EY-0016E-TP-0001

VMS Internals

Tests/Exercises

digital

EY-0016E-TP-0001

VMS Internals

Tests/Exercises

Prepared by Educational Services
of
Digital Equipment Corporation

Printed in U.S.A.

CONTENTS

System Components

TEST/EXERCISE

For each system component named below, fill in the required information.

a. Under "Implementation", specify system process (PCS), procedure (PCR), exception service routine (EXC), interrupt service routine (INT), or shared image (SHR).

b. Under "Context", indicate system (SYS) or process (PCS).

c. Under "Address Region", specify program (PGM), control (CTL), or system (SYS).

d. Under "Purpose", briefly describe the primary function of the component.

Component Name	Implementation	Context	Address Region	Purpose
system service	PCR	PCS	SYS	common internal function
1. scheduler				
2. swapper				
3. symbiont				
4. AME				
5. ACP				
6. run-time library				
7. error logger				
8. pager				
9. CLI				
10. RMS				

ANSWER SHEET

Component Name	Implementation	Context	Address Region	Purpose
system service	PCR	PCS	SYS	common internal function
1. scheduler	INT	SYS	SYS	chooses next process to execute
2. swapper	PCS	PCS	SYS	system-wide mem.management
3. symbiont	PCS	PCS	PGM	input/output spooling
4. AME	EXC	PCS	PGM	implements compatibility mode
5. ACP	PCS	PCS	PGM	formatting or protocol processing
6. run-time library	PCR	PCS	PGM	common subroutines and functions
7. error logger	PCS	PCS	PGM	records hardware errors
8. pager	EXC	PCS	SYS	process memory management
9. CLI	SHR	PCS	CTL	command language processing
1Ø. RMS	PCR	PCS	SYS	record/file management

TEST/EXERCISE

For each feature/resource associated with or used by a process and listed on the following page:

1. Name the data structure/component that implements or controls it.

2. State the region (program, control , or system) in which the data structure/component resides.

3. State whether the data structure/component is paged.

4. State whether the data structure/component is included in the working set of the process and swapped.

For resources that are **not** part of a larger data structure (an example is the user stack), simply copy the name into the data structure column. For resources that occur in multiple locations, answer for each location.

TEST/EXERCISE

Resources	Data Structure	Region	Paged?	Swapped?
user stack	user stack	control	yes	yes
page tables				
privilege mask				
CLI data areas				
run-time library				
general purpose regs. when process is not the current one				
process priority				
quotas/limits on system resources				
VAX-11 RMS code				
image of user program				
working set list				
kernel stack				
process I/O data structures				
process ID				
CLI code				
interrupt stack				

ANSWER SHEET

Resources	Data Structure	Region	Paged?	Swapped?
user stack	user stack	control	yes	yes
page tables	process header	system	yes	yes
privilege mask	process header software PCB pointer page	system* system control	no no no	yes no yes
CLI data areas	CLI data areas	control	yes	yes
run-time library	run-time library	program	yes	yes**
general purpose regs. when process is not the current one	hardware PCB	system*	no	yes
process priority	software PCB	system*	no	yes
quotas/limits on system resources	software PCB JIB	system system	no no	no no
VAX-11 RMS code	RMS code	system	yes	no
image of user program	image	program	yes	yes**
working set list	process header	system*	no	yes
kernel stack	kernel stack	control	no	yes
process I/O data structures	process I/O data structures	control	yes	yes
process ID	software PCB	system	no	no
CLI code	CLI code	control	yes	yes**
interrupt stack	interrupt stack	system	no	no

*These portions of the process header are also mapped by the "window" in the **control** region of the process.

These software components are or may be **global read only sections. As such, they are included in the working set of the process, but in certain situations are not outswapped with the rest of the working set. See the Internals and Data Structures manual, for further details.

TEST/EXERCISE

1. A system feature called set attention ASTs is employed by several device drivers in the system. A process can specify that it receive an AST when someone writes or reads its mailbox. A process can receive an AST when someone types a CTRL-Y or a CTRL-C at the terminal.

   In these cases, the mailbox driver or the terminal driver must call a routine to queue an AST to the process, which requires modifying scheduling data structures.

   Before this routine (SCH$QAST) is called, the driver requests fork processing at IPL 6. This fork processing raises IPL to 7 (IPL$_SYNCH), and then calls SCH$QAST.

   a. Why can't these drivers simply request fork processing at IPL 7?

   b. Why is IPL 6 used rather than having the driver simply lower the IPL from the present level to IPL 7?

   c. Why can't a mutex be used to lock the scheduler's data structures?

TEST/EXERCISE

2. When an exception or interrupt occurs, the PSL and the PC are pushed onto the stack, and a new PC and PSL are created.

   a. Which stack is used?

   b. How is the new PC value formed?

   c. What are the contents of the current mode and previous mode fields of the new PSL?

   d. What is the new IPL?

   e. When an REI instruction is executed, is the previous mode field of the PSL significant? Explain.

3.

   a. The following table illustrates a hypothetical sequence of hardware and software interrupts. At each step, fill in the contents of the indicated items. In the "Saved IPL" column, indicate the stack which contains the saved IPL. Indicate where control is passed after each REI instruction. All numbers are decimal. Assume that software interrupts above IPL 6 are handled on the interrupt stack, and that those at IPL 1 through IPL 6 are handled on the kernel stack. Further assume that all device interrupts are handled on the interrupt stack.

