TDIU11: Operating Systems

I. Introduction

Course organization, Interrupts, system calls

SGG10: 1.1-1.6, 2.3
SGG09: 1.1-1.9, 2.3-2.4

Ahmed Rezine, Linköping University

Course Organization

- Typically: Lectures on Mondays, seminars on Tuesdays (three groups)
  - Problem assignments (UPG1, 3hp): 5 seminars, with 8 problems each
  - Article summary (UPG2, 1hp) : 2 seminars
  - Final exam (TEN, 2hp): similar problems to those discussed in the seminars.
- You will need to:
  - Hand in two article summaries, and attend the corresponding seminars
  - Solve, and be ready to present, at least 16 problems. You will need to attend each seminar for which you claim the points.
  - Pass the exam!

Weeks with assignments

- *Announce by mail to your assistant (subject "[TDIU11] Challenge x").*
- *When announcing, send a sketch of your solutions (e.g., picture).*
- *Attend seminars for which you claim points.*
- *The solution of each assignment will be presented by a student that solved it.*
  - *Aim for 10 minutes.*
  - *Important to have a clear solution and to be convinced of it!*

Weeks with an article summary

- *Send a text document with your summary and your discussion questions to both your group supervisor and to Ahmed's Urkund address (*ahmed.rezine.liu@analys.urkund.se) with "[TDIU11] summary x" as subject.
- You will need to bring a printout of your summary
  - *(no credits without a printout!).*
- *Choose English or Swedish.*

Webreg

- Register to BOTH Webreg activities before January 23rd to get your grades:
  - *https://www.ida.liu.se/webreg3/TDIU11-2023-1/UPG1*
  - *https://www.ida.liu.se/webreg3/TDIU11-2023-1/UPG2*

Advises

- *Skim through the material before each lecture*
- Be prepared to demonstrate your solutions to the class.
- Mistakes are ok. You have however to solve all questions of a problem and to be convinced of your solution!

Plagiarism vs Cooperation

- Problem assignments: you are encouraged to cooperate. If you do not manage to present your solution, you lose all points for that set.
- Article summaries are individual. Urkund is used to check. Honest and dishonest authors will be reported to the disciplinary board in case of plagiarism. Properly cite your sources!

