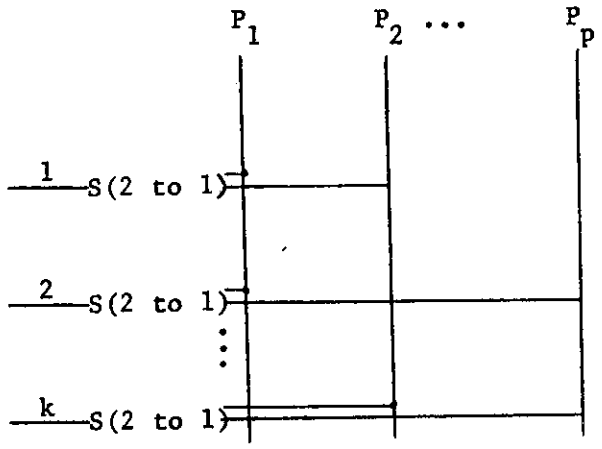
Skp Structural Alternatives

At the present time we do not know the exact structure of Skp. In the trivial case this could be a null switch with various devices just connected to the p busses. Unfortunately, an arrangement of this type will force a processor and all associated devices to be taken out of the system when a single device is connected or removed, since the processor will have to be stopped.

---

*The interrupt service to handle a single Teletype character on the PDP-10 is on the order of 500 μs. Assuming 10 char/sec would require 1/2% processor capacity per line. The device data transmission is handled roughly in the same way as PDP-10, except that the better interrupt facility of PDP-11 should further reduce service times.

A more desirable structure is k two-position switches, S(2 to 1) which allow either of two processors to connect to a single Unibus. A structure of this type might look like:



This structure allows us to expand k after the initial configuration, yet provides multiple control-data Unibusses for a set of controllers connected to one of the k busses. This gives a particularly simple structure if $k = p/2$, by having all Unibusses accessible to at least two processors, with only one switch per processor and only devices on 1/2 the processors.

The most desirable alternative for the switch, from a performance viewpoint, is the cross-point. The only limitation is its capability for future growth, given that a central one of the type proposed in Smp is used. The cross-point gives us the capability of allowing any of the k busses to be connected to any number of the p processors. S(cross-point) for Skp has the further advantage of allowing us to construct configurations which have either all devices on a single processor, or a small number of devices per processor. In this way, for certain specialized tasks, the devices can be handled with a minimum of interprocessor communication.

Skp is not critical in the early configurations, and we can start with mostly null switching with several S(2 to 1) and eventually construct a full cross-point. Doing this, we would place duplicate equipment on two separate processors. Non-duplicated, but reasonably critical, components would be placed on S(2 to 1)'s. Specialized equipment could only be placed on P's which had no S(2 to 1) or other critical devices.

PROCESSOR AND Dmap - PROTECTION, RELOCATION, AND INTERRUPTS

Most of the PDP-11 processor models can be used with the system, with minor modifications to the bus interface logic. Higher performance processors than those currently available might require extensive modification, if they used a second bus for faster solid-state memories.

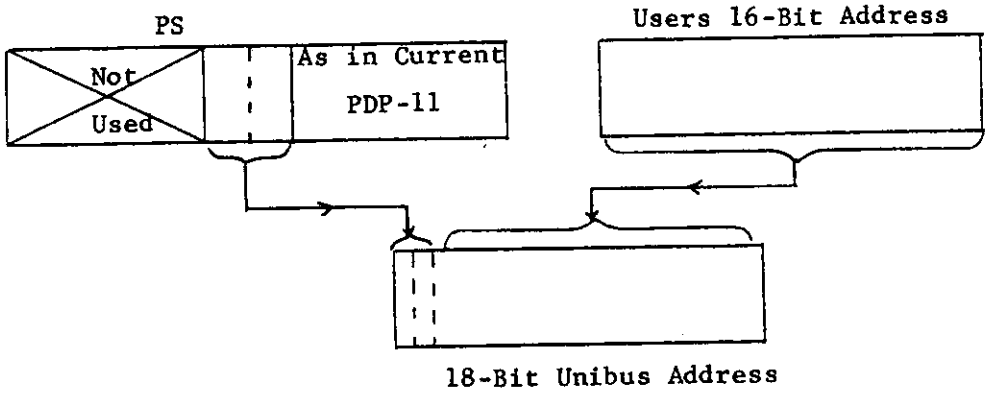
The Dmap is a Data Operations component which takes the addresses generated in the processor and converts them to addresses to use on the Memory and Unibusses emanating from the Dmap. There are four sets of eight registers in Dmap, enabling each of eight 8,192 byte blocks to be relocated in the large physical memory. The size of the physical Mp is 2^{20} words or 2^{21} bytes.* Two bits in the processor, together with the address type are used to specify which of the mapping registers is to be used.

The logical structure, as seen by the systems programmer, of the address map is described below, together with its implications for the user and the monitor. For the simple user, the conventional PDP-11 addressing structure is retained - except that he does not have access to the "i/o page", and hence the full 16-bit address space refers to primary memory.

---

*Provision has been made to expand the physical address space to 2^{24} (16×10^6) words at some subsequent time; however, this expansion is not currently planned.

A PDP-11 program can only generate a 16-bit address, but the Unibus has 18-bit addressing capability. In the proposed scheme the additional two bits of address will be obtained from two unused positions of the program status (PS) register. (Note, this register is inaccessible to user programs.)

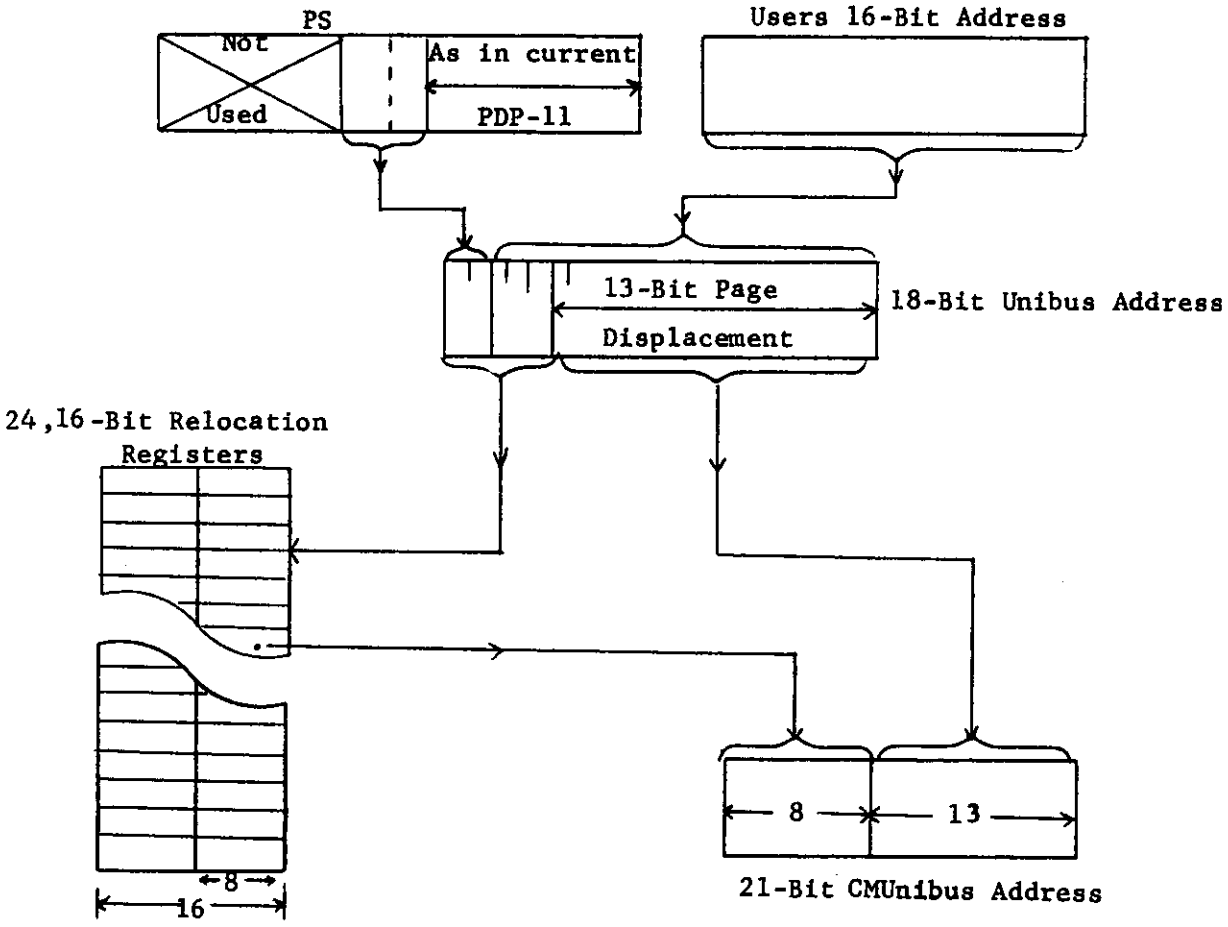


In the sequel we refer to ØØ-, Ø1-, 1Ø-, and 11- mode addressing to refer to the cases arising from the four possible bit configurations obtained from PS. These cases are:

ØØ-mode, Ø1-mode, 1Ø-mode — These addresses are always mapped, and always refer to the shared, large, primary memory.

