digital

VAX/VMS
Summary Description and Glossary

Order No. AA-D022B-TE

VAX11

March 1980

This document introduces the basic concepts of the VAX/VMS operating system.

VAX/VMS
Summary Description and Glossary

Order No. AA-D022B-TE

SUPERSESSION/UPDATE INFORMATION:	This revised document replaces the VAX/VMS Summary Description (Order No. AA-D022A-TE).
OPERATING SYSTEM AND VERSION:	VAX/VMS V02
SOFTWARE VERSION:	VAX/VMS V02

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, August 1978
Revised, March 1980

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

Contents

Chapter 5 VAX/VMS I/O System

Chapter 6 Languages, Libraries, Linking, and Sharing

Chapter 7 System Management and Operation

Chapter 8 Compatibility Mode

Appendix A Process States and System Events

Glossary

Index

Figures

Tables

Preface

Together, the *VAX/VMS Primer* and the *VAX/VMS Summary Description and Glossary* introduce the VAX/VMS operating system.

The primer introduces new VAX/VMS users and application programmers to the operation of the system; that is, it provides tutorial information about the DIGITAL command language (DCL) and other VMS features.

The summary description, on the other hand, presents system concepts of interest to such advanced users as system programmers and system managers. The first chapter describes aspects of the VAX–11 processor that are closely related to system programming; the remaining chapters describe the VAX/VMS operating system.

Also included in this document is the VAX–11 glossary. Key terms that are in boldface in the main part of the document are defined in the glossary.

The *VAX–11 Information Directory and Index* describes the full VAX–11 document set.

Chapter 1
Introduction

The VAX/VMS **virtual memory** operating system runs on a VAX-11 processor and takes full advantage of the processor's capabilities. The following VAX-11 features used by VAX/VMS are discussed in this chapter:

- Hardware-maintained context for each user program
- 32-bit addresses
- The four processor access modes
- A stack for each processor access mode and an interrupt stack
- Native and compatibility mode instruction sets

Refer to the *VAX-11 Architecture Handbook* and the appropriate hardware handbook for details on the full range of VAX-11 features.

1.1 Hardware Definition of a Process

The VAX-11 hardware executes processes; these, in turn, execute user programs. That is, a **process** is the execution agent recognized by the VAX-11 processor. The VAX-11 processor defines a process using the values that are loaded into the processor registers when the process is scheduled for execution. The *VAX-11/780 Hardware Handbook* describes these registers.

Only one process can execute at a time, although many may be known to the system. When a process is not being executed, its **hardware process control block (hardware PCB)** contains the process's register values. When VAX/VMS requests that the execution of one process be interrupted and the execution of another process start, the processor stores the interrupted process's register values in its hardware PCB and loads the next process's register values from its hardware PCB. This sequence is called **context switching**.

The VAX/VMS system also provides each process with a software definition, as described in Section 2.3.

1.2 32-Bit Addresses

The VAX/VMS system uses the processor's 32-bit addresses to provide an extensive virtual address space that is available to user processes. In a virtual memory system, a process need not reside entirely in physical memory to execute. Portions not required at a given time can reside on secondary storage; they are read into physical memory in 512-byte segments, called **pages**, when they become necessary to the stream of execution. As a result, programs with a large virtual address space execute in a much smaller amount of physical memory. The physical memory required for a process to execute a program is called its **working set**.

Not all of the processes known to VAX/VMS have their working sets in physical memory. For example, a process in a wait state may reside temporarily on secondary storage. The processes having their working sets in physical memory are called the **balance set**.

1.3 User, Supervisor, Executive, and Kernel Modes

The VAX-11 processor provides four processor **access modes**: user, supervisor, executive, and kernel. User mode is the least privileged; kernel mode is the most privileged. The VAX-11 processor uses access modes to determine:

- Instruction execution privilege; that is, which instructions the processor will execute
- Memory access privilege; that is, which locations of memory the current instruction can access

The VAX/VMS system also uses access modes to protect itself from user processes executing in user mode and to separate parts of the operating system according to their degree of privilege. For example, the **command language interpreter** runs in supervisor mode (less privileged) while the core of the operating system runs in kernel mode (more privileged).

Because each process has a **stack** and **Stack Pointer (SP)** for each access mode, VAX/VMS can separate various levels of execution within a process. For example, VAX/VMS provides a number of procedures, called system services, that execute in more privileged access modes. A user process can call these services, which, in turn, execute on behalf of the process in more privileged access modes. They execute in the context of the calling process using the process's stack for the associated mode. By providing these system services, VAX/VMS allows user processes to perform sensitive functions under the control of the operating system.

A special stack referred to as the **interrupt stack** is used to handle interrupt requests. When an interrupt occurs, the processor operates in a system-wide context in kernel mode.

1.4 Native and Compatibility Mode Instruction Sets

The VAX-11 processor contains two instruction sets: a native mode instruction set and a compatibility mode instruction set. The native mode instruction set provides an extensive number of data types, operation codes, and addressing modes tailored to the 32-bit environment.

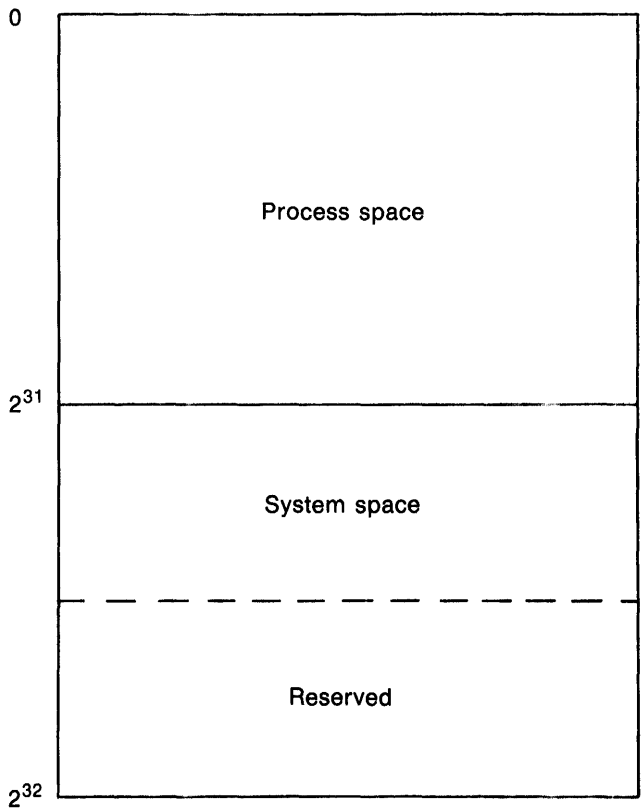
The compatibility mode instruction set allows execution of user mode PDP-11 programs that meet specific requirements. In particular, VAX/VMS emulates the RSX-11M Version 3.2 programming environment so that RSX-11M Version 3.2 user mode tasks can execute under VAX/VMS.

Native and compatibility mode instructions cannot be mixed within the same program.

Chapter 2
Processes

The VAX/VMS operating system provides a virtual address space of 2^{32} bytes of memory apportioned for user programs and system use as shown in Figure 2–1.

Figure 2–1: VAX/VMS Virtual Memory



The lower-addressed half of virtual memory, called **process space**, is where user programs execute. The higher-addressed half, called **system space**, is divided into two parts. The lower part of system space is where system procedures such as VAX/VMS system services and memory execute. The upper part of system space is reserved.

Users need only be concerned with the process space and their own programs.

2.1 Process

A process is the basic schedulable entity executed by the VAX–11 processor. VAX/VMS automatically creates a process for any of the following reasons:

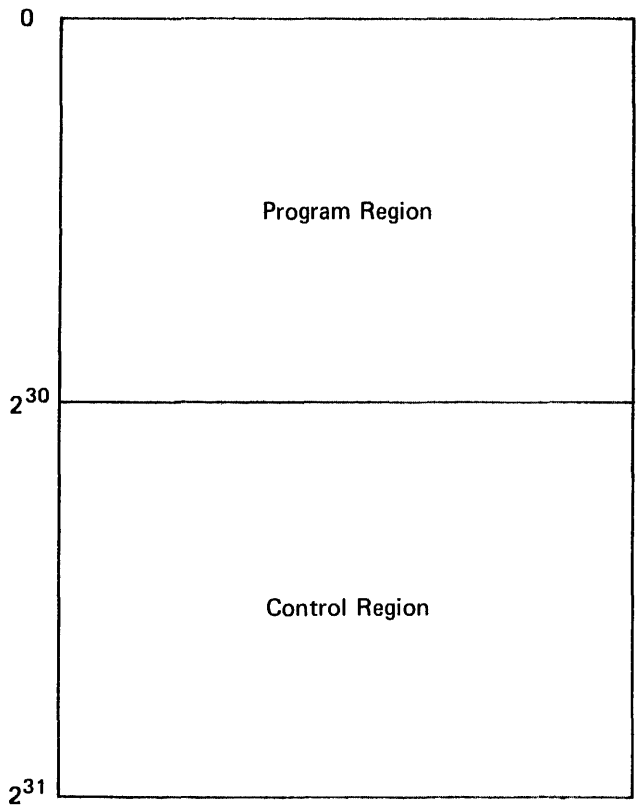
- A user logs into the system at an interactive terminal
- The system starts execution of a batch job
- DECnet creates a process on behalf of a remote user

In all cases, the process created provides a controlled environment in which the system performs the user's requests. Each process provides a virtual address space where programs are run and contains the information required by the operating system to schedule the process for execution and to control its interaction with other processes in the system.

2.1.1 Process Virtual Address Space

The process space of VAX/VMS virtual memory starts at virtual address 0 and extends to virtual address 2^{31}. It consists of two regions, as shown in Figure 2–2.

Figure 2–2: Virtual Memory of a Process



The **program region** is the lower-addressed half of the process space. It provides the virtual memory in which programs run. The program region starts at virtual address 0 and extends toward the higher addresses. The address that separates the program region from the control region is fixed at 2^{30}.

The program region is also called the P0 region.

The **control region** is the higher-addressed half of the process space (see Figure 2-2). This region normally contains the user stack; the supervisor, executive, and kernel stacks; system components; and system control information. The system-reserved portion of the control region is only a small portion of that region; most of the control region (for example, the user stack) remains accessible to user programs. The control region starts at virtual address 2^{31} and extends downward toward the lower addresses.

The control region is also called the P1 region.

2.2 Image

An **image** is a program that is executed in a process; that is, it is the output of the VAX-11 Linker.[1] An image consists of 1) programmed procedures (routines) that are called during image execution, and 2) data areas. To request execution of an image, a user specifies the file name of the image. The operating system then activates the image in the user's process and calls the image at its main procedure.

Images execute serially in the program region of a process; after one image terminates, the next can be requested. Section 6.3 describes the allocation of virtual memory to an image.

2.3 Sources of Process Characteristics

The system provides each process with a set of distinguishing characteristics. These characteristics consist of information maintained by three sources:

- The user authorization file
- The VAX/VMS operating system
- The VAX-11 processor, described in Section 1.1

2.3.1 User Authorization File

The system manager maintains a **user authorization file** that contains an entry for each user allowed access to the system. A user authorization file entry describes the characteristics of a process that are associated with the user for whom the process is created. Additional information in a user authorization file entry includes:

- User name, password, and account name
- Privileged system functions to which the user is allowed access
- Limits and quotas for the system resources that can be used
- Priority of the process created for the user at login
- Default command interpreter
- Default disk device and directory

1. Images produced by the RSX-11M Version 3.2 task builder also can be executed under VAX/VMS; refer to Chapter 8.

When VAX/VMS creates a process, it reads the user authorization file entry for that user to obtain user-related characteristics of the process.

2.3.2 VAX/VMS Definition of a Process

The VAX/VMS operating system uses three data structures to define a process: the **software process control block (software PCB)**, the **process header**, and the job information block (JIB). The software PCB is the central control mechanism for the process. It includes the current state of the process, the address of the process if it is on secondary storage rather than in memory, a unique identification number, and the process's priority, taken from the user authorization file.

The process header for each process contains the privilege mask for the process, taken from the user authorization file; and memory management information. It also contains the process's hardware PCB.

The job information block for each process contains the accounting and quota statistics for the process. The process and the process's subprocesses share the job information block.

2.4 Process Priorities

The VAX/VMS system defines 32 levels of software priorities for the purpose of scheduling. Priorities range from 0 through 31, with 31 representing the highest priority. Priorities 0 through 15 are used for normal processes; priorities 16 through 31 are reserved for **real-time processes**.

The VAX/VMS system schedules processes having real-time priorities strictly according to their priority. That is, the executable process with the highest real-time priority has access to the VAX-11 processor.

For processes with normal priorities, VAX/VMS uses a scheduling algorithm that slightly modifies priorities to achieve maximum overlap of compute-bound and I/O-bound operations. It achieves the overlap using a **base priority** and a current priority. VAX/VMS obtains the base priority from the user authorization file entry when it creates a process. It uses the base priority and an increment to calculate the current priority. When a process becomes eligible for execution, VAX/VMS adds an increment to the base priority to form the current priority. Each time the process is subsequently scheduled for execution, VAX/VMS decreases the current priority until it reaches the base priority, or until the process once again enters a **wait** state.

The result is that a process having a normal priority and using small amounts of processor time between waits (for example, one that issues many I/O requests) tends to have a higher current priority than a process having the same base priority but performing large amounts of processing between waits.

The VAX/VMS system never raises the current priority to the value of a real-time priority.

2.5 Scheduling

The VAX/VMS system schedules processes using a method of process priorities, process **state queues**, and system **events**. Each process in the system has its software PCB linked into a state queue.

The two main sets of state queues are for executable processes: those that are in memory (that is, in the balance set) and those that are on secondary storage (that is, out of the balance set). Each set consists of 32 queues, one for each priority level. This multiple queue arrangement eliminates the need for queue searching when context switching. For example, the next process to execute is always at the head of the highest priority queue for which an executable process exists.

The remaining state queues are for processes that are waiting for a condition to be satisfied before they can become eligible for execution. These queues contain all processes in the appropriate state regardless of priority.

Processes make the transition from one state queue to another as a result of system events. System events are occurrences that favorably or adversely affect the ability of one or more processes to execute. An executing process can cause a system event by putting itself in a wait state, or it can cause a system event for another process. All system events are reported to the scheduler.

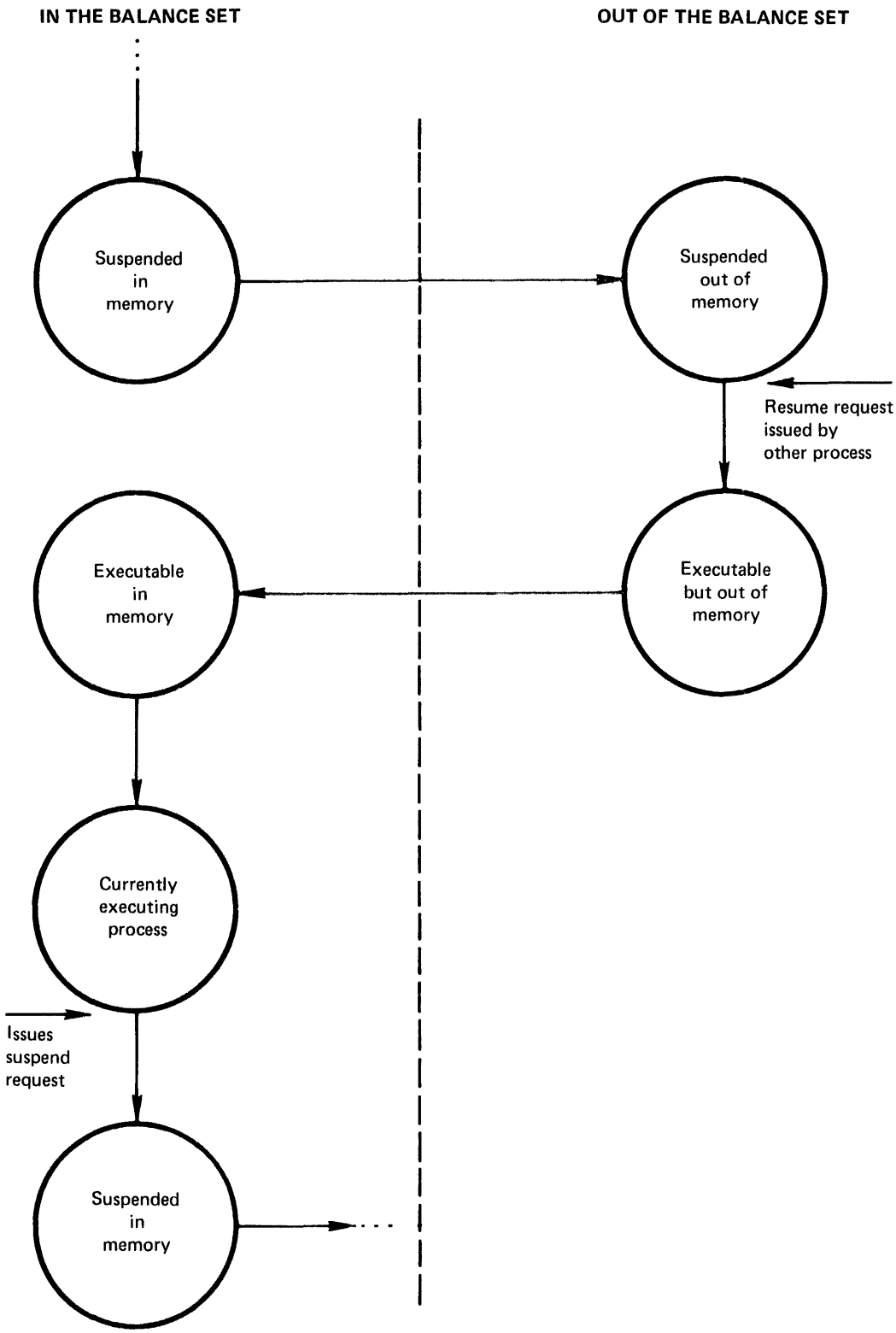
Appendix A lists process states and system events.

Figure 2–3 illustrates a process making the transition from one state queue to another.

2.6 Process Privilege, Limits, and Quotas

The VAX/VMS operating system uses the **process privileges** contained in the user authorization file to determine the protected functions to which a process has access. The operating system also uses the **quotas** and **limits** contained in the user authorization file to restrict a process's use of system resources, for example, physical memory. Subsequent chapters of this summary note process functions restricted by privileges and limits or quotas.

Figure 2–3: Example of Process States and Scheduling



Chapter 3
Memory Management

In a virtual memory system, the composite size of all processes' virtual address space and system space exceeds the actual amount of physical memory available. **Memory management** determines how the limited physical memory is allocated.

Under VAX/VMS, memory management mediates requests for physical memory on two levels:

- Within a process
- Among all processes in the system

To mediate requests for memory within a process, VAX/VMS uses the concept of a working set. To mediate requests among all processes, VAX/VMS uses the concept of a balance set.

3.1 Working Set

When a process executes, only a subset of its pages need be in physical memory. These pages are referred to as the process's working set. The remaining pages reside on secondary storage. When VAX/VMS creates a process, it uses the requesting user's authorization file entry to determine the size of the working set for that process. The size of the working set determines the number of pages of physical memory needed to run the process.

The system reads pages residing on secondary storage into physical memory when a hardware **page fault** occurs for the page. A hardware page fault occurs when a process attempts to refer to a virtual page that is not in physical memory. Once in memory, the code or data contained in that page can be executed or accessed. A page brought into memory because of a hardware page fault remains in memory as part of the working set until it is no longer needed and its space is required for a new page.

The system allows a process to control the residency of its pages in physical memory. For example, a process can state how many of its pages are to be kept in memory, lock individual pages in its working set, and, with the suitable privilege, lock individual pages in physical memory. The user authorization file entry places a limit on the number of pages that a process can have in its working set.

In addition, users can specify a **page fault cluster size** when linking an image. The page fault cluster size determines the number of contiguous virtual pages of the image that memory management attempts to bring into the process's working set when a hardware page fault occurs. Bringing a cluster of pages into physical memory in one read operation reduces the number of I/O operations required, especially when an image first begins to execute. Once

the process's working set is full, memory management brings pages into physical memory one at a time; that is, when a fault occurs for the page. Then each page brought into the working set causes another page to be removed from it.