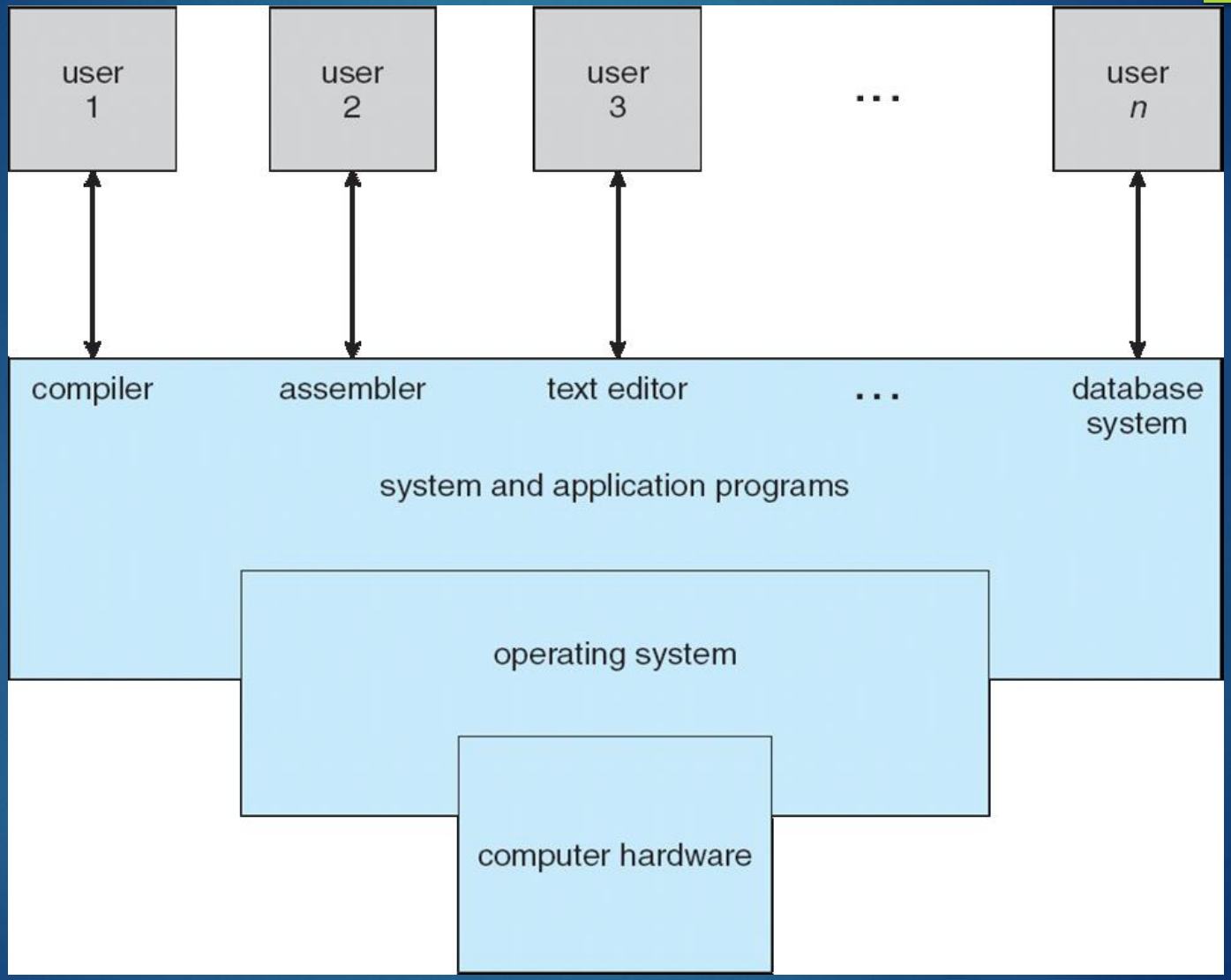
For more information

- https://www.ida.liu.se/~TDIU11/current/index.en.shtml
- https://www.ida.liu.se/~TDIU11/current/info/homework.en.shtml

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Four Components of a Computer System



What Should Operating Systems Do?

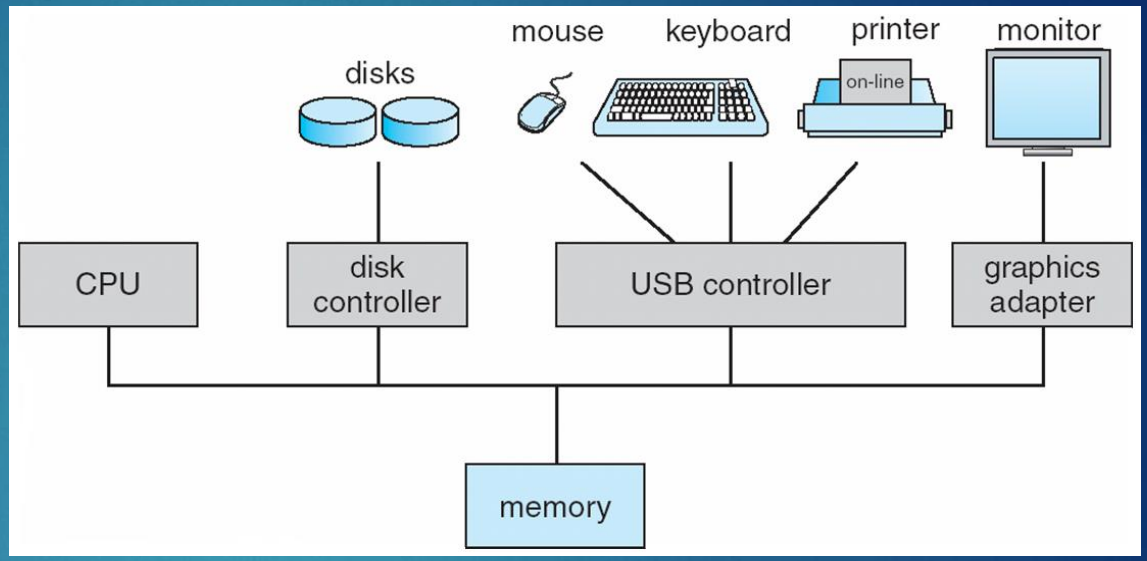
It depends …

- Users want convenience, ease of use and good performance
- Shared computers, e.g., mainframe or minicomputer must keep all users happy
- Users of dedicated systems such as workstations have dedicated resources but frequently use shared resources from servers
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

Operating System Definition

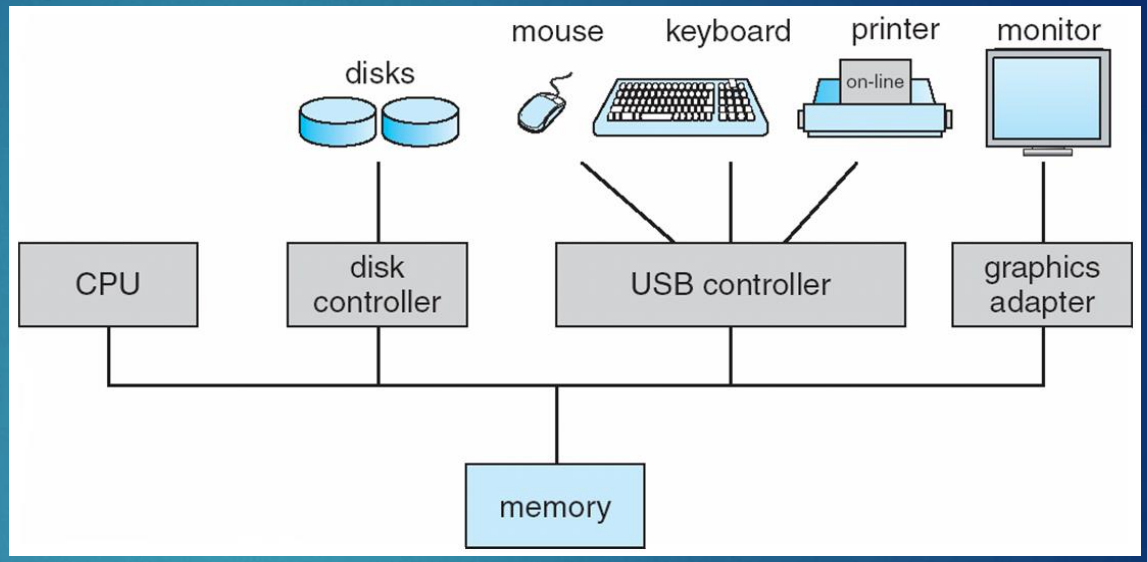
- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer

