C.ai: A COMPUTING ENVIRONMENT FOR AI RESEARCH

Overview, PMS and Operating System Considerations

by

G. Bell (Co-Chairman)

P. Freeman (Co-Chairman)

and

M. Barbacci
S. Bhatia
W. Broadley
R. Chen
L. Erman
H. Goldberg
W. Huen
M. Knudsen
D. McCracken
A. Newell
R. Reddy
S. Rege
G. Robertson

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

May 7, 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. This document has been approved for public release and sale; its distribution is unlimited.

TABLE OF CONTENTS

ABSTRACT

A computer for artificial intelligence research is examined. The design is based on a large, straightforward primary memory facility (about 8 million 74 bit words). Access to the memory is via at least 16 ports which are hardware protected; there is dynamic assignment of the memory to the ports. The maximum port bandwidth is 8,600 million bits/sec. Processors for languages (e.g., LISP) and specialized terminals (e.g., video input/output) can be reliably connected to the system during its operation. The approach is evolutionary in that high performance processors, such as the Stanford AI Processor, can be connected to the memory structure, giving an overall power of at least 100 times a PDP-10 (and 200 to 300 times a PDP-10 for list processing languages) for 10 processors -- although 20 processors can be attached. Using this approach we might expect 40 - 80 million PDP-10 operations/second.

At the same time, special language processors (P.ℓ) can be designed and attached. These processors give even larger power increases, but for restricted language use. Two processors, P.LISP and P.L*; were examined for the LISP and L* languages and are reported on separately.

A plan for building the machine in increments over the next three to five years is examined. Specific schedules are proposed.

Concurrent with the operation of the machine, there should be research into the design of hardware, software and theory of constructing large scale computing facilities with maximum modularity.

ACKNOWLEDGMENTS

The basic PMS structure of C.ai and that of the L* and LISP language processors was developed in the Carnegie-Mellon University Computer Science Department, in a project seminar on list processing machines. This report is based on the working papers developed in the CVAM seminar.

The group was informally organized and met frequently as a whole, but the main work for the reports was carried out in the following sub-groups:

CVAM:PMS - W. Broadley (Chairman), S. Bhatia, and S. Rege

Specification of PMS, circuitry, technology, and overall performance.

CVAM:OS - P. Freeman (Chairman), S. Bhatia, L. Erman, W. Huen, D. McCracken, R. Reddy and G. Robertson

Specification of an operating system.

CVAM:P.L* - D. McCracken and G. Robertson (Chairmen), R. Chen

Design of a hardwired processor for an L* kernel.

CVAM:P.LISP - M. Barbacci (Chairman), H. Goldberg, and M. Knudsen

Design of a hardwired processor for LISP.

Many members of the Carnegie-Mellon Computer Science Department served as critics and consultants on the design, especially A. Kendziora, J. McCredie, A. Newell, R. Reddy and W. Wulf. Alan Kay of Stanford also interacted with the group frequently and in detail.

A draft of this document was presented at a working meeting held at Bolt, Beranek and Newman in Boston, April 26-27, 1971. Representatives

from the Advanced Research Projects Agency, Bolt, Beranek and Newman, Carnegie-Mellon University, Massachusetts Institute of Technology, Stanford Research Institute, Stanford University, and System Development Corporation were present. Their comments and suggestions were of value in preparing this final version.

Gordon Bell and Peter Freeman integrated the working papers into this document and did the final editing.

OVERVIEW OF THE SYSTEM

Consideration of the problem of designing and building an optimal computer for ai research quickly leads one to the realization that there may not be a feasible solution. The numerous constraints, wide variations in computing style, and the impossibility of defining the ai problem narrowly seem to make this a certainty. Thus, the major premise of the design we are about to present is that if one wishes to provide ai researchers with better computing tools, one must, in fact, provide an environment in which many varied tools may be developed and used. Our design should be viewed as a specification of such an environment.

REQUIREMENTS FOR AI COMPUTING

In Bell and Newell's Computer Structures (McGraw-Hill, 1971)* a number of special function computers ranging from business to scientific are described. The characteristics of machines used for ai research appear to span this spectrum, exhibiting the max of each characteristic attribute.

Memory Size

The primary memory is larger in an ai environment than with almost all scientific computers because the local program data base is typically larger than for scientific applications. Here we assume that the average program size in this environment is 250,000 74-bit words (64 bits of

---

*The PMS notation presented therein and used throughout this report is based on seven primitive component types: P-processor, M-memory, S-switch, L-link, K-controller, T-transducer, D-data operation. A computer composed of primitive components is represented by C; hence, C.ai for "ai computer."

seems to be adequate because decisions can be bound in software and later changed.

There is almost uniform agreement (at the meetings on the ARPA list processing machine) that a large (and probably linear) addressing space is essential. The other noticeable point of agreement is that the PDP-10 instruction set with only a few modifications is all that is needed for the immediate future.

There is clearly a need for general-purpose processors with clean order codes and specialized instructions for characteristic ai processing operations. But, in addition, the magnitude and closely defined nature of many ai tasks makes it very desirable to have highly specialized processors as well.

The PDP-10 As The Current ai Computer

Because the DEC PDP-10 is the current most popular machine for ARPA-supported ai research, it is worth briefly examining the reasons for its popularity.

1. Price: It is the only computer that the group can afford. (A 360 Model 44 is also a candidate in this range that has been overlooked and it is worth asking why.)

2. ISP: The instruction set is very straightforward, providing power but with none of the anomonalies in addressing, instruction set size, data types, etc. that accompany most machines (e.g., 360, 1108 or Sigma 7 family).

