DEC Vax-11

VAX-11/780—A virtual address extension to the DEC PDP-11 family

by W. D. STRECKER
Digital Equipment Corporation
Maynard, Massachusetts

INTRODUCTION

Large virtual address space minicomputers

Perhaps the most useful definition of a minicomputer system is based on price: depending on one's perspective such systems are typically found in the $20K to $200K range. The twin forces of market pull—as customers build increasingly complex systems on minicomputers—and technology push—as the semiconductor industry provides increasingly lower cost logic and memory elements—have induced minicomputer manufacturers to produce systems of considerable performance and memory capacity. Such systems are typified by the DEC PDP-11/70. From an architectural point of view, the characteristic which most distinguishes many of these systems from larger mainframe computers is the size of the virtual address space: the immediately available address space seen by an individual process. For many purposes the 65K byte virtual address space typically provided on minicomputers (such as the PDP-11) has not been and probably will not continue to be a severe limitation. However, there are some applications whose programming is impractical in a 65K byte virtual address space, and perhaps most importantly, others whose programming is appreciably simplified by having a large virtual address space. Given the relative trends in hardware and software costs, the latter point alone will insure that large virtual address space minicomputers play an increasingly important role in minicomputer product offerings.

In principle, there is no great challenge in designing a large virtual address minicomputer system. For example, many of the large mainframe computers could serve as architectural models for such a system. The real challenge lies in two areas: compatibility—very tangible and important; and simplicity—intangible but nonetheless important.

The first area is preserving the customer's and the computer manufacturer's investment in existing systems. This investment exists at many levels: basic hardware (principally busses and peripherals); systems and applications software; files and data bases; and personnel familiar with the programming, use, and operation of the systems. For example, just recently a major computer manufacturer abandoned a major effort for new computer architectures in favor of evolving its current architectures.[1]

The second intangible area is the preservation of those attributes (other than price) which make minicomputer systems attractive. These include approachability, understandability, and ease of use. Preservation of these attributes suggests that simply modelling an extended virtual address minicomputer after a large mainframe computer is not wholly appropriate. It also suggests that during architectural design, tradeoffs must be made between more than just performance, functionality, and cost. Performance or functionality features which are so complex that they appreciably compromise understanding or ease of use must be rejected as inappropriate for minicomputer systems.

VAX-11 overview

VAX-11 is the Virtual Address eXtention of PDP-11 architecture.[2,3] The most distinctive feature of VAX-11 is the extension of the virtual address from 16 bits as provided on the PDP-11 to 32 bits. With the 8-bit byte the basic addressable unit, the extension provides a virtual address space of about 4.3 gigabytes which, even given rapid improvement in memory technology, should be adequate far into the future.

Since maximal PDP-11 compatibility was a strong goal, early VAX-11 design efforts focused on literally extending the PDP-11: preserving the existing instruction formats and instruction set and fitting the virtual address extension around them. The objective here was to permit, to the extent possible, the running of existing programs in the extended virtual address environment. While realizing this objective was possible (there were three distinct designs), it was felt that the extended architecture designs were overly compromised in the areas of efficiency, functionality, and programming ease.

Consequently, 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 a PDP-11 *compatibility mode*. Compatibility

mode provides the basic PDP-11 instruction set less only privileged instructions (such as HALT) and floating point instructions (which are optional on most PDP-11 processors and not required by most PDP-11 software).

In addition to compatibility mode, a number of other features to preserve PDP-11 investment have been provided in the VAX-11 architecture, the VAX-11 operating system VAX/VMS, and the VAX-11/780 implementation of the VAX-11 architecture. These features include:

1. The equivalent native mode data types and formats are identical to those on the PDP-11. Also, while extended, the VAX-11 native mode instruction set and addressing modes are very close to those on the PDP-11. As a consequence VAX-11 native mode assembly language programming is quite similar to PDP-11 assembly language programming.
2. The VAX-11/780 uses the same peripheral busses (Unibus and Massbus) as the PDP-11 and uses the same peripherals.
3. The VAX/VMS operating system is an evolution of the PDP-11 RSX-11M and IAS operating systems, offers a similar although extended set of system services, and uses the same command languages. Additionally, VAX/VMS supports most of the RSX-11M/IAS system service requests issued by programs executing in compatibility mode.
4. The VAX/VMS file system is the same as used on the RSX-11M/IAS operating systems permitting interchange of files and volumes. The file access methods as implemented by the RMS record manager are also the same.
5. VAX-11 high level language compilers accept the same source languages as the equivalent PDP-11 compilers and execution of compiled programs gives the same results.

The coverage of all these aspects of VAX-11 is well beyond the scope of any single paper. The remainder of this paper discusses the design of the VAX-11 native mode architecture and gives an overview of the VAX-11/780 system.

VAX-11 NATIVE ARCHITECTURE

Processor state

Like the PDP-11, VAX-11 is organized around a general register processor state. This organization was favored because access to operands stored in general registers is fast (since the registers are internal to the processor and register accesses do not need to pass through a memory management mechanism) and because only a small number of bits in an instruction are needed to designate a register. Perhaps most importantly, the registers are used (as on the PDP-11) in conjunction with a large set of addressing modes which permit unusually flexible operand addressing methods.

Some consideration was given to a pure stack based architecture. However it was rejected because real program data suggests the superiority of two or three operand instruction formats.[4] Actually VAX-11 is quite stack oriented, and although it is not optimally encoded for the purpose, can easily be used as a pure stack architecture if desired.

VAX-11 has 16 32-bit general registers (denoted R0-R15) which are used for both fixed and floating point operands. This is in contrast to the PDP-11 which has eight 16-bit general registers and six 64-bit floating point registers. The merged set of fixed and floating registers were preferred because it simplifies programming and permits a more effective allocation of the registers.

Four of the registers are assigned special meaning in the VAX-11 architecture:

1. R15 is the *program counter* (PC) which contains the address of the next byte to be interpreted in the instruction stream.
2. R14 is the *stack pointer* (SP) which contains the address of the top of the processor defined stack used for procedure and interrupt linkage.
3. R13 is the *frame pointer* (FP). The VAX-11 procedure calling convention builds a data structure on the stack called a stack frame. FP contains the address of this structure.
4. R12 is the *argument pointer* (AP). The VAX-11 procedure calling convention uses a data structure called an argument list. AP contains the address of this structure.

The remaining element of the user visible processor state (additional processor state seen mainly by privileged procedures is discussed later) is the 16-bit *processor status word* (PSW). The PSW contains the N, Z, V, and C condition codes which indicate respectively whether a previous instruction had a negative result, a zero result, a result which overflowed, or a result which produced a carry (or borrow). Also in the PSW are the IV, DV, and FU bits which enable processor trapping on integer overflow, decimal overflow, and floating underflow conditions respectively. (The trapping on conditions of floating overflow and divide by zero for any data type are always enabled.)

Finally, the PSW contains the T bit which when set forces a trap at the end of each instruction. This trap is useful for program debugging and analysis purposes.

Data types and formats

The VAX-11 data types are a superset of the PDP-11 data types. Where the PDP-11 and VAX-11 have equivalent data types the formats (representation in memory) are identical. Data type and data format identity is one of the most compelling forms of compatibility. It permits free interchange of binary data between PDP-11 and VAX-11 programs. It facilitates source level compatibility between equivalent PDP-11 and VAX-11 languages. It also greatly facilitates hardware implementation of and software support of the PDP-11 compatibility mode in the VAX-11 architecture.

The VAX-11 data types divide into five classes:

1. Integer data types are the 8-bit *byte*, the 16-bit *word*, the 32-bit *longword*, and the 64-bit *quadword*. Usually these data types are considered signed with negative values represented in two's complement form. However, for most purposes they can be interpreted as unsigned and the VAX-11 instruction set provides support for this interpretation.
2. Floating data types are the 32-bit *floating* and the 64-bit *double* floating. These data types are binary normalized, have an 8-bit signed exponent, and have a 25- or 57-bit signed fraction with the redundant most significant fraction bit not represented.
3. The *variable bit field* data type is 0 to 32 bits located arbitrarily with respect to addressable byte boundaries. A bit field is specified by three operands: the address of a byte, the starting bit position P with respect to bit 0 of that byte, and the size S of the field. The VAX-11 instruction set provides for interpreting the field as signed or unsigned.
4. The *character string* data type is 0 to 65535 contiguous bytes. It is specified by two operands: the length and starting address of the string. Although the data type is named "character string", no special interpretation is placed on the values of the bytes in the character string.
5. The *decimal string* data types are 0 to 31 digits. They are specified by two operands: a length (in digits) and a starting address. The primary data type is *packed decimal* with two digits stored in each byte except that the byte containing the least significant digit contains a single digit and the sign. Two ASCII character decimal types are supported: *leading separate* sign and *trailing embedded* sign. The leading separate type is a "+", "−", or "⟨blank⟩" (equivalent to "+") ASCII character followed by 0 to 31 ASCII decimal digit characters. A trailing embedded sign decimal string is 0 to 31 bytes which are ASCII decimal digit characters except for the character containing least significant digit which is an arbitrary encoding of the digit and sign.

All of the data types except field may be stored on arbitrary byte boundaries—there are no alignment constraints. The field data type, of course, can start on an arbitrary bit boundary.

Attributes of and symbolic representations for most of the data types are given in Table I and Figure 1.

TABLE I.—Data Types

DATA TYPE	SIZE	RANGE (decimal)	
Integer		Signed	Unsigned
Byte	8 bits	-128 to +127	0 to 255
Word	16 bits	-32768 to +32767	0 to 65535
Longword	32 bits	-2^{31} to $+2^{31}-1$	0 to $2^{32}-1$
Quadword	64 bits	-2^{63} to $+2^{63}-1$	0 to $2^{64}-1$
Floating Point		$\pm 2.9 \times 10^{-37}$ to 1.7×10^{38}	
Floating	32 bits	approximately seven decimal digits precision	
Double Floating	64 bits	approximately sixteen decimal digits precision	
Packed Decimal String	0 to 16 bytes (31 digits)	numeric, two digits per byte sign in low half of last byte	
Character String	0 to 65535 bytes	one character per byte	
Variable-length Bit Field	0 to 32 bits	dependent on intrepretation	

Instruction format and address modes

Most architectures provide a small number of relatively fixed instruction formats. Two problems often result. First, not all operands of an instruction have the same specification generality. For example, one operand must come from memory and another from a register; or one must come from the stack and another from memory. Second, only a limited number of operands can be accommodated: typically one or two. For instructions which inherently require more operands (such as field or string instructions), the additional operands are specified in ad hoc ways: small literal fields in instructions, specific registers or stack positions, or packed in fields of a single operand. Both these problems lead to increased programming complexity: they require superfluous move type instructions to get operands to places where they can be used and increase competition for potentially scarce resources such as registers.

To avoid these problems two criteria were used in the design of the VAX-11 instruction format: (1) all instructions should have the "natural" number of operands and (2) all operands should have the same generality in specification. These criteria led to a highly variable instruction format. An instruction consists of a one or two* byte *opcode* followed by the specifications for n operands ($n \geq 0$) where n is an implicit property of the opcode. An operand specification is one to 10 bytes in length and consists of a one or two byte *operand specifier* followed by (as required) zero to eight

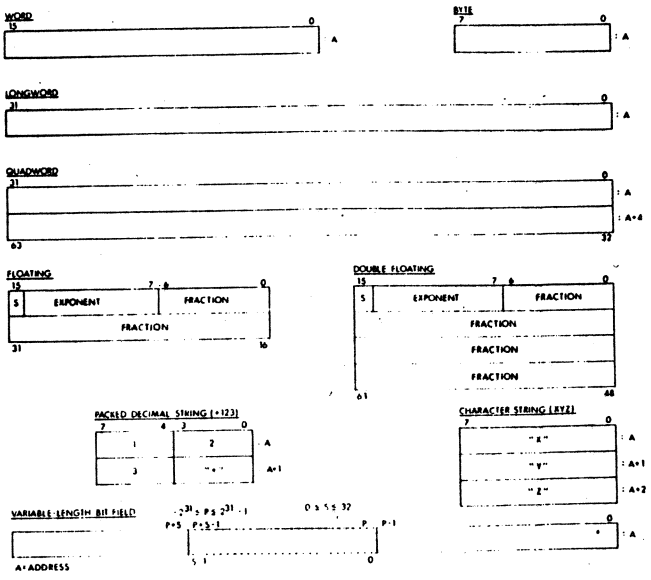


Figure 1—Data formats

* No currently defined instructions use two byte opcodes.

bytes of *specifier extension*. The operand specifier includes the address mode and designation of any registers needed to locate the operand. A specifier extension consists of a displacement, an address, or immediate data.

The VAX-11 address modes are with one exception a superset of the PDP-11 address modes. The PDP-11 address mode autodecrement deferred was omitted from VAX-11 because it was rarely used.

Most operand specifiers are one byte long and contain two 4-bit fields: the high order field (bits 7:4) contains the address mode designator and the lower field (bits 3:0) contains a general register designator. The address modes include:

1. *Register mode* in which the designated register contains the operand.
2. *Register deferred mode* in which the designated register contains the address of the operand.
3. *Autodecrement mode* in which the contents of the designated register are first decremented by the size (in bytes) of the operand and then used as the address of the operand.
4. *Autoincrement mode* in which the contents of the designated register are first used as the address of the operand and are then incremented by the size of the operand. Note that if the designated register is PC, the operand is located in the instruction stream. This use of autoincrement mode is called *immediate* mode. In immediate mode the one to eight bytes of data are the specifier extension.

   Autoincrement mode can be used sequentially to process a vector in one direction and autodecrement mode used to process a vector in the opposite direction. Autoincrement, register deferred, and autodecrement modes can be applied to a single register to implement a stack data structure: autodecrement to "push", autoincrement to "pop", and register deferred to access the top of the stack.
5. *Autoincrement deferred mode* in which the contents of the designated register are used as the address of a longword in memory which contains the address of the operand. After this use, the contents of the register are incremented by four (the size in bytes of the longword address). Note that if PC is the designated register, the absolute address of the operand is located in the instruction stream. This use of autoincrement deferred mode is termed *absolute* mode. In absolute mode the 4-byte address is the specifier extension.
6. *Displacement mode* in which a displacement is added to the contents of the designated register to form the operand address. There are three displacement modes depending on whether a signed byte, word, or longword displacement is the specifier extension. These modes are termed byte, word, and longword displacement respectively. Note that if PC is the designated register, the operand is located relative to PC. For this use the modes are termed byte, word, and longword *relative* mode respectively.
7. *Displacement deferred mode* in which a displacement is added to the designated register to form the address of a longword containing the address of the operand. There are three displacement deferred modes depending on whether a signed byte, word, or longword displacement is the specifier extension. These modes are termed byte, word, and longword displacement respectively. Note that if PC is the designated register, the operand address is located relative to PC. For this use the modes are termed byte, word, and longword *relative deferred* mode respectively.
8. *Literal mode* in which the operand specifier itself contains a 6-bit literal which is the operand. For integer data types the literal encodes the values 0-63; for floating data types the literal includes three exponent and three fraction bits to give 64 common values.
9. *Index mode* which is not really a mode but rather a one byte prefix operator for any other mode which evaluates to a memory address (i.e., all modes except register and literal). The index mode prefix is cascaded with the operand specifier for that mode (called the base operand specifier) to form an aggregate two byte operand specifier. The base operand specifier is used in the normal way to evaluate a base address. A copy of the contents of the register designated in the index prefix is multiplied by the size (in bytes) of the operand and added to the base address. The sum is the final operand address. There are three advantages to the VAX-11 form of indexing: (a) the index is scaled by the data size and thus the index register maintains a logical rather than a byte offset into an indexed data structure, (b) indexing can be applied to any of the address modes which generate memory addresses and this results in a comprehensive set of indexed addressing methods, and (c) the space required to specify indexing and the index register is paid only when indexing is used.