TEST/EXERCISE

Note that this example is hypothetical and bears little resemblance to the VAX/VMS operating system. Its purpose is to explore the workings of interrupts, especially software interrupts.

Event	Stack	IPL	SISR(hex)	Saved IPL
1. Executing user image				
2. Device int. at IPL 21				
3. SOFTINT 8				
4. REI to _____				
5. SOFTINT 5				
6. SOFTINT 3				
7. REI to _____				
8. Device int. at IPL 20				
9. SOFTINT 8				
10. REI to _____				
11. SOFTINT 4				
12. REI to _____				
13. REI to _____				
14. REI to _____				
15. REI to _____				

TEST/EXERCISE

b. In steps 7 and 12, a switch is made from the interrupt stack to the kernel stack. Why?

4.

a. Briefly describe how system services are dispatched. Assume that **no** errors occur. Include all steps from the program's initial call until control is passed back to that program.

b. Why does the routine SRVEXIT issue an REI instruction?

c. Several system services have access mode as one of their arguments. The service routines which perform these requests first call a routine called Maximize Access Mode which chooses the **least** privileged access mode of the one requested and the access mode of the caller. Describe how this might be done. Why is it done?

TEST/EXERCISE

5. List two differences between the exception dispatching within the executive and the common run-time library procedure LIB$SIGNAL.

ANSWER SHEET

1.

a. The SCB is set up so that the IPL 7 software interrupts are serviced by the software timer routine. If a driver mistakenly issued a request for fork processing at IPL 7, the system would become confused because the requested software interrupt would be serviced not by the fork dispatcher but by the software timer interrupt service routine (which was not expecting the interrupt).

b. The principle behind using IPL as a synchronization tool is that all competing software must raise IPL to synchronization level. Lowering IPL defeats the purpose of synchronization. Consider the following sequence:

   i. A routine following the synchronization conventions correctly raises IPL to 7 before looking at some scheduling data.

   ii. A device interrupts and the associated interrupt service routine requests fork processing at IPL 8.

   iii. The fork processing entry point of the driver lowers IPL to 7, writes into the scheduler's data and issues an REI instruction back to ...

   iv. ... the previously interrupted code (at IPL 7) which is now looking at changed and possibly corrupted data.

c. Mutexes are a synchronization technique available to processes. When on the interrupt stack, the system is not in any process context and hence the method of elevating IPL is the only synchronization technique available.

2.

a. The entry to an exception or interrupt service routine must be longword aligned. Thus, the two low bits in the SCB can be used for other purposes. Bit Ø determines whether the interrupt is handled on the kernel stack (bit Ø clear) or on the interrupt stack (bit Ø set).

ANSWER SHEET

All device interrupts are handled on the interrupt stack. All software interrupts (except ASTDEL at IPL 2 and RESCHED at IPL 3) are handled on the interrupt stack.

CHMx exceptions are placed on the resultant per-process stack. Machine Check, Power Fail, and Kernel Stack Not Valid exceptions are handled on the interrupt stack. The rest of the exceptions are handled on the kernel stack.

b. The new PC value is the address found in bits<31:2> of the SCB entry for this particular exception or interrupt. (PC bits<1:Ø> are always cleared.)

c. For all exceptions **except** CHMU, CHMS and CHME, the current mode will be zero, kernel access mode.

For exceptions, the previous mode field will be the access mode that the CPU was in when the exception occurred. In fact, PSL<previous mode> is the same as the current mode field of the saved PSL on the stack.

The previous mode field of the PSL is set to Ø (kernel mode) following an interrupt.

d. The new IPL depends upon the interrupt or exception:

Exceptions	IPL (decimal)
Machine check	31
Kernel stack not valid	31
all other exceptions	unchanged!
Software Interrupts	IPL raised to corresponding level
Hardware Interrupts	
Interval timer	24
Console	2Ø
Other devices	2Ø-23
Power fail	3Ø

ANSWER SHEET

e. No, the previous mode field of the PSL is not significant when an REI executes. The previous mode field is an historical parameter, recording where the processor came from. The previous mode field is used by the PROBEx instructions.

The relevant field (and the one checked by the REI instruction microcode) is the current mode field of the PSL **on the stack.** If privileged software wishes to alter its destination, IPL, or mode, then this longword is what should be changed.

3.

a.

Event	Stack	IPL	SISR(hex)	Saved IPL
1. Executing user image	user	Ø	Ø	--
2. Device int. at IPL 21	interrupt	21	Ø	Ø(I)
3. SOFTINT #8	interrupt	21	1ØØ	Ø(I)
4. REI to IPL 8 serv. routine	interrupt	8	Ø	Ø(I)
5. SOFTINT #5	interrupt	8	2Ø	Ø(I)
6. SOFTINT #3	interrupt	8	28	Ø(I)
7. REI to IPL 5 serv. routine	kernel	5	8	Ø(K)
8. Device int. at IPL 2Ø	interrupt	2Ø	8	5(I),Ø(K)
9. SOFTINT #8	interrupt	2Ø	1Ø8	5(I),Ø(K)
1Ø. REI to IPL 8 serv. routine	interrupt	8	8	5(I),Ø(K)

ANSWER SHEET

3.a. (continued)

11. SOFTINT #4	interrupt	8	18	5(I),Ø(K)
12. REI to interrupted IPL 5 serv. routine	kernel	5	18	Ø(K)
13. REI to IPL 4 serv. routine	kernel	4	8	Ø(K)
14. REI to IPL 3 serv. routine	kernel	3	Ø	Ø(K)
15. REI to interrupted user image	user	Ø	Ø	--

b. At step 7, the REI triggers a software interrupt at IPL 5. One of the assumptions was that IPL 5 (actually IPL 6 and below) interrupts were to be handled on the kernel stack.