3. PMS Structure: The machine has been easily approachable by all to interconnect any kind of device from foreign memory to TV camera. Indeed, the PMS structure, now eight years old, is only being slightly revised to handle larger and faster memories. This ease of interfacing was initially used internally by DEC to administer various memory and peripheral designs; later, this policy boomeranged by allowing anyone to connect any kind of memory to a PDP-10. In contrast, the IBM processor-memory interfaces are usually so well guarded and obscure that

1. Mp - A simple primary memory structure with 16 multiple ports for connecting a variety of high speed processors of the Stanford AI processor design, special language processors, secondary memories and specialized i/o transducers.

    a. A primary memory size of 620 megabits with individual port rates of 74, 148, 222 or 296 bits/port memory access (550 ns); 135, 270, 405, or 540 megabits/sec per port;

    b. A total information rate to all ports of 2.1 to 8.6 gigabits/sec;

    c. Memory port widths of 72+2 parity or 64+10 single error correction and double error detection bits;

    d. The 16 ports can be further demultiplexed to provide more ports;

    e. Each port has a memory port control for dynamically reassigning 64k word blocks to each processor and protecting the memory from neighboring processors. The port control also includes statistics data gathering and error control.

    f. A mode of operation which provides nearly 100% uptime.

2. Ms - Secondary memory bandwidth to allow swapping. The loading time for a single 100,000 word program would be roughly .14 sec. The worst case swap time would be .3 seconds. Thus, assuming five drums, 15 programs could be swapped per second.

3. P - Multiple approaches for providing processing power. These range from a conventional PDP-10 to specialized hardwired list processors. This permits development of highly functionally specialized processors.

times a PDP-10 for multiple processors. The cost of these processors appears to be quite low ($50,000-$100,000 each). They are described in two reports of the CMU Computer Science Department: "C.ai (P.L*) -- An L* Processor for C.ai," D. McCracken and G. Robertson, and "C.ai (P.LISP) -- A LISP Processor for C.ai," M. Barbacci, H. Goldberg, and M. Knudsen. The abstracts of these reports appear as Appendices 2 and 3 of this report.

3e. Another alternative would be to construct a central processor to give an instruction encoding improvement over the PDP-10. For example, a processor based on the stack and instruction format of the PDP-11 might be desirable. This approach may allow high performance processors to be built more easily.

4. Modularity in PMS structure. Here we hope to do significantly better than the PDP-10 by providing better protection among the processors at the memory ports. We believe that it will never be necessary to turn off computer power for any reason (except for cooling equipment failure).

5. Uniform interprocessor communication. A second type of interface has been added which has a protocol like the PDP-11 and allows intercommunication among the various processors. Like the PDP-10 i/o and memory busses, this type bus should be able to be used over a significant period of time. It allows the transmission of data at high rates. Unlike most other busses,

a. A large inventory of modules that would facilitate design of special experiments. The module types would include: caches, general purpose microprogram controls, arithmetic units, buffers (queues), port switches (to increase the number of memory ports), mapping, interfaces to other computers, component exercisors, etc.

b. Programmers and engineers to carry out many of the designs and assist in system integration of the devices.

The design as proposed would operate at a single central site to give large memory and to decrease the operating cost. However, the design and the construction strategy are such that at almost anytime in the project the machine could be partitioned physically for multiple site operation. The most practical size for economy and reliability would no doubt require at least two processors and 1 to 2 million words of primary memory.

9. The system could be operational within one year at a central site. At this time although any amount of primary memory could be available, only two PDP-10 Tenex processors need be available. Over the next few years, more processors and memories could be added.

10. Research should go along with making such a modular system laboratory. The general direction would be to explore hardware, software, and theory that made the interconnection of modules of the above type easily interconnected. In this way, a system of any type could be constructed easily.