3.2 Balance Set

The balance set consists of the set of processes that reside in physical memory. These processes have memory requirements that balance with the available physical memory of the system. At certain points during the execution of a process, its entire working set can be written to secondary storage, thereby freeing physical memory for another process. For example, a suspended process's working set can be written to secondary storage while the process is waiting to be resumed. After the resume request is issued, VAX/VMS brings the process back into memory. This method of controlling memory use by removing some processes from and adding other processes to the balance set is called **swapping**.

Swapping and scheduling work with each other to provide maximum throughput. Section 3.4 describes swapping.

3.3 Mapping Process Virtual Pages to Physical Memory Pages

The VAX/VMS operating system controls the mapping of a process's virtual memory to physical memory on a page-by-page basis. As a hardware page fault occurs for a specific page, that page is brought into physical memory and into the process's working set. Although a process's virtual memory is contiguous, the pages of physical memory in which it executes are not. VAX/VMS uses structures called page tables to maintain the correlation between a process's virtual pages and the physical pages of memory that the process uses.

Each process has two page tables: one for the program region and one for the control region. The P0 page table contains an entry for each virtual page in the program region, and the P1 page table contains an entry for every page of the control region, as shown in Figure 3–1. Each **page table entry** describes the location of its associated page, as shown in Figure 3–2.

Any page of a process can be in either of the following states:

- In physical memory in the working set or on the free or modified page list
- On secondary storage in an image file or paging file

3.3.1 Free Page List, Modified Page List, and Paging File

When bringing one page into a process's working set causes another page to be removed from the working set, VAX/VMS chains the removed page into either of two lists:

- Free page list
- Modified page list

Figure 3-1: Process Pages and Page Table Entries

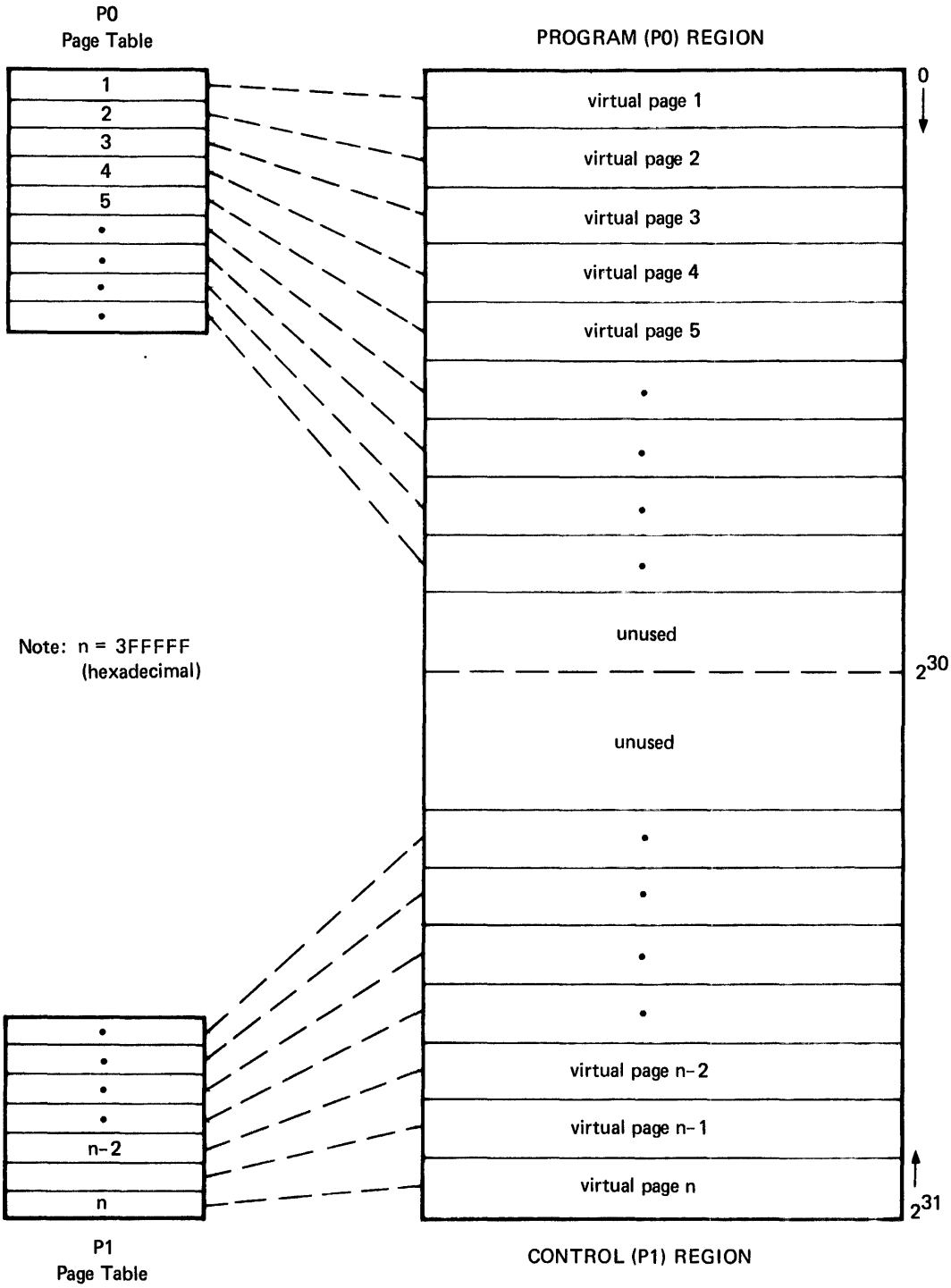
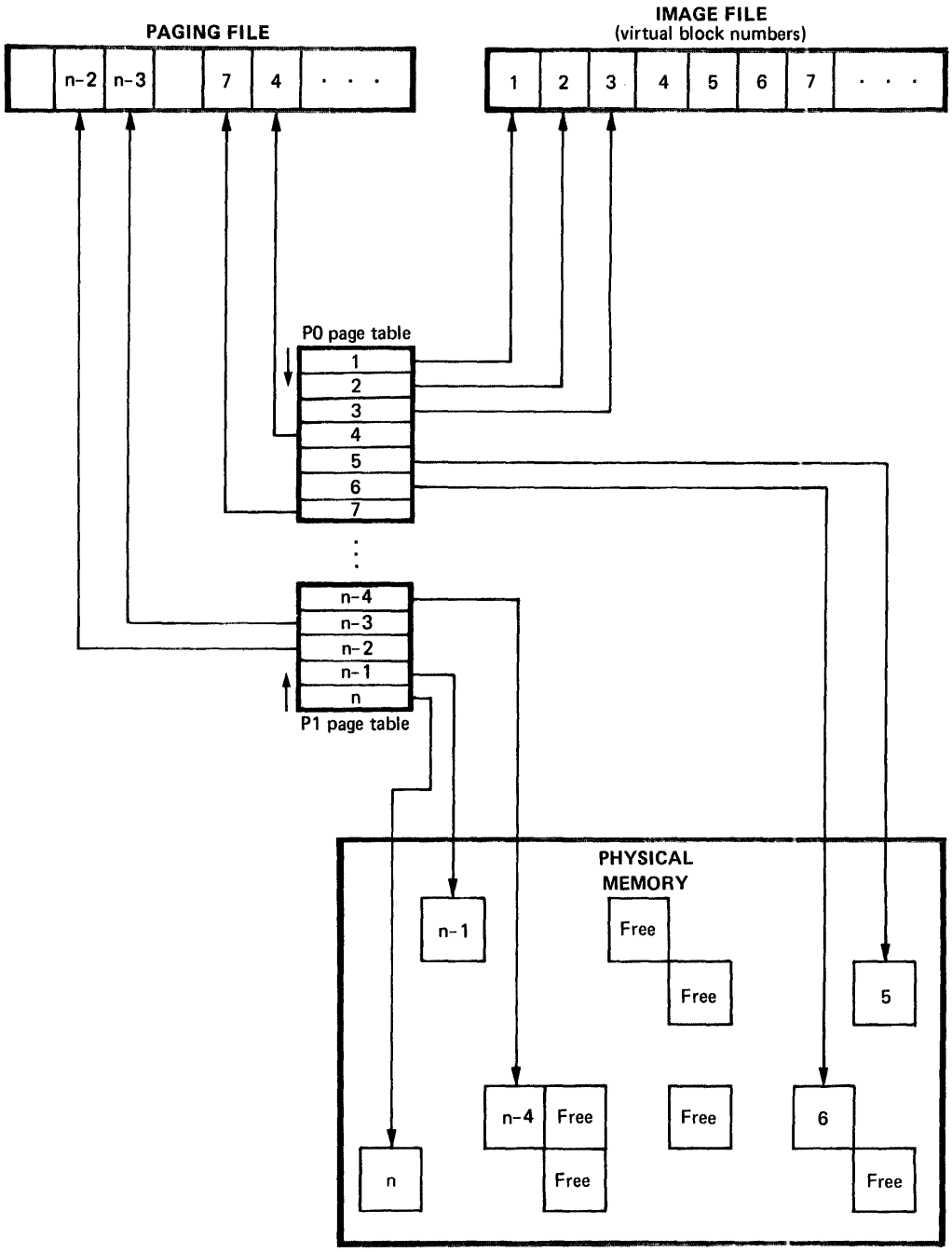


Figure 3-2: Use of Page Table Entries



The free page list contains pointers to pages that do not have to be written back to secondary storage; that is, pages that were not modified while in memory. The free page list serves two purposes:

- It is the list of available pages of physical memory
- It serves as a cache for recently removed working set pages

Pointers to pages are placed on the bottom of the list and are removed from the top of the list. Because a virtual page remains in memory for a period of time, a page fault for a virtual page on the list can be satisfied by taking the page from the free list and placing it back in the working set. It is not necessary to read the page back from disk.

The modified page list contains pointers to pages that must be written back to disk. Some modified pages (for example, pages of a **global section** containing read/write data) are written back to a data file; for further information, see Section 6.4. Other pages (for example, pages that correspond to image pages that must not be modified in the image file or that have no image file address) are written to a system-defined file on secondary storage, called a paging file. Like the free page list, the modified page list also serves as a cache.

Once virtual pages on the modified page list have been written to secondary storage (either the image file or a paging file), pointers to the physical pages that contained them are added to the free list. The user authorization file places a limit on the number of pages a process can have in a paging file.

3.4 Swapping In and Out of the Balance Set

The operating system mediates among executable processes in and out of the balance set by performing working set swapping. Swapping is motivated by the need to bring into memory a process that would be executable if its working set were in physical memory. VAX/VMS obtains sufficient memory to swap the highest-priority executable process into memory.

The system obtains that memory by collecting available pages (for example, pages from the free list) and by swapping nonexecutable processes (that is, processes in a wait state) out of memory. If necessary, the system also swaps lower-priority executable processes out of memory to make room for higher-priority processes. Once a process is swapped into memory, it remains resident until it has had access to a reasonable amount of processor time. A process with the appropriate privilege can lock itself in the balance set.

Individual installations determine the number of processes allowed in their balance sets, as described in the *VAX/VMS System Manager's Guide*. That number is affected by considerations such as the amount of physical memory available and typical working set sizes.

Chapter 4
Process Control and Interprocess Communication

Process control occurs within a process and among cooperating processes in the system. A process provides each image it executes with mechanisms that allow the image to coordinate its activities and respond to error conditions. VAX/VMS also provides the mechanisms that allow cooperating processes to coordinate their activities, control each other, and communicate with each other.

4.1 Intraprocess Control

An image can coordinate its flow of execution through the use of local **event flags**, **asynchronous system traps**, and **condition handlers**. In addition, an image can cause the process to hibernate and request certain time-related services for the process. The *VAX/VMS System Services Reference Manual* describes these features.

4.1.1 Local Event Flags

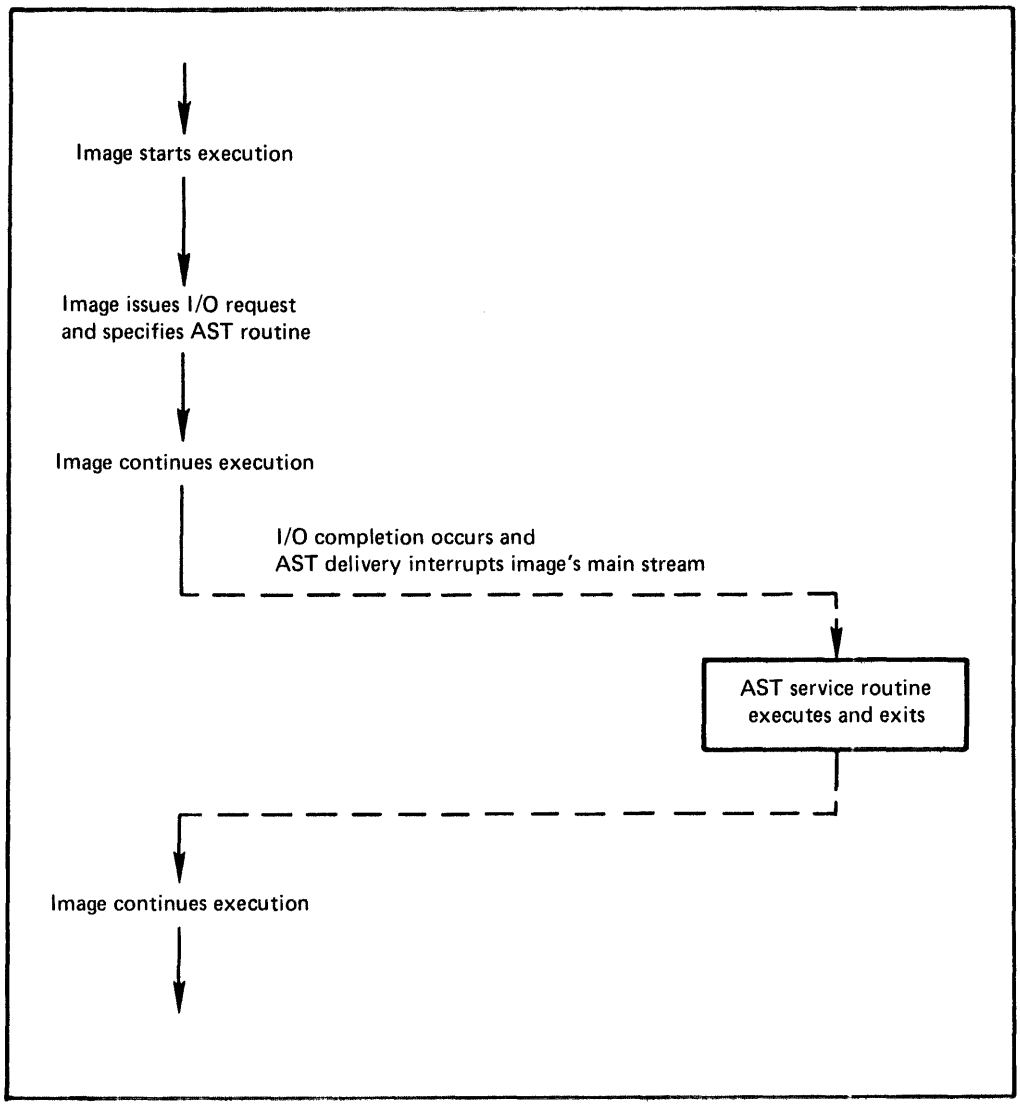
A local event flag is a bit that can be set or cleared to indicate the completion of a function. Each process has two clusters of local event flags; each cluster contains 32 flags. Various procedures within an image use event flags to communicate event completion and to coordinate their activities. Images can set, clear, and read flags, or they can request that process execution be suspended until a designated flag is set.

I/O completion is a prime example of local event flag use. An image can request the system to set a designated event flag when an I/O transfer is completed. The image can either wait for the flag to be set before process execution continues, or the image can continue and test the flag periodically to determine when I/O completion has occurred.

4.1.2 Asynchronous System Traps

Asynchronous system traps (ASTs) allow an image to request that a specific routine be executed in response to an event, regardless of when the event occurs in relation to image execution. When the event occurs, image execution is interrupted at its current point, the AST service routine is executed, and then the image resumes at the point of interruption. An image can request ASTs for I/O completion, power failure recovery, and after a specified time interval or at a specified time of day. Figure 4–1 is an example of an I/O completion AST.

Figure 4–1: Example of I/O Completion AST



The system can queue ASTs for a process at any of the four access modes. An image can enable and disable AST delivery at its current access mode and less privileged access modes; normally, images deal only with user mode ASTs. The queue of pending ASTs is ordered by access mode. ASTs for more privileged access modes take precedence over ASTs for less privileged access modes.

4.1.3 Condition Handlers

Condition handlers are procedures that are given control when an **exception** occurs. Exceptions are hardware- or software-detected conditions (usually error conditions) that interrupt the execution of an image. Because an image can declare condition handlers for possible exception conditions, the image has the opportunity to react to a wide variety of errors. If an image incurring

the exception has declared a condition handler, the condition handler can either perform error recovery and allow the image to continue, or perform any appropriate clean-up operations and cause the image to exit.

Exception conditions include errors from which the processor cannot normally recover, such as a divide-by-zero error, and special conditions for which an image does not wish to test continually, such as floating point overflow.

An image can declare more than one condition handler. When a condition handler starts execution, it can examine the condition name and either handle the error or resignal the condition for another condition handler. Condition handlers can be declared for the duration of an image, or for the duration of a procedure within the image.

4.1.4 Hibernation

An image can request that its process be placed in a state of **hibernation**, that is, a state of inactivity. Even when a process becomes inactive, it is still known to the system. Consequently, it can be removed from inactivity by either of the following:

- Delivery of an AST for which the image has declared an AST service routine; once the AST has been handled, the process returns to hibernation
- A **wake** request scheduled by the image, or a wake request issued from another process, as described in Section 4.2.3, or within an AST service routine

4.1.5 Timer System Services

Timer system services allow an image to request an action for a future time. The future time can be designated either as a specific time or as an interval measured from the current time. Timer services allow an image to schedule the following:

- The setting of an event flag
- The queuing of an AST for its process
- The issuing of a wake request for its process or another process

4.2 Interprocess Control and Communication

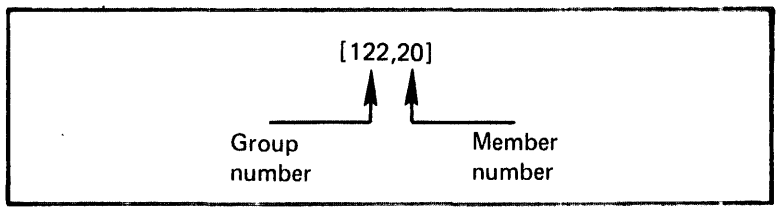
A process can create processes and interact with other processes in the system. Processes interact through the use of process control system services, common event flags, **mailboxes**, and global sections. VAX/VMS uses each process's **user identification code (UIC)** to determine the processes it can affect and the common event flags and mailboxes to which it has access.

Processes can interact also by use of DECnet logical links, as described in the *DECnet-VAX User's Guide*.

4.2.1 UIC-Based Protection and Process Control Privileges

Every user of the system has a unique user identification code that consists of two 16-bit numbers. The first number represents the user's group, and the second number represents the user's member number within the group. Users performing related tasks have their UIC group numbers in common; they have unique member numbers to allow differentiation among users. Figure 4–2 illustrates a UIC. Group and member are expressed as octal numbers, separated by a comma, and are enclosed in square brackets.

Figure 4–2: Example of a User Identification Code



The system manager for an installation assigns a UIC to each user.

The operating system provides system services that allow one process to affect the execution of another; for example, one process can suspend another. VAX/VMS uses the group number of the process's UIC with process privilege to control processes' effect on each other. Two privileges are used in this respect:

- Group process control privilege (GROUP) — The right of a process to affect any other process having the same group number in its UIC
- World process control privilege (WORLD) — The right of a process to affect any other process in the system, regardless of UIC

Unless a process has one of the above privileges, it can affect only its subprocesses. Group privilege allows a set of processes to execute related images and interact with one another without being able to interfere with other processes in the system.

UIC-based protection is also applied to interprocess communication mechanisms such as mailboxes, as described in Section 4.2.5, and global sections, as described in Section 4.2.6.