The VAX-11 assembler syntax for the address modes is given in Figure 2. The bracketed ({ }) notation is optional

Literal (Immediate)	{S↑ I↑} # *constant*	
Register	R*n*	
Register Deferred	(R*n*)	Indexed [Rx]
Autodecrement	-(R*n*)	
Autoincrement	(R*n*)+	
Autoincrement Deferred	@(R*n*)+	
(Absolute)	@#*address*	
Displacement	{B↑ W↑ L↑} *displacement*(R*n*) *address*	
Displacement Deferred	@{B↑ W↑ L↑} *displacement*(R*n*) *address*	

n = 0 through 15
x = 0 through 14

Figure 2—Assembler syntax

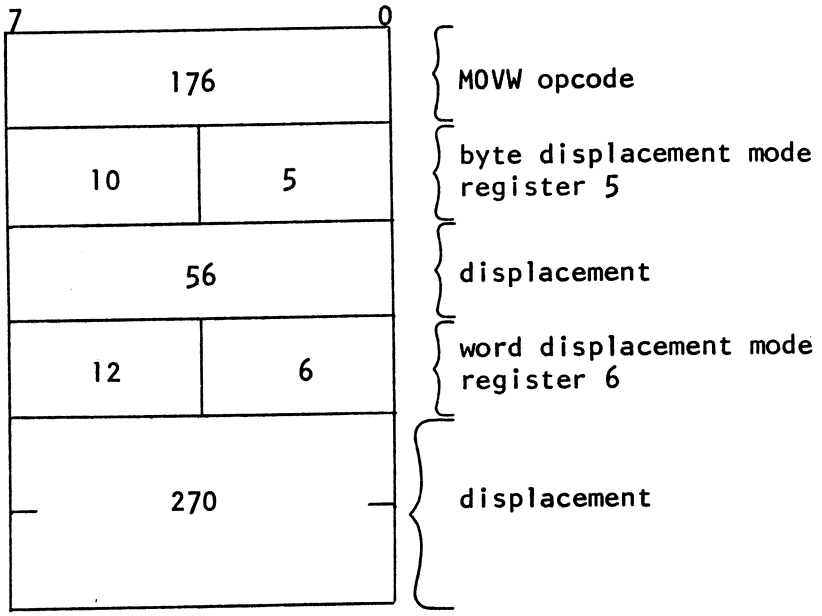


Figure 3—MOVW 56 (R5), 270(R7)

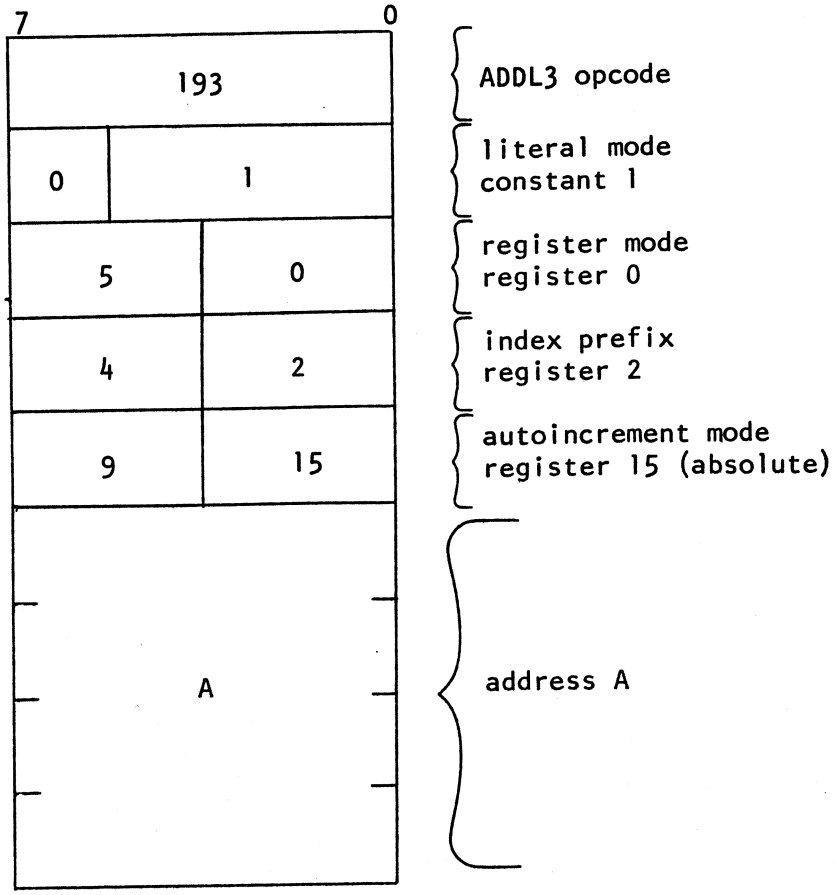


Figure 4—ADDL3 #1, RO, @#A[R2]

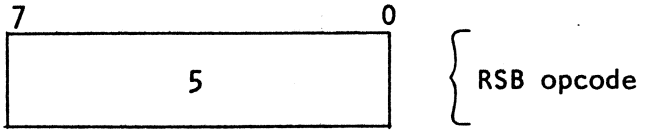


Figure 5—RSB

and the programmer rarely needs to be concerned with displacement sizes or whether to choose literal or immediate mode. The programmer writes the simple form and assembler chooses the address mode which produces the shortest instruction length.

In order to give a better feeling for the instruction format and assembler notation, several examples are given in Figures 3-5. In Figure 3 is an instruction which moves a word from an address which is 56 plus the contents of R5 to an address which is 270 plus the contents of R6. Note, that the displacement 56 is representable in a byte while the displacement 270 requires a word. The instruction occupies 6 bytes. In Figure 4 is an instruction which adds 1 to a longword in R0 and stores the result at a memory address which is the sum of A and 4 times the contents of R2. This instruction occupies 9 bytes. Finally, in Figure 5 is a return from subroutine instruction. It has no explicit operands and occupies a single byte.

The only significant instance where there is non-general specification of operands is in the specification of targets for branch instructions. Since invariably the target of a branch instruction is a small displacement from the current PC, most branch instructions simply take a one byte PC relative displacement. This is exactly as if byte displacement mode were used with the PC used as the register, except that the operand specifier byte is not needed. Because of the pervasiveness of branch instructions in code, this one byte saving results in a non-trivial reduction in code size. An example of the branch instruction branch on equal is given in Figure 6.

Instruction set

A major goal of the VAX-11 instruction set design was to provide for effective compiler generated code. Four decisions helped to realize this goal:

1. A very regular and consistent treatment of operators. Thus, for example, since there is a divide longword instruction, there are also divide word and divide byte instructions.

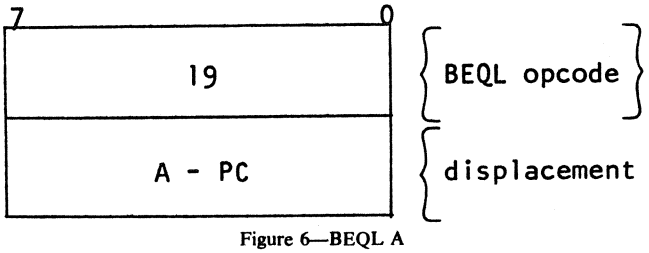


Figure 6—BEQL A

2. An avoidance of instructions unlikely to be generated by a compiler.
3. Inclusion of several forms of common operators. For example the integer add instructions are included in three forms: (a) one operand where the value one is added to an operand, (b) two operands where one operand is added to a second, and (c) three operands where one operand is added to a second and the result stored in a third. Since the VAX-11 instruction format allows fully general specifications of the operands, VAX-11 programs often have the structure (though not the encoding) of the canonic program form proposed in Reference 5.
4. Replacement of common instruction sequences with single instructions. Examples of this include procedure calling, multiway branching, loop control, and array subscript calculation.

The effect of these decisions is reflected in several observations. First, despite the larger virtual address and instruction set support for more data types, compiler (and hand) generated code for VAX-11 is typically smaller than the equivalent PDP-11 code for algorithms operating on data types supported by the PDP-11. Second, of the 243 instructions in the instruction set about 75 percent are generated by the VAX-11 FORTRAN compiler. Of the instructions not generated, most operate on data types not part of the FORTRAN language.

A complete list of the VAX-11 instructions is given in the Appendix. The following gives an overview of the instruction set.

1. *Integer logic and arithmetic*—Byte, word, and longword are the primary data types. A fairly conventional group of arithmetic and logical instructions is provided. The result generating dyadic arithmetic and logical instructions are provided in two and three operand forms. A number of optimizations are included: clear which is a move of zero; test which is a compare against zero; and increment and decrement which are an optimization of add one and subtract one respectively. A complete set of converts is provided which covers both the integer and the floating data types. In contrast to other architectures only a few shift type instructions are provided: it was felt that shifts are mostly used for field isolation which is much more conveniently done with the field instructions described later. In order to support greater than longword precision integer operations, a few special instructions are provided: extended multiply and divide and add with carry and subtract with carry.
2. *Floating point instructions*—Again a conventional group of instructions are included with result producing dyadic operators in two and three operand forms. Several specialized floating point instructions are included. The extended modulus instruction multiplies two floating operands together and stores the integer and fraction parts of the product in separate result operands. The polynomial instruction computes a polynomial

from a table of coefficients in memory. Both these instructions employ greater than normal precision and are useful in high accuracy mathematical routines. A convert rounded instruction is provided which implements the ALGOL rather than FORTRAN conventions for converting from floating point to integer.

3. *Address instructions*—The move address instructions store in the result operand the effective address of the source operand. The push address optimizations push on the stack (defined by SP) the effective address of the source operand. The latter are used extensively in subroutine calling.
4. *Field instructions*—The extract field instructions extract a 0 to 32-bit field, sign- or zero-extend it if it is less than 32 bits, and store it in a longword operand. The compare field instructions compare a (sign- or zero-extended if necessary) field against a longword operand. The find first instructions find the first occurrence of a set or clear bit in a field.
5. *Control instructions*—There is a complete set of conditional branches supporting both a signed and, where appropriate, an unsigned interpretation of the various data types. These branches test the condition codes and take a one byte PC relative branch displacement. There are three unconditional branch instructions: the first taking a one byte PC relative displacement, the second taking a word PC relative displacement, and the third—called jump—taking a general operand specification. Paralleling these three instructions are three branch to subroutine instructions. These push the current PC on the stack before transferring control. The single byte return from subroutine instruction returns from subroutines called by these instructions. There is a set of branch on bit instructions which branch on the state of a single bit and, depending on the instruction, set, clear, or leave unchanged that bit.

   The add compare and branch instructions are used for loop control. A step operand is added to the loop control operand and the sum compared against a limit operand. The result of the comparison determines whether the branch is taken. The sense of the comparison is based on the sign of the step operand. Optimizations of loop control include the add one and branch instructions which assume a step of one and the subtract one and branch instructions which assume a step of minus one and a limit of zero.

   The case instructions implement the computed go to in FORTRAN and case statements in other languages. A selector operand is checked to see that it lies in range and is then used to select one of table of PC relative branch displacements following the instruction.
6. *Queue instructions*—The queue representation is a doubly linked circular list. Instructions are provided to insert an item into a queue or to remove an item from a queue.
7. *Character string instructions*—The general move character instruction takes five operands specifying the lengths and starting addresses of the source and destination strings and a fill character to be used if the source string is shorter than the destination string. The instruction functions correctly regardless of string overlap. An optimized move character instruction assumes the string lengths are equal and takes three operands. Paralleling the move instructions are two compare character instructions. The move translated characters instruction is similar to the general move character instruction except that the source string bytes are translated by a translation table specified by the instruction before being moved to destination string. The move translated until escape instruction stops if the result of a translation matches the escape character specified by one of its operands. The locate and skip character instructions find respectively the first occurence or non-occurence of a character in a string. The scan and span instructions find respectively the first occurence or non-occurence of a character within a specified character set in a string. The match characters instruction finds the first occurence of a substring within a string which matches a specified pattern string.
8. *Packed decimal instructions*—A conventional set of arithmetic instructions is provided. The arithmetic shift and round instruction provides decimal point scaling and rounding. Converts are provided to and from longword integers, leading separate decimal strings, and trailing embedded decimal strings. A comprehensive edit instruction is included.

VAX-11 procedure instructions

A major goal of the VAX-11 design was to have a single system wide procedure calling convention which would apply to all inter-module calls in the various languages, calls for operating system services, and calls to the common run time system. Three VAX-11 instructions support this convention: two call instructions which are indistinguishable as far as the called procedure is concerned and a return instruction.

The call instructions assume that the first word of a procedure is an *entry mask* which specifies which registers are to be used by the procedure and thus need to be saved. (Actually only R0-R11 are controlled by the entry mask and bits 15:12 of the mask are reserved for other purposes.) After pushing the registers to be saved on the stack, the call instruction pushes AP, FP, PC, a longword containing the PSW and the entry mask, and a zero valued longword which is the initial value of a condition handler address. The call instruction then loads FP with the contents of SP and AP with the argument list address. The appearance of the stack frame after the call is shown in the upper part of Figure 7.

The form of the argument list is shown in the lower part of Figure 7. It consists of an argument count and list of longword arguments which are typically addresses. The CALLG instruction takes two operands: one specifying the procedure address and the other specifying the address of the argument list assumed arbitrarily located in memory.

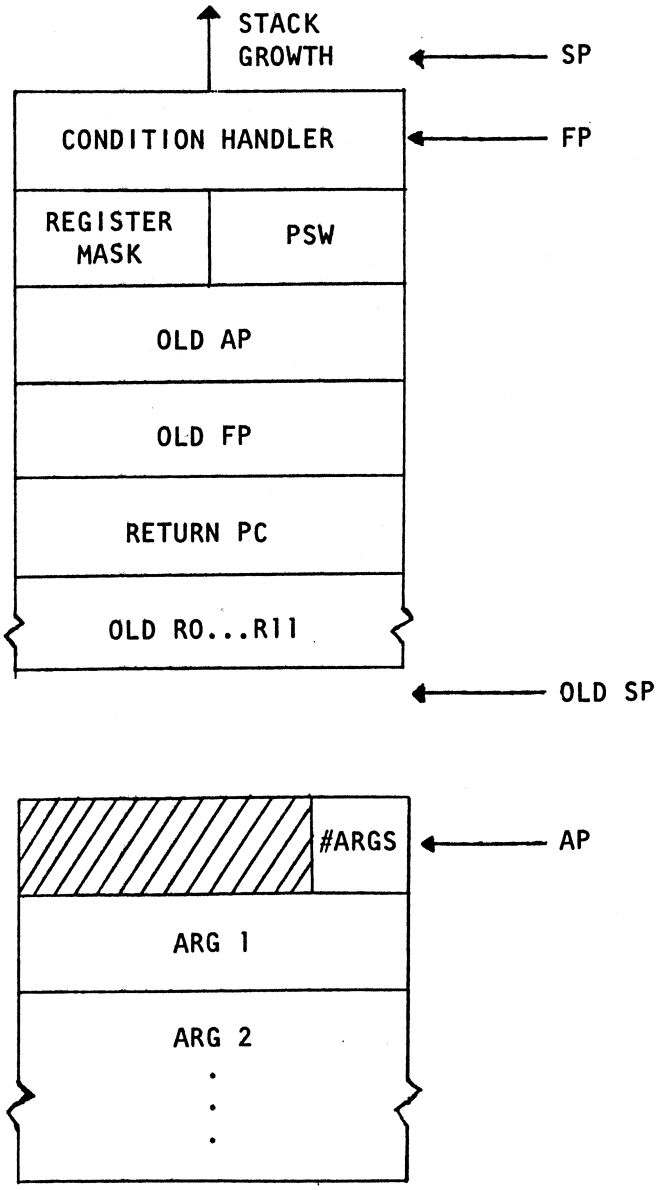


Figure 7—Stack frame

The CALLS instruction also takes two operands: one the procedure address and the other an argument count. CALLS assumes that the arguments have been pushed on the stack and pushes the argument count immediately prior to saving the registers controlled by the entry mask. It also sets bit 13 of the saved entry mask to indicate a CALLS instruction was used to make the call.

The return instruction uses FP to locate the stack frame. It loads SP with the contents of FP and restores PSW through PC by popping the stack. The saved entry mask controls the popping and restoring of R11 through R0. Finally if the bit indicating CALLS was set, the argument list is removed from the stack.

Memory management design alternatives

Memory management comprises the mechanisms used (1) to map the virtual addresses generated by processes to physical memory addresses, (2) to control access to memory (i.e., to control whether a process has read, write, or no access to various areas of memory), and (3) to allow a process to execute even if all of its virtual address space is not simultaneously mapped to physical memory (i.e., to provide so called virtual memory facilities). The memory management proved to be the most difficult part of the architecture to design. Three alternatives were pursued and full designs were completed for the first two alternatives and nearly completed for the third. The three alternatives* were:

1. A paged form of memory management with access control at the page level and a small number (4) of hierarchical access modes whose use would be dedicated to specific purposes. This represented an evolution of the PDP-11/70 memory management.
2. A paged and segmented form with access control at the segment level and a larger number (8) of hierarchical access modes which would be used quite generally. Although it differed in a number of ways, the design was motivated by the Multics[6,7] architecture and the Honeywell 6180 implementation.
3. A capabilities[8,9] form with access control provided by the capabilities and the ability to page larger objects described by the capabilities.

The first alternative was finally selected. The second alternative was rejected because it was felt that the real increase in functionality provided inadequately offset the increased architectural complexity. The third alternative appeared to offer functionality advantages that could be useful over the longer term. However, it was unlikely that these advantages could be exploited in the near term. Further it appeared that the complexity of the capabilities design was inappropriate for a minicomputer system.

Memory mapping

The 4.3 gigabyte virtual address space is divided into four regions as shown in Figure 8. The first two regions—the *program* and *control* regions—comprise the per process virtual address space which is uniquely mapped for each process. The second two regions—the *system* region and a region reserved for future use—comprise the system virtual address space which is singly mapped for all processes.

Each of the regions serves different purposes. The program region contains user programs and data and the top of the region is a dynamic memory allocation point. The control

* It should not be construed that memory management is independent of the rest of the architecture. The various memory management alternatives required different definitions of the addressing modes and different instruction level support for addressing.