Computer System Operation



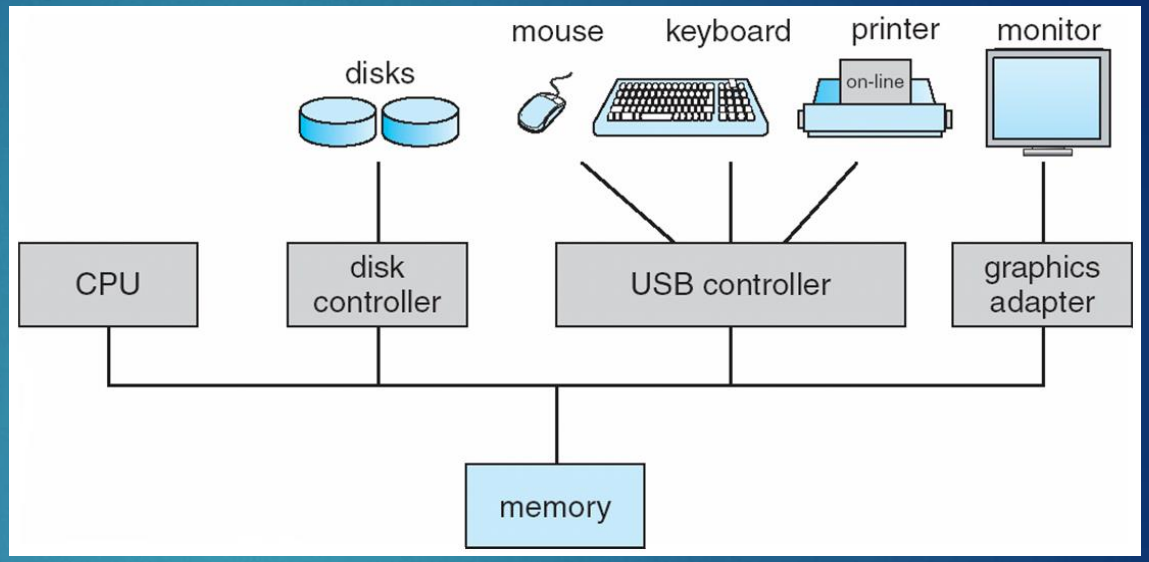
- One or more CPUs and device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles

Computer Startup



- Bootstrap program is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as firmware
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

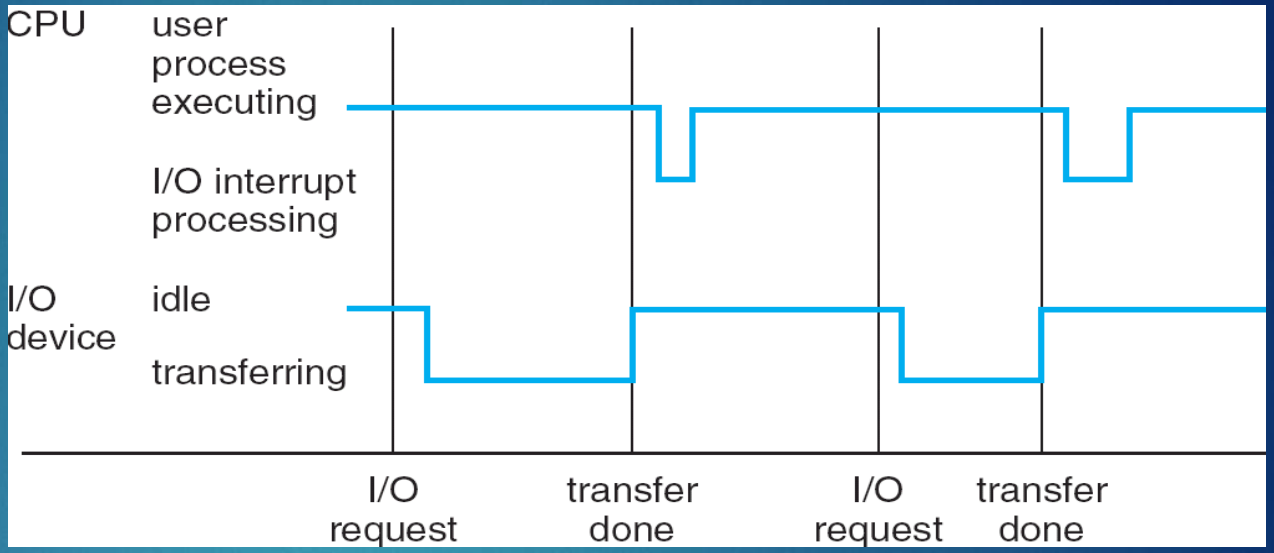
CPU – I/O Device Interaction



- I/O devices and the CPU can execute concurrently.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Remark: Alternative to interrupt usage: *Polling / Busy-waiting*, see [SGG10] Ch. 12.2.2

CPU – I/O Device Interaction (cont.)

