Architectural Evolution in DEC's 18b Computers

Bob Supnik, revised 08-Oct-2006

Abstract

DEC built five 18b computer systems: the PDP-1, PDP-4, PDP-7, PDP-9, and PDP-15. This paper documents the architectural changes that occurred over the lifetime of the 18b systems and analyses the benefits and tradeoffs of the changes made.

Introduction

From 1961 to 1975, Digital Equipment Corporation (DEC) built five 18b computer systems: the PDP-1, PDP-4, PDP-7, PDP-9, and PDP-15 (see table below). Each system differed from its predecessors, sometimes in major ways representing significant architectural breaks, and sometimes in minor ways representing new features or incompatibilities. The architectural evolution of these systems demonstrates how DEC's ideas about architectural versus implementation complexity, I/O structures, and system features evolved over the period of a decade.

	PDP-1	PDP-4	PDP-7	PDP-9	PDP-15
First ship	Nov 1960	Jul 1962	Dec 1964	Aug 1966	May 1970
Number built	50	45	120	445	790
Memory cycle	5usec	8usec	1.75usec	1usec	0.8usec
Base price	$120K	$65.5K	$45K	$25K	$19.8K

Reproduced from Computer Engineering: A DEC View Of Hardware Systems Design

The PDP-1

The PDP-1 was DEC's first computer system. Introduced in 1960, the PDP-1 reflected ideas from Lincoln Labs' TX-2 project as well as the existing capabilities of DEC's module logic family. It was implemented in 5Mhz logic.

Arithmetic System

The PDP-1 was a 1's complement arithmetic machine. In 1's complement arithmetic, negative numbers are represented by the bit-for-bit inversion of their positive counterparts:

```
+1    =          000001
-1    =          777776

+4    =          000004
-4    =          777773
```

One's complement arithmetic has two problems. First, zero has two representations, +0 and -0:

```
+0    =          000000
-0    =          777777
```

Second, addition of negative numbers requires an “end around carry” from the high order position to the low order position:

```
-1    =             777776
-1    =             777776
--                 ---------
sum                1 777774
                   |----->1
-2    =             777775
```

The PDP-1 tried to solve the zero-representation problem by guaranteeing that arithmetic operations never produced –0. To do this, it performed an extra logic step during addition, checking the result for –0 and converting it to 0. However, the PDP-1 performed subtraction by complementing the AC, adding the memory operand, and recomplementing the result. The recomplementation step occurred in the same time slot as the –0 detect during add. As a result, subtract had one special case: -0 – (+0) yielded –0.

Character Sets

The PDP-1’s first console typewriter was a Friden Flexowriter. (Production units used a Soroban typewriter, which was a modified IBM Model B.) The console’s six bit character set was called FIODEC, which stood for **F**riden **I**nput **O**utput for **D**igital **E**quipment **C**orporation. This code included both upper and lower case letters, using shift characters to move between sets. The console echoed characters locally, a feature that would be retained in all later 18b systems (and no other DEC systems).

The PDP-1’s line printer used Hollerith (BCD) coding. FIODEC and Hollerith had common encodings for letters but not for symbols, requiring character conversions throughout the software.

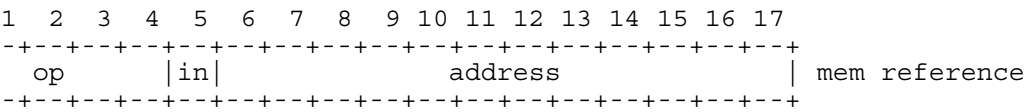
Instruction Set Architecture

The PDP-1’s visible state included the following registers and capabilities:

AC<0:17>	accumulator
IO<0:17>	I/O register
OV	overflow flag
PC<0:11>	program counter
EPC<0:3>	extended program counter (if memory > 4K)
EXTM	extend mode
PF<1:6>	program flags
SS<1:6>	sense switches
TW<0:17>	test word (front panel switches)
IOSTA<0:17>	I/O status

In addition, the PDP-1 had non-observable state in the I/O system for I/O timing (see below).

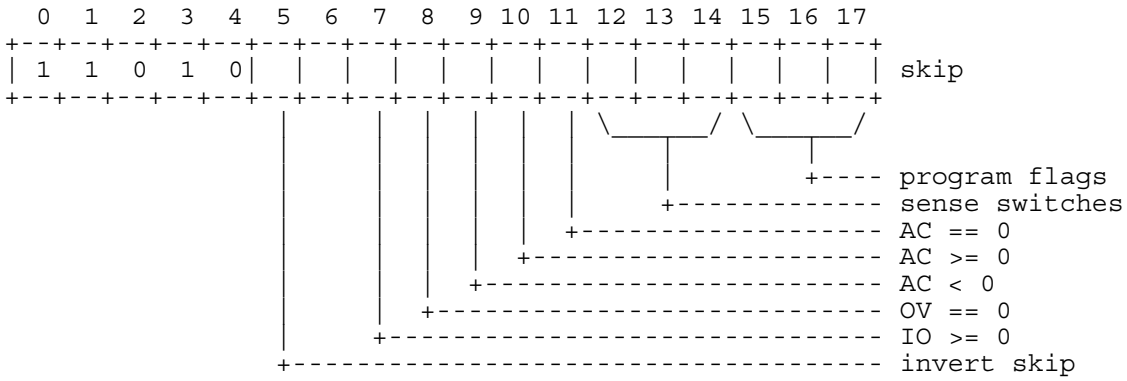
The PDP-1 had 32 opcodes and implemented six instruction formats: memory reference, skip, shift, operate, I/O, and load immediate. The memory reference format was:



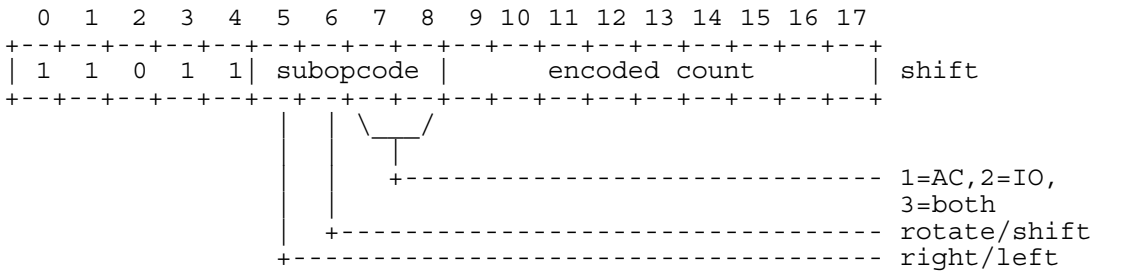

```
<0:4> <5>   mnemonic    action
```

```
00
02          AND         AC = AC & M[MA]
04          IOR         AC = AC | M[MA]
06          XOR         AC = AC ^ M[MA]
10          XCT         M[MA] is executed as an instruction
12
14
16     0    CAL         M[100] = AC, AC = PC, PC = 101
16     1    JDA         M[MA] = AC, AC = PC, PC = MA + 1
20          LAC         AC = M[MA]
22          LIO         IO = M[MA]
24          DAC         M[MA] = AC
26          DAP         M[MA]<6:17> = AC<6:17>
30          DIP         M[MA]<0:5> = AC<0:5>
32          DIO         M[MA] = IO
34          DZM         M[MA] = 0
36
40          ADD         AC = AC + M[MA]
42          SUB         AC = AC - M[MA]
44          IDX         AC = M[MA] = M[MA] + 1
46          ISP         AC = M[MA] = M[MA] + 1, skip if AC >= 0
50          SAD         skip if AC != M[MA]
52          SAS         skip if AC == M[MA]
54          MUL         AC'IO = AC * M[MA]
56          DIV         AC, IO = AC'IO / M[MA]
60          JMP         PC = MA
62          JSP         AC = PC, PC = MA
```

The skip format was:

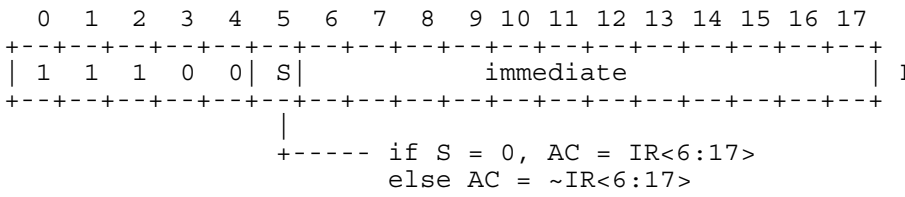


The shift format was:

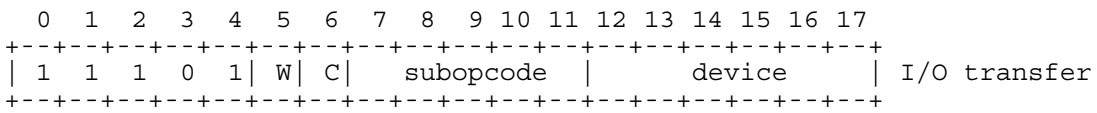


The shift count was the number of 1’s in bits <9:17>.

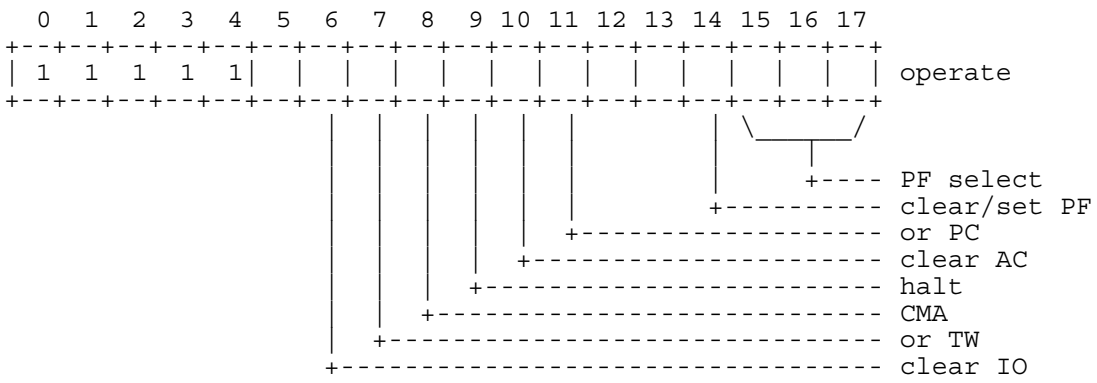
The load immediate format was:



The I/O transfer format was:



The operate format was:



There are significant discrepancies in the PDP-1 documentation about memory extension options. The original 1960 User Handbook (F15) didn't mention any. The 1961 Handbook (F15B) described two, the Type 13 and Type 14. The 1962 and 1963 Handbooks (F15C and F15D, respectively), and the Maintenance Manual, described only one, the Type 15. This option expanded memory to 64K words. The address space was divided into sixteen 4K word fields. An instruction could directly address, via its 12b address, the entire current field. If extend mode was off, indirect addresses accessed the current field, and multi-level indirect addressing was enabled; if on, indirect addresses could access all 64K, and indirect addressing was single level. The state of extend mode was captured by subroutine calls and sequence breaks, and extend mode was cleared at the start of a sequence break.

BBN built a custom memory manager for its PDP-1 timesharing system. There are also scattered referenced to a PDP-1D, built by DEC itself for timesharing. Gordon Bell believes two were built, one for BBN and one for Stanford.

I/O System

The PDP-1's I/O system offered multiple modes for I/O instructions, including synchronous waiting, timed waiting, asynchronous, and sequence break (interrupt) driven. This multiplicity made the I/O system complex and redundant.

I/O operations were initiated by a single instruction, Input/Output Transfer (IOT). Bits<12:17> addressed a particular device; bits <7:11> provided additional control or opcode bits. Bits<5:6> specified the mode for the I/O transfer:

<5:6> mode

00	asynchronous - no wait, no device completion pulse
01	timed wait - no wait, device completion pulse
10	synchronous - wait for completion
11	not used - wait, no completion pulse (hung the system if <12:17> != 0)

In synchronous wait, the CPU effectively stalled until the I/O operation completed. If synchronous wait was not specified, three different mechanisms were available for I/O completion:

- Timed wait. Execution proceeded. Eventually, the CPU issued a wait instruction. The CPU then stalled until the I/O operation completed and the device issued a completion pulse.
- Polled wait. Execution proceeded. The CPU monitored the device's flag in the I/O status word until the I/O operation completed.
- Sequence break driven. Execution proceeded. When the I/O operation completed, a sequence break (interrupt) occurred, signaling I/O done.