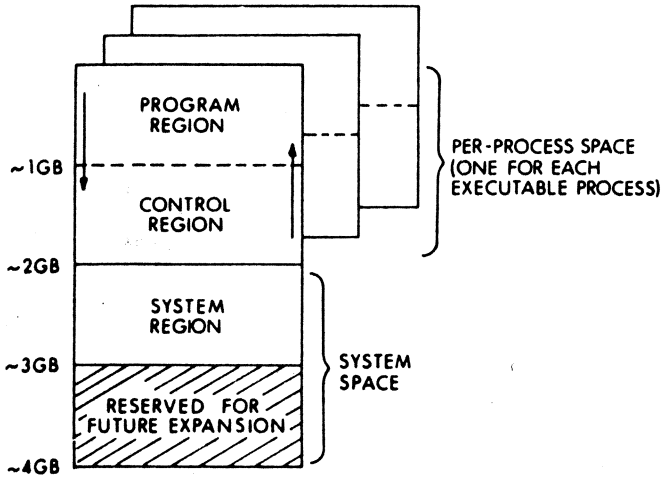


Figure 8—Virtual address space

region contains operating system data structures specific to the process and the user stack. The system region contains procedures which are common to all processes (such as those that comprise the operating system and RMS) and (as will be seen later) page tables.

A virtual address has the structure shown in the upper part of Figure 9. Bits 8:0 specify a byte within a 512 byte *page* which is the basic unit of mapping. Bits 29:9 specify a *virtual page number* (VPN). Bits 31:30 select the virtual address region. The mechanism of mapping consists of using the region select bits to select a *page table* which consists of *page table entries* (PTEs). After a check that it is not too large, the VPN is used to index into the page table to select a PTE. The PTE contains either (1) 21-bit physical *page frame number* which is concatenated with the nine low order byte in page bits to form a 30-bit physical address shown in the lower part of Figure 9, or (2) an indication that the virtual page accessed is not in physical memory. The latter case is called a *page fault*. Instruction execution in the current procedure is suspended and control is transferred to an operating system procedure which will cause the missing virtual page to be brought into physical memory. At this point instruction execution in the suspended procedure can resume transparently.

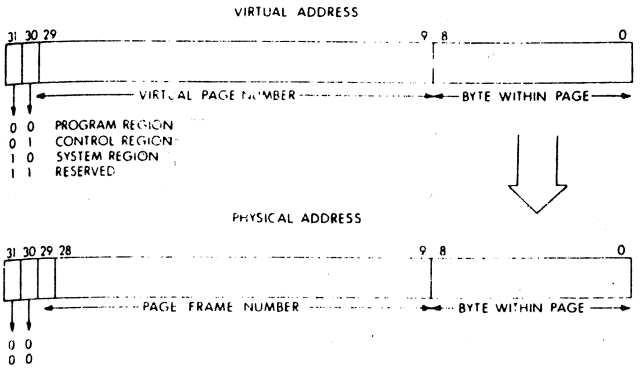


Figure 9—Virtual and physical addresses

The page table for the system region is defined by the *system base register* which contains the physical address of the start of the system region page table and the *system length register* which contains the length of the table. Thus the system page table is contiguous in physical memory.

The per process space page tables are defined similarly by the *program* and *control region base registers* and *length registers*. However, the base registers do not contain physical addresses: rather, they contain system region virtual addresses. Thus the per process page tables are contiguous in the system region virtual address space and are not necessarily contiguous in physical memory. This placement of the per process page tables permits them to be paged and avoids what would otherwise be a serious physical memory allocation problem.

Access control

At a given point in time a process executes in one of four *access modes*. The modes from most privileged to least are called *kernel, executive, supervisor* and *user*. The use of these modes by VAX/VMS is as follows:

1. Kernel—Interrupt and exception handling, scheduling, paging, physical I/O, etc.
2. Executive—Logical I/O as provided by RMS.
3. Supervisor—The command interpreter.
4. User—User procedures and data.

The accessibility of each page (read, write, or no access) from each access mode is specified in the PTE for that page. Any attempt to improperly access a page is suppressed and control is transferred to an operating system procedure. The accessibility is assumed hierarchically ordered: if a page is writable from any given mode, it is also readable; and if a page is accessible from a less privileged mode, it is accessible from a more privileged mode. Thus, for example, a page can be readable and writable from kernel mode, only readable from executive mode, and inaccessible from supervisor and user modes.

A procedure executing in a less privileged mode often needs to call a procedure which executes in a more privileged mode: e.g., a user program needs an operating system service performed. The access mode is changed to a more privileged mode by executing a change mode instruction which transfers control to a routine executing at the new access mode. A return is made to original access mode by executing a return from exception or interrupt instruction (REI).

The *current access mode* is stored in the *processor status longword* (PSL) whose low order 16 bits comprise the PSW. Also stored in the PSL is the *previous access mode*; i.e., the access mode from which the current access mode was called. The previous mode information is used by the special probe instructions which validate arguments passed in cross access mode calls.

Procedures running at each of the access modes require a

separate stacks with appropriate accessibility. To facilitate this, each process has four copies of SP which are selected according to the current access mode field in the PSL. A procedure always accesses the correct stack by using R14.

In an earlier section, it was stated that the VAX-11 standard CALL instruction is used for all calls including those for operating system services. Indeed procedures do call the operating system using the CALL instruction. The target of the CALL instruction is the minimal procedure consisting of an entry mask, a change mode instruction, and a return instruction. Thus access mode changing is transparent to the calling procedure.

Interrupts and exceptions

Interrupts and exceptions are forced changes in control flow other than that explicitly indicated by the executing program. The distinction between them is that interrupts are normally unrelated to the currently executing program while exceptions are a direct consequence of program execution. Examples of interrupt conditions are status changes in I/O devices while examples of exception conditions are arithmetic overflow or a memory management access control violation.

VAX-11 provides a 31 priority level interrupt system. Sixteen levels (16-31) are provided for hardware while 15 levels (1-15) are provided for software. (Level 0 is used for normal program execution.) The current *interrupt priority level* (IPL) is stored in a field in the PSL. When an interrupt request is made at a level higher than IPL, the current PC and PSL are pushed on the stack and new PC obtained from a vector selected by the interrupt requester (a new PSL is generated by the CPU). Interrupts are serviced by routines executing with kernel mode access control. Since interrupts are appropriately serviced in a system wide rather than a specific process context, the stack used for interrupts is defined by another stack pointer called the *interrupt stack pointer*. (Just as for the multiple stack pointers used in process context, an interrupt routine accesses the interrupt stack using R14.) An interrupt service is terminated by execution of an REI instruction which loads PC and PSL from the top two longwords on the stack.

Exceptions are handled like interrupts except for the following: (1) since exceptions arise in a specific process context, the kernel mode stack for that process is used to store PC and PSL and (2) additional parameters (such as the virtual address causing a page fault) may be pushed on the stack.

Process context switching

From the standpoint of the VAX-11 architecture, the process state or context consists of:

1. The 15 general registers R0-R13 and R15.
2. Four copies of R14 (SP): one for each of kernel, executive, supervisor, and user access modes.
3. The PSL.
4. Two base and two limit registers for the program and control region page tables.

This context is gathered together in a data structure called a *process control block* (PCB) which normally resides in memory. While a process is executing, the process context can be considered to reside in processor registers. To switch from one process to another it is required that the process context from the previously executing process be saved in its PCB in memory and the process context for the process about to be executed to be loaded from its PCB in memory. Two VAX-11 instructions support context switching. The save process context instruction saves the complete process context in memory while the load process context instruction loads the complete process context from memory.

I/O

Much like the PDP-11, VAX-11 has no specific I/O instructions. Rather, I/O devices and device controllers are implemented with a set of registers which have addresses in the physical memory address space. The CPU controls I/O devices by writing these registers; the devices return status by writing these registers and the CPU subsequently reading them. The normal memory management mechanism controls access to I/O device registers and a process having a particular device's registers mapped into its address space can control that device using the regular instruction set.

Compatibility mode

As mentioned in the VAX-11 overview, compatibility mode in the VAX-11 architecture provides the basic PDP-11 instruction set less privileged and floating point instructions. Compatibility mode is intended to support a user as opposed to an operating system environment. Normally a compatibility mode program is combined with a set of native mode procedures whose purpose is to map service requests from some particular PDP-11 operating system environment into VAX/VMS services.

In compatibility mode the 16-bit PDP-11 addresses are

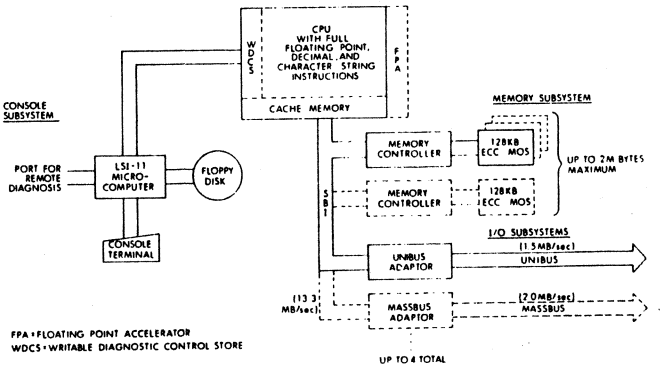


Figure 10—VAX-11/780 system

zero-extended to 32-bits where standard native mode mapping and access control apply. The eight 16-bit PDP-11 general registers overmap the native mode general registers R0-R6 and R15 and thus the PDP-11 processor state is contained wholly within the native mode processor state.

Compatibility mode is entered by setting the compatibility mode bit in the PSL. Compatibility mode is left by executing a PDP-11 trap instruction (such as used to make operating service requests), and on interrupts and exceptions.

VAX-11/780 IMPLEMENTATION

VAX-11/780

The VAX-11/780 computer system is the first implementation of the VAX-11 architecture. For instructions executed in compatibility mode, the VAX-11/780 has a performance comparable to the PDP-11/70. For instructions executed in native mode, the -11/780 has a performance in excess of the -11/70 and thus represents the new high end of the -11 (LSI-11, PDP-11, VAX-11) family.

A block diagram of the -11/780 system is given in Figure 10. The system consists of a central processing unit (CPU), the console subsystem, the memory subsystem, and the I/O subsystem. The CPU and the memory and I/O subsystems are joined by a high speed synchronous bus called the *Synchronous Backplane Interconnect* (SBI).

CPU

The CPU is a microprogrammed processor which implements the native and compatibility mode instruction sets, the memory management, and the interrupt and exception mechanisms. The CPU has 32-bit main data paths and is built almost entirely of conventional Shottky TTL components.

To reduce effective memory access time the CPU includes an 8K byte write through cache or buffer memory. The cache organization is 2-way associative with an 8-byte block size. To reduce delays due to writes, the CPU includes a write buffer. The CPU issues the write to the buffer and the actual memory write takes place in parallel with other CPU activity.

The CPU contains a 128 entry address translation buffer which is a cache of recent virtual to physical translations. The buffer is divided into two 64 entry sections: one for the per process regions and one for the system region. This division facilitates permitting the system region translations to remain unaffected by a process context switch.

A fourth buffer in the CPU is the 8-byte instruction buffer. It serves two purposes. First, it decomposes the highly variable instruction format into its basic components and, second, it constantly fetches ahead to reduce delays in obtaining the instruction components.

The CPU includes two standard clocks. The programmable real-time clock is used by the operating system for local timing purposes. The time-of-year clock with its own battery backup is the long term time reference for the operating system. It is automatically read on system startup to eliminate the need for manual entry of data and time.

The CPU includes 12K bytes of writable diagnostic control store (WDCS) which is used for diagnostic purposes, implementation of certain instructions, and for future microcode changes. As an option for very sophisticated users, another 12K bytes of writable control store is available.

A second option is the floating point accelerator (FPA). Although the basic CPU implements the full floating point instruction set, the FPA provides high speed floating point hardware. It is logically invisible to programs and only affects their running time.

Console subsystem

The console subsystem is centered around an LSI-11 computer with 16K bytes of RAM and 8K bytes of ROM (used to store the LSI-11 bootstrap, LSI-11 diagnostics, and console routines). Also included are a floppy disk, an interface to the console terminal, and a port for remote diagnostic purposes.

The floppy disk in the console subsystem serves multiple purposes. It stores the main system bootstrap and diagnostics and serves as a medium for distribution of software updates.

SBI

The SBI is the primary control and data transfer path in the -11/780 system. Because the cache and write buffer largely decouple the CPU performance from the memory access time, the SBI design was optimized for bandwidth and reliability rather than the lowest possible access time.

The SBI is a synchronous bus with a cycle time of 200 nsec. The data path width of the SBI is 32 bits. During each 200 nsec cycle either 32 bits of data or a 30-bit physical address can be transferred. Since each 32-bit read or write requires transmission of both address and data, two SBI cycles are used for a complete transaction. The SBI protocol permits 64-bit reads or writes using one address cycle and two data transfer cycles: the CPU and I/O subsystem use this mode whenever possible. For read transactions the bus is reacquired by the memory in order to send the data: thus the bus is not held during the memory access time.

Arbitration of the SBI is distributed: each interface to the SBI has a specific priority and its own bus request line. When an interface wishes to use the bus, it asserts its bus request line. If at the end of a 200 nsec cycle there are no interfaces of higher priority requesting the bus, the interface takes control of the bus.

Extensive checking is done on the SBI. Each transfer is parity checked and confirmed by the receiver. The arbitration process and general observance of the SBI protocol are checked by each SBI interface during each SBI cycle. The processor maintains a running 16-cycle history of the SBI:

any SBI error condition causes this history to be locked and preserved for diagnostic purposes.

Memory subsystem

The memory subsystem consists of one or two memory controllers with up to 1M bytes of memory on each. The memory is organized in 64-bit quadwords with an 8-bit ECC which provides single bit error correction and double bit error detection. The memory is built of 4K MOS RAM components.

The memory controllers have buffers which hold up to four memory requests. These buffers substantially increase the utilization of the SBI and memory by permitting the pipelining of multiple memory requests. If desired, quadword physical addresses can be interleaved across the memory controllers.

As an option, battery backup is available which preserves the contents of memory across short term power failures.

I/O subsystem

The I/O subsystem consists of buffered interfaces or adapters between the SBI and the two types of peripheral busses used on PDP-11 systems: the Unibus and the Massbus. One Unibus adapter and up to four Massbus adapters can be configured on a VAX-11/780 system.

The Unibus is a medium speed multiplexor bus which is used as a primary memory as well as peripheral bus in many PDP-11 systems. It has an 18-bit physical address space and supports byte and word transfers. In addition to implementing the Unibus protocol and transmitting interrupts to the CPU, the Unibus adapter provides two other functions. The first is mapping 18-bit Unibus addresses to 30-bit SBI physical addresses. This is accomplished in a manner substantially identical to the virtual to physical mapping implemented by the CPU. The Unibus address space is divided into 512 512-byte pages. Each Unibus page has a page table entry (residing in the Unibus adapter) which maps addresses in that page to physical memory addresses. In addition to providing address translation, the mapping permits contiguous transfers on the Unibus which cross page boundaries to be mapped to discontiguous physical memory page frames.

The second function performed by the Unibus adapter is assembling 16-bit Unibus transfers (both reads and writes) into 64-bit SBI transfers. This operation (which is applicable only to block transfers such as from disks) appreciably reduces SBI traffic due to Unibus operations. There are 15 8-byte buffers in the Unibus adapter permitting 15 simultaneous buffered transactions. Additionally there is a un-buffered path through the Unibus adapter permitting an arbitrary number of simultaneous un-buffered transfers.

The Massbus is a high speed block transfer bus used primarily for disks and tapes. The Massbus adapter provides much the same functionality as the Unibus adapter. The physical addresses into which transfers are made are defined by a page table: again this permits contiguous device transfers into discontiguous physical memory.

Buffering is provided in the Massbuss adapter which minimizes the probability of device overruns and assembles data into 64-bit units for transfer over the SBI.

ACKNOWLEDGMENTS

Although the final architecture is the result of several design iterations involving many hardware and software engineers, the author would like to acknowledge the other members of the initial architectural group: Gordon Bell, Peter Conklin, Dave Cutler, Bill Demmer, Tom Hastings, Richy Lary, Dave Rodgers and Steve Rothman. Mary Jane Forbes and Louise Principe deserve special thanks for typing this manuscript.

REFERENCES

1. McLean, J., "Univac Disbanding Future Systems Plan," *Electronic News*, December 12, 1977.
2. Bell, G., *et al.*, "A New Architecture for Minicomputers—the DEC PDP-11," *AFIPS Conference Proceedings*, Vol. 36, 1970.
3. Bell, G. and W. D. Strecker, "Computer Structures: What Have We Learned from the PDP-11," *Conference Proceedings: 3rd Annual Symposium on Computer Architecture*, 1976.
4. Myers, G. J., "The Case Against Stack-Oriented Instruction Sets," *Sigarch News*, August 1977.
5. Flynn, M. J., "The Interpretive Interface: Resources and Program Representation in Computer Organization," *High Speed Computer and Algorithm Organization* (Kuck, Lawrie, and Sameh, editors), Academic Press, New York, 1977.
6. Organick, E. I., *"The Multics Systems: An Examination of its Structure,"* MIT Press, Cambridge, 1972.
7. Schroeder, M. D. and J. H. Saltzer, "A Hardware Architecture for Implementing Protection Rings," *Proceedings Third Symposium on Operating Systems Principles*, 1971.
8. Needham, R. M., "Protection Systems and Protection Implementations," *AFIPS Conference Proceedings*, Vol. 41, 1972.
9. Needham, R. M. and R. D. H. Walker, "The Cambridge CAP Computer and Its Protection System," *Proceedings Sixth Symposium on Operating Systems Principles*, 1977.

