PDP-X Technical Memorandum # 16

Title: PDP-X System Architecture (Revised)

Author(s): R. Burkhardt
L. Seligman

Index Keys: Architecture
System
Design Decisions

Distribution
Key: On Request

Obsolete: Technical Memorandum # 6

Revision: None

Date: August 1, 1967

Group	Field	Number of bits	Description
A Bus, FM address (Read), Data Read (Cont.)	DAT	1	DATa - This bit indicates that memory data is to be selected. If the previous memory start referenced fast memory, the fast memory will be accessed and selected. This bit should be on whenever DBR, DBL, DATC, or SE is on.
	Total	8	

PDP-X System Architecture

Introduction

The system architecture* of PDP-X is described in this technical memorandum. Both design innovations of importance to the user and system concepts which lead to production economies are discussed.

This system architectural description, together with the more detailed descriptions found in associated technical memorandums, do not merely define a single processor; PDP-X lends itself to a number of implementations of varying complexity and relative costs. Since the architecture is constant across several possible models, many of the programs and peripherals are interchangeable. The first implementation of PDP-X falls in the cost/complexity range occupied of the PDP-8 implementations; future implementations are possible, with the smallest very much competitive with PDP-8/I and the largest filling in the current gap between the large and small computers now available from DEC.

PDP-X has grown out of experience with DEC's large and small computer lines, the design strives to achieve performance levels comparable to the largest machines through optional features added to a basically simple structure. In many ways PDP-X is a refinement of PDP-8, sharing the same basic word structure and basic instructions, but also including important design advances that will make it competitive with other products over the next several years. The appendix shows the evolution of basic DEC small computer instruction code structures.

---

*The term system architecture is intended to convey the concept of logical structure as opposed to its physical realization or implementation. The PDP-8 architecture, for example, has been re-implemented several times; the last, PDP-8/I, the fastest, least expensive, and smallest of the implementations, executes the same programs and runs the same peripherals as its predecessors. The lifetime of an architecture, with its system and diagnostic software, interface options, peripherals, customer acceptance, and training, is far longer than that of any implementation; thus, a company's profits in a good architectural design is far higher.

Introduction

The first four of the eight sections below discuss such a user-oriented feature as efficient use of core memory, efficiency achieved through innovation in instruction format and a multi-accumulator register structure. The final four sections discuss implementation concepts which, primarily through subassembly sharing techniques, lead to low overall system manufacturing costs.

Further details of the initial implementation must await experience gained from PDP-8/I, which, it is hoped, will expose IC problem areas and potential packaging traps. PDP-8/I will also expose the new cost relationships so that PDP-X design strategy can focus effectively on achieving minimum manufacturing cost.

Efficient Memory Utilization

One of the more obvious problems with current small-machine designs has been their inefficient use of core memory, a system resource whose cost has been rising relative to the cost of the processor logic. Indeed, a major cost-reduction technique used in PDP-8/S design was the development of a far less-expensive memory as is indicated by the cost breakdown in the appendix. PDP-X architecture makes more efficient use of available core memory through the introduction of a more powerful instruction set and an addressing structure that eliminates the (sometimes hidden) memory waste in sector addressing and single accumulator small-computer designs.

The great majority of instructions written for and executed by small computers, regardless of instruction set, fall into the PDP-8 repertoire. In addition, most others have certain little information, as they usually reference memory words close to index quantities, close to the program counter, or the lowest words in memory (sector 0). PDP-X allows compact (16-bit) representation of instructions where possible and permits complete (32-bit) representation when necessary. Hence, although such an instruction set has much of the potential power and scope of a 32-bit processor, only 16-bits are necessary for frequently used instructions. The ability to directly address all of memory easily, when necessary, simplifies the task facing the programmer by eliminating the need for complex linking structures as found, for example, in the PDP-9. In addition, the more casual customer or application programmer can generate working programs far more quickly. The appendix contains a more detailed analysis of the instruction formats.

The ultimate test of the efficiency of the structure lies in the programming package supplied with the hardware. Since no manufacturer of a PDP-9 class machine has been able to supply a version of FORTRAN for a 4K word-memory system, such a compiler would represent a major competitive advantage as well as testify to the accomplishment of a design goal.

