(8)

# PMS: A NOTATION TO DESCRIBE COMPUTER STRUCTURES

C. G. Bell, M. Knudsen and D. Siewiorek
Carnegie-Mellon University
Pittsburgh, Pa.

## ABSTRACT

PMS (for Processors, Memories and Switches) is a notation that was initially used in a textbook (Bell and Newell, 1971) to describe the physical structure of parts of computers, computers, and computer networks. The use of PMS components appears to be a natural way to describe information processing systems. PMS has been used: pedagogically; to focus analysis on certain structures; at lower system levels to describe register transfer and switching circuits structures; and as the basis of a computer implemented data base and analysis language.

## INTRODUCTION

Although the PMS notation was developed to describe the physical structure of computer systems for a textbook (Bell and Newell, 1971), it has subsequently evolved to a basic set of components for information processing. PMS was developed to provide a systematic treatment of physical structures, since many functional names are often used to describe the same object (e.g., memory and storage). Also many objects may have the same functional name. For example, within a single organization the name Multiplexor might be applied to a time-multiplexed processor, a communications link, a switch used to connect multiple processors to a single memory, or a combinational circuit with multiple inputs and a single output. It was also our aim to give a uniform meaning and definition to components (e.g., processors) to facilitate better communication among professionals.

There are seven basic component types, each distinguished by the kinds of operations it performs:

Memory, M. A component that holds or stores information (i.e., i-units) over time. Its operations are reading i-units out of the memory, and writing i-units into the memory. Each memory that holds more than a single i-unit has associated with it an addressing system by means of which particular i-units can be designated or selected. A memory can also be considered as a switch to a number of submemories. The i-units are not changed in any way by being stored in a memory.

Link, L. A component that transfers information (i.e., i-units) from one place to another in a computer system. It has fixed terminals. The operation is that of transmitting an i-unit (or a sequence of them) from the component at one terminal to the component at the other. Again, except for the change in spatial position, there is no change of any sort in the i-units.

Control, K. A component that evokes the operations of other components in the system. All other components are taken to consist of a set of discrete operations, each of which -- when evoked -- accomplishes some discrete transformation of state. With the exception of a processor, P, all other components are essentially passive and require some other active agent (a K) to set them into small episodes of activity.

Switch, S. A component that constructs a link between other components. Each switch has associated with it a set of possible links, and its operations consist of setting some of these links and breaking others.

Transducer, T. A component that changes the i-unit used to encode a given meaning (i.e., a given referent). The change may involve the medium used to encode the basic bits (e.g., voltage levels to magnetic flux, or voltage levels to holes in a paper card) or it may involve the structure of the i-unit (e.g., bit-serial to bit-parallel). Note that T's are meaning preserving, but not necessarily information preserving (in number of bits), since the encodings of the (invariant) meaning need not be equally optimal.

Data-operation, D. A component that produces i-units with new meanings. It is this component that accomplishes all the data operations, e.g., arithmetic, logic, shifting, etc.

Processor, P. A component that is capable of interpreting a program in order to execute a sequence of operations. It consists of a set of operations of the types already mentioned -- M, L, K, S, T and D -- plus the control necessary to obtain instructions from a memory and interpret them as operations to be carried out.

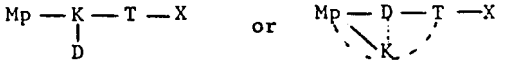
## COMPUTER MODEL (IN PMS)

Components of the seven types can be connected to make stored program digital computers, abbreviated by C. For instance, the classical configuration for a computer is:

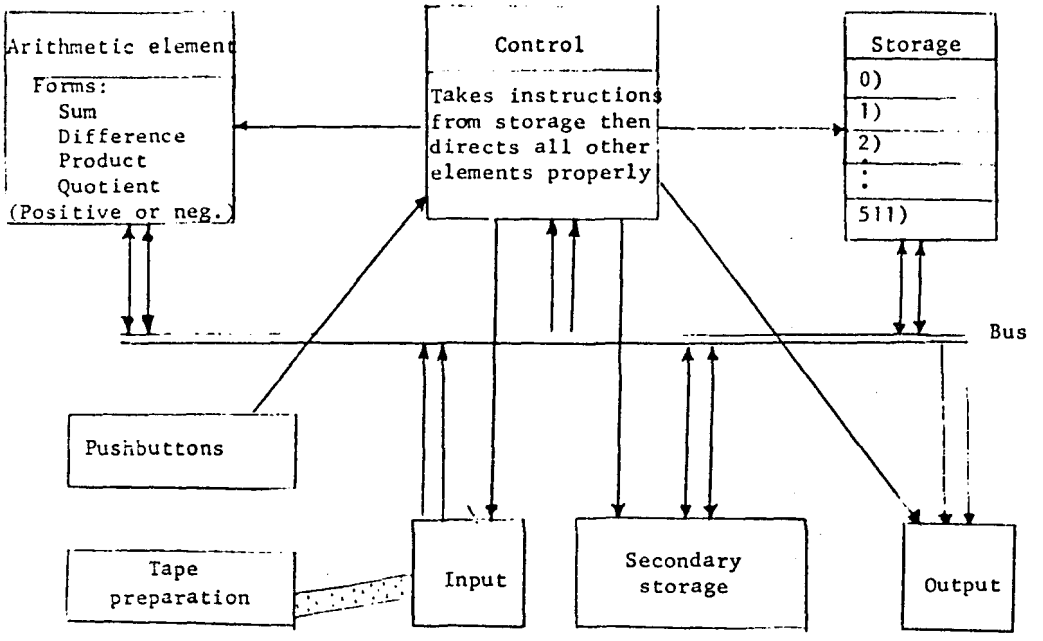
C := Mp——Pc——T——X

Here Pc indicates a central processor and Mp a primary memory, namely, one which is directly accessible from a P and holds the program for it. T (input/output device) is a transducer connected to the external environment, represented by S. (The colon-equals (:=) indicates that C is the name of what follows to the right.)

The classic diagram had four components, since it decomposed the Pc into a control and an arithmetic unit:

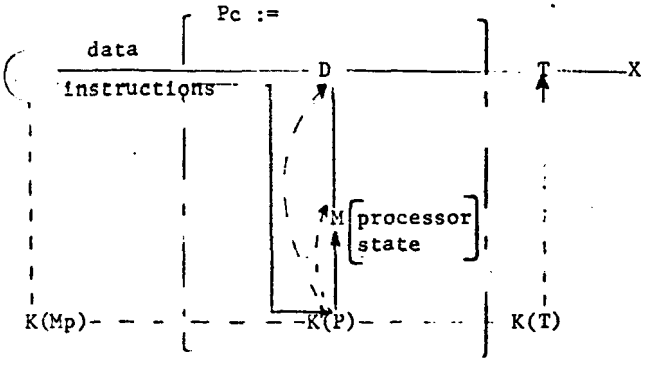



where the heavy information carrying lines are for instructions and their data, and the dotted lines signify control. Diagrams such as these correspond roughly to the conventional, simplified block diagrams of computers. The following one from the MIT Whirlwind computer is one of the earliest. Later we will diagram Whirlwind in PMS notation.




Often logic operations were lumped with control, instead of with data operations -- but this no longer seems to be the appropriate way to functionally decompose the system.