APPENDIX—VAX-11 INSTRUCTION SET

Integer and Floating Point Logical Instructions

MOV-	Move(B,W,L,F,D,Q)*
MNEG-	Move Negated(B,W,L,F,D)
MCOM-	Move Complemented(B,W,L)
MOVZ-	Move Zero-Extended(BW,BL,WL)
CLR-	Clear(B,W,L=F,Q=D)
CVT--	Convert(B,W,L,F,D)(B,W,L,F,D)
CVTR-L	Convert Rounded(F,D) to Longword
CMP-	Compare(B,W,L,F,D)
TST-	Test(B,W,L,F,D)
BIS-2	Bit Set(B,W,L)2-Operand
BIS-3	Bit Set(B,W,L)3-Operand
BIC-2	Bit Clear(B,W,L)2-Operand

* B = byte, W = word, L = longword, F = floating, D = double floating, Q = quadword, S = set, C = clear.

BIC-3	Bit Clear(B,W,L)3-Operand
BIT-	Bit Test(B,W,L)
XOR-2	Exclusive OR(B,W,L)2-Operand
XOR-3	Exclusive OR(B,W,L)3-Operand
ROTL	Rotate Longword
PUSHL	Push Longword

Integer and Floating Point Arithmetic Instructions

INC-	Increment(B,W,L)
DEC-	Decrement(B,W,L)
ASH-	Arithmetic Shift(L,Q)
ADD-2	Add(B,W,L,F,D)2-Operand
ADD-3	Add(B,W,L,F,D)3-Operand
ADWC	Add with Carry
ADAWI	Add Aligned Word Interlocked
SUB-2	Subtract(B,W,L,F,D)2-Operand
SUB-3	Subtract(B,W,L,F,D)3-Operand
SBWC	Subtract with Carry
MUL-2	Multiply(B,W,L,F,D)2-Operand
MUL-3	Multiply(B,W,L,F,D)3-Operand
EMUL	Extended Multiply
DIV-2	Divide(B,W,L,F,D)2-Operand
DIV-3	Divide(B,W,L,F,D)3-Operand
EDIV	Extended Divide
EMOD-	Extended Modulus(F,D)
POLY-	Polynomial Evaluation (F,D)

Index Instruction

INDEX	Compute Index

Packed Decimal Instructions

MOVP	Move Packed
CMPP3	Compare Packed 3-Operand
CMPP4	Compare Packed 4-Operand
ASHP	Arithmetic Shift Round and Packed
ADDP4	Add Packed 4-Operand
ADDP6	Add Packed 6-Operand
SUBP4	Subtract Packed 4-Operand
SUBP6	Subtract Packed 6-Operand
MULP	Multiply Packed
DIVP	Divide Packed
CVTLP	Convert Long to Packed
CVTPL	Convert Packed to Long
CVTPT	Convert Packed to Trailing
CVTTP	Convert Trailing to Packed
CVTPS	Convert Packed to Separate
CVTSP	Convert Separate to Packed
EDITPC	Edit Packed to Character String

Character String Instructions

MOVC3	Move Character 3-Operand
MOVC5	Move Character 5-Operand
MOVTC	Move Translated Characters
MOVTUC	Move Translated Until Character
CMPC3	Compare Characters 3-Operand
CMPC5	Compare Characters 5-Operand
LOCC	Locate Character
SKPC	Skip Character
SCANC	Scan Characters
SPANC	Span Characters
MATCHC	Match Characters

Variable-Length Bit Field Instructions

EXTV	Extract Field
EXTZV	Extract Zero-Extended Field
INSV	Insert Field
CMPV	Compare Field
CMPZV	Compare Zero-Extended Field
FFS	Find First Set
FFC	Find First Clear

Branch on Bit Instructions

BLB-	Branch on Low B(S,Cl)
BB-	Branch on Bit(S,Cl)
BBS-	Branch on Bit Set and(S,Cl)Bit
BBC	Branch on Bit Clear and(Set,lear)Bit
BBSSI	Branch on Bit Set and Set Bit Interlocked
BBCCI	Branch on Bit Clear and Clear Bit Interlocked

Queue Instructions

INSQUE	Insert Entry in Queue
REMQUE	Remove Entry from Queue

Address Manipulation Instructions

MOVA-	Move Address(B,W,L=F,Q=D)
PUSHA-	Push Address(B,W,L=F,Q=D)on Stack

Processor State Instructions

PUSHR	Push Registers on Stack
POPR	Pop Registers from Stack
MOVPSL	Move from Processor Status Longword
BISPSW	Bit Set Processor Status Word
BICPSW	Bit Clear Processor Status Word

Unconditional Branch and Jump Instructions

BR-	Branch with(B,W)Displacement
JMP	Jump

Branch on Condition Code

BLSS	Less Than
BLSSU	Less Than Unsigned
(BCS)	(Carry Set)
BLEQ	Less Than or Equal
BLEQU	Less Than or Equal Unsigned
BEQL	Equal
(BEQLU)	(Equal Unsigned)
BNEQ	Not Equal
(BNEQU)	(Not Equal Unsigned)
BGTR	Greater Than
BGTRU	Greater Than Unsigned
BGEQ	Greater Than or Equal
BGEQU	Greater Than or Equal Unsigned
(BCC)	(Carry Clear)
BVS	Overflow Set
BVC	Overflow Clear

Loop and Case Branch

ACB-	Add, Compare and Branch(B,W,L,F,D)
AOBLEQ	Add One and Branch Less Than or Equal
AOBLSS	Add One and Branch Less Than
SOBGEQ	Subtract One and Branch Greater Than or Equal
SOBGTR	Subtract One and Branch Greater Than
CASE-	Case on(B,W,L)

Subroutine Call and Return Instructions

BSB	Branch to Subroutine with(B,W)Displacement
JSB	Jump to Subroutine
RSB	Return from Subroutine

Procedure Call and Return Instructions

CALLG	Call Procedure with General Argument List
CALLS	Call Procedure with Stack Argument List
RET	Return from Procedure

Access Mode Instructions

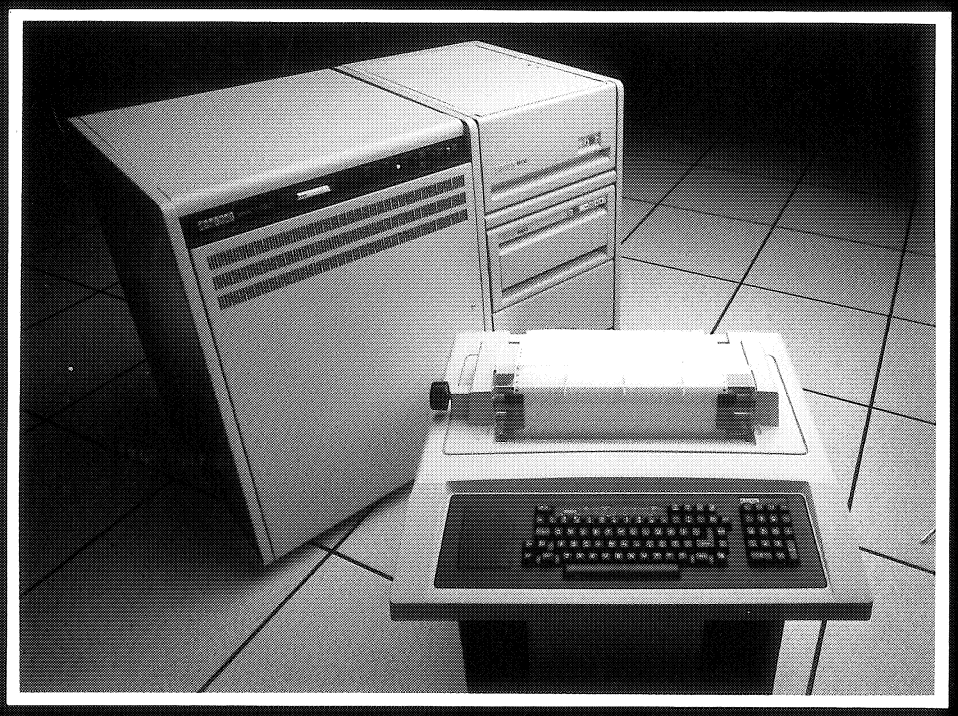
CHM	Change Mode to (Kernel,Executive,Supervisor,User)
REI	Return from Exception or Interrupt
PROBER	Probe Read
PROBEW	Probe Write

Privileged Processor Register Control Instructions

SVPCTX	Save Process Context
LDPCTX	Load Process Context
MTPR	Move to Process Register
MFPR	Move from Processor Register

Special Function Instructions

CRC	Cyclic Redundancy Check
BPT	Breakpoint Fault
XFC	Extended Function Call
NOP	No Operation
HALT	Halt



VAX11 750

digital

DESCRIPTION

The VAX-11/750 computer system is the mid-range, high-performance system that implements VAX Family architecture, thus making it software compatible with other VAX Family systems. User programs written for one VAX system can run without modification on any other VAX system. Data and software compatibility with PDP-11 computer systems is provided by utilization of the same disk or tape file formatting and through compatibility mode PDP-11 system instructions.

A variety of complete VAX-11/750 systems is available. Each system is based on the VAX-11/750 processor, the VAX/VMS operating system, a choice of mass storage subsystems, a console terminal, an 8-line communication multiplexer, and a minimum of one Mbyte of memory.

CENTRAL PROCESSOR

The Central Processing Unit (CPU) of the VAX-11/750 system is a microprogrammed computer that executes a large set of variable-length instructions in VAX native mode, and nonprivileged PDP-11 system instructions in compatibility mode.

VAX Native Mode

The native-mode instruction set is highly versatile and bit-efficient, resulting in concise, fast-running programs. Each native mode instruction can use any of nine different addressing modes. This yields smaller programs and less complicated assembly language programming.

The processor has built-in floating-point instructions that handle single and double precision (32- and 64-bit) computation. Full decimal arithmetic and text string instructions equip the VAX-11/750 processor for commercial and character-oriented data processing applications. The native mode instruction set also includes program control and special instructions that improve both the performance and the memory utilization of system and application software. Instructions are variable in length and can start on any byte boundary.

Compatibility Mode

In compatibility mode, the processor executes the user set of PDP-11 instructions, recognizes integer data, and uses eight 16-bit general purpose registers. This allows the VAX-11/750 system user to run most existing nonprivileged RSX-11M™ and RSX-11S operating system software, including compilers, utilities and user programs. Programs that use compatibility mode instructions can execute concurrently with native mode programs.

Registers and Cache Memory

The processor provides sixteen 32-bit registers that can serve as temporary storage, accumulators, index registers, and base registers for user application programs.

The Memory Cache of the processor is a 4 Kbyte, direct-mapped, write-through cache. It reduces the effective CPU access time to memory to 400 nanoseconds with a 90% hit rate. The write-through feature protects the integrity of memory because memory contents are updated immediately after the processor performs a write.

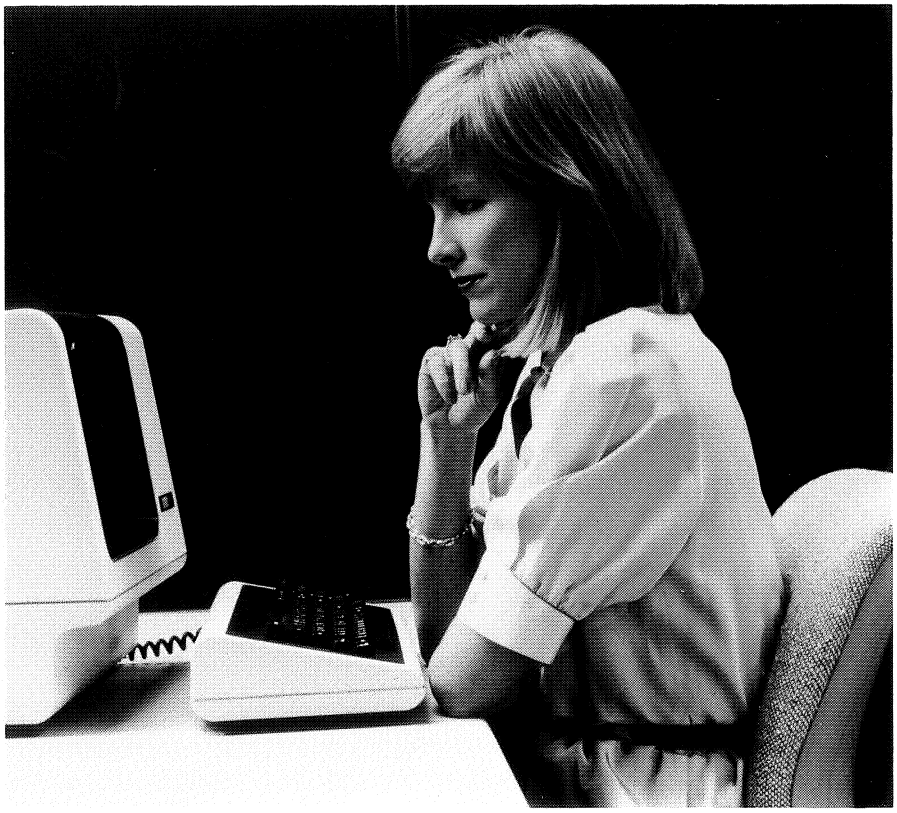
The translation buffer of the processor contains 512 virtual-to-physical page address translations. This significantly reduces the time spent on virtual to physical address translation. An 8-byte instruction prefetch buffer improves processor performance by prefetching and decoding data in the instruction stream. The control logic fetches long-word data from memory to keep the 8-byte buffer full.

Clocks

Two clocks are contained in the VAX-11/750 processor. The high-resolution, programmable, realtime clock is used by systems diagnostics and by the VAX/VMS operating system for accounting and scheduling. The time-of-year clock, with its own battery backup, is used by the operating system to enable unattended automatic restart following any service interruption, including a power failure.

HIGHLIGHTS

- 32-bit architecture with 4 gigabytes of virtual address space allowing for large programs without overlays
- Full VAX Instruction Set, including over 240 basic operations, nine addressing modes, six data types, and compatibility mode PDP-11™ instructions
- Sixteen 32-bit general purpose registers that can be used for temporary storage and as accumulators, index registers, and base registers
- A multiuser, multifunction virtual memory operating system that supports a wide variety of languages and program development tools
- Dependability through extensive reliability, availability and maintainability features including automatic error checking, remote diagnosis, VAX/VMS operating system on-line diagnostics, and automatic restart
- Advanced semiconductor technology implementation that reduces the physical size and power requirements, and increases reliability
- Expansion to eight Mbytes of ECC MOS Memory
- Direct memory access I/O for up to two UNIBUS or up to three MASSBUS (optional) adapters
- Hardware Bootstrap Load for up to 4 different devices
- High-speed intelligent parallel I/O option
- High-performance floating-point accelerator option



The VAX-11/750™ computer system is the mid-range member of the VAX family of computers. Like the VAX-11/780 it implements the full 32-bit VAX architecture, yet at a lower price. In addition to sharing the same set of system software with the VAX-11/780 and the VAX-11/730 systems, it also offers compatibility with the peripheral set via the UNIBUS™ and the optional MASSBUS™ adapters. The processor uses a 32-bit architecture with addressing space for very large programs, an extensive instruction set with numerous data types, and 32-bit bus structure for high throughput.

The VAX-11/750 system hardware is complemented by the VAX/VMS™ (Virtual Memory System) operating system. This general purpose, multiprogramming operating system simultaneously handles multiuser, realtime, and multistream batch applications, plus online program development.

Console Subsystem

The VAX-11/750 console subsystem is designed to allow the user to give commands interactively to the processor using the console command language. Three elements combine to give the user access to the system's capabilities: a set of machine status switches on the processor's front panel, the integral TU58 tape cartridge unit, and the console terminal. The DECwriter™ terminal serves as an operating system terminal for control and error logging, and as a system console. Additionally, the console terminal is a diagnostic console with the Remote Diagnostic Module (RDM) option. The TU58 tape cartridge unit may be used as a load device for diagnostics, optional software products, and software updates, as well as a convenient storage medium (up to 256 Kbytes) for personal data and programs.

The remote diagnosis module is an option available to customers with a full, DIGITAL maintenance contract. It provides the customer with dial-in diagnosis (24 hours a day, seven days a week, with a 15 minute response time), including detection of failures down to the board and chip level, and diagnostic capability even on an inoperative CPU.

SUPPORTED SOFTWARE

VAX/VMS Operating System

The VAX-11/750 system hardware is complemented by the high-performance VAX/VMS operating system. This operating system can support many user environments, including realtime processing, interactive program development, and batch processing, running independently, concurrently, or in any combination. It includes a highly efficient scheduling algorithm, extensive record and file management capabilities, and virtual memory features achieved by an extremely efficient paging algorithm.