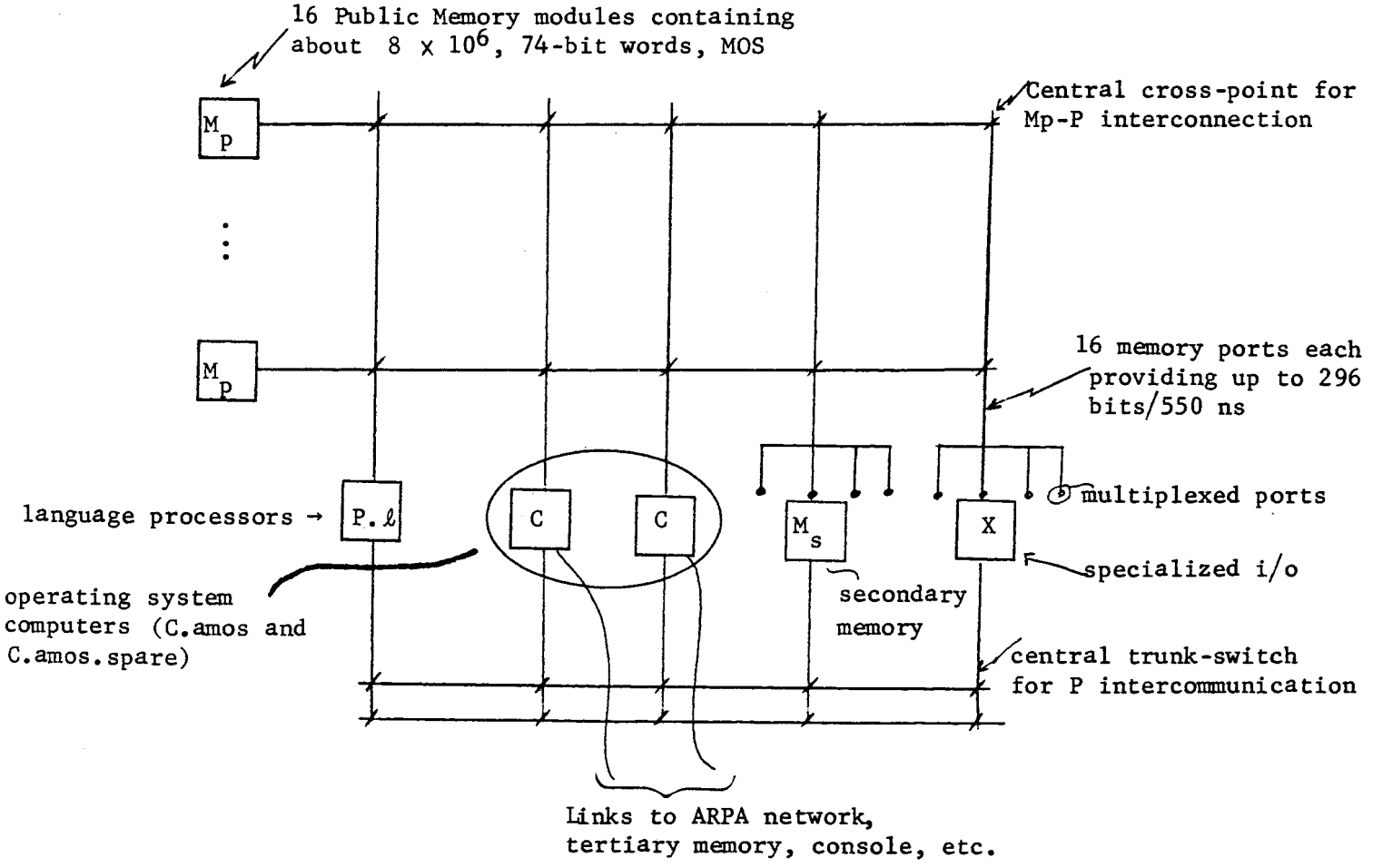
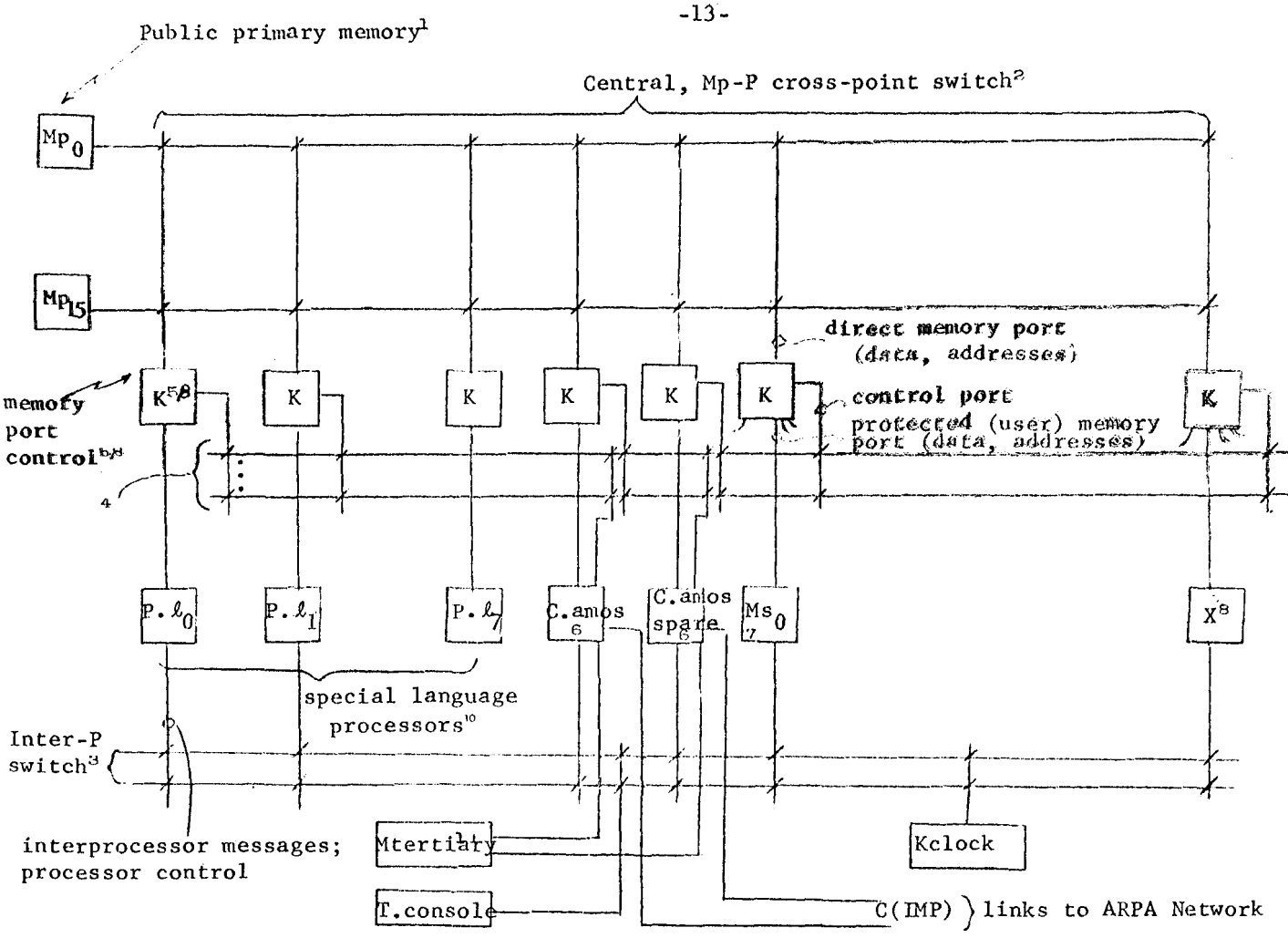
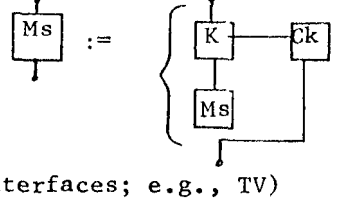


