

CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

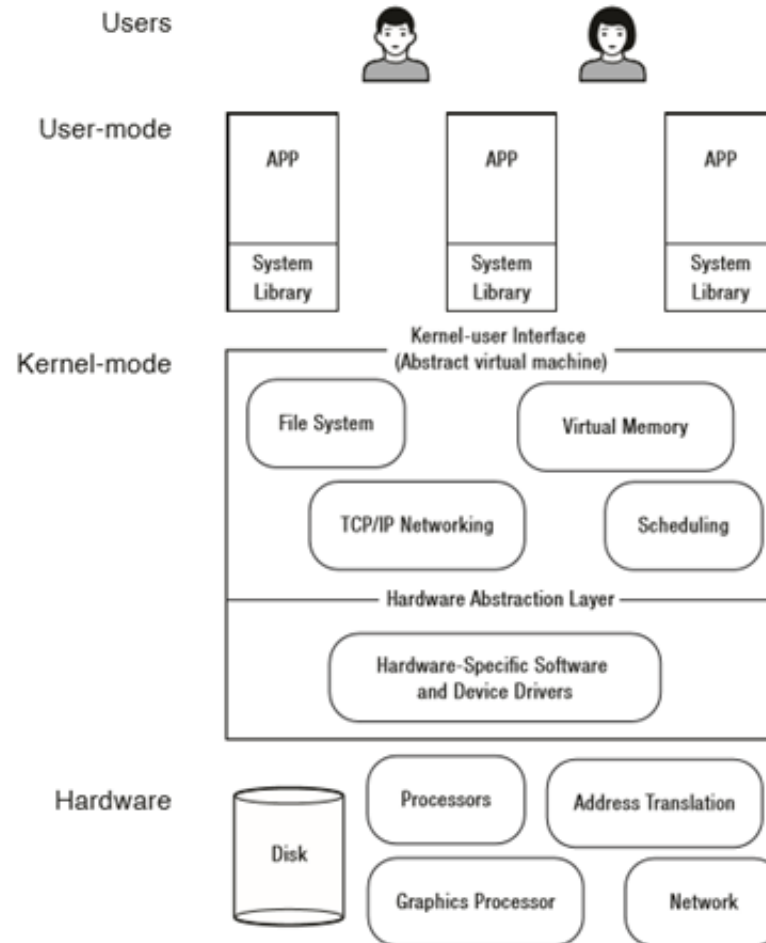
Fall 2016 Lecture 3



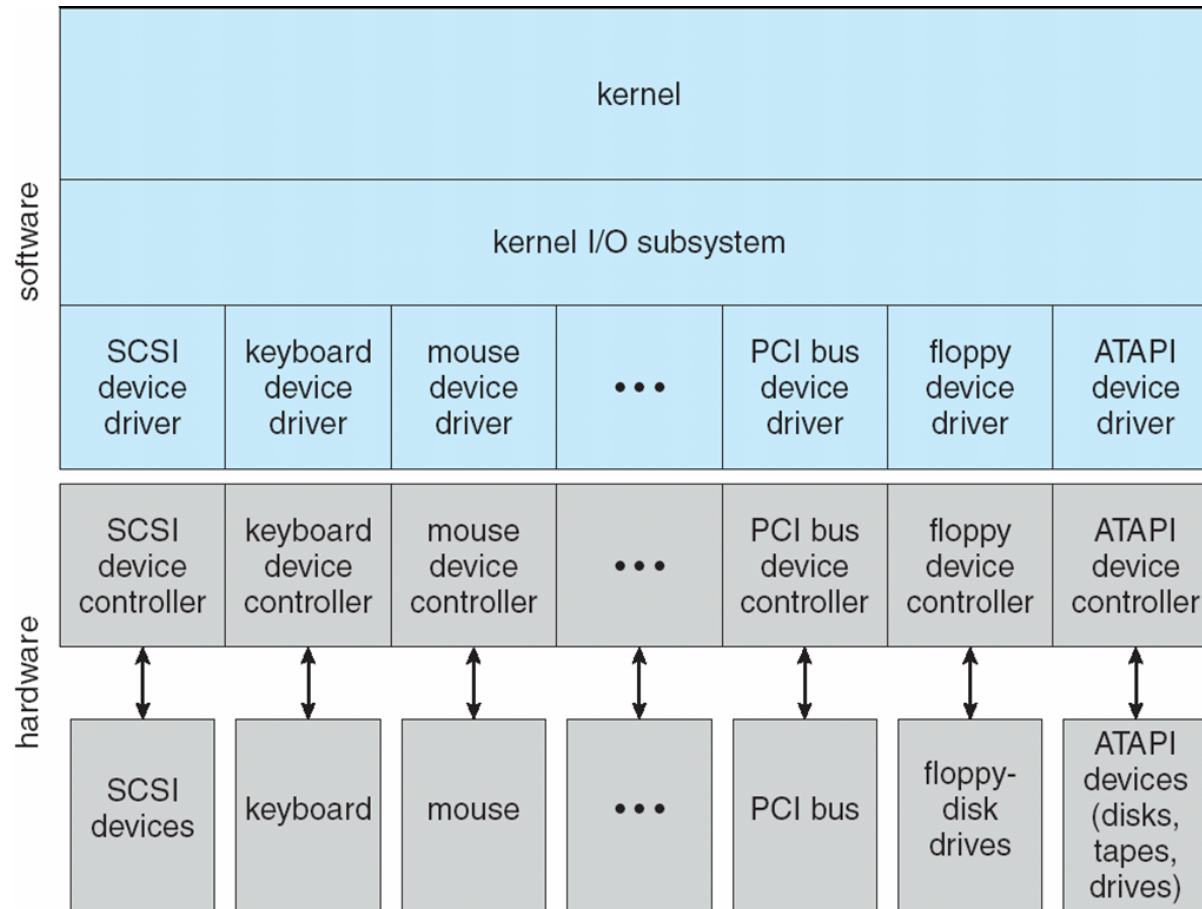
Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

What is an Operating System?



A Kernel I/O Structure



Interrupts

- Polling is slow
- Interrupts used in practice
- CPU **Interrupt-request line** triggered by I/O device
 - Checked by processor after each instruction
- Interrupt handler receives interrupts
 - Maskable to ignore or delay some interrupts
- **Interrupt vector** to dispatch interrupt to correct handler
 - **Context switch at start and end**
 - Based on priority
 - Some nonmaskable
 - Interrupt chaining if more than one device at same interrupt number

Direct Memory Access

- for movement of a block of data
 - To/from disk, network etc.
- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
 - Source and destination addresses
 - Read or write mode
 - Count of bytes
 - Writes location of command block to DMA controller
 - Bus mastering of DMA controller – grabs bus from CPU
 - **Cycle stealing** from CPU but still much more efficient
 - When done, interrupts to signal completion

Questions from last time

Interrupts: Why? How?

I/O implementation: IO devices are much slower

- Options: Interrupts vs polling
- Interrupt request line is hardware
- Interrupt causes transfer of control to *Interrupt Service Routine*
- Hence need to save context. Context restored when returning.