Now we associate local control of each component with the appropriate component to get:



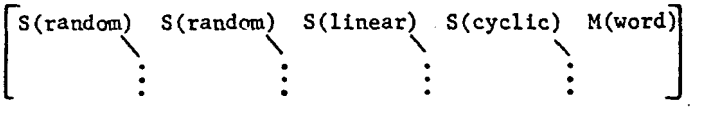

where the heavy lines carry the information in which we are interested, and the dotted lines carry information about when to evoke operations on the respective components. The heavy information carrying lines between K and Mp are instructions. Now, suppressing the K's, then lumping the processor state memory, the data operators, and the control of the data, operators and processor state memory to form a central processor, we again get:

Mp — Pc — T — X

Computer systems can be described in PMS at varying levels of detail. For instance, we did not write in the links (L's) as separate components. These would be of interest only if the delays in transmission were significant to the discussion at hand. Similarly, often the encoding of information unimportant; then there is no reason to show the . The same statement holds for K's. Sometimes one wants to show the locus of control, say when there is one control for many components, as in a tape controller. But often this is not of interest.

Components are themselves decomposable into other components. Memories are composed of a switch (the addressing switch) and a number of submemories. Thus a memory is recursively defined as either a memory or a switch to other memories. The decomposition stops with the unit-memory, which is one that stores only a single i-unit, hence requires no addressing. Likewise, a switch is often composed of a cascade of 1-way to n-way switches. For example, the switch that addresses a word on a multiple-headed disk might look like:

M.disk :=




The first S(random) selects a specific Ms.disk drive unit; the second S(random) is a switch with random addressing that selects the head (the platter and side); S(linear) is a switch with linear accessing that selects the track; and S(cyclic) is a switch with cyclic addressing that finally selects the M(word) along the circular recurring track. Note that the switches are realized by differing technologies and thus have varying performance.

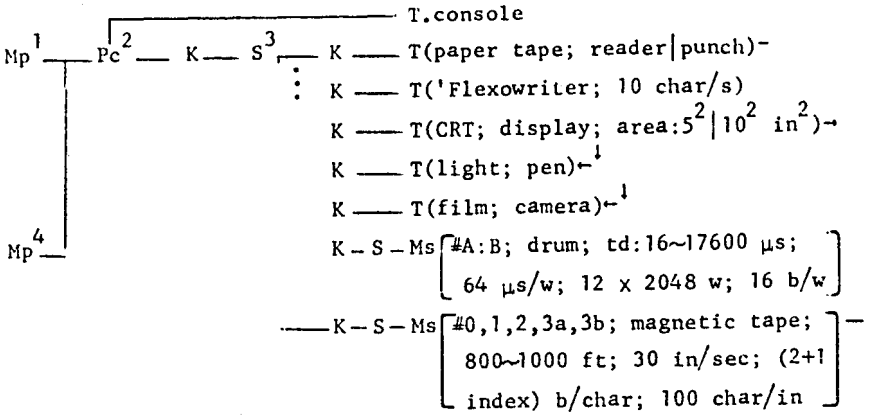
Various notational conventions designate specifications for a component, e.g., Mp for a functional classification, and S(cyclic) for a type of switch access function in the case of rotating memory devices like drums. There are many other additional specifications one wants to give -- a single general way of providing additional specifications is used so that if X is a component, we can write:

$$X(a_1:v_1;\ a_2:v_2;\ \ldots.)$$

to indicate that X is further specified by attribute $a_1$ having value $v_1$, attribute $a_2$ having value $v_2$, etc. Each parameter (as we call the pair a:v) is

well defined independently of what other parameters are given; hence, there is no significance to the order in which they are written, or the number which have to be written.

The following PMS diagrams describe actual computer systems at varying degrees of detail. The Whirlwind computer is represented as:




[1] M(toggle switch; 8 μs/w; 32 w; 16 b/w)

