The PDP-11 after Three Design Generations
~~WHAT HAVE WE LEARNED FROM THE PDP-11?~~

[This needs sections on CIS, 11/74MP, dirty dozen and I/O benchmarks]
[The cost/performance figs. need 11/74, 11/34 with cache / CIS]

1. INTRODUCTION

A computer is not solely determined by its architecture; it reflects the
technological, economic, and human aspects of the environment in which it
was designed and built. In Chapters 1 and 2 we discussed the non-architectural
design factors: the availability and price of the basic electronic
technology, the various government and industry rules and standards, the
current and future market conditions. ~~The finished computer is a product~~
~~of the total design environment.~~

In this chapter, we use the evolution of the PDP-11 to provide a concrete
example of how the various forces interact. We reflect on the PDP-11: it
goals, its architecture, its various implementations, and the people who
designed it. We examine the design, beginning with the architectural
specifications, and observe how it was affected by technology, and by the
development organization, the sales, application, and manufacturing
organizations, and the nature of the final users. [GB: Do we cover all,
e.g., manufacturing?]

2. BACKGROUND: THOUGHTS BEHIND THE DESIGN

It is the nature of computer engineering to be goal-oriented, with pressure
to produce deliverable products. It is therefore difficult to plan for an

Table Cady   PDP-11 Family Projection as of April 3, 1969.   4/3/69

MODEL	CP	LOGICAL POWER	ARITH POWER	SPEED (microsec.)	PRICE	CONFIGURATION	SOFTWARE PAPER TAPE	DISK
PDP-11/10		.7	.7	2-3μ	4K	Technilogically cost reduced 11/20 with MOS		
PDP-11/20	KA11	1	1	2.2μ	5.2K	CP, 1KBROM, 128 by R/W, Turnkey Console		
PDP-11/30	KA11	1	1	2.2μ	9.3K	CP, 8KB Core, Console, TTY .	Assembler, Editor, Math Utility FOCAL, BASIC, (ASA Basic Fortran)[3]	8-like monitor (syst.builder w/ODT, DOT, PIP)[2]
PDP-11/40	KB11	2^1	10-20	1.2μ	13K	adds *, /, normalize, etc. Possible micro-programmed processor, no EAE saves $1000	Possible 16KB Fortran IV Improved Assembler	Fortran IV
PDP-11/45	KB11	2^1	10-20	1.2μ	15K + Disk	11/45 with memory protect/relocate max core 262KB, Max phys memory (using disk) 2^{22} Bytes		Super Monitor[4] 65KB virtual mem/user for either small or large Disk
PDP-11/50	KC11	2^1	50-100	1.2μ	25K	adds hardware floating point 32 bit processor, 16 bit memory (16KB)		
PDP-11/55	KC11	2^1	50-100	1.2μ	27K + Disk	with memory protect/relocate		
PDP-11/65	KD11	4	100-200	1.2μ / 32 bit	45K + Disk	32 bit separate memory bus 32 bit processor.		

COMPANY CONFIDENTIAL

NOTES:

1. If microprogrammed, then logical power could be tailored to user and go to 20-50, 40-100 for 11/65.

2. Business language system under consideration.

3. Possible by-product of FOCAL.

4. Super monitor for 11-45, 55, 65 is priority multi-user real-time system.

----

extensive lifetime. Nevertheless, the PDP-11 evolved rapidly, and over a

much wider range than we expected. This rapid evolution would have placed

unusual stress even on a carefully planned system. The PDP-11 was not

extremely well planned or controlled; rather it evolved under pressure from

implementation and marketing groups. *However there was a plan for more models. Roger Cady who headed the 11 group outlined the ~~to to~~ + several subsequent machines in a memo on _____ (see Table Cady).*

Because of the many pressures on the design, the planning was asynchronous

and diffuse; development was distributed throughout the company. This sort

of decentralized design organization provides a system of checks and

balances, but often at the expense of perfect hardware compatibility. This

compatibility can hopefully be provided in the software, and at lower cost

to the user. *[Insert Table Cady]*

Despite its evolutionary planning, the PDP-11 has been quite successful in

the marketplace: over 50,000 have been sold in the *first* eight years that it has

been on the market (1970-1977). It is not clear how rigorous a test (aside

from the marketplace) we have given the design, since a large and

aggressive marketing organization, armed with software to correct

architectural inconsistencies and omissions, can save almost any design.

It has been interesting to watch as ideas from the PDP-11 migrate to other

computers in newer designs. Although some of the features of the PDP-11

are patented, machines have been made with similar bus and ISP structures.

One company has manufactured a machine said to be "plug compatible" with a

PDP-11/40. Many *computer* designers have adopted the UNIBUS as their fundamental

architectural component. Many microprocessor designs incorporate the

UNIBUS notion of mapping I/O and control registers into the memory address

----

space, eliminating the need for I/O instructions without complicating the

I/O control logic. When the LSI-11 was being designed, no alternative to

the UNIBUS-style architecture was even considered.

An earlier paper [Chapter 6] described the design goals and constraints for

the PDP-11, beginning with a discussion of the weaknesses frequently found

in minicomputers. The designers of the PDP-11 faced each of these known

minicomputer weaknesses, and our goals included a solution to each one. In

this section we shall review the original design goals and constraints,

commenting on the success or failure of the PDP-11 at meeting each of them.

The first weakness of minicomputers was their limited addressing

capability. The biggest (and most common) mistake that can be made in a

computer design is that of not providing enough address bits for memory

addressing and management. The PDP-11 followed this hallowed tradition of

skimping on address bits, but it was saved by the principle that a good

design can evolve through at least one major change.

For the PDP-11, the limited-address problem was solved for the short run,

but not with enough finesse to support a large family of minicomputers.

That was indeed a costly oversight, resulting in both redundant development

and lost sales. It is extremely embarassing that the PDP-11 had to be

redesigned with memory management only two years after writing the paper

that outlined the goal of providing increased address space. All

predecessor DEC designs have suffered the same problem, and only the PDP-10

evolved over a long period (fifteen years) before a change occurred to

increase its address space. In retrospect, it is clear that since memory

prices decline 26 to 41% yearly, and users tend to buy "constant-dollar"

systems, then every two or three years another address bit will be

required.

A second weakness of minicomputers was their tendency not to have enough

registers. This was corrected for the PDP-11 by providing eight 16-bit

registers. Later, six 64-bit registers were added for floating-point

arithmetic. This number seems to be adequate: there are enough registers

to allocate two or three (beyond those already dedicated to program counter

and stack pointer) for program global purposes and still have registers for

local statement computation. More registers would increase the

multiprogramming context switch time and confuse the user.

A third weakness of minicomputers was their lack of hardware stack

capability. In the PDP-11, this was solved with the

autoincrement/autodecrement addressing mechanism. This solution is unique

to the PDP-11 and has proven to be exceptionally useful. (In fact, it has

been copied by other designers.)

A fourth weakness, limited interrupt capability and slow context switching,

was essentially solved with the device of UNIBUS interrupt vectors, which

direct device interrupts. Implementations could go further by providing

automatic context saving in memory or in special registers. This detail

was not specified in the architecture, nor has it evolved from any of the

implementations to date. The basic mechanism is very fast, requiring only

four memory cycles from the time an interrupt request is issued until the

first instruction of the interrupt routine begins execution.

A fifth weakness of prior minicomputers, inadequate character-handling
capability, was met in the PDP-11 by providing direct byte addressing
capability. Although string instructions are not yet provided in the
hardware, the common string operations (move, compare, concatenate) can be
programmed with very short loops. Early benchmarks showed that this
mechanism was adequate. However, as COBOL compilers have improved and as
more understanding of operating systems string handling has been obtained,
there appears to be a need for a string instruction set.

A sixth weakness, the inability to use read-only memories, was avoided in
the PDP-11. Most code written for the PDP-11 tends to be pure and
reentrant without special effort by the programmer, allowing a read-only
memory (ROM) to be used directly. ROMs are used extensively for bootstrap
loaders, program debuggers, and for normal simple functions. Because large
ROMs were not available at the time of the original design, there are no
architectural components designed specifically with large ROMs in mind.