At step 12, the restored PSL requires IPL 5 but also PSL<IS> is clear. The REI instruction microcode then switches stacks, in this case to the kernel stack.

4.

a. The user program issues a CALLx instruction to the vector area of system virtual address space. A CHMK or CHME instruction transfers control to a change mode dispatcher which builds a call frame and then executes a CASE instruction to dispatch to the service specific procedure.

When that procedure completes its operations, it executes an RET instruction which returns control to a routine SRVEXIT. Because no error occurred (as assumed), an REI instruction is executed to pass control back to the vector area where another RET instruction returns control to the user program.

ANSWER SHEET

b. The CHMK and CHME instructions cause corresponding exceptions which push a PSL and PC pair plus a service code used in dispatching and change access mode to the required mode. The exit from the exception service routine must be an REI instruction to restore the previous access mode and reset the PC and PSL.

c. The caller's access mode can be obtained from either the previous mode field from the current PSL or from the current mode field of the saved PSL.

   Because the saved PSL may be at an unspecified offset from the top of the stack, the previous mode field of the current PSL is simply compared to the access mode passed as an argument to the system service. The larger (less privileged) access mode is the one used by the system service.

   This operation is performed to insure that a nonprivileged image does not gain access rights by, for example, queuing an executive or kernel mode AST to itself.

5. LIB$SIGNAL may be invoked by any code on detection of an error that is to be treated as an exception. Software makes the decision.

   The exception dispatcher is entered as a result of hardware exceptions and a small set of software exceptions.

   LIB$SIGNAL, through its alternate entry point LIB$STOP, can force an image to exit. The exception dispatcher has no such feature, although a condition handler could issue a $EXIT system service.

TEST/EXERCISE

1. Which debugger would you use under the following conditions?

   a. Examine the current system

   b. Examine a crash dump

   c. Debug a user mode image at IPL Ø

   d. Debug a driver

2. Which is NOT a reason for a crash dump to occur?

   a. Exception at elevated IPL

   b. User mode image error

   c. Machine check in kernel mode

TEST/EXERCISE

3. Use the available executive listings and SYS.MAP to answer the following questions about the $SUSPND system service and AST delivery.

   $SUSPND System Service

   a. Which module contains the code that implements the $SUSPND system service? (Remember that all system services have two entry points, one of the form SYS$name which is the starting address of the vector entry, and one of the form EXE$name which is the starting point of the actual code.)

   b. What other routines are defined in this module?

   c. How long (in bytes) is this module?

   d. Which system mechanism is used to suspend a process?

TEST/EXERCISE

e. List all of the system subroutines that are called by the $SUSPND system service.

f. A process can suspend another process only if it is in the same group and the issuing process has GROUP privilege, or if the issuing process has WORLD privilege. Where in the code is this check made? What other system services need to make this check?

g. The $HIBER system service does not make the same UIC and privilege check that $SUSPND does (see question f.). Why?

h. Why does the $SUSPND system service set IPL to zero in line 125?

AST Delivery

i. What line of the $SUSPND system service actually queues the AST?

TEST/EXERCISE

j. What section of code in the routine SCH$NEWLVL computes the ASTLVL value and stores the value in the hardware PCB and ASTLVL processor register?

k. Assume that the current process is issuing a $SUSPND for itself, and that it will be able to complete the $SUSPND system service without interruption. At what point in the system service dispatching sequence will the AST delivery code (the IPL 2 interrupt service routine) be entered? (This is the code that will transfer control to the AST routine.)

l. Once the AST delivery code has been entered and the special kernel mode AST routine begins to execute, something must happen to change the state of the current process (since a process cannot continue to execute after a $SUSPND). What line of code in the $SUSPND special kernel mode AST routine (page 5 of the module SYSPCNTRL) causes the current process to change states and give up the CPU? How is this operation accomplished?

ANSWER SHEET

1.

a. To examine the current system, use the System Dump Analyzer.

b. To examine a crash dump, use the System Dump Analyzer.

c. The symbolic debugger is used to debug user mode images at IPL Ø. For other access modes at IPL Ø, use the DELTA debugger.

d. Use XDELTA to debug a driver, which operates at elevated IPL in kernel access mode.

2. A user mode image error will not cause a crash dump to occur. What will occur is a traceback, and any condition handling that has been set up.

ANSWER SHEET

3.

$SUSPND System Service

a. SYSPCNTRL is the module which defines the symbol EXE$SUSPND. See page 34 of SYS.MAP.

b. There are two ways to find the routines defined in SYSPCNTRL. The easiest way is to look at the table of contents of the SYSPCNTRL module listing. This lists all the entry points:

EXE$SUSPND	EXE$WAKE
EXE$RESUME	EXE$NAMPID
EXE$HIBER	EXE$SETPRN

Another way to answer this question is to first find the PSECT in which the SYSPCNTRL module resides. This is accomplished by searching sequentially through the PSECTs starting on page 5 of SYS.MAP until SYSPCNTRL is found. Ignore any reference that shows identical base and end virtual addresses such as the one under the .BLANK. PSECT on page 9. SYSPCNTRL appears on page 10 under the AEXENONPAGED PSECT with a base of 80009FF0 and an end of 8000A1BC. Note that the length of 1CD also appears here, which answers question c. as well. Any routines defined by SYSPCNTRL must have entry points that fall between the base and end addresses.

