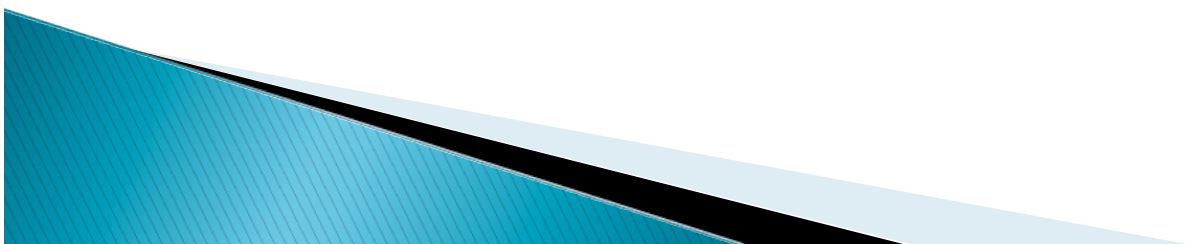
Chapter 1: Introduction

What is an Operating System?

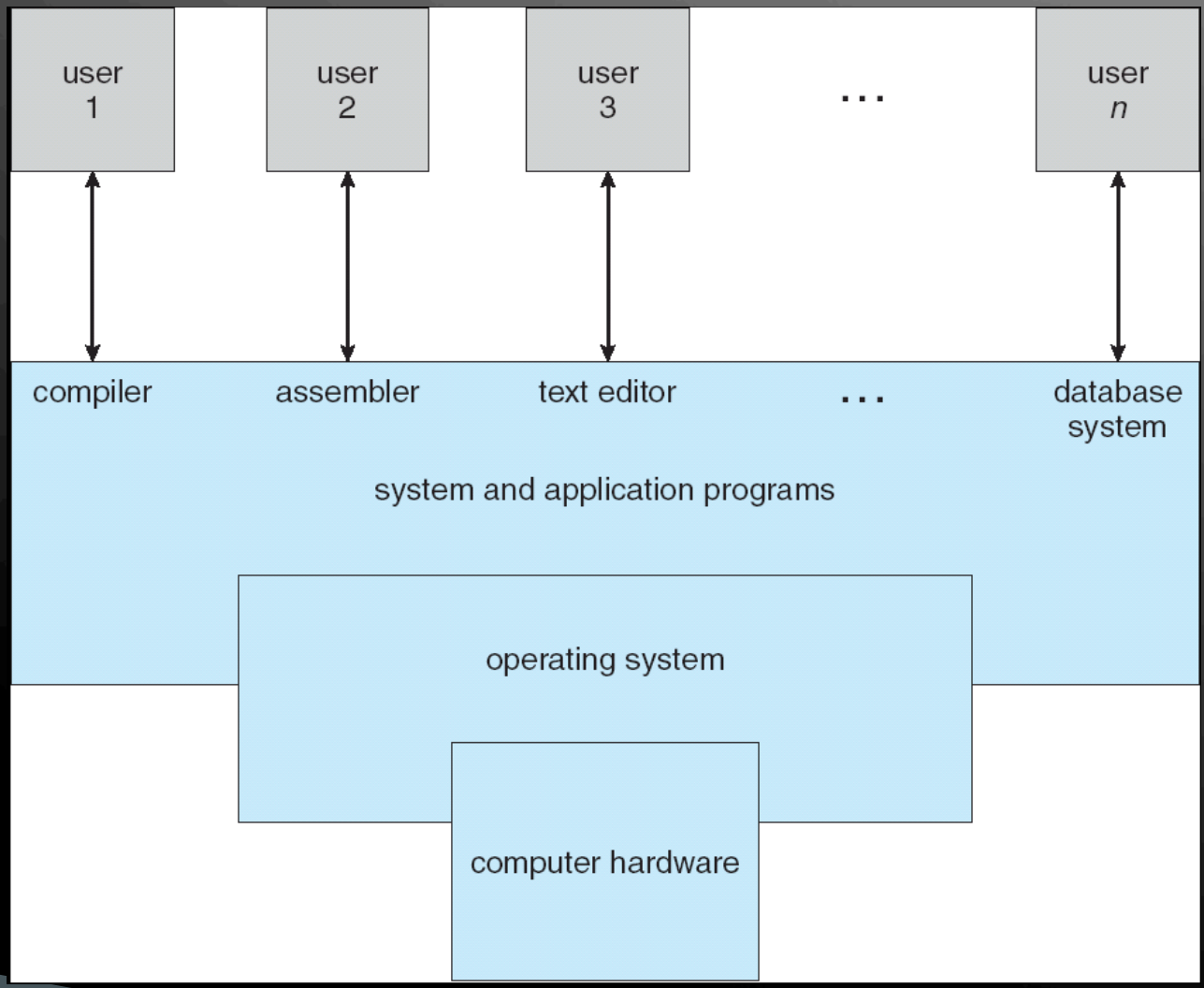
- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier. (primary goal)
  - Coordinates access to physical resources e.g CPU, memory, disk, i/o devices, etc. (primary goal)
  - Make the computer system convenient to use. (secondary goal)
  - Use the computer hardware in an efficient manner. (secondary goal)



Computer System Structure

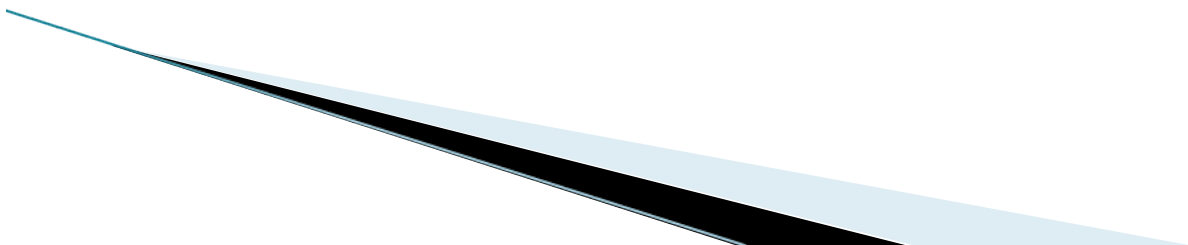
- Computer system can be divided into **FOUR** components
  - ***Hardware*** – provides basic computing resources
    - CPU, memory, I/O devices
  - ***Operating system***
    - Controls and coordinates use of hardware among various applications and users
  - ***Application programs*** – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - ***Users***
    - People, machines, other computers

Four Components of a Computer System



What Operating Systems do?

