Operating Systems - Chapter 6 Summary (Detailed)

Mechanism: Limited Direct Execution (LDE)

• Goal: Efficiently virtualize the CPU while ensuring OS control.
• Idea: Run programs directly on the CPU for speed, but under restrictions.
• Challenge: Prevent programs from monopolizing CPU or accessing privileged operations.

Restricted Operations and System Calls

• CPU provides two modes:
– User mode: restricted, normal applications run here.
– Kernel mode: privileged, OS can execute all instructions.
• Programs request privileged actions via system calls, which use the trap instruction.
• Trap instruction: saves registers, switches to kernel mode, jumps to OS handler.
• After servicing, return-from-trap restores user state and resumes program.
• Trap table: initialized at boot, maps system calls and interrupts to handlers.

Regaining Control of the CPU

• Cooperative approach: processes voluntarily yield CPU via system calls or errors.
• Limitation: buggy/malicious programs can loop forever, blocking system.
• Non-cooperative approach: hardware timer interrupts return control to OS periodically.
• Timer interrupts ensure OS can always preempt processes.

Context Switching

• A context switch is the act of saving the state of one process and restoring the state of another, allowing multiple processes to share the CPU safely.
• Triggered by: system call, timer interrupt, or I/O completion.
• Steps in a context switch:
1. Hardware saves user registers automatically (PC, flags, etc.) onto kernel stack.
2. OS saves additional kernel registers and stack pointer into process structure (PCB).
3. OS scheduler chooses next process to run.
4. OS restores saved registers, stack, and PC of the chosen process.
5. return-from-trap instruction resumes execution of the new process.
• This mechanism allows multitasking and CPU virtualization.

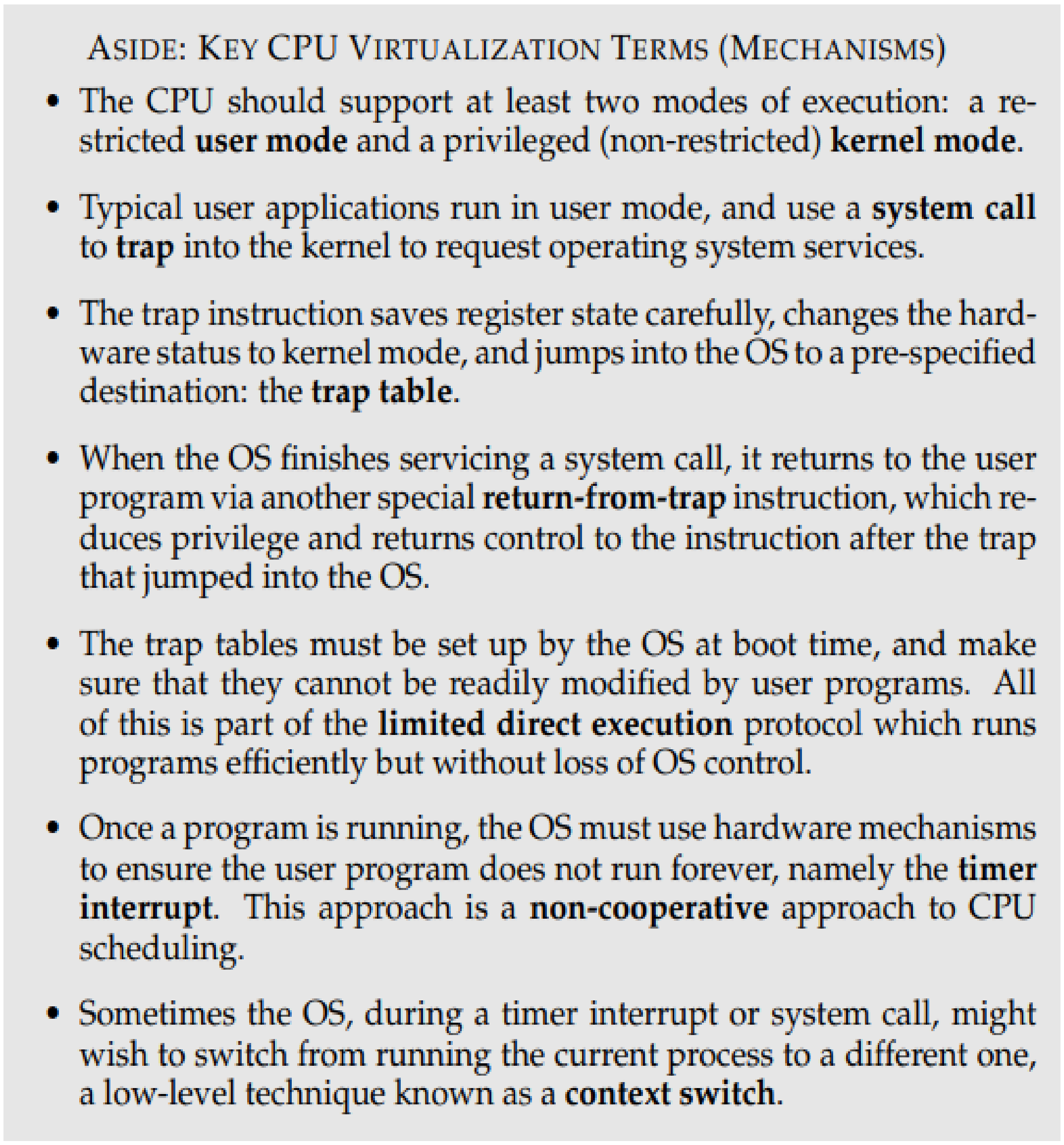
Concurrency Concerns

• Interrupts may occur during system calls or while handling another interrupt.
• OS solutions:
– Temporarily disable interrupts while handling one.
– Use locks to protect OS data structures.

Summary

Limited Direct Execution balances efficiency and control. Programs run directly on hardware but within limits enforced by the OS. Key mechanisms: user/kernel modes, traps and system calls, timer interrupts for preemption, and context switching to enable multitasking. Together, these mechanisms allow the OS to virtualize the CPU safely and efficiently.

Key CPU Virtualization Terms (Mechanisms)



ASIDE: KEY CPU VIRTUALIZATION TERMS (MECHANISMS)

- The CPU should support at least two modes of execution: a restricted **user mode** and a privileged (non-restricted) **kernel mode**.
- Typical user applications run in user mode, and use a **system call** to **trap** into the kernel to request operating system services.
- The trap instruction saves register state carefully, changes the hardware status to kernel mode, and jumps into the OS to a pre-specified destination: the **trap table**.
- When the OS finishes servicing a system call, it returns to the user program via another special **return-from-trap** instruction, which reduces privilege and returns control to the instruction after the trap that jumped into the OS.
- The trap tables must be set up by the OS at boot time, and make sure that they cannot be readily modified by user programs. All of this is part of the **limited direct execution** protocol which runs programs efficiently but without loss of OS control.
- Once a program is running, the OS must use hardware mechanisms to ensure the user program does not run forever, namely the **timer interrupt**. This approach is a **non-cooperative** approach to CPU scheduling.
- Sometimes the OS, during a timer interrupt or system call, might wish to switch from running the current process to a different one, a low-level technique known as a **context switch**.