All symbols are listed in numerical order beginning on page 62 of SYS.MAP. On page 89 you will find the following entry points:

80009FF0	EXE$SUSPND
8000A046	EXE$RESUME
8000A05B	EXE$HIBER
8000A06F	EXE$WAKE
8000A080	EXE$NAMPID
8000A14B	EXE$SETPRN

c. The length of the module is 1CD bytes hexadecimal or 461 bytes decimal. This can be found on page 10 of SYS.MAP as described in question b, or by looking at the last line of code in the SYSPCNTRL module (which occurs on page 14 of the SYSPCNTRL listing).

ANSWER SHEET

d. The system suspends a process by queuing a special kernel mode AST to the target process, as mentioned in the comments on page 3 of SYSPCNTRL (under Functional Description).

e. The following system subroutines are used:

Line referenced	Subroutine name
124	EXE$NAMPID
129	EXE$ALLOCIRP
136	SCH$QAST

These subroutines are invoked on pages 3 and 4 of SYSPCNTRL.

f. The UIC and privilege check is made in the EXE$NAMPID routine, which begins on page 1Ø of the SYSPCNTRL module. The actual check occurs in line 439 for group privilege and line 438 for world privilege.

The other system services that need to make this check are:

$DELPRC	$SCHDWK
$RESUME	$FORCEX
$WAKE	$SETPRI
$CANWAK	$GETJPI

Probably the best way to find this answer is to look in the table summarizing the system services in Chapter One of the VAX/VMS System Services Reference Manual. However, most of these services can be deduced from the names of the modules which reference EXE$NAMPID, found on page 33 of SYS.MAP:

SYSPCNTRL	SYSFORCEX
$SUSPND	$FORCEX
$RESUME	SYSGETJPI
$WAKE	$GETJPI
SYSCANEVT	SYSSCHEVT
$CANWAK	$SCHDWK
SYSDELPRC	SYSSETPRI
$DELPRC	$SETPRI

To verify the check in each case, locate the call to EXE$NAMPID in the code for each service. (Merely understanding the process and perhaps doing it in the case of the SYSPCNTRL module, is sufficient for this exercise.)

ANSWER SHEET

g. $HIBER makes no privilege check because a process is only allowed to hibernate itself (not others), although it can be awakened by other processes. This is not mentioned explicitly in the code comments, but could perhaps be deduced from the absence of the privilege check or from the fact that the $HIBER system service does not have any arguments.

h. $SUSPND sets IPL to zero after the call to EXE$NAMPID because that routine raises IPL to synchronization level to access the scheduler's data base and does **not** lower it. This is emphasized in the comments under OUTPUT PARAMETERS for this subroutine (line 360 on page 10 of SYSPCNTRL). IPL is actually raised on line 425, page 11.

AST Delivery

i. Line 136 of SYSPCNTRL invokes SCH$QAST to actually queue the special kernel mode AST to the target process. The routine SCH$QAST is located in the module ASTDEL, as indicated on page 54 of SYS.MAP.

j. Lines 603-625 of module ASTDEL calculate the ASTLVL value and store it. Line 613 extracts the access mode of the first AST in the queue. Line 618 stores the ASTLVL value in the hardware PCB field, while line 619 performs the same operation for the ASTLVL processor register.

k. The AST delivery mechanism begins with an REI instruction detecting the deliverability of an AST and causing a software interrupt at IPL 2. If the process is **not** interrupted between the queuing of the AST in SCH$QAST and the REI instruction in the SRVEXIT routine, then the first REI instruction encountered will be that one. As a special kernel mode AST, the suspend AST will be delivered next regardless of the previous access mode and any other ASTs in the queue of the process.

ANSWER SHEET

1. The state change in the process is not performed in the SYSPCNTRL module code. The actual location is indicated by line 174 in SYSPCNTRL, a branch to the system subroutine SCH$WAITK. This routine is located in the module SYSWAIT (use SYS.MAP again) beginning on line 311, page 9. SCH$WAITK changes the state field in the software PCB (line 314), inserts the PCB into the appropriate wait queue (line 313), saves the hardware context of the current process (line 320), and jumps to the scheduler (line 327). Note that an IPL 3 software interrupt is not requested, because there no longer is a current process to place into a computable wait queue (see also the "Scheduling" module of this course).

TEST/EXERCISE

1. For each state described below, briefly discuss the properties of a process in the state (for example, memory-resident, or executable), what event or system service placed the process in the state, what system events must occur before the process can leave the state, and what the next process state can be.

   a. CUR

   b. HIB

   c. SUSPO

   d. CEF

   e. COLPG

   f. PFW

   g. COMO

TEST/EXERCISE

2. Assuming the same initial conditions (stated below) for each question, state

   - what happens to the currently executing process,
   - which process is next selected for execution, and
   - at what software priority that process executes.

   Initial Conditions:

Process Name	Software Priority	Process State
A	5	COM
B	7	LEF
C	17	HIB
D	5	CUR

   a. System event: quantum end for Process D.

   b. System event: post event flag (terminal output completed) for Process B.

   c. System event: scheduled wakeup (from software timer) for Process C.