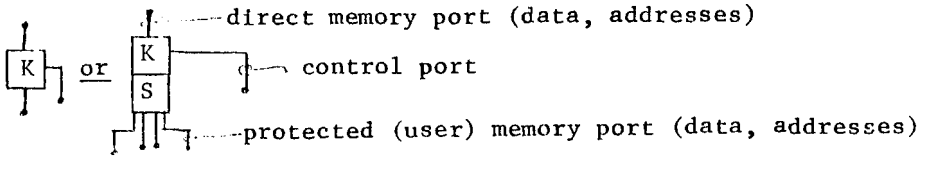
Figure 1. PMS diagram (simplified) of C.ai



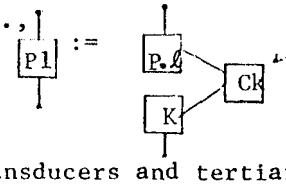
[1] Public, primary memory; $2^{23} \sim 2^{24}$ words; 74, 148, 222, or 296 b/w; 550 ns/w; MOS
[2] Central; cross-point switch; Mp-Pc|Ms dialogues; 16x16; width: 74 bits
[3] Central, trunk switch; inter P dialogues; ≥ 2 trunks
[4] Central, trunk switch; P-K(Mp.port) dialogues; ≥ 2 trunks
} identical protocols
[5] K(Mp.port) Relocation, protection, error correction, error detection; see[9]
[6] C(Operating-System) - with local primary and secondary memory
[7] Ms(secondary memory; drum; 1.4 gbits)



[8] X(special i/o interfaces; e.g., TV)
[9]



[10] E.g.,



[11] Transducers and tertiary memory - managed by a Ck (e.g., terabit memory)

Figure 2. PMS diagram of C.ai

Table 1. Comparison of C.ai with Other Computers

	Mp.width (b/w)	Mp.size (mwords)	Mp.size (mbits)	Mp.i-rate (mwords/sec)	Mp.i-rate (mbits/sec)	Ms.i-rate (mbits)	Pc.i-rate (mop/s)	Cost($m); Bitxops/sec; Bitxops/sec/$
PDP-10	36+1	0.26	9.7	4	144	9	0.4	1; 18; 18
Stanford AI-10	144+4						4.0	1; 180; 180
Model 91	64+8	0.52	37	21	1370	10x(1~2)	6.0	7.7; 500; 65
CDC 6600	60	.26	15.4	32	1920	12x10 (i/o) 600 (ECS)	3.0	5.5; 145; 27
C.ai	74~296	8.3	620	29~120	2150~ 8600	50x5	(4~12)x10 to (4~12)x20	13; 1440~4320[3] 110~330 16; 2880~8640; 180~540
C.ai/4	74~296	2.3	155	8~30	504 ~ 2150	50x2	(4~12)x3 to (4~12)x6	
CDC STAR	512	.131	74	25	12,800	(50~100)x (1~5)	100	10; 3200; 320
ILLIAC IV[1]	64+	.131	8.2	64x4.1 261	16,800	1000	256	10; 8200; 820

[1]Specifications taken from early ILLIAC IV paper by Barnes, et al.
[2]L. Roberts - Data Processing Technology Forecast, April 23, 1969.
[3]Assumes $8m for memory, $2m for peripherals and $300K per processor. Adjusting the memory size to that of STAR, yields $7m (total); 1440 ~ 4320; 205 ~ 620 to $10m; 2880 ~ 8640; 288 ~ 864.

Table 2. C.ai Memory Characteristics

Device	function[3]	t.access (t.cycle)	i.rate mb/s	i-width (bits)	total i.rate (mb/s)	power (watts)	floor space ft^2	size module 10^6b	cost (k$)	cost/bit ¢/b	controller costs (k$)
photomemory	t							≥ $1x10^6$			
moving-head disk (e.g., Calcomp)	t	0~55 + 20 ms	3.3[2]	1x3	3.3x3 10	30 kw (15 kw/ unit)	10x20 = 200	200 x20 = 4000	300	0.0075[2]	300
drum (e.g., GI)	s	8 ms	3.3	8~16/ drum	50 drum	20 kw (1 kw/ unit)	10x20 =200	70x20 =1400	900	0.048	5x40 200
shift register (AMS)	s	131 μs 780 μs	1	296	296	200 kw		1400	1800	1.25 0.7	5x40 200
core (Ampex)	p	.7 μs (1.8 μs)	.55	296 x16	163 2605	200 kw		620	10,240	1.65	500
Lockheed core	p	275 nsec (650 ns)	1.54	296 x16	455 7286	200 kw		620	8070	1.3	500
MOS AMS	p	500 nsec (1.2 μsec)	.83	296 x16	246 3947	200 kw		620	5760	.93	500
MOS (AMS)	p	250 ns (400 ns)	2.5	296 x16	740 8600	200 kw (.2 mw/ bit)		620	9500[1]	1.52	500
Bipolar (Cogar)	i	40 ns (80 ns)	10	296	2960			100 ~ 2000 words	36	6	
Bipolar ROM	i	40 ns (80 ns)	10	50~296	2960			100 ~ 2000 words	12	1 ~ 2	
									11,049		1000

[1] Estimate - Cogar has quoted such a system at $13,100K for delivery in 1973.

[2] These assume current densities. We can safely assume double density, hence lower cost, more storage, and higher transfer rates.

[3] t/tertiary; s/secondary; p/primary; and i/internal processor (program control, accumulators, etc.)