The IOT wait mechanism was implemented with four control flip-flops:

- IOC (I/O command): when asserted, allowed IOT pulses to be sent to a device; when clear, IOT was effectively a NOP.
- IOH (I/O halt): when asserted, stalled the CPU by re-executing the current instruction.
- IHS (I/O halt save): saved the state of IOH on a no-wait IOT.
- IOS (I/O synchronization): when asserted, terminated I/O wait state.

An IOT that specified wait would set IOH, execute the IOT, and, if IOS was clear, clear IOC and decrement the PC. Thus, subsequent re-executions of the IOT would do nothing, because IOC was not asserted. When the I/O operation completed, the device would set IOS. This caused the IOT to set IOC and not decrement the PC, allowing execution to proceed.

An IOT that did not specify wait copied IOS to IHS, set IOC, executed the IOT, and copied IHS back to IOH. If IOH was set as a result of the copy, IOC was cleared. This implemented a one-level memory for wait state. If an IOT with wait was interrupted, the interrupt routine could execute no-wait IOT's while preserving wait state for the main line program.

The sequence break mechanism recorded break requests in a single pulse sensitive flip flop. Thus, like the PDP-11 but unlike the other 18b systems, break requests were independent of the device completion flags. If the sequence break system was enabled, and a break request occurred, the CPU automatically stored the state of the machine and initiated a new program by:

- storing AC in location 0
- storing EPC and PC, plus overflow and extend mode, in location 1
- storing IO in location 2
- clearing overflow and extend mode
- setting the PC to 3
- setting the sequence break in progress flag

The sequence break in progress flag blocked further breaks.

The end of the break was recognized when the CPU decoded a JMP I 1 (from field 0 in a multi-field system) while the sequence break system was enabled. At that point, the CPU automatically restored the state of the system by:

- temporarily turning on extend mode
- obtaining the new PC from location 1
- restoring the original values of overflow and extend mode

- clearing sequence-break-in-progress

A CPU option expanded the standard sequence break system from one channel to sixteen. Each channel was a unique priority level and had a dedicated four location memory block (0 – 3 for the highest priority channel, 4 – 7 for the next, etc.). The first three locations of the block were used to store AC, PC, and IO when a break occurred; the PC was then set to point to the fourth location.

Software

The PDP-1 featured some notable software offerings, including an interactive editor (called Expensive Typewriter), a macro assembler (Macro), a symbolic debugger (DDT), a Lisp interpreter, and the world's first computer video game, Spacewar. Sources to Lisp and Spacewar are available on the Internet, and source listings for Macro and DDT are in the Computer History Museum collections.

The PDP-4

The PDP-4 was intended to be substantially lower cost than the PDP-1. Part of the cost reduction was achieved by using slower and less expensive logic (500Khz instead of 5Mhz), but part was achieved by simplifying the system and reducing the number of gates. Thus, the PDP-4 (and its closely related successors, the PDP-7 and PDP-9) simplified the architecture of the PDP-1 along multiple dimensions.

Arithmetic Systems

The PDP-4 introduced two's complement arithmetic in parallel with the PDP-1's one's complement arithmetic. Two's complement arithmetic eliminated the need for -0 detection and made implementation of multi-precision arithmetic much easier. However, 1's complement capability was not dropped; indeed, it remained the predominant arithmetic system, as reflected in architectural extensions such as the EAE. Thus, the PDP-4 still needed end around carry propagation, as well as 1's complement overflow detection. The result was greater, rather than lesser complexity, in the hardware, and loss of valuable opcode space in the architecture. Gordon Bell commented that the retention of 1's complement arithmetic was, simply, "a mistake". By the PDP-5, it had vanished from DEC's architectures.

Character Sets

The PDP-4's console typewriter was an ASR-28 Teletype. Its five bit character code was called Baudot. It supported only upper case letters and required shift characters to get from letters to figures and back again. The line printer was unchanged and continued to use Hollerith coding.

Instruction Set Architecture

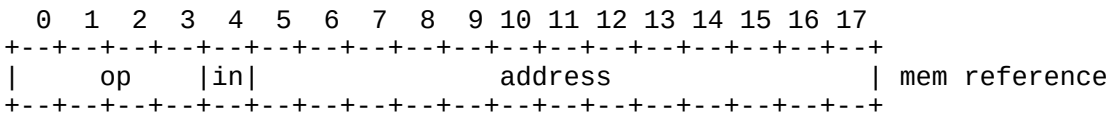
The PDP-4 and its successors reduced the amount of visible state in the CPU. Specifically,

register	PDP-1	PDP-4,-7,-9
AC	arithmetic register	same, plus I/O register
IO	I/O register	removed (MQ with EAE option)
OV	overflow indicator	replaced by Link register
PF	program flags	removed

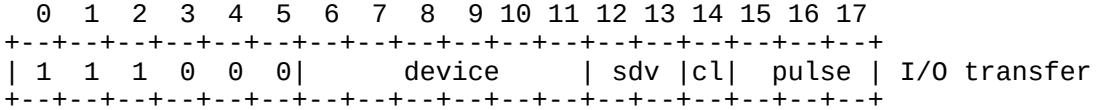
SS	sense switches	removed
TW	test word	front panel switches
EXTM	extend mode	same
IOSTA	IO flags	same

The register changes simplified the logic implementation. The L was essentially the 19th bit of the AC, rather than a special flag. The AC no longer implemented -0 detection. I/O now used the existing access paths to the AC rather than separate paths to an IO register. The elimination of the program flags, and the sense switches, was pure gain.