TEST/EXERCISE

3. Describe, for each of the general process categories listed below, how processes in the category may be included in multiprocess applications. Indicate any possible interactions with system processes that must be considered in assigning processes to these categories and the expected execution behavior of processes in the category.

   a. time-critical processes

   b. normal processes with elevated base priorities

   c. normal processes with normal (default) base priorities

   d. normal processes with lowered base priorities

ANSWER SHEET

1.

a. CUR -- The process is the current executing process. It is memory-resident. The state is only entered from the computable, memory-resident state (COM) as a result of a scheduling operation. A process leaves the CUR state as a result of quantum end, process deletion, a wait condition, or preemption by a higher priority COM process.

b. HIB -- The process is memory-resident, but not computable. The hibernate state is entered by issuing a request to the $HIBER system service (from the CUR state) or requesting the action as part of a create process request ($CREPRC). A process outswapped while hibernating is placed in the HIBO wait state. A process can be made computable (COM) by receiving an AST, a $WAKE request, or a process deletion request.

c. SUSPO -- The process is neither memory-resident nor computable. The state is entered from the CUR state as a result of a $SUSPND system service request, followed at some point, by an outswap operation. A process leaves this state only after a $RESUME system service request issued by another process, or as a result of a process deletion request. In each case, the process is next placed in the appropriate COMO queue.

d. CEF -- The process is waiting for one or more event flags in a common event flag cluster. Memory-resident and outswapped CEF processes share the same wait state queue (for a particular common event flag cluster). When the combination of event flags is satisfied, the process is placed into either the computable, resident (COM) or computable, outswapped (COMO) state depending on the memory-resident status bit in the software PCB. The process can also be made computable as a result of AST delivery and process deletion.

ANSWER SHEET

e. COLPG -- The process referenced a page already being read into memory as a result of other activity in the system. When the page is available, the process will be made computable or computable outswapped, depending upon its memory residence status when the page becomes available. AST delivery and process deletion also make COLPG processes computable.

f. PFW -- The process is waiting for a paging operation (page read I/O) to complete. When the page becomes available, the process enters the COM or COMO state, depending upon the memory residence status. A PFW process can also be made computable as a result of either AST delivery or process deletion.

g. COMO -- The process is computable but not resident in memory. The state may be entered from the various outswapped wait states after any of the system events that make such a process computable. The COMO state is also the initial state of a newly created process. The only transition is to the computable, resident (COM) state, the event for which the process is waiting.

2.

a. Process D will be rescheduled into the tail of the priority 5 COM state queue. Process A will be scheduled by removing it from the head of the priority 5 state queue and executing it at priority 4.

b. Process D will be rescheduled as in answer a. The event flag service will make Process B computable at priority 11 (after the terminal input boost is applied). The scheduler brings Process B into execution at priority 10.

c. Process D will be rescheduled as in answer a. Awakening Process C makes it computable at priority 17, and it will be scheduled at priority 17.

ANSWER SHEET

3.

a. Time-critical processes are useful for traditional real-time applications. They are characterized by fast response times, fixed execution priorities, and invulnerability to quantum end events. For predictable scheduling, time-critical processes should be assigned unique priorities. Otherwise, there is a potential for round robin scheduling of computable real-time processes. In addition, these processes should disable swapping to prevent scheduling conflicts with the swapper, a time-critical process at priority 16.

b. Normal processes with elevated base priorities are characterized by fast response times, but they are susceptible to quantum end events, including working set adjustment and CPU time expiration. As the base priority approaches 15, the current priority level tends to remain more constant than for default processes. Normally, interaction with the system processes (which are mostly implemented as processes of this type) is not a serious concern, because their normal process states are either HIB or LEF.

c. Normal processes with default or normal base priorities typically represent the majority of the processes on a system. The full range of scheduling-related operations apply -- round robin scheduling, dynamic priority recomputation, and quantum end (with working set adjustment and CPU time limit checking). Interactive processes in this category tend to be favored over compute-bound processes because of the priority boost mechanism.

d. Normal processes with lowered base priorities are background processes. On a busy system, these processes will only experience occasional scheduling. This category, if used at all, is typically reserved for batch streams, where response time is less critical.

TEST/EXERCISE

1. What is the page replacement algorithm used by the pager? Explain how this algorithm combines with the page cache (free and modified page lists) to effectively implement a least recently used algorithm.

2. What limitation is imposed on the size of a page file by memory management data structures? Is this a restrictive limitation at the present time?

TEST/EXERCISE

3. The VAX/VMS operating system uses page table entries of invalid pages to locate those pages in secondary storage. In view of the sequence of steps followed in address translation, what must be the contents of PTE<30:27>? Why?

4. A common programming error involves inadvertently transferring control to location 0. This problem is neatly caught in VAX native images using the memory management protection mechanism. What is the mechanism used? How could the programmer distinguish this exception from other problems?

TEST/EXERCISE

5. In translating a process virtual address, two address translations are potentially involved, and thus, two distinct translation-not-valid faults could occur.

   a. What are the translations?

   b. Why are two translations not always required?

   c. How could one distinguish the two faults?

   d. What is the difference between the state of the stack in the two cases?

6. Explain how the VAX hardware uses the modify bit in the page table entry.

7. State one instance when the VAX/VMS operating system must invalidate a single entry in the translation buffer.

TEST/EXERCISE

Questions 8 through 12 represent a sequence of operations involving the interactions among three user processes and VAX/VMS. The processes have the following initial characteristics:

Process Name	Software Priority	Scheduling State
LOW	4	CUR
MEDIUM	1Ø	LEF
HIGH	15	LEF

8. Process LOW causes a page fault in referencing a page in VAX-11 RMS (a mapped system section in the system region). The corresponding page table entry (PTE) points to the image file (SYS$SYSTEM:RMS.EXE).

   a. What is the action of the pager?

   b. Into what scheduling state is process LOW placed?

   c. Into what page state is the physical page (PFN database entry) placed?