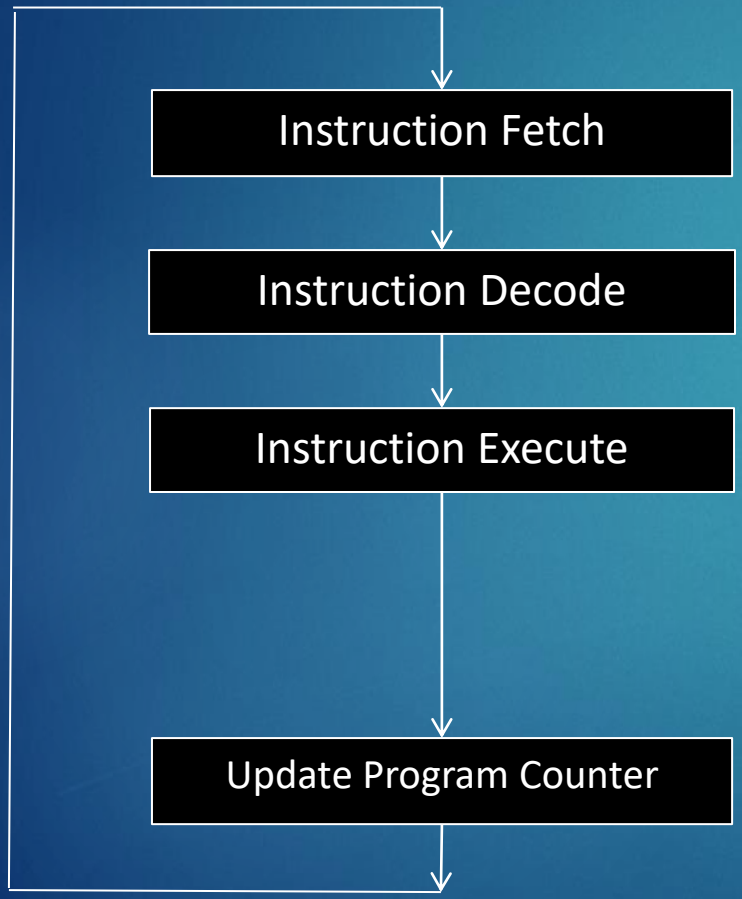


- I/O devices and the CPU can execute concurrently.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Remark:
Alternative to interrupt usage: *Polling / Busy-waiting*, see [SGG10] Ch. 12.2.2

Background: Interrupt

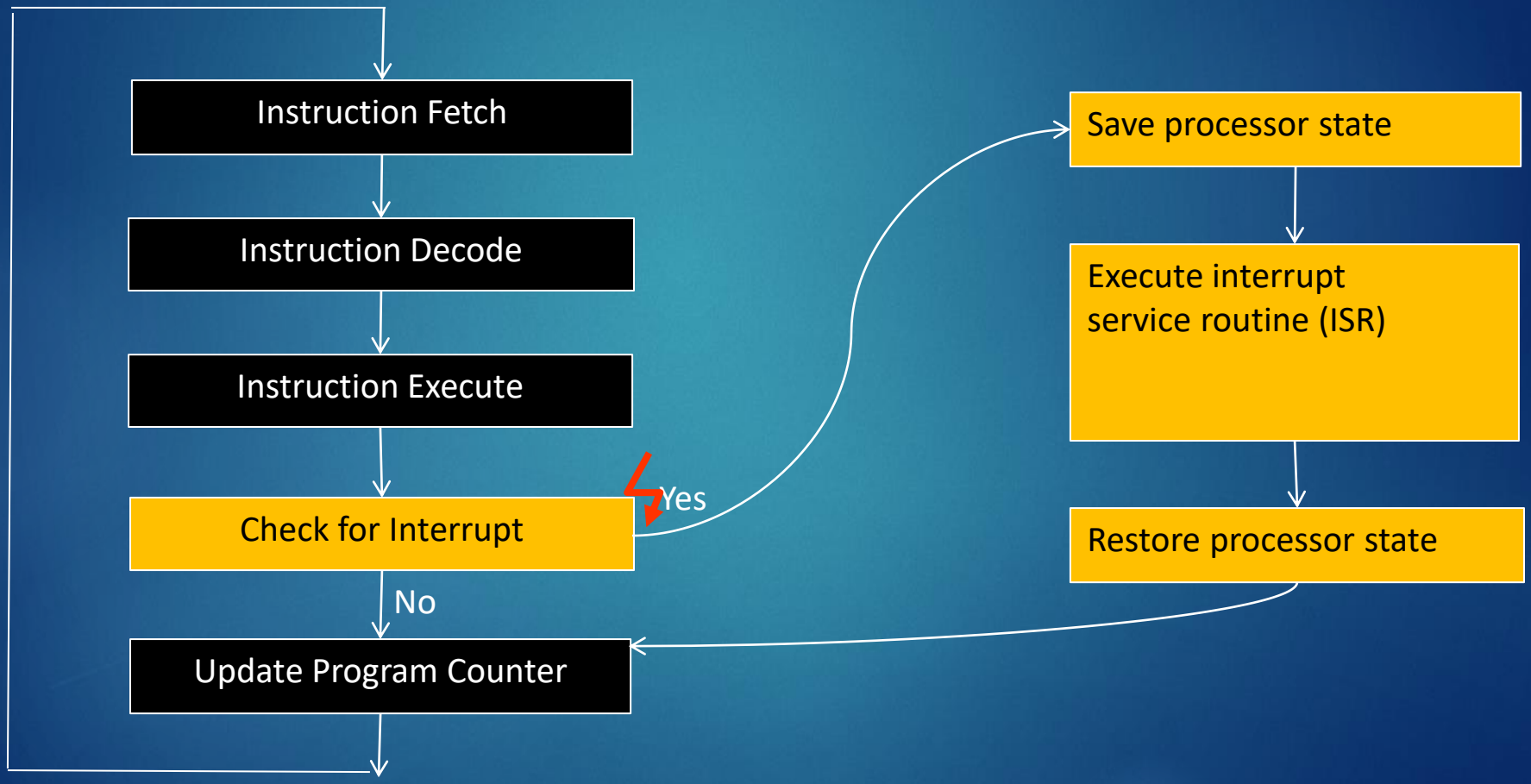
- Program execution (Von-Neumann cycle) by a processor