Wide Range of Possible Processor Performance

The architecture is implementable in several processor models whose price and performance span the entire small computer market and include a model small enough to use as part of an IO device controller or selector channel. Smooth evolution and re-implementation should be possible over the next several years as the architecture leads to many new models, each particular model also exhibits a fairly wide range of performance depending on the number and type of internal options and peripherals purchased. To avert proliferation of software systems, however, certain standard configurations will be defined and the software written around them.

The major reasons for the wide range of possible processor performance are:

a. large, partially implemented, OP code set

b. variable number of interrupt levels with associated register sets

c. use of main core memory to replace hardware registers

d. facility for multi-way/bit-by-bit configurations without drastic alterations to basic processor

e. use of BOS to create dedicated IO or OP controllers.

Software and Hardware Integration

PDP-X systems will consist of many diverse configurations ranging from the small, dedicated data gathering system to the large real-time/multiuser installation; useful system software should be available to aid each user in fully realizing the potential power of his particular configuration.

Ease of use, rather than sophistication, is one of the major goals of the software system. Inhibiting numbers of conventions, command mnemonics, data formats, and calling sequences are to be avoided. Since most of the users will be relatively inexperienced, error detection and recovery is included in all of the major system such as the assembler and compiler. All IO data will be parity-checked, check-summed, or both recorded on the media.

The major system will be written in modular pieces with clean calling sequences; [illegible] versions will be available for the smaller configurations. Programs will not modify themselves; instructions and alterable data will be independently located. All input/output will be done through a common interface that is also accessible to the user. Refer to the appendix for more detailed information on required software components.

The importance of external and internal documentation cannot be overstressed; many customers will want to alter or modify portions of the software system. Software performance and maintenance will become as important in the next few years as it is today for hardware. Many of the problems of documentation, training, employee turnover, and unexpected results can be solved by concurrent design and documentation.

*Since, in very small systems, modularity and minimum core usage are often conflicting goals, it may be necessary to sacrifice some modularity in order to make the software available on such minimal systems.

Real-time and Multiuser Environment

Real-time usage has become an important factor in computer sales and applications; three very distinct usages stand out. The first is the dedicated on-line system where the cost of the device (e.g., a particle accelerator) interfaced to the computer far outweighs the latter's cost. PDP-X hardware provides a good order code, extremely fast interrupt response time, high speed core memory, and a fast IO system; in addition, the software package includes a set of highly optimized arithmetic and utility routines, re-entrant at some level, to cater this sort of real-time usage.

The second is used by an OEM who requires the smallest and simplest processor possible to imbed in his product. Such customers will use much of the same hardware used by DEC in dedicated IO controllers.

The third usage occurs where several real-time operations together occupy only a small fraction of the available processor time. Here the customer (DEC's traditional laboratory user) looks for a multiuser software system. The major hardware feature required for such a real-time, non-dedicated environment is rapid problem switching. This implies fast interrupt response, low overhead in switching users, and sufficient core memory to allow resident programs to handle the peak service demands. The problems of core usage have already been discussed. User change overhead has always been due to a combination of both software and hardware considerations. The intent of PDP-X design is to reduce problem switching (due to interrupt) to an absolute minimum by providing multiple sets of general registers and a 2 memory map. With the addition of this optional hardware, few interrupt cases will require that processor status be explicitly saved and restored.

A complete set of general registers, hardware on the larger models, is provided at each interrupt level. The cost of adding these registers is easily outweighed by the advantages of automatically saving and restoring the program status double-word, accumulators, and index registers. The dual map provides a separate set of mapping registers for the user and the real time or monitor program. Mapping rather than relocation, particularly for read, or protection bits per block has been selected since it leads to simplified system programming and better core usage. Additionally shared code, and as the most general of high level systems, is most likely to fit a customer's existing needs.

Structure changes to facilitate system implementation

In the next few years as large-scale integration (LSI) becomes more readily available, later implementations of the new architecture can effectively take advantage of the "standard" LSI devices now being developed. Current computer products could be constructed around "custom built" LSI but the economic advantages are only marginal, primarily because of the high development costs of such custom circuits.