11-mode — These addresses are (with two exceptions noted below) are mapped as above. The exceptions are that 8 kw of this space are not mapped and refer to the private Unibus of each processor.

For those references that are mapped, the mapping itself consists of using the top five bits of the 18-bit address to select one of 30 relocation registers, and replacing these by the contents of the 8 low order bits of that register:

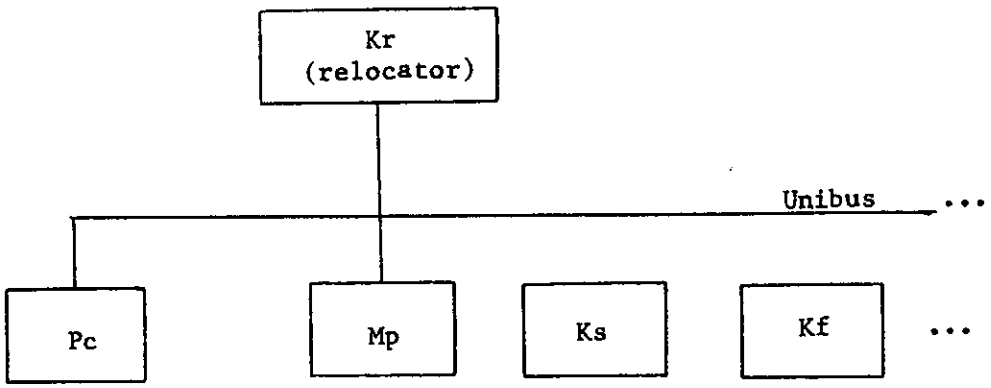


The leftmost five bits of the 18-bit Unibus address may be thought of as selecting one of 30 relocation registers, as described above. A better description, however, is that the two bits of the PS select one of four banks of relocation registers and the leftmost three bits of the users (16-bit) address selects one of the eight registers in this bank. This latter description is more appropriate, since

a process is effectively bound to a particular bank of relocation registers by its PS word, while it may (by appropriate monitor calls) alter the contents of the relocation registers within that bank and thus alter its "instantaneous virtual memory"--that is, the set of directly addressable pages.

Although not implemented in exactly this way, the relocator may be thought of as a controller on the Unibus as shown below:

21-Bit CMUnibus to large Mp

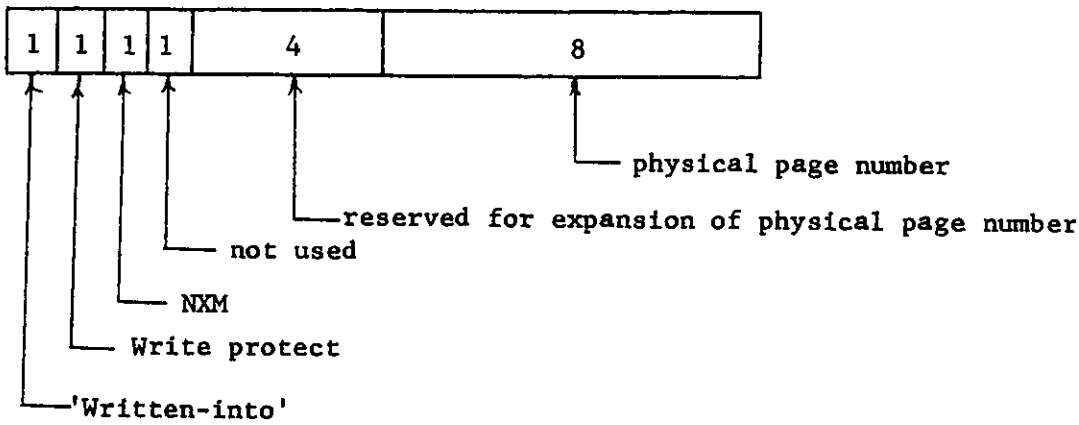


Under this conceptualization the relocator, Kr, behaves as a controller which: (1) responds to all ØØ-, Ø1-, and 1Ø- mode addresses by performing the mapping described above and passing the request along to the switch, (2) responds to six (of eight possible) 11-mode page addresses by performing the mapping described above, (3) responds to 30 11-mode addresses for the relocation registers themselves (in the 'device space').

There are three other properties of the mapping mechanism which have not been mentioned previously:

1. All accesses to trap and interrupt vectors are forces to the processor's 'local' memory. Note that traps and interrupts save the current PS and initialize PS from the trap vector. Thus, by appropriate use of the relocation registers the cost of a context swap can be substantially reduced.

2. The format of each of the 30 relocation registers is as shown below:



- the 'written-into' bit is set by the hardware whenever a write operation is performed on the specified page.
- the 'write protect' bit, when set, will cause a trap on (before) an attempted write operation into the specified page.
- the NXM, 'non-existent memory', will cause a trap on any attempted write into the specified page. Note: this is not adequate for, and not intended for, 'page fault' interruption.
- the 8-bit 'physical page number' is the actual relocation value.

3. Relocation register zero in each bank is identical and, moreover, the stack will be forced into this page (by ignoring the three high-order bits of SP). Thus monitor service routines (entered via a trap) will use the users stack; this is necessary in order to insure that an RTI (return from interrupt) instruction may be properly executed.

Now consider some implications of the mapping scheme with respect to user programming, protection, sharing, interrupts, and i/o.

User Programming

User processes will run under ØØ-, Ø1-, or 1Ø- mode addressing, and probably under ØØ-mode by convention for all but real-time processes where context switching time is especially critical. Under these modes the PDP-11 appears essentially identical to the non-mapped version except that the i/o page is inaccessible and certain instructions will be disallowed (e.g., HALT). In particular the PS and relocation registers are inaccessible and thus there is absolute protection between processes. Since all i/o is in the 11-space, standard devices (e.g., disks) cannot be inadvertently operated. All pages of a (running) process's instantaneous virtual memory (IVM) must necessarily be physically present in Mp. Monitor request (via traps) will be provided to do such things as change the IVM, request a pre-paging operation, lock a page into Mp, release a page, etc.

Protection and Sharing

As described above, at the hardware level protection is absolute since a process cannot reference outside the set of pages specified by its bank

of relocation registers. Also, at the hardware level, sharing is trivial since the relocation registers of several processes may reference the same physical page. Inadvertent destruction of shared pages (e.g., code) is prohibited by write protection. If the shared page is code, however, its author must exercise some care to make it "position independent" (not difficult on the PDP-11), since it may execute from different "page positions" (relocation registers) in the IVM's of different processes. In particular, for example, the FORTRAN library would be written to execute in this fashion. Standard compilers, e.g., Bliss, will produce this form of code. Monitor requests (via traps) will be provided for establishing and controlling such sharing.

The NXM bit mentioned above will be set for all relocation registers not being used by a process (i.e., if its IVM is less than 32kw). Should the process reference one of the registers containig this bit set, it will be assumed to be running amuck and will be interrupted and killed by the monitor. No attempt has been made to provide a "page-fault" mechanism because: (1) the "working set" model strongly suggests it is futile to try to run a process unless it has a reasonable subset of its pages already present, and (2) it is extremely difficult (on the PDP-11) to record the state at the time at which the fault occurred during an instruction.