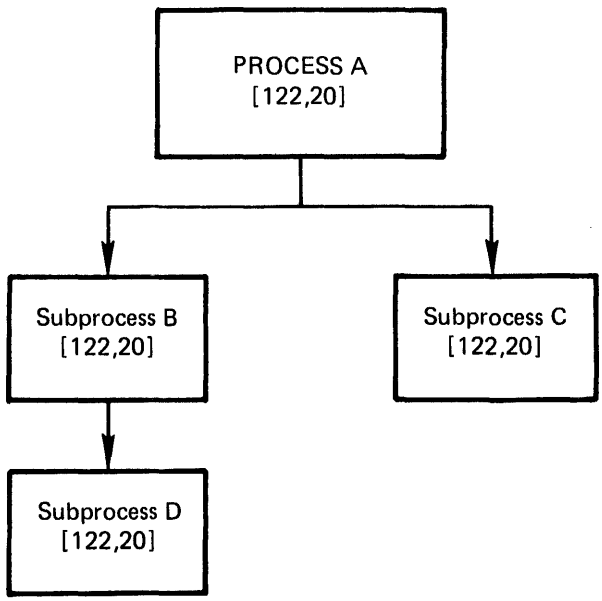
4.2.2 Subprocesses and Detached Processes

A process can create **subprocesses** and, if the process has the appropriate privilege, it can create independent processes called **detached processes**.

When a process creates a subprocess, the process and its subprocess share a pool of resources and quotas. No process privilege is required to create a subprocess. VAX/VMS limits the number of subprocesses that a process can create because it requires the process to share its resources with its subprocesses. It also limits the number of subprocesses that a process can have at any one time. The system manager indicates in the user authorization file entry the resource limits for a user's process.

A process and its subprocesses form a job, that is, an accounting unit, as shown in Figure 4–3.

Figure 4–3: Process and Its Subprocess



All of the resources used by a job are charged to the main process's account name.

When a process creates a detached process, it specifies the resources and privileges allowed the detached process. Because the resources granted are not subtracted from the creating process's resources, VAX/VMS requires a process to have a privilege to create a detached process. In addition, a process must have the appropriate privilege to grant a process any privilege that it does not have itself.

A detached process can have a UIC different from that of its creator.

4.2.3 Process Control System Services

Process control system services allow one process to affect another process's ability to execute. Using process control system services, a process can perform the following functions:

- Create or delete a process
- Suspend a process or cause it to resume
- Place itself in hibernation or wake a hibernating process
- Terminate execution of an image in another process

A process must have either group or world process control privilege to affect other processes unless they are the requesting process's subprocesses.

Suspension and hibernation are both inactive states of a process; however, the following distinctions are significant:

- A hibernating process can be awakened to receive an AST. After the AST service routine exits, the process automatically returns to hibernation. ASTs for a suspended process are queued; they are not delivered until the process is resumed.
- A process can suspend another process. A process can place only itself in hibernation.

4.2.4 Common Event Flags

In addition to its local event flag clusters, each process can associate with up to two **common event flag clusters**, each containing 32 flags. Once a process has created a common event flag cluster, other processes in the same group as the creator can associate with the cluster. Processes can read, set, and clear event flags and they can wait for a designated flag or flags to be set. One process can indicate that a function has been completed by setting a common event flag that is tested by another process or processes.

Event flag clusters can be either temporary or permanent. VAX/VMS automatically deletes a temporary cluster when all associated processes have disassociated from it. Permanent event flag clusters remain in the system until explicitly deleted. A process must have the appropriate privilege to create a permanent event flag cluster.

Common event flag clusters can be in shared memory (that is, **multiport memory**) or in memory that is local to a particular processor.

4.2.5 Mailboxes

A mailbox is a record-oriented virtual I/O device that processes can use to exchange status information and messages. Typically, a process creates a mailbox from which it reads data that was placed in the mailbox by other cooperating processes. Messages can be read from and written to mailboxes using standard VAX/VMS I/O facilities.

The VAX/VMS system uses UIC-based protection to control access to the mailbox. The mailbox creator (**owner**) can allow read and/or write access or deny access to several categories of users including those in the same group as the owner. The protection applied to mailboxes is very similar to that applied to data files, as described in Section 5.5.

A process also can provide the name of a mailbox when it creates a process or subprocess. When the created process is deleted, its creator receives notification in the mailbox. The mailbox message indicates the exit status of the image that the created process executed.

Like common event flags, mailboxes can be either temporary or permanent. Permanent mailboxes must be explicitly deleted. A process must have the appropriate privilege to create either a temporary or permanent mailbox.

Mailboxes can be in shared (that is, multiport) memory or in memory that is local to a particular processor.

4.2.6 Global Section

A global section can be a disk file section or a page frame section. A disk file section is data or code from a disk file that can be brought into memory and made available, either only to the process that creates it (private section) or to all processes that map to it (global section). A page frame section consists of one or more page frames in physical memory or I/O space.

Global sections can be in shared (that is, multiport) memory or in memory that is local to a particular processor.

Section 6.4 provides additional information about global sections.

Chapter 5
VAX/VMS I/O System

The VAX/VMS I/O system consists of:

- System services that an image can call to request I/O transfers and other related functions
- **Ancillary control processes (ACPs)** that perform file- and directory-related functions, magnetic tape handling functions, and networks functions
- I/O drivers that perform device-level operations

User images can either interface directly with the I/O system by means of system services, or they can use VAX-11 **Record Management Services** (VAX-11 RMS) to interface with the I/O system on their behalf. Figure 5-1 illustrates the interaction among an image and components of the I/O system.

5.1 Overview of I/O

Before an image can request an I/O operation, it must establish a path of reference to the device on which the operation is to be performed. Under VAX/VMS, an image creates a path of reference by calling a system service that returns a channel number (path designator) for the assigned device. The image can then request I/O operations by specifying the **channel** number assigned to that particular device.

Once the image has issued an I/O request, VAX/VMS allocates in system space an I/O request packet that describes the operation to be performed. Using information in the I/O packet, the driver validates the request. VAX/VMS places the packet in the appropriate device unit's request queue according to the priority of the requesting process and returns the request status to the image. That unit's I/O driver is activated if the unit is not currently busy.

The driver initiates the actual hardware operation on the device. The driver then returns control to the system.

When the actual transfer completes, an interrupt causes the driver to be reentered so that it can complete the processing of the I/O request. The driver determines the final I/O request status, which is returned to the image initiating the request. At that time, any event flags that the image requested are set, and a user-specified I/O completion AST is queued, if requested.

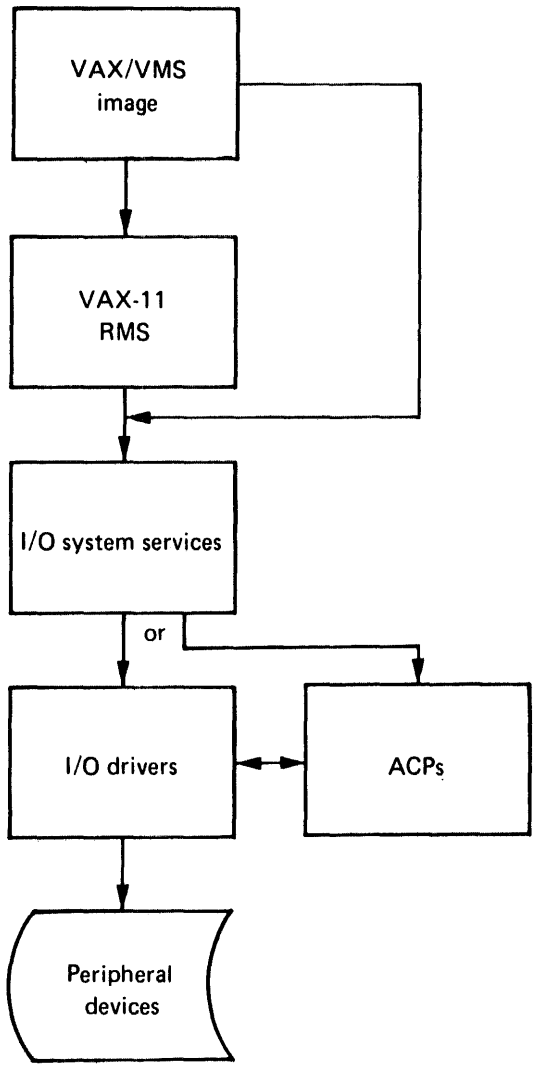
If an image issues an I/O request that the driver cannot perform because it requires understanding of file-structured volumes, magnetic tape, or networks, ancillary control process (ACP) intervention is required. Images can make an explicit request for an ACP function, for example, to create a file

entry in a directory, or they can make an implicit request, for example, when an I/O transfer causes the file **window** mapping pointers to be updated. VAX/VMS provides five ACPs:

- Two **Files-11** ACPs (F11ACP) for disk volumes
- Magnetic tape ACP (MTAACP)
- Networks ACP (NETACP)
- Remote I/O ACP (REMACP)

Once the ACP provides the needed service, it requeues the I/O request for the driver, which processes it in the normal manner. The *VAX/VMS I/O User's Guide* provides additional information about drivers and ACPs.

Figure 5-1: Components of VAX/VMS I/O System



5.2 I/O System Services

An image can call VAX/VMS I/O system services to request the following functions, which are described in detail in the *VAX/VMS System Services Reference Manual*:

- Allocate a device for exclusive use by the image and later deallocate it
- Assign a channel to a device and later deassign it
- Queue an I/O request and, optionally, wait for its completion
- Create a mailbox and later delete it; Section 4.2.5 describes mailboxes
- Get information about a device
- Cancel I/O on a channel

The Queue I/O Request system service allows an image to issue I/O requests directly, rather than going through VAX-11 RMS. Each request contains an **I/O function code** indicating the type of operation to be performed. Depending on the function code specified, either a device driver or an ACP handles the request. All function codes are classified as either device independent (for example, read a block of data) or device dependent (for example, rewind a magnetic tape). Images written with device-independent function codes can perform I/O using different device types (for example, disk, tape, or a network logical link) without reprogramming.

Function codes allow images access to devices at the virtual, logical, and physical levels. **Virtual I/O function** codes request file-oriented I/O operations; they require ACP intervention. The ACP performs privilege and protection checking for the I/O operation. **Logical I/O function** codes allow images with the appropriate privilege direct access by logical block number, rather than file-relative access, to a mass storage volume. **Physical I/O function** codes allow the image access to all device hardware functions. Normally, only diagnostic programs perform physical I/O; the appropriate privilege is required.

5.3 VAX-11 RMS

VAX-11 RMS allows images to perform device-independent I/O without having to be concerned with physical devices and **file structures**. Images issue commands to open a file, get and put records or read and write blocks, and close the file. VAX-11 RMS, in turn, assigns the channels, issues I/O system services, and considers physical device requirements and volume and file structures. These requests from VAX-11 RMS cause the driver or ACP to perform the image's requests.

Records read or written using VAX-11 RMS can be fixed or variable length or variable length with a fixed-length control field.

A programmer can select the appropriate file organization and record access mode from among the following supported by VAX-11 RMS on Files-11 volumes:

- **Sequential file organization** accessed sequentially or randomly by a record's file address
- **Relative file organization** accessed sequentially, or randomly by a record's file address, or randomly by relative record number
- **Indexed file organization** accessed sequentially, randomly by a record's file address, or randomly by key

In addition, during the processing of files, users can change from one access mode to another dynamically. For additional information about VAX-11 RMS file organizations and record access modes, refer to the *Introduction to VAX-11 Record Management Services.*

5.4 Files-11 Disk Structure

VAX-11 RMS interacts with the Files-11 ACP to create Files-11 disk volumes. Files-11 is the name of the directoried disk volume structure supported by VAX/VMS. Using VAX-11 RMS, images can read and write both **Files-11 Structure Level 1** and Structure Level 2 volumes.

VAX-11 RMS supports structure level 1 to provide compatibility with other DIGITAL operating systems, that is, for compatibility with RSX-11M. As a result, Files-11 disk volumes can be transported between a VAX/VMS system and these systems.

Files-11 Structure Level 2 is an advanced version of structure level 1. It provides an improved directory structure, named directories and subdirectories, more encompassing user-specified volume protection, and improved performance for large files.

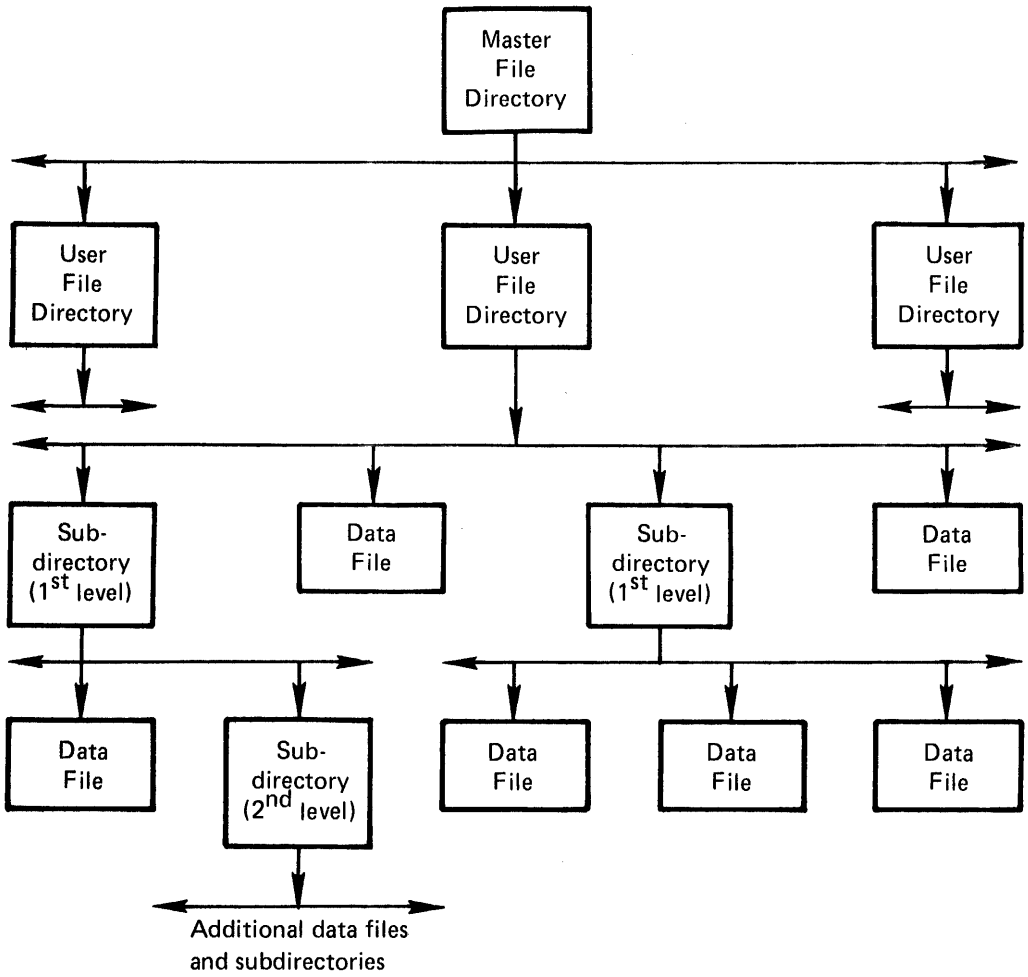
Each Files-11 volume contains a master file directory that locates the top level user file directories on the volume. Each user file directory, in turn, locates the files in that directory and, in the case of structure level 2, the first level subdirectories, as illustrated in Figure 5-2. As many as seven nested subdirectory levels are allowed.

5.5 File Protection

The VAX/VMS system of data protection allows users to specify the protection to be applied to entire volumes, to individual directories and subdirectories, and to data files. Users are categorized, according to their UIC group and member numbers, as:

- **System** — Users of the system whose group numbers are in the range 0 through 10 (octal), for example, [1,1], or who have certain I/O-related privileges
- **Owner** — The user whose UIC group and member numbers are identical to the UIC of the file or volume creator, for example, [122,20]

Figure 5-2: Files-11 Directories and Subdirectories



- **Group** — Users of the system whose UIC group number is identical to the file or volume owner's group number, for example, [122,30]
- **World** — All users not included in the categories above, for example, [115,10]

The owner of a volume or file can permit four types of access for each user category:

- Read (R) access — The right to read the file for any purpose, for example, to copy, print, or display it
- Write (W) access — The right to update or extend the file
- Execute (E) access — The right to execute an image file or, in the case of disk volume protection, the right to create directories on the volume
- Delete (D) access — The right to delete the file

Each user has a default protection that the system applies to each newly-created file unless the user specifically requests a modified protection. A

typical default protection allows read, write, execute, and delete access for the system and owner categories and read, write, and execute access to the group category; it denies all except read access to the world.

File protection also applies to mailboxes and global sections.

5.6 Logical Names

Regardless of whether an image interfaces with the I/O system using system services or VAX–11 RMS, it has access to the VAX/VMS **logical name** facility. A logical name is a character string used to refer to a file or device by other than its specific physical name. A logical name is defined by equating it to a physical device name or all or a portion of a file specification. The *VAX/VMS Command Language User's Guide* provides additional information about file specifications and logical names. The following are examples of logical names and their **equivalence names**:

Logical Name	Equivalence Name	Notes
TAPE	MTA1:	The logical name TAPE translates to the physical name for controller A and unit 1 of a TU16 Magnetic Tape Drive.
INPDAT	DBB2:[JONES]	The logical name INPDAT translates to the physical device and unit DBB2 and the directory [JONES].
INFILE	DBB2:[MYDIR]PAYROL.DAT;3	The logical name INFILE translates to the full file specification that provides the device name and unit number (DBB2); the directory name ([MYDIR]); and the file name, file type,and file version number (PAYROL.DAT;3).

Logical names permit images to be independent of the physical devices or files that they use for I/O. Before running an image, a user defines the logical names and their equivalences. Images in execution refer to devices by the assigned logical names. As part of I/O processing, VAX–11 RMS and the system services translate the logical names to equivalence names.

For example, an image can assign a channel to a logical device name. The system service performing the actual channel assignment translates the logical name and assigns the channel to the equivalent physical device. Thus, all I/O operations specifying that channel occur on the associated physical device. Each time the image executes, the logical name can be associated with a different physical device, depending on device availability.

The VAX/VMS operating system defines three levels of logical names, each of which is contained in its own logical name table:

- Process logical names — names defined for use only by the images that the process executes

- Group logical names — names that can be shared among users having the same group number in their UICs
- System logical names — names that can be shared by all users of the system

When the system translates a logical name, it searches the process, group, and system logical name tables in that order. As a result, process logical names take precedence over group and system logical names, and group logical names take precedence over system logical names.

5.7 Process-Permanent Files

So far, this chapter has discussed only I/O operations requested by the images that a process executes. However, VAX/VMS also establishes I/O paths that exist for the duration of the process. These paths are called process-permanent files. When VAX/VMS creates a process to handle a user's job, it opens the process-permanent files needed for user command input to the system and for the system to communicate informational and error messages to the user. These process-permanent files include SYS$INPUT and SYS$COMMAND for user command input, SYS$OUTPUT for informational messages from the system, and SYS$ERROR for error messages from the system.

For an interactive user, these four process-permanent files normally are assigned to the terminal; however, they can be assigned to other devices. For example, in a batch job, SYS$INPUT normally is a disk file of user commands.

Using VAX-11 RMS, images executing in the process can read from and write to process-permanent files. For example, an application program can accept user queries from SYS$INPUT and display responses on SYS$OUTPUT. If an application assigns SYS$INPUT to another device (for example, a disk file from which it wishes to read data), VAX/VMS communicates with the user by means of SYS$COMMAND. SYS$COMMAND is a process-permanent file that always remains assigned to the initiating user's terminal.

5.8 Networks

DECnet is the DIGITAL networking facility which consists of the protocols and utility programs needed for various DIGITAL operating systems to be connected in a **network**. DECnet-VAX allows a suitably configured VAX/VMS system to participate in point-to-point communication as a Phase II DECnet **node**. DECnet-VAX provides task-to-task communication and network resource-sharing capabilities using the DIGITAL Network Architecture (DNA) protocols.

Using DECnet, a VAX/VMS process can communicate over the network to perform the following functions:

- Task-to-task communication with a user program executing at a remote node

- I/O operations on a remote file
- Log into a remote node

For additional information about networks, refer to the *DECnet–VAX User's Guide*.