Such standard LSI devices include:

a. Internal Scratch Pads. The multi-accumulator organization of the new architecture will allow yet machines to make efficient use of high-speed scratch pads. The larger model processors can also use these memory arrays as temporary storage when executing more complex instructions.

b. Read only storage (ROS). The new architecture has many unimplemented operation codes. As LSI ROS becomes available, its implementation could be added without significantly effecting cost. The ROS approach to the control allows elimination of much of the "random" logic that is so difficult to package and test. This approach also allows use of the same processor for diverse functions, such as a dedicated I/O device controller.

c. Uniform cell arrays, address decoders, and registers. As these elements LSI variations of current products become available, they may be readily incorporated into the processor.

Processor mobile concept

As advanced engineering/manufacturing methods shrink the cost of computer arithmetic processors, the cost of IO controllers grows relative to them. Today, one finds tape systems, displays, etc., almost as complex as the arithmetic processor and certainly more difficult to manufacture. To satisfy user demand for still more powerful IO command structures, including more flexible interfaces, higher bandwidth, and less main (arithmetic) processor interference, these controllers must grow even more complex.

The most common approach to increasing the IO controller capability while reducing system cost has been the imbedding of part of the controller in the arithmetic processor. The success of this approach has been limited by the amount of processor time stolen relative to the cost saved. Historically single controllers, relying heavily on main processor assistance, have been very unsuccessful for reasons of efficiency.

A new approach may be termed the processor mobile concept. Here, the specialized IO controllers are replaced by small general purpose processors dedicated to IO control. These are, in fact, the same processors found in the smallest model PDP-X. Much of the special purpose, normally hardware normally found in the controllers is replaced by appropriate software and DMC programming (additional, specialized instructions). Devices which, by their complexity, lend themselves to this form of implementation include:

IBM compatible Magnetic Tape
DECtape
Buffered Display
Multistation Teletype Control
Buffered Line Printer

It should be noted, however, that the intent at this time is not general purpose multiprocessing. These dedicated processors will be sold only as IO controllers where implementation just happens to include a modified arithmetic processor.

Since, however, most of the new software systems are designed to achieve some form of simultaneity of calculation or motion, it would appear that hardware explicitly designed to aid this multiprocessing is the natural extension of current design.

PMP-X architecture permits it to serve as a vehicle for software development of more general multiprocessor systems. Multiprocessors offer:

a. optimization over diverse problem mixes through dynamic restructuring

b. system size scales, by adding identical units, over an extremely wide performance range.

c. extreme reliability since redundancy merely lowers system capacity.

A typical system component interconnection is given in the appendix. The memory bus system has been designed to explicitly aid multiprocessing.

Modular Implementation

One of the most obvious, and perhaps expected, facts of digital system manufacture at DEC is that the labor cost and time of test rises as a square law rather than linearly with module count. Some statistics are presented in the appendix to justify these conclusions; although they are not as accurate as one might desire, the general trend is clear*. Independent subassembly construction and testing seems to be one effective method of minimizing system manufacturing cost and in-process construction time. Indeed, even if test were a small fraction of system cost, the (un)availability of properly skilled manpower strongly influences the production rate. As labor costs rise and component costs drop, the need for modular construction techniques increases. A PDP-X processor (memory, or major optic) is partitioned into a number of independent subassemblies which are small enough to be suitable for automated test equipment yet, which reduce the number and cost of interconnections. These subassemblies are considered repairable only at the subassembly construction level and, like most of todays modules, they are replaced, not repaired, by system test and maintenance personnel.

* It may be argued that sheer number of modules is unimportant, rather the complexity of the system is the significant factor. PDP-6 is not only larger but far more complex than PDP-8, a fact reflected in the module count. However, note a.) that both PDP-9 and PDP-7 fall on the same curve and b.) that the 338, a more highly-structured (logically) system, lies below the curve. Also consider the case where an assembly/test step has only a few components; an unskilled person can quickly perform single substitution tests to find the defective component. Such tests are not feasible in systems composed of many components.

Integrated Option Design

All of the central processor features required for extended arithmetic functions, large core memory, and real-time and multi-user operation are an integral part of the PDP-X architecture. These features are standard or optional depending on the model purchased. A general logic for peripheral control structure has also been formed; this structure eliminates many of the idiosyncrasies often found in input/output programming systems.