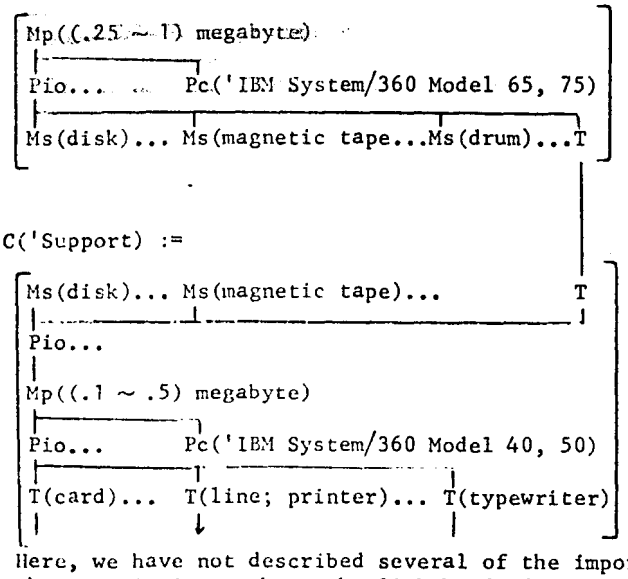
[2] Pc(50 kop/s; 16 b/w; 1 instruction/w; 1 address/instruction; M.processor state (3 w); technology; vacuum tube; 1948 ~ 1966)

[3] S(fixed; from: Pc; to: 8 K; concurrency: 1)

[4] Mp(#0:1; core; 8 μs/w; 1024w; 16 b/w; taccess: 2 μs)

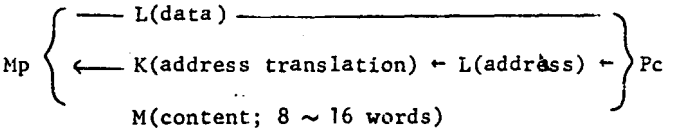
Note that most of the important attribute:value characteristics about the machine are given. In addition, it might be noted that the machine has only limited processor/input-output concurrency due to the switching structure.

At a somewhat higher level, PMS is useful for describing the structure and the interaction of various larger components in a computer network. The IBM two computer, ASP system can be represented as:




Here, we have not described several of the important characteristics such as the link bandwidth, and various characteristics of the interconnected computers although we could have.

Similarly, lower level features can be shown as in the mapping structure between a processor and its memory:




Again, a diagram might also include the information rate and width of the links, and the access time of the memory.

PMS can also be used at the register transfer level. In fact, we have used PMS to name and describe a system of register transfer modules called RTMs (Bell, Grason and Newell, 1972). In this system, a wiring diagram for a digital system to compute the sum-of-integers to N is shown in Figure 1.

At a much lower level of detail, the PMS names adequately and clearly describe the structures of switching circuits. Most combinational circuits correspond to data-operations D's or switches S's, and combinational circuit design consists of making more complex D's and S's. Sequential circuits take small amounts of memory M, and proceed to build controls K's. At a higher level more complex sequentially controlled P's are formed from D's and K's. Finally, the special name of P's and C's are used for particular structures.

## USES OF PMS

Although we have no specific data, the fact that PMS provides a small number of information processing primitives to learn is indeed a comfort to both the novice and professional alike. Namely, information processing is presented in an orderly, easy to understand manner. There is no barrage of

what appear to be newly invented devices, which turn out to be mere inventions in marketing names and numbers. PMS can thus be used for brochures and catalogues to provide concise descriptions of commercially available computers, and to specify a buyer's requirements in a call for bids from suppliers.

Pedagogically, the PMS notation can be used to develop an understanding of the overall behavior of computing systems. For example, several computing systems can be compared using PMS notation. The PMS representation can be designed so that only a few key parameters are presented to the student, thus requiring a step-by-step analysis to assess the capabilities of each structure.

The fact that there is a well-defined level has a positive effect on the formulation of problems for analysis. PMS can be used to analyze various aspects of computer systems, such as: distributing of I/O devices and controllers for increased I/O rate and/or throughput; tweaking of existing structures to increase throughput; examining I/O priority schemes; predicting system performance with the addition of specialized hardware (such as a cache memory, associative memory, etc.); and modeling system reliability. We are in the process of compiling a set of problems for these structures which have been developed at various universities.

There has also been research which carries out analysis at the PMS level, including creation of programs to mechanize the analyses. Strecker (1970) developed a closed form analysis for performance of multiprocessor systems.

