What Was The PDP-X?

Bob Supnik, 10-Jan-2004 [revised 03-February-2008]

Introduction

The PDP-X was one of Digital Equipment Corporation's legendary lost designs. The leaders of the project, Edson DeCastro and Henry Burkhardt, left DEC when the project was cancelled to found Data General Corporation, amid charges of bad faith and IP theft. The PDP-X was rumored to be the prototype for the Nova, the PDP-11, both, or neither.

Recently uncovered documents in the DEC Archive (now in possession of the Computer History Museum in Mountain View, California) make it possible to debunk these rumors. The "PDP-X" technical memorandum series shows conclusively that the proposed PDP-X had little similarity to either the DG Nova or the PDP-11. Both the Nova and the PDP-11 demonstrate substantial advances in architectural thinking over the PDP-X, with the Nova pointing the way to future RISC processors, and the PDP-11 to the VAX.

The PDP-X Project

The documentary record for the PDP-X begins in June, 1967, with an introductory memo about the Technical Memorandum series, and ends in February, 1968 with a note about proposed assembler syntax. Critical memos include the Processor Architecture (#13, revised in #29) and the System Architecture (#16), both dating from the summer of 1967. By the spring of 1968, the project had been rejected, its value vis-à-vis the established 12b (PDP-8) and 18b (PDP-9) product lines insufficiently proven to warrant further development.

The PDP-X proposal represented a way-station between the "one off" system design embodied in DEC's 12b and 18b systems and the "family" concept of the PDP-11. From the outset, the PDP-X was intended to include a variety of models at a variety of price points. These models would have (upward) compatible features and capabilities but would share common peripherals and software. The lower cost model (the model I) was intended to be price competitive with the PDP-8, the higher cost model (the model II) with the PDP-9.

Architecturally, the PDP-X was also a way-station between the accumulator-oriented systems of the early 60's and the more radical Nova and PDP-11. Multiple accumulators and index registers gave the architecture more flexibility, at the cost of greater complexity (including variable length instructions). The instruction set followed a register-memory model, like the PDP-10, rather than the load-store model of the Nova or the generalized operands of the PDP-11. Real-time processing was a central concern, with fast context switching through multiple register sets.

The PDP-X Architecture

Data Types

The PDP-X was a word-oriented, multiple accumulator, variable length instruction computer. A minimal system had 4KW. A system without memory protection could support 32KW, with memory protection, 128KW. There were five basic data types:

- 16b unsigned integers
- 16b signed integers – 2's complement
- 8b bytes – stored two per word, with the "first" byte on the right ("little endian")
- 32b floating point – IBM "hex" format
- 64b floating point – IBM "hex" format

Bits in memory were numbered left to right, starting with bit 0. Although floating point data formats were defined, the architecture spec had no hardware support for floating point.

Memory

Memory consisted of 16b words. A minimal system had 4KW. A system without memory protection could support 32KW, with memory protection, 128KW. Memory was contiguous; references to non-existent memory caused a trap.

Registers

Processor state was organized around 16 registers. These registers occupied memory addresses 0-15, as in the PDP-10. The first eight registers could be implemented in discrete logic, again following the model of the PDP-10:

R0	program status word
R1	program counter, "index register"
R2	accumulator, subroutine linkage, index register
R3	accumulator, index register
R4	accumulator
R5	accumulator
R6	accumulator
R7	accumulator

The second eight registers were always in memory and had dedicated purposes:

R8	extended op PC
R9	extended op instruction
R10	extended op effective address
R11	extended op entry address

R12 push down pointer
R13 push down counter
R14 trap PC
R15 trap entry address

Each interrupt priority level had its own register set. A minimal system had two priority levels, user and interrupt (monitor); a fully populated system had eight.