No way to react to events not explicitly anticipated in the (user) program code!

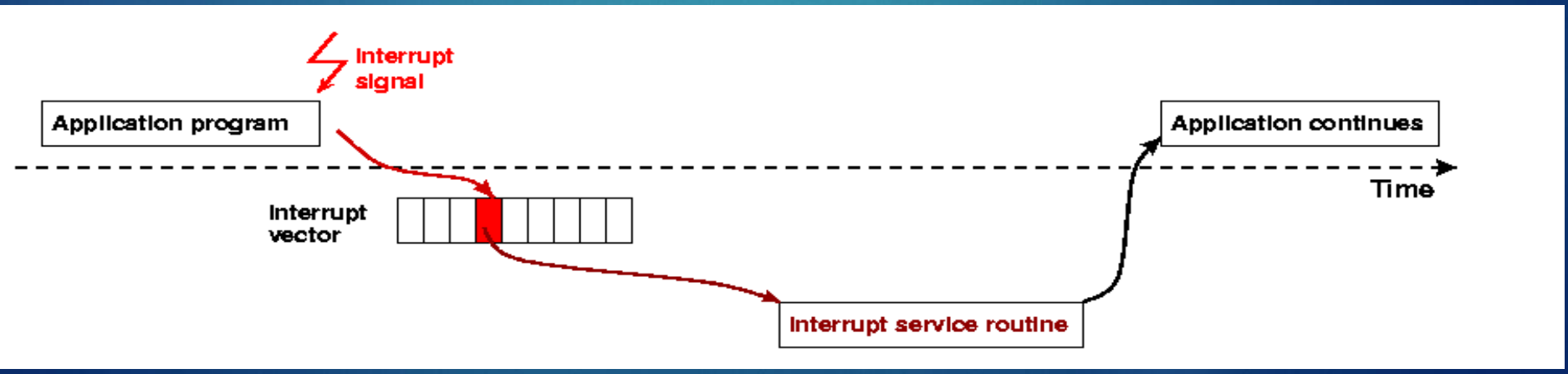
Background: Interrupt

- Program execution (Von-Neumann cycle) by a processor with interrupt logic



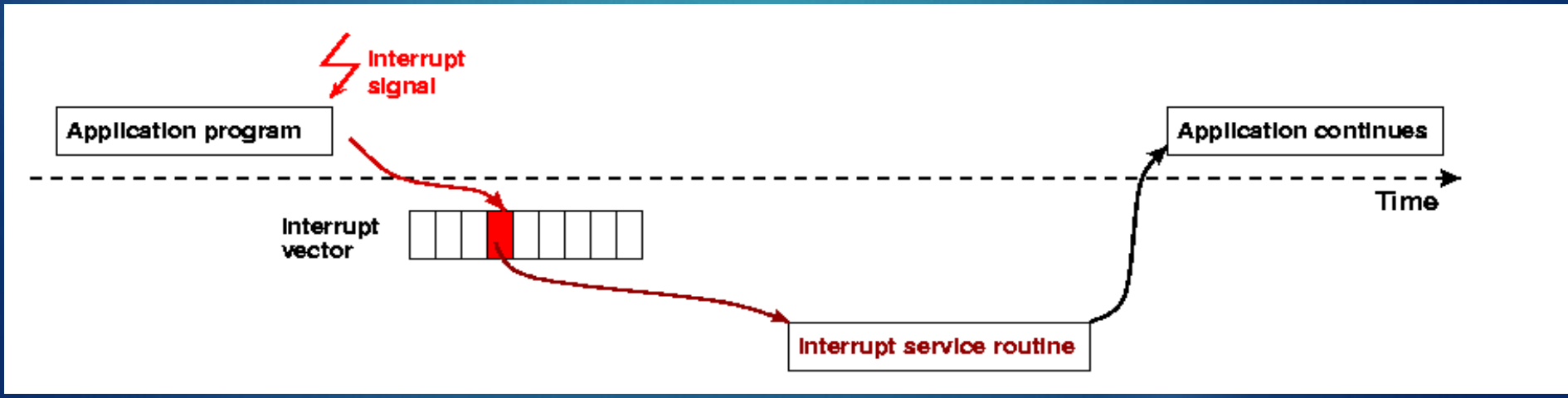
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request for an OS service
- An operating system is **interrupt driven**