All connections between I/O device control units and processors are through the standard I/O bus. This bus is used for all processor models and configurations to reduce redundant peripheral development. DC interlocked control signals are used to insure reliable operation over extremely long distances while still permitting arbitrarily fast devices to operate physically close to the processor.

The I/O bus has all of the capability currently found on the PDP-9 class computer; it is used for all data transfers under program control, multiplexer channel control, and selector channel control. Communications between High- and Medium-speed transports and their associated control processors are also through the I/O bus.

Appendix II

PDP-8, PDP-8/S Cost Analysis

Detailed information is hard to obtain, but M. Mendrala has verified that the following relative estimates, scaled so that the absolute values are meaningless, accurately reflect true manufacturing costs:

	Total	Memory	Processor, etc.
PDP-8	15	7.3	7.7
PDP-8/S	10	2.5	7.5

Appendix III Instruction format analysis

The compressed instruction format used on PDP-X and other modern 16-bit computers is satisfactory to code the great bulk of program executed on these small computers since most instructions written fall into the load/store/branch group and require only short format addresses. There are, however, two significant cases when the compressed format is grossly inefficient. First, when the need arises to access arbitrary memory words which may be beyond the sector boundaries, complex linkages (tables in sector 0, indirect addressing through literals, etc.) are usually required. As shown in the figure on the following page, PDP-X permits a long form (32-bit instruction) which contains a full address for access to any word in the memory system without resort to base registers or other inconveniences.

Secondly, small machines have not been expandible since their instruction word could not afford the luxury of unused operation codes. The long form of each instruction provides the opportunity to augment the processor with some 256 additional operation codes, more than ever might be necessary. The basic instruction class that permits this expansion is trapped on the smaller machines, permitting programmed operators that preserve downward compatibility. All such extended instructions require long format; indeed, they make the machine seem like a small 32-bit processor. Instruction frequency data shows that these instructions are needed only infrequently

Appendix IV - Major Software Components

I. Input Output System.

In the small configurations, this should consist mainly of Teletype and Paper Tape Peripherals. In larger systems, other devices such as cards, magnetic tape and discs should be available. Since all I/O interfaces are well defined, the I/O System could be replaced with a real-time multi-user monitor.

II. Assembler.

The assembler should be structured around MACRO X for the PDP-10. The assembler should be written in such a fashion that individual option modules may be added to obtain more features or deleted to reduce core requirements. A good assembler is one of the most useful system software components supplied to the customer.

Standard features should be:

a. Relocatable output

b. [illegible] (by use of [illegible] mode)

c. Radix change (at local level)

d. Arithmetic operations

e. Text facilities

Optional features should be:

a. Full optimization of addresses and literals

b. Conditional assembly

c. Macros, [illegible]

III. Compiler

[illegible]

the PDP-10 compiler is required. The compiler, like the assembler, should be written so that pieces of it may be removed. Real-time capabilities for the object code are very important, but, as yet, these remain undefined.

IV. Loader

A linking loader capable of linking assembler or compiler-produced binary is required. The output formats for the assembler and compiler should be identical to, or at least subsets of, the basic relocatable format. The loader should be capable of performing library searches and conditional loads. As in II and III above, these features should be optional so that the loader can work on earlier configurations.

V. Editor

The Editor should be similar to [illegible]. The command format, however, should be altered so that commands such as:

PUNCH

READ

are valid, as well as their abbreviations [illegible]. The error detection and messages [illegible] should be thoroughly comprehensible.

VI. Peripheral Interchange.

The total number of different data formats should be held to a minimum so that many of the processing routines found in PDP-10 PIP will not be necessary. Full [illegible] character set compatibility should be maintained on all media.