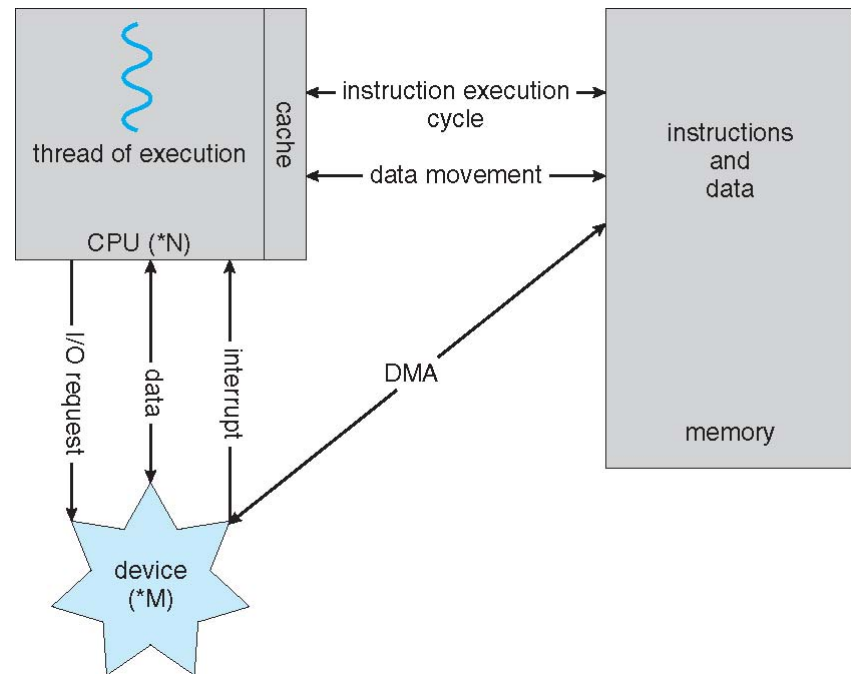
Questions from last time

Block transfer using DMA Controller vs CPU

- CPU needs to fetch instructions for each word
- DMAC, once initialized, doesn't need to fetch instructions
- DMA: direct connection between memory and IO device

Direct Memory Access Structure

- high-speed I/O devices
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block