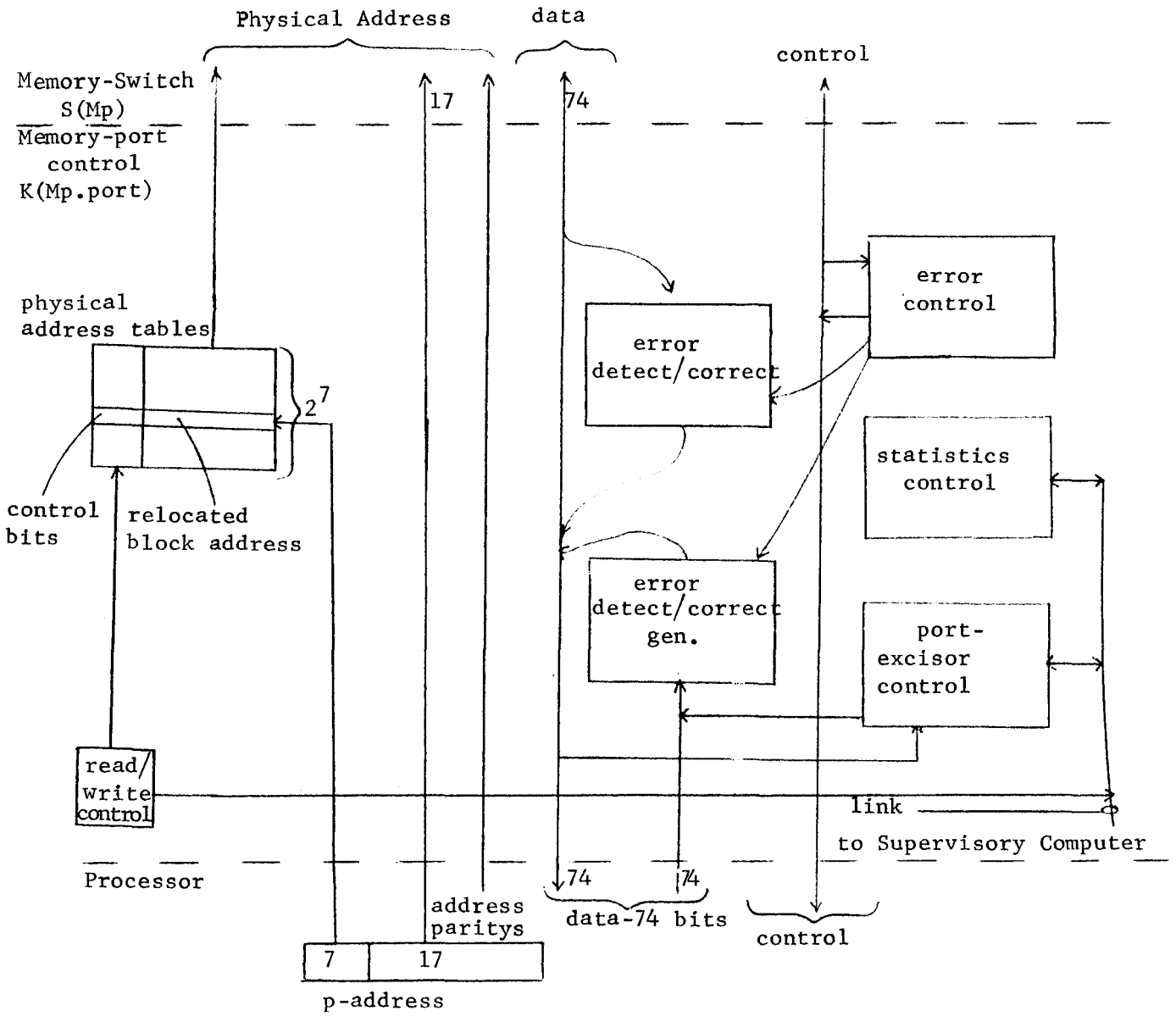
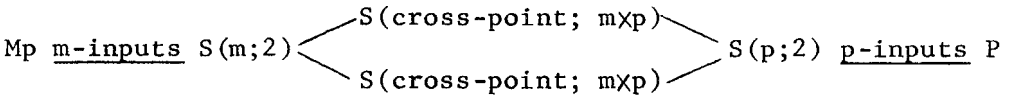


Figure 3. K(Mp.port) Memory-port relocation, error-detection, error-correction and control.

the 8,192k word memory (74 bits) would cost approximately $9,500,000. This is a projected price, but it is possible that memory prices will decrease even more than projected. AMS is the lowest bidder, but there is some question as to whether they could manufacture a 620 megabit memory in the time frame desired in addition to their current commitments. Cogar has quoted a price of $13,100,000 for a similar system for delivery in 1973. The memory prices are quotes for memory systems with TTL-compatible interfaces. The power dissipation for the AMS system is 180 milliwatts per 1024 bits or approximately 100kw for proposed system; support circuitry will double this to 200kw.

The memory switch will utilize MSI (medium scale integration) logic whenever possible. The data switch is currently envisioned as utilizing the Texas Instrument SN74150N 16 bit multiplexor with a typical data propagation time of 10 nsec (assuming the data select lines have been settled for ≥ 30 nsec). By the time the switch is constructed, faster circuits, such as Schotky TTL, will probably be available.

Since the switch is so central, a dual cross-point probably should be used. This is essentially a compound switch consisting of two cross-point switches and a (m+p) x S(1-input, 2-output) switch as shown:



The memory switch itself should not exceed $200,000 in cost (parts and labor). A quick calculation shows that in just the data and address switching logic, 2800 SN74150N 16 input multiplexors would be needed at

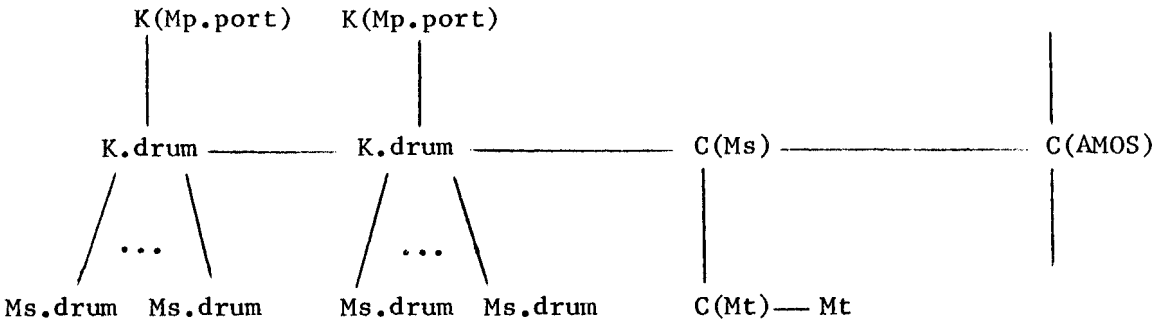


Figure 4. Eventual secondary-tertiary memory structure.