5.8.1 Task-to-Task Communication

Under the VAX/VMS system, programs written in VAX–11 MACRO and high-level languages can perform task-to-task communication. High-level language programs use VAX–11 RMS to connect to a remote task and communicate with it. MACRO programs can use the I/O system services as well as VAX–11 RMS.

Because DECnet–VAX is implemented as an integral part of the VAX/VMS operating system, VAX/VMS processes request remote communications with the same I/O facilities used for local I/O requests. By issuing system services, MACRO programs can use the network without being concerned with the details of network I/O (transparent I/O); or they can issue system service requests that provide additional information about and control over the network (nontransparent I/O). VAX–11 RMS provides transparent use of the network for high-level language programs.

5.8.2 Remote File Operations

Remote file operations can be requested either at the program level or the command level. Programs can issue transparent I/O operations to read and write records and blocks in a file at a remote node using VAX–11 RMS.

Users can move sequential and relative files between Phase II DECnet nodes with the same DCL commands used to move files from device to device locally. File organizations other than sequential can be transferred between compatible nodes. Files containing commands, called command procedures, can be transferred to and from a remote node and submitted for processing as long as commands are acceptable to the node on which they will be executed.

To summarize, DECnet–VAX supports the following DCL commands:

APPEND
ASSIGN
COPY
DEASSIGN
DEFINE
DELETE
DIRECTORY
SUBMIT
TYPE

DECnet is not supported by compatibility mode; that is, users cannot run RSX–DECnet programs.

5.8.3 Remote Command Terminals

To enhance remote communications, VAX/VMS supports **remote command terminals** using DECnet–VAX as an interconnect mechanism. Remote command terminals are terminals that are logically connected to another node. This node is referred to as the host node. The node to which the terminal is physically connected is referred to as the local node.

A logical connect to a node is transparent to the user. Consequently, once a logical connect is established, the terminal appears to be physically connected to the host node. The user may then log in and take adavantage of all the resources the host node offers.

Any node in a DECnet–VAX network is capable of being a host node for a remote command terminal.

Chapter 6
Languages, Libraries, Linking, and Sharing

Before a program can be executed under VAX/VMS, it must go through a series of steps that transform it into an executable image. Typically, these steps involve a combination of the following VAX/VMS components:

- Compiler or assembler
- Librarian or libraries
- Linker

All these components cooperate to produce object modules, libraries, and executable and shareable images that are shareable on disk. With the cooperation of VAX/VMS memory management, code and data can be shared in physical memory as well.

6.1 Programming Languages

VAX/VMS provides a variety of programming languages including:

- VAX-11 MACRO
- VAX-11 FORTRAN
- VAX-11 BASIC
- VAX-11 COBOL-74
- VAX-11 PASCAL
- VAX-11 BLISS-32

VAX-11 MACRO is the assembly language provided by VAX/VMS. VAX-11 FORTRAN is an optional language based on FORTRAN-77 ANSI X3.9-1978; it also provides optional support for programs that conform to the previous FORTRAN standard, X3.9-1966. Both the VAX-11 MACRO assembler and the VAX-11 FORTRAN compiler accept one or more source modules as input and produce, as output, a relocatable object module containing native VAX-11 code.

VAX-11 BASIC and VAX-11 COBOL 74 are both optional language processing systems. VAX-11 BASIC uses a superset of PDP-11 BASIC-PLUS-2 features and incorporates many of the features found in RSTS/E BASIC-PLUS. VAX-11 COBOL 74 is based on the ANSI specification X3.23-1974, the industry-wide standard for COBOL.

VAX-11 PASCAL and VAX-11 BLISS-32 are also optional languages supported by VAX/VMS. VAX-11 PASCAL is an extended implementation of the Pascal language, used for educational and general purpose program-

ming. VAX-11 BLISS-32 is a VAX-specific dialect of BLISS (the DIGITAL system implementation language) designed for transportable system programming.

For additional information about the languages supported by VAX/VMS, refer to the appropriate language documentation, as described in the *VAX-11 Information Directory and Index*.

6.2 Libraries

The VAX/VMS librarian allows the creation and maintenance of four types of disk-resident libraries:

- Object module libraries
- Macro routine libraries
- Help libraries
- General text libraries

Libraries facilitate the use of frequently needed modules at assembly or compilation time and link time. They make a single copy of a module or macro routine available to many users. Libraries conserve time by eliminating the need to open and close multiple files. VAX/VMS uses the standard Files-11 UIC-based protection to control access to libraries.

The librarian and linker cooperate to provide a flexible approach to the selection of modules from an object module library, as described in Section 6.3.2.

The VAX/VMS operating system provides a default system object module library and the Common Run-Time Procedure Library, which includes the following types of object modules, as described in the *VAX-11 Run-Time Library Reference Manual*:

- General utility procedures
- Mathematics procedures
- Resource allocation procedures
- Signaling and condition handling procedures
- Language-independent support procedures
- VAX-11 FORTRAN language-specific procedures

The VAX/VMS librarian is invoked with the LIBRARY command, as described in the *VAX/VMS Command Language User's Guide*.

6.3 VAX-11 Linker

The VAX-11 Linker accepts relocatable object modules produced by the assembler or compiler, or both, places them in the virtual address space, and describes them in a manner understood by VAX/VMS memory management.

The output from the linker is an image file. The linker performs two basic steps in producing an image:

- Allocation of virtual memory to the image
- Resolution of intermodule symbolic references

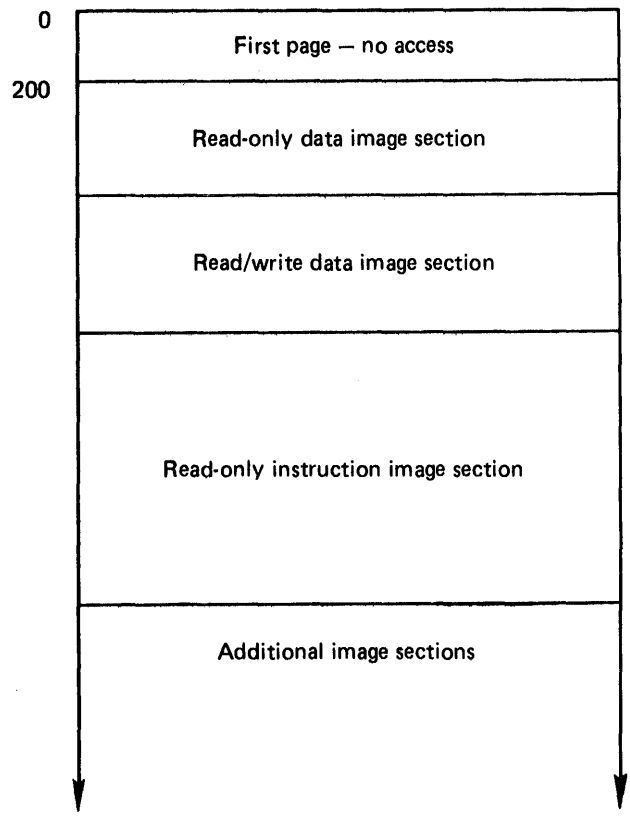
The *VAX-11 Linker Reference Manual* details the use and operation of the linker.

6.3.1 Virtual Memory Allocation

Neither the assembler nor the compiler computes any virtual addresses in a module. Because the assembler or compiler handles each module separately, it cannot determine how modules interrelate in virtual memory. Rather, it describes each program as a number of separate areas called **program sections**. Each program section has a set of attributes. For example, some contain data, others contain instructions; some can be modified, others cannot.

When determining the virtual memory allocation of a program, the linker uses the attributes of each program section: the linker groups program sections that have similar attributes into **image sections**. For example, an image section can contain read-only program sections or read/write program sections. Once the linker has grouped program sections into image sections, it assigns the virtual address space to each image section, as shown in Figure 6–1.

Figure 6–1: Example of Virtual Address Space Allocation



In addition, the linker places sufficient information in the image file for memory management to determine the location and characteristics of each image section. This information is used to activate the image in a process and to perform paging. For example, the linker provides memory management with the page fault cluster size for the image and indicates image sections that are **demand-zero** and **copy-on-reference**. Section 3.1 describes the page fault cluster size.

A demand-zero image section is a writeable image section containing uninitialized data; when a fault occurs for a page of a demand-zero image section, memory management provides the image with a page containing all zeros. A copy-on-reference image section is a writeable image section containing preinitialized data; when a fault occurs for a page of such an image section, memory management provides the image with a private copy of the page. Typically, copy-on-reference image sections are global sections; see Section 6.4. When removed from the working set, both demand-zero and copy-on-reference pages are written back to the paging file, rather than to the image file, as described in Section 3.3.1.

6.3.2 Resolution of Symbolic References

An object module can contain calls to other modules, and it can refer to literals and variables by symbolic name. A **global symbol** is an address or value name that is available for reference by all object modules that are linked in the same operation. The linker resolves global symbols; that is, it matches external references with external definitions.

The linker resolves global symbols using four sources for object modules:

- The modules that the user names in the LINK command
- Modules contained in libraries that the user names in the LINK command
- The default user libraries
- The default system library (including the Common Run-Time Procedure Library)

Modules contained in libraries can be extracted for inclusion in the output image by specifying module names explicitly to the linker, or by providing the linker with the name of a library and letting it extract only those modules that resolve symbolic references. Once the linker has searched the user-specified libraries to resolve references, it searches the default libraries to resolve any that may remain.

The MACRO assembler and the high-level language compilers also allow the use of global symbols in expressions. Because neither the assembler nor the compiler knows the value of externally defined symbols in expressions, the linker must evaluate such expressions and insert their values in the image.

6.3.3 Images

The VAX-11 Linker produces three types of images:

- **Executable images**
- **Shareable images**
- **System images**

An executable image is an image that can be executed by a process, for example, as a result of a user's issuing a RUN command. Executable images are formed by binding together one or more object modules and, optionally, shareable images.

A shareable image is one designed for use in multiple executable images. To produce a shareable image, the linker resolves all internal references. When the shareable image is subsequently bound with other object modules to form an executable image, the linker need only resolve references from the object modules to the shareable image and perform the normal binding for the modules. Because the shareable image is prelinked, the linker does not have to perform a complete binding operation each time modules are linked with a shareable image.

A shareable image cannot be run; it must first be linked with other object modules to form an executable image. Usually, shareable images also are installed as global sections and are shared in physical memory, as described in Section 6.4.

System images are intended for stand-alone operations; they do not run under the VAX/VMS operating system. Examples of system images are the VAX/VMS operating system and stand-alone memory diagnostics.

6.4 Sharing

In addition to the sharing of libraries and shareable images that the librarian and the linker provide, VAX/VMS memory management allows more than one process to share pages of physical memory. That is, one page of physical memory can be mapped into the virtual address space of many processes simultaneously. Sharing of physical memory is accomplished using global sections.

Memory management treats a process's virtual address space as a number of sections. Each section of a process maps to all or a portion of a disk file containing data or code. The data or code can be paged into memory and made available to a process for manipulation and execution. These sections can be either private or global. **Private sections** are accessible only by the process that contains them; for example, a private section can correspond to an image section of the image it is executing.

Global sections, on the other hand, can be accessed by more than one process. Memory management creates a global section as a result of either of the following:

- An image's issuing a system service request to create a global section for the sharing of data
- The system manager's installing a shareable image, thus creating a group of global sections for the sharing of code or data

A global section used to share data maps to all or a portion of a disk file containing the data. A global section used to share code maps to an image file. An image can add new pages to a process's virtual address space and it can delete them. For example, an image can create new pages to map a global section and can delete those pages when finished with the global section.

Global sections can be either temporary or permanent. VAX/VMS automatically deletes temporary global sections when all processes that mapped to the section unmap from it. Permanent global sections remain known to the system until explicitly deleted. You must have the appropriate privilege to create a permanent global section.

6.4.1 Global Page Table

Memory management uses a global page table to record the status of each page in a global section. When a process incurs a page fault for a page of a global section, memory management uses the page table entry for that page in the process's page table to index to the master page table entry for the page in the **global page table**.

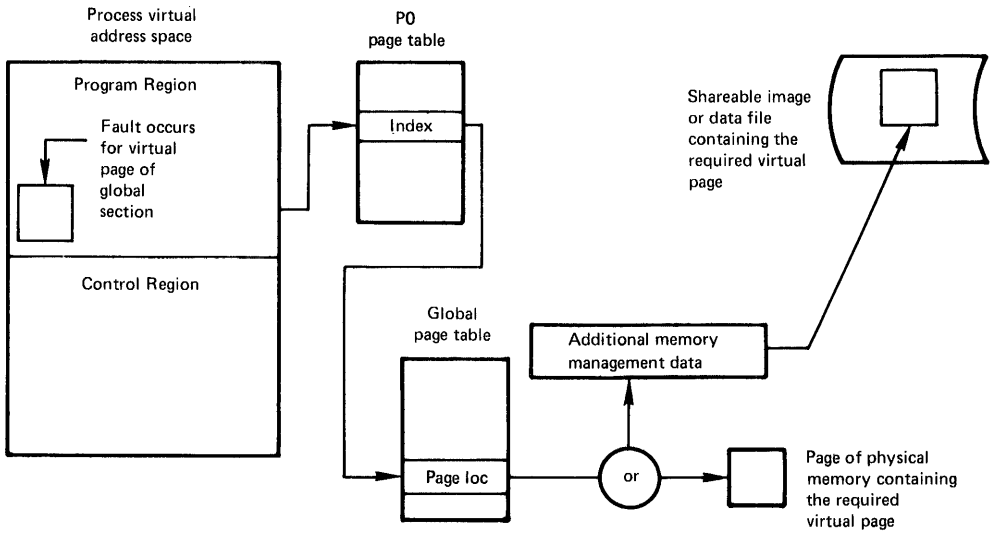
Using the global page table entry, memory management can determine whether:

- The global page is already in memory but is not in the working set of the process incurring the fault; in this case, the page is placed in the working set
- The global page is currently on disk; in this case, the page is brought into memory and into the working set of the process incurring the fault

Figure 6–2 illustrates the mapping of global sections into a process's virtual address space.

Once the needed page is in the process's working set, the process's page table contains the physical address of the page; subsequent references to the page do not require the use of the global page table. Memory management maintains a reference count for each page of a global section so that the page can be removed from memory when it is no longer needed.

Figure 6–2: Mapping Global Sections



6.5 VAX–11 Symbolic Debugger

The VAX–11 Symbolic Debugger is a shareable image that can be linked with an executable image to control program execution during development. The debugging language is similar to the VAX/VMS command language. Expressions and data references are similar to those of the source language being debugged. Debugging commands include the capability to start and interrupt program execution; step through instruction sequences; call routines; set breakpoints, watchpoints, or tracepoints; define symbols; change defaults; and deposit, examine, or evaluate virtual memory locations.

The VAX–11 assembler, VAX–11 compilers, and linker cooperate to perpetuate symbol tables in images that are to be linked with the debugger. The result is that symbol names (labels and value names) can be used in commands to the debugger.

The debugger cannot be used with compatibility mode images.

The *VAX–11 Symbolic Debugger Reference Manual* describes the debugger available under VAX/VMS.

Chapter 7
System Management and Operation

In addition to being responsible for the user authorization file, a VAX/VMS system manager is responsible for controlling the system's operation and accounting for its use. Areas of control include:

- **Spooling**
- Batch processing
- Accounting
- Monitoring system activity
- System recovery
- System update and installation of optional software

A system operator can also control these functions, as detailed in the *VAX/VMS System Manager's Guide*, the *VAX/VMS Operator's Guide*, and the *VAX-11 Software Installation Guide*.

7.1 Spooling

VAX/VMS supports both input and output spooling. The system manager defines and initiates the queues for spooled devices.

Input spooling occurs automatically when a card deck containing a batch **job** is placed in a card reader. VAX/VMS places the batch job in the batch **queue** associated with the card reader. Output spooling occurs when an interactive user or batch job issues a command to print a file, or when a program writes a file to a spooled device.

For output spooling, the system manager can define print queues as needed and assign them to a number of devices, including line printers and terminals. The system manager has complete control over each queue and every job within the queue.

7.2 Batch Processing

Any user logged into the system can submit a file containing commands and data for batch processing. VAX/VMS provides for multiple batch queues, each with multiple active jobs. The system manager defines both the number of batch queues and the number of jobs that can be active in each queue.

When a user submits a batch job, the job is placed in the designated queue according to the user's priority. As each job is dequeued, VAX/VMS creates a process in which to run the job. When the number of jobs (that is, the number of processes created to run batch jobs) reaches the limit that the system

manager imposed for a queue, VAX/VMS stops dequeuing jobs from that queue. As each job terminates, VAX/VMS initiates another job from the queue.

The system manager can control dynamically the number of batch queues present and the number of jobs allowed to execute concurrently from the queues. By exercising these controls, the system manager controls the amount of system resources available for batch processing. In addition, jobs can be controlled on an individual basis, for example, by changing job priorities, placing them in a hold status, or terminating them.

7.3 Accounting

VAX/VMS maintains an accounting log file for collecting cumulative resource usage statistics. The system updates the accounting log file when one of the following conditions is met:

- An interactive process terminates
- A batch process terminates
- A subprocess or a detached process terminates
- A printing job is completed
- A login failure occurs
- A user sends a message to the accounting log file by use of the Send Message to Accounting Manager system service

The accounting log file consists of detailed records that identify the **account name** and **user name** of each statistic. The accounting log file can be used to calculate billing information and to report by account name or user name. Because the system collects all detail records, the system manager or system programmmer can define individual algorithms for resource billing.

7.4 Error Logging

The **error logger** is a job that runs continuously to log errors detected by both hardware and software. The errors include:

- Device errors
- Interrupt timeouts
- Interrupts received from nonexistent devices
- Memory, translation buffer, and cache parity errors
- Datapath errors

In addition, system software sends complete recovery information to the error logger following a power interruption or hardware or software failure.

The error logger writes all messages it receives into an error log file, noting vital system statistics at the time of the message. The error logger also notes

benign events when they occur, such as when volumes are mounted and dismounted, and provides periodic time stamps indicating that no entries have occurred for a specified period of time. The error logger can accept messages from system operators at any time, and from any programs privileged to send messages to the error logger.

The system includes a utility called the error report generating utility program (SYE) that converts the information in the error log file into a text file that can be printed for later study.

7.5 Online Diagnostic Programs

An operator can run diagnostic programs to check the operation of both hardware and software. An operator can run system exercisers and device verification diagnostic programs while normal operations proceed. System exercisers test general purpose software and compare the results with known answers, reporting any discrepancies to the error logger.

Operators can run device verification diagnostic programs either as stand-alone tests or concurrently with other processes. Diagnostic programs check the peripheral functions, including disk head alignment. In addition, fault isolation diagnostics, which isolate problems to replaceable units, are available for stand-alone use.

7.6 System Recovery

An operator can select manual or automatic system recovery following a power interruption or a hardware or software failure. On automatic system recovery after power interruption, the system determines whether the contents of memory are still valid. If they are, the system restarts all possible I/O operations in progress at the time of the power interruption and continues operations from the point of interruption. If the contents of memory are not valid, either because memory battery back-up is not included in the configuration, or because the power failure lasted longer than the battery, the system automatically boots itself from disk and executes the start-up command procedures.

7.7 Operator Utilities

In addition to the system management tools described above, VAX/VMS also provides utilities that often are run by a system operator, possibly at another user's request. These utilities include:

- A disk quota program (DISKQUOTA) — Enables the system manager and operators to regulate the amount of disk space apportioned to individual system users.
- A disk save and compress utility (DSC) — Permits the system manager and operators to back up and restore system disks, public and private disks, and tapes. There are three types of DSCs: DSC1, which is used to back up and restore Files-11 Structure Level 1 volumes; DSC2, which is used to

back up and restore Files-11 Structure Level 2 volumes; and stand-alone DSC-2, which is used to back up and restore Files-11 Structure Level 2 volumes on single-disk-drive configurations.