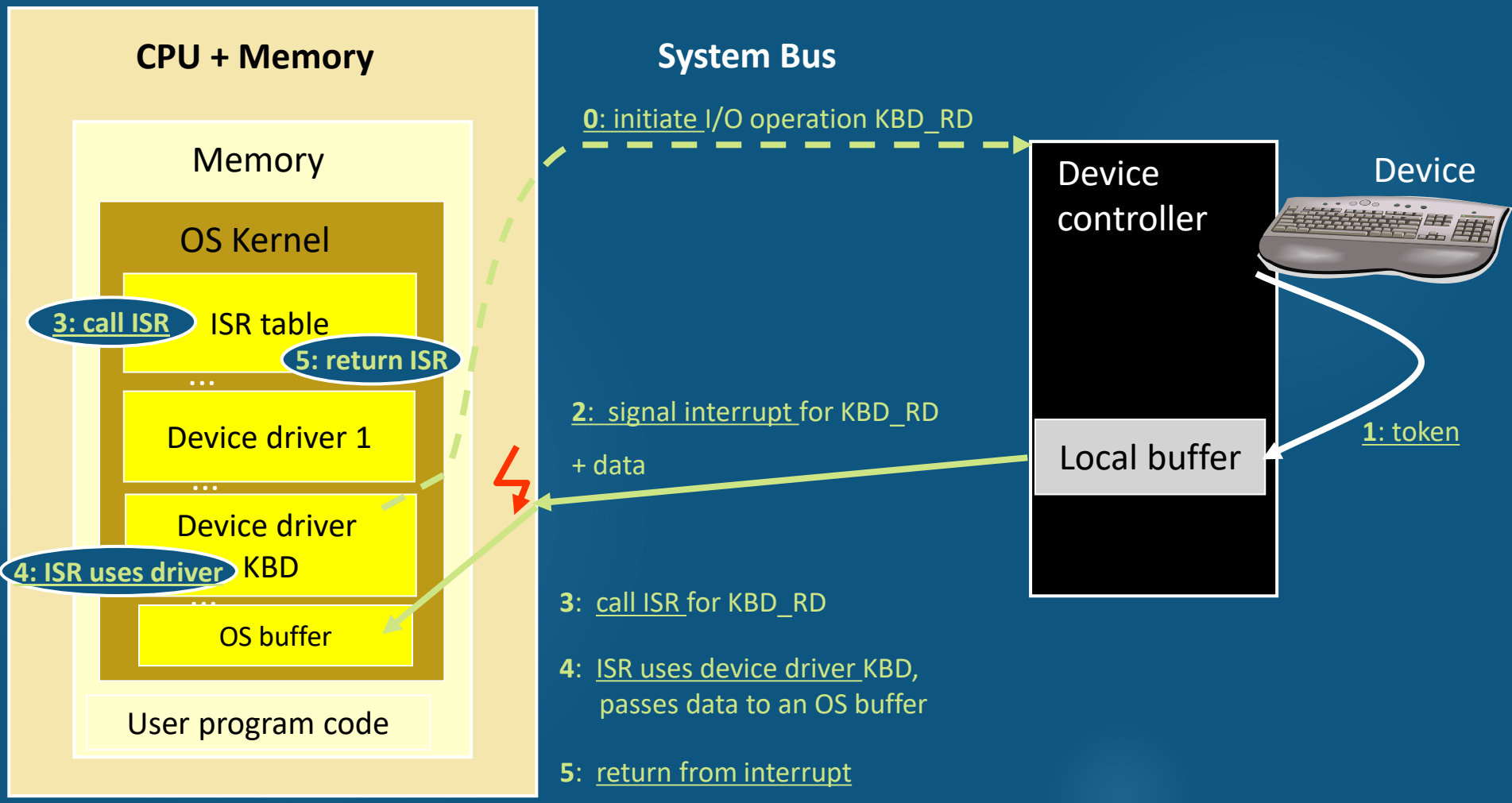
Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred: (either by **polling or via vectored** interrupt system)
- Separate segments of code determine what action should be taken for each type of interrupt



I/O Interaction using Interrupt

- **Example**: Read from the keyboard (KBD)