TERTIARY MEMORY

Any storage of programs and data outside the local C.ai environment will tend to put a fairly heavy transmission load on the current, 50,000 kb ARPA network. For example, we have assumed average program sizes of 250,000 74-bit words or 18.5 million bits. The transmission of a program this size on the present ARPA network would require approximately 460 seconds. If a session lasted an hour, about 15 minutes of the hour would be spent in file transmission. If 4 such users were on the system, then the whole ARPA network would be swamped.

Clearly, some on-site permanent storage must be provided. Programs can reside on a tertiary memory until they are brought into either primary or secondary memory for more rapid access.

Exact specification of C.amos will require further refinement of C.ai. It appears, however, that a PDP-11 Model 45 size machine will be sufficient if enough other minis are used for Ms, Mt and network control.

CONSOLE

Scopes would be used to display the overall allocation of resources to tasks, and the status of the computer. Several scopes might also be employed for human intervention required in the management of the computer.

Each of the processors would occasionally require a certain amount of console activity which would be done by small computers as described in the section on the language processors. Due to the nature of the switching involved in the individual processors, it might be possible to use only one small computer to serve several large processors.

2. The functions provided by AMOS must be a minimal set consistent with managing the hardware resources of C.ai. Complicated systems take a long time to build and are more open to problems.

3. The "users" of AMOS are the operating systems for each special-purpose P.ℓ. Thus the total operating system is a two-stage object: an overall operating system (AMOS) plus distinct operating systems on each processor. In most cases a human user and/or his program see only one of the individual systems, not AMOS.

4. Specification and construction of the operating system for a P.ℓ is up to the group responsible for that processor.

5. AMOS should usurp as few design perogatives as possible. That is, it should influence only minimally the design of operating systems and programs on individual P.ℓ's. Further, it should not greatly influence the design of C.ai as a whole in order that it will be possible in the future to replace AMOS with a completely different operating system.*

6. It should be possible to build very simple operating systems on the P.ℓ's if desired. They should not have to worry about physical level i/o and their communications with AMOS should be as simple as possible.

7. AMOS should be simple. This not only aids construction time and effort but also understanding (an essential point to insure correct operation and usage).

FUNCTIONS TO BE PROVIDED

It is easiest to specify what AMOS is to do by listing the major functions it is to provide. Elaborations of these functions will be

---

* C.ai is clearly a unique opportunity for implementing radically new virtual machines that exploit its parallel and functionally specialized parts. It is hoped that the computer will be available for research along this line. The understanding of such a machine, how to break up a load computationally, the characteristics of the programs run on it, etc. is so meager at present that initially the only sensible way to use it is as a collection of independent systems that happen to share some physical resources. AMOS and its hardware should not unduly impede research on more advanced modes of usage, however.

- the potential for processes on separate processors to communicate and share resources;

- a large on-site Ms for temporary use and a very large Mt for permanent storage of information.

What an Operating System on a P.ℓ Sees

- a device for allocating and overseeing sharing of Mp, Ms, and Mt;

- a device that will move files between Mp, Ms and Mt upon request;

- a device to handle the mechanics of transmitting and receiving information over the ARPA network.

What AMOS sees

- P.ℓ's competing for Mp, Ms and Mt;

- requests to share resources between processors;

- logical communication channels between P.ℓ's and the ARPA network to establish and maintain;

- files to be created and moved;

- M's to housekeep;

- accounting information to be logged and displayed.

STRUCTURES PROVIDING THE REQUIRED FUNCTIONS OF AMOS

Different structures can be chosen to provide the functions of AMOS. Those we have selected below seem to be sufficient for the task and consistent with our design objectives.

Mp Allocation

The opaqueness of how Mp allocations to P.ℓ's are being used and their size (64K wds) implies using an extremely simple algorithm. A P.ℓ will send a request to AMOS over the bus to allocate or deallocate

The request can be made with a priority, thus allowing swapping or paging information to be handled just like any other file only with higher priority for performing the transfer. Likewise, files can be transferred from Ms or Mt to Mp. Alternatively, information can be sent and received over the network directly to a file (see below). Files can be erased by request.

Note that the P.ℓ's specify where they want their files to reside. This seems essential since only they will know what they are being used for. Pricing structure, time limits, and allocation limits can be used to insure proper migration.

It is assumed that Ms and Mt provide hardware detection of record and file ends so that transfers of partial files may be made. On the other hand, record transfer may impose too much additional complexity on AMOS.

Communication Over the Network

AMOS will know nothing about users. It will have only logical channels that it can connect between a P.ℓ and some entity transmitting messages to C.ai over the network.

AMOS may receive messages on the network from entities for which it has no logical channel set up requesting access to a given P.ℓ. The P.ℓ may have told AMOS that it will take all comers, only certain ones, or that it wants to be informed of all requests for connection so that it can make a dynamic decision. If the requestor cannot be attached, he will be so informed.

Initialization

C.amos will have an autoload button that will load its local memory from a startup disk with a program to initialize Mp bounds registers and load its main Mp from Ms. Its bootstrap will also be able to retrieve from its local Ms various debug, checkout, and recovery routines.