- A disk structure verification utility (VFY) — Allows the system manager and operators to check data integrity on Files-11 Structure Level 1 and Files-11 Structure Level 2 volumes.
- A bad block locator utility (BAD) — Lets the system manager and operators test whether blocks on Files-11 Structure Level 1 and Files-11 Structure Level 2 volumes are corrupt.
- A system dump analyzer (SDA) — Enables the system manager and operators to interrogate information in the system dump file and determine the reason for the system failure.
- A display utility program — Permits the system manager and operators to display system performance measurement statistics that can aid in improving system performance.

Also available are various RMS utilities. For additional information, see the *VAX-11 Information Directory and Index*.

7.8 Maintenance Updates and Optional Software Installations

System management may also include the task of installing maintenance updates to the VAX/VMS operating system or an optional software product on the VAX-11 processor. Both maintenance update kits and optional software kits are distributed on floppy diskettes (generally, two or more, depending on the update or the software product).

To facilitate the installation of maintenance updates and optional software, the floppy diskettes include a command procedure that, when invoked, executes the copying of files from the floppy diskettes to the system disk. The system manager need only answer the queries issued by the command procedure to ensure proper installation.

The *VAX-11 Software Installation Guide* provides a complete description of the maintenance update procedure. It also describes the preparatory steps for installing optional software products. Each language documentation set then describes its own installation procedure.

Chapter 8
Compatibility Mode

VAX-11 systems provide a compatibility mode of operation that allows PDP-11 programs that meet certain requirements to run under VAX/VMS. The VAX-11 hardware contains a subset of the PDP-11 instruction set. All user mode instructions, except FPP and FIS floating-point instructions, are included. FPP floating-point instructions are emulated in the software.

VAX/VMS emulates the RSX-11M Version 3.2 programming environment. That is, VAX/VMS emulates most RSX-11M executive directives (EMT 377s). Many RSX-11M task images can run under VAX/VMS without rebuilding. Other images compatible with RSX-11M (for example, IAS and RSX-11D) can run under VAX/VMS after task building by the RSX-11M Version 3.2 task builder. The *VAX-11/RSX-11M Programmer's Reference Manual* describes the requirements for RSX-11M image execution under VAX/VMS.

VAX/VMS also supports a version of the MCR command language and MCR directives available under RSX-11M. As a result, VAX/VMS can serve as the host system for RSX-11M/S system generation.

Because many RSX-11M utility programs (for example, PIP and EDI) run under VAX/VMS in compatibility mode, users can interface with the same utilities used in the RSX-11M environment just as they are invoked from RSX-11M MCR. In addition, the MCR command language has been enhanced to allow access to many VAX/VMS features not available in RSX-11M. The *VAX-11/RSX-11M User's Guide* describes VAX/VMS MCR commands and directives.

Appendix A
Process States and System Events

Table A-1: Process States

State Name	Descriptions
Collided page wait (COL PG)	A wait state for processes that have faulted a page currently in transition. The queue contains both resident and nonresident processes.
Compute (COM)	A state for executable processes contained in the balance set. This state is subdivided into 32 queues, one for each process priority.
Compute, out of balance set (COMO)	A state for executable processes not currently contained in the balance set. Actually, this is the set of inswap candidates. This state, like the resident compute state, is subdivided into 32 queues, one for each priority.
Common event flag wait (CEF)	A wait state for processes waiting for some combination of event flags to be set in a common event block (CEB). There is a wait queue for each of the common event blocks linked into a list of common event blocks.
Free page wait (FPG WAIT)	A wait state for a process that requires a free page of memory. This state queue contains both resident and nonresident processes.
Hibernate (HIB)	A wait state for processes that have made a hibernate request. This state queue contains only resident processes.
Hibernate, out of balance set (HIBO)	A wait state for hibernating processes that have been swapped out of the balance set.
Local event flag wait (LEF)	A wait state for resident processes waiting for some combination of local event flags.
Local event flag wait, out of balance set (LEFO)	A wait state for nonresident processes waiting for some combination of local event flags.
Suspended (SUSP)	A wait state for suspended processes currently resident in the balance set.
Suspended, out of balance set (SUSPO)	A wait state for nonresident, suspended processes.
Current process (CUR)	The state of a process actively being executed by the processor.
Miscellaneous wait (MWAIT)	A wait state for both resident and nonresident processes awaiting the availability of a **mutex** semaphore or a dynamic resource.
Page fault wait (PFW)	A wait state for processes that have initiated the read of a page as the result of a page fault. This state queue contains resident processes only.

Table A-2: System Events

System Event Name	Description
AST enqueuing (AST)	The enqueuing of an AST control block that can be delivered to its intended access mode.
Delete process (DEL)	A request to delete a process was made.
Event flag setting (EVENT)	The setting of an event flag has satisfied a wait condition.
Hibernate (HIB)	A request to hibernate was made by the current process.
Inswap (INSWAP)	The swapper has moved a process into the balance set.
Outswap (OUTSWAP)	The swapper has moved a process out of the balance set.
Reschedule (RESCHED)	A higher priority process has become executable and preempted the current process.
Resume (RESUME)	A request has been made to resume the process if it was suspended.
Schedule (SCHED)	The process is assigned to a processor and placed in execution.
Suspend (SUSP)	A suspend request has been issued for the process specified.
Wait for common event flag (WAIT CEF)	A request has been made to wait for some combination of event flags contained in a common event block.
Wait for local event flag (WAIT LEF)	A request has been made to wait for some combination of event flags contained in the software PCB.
Wake (WAKE)	A request has been made to wake the process if it was hibernating.
Wait for mutex (WAIT MUTEX)	The current process requires a mutex semaphore that is currently busy.
Mutex available (MUTEX AVAIL)	A mutex semaphore has become available for a waiting process.
Wait for resource (R WAIT)	The current process requires a dynamic resource that is temporarily unavailable. The software PCB contains the identification number of the resource for which the process is waiting.
Resource available (RES AVAIL)	A dynamic resource is now available. All processes waiting for the specified resource become executable.
Collision page wait (COL PG WAIT)	A process has faulted a page currently in transit.
Collision page in (COL PG IN)	An in-transit page involved in a collision has arrived in memory.
Free page wait (FPG WAIT)	The current process requires a free page and none is currently available.
Free page available (FPG AVAIL)	A free page has become available and processes are waiting for more pages.
Page Fault complete (PF COM)	The I/O operation initiated to read the content of a disk page as a result of a page fault has completed.
Page fault wait (PF WAIT)	The current process must wait for paging I/O to complete.

Glossary

abort

An exception that occurs in the middle of an instruction and sometimes leaves the registers and memory in an indeterminate state, such that the instruction cannot necessarily be restarted.

absolute indexed mode

An indexed addressing mode in which the base operand specifier is addressed in absolute mode.

absolute mode

A mode of address in which the program counter (PC) is used as the register in autoincrement deferred mode. The contents of the PC is the address of the location containing the actual operand.

absolute time

Values expressing a specific date (month, day, and year) and time of day. Absolute time values are always expressed in the system as positive numbers.

access mode

(1) Any of the four processor access modes in which software executes. Processor access modes are, in order from most to least privileged and protected: kernel (mode 0), executive (mode 1), supervisor (mode 2), and user (mode 3). When the processor is in kernel mode, the executing software has complete control of, and responsibility for, the system. When the processor is in any other mode, the processor is inhibited from executing privileged instructions. The processor status longword contains the current access mode field. The operating system uses access modes to define protection levels for software executing in the context of a process. For example, the executive runs in kernel and executive mode and is most protected. The command interpreter is less protected and runs in supervisor mode. The debugger runs in user mode and is not more protected than normal user programs. (2) See also *record access mode.*

access type

(1) The way in which the processor accesses instruction operands. Access types are: read, write, modify, address, and branch. (2) The way in which a procedure accesses its arguments. (3) See also *record access type.*

access violation

An attempt to reference an address that is not mapped into virtual memory or an attempt to reference an address that is not accessible by the current access mode.

accounting manager

The function in the system process called the job controller that writes accounting records to a system accounting log file to track job activity for user process termination, printer and batch jobs, etc.

account name

A string that identifies a particular account used to accumulate data on a job's resource use. All user resources, except disk quotas, are charged to user account names. Disk quotas are charged to user UICs.

ACP

See *ancillary control process.*

adapter control block (ADP)

A structure in the I/O data base that describes either a UNIBUS or MASSBUS adapter.

address

A number used by the operating system and user software to identify a storage location. See also *virtual address* and *physical address.*

address access type

The specified operand of an instruction is not directly accessed by the instruction. The address of the specified operand is the actual instruction operand. The context of the address calculation is given by the data type of the operand.

address space

The set of all possible addresses available to a process. Virtual address space refers to the set of all possible virtual addresses. Physical address space refers to the set of all possible physical addresses sent out on the SBI.

addressing mode

The way in which an operand is specified; for example, the way in which the effective address of an instruction operand is calculated using the general registers. The basic general register addressing modes are: register, register deferred, autoincrement, autoincrement deferred, autodecrement, displacement, and displacement deferred. In addition, there are six indexed addressing modes using two general registers, and literal mode addressing. The PC addressing modes are called: immediate (for register deferred mode using the PC), absolute (for autoincrement deferred mode using the PC), and branch.

ADP

See *adapter control block.*

allocate a device

To reserve a particular device unit for exclusive use. A user process can allocate a device only when that device is not allocated by any other process.

alphanumeric character

An upper- or lowercase letter (A–Z, a–z), a dollar sign ($), an underscore (/), or a decimal digit (0–9).

American Standard Code for Information Interchange (ASCII)

A set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol.

ancillary control process (ACP)

A process that acts as an interface between user software and an I/O driver. An ACP provides functions supplemental to those performed in the driver, such as file and directory management. Three examples of ACPs are: the Files–11 ACP, the magnetic tape ACP, and the network ACP.

AP

See *Argument Pointer.*

Argument Pointer (AP)

General register 12 (R12). By convention, AP contains the address of the base of the argument list for procedures initiated using the CALL instructions.

ASCII

See *American Standard Code for Information Interchange.*

assign a channel

To establish the necessary software linkage between a user process and a device unit before a user process can communicate with that device.

AST

See *asynchronous system trap.*

ASTLVL

See *asynchronous system trap level.*

asynchronous record operation

A mode of record processing in which a user program can continue to execute after issuing a record retrieval or storage request without having to wait for the request to be fulfilled.

asynchronous system trap (AST)

A software-simulated interrupt to a user-defined service routine. ASTs enable a user process to be notified asynchronously with respect to its execution of the occurrence of a specific event. If a user process has defined an AST routine for an event, the system interrupts the process and executes the AST routine when that event occurs. When the AST routine exits, the system resumes the process at the point where it was interrupted.

asynchronous system trap level (ASTLVL)

A value kept in an internal processor register (ASTLVL) that is the most privileged access mode for which an AST is pending. The AST does not occur until the current access mode drops in privilege (rises in numeric value) to a value greater than or equal to ASTLVL. Thus, an AST for an access mode will not be serviced while the processor is executing in a more privileged access mode.

authorization file

See *user authorization file.*

autodecrement indexed mode

An indexed addressing mode in which the base operand specifier uses autodecrement mode addressing.

autodecrement mode

An addressing mode in which the contents of the selected register are decremented, and the result is used as the address of the actual operand for the instruction. The contents of the register are decremented according to the data type context of the register: 1 for byte, 2 for word, 4 for longword and floating, 8 for quadword and double floating.

autoincrement deferred indexed mode

An indexed addressing mode in which the base operand specifier uses autoincrement deferred mode addressing.

autoincrement deferred mode

An addressing mode in which the contents of the specified register is the address of a longword containing the address of the actual operand. The contents of the register are incremented by 4 (the number of bytes in a longword). If the PC is used as the register, this mode is called absolute mode.

autoincrement indexed mode

An indexed addressing mode in which the base operand specifier uses autoincrement mode addressing.

autoincrement mode

An addressing mode in which the contents of the specified register are used as the address of the operand, then the contents of the register are incremented by the size of the operand.

automatic record locking

A VAX-11 RMS capability by which a user can have only one record in a specific file locked at any given time. The lock occurs on every execution of a $FIND or $GET macro instruction (unless the NLK bit is set in the record processing field). The lock is released when the next record is accessed, the current record is updated or deleted, the record stream is disconnected, or the file is closed.

balance set

The set of all process working sets currently resident in physical memory. The processes whose working sets are in the balance set have memory requirements that balance with available memory. The balance set is maintained by the system's swapper process.

base operand address

The address of the base of a table or array referenced by index mode addressing.

base operand specifier

The register used to calculate the base operand address of a table or array referenced by index mode addressing.

base priority

The process priority that the system assigns a process when it is created. A base priority generally comes from the authorization file. The scheduler never schedules a process below its base priority. The base priority can be modified only by the system manager or the process itself. The base priority of a running process can be altered by any user with ALTPRI.

base register

A general register used to contain the address of the first entry in a list, table, array, or other data structure.

binding

See *linking*.

bit string

See *variable-length bit field*.

block

(1) The smallest logically addressable unit of data that a specified device can transfer in an I/O operation (512 contiguous bytes for most disk devices). (2) An arbitrary number of contiguous bytes used to store logically-related status, control, or other processing information.

block I/O

A data accessing technique in which the program manipulates the blocks (physical records) that make up a file, instead of its logical records; allows for the direct access to the blocks in a file without regard for the file organization or record format.

branch access type

An instruction attribute which indicates that the processor does not reference an operand address, but that the operand is a branch displacement. The size of the branch displacement is given by the data type of the operand.

branch mode

An addressing mode in which the instruction operand specifier is a signed byte or word displacement. The displacement is added to the contents of the updated PC (which is the address of the first byte beyond the displacement), and the result is the branch address.

bucket

A storage structure of 1 through 32 blocks, used for building and processing relatively organized files. A bucket contains one or more records or record cells.

bucket locking

A facility that prevents access to any record in a bucket by more than one user until that user releases the bucket.

buffered data path

A UNIBUS adapter data path that transfers 32 or 64 bits of data in a single SBI transfer.

buffered I/O

See *system-buffered I/O*.

bug check

The operating system's internal diagnostic check. The system logs the failure and crashes the system.

byte

A byte is 8 contiguous bits starting on any addressable boundary. Bits are numbered from the right, 0 through 7, with bit 0 the low-order bit. When interpreted arithmetically, a byte is a 2's complement integer with significance increasing from bits 0 through 6. Bit 7 is the sign bit. The value of the signed integer is in the range –128 to 127 decimal. When interpreted as an unsigned integer, significance increases from bits 0 through 7 and the value of the unsigned integer is in the range 0 to 255 decimal. A byte can be used to store one ASCII character.

cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor, and fetches several bytes of data from memory in anticipation that the processor will access the next sequential series of bytes.

call frame

See *stack frame.*

call instructions

The processor instructions CALLG (call procedure with general argument list) and CALLS (call procedure with stack argument list).

call stack

The stack and conventional stack structure used during a procedure call. Each access mode of each process context has one call stack, and interrupt service context has one call stack.

CCB

See *channel control block.*

CF

Current Frame Pointer.

channel

A logical path connecting a user process to a physical device unit. A user process requests the operating system to assign a channel to a device so the process can communicate with that device. See also *controller data channel.*

channel control block (CCB)

A structure in the I/O data base created by the Assign I/O Channel system service to describe the device unit to which a channel is assigned.

channel request block (CRB)

A structure in the I/O data base that describes the activity on a particular controller. The channel request block for a controller contains pointers to the wait queue of drivers ready to access a device through the controller.

character

A symbol represented by an ASCII code. See also *alphanumeric character.*

character string

A contiguous set of bytes. A character string is identified by two attributes: an address and a length. Its address is the address of the byte containing the first character of the string. Subsequent characters are stored in bytes of increasing addresses. The length is the number of characters in the string.

character string descriptor

A quadword data structure used for passing character data (strings). The first word of the quadword contains the length of the character string. The second word can contain type information. The remaining longword contains the address of the string.

cluster

(1) A set of contiguous blocks that is the basic unit of space allocation on a Files–11 disk volume. (2) A set of pages brought into memory in one paging operation. (3) An event flag cluster.

CMP

The compatibility mode bit in the hardware processor status longword (PSL).

command

An instruction, generally an English word, typed by the user at a terminal or included in a command procedure that requests the software monitoring a terminal or reading a command procedure to perform some well-defined activity. For example, typing the COPY command requests the system to copy the contents of one file into another file.

command file

See *command procedure.*

command language interpreter

A procedure-based system code that executes in supervisor mode in the context of a process to receive, to check the syntax of, and to parse commands typed by the user at a terminal or submitted in a command file.

command level

Input stream for the command interpreter. The initial input stream is always command level 0. Each subsequent execution of a procedure changes the command level.

command parameter

The positional operand of a command delimited by spaces, such as a file specification, option, or constant.

command procedure

(1) A file containing commands and data that the command interpreter can accept in lieu of the user's typing the commands individually on a terminal. (2) A set of commands punched on data cards and submitted to the system for processing as a batch job.

command string

A line (or set of continued lines) containing a command and, optionally, information modifying the command. A complete command string consists of a command, its qualifiers, if any, and its parameters (file specifications, for example), if any, and their qualifiers, if any. A command string is normally terminated by pressing the carriage return key.

common

A FORTRAN term for a program section that contains only data.

common event flag cluster

A set of 32 event flags that enables cooperating processes to post event notification to each other. Common event flag clusters are created as they are needed. A process can associate with up to two common event flag clusters.

compatibility mode

A mode of execution that enables the central processor to execute nonprivileged PDP–11 instructions. The operating system supports compatibility mode execution by providing an RSX–11M execution environment for an RSX–11M task image. The operating system compatibility mode procedures intercept calls to the RSX–11M executive and convert them to the appropriate operating system functions.

condition

An error state that exists when an exception occurs. See also *exception* and *condition handler*.

condition codes

The 4 bits in the processor status word (PSW) that indicate the results of previously executed instructions.

condition handler

A procedure that a process wants the system to execute when an exception occurs. When an exception occurs, the operating system searches for a condition handler and, if found, initiates the handler immediately. The condition handler may perform some action to change the situation that caused the exception and continue execution for the process that incurred the exception. Condition handlers execute in the context of the process at the access mode of the code that incurred the exception.

condition value

A 32-bit value that uniquely identifies the exception that caused the condition.

configuration register

A control/status register for an adapter, for example, a UNIBUS adapter. It resides in the adapter's I/O space.

connect-to-interrupt

A function by which a process connects to a device interrupt vector. To perform a connect-to-interrupt, the process must map to the program I/O space containing the vector. See also *page frame number mapping*.

console

The manual control unit integrated into the central processor. The console includes a serial line interface connected to a hard-copy terminal. This enables the operator to start and stop the system, monitor system operation, and run diagnostics.

console terminal

The hard-copy terminal connected to the central processor console.

context

The environment of an activity. See also *process context*, *hardware context*, and *software context*.

context indexing

The ability to index through a data structure automatically because the size of the data type is known and used to determine the offset factor.

context switching

Interrupting the activity in progress and switching to another activity. Context switching occurs as one process after another is scheduled for execution. The operating system saves the interrupted process's hardware context in its hardware PCB using the Save Process Context instruction, loads another process's hardware PCB into the hardware context using the Load Process Context instruction, scheduling that process for execution.

continuation character

A hyphen at the end of a command line signifying that the command string continues on to the next command line.

controller data channel

A logical path to which a driver for a device on a multidevice controller must be granted access before it can activate a device.

control region

The higher-addressed half of per-process space (the P1 region). Control region virtual addresses refer to the process-related information used by the system to control the process, such as: the kernel, executive, and supervisor stacks, the permanent I/O channels, exception vectors, and dynamically used system procedures (such as the command interpreter). The user stack is also normally found in the control region.

Control Region Base Register (P1BR)

The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of a process control region page table.

Control Region Length Register (P1LR)

The processor register, or its equivalent in a hardware process control block, that contains the number of nonexistent page table entries for virtual pages in a process control region.

Control/Status Register (CSR)

A control/status register for a device or controller. It resides in the processor's I/O space.

copy-on-reference

A method used in memory management for sharing data until a process accesses it, in which case it is copied and made private before the access. Copy-on-reference allows sharing of the initial values of a global section whose pages have read/write access but contain pre-initialized data available to many processes.

counted string

A character-string data structure consisting of a byte-sized length followed by the string. Although a counted string is not used as a procedure argument, it is a convenient representation in memory.

CPU

Central Processing Unit.

CRB

See *channel request block.*

CRC

Cyclic redundancy check.

