MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 132

Proposed Research on Architectural Principles for Large Memory Systems

by

J. B. Dennis

(A proposal submitted to the Advanced Research Projects Agency, the Office of Naval Research)

October 1975

Table of Contents

Massachusetts Institute of Technology

Project MAC

Proposal for Research on

Architectural Principles for Large Memory Systems

Principal Investigator: Jack B. Dennis

Statement of Work

This research program concerns a new approach to the architecture of large, general purpose, memory systems capable of achieving concurrent processing of large numbers of memory transactions. The systems studied will be organized as interconnections of units that operate independently, and communicate by sending discrete packets of information to other units. These memory systems comprise several physical levels spanning a wide range of access time versus capacity, with built-in automatic redistribution of information among the physical levels ("percolation"); they are modular in structure within each level to provide extensibility, ease of maintenance, and the possibility of fault-tolerant performance.

The behavior of the memory systems studied will be specified by formal abstract memory models, and the structure of the memory systems will be chosen so they may be formally proved to implement the specified abstract memory model, on the basis of formal subunit specifications and the properties of their discipline of interaction.

The formal memory models used in this project will reflect the qualities essential to the modular construction of programs, the support of advanced high-level programming languages, and the management of large data bases.

The tasks of the research program are: (1) The design of appropriate formal memory models; (2) The development of organizational structures for memory systems that provably implement the formal models; (3) The analysis and estimation of their potential performance; (4) The study of technological requirements for promising architectural concepts, and (5) The verification of designs and evaluation of their potential performance through simulation and measurement using a network of microprocessors.

The evaluation of the performance of packet memory systems will be carried out using illustrative computations of interest to the Department of Defense in such areas as searching of large data bases, associative retrieval of information, transaction processing, and signal processing.

B. Packet Memory Systems

We propose to study and evaluate the design of packet memory systems, which differ radically from conventional computer memory systems. A Packet Memory System is made up of many physical units or modules, of a few distinct types, that operate independently. An important characteristic is that information is transmitted between units in fixed-size packets, and each unit is designed so it never has to wait for a response to a packet it has transmitted while other transactions are ready for its attention. In this way, the effectiveness of a unit is only indirectly dependent on the time for information to be processed by other units in the system, and a high level of concurrent processing of memory transactions should be achievable.

The behavior of a packet memory system can be specified in terms of a formal memory system model, and the formal memory model may be designed to satisfy criteria for support of modular programming and to provide direct support for useful data abstractions. It will be feasible to prove that a packet memory system correctly implements its specification expressed as a formal memory model.

Because they support a high level of concurrent activity, and can realize a formal memory model well suited to advanced language and data base applications, computer systems designed around packet memory systems will achieve far greater performance and be much easier to program than computers using conventional memory structures.

A computer system designed around a packet memory system requires instruction processing units capable of generating many concurrent and independent store and retrieval requests, if the maximum potential of a packet memory system is to be realized. Processors using data-flow program representations, such as those described in other publications of the Project MAC Computation Structures Group [4, 5, 10, 11, 12, 13], have this property.

In the following section we discuss a specific example of a formal memory model, and the behavior and structure of a Packet Memory System that implements the memory model. Other memory models and memory system structures will also be investigated in our search for the most attractive structures for achieving high performance and supporting important concepts of program and data base structure.

- Let U be a finite set of unique identifiers. (U contains one element for each physical site in M that can hold a distinct item.)

- The collection I contains items of two kinds: An item

$$v = \langle \underline{\text{elem}}, i, e, r \rangle$$

where	$i \in U$	unique identifier
	$e \in E$	elementary value
	$r \in N$	reference count

is an <u>elementary item</u>. An item

$$p = \langle \underline{\text{pair}}, i, j, k, r \rangle$$

where	$i, j, k \in U$	unique identifiers
	$r \in N$	reference count

is a <u>pair item</u> and represents a pair whose component values are uniquely identified by j and k. In both kinds of items i is the unique identifier of the item.

- The collection T contains the unique identifiers not in use.