Interrupts

One of the nice features of the mapping scheme is that, since the PS is uniquely loaded for each interrupt (or trap), a portion of the context swap is inexpensive. For example, monitor request traps can directly enter an address space in which the process description, page table, etc., are

stored. Similarily, device interrupts can activate a process in the (11) address space in which that device service can/must be performed.

I/O

Direct-memory-access devices, such as disk and drum, specify an 18-bit bus address when they read or write to Mp. Thus, these addresses may be automatically relocated into Mp. Since an i/o operation is not likely to be for the process using the Pc at that instant, the ∅1 and/or 1∅ relocation banks can, for example, be used for i/o while the user (Pc) process is using the ∅∅ bank.

TIME OF DAY CLOCK, TIMER, PROCESSOR IDENTIFICATION

Each processor has a local control, Kc, which is used for (1) intercommunication among the other processors, (2) to collect the exact time of day from the central time of day clock, Kclock, and (3) to collect time events such that software times can be constructed in each processor. Information for these three functions is passed on a single bus, called the Processor Intercommunication and Clock-Timer Bus.

Time of Day Clock

The clock is used in several ways: for communication to the user requiring time labeling (e.g., printouts, file labels); for software and system function time measurements; and for internal naming of 'objects' (e.g., pages, files, etc.). For this latter no two objects can have the same identifier, i.e., the same time.

The K.clock operates continuously, providing a count, in microseconds,

post A.B.[1,2] In addition to the 60 bits giving the clock information, four bits are also included in the words to specify the processor number. Each processor may read this information by executing four instructions to read four words. In this way a clock number read by any processor is always unique, even if the clock is read simultaneously by all processors.

In order to avoid timing problems inherent in reading clocks which are subject to change during the reading interval, reading the first word of the clock causes the 60 bit time to be placed in a register. On subsequent reads of the remaining parts of the register the register will not change, thus insuring that the clock does not change during the interval which it is being read. Electrically, this process is accomplished by having the central clock, K.clock, continuously broadcasting the 60 bit number on a bus, together with information telling when the clock may be read. Physically, this can take the form of broadcasting on a time multiplexed basis (e.g., every 0.5 microseconds a different quadrant of the clock count is broadcast) to avoid using a 60-bit wire bus. Alternatively, it may be desirable to broadcast continuously the least significant word, while time multiplexing the most significant part.

Processor Number

The processor number is read by looking at the clock count. This number is variable, by toggle switches. It is also used for signaling other processors using the Intercommunication bus.

---

[1] 'Anno Babbage'

[2] We rejected using a direct encoding, e.g., year (12-bits), month (4-bits), day (5-bits), hour (5-bits), minute (6-bits), second (6-bits), milliseconds (10-bits) and microseconds (10-bits) (for a total of 58-bits) because of the difficulty of obtaining time differences.

Timers

The K.clock also broadcasts periodic time events for use by each processor in generating time-interval interrupts. Each processor is equipped with a counter and the ability to select the frequency with which the counter is to be decremented. An interrupt is generated on a processor when its counter is decremented below zero.

Processor Intercommunication

Intercommunication is carried out among the processors fundamentally by placing messages in memory and having the various processors look at the messages. In order to signal another processor to look for messages, a processor may cause an interrupt to any of the other processors. That is, each processor has a wire on the intercommunication bus which is used to carry the input event from all the other processors. There is no information on the intercommunication bus that identifies the processor requesting the interrupt. Instead, the interrupter is identified by looking in memory at some predetermined location. In order to allow the system to be partitioned into arbitrary, totally isolated, subsets, manual switches are provided to prohibit such subsets from generating mutual inter-processor interrupts.

PERFORMANCE ANALYSIS HARDWARE

In order that we can effectively do research into the nature of the multiprocessor behavior, it is necessary to have performance measuring ports throughout the system. The basic philosophy in the design of the clock was to have a very accurate clock which would enable software monitoring. In addition, it is necessary that we be able to measure the processor-memory

performance. This requires hardware because the times are short, and the data is not accessible by other means.

The most accurate method under consideration is to associate a small memory with each crosspoint intersection. This can be constructed efficiently by having a memory array for each of the m rows, since control is on a row (per memory) basis. When each request for a particular row is acknowledged a 1 is added to the register corresponding to the processor which gets the request. In this way we can measure the exact amount of work done by each processor. Note that i/o and file traffic is known since the file sizes, words transferred, etc. are known. Such a scheme does have the drawback of adding significant hardware to the switch, hence lowering reliability.

The performance with even a large number of processors seems quite high (see Performance of the System, Section VI). Therefore, little may be gained by measuring the performance accurately. Knowing the performance of individual P's may be more interesting, and is somewhat easier to implement.

It would be most desirable to measure data about processor instruction performance as measured by execution. The information which could be obtained includes:

1. instructions executed
2. memory accesses
3. instruction types (relatively difficult)
4. instructions, accesses, instruction types.

Information of the above type would be particularly useful in regard to generating very accurate, repeatable billing statistics. For example, the

actual number of instructions executed could be billed. Any scheme based on time has some error because of background interrupt handling, and memory interference.

PROCESSOR MODIFICATIONS

Processor modifications to implement the relocation scheme as well as the switch are minimal and straightforward. The module on the 11/20 Pc that requires the most modification will be the M725 Bus Interface and IR card. This card contains not only the two high order address bits of the 18 bit bus address but also the necessary signals and gating to read and load the status word. The necessary modifications are made by disabling the conflicting functions on the M725 and providing alternative functions on an additional card; hence we use as much existing logic as possible.

The relocation registers will be in close proximity to the processor and therefore can be wired directly. Thus bus drivers and receivers are not needed.

The memory bus address decoders (M109) for local memory appear to be adequate. The device address decoders (M105) are not usable because of their slow speed. Therefore, a new device address decoder card is being constructed.

Relocation Register Additions

Each processor port will have 30 relocation registers, each 16 bits wide. The processor is such that the registers should have a fast access time, but the cycle time can be rather long. Using five 5N7489N Texas

Instrument 64 bit memory chips, a 30X16 bit memory with an access time of 35 ns and a cycle time of 100 nsec. can be built for about $100.

CONNECTION TO PDP-10

C.mmp will be connected to PDP-10 System A, via a PDP-10, DL10 adapter. This adapter allows up to four minicomputers to access directly the PDP-10 memory at relatively high data rates (up to 4 Mhz bit rates). The data is transmitted under program control of the PDP-11 processor, and transfers are variable length byte, variable length character strings. Each computer transferring data into the PDP-10 memory is provided 64 channels (via one physical channel). Each channel has control status words kept in the PDP-10 memory, which control the format and location for packing and unpacking in the PDP-10. That is, a byte pointer to the character string being transferred is kept for each channel. The PDP-11 accesses each of the channels by unique addresses in its memory space. By continuously writing information in a particular PDP-11 address causes a byte (or word) string to be written in PDP-10 memory at the location and form specified by the PDP-10 control words. Similarly, a byte string in PDP-10 can be read into the PDP-11 memory, by having the PDP-11 continuously read the fixed address corresponding to the channel number. An interrupt channel is also provided for signaling task completion, errors, etc., between the two computers.

V. OPERATING SYSTEM CONSIDERATIONS AND IMPLICATIONS

The operating system for the multiprocessor, which we will call Hydra, is intended (initially) to support the following kinds of activities:

1. TTY handling (for several, possibly dissimilar hosts)
2. Display processing
3. Speech/Vision device handling
4. Speech/Vision real-time processing
5. Synchronous communication switching
6. "Dedicated" systems such as BASIC, APL, and text editing

Later versions of Hydra will support more general user-type programming. Note, however, that the initial applications have real-time and/or system characteristics. This has an important influence of the design of Hydra. It implies that the initial system must provide good multiprogrammed/multiprocessor scheduling, good process communication and synchronization mechanisms, etc. In short, it must provide clean interface and good primitives for systems building. On the other hand, it need not initially provide fancy device-independent file i/o, an elaborate user-terminal interface, etc. These features will be built, in a layered fashion, on top of the kernel so that they may be easily altered and so that several versions may be run simultaneously.

Hydra will be coded in BLISS-11,which runs on the PDP-10.