Interrupt driven operating systems

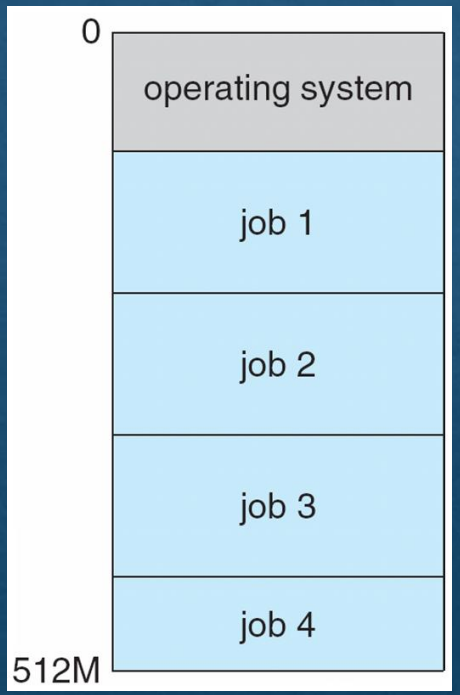
Interrupt driven (hardware and software)

- Hardware interrupt by one of the devices
- Software interrupt (exception or trap):
  - Software error (e.g., division by zero) that need to be handled (e.g., by terminating the program)
  - Request for operating system service: Important since the OS is also a program

Multiprogramming

In a **batch system** needed for efficiency

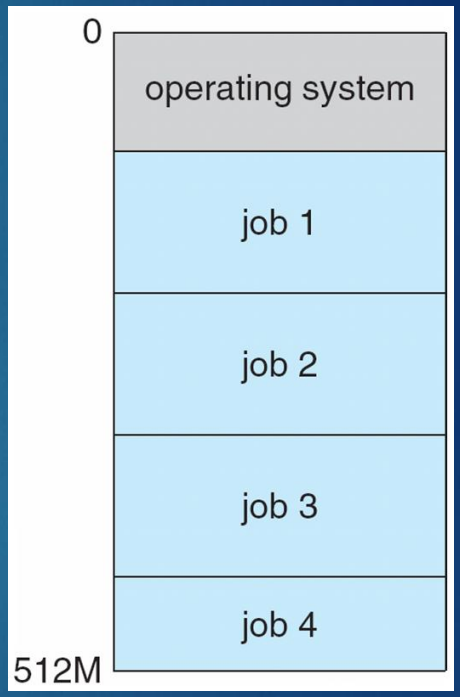
- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory. One job selected and run via **job scheduling**
- When it must wait (for I/O for example), OS switches to another job



Timesharing

Timesharing (**multitasking**) CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

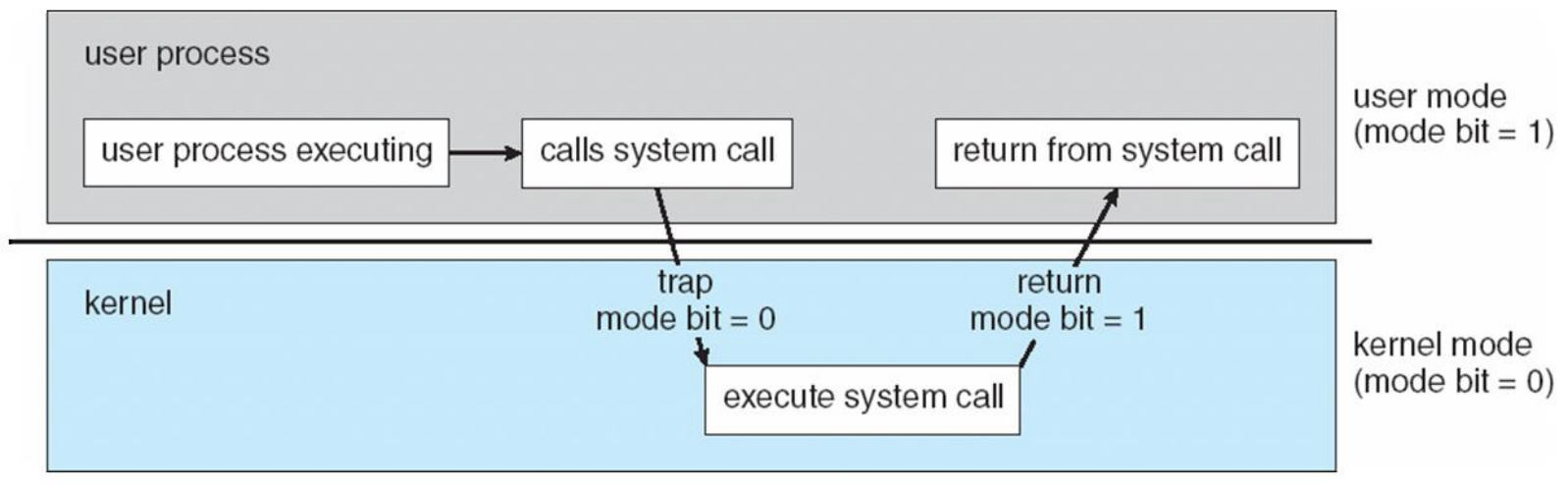
- **Response time** should be short
- Each user has at least one program executing in memory ⇨**process**
- If several jobs ready to run at the same time ⇨ **CPU scheduling**
- If processes don't fit in memory, **swapping** moves them in and out to run
- **Virtual memory** allows execution of processes not completely in memory