Bhatia (1972) has studied a model for time shared multi-processor computer systems where each processor has an associated cache memory. An APL program was developed which allows a designer to interactively balance a system design. A detailed study of cache memories was also made to determine cost-performance as a function of cache parameters.

Other theses in progress are directed toward further development of models and analytic tools for combining PMS components to form processors, computers and networks. At one extreme, this work assumes that information flow is continuous, but with stochastic behavior, and thus can be modeled using both steady state and signal flow techniques. In more difficult cases, continuous systems simulators are used. At the other extreme, it is necessary to know the behavior (load) of the system, and to model it discretely.

The main effort to make PMS more than a notation scheme has been carried out by Knudsen, in his Ph.D. research (Knudsen, 1972). PMS was implemented as a man-machine language for building up data files of computer descriptions. Several subprograms using these descriptions tally characteristics such as cost. Two analytical routines carry out reliability estimation and graphical displays of multiprocessor performance and efficiency as functions of any two independent variables, based on the Strecker formulation. Future work will include: comparison of computers, checking for correct configurations, pinpointing of bottlenecks, etc.

## CONCLUSIONS

We have been working with the PMS notation for several years, and have found that it more than satisfies our initial pedagogical and descriptive requirements. Subsequently we and others have used and extended it to formulate and solve other problems at all levels of digital systems.

## REFERENCES


1. Bell, C. G. and A. Newell, Computer Structures: Readings and Examples, McGraw-Hill, 1971.

2. Bell, C. G., J. Grason and A. Newell, Register Transfer Design: Computers and Digital Systems Using the PDP-16, forthcoming from Digital Press, late summer 1972.

3. Bhatia, S., Design Techniques for Balancing Computer Systems, Technical Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa., 1972.

4. Knudsen, Michael, PMSL - An Interactive Language for High-Level Description and Analysis of Computer Systems, Technical Report, Computer Science Department, Carnegie-Mellon University, 1972.

5. Strecker, W. D., An Analysis of the Instruction Execution Rate in Certain Computer Structures, EE Dept., Carnegie-Mellon University, 1970.

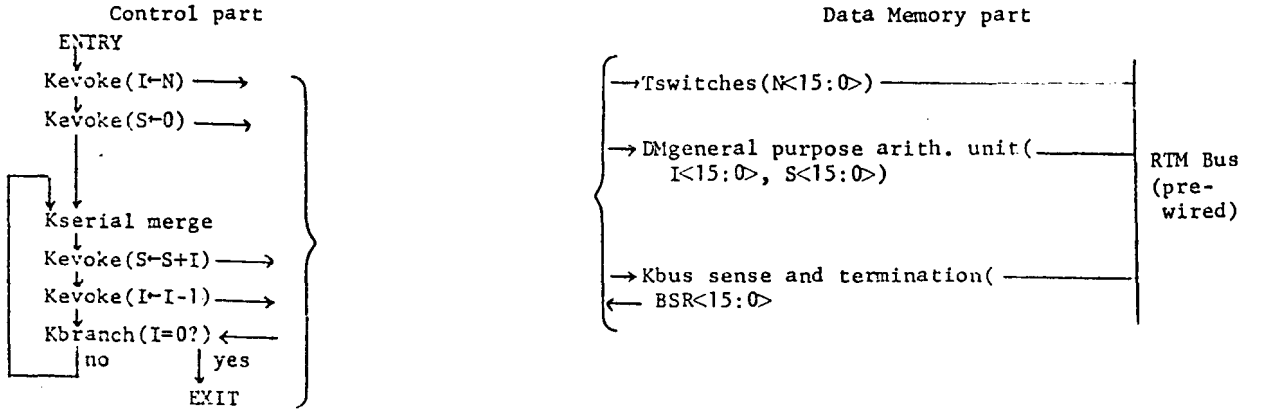




Figure 1. RTM diagram for S = sum of integers from 1 to N.