9. While the paging operation is in progress, Process HIGH becomes computable and also makes a reference to the same page in RMS as Process LOW referenced.

   a. Into what scheduling state is process HIGH placed?

   b. What page state is the physical page (PFN database entry) in now?

TEST/EXERCISE

10. While the paging operation continues, process MEDIUM also becomes computable and also refers to the same RMS page as processes HIGH and LOW.

    a. Into what scheduling state is process MEDIUM placed?

    b. What page state is the physical page (PFN database entry) in now?

11. The paging read operation completes. Further processing is performed at IPL 4 by the I/O post processing routine.

    a. Into what scheduling state is process LOW placed?

    b. Into what scheduling state is process MEDIUM placed?

    c. Into what scheduling state is process HIGH placed?

    d. Into what page state is the physical page (PFN database entry) placed?

TEST/EXERCISE

12. IOPOST completes its processing and dismisses the IPL 4 interrupt. A scheduling interrupt (IPL 3) occurs as a result of the IOPOST operations.

    a. Which of the three processes will be scheduled first?

    b. Why is this process selected for execution?

13. Several components and utilities of VAX/VMS are required to cooperate in the implementation of shared sections.

    a. How does this feature contribute to reducing the consumption of disk storage and physical memory?

    b. Three different forms of installation of a shareable image file affect the amount of time necessary to begin use of the image:

```
INSTALL
INSTALL/OPEN
INSTALL/OPEN/HEADER
```

    What does each of these options do to affect the startup time in using shareable images?

TEST/EXERCISE

c. A shareable image requires the use of the global page table and the global section table to resolve page faults within the image. What does this fact imply about the speed of an individual page fault resolution within a global section? What is the implication of page fault resolution considering all of the processes on the system? Why?

14. Using the System Dump Analyzer (optional)

a. Using a copy of the system page table obtained via SDA, construct a map of the actual placement in physical memory of the components of the permanently resident portion of the executive. These include:

the system page table itself
the PFN database
the system header
the nonpaged executive code and data
the interrupt stack
non-paged dynamic memory

HINT

In the Internals and Data Structures manual table detailing the layout of system virtual address space, the memory access codes for these components are given. These can be used to identify which pages in the SPT are associated with each component. You might find it easiest to work from the end of the SDA listing of the system page table. The components listed in the table are in the order that they will appear in the SPT. The actual page frame for each page is also listed in the SPT.

TEST/EXERCISE

b. Using a copy of the PFN database obtained by using SDA, determine how many pages of physical memory are available for paging. Determine how much memory must be used by the permanently resident executive. Go back to the system page table and figure out how many pages are required by each component from question a above, and add the values together. Does this agree with the value (computed above) from the PFN database? It should.

ANSWER SHEET

1. The pager replaces the oldest page in the process working set. The process working set list is a circular buffer, with a single pointer advancing to the next replacement candidate.

   The contents of the physical page are not discarded when the page is removed from the working set. Rather, the physical page is placed on either the free page list or the modified page list. If a page fault occurs while the page is on either of these lists, the pager simply removes the page from the list and puts it back into the process working set.

   Virtual pages that are frequently referenced will occasionally be removed from the process working set. However, it is highly likely that the page will still be on one of the lists when a subsequent page fault occurs.

2. The page file control block imposes no limitation on the size of the page file. The form of page table entry that indicates that a virtual page is in the page file allows 22 bits for virtual block number. This requires that the page file be less than four Megablocks. Because disks do not normally exceed about one Megablock, the maximum size of a single page file is much larger than the disks available. No limitation is currently imposed by the data structures.

3. PTE<30:27> must contain a protection code, even for invalid pages. Because the access check is performed before the valid bit is tested, the PTE for each page in process or system virtual space (specified by the contents of the appropriate region length register) must contain a protection code in these four bits.

4. The VAX linker sets up the first page of a native image as NO ACCESS for any access mode (PTE<30:27> = 0). A transfer of control to location 0 (via a CALLX, JMP, BRx, JSB, or BSBx instruction) will cause a protection code access violation.

   The top two longwords on the stack will both be zero. The reason mask is a zero and the virtual address causing the exception is also a zero. This is the key to this type of programming error.

   The third longword on the stack is the PC of the offending instruction, and the fourth longword is the PSL at the time of the exception.

ANSWER SHEET

5.

a. Both the process virtual address and the system virtual address of the corresponding PxPTE must be translated.

b. If a translation buffer hit occurs on the process page table entry, the physical address can be formed immediately.

Note that if a translation buffer hit occurs on the SPTE which maps the PxPTE, two translations are still required.

c. If the translation-not-valid fault occurs on the associated page table entry, bit 1 in the reason mask (on the top of the kernel stack) will be set. The second longword will contain the **process** virtual address in both cases.

d. The only difference in the state of the stack is bit 1 in the reason mask. The faulting virtual address is the **process** virtual address in both cases.

6. When a page is brought into a process working set, the modify bit is initially clear. Each time a write or modify access is made to a page, the modify bit is checked. If the bit is clear, it will be set by hardware both in the translation buffer and in the page table entry in physical memory.

