ABSTRACT

Gordon Bell, William D. Strecker
November 8, 1975

COMPUTER STRUCTURES: WHAT HAVE WE LEARNED FROM THE PDP-11?

Over the PDP-11'S six year life about 20,000 specimens have been built based on 10 species (models). Although range was a design goal, it was unquantified; the actual range has exceeded expectations (500:1 in memory size and system price). The range has stressed the basic mini(mal) computer architecture along all dimensions. The main PMS structure, i.e. the UNIBUS, has been adopted as a de facto standard of interconnection for many micro and minicomputer systems. The architectural experience gained in the design and use of the PDP-11 will be described in terms of its environment (initial goals and constraints, technology, and the organization that designs, builds and distributes the machine).

1.0 INTRODUCTION

Although one might think that computer architecture is the sole determinant of a machine, it is merely the focal point for a specification. A computer is a product of its total environment. Thus to fully understand the PDP-11, it is necessary to understand its environment.

Figure Org. shows the various groups (factors) affecting a computer. The lines indicate the primary flow of information for product functional behavior and for product specifications. The physical flow of goods is nearly along the same lines, but more direct: starting with applied technology (e.g., semiconductor manufacturers), going through computer manufacturing, and finally to the service personnel before being turned over to the final user.

The relevant parts, as they affect the design are:

1. The basic technology--it is important to understand the components that are available to build from, as they directly affect the resultant designs.

2. The development organization--what is the fundamental nature of the organization that makes it behave in a particular way? Where does it get inputs? How does it formulate and solve problems?

3. The rest of the DEC organization--this includes applications groups associated with market groups, sales, service and manufacturing.

4. The user, who receives the final output.

Note, that if we assume that a product is done sequentially, and each stage has a gestation time of about two years, it takes roughly eight years for an idea from basic research to finally appear at the user's site. Other organizations also affect the design: competitors (they establish a design level and determine the product life); and government(s) and standards.

There are an ever increasing number of groups who feel compelled to control all products bringing them all to a common norm: the government(s), testing groups such as Underwriters Laboratory, and the voluntary standards groups such as ANSI and CBEMA. Nearly all these groups affect the design in some way or another (e.g. by requiring time).

2.0 BACKGROUND

It is the nature of engineering projects to be goal oriented--the 11 is no exception, with much pressure on deliverable products. Hence, it is difficult to plan for a long and extensive lifetime. Nevertheless, the 11 evolved more rapidly and over a wider range than we expected, placing unusual stress on even a carefully planned system. The 11 family has evolved under market and implementation group pressure to build new machines. In this way the planning has been asynchronous and diffuse, with distributed development. A decentralized organization provides checks and balances since it is not all under a single control point, often at the expense of compatibility. Usually, the hardware has been designed, and the software is modified to provide compatibility.

Independent of the planning, the machine has been very successful in the marketplace, and with the systems programs written for it. In the paper (Bell et al, 1970) we are first concerned with market acceptance and use. Features carried to other designs are also a measure of how it contributes to computer structures and are of secondary importance.

The PDP-11 has been successful in the marketplace with over 20,000 computers in use (1970-1975). It is unclear how rigid a test (aside from the marketplace) we have given the design since a large and aggressive marketing and sales organization, coupled with software to cover architectural inconsistencies and omissions, can save almost any design. There was difficulty in teaching the machine to new users; this required a large sales effort. On the other hand, various machine and operating systems capabilities still are to be used.

2.1 GOALS AND CONSTRAINTS - 1970

The paper (Bell et al, 1970) described the design, beginning with weaknesses of minicomputers to remedy other goals and constraints. These will be described briefly in this section, to provide a framework, but most discussion of the individual aspects of the machine will be described later.

Weakness 1, that of limited address capability, was solved for its immediate future, but not with the finesse it might have been. Indeed, this has been a costly oversight in redundant development and sales.

There is only one mistake that can be made in a computer design that is difficult to recover from--not providing enough address bits for memory addressing and memory management. The PDP-11 followed the unbroken tradition of nearly every known computer. Of course, there is a fundamental rule of computer (and perhaps other) designs which helps to alleviate this problem: any well-designed machine can be evolved through at least one major change. It is extremely embarrassing that the 11 had to be evolved with memory management only two years after the paper was written outlining the goal of providing increased address space. All predecessor DEC designs have suffered the same problem, and only the PDP-10 evolved over a ten year period before a change was made to increase its address space. In retrospect, it is clear that since memory prices decline at 26% to 41% per year, and many users tend to buy constant dollar systems, then every two or three years another bit is required for the physical address space.

Weakness 2 of not enough registers was solved by providing eight 16-bit registers; subsequently six more 32-bit registers were added for floating point arithmetic. The number of registers has proven adequate. More registers would just increase the context switching time, and also perhaps the programming time by posing the allocation dilemma for a compiler or a programmer.

Lack of stacks (weakness 3) has been solved, uniquely, with the auto-increment/auto-decrement addressing mechanism. Stacks are used extensively in some languages, and generally by most programs.

Weakness 4, limited interrupts and slow context switching has been generally solved by the 11 UNIBUS vectors which direct interrupts when a request occurs from a given I/O device.

Byte handling (weakness 5) was provided by direct byte addressing.