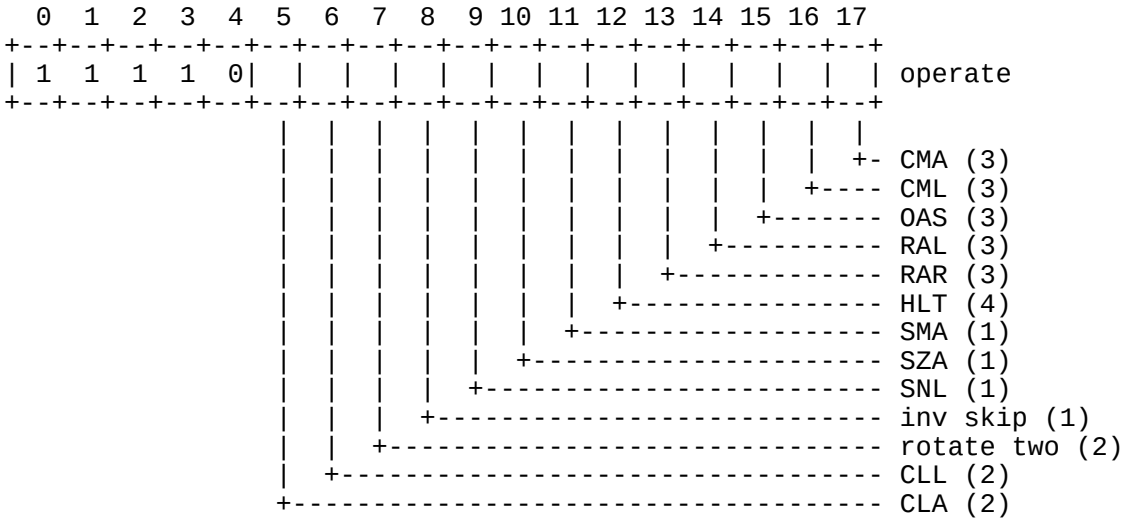
The PDP-4 halved the number of instructions, from 32 to 16, and reduced the number of instruction formats from 6 to 4. The memory reference format was:



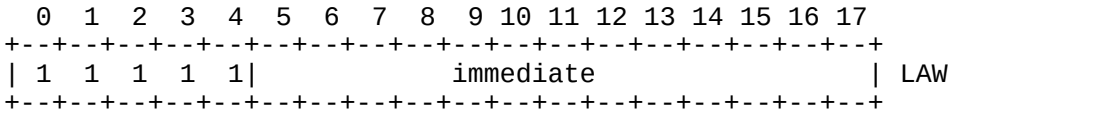
The I/O transfer format was:



The operate format was:



The immediate format was:



The following table shows the reduction in instruction count between the PDP-1 and the PDP-4:

PDP-1 instruction	PDP-4 instruction
AND	AND

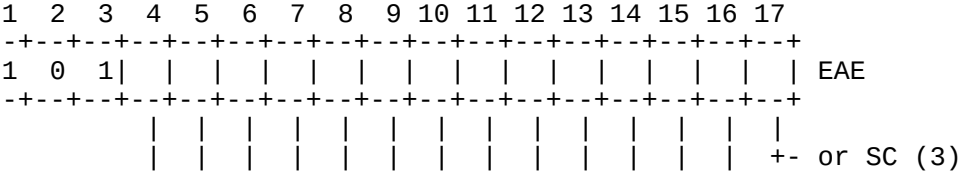
IOR	removed
XOR	XOR
LAC	LAC
DAC	DAC
DZM	DZM
DIP	removed
DAP	removed
LIO	removed
DIO	removed
ADD	ADD; L used in place of overflow
SUB	removed
MUL	(EAE option)
DIV	(EAE option)
not present	TAD (2's complement add)
IDX	removed
ISP	ISZ
XCT	XCT
SAD	SAD
SAS	removed
CAL	CAL
JDA	JMS
JSP	removed
JMP	JMP
skips	OPR skips
operate	OPR operates
shifts	(EAE option)
LAW	LAW
IOT	IOT

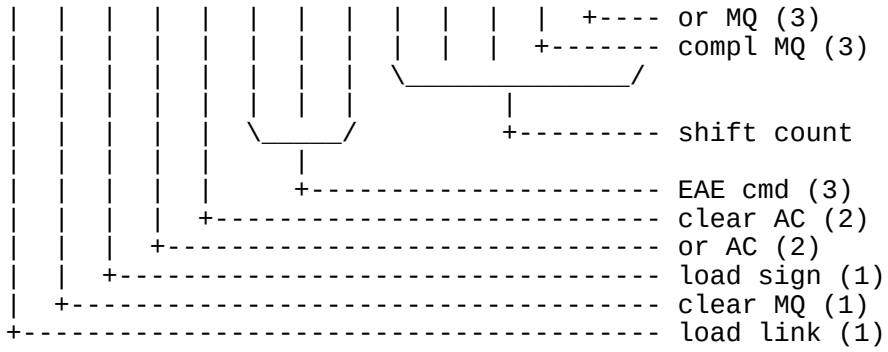
Beyond the reduction in instruction count, the PDP-4's instruction set required less logic to implement.

- Instructions were encoded to minimize logic. For example, all instructions with IR<0:1> = 00 (CAL, DAC, JMS, DZM) did not read a memory operand. All instructions with IR<0:1> = 11 (JMP, EAE, IOT, OPR/LAW) were single cycle.
- ISZ (replacing IDX and ISP) did not modify the AC. By using 2's complement arithmetic, it did not need to detect -0.
- JMS (replacing JDA and JSP) did not modify the AC. This eliminated the transfer path from the PC to the AC. JMS (and interrupts) saved PC and L, and in later systems, the memory extend and memory protection flags.
- LAW did not mask or modify the address but instead copied the entire instruction to AC.
- OPR no longer guaranteed conflict-free execution of any combination of bits.

Finally, indirect addressing was simplified by the elimination of multi-level indirection.

The PDP-4 replaced the PDP-1's multiply, divide, and multi-bit shifts with an option, the Extended Arithmetic Element (EAE). The EAE added a second 18b arithmetic register, the MQ, and a shift/multiply/divide instruction. The EAE instruction was microprogrammed and could implement a wide variety of unsigned and signed (one's complement) operations:





The EAE architecture remained unchanged in the PDP-7, PDP-9 and PDP-15.

The PDP-4 included an extended addressing option (Type 16); two of the surviving PDP-4's in the 1972 census have more than 8KW of memory. No documentation has yet been found on this option, but it's reasonable to assume that it was the same as the PDP-7's. If that is true, the PDP-4's extended memory model was essentially the same as the PDP-1's, with 13 direct address bits instead of 12. Addressable memory was divided into four 32K word banks. Direct addresses always referenced the current memory bank; indirect addresses accessed either the current memory bank or all of memory, depending on the extend mode flag. As on the PDP-1, subroutine calls and interrupts saved the state of extend mode automatically.

In all, the architectural tradeoffs in the PDP-4 substantially reduced control logic at the cost of complete software incompatibility with the PDP-1. There were also a few oversights; in particular, the lack of a "complement and increment" operate (present in the PDP-5) made two's complement subtract an instruction longer. The PDP-15 finally corrected this oversight.

The PDP-4 (and the PDP-5) introduced a new feature, the concept of "auto-index" memory locations, that is, locations which, when used as indirect addresses, incremented before use. This feature allowed efficient traversal of linear data structures and made the IDX and DAP instructions unnecessary.