A seventh weakness, one common to many minicomputers, was primitive I/O
capabilities. The PDP-11 answers this to a certain extent with its
improved interrupt structure, but the more general solution of I/O
processors has not yet been implemented. The I/O-processor concept is used
extensively in the GT4X display series, and for signal processing. Having
a single machine instruction that would transmit a block of data at the
interrupt level would decrease the CPU overhead per character by a factor
of three, and perhaps should have been added to the PDP-11 instruction set
for implementation on all machines. Provision was made in the 11/60 for
invocation of a micro-level interrupt service routine in WCS, but the
architecture is yet to be extended.

Another common minicomputer weakness was the lack of system range. If a
user had a system running on a minicomputer and wanted to expand it or
produce a cheaper turnkey version, he frequently had no recourse, since
there were often no larger and smaller models with the same architecture.
The problem of range and how it is handled in the PDP-11 is discussed
extensively in a later section.

A ninth weakness of minicomputers was the high cost of programming them.
Many users program in assembly language, without the comfortable
enviornment of editors, file systems, and debuggers available on bigger
systems. The PDP-11 does not seem to have overcome this weakness, although
it appears that more complex systems are being built successfully with the
PDP-11 than with its prececessors, the PDP-8 and PDP-15. Some systems
programming is done using higher-level languages; the optimizing compiler
for BLISS-11, however, at first ran only on the PDP-10. The use of BLISS
has been slowly gaining acceptance. It was first used in implementing the
FORTRAN-IV PLUS compiler. Its use in PDP-10 and VAX-11 systems programming
has been more widespread.

One design constraint that turned out to be expensive, but probably wor[h]t
it in the long run, was that the word length had to be a multiple of eight
bits. Previous DEC designs were oriented toward 6-bit characters, and DEC
has a large investment in 12-, 18-, and 36-bit systems. The notion of word
length is somewhat meaningless in machines like the PDP-11 and the IBM
System/360, because data types are of varying length, and instructions have
varying length: one or more groups of 16 bits.

Microprogrammability was not an explicit design goal, partially since the
large ROMs which make it feasible were not available at the time of the
original Model 20 implementation. All subsequent machines have been
microprogrammed, but with some difficulty and expense.

Understandability as a design goal seems to have been minimized. The
PDP-11 was initially a hard machine to understand, and was marketable only
to those who really understood computers. Most of the first machines were
sold to knowledgeable users in universities and research laboratories. The
first programmers' handbook was not very helpful, and the second, arriving
in 1972, helpted only to a limited extent. It is still not clear whether a
user with no previous computer experience can figure out how to use the
machine from the information in the handbooks. Fortunately, several
computer science textbooks [Gear 74, Eckhouse 75, and Stone and Siewiorek
75] have been written based on the PDP-11; their existence should assist
the learning process.

We do not have a very good understanding of the style of programming our
users have adopted. Since the machine can be used in so many ways, there
and there are discussed in the section on the ISP.
have been many programming styles. Former PDP-8 users adopt a
one-accumulator convention; novices use the two-address form; some
compilers use it as a stack machine; probably most of the time it is used
as a memory-to-register machine with a stack for procedure calling.
Frequencies of the various addressing modes have been tabulated from
Strecker's program traces and are given in Appendix A of Chapter 11. The
high frequency of destination mode zero suggests high use of a
memory-to-register programming style.

Structural flexibility (modularity) for hardware configurations was an important goal. This succeeded

beyond expectations, and is discussed extensively in the UNIBUS section.

3. TECHNOLOGY: COMPONENTS OF THE DESIGN

In Chapter 2, we observed that computers are very strongly influenced by

the basic electronic technology of their components. The PDP-11 family

provides the best example, of all DEC computers, of designing with improved

technologies. Because design resources have been available to do

concurrent implementations spanning a cost/performance range, we have a

rich source of examples of the three different design styles: constant

cost with increasing functionality, constant functionality with decreasing

cost, and growth-path.

Memory technology has had a much greater impact on PDP-11 evolution than

logic technology. Except for the LSI-11, the one logic family (7400 series

TTL) has dominated PDP-11 implementations since the beginning. Except for

a small increase following the 11/20, gate desnity has not improved

markedly. Speed improvement has taken place -- with Schottky TTL -- as has

a power improvement -- LS series. Departures from MSI TTL, in terms of

gate density, have been few -- but very effective. Examples are the 2901

bit-slice in the 11/34 floating-point processor, the use of PLA's in the

11/04 and 11/34 control units, and the use of ECL in some clock circuitry.

                                                              1969
Memory densities and costs have improved rapidly since 1970 and have thus

the most impact. Read/write memory chips have gone from 16 bits to 4096

                                       now   in 1978.
bits in density and ROM's with 8K or 16 Kbits are widely available. This

Various figures for semiconductor memory size characteristics are given in tables 00 and 00 of chapter 2 (page 00) available since 1962 doubling each year (see chapter 2, page 00) according to both/TC 2 - 1972 using the model of semiconductor density

section discusses the PDP-11 evolution through memory technologies.

The memory technology of 1969 imposed several constraints. First, core
memory was cost effective for the primary (program) memory, but a clear
trend toward semiconductor primary memory was visible. Second, since the
largest high-speed read/write memories available were 16 words, then the
number of processor registers should be kept small. Third, there were no
large high-speed read-only memories that would have permitted a
microprogrammed approach to the processor design.

These constraints established four design attitudes toward the PDP-11's
architecture. First, it should be asynchronous, and thereby capable of
accepting different configurations of memory that operate at different
speeds. Second, it should be expandable to take eventual advantage of a
larger number of registers, both user registers for new data types and
internal registers for improved context switching, memory mapping and
protected multiprogramming. Third, it could be relatively complex, so that
a microcode approach could eventually be used to advantage: new data types
could be added to the instruction set to increase performance, even though
they might add complexity. Fourth, the UNIBUS width should be relatively
large, to get as much performance as possible, since the amount of
computation possible per memory cycle is relatively small.

As semiconductor memory of varying price and performance became available,
it was used to trade cost for performance across a reasonably wide range of
models. Different techniques were used on different models to provide the
for all models except the 11/20 a
range. These techniques include; microprogramming to enhance performance

as described in Chapter 00.

(for example, faster floating point); use of faster program memories for

(e.g. model 11/55 and the 11/60 with writable control store)

brute-force speed improvements, use of fast caches to optimize program

(models 11/70, 11/60 and cached 11/34);

memory references, and expanded use of fast registers inside the processor.

(models > 11/45)

+Observe how semiconductor technology availability

has "driven" the various

PDP-11 designs.