Operating-System Operations

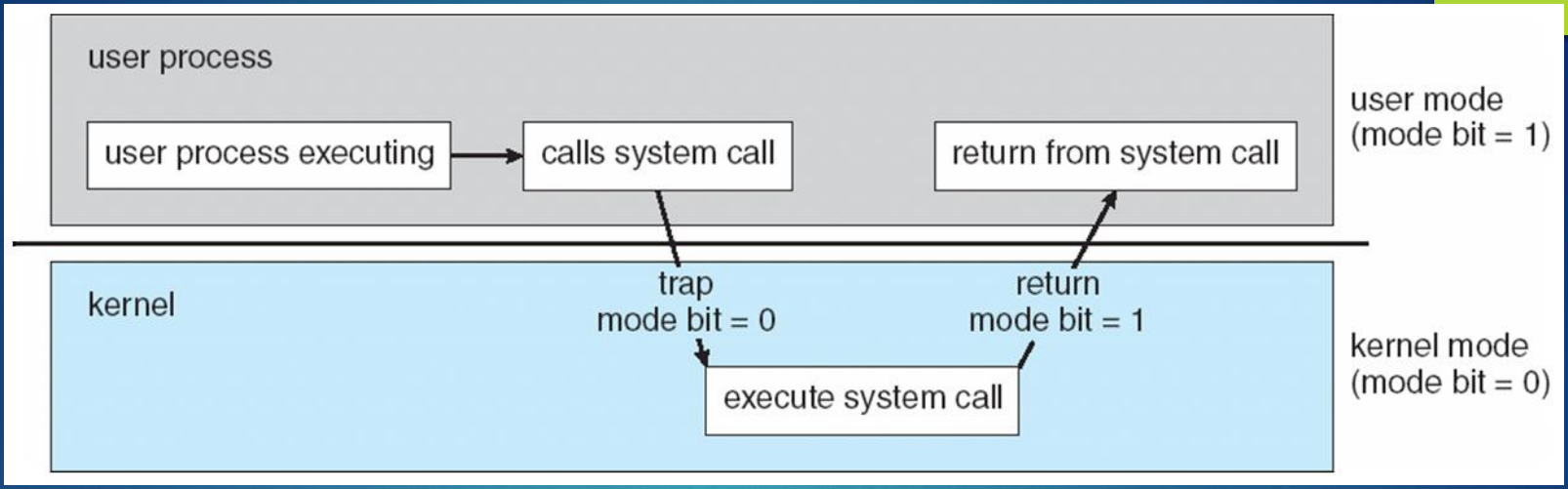
- Operating system and users share hardware and software.
- Make sure that an error in a user program does not cause problems for other programs. Examples:
  - Infinite loops,
  - processes modifying each other or the operating system
  - Forbidding access to the resources to the other users
  - …

Dual mode



- Dual-mode operation allows OS to protect itself and other system components
  - Typically: User mode and kernel mode
  - Mode bit provided by hardware

Dual mode and system calls

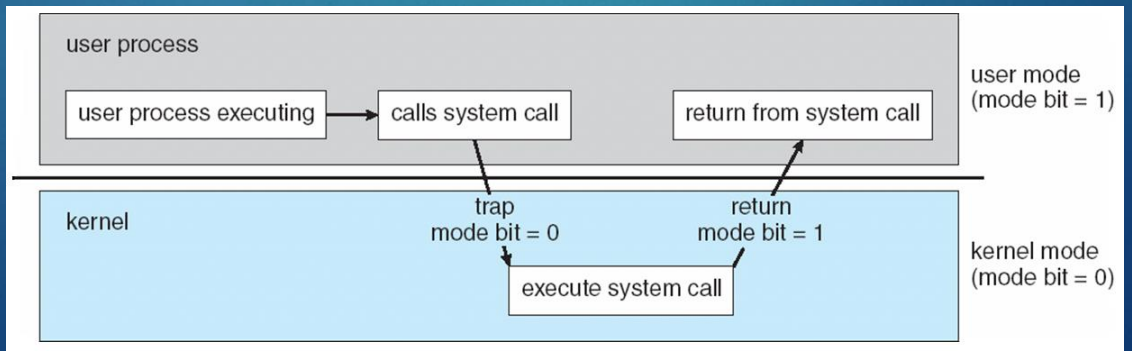


- **Mode bit** provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - **System call** changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode

Timer to prevent infinite loop / process hogging resources

- Timer is set to interrupt the computer after some time period
- Keep a counter that is decremented by the physical clock.
- Operating system set the counter (privileged instruction)
- When counter zero generate an interrupt
- Set up before scheduling process to regain control or terminate program that exceeds allotted time



Operating System Operations

- Dual mode, system calls (challenge 1)
- CPU management (challenge 1)
  - Uniprogramming, Multiprogramming, Multitasking
  - Process management
- Memory management (challenge 2)
- File system and mass storage management (challenge 3)
- Protection and security (challenge 4)