I/O System

The I/O system was pruned even more dramatically than the CPU. Synchronous waits and timed waits were dropped. Instead, only two mechanisms were supported: polled waits and interrupts. Further, the two mechanisms were integrated by having the device flag for polling be the triggering mechanism for device interrupts. Finally, polled waiting was implemented more efficiently by allowing devices to increment the PC (skip) in response to an IO instruction. The PDP-5 also used this I/O paradigm, and it was retained throughout the life of the 12b and 18b families.

In the PDP-4, an ideal I/O device had one flag representing the state of an I/O operation. This flag was cleared when the device initiated I/O; it was set when the device completed I/O. For example, in the paper tape reader, the reader flag was cleared by a request to read a character or by explicit command, and set when the character was in the I/O buffer.

Interrupts (as sequence breaks were now called) were simplified, and control was made explicit rather than implicit.

Function	PDP-1	PDP-4
interrupt request	request flip-flop	logical or of device flags
interrupt block	request in progress flop	interrupts turned off

interrupt action	save AC	--
	save PC + flags	save PC + flags
	save IO	--
	clear OV	--
	clear extend mode	clear extend mode
	set break in progress	turn off interrupts
	set PC = 3	set PC = 1
interrupt complete	monitor for JMP I 1	turn on interrupts, one cycle delay to allow for JMP I 0

The PDP-4 offered a multi-level interrupt option. As in the PDP-1, each interrupt vectored to a unique memory block. Unlike the PDP-1, the memory block was a single location, which was executed. If the location contained a JMS, control transferred to an interrupt service routine. If the location contained any other instruction, the instruction was executed, but control returned to the main line program.

Software

Because the PDP-4 was not compatible with the PDP-1, it required new software. DEC provided an editor, an assembler, and, most notably, a Fortran II compiler, all paper-tape based. While the Fortran compiler was a significant advance, the assembler was actually a step backward: the PDP-1's assembler had supported macros, the PDP-4's did not. But it offered some consolation by being a one pass assembler, obviating the need to read the source paper tape twice. The assembler assumed that unresolved references would in fact be resolved and punched unresolved binary code as it processed the source, with a resolution dictionary at the end of the output tape. The resulting tape was then read, upside down and backward, by the loader, which used the resolution dictionary to "fix up" the broken references in the binary.

The PDP-4's programs later became the basis for the PDP-7's software offerings, which accounts for lingering use of Baudot code on the PDP-7. However, the presence of FIODEC on the PDP-4 (and thus on the PDP-7) is a mystery, since the PDP-1 software base was not carried forward.

Early Mass Storage

The PDP-1 and PDP-4 started out as paper tape based systems. The development software was paper tape based; magnetic tape, if used at all, was used strictly for data. This situation was clearly unsatisfactory, and by 1963 DEC was experimenting with mass storage.

The first mass storage products were based on Vermont Research Drums. The Type 23 parallel and Type 24 serial drums offered 131,072 words of storage with rapid access. But the drums were big (two six-foot cabinets for the Type 23, one for the Type 24), expensive, and inflexible: storage was tied to the computer. This didn't fit with the typical use of the 18b computers as "personal" or serially shared systems.

To find a solution, DEC again turned to Lincoln Labs. In 1962, Wes Clark had demonstrated the prototype of the LINC computer. It featured LINCtape, a block-replaceable tape system with a simple, rugged transport and small, inexpensive tape reels. LINCtape offered exactly the kind of "personal" storage needed to complement DEC's computers. With some changes in tape format, DEC offered "MicroTape" (later renamed DECtape) on the PDP-1 and PDP-4 in 1963. The product also included a stand-alone program librarian, Microtrieve. DECtape was to remain the

dominant form of mass storage on DEC's 12b and 18b systems into the early 1970's, when it was supplanted by the RK05 (2315-style) cartridge disk drive.

The PDP-7

According to the history of the 18b series in Computer Architecture, the PDP-4 was not a success. The use of slower logic yielded a system that was 5/8 the performance of the PDP-1 at ½ the price. What the market required was a system that was both higher performance and lower cost. That system was the PDP-7. Implemented (primarily) in 10Mhz logic, its basic 1.75 usec cycle time was almost three times the speed of the PDP-1, at 1/3 the cost.

The PDP-7's basic architecture consisted of minor refinements of the PDP-4's instruction set, accompanied by one interesting architectural extension: multi-user protection, the first in the 18b family. The PDP-7 also was the first 18b PDP to use ASCII coding.

Arithmetic Systems and Character Sets

The PDP-7's arithmetic systems were identical to the PDP-4. The console typewriter was an ASR-33 Teletype. Its eight-bit character set was an early version of ASCII, with the high order bit always forced on. The character set supported both upper and lower case letters, although the console only supported upper case. The line printer's SIXBIT character set was derived from ASCII by truncating codes 040 - 0137 to six bits. The rapid evolution of character sets in the 18b family was embodied in the PDP-7's DECtape-based operating system DECsys. DECsys stored information in FIODEC, Baudot, and SIXBIT, depending on whether the underlying software was derived from the PDP-4 or newly written.

Instruction Set Architecture and I/O System

The PDP-7 used the same instruction set architecture as the PDP-4, including the EAE. The extended memory model was the same as the PDP-4's. The PDP-7's I/O architecture was identical to the PDP-4's, and it used the same controllers for major I/O devices such as DECtape, magnetic tape, and the serial drum. A few new IOT's were added, for management of the trap system. The PDP-7 featured an interprocessor link; this device set the model for the general purpose parallel I/O options in subsequent DEC computers. Like the PDP-1 (but unlike the PDP-4), the PDP-7 console featured a "read-in" switch, to automate system bootstrapping from paper tape. The "read-in" function did not use the PDP-4's RIM format but instead loaded memory sequentially from the tape. Therefore, loading software required three steps: use the "read-in" switch to load the RIM loader; use the RIM loader to load the binary loader; and finally use the binary loader to load the software.

Memory Management

The PDP-7 implemented a primitive form of multi-user protection called trap mode. If trapping was enabled, IOT's and HLT became privileged instructions. If extend mode was simultaneously disabled, indirect addresses were confined to the current bank. This allowed for simple time-sharing, with each user in a separate memory bank. (An option, the KA70A, added a small bounds control register to protect memory within a bank.)