Languages and Utilities

The VAX/VMS operating system offers a large choice of languages and utilities:

VAX-11 COBOL	VAX-11 DSM
VAX-11 FORTRAN	VAX-11 DATATRIEVE
VAX-11 BASIC	VAX-11 DBMS
VAX-11 PL/I	VAX-11 CDD
VAX-11 PASCAL	VAX-11 FMS
VAX-11 C	DECnet-VAX™
VAX-11 CORAL 66	MUX200/VAX
VAX-11 BLISS-32	VAX-11 2780/3780
VAX-11 MACRO	VAX-11 3271
VAX-11 DEC/CMS™	VAX-11 PSI
VAX-11 DIBOL™	

The VAX/VMS operating system also provides program development and maintenance utilities including an interactive and a batch editor, a linker with cross-reference ability, and a symbolic debugger.

I/O SUBSYSTEM

A UNIBUS adapter supports general purpose options and devices. The maximum I/O rate through one of the three buffered data paths is 1.5 megabytes per second. This adapter has direct vectored interrupts. As much as five megabytes of data can be moved per second among the system's major hardware components. An optional second UNIBUS can be added.

Up to three optional MASSBUS adapters can be plugged into the basic VAX-11/750 processor. This allows connection for up to 24 MASSBUS disk or tape drives on a single processor. An individual MASSBUS adapter has a maximum I/O rate of two Mbytes per second.

The VAX-11/750 also supports the UDA50, a microprocessor based controller that provides performance optimization to improve disk throughput.

DMF32 Multifunction Communications Controller

The VAX-11/750 UNIBUS accepts a number of VAX/VMS supported device and communication controllers, including the DMF32 multifunction communications controller. The DMF32 consists of four controllers including: an 8-line asynchronous multiplexer featuring programmed DMA or silo transmit lines; a DMA synchronous line supported by DECnet communications software; and *either* a DMA lineprinter controller *or* a general purpose 16-bit DR11 parallel interface. The DMF32 is programmed for three interfaces to run concurrently; the asynchronous multiplexer, synchronous multiplexer, and LP or DR device.

DEPENDABILITY

VAX systems, designed for careful cooperation between hardware and software, perform continual checking, detecting, and correcting of error conditions. The result is a fault-tolerant environment where errors are corrected before they impede system operation. VAX maintenance operations, which are aided in part by the system's remote diagnosis capability, proceed quickly and easily when a failure does occur. Components are tested and repaired in minimal time, often without disturbing users or bringing the system down. Some of the VAX-11/750 dependability features are listed below.

- The operating system has powerfail recovery code; online system error logging; dynamic bad block handling; online functional diagnostics concurrent with user programs; and a User Environment Test Package (UETP).
- The CPU has ECC MOS Memory with a battery backup option; ability to turn off malfunctioning cache and continue to operate in degraded mode; CPU integrity diagnostic run during power up; VAX-11/750 processor board count is reduced to five modules (due to custom LSI technology); parity checking on MASSBUS data, cache, translation buffer, and CPU microcode; and single board options for MBA (MASSBUS Adapter) and UCS (User Control Store).
- The power supply has a simple removal and replacement procedure; sensing and visual indicators for short circuit, overvoltage, loss of power and regulator board failure; and airflow and temperature sensors that power the system down on blower failure.
- The system has been packaged so that all components are easily accessible for repair; cables are fixed in place; CPU modules, CPU options, and memory arrays have no cables connected to them; almost all interconnections are etched into the backplane printed circuit board with very little wirewrapping required; and the printed circuit card cage and the backplane are fixed mounted into the system cabinet to eliminate cable service loops.

SPECIFICATIONS

Processor

Micro-control store instruction time	320 nanoseconds
Control store size	6 Kwords (80-bit words, read only memory)
Internal data path	32 bits
Maximum system I/O rate	5 MB/s
Instruction buffer size	8-byte lookahead
Memory cache	4 KB direct mapped

VAX Instruction Set

16 32-bit registers

248 basic operations

56 optional instructions

32 priority interrupt levels

9 addressing modes

Data types: Four integer types, four floating-point types, packed decimal, character string, variable bit fields, numeric strings

Main Memory

Virtual address space	4 gigabytes
Physical expansion	8 megabytes in 1 Mbyte increments
Parity	7-bit error correcting code per 32-bit longword
Cycle times	800 nanoseconds per 32-bit read 640 nanoseconds per 32-bit write 400 nanoseconds effective with cache

I/O UNIBUS Adapter

Maximum UNIBUS I/O rate	1.5 MB/s through buffered data paths per UNIBUS
Buffered data paths	3 total, 4-byte buffer in each per UNIBUS

POWER REQUIREMENTS*

Maximum

AC line voltage tolerance	90–128 or 180–256 V
Frequency tolerance	47–63 Hz
Phases	1
Surge current	100 A
Maximum AC power consumption	1700 W

Physical Characteristics*

Weight	182 kg (400 lbs) maximum
Height	106.2 cm (41.8 in)
Width	73.7 cm (29.0 in)
Depth	76.3 cm (30.0 in)

Operating Environment*

Temperature	10° to 40°C (50° to 104°F)
Relative humidity	10 to 90%
Maximum altitude	2.4 km (8000 ft)
Heat dissipation	1460 kcal/h (5800 BTU/h)

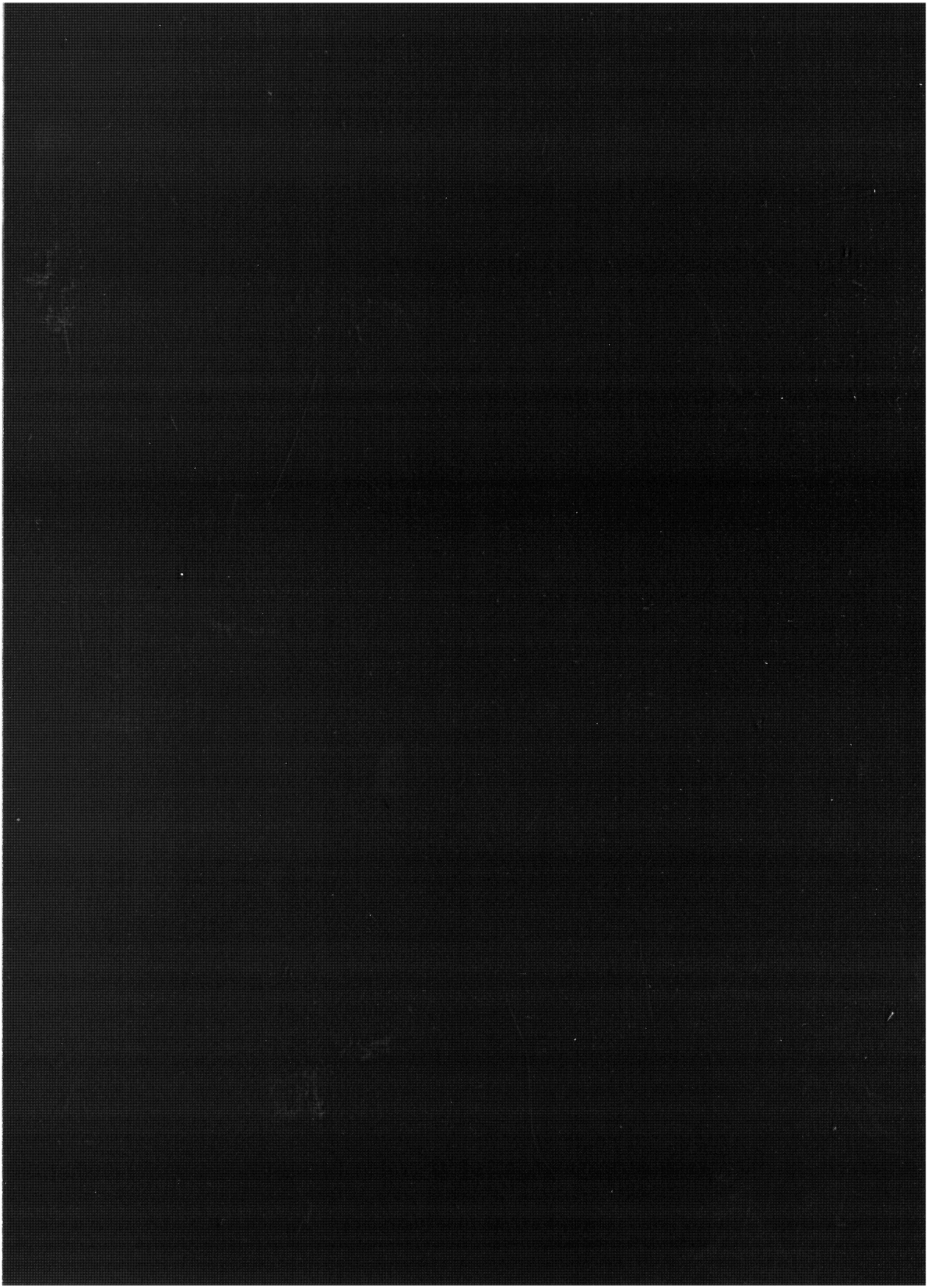
***NOTE: These specifications are given for the VAX-11/750 processor only.**

ORDER CODE: ED-23831-18

VAX 8600

Now, the VAX/VMS Environment for Your Large Applications.

digital



Change and Growth: The Challenges for Your Computing Strategy.

Your business is successful because of the decisions you've made. You've been willing to assess your alternatives when faced with difficult choices. Willing to invest now for future gain. The reason for the success of this strategy is that while making choices for the present you've been able to anticipate future change.

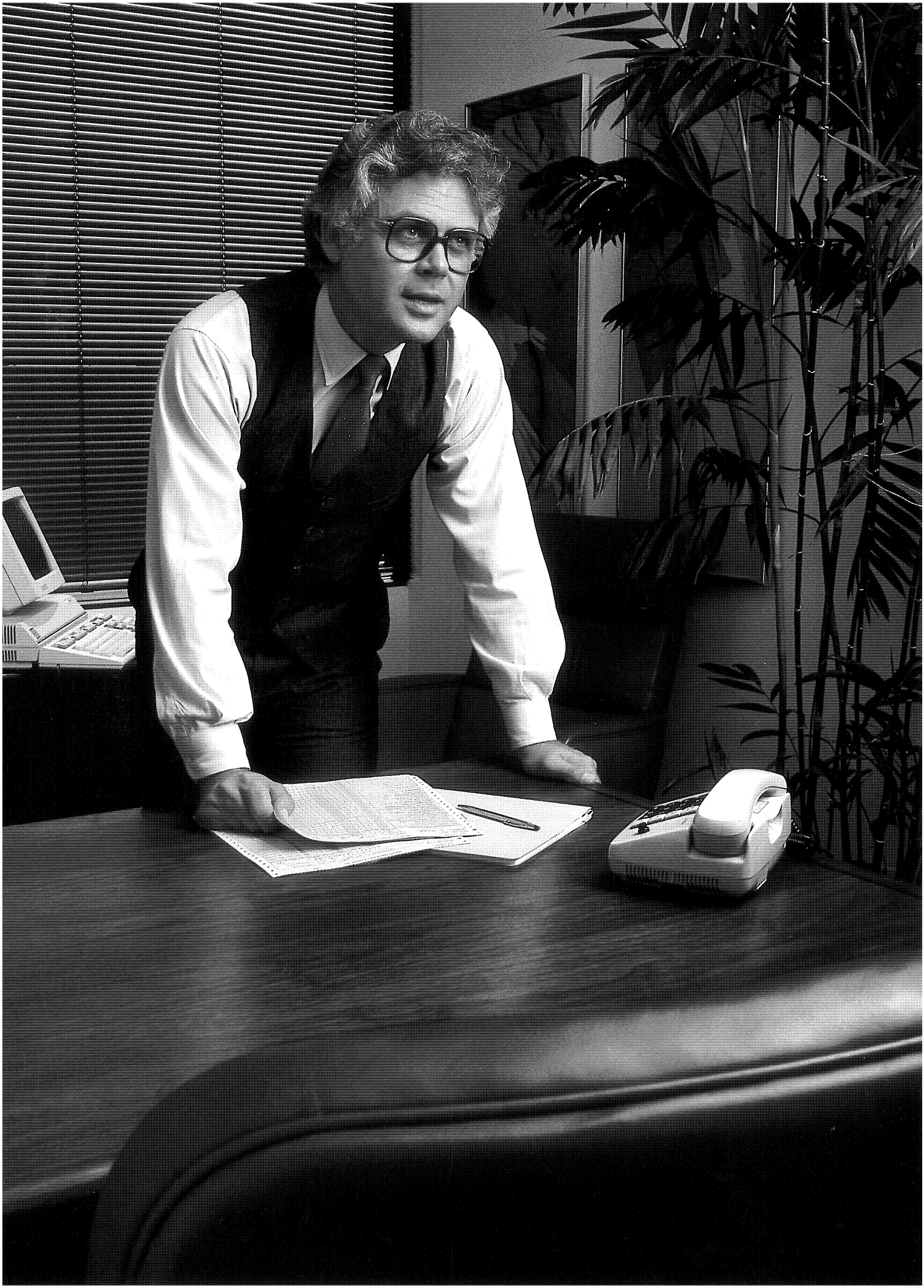
Ten years ago, few of us used the words "computer strategy." Today, we have to. Both users and manufacturers understand that each organization has specific needs to be met with its computer resources – data processing, program development, office automation, networking and a host of other applications. As your organization grows, these applications change or are replaced by new ones. You've had to analyze your various information management needs and determine which could be met most efficiently with your computer resources. The key for you is to anticipate changes in your organization and incorporate this knowledge into your computer strategy.

The challenge for Digital is to use our engineering skill to provide computer resources that can meet your ever-changing needs as you implement your computer strategy. From micros to our largest systems, we've put that skill to work for you. All systems in the VAX family of computers work together, or independently, and all run the same VMS operating system software. New developments in the family must comply with our own strict guidelines to keep it that way, and to keep up with change and growth within organizations such as yours.

Your strategy and planning are most important when considering a major investment, such as a large computer system. You must have the power and the storage to satisfy a growing number of users. You must be able to link the new system to your existing hardware – locally or over a network. You must have the tools to develop and run the applications that keep your organization running smoothly. And you must have the assurance that your investment will be preserved when it's time to grow again.

We listened to your request for more performance in a large system – a system that's compatible with your existing resources at the desk, department, and data center levels. And we responded with the VAX 8600 – our largest VAX system – the VAX with improved performance and reliability in the same physical space as our industry-standard VAX-11/780 system.

Now, with the VAX 8600 and VMS software, you can stay within the VAX computing environments to run all of your applications.



Real Performance Improvement Demands Advanced Technology.

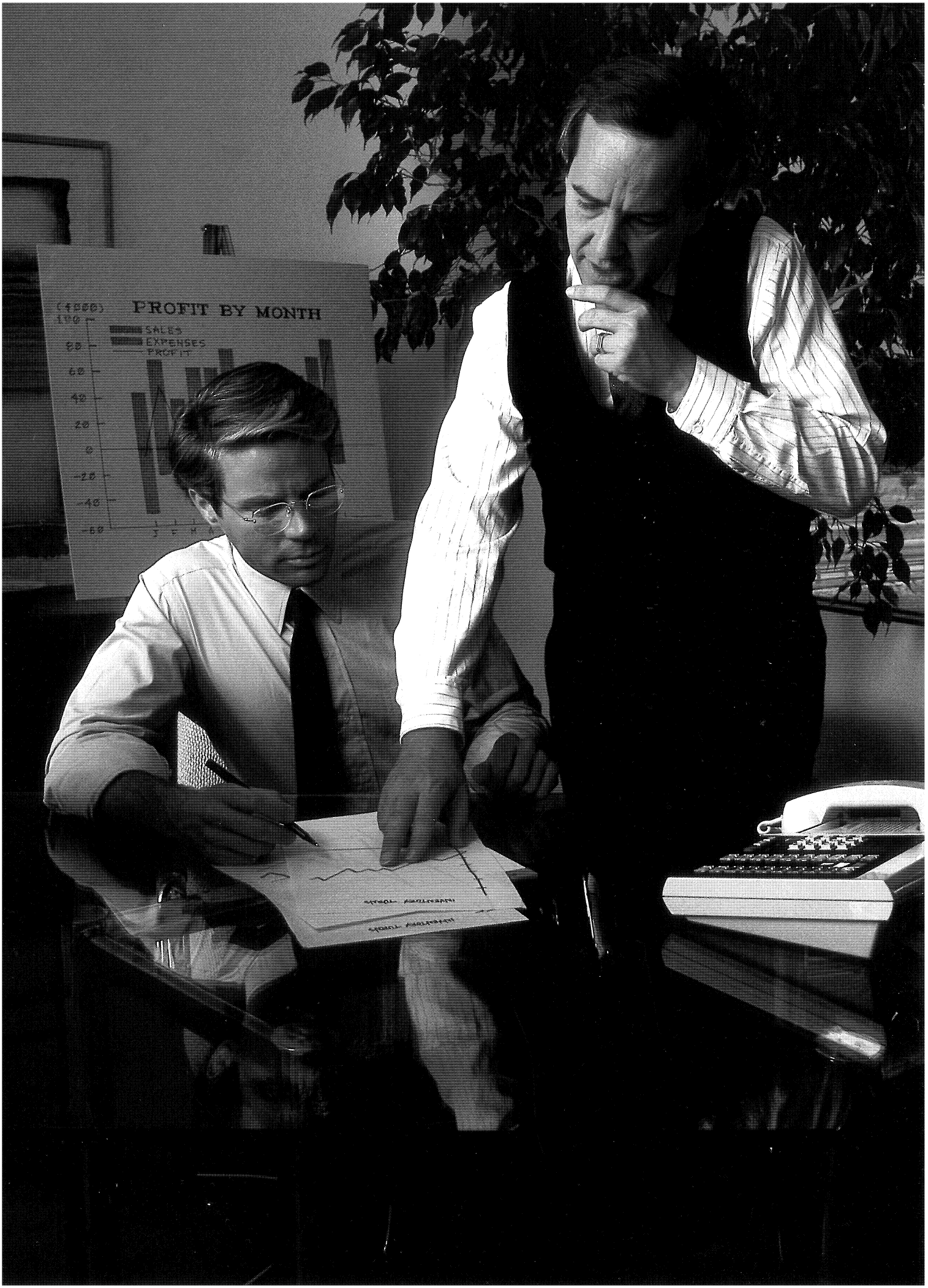
Adding memory or storage isn't always the answer to gaining more performance. In order to meet your demands for higher performance computers, we've employed advanced technologies. We've made an extremely efficient processor that makes better use of memory without creating an oversized machine. The VAX 8600 reduces the time needed to retrieve instructions from memory through pipelined processing. This means that tasks requiring a great deal of processor power, such as reports and payroll, are completed quickly, freeing the system for an even greater workload. In fact, with the VAX 8600 in a VAXcluster, jobs from virtually every department in your organization can run simultaneously.