The use of semiconductors versus cores for primary memory is a purely economic consideration as we discuss in Chapter 2.
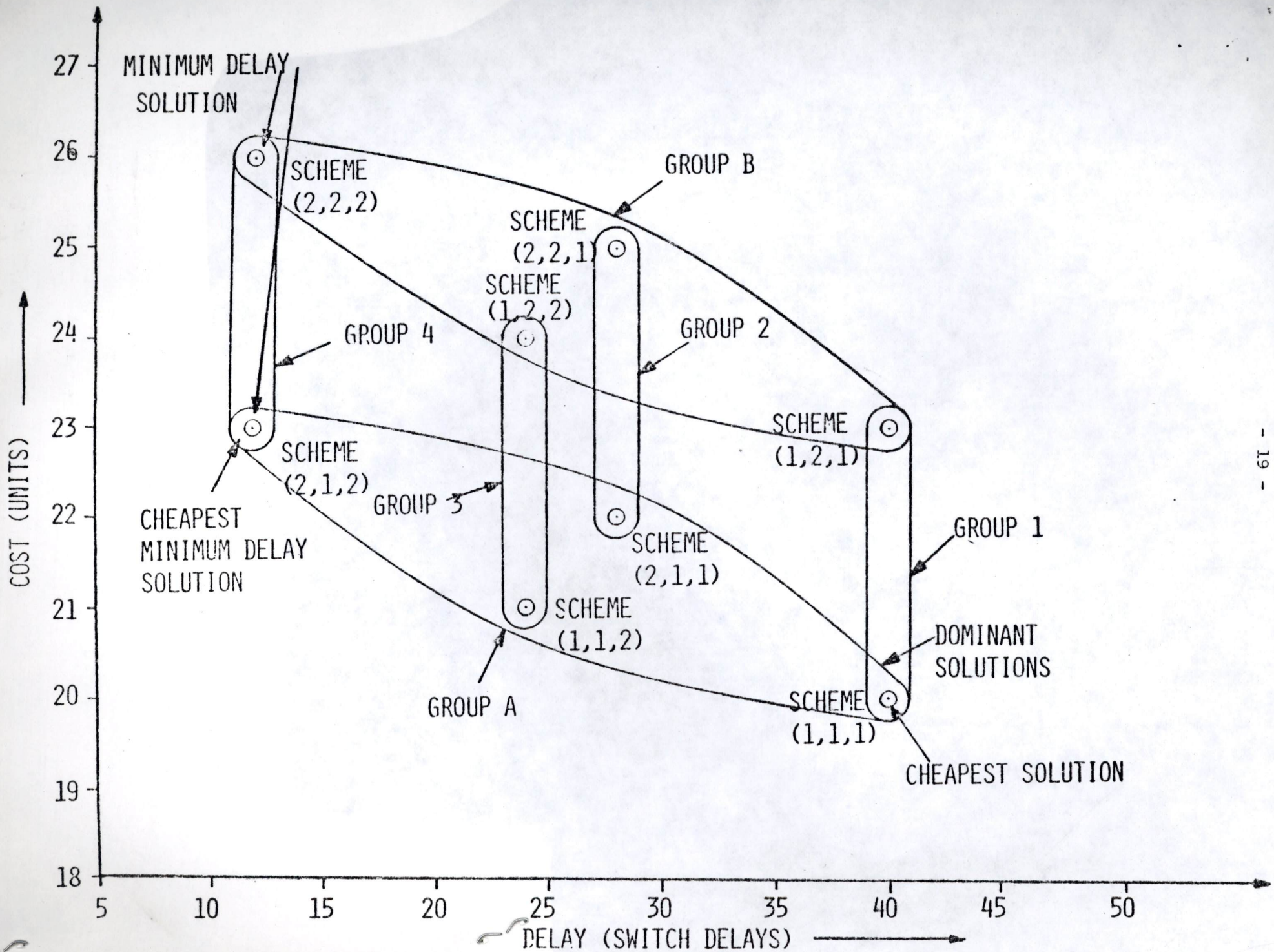
note pages

10 - 14 are removed

FIGURE 7: PLOT OF COST VERSUS DELAY FOR VARIOUS SCHEMES OF THE MULTIPLIER ALGORITHMS

In this section we shall outline the evolutionary process of the PDP-11 design, describing how the flavor of the design was subtly determined by ~~this~~ the style of the designers

4.1.   ~~THE SYSTEM ARCHITECTURE~~

Some of the initial work on the architecture of the PDP-11 was done at Carnegie-Mellon University by Harold McFarland and Gordon Bell. Two of the useful ideas, the UNIBUS and the generalized use of the program registers (such as for stack pointers and program counters), came out of earlier work by Gordon Bell and were described in Bell and Newell [71]. The detailed design specification was the work of Harold McFarland and Roger Cady.

The PDP-11/20 was the first model designed. Its design and implementation took place more or less in parallel, but with far less interaction between architect and builder than for previous DEC designs, where the ~~first~~ architect was the first implementor. As a result, some of the architectural specifications caused problems in subsequent designs, especially in the ~~area of~~ ability to build a machine that could be easily implemented using microprogramming.

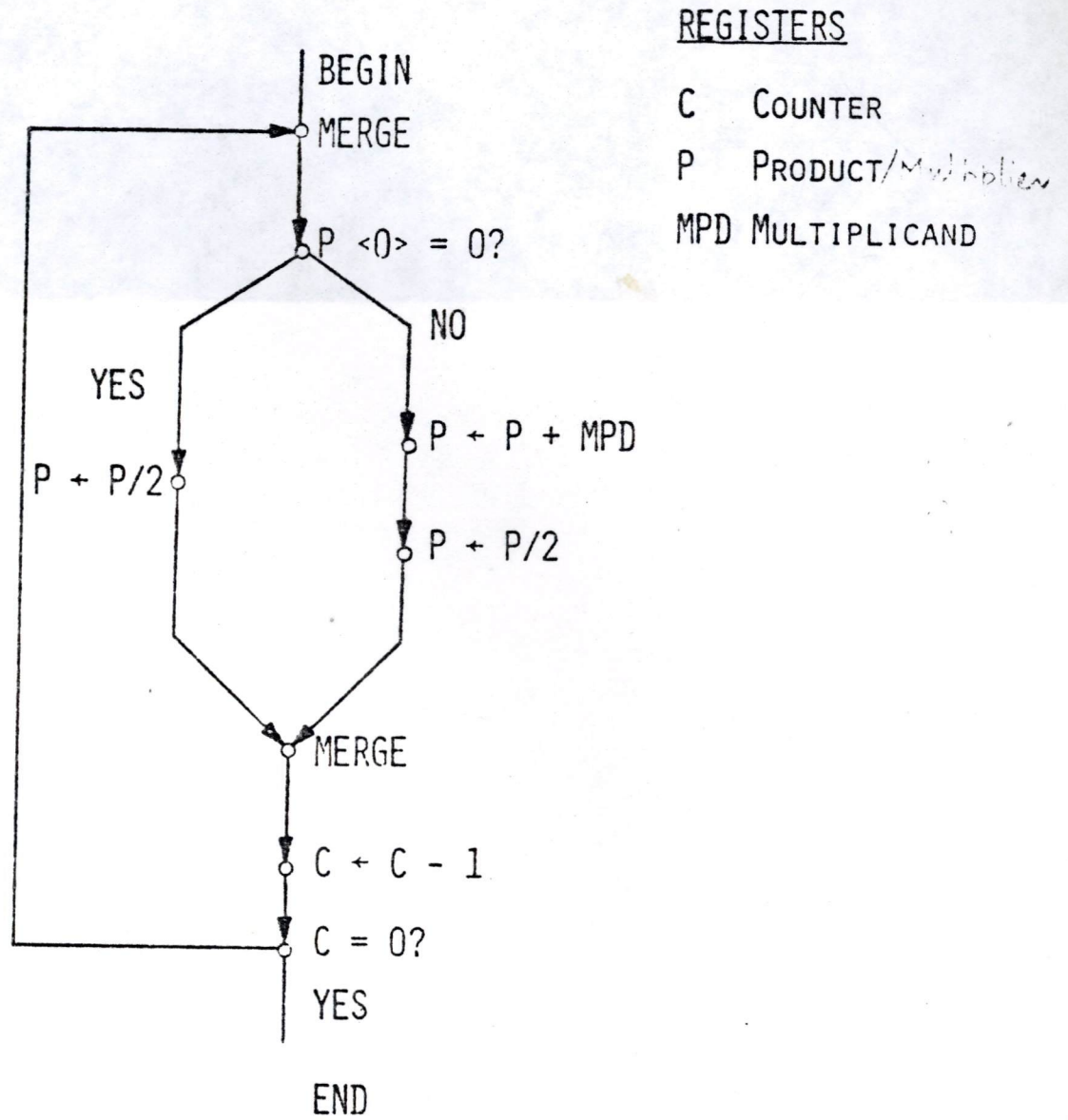As there began to appear other models besides the original Model 20, strong

REGISTERS

C   COUNTER

P   PRODUCT/Multiplier

MPD MULTIPLICAND

FIGURE 5:   PHYSICAL CONTROL PROGRAM FOR 16 BIT
MULTIPLICATION

(ALL REGISTERS ARE 16 BIT WIDE)

----

architectural controls disappeared; there was no one person responsible for

the family-wide design. A similar loss of control occurred in the design

of the peripherals after the basic design.

4.2.   A CHRONOLOGY OF THE DESIGN

The internal organization of DEC design groups has through the years

oscillated between market orientation and product orientation. Since the

company has been growing at a rate of 30 to 40% a year, there has been a

constant need for reorganization. During a high growth rate year, one third of the

staff has been with the company less than a year.