Read-only memory (weakness 6) can be used directly without special programming since all procedures tend to be pure (and reentrant) and can be programmed to be recursive (or multiply reentrant). Read-only memories are used extensively for bootstrap loaders, debugging programs, and now provide normal console functions (in program) using a standard terminal.

Very elementary I/O processing (weakness 7) is partially provided by a better interrupt structure, but so far, I/O processors per se have not been implemented.

Weakness 8 suggested that we must have a family. Users would like to move about over a range of models.

High programming costs (weakness 9) should be addressed because users program in machine language. Here we have no data to suggest improvement. A reasonable comparison would be programming costs on an 11 versus other machines. We built more complex systems (e.g., operating systems, computers) with the 11 than with simpler structures (e.g. PDP-8 or 15). Also, some systems programming is done using higher level languages.

Another constraint was the word length had to be in multiples of eight bits. While this has been expensive within DEC because of our investment in 12, 18 and 36 bit systems, the net effect has probably been worthwhile. The notion of word length is quite meaningless in machines like the 11 and the IBM 360 because data-types are of varying lengths, and instructions tend to be in multiples of 16 bits. However, the addressing of memory for floating point is inconsistent.

Structural flexibility (modularity) was an important goal. This succeeded beyond expectations, and is discussed extensively in the part on PMS, in particular the UNIBUS section.

There was not an explicit goal of microprogrammed implementation. Since large read-only memories were not available at the time of the Model 20 implementation, microprogramming was not used. Unfortunately, all subsequent machines have been microprogrammed but with some additional difficulty and expense because the initial design had poorly allocated opcodes, but more important the condition codes behavior was over specified.

Understandability was also stated to be a goal, that seems to have been missed. The initial handbook was terse and as such the machine was only saleable to those who really understood computers. It is not clear what the distribution of first users was, but probably all had previous computing experience. A large number of machines were sold to extremely knowledgeable users in the universities and laboratories. The second handbook came out in 1972 and helped the learning problem somewhat, but it is still not clear whether a user with no previous computer experience can determine how to use a machine from the information in the handbooks. Fortunately, two computer science textbooks (Eckhouse, 75; and Stone and Siewiorek, 75) have been written based on the 11 to assist in the learning problem.

2.2 FEATURES THAT HAVE MIGRATED TO OTHER COMPUTERS AND OFFSPRINGS

A suggested test (Bell et al 1970) was the features that have migrated into competitive designs. We have not fully permitted this test because some basic features are patented; hence, non-DEC designers are reluctant to use various ideas. At least two organizations have made machines with similar bus and ISP structures (use of address modes, behavior of registers as program counter and stack); and a third organization has offered a plug-replacement system for sale.

The UNIBUS structure has been accepted by many designers as the PMS structure. This interconnection scheme is especially used in microprocessor designs. Also, as part of the UNIBUS design, the notion of mapping I/O data and/or control registers into the memory address space has been used often in the microprocessor designs since it eliminates instructions in the ISP and requires no extra control to the I/O section.

Finally, we were concerned in 1970 that there would be offsprings--clearly no problem; there have been about ten implementations. In fact, the family is large enough to suggest need of family planning.

3.0 TECHNOLOGY

The computers we build are strongly influenced by the basic electronic technology. In the case of computers, electronic information processing technology evolution has been used to mark the four generations.

3.1 Effects Of Semiconductor Memory On The PDP-11 Model Designs

The PDP-11 computer series design began in 1969 with the Model 20. Subsequently, 3 models were introduced as minimum cost, best cost/performance, and maximum performance machines. The memory technology in 1969 formed several constraints:

1. Core memory for the primary (program) memory with an eventual trend toward semiconductor memory.

2. A comparatively small number of high speed registers for processor state (i.e. general registers), with a trend toward larger, higher speed register files at lower cost. Note, only 16 word read-write memories were availableat design time.

3. Unavailability of large, high speed read-only memories,

permitting a microprogrammed approach to the design of the control part. Note, not for ca paper, read-only memory was unavailable although slow, read-only MOS was available for character generators.

These constraints established the following design principles and attitudes:

1. It should be asynchronous and capable of accepting various configurations of memories in size and speed.

2. It should be expandable, to take advantage of an eventually larger number of registers for more data-types and improve context switching time. Also, more registers would permit eventually mapping memory to provide a virtual machine and protected multiprogramming.

3. It could be relatively complex, so that an eventual microcode approach could be used to advantage. New data-types could be added to the instruction set to increase performance even though they added complexity.

4. The UNIBUS width would be relatively wide, to get as much performance as possible, since LSI was not yet available to encode functions.

3.2 Variations In PDP-11 Models Through Technology

Semiconductor memory (read-only and read-write) were used to tradeoff cost performance across a reasonably wide range of models. Various techniques based on semiconductors are used in the tradeoff to provide the range. These include:

1. Improve performance through brute force with faster memories. The 11/45 and 11/70 uses bipolar and fast MOS memory.

2. Microprogramming (see below) to improve performance through a more complex ISP (i.e., floating point).

3. Multiple copies of processor state (context) to improve time to switch context among various running programs.

4. Additional registers for additional data-types--i.e., floating point arithmetic.

5. Improve the reliability by isolating (protecting) one program from another.

6. Improve performance by mapping multiple programs into the same physical memory, giving each program a virtual machine. Providing the last two points requires a significant increase in the number of registers (i.e. at least 64 word fast memory arrays).