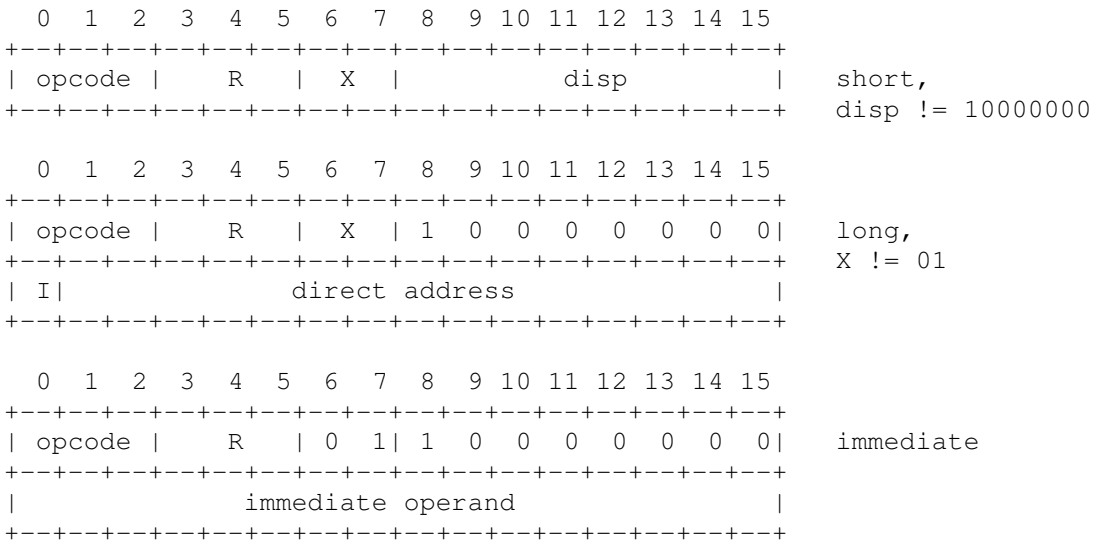
The program status word (PSW) consisted of 16b of status information and the 15b program counter. The PSW provided trap information, condition codes, and priority level control (register set select):

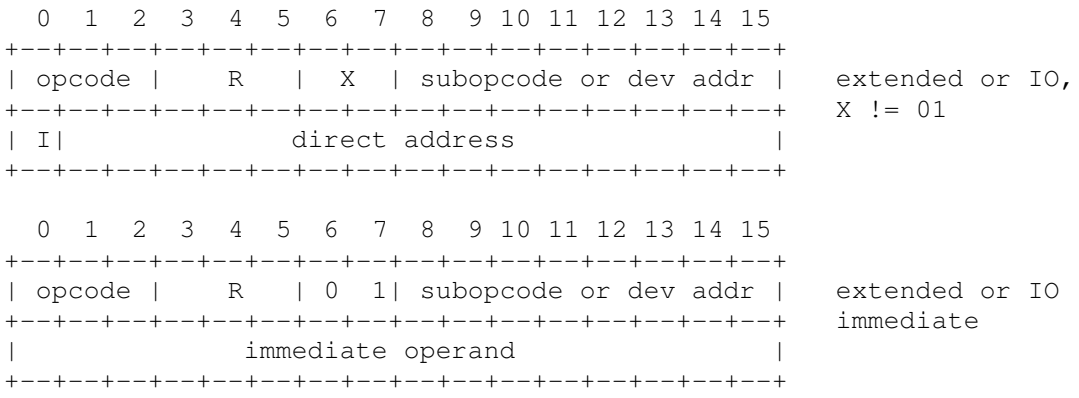
bit<0>	arithmetic trap enable
bit<1>	error trap enable
bit<2>	arithmetic trap flag
bit<3>	push down list error
bit<4>	non-existent memory error
bit<5>	address exception
bit<6>	I/O device error (timeout)
bit<7>	privileged instruction violation
bit<8>	memory protection violation
bits<10:12>	priority (register set select)
bits<13:15>	condition codes

The three condition codes were carry/borrow, negative, and not-zero, respectively.

Instructions

PDP-X instructions were either 16b or 32b in length, depending on the opcode and the addressing mode. There were multiple instruction formats:





Effective address calculation was controlled by the opcode, the X field (bits<6:7>), and by the displacement field (bits<8:15>), as follows:

Opcode <=5 && displacement != 0x80		
X == 0	field 0 direct	ea = displacement
X == 1	PC relative	ea = PC + SEXT8 (displacement)
X == 2	R2 (link) relative	ea = R2 + SEXT8 (displacement)
X == 3	R3 (index) relative	ea = R3 + SEXT8 (displacement)

Opcode >5 || displacement == 0x80		
X == 0	direct	ea = direct address
X == 1	immediate	ea = PC + 1
X == 2	R2 (link) relative	ea = R2 + direct address
X == 3	R3 (index) relative	ea = R3 + direct address

Long addresses supported indirection. Indirect addressing was multi-level.