Some of the other objectives of the initial Hydra (not in any particular order) are:

1. One objective in the multiprocessor design is to allow the total configuration to be partitioned into disjoint subconfigurations. It is not the intention that Hydra cope with this partitioning other than: (a) it must be able to run with whatever resources are available so long as they include certain minimal facilities, and (b) it must be able to (software) lockout a subset of resources in preparation for partitioning.

2. The virtual address space instantaneously available to a process is limited by the 16-bit addresses of the PDP-11. More specifically, the instantaneous virtual memory (IVM) consists of eight 4k (16b word) pages named by the relocation registers. However, a process will be allowed a much larger total virtual memory (VM), perhaps 4000 pages. Monitor traps will be provided for the user to re-define that portion of his VM which is to be his IVM.

3. The relocation registers are not fitted with page-fault detection, implying that all pages of the IVM must be in core when a process is running. The IVM is the user's "working set" and will be kept core-resident for high priority processes. Monitor traps will be provided for a process to request prepaging as well as to mark that specific pages are to be kept core-resident.

4. A premise of the multiprocessing design is that not all Pc's need be identical, either in terms of their instruction sets

or in terms of the devices accessible to them (e.g., displays may be on specific processors). Therefore, a process may be eligible to run only on a subset of the processors. The scheduling algorithm must cope with this problem. The current plan is to associate a mask defining the set of processors on which it is eligible to run, and to use a (dynamic) priority scheduling algorithm to schedule the highest priority process able to run on the available processor.

5. In operating systems it is not uncommon for a "job" to consist of several inter-dependent processes; however, there is usually an enforced ancestral relation between these processes. Exploiting the multiprocessor, as in the speech/vision task, makes such mandatory relations undesirable. (It implies, among other things, too many levels of interrupt handling.)

6. Context switching time can be a problem. In general (on the 11/20) 13 registers must be saved/restored, 11 of them under program control. Thus, in the best case 48 memory references will be made. This is too many for some device service routines (e.g., for a scope). For these, relocation registers (from the ∅1 or 1∅ sets) will be set aside and they will save only used registers. This can reduce the time to five memory references (one accumulator saved).

7. Processor performance can be improved by minimizing conflicts for memory banks. We are studying this problem, but do not have a proposal yet. (See the Section on Performance for estimates of degradations assuming random references.)

VI. PERFORMANCE OF THE SYSTEM

The performance of the multiprocessor can be computed almost exactly given m (the number of memories), p (the number of processors), t_d (the delay introduced by the switch), and the following parameters:

t_p The mean of a distribution of the processor time between the completion of one memory request and the next request.

t_a, t_c The access time and cycle time for the memories to be used.

$t_w = t_c - t_a$ The rewrite time of the memory.

t_{io} The average transfer times of high speed i/o transfers, e.g., drum or disk, which interfere with processor requests; in this analysis we ignore this effect.

Strecker (1970) gives closed form solutions for the interference in terms of a defined quantity, the UER (unit execution rate). The UER is, effectively, the rate of memory references and, for the PDP-11, is approximately twice the actual instruction execution rate. (A single instruction on the 11 may make from one to five memory references, but is about two on the average.) Strecker give the following relations, neglecting i/o transfers,* and assuming random memory references:

---

* A simple argument indicates that i/o traffic, t_{io}, is relatively insignificant and so was not considered in these figures. For example, transferring with four drums or 15 fixed head disks at full rate is comparable to one Pc.

$t_p = t_w$: $\quad UER = (m/t_c)\ (1 - (1 - 1/m)^p)$

$t_p < t_w$: $\quad UER = \frac{m}{t}\ \frac{1 - (1 - 1/m)^p}{1 - (1 - 1/m)^p}\ (t_w - t_p)/t_c$

$t_p > t_w$: $\quad UER = (m/t_c)\ (1 - (1 - P_m/m)^p)$

where $P_m + (m/p)(\frac{t_p - t_w}{t_c})(1 - (1 - P_m/m)^p)) - 1 = 0$

Various speed processors, starting with the Model 20, various types of memories, and various switch delays, t_d can be studied by means of these formulas. Switch delays effects are calculated by adding to t_a and t_c, i.e., $t_a' = t_d + t_a$; and $t_c' = t_d + t_c$. In particular the following cases are given in the attached graphs (Figures 4a-f). The plots show UER x 10^6 as a function of p for a fixed m, for various parameters of m. m-parameters are the triplet: (t_c, t_a, t_d).

$m = 8, 16$

$p = 1, 5, 10, \ldots, 35$

t_p = 700 ns (11/20), 450 ns, and 200 ns

t_d = 190, 270 ns

t_c, t_a = (300,0); (400,250); (650,350); (900,350); and (1200,500) ns

The two values of t_d correspond to the estimated switch delay in two cable-length cases: 10' and 20'. The t_c, t_a values correspond to the six memory systems being considered.

In addition to the Unit Execution Rates for Mp references, two separate measures of the degradation were obtained:

Fig. 4a. Performance vs Pc's for 8 Mp; Pc(Model 20; tp: 700 ns)

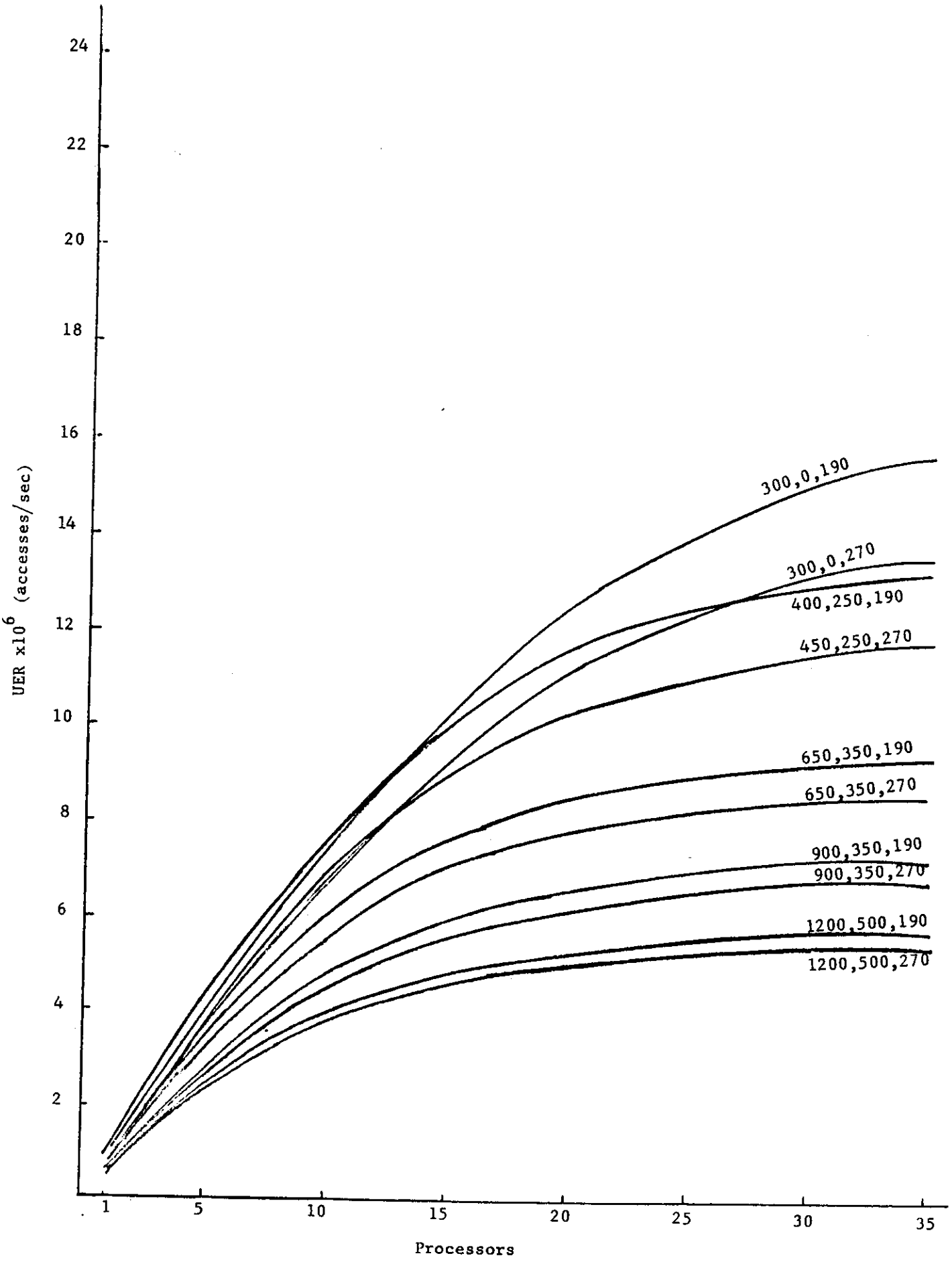


Fig. 4b. Performance vs Pc's for 8 Mp; Pc(#2; tp: 450 ns)