The majority of computer instructions, those involving a single operation, require several successive actions by the processor to complete. Low-speed computer systems perform the entire sequence of actions for one instruction at a time. Actions on the next instruction can't begin until those on the current instruction have been completed, wasting time and compute power. Cache, faster auxiliary memory, was developed to solve this problem and is a simple level of pipelining. The VAX-11/780 shortened the execution time of instructions by prefetching. While the sequence of actions is being performed on one instruction, another instruction is taken from memory, or prefetched, and stored in cache memory. The VAX 8600 carries this technique a step further by working on the entire sequence of actions at once, keeping the processor circuits busy with successive instructions – pipelining.

Pipelined processing is much like a sophisticated car assembly plant. A series of cars (instructions) is worked on – successive actions being performed on each car through the assembly line (pipeline). As the engine is secured onto the chassis of a car, the same car proceeds to the next station in the assembly line and is fitted with a radiator. It enters the painting station as the car behind it enters the radiator station. Time is not wasted while one car is completed but, instead, all stations are kept in constant use. Just as the assembly line is kept filled with cars, the pipeline is kept filled with instructions to maximize its performance. All VAX 8600 memory operations are pipelined, making it our most efficient VAX.

Performance and reliability go hand in hand. – The highest-performance computers available are valuable to you only when your applications can be run on them. New methods of error logging, analysis, and recovery, formerly used in much larger machines, have been built into the VAX 8600. The VAX 8600 features an error analysis tool, VAXsim, used by all VAX processors, that displays a graphic image of the system, even an entire VAXcluster, highlighting any component that is exhibiting a fault. In the high-availability environment of a VAXcluster, this increased reliability of the VAX 8600 equates to more service to your users as they face increasingly complex tasks in their work.

The combination of performance and reliability that's built into every VAX 8600 will surpass even the high expectations that your users have for VAX systems. Combined with your computer strategy, your present investment can only result in future gain.


PROFIT BY MONTH
SALES
EXPENSES
PROFIT

You Expect Your Business to Outgrow Its Resources.

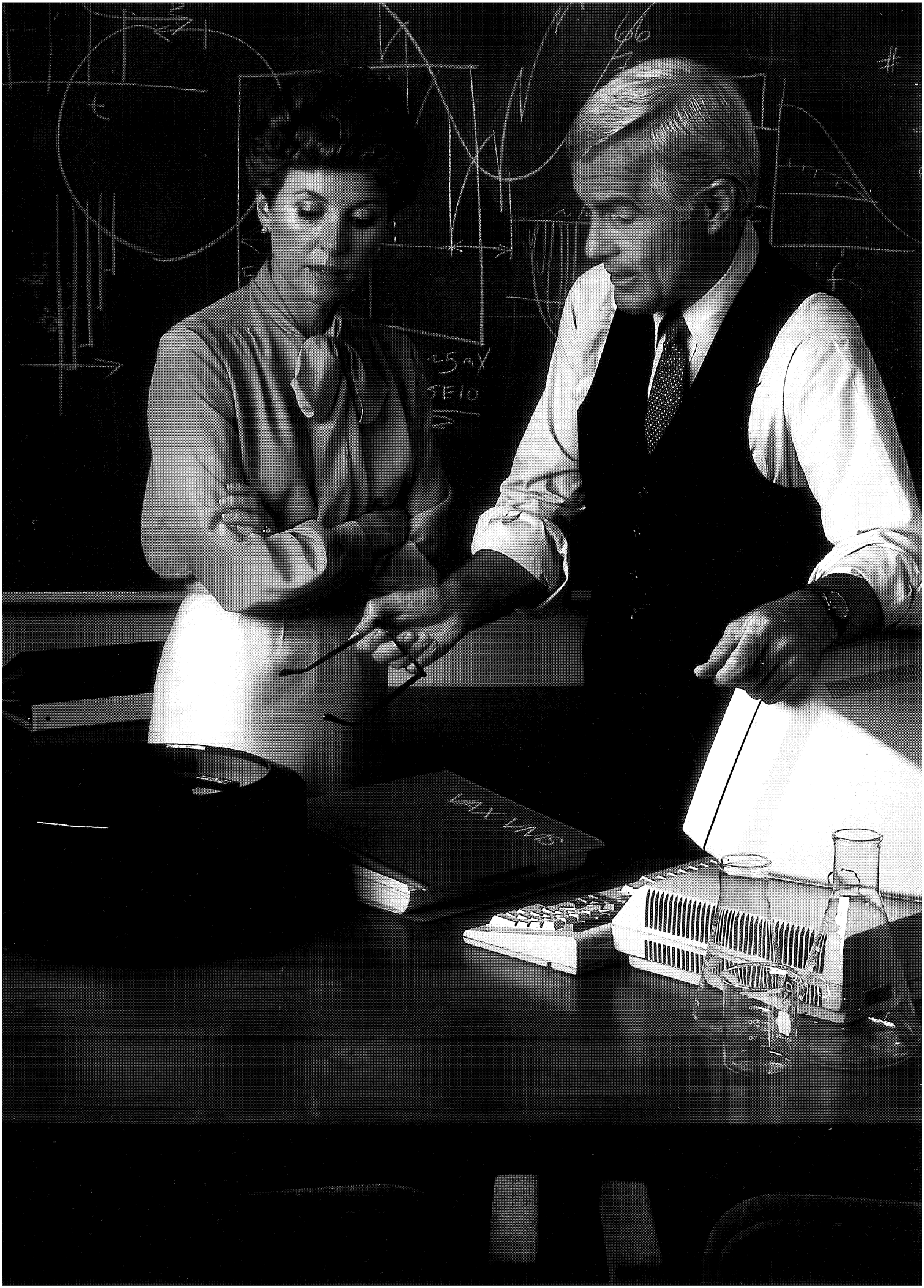
Just as you expect to outgrow your workspace, you expect your business to outgrow its computer resources. If you can add to the computer resources that you own, expand these resources later, and link all of this together, your computer strategy has taken care of future growth.

In engineering and manufacturing environments the VAX 8600 in your VAXcluster will run the very large programs that always had to be run on mainframes. This means that applications such as stress and finite element analysis can be run on-site along with your other applications. This same VAX 8600 runs your existing VMS programs without change—at up to 4.2 times the speed of the industry-standard VAX-11/780. Now, you can stay in the VAX/VMS computing environment to do even your most ambitious data processing.

Unique in the industry, a VAXcluster is a system of several VAX computers and storage devices. Up to 16 components in designated configurations, including VAX 8600, VAX-11/780 series (VAX-11/780, VAX-11/782, and VAX-11/785), and VAX-11/750 processors, and intelligent storage subsystems, function as a single, large, highly powerful system. A VAXcluster with two of the above mentioned VAX processors adds high availability to the list of benefits. Information can be compiled for your Board of Directors meeting at the same time that engineers are designing models with CAD/CAM workstations. You can expand almost without limit to take care of new applications. If one of your departments grows faster than others, add a system that will support only that department—linked to the VAXcluster to access important data. In a VAXcluster system users can share all data, which is updated efficiently so you don't have to worry about lack of current information. Place a VAX 8600 in the departments that need more power and smaller VAX systems in departments needing less.

Advanced clustering capabilities and more...with our VMS operating system.—Many users in your organization need access to the same data. Until recently, the process of sharing information among computer systems was cumbersome. VAXclusters and the clustering features of the VMS operating system streamline this process. With the clustering capabilities of our VMS software, you can take advantage of VAXcluster hardware features. With two paths for transferring data, the hardware will always use one path if the other one is inoperative. If one VAX processor in the VAXcluster becomes unavailable, your users can continue running applications on the remaining processors. Your data and computer resources are ready when your users need them. So, while VMS takes care of availability, your data processing managers have more time to take care of business.

The Digital Storage Architecture (DSA) and the VMS operating system work together to ensure that your data is always available to be shared by *all* VAXcluster users. When intelligent storage subsystems are used in a VAXcluster, VMS transparently switches to the working storage subsystem if one becomes unavailable. That's the resource sharing and efficiency that you need to keep users in your organization ahead of increasing workloads. Our VMS operating system software is the same for all VAX processors, so users don't have to relearn or reprogram when they move from system to system or department to department. Both software and people move with ease.


VAX VMS

Sound Business Reasons for Insisting on Portable Applications.

Software development and maintenance are costly and time-consuming. The more you can do to shorten that time, the more efficiently you can use your computer resources. With the VAX family of computer systems, your applications can move from one department to another, from one system to another, without change, without rewrite. You could even move an entire department to a new building in another city, reconfiguring new and old VAX processors to meet your needs, and your software would run and all of your users could begin work on any VAX with the same VMS operating system that they're familiar with. Our computer systems are compatible and that makes your applications portable.

With Digital and VAX, you aren't restricted to today's methods of communicating or to today's products. Later, if you have to expand to other facilities, add users, or purchase additional computer resources, the flexibility and choice in the VAX family plus the wide range of application possibilities let you do what's best for your business needs. Because the VAX family has always been engineered to a plan, what you buy today will still be useful to you tomorrow.

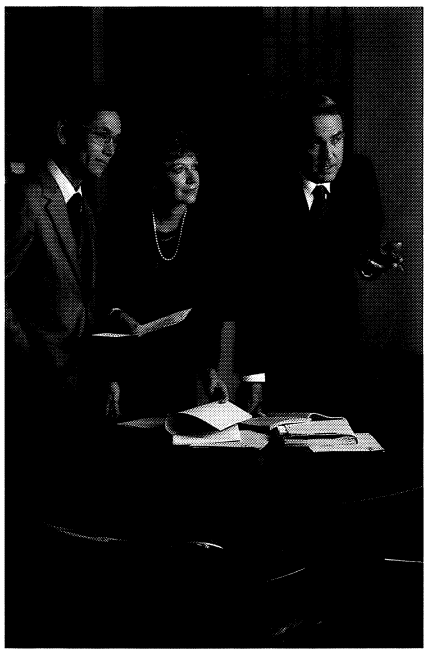
From our smallest VAX, the MicroVAX I, to our largest single processor, the VAX 8600, and VAXcluster Systems, all processors run the same VMS software. With the introduction of the VAX 8600, the VAX family can handle all of your needs, right up through your large programs. The similar implementation of central processing units throughout the VAX family assures you that all of your VAX systems will work together. For more specific information on the entire VAX family, ask for the VAX Family Brochure. Now, experience the efficiency of the VAX/VMS environment for all of your computing requirements.

Your data processing managers may have information management problems like no one else.—No one can write software to suit everyone. That's why many companies develop their own. One method of managing information is right for your financial department while another method may be suitable for your personnel department. With the VAX/VMS software products, you have the programming languages and tools to develop what you need—when you need it—on the most versatile operating system in the industry.

The VAX Information Architecture is an integrated system of information management tools. By *integrated system* we mean that all of the tools can work together and can access information from the same database. Whether users need to develop easy-to-understand tables or complex reports, create graphs and slides for presentations, or develop their own forms to keep track of information, the VAX Information Architecture can help.

Choose the combination of these tools to streamline operations now and add new methods as your needs change. Our architecture assures you that future information management capabilities will work alongside the proven ones that meet your needs today.

Communication with Other Systems Won't Complicate Your Workday.



Your organization may have computer systems at the desk, department, and data center levels. You may even have equipment from several different computer manufacturers. Dealing with several types of equipment can be confusing and time-consuming. Digital's networking products take care of the problems caused by mixing equipment from several sources. And, our DECnet software and hardware connect users from one Digital computer to another. Whether they communicate within a building or around the world, people in all parts of your organization can keep informed–without complicating their workday.

Our DECnet/SNA Gateway gives you the capability to communicate and share information between a Digital network and an IBM SNA network. You can work on a program-to-program basis or send large batch jobs over the network to a computer system in another department. Other industry-standard communication protocols allow you to exchange information from country to country via telephone lines.

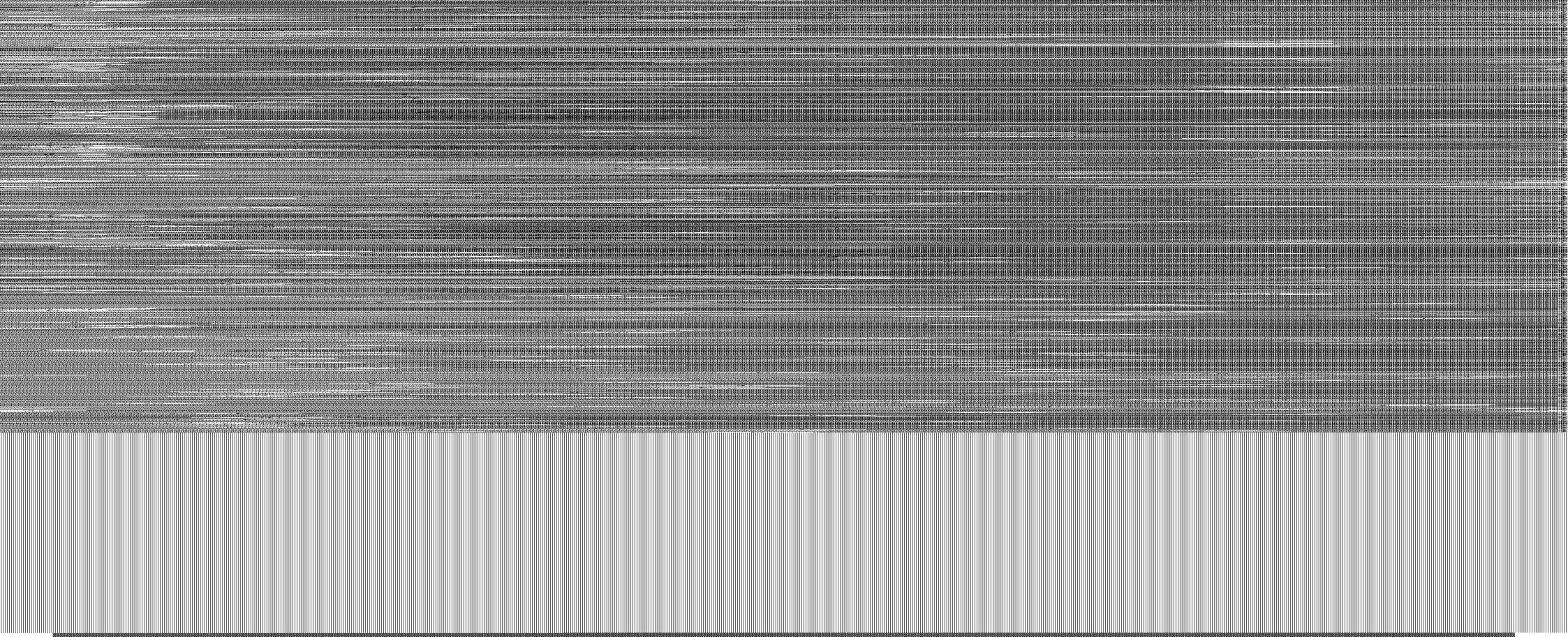
Because your VAX 8600 takes advantage of this networking architecture, you're assured that it will fit in with your current systems without disrupting your employees. As we constantly expand our networking capabilities your business will benefit from the latest in communication technology because you can add what you need in the future.