C.amos will be able to startup any of the other P.ℓ's by a signal over the bus. Once started, however, AMOS has no control over the P.ℓ. This means that AMOS will have available the operating system (or a proper bootstrap) for each P.ℓ. In some cases this may include loading a micro-code store.

File Movement

All file operations are logical (not physical) as described above. AMOS will have one or more minicomputers that will initiate transfers between memory hierarchies and perform housekeeping chores.

Resource Sharing

The mechanism for sharing is basically the same for all resources. Processor A (the owner of a resource) tells AMOS over the bus that processor B may have a given type of access to that resource. If later, B requests that access, it will be granted (unless A has rescinded the access rights).

In the case of Mp this is implemented by setting bounds registers. For files AMOS must keep lists of processors (not processes) that can access given files. It is then up to the processors to control the access of their individual processes.

(a) Pc-Mp-Ms type of structure;

(b) reduce its own data and keep current system information available for AMOS;

(c) write to its own slow Ms for later display and analysis.

Some examples of information of interest are: (a) K(Mp.port) errors could be counted, and transmitted to C.amos, (b) the number of memory references could be counted and waiting times tabulated, (c) a central clock may be provided which all P's may access. A central timing facility might also be included at the clock. In order to keep the traffic low, a facility such as the clock might broadcast the time so that each processor could maintain its own timers (which would undoubtedly be in software).

AMOS should have a number of software monitors built into its modules. Such hooks are best when implemented in parallel with the operating system. Selected information would be either written by AMOS or read from registers by C.pm. If AMOS is to be a resource allocator, some P.ℓ information may be required. Information of this type would place certain constraints on P.ℓ implementors, but the sharing of common resources requires some standardization.

Each P.ℓ should also include its own hardware and software measurement devices with which AMOS can communicate. A mixture of software and independent hardware monitors integrated into the design of C.ai will allow for future study of this new structure, and encourage growth based on a knowledge of actual performance and utilization.

twice as wide (74 points), and must accept up to four words transmitted serially. We also proposed that this switch structure be duplicated centrally so that 1/2 can fail or be worked on without bringing the system down.

Memory would be added slowly, as the processing power is increased. An initial configuration could be operating in only one year, composed of the nucleus of the memory (including the secondary memory) together with some small amount of processing power (two PDP-10 processors, running Tenex). This configuration would provide immediate benefit to some users by having a larger memory. Even more memory could be added if needed.

By proceeding slowly the operation of the computer could also evolve and the tie with the network could be made gradually. During the second year of operation the first and second Stanford AI processors could be integrated to operate with the memory system and the first two processors. The necessity for two processors is based on the fact that the facility should be able to supply a constant resource to its users and, thus, two processors would be necessary to meet the reliability requirement. At the end of the second year almost any configuration of processors and computers would be possible.

CONTINGENCIES AND RESEARCH

The center section of Schedule 1 deals with the scheduling of specialized processors for languages. Their progress would not influence the main schedule line to any great extent, but could provide backup for the extra computing power. If specialized processors prove as valuable as

we believe they will, then such an approach would pay off and possibly negate the need for the large processors. Two of the contingencies (KI10 and a small microprogram-based PDP-10) could no doubt be contracted to DEC because they would be part of their standard product line.

NEW PROCESSOR ALTERNATIVES

The final line on Schedule 1 shows what we might expect (optimistically) from a new processor based design. In three years we might have such a system operating ready for software debugging (assuming that the design and fabrication had gone on in parallel with the hardware development). In possibly as early as four years, users might be on the system. (For example, by comparison, the PDP-11 is now about $2\frac{1}{2}$ to $4\frac{1}{2}$ years old (depending on when you count the start).

The problem with such a design appears to be that it isn't clear why any company would want to build it. The only company which has both the resources and the diverse product line strategy is Honeywell. A GE645-based processor has a large address space and possibly might be considered.

There is some evidence that a straightforward machine based on the PDP-11 structure, but with large addresses would be useful and could provide better encoding efficiency than a PDP-10. It might be much easier to build than the Stanford AI Processor, but if it were not part of a main product line it would be difficult to get it built. Buying computers which are part of another product line allows both a much lower cost and higher reliability (by not having a one of a kind).

data structure with large pointers. While all of these methods may be regarded as somewhat ad hoc, they should be sufficient. Thus, if we compare the modified version with some of the alternative off-the-shelf processors, making these modifications will probably make the PDP-10 no worse than the alternatives. Some of the desirable modifications will deal with list structure manipulation and typed variables. All in all, it appears from the early explorations about modifications between R. Greenblatt (M.I.T.) and the Stanford project, that they can be made without unduly distorting the Stanford design.

ECONOMIES OF SCALE

We have not carried out calculations which would allow one to decide whether it would be worthwhile making a system smaller than the one proposed. The considerations for not scaling down the computer to 1/4 the size would be:

1. Fixed operating costs of the laboratory facility (overhead), engineers, programmers, cooling, lab supplies, space.

2. The design task is fixed almost independent of machine size. Thus, it would be desirable to get as much power as possible in a single place. This would give us the highest performance/ total cost.

3. Desirability of having a central facility with a very large amount of processing power attached to a single, large memory.

4. Desirability of having a single, very large memory. This would permit more users through better user-load-averaging. Also, larger programs could be run.

5. There may be some quantity discount for drums, disks and memory.

6. Higher reliability. All facilities would be duplicated in a larger computer.

CONCLUSIONS

We have given an overall argument as to the feasibility and desirability of building a computer to be used in ai research. It is basically a very conservative approach built on current computers together with what we think can be done easily with the technology. We have not assumed very high performance processors. For example, one processor we think feasible is the Stanford AI version of the PDP-10. Such a processor should not require more than two years to develop, and should be replicatable for in the neighborhood of $100,000.