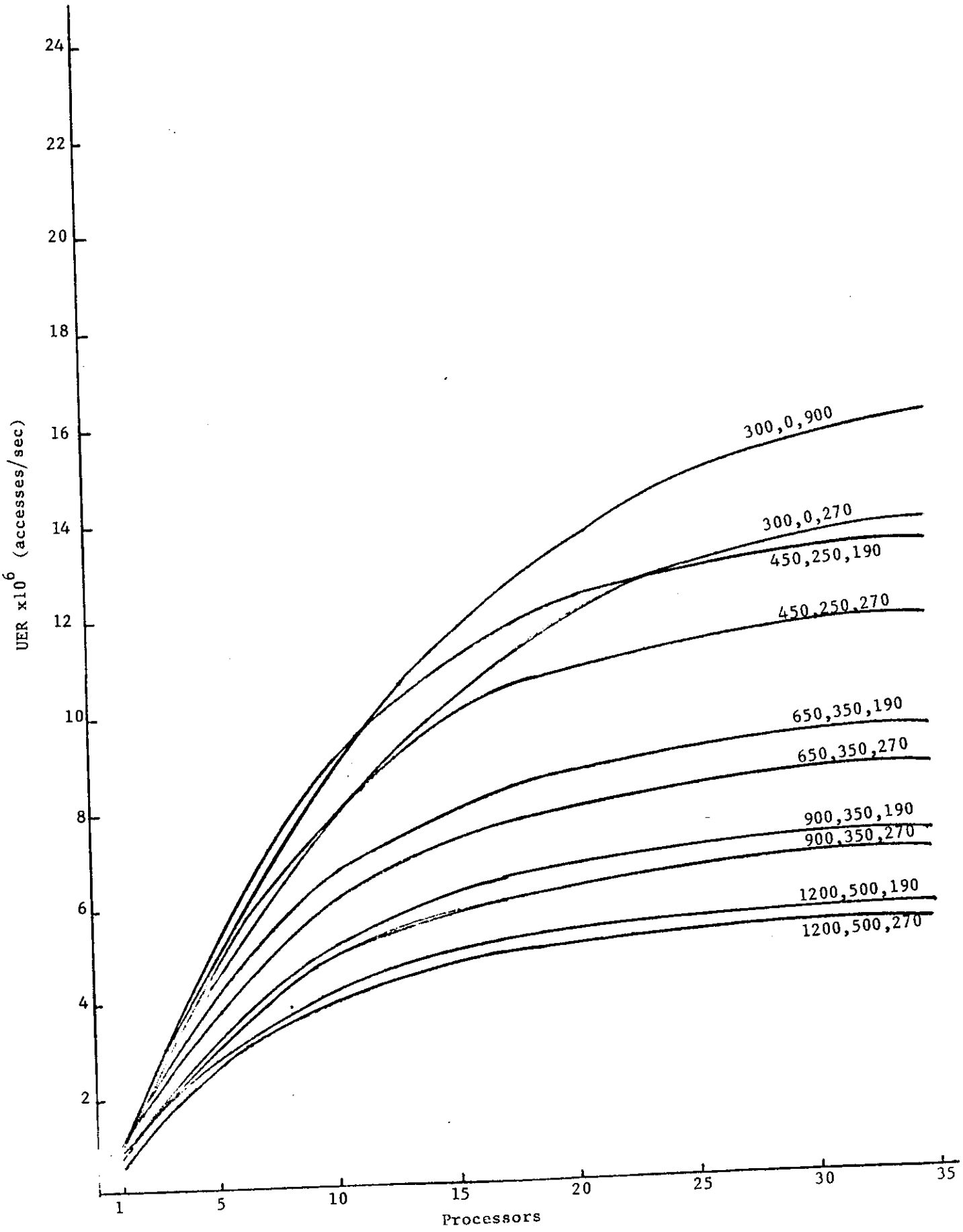


Fig. 4c. Performance vs Pc's for 8 Mp; Pc(#3; tp: 200 ns)

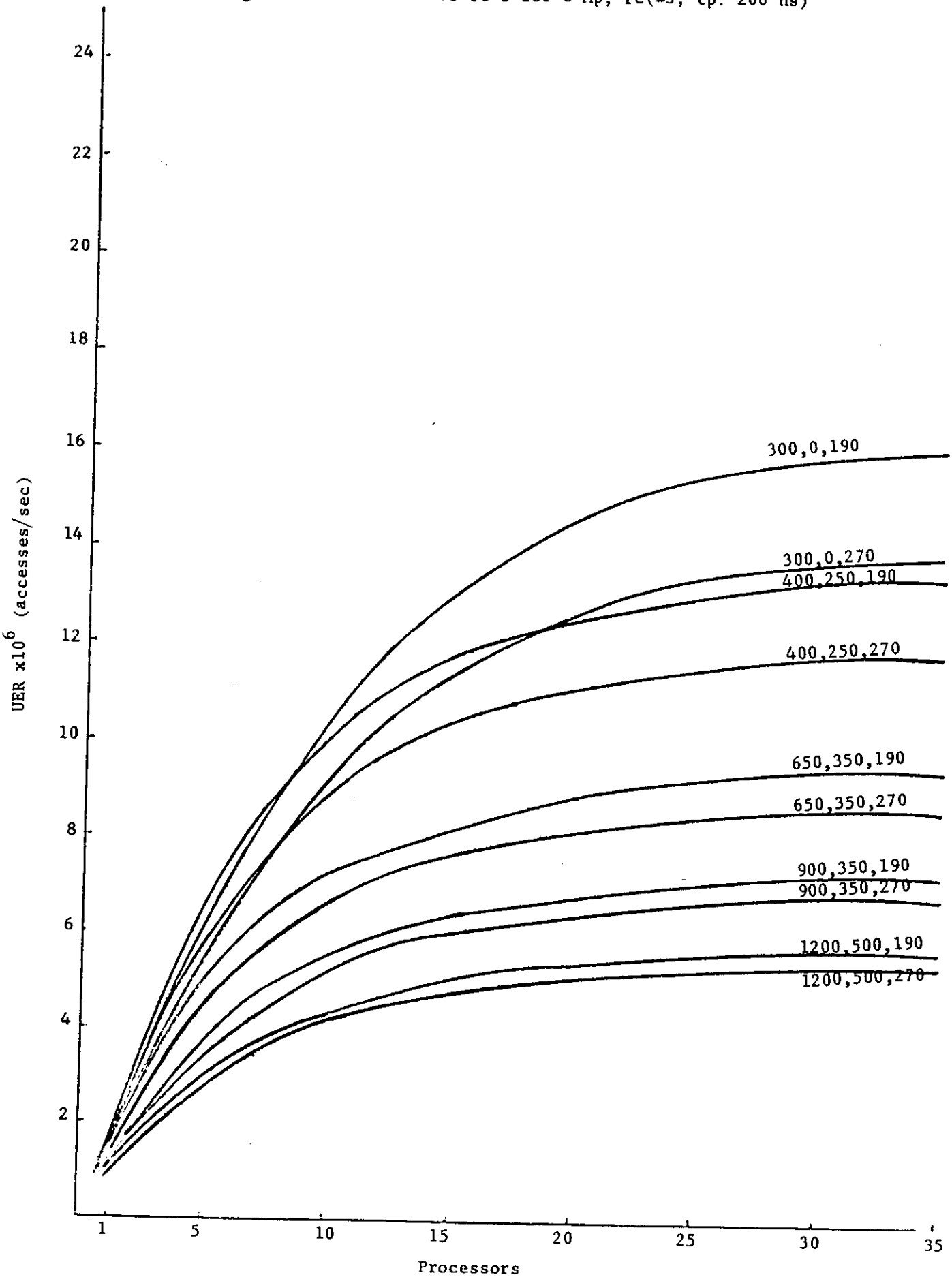


Fig. 4d. Performance vs Pc's for 16 Mp; Pc(Model 20; tp 700 ns)