Storage

Storage Structure

Memory
for short

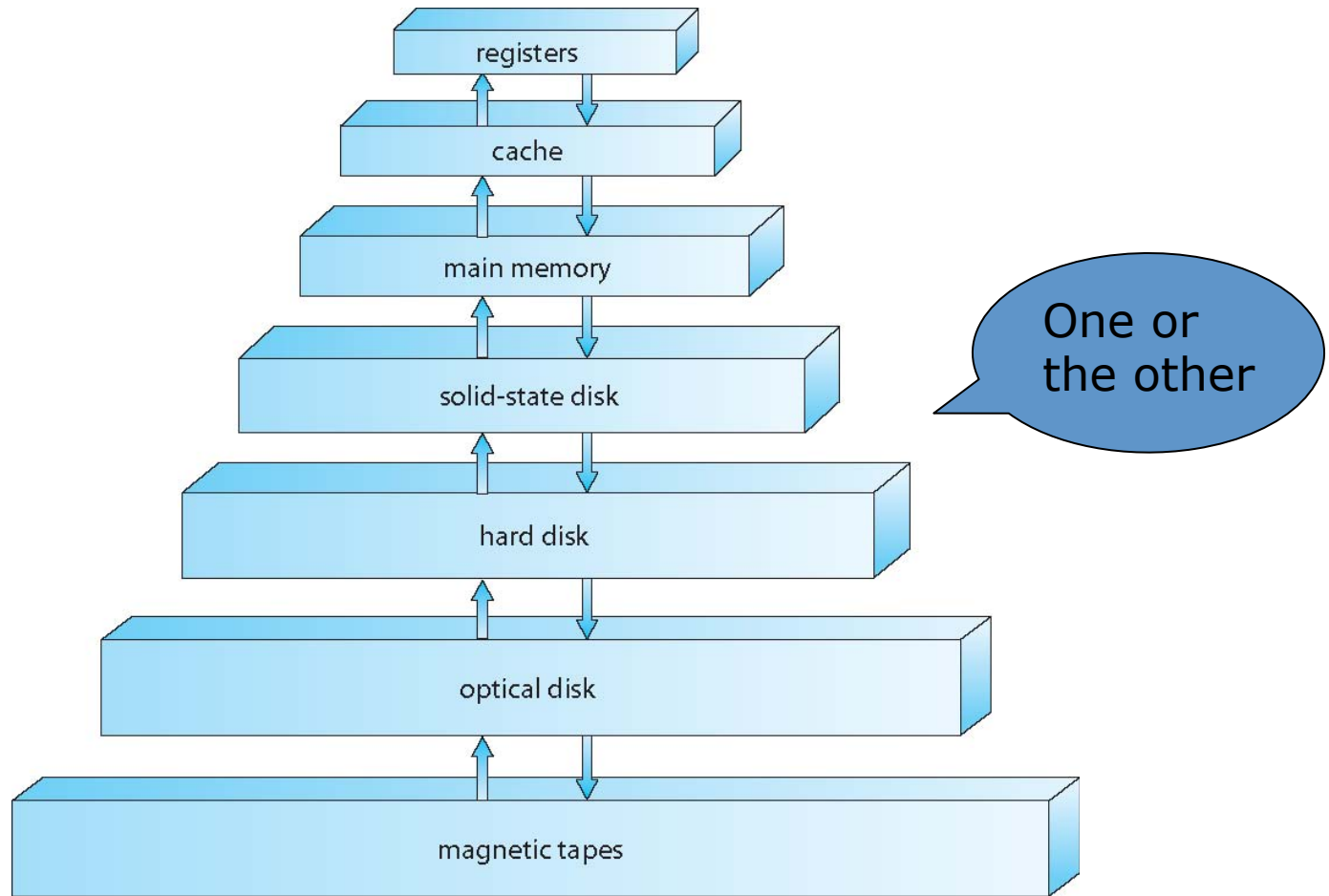
Disk
for short

- Main memory – only large storage media that the CPU can access directly
 - **Random access**
 - Typically **volatile (except for ROM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
 - Hard disks (HDD) – rigid platters covered with magnetic recording material
 - Disk surface divided into **tracks**, which are subdivided into **sectors**
 - The **disk controller – transfers** between the device and the processor
 - **Solid-state disks (SSD)** – faster than hard disks, lower power consumption
 - More expensive, but becoming more popular