4.0 THE ORGANIZATION OF PEOPLE

Three types of design are based both on the technology and the cost and performance considerations. The nature of this tradeoff is shown in Figure DS. Note, that one starts at 0 cost and performance, proceeds to add cost, to achieve a base (minimum level of functionality). At this point, certain minimum goals are met: for the computer, it is simply that there is program counter, and the simplest arithmetic operations can be carried out. It is easy to show (based on the Turing machine) that only a few instructions are required, and from these, any program can be written. From this minimal point, performance increases very rapidly in a step fashion (to be described later) for quite sometime (due to fixed overhead of memories, cabinets, power, etc.) to a point of inflection where the cost-effective solution is found. At this point, performance continues to increase until another point where the performance is maximized. Increasing the size implies physical constraints are exceeded, and the machine becomes unbuildable, and the performance can go to 0. There is a general tendency of all designers to "n+1" (i.e., incrementally add to the design forever). No design is so complete, that a redesign can't improve it.

The two usual problems of design are: inexperience and "second-systemitis". The first problem is simply a resources problem. Are there people available? What are their backgrounds? Can a small group work effectively on architectural definitions? Perhaps most important is the principle, that no matter who is the architect, the design must be clearly understood by at least one person.

Second-systemitis is the phenomenon of defining a system on the basis of past system history.

Invariably, the system solves all past problems...bordering on the unbuildable.

4.1 PDP-11 Experience

Some of the PDP-11 architecture was initially carried out by at Carnegie-Mellon University (HM with GB). Two of the useful ideas: the UNIBUS, and the use of general registers in a substantially more general fashion (e.g. as stack pointers) came out of earlier work (GB) at CMU and was described in COMPUTER STRUCTURES (Bell and Newell, 1971). During the detailed design amelioration, 2 persons (HM, and RC) were responsible for the specification.

Although the architectural activity of the 11/20 proceeded in parallel with the implementation, there was less interaction than in previous DEC designs where the first implementation and architecture were carried out by the same person. As a result, a slight penalty was paid to build subsequent designs, especially vis a vis microprogramming.

As the various models began to be built outside the original PDP-11/20 group, nearly all architectural control (RC) disappeared, and the architecture was managed by more people, and design resided with no one person! A similar loss of control occurred in the design of the peripherals after the basic design.

The first designs for 16-bit computers came from a group placed under the PDP-15 management (a marketing person, with engineering background). It was called PDP-X, and did include a range. As a range architecture, it was better thought out than the later PDP-11, but didn't have the innovative aspects. Unfortunately, this group was intimidating, and some members lacked credibility. The group also managed to convince management that the machine was potentially as complex as the PDP-10 (which it wasn't); since no one wanted another large computer disconnected from the main business, it was a sure suicide. The (marketing) management had little understanding of the machine. Since the people involved in the design were apparently simultaneously designing Data General, the PDP-X was not of foremost importance.

As the PDP-X project folded and the DCM (for Desk Calculator Machine for security) project started up, design and planning were in disarray, since Data General had been formed and was competing with the PDP-8 using a very small 16-bit computer. Although the Product Line Manager, a former engineer (NM) for the PDP-8, had the responsibility this time, the new project manager was a mathematician/programmer followed by another manager (RC) who had managed the PDP-8. Work proceeded for several months based on the DCM and with a design review at Carnegie-Mellon University in late 1969. The DCM review took only a few minutes. Aside from a general dullness and a feeling that it was too little too late to compete. It was difficult to program (especially by compilers). However, it's benchmark results were good. (We believe it had been tuned to the benchmarks, hence couldn't do other problems very well.) One of the designers (HM) brought along the kernel of an alternative, which turned out to be the PDP-11. We worked on the design all weekend, recommending a switch to the basic 11 design.

At this point, there were reviews to ameliorate the design, and each suggestion, in effect, amounted to an n+1; the implementation was proceeding in parallel (JO) and since the logic design was conventional , it was difficult to tradeoff extensions. Also, the design was constrained with boards and ideas held over from the DCM. (The only safe way to design a range is simultaneously do both high and low end designs.) During the summer at DEC, we tried to free op code space, and increased (n+1'ed) the UNIBUS bandwidth (with an extra set of address lines), and outlined alternative models.

The advent of large, read-only memories, made possible the various follow-on designs to the 11/20. Figure "Models" sketches the cost of various models versus time, with lines of consistent performance. This very clearly shows the design styles (ideologies). The 11/40 design was started right after the 11/20, although it was the last to come on the market (the low and high ends had higher priority to get into production as they extended the market). Both the 11/04 and 11/45 design groups went through extensive buy in processes, as they came into the 11 by first proposing alternative designs. In the case of the 11/45, a larger, 11-like 18-bit machine was proposed by the 15 group; and later, the LINC engineering group proposed an alternative design which was subset compatible at the symbolic program level. As the groups considered

the software ramifications, buy-in was rapid. Figure Models shows the minimum cost-oriented group has two successors providing lower cost (yet higher performance) and the same cost with the ability to have larger memories and perform better. Note, both of these came from a backup strategy to the LSI-11. These come from larger read-only memories, and increased understanding of how to implement the 11.

The 11/70 is, of course, a natural follow on to extend the performance of the 11/45.