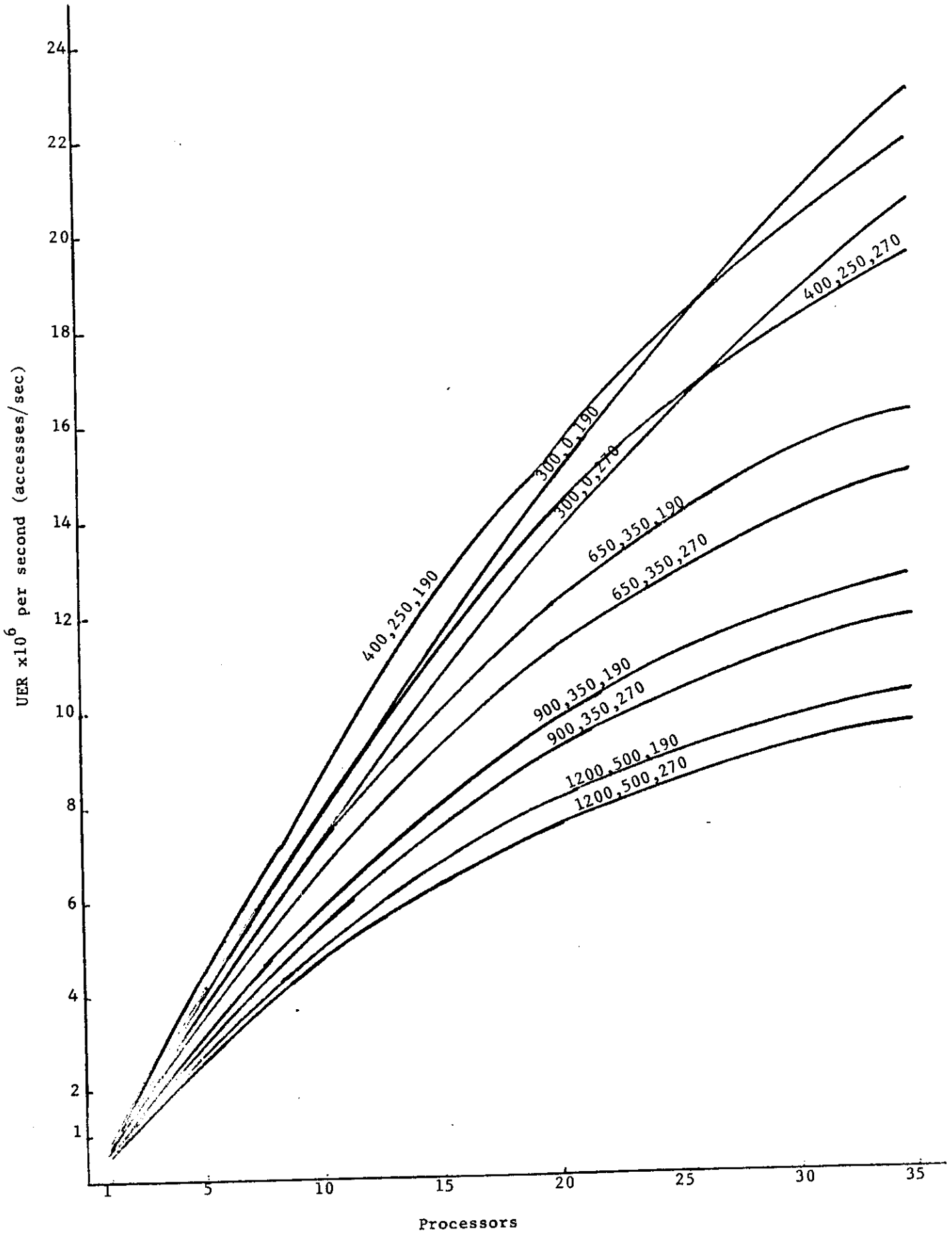


Fig. 4e. Performance vs Pc's for 16 Mp; Pc(#2; tp: 450 ns)

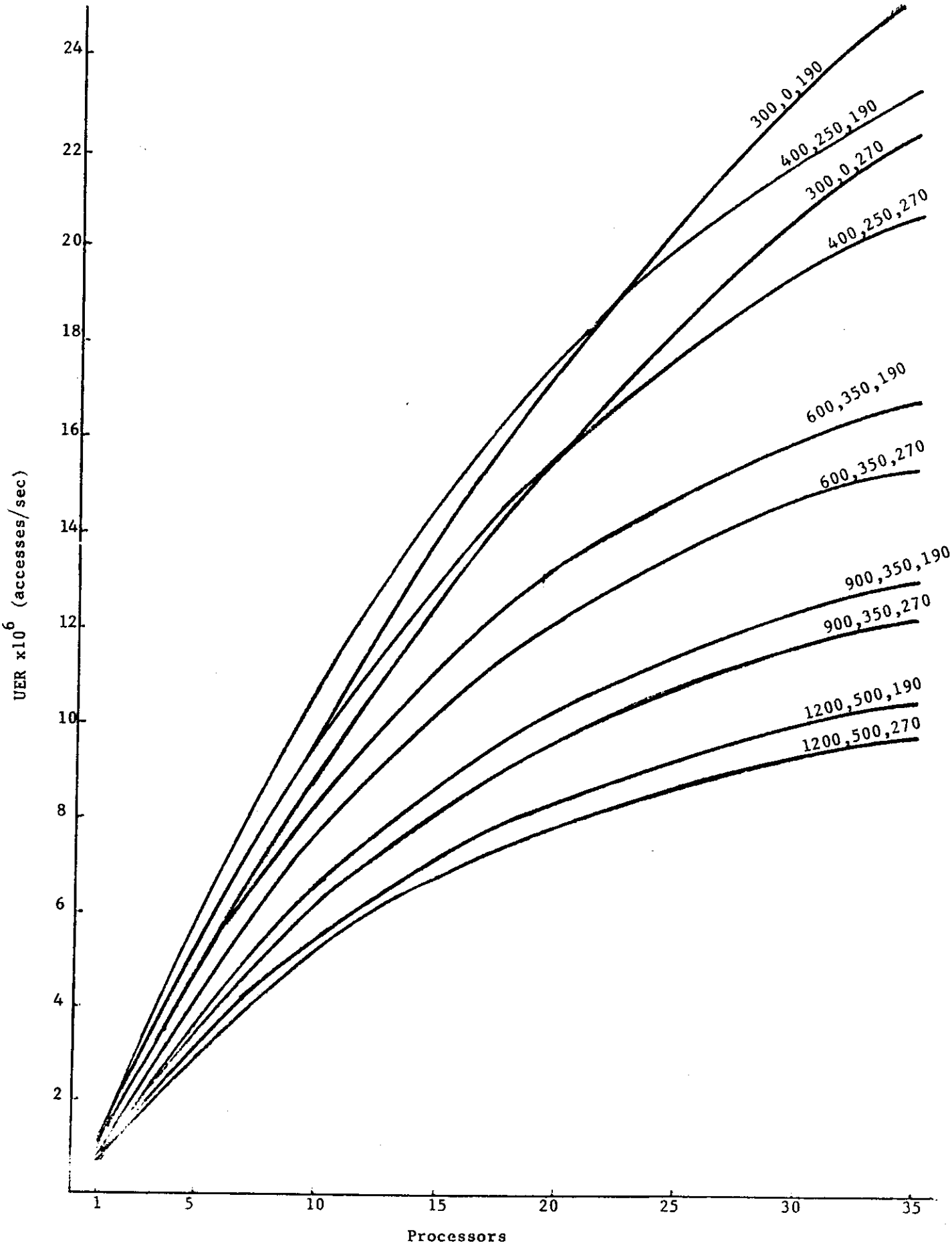
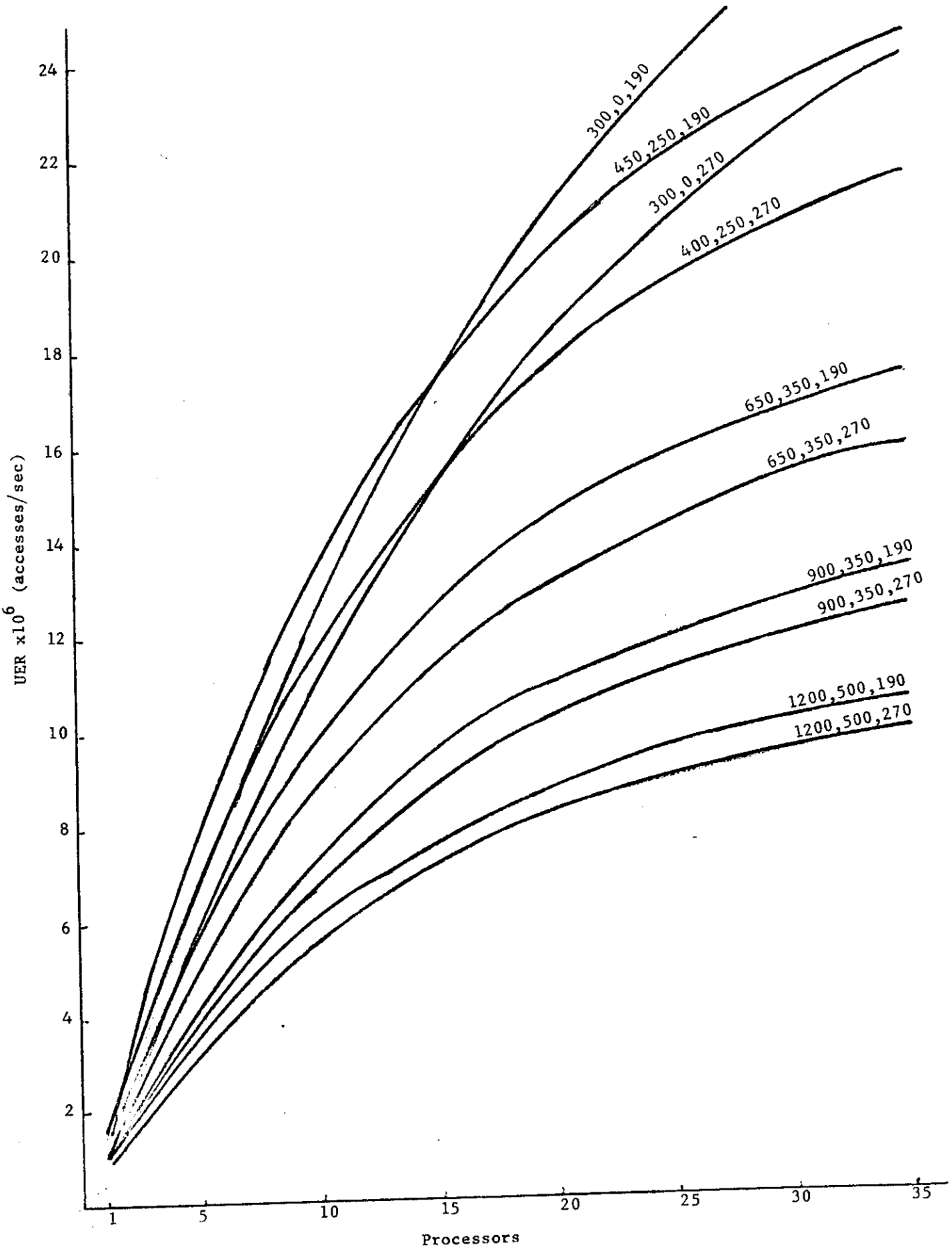