Thus, the first write or modify access will cause the bit to be set. All subsequent accesses (until the page is removed from the working set) will have no effect on the modify bit.

The state of the modify bit will be checked when the page is removed from the working set. If the bit is set, the page must be put on the modified page list and written to secondary storage before the physical page can be reused by another process.

ANSWER SHEET

7. The most common example of invalidating a single page table entry in the translation buffer is when the page is removed from the working set. If virtual addresses are deleted from a process (as a result of $CNTREG, $DELTVA, or $DGBLSC system services or at image exit) their associated translation buffer entries must be invalidated.

   If page protection is changed by using the $SETPRT system service, the corresponding translation buffer entries are invalidated.

8.

   a. The pager

      - o determines that the page is in an image file,
      - o allocates a physical page,
      - o allocates a working set list entry (WSLE) from the system working set list,
      - o initiates the read operation, and
      - o sets the process scheduling state to page fault wait (PFW).

   b. Page fault wait state (PFW) (see question 9)

   c. Read-in-progress

9.

   a. Collided page wait state (COLPG)

   b. Read-in-progress (as in question 8) but with the collided page bit set

ANSWER SHEET

10.

a. Collided page wait state (COLPG)

b. No further change from answer 8b. The collided page bit is already set.

11.

a. Computable (COM)

b. Computable (COM)

c. Computable (COM)

d. Active and valid

12.

a. Process HIGH

b. Scheduling is based strictly upon the relative priorities of computable processes, and not upon circumstances such as which process caused the initial page fault. Thus, although process LOW caused the initial page fault, and most of the work was performed by the pager in its context, process HIGH is likely to be the first process to use the valid page as a result of its higher priority.

13.

a. Disk storage is reduced because each image file does not require a separate copy of the shared sections. Physical memory requirements are reduced because only one copy of a shared section needs to exist in the system (and only those pages of a section actually used by one or more processes occupy physical memory).

ANSWER SHEET

b. INSTALL allows a file to be opened by file ID and sequence number. This improves the speed of the OPEN operation.

INSTALL/OPEN makes the file permanently opened. There is no wait for the OPEN operation.

INSTALL/OPEN/HEADER makes the file open and the file header permanently resident. Not only is there no OPEN processing, but one less disk read operation is required during the section mapping or the image activation.

c. Although there is an additional level of indirection involved in resolving addresses within a shared image, address resolution only seems longer. With several processes referring to the section, there is a higher probability that the global page table entry (GPTE) is active and valid. If this is the case, page fault resolution is rapid. The working set list must be modified, the contents of the GPTE copied into the process PxPTE, and the share count for the physical page incremented in the PFN database.

TEST/EXERCISE

1. Describe the special treatment given to pages with direct I/O in progress both at outswap and inswap times. Be sure to include the special case of the inswap occurring before the read or write operation completes.

2. Discuss the special treatment given global pages by the swapper. Include both global read-only and global read-write pages in your discussion.

TEST/EXERCISE

3. List one advantage of implementing swapping in an operating system that already implements paging.

Questions 4 through 7 describe the interaction of two real-time processes and the swapper over an interval of time. Each question describes a particular event. For each process, indicate which process state will be occupied by that process. If a process does not exist, indicate this instead of a process state.

The initial process characteristics are:

Name	Priority	State
SWAPPER	16	HIB
LOW	2Ø	CUR
HIGH	22	not yet created

4. Process LOW issues a $CREPRC system service request and continues to execute.

   a. SWAPPER

   b. LOW

   c. HIGH

TEST/EXERCISE

5. Process LOW issues a $HIBER system service request.

   a. SWAPPER

   b. LOW

   c. HIGH

6. The inswap operation completes and is reported to the scheduler. Assume that the SWAPPER performs further operations at IPL 7 before dropping the interrupt priority level.

   a. SWAPPER

   b. LOW

   c. HIGH

7. The SWAPPER drops the interrupt priority level from IPL 7 to IPL Ø.

   a. SWAPPER

   b. LOW

   c. HIGH

ANSWER SHEET

1. For either read-in-progress or write-in-progress, the pages in question are written to the swap file with the rest of the working set. However, because the reference count will not go to zero at outswap completion if the read or write is still outstanding, the pages will not be released to the free page list.

   If the read or write is still outstanding when the process is swapped back into memory, the swapper will take this into account by putting the page left behind into the rebuilt working set of the process and releasing the page frame from the swapper's special I/O page table.

   If the operation in progress was a write, the contents of the swap file are accurate, and the page is released to the free page list when the write operation completes.

   If the operation is progress was a read, the contents of the swap file are out of date. The write of the page to the swap file merely served to reserve a place in the swap file. This block is noted in the SWPVBN array in the PFN database. When the read operation completes, the page will be released to the **modified** page list. Subsequently, the modified page writer will write this page not to the page file but to its reserved location in the swap file. (If the inswap occurs before the modified page writer writes this page to the swap file, the page is simply faulted in from the modified page list while the swapper rebuilds the working set.)

   Note that the only I/O that is relevant here is direct I/O because only direct I/O locks pages in the working set until I/O completion. Buffered I/O uses an intermediate buffer in system virtual address space (non-paged dynamic memory). Thus, buffered operations do not require the user buffer to be in memory while the request is being processed. On a buffered write, the appropriate FDT routine transfers (perhaps with modification) data from the user buffer to a system buffer. On a buffered read, the I/O completion special kernel mode AST routine transfers the data from the system buffer into the specified user buffer.