The state of the Packet Memory System is the pair ⟨I, T⟩. Initially the state is ⟨∅, U⟩, that is, the memory holds no items and every unique identifier is free.

State transitions of M are associated with acceptance of command packets by M:

1. <u>Store transaction</u>. A <u>store command packet</u>

$$\langle \underline{\text{store}}, i \rangle$$

is accepted at the command port (<u>cmd</u>) and a <u>store packet</u> of one of the forms

$$\begin{cases} \langle \underline{\text{elem}}, i, e \rangle \\ \langle \underline{\text{pair}}, i, j, k \rangle \end{cases}$$

is accepted at the store port (<u>store</u>). One of the items

$$\begin{cases} \langle \underline{\text{elem}}, i, e, 1 \rangle \\ \langle \underline{\text{pair}}, i, j, k, 1 \rangle \end{cases}$$

(with reference count equal to 1) is added to M's collection of items. T does not change.

Modular Structure

Figure 2 shows how a Packet Memory System may be organized as a set of smaller packet memory system modules. Let $F:U \rightarrow [1, \ldots, n]$ be a function that maps each unique identifier $i \in U$ into an integer r between 1 and n. A module M_r holds each item having a unique identifier i with $F(i) = r$. Command packets and store packets are distributed among the modules $M_1, \ldots, M_n$ by the Distribution Network, and retrieval packets, ref count packets and unid packets are collected into common streams by the Arbitration Network. The Distribution Network and the Arbitration Network can themselves be built out of simpler units, as we have discussed in published work [4, 5]. The Distribution Network, for example, may be a tree of switching modules and buffer modules arranged to route packets to exit channels according to the property F(i) of the unique identifier i in the packet.

Hierarchy

The principle of hierarchical organization of a Packet Memory System is shown in Figure 3. Two levels M_H and M_L are shown although the concept extends naturally to more levels. The higher level M_H acts as a cache memory for items held in the lower level M_L. The particular scheme illustrated works as follows: Every item held by the memory system is represented by an item in M_L which contains its reference count. The most active items are held in M_H without reference counts. Unique identifiers are identified as free by M_L and delivered as unid packets. Store packets arriving at the memory system are sent to both M_H and M_L. Command packets are presented to M_H. If a retrieval command refers to an item in M_H, a retrieval packet is produced and the command packet vanishes. Otherwise, the retrieval command packet is sent on to M_L. A store command packet instructs M_H that a store packet for a designated unid will arrive to be held by M_H; M_H disards some less active item if necessary to make room for the new item. Once the store packet arrives and is stored in M_H, the store command packet is forwarded to M_L. Up and down command packets are simply transmitted through M_H to M_L. In addition, whenever an item is discarded from M_H, a command packet of the form ⟨<u>done</u> i⟩ must be sent to inform M_L that the unique identifier i may be released at the unid port for reuse. Command packets are transmitted to M_L in the same sequence as they arrived at M_H. In particular, the order of storing and retrieving any item must be maintained. Note that if M_H itself has modular structure, it is sufficient to maintain command sequencing independently for each module of M_H.

In this hierarchical memory scheme, store packets are always sent to both M_H and M_L. In consequence, items displaced from M_H may be discarded -- transmission to M_L is not necessary. Furthermore, reference counts can be maintained at the lowest memory level only. An alternative scheme is that store packets are sent only to M_H, and items are sent to M_L only as they are displaced from M_H. In this case reference counts would have to be maintained in both M_H and M_L.

For a more complete picture of the kinds of memory system structure to be studied, let us consider briefly the structure of memory modules that might be used (in large numbers) to construct levels M_H and M_L in Figure 3. In the case of M_H, each module M_r functions as the usual cache or buffer memory for each command referring to an item whose unique identifier i satisfies $F(i) = r$. The module M_r contains a limited number of storage locations or <u>cells</u> which can hold stored items. For each retrieval command packet routed to M_r, its unique identifier is tested to determine if the corresponding item is held in some cell of M_r. A store command packet causes the accompanying item to be placed in a vacant cell of M_r, or to displace some relatively inactive item from an occupied cell. In a full-scale realization of a packet memory system, each module could be fabricated on a single LSI chip, and perhaps hundreds of these modules would make up M_H in a large computer system.