Given our overall numbers, we think it is worth trying to specify the next level of detail based on our evolutionary approach. We believe that an approach that departs from a conventional structure (e.g., by placing specific interpretation on addressing) will both decrease the performance and also make the memory too special purpose, thereby eliminating unspecified future use that might be made of that large facility.

We think the real point of the structure is that it should be simple, yet provide as much potential power (bandwidth) as possible, and should not be very presumptious about how particular future processors would use the facility. In particular, we strongly believe that additional power will be gained by developing special purpose processors to be used on C.ai and that this course should be thoroughly explored.

Allow all contractors to engage in certain kinds of improvements.

High reliability -- probably affects more users (say 50 ~ 100 versus 10 ~ 25 at a site).

Multiple users simultaneously.

6. This is a machine for artificial intelligence research.

   Consequences: Variety of specialized devices connected to it (e.g., robotic equipment); powerful arithmetic processing; large memory.

7. This is a symbol and list processing machine.

   Consequences: Access to very wide words so that different types of encodings are possible for various languages. The (memory size)/(memory accessed) ratio is probably large.

8. Examine all techniques (especially those which have been found to work in hardware and software):

   Examples: cache (1 or more);

   microprogramming;

   hash coding;

   highly specialized processors.

9. Design should be highly likely to materialize.

   Consequences: must be understandable;

   maximum amount of project parallelism;

   minimum amount of interaction among parts (hence well defined interfaces).

10. Base system performance on easy-to-identify techniques.

    Consequences: parallelism in
    word length,
    processors,
    memories;

APPENDIX 2

OUTLINE OF P.L* PROCESSOR DESIGN

The following is an abstract of the report "C.ai (P.L.*) -- An L* Processor for C.ai" by D. McCracken and G. Robertson. It is available from the Carnegie-Mellon Computer Science Department or the Defense Documentation Center.

The results of a preliminary design study for a specialized language processor (P.ℓ) for L* are presented. The objective of the study is to give an example of a specialized processor for C.ai.

The L* processor is to run 20-30 simultaneous L* users with very large address spaces at a speed improvement of better than 10 times a typical PDP-10 L* system. Its cost should be low relative to the memory resources of C.ai.

The design presented is that of an L* central processor (Pc.L*) with a very low-level instruction set (about the level of typical micro-code). Pc.L* is time-shared by a mini-computer that sits to the side, so that each L* user sees himself as running on a base L* processor. User contexts are switched by swapping processor status information in Pc.L*.

Each L* user has complete access to the central processor status through his address space. His machine code (microcode) can appear anywhere in the large address space, but executes out of a fast cache memory. It thus runs at microcode speeds. L* programs and data being interpreted by the machine code are accessed explicitly from a second

APPENDIX 3

OUTLINE OF P.LISP PROCESSOR DESIGN

The following is an abstract of the report "C.ai (P.LISP) -- A LISP processor for C.ai" by M. Barbacci, H. Goldberg, and M. Knudsen. It is available from the Carnegie-Mellon Computer Science Department or the Defense Documentation Center.

A special processor designed for efficient LISP processing is described. The processor is micro-programmable and makes heavy use of a fast scratchpad memory with several special purpose registers (small function units) and general byte manipulation capabilities. The approach taken has been to avoid unorthodox schemes of implementation and employs little in the way of unusually new (and untried) hardware.

Such a conservative approach will enable a fairly good implementation in a reasonable time. One of the places where efficiency in list processing (and in most programming applications) can be enhanced is in the ratio of instruction fetches/data fetches. To that end two things that are not usually available were required: writeable (up-datable) microcode and recursive control of microcode. With them, it is possible to implement the language interpreter as close as possible to the real hardware machine. Such a machine could also be a "shell" language processor. However, this was not a goal of the design, but rather a by-product.

The microprogrammed processes include a storage-compacting garbage-collector, which can be made to operate incrementally in parallel with user-program execution. This option avoids interruptions in LISP execution for garbage collection.

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)

Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pa. 15213

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

C.ai: A COMPUTING ENVIRONMENT FOR AI RESEARCH

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

Scientific Interim

5. AUTHOR(S) (First name, middle initial, last name)

G. Bell, P. Freeman, M. Barbacci, S. Bhatia, W. Broadley, R. Chen, L. Erman, H. Goldberg, W. Huen, M. Knudsen, D. McCracken, A. Newell, R. Reddy, S. Rege, G. Robertson.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May, 1971	55	none

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F44620-70-C-0107	
b. PROJECT NO A0827-5	
c. 61101D	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH, OTHER	Air Force Office of Scientific Research (SRMA) 1400 Wilson Blvd. Arlington, Va. 22209

13. ABSTRACT

A computer for artificial intelligence research is examined. The design is based on a large, straightforward primary memory facility (about 8 million 74 bit words). Access to the memory is via at least 16 ports which are hardware protected; there is dynamic assignmentof the memory to the ports. The maximum port bandwidth is 8,600 million bits/sec. Processors for languages (e.g., LISP) and specialized terminals (e.g., video input/output) can be reliably connected to the system during its operation. The approach is evolutionary in that high performance processors, such as the Stanford AI Processor, can be connected to the memory structure, giving an overall power of at least 100 times a PDP-10 (and 200 to 300 times a PDP-10 for list processing languages) for 10 processors -- although 20 processors can be attached. Using this approach we might expect 40 - 80 million operations/second.

At the same time, special language processors (P.$\mathcal{L}$) can be designed and attached. These processors give even larger power increases, but for restricted language use. Two processors, P.LISP and P.L*, were examined for the LISP and L* languages and are reported on separately.

A plan for building the machine in increments over the next three to five years is examined. Specific schedules are proposed.

Concurrent with the operation of the machine, there should be research into the design of hardware, software and theory of constructing large scale computing facilities with maximum modularity.

DD FORM 1 NOV 65 1473

Security Classification