- OS is a **resource allocator**
  - Manages all resources e.g. CPU time, memory space, file-storage space, I/O devices, printers, scanner etc
  - Decides between conflicting requests for efficient and fair resource use
  - ***Undesirable scenario*** : Can occur in a multiprogramming environment and also in a distributed system
    - Suppose a process holds resource A and requests resource B
    - At the same time another process holds B and requests A
    - Both are blocked and remain so, waiting for each other
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

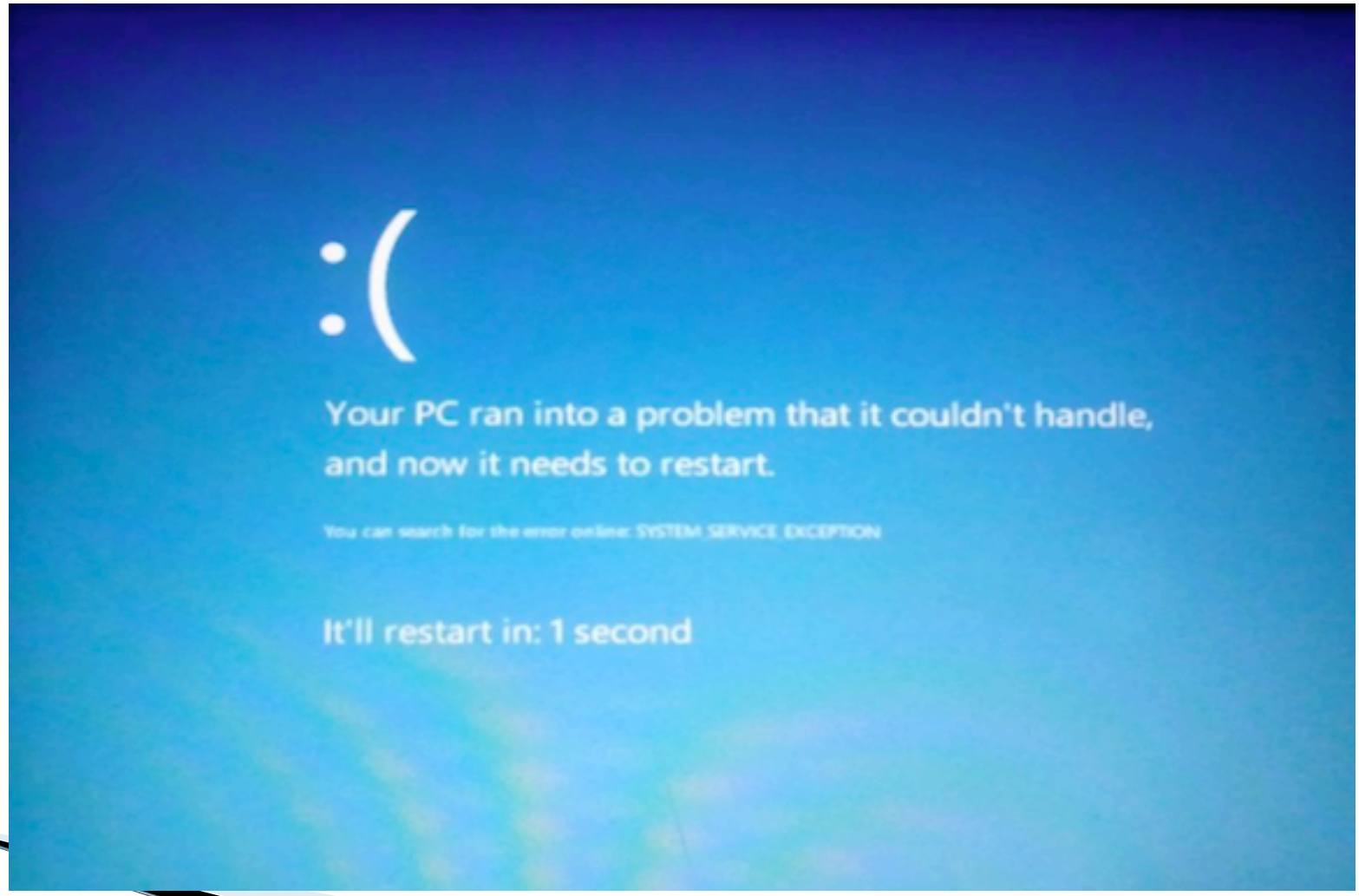


Operating System Definition

- No universally accepted definition
- The matter of what constitutes an operating system has become increasingly important. In 1998, the United States Department of Justice filed suit against Microsoft, in essence claiming that Microsoft included too much functionality in its operating systems and thus prevented application vendors from competing. For example, a web browser was an integral part of the operating system. As a result, Microsoft was found guilty of using its operating system monopoly to limit competition.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the **kernel**), with all else being systems programs and application programs. This last definition is the one that we generally follow.

Operating System Definition(cntd)

- Kernel panic?