Each module of M_L might include one track or a group of tracks of a magnetic disc, or a set of long shift registers fabricated in CCD or magnetic bubble technology, where unique identifiers correspond to specific positions within the circulating stream of information. The logic of each module would be fabricated in a LSI technology and would include a small associative memory so that many items could be simultaneous objectives for retrieval from the recirculating stream.

D. Potential Advantages of Packet Memory Systems

The potential advantages of packet memory systems relative to contemporary memory systems seem enormous to us. The structure of these systems is sufficiently straightforward and modular that it will be feasible to formally establish their correctness of operation and freedom from deadlock -- a possibility not yet contemplated for systems as complex and *ad hoc* as conventionally organized memory hierarchies for general purpose computers. These systems will support universal information structuring concepts that meet requirements for modular programming -- requirements that no existing computer system satisfies, especially for large programs. Furthermore, packet memory systems will be able to achieve a level of concurrency in their operation at least one hundred times greater than current large scale computer memory systems. Note also that in a computer using a packet memory system, the instruction processing units are not involved in the redistribution of information among memory levels -- in contrast to essentially all existing systems, in which a large fraction of processing unit capacity is devoted to memory management.

E. Proposed Research Program

The proposed research program will include activities in four areas: formal memory models; memory system structure; technology; and evaluation.

Formal Memory Models

The relation of formal memory models to the overall architecture of computer systems for advanced programming languages and data base applications will be studied. Emphasis will be on resolving issues concerning concurrency and shared data in a manner consistent with modern views of language design and requirements for efficient memory system structure.

Memory System Structure

A variety of memory system structures will be studied using formal memory models as the specification of their behavior. Methods of proving that the memory systems implement the formal models will be developed and applied. The potential performance of the systems studied will be analyzed and estimated.

Technology

Different choices of technology for realization of components of packet memory systems will be evaluated and the technological requirements for realizing the performance potential of packet memory systems will be determined.

Evaluation

The most promising concepts of memory system architecture will be validated and their performance potential evaluated through simulation using the facility described below.

The evaluation will be performed by determining the throughput of packet memory systems in support of specific illustrative computational tasks of interest to the Department of Defense:

1. Files Searching: The problem of searching a large text file for occurrences of a specified pattern or key word.
2. Associative Retrieval: The problem of performing simple retrieval requests on a large relational data base.
3. Transaction Processing: The problem of handling large numbers of access and update transactions for a large shared data base.
4. Signal Processing: Specific signal processing computations such as the fast Fourier transform, and sound (e.g. speech) synthesis.

F. Prototype Evaluation Facility

We plan to verify the correctness of our designs and to evaluate their performance potential through simulation of packet memory systems. For this purpose we propose to augment an existing ARPA funded facility to support evaluation of architectural concepts using networks of microprocessors and packet buffer modules. In using this facility to evaluate a particular system organization, each microprocessor will simulate one unit or a group of units of the system being evaluated, and buffer modules will hold first in, first out queues of packets waiting for processing. Each microprocessor will communicate with other microprocessors by placing packets in their input buffers.

Such a facility will allow validation and evaluation of a wide variety of architectural concepts, and will permit a particular architecture to be modelled in varying degrees of detail. A very attractive possibility is to program the microprocessors so that operation of a full scale system is modelled with precise scaling in time, for then an accurate extrapolation of performance to a full scale version of the system may be made.

We believe the microprocessor approach to design validation and performance prediction is preferable to programmed simulation on a conventional computer in terms of cost, accuracy of modelling, and speed of simulation.

