1. Which of the following instructions should be privileged(in kernel mode)?

a. Set value of timer.
b. Read the clock.
c. Clear memory.
d. Issue a trap instruction.
e. Turn off interrupts.
f. Modify entries in device-status table.
g. Switch from user to kernel mode.
h. Access I/O device.

Answer:

The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. (a,c,e,f,h)

The rest can be performed in user mode.(b,d,g)

2. Name three ways in which the processor can transition form user mode to kernel mode?

Answer:

1) The user process can execute a trap instruction (e.g. system call). A trap is known asa synchronous software interrupt.
2) The user process can cause an exception (divide by zero, access bad address, bad instruction, page fault, etc).
3) The processor can transition into kernel mode when receiving an interrupt.

3. **(2.3)**Describe three general methods for passing parameters to the operating system.

Answer:

a. Pass parameters in registers
b. Registers pass starting addresses of blocks of parameters
c. Parameters can be placed, or pushed, onto the stack by the program, and popped off the stack by the operating system.

4. What is the difference between kernel mode and user mode? Why is the difference important to an operating system?

Answer:

In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address .

In User mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory.

The difference rather protect the computer system resources, while preventing from errant users.

5. What are exceptions? Name two different types of exceptions and give an example of each type.

Answer:

Exceptions are events that stop normal execution, switch the execution mode into kernel mode, and begin execution at special locations within the kernel. Examples include

system calls, divide by zero errors, illegal instructions, and page faults.