At the time of the PDP-11 design, the company was structured along product

lines. The design talent in the company was organized into tight groups:

the PDP-10 group, the PDP-15 (an 18-bit machine) group, the PDP-8 group,

an ad hoc PDP-8/S subgroup, and the LINC-8 group. Each group included

marketing and engineering people responsible for designing a product,

software and hardware. As a result of this organization, architectural

experience was diffused among the groups, and there was little

understanding of the notion of a range of products.

The PDP-10 group was the strongest group in the company. They built large,

powerful time-shared machines. It was essentially a separate division of

the company, with little or no interaction with the other groups. Although

the PDP-10 group as a whole had the best understanding of system

architectural controls, they had no notion of system range, and were only

interested in building higher-performance computers.

The PDP-15 group was relatively strong, and was an obvious choice to build

the new mid-range 16-bit PDP-11. The PDP-15 series was a constant-cost

series that tended to be optimized for cost performance. However, the

PDP-11 represented direct competition with their existing line. Further,

the engineering leadership of that group changed from one implementation to

the next, and thus there was little notion of architectural continuity or

range.

The PDP-8 group was a close-knit group who did not communicate very much

with the rest of the company. They had a fair understanding of

architecture, and were oriented toward producing minimal-cost designs with

an occasional high-performance model. The PDP-8/S "group" was actually one

person, someone outside the regular PDP-8 group. The PDP-8/S was an

attempt to build a much lower-cost version of the PDP-8 and show the group

engineers how it should be done. The 8/S worked, but it was not terribly

successful because it sacrificed too much performance in the interests of

economy.

The LINC-8 group produced machines aimed at the biomedical and laboratory

market, and had the greatest engineering strength outside the PDP-10 group.

The LINC-8 people were really the most systems oriented. The LINC design

came originally from MIT's Lincoln Laboratory, and there was dissent in the

company as to whether DEC should continue to build it or to switch software

to the PDP-8.

The first design work for a 16-bit computer was carried out under the eye

of the PDP-15 manager, a marketing person with engineering background.

This first design was called PDP-X, and included specification for a range

of machines. As a range architecture, it was better designed than the

later PDP-11, but was not otherwise particularly innovative.

Unfortunately, this group managed to convince management that their design

was potentially as complex as the PDP-10 (which it was not), and thus

ensured its demise, since no one wanted another large computer unrelated to

the company's main large computer. In retrospect, the people involved in

designing PDP-X were ~~apparently~~ working simultaneously on the design of

Data General. ~~[GB: Is this statement too catty?]~~

As the PDP-X project folded, the DCM (Desk Calculator Machine, a code name

chosen for security) was started. Design and planning were in disarray, as

Data General had been formed and was competing with the PDP-8, using a very

small 16-bit computer. Work on the DCM progressed for several months,

culminating in a design review at Carnegie-Mellon University in late 1969.

The DCM review took only a few minutes; the general feeling was that the

machine was dull and would be hard to program. Although its benchmark

results were good, we now believe that it had been tuned to the benchmarks

and would not have fared well on other sorts of problems.

One of the DCM designers, Harold McFarland, brought along the kernel of an

alternative design, which ultimately grew into the PDP-11. Several people

worked on the design all weekend, and ended by recommending a switch to the

new design. The machine soon entered the design-review cycle, each step

being an n+1 of the previous one. As part of the design cycle, it was

necessary to ensure that the design could achieve a wide cost/performance

range. The only safe way to design a range is to simultaneously do both

----

the high- and low-end designs. 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.

Meanwhile an implementation was underway, led by Jim O'Laughlin. The logic

design was conventional, and the design was hampered by the holdover of

ideas and circuit boards from the DCM. As ideas were tested on the

implementation model, various design changes were proposed; for example,

the opcodes were adjusted and the UNIBUS width was increased with an extra

set of address lines.

the relationship of the various designs is shown in Fig. 11Tree.

With the introduction of large read-only memories, various follow-on

designs to the Model 20 were possible. Figure 2 sketches the cost of

various models over time, showing lines of constant performance. The

graphs show clearly the differing design styles used in the different

models.

Note there are roughly three generations of designs. Here we only discuss the first two and the extensions which created the VAX-11 architecture.

organizations that designed the second generation

The 11/40 and 11/45 design groups went through extensive "buy-in"

45   05

processes, as they each came to the PDP-11 by first proposing alternative

designs. The people who ultimately formed the 11/45 group had started by

One designer-architect, Ad van der Goor, who did many of the 11/45 architectural extensions

proposing a PDP-11-like 18-bit machine with roots in the PDP-15. Later a

design and design group.

totally different design was proposed, with roots in the LINC group, that

was instruction subset-compatible at the source program level. As the

groups considered the impact of their changes on software needs, they

rapidly joined the mainstream of the PDP-11 design.

architecturally incompatible

The 11/05 designers also came from the LINC group. Steve Teicher headed the group, and Bob Armstrong carried out the detailed logical design with Bob Kusik writing the microprogram.

generations (i.e. Model 20; and models 45, 40 and 05)

Dick Clayton, who had just headed the LINC Production line, led the group, for that produced the 11/45, Bruce DeLagi was responsible for the logical design, and Len Hughes for designing the fast floating point unit.

Note from Fig. 2 that the minimum-cost group had two successors to their

original design, one cheaper with slightly improved performance, the other

the same price with greatly improved performance and flexibility.

→ Subseg since subseg

5. THE PDP-11: AN EVALUATION

The end product of the PDP-11 design is the computer itself, and in the

evolution of the architecture we can see images of the evolution of ideas.

In this section, we outline the architectural evolution, with a special

emphasis on the UNIBUS.

In general, the UNIBUS has behaved beyond all expectations. Several

hundred types of memories and peripherals have been interfaced to it; it

has become a standard architectural component of systems in the $3K to

$100K price range (1975). The UNIBUS is a price and performance optimizer:

it limits the performance of the fastest machines and penalizes the

lower-performance machines with a higher cost. For larger systems,

supplementary buses were added for Pc-Mp and Mp-Ms traffic. For very small

systems like the LSI-11, a narrower bus (called a Q-bus) was designed.

The UNIBUS, as a standard, has provided an architectural component for

easily configuring systems. Any company, not just DEC, can easily build

components that interface to the bus. Good buses make good engineering

neighbors, since people can concentrate on structured design. Indeed, the

UNIBUS has created a secondary industry providing alternative sources of

supply for memories and peripherals. With the exception of the IBM 360

Multiplexor/Selector bus, the UNIBUS is the most widely used computer

Recently it has become clear that the Unibus is
adequate for large, high performance systems because the
cache structure reduces the traffic between Mp and Pc by a factor of the
miss ratio.


interconnection standard.

5.1.   THE ARCHITECTURE AND THE UNIBUS

The UNIBUS is the architectural component that connects together all of the
other major components. It is the vehicle over which data flow takes
place. Its structure is shown in Fig. 3. Traffic between any pair of
components moves along the UNIBUS. The original design anticipated the
following traffic flows.

1.   Pc-Mp for the processor's programs and data.

2.   Pc-K for the processor to issue I/O commands to the controller K.

3.   K-Pc, for the controller K to interrupt the Pc.

4.   Pc-K for direct transmission of data from a controller to Mp under
     control of the Pc.

5.   K-Mp for direct transmission of data from a controller to Mp; i.e., DMA
     data transfer.

6.   K-T-K-Ms, for direct transmission of data from a device to secondary
     memory without intervening Mp bufferiing; e.g., a disk refreshing a
     CRT.

Experience has shown that paths 1 through 5 are used in every system that
has a DMA (direct memory access) device. An additional communications path
has proved useful: demons, i.e., special Kio/Pio/Cio communicating with a
conventional K. The These demons are used for direct control of another K in
order to remove the processing load from Pc.