Software

The PDP-7 offered the 18b product line's first mass-storage operating system, the DECtape-based DECsys. (DECsys also ran on the PDP-4.) DECsys was a modest first step in operating

system development. It consisted of a simple memory-resident DECtape I/O library, a keyboard monitor, a Fortran II compiler, an assembler, a linking loader, and a symbolic debugger. All of the components were based on PDP-4 and PDP-7 paper-tape counterparts, with calls to the DECtape I/O library replacing paper-tape I/O. The internals of DECsys reflect its heterogeneous origins, with directory information stored in Baudot (3 six-bit characters per word, each character consisting of a 5b Baudot code plus a 1b shift flag) and source files in FIODEC.

A DECsys system tape contained a tape label in block 1, the system directory in block 2, the library directory in block 3, and the keyboard monitor in blocks 4-6. The first word of the system directory contained the directory length; the last word contained the address of the first free block on the tape. Directory entries consisted of 5 or 6 words:

Word 1:	Type (1 for **S**ystem, 2 for **W**orking)
Words 2-3:	File name, in Baudot
S, word 4:	starting block on tape
S, word 5:	starting address in memory
W, word 4:	starting block on tape for F (Fortran) version
W, word 5:	starting block on tape for A (assembler) version
W, word 6:	starting block on tape for R (relocatable binary) version

Files were simply linked DECtape blocks, with the first word of a block pointing to the next; a pointer of 0 signified end of file.

The first word of the library directory contained the directory length in words. Directory entries were variable length, depending on the number of entry points in the routine:

Words 1-2:	entry name, in Baudot
Words 3-4:	second entry name (if any, in Baudot)
:	
Word 2n+1:	777777, marking end of entry names
Word 2n+2:	starting block for the library
Word 2n+3:	777777, end of ldirectory entry

After many unsuccessful years of searching, a copy of DECsys was found with the last functioning PDP-7 on the planet, Professor Harlan Lefevre's system at the University of Oregon (soon to reside at Paul Allen's PDP Planet in Seattle). But so far, no copies have been found of an even more historic system for the PDP-7, UNIX. The PDP-7's multi-user protection, crude as it was, sufficed for implementation of the first version of UNIX, making the PDP-7 a significant system in the history of computing. Unfortunately, all copies of UNIX for the PDP-7 have been lost. Some details of the PDP-7 version can be found on Dennis Ritchie's personal web site.

The PDP-9

The PDP-7 was considerably more successful than its predecessors, selling more than 100 systems thanks to its significant price/performance improvements. The PDP-9 was intended to carry the line forward. The arithmetic system and character sets were unchanged, and the instruction set and I/O architecture changed only minimally. The I/O subsystem changed from a radial to a bus design, necessitating redesign of all peripherals. Interfaces to programmed I/O peripherals (paper tape, console, line printer) remained basically the same as the PDP-4 and PDP-7; however, interfaces to mass storage peripherals (magnetic tape, DECtape) changed significantly. An entirely new multi-level interrupt option, called the Automatic Priority Interrupt (API), was designed. The PDP-9 carried over little of the PDP-7's admittedly small software base.

Instruction Set Architecture and I/O System

Although intended to be upward compatible with the PDP-7, the PDP-9 introduced a number of differences:

- Auto-indexing. In the PDP-7, each bank of memory had auto-index registers. In the PDP-9, only bank 0 had auto-index registers, and indirect references through addresses 00010-00017 were forced to reference bank 0.
- Extend mode restore. The PDP-7 used EMIR to prepare the system to restore extend mode at the end of an interrupt. The PDP-9 introduced the more ambitious RES, which prepared the system to restore the link, extend mode, and memory protect mode. This removed two instructions from the end of all interrupt routines.
- Extend mode behavior. The PDP-7 set extend mode on a protection trap but cleared it on an interrupt; the PDP-9 cleared it on both. The PDP-7 performed a modified JMS, storing the program state in location 0 but taking the next instruction from location 2; the PDP-9 performed a JMS 0 or JMS 20, depending on whether interrupts were on or off.

The PDP-9's I/O architecture contained some modest improvements in flexibility and error detection. Status flags were added for reader and punch errors. The line printer controller implemented a device-specific interrupt enable/disable. The new DECtape, magnetic tape, and fixed head disk controllers implemented better programming models than their PDP-7 counterparts, and used up fewer device numbers in the process.

The PDP-9 also implemented an entirely new design for multi-level interrupts. Called the Automatic Priority Interrupt (API) option, the API separated the concept of interrupt channel from priority. The API option supported 32 channels (interrupting devices), but the channels were grouped into eight priority levels. Four channels, on the four lowest priorities, were reserved for software interrupts. When an API break occurred, the memory location corresponding to the channel was executed. The location had to contain a JMS to an interrupt service routine; use of other instructions was not supported. The API was carried over unchanged to the PDP-15.

Memory Management

The PDP-9 introduced a more flexible form of memory management, with a bounds register separating system (lower) memory from user (upper) memory. The PDP-7's trap flag became the PDP-9's user mode flag. Memory management was partially integrated with the API subsystem: the CAL instruction, automatically activated API 4, effectively locking out software interrupts.

Software

The PDP-9's close compatibility with the PDP-7 allowed the latter's software to be brought forward. However, that code base, dating from the PDP-4, was considered inadequate and relegated to use in the smallest systems. For mainstream use, a new software suite was written from scratch. The three-step software loading process was simplified by eliminating the intermediate RIM loader. The hodge-podge of I/O routines and libraries was replaced by a standard I/O executive that maintained compatible interfaces from the paper-tape environment through the mass-storage based operating systems (Advanced Monitor System, its foreground/background extension, and DOS). The PDP-4/7 assembler syntax and binary formats were scrapped and replaced with a new macro assembler, Macro 9. Fortran II was replaced by Fortran IV. The intent versus the practice for PDP-9 software is illustrated by the changes in the manual set. The examples in the Systems Reference Manual all follow PDP-7 assembler syntax, but most surviving software is written in Macro 9.

The PDP-15

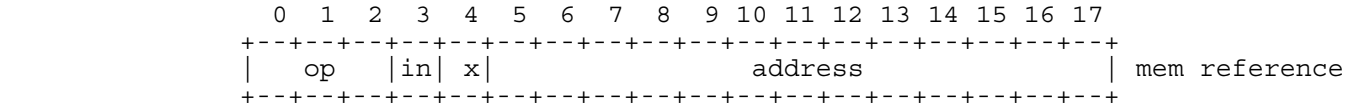
The PDP-15 introduced the most significant set of architectural changes in the 18b product line since the transition from the PDP-1 to the PDP-4. It represented a major technology shift, from discrete transistors to TTL integrated circuits. The PDP-15 was the fastest and most popular 18b computer in Digital's history. It was also the last.