CSR

See *Control/Status Register.*

current access mode

The processor access mode of the currently executing software. The current mode field of the processor status longword indicates the access mode of the currently executing software.

cylinder

The tracks at the same radius on all recording surfaces of a disk.

data base

(1) All the occurrences of data described by a data base management system. (2) A collection of related data structures.

data structure

Any table, list, array, queue, or tree whose format and access conventions are well-defined for reference by one or more images.

data type

In general, the way in which bits are grouped and interpreted. In reference to the processor instructions, the data type of an operand identifies the size of the operand and the significance of the bits in the operand. Operand data types include: byte, word, longword, and quadword integer; floating and double-floating character string; packed decimal string; and variable-length bit field.

DDB

See *device data block.*

DDT

See *driver dispatch table.*

deferred echo

Refers to the fact that terminal echoing does not occur until a process is ready to accept input entered by type ahead.

delta time

A time value expressing an offset from the current date and time. Delta times are always expressed in the system as negative numbers whose absolute value is used as an offset from the current time.

demand zero page

A page, typically of an image stack or buffer area, that is initialized to contain all zeros when dynamically created in memory as a result of a page fault. This feature eliminates the waste of disk space that would otherwise be required to store blocks (pages) that contain only zeros.

descriptor

A data structure used in calling sequences for passing argument types, addresses, and other optional information. See also *character string descriptor*.

detached process

A process that has no owner. The parent process of a tree of subprocesses. Detached processes are created by either the job controller when a user logs on the system, when a batch job is initiated, or when a logical link connect is requested. The job controller does not own the user processes it creates; these processes are therefore detached.

device

The general name for any physical terminus or link connected to the processor that is capable of receiving, storing, or transmitting data. Card readers, line printers, and terminals are examples of record-oriented devices. Magnetic tape devices and disk devices are examples of mass storage devices. Terminal line interfaces and interprocessor links are examples of communications devices.

device data block (DDB)

A structure in the I/O data base that identifies the generic device/controller name and driver name for a set of devices attached to the same controller.

device interrupt

An interrupt received on interrupt priority levels 16 through 23. Device interrupts can be requested only by devices, controllers, and memories.

device name

The field in a file specification that identifies the device unit on which a file is stored. Device names also include the mnemonics that identify an I/O peripheral device in a data transfer request. A device name consists of a mnemonic followed by a controller identification letter (if applicable), followed by a unit number (if applicable), and ends with a colon (:).

device queue

See *spool queue.*

device register

A location in device controller logic used to request device functions (such as I/O transfers) and/or report status.

device unit

One drive and its controlling logic, for example, a disk drive or terminal. Some controllers can have several device units connected to a single controller; for example, mass storage controllers.

diagnostic

A program that tests hardware, firmware, peripheral operation, logic, or memory and reports any faults it detects.

direct data path

A UNIBUS adapter data path that transfers 16 bits of data in a single SBI transfer.

direct I/O

An I/O operation in which the system locks the pages containing the associated buffer in physical memory for the duration of the I/O operation. The I/O transfer takes place directly from the process buffer. Contrast with *system-buffered I/O.*

direct mapping cache

A cache organization in which only one address comparison is needed to locate any data in the cache because any block of main memory data can be placed in only one possible position in the cache. Contrast with *fully associative cache.*

directory

A file used to locate files on a volume. It contains a list of file names (including type and version number) and their unique internal identifications.

directory name

The field in a file specification that identifies the directory file in which a file is listed. The directory name begins with a left bracket ([or <) and ends with a right bracket (] or >).

displacement deferred indexed mode

An indexed addressing mode in which the base operand specified uses displacement deferred mode addressing.

displacement deferred mode

An address mode in which the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of a longword that contains the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The updated contents of the PC is the address of the first byte beyond the specifier extension.

displacement indexed mode

An indexed addressing mode in which the base operand specifier uses displacement mode addressing.

displacement mode

An addressing mode in which the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The updated contents of the PC is the address of the first byte beyond the specifier extension.

double floating data

Eight contiguous bytes (64 bits) starting on an addressable byte boundary which are interpreted as containing a floating point number. The bits are labeled from right to left, 0 to 63. An 8-byte floating point number is identified by the address of the byte containing bit 0. Bit 15 contains the sign of the number. Bits 14 through 7 contain the excess 128 binary exponent. Bits 63 through 16 and 6 through 0 contain a normalized 56-bit fraction with the redundant, most-significant fraction bit not represented. Within the fraction, bits of decreasing significance go from 6 through 0, 31 through 16, 47 through 32, then 63 through 48. Exponent values of 1 through 255 in the 8-bit exponent field represent true binary exponents of -128 to 127. An exponent value of 0 together with a sign bit of 0 represent a floating value of 0. An exponent value of 0 with a sign bit of 1 is a reserved representation; floating point instructions processing this value return a reserved operand fault. The value of a floating data is in the approximate range (+ or -) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 2^{55} or 16 decimal digits.

DPT

See *driver prologue table.*

drive

The electromechanical unit of a mass storage device system on which a recording medium (disk cartridge, disk pack, or magnetic tape reel) is mounted.

driver

The set of code and tables that handles physical I/O to a device.

driver dispatch table (DDT)

A table in the I/O driver that lists the entry point addresses of standard driver routines and the sizes or diagnostic and error logging buffers for the device type.

driver fork level

The interrupt priority levels at which a driver fork processes executes, that is, IPLs 8 through 11. Every unit control block indicates the driver fork level for its unit.

driver prologue table (DPT)

A table in the driver that describes the driver and the device type to the VAX/VMS procedure that loads drivers into the system.

driver start I/O routine

See *start I/O routine*

DST

Debug symbol table.

DV

Decimal overflow trap enable bit in the processor status word (PSW).

dynamic access

A technique in which a program switches from one record access mode to another while processing a file.

ECB

Exit control block.

ECC

Error correction code.

echo

A terminal handling characteristic in which the characters typed by the user on the terminal keyboard are also displayed on the screen or printer.

effective address

The address obtained after deferred or indexing modifications are calculated.

entry mask

A word whose bits represent the registers to be saved or restored on a subroutine or procedure call using the call and return instructions.

entry point

A location that can be specified as the object of a call. It contains an entry mask and exception enables known as the entry point mask.

equivalence name

The string associated with a logical name in a logical name table. An equivalence name can be, for example, a device name, another logical name, or a logical name concatenated with a portion of a file specification.

error logger

A system process that empties the error log buffers and writes the error messages into the error file. Errors logged by the system include memory system errors, device errors and timeouts, and interrupts with invalid vector.

escape sequence

An escape is a transition from the normal mode of operation to a mode outside the normal mode. An escape character is the code that indicates the transition from normal to escape mode. An escape sequence refers to the set of character combinations starting with an escape character that the terminal transmits without interpretation to the software set up to handle escape sequences.

ESP

Executive Mode Stack Pointer.

ESR

Exception service routine.

event

A change in process status or an indication of the occurrence of some activity that concerns an individual process or cooperating processes. An incident reported to the scheduler that affects a process's ability to execute. Events can be synchronous with the process's execution (a wait request), or they can be asynchronous (I/O completion). Some other events include: swapping, wake request, page fault.

event flag

A bit in an event flag cluster that can be set or cleared to indicate the occurrence of the event associated with that flag. Event flags are used to synchronize activities in a process or among many processes.

event flag cluster

A set of 32 event flags used for event posting. Four clusters are defined for each process: two process-local clusters and two common event flag clusters. Of the process-local flags, eight are reserved for system use.

exception

An event detected by the hardware or software (other than an interrupt or jump, branch, case, or call instruction) that changes the normal flow of instruction execution. An exception is always caused by the execution of an instruction or set of instructions (whereas an interrupt is caused by an activity in the system independent of the current instruction). There are three types of hardware exceptions: traps, faults, and aborts. Examples are: attempts to execute a privileged or reserved instruction, trace traps, compatibility mode faults, breakpoint instruction execution, and arithmetic traps such as overflow, underflow, and divide by 0.

exception dispatcher

An operating system procedure that searches for a condition handler when an exception condition occurs. If no exception handler is found for the exception or condition, the image that incurred the exception is terminated.

exception enables

See *trap enables*.

exception vector

See *vector*.

executable image

An image that is capable of being run in a process. When run, an executable image is read from a file for execution in a process.

executive

The generic name for the collection of procedures included in the operating system software that provides the basic control and monitoring functions of the operating system.

executive mode

The second most privileged processor access mode (mode 1). The record management services (RMS) and many of the operating system's system service procedures execute in executive mode.

exit

An image rundown activity that occurs when image execution terminates either normally or abnormally. Image rundown activities include deassigning I/O channels and disassociation of common event flag clusters. Any user- or system-specified exit handlers are called.

exit handler

A procedure executed when an image exits. An exit handler enables a procedure that is not on the call stack to gain control and clean up procedure-owned data bases before the actual image exit occurs.

extended attribute block (XAB)

An RMS user data structure that contains additional file attributes beyond those expressed in the file access block (FAB), such as boundary types (aligned on cylinder, logical block number, virtual block number) and file protection information.

extension

The amount of space to allocate at the end of a file each time a sequential write exceeds the allocated length of the file.

extent

The contiguous area on a disk containing a file or a portion of a file. Consists of one or more clusters.

F11ACP

Files–11 ancillary control process.

FAB

See *file access block.*

failure exception mode

A mode of execution selected by a process indicating that it wants an exception condition declared if an error occurs as the result of a system service call. The normal mode is for the system service to return an error status code for which the process must test.

fault

A hardware exception condition that occurs in the middle of an instruction and that leaves the registers and memory in a consistent state, such that elimination of the fault and restarting the instruction will give correct results.

FCB

File control block.

FCS

File control system.

FDT

Function decision table.

EDT routine

Driver routines called by the Queue I/O Request system service to perform device-dependent preprocessing of an I/O request.

field

A set of contiguous bytes in a logical record. See also *variable-length bit field*.

file access block (FAB)

An RMS user data structure that describes a particular file and contains file-related information needed for data operations, such as OPEN, CLOSE, or CREATE.

file header

A block in the index file describing a file on a Files–11 disk structure. The file header identifies the locations of the file's extents. There is at least one file header for every file on the disk.

file name

The field preceding a file type in a file specification that contains a 1- through 9-character logical name for a file.

file name extension

See *file type*.

file organization

The particular file structure used as the physical arrangement of the data comprising a file on a mass storage medium. RMS file organizations are: sequential, relative, and indexed.

file sharing

An ability to have multiple readers and writers concurrently accessing a particular relative or indexed file.

file specification

A unique name for a file on a mass storage medium. It identifies the node, the device, the directory name, the file name, the file type, and the version number under which a file is stored.

file structure

The way in which the blocks forming a file are distributed on a disk or magnetic tape to provide a physical accessing technique suitable for the way in which the data in the file is processed.

file system

A method of recording, cataloging, and accessing files on a volume.

file type

The field in a file specification that consists of a period (.) followed by a 0- through 3-character type identification. By convention, the type identifies a generic class of files that have the same use or characteristics, such as compiler and assembler listing files, binary object files, etc.

Files–11

The name of the on-disk structure used by the RSX–11, IAS, and VAX/VMS operating systems. Refer also to *Files–11 Structure Level 1* and *Files–11 Structure Level 2.*

Files–11 Structure Level 1

The original Files–11 structure used by IAS, RSX–11M, and RSX–11D for disk volumes. VAX/VMS supports structure level 1 for reasons of compatibility.

Files–11 Structure Level 2

The second generation disk file structure supported by VAX/VMS. It offers improved performance, reliability, and named directories and subdirectories.

fixed-length control area

An area, prefixed to a variable-length record, containing additional information about the record that may have no bearing on the other contents of the record. The fixed-length control area may be used, for example, to contain line numbering or carriage control information.

fixed-length record format

A file format in which all records have the same length.

floating (point) data

Four contiguous bytes (32 bits) starting on an addressable byte boundary. The bits are labeled from right to left from 0 to 31. A 4-byte floating point number is identified by the address of the byte containing bit 0. Bit 15 contains the sign of the number.

Bits 14 through 7 contain the excess 128 binary exponent. Bits 31 through 16 and 6 through 0 contain a normalized 24-bit fraction with the redundant most significant fraction bit not represented. Within the fraction, bits of decreasing significance go from bit 6 through 0, then 31 through 16. Exponent values of 1 through 255 in the 8-bit exponent field represent true binary exponents of -128 to 127. An exponent value of 0 together with a sign bit of 0 represent a floating value of 0. An exponent value of 0 with a sign bit of 1 is a reserved representation; floating point instructions processing this value return a reserved operand fault. The value of a floating data is in the approximate range (+ or -) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 2^{23} or 7 decimal digits.

foreign volume

Any volume other than a Files-11 formatted volume which may or may not be file structured.

fork block

That portion of a unit control block that contains a driver's context while the driver is waiting for a resource. A driver awaiting the processor resource has its fork block linked into the fork queue.

fork dispatcher

A VAX/VMS interrupt service routine that is activated by a software interrupt at a fork interrupt priority level (IPL). Once activated, it dispatches driver fork processes from a driver fork queue until no processes remain in the queue for that IPL.

fork process

A fork process is a minimal context process that executes code under a series of constraints: it executes at raised interrupt priority levels; it uses R0 through R5 only (other registers must be saved and restored); it executes in system virtual address space; it is only allowed to refer to and modify static storage that is never modified by higher interrupt priority level code. VAX/VMS uses software interrupts and fork processes to synchronize executive operations.

fork queue

A queue of driver fork blocks that is awaiting activation at a particular interrupt priority level (IPL) by the VAX/VMS fork dispatcher.

FP

See *Frame Pointer*.

FPD

First part (of an instruction) done.

Frame Pointer (FP)

General register 13 (R13). By convention, FP contains the base address of the most recent call frame on the stack.

FU

Floating underflow trap enable bit in the processor status word (PSW).

fully associative cache

A cache organization in which any block of data from main memory can be placed anywhere in the cache. Address comparison must take place against each block in the cache to find any particular block. Contrast with *direct mapping cache.*

function code

See *I/O function code.*

function decision table (FDT)

A table in the driver that lists all valid function codes for the device and lists the addresses of I/O preprocessing routines associated with each valid function.

function modifier

See *I/O function modifier.*

general register

Any of the sixteen 32-bit registers used as the primary operands of the native mode instructions. The general registers include 12 general purpose registers which can be used as accumulators, as counters, and as pointers to locations in main memory, and the FP, AP, SP, and PC.

generic device name

A device name that identifies the type of device but not a particular unit; a device name in which the specific controller and/or unit number is omitted.

global page table

The page table containing the master page table entries for global sections.

global section

A data structure (e.g., FORTRAN global common) or shareable image section potentially available to all processes in the system. Access is protected by privilege and/or group number of the UIC.

global symbol

(1) A symbol defined in a module that is potentially available for reference by another module. The linker resolves (matches references with definitions) global symbols. Contrast with *local symbol.* (2) A command language symbol that is accessible at all command levels.

global symbol table (GST)

In a library, an index of strongly defined global symbols used to access the modules defining the global symbols. The linker will also put global symbol tables into an image. For example, the linker appends a global symbol table to executable images that are intended to run under the symbolic debugger, and it appends a global symbol table to all shareable images.

group

(1) A set of users who have special access privileges to each other's directories and files within those directories (unless protected otherwise), as in the context system, owner, group, world, where group refers to all members of a particular owner's group. (2) A set of jobs (processes and their subprocesses) with access to a group's common event flags and logical name tables.

group number

The first number in a User Identification Code (UIC).

GST

See *global symbol table.*

hardware context

The values contained in the following registers while a process is executing: the PC; the PSL; the 14 general registers (R0 through R13); the four processor registers (P0BR, P0LR, P1BR and P1LR) that describe the process virtual address space; the SP for the current access mode in which the processor is executing; plus the contents to be loaded in the SP for every access mode other than the current access mode. While a process is executing, its hardware context is continually being updated by the processor. While a process is not executing, its hardware context is stored in its hardware PCB.

hardware process control block (hardware PCB)

A data structure known to the processor that contains the saved hardware context when a process is not executing. A process's hardware PCB resides in its process header.

hibernation

A state in which a process is inactive, but known to the system with all of its current status. A hibernating process becomes active again when a wake request is issued. It

can schedule a wake request before hibernating, or another process can issue its wake request. A hibernating process can also become active long enough to service any AST it may receive while it is hibernating. Contrast with *suspension*.

home block

A block in the index file that contains the volume identification, such as volume label and protection.

IDB

See *interrupt data block*.

image

An image consists of procedures and data bound together by the linker. There are three types of images: executable, shareable, and system.

image activator

A set of system procedures that prepares an image for execution. The image activator establishes the memory management data structures required both to map the image's virtual pages to physical pages and to perform paging.

image exit

See *exit*.

image I/O segment

That portion of the control region that contains the RMS internal file access blocks (IFAB) and I/O buffers for the image currently being executed by a process.

image name

The name of the file in which an image is stored.

image privileges

The privileges assigned to an image when it is installed. See also *process privileges*.

image section (isect)

A group of program sections (psects) with the same attributes (such as read-only access, read/write access, absolute, relocatable, etc.) that is the unit of virtual memory allocation for an image.

immediate mode

An addressing mode in which the PC is used as the register in autoincrement mode addressing.

index file

The file on a Files-11 volume that contains the access information for all files on the volume and enables the operating system to identify and access the volume.

index file bit map

A table in the index file of a Files-11 volume that indicates which file headers are in use.

index register

A register used to contain an address offset.

indexed addressing mode

An addressing mode in which two registers are used to determine the actual instruction operand: an index register and a base operand specifier. The contents of the index register are used as an index (offset) into a table or array. The base operand specifier supplies the base address of the array (the base operand address or BOA). The address of the actual operand is calculated by multiplying the contents of the index register by the size (in bytes) of the actual operand and adding the result to the base operand address. The addressing modes resulting from index mode addressing are formed by adding the suffix "indexed" to the addressing mode of the base operand specifier: register deferred indexed, autoincrement indexed, autoincrement deferred indexed (or absolute indexed), autodecrement indexed, displacement indexed, and displacement deferred indexed.

indexed file organization

A file organization in which a file contains records and a primary key index (and optionally one or more alternate key indices) used to process the records sequentially by index or randomly by index.

indirect command file

See *command procedure.*

input stream

The source of commands and data. It is either the user's terminal, the batch stream, or an indirect command file.

instruction buffer

An 8-byte buffer in the processor used to contain bytes of the instruction currently being decoded and to prefetch instructions in the instruction stream. The control logic continuously fetches data from memory to keep the 8-byte buffer full.

interleaving

Assigning consecutive physical memory addresses alternately between two memory controllers.

interprocess communication facility

A common event flag cluster, mailbox, or global section used to pass information between two or more processes.

interrecord gap

A blank space deliberately placed between data records on the recording surface of a magnetic tape.

interrupt

An event other than an exception or branch, jump, case, or call instruction that changes the normal flow of instruction execution. Interrupts are generally external to the process executing when the interrupt occurs. See also *device interrupt*, *software interrupt*, and *urgent interrupt*.

interrupt data block (IDB)

A structure in the I/O data base that describes the characteristics of a particular controller and points to devices attached to that controller.

interrupt priority level (IPL)

The interrupt level at which a software or hardware interrupt is generated. There are 31 possible interrupt priority levels: IPL 1 is lowest, 31 is highest. The levels arbitrate contention for processor service. For example, a device cannot interrupt the processor if the processor is currently executing at an interrupt priority level equal to or greater than the interrupt priority level of the device's interrupt service routine.

interrupt service routine (ISR)

The routine executed when an interrupt occurs.

interrupt stack (IS)

The system-wide stack used when executing in interrupt service context. At any time, the processor is either in a process context executing in user, supervisor, executive or kernel mode, or in system-wide interrupt service context operating in kernel mode, as indicated by the interrupt stack and current mode bits in the PSL. The interrupt stack is not context switched.

Interrupt Stack Pointer (ISP)

The stack pointer for the system-wide interrupt stack.

Interrupt vector

See *vector*.

I/O data base

A collection of data structures that describes I/O requests, controllers, device units, volumes, and device drivers in a VAX/VMS system. Examples are the driver

dispatch table, driver prologue table, device data table, unit control block, channel request block, I/O request packet, and interrupt data block.