5.0 PMS STRUCTURE

In this section, we give an overview of the evolution of the PDP-11 in terms of its PMS structure, and compare it with expectations (Bell et al, 1970). The aspects include: the UNIBUS structure; UNIBUS performance; use for diagnostics; architectural control required; and multi-computer and multi-processor computer structures.

5.1 The UNIBUS - The Center Of The PMS Structure

In general, the UNIBUS has behaved beyond expectations, acting as a standard for intercommunication of peripherals. Several hundred types of memories and peripherals have been attached to it. It has been the principle PMS interconnection media of Mp-Pc and peripherals for systems in the range 3K dollars to 100K dollars (1975). For larger systems supplementary buses for Pc-Mp and Mp-Ms traffic have been added. For very small systems, like the LSI-11, a narrower bus (Q-bus) has been designed.

The UNIBUS by being a standard has provided us with a PMS architecture for easily configuring systems; any other organization can also build components which interface the bus...clearly ideal for buyers. Good busses (standards) make good neighbors (in terms of engineering), since people can concentrate on design in a structured fashion. Indeed, the UNIBUS has created a complete secondary industry dealing in alternative sources of supply for memories and peripherals. Outside of the IBM 360 I/O Multiplexor/Selector bus, the UNIBUS is the most widely used computer interconnection standard. Although it has been difficult to fully specify the UNIBUS such that one can be certain that a given system will work electrically and without missed data, specification is the key to the UNIBUS. The bus behavior specification is a yet unsolved problem in dealing with complexity--the best descriptions are based on behavior (i.e., timing diagrams).

There are also problems with the design of the UNIBUS. Although parity was assigned as two of the bits on the bus (parity and parity is available), it has not been widely used. Memory parity was implemented directly in the memory, since checking required additional time. Memory and UNIBUS parity is a good example of nature of engineering optimization. The tradeoff is one of cost and decreased performance versus decreased service cost and more data integrity for the user. The engineer is usually measured on production cost goals, thus parity transmission and checking are clearly a capability to be omitted from design...especially in view of lost performance. The internal Field Service organization has been unable to quantify the increase in service cost savings due to shorter MTTR by better fault isolation. Similarly, many of the transient errors which parity detects can be detected and corrected by software device drivers and backup procedures without parity. With lower cost for logic and increased responsibility (scope) to include warranty costs as part of the product design cost forces much more checking into the design.

The interlocked nature of the transfers is such that there is a deadlock when two computers are joined together using the UNIBUS window. With the window a computer can map another computer's address space into its own address space in a true multiprocessor fashion. Deadlock occurs when the two computers simultaneously attempt to access the other's addresses through each window. A request to the window is in progress on one UNIBUS, and at the same time a request to the other UNIBUS is in progress on the requestee's UNIBUS, hence neither request can be answered, causing a deadlock. One or both requests are aborted and the deadlock is broken by having the UNIBUS time out since this is equivalent to a non-existent address (e.g., a memory). In this way the system recovers and requests can be reissued (which may cause deadlock). The UNIBUS window is confined to applications where there is likely to be a low deadlock rate.

5.2 UNIBUS and Performance Optimality

Although we always want more performance on one hand, there is an equal pressure to have lower cost. Since cost and peformance are almost totally correlated the two goals perfectly conflict. The UNIBUS has turned out to be optimum over a wide dynamic range of products, (argued below). However, at the lowest size system, the Q-bus has been introduced, which contains about 1/2 the number of conductors; and at the largest systems, the data path width for the processor and memory has been increased to 32-bits for added performance although the UNIBUS is still used for communication with most I/O controllers.

Since all interconnection schemes are highly constrained, it is clear that future lower and higher systems cannot be accomplished from a single design unless a very low cost, high performance communication media (e.g. optical) is found.

The optimality of the UNIBUS comes about because memory size (number of address bits) and I/O traffic are correlated with the processor speed. Amdahl's rule-of-thumb for IBM computers (including the 360) is: one byte of memory is required per instruction/sec and one bit of I/O is required for each instruction executed. For our applications, we believe there is more computation required for each memory word, because of the bias toward control and scientific applications. Also, there has been less use of complex instructions typical of the IBM computers. Hence, we assume one byte of memory is required for each two instructions executed, and assume one byte of I/O is an upper bound (for real time applications) for each instruction executed. In the PDP-11, an average instruction accesses three to five bytes of memory, and with one byte of io, up to six bytes of memory are accessed for each instruction/sec. Therefore, a bus which can support two megabyte/sec traffic permits instruction execution rates of .33 to .5 mega instruction/sec. This imputes to meory sizes of .16 to .25 megabytes; the maximum allowable memory is .3 to .256 megabytes. By using a cache memory with a processor, the effective memory processor rate can be increased to further balance the processor. Alternatively, faster floating point operations will bring the balance to be more like the IBM data, requiring more memory.

5.3 Evolution Of Models: Predicted Versus Actual

The original prediction (Bell et al, 1970) was that models with increased performance would evolve using: increased path width for data; multi-processors; and separated bus structures for control and data transfers to secondary and tertiary memory. Nearly all of these forms have been used, though not exactly as predicted. (Again, this points to lack of overall architectural planning versus our willingness and belief that the suggestions and plans for the evolution must come from the implementation groups.)