Figure 3: UNIBUS structure

Pc	Mp		K --- T		K --- Ms
|	|		|		|
|	|		|		|
|	|		|		|
|	|		|		|
|	|		|		|
|	|		|		|

--(UNIBUS)
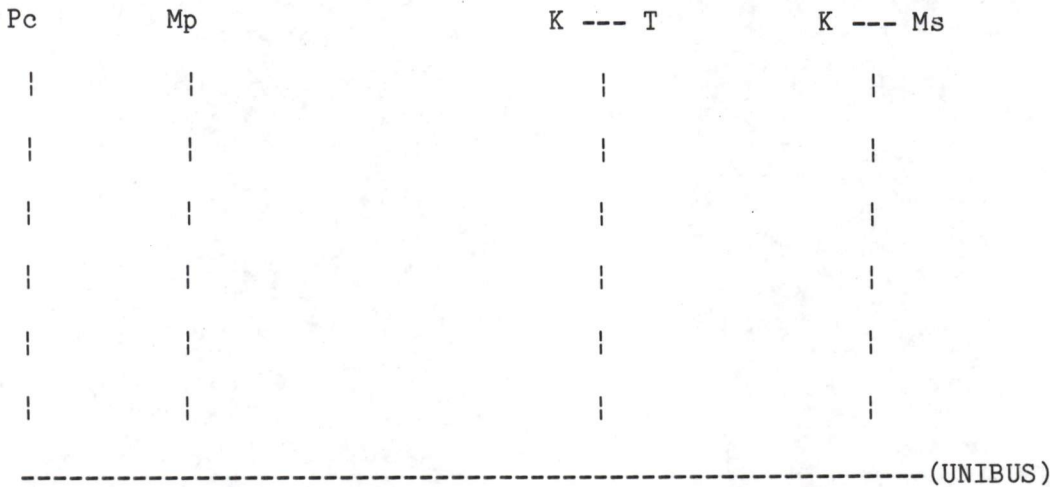
Several examples of a demon come to mind: a K that handles all

communication with a conventional subordinate Kio (e.g., an A/D converter

interface or communications line); a full processor executing from Mp a

program to control K; or a complete I/O computer, Cio, which has a program

in its local memory and which uses Mp to communicate with Pc. Effectively,

Pc and the demon act together, and the UNIBUS connects them. Demons

provide a means of gracefully off-loading the Pc by adding components, and

is useful for handling the trivial pre-processing found in analog,

communications, and process-control I/O. The DMC-11, ~~communications~~

~~processor~~ is an example.

with KMC-11 control processor for
high speed data communications link

5.1.1.   UNEXPECTED BENEFITS FROM THE DESIGN

The UNIBUS has *also* turned out to be invaluable as an "umbilical cord" for

factory diagnostic and checkout procedures. Although such a capability was

not part of the original design, the UNIBUS is almost capable of dominating

----
the Pc, Tp's, and Mp during factory checkout and diagnostic work.

Ideally, the scheme would let all registers be accessed during full

operation. This is now possible for all devices except Pc. By having all

Pc registers available for reading and writing in the same way that they

are now available from the console switches, a second system could fully

monitor the computer in the same fashion as a human. Although the DEC

factory uses a UNIBUS umbilical cord to watch systems under test, human

intervention is occasionally required.

In most recent PDP-11 models, a serial communications line is connected to

the console, so that a program may remotely examine or change any

information that a human operator could examine or change from the front

panel, even when the system is not running. In this way computers can
be remot diagnosed from a remote site.

5.1.2.   **DIFFICULTIES WITH THE DESIGN**

The UNIBUS design is not without problems. Although two of the bus bits

were in the original design set aside as parity bits, they have not been

widely used as such. Memory parity was implemented directly in the memory;

this phenomenon is a good example of the sorts of problems encountered in

engineering optimization. The trading of bus parity for memory parity

exchanged higher hardware cost and decreased performance for decreased

service cost and better data integrity. Since engineers are usually judged

on how well they achieve production cost goals, parity transmission is an

obvious choice to pare from a design, since it increases the cost and

decreases the performance. As logic costs decrease and pressure to include

warranty costs as part of the product design cost increases, the decision

to transmit parity might be reconsidered.

Early attempts to build multiprocessor structures (by mapping the address

space of one UNIBUS onto the memory of another) *Called a UNIBUS Window* were beset with deadlock

problems. The UNIBUS design does not allow more than one master at a time.

Successful multiprocessors required much more sophisticated sharing

mechanisms *(e.g. a shared primary memory)* than this UNIBUS Window.

At the time the UNIBUS was designed, it was felt that allowing 4K bytes of

the address space for I/O control registers was more than enough. However,

so many different devices have been interfaced to the bus over the years

that it is no longer possible to assign unique addresses to every device.

The architectural group has thus been saddled with the chore of device

address bookkeeping. Many solutions have been proposed, but none was soon

enough; as a result, they are all so costly that it is cheaper just to live

with the problem and the attendant inconvenience.

5.2.   UNIBUS COST AND PERFORMANCE

Although performance is always a design goal, so is low cost; the two goals

conflict directly. The UNIBUS has turned out to be nearly optimum over a

wide range of products. It served as an adequate memory-processor

interconnect for six of the ten models. However, in the smallest system,

we introduced the Q-bus, which uses about half the number of conductors.

For the largest systems, we use a separate 32-bit data path between

processor and memory, although the UNIBUS is still used for communication

----

with most I/O controllers. The UNIBUS slows down the high-performance

machines and increases the cost of low-performance machines; it is optimum

over the middle range. Levy [Chapter 9] discusses the evolution in more

detail.

There are several attributes of a bus that affect its cost and performance.

One factor affecting performance is simply the data rate of a single

conductor. There is a direct tradeoff among cost, performance, and

reliability. Shannon [48] gives a relationship between the fundamental

signal bandwidth of a link and the error rate (signal-to-noise ratio) and

data rate. The performance and cost of a bus are also affected by its

length. Longer cables cost proportionately more, and the longer

propagation times necessitate more complex circuitry to drive the bus.

Since a single-conductor link has a fixed data rate, the number of

conductors affects the net speed of a bus. The cost of a bus is directly

proportional to the number of conductors. For a given number of wires,

time-domain multiplexing and data encoding can be used to trade performance

and logical complexity. Since logic technology is advancing faster than

wiring technology, we suspect that fewer conductors will be used in all

future systems. There is also a point at which time-domain multiplexing

impacts performance.

If during the original design of the UNIBUS we could have forseen the wide

range of applications to which it would be applied, its design would have

been different. Individual controllers might have been reduced in

complexity by more central control. For the largest and smallest systems,

----

it would have been useful to have a bus that could be contracted or
expanded by multiplexing or expanding the number of conductors.

The cost-effective success of the UNIBUS is due in large part to the high
correlation between memory size, number of address bits, I/O traffic, and
processor speed. Amdahl's rule of thumb for IBM computers is that 1 byte
of memory and 1 byte/sec of I/O are required for each instruction/sec. For
DEC applications, with emphasis in the scientific and control applications,
there is more computation required per memory word. Further, the PDP-11
instruction sets do not contain the complex instructions typical of IBM
computers, so a larger number of instructions may be executed to accomplish
the same task. Hence, we assume 1 byte of memory for each 2
instructions/sec, and that 1 byte/sec of I/O occurs for each
instruction/sec.

[This paragraph to be moved to 8 range]

In the PDP-11, an average instruction accesses 3-5 bytes of memory, so
assuming 1 byte of I/O for each instruction/sec, there are 4-6 bytes of
memory accessed on the average for each instruction/sec. Therefore, a bus
that can support 2 megabyte/sec traffic permits instruction execution rates
of 0.33-0.5 megainstructions/sec. This implies memory sizes of 0.16-0.25
megabytes; the maxiumum allowable memory is 0.064-0.256 megabytes. by
using a cache memory on the processor, the effective memory processor rate
can be increased to balance the system further. If fast floating point
instructions were added to the instructions were added to the instruction
set, the balance would approach that used by IBM and thereby require more
memory (seen in the 11/70).

5.3.   **EVOLUTION OF THE DESIGN**

The market life of a computer is determined in part by how well the design

can gracefully evolve to accommodate new technologies, innovations, and

market demands. As component prices decrease, the price of the computer

can be lowered, and by compatible improvements to the design (the "mid-life

kicker"), the useful life can be extended. An example of a mid-life kicker

is the writable control store for user microprogramming of the 11/40 [Almes

et al. 75]. The PDP-11 designs have used the mid-life kicker technique

occasionally. In retrospect, this was probably poor planning. Now that we

understand the problem of extending a machine's useful life, this

capability can be more easily designed in.