ANSWER SHEET

2. At outswap time, each global read-write page is removed from the working set of the process. Each page must be refaulted into the working set after inswap only if it is referenced after inswap.

   Each global read-only page is written to the swap file if either the PFN database SHRCNT value is one (only this process is using this page). Otherwise, the global page is removed from the working set and will need to be refaulted if it is referenced after inswap.

   At inswap time, global read-only pages are read along with the rest of the working set of the process. If the corresponding global page table entry (GPTE) is either valid or in transition, then the PxPTE points to the existing physical page, and the duplicate page is released to the free page list. If the GPTE is pointing to the global section table entry, the page is retained and both the PxPTE and GPTE are made valid.

3. The primary advantage of swapping is that it provides a way of obtaining a large amount of physical memory, an entire process working set (typically 1ØØ pages or more). This does not imply that paging is slow. In fact, paging operations are both simpler and faster than swapping operations. However, the swapper frees much larger amounts of physical memory in single operations.

   A second advantage is that implementing swapping provides further flexibility in supporting a wide range of processes with differing demands on the resources of the system. An example of this second advantage is the use of the swapper in process creation (see also the next module). Each process is brought into being in the COMO state with its initial swap image in a pseudo swap file known as the shell process. Further, the swapper's role in balancing the system-wide demands on the physical memory resource isolates these responsibilities in a single software component.

ANSWER SHEET

4.

a. COM -- process creation will awaken the swapper process.

b. CUR -- the stated assumption.

c. COMO -- the initial process state for every process.

5.

a. CUR -- the highest priority computable process.

b. HIB -- the stated assumption.

c. COMO -- same as 4c.

6.

a. CUR -- the swapper is still executing.

b. HIB -- same as 5b.

c. COM -- the purpose of the inswap operation is to make this process computable.

7.

a. COM -- dropping IPL will enable an IPL 3 scheduling interrupt to occur.

b. HIB -- same as 5b.

c. CUR -- the highest priority computable process.

TEST/EXERCISE

1. List two advantages to performing process deletion in the context of the process being deleted.

2. Name two errors that can result from process creation. One of the errors should be returned from the $CREPRC system service request and the other only through a termination mailbox. Explain why the $CREPRC system service is not capable of detecting the second type of error.

TEST/EXERCISE

3. Explain why an interactive process is not deleted when an image exits.

ANSWER SHEET

1. When executing in the context of the process being deleted, all the virtual address space of that process is accessible. In particular, the contents of the control region (P1 space) that describe the state of the process at the time of deletion is readily available.

   In addition, the full support of VAX/VMS (including RMS and all the system services) is available to aid in the process deletion. Much of this support is not available to code executing outside of process context.

2. The complete list of errors that can be detected by the $CREPRC system service is listed in the description of $CREPRC in the VAX/VMS System Service Reference Manual. Possible errors include privilege violation, insufficient quota, and process name errors.

   Several errors can be detected only when the newly created process executes. These errors include the specification of an image that does not exist or bad equivalence strings for SYS$INPUT, SYS$OUTPUT, or SYS$ERROR.

   By the time that the new process is placed into execution, the $CREPRC system service has already completed its work for the creator and returned a status code. All errors that cannot be detected **except in the context of the newly created process** can only be reported to the creator through a termination mailbox.

ANSWER SHEET

3. Image exit results in all previously declared termination handlers being called. The command language interpreter (DCL or MCR) has declared a handler that runs the image down, restores the supervisor stack to its state before the image was initially called, and looks for the next command from SYS$INPUT. This allows multiple images to execute sequentially in the same process. Only a special action, such as a LOGOUT command within the process, an external STOP/ID= command, can cause such a process to be deleted.

TEST/EXERCISE

Differentiate the two programs SYSBOOT and SYSGEN, including their

- purposes,
- environments, and
- command syntax.

ANSWER SHEET

SYSBOOT

- **Purpose:** SYSBOOT is the program that performs the secondary phase of the bootstrap sequence. It reads parameters from the system image and, optionally, from a parameter file. All adjustable parameters are calculated. The system page table is set up. The system image is read into memory.

  SYSBOOT is **not** involved in determining which devices are present or in loading the drivers and associated data structures for these devices

- **Environment:** SYSBOOT executes in a standalone environment with memory management turned off. All communication with the console terminal and all file operations must be performed by code contained in the SYSBOOT image, because there is no RMS or ACP to provide these services.

- **Command Syntax:** SYSBOOT does **not** recognize those commands associated with loading device drivers. The WRITE command is also ignored by SYSBOOT.

  SYSBOOT begins its operation by reading the values of adjustable parameters from the system image file. This is an implied USE CURRENT command.

SYSGEN

- **Purpose:** SYSGEN is not directly involved in the bootstrap operation. Its primary purpose is to create a parameter file that will be used by SYSBOOT during future bootstrap operations.

  SYSGEN also loads device drivers for all devices that it finds on the system or in response to explicit commands. The data structures required by the driver are allocated and initialized by SYSGEN.

- **Environment:** SYSGEN is a normal image that executes in full process context. This means that services of the VAX/VMS operating system are available for file operations including terminal communication.

ANSWER SHEET

- **Command Syntax:** All commands can be performed by SYSGEN. However, SET commands do **not** normally affect the current system, but merely change the values in a table that will be written to a parameter file. A WRITE CURRENT command will establish the parameter values used in the **next** system initialization. A WRITE ACTIVE command can change the values of dynamic system parameters on the running system.