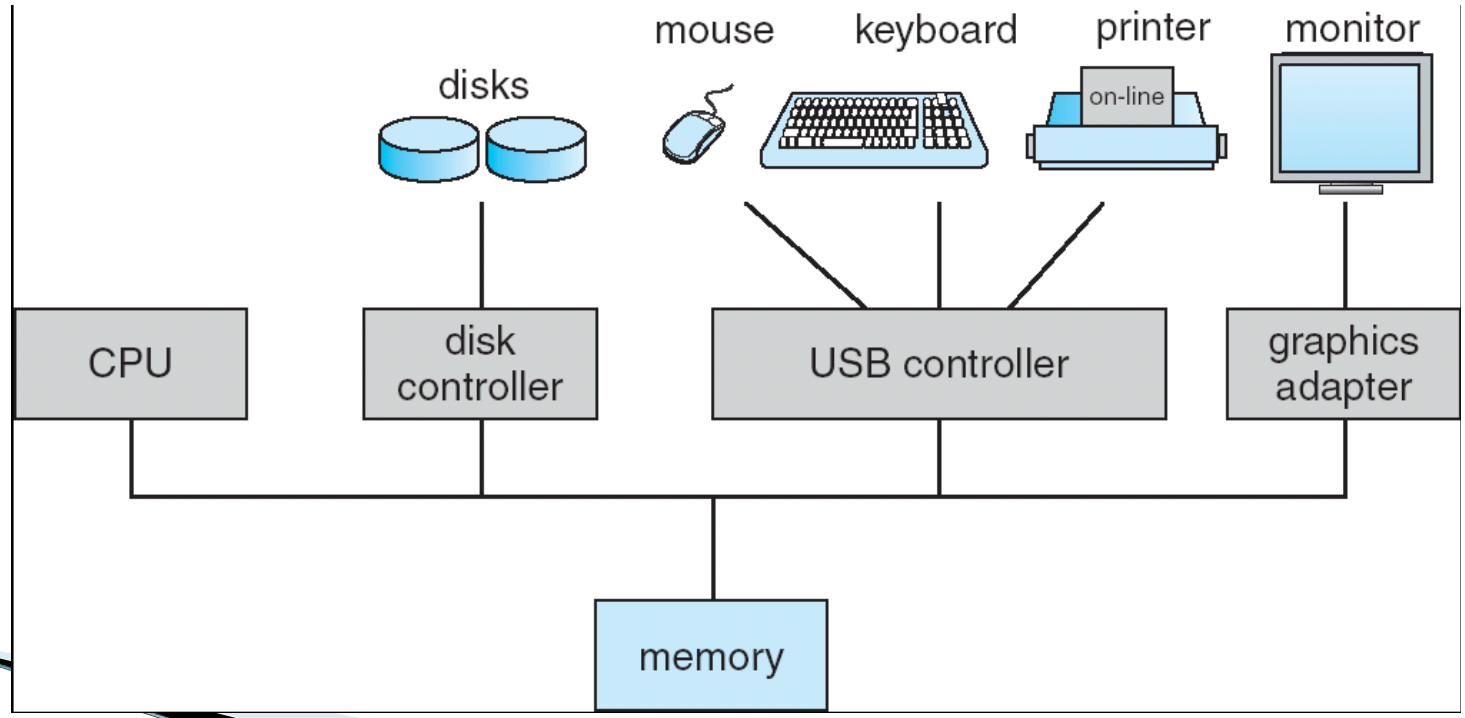
Computer Startup

- For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run.
- This initial program, or **bootstrap program**, tends to be simple. Typically, it is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term **firmware**, within the computer hardware.

- The bootstrap program knows how to load the operating system and to start executing that system. It load into memory the operating system kernel. The operating system then starts executing and waits for some event to occur.