Instruction Set Architecture and I/O System

The PDP-15 introduced three architectural extensions:

- two new registers, an 18b index register and a 9b limit register
- extended addressing to 128K words
- hardware floating point

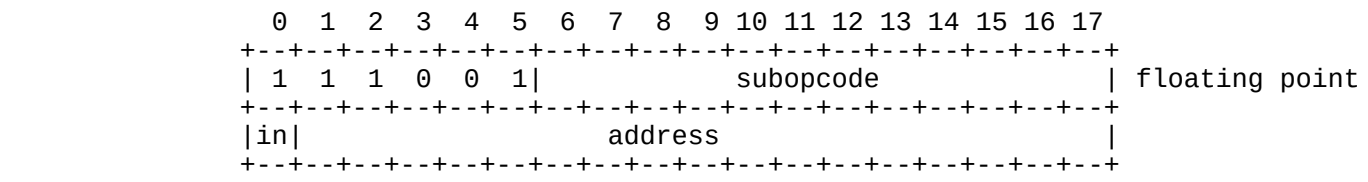
The introduction of the index register made the PDP-15 more competitive with contemporary machines such as the SDS 940 and DDP 516, both of which had indexing. To get an index register select into the memory reference instructions, the directly addressable memory range was reduced from 8K to 4K:



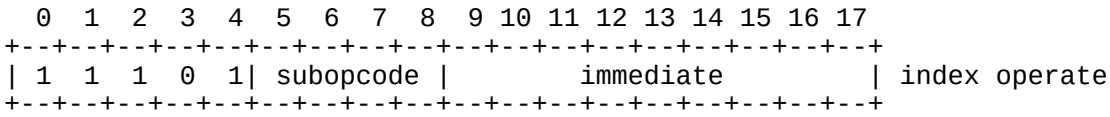
Direct addressing beyond 4K words could be done by indirect addressing (maximum 32K words), or by indexing (maximum 128K words). However, until the XVM memory management option was introduced, return addresses remained limited to 15b; thus the maximum practical code segment size remained 32K words. Extended memory worked best with the new memory relocation and protection option; in that environment, multiple 32K word programs could reside in memory simultaneously.

The addition of indexing created a serious compatibility problem with the PDP-9. To ameliorate migration issues, the PDP-15 redefined the PDP-7's and PDP-9's extend mode flag as PDP-9 compatibility mode, or bank mode. If bank mode was enabled, memory reference decoding was identical to the PDP-9, without index capability. The PDP-15 did not implement the PDP-9's extend mode capability within bank mode, because extend mode, which was a compatibility aid for PDP-4 and PDP-7 programs, was no longer needed.

The hardware floating point unit was another new addition to the architecture. It dramatically improved the performance of the system in scientific applications. To support indexing and floating point, the PDP-15 introduced two new instructions, both carved out of the IOT instruction. Bits <4:5> of the IOT instruction had been defined as sub-device selects but in practice were unused. The PDP-15 used them to differentiate between IOT instructions (<4:5> = 00), floating point instructions (<4:5> = 01),



and index operate instructions (<4:5> = 1x):



In addition to the major changes outlined above, the PDP-15 had its own set of tweaks and incompatibilities compared to its predecessor:

- Two meaningless operates were redefined as IAC (increment AC) and BSW (byte swap). The former facilitated a one-instruction 2's complement, thereby correcting a hole in the arithmetic system.
- On the PDP-9, DBR and RES were triggered by a JMP indirect, on the PDP-15 by any indirect.
- The PDP-15 implemented new IOT skips for bank mode, and redefined the SKP7 IOT to test whether the high-speed reader was present in the configuration.
- A mid-life ECO (called the "re-entrancy ECO") added two additional IOT's to inhibit and enable interrupts. Also, in monitor mode it suppressed interrupts for one instruction following a PI, JMS, or CAL (two after a NORM).
- The PDP-15 EAE did not require that the link be cleared for IMUL and IDIV.
- The PDP-15 API placed the program interrupt priority between the API hardware and software interrupts, rather than below the software interrupts.
- The PDP-15 CAL instruction only set API 4 if API's 0-3 were inactive, rather than unconditionally.

From a programming viewpoint, the PDP-15's I/O architecture was the same as the PDP-9's, but the implementations were quite different. The PDP-15 implemented a separate I/O processor, providing greater expandability and flexibility, and a different I/O bus. It had more powerful peripherals, including the RP15/RP02 disk pack and the LP15 DMA line printer. Some PDP-9 controllers, such as the TC09 DECtape controller and the RF09 fixed head disk controller, were redesigned to connect directly to the PDP-15's I/O bus; others were interfaced through a backwards-compatible bus converter.

Memory Management

Over its lifetime, the PDP-15 implemented three different memory management options:

- The KM15 memory protection option implemented boundary register protection. It was programmatically identical to the PDP-9's KX09 option.
- The KT15 memory relocation option implemented base register relocation and boundary register protection. This base-and-bounds style of memory management was used in DOS-15 and RSX-15.
- The XM15 (XVM) memory management option. This option provided not only the base-and-bounds relocation of the KT15, but additional options for shared segments and >32KW virtual addressing. It was supported by the XVM series of software releases, particularly XVM/RSX.

System Extensions

Although the PDP-15 was more successful than any prior 18b system, compared to the PDP-11 its volume was low. This made continuing investment in new technology and options difficult. The CPU was never re-implemented to take advantage of advances in component integration. Investments in new peripheral types and controllers had to be limited. The PDP-15 group responded with great ingenuity to these constraints. Notable developments included:

- Multiprocessing. Two CPU's could share memory and I/O subsystems, for increased throughput in a multiprogramming environment.
- PDP-11 add-on processor. The Unichannel-15 was a PDP-11/05 CPU that functioned as an I/O controller. The Unibus tied in directly to the PDP-15's memory system, using the two data parity lines as extra data lines. This gave the PDP-15 access to inexpensive PDP-11 peripherals, such as the RK05 and LP11.
- XVM memory subsystem. The XVM project was the final spin on the PDP-15. It replaced the initial memory relocation option with a more sophisticated unit that allowed individual programs to extend beyond 32K words. It added instruction prefetching to improve performance and a high-resolution clock for task-level accounting.