Fig. 4f. Performance vs Pc's for 16 Mp



D_1 The (percent) degradation due to both memory interference and the switch delay

D_2 The (percent) degradation due to only memory interference effect

Only sample values have been incldued because of the volume of data, and the following cases are shown in Table 2:

$m = 8, 16$

$p = 1, 5, 10, 15$

$t_p = 700, 450, 200$ ns

$t_d = 190$ ns

$t_c, t_a = (300, 0)$; $(400, 250)$; and $(650, 350)$ ns

Note, in particular, that D_1 for the p=1 case is the performance degradation due to the delay introduced by the switch. In effect, this indicates the number of P's necessary to overcome switching delays.

Two unmistakable conclusions can be drawn from the data: (1) the delay in the switch, t_d, is the dominant factor, and (2) it is better to use the fastest memory you can get, unless the cost is too high. Also, because of the long switch delay, faster processors are not needed, or rather, cannot be used effectively.

More processors will yield higher performance until a sharp cut-off occurs when processing capacity has absorbed the total Mp capacity. This can be seen from the figures when a slow memory or fast processor is used. Since the processor costs are small (in the order of \$12K/processor with a local Mp and some terminals), an objective function based solely on maximizing performance/cost should not be used. The performance/cost is a fairly flat function near maximum, since the processor cost is a small part of the system cost.

Table 2 Switching and Mp Degradation

note td = 190 ns

11/20 Processor (tp: 700 ns)

P	m = 8						m = 16					
	(300,0)		(400,250)		(650,350)		(300,0)		(400,250)		(650,350)	
	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2
1	16.0	--	18.3	--	16.0	--	16.0	--	18.3	--	16.0	--
5	19.8	4.6	25.2	8.3	26.5	12.6	17.8	2.2	21.6	4.1	21.2	6.3
10	25.1	10.9	34.1	19.3	38.7	27.0	20.2	5.1	26.0	9.4	27.6	13.9
15	31.1	18.0	42.7	29.9	49.0	39.3	22.8	8.2	30.3	14.7	33.7	21.1

Processor, Pc#2 with tp: 450 ns

P	m = 8						m = 16					
	(300,0)		(400,250)		(650,350)		(300,0)		(400,250)		(650,350)	
	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2
1	20.2	--	24.1	--	20.2	--	20.2	--	24.0	--	20.2	--
5	26.0	7.2	34.6	13.9	35.1	18.6	23.0	3.5	29.3	7.0	27.9	9.6
10	33.6	16.8	46.2	29.2	48.3	36.4	26.6	8.0	35.6	15.3	36.4	20.3
15	41.4	26.5	55.8	41.8	59.6	49.4	30.4	12.7	41.5	23.0	43.7	29.5