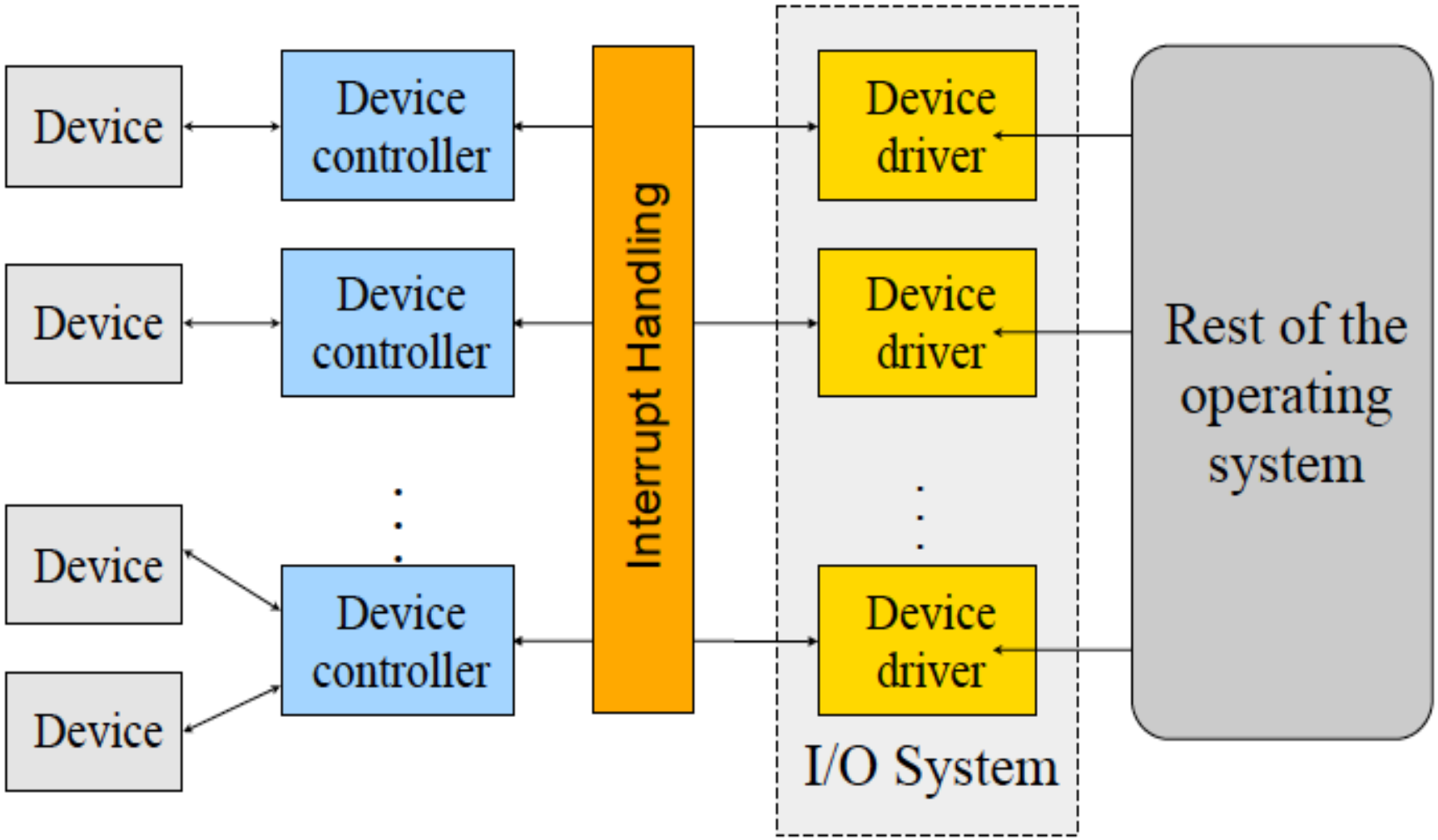
Computer System Organization

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



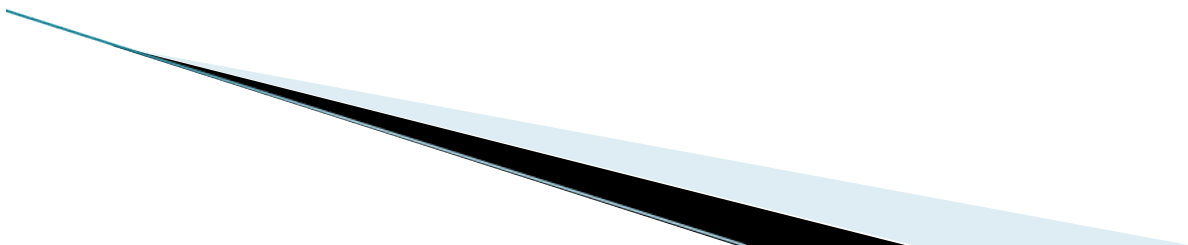
Device Controllers and Device Drivers

- A device controller is a physical part of a computer system that makes sense of the signals going to, and coming from the CPU for the different input/output devices.
- The Device Controller receives the data from a connected device and stores it temporarily in some special purpose registers (i.e. local buffer) inside the controller. Then it communicates the data with a Device Driver.
- For each device controller there is an equivalent device driver which is the standard interface through which the device controller communicates with the Operating Systems through Interrupts.
- Device controller is a hardware whereas device driver is a software.


Device
Device
Device
Device
Device controller
Device controller
Device controller
Interrupt Handling
Device driver
Device driver
Device driver
I/O System
Rest of the operating system

Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- 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***.



Common Functions of Interrupts

- When interacting with a device controller, the CPU can wait for a response by polling the status register(s), i.e., by periodically checking whether the status of the device has changed. This is known as **Polling**.
- Problem with polling: The CPU is busy waiting for some event to happen. CPU utilization will be low.

- So **Interrupts** are used by devices for asynchronous event notification. Interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

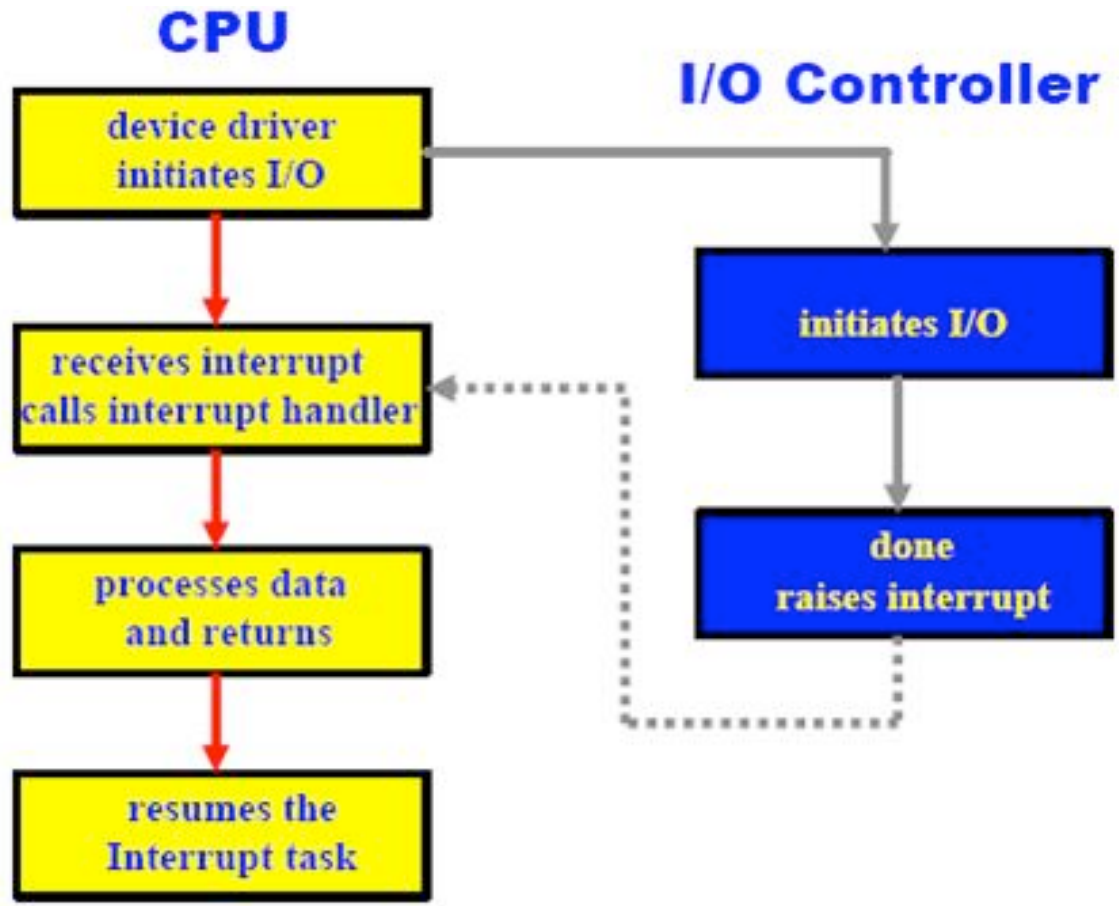
- When an interrupt is fired, the CPU jumps to a predefined position in the kernel's address space and executes an interrupt handler. When an interrupt occurs, the CPU can start reading data from the device controller's data registers.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a ***lost interrupt***.

H/w and S/w Interrupts

- A **hardware interrupt** is an electronic alerting signal sent to the processor from an external device, either a part of the computer itself such as a disk controller or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.
- A **software interrupt** (also called a monitor call) is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed.
- The former is often called a **trap** or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. For example divide a number by zero, this impossible demand will cause a divide-by-zero exception
- Software interrupt instructions function similarly to functions and are used for a variety of purposes. For example, computers often use software interrupt instructions to communicate with the disk controller to request data be read or written to the disk.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Separate segments of code determine what action should be taken for each type of interrupt (interrupt service routine).