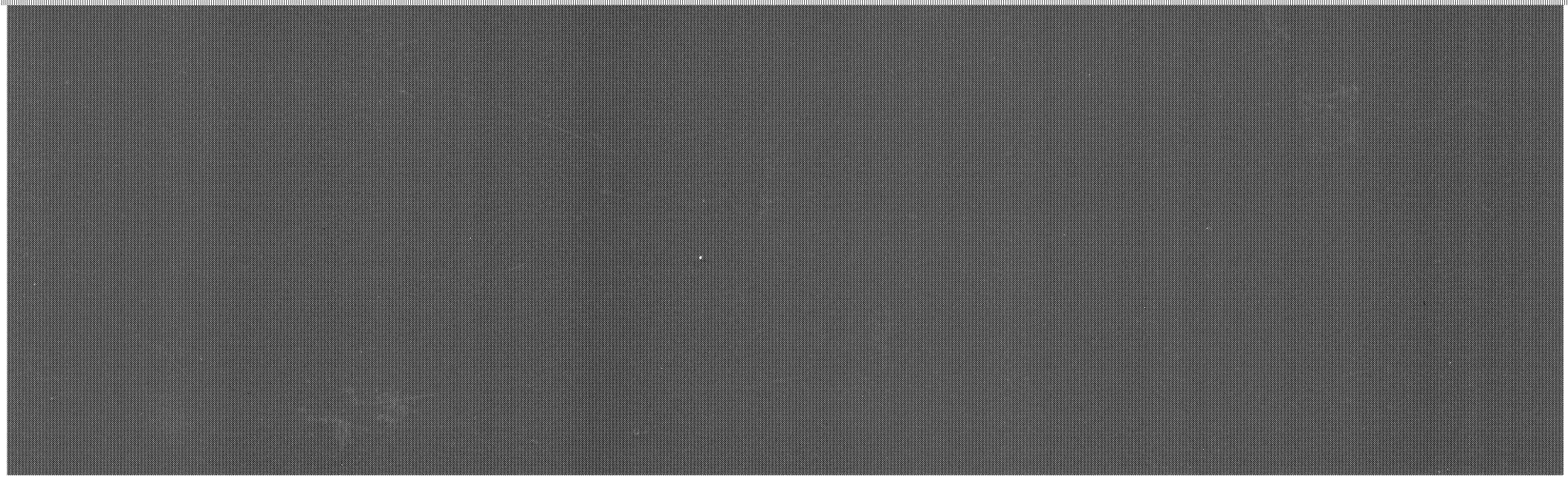
Don't Trust Just Any Computer Company With Your Success.

Your business has been successful because you've made it that way. As any organization grows, there are more responsible parties, more decision-makers, more complications. You need to know that your data processing people have the best tools to work with so that they can do their work and keep you informed. And that the systems you buy are backed by a reputation for quality and reliability.

It doesn't make sense to buy a system that is efficient only for today, that has little chance of growth to keep up with your needs. To get the most from your investment in a large system, also consider its potential for connecting to other systems, for sharing data, and for using a standard operating system–as well as its power to run very large programs.

The advantages are clear. The VAX 8600 is a VAX. And with the VAX family you can place your users in a homogeneous VAX/VMS environment that includes any combination of VAX systems, with software advantages that can be shared by every person in your organization, and with Digital service and support. There's no need to buy a larger, more expensive system when the VAX family–now with the VAX 8600–can handle your success.





digital

VAX-11/785 Product Bulletin

Extending VAX Performance Upward

The VAX-11/785. Raising the Industry Standard VAX Performance.
Digital Equipment Corporation builds the industry standard series of 32-bit computers. It's called the VAX Family of Systems.

No one else has a series that offers you so many system choices and so many proven subsystem options – with one common operating environment. No matter what level of performance you require, we have a VAX system to meet your needs.

We began the VAX family with the powerful VAX-11/780 system. It quickly became the standard for 32-bit computing. Next, we extended the family downward to smaller systems, including the midsized VAX-11/750 system, the VAX-11/730, the VAX-11/725, and the MicroVAX. This range of systems means you can offload your VAX applications downward or extend your VAX applications upward without converting software.

Now, we add the VAX-11/785 system to the VAX family. This is the first single-processor system to offer higher performance than the VAX-11/780 system. This extended VAX performance results from recent advances in CPU technology. These advances speed up the flow of data through the processor. Higher throughput improves response time for realtime and office information applications. Higher throughput also increases the capacity for more computation in compute-bound applications. And higher throughput means you can add more users in commercial timesharing applications.

The VAX-11/785 system offers you more performance than any VAX that has come before. And that means the industry standard for performance has just been raised.

Highlights

- The high-speed VAX-11/785 CPU shortens program execution time in all VAX applications.
- Rapid response time improves performance in realtime applications, such as simulation and monitoring.
- An optional FP785 floating point accelerator uses high-speed CPU design to provide faster scientific computation.
- The VAX-11/785 system is compatible with the VAX-11/780 subsystem options.
- The VAX-11/785 computer enhances the performance of VAXcluster systems.

And the HSC50 controller does more. This intelligent subsystem handles the storage devices for our VAXclusters. Linking as many as 16 nodes of mid-sized VAX systems, larger VAX systems, and HSC50 controllers, a VAXcluster system offers you something unique – more system management flexibility for your VAX computers.

A VAXcluster system can be viewed as both an expandable single system, and a more efficient and powerful distributed processing system. With a VAXcluster system, you can increase your processing capabilities and yet retain common access to information in shared mass storage. You can pool a number of systems for high availability and high data integrity. And you can optimize the workload of your applications among several systems at once.

Obviously, the more powerful the individual processors in your VAXcluster, the more powerful the overall system. That's why the VAX-11/785 computer is an ideal system for use in a VAXcluster: it brings high-speed performance and increased system capacity to these applications. That can mean more powerful monitoring and controlling, data reduction and analysis, simulation, and extensive transaction processing.

No Vendor Offers You More Comprehensive Networking.
VAXclusters include the VAX-11/750, VAX-11/780, VAX-11/782, and VAX-11/785 systems. Whether or not you use the VAX-11/785 in a VAXcluster, you can always provide communications from your VAX-11/785 to the smaller VAX systems and other Digital systems – like the PDP-11 series and the personal computer series – as well as non-Digital systems. Digital can be your single source for all the hardware and software you need for communications.

In fact, no other vendor offers you such comprehensive networking capabilities as Digital. Whether your systems are close enough to link by direct cable or are separated by greater distances, Digital has the hardware and software devices for every need. Like Ethernet. Digital's Ethernet hardware makes it easy and efficient to directly link many heterogeneous computers in the same general location, such as a large building. Plus, we offer modems to extend communications beyond this range.

DECnet layered software handles the protocols and interfaces required to communicate with most of today's computers. DECnet-VAX is designed specifically to link your VAX-11/785 processor with the world at large. It handles all interprocessor messages on the network itself, including remote access, and the software provides gateway access to public packet-switching networks.

Our Software Makes the VAX-11/785 Ideal for Office Systems and Information Management.
Large timesharing systems with information management applications demand I/O activity for the many users who frequently access files from their terminals. Our VAX Information Architecture is designed to help off-load the processor from the heavy I/O load these applications demand. This software includes programs for distributed access, CODASYL and relational database management, and timesharing terminal management.

Our information management capabilities also include online query and report writing, forms, and graphics so that you can easily access data and produce information, on a terminal screen or in hardcopy, right in your own office.

Our ALL-IN-1 office information software makes the VAX-11/785 as easy to use as any desktop personal computer. The ALL-IN-1 package has electronic mail, an online calculator, a calendar facility, electronic filing, and extensive text processing capabilities. So your VAX-11/785 system is instantly suitable for all office uses. And people accustomed to word processing equipment will find it familiar and easy to use the VAX-11/785 system as an office machine with the ALL-IN-1 software.

Tailor Your Own VAX-11/785 System.
When you order the VAX-11/785, you can tailor it to the needs of your specific application. Choose from four VAX-11/785 system building blocks: a stand-alone system, an upgrade kit for a stand-alone system, a VAXcluster starter, or a VAXcluster upgrade kit. Once you select the system model, you can configure the rest of the system from an extensive menu of hardware, software, and service options. Call or write your nearest Digital Equipment Corporation sales office for further details about ordering VAX-11/785 systems.


digital VAX 11/785

Now There Is a More Powerful VAX System.
Demands on computer resources increase over time. That's why you need to select a computer system that provides upward mobility for your applications. So that you can add new users or improve performance without disrupting ongoing applications.

The VAX family of computers offers this mobility. Since VAX software runs on all VAX systems, you can switch from one VAX system to another without modifying your software.

For instance, you can start with the MicroVAX. When you need to add more users or faster performance, you can move to the VAX-11/725, VAX-11/730, or the VAX-11/750. Later, and again without converting software or rewriting code, you can move your applications to the more powerful VAX-11/780 system.

Now you can go one more step upward – with the high-speed performance of the VAX-11/785 system. The new VAX-11/785 central processor delivers faster program execution and increased system capacity to your larger VAX applications.

The Real Added Value of a Faster VAX.
What makes the high speed of the VAX-11/785 processor so attractive is that it gives you greater system capacity. Faster processing frees up the system sooner – allowing more program execution in less time. This increases the overall capacity of your system.

Almost any large-scale VAX application can be expanded with the VAX-11/785 system. Timesharing environments – like department-wide software development or database management, office automation or computer-based instruction – can take on new users. And most compute-intensive applications, such as high-energy physics, econometrics, engineering, CAD/CAM, and simulation can accommodate more programs.

No matter what your application, the added capacity can save you money. Because it can do the work of two smaller systems. This saves on overhead and reduces the cost-of-computation and cost per user.

Advanced CPU Design for Faster Performance.
The accelerated speed of the VAX-11/785 processor is the result of advances in CPU design. These advances implement technology recognized by the industry as proven and reliable. Advanced circuitry and a faster internal clock accelerate the CPU cycle time to 133 nanoseconds, significantly faster than any other VAX processor.

With the accelerated processor design you get better interrupt response time. This contributes to more efficient performance in realtime applications, including simulation, device tracking and other monitoring and control systems.

And the large VAX-11/785 cache, at 32 Kbytes, accommodates more data and instructions than most other computer caches, so it needs to be updated less frequently. Programs with many subroutine jumps, such as CAD/CAM, simulation, and graphics systems, benefit most from larger cache size.

And the number of standard data types have been increased. Floating point G- (double precision) and H- (quadruple precision) data types, which are standard on the VAX-11/785 system, provide up to 33 digit accuracy. As an option, you may add a floating point accelerator, which accelerates throughput up to 40 percent in most applications.

The Flexibility to Tailor Your VAX-11/785 System with Proven VAX Options.
In addition to high-speed processor performance, you need a vendor who offers a wide choice of hardware and software subsystems.

You need all kinds of peripheral devices, as well as the means to communicate with other computers. You need operating system software that makes it easy for you to take programs developed in multiuser environments and run them in realtime environments. And you need software packages that make it easy for people to use computers for office and information management tasks.

Digital Equipment Corporation's VAX hardware and VMS operating software meet all these computer needs. Your Digital sales representative will help you choose the options best suited to your applications.

The Added Value of Intelligent Mass Storage. From Standalone Systems to VAXcluster Systems.
Digital's new intelligent mass storage controllers, including the HSC50 hierarchical storage controller for the VAX-11/785 system, contain microcomputer-based processors. These processors control storage activities for a number of large mass storage devices – including the RA-80, RA81, and RA60 disk drives – attached to your VAX-11/785 system. These control activities have to be handled by the system CPU in other systems. By offloading storage control activities, the HSC50 controller helps the VAX-11/785 CPU maintain its high-speed performance. Especially in timesharing applications, office applications, and other applications with frequent disk I/O activity.

digital

Processor	
Microcontrol store instruction time	133 ns
Control store size	8K words (.5 ROM, 7.5K Read/Write –including 1K reserved for UCS)
Internal data path	32 bits
Maximum system I/O rate	13.3 Mbytes/sec
Instruction buffer size	8-byte lookahead
Memory cache	32 Kbytes, 2-way set associative
VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating point (G and H), packed decimal, character string, variable bit fields, and numeric strings
Main Memory	
Virtual address space	4 Gbytes
Physical expansion	32 Mbytes in 2 Mbyte increments (plus 4 Mbytes multiport memory for 36 Mbyte maximum)
Parity	8-bit error correcting code per 64-bit quadword
Cycle times	600 ns per 64-bit read (700 ns with single bit errors), 700 ns per 64-bit write
Operating Environment*	
Temperature	15-32°C (59-90°F)
Relative humidity	20-80%
Maximum altitude	2,400 m (8,000 ft)
Maximum heat dissipation	2,166 kcal/h (8,530 Btu/h)
Processor Power Requirements*	
AC line voltage	120/208 V
Frequency tolerance	59-61 Hz
Phases	3
Also available	240/415 V 50 Hz
Maximum AC power consumption	2,500 W
Physical Characteristics*	
Weight	498 kg (1,100 lbs)
Height	153.7 cm (60.5 in)
Width	118.1 cm (46.5 in)
Depth	76.2 cm (30 in)

*Note: These specifications for processor with CPU, 1 DW780, 2 MB MS780-E Memory, and without Floating Point Accelerator option.

digital

The First System Designed for Large-Scale VAX Computing.

Today, one system can deliver 4.2 times the performance of the industry-standard VAX-11/780 computer.

The VAX 8600 system, the most powerful of the VAX family of systems, has the speed, performance, and capacity you need to expand applications into the realm of large-scale computing. This single VAX system provides the high-speed processing large applications need, the powerful performance large complex and large multiuser applications need, and the I/O capacity large timesharing applications need, in technical, scientific, commercial, academic, and research environments.

And with the VAX 8600 system, years of proven VAX experience are brought to your large-scale applications. Proven VMS software for program development, networking, and information management can now be applied to these larger VAX applications. And if you are already using VAX systems, all of your investments in equipment, training, and software are preserved. So put your large applications on a VAX, the powerful VAX 8600 system.

Highlights

- Provides the greatest VAX performance ever—up to 4.2 times the industry-standard VAX-11/780.
- Supplies the highest CPU performance for VAXcluster multiprocessing systems.
- Has the fastest VAX throughput in scientific and technical applications.
- Offers the most I/O throughput for VAX subsystems.
- Protects your investment in VAX subsystems and VMS software while letting you expand VAX applications.

Introducing Large-Scale VAX Computing.

With the VAX 8600 system, Digital raises the range of VAX computing to encompass large-scale applications. The VAX 8600 system can support far more extensive scientific, engineering, and realtime applications – such as artificial intelligence, simulation, and computer-aided design – than any previous VAX system. And the VAX 8600 system can easily support the large general purpose timesharing applications often found in government, commercial, administrative, and academic environments.

Yet the VAX 8600 system is far less expensive to own and operate than the systems that typically support this size of application. That's because the VAX 8600 processor brings proven technologies, once thought to be the domain of much larger systems, to VAX computing.

Technologies and Techniques Speed Up System Throughput.

The VAX 8600 system's central processing unit (CPU) incorporates the fastest technologies of any VAX processor. High-density semiconductor components, using emitter-coupled logic (ECL) gate array circuit technology, lower the VAX 8600 cycle time to 80 nanoseconds. This provides the basis for very fast processing in any VAX 8600 application.

A write-back cache helps the VAX 8600 CPU maintain its processing speed. With this cache technique, the CPU never waits for the updating of main memory, which operates at a slower cycle time.

And a floating point accelerator speeds up throughput in scientific and technical applications. It speeds up all four VAX floating point data types, including IEEE double-precision and extended-precision floating point types, and improves integer multiply performance.

New VAX CPU Design Increases Performance.

It takes more than just faster components to make a system able to handle the extensive workloads of large-scale applications. A large system must also be able to work harder. And the VAX 8600 system does just that – thanks to a CPU designed to provide greater performance.

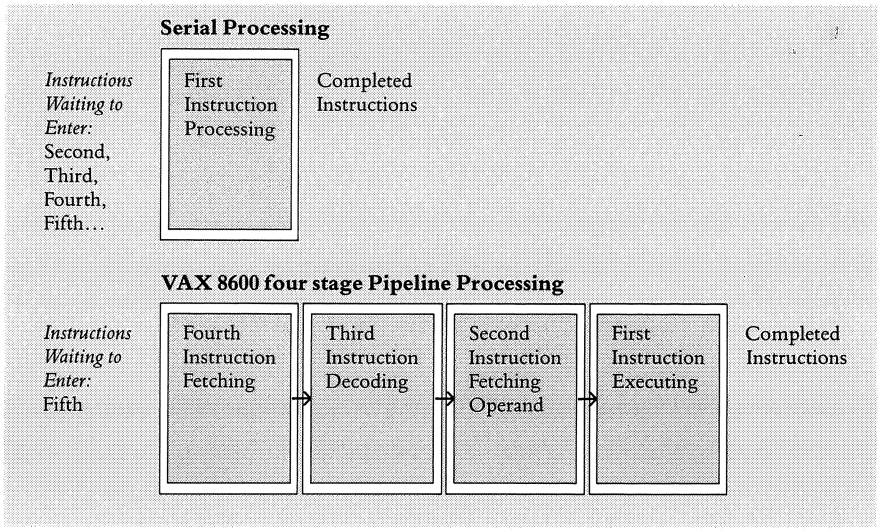
The VAX 8600 CPU uses sophisticated four-stage pipelined processing. This means that the processor can work on four instructions simultaneously.