- Tertiary/removable storage
 - External disk, thumb drives, cloud backup etc.

Storage Hierarchy

- Storage systems organized in hierarchy
 - Speed
 - Cost
 - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
 - Provides uniform interface between controller and kernel

Storage-Device Hierarchy



Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

- Cache managed by hardware. Makes main memory appear much faster.
- Disks are several orders of magnitude slower.

Caching

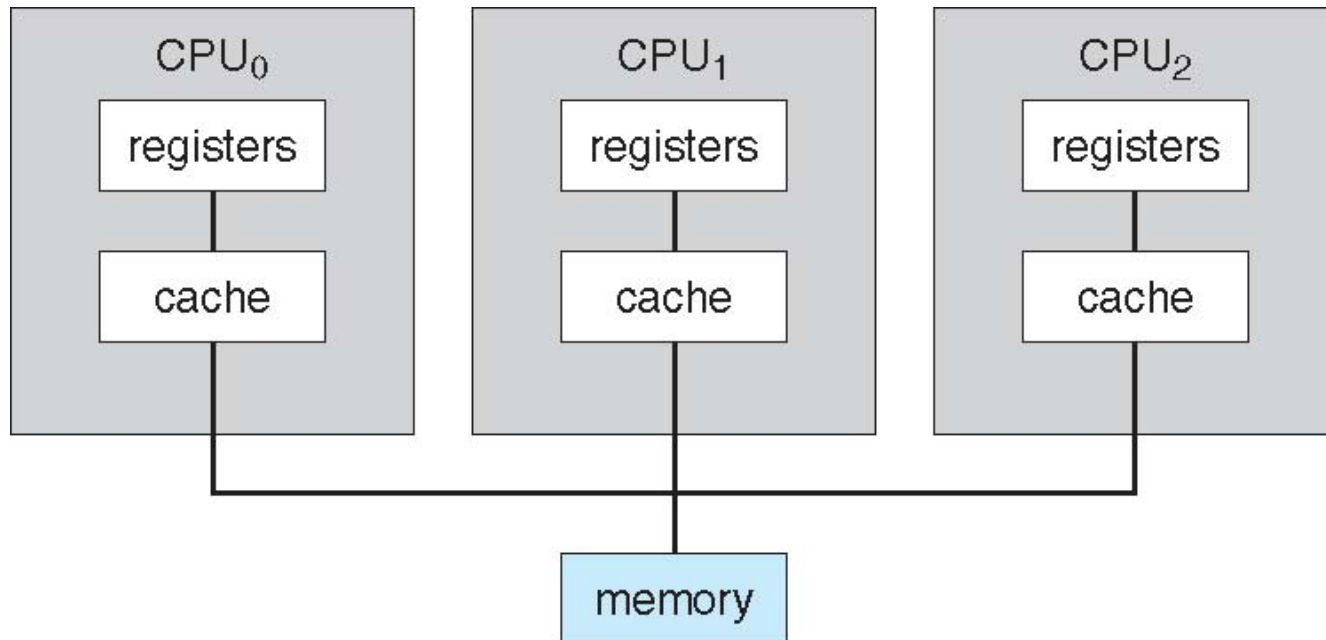
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (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