These structural innovations stretched the lifetime of the product line but could not reverse its status as a niche rather than a volume product. By the mid 1970's, the PDP-15's position in DEC's product line was eclipsed by the success of the more flexible PDP-11 (as the position of the PDP-10 would be by the VAX). In 1977, the PDP-15 was retired, ending the history of the 18b product family.

Software

The PDP-15 built on the PDP-9's software base. The Advanced Monitor System was retained and extended to create DOS-15 and its batch extension, BOS-15. A new real-time operating system, RSX15, evolved from an execution-only environment into a full-featured multiprogramming system, RSX15-Plus III, that exploited the memory relocation hardware and multiprocessing capabilities to provide simultaneous timesharing, batch, and real-time capabilities. Another notable system was MUMPS (**M**GH **U**tility **M**ulti **P**rogramming **S**ystem), a timesharing system developed at Massachusetts General hospital for processing medical records. Descendents of MUMPS (now known as the M language) continue to be used today in medical systems. DOS, RSX-Plus III, and MUMPS were all substantially rewritten in the mid-70's to take advantage of XVM memory management.

18b Systems Today

Because of the low numbers produced (< 1500), and the early retirement of the product line, relatively few examples of the DEC 18b computers are still extant (a fate shared by the early 36b products as well). Surviving systems are scattered and often in private collections, making an accurate census difficult.

- PDP-1: The Computer History Museum (Mountain View, Ca) has three PDP-1's. One of these was running as recently as 1995 and is being restored to operation. The other two are from DEC's history collection.
- PDP-4: The Computer History Museum has three PDP-4's, all from DEC's history collection. None are considered restorable.
- PDP-7: The Computer History Museum has a PDP-7, from DEC's history collection. Max Burnet (Sydney, Australia) has a PDP-7 in his collection. Neither is considered restorable. There is a partially running PDP-7 in Norway and, incredibly, one still in operation in Oregon, which has been donated to PDP Planet in Seattle Washington.
- PDP-9, 9/L: The Computer History Museum has both a PDP-9 and a –9/L. Max Burnet also has one of each, and the PDP-9/L works. The Rhode Island Computer Museum has a PDP-9, which is being restored. There are two PDP-9's at ACONIT (Grenoble, France); Hans Pufal and his team have restored one to working order.
- PDP-15: Multiple examples in private hands.

Sources

The primary source for this article was DEC's documentation archive. The author was fortunate to have access to the archive while it was still being staffed and maintained (Compaq dismissed the archive staff and dispersed the documents; HP has donated the archive to the Computer History Museum). Max Burnet has graciously shared his unique collection of DEC documents and hardware. In addition, Al Kossow and Dave Gesswein have done the field of "computer archaeology" a tremendous service by scanning, transcribing, and publishing online, surviving documents, DECtapes, and paper-tapes from the 18b family. Davis Johnson located a fascinating memo documenting differences between the PDP-9 and the PDP-15. Last, but hardly least, the staff of the Computer History Museum has made available its significant archive of DEC material. Among the items consulted:

Family

- 1972 Field Service Census of Systems under contract – Computer History Museum

PDP-1

- PDP-1 Handbook (F-15, 1960 edition) – online
- PDP-1 Handbook (F-15B, 1961 edition) – online
- PDP-1 Handbook (F-15C, 1962 edition) – Max Burnet's collection, now online
- PDP-1 Handbook (F-15D, 1963 edition) – Computer History Museum, now online
- PDP-1 Maintenance Manual (F-17) – Max Burnet's collection, now online
- PDP-1 Input-Output Systems Manual (F-25) – DEC archive, now online

PDP-4

- PDP-4 Handbook (F-45, 1962 edition) – DEC archive, now online
- PDP-4 Maintenance Manual (F-47) – Max Burnet's collection, now online
- PDP-4 Technical Specification (DEC memo M-1142) – online
- PDP-4 Fortran Users' Manual (J-4FT) – DEC library, now online
- PDP-4 EAE Option Bulletin (F-43(18)P) – Computer History Museum
- PDP-4 Paper, Gordon Bell, August 1977 – Computer History Museum

PDP-7

- PDP-7 Reference Manual (F-75, 1964 edition) – DEC archive, now online
- PDP-7 Maintenance Manual (F-77) – Max Burnet's collection, now online
- DECSYS-7 Operating Manual (7-5-S) – DEC library, now online

PDP-9

- PDP-9 User's Handbook (F-95, 1968 edition) – online
- PDP-9 Maintenance Manual (F-97) – online
- PDP-9 schematics – online
- KE09A Extended Arithmetic Element Instruction Manual – online
- PDP-9 – Design History, Don Vonada, undated – Computer History Museum

PDP-15

- PDP-15 Reference Manual (first and sixth editions) – online
- PDP-15 Maintenance Manual – online
- XVM System Reference Manual – online
- XVM Maintenance Manual – online
- XVM Processor and Memory Subsystem schematics – online
- PDP-15 processor diagnostics – online
- PDP-15 Development Project History, Jerry Butler, September 1977 – Computer History Museum

Another critical source was Computer Engineering: A DEC View Of Hardware Systems Design. The article “The PDP-1 and Other 18-Bit Computers”, by Gordon Bell, Gerald Butler, Robert Gray, John McNamara, Donald Vonada, and Ronald Wilson, contains unique hardware, marketing, and technology information about the 18b family. The book, out of print for years, is now online, thanks to the efforts of Gordon Bell.

Most recently, Professor Harlan Lefevre of the University of Oregon made an invaluable contribution to the software archive for the 18b PDP’s by providing a copy of DECsys, DEC’s first mass storage operating system. Professor Lefevre’s careful preservation of a functioning PDP-7 and all its software made possible the restoration of a long-missing piece of DEC history.

Lastly, the author had the benefit of the recollections of people who worked on the 18b family, including Gordon Bell, Dennis Ritchie, and Barry Rubinson, as well as access to the surviving archive of PDP-7 software from Applied Data Research.

18b PDP Web Sites

Gordon Greene’s PDP-1 web site, http://www.dbit.com/~greeng3/pdp1/

Al Kossow’s documentation archive, including the 18b PDP’s, http://bitsavers.org/pdf/

Dennis Ritchie and Ken Thompson memoir of early UNIX, http://www.bell-labs.com/history/unix/pdp7.html

SIMH simulation site, http://simh.trailing-edge.com