Because the opcode field was so small (3 bits), the number of basic operations was very small and was almost the same as the PDP-8:

op == 0	LDA	Rn = M[ea], CC unchanged
op == 1	STA	M[ea] = Rn, CC unchanged
op == 2	ADD	Rn = Rn + M[ea], set CC 0-2
op == 3	AND	Rn = Rn & M[ea], set CC 1-2
op == 4	branch	Rn selects one of 8 branch functions
op == 5	modify	Rn selects one of 8 modify operations

The 8 available branches were:

R == 0	BCN	branch if CC 0 (carry) == 1
R == 1	BM	branch if CC 1 (minus) == 1
R == 2	BN	branch if CC 2 (non-zero) == 1
R == 3	B	unconditional branch

R == 4	BCZ	branch if CC 0 (carry) == 0
R == 5	BP	branch if CC 1 (minus) == 0
R == 6	BZ	branch if CC 2 (non-zero) == 0
R == 7	BAL	R2 = PC + 1, unconditional branch

The 8 available modify functions were:

R == 0	TST	set CC 1 and 2 from M[ea]
R == 1	COM	M[ea] = ~M[ea], set CC 1-2
R == 2	INC	M[ea] = M[ea] + 1, set CC 1-2
R == 3	NEG	M[ea] = -M[ea], set CC 1-2
R == 4	RR	rotate M[ea] right through CC 0, set CC 1-2
R == 5	RL	rotate M[ea] left through CC 0, set CC 1-2
R == 6	SWP	swap bytes in M[ea], set CC 1-2
R == 7	CLR	M[ea] = 0, set CC 1-2

The extended operation instructions (opcode 6) provided an “escape” for more complex instructions, at the cost of an additional word. The extended operation class provided room for 256 additional instructions. The first 64 were reserved as UUO’s (unused operation orders), for program/monitor communication (again, like the PDP-10). Of the remaining 192, the following were defined:

subop == 100	LMUL	Rn’Rn v 1 = Rn * M[ea], unsigned, set CC 1-2
subop == 101	MUL	Rn’Rn v 1 = Rn * M[ea], signed, set CC 1-2
subop == 102	LDIV	Rn,Rn v 1 = Rn’Rn v 1 / M[ea], unsigned, set CC 1-2
subop == 103	DIV	Rn,Rn v 1 = Rn’Rn v 1 / M[ea], signed, set CC 1-2
subop == 104	TSTN	Rn & M[ea], set CC 1-2
subop == 105	TSTZ	Rn & M[ea], set CC 1-2; Rn = Rn & ~M[ea]
subop == 106	TSTO	Rn & M[ea], set CC 1-2; Rn = Rn | M[ea]
subop == 107	TSTC	Rn & M[ea], set CC 1-2; Rn = Rn ^ M[ea]
subop == 110	LCMP	Rn : M[ea], unsigned, set CC 1-2
subop == 111	CMP	Rn : M[ea], signed, set CC 1-2
subop == 112	SUB	Rn = Rn – M[ea], set CC 0-2
subop == 113	shift	Rn = Rn (shftop) SEXT8(M[ea]<8:15>)
subop == 114	LDC	Rn<8:15>> = M-byte[ea]; Rn<0:7> = 0
subop == 115	STC	M-byte[ea] = Rn<8:15>
subop == 116	push/pop	one of 8 push-down list operations, selected by R

The shift operation used the effective operand as a control word. Bits<6:7> specified the type of shift:

- Bits<6:7> == 00: arithmetic shift
- Bits<6:7> == 01: rotate through CC 0
- Bits<6:7> == 10: rotate without CC 0
- Bits<6:7> == 11: logical shift

while bits<8:15>, sign extended, controlled the direction and amount of the shift.

The push-down list operations used R12 as the push-down pointer and R13 as the push-down counter. The counter had two bytes; the left for tracking pops, the right for tracking pushes. Push and pop were defined as follows:

- void push (operand): M[R12++] = operand; R13<0:7>++; R13<8:15>--
- int16 pop (void): result = M[--R12]; R13<0:7>--; R13<8:15>++

If either half of the counter was decremented past 0, a trap occurred. This provided both overflow and underflow detection but limited the push-down list to 256 entries. The push-down list operations were:

R == 0	PUC	push but no memory store
R == 1	PUSH	push (M[ea])
R == 2	PUB	push (PC); PC = ea
R == 3	PUL	push (R2); push (PC); R2 = PC; PC = ea
R == 4	POC	pop but no memory read
R == 5	POP	M[ea] = pop ()
R == 6	POB	PC = ea + pop ()
R == 7	POL	PC = ea + pop (); R2 = pop ()

Other extended operations were reserved for floating point and future extensions.

I/O

The I/O architecture was fairly standard for the day. I/O devices were addressed via ports rather than memory locations. There were seven basic I/O primitives:

- read byte
- read word
- read status
- write byte
- write word
- write command
- test status

Data transfers were 8b or 16b. Direct memory access was implemented via a medium speed multiplexor channel or a dedicated selector channel.