G. Recent Relevant Research

The Computation Structures Group of Project MAC has developed concepts of data flow computer architecture that promise highly parallel execution of programs expressed in well structured source programming languages. The proposed research is an extension and application of these concepts to the problem of very large memory system architecture. The relevance of our recent work is indicated below:

1. J. B. Dennis and D. P. Misunas. A computer architecture for highly parallel signal processing. Proceedings of the 1974 ACM National Conference, Association for Computing Machinery, New York, November 1974 [5].

   This is our original and most primitive form of data flow computer. The idea of using an Arbitration Network and a Distribution Network for efficient Processor/Memory communication is explained here.

2. J. B. Dennis and D. P. Misunas. A preliminary architecture for a basic data flow processor. Proceedings of the Second Annual Symposium on Computer Architecture [4].

   This paper extends our primitive data flow architecture to programs containing conditional and iteration constructs. Also, our first detailed consideration of a hierarchical memory is presented. Current work [10, 11, 12] is directed at extending this architecture to handle procedures and data structures.

3. J. B. Dennis. First version of a data flow procedure language. Symposium on Programming, Paris, 1974 [3].

   This paper presents a data flow language which serves as a goal for our architecture studies. In expressive power, this language encompasses languages such as Algol 60, Pure Lisp, and APL.

4. J. B. Dennis. On storage management for advanced programming languages [6].

   This paper presents a conventional programming language designed so its translation into efficient data flow form is straightforward. It is shown that a memory system based on acyclic directed graphs provides a satisfactory storage model for the language. This supports our proposal to base memory system architecture on an acyclic graph model.

5. J. E. Rumbaugh. Ph.D thesis in preparation [13].

   This thesis presents an alternative computer architecture using the data flow concept and a language similar to the language in [3]. It is shown that the proposed machine correctly implements the language and is free of deadlock under several reasonable assumptions (for example, that the machine has enough storage to carry out the requested tasks). On the basis of this work, we believe we can construct similar proofs for packet memory systems.

6. D. Isaman. Ph.D thesis in preparation.

   This work is a study of memory system models comparing the power and efficiency of alternatives: In particular, the restriction to acyclic graphs, and the use of cells having changeable contents, are studied.

7. I. Hawryszkiewycz. Semantics of Data Base Systems. Project MAC Technical Report TR-112, December 1973.

   This work shows how a relational data base system model founded on Codd's work can be formally implemented in terms of a base language whose objects are acyclic directed graphs. This work supports our position that a memory model using acyclic graphs is adequate for expressing relational data bases.

Other Related Work at Project MAC

The proposed research program has strong ties with four other research efforts at Project MAC.

1. Data Flow Computer Architecture

   This research concerns the development of computer architecture based on data flow representations of programs and is supported by a National Science Foundation grant. The prototype evaluation facility described above will be partially supported by this grant, and the facility will be used for both studies of data flow architecture and memory system architecture.

6. Dennis, J. B. On Storage Management for Advanced Programming Languages. Computation Structures Group Memo 109-1, Project MAC, M.I.T., Cambridge, Mass., November 1974.

7. Henderson, D. A., Jr. The Binding Model: A Semantic Base for Modular Programming Systems. Technical Report TR-145, Project MAC, M.I.T., Cambridge, Mass., February 1975.

8. Liskov, B. H., and S. N. Zilles. Programming With Abstract Data Types. Computation Structures Group Memo 99, Project MAC, M.I.T., Cambridge, Mass., March 1974.

9. McCarthy, J., et al. The LISP 1.5 Programmers' Manual. M.I.T. Press, Cambridge, Mass., 1962.

10. Misunas, D. P. Structure Implementation in a Data-Flow Architecture. Paper submitted for presentation at the 1975 Sagamore Conference.

11. Misunas, D. P. Performance Analysis of a Data-Flow Computer Architecture. Paper submitted for presentation at the 1975 Sagamore Conference.

12. Misunas, D. P. Procedure Representation in a Data-Flow Processor. Paper submitted for presentation at the 1975 Sagamore Conference.

13. Rumbaugh, J. E. A Parallel Asynchronous Computer Architecture for Data Flow Programs. Ph.D Thesis, Department of Electrical Engineering and Computer Science, M.I.T., in preparation.