In the earlier 11/45, a separate bus was added for direct access of either bipolar (300ns) or fast MOS (400ns) memory. In general, it was assumed that these memories would be small, and the user would move the important part of his algorithm to the fast memory for direct execution. The 11/45 provided a second UNIBUS for direct transmission of information to the fast memory without Pc interference. The 11/45 also increased performance by adding a second autonomous data operation unit called the Floating Point Processor (actually not a processor). In this way, both integer and floating point computation could proceed concurrently.

The 11/70, a cache based processor, is a logical extension of using fast, local memories, but without need for expert movement of data. It has a memory path width of 32-bits, and the control portion and data portion of I/O transfers have been separated as originally suggested. The performance limitation of the UNIBUS are removed, since the second Mp system permits data transfers of up to five megabytes/sec (2.5 times that of the UNIBUS). Note, that a peripheral memory map control is needed since Mp address space (two megabytes) exceeds the UNIBUS. In this way, direct memory access devices on the UNIBUS transfer data into a mapped portion of the larger address space.

5.4 Multi-processor Computer Structures

Although it is not surprising that multi-processors have not been used except on a highly specialized basis, it is depressing. In Computer Structures (Bell and Newell, 71) we carried out an

analysis of the IBM 360, predicating a multi-processor design. The range of performance covered by the PDP-11 models is substantially worse than with the 360, although the competitive environment of the two companies is substantially different. For the 360, smaller models appear to perform worse than the technology would predict. The reasons why multiprocessors have not materialized may be:

1. The basic nature of engineering is to be conservative. this is a classical deadlock situation: we cannot learn how to program multiprocessors until such systems exist; a system canot be built before programs are ready.

2. The market doesn't demand them. Another deadlock: how can the market demand them, since the market doesn't even know that such a structure could exist? IBM has not yet blessed the concept.

3. We can always build a better single, special processor. This design philosophy stems from local optimization of the designed object, and ignores global costs of spares, training, reliability and the ability of the user to dynamically adjust a configuration to his load.

4. There are more available designs for new processors than we can build already.

5. Planning and technology are asynchronous. Within DEC, not all products are planned and built at a particular time, hence, it is difficult to get the one right time when a multiprocessor would be better than an existing Uniprocessor together with one or two additional new processors.

6. Incremental market demands require specific new machines. By having more products, a company can better track competitors by specific uniprocessors.

5.4.1 Existent Multiprocessors -

Figure MP gives some of the multiprocessor systems that have been built on the 11 base. The top most structure has been built using 11/05 processors, but because of improper arbitration in the processor, the performance expected based on memory contention didn't materialize. We would expect the following results for multiple 11/05 processors sharing a single UNIBUS:

#Pc	Mp	Pc. PERF	Pc. PRICE	PRICE/ PERF*	SYS Price	Price/ PERF**
1	.6	1	1	1	3	1
2	1.15	1.85	1.23	.66	3.23	.58
3	1.42	2.4	1.47	.61	3.47	.48
40		2.25	1.35	.6	3.35	.49

*Pc cost only
** Total system, assuming 1/3 of system is Pc.cost

From these results we would expect to use up to three processors, to give the performance of a model 40. More processors, while increasing the performance, are less cost-effective. This basic structure has been applied on a production basis in the GT4X series of graphics processors. In this scheme, a second P.display is added to the UNIBUS for display picture maintenance.

The second type of structure given in Figure MP is a conventional multiprocessor using multiple port memories. A number of these systems have been installed and operate quite effectively, however, they have only been used for specialized applications.

The most extensive multiprocessor structure, C.mmp, has been described elsewhere. Hopefully, convincing arguments will be forthcoming about the effectiveness of multiprocessors from this work in order to establish these structures on an applied basis.

6.0 THE ISP

Determining an ISP is a design problem. The initial 11 design was based substantially on benchmarks, and as previously indicated this approach yielded a predecessor (not built) that though performing best on the six benchmarks, was difficult to program for other applications.

6.1 General ISP Design Problems

The guiding principles for ISP design in general, have been especially difficult because:

1. The range of machines argues for different encoding over the range. At the smallest systems, a byte-oriented approach with small addresses

is optimum, whereas larger implementations require more operations, larger addresses and encoding efficiency can be traded off to gain performance.

The 11 has turned out to be applied (and hopefully effective) over a range of 500 in system price ($500 to $250,000) and memory size (8k bytes to 4 megabytes). The 360 by comparison varied over a similar range: from 4k bytes to 4 megabytes.

2. At a given time, a certain style of machine ISP is used because of the rapidly varying technology. For example, three address machines were initially used to minimize processor state (at the expense of encoding efficiency), and stack machines have never been used extensively due to memory access time and control complexity. In fact, we can observe that machines have evolved over time to include virtually all important operations on useful data-types.

3. The machine use varies over time. In the case of DEC, the initial users were sophisticated and could utilize the power at the machine language level. The 11 provided more fully general registers and was unique in the minicomputer marketplace, which at the time consisted largely of 1 or 2 accumulator machines with 0 or 1 index registers. Also, the typical minicomputer operation codes were small. the 11 extended data-typing to the byte and to reals. by the extension of the auto-indexing mode, the string was conveniently programmed, and the same mechanism provided for stack data-structures.