States of Process

- A process which is Executed by the Processor have various States. The various States of the Process are as Followings:-

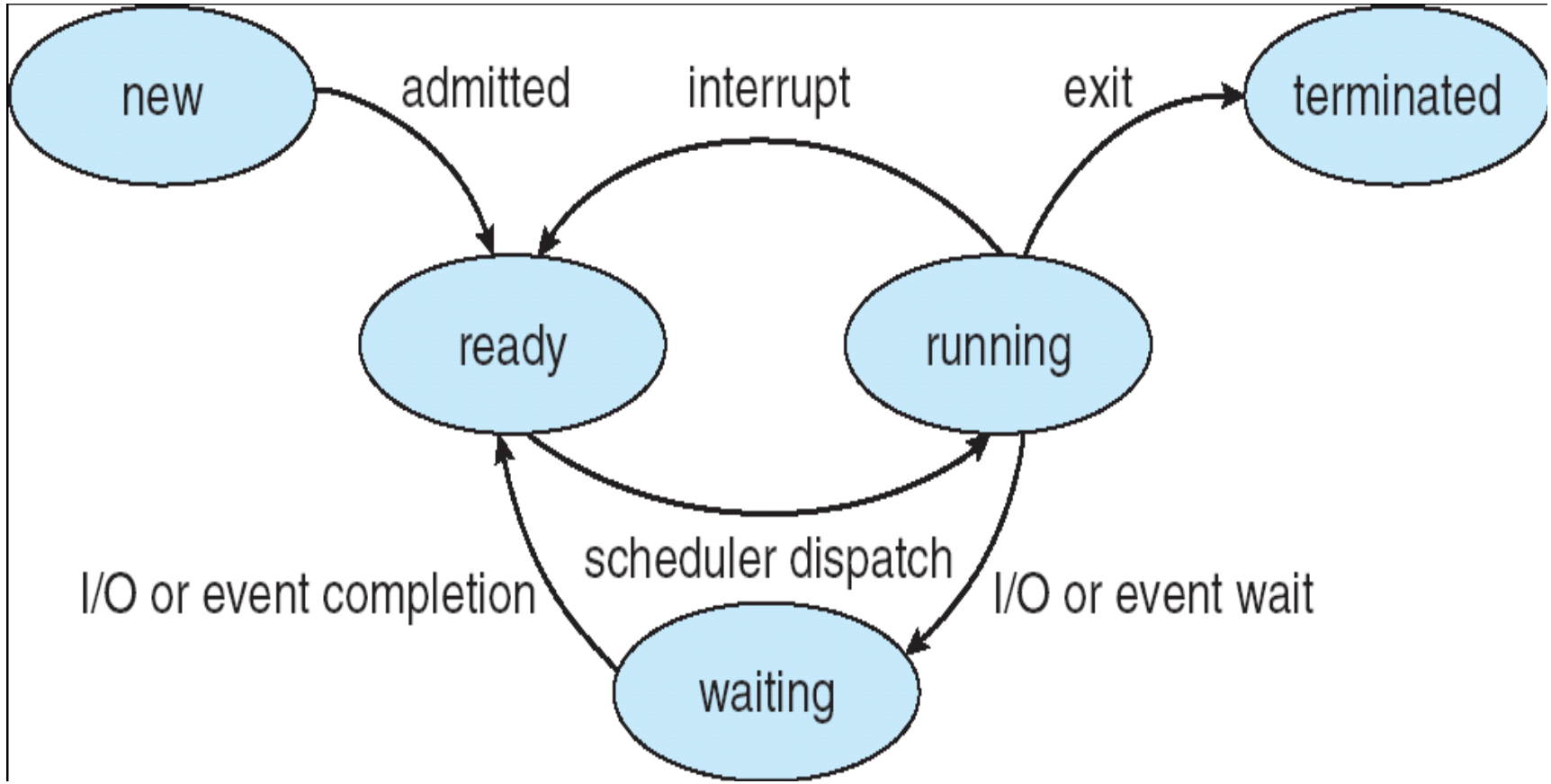
- 1) New State : When a user request for a Service from the System , then the System will first initialize the process or the System will call it an initial Process . So Every new Operation which is Requested to the System is known as the New Born Process.

- 2) Running State : When the Process is Running under the CPU, or When the Program is Executed by the CPU , then this is called as the Running process.

- 3) Waiting : When a Process is Waiting for Some Input and Output Operations then this is called as the Waiting State.

- 4) Ready State : When the Process is Ready to Execute but he is waiting for the CPU to Execute then this is called as the Ready State. After the Completion of the Input and outputs the Process will be on Ready State means the Process will Wait for the Processor to Execute.

- 5) Terminated State : After the Completion of the Process , the Process will be Automatically terminated by the CPU . So this is also called as the Terminated State of the Process.

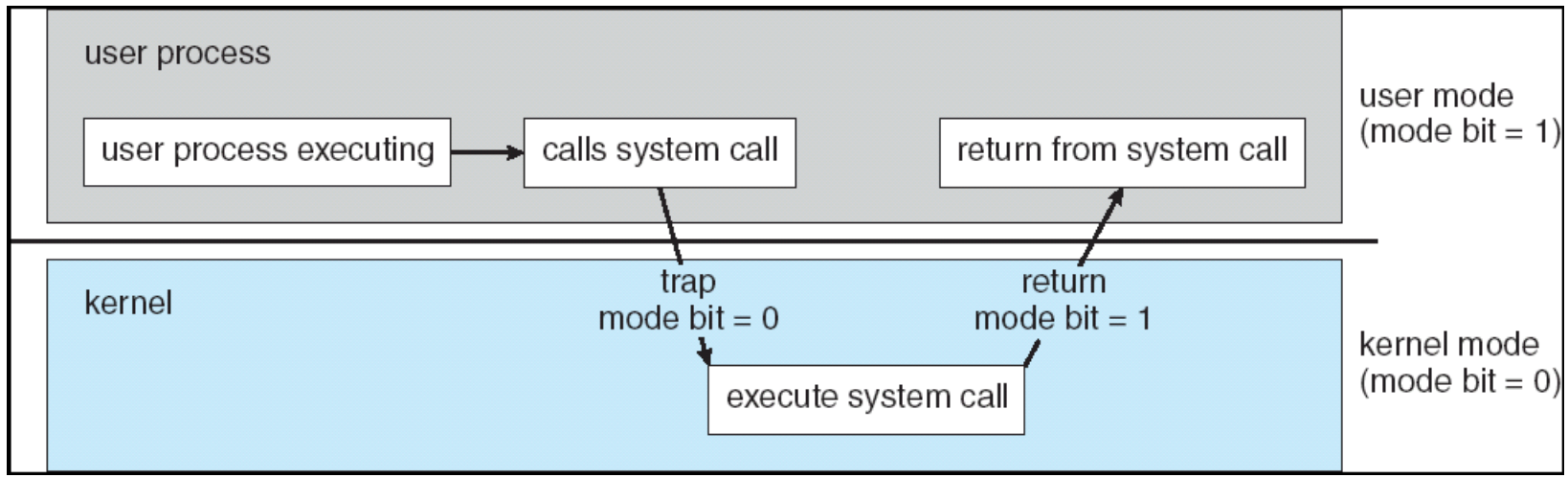

new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

Operating-System Operations

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **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
    - For example, if an I/O instruction is executed under user mode, the hardware does not execute it, although identifies it as an illegal execution and traps it to the operating system.