VII. DDT

DDT should be written to run in either a single user environment [illegible]

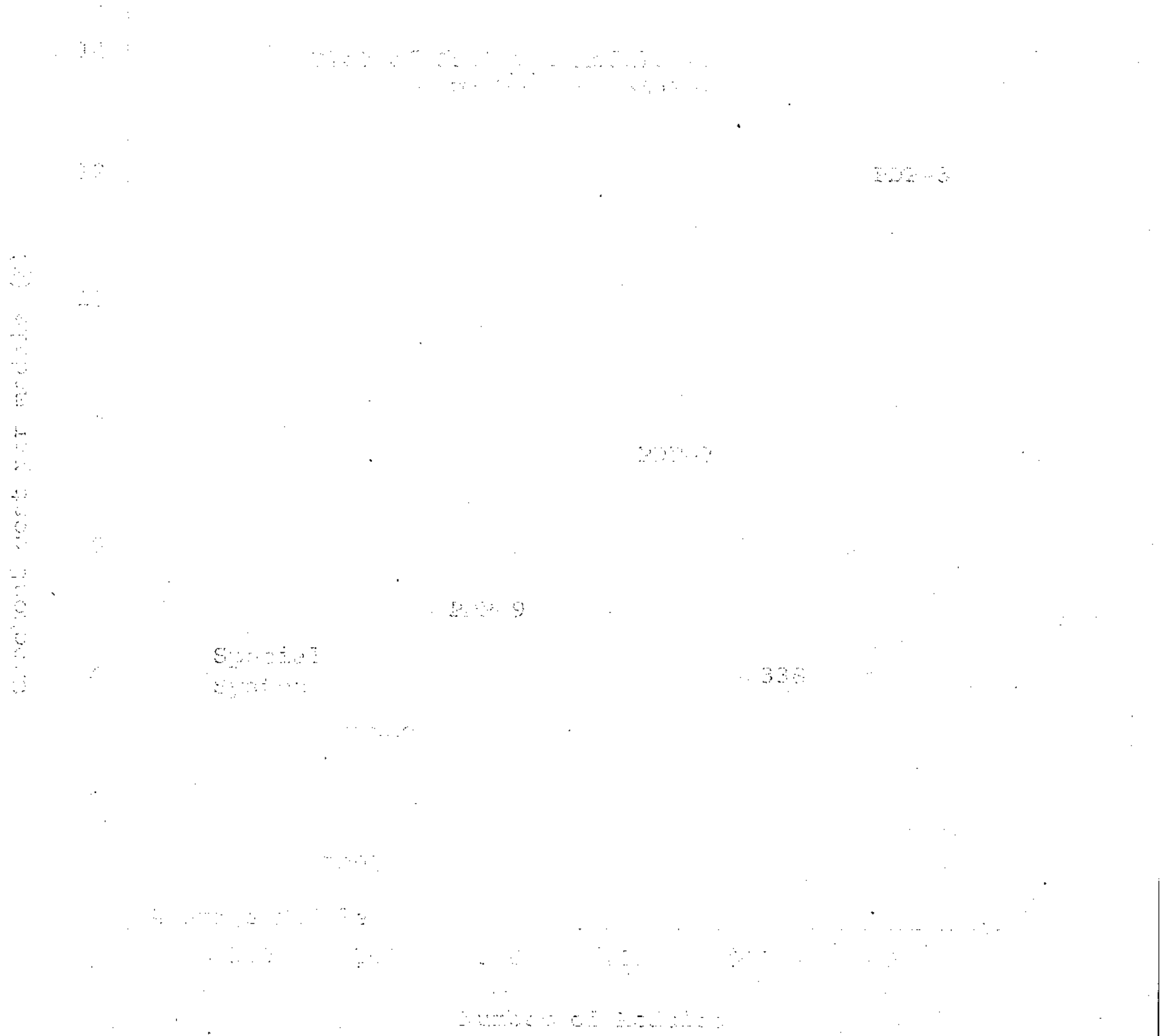
Frequency order/ranked PDP-9 System Code

	% within	Covered by PDP-X Slave	Covered by PDP-X Only	Covered by PDP-X Mon-itor
DSP	17.9	X		
CRT	16.4	X		
LPT	16.4	X		
LTC	15.3	X		
CRS	7.6	X		
TTY	5.0	X		
DC	4.3		X	
DIA	3.8	X		
SCP	3.6			X
PTR	3.5	X		
DTA	2.8	X		
PTP	2.4	X		
MON	.5			X
XXX	.3			-
BAD	0			X
CRA	0		X	

Appendix V

Checkout costs and logical complexity

Note that systems include only digital logic, the time and costs for [illegible], transponders, CRTs, etc., has been explicitly deleted.



Appendix VI