Multiprocessors

Multiprocessors

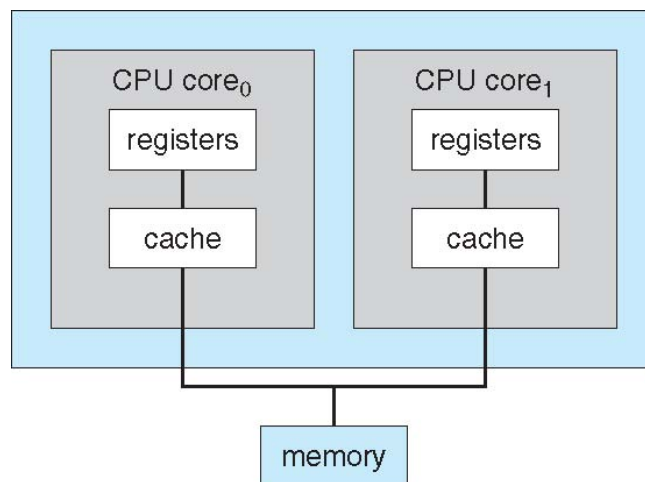
- Most systems use a single general-purpose processor
 - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
 - Also known as **parallel systems**, **tightly-coupled systems**
 - Advantages include:
 1. **Increased throughput**
 2. **Economy of scale**
 3. **Increased reliability** – graceful degradation or fault tolerance
 - Two types:
 1. **Asymmetric Multiprocessing** – each processor is assigned a specific task.
 2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



A Dual-Core Design

- Multi-chip and **multicore**
- Systems containing all chips
 - Chassis containing multiple separate systems

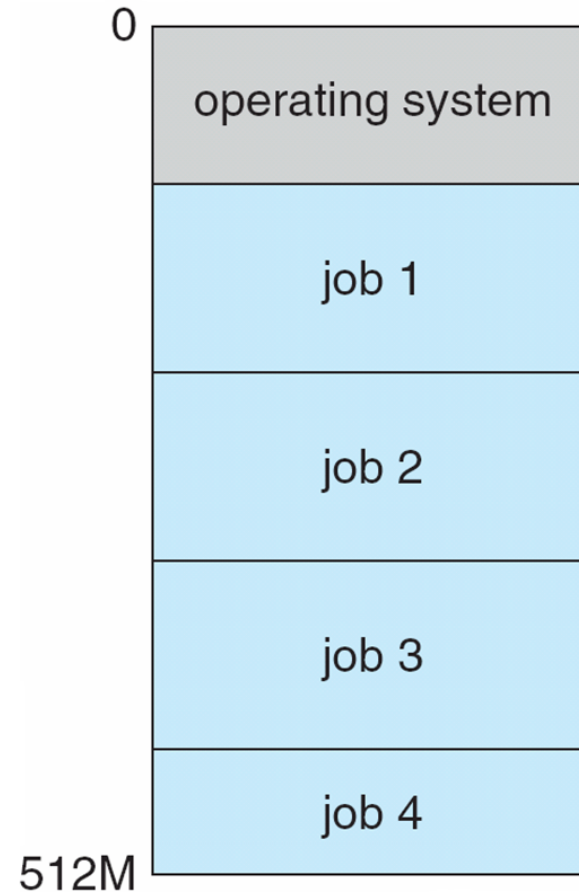


Operating System Structure

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
- **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
 - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System



Operating-System Operations

- **Interrupt driven** (hardware and software)
 - Hardware interrupt by one of the devices
 - Software interrupt (**exception** or **trap**):
 - Software error (e.g., division by zero)
 - Request for operating system service
 - Other process problems include infinite loop, processes modifying each other or the operating system

Operating-System Operations (cont.)