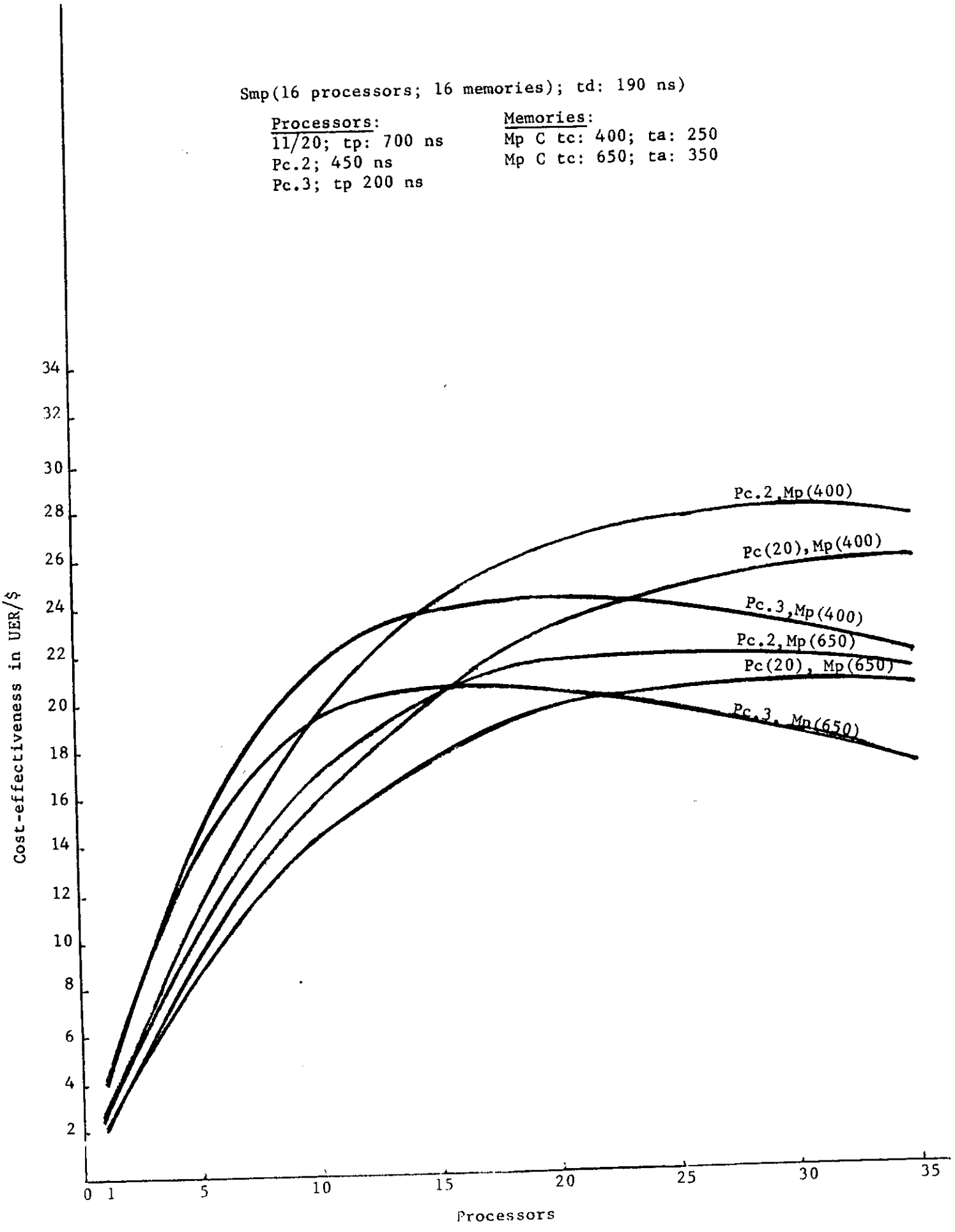
Processor, Pc#3 with tp: 200 ns

P	m = 8						m = 16					
	(300,0)		(400,250)		(650,350)		(300,0)		(400,250)		(650,350)	
	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2
1	27.5	--	34.8	--	26.8	--	27.5	--	34.9	--	27.2	--
5	36.7	12.7	50.8	24.6	47.0	27.6	32.1	6.3	43.7	13.4	38.5	15.6
10	47.3	27.2	63.6	44.2	61.8	47.8	37.7	14.0	52.3	26.7	49.1	30.1
15	56.2	39.6	71.8	56.8	70.8	60.1	43.0	21.3	59.1	37.1	56.9	40.9

In Figure 5 we have plotted the performance/cost. Here note that if we start with a small size configuration of five Model 20's, the cost is only $375K, while the performance is 4.5×10^6 accesses/second (UER), giving a cost-effectiveness of 12. Going to 1 Pc#2's later, provides about a UER of 15×10^6. While the cost is only $625K (cost-effectiveness is 24). Following this strategy provides a very cost-effective system, once a reasonably large number of processors are used. In fact, in the range of 15 - 30 processors the cost-effectiveness is relatively constant, while the absolute performance nearly doubles. The most impressive region is with a fast Pc of 200 ns from 10 - 35 P's. Here performance/cost varies by $\pm$ 5% and performance ranges from 13×10^6 to 26×10^6.

Our interest in this exercise is to determine whether the faster memory seems worthwhile. We think it is, solely on the current configuration. Since we would like to fabricate specialized, faster processors eventually, the additional bandwidth seems essential.

Fig. 5. Cost effectiveness (UER/$) vs Pc's.



VIII. CONCLUSIONS

We have given an overview of the design of the proposed C.mmp. We have described the computational requirements of our present research that the system is to serve, and translated these into a set of specific constraints used to shape the design. We have also described the wide range of additional research payoffs that can rather clearly be expected on the basis of the system.

Many details of the design are left unspecified. These are, and will be, given in detailed memoranda on specific parts. For example, a separate paper is under preparation on the details of the Smp design: its fabrication and precise operation, including processor-port assignment control, manual switching, etc. Similarly, there will be a detailed description of the operating system structure and operation. But the essential features of the system are now firm.

REFERENCES

Barbacci, M., H. Goldberg and M. Knudsen, C.ai (P.LISP) -- a LISP processor for C.ai, Department of Computer Science, Carnegie-Mellon University, 1971.

Bell, C. G., "Minicomputer Architecture," IEEE Conference, March 1971.

Bell, C. G., R. Cady, H. McFarland, B. Delagi, J. O'Laughlin, R. Noonan and W. Wulf, "A New Architecture for Minicomputers - The DEC PDP-11," SJCC 1970, pp. 657-675.

Bell, C. G., P. Freeman et al, C.ai: A Computing Environment for AI Research - Overview, PMS,and Operating System Considerations, Department of Computer Science, Carnegie-Mellon University, May 1971.

Bell, C. G., J. Grason, S. Mega, R. Van Naarden and P. Williams, The Design, Description and Use of the DEC Register Transfer Modules (RTM), Department of Computer Science, Carnegie-Mellon University, 1970.

Bell, C. G., A. N. Habermann, J. McCredie, R. Rutledge and W. Wulf, "Computer Networks," Computer Science Research Review, Carnegie-Mellon University, 1969.

Bell, C. G., H. C. Lauer, and B. Randall, A Switch for Connecting Computer Components, The University of Newcastle Upon Tyne (U.K.), Technical Report SRM/16, June 3, 1971.

Bell, C. G., H. C. Lauer and B. Randall, S(Magnabus) - A Multi-Unibus Switch for PDP-11, CMU memorandum, May 27, 1971.

Bell, C. G., and A. Newell, Computer Structures, McGraw-Hill Book Company, 1971a.

Bell, C. G., and A. Newell, "Possibilities for Computer Structures, 1971," FJCC 1971b. (in press)

DEC PDP-11 Documents: Programmer Reference Manual and Unibus Interface Manual.

Krutar, R., Personal communication, 1971.

McCracken, D. and G. Robertson, C.ai (P.L*) -- a L* processor for C.ai, Department of Computer Science, Carnegie-Mellon University, 1971

Strecker, W. D., "An Analysis of the Instruction Execution Rate in Certain Computing Structures," Ph.D. Dissertation, Carnegie-Mellon University, ARPA Report, 1971.

Wulf, W., Personal communication, 1971

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
C.mmp: THE CMU MULTIMINIPROCESSOR COMPUTER (Requirements and Overview of the Initial Design)

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific final

5. AUTHOR(S) *(First name, middle initial, last name)*
C. G. Bell, W. Broadley, W. Wulf, A. Newell, C. Pierson, R. Reddy, S. Rege

6. REPORT DATE
August 24, 1971

7a. TOTAL NO. OF PAGES
63

7b. NO. OF REFS
15

8a. CONTRACT OR GRANT NO. F44620-70-C-0107

b. PROJECT NO. 9769

c. 61102F

d. 681304

9a. ORIGINATOR'S REPORT NUMBER(S)
CMU-CS-72-112

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES
TECH OTHER

12. SPONSORING MILITARY ACTIVITY
Air Force Office of Scientific Research(NM)
1400 Wilson Blvd.
Arlington, Va. 22209

13. ABSTRACT

This document describes a proposed CMU multiprocessor system to be constructed around a set of PDP-11 computers connected through a crosspoint switch to a large sharable primary memory. The present design constitutes a solution to a specific set of needs existing in our environment. The system has research consequences that reach well beyond the particular demands it was designed to satisfy. For although multiprocessors have been much talked about and advocated, there are remarkably few operational systems more complex than dual-processor systems, and even fewer documented scientific investigations into their performance and operating structure.

This document is limited to a presentation and analysis of the (hardware) system. It gives enough description of the usage requirements, software, and the research potentials and problems to make clear why we believe the effort to be a sound one. It does not attempt a systematic discussion of the field of multiprocessor research, nor of alternative systems that might be of interest, either to meet our computing demands or as research directions.

Section II discusses the requirements and research potential. Section III lists the design constraints adopted. Section IV lays out the PMS structure of the system. Section V describes the main specifications of the operating system. Section VI provides some details on a performance analysis.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
multiprocessor						
crosspoint switch						
operating system						
performance analysis						