Fig. 4. Use of dual Pc multiprocessor system with processorless UNIBUS for
I/O data transmission (from Bell et al. [70]).

and in Cady's memo, page 00,

In the original PDP-11 paper [Bell et al. 70], it was forecast that there

----

would evolve models with increased performance, and that the means to

achieve this increased performance would include wider data paths,

multiprocessors, and separate data and control buses for I/O transfers.

Nearly all of these devices have been used, though not always in the style

that had been expected.

Figure 4 shows ~~a~~ dual-processor system ~~as~~ was originally suggested. *in chapter 00. (see Fig. 8, page 00).* A number

of systems of this type have been built, but without the separate I/O data

and control buses, and with minimal sharing of Mp. The switch S permitting

two computers to access a single UNIBUS, has been widely used in

high-availability high-performance systems.

Fig. 5.   PMS structure of 11/45.

In designing higher-performance models, additional buses were added so that

a processor could access fast local memory. The original design ~~never~~ did not

anticipated the availability of large fast semiconductor memories. In the

----

past, high-performance machines have parlayed modest gains in component

technology into substantially more performance by making architectural

changes based on the new component technologies. This was the case with

both the PDP-11/45 (see ~~Fig. 5~~ *page 00*) and the PDP-11/70 (see ~~Fig. 6~~ *page 00*).

Fig. 6. ~~PMS structure of 11/70.~~

Fig. 7a. PDP-11/03 (LSI-11) block diagram. (*indicates one LSI chip each

and one for data and registers.)

In the PDP-11/45, a separate bus was added for direct access to either

300-nsec bipolar or 350-nsec MOS memory. It was assumed that these

memories would be small, and that the user would move the important parts

of his program into the fast memory for direct execution. The 11/45 also

provided a second UNIBUS for direct transmission of data to the fast memory

without processor interference. The 11/45 also used a second autonomous

data operation unit called a Floating Point Processor (not a true

processor), which allowed integer and floating-point calculations to

proceed concurrently.

Fig. 7b.   PDP-11/05 block diagram.

The PDP-11/70 derives its speed from the cache, which allows it to take

advantage of fast local memories without requiring the program to suffle

data in and out of them. The 11/70 has a memory path width of 32 bits, and

has separate buses for the control and data portions of I/O transfer. The

performance limitations of the UNIBUS are circumvented; the second Mp

----

system permits transfers of up to 5 megabytes/sec., 2.5 times the UNIBUS

limit. If direct memory access devices are placed on the UNIBUS, their

address space is mapped into a portion of the larger physical address

space, thereby allowing a virtual-system user to run real devices.

Chapter 00 gives
~~Figure 7~~ ~~shows~~ the block diagrams of the ~~LSI-11, the 11/05, and the 11/45~~. various processor implementations so they may be compared.
~~It includes the smallest and largest (except the 11/70) models.~~ Note ~~that~~
~~the 11/45 block diagram does not include the floating-point operations, but~~
~~does show the path to fast local memory. It~~ has duplicate sets of
to get increased parallelism
registers, even to a separate program counter. The local Mp.MOS and

Mp.Bipolar provide the greatest performance improvements by avoiding the

UNIBUS protocols. When only core memory is used, the 11/45 floating-point

performance is only twice that of the 11/40. Table III charts the

implementation of each design and its performance and parallelism as

measured by the microprogram memory width. Note that the brute-force speed

of the 11/45 core is only 2 to 4 times faster than the 11/05 for simple

data types, i.e., for the basic instruction set. The 11/45 has roughly

twice the number of flip-flops.

~~Fig. 7c.   PDP-11/45 block diagram.~~

5.4.   ISP DESIGN

Designing the ISP level of a machine -- that collection of characteristics
such as instruction set, addressing modes, trap and interrupt sequences,
register organization, and other features visible to a programmer of the
bare machine -- is an extremely difficult problem. One has to consider the
performance (and price) ranges of the machine family as well as the
intended applications, and there are always difficult tradeoffs. For
example, a wide performance range argues for different encodings over the
range. For small systems a byte-oriented approach with small addresses is
optimal, whereas larger systems require more operation codes, more
registers, and larger addresses. Thus, for larger machines, instruction
coding efficiency can be traded for performance.

The PDP-11 was originally conceived as a small machine, but over time its
range was gradually extended so that there is now a factor of 500 in price
($500 to $250,000) and memory size (8K bytes to 4 megabytes) between the
smallest and largest models. This range compares favorably with the range
of the 360 family (4K bytes to 4 megabytes). Needless to say, a number of
problems have arisen as the basic design was extended.

For one thing, the initial design did not have enough opcode space to
accommodate instructions for new data types. Ideally, the complete set of
operation codes should have been specified at initial design time so that
extensions would have fit. Using this approach, the uninterpreted

----

operation codes could have been used to call the various operation

functions (e.g., floating-point add). This would have avoided the

proliferation of runtime support systems for the various hardware/software

floating point arithmetic methods (Extended Arithmetic Element, Extended

Instruction Set, Floating Instruction Set, Floating Point Processor). This

technique was used in the Atlas and SDS designs, but most computer

designers don't know or remember the techniques. By not specifying the ISP at the

initial design, completeness and orthogonality have been sacrificed.

At the time the 11/45 was designed, several extension schemes were

examined: an escape mode to add the floating point operations, bringing

the 11 back to being a more conventional general-register machine by

reducing the number of addressing modes, and finally, typing the data by

adding a global mode that could be switched to select floating point

instead of byte operations for the same opcodes. The FPP of the PDP-11/45

is a version of the second alternative.

It also became necessary to do something about the small address space of

the processor. The UNIBUS limits the physical memory to 262,144 bytes

(addressable by 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 256K UNIBUS space could transfer into the 4 megabyte space via

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 many small tasks. This is a

----

trend in operating systems, especially for process control and transaction
processing. It does, however, enforce a structuring discipline in (user)
program organization. The RSX series operating systems for the PDP-11 are
organized this way, and the need for large addresses is minimized.

The initial memory management proposal to extend the virtual memory was
predicted 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 dynamically).

With dynamic assignment, a user can address a number of segment names, via
a table, and directly load the appropriate segment registers. The segment
registers act to concatenate additional address bits in a base address
fashion. There have been other schemes proposed that extend the addresses
by extending the length of the general registers -- of course, extended
addresses propagate throughout the design and include double length address
variables. In effect, the extended part is loaded with a base address.

With larger machines and process-oriented operating systems, the context
switching time becomes an important performance factor. By providing
additional registers for more processes, the time (overhead) to switch
context from a process (task) to another process can be reduced. This
option has not been used in the implementations of the 11's to date.
Various alternatives have been suggested, and to accomplish this most
effectively requires additional operators to handle the many aspects of
process scheduling. This extension appears to be relatively unimportant

----

since the range of computers coupled with networks tend to alleviate the
need by increasing the real parallelism (as opposed to the apparent
parallelism) by having various independent processors work on the separate
processes in parallel. The extensions of the 11 for better control of I/O
devices is clearly more important in terms of improved performance.

The criteria used to decide whether or not to include a particular
capability in an instruction set are highly variable and border on the
artistic. We 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. Having completely general facilities (e.g., registers) which
are not context dependent assists in minimizing the number of instruction
types, and greatly aids in increasing understandability (and usefulness).
We feel the 11 provided this.

Techniques for generating code by the human and compiler vary widely and
thus affect ISP design. The 11 provides more addressing modes than nearly
any other computer. The 8 modes for source and destination with dyadic
operators provide what amounts to 64 possible add instructions. By
associating the Program Counter and Stack Pointer registers with the modes,
even more data accessing methods are provided. For example, 18 varieties
of the MOVE instruction can be distinguished (see Chapter OO) [Bell et al. 70] as the
machine is used in two-address, general-register and stack machine program
forms. (There is a price for this generality -- namely, fewer bits could
have been used to encode the address modes that are actually used most of
the time.)