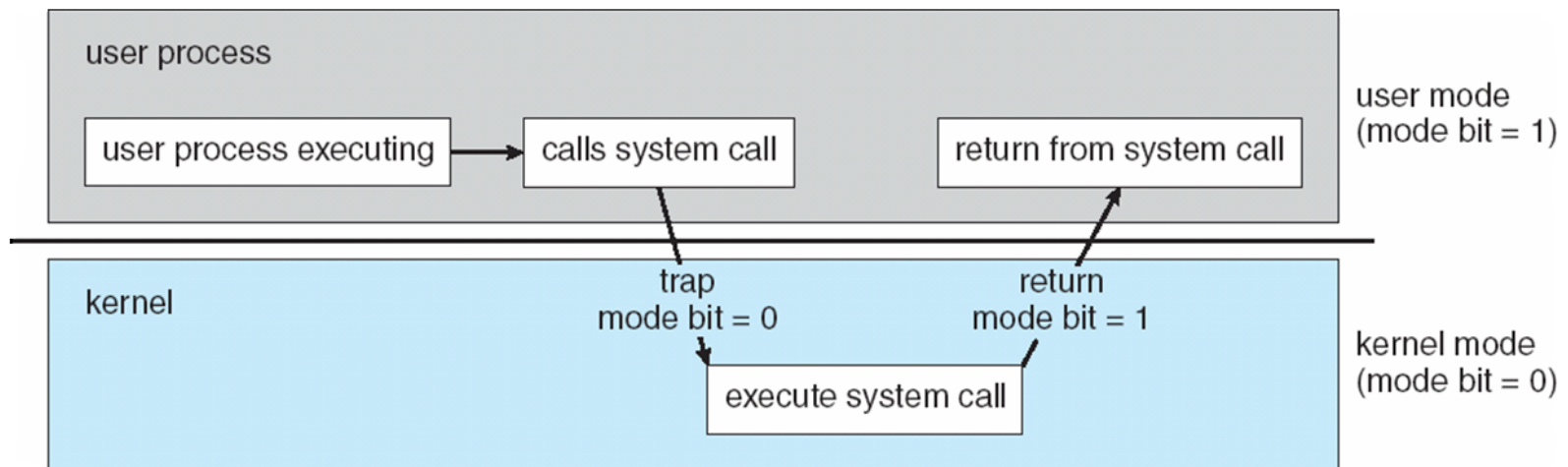
- **Dual-mode** operation allows OS to protect itself and other system components
 - **User mode** and **kernel mode**
 - **Mode bit** provided by hardware
 - Provides ability to distinguish when system is running user code or kernel code
 - Some instructions designated as **privileged**, only executable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



called Supervisor mode
in LC3

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



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 & Storage Management

Means main
memory here

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and 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

CPU
scheduling

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 directories
 - 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
- 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 – by OS or applications
 - Varies between WORM (write-once, read-many-times) and RW (read-write)

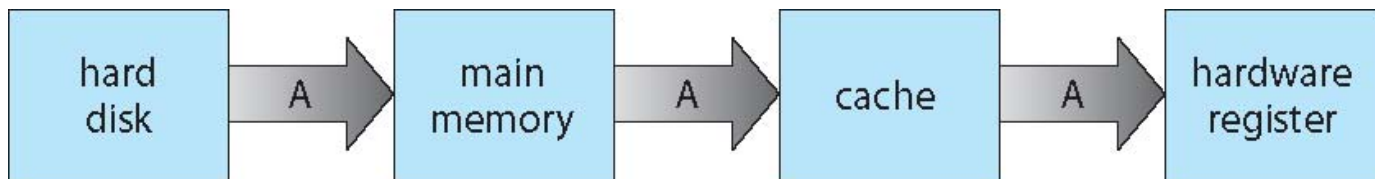
Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
 - Several copies of a datum can exist
 - Various solutions covered in Chapter 17 (*will not get to it*)

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),
 - **spooling** (the overlapping of output of one job with input of other jobs) like printer queue
 - General device-driver interface
 - Drivers for specific hardware devices