I/O driver

See *driver*.

I/O function

An I/O operation interpreted by the operating system and typically resulting in one or more physical I/O operations.

I/O function code

A 6-bit value specified in a Queue I/O Request system service that describes the particular I/O operation to be performed (e.g., read, write, rewind).

I/O function modifier

A 10-bit value specified in a Queue I/O Request system service that modifies an I/O function code (e.g., read terminal input no echo).

I/O lockdown

The state of a page when it cannot be paged or swapped out of memory.

I/O request packet (IRP)

A structure in the I/O data base that describes an individual I/O request. The Queue I/O Request system service creates an I/O request packet for each I/O request. VAX/VMS and the driver of the target device use information in the I/O request packet to process the request.

I/O rundown

An operating system function in which the system cleans up any I/O in progress when an image exits.

I/O space

The region of physical address space that contains the configuration registers, and device control/status and data registers. These regions are not physically contiguous.

I/O status block (IOSB)

A data structure associated with the Queue I/O Request system service. This service optionally returns a status code, number of bytes transferred, and device/function-dependent information in an I/O status block. It is not returned from the service call, but filled in when the I/O request completes.

IPL

See *interrupt priority level.*

IRP

See *I/O request packet.*

isect

See *image section.*

IS

See *interrupt stack.*

ISP

See *Interrupt Stack Pointer.*

ISR

See *interrupt service routine.*

IV

Integer overflow trap enable bit in the processor status word (PSW).

job

(1) The accounting unit equivalent to a process and the collection of all the subprocesses, if any, that it and its subprocesses create. Jobs are classified as batch and interactive. For example, the job controller creates an interactive job to handle a user's requests when the user logs onto the system and it creates a batch job when the symbiont manager passes a command input file to it. (2) A print job.

job controller

The system process that establishes a job's process context, starts a process running the LOGIN image for the job, maintains the accounting record for the job, manages symbionts, and terminates a process and its subprocesses.

kernel mode

The most privileged processor access mode (mode 0). The operating system's most privileged services, such as I/O drivers and the pager, run in kernel mode.

KSP

Kernel Mode Stack Pointer.

lexical function

A command language construct that the command interpreter evaluates and substitutes before it parses a command string. Lexical functions return information about the current process (the UIC or default directory, for example) and about character strings, (their length or the location of substrings, for example).

librarian

A program that allows the user to create, update, modify, list, and maintain object library, help library, text library, and assembler macro library files.

library file

A direct access file containing one or more modules of the same module type.

limit

The size or number of given items requiring system resources (such as mailboxes, locked pages, I/O requests, open files, etc.) that a job is allowed to have at any one time during execution, as specified by the system manager in the user authorization file. See also *quota*.

line number

A number used to identify a line of text in a file processed by a text editor.

linker

A program that reads one or more object files created by language processors and produces an executable image file, a shareable image file, or a system image file.

linking

The resolution of external references between object modules used to create an image, the acquisition of referenced library routines, service entry points, and data for the image, and the assignment of virtual addresses to components of an image.

literal mode

In literal mode addressing, the instruction operand is a constant whose value is expressed in a 6-bit field of the instruction. If the operand data type is byte, word, longword, or quadword, the operand is zero extended and can express values in the range 0 through 63 (decimal). If the operand data type is floating or double floating, the 6-bit field is composed of two 3-bit fields, one for the exponent and the other for the fraction. The operand is extended to floating or double floating format.

local symbol

(1) A symbol meaningful only to the module that defines it. Symbols not identified to a language processor as global symbols are considered to be local symbols. A language

processor resolves (matches references with definitions) local symbols. They are not known to the linker and cannot be made available to another object module. They can, however, be passed through the linker to the symbolic debugger. Contrast with *global symbol.* (2) A command language symbol name that is accessible only at the current command level and subsequently invoked levels. It is deleted when the command level at which it is defined exits.

locality

See *program locality.*

locate mode

A record access technique in which a program accesses records in an RMS block buffer working storage area to reduce overhead. See also *move mode.*

locking a page in memory

Making a page in a process ineligible for either paging or swapping. A page stays locked in physical memory until VAX/VMS specifically unlocks it.

locking a page in the working set

Making a page within a process ineligible for paging out of the working set for the process. The page can be swapped when the process is swapped. A page stays locked in a working set until it is specifically unlocked.

logical block number

A volume-relative address for identifying a block on a mass storage device. This is in contrast to the block's physical (device-oriented) address and its virtual (file-relative) address. The blocks that form the volume are labeled sequentially starting with logical block 0.

logical I/O function

A set of I/O operations (for example, read and write logical block) that allow restricted direct access to device level I/O operations using logical block addresses.

logical name

A user-specified name for any portion or all of a file specification. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by a user. Logical name assignments are maintained in logical name tables for each process, each group, and the system.

logical name table

A table that contains a set of logical names and their equivalence names for a particular process, a particular group, or the system.

logical record

A group of related fields treated as a unit.

login file

A command procedure that is automatically executed at login and at the beginning of a batch job.

longword

Four contiguous bytes (32 bits) starting on any addressable byte boundary. Bits are numbered from right to left, 0 through 31. The address of the longword is the address of the byte containing bit 0. When interpreted arithmetically, a longword is a 2's complement integer with significance increasing from bit 0 to bit 30. When interpreted as a signed integer, bit 31 is the sign bit. The value of the signed integer is in the range -2,147,483,648 to 2,147,483,647. When interpreted as an unsigned integer, significance increases from bit 0 to bit 31. The value of the unsigned integer is in the range 0 through 4,294,967,295.

macro

A statement that requests a language processor to generate a predefined set of instructions.

mailbox

A software data structure that is treated as a record-oriented device for general interprocess communication. Communication using a mailbox is similar to other forms of device-independent I/O. Senders write to a mailbox, the receiver reads from that mailbox. Some system-wide mailboxes are defined: the error logger and OPCOM read from system-wide mailboxes.

main memory

See *physical memory*.

manual record locking

A capability that allows users to lock multiple records in a file simultaneously. The user has explicit control over the locking and unlocking of records. A lock occurs when the ULK bit is set in the record procesing options field on the execution of a $GET, $FIND, or $PUT macro instruction. Once a record is manually locked, it will remain in that state until it is explicitly unlocked by either the free or release service, or until the stream terminates.

mapping window

A subset of the retrieval information for a file that is used to translate virtual block numbers to logical block numbers.

mass storage device

A device capable of reading and writing data on mass storage media such as a disk pack or a magnetic tape reel.

MBA

MASSBUS adapter.

MBZ

Must be zero.

MCR

See *monitor console routine.*

member number

The second number in a user identification code that uniquely identifies that code.

memory management

The system functions that include the hardware's page mapping and protection and the operating system's image activator and pager.

memory mapping enable (MME)

A bit in a processor register that governs address translation.

MFD

Master file directory.

MFPR

Move From Process Register instruction.

MME

See *memory mapping enable.*

modify access type

The specified operand of an instruction or procedure is read, and is potentially modified and written, during that instruction's or procedure's execution.

module

(1) A portion of a program or program library, as in a source module, object module, or image module. (2) A board, usually made of plastic covered with an electrical conductor, on which logic devices (such as transistors, resistors, and memory chips) are mounted, and circuits connecting these devices are etched, as in a logic module.

monitor console routine (MCR)

The command interpreter in an RSX–11 system. Also a command interpreter in a VAX/VMS system.

mount a volume

(1) To logically associate a volume with the physical unit on which it is loaded (an activity accomplished by system software at the request of an operator). (2) To load or place a magnetic tape or disk pack on a drive and place the drive online (an activity accomplished by a system operator).

move mode

A record I/O access technique in which a program accesses records in its own working storage area. See also *locate mode*.

MTAACP

Magnetic tape ancillary control process.

MTPR

Move To Process Register instruction.

multiport memory

A memory unit that can be connected to multiple processors and that can contain resources (for example, mailboxes, common event flag clusters, and global sections) for use by processes running on different processors.

mutex

A semaphore that is used to control exclusive access to a region of code that can share a data structure or other resource. The mutex (mutual exclusion) semaphore ensures that only one process at a time has access to the region of code.

NAM

See *name block*.

name block (NAM)

An RMS user data structure that contains supplementary information used in parsing file specifications.

native image

An image whose instructions are executed in native mode.

native mode

The processor's primary execution mode in which the programmed instructions are interpreted as byte-aligned, variable-length instructions that operate on the following data types: byte, word, longword, and quadword integers; floating and double floating character strings; packed decimals; and variable-length bit fields. The other instruction execution mode is compatibility mode.

NETACP

Network ancillary control process.

network

A collection of interconnected individual computer systems.

node

An individual computer system in a network that can communicate with other computer systems in the network.

NSP

Network services protocol.

null process

A small system process that is the lowest priority process in the system and takes one entire priority class. The only function of the null process is to accumulate idle processor time.

numeric string

A contiguous sequence of bytes representing up to 31 decimal digits (one per byte) and possibly a sign. The numeric string is specified by its lowest addressed location, its length, and its sign representation.

object module

The binary output of a language processor such as the assembler or a compiler, which is used as input to the linker.

object time system

See *Run-Time Procedure Library*.

offset

A fixed displacement from the beginning of a data structure. System offsets for items within a data structure normally have an associated symbolic name used instead of

the numeric displacement. Where symbols are defined, programmers always reference the symbolic names for items in a data structure instead of using the numeric displacement.

On-Disk Structure Level 1 (ODS–1)

Refer to *Files–11 Structure Level 1.*

On-Disk Structure Level 2 (ODS–2)

Refer to *Files–11 Structure Level 2.*

opcode

The pattern of bits within an instruction that specifies the operation to be performed.

OPCOM

See *operator communication manager.*

operand specifier

The pattern of bits in an instruction that indicates the addressing mode and register, or a displacement that identifies an instruction operand.

operand specifier type

The access type and data type of an instruction's operand(s). For example, the test instructions are of read access type, since they only read the value of the operand. The operand can be of byte, word, or longword data type, depending on whether the opcode is for the TSTB (test byte), TSTW (test word), or TSTL (test longword) instruction.

operator communication manager (OPCOM)

A system process that receives input from a process that wants to inform an operator of a particular status or condition, passes a message to the operator, and tracks the message. OPCOM is always active.

operator's console

Any terminal identified as a terminal attended by a system operator.

owner

In the context system, owner, group, world, an owner is the particular member (of a group) to which a file, global section, mailbox, or event flag cluster belongs.

owner process

The process or subprocess that created a subprocess.

P0

See *program region.*

P0BR

See *Program Region Base Register.*

P0LR

See *Program Region Length Register.*

P0PT

Program region page table.

P1

See *control region.*

P1 through P8

See *parameter.*

P1BR

See *Control Region Base Register.*

P1LR

See *Control Region Length Register.*

P1PT

Control region page table.

packed decimal

A method of representing a decimal number by storing a pair of decimal digits in 1 byte, taking advantage of the fact that only 4 bits are required to represent the numbers 0 through 9.

packed decimal string

A contiguous sequence of up to 16 bytes interpreted as a string of 4-bit fields. Each field represents a digit except the low-order four bits of the highest addressed byte, which represents the sign. The packed decimal string is specified by its lowest addressed location and the number of digits.

page

(1) A set of 512 contiguous byte locations beginning at an even 512-byte boundary used as the unit of memory mapping and protection. (2) The data between the beginning of file and a page marker, between two markers, or between a marker and the end of a file.

page fault

An exception generated by a reference to a page which is not in the faulting process's working set.

page fault cluster size

The number of pages read in on a page fault.

page frame number (PFN)

The high-order 21 bits of the physical address of a page in physical memory.

page frame number mapping (PFN mapping)

Mapping a section to one or more pages in physical memory or I/O space (as opposed to mapping it to a disk file).

page marker

A character or characters (generally a form feed) that separates pages in a file that is processed by a text editor.

pager

A set of kernel mode procedures that executes as the result of a page fault. The pager makes the page for which the fault occurred available in physical memory so that the image can continue execution. The pager and the image activator provide the operating system's memory management functions.

page table entry (PTE)

The data structure that identifies the physical location and status of a page of virtual address space. When a virtual page is in memory, the PTE contains the page frame number needed to map the virtual page to a physical page. When it is not in memory, the page table entry contains the information needed to locate the page on secondary storage (disk).

paging

The action of bringing pages of an executing process into physical memory when referenced. When a process executes, all of its pages are said to reside in virtual memory. Only the actively used pages, however, need to reside in physical memory. The remaining pages can reside on disk until they are needed in physical memory. In VMS, a process is paged either when it references more pages than it is allowed to have in its working set or when it first activates an image in memory. When the process refers to a page not in its working set, a page fault occurs. This causes the operating system's pager to read in the referenced page if it is on disk (and, optionally, other related pages depending on a cluster factor), replacing the least recently faulted pages as needed. This system only pages a process against itself. The operating system's pager does not read in a referenced page if that page is on the free or modified list.

parameter

A value passed to a command procedure equated to a symbol ranging from P1 through P8. See also *command parameter*.

PC

See *Program Counter*.

PCB

See *process control block*.

PCBB

Process Control Block Base Register.

per-process address space

See *process address space*.

PFN

See *page frame number*.

PFN mapping

See *page frame number mapping*.

physical address

The address used by hardware to identify a location in physical memory or on directly-addressable secondary storage devices such as disk. A physical memory address consists of a page frame number and the number of a byte within the page. A physical disk block address consists of a cylinder or track and sector number.

physical address space

The set of all possible 30-bit physical addresses that can be used to refer to locations in memory (memory space) or device registers (I/O space).

physical block number

A physical (device-oriented) address for identifying a block on a mass storage device. This is in contrast to the block's logical (volume-relative) address and its virtual (file-relative) address.

physical I/O functions

A set of I/O functions that allows access to all device level I/O operations except maintenance mode.

physical memory

The memory modules connected to the SBI that are used to store: (1) instructions that the processor can directly fetch and execute, and (2) any other data that a processor is instructed to manipulate. Also called main memory.

PID

See *process identification.*

PME

Performance monitor enable bit in PCB.

position-dependent code

Code that can execute properly only in the locations in virtual address space that are assigned to it by the linker.

position-independent code

Code that can execute properly without modification wherever it is located in virtual address space, even if its location is changed after it is linked. Generally, this code uses addressing modes that form an effective address relative to the PC.

primary vector

A location that contains the starting address of a condition handler to be executed when an exception condition occurs. If a primary vector is declared, that condition handler is the first handler to be executed.

private section

An image section of a process that is not shareable among processes. See also *global section.*

privilege

See *process privileges*, *user privileges*, and *image privileges.*

privileged instructions

In general, any instructions intended for use by the operating system or privileged system programs. In particular, instructions that the processor will not execute unless the current access mode is kernel mode (e.g., HALT, SVPCTX, LDPCTX, MTPR, and MFPR).

procedure

A routine entered by means of a call instruction. See also *command procedure.*

process

The basic entity scheduled by the system software that provides the context in which an image executes. A process consists of an address space and both hardware and software context.

process address space

See *process space*.

process context

The hardware and software contexts of a process.

process control block (PCB)

A data structure used to contain process context. The hardware PCB contains the hardware context. The software PCB contains the software context, which includes a pointer to the hardware PCB.

process header

A data structure that contains the hardware PCB, accounting and quota information, process section table, working set list, and the page tables defining the virtual layout of the process.

process header slots

That portion of the system address space in which the system stores the process headers for the processes in the balance set. The number of process header slots in the system determines the number of processes that can be in the balance set at any one time.

process identification (PID)

A 32-bit binary value that uniquely identifies a process. Each process has a process identification and a process name.

process I/O channel

See *channel*.

process I/O segment

That portion of a process control region that contains the process permanent RMS internal file access block for each open file, and the I/O buffers, including the command interpreter's command buffer and command descriptors.

process name

A 1- to 15-character ASCII string that can be used to identify processes executing under the same group number.

process page tables

The page tables used to describe process virtual memory.

process priority

The priority assigned to a process for scheduling purposes. The operating system recognizes 32 levels of process priority, where 0 is low and 31 high. Levels 16 through 31 are used for real-time processes. The system does not modify the priority of a real-time process (although the system manager or process itself may). Levels 0 through 15 are used for normal processes. The system may temporarily increase the priority of a normal process based on the activity of the process.

process privileges

The privileges granted to a process by the system; these privileges are a combination of user privileges and image privileges. They include, for example, the privilege to: affect other processes associated with the same group as the user's group, affect any process in the system regardless of UIC, set process swap mode, create permanent event flag clusters, create another process, create a mailbox, perform direct I/O to a file-structured device, perform network operations.

process section

See *private section.*

process space

The lowest-addressed half of virtual address space, where process instructions and data reside. Process space is divided into a program region and a control region.

processor register

A part of the processor used by the operating system software to control the execution states of the computer system. Process registers include, for example, the system base and length registers, the program and control region base and length registers, the system control block base register, and the software interrupt request register.

processor status longword (PSL)

A privileged processor register consisting of a word of privileged processor status and the PSW. The privileged processor status information includes: the current IPL (interrupt priority level), the previous access mode, the current access mode, the interrupt stack bit, the trace trap pending bit, and the compatibility mode bit.

processor status word (PSW)

The low-order word of the processor status longword. Processor status information includes: the condition codes (carry, overflow, 0, negative), the arithmetic trap enable bits (integer overflow, decimal overflow, floating underflow), and the trace enable bit.

Program Counter (PC)

General register 15 (R15). At the beginning of an instruction's execution, the PC normally contains the address of a location in memory from which the processor will fetch the next instruction it will execute.

program locality

A characteristic of a program that indicates how close or far apart the references to locations in virtual memory are over time. A program with a high degree of locality does not refer to many widely scattered virtual addresses in a short period of time.

program region

The lower-addressed half of process address space (P0 region). The program region contains the image currently being executed by the process and other user code called by the image.

Program Region Base Register (P0BR)

The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of the page table entry for virtual page number 0 in a process program region.

Program Region Length Register (P0LR)

The processor register, or its equivalent in a hardware process control block, that contains the number of entries in the page table for a process program region.

program section (psect)

A portion of a program with a given protection and set of storage management attributes. Program sections that have the same attributes are gathered together by the linker to form an image section.

programmer number

See *member number.*

project number

See *group number* or *account number.*

psect

See *program section.*

PSL

See *processor status longword.*

PSW

See *processor status word.*

PTE

See *page table entry.*

pure code

See *re-entrant code.*

QIO

Queue I/O Request system service. The VAX/VMS system service that services $QIO and $QIOW requests. The Queue I/O Request system service prepares an I/O request for processing by the driver and performs device-independent preprocessing of the request. This system service also calls driver FDT routines.

quadword

Four contiguous words (64 bits) starting on any addressable byte boundary. Bits are numbered from right to left, 0 to 63. A quadword is identified by the address of the word containing the low-order bit (bit 0). When interpreted arithmetically, a quadword is a 2's complement integer with significance increasing from bit 0 to bit 62. Bit 63 is used as the sign bit. The value of the integer is in the range -2^{63} to $2^{63}-1$.

qualifier

A portion of a command string that modifies a command verb or command parameter by selecting one of several options. A qualifier, if present, follows the command verb or parameter to which it applies and is in the format: /qualifier[=option]. For example, in the command string PRINT filename /COPIES=3, the COPIES qualifier indicates that the user wants three copies of a given file printed.

queue

n: (1) A circular, doubly-linked list. (2) Batch job queue or printer job queue. See also *state queue* and *system queue.* v: To make an entry in a list or table, perhaps using the INSQUE instruction.

queue priority

The priority assigned to a job placed in a spooler queue or a batch queue.

quota

The total amount of a system resource, such as CPU time, that a job is allowed to use in an accounting period, as specified by the system manager in the user authorization file. See also *limit.*

RAB

See *record access block.*

random access by key

The retrieval or storage of a record by specifying the key value. This method of record retrieval and storage applies only to indexed files.

random access by record's file address

The retrieval of a record by its unique address, which is provided to the program by RMS upon successful $GET or $FIND operations. The record's file address (RFA) can subsequently be used to randomly access that same record.

random access by relative record number

The retrieval or storage of a record by specifying its position relative to the beginning of the file. This method of record storage and retrieval applies only to sequential files with fixed-length records and relative files.

read access type

An instruction or procedure operand attribute indicating that the specified operand is only read during instruction or procedure execution.

real-time process

A process assigned to a software priority level between 16 and 31, inclusive. The scheduling priority assigned to a real-time process is never modified by the scheduler, although it can be modified by the system manager or the process itself.

record access block (RAB)

An RMS user control block allocated at either assembly or run time to communicate with VAX-11 RMS. The control block describes the records in a particular file and associates with a file access block to form a record access stream. A RAB defines the characteristics needed to perform record-related operations, such as UPDATE, DELETE, or GET.

record access mode

The method used in RMS for retrieving and storing records in a file. Access is by one of four methods: sequential, random by key, random by record's file address, and random by relative record number.

record cell

A fixed-length area in a relatively organized file that is used to contain one record.

record locking

The ability to control operations being performed on relative and indexed files that are being simultaneously accessed by more than one program and/or more than one record stream. Record locking makes certain that when a program is adding, deleting, or modifying a record on a given stream, another program or stream is not allowed to access the same record or record cell. See also *automatic record locking* and *manual record locking*.