(i.e. memory to registers)

In general, the 11 has been used mostly as a general register machine. In

one case, it was observed that a user who previously used a 1-accumulator

computer (e.g., PDP-8), continued to do so. ~~Normally, the machine is used~~

~~as a memory to registers machine.~~ *A general register machines* ~~This~~ provides the greatest performance,

and the cost (in terms of bits) is the same as when used as a stack

machine. Some compilers, particularly the early ones, are stack oriented

since the code production is easier. Note, that in principle, and with

much care, a fast stack machine could be constructed. However, since most

stack machines use Mp for the stack, there is a loss of performance even if

the top of the stack is cached. The stack machine is perhaps the most

poorly understood concept in computing. While a stack is natural (and

necessary) structure to interpret the nested block structure languages, it

doesn't necessarily follow that the interpretation of all statements should

occur in the context of the stack. In particular, the predominance of

register transfer statements are of the simple 2- and 3-address forms.

This can be seen by observing the use frequency from Strecker's data, page oo.

D <-- S

and

D1 (index 1) <-- f (S2(index 2), S3 (index 3)).

These don't require the stack organization. In effect, appropriate

assignment allows a general register machine to be ued as a stack machine

for most cases of expression evaluation. It has the advantage of providing

temporary, random access to common sub-expressions, a capability that is

usually hard to exploit in stack architectures.

[Need CIS extension statent]

5.5.   MULTIPROCESSORS

Although it is not surprising that multiprocessors have not been used save

in highly specialized applications, it is depressing. One way to extend

the range of a family is to build multiprocessors. In this section we

examine some factors affecting the design and implementation of

multiprocessors, and their affect on the PDP-11.

It is the nature of engineering to be conservative. Given that there are

already a number of risks involved in bringing a product to the market, it

is not clear why one should build a higher-risk structure that may require

a new way of programming. What has resulted is a sort of deadlock

situation: we cannot learn how to program multiprocessors until such

machines exist, but we won't build the machine until we are sure that there

will be a demand for it, i.e, that the programs will be ready.

While on the subject of demand for multiprocessors, we should note that

there is little or no market presssure for them. Many Most users don't even

know that multiprocessors exist. Even though multiprocessors are used

extensively in the high-performance systems built by Burroughs, DEC

(PDP-10), and Univac, the concept has not yet been blessed by IBM.

even though there is a need for increased reliability and availability of machines.

One reason that there is not a lot of demand for multiprocessors is

acceptance of the philosophy that we can always build a better

single-processor system. Such a processor achieves performance at the

considerable expense of cost of spares, training, reliability, and

flexibility. Although a multiprocessor architecture provides a measure of

----

reliability, backup, and system tunability unreachable on a conventional

system, the biggest, fastest machines are always uniprocessors.

5.5.5.   MULTIPROCESSORS BASED ON THE PDP-11

Multiprocessor systems have been built out of PDP-11's. Figure 8

summarizes the design and performance of some of these machines. The

topmost structure was built using 11/05 processors, but because of improper

arbitration techniques in the processor, the expected performance did not

materialize. Table IV shows the expected results for multiple 11/05

processors sharing a single UNIBUS:

From these results we would expect to use as many as three 11/05 processors

to achieve the performance of a Model 40. More than 3 processors will

increase the performance at the expense of the cost-effectiveness. 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 picutre maintenance. A similar structure is used for

connecting special signal-processing computers to the UNIBUS although these

structures are technically coupled computers rather than multiprocessors.

Fig. 8. Multiprocessor computer structures implemented using PDP-11.

As an independent check on the validity of this approach, a multiprocessor

system has been built, based on the Lockheed SUE[Ornstein et al. 72]. This

machine, used as a high-speed communications processor, is a hybrid design:

it has seven dual-processor computers with each pair sharing a common bus

as outlined above. The seven pairs share two shared multiport memories.

                            Table IV

| | Pc perf. | | | | |
#Pc	(rel.)	Pc price	Price[a]/perf.	SYS price	Price[b]/perf.
1	1.00	1.00	1.00	3.00	1.00
2	1.85	1.23	0.66	3.23	0.58
3	2.4	1.47	0.61	3.47	0.48
40	2.25	1.35	0.60	3.35	0.49

[a]Pc cost only.

[b]Total-system cost assuming one-third of system is Pc cost.

The second type of structure given in Fig. 8 is a conventional

multiprocessor using multiple-port memories. A number of these systems

have been installed, and they operate quite effectively. However, they

have only been used for specialized applications.

The most ambitious multiprocessor structure made from PDP-11's, C.mmp, is

----

amply described in the literature [Wulf et al. 72]. As it becomes a user

machine, we will gather data about its effectiveness. Hopefully, data from

this and other multiprocessor efforts will establish multiprocessors as

applicable and useful in a wide variety of situations. *The DEC 16 LSI-11 processor*

PULSAR System (see page 00) described on page 00 is ~~an attempt to st~~ a vehicle to investigate the use of

[The 11/74 Multiprocessor]

6. PDP-11 FAMILY EVALUATION

Dirty dozen CPU   [Benchmarks]   ~~dirty dozen CPU benchmarks~~

go here

[I/O workload]   ~~i/o workload~~ study

6.1 COMPATIBILITY

a large number of processors LSI for such structures.

6.2 FAMILY RANGE

7. VAX-11

Enlarging the virtual-address space of an architecture has far more

implications than enlarging the physical-address space. The simple device

of relocating program generated addresses can solve the latter problem.

The physical address space, the amount of physical memory that can be

addressed, has been increased in two steps in the PDP-11 family. The KT-11

memory management unit expanded the address field from 16 to 18 bits and

then from 18 to 22 bits on the 11/70.

The virtual address space, or name space, is a much more fundamental part

of an architecture. Such addresses are programmer generated: he uses

these to name data objects, their aggregates (whether they be vectors,

matrices, lists, or shareable data segments) and instructions (subroutine

addresses, for example). Names seen by an individual program are part of a

larger name space -- that managed by an operating system and its associated

----

language translators and object-time systems. An operating system provides

sharing and protection, for example, using the name space of the

architecture.

As the 11/70 design progressed, we realized that for some large

applications there would soon be a bad mismatch between the 64 Kbyte name

space and 4 Mbyte memory space. Two trends could be clearly seen: (1)

minicomputer users would be processing large arrays of data, particularly

in FORTRAN programs (only 8096 double precision floating point numbers are

needed to fill a 16-bit name space), and (2) applications prrograms were

growing rapidly in size, particularly COBOL programs for

transaction-oriented processing. Moreover, anticipated memory price

declines made the problem worse. The need for a 32-bit integer data type

was felt, but this was far less important than the need for 32-bit

addressing of a name space.

Thus, in 1974, architectural work seriously began on extending the virtual

address space of the 11. Strecker and Mudge led the efforts. The

principal goal was compatibility with the PDP-11. Each of the general

registers, R0-R7, was extended to 32 bits. The addressing modes, and hence

address arithmetic, inherent in the PDP-11 allowed this to be a natural,

easy extension.

The design of the structure to be placed on a 32-bit virtual address

presented the "most" difficulty. The most PDP-11-compatible structure

would view a 32-bit address as 2^{16} 16-bit PDP-11 segments each having the

substructure of the KT11 memory management architecture. This segmented

----

address space, although PDP-11 compatible, was ill-suited to FORTRAN, which

expects a linear address space.

A severe design constraint was that existing PDP-11 subroutines must be

callable from programs which ran in extended mode. The main problem areas

were in establishing a protocol for communicating addresses (between

programs between the operating systems and programs on the occurrence of

interrupts). Saving state (the program counter and its extension) on the

stack was straightforward. However, the accessing of linkage addresses on

the stack after a subroutine call or interrupt was not straightforward.

Complicated sequences were necessary to ensure that the correct number of

bytes (representing a 32-bit or 16-bit address) were popped from the stack.

Our understanding of the thoroughness of the solution was hampered by the

fact that DEC customers programmed the PDP-11 at all levels -- there was no

clear user level, below which DEC had complete control, as is the case with

the IBM S/360.

The proposed architecture was the result of work by engineers, architects,

operating system designers and compiler designers. Moreover, it was

subjected to close scrutiny by a wider group of engineers and programmers.