Contrasted with the serial processor, in which each instruction must be completed before another can begin, a fully loaded pipelined processor can be doing much more work at once, which means greater overall system performance.

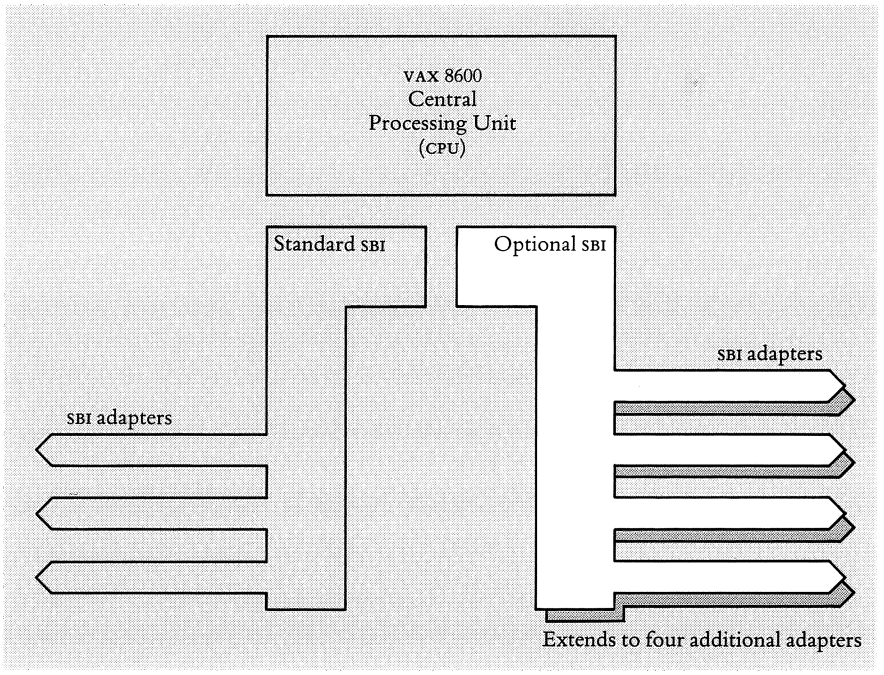
The internal communications of a VAX 8600 CPU are also uniquely designed. Instead of using the synchronous backplane interconnect (SBI) for all CPU communications, the VAX 8600 processor uses a separate channel for main memory, reserving the SBI for peripherals. This keeps the CPU constantly supplied with data and instructions, while it doubles SBI efficiency for peripheral I/O.

A Second SBI Option Extends Adapters for Input and Ouput.

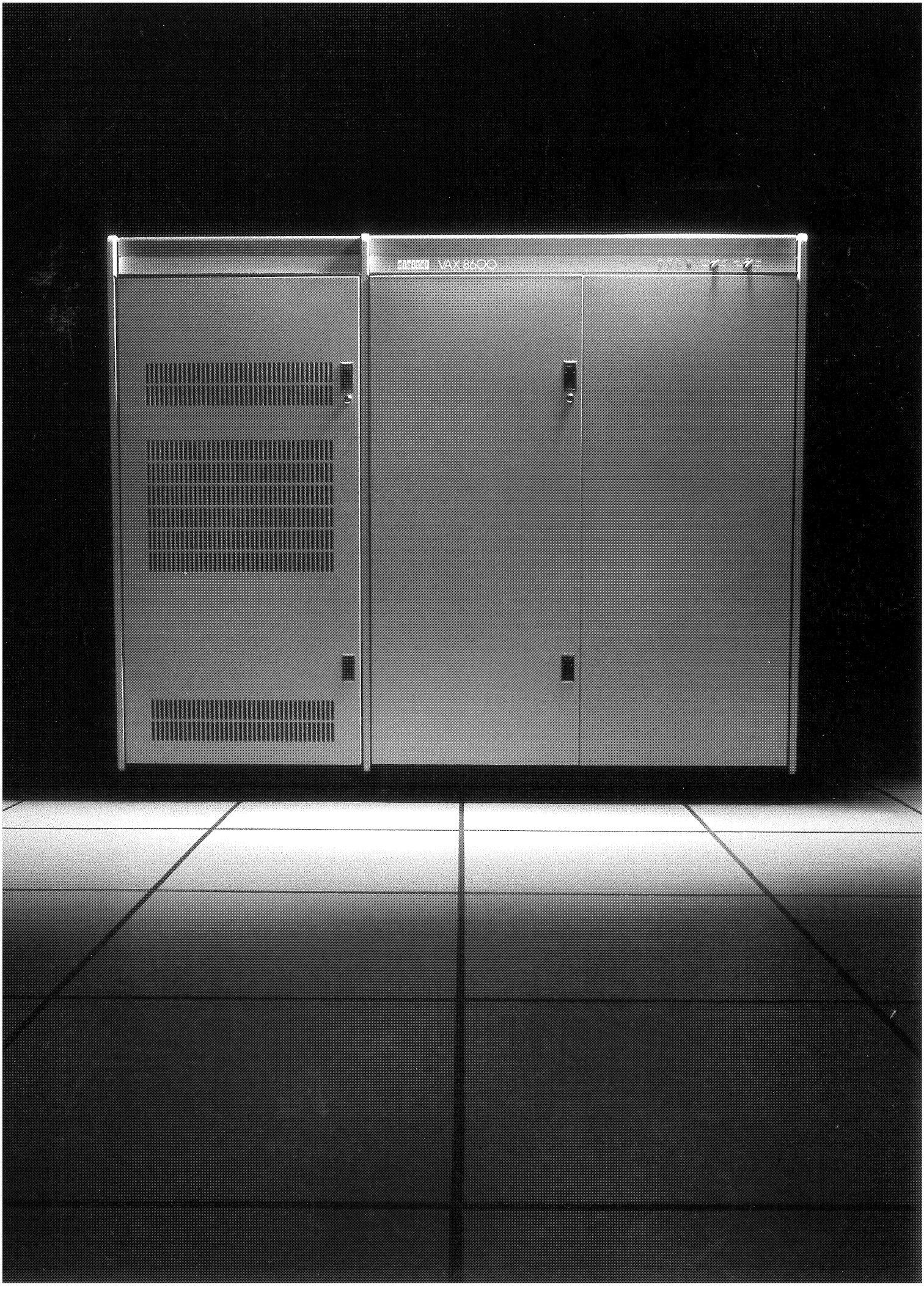
And, the VAX 8600 system uses the same SBI adapters that the VAX-11/780 uses so you get the same configuration flexibility to support a wide variety of VAX applications. In addition, you can enlarge your configuration with even more devices. Up to 8 SBI adapters on an optional second SBI give you a VAX 8600 system with a grand total of 11 SBI adapters. No previous VAX system offers this much I/O.



Symbolic comparison of processing techniques. One generalized processing unit means the standard serial processor can handle only one instruction at a time. Four sequential and specialized processing units mean the VAX 8600 pipelined processor can handle as many as four instructions at a time, for a much higher rate of throughput.



Unique to the VAX 8600 system, the optional second SBI extends VAX configuration flexibility to more I/O devices.


VAX 8600

Fewer Chips and More Diagnostics Lower Maintenance Costs.

System dependability is a vital ingredient of cost-effective computing in large applications. If a larger system is not available or has high maintenance costs, how can you realize its economies of scale? This is why the VAX 8600 is designed inside and out with self-diagnostic and self-maintenance features: to ensure dependable performance for large applications.

First, there are actually fewer components to maintain – thanks to high-density chips. Second, error checking and correcting (ECC) memory and cache automatically correct many types of errors without interrupting operations. If there ever is a problem, a diagnostic bus isolates it – and the system often circumvents it. A floating point accelerator diagnosed with a problem, for example, is simply turned off so that programs can continue to execute.

Environmental sensors are used to monitor the VAX 8600 system's physical environment. This safeguards the system from adverse conditions. If air conditioning fails, for instance, the environmental sensors issue warnings and stand by to shut the whole system down before costly damages can occur.

Compact System Design Minimizes Operating Costs.

All other things being equal, VAX 8600 overhead costs are far less than those of other large systems. In fact, the cost of ownership, including initial investment and five years of maintenance, is about half what you'd pay for previous large systems.

This is only partly due to the maintenance features. It is also because the VAX 8600 system is physically smaller than most large systems. The VAX 8600 CPU chips, including the latest 256K memory chips, take up less space. In fact, the VAX 8600 processor cabinet can hold up to 32 Mbytes of memory, which means your fully configured VAX 8600 system could have an even smaller footprint than a comparably configured VAX-11/780. So floorspace, electrical power, and air conditioning all cost less.

VMS Makes Program Development Easy and Applications Portable.

While the VAX 8600 *processor* is unique, the VAX 8600 *system* has much in common with the other VAX systems. The VMS virtual memory operating system provides the same software operating environment for the whole range of VAX computers, from the MicroVAX I to the VAX 8600 system. This gives you application "portability" among the different VAX systems, which makes it easy to move existing VMS applications upward to the VAX 8600 system, and to move programs developed on the VAX 8600 system to other VAX computers.

The VMS operating system, using the common VAX instruction set and having access to 4 gigabytes of virtual address space, can serve any type of VAX application – realtime, timesharing, and multiprocessing. And VMS utilities make programming easier by creating a common language environment that lets programmers share code to shorten development time and conserve system space.

Digital and other vendors have developed thousands of VMS software application packages, ranging from accounting to word processing to scientific data analysis. So chances are there's software already developed for your application. And Digital's software leads the industry in many areas, including networking and information management products.

VAX Information Architecture: To Organize Large Amounts of Data.

Large computers often support information systems used by entire organizations. These systems can grow very complex very quickly. Digital's VAX Information Architecture can help you keep large-scale information systems, such as transaction processing and commercial timesharing, organized and easy to maintain.

The VAX Information Architecture software is a method and style of design that makes information systems easier to develop and easier to use – from the point of view of novice computer users, managers, data processing specialists, as well as programmers. And since VAX Information Architecture products run across the entire spectrum of VAX computers, previous VAX information system training is preserved. What's more, users benefit from the ability to choose software that meets their own specific needs. They can use spreadsheet, query and report writers, forms generators, and now videotext for the office.

Programmers and system managers benefit from the central data dictionary (CDD). Along with code and program library management tools, the CDD eliminates redundancy in the creation, storage and retrieval of data for information system applications. And it's possible to match database software to information system complexity – with DBMS for more complex hierarchical databases – to Rdb/VMS for relational databases. Then, using DECnet-VAX networking software, VAX information systems can be distributed across several VAX computers for remote access and data manipulation.

Digital's Network Capabilities are Far-Reaching.

Large systems are, and often have attached to them, scarce and expensive resources that users at remote sites need to access. If you need to provide remote access to your VAX 8600 system, you'll find all the tools you need with the integrated hardware and software of the Digital Network Architecture (DNA).

DNA software and Ethernet hardware provide easy access and growth potential for local area networks (LANs). Local area networks permit the sharing of resources among all systems within a limited area such as a large building. LANs also provide automatic failover and distributed processing. And Ethernet hardware makes it easy to install up to a thousand systems on a LAN, while providing higher reliability with less cabling than most other methods.

For access beyond the local area, DNA supports the protocols required for all types of wide area networks, including DECnet networks, SNA gateways to other vendors' networks, and packet-switched data networks such as X.25.

Proven Subsystems Supported by the VAX 8600.

Another advantage of the VAX 8600 system is that it uses existing, proven VAX subsystems that are already available. Standard UNIBUS, MASSBUS, general purpose, and Computer Interconnect subsystems are all supported by the VAX 8600 system. So are the same communications interfaces, terminal servers, printers, terminals, and workstations many other systems use. This protects the investments you may already have, as well as those you make in VAX equipment now and in the future.

VAX 8600 Systems and VAXclusters.

And if you are planning on a very large application, there is one VAX system that guarantees future expansion – on a very grand scale – in very reasonable increments. This is the VAXcluster, which gives you the opportunity to obtain tremendously powerful levels of VAX computing for large applications.

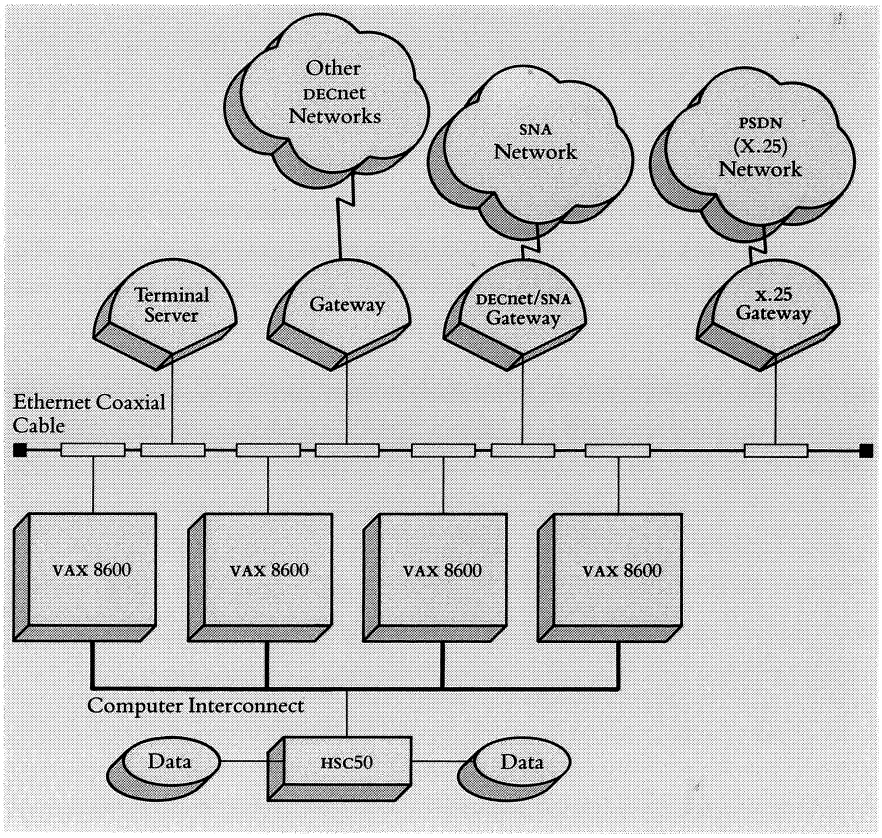
The VAXcluster helps you optimize the power of your VAX computer resources. A VAXcluster makes it possible to balance I/O and processing loads among as many as 16 VAX systems and HSC50 intelligent mass storage controllers at once. And your users get high system availability and global data sharing from VAXcluster

systems. VAX-11/750, VAX-11/780, VAX-11/782 and VAX-11/785 systems, as well as the 8600, can all be interconnected–so you retain system continuity as you pool these computers into VAXclusters for expanded applications.

And, when you consider that the VAX 8600 system can deliver the performance of over four VAX-11/780s, for the price of two VAX-11/780s, in the space of only one VAX-11/780, you can see that the VAX 8600 system is the most economical choice for very large VAXcluster applications.

Putting Your Large Application On a VAX 8600 System.

So call your local Digital sales office today. A Digital sales representative will help you determine which of the VAX 8600 system options, as well as field service and software licensing agreements, best serve your large system needs. And we'll help your programmers and operators with education and training on the VAX 8600 and other VAX systems at your own site or at one of Digital's many training centers worldwide.



A VAXcluster allows you to couple your VAX systems into one single multiprocessing environment in order to balance workloads, share data, and provide high availability. In addition, Digital Network Architecture provides full networking capabilities from each VAX 8600 system to other computers in and beyond the local area.

digital

VAX 8600 System Specifications

Processor	
Microcontrol store instruction time	80 ns
Control store size	8K words–86 bits
Internal data path	32 bits
Instruction buffer size	8-byte lookahead
Memory cache	16 Kbyte write-back
VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating point (F, D, G and H), packed decimal, character string, variable bit fields, and numeric strings
Main Memory	
Virtual address capacity	4 Gbytes
Physical expansion capacity	32 Mbytes in 4-Mbyte increments
Parity	7-bit error correcting code per 32-bit quadword
Memory cycle times	560 ns per 128-bit read
Operating Environment	
Temperature	15-32°C (59-90°F)
Relative humidity	20-80%
Maximum altitude	2,400 m (8,000 ft)
Maximum heat dissipation	22,000 Btu/h
Processor Power Requirements	
AC line voltage	120/208 VRMS
Frequency tolerance	47-63 Hz
Phases	3
Also available	240/415 V 50 Hz
Maximum AC power consumption	6,500 W/10 KVA
Physical Characteristics	
Weight	782.1 kg (1,725 lb)
Height	153.7 cm (60.5 in)
Width	188.1 cm (74 in)
Depth	76.2 cm (30 in)

Printed in U.S.A. ED 25992 46/84 09 08 1500 Copyright © 1984 Digital Equipment Corporation. All rights reserved