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - A timer is another mechanism, in addition to dual-mode operation, that maintains an operating system's overall control of the computer system
  - The timer works by triggering an interrupt after a specified time period; the operating system. The timer interrupt guarantees a "drop" into the operating system, regardless of a user program's state



I/O Structure

- There are two types of input/output (I/O) synchronization: **synchronous I/O and asynchronous I/O**. Asynchronous I/O is also referred to as overlapped I/O.

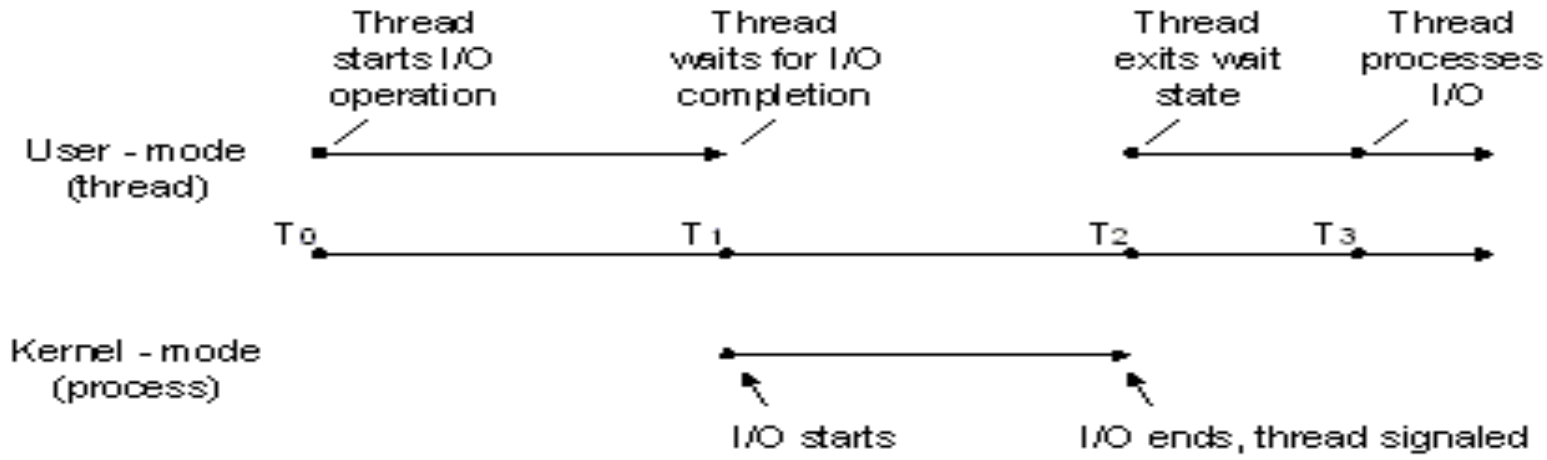
- In **synchronous** I/O, a thread starts an I/O operation and immediately enters a **wait state** until the I/O request has completed.
- A thread performing **asynchronous** file I/O sends an I/O request to the kernel by calling an appropriate function. If the request is accepted by the kernel, the calling thread continues processing another job until the kernel signals to the thread that the I/O operation is complete. It then interrupts its current job and processes the data from the I/O operation as necessary.

I/O Structure

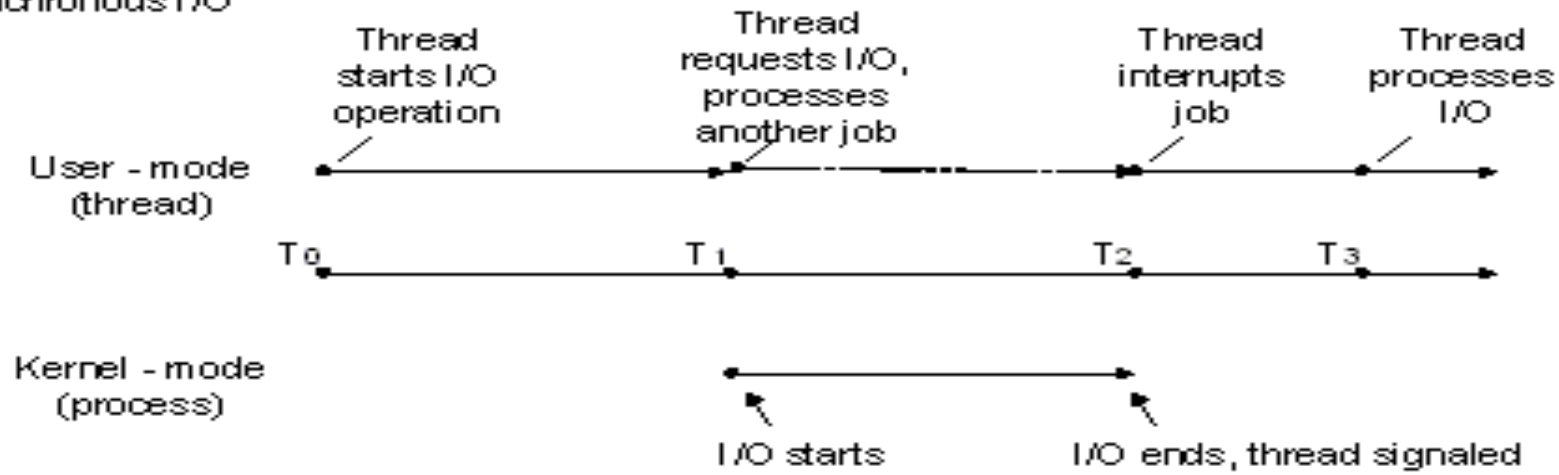
- In situations where an I/O request is expected to take a large amount of time, such as a refresh or backup of a large database or a slow communications link, asynchronous I/O is generally a good way to optimize processing efficiency.

- However, for relatively fast I/O operations, the overhead of processing kernel I/O requests and kernel signals may make asynchronous I/O less beneficial. In this case, synchronous I/O would be better.