Much was learned about the consequences of strict PDP-11 compatibility, the

notions of degree of compatibility, e.g., KT-11 are or not, and the software

costs which would be incurred by an extended PDP-11 architecture.

Fortunately, the project was shelved. There were many reservations about

its viability. The two major reasons were (a) it was felt that the

11-compatibility constraint caused too much compromise. Any new

architecture would require a large software investment; it was essential

that it be a quantum jump over the PDP-11 to justify the effort. (b) there

was not the necessary "buy in" from a group working on a low cost

implementation of the PDP-10.

In April, 1975, work on a 32-bit architecture was reviewed and led to

directly to VAX-11. Strecker was the principal architect. As a result of

the experience with the extended -11 designs, it was decided to drop the

constraint of the PDP-11 instruction format in designing the extended

virtual address space or native mode of the VAX-11 architecture. However,

in order to run existing PDP-11 programs, VAX-11 includes an 11

compatibility mode. This mode provides the basic PDP-11 instruction set

less privileged instructions (as defined by the RSX-11M operating system)

and floating point instructions. Neither is the KT-11 memory management

architecture preserved in this mode.

Preserving the existing instruction formats would have enacted too high a

price in dynamic bit efficiency. Whereas the PDP-11 has a high level of

efficiency in this area (the Army/Navy CFA project measured this), adding

the 50(?) new operation codes for the anticipated data types would have

lowered the efficiency instruction stream bit. An opcode extension field

would have been required. We felt that data stream bit efficiency could be

improved, because measurements showed that 98% of all literals were 6 bits

or less in length.

Besides the desire to add the data types for string, integer 32, integer

64, and decimal arithmetic, there were many other extenstions proposed.

These included a common CALL protocol, demand paging, true indexing,

context-sensitive indexing, and ~~good~~ more I/O addressing.

Along the way, some major perturbations to the 11 style were considered and

rejected. The major ones are discussed below.

Typed data and descriptor addressing were rejected on the grounds of

dynamic bit efficiency. Although system software costs may be lower with

such architectures, we were unable to quantify the gain convincingly. *Also,*
Such an architecture destroyed any compatibility with PDR 11

Our experience with PDP-11 (floating point, in particular) led us to reject

a soft-machine architecture, i.e., one with an instruction set (and highly

microprogrammed implementations) for general purpose emulation. Our PDP-11

experience showed that embedding a data type (once it is understood) in the

architecture gives a higher performance gain than embedding the

higher-level language control constructs. We also had a general objection

to soft machines: with them, the guidance necessary for clean moves from a

central group to a number of small software groups. Moreover, it

jeopardizes the ability to have communication between programs that are

written in different languages.

A capabilities-based architecture was rejected because we did not fully

udnerstand it and because there was no performance or reliability data

available from the few experimental machines which have been built.

Out

8. **FUTURE PLANS AND DIRECTIONS**   *tar.*

The problems encountered on the PDP-11 project are not peculiar to that

machine, or to any machine or style of architecture. In the course of the

project, we have isolated several specific problems in computer design. We

intend to explore each of them further.

8.1.   THE BUS SPECIFICATION PROBLEM

It has taken a long time to understand the UNIBUS in terms of its

electrical, performance, and logical capabilities. The existing bus

specifications, however inadequate, are the result of many iterations of

respecification based on experience and redesign. Several description

techniques have been tried: timing diagrams, threaded diagrams showing the

cause and effect of signals, and partial state flowcharts showing state in

master and slave components. A rigorous specification language, such as

BNF, would be helpful. BNF has proven helpful in the specification of

communication links, but is too clumsy for general use, and is not widely

understood by engineers and programmers.

The most important use of a rigorous bus specification is the testing of

faulty components rather than the exercising of good ones. A bus

specification would provide a behavior standard against which to check

faulty components. It is not clear how one should best attach the problem

of bus behavior specification. A safe place to start would be an

exhaustive set of examples.

8.2.   CHARACTERIZING COMPUTATION PROBLEMS

When a user comes to us with a task needing computerization, we don't have
a good way to describe the computational needs of the task. The needs are
multidimensional, consisting of the procedural algorithms, the file
structure, the interface transducers, reliability, cost, and development
deadline. This communications difficulty exists between computer designers
and operating-system designers as much as between computer designers and
end users.

Even when there is a good way to specify to the system designer exactly
what the user's computational needs might be, there is still a lot of work
in finding an architecture to best solve that problem and finding an
implementation to best build that architecture.

8.3 OPERATING SYSTEMS

A taxonomy and notation is needed to describe the functions of a system,
especially the operating systems. There is no good methodology for talking
about tradeoffs, because the functions and structure of a system are so
vague.

There exist numerous operating systems for the PDP-11. One of the reasons
for this situation is that there is no easy way to compare an existing
system with a design for a new one. Instead, an engineering-marketing
conspiracy invents a new system because it is oriented toward a particular
market in some nebulous way. If we had the ability to specify operating
system behavior in a uniform and comprehensible way, then a system could be
analyzed before it is programmed.

8.4 PROBLEMS WITH ARCHITECTURAL RANGE

In a growing family of computers, the designer is constantly faced with the
question of whether or not to build a certain model or provide an certain
point on the price/performance curve. The decision is colored by
technology, user requirements, competitor offerings, and available design
staff. It is difficult to answer precisely even a question so simple as
whether to build two models that are close together (as the 11/40 and
11/45), or to make a single model and expand it with a multiprocessor
option.

The range problem occurs at other levels. Consider memory. The number of
memory technologies available is growing constantly, and the once-clear
boundaries between memory classes based on memory speed are blurring. Some
of the new electronic-based technologies such as CCD and magnetic bubbles
have an access time in the 100-microsecond range, and fill the gap between
traditional random-access memories (.1 to 1 microsecond) and
electromechanical memories like disks or drums (1 millisecond to 100
millisecond). The system designer must decide how much of which kinds of
memory will be used in each implementation. It may well be that a solution
to problems of this sort will be dependent on the ability to characterize
the computational needs.

9. SUMMARY

In this paper we have reexamined the PDP-11 in the light of ~~six years of~~
experience, and have compared its successes and failures with the goals and

----

problems of the initial ideas. With the clarity of hindsight, we now see

the initial design problems. Many mistakes were made through ignorance,

and many more because the design work was started too late. As we continue

to evolve and improve the PDP-11 computer over the next ~~five years~~ generations, it will

indeed be interesting to observe whether the PDP-11 can continue to be a

significant, cost-effective minicomputer. We believe it can. The ultimate

test is its use.

ACKNOWLEDGMENT

REFERENCES

Almes, G. T., Drongowski, P. J., and Fuller, S. H., Emulating the Nova on the PDP-11/40: a case study. Proc. COMPCON, Washington, D. C., September 1975.

Bell, C. 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 36, 657-675 (1970).

Bell, C. G., and Newell, A., Computer Structures. McGraw-Hill, New York, 1971.

Eckhouse, R. H., Minicomputer Systems: Organization and Programming (PDP-11). Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

Gear, C. W., Computer Organization and Programming, Second Edition, McGraw-Hill, New York, 1974.

McWilliams, T., Sherwood, W., and Fuller, S., PDP-11 implementation using the Intel 3000 microprocessor chips. Proc. NCC 46, 243-253 (1977).

O'Loughlin, J. F., Microprogramming a fixed architecture machine.

Microprogramming and Systems Architecture Infotech State of the Art Rep.

23, 205-244 (1975).

Ornstein, S. M., Heart, F. E., Crowther, W. R., Rising, H. K., Russell, S.

B., and Michael, A., The terminal IMP for the ARPA computer network. Proc.

SJCC 40, 243-254 (1972).

Shannon, C. E., A mathematical theory of communication. Bell Sys. Tech. J.

27, 279-423, 623-656 (1948).

Stone, H. S., and Siewiorek, D. P., Introduction to Computer Organization

and Data Structures: PDP-11 Edition. McGraw-Hill, New York, 1975.

Wulf, W. A., and Bell, C. G., C.mmp: a multi-mini-processor, Proc. FJCC 41,

765-778 (1972).