The ‘seven primitives’ I/O model reflected current competitive practices; a similar model could be found in the Interdata minicomputers. The multiplexor channel extended DEC’s existing 3-cycle data break designs; the selector channel was, from an architectural viewpoint, invisible.

The eight I/O sub-opcodes implemented overall control functions, such as I/O reset, halt, read switches and write indicators, and priority interrupt subsystem control.

The PDP-X and the Nova

The PDP-X bears little resemblance to the Nova. To list the most obvious differences:

- The PDP-X had a register-memory instruction architecture; the Nova had a load-store instruction architecture.
- The PDP-X was little-endian; the Nova was big-endian.
- The PDP-X was architected for a microcoded implementation; the Nova was architected for a hard-wired implementation.
- The PDP-X had 8 accumulators; the Nova had 4.
- The PDP-X's accumulators could be addressed as memory locations; the Nova's could not.
- The PDP-X had multiple register sets; the Nova did not.
- The PDP-X had variable length instructions; the Nova had fixed length instructions.
- The PDP-X had condition codes and used branches; the Nova had a single carry bit and used skips.
- The PDP-X had many specific single-register operate instructions; the Nova had eight generalized dual-register operate instructions.

Indeed, the only bit of resemblance is in the addressing modes for single-word memory reference instructions. The PDP-X's four modes:

Opcode <=5 && displacement != 0x80		
X == 0	field 0 direct	ea = displacement
X == 1	PC relative	ea = PC + SEXT8 (displacement)
X == 2	R2 (link) relative	ea = R2 + SEXT8 (displacement)
X == 3	R3 (index) relative	ea = R3 + SEXT8 (displacement)

are pretty much the same as the Nova's (although the Nova reversed the roles of R2 and R3). However, there are also differences: the Nova provided indirect addressing for its 16b load-store instructions; the PDP-X did not.

The Nova demonstrates a substantial advance in architectural simplicity, elegance, and orthogonality over the PDP-X. Except for the loop instructions ISZ/DSZ, the Nova was a strict load-store machine, foreshadowing the later RISC processor movement. The I/O system was more flexible than the Interdata-like PDP-X model. The simplicity of the architecture (and the newly available S181 ALU slice) made it possible to build a system that was smaller, faster, and less expensive than the PDP-X would have been.

The PDP-X and the PDP-11

The PDP-X also has little relationship to DEC's eventual 16b architecture, the PDP-11. To list the most obvious differences:

- The PDP-X was a multi-accumulator architecture; the PDP-11 was a general-register architecture.
- The PDP-X had a register-memory instruction architecture; the PDP-11 had a generalized operand instruction architecture.
- The PDP-X addressed memory as words; the PDP-11 addressed memory as bytes.
- The PDP-X's accumulators could be addressed as memory locations; the PDP-11's general registers could not.
- The PDP-X had 16b and 32b instructions; the PDP-11 had 16b, 32b, and 48b instructions.
- The PDP-X had an explicit push down list mechanism; the PDP-11 integrated stacks into the overall addressing modes.
- The PDP-X used the PC-as-general-register only to implement relative addressing; the PDP-11 used the PC as a general register in all addressing modes.
- The PDP-X had multiple register sets; the PDP-11 had only one (until the 11/45, which added a second). The PDP-X register sets were tied to the processor mode; the PDP-11's were not.
- The PDP-X used separate instructions and addressing for devices; the PDP-11 integrated device addressing into standard addressing and used standard instructions for I/O.

There are some similarities. Both designs had a processor status word; both had condition codes and branches rather than skips; the list of single operand instructions is similar.

Like the Nova, the PDP-11 is a substantial advance in architectural thinking over the PDP-X. The major advances:

- Generalized addressing modes integrating indexing and stack
- Generalized two operand instructions
- Use of the PC as a full general register for addressing
- Integration of I/O with memory

represented a significant break with prior systems. The PDP-11 set the model for most minicomputer and microcomputer architecture of the 1970's and was considered the epitome of architectural ingenuity until the VAX.

Summary

The PDP-X was not the direct architectural precursor of either the Nova or the PDP-11. Indeed, its most obvious relationship is not to those systems but to contemporary competitive minicomputers. Its I/O system borrowed heavily from the Interdata model. The Nova abandoned all the complexity of the PDP-X; and the PDP-11 rethought it from scratch. Both proved to be major advances in computer architecture. The PDP-X, despite the nine months of hard work that went into it, was just another minicomputer.