MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Machine Structures Group
Memo No. 2

Memorandum MAC-M-143
March 6, 1964

RESEARCH PROGRAM IN MACHINE STRUCTURE

J. B. Dennis

The machine structures group is concerned with developing an adequate theoretical basis for the design, analysis, and evaluation of advanced computer structures that can meet the needs of Project MAC. The concept of a computer utility depends on two fundamental points:

1. The power of a large computer is a better match to the union of many tasks than to any particular one.

2. There is a wealth of knowledge- procedures and data - of interest to many users.

A successful computer utility must, therefore, possess certain properties:

1. Users must be able to create, test, correct and run arbitrary procedures through appropriate use of their terminals.

2. It must be impossible for one program to interfere with the correct execution of another.

3. The users must have access to a large file of public information.

4. The capacity of the system must be readily and appropriately divisible among the users.

A multi-access computer contains five major structural elements:

1) main directly addressable memory -- contains data and programs at a time of program execution.
2) processing units -- interpret programs as sequences of actions on data.
3) auxiliary storage -- contains programs and data that are active but not in execution.
4) file memory -- provides long term storage of program and data files.
5) terminal devices -- provide communication between the system's environment and active procedures within the machine.

While there can never be a precise definition of machine capacity, because it has many aspects and different users make widely differing uses of them, we may identify the more important aspects as:

a) amount of main memory;
b) amount of processing capacity;
c) rate of interaction with environment;
d) the channel capacity for information transmission to and from auxiliary storage;
e) amount of file memory.

The demands for these aspects of capacity will vary from program to program, and also will fluctuate with time for any single program; these variations will occur at least as fast as the interactions of the programs with their input-output environments, that is on the time scale of human reactions. Thus it is essential that these aspects of machine capacity be promptly and effectively allocatable among many users.

Multi-Processor Systems A large memory is necessary in a multi-access system for two reasons. First of all, there are some individual programs that could easily take advantage of a great deal more memory. Secondly, it is clearly desirable to keep in active memory many tasks at a time. If this can be done, there will generally be many tasks in the main memory in various states: some ready for execution, some ready for responses from input/output units, some waiting for access to secondary storage, and so on. Thus the inclusion in the system of a number of processors should permit the parallel execution of many programs. A multi-processor system is particularly attractive for several reasons.

1) Improved balance of processing capacity in relation to main memory is realized.

A single processing unit cannot make effective use of a large main memory except for certain classes of procedures, and for these, the average demand is likely to be considerably less than peak demand.

2) Larger absolute computation capacity can be attained.

It appears that the creation of faster computing structures in the future will depend more on the use of parallelism than on the use of faster components.

3) Better utilization of memory through the possibility of several processors executing the same procedure in parallel.

A procedure may be represented only once in main memory, yet be interpreted simultaneously by several processing units.

4) Less switching of processors among tasks is required.

More processing tasks can run to completion without interruption, thus reducing executive overhead.

The foregoing factors are concerned with better utilization of machine capacity. Other reasons for interest in multi-processor systems relate to operational characteristics.

5) The system offers reliability.

The modular structure of a multi-processor system ensures that a fault in any one component is unlikely to disable the system as a whole.

6) The system is adaptable to changing requirements. Modules of processing capacity or memory can be added or deleted to adapt a multi-processor system to changing needs.

7) Processors may be assigned to individual users if desired. In some applications -- for example, real-time computation in connection with an on-line experiment -- a user must have access to the full capability of a processing unit on very short notice. A processor may be assigned to such a user for the duration of his experiment, or may be made available for his immediate preemption as required.

A clear limitation on a multi-processor system is placed by the switching arrangement used to connect processing units with memory modules. If all possible direct paths are included in hardware, systems with more than several dozen processing units become impractical. Machine structures that avoid this limitation are under study.

The design of a multi-access computer system suitable to the objectives of Project MAC raises a number of questions which are at present not adequately answered. These questions may be divided into three broad classes:

1) How can a computing structure be organized so that its resources are dynamically re-assignable among a large number of tasks?

The evolution of the interrupt feature has made the processing unit of a computer readily re-assignable. With regard to assignment of memory capacity and input/output devices, however, little progress has been made from the earliest machines.

2) What are the appropriate policies for governing the assignment of machine capacity to insure its effective utilization? By what mechanism are policies determined, and by what techniques should they be implemented?

Present allocation policies for multi-programmed systems are not keyed to the time scale of human reactions, and deal with small numbers of program and data objects in main memory.

3) How can machines be organized to improve their programmability?

Certain features of modern algebraic compilers are cumbersome and wasteful to implement in existing machine structures. Examples are push-down storage for nested and recursively applied procedures, organization of program and data overlays, and retrieving file sectors from mass memory.

Theoretical Studies For the realization of important advances in the structure of multi-access computers, theoretical studies are sorely needed in two areas. The foremost is the development of models of program structure that are valid for the study of machine structure problems. Several elements of such a theory are already available: One element is the concept of push-down storage allocation for nested procedures as implied by the Algol programming language. Another is the view of a program as a collection of procedure and data objects called segments. References within a segment are by the usual linear addressing technique. References among segments are by segment name. Segments are considered as the allocatable entities with regard to memory assignment.

The second area of theoretical study is the development of an adequate model for consideration of the allocation and scheduling problem in multi-programmed computer systems. Most previous work in this area has assumed knowledge of memory requirements and running times prior to the scheduling process. This assumption is totally unjustified in a multiple-access computer system: Not only is it unreasonable to expect users to supply the assumed data, but the requirements will fluctuate in frequently unpredictable ways during a typical computation. A theory of scheduling is needed that assumes ready reassignability of system resources and takes advantage of the large number of tasks that will be simultaneously active in the system. The theory must define principles for establishing scheduling policies based on averages of the aspects of user demand. The statistical law of large numbers should be applied so that the details of task execution do not enter into policy formulation.

There are two areas in which contemporary means of representing procedures (e.g., Algol, LISP) give no clue to the machine designers of an elegant hardware solution. One is the parallel execution (by several processing units) of program sequences that operate on a common data object. The other is the interaction of procedures with outside environment (the human users and external systems in communication with the machine). These problems need to be attacked from the level of source language through the level of hardware to achieve elegant solutions consistent with the principles of mult-programming, and the view of a machine as an allocatable computation structure. Any complete model of program structure for purposes of machine design must acknowledge these problems.

Machine structures and design features are required that permit fast redistribution of the aspects of machine capacity among active tasks. As an example a memory structure has been proposed[1] that allows main memory to be readily reallocated among the various segments of users' programs. Extensions of this philosophy to other situations are being investigated.'

As components become progressively more reliable and as greater demands are made on the performance of computers, machine designs have become more complex and this trend will undoubtedly continue as techniques for producing complex logical structures are continually improved. One can readily foresee the need for new tools for the logical designer to cope with the problem of evolving the organization of these systems. A language for describing machine structures is needed in which the designer may specify a logical design, test it for consistency, simulate its operation, and readily modify the description to include changes and additions.

---

1. J. B. Dennis, A Machine Structure for Dymanic Storage Allocation, Memorandum MAC-M-137, January 31, 1964.