Synchronous I/O



Asynchronous I/O



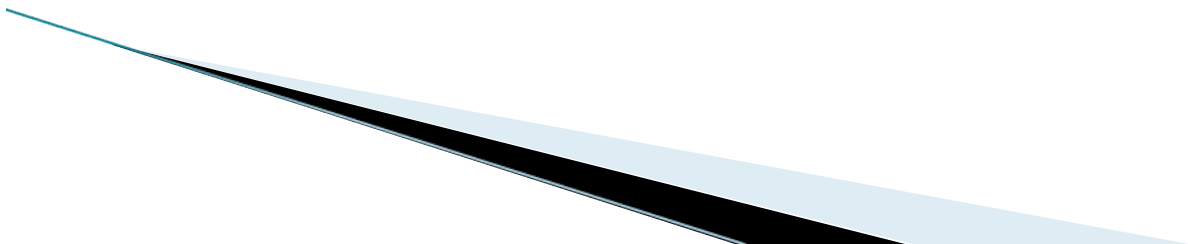
Types of I/O devices

Block devices

- Organize data in fixed-size blocks
- Transfers are in units of blocks
- Blocks have addresses and data are therefore addressable
- E.g. hard disks, USB disks, CD-ROMs

Character devices

- Delivers or accepts a stream of characters, no block structure
- Not addressable, no seeks
- Can read from stream or write to stream
- Printers, network interfaces



Direct Memory Access Structure

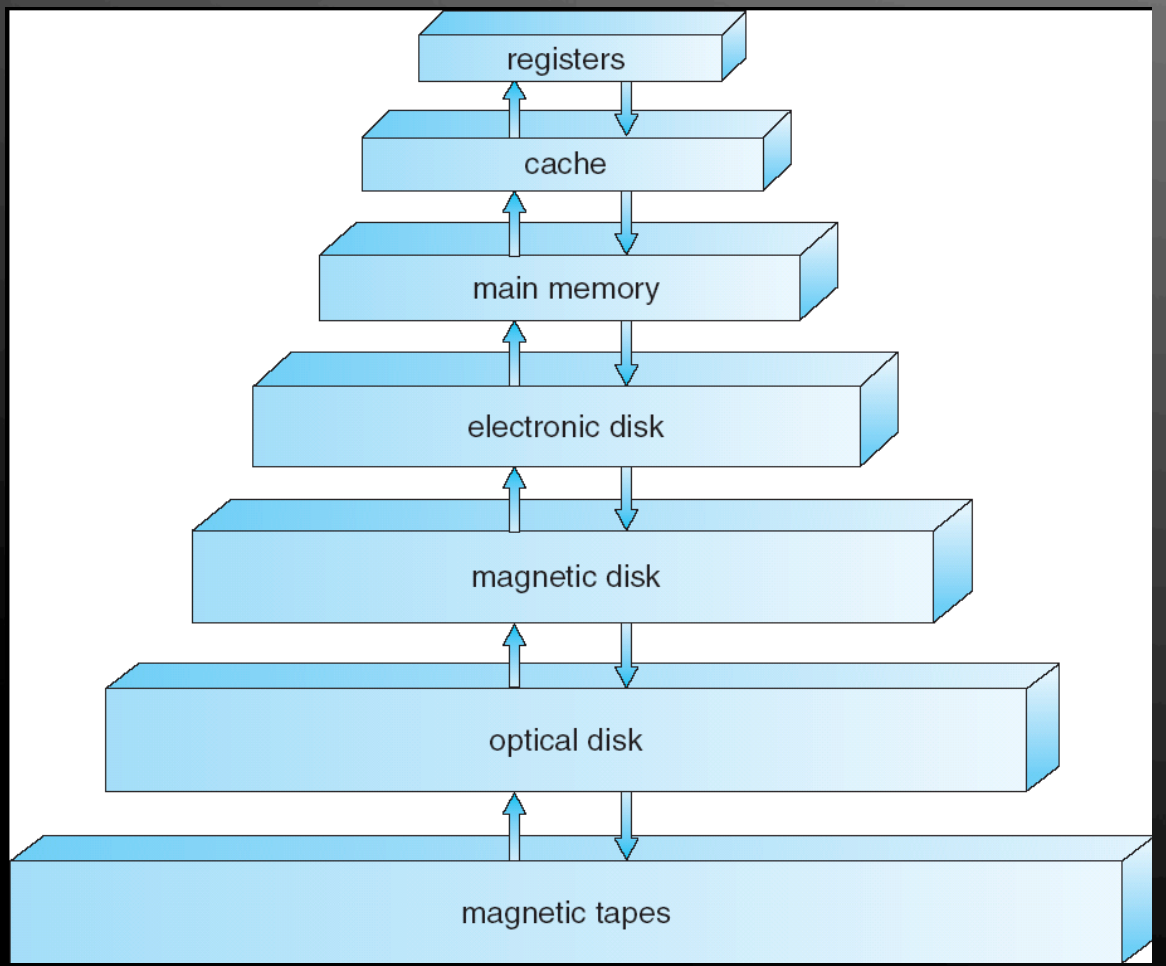
- DMA is a method of transferring data from the computer's RAM to another part of the computer without processing it using the CPU.
- While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device. In these situations, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices.
- With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done. Only one interrupt is generated per block, rather than the one interrupt per byte.
- For example, a sound card may need to access data stored in the computer's RAM, but since it can process the data itself, it may use DMA to bypass the CPU.

Storage Structure

- Main memory – only large storage media that the CPU can access directly.