4. The machine is applied into widely different markets. Initially the 11 was used at the machine language level. The user base broadened by applications with substantially higher level languages. These languages initially were the scientific based register transfer languages such as BASIC, FORTRAN, DEC'S FOCAL, but the machine eventually began to be applied in the commercial marketplace for the RPG, COBOL, DIBOL, and BASIC-PLUS languages which provided string and decimal data-types.

5. The criteria for a capability in an instruction set is highly variable, and borders on the artistic. Ideal goals are thus to have a complete set of operations for a given basic data-type (e.g. integers)--completeness, and operations would be the same for varying length data-types--orthogonality. Selection of the data-types is totally a function of the application. That is, the 11 considers both bytes and full words to be integers, yet doesn't have a full set of operations for the byte; nor are the byte and word ops the same. By adhering to this principle, the compiler and human code generators are greatly aided.

   We would therefore ask that the machine appear elegant, where elegance is a combined quality of instruction formats relating to mnemonic significance, operator/data-type completeness and orthogonality, and addressing consistency. By having completely general facilities (e.g., registers) and which are not context dependent assists in minimizing the number of instruction types, and greatly aids in increasing the understandability (and usefulness).

6. Techniques for generating code by the human and compiler vary widely. With the 11, more addressing modes are provided than any other computer. The 8 modes for source and destination with dyadic operators provide what amounts to 64 possible instructions; and by associating the Program Counter and Stack Pointer registers with the modes, even more data accessing methods are provided. For example, 18 forms of the MOVE instruction can be seen (Bell et al, 1971) as the machine is used as a two-address, general registers and stack machine program forms. (The price for this generality is extra bits). In general, the machine has been used mostly as a general register machine.

7. Basic design can take the very general form or be highly specific, and design decisions can be bound in some combination of microcode or macrocode with no good criteria for tradeoff.

6.2 Problems In Extending The Machine Range

Several problems have arisen as the basic machine has been extended:

1. The operation-code extension problem--the initial design did not leave enough free opcode space for extending the machine to increase the data-types.

   At the time the 11/45 was designed (FPP was added), several extension schemes were examined: an escape mode to add the floating point operations; bringing the 11 back to a more conventional general register machine by reducing the modes and finally, typing the data by adding a global mode which could be switched to select floating point (instead of byte operations).

2. Extending the addressing range--the UNIBUS limits the physical memory to 262,144 bytes (18-bits). In the implementation of the 11/70, the physical address was extended to 4 megabytes by providing a UNIBUS map so that devices in a 262K UNIBUS space could transfer into the 4 megabyte space by mapping registers.

   While the physical address limits are acceptable for both the UNIBUS and larger systems, the address for a single program is still confined to an instantaneous space of 16 bits, the user virtual address.

   The main method of dealing with relatively small addresses is via process-oriented operating systems that handle large numbers of smaller tasks. This is a trend in operating systems, especially for process control and transaction processing. It also enforces a structuring discipline in the (user) program organization. The RSX series operating systems are organized this way, and the need for large addresses except for problems where large arrays are accessed is minimized.

   The initial memory management proposal to extend the virtual memory was predicated on dynamic, rather than static assignment of memory segment registers. In the current memory management scheme, the address registers are usually considered to be static for a task (although some operating systems provide functions to get additional segments).

7.0 SUMMARY

This paper has re-examined the PDP-11 and compared it with the initial goals and constraints. With hindsight, we now clearly see what the problems with the initial design were. Design faults occurred not through ignorance, but because the design was started too late. As we continue to evolve and improve the PDP-11 over the next five years, it will indeed be interesting to observe, however, the ultimate test is use.

BIBLIOGRAPHY

Ames, G.T., Drongowski, P.J. and Fuller, S.H. Emulating the Nova on the PDP-11/40: a case study. Proc. COMPCON (1975).

Bell, G., Cady, R., McFarland, H., Delagi, B., O'Loughlin, J., Noonan, R., and Wulf, W. A new architecture of minicomputers-- the DEC PDP-11. Proc. SJCC (1970) Vol36, pp.657-675.

Bell, C.G. and Newell, A. Computer Structures. McGraw Hill (1971)

Bell, J.R. Threaded code. COMM ACM (June 1973) Vol 16, No. 6, pp 370-372.

Eckhouse, R.H. Minicomputer Systems: organization and programming (PDP-11). Prentice-Hall,(1975)

Fusfeld, A. R. The technological progress function. Technology Review (Feb. 1973) pp.29-38

McWilliams, T., Sherwood, W., Fuller, S., PDP-11 Implementation using the Intel 3000 microprocessor chips. Submitted to NCC (May 1976)

O'Loughlin, J.F. Microprogramming a fixed architecture machine. Microprogramming and Systems Architecture Infotech State of the Art Report 23. pp205-224

Ornstein, 1972? (page 28)

Stone, H.S. and Siewiorek, D.P. Introduction to computer organization and data structures: PDP-11 Edition. McGraw-Hill, (1975)

Turn, R. Computers in the 1980's. Columbia University Press 1974.