Record Management Services (RMS)

A set of operating system procedures that is called by programs to process files and records within files. RMS allows programs to issue GET and PUT requests at the record level (record I/O) as well as read and write blocks (block I/O). VAX-11 RMS is an integral part of the system software. VAX-11 RMS procedures run in executive mode.

record-oriented device

A device such as a terminal, line printer, or card reader, on which the largest unit of data a program can access in one I/O operation is the device's physical record.

record's file address (RFA)

The unique address of a record in a file that allows records, previously accessed, to be accessed randomly at a subsequent time. This occurs regardless of file organization.

re-entrant code

Code that is never modified during execution. It is possible to let many users share the same copy of a procedure or program written as re-entrant code.

register

A storage location in hardware logic other than main memory. See also *general register*, *processor register*, and *device register*.

register deferred indexed mode

An indexed addressing mode in which the base operand specifier uses register deferred mode addressing.

register deferred mode

An addressing mode in which the contents of the specified register are used as the address of the actual instruction operand.

register mode

An addressing mode in which the contents of the specified register are used as the actual instruction operand.

relative file organization

The arrangement of records in a file where each record occupies a cell of equal length within a bucket. Each cell is assigned a successive number, which represents its position relative to the beginning of the file.

REMACP

Remote I/O ACP.

remote command terminal

A terminal that is logically connected to another node by means of a network, in the way that a command terminal is physically connected to a node by means of a dial-up line.

resource

A physical part of the computer system such as a device or memory, or an interlocked data structure such as a mutex. Quotas and limits control the use of physical resources.

resource wait mode

An execution state in which a process indicates that it will wait until a system resource becomes available when it issues a service request requiring a resource. If a process wants notification when a resource is not available, it can disable resource wait mode during program execution.

return status code

See *status code.*

RFA

See *record's file address.*

RMS

See *Record Management Services.*

Run-Time Procedure Library

The collection of procedures available to native mode images at run time. These procedures may be used by all native mode images, regardless of the language processor used to compile or assemble the program. These procedures also provide support routines for high-level language compilers.

RWED

Read, Write, Execute, Delete.

SBI

See *Synchronous Backplane Interconnect.*

SBR

See *System Base Register.*

scatter/gather

The ability to transfer in one I/O operation data from discontiguous pages in memory to contiguous blocks on disk, or data from contiguous blocks on disk to discontiguous pages in memory.

SCB

See *System Control Block.*

SCBB

See *System Control Block Base Register.*

secondary storage

Random access mass storage.

secondary vector

A location that identifies the starting address of a condition handler to be executed when a condition occurs and (1) the primary vector contains 0 or (2) the handler to which the primary vector points, chooses not to handle the condition.

section

A portion of process virtual memory that has common memory management attributes (protection, access, cluster factor, etc.). It is created from an image section, a disk file, or as the result of a Create Virtual Address Space system service. See also *global section, private section, image section,* and *program section.*

sequential access mode

The retrieval or storage of records in which a program successively reads or writes records one after the other in the order in which they appear, starting and ending at any arbitrary point in the file.

sequential file organization

A file organization in which records appear in the order in which they were originally written. The records can be fixed length or variable length. Sequential file organization permits sequential record access and random access by record's file address. Sequential file organization with fixed length records also permits random access by relative record number.

shareable image

An image that has all of its internal references resolved, but which must be linked with one or more object modules to produce an executable image. A shareable image cannot be executed. A shareable image file can be used to contain a library of routines. A shareable image can be used to create a global section by the system manager.

shared memory

See *multiport memory*.

shell process

A predefined process that the job initiator copies to create the minimum context necessary to establish a process.

signal

(1) An electrical impulse conveying information. (2) The software mechanism used to indicate that an exception condition was detected.

slave terminal

A terminal from which it is not possible to issue commands to the command interpreter. A terminal allocated to application software.

SLR

See *System Length Register*.

software context

The context maintained by the VAX/VMS to describe a process. See software process control block (PCB).

software interrupt

An interrupt generated on interrupt priority levels 1 through 15, which can be requested only by software.

software priority

See *process priority* and *queue priority*.

software process control block (software PCB)

The data structure used to contain a process's software context. The operating system defines a software PCB for every process when the process is created. The software PCB includes the following kinds of information about the process: current state; storage address if it is swapped out of memory; unique identification of the

process; and address of the process header (which contains the hardware PCB). The software PCB resides in system region virtual address space. It is not swapped with a process.

SP

See *Stack Pointer.*

spool queue

The list of files supplied by processes that are to be processed by a symbiont. For example, a line printer queue is a list of files to be printed on the line printer.

spooling

The technique of using a high-speed mass storage device to buffer data passing between low-speed I/O devices and high-speed memory. (1) Output spooling: The method by which output to a low-speed peripheral device (such as a line printer) is placed into queues maintained on a high-speed device (such as disk) to await transmission to the low-speed device. (2) Input spooling: The method by which input from a low-speed peripheral (such as the card reader) is placed into queues maintained on a high-speed device (such as disk) to await transmission to a job processing that input.

SPT

See *system page table.*

SSP

Supervisor Mode Stack Pointer.

stack

An area of memory set aside for temporary storage, or for procedure and interrupt service linkages. A stack uses the last-in, first-out concept. As items are added to ("pushed on") the stack, the SP decrements. As items are retrieved from ("popped off") the stack, the SP increments.

stack frame

A standard data structure built on the stack during a procedure call, starting from the location addressed by the FP to lower addresses, and popped off during a return from procedure. Also called call frame.

Stack Pointer (SP)

General register 14 (R14). SP contains the address of the top (lowest address) of the processor-defined stack. Reference to SP will access one of the five possible stack pointers, kernel, executive, supervisor, user, or interrupt, depending on the value in the current mode and interrupt stack bits in the PSL.

start I/O routine

The routine in a device driver that is responsible for obtaining necessary resources (for example, the controller data channel) and activating the device unit.

state queue

A list of processes in a particular processing state. The scheduler uses state queues to keep track of processes' eligibility to execute. They include: processes waiting for a common event flag, suspended processes, and executable processes.

status code

A longword value that indicates the success or failure of a specific function. For example, system services always return a status code in R0 upon completion.

store through

See *write through*.

strong definition

Definition of a global symbol that is explicitly available for reference by modules linked with the module in which the definition occurs. The linker always lists a global symbol with a strong definition in the symbol portion of the map. The librarian always includes a global symbol with a strong definition in the global symbol table of a library. Contrast with *weak definition*.

strong reference

A reference to a global symbol in an object module that requests the linker to report an error if it does not find a definition for the symbol during linking. If a library contains the definition, the linker incorporates the library module defining the global symbol into the image containing the strong reference.

subprocess

A subsidiary process created by another process. The process that creates a subprocess is its owner. A process and its subprocesses share a pool of quotas and limits. When an owner process is removed from the system, all its subprocesses (and their subprocesses) are also removed.

supervisor mode

The third most privileged processor access mode (mode 2). The operating system's command interpreter runs in supervisor mode.

suspension

A state in which a process is inactive, but known to the system. A suspended process becomes active again only when another process requests the operating system to resume it. Contrast with *hibernation*.

SVA

See *system virtual address.*

swap mode

A process execution state that determines the eligibility of a process to be swapped out of the balance set. If process swap mode is disabled, the process working set is locked in the balance set.

swapping

The method for sharing memory resources among several processes by writing an entire working set to secondary storage (swap out) and reading another working set into memory (swap in). For example, a process's working set can be written to secondary storage while the process is waiting for I/O completion on a slow device. It is brought back into the balance set when I/O completes. Contrast with *paging.*

symbiont

A full process that transfers record-oriented data to or from a mass storage device. For example, an input symbiont transfers data from card readers to disks. An output symbiont transfers data from disks to line printers.

symbiont manager

The function (in the system process called the job controller) that maintains spool queues, and dynamically creates symbiont processes to perform the necessary I/O operations.

symbol

See *local symbol, global symbol,* and *universal symbol.*

Synchronous Backplane Interconnect (SBI)

The part of the hardware that interconnects the processor, memory controllers, MASSBUS adapters, the UNIBUS adapter.

synchronous record operation

A mode of record processing in which a user program issues a record read or write request and then waits until that request is fulfilled before continuing to execute.

system

In the context system, owner, group, world, the system refers to the group numbers of less than or equal to 10 (octal) which are used by operating system and its controlling users, the system operators, and the system manager.

system address space

See *system space* and *system region.*

System Base Register (SBR)

A processor register containing the physical address of the base of the system page table.

system-buffered I/O

An I/O operation, such as terminal or mailbox I/O, in which an intermediate buffer from the system buffer pool is used instead of a process-specified buffer. Contrast with *direct I/O.*

system control block (SCB)

The data structure in system space that contains all the interrupt and exception vectors known to the system.

System Control Block Base Register (SCBB)

A processor register containing the base address of the system control block.

system device

The random access mass storage device unit on which the volume containing the operating system software resides.

system dynamic memory

Memory reserved for the operating system to allocate as needed for temporary storage. For example, when an image issues an I/O request, system dynamic memory is used to contain the I/O request packet. Each process has a limit on the amount of system dynamic memory that can be allocated for its use at one time.

System Identification Register

A processor register which contains the processor type and serial number.

system image

The image that is read into memory from disk when the system is started up.

System Length Register (SLR)

A processor register containing the system page table in longwords.

system page table (SPT)

The data structure that maps the system region virtual addresses, including the addresses used to refer to the process page tables. The SPT contains one PTE for each page of system region virtual memory. The physical base address of the SPT is contained in a register called SBR.

system programmer

A person who designs and/or writes operating systems, or who designs and writes procedures or programs that provide general purpose services for an application system.

system queue

A queue used and maintained by operating system procedures. See also *state queue*.

system region

The third quarter of virtual address space. The lowest-addressed half of system space. Virtual addresses in the system region are shareable between processes. Some of the data structures mapped by system region virtual addresses are: system entry vectors, the SCB, the SPT, and process page tables.

system services

Procedures provided by the operating system that can be called by user images.

system space

The highest-addressed half of virtual address space. See also *system region*.

system virtual address (SVA)

A virtual address identifying a location in system space.

system virtual space

See *system space*.

task

An RSX–11/IAS term for a process and image bound together.

terminal

The general name for peripheral devices that have keyboards and video screens or printers. Under program control, a terminal enables users to type commands and data on the keyboard and receive messages on the video screen or printer. Examples of terminals are the LA36 DECwriter hard-copy terminal and VT52 video display terminal.

timeout

The expiration of the time limit in which a device is to complete an I/O transfer. The driver's wait for interrupt request specifies the timeout limit.

timer

Two system processes: one that maintains the time of day and the date, and another that scans for device timeouts and performs time-dependent scheduling upon request. The timer interrupt service routine creates the timer process.

traceback

The system facility that examines and displays the status of the user call stack when an image terminates abnormally.

track

A collection of blocks at a single radius on one recording surface of a disk.

transfer address

The address of the location containing a program entry point (the first instruction to execute).

translation buffer

An internal processor cache containing translations for recently used virtual addresses.

trap

An exception condition that occurs at the end of the instruction that caused the exception. The PC saved on the stack is the address of the next instruction that would normally have been executed. All software can enable and disable some of the trap conditions with a single instruction.

trap enables

Three bits in the PSW that control the processor's action on certain arithmetic exceptions.

2's complement

A binary representation for integers in which a negative number is one greater than the bit complement of the positive number.

two-way associative cache

A cache organization which has two groups of directly mapped blocks. Each group contains several blocks for each index position in the cache. A block of data from

main memory can go into any group at its proper index position. A two-way associative cache is a compromise between the extremes of fully associative and direct mapping cache organizations that takes advantage of the features of both.

type-ahead

A terminal handling technique in which the user can enter commands and data while the software is processing a previously entered command. The commands typed ahead are not echoed on the terminal until the command processor is ready to process them. They are held in a type-ahead buffer.

UBA

UNIBUS adapter.

UCB

See *unit control block.*

UETP

See *User Environment Test Package.*

UFD

See *directory.*

UIC

See *User Identification Code.*

unit control block (UCB)

A structure in the I/O data base that describes the characteristics of and current activity on a device unit. The unit control block also holds the fork block for its unit's device driver; the fork block is a critical part of a driver fork process. The UCB also provides a dynamic storage area for the driver.

unit record device

A device such as a card reader or line printer.

universal symbol

A global symbol in a shareable image that can be used by modules linked with that shareable image. Universal symbols are typically a subset of all the global symbols in a shareable image. When creating a shareable image, the linker ensures that universal symbols remain available for reference after symbols have been resolved.

unwind the call stack

To remove call frames from the stack by tracing back through nested procedure calls using the current contents of the FP register and the FP register contents stored on the stack for each call frame.

urgent interrupt

An interrupt received on interrupt priority levels 24 through 31. These can be generated only by the processor for the interval clock, serious errors, and power fail.

user authorization file

A file containing an entry for every user that the system manager authorizes to gain access to the system. Each entry identifies the user name, password, default account, UIC, quotas, limits, and privileges assigned to individuals who use the system.

User Environment Test Package (UETP)

A collection of routines that verify that the hardware and software systems are complete, properly installed, and ready to use.

user file directory (UFD)

See *directory*.

user identificaton code (UIC)

The pair of numbers assigned to users and to files, global sections, common event flag clusters, and mailboxes that specifies the type of access (read and/or write access; and in the case of files, execute and/or delete access) available to the owners, group, world, and system. The UIC consists of a group number and a member number separated by a comma and enclosed within square brackets.

user mode

The least privileged processor access mode (mode 3). User processes and Run-Time Library Procedures run in user mode.

user name

The name that a user types on a terminal to log on to the system.

user number

See *member number*.

user privileges

The privileges granted a user by the system manager. See also *process privileges*.

USP

User Mode Stack Pointer.

utility

A program that provides a set of related general purpose functions, such as a program development utility (an editor, a linker, etc.), a file management utility (file copy or file format translation program), or operations management utility (disk quotas, diagnostic program, etc.).

value return registers

The general registers R0 and R1 used by convention to return function values. These registers are not preserved by any called procedures. They are available as temporary registers to any called procedure. All other registers (R2, R3,...,R11, AP, FP, SP, PC) may be preserved across procedure calls.

variable-length bit field (VBF)

A set of 0 to 32 contiguous bits located arbitrarily with respect to byte boundaries. A variable bit field is specified by four attributes: 1) the address A of a byte, 2) the bit position P of the starting location of the bit field with respect to bit 0 of the byte at address A, 3) the size, in bits, of the bit field, and 4) whether the field is signed or unsigned.

variable-length record format

A file format in which records are not necessarily the same length.

variable with fixed-length control record format

A file format in which records of variable length contain an additional fixed-length control area. The control area may be used to contain file line numbers and/or print format controls.

VBF

See *variable-length bit field.*

VCB

Volume control block.

vector

(1) An interrupt or exception vector is a storage location known to the system that contains the starting address of a procedure to be executed when a given interrupt or exception occurs. The system defines separate vectors for each interrupting device controller and for classes of exceptions. Each system vector is a longword. (2) For the purposes of exception handling, users can declare up to two software exception vectors (primary and secondary) for each of the four access modes. Each vector contains the address of a condition handler. (3) A one-dimensional array.

version number

(1) The field following the file type in a file specification. It begins with a semicolon (;) or period (.) and is followed by a number which generally identifies it as the latest file created of all files having the identical file specification but for version number. (2) The number used to identify the revision level of program.

virtual address

A 32-bit integer identifying a byte location in virtual address space. The memory management hardware translates a virtual address to a physical address. The term virtual block number (VBN) refers to the address used to identify a virtual block on a mass storage device.

virtual address space

The set of all possible virtual addresses that an image executing in the context of a process can use to identify the location of an instruction or data. The virtual address space seen by the programmer is a linear array of 4,294,967,296 (2^{32}) byte addresses.

virtual block

A block on a mass storage device referred to by its file-relative address rather than its logical (volume-oriented) or physical (device-oriented) address. The first block in a file is always virtual block 1.

virtual I/O functions

A set of I/O functions that must be interpreted by an ancillary control process.

virtual memory

The set of storage locations in physical memory and on disk that is referred to by virtual addresses. From the programmer's viewpoint, the secondary storage locations appear to be locations in physical memory. The size of virtual memory in any system depends on the amount of physical memory available and the amount of disk storage used for nonresident virtual memory.

virtual page number (VPN)

The virtual address of a page of virtual memory.

volume

A mass storage medium such as a disk pack or reel of magnetic tape.

volume set

The file-structured collection of data residing on one or more mass storage media.

VPN

See *virtual page number.*

wait

To become inactive. A process enters a process wait state when the process suspends itself, hibernates, or declares that it needs to wait for an event, resource, mutex, etc.

wait for interrupt request

A request made by a driver's start I/O routine after it activates a device. The request causes the driver fork process to be suspended until the device requests an interrupt or the device times out.

wake

To activate a hibernating process. A hibernating process can be awakened by a time-scheduled wake-up call.

WCB

Window control block.

WCS

Writeable control store.

WDCS

Writeable diagnostic control store.

weak definition

Definition of a global symbol that is not explicitly available for reference by modules linked with the module in which the definition occurs. The librarian does not include a global symbol with a weak definition in the global symbol table of a library. Weak definitions are often used when creating libraries to identify those global symbols that are needed only if the module containing them is otherwise linked with a program. Contrast with *strong definition.*

weak reference

A reference to a global symbol that requests the linker not to report an error or to search the default library's global symbol table to resolve the reference if the definition is not in the modules explicitly supplied to the linker. Weak references are often used when creating object modules to identify those global symbols that may not be needed at run time.

wild card character

A symbol, such as an asterisk or percent sign, that is used within or in place of a file name, file type, directory name, or version number in a file specification to indicate "all" for the given field.

window

See *mapping window.*

word

Two contiguous bytes (16 bits) starting on an addressable byte boundary. Bits are numbered from the right, 0 through 15. A word is identified by the address of the byte

containing bit 0. When interpreted arithmetically, a word is a 2's complement integer with significance increasing from bit 0 to bit 14. If interpreted as a signed integer, bit 15 is the sign bit. The value of the integer is in the range -32768 to 32767. When interpreted as an unsigned integer, significance increases from bit 0 through bit 15 and the value of the unsigned integer is in the range 0 through 65535.

working set

The set of pages in process space to which an executing process can refer without incurring a page fault. The working set must be resident in memory for the process to execute. The remaining pages of that process, if any, are either in memory and not in the process working set or they are on secondary storage.

working set swapper

A system process that brings process working sets into the balance set and removes them from the balance set.

world

In the context system, owner, group, world, world refers to all users, including the system operators, the system manager, and users both in an owner's group and in any other group.

write access type

The specified operand of an instruction or procedure is only written during that instruction's or procedure's execution.

write allocate

A cache management technique in which cache is allocated on a write miss as well as on the usual read miss.

write back

A cache management technique in which data from a write operation to cache is copied into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. Contrast with *write through*.

write through

A cache management technique in which data from a write operation is copied in both cache and main memory. Cache and main memory data are always consistent. Contrast with *write back*.

XAB

See *extended attribute block*.

XDELTA

A tool for debugging operating systems and drivers.

Index

Q

R

S

T

U

V

W

VAX/VMS
Summary Description and Glossary
AA-D022B-TE

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

— — Do Not Tear - Fold Here —