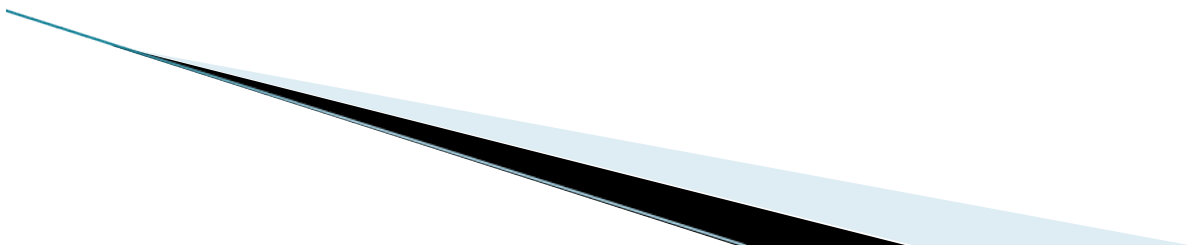
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
  - Magnetic disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
    - The *disk controller* determines the logical interaction between the device and the computer.
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Storage-Device Hierarchy

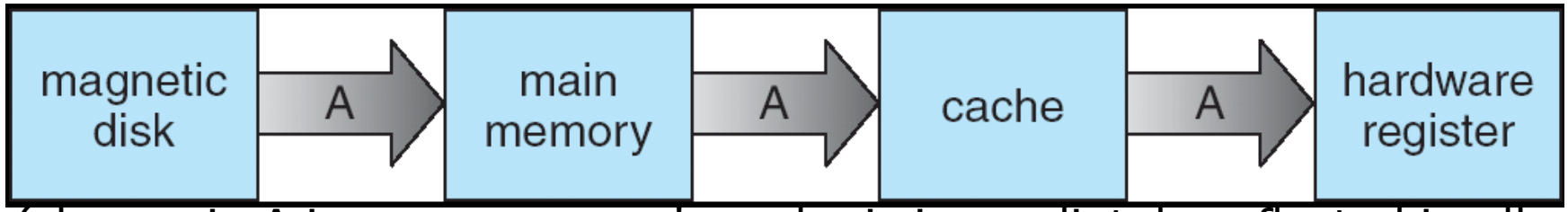


Caching

- Important principle, performed at many levels in a computer system.
- Information in use is copied from slower to faster storage temporarily so that future requests for that data can be served faster.
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy



Migration of Integer A from Disk to Register

- In a multitasking environment, where the CPU is switched back and forth among various processes, extreme care must be taken to ensure that, if several processes wish to access A, then each of these processes will obtain the most recently updated value of A
- 

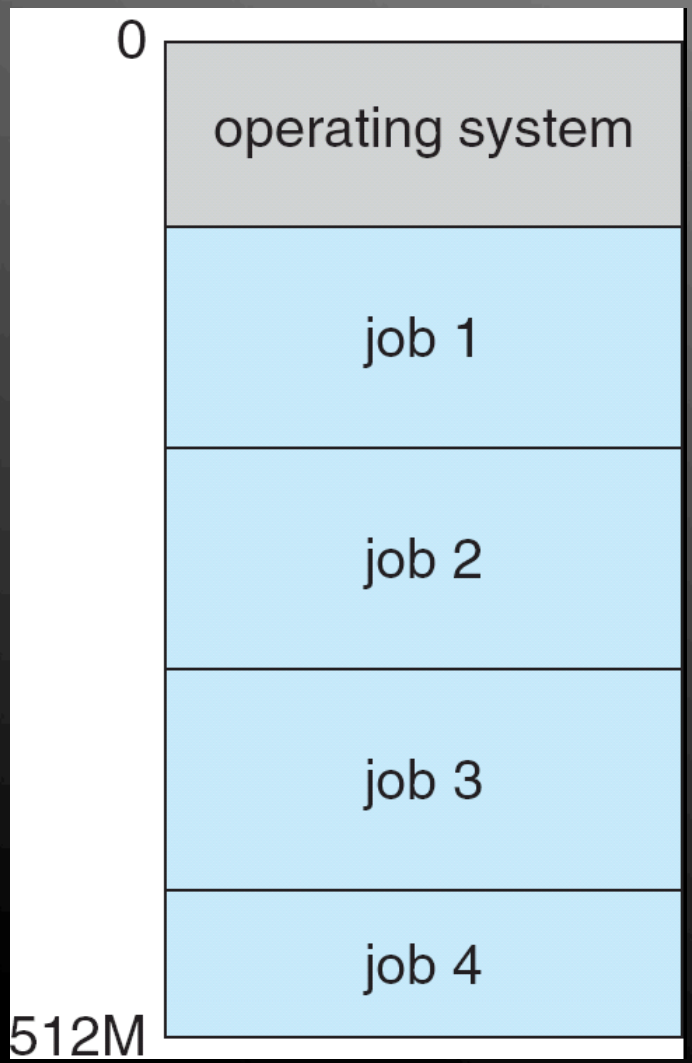
(change in A in one processor's cache is immediately reflected in all other processor's cache where A resides.) in hardware such that all CPUs have the most recent value in their cache.

Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - 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 has to wait (for I/O for example), OS switches to another job.In non-multiprogrammed system, the CPU would sit idle

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - 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

Memory Layout for Multiprogrammed System



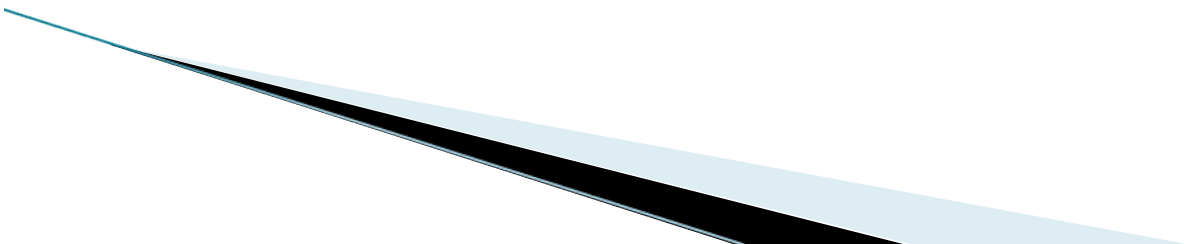
Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling



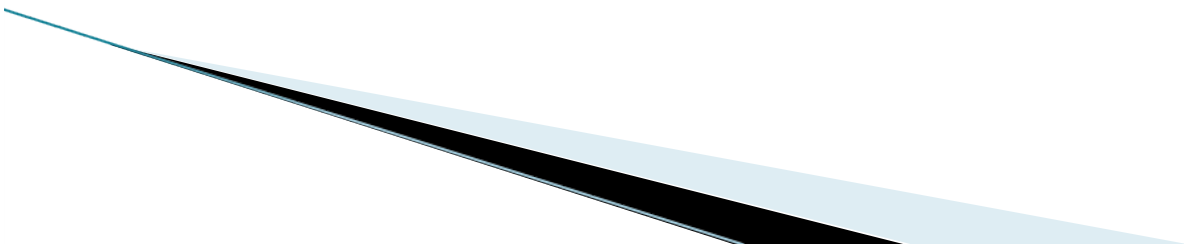
Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media



Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time.
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance),
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS

- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service

- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (g**roup ID**) allows set of users to be defined and controls managed, then also associated with each process, file