Wulf, W.A., Bell, C.G., C.mmp: A multi-mini-processor. FJCC (1972)

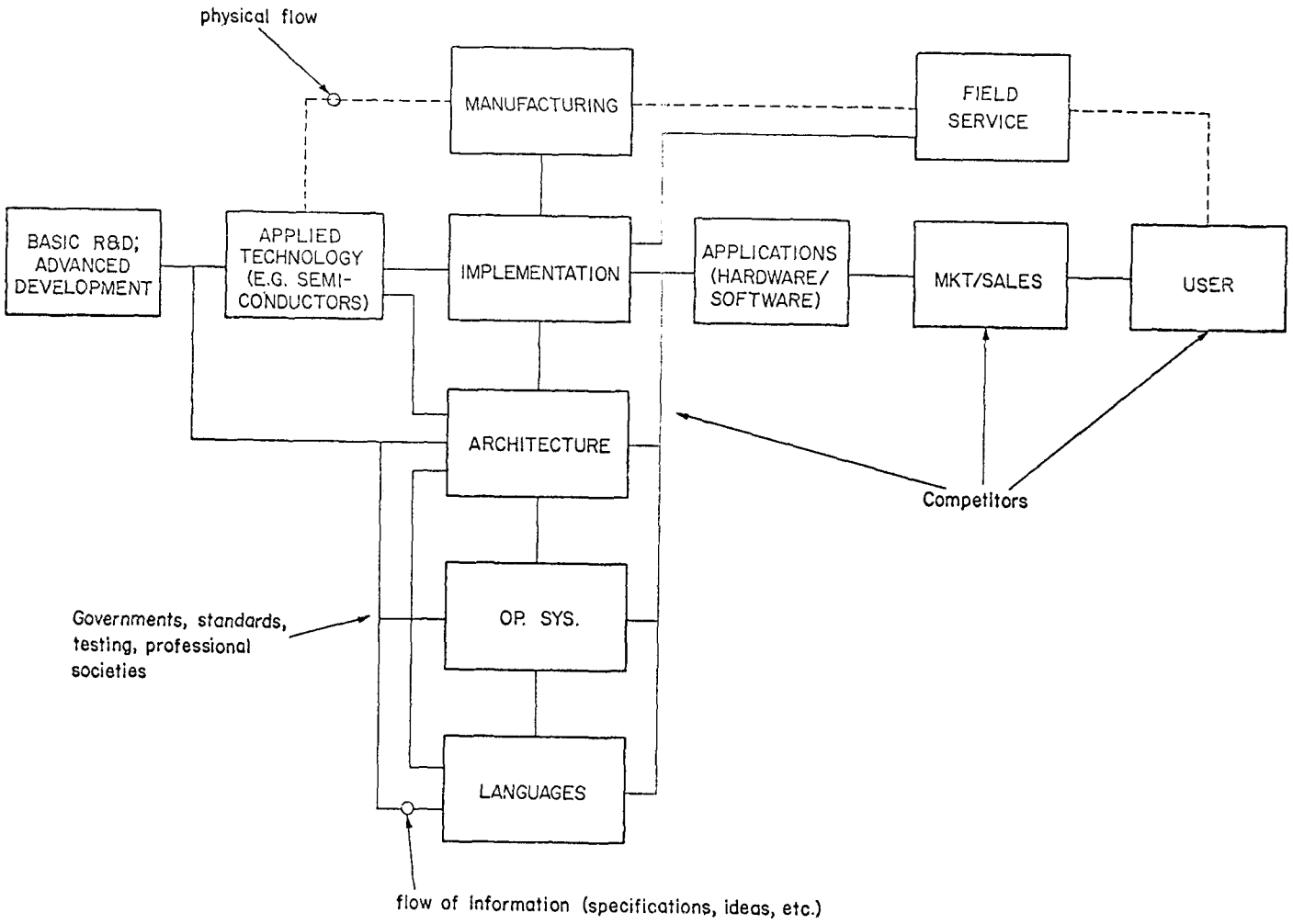
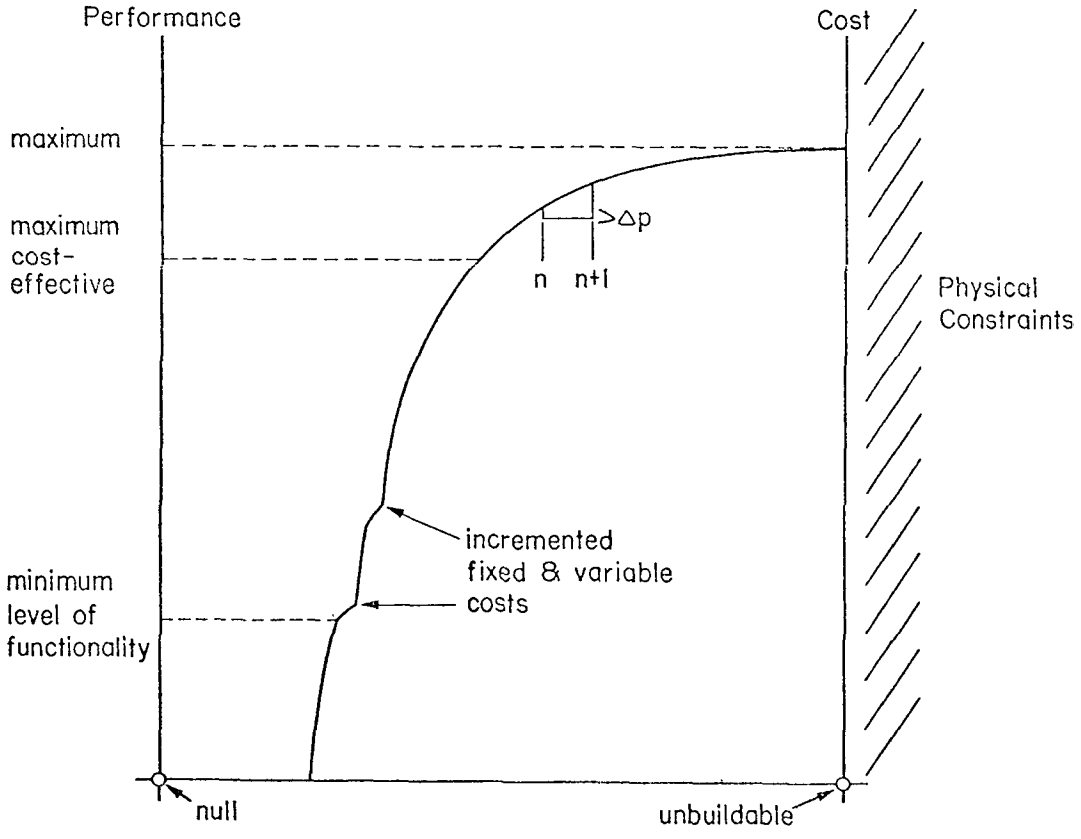


FIGURE ORG. STRUCTURE OF ORGANIZATION AFFECTING A COMPUTER DESIGN

Fig. DS DESIGN STYLES (IDEOLOGIES) IN TERMS OF COST AND PERFORMANCE



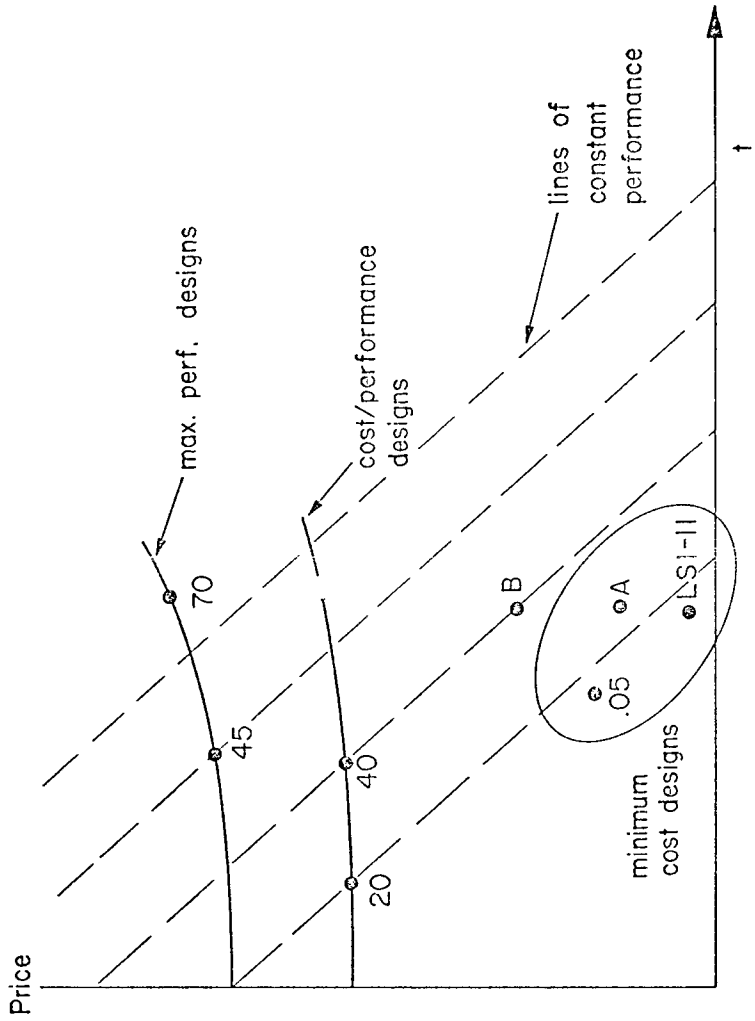
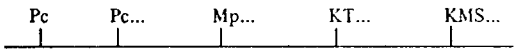
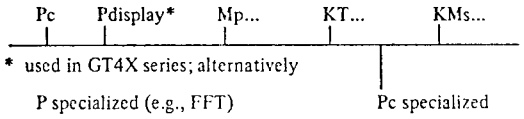


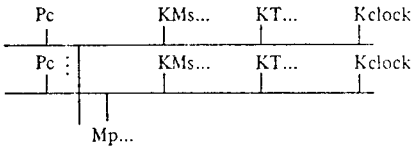
Fig. MODELS PDP-11 MODELS PRICE VERSUS TIME WITH LINES OF CONSTANT PERFORMANCE



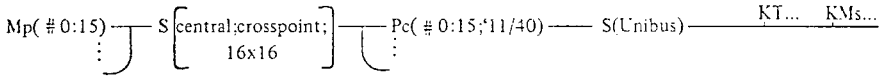
a. Multi-Pc structure using a single Unibus.



b. Pc with P.display using a single Unibus.



c. Multiprocessor using multiport Mp.



d. C.mmp CMU multi-mini-processor computer structure.

Figure MP Multi-Processor Computer Structures Implemented using PDP-11