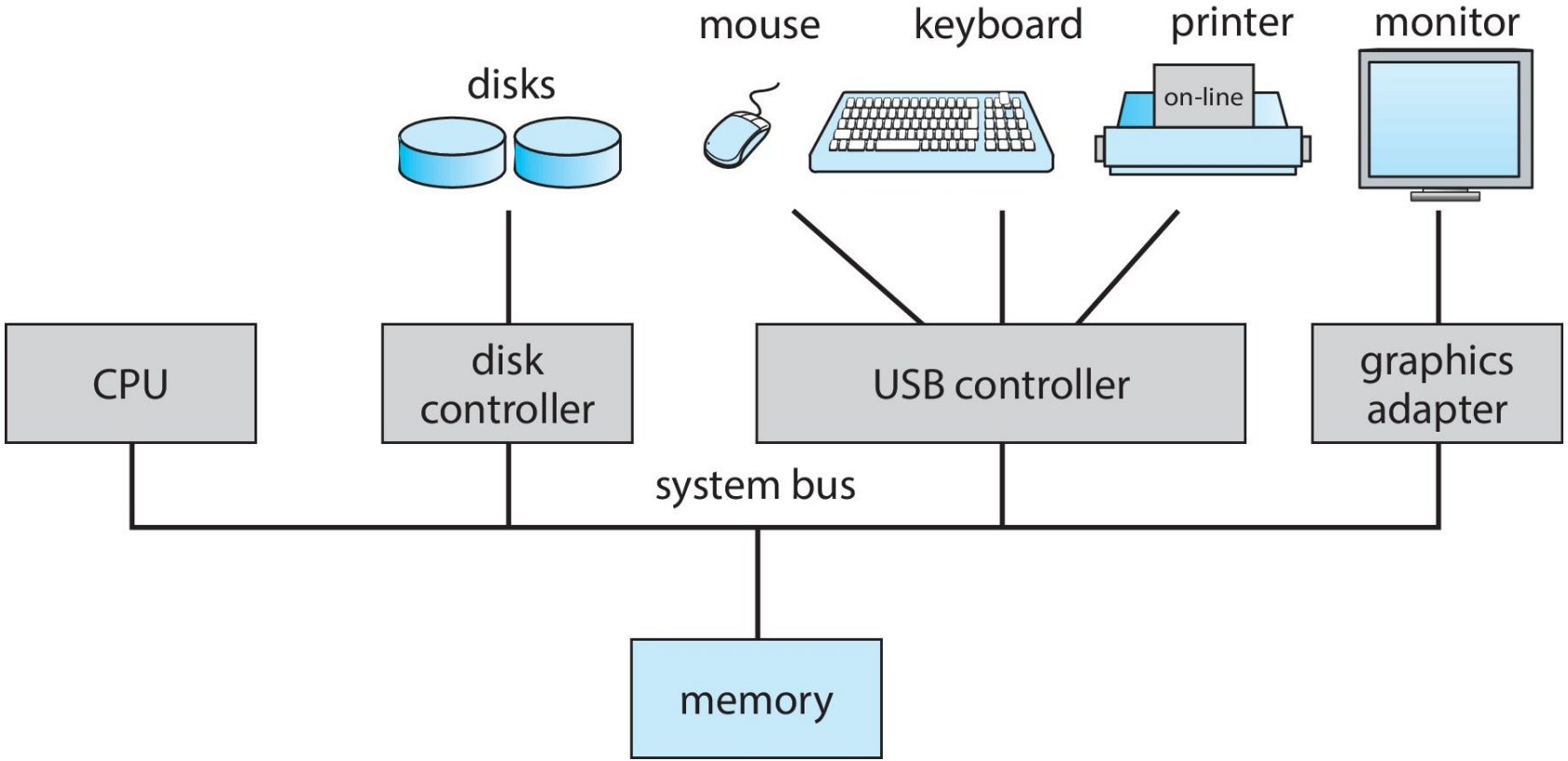
1. Introduction

CS 4352 Operating Systems

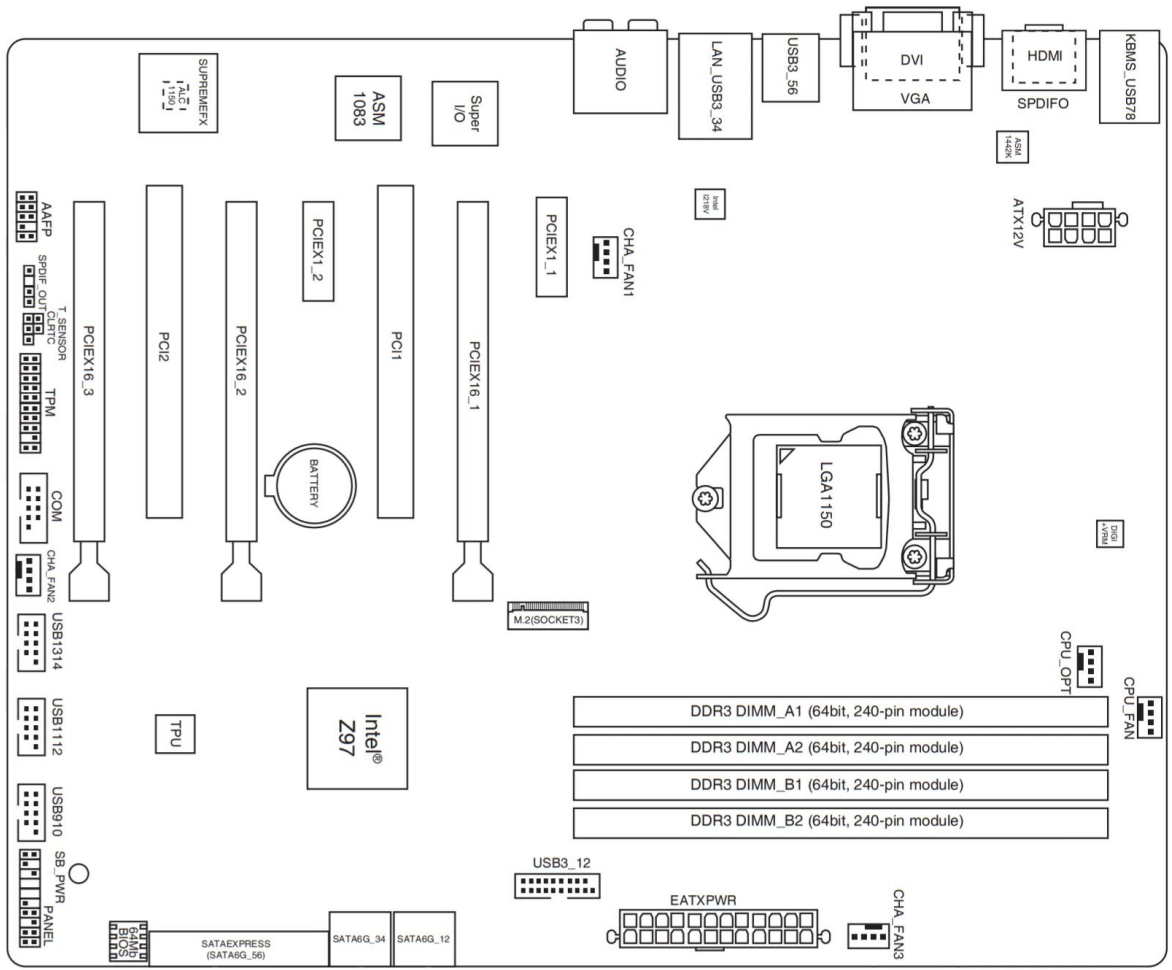
Zhenkai Zhang

Adopted from slides of Drs. Silberschatz, Huang, & Bryant

Computer Hardware



PC Motherboard

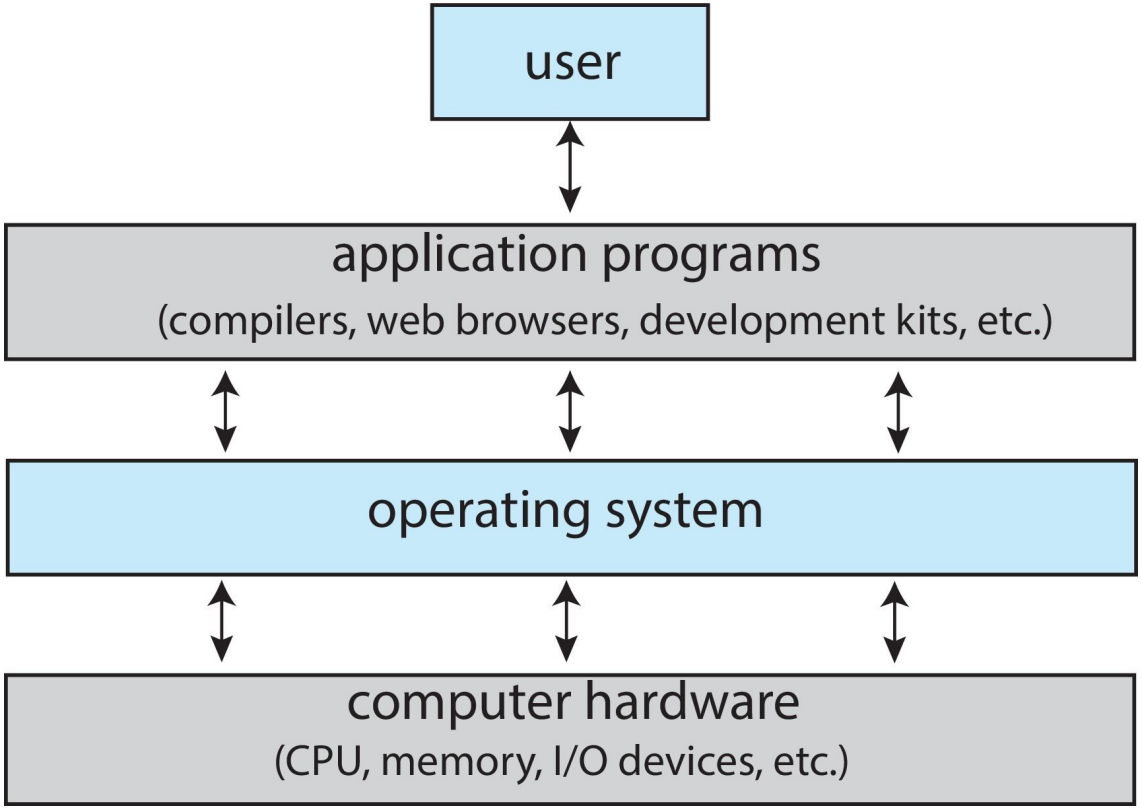


Credit: https://www.techspot.com/article/1965-anatomy-motherboard/

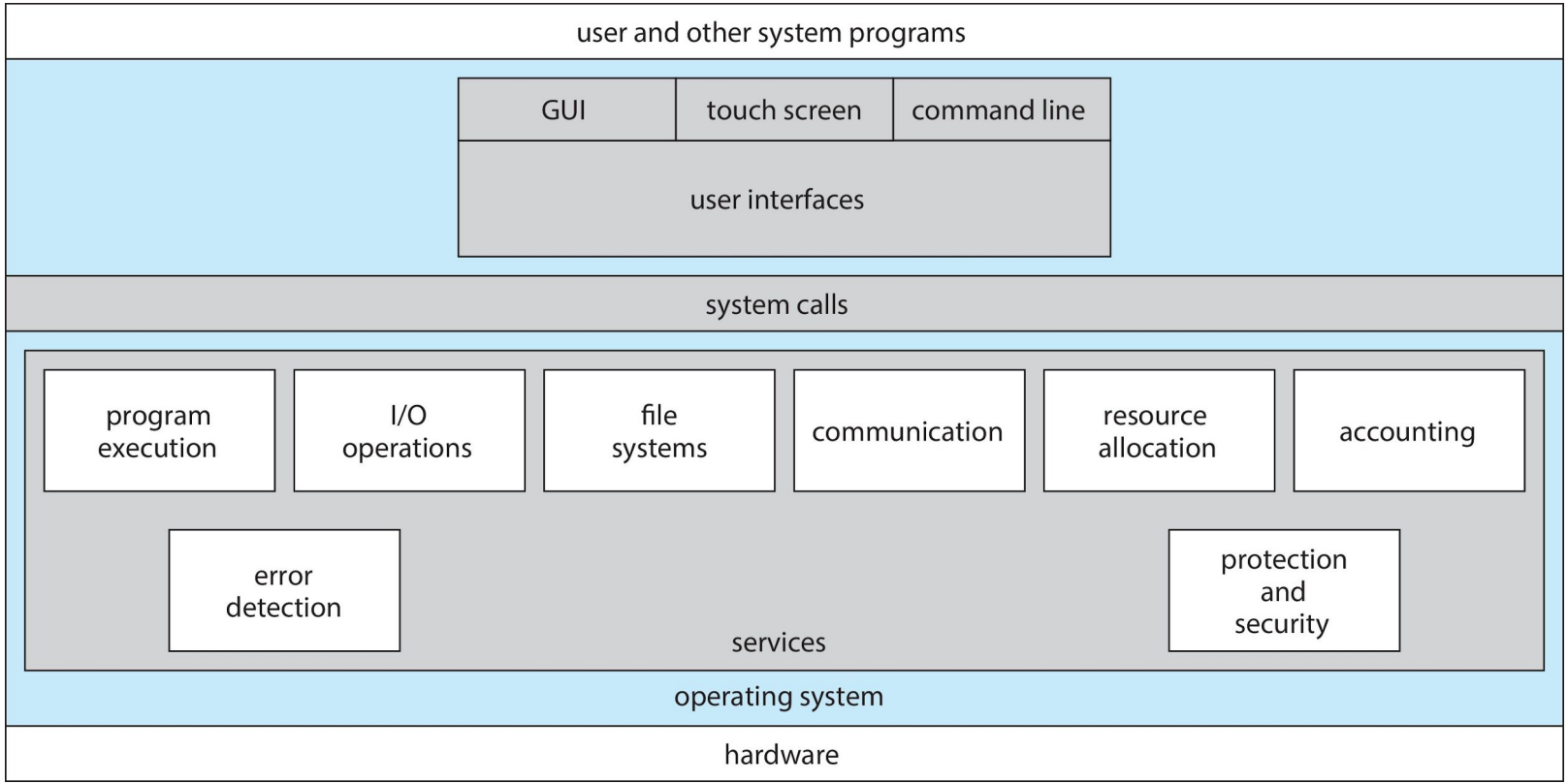
What is an Operating System?

- An OS is software that acts as an intermediary between a user and the computer hardware
  - Manage hardware resources
  - Provide abstractions to hide details from applications
- OS goals:
  - Make the computer hardware convenient to use
  - Use the computer hardware in an efficient manner
  - Execute user programs and make solving user problems easier
  - Enforce protection and security

Abstract View of Components of Computer System



More Detailed …



What Operating Systems Do?

- Depends on the point of view
- Users want **convenience**, **ease of use** and **good performance**
  - Don’t care about resource utilization
- Shared computer like mainframe or minicomputer must keep all users happy
  - OS is a resource allocator and control program making efficient use of hardware and managing execution of user programs
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention

Operating System Definition

- No universally accepted definition (many definitions), e.g.,
  - A program that shares a computer among multiple programs and provides a more useful set of services than the hardware alone
  - A program that makes the hardware do “something useful”
  - Everything a vendor ships when you order an operating system
- The core part of the OS is called the OS kernel, or just kernel
- Today’s OSes for general purpose and mobile computing also include middleware – a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics
- Key point: the operating system is itself a program!
  - But unlike other programs, it should have full control over all resources!

An OS can be treated as a resource manager!

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:
  - Process creation
  - Process termination
  - Interprocess communication
  - Process synchronization
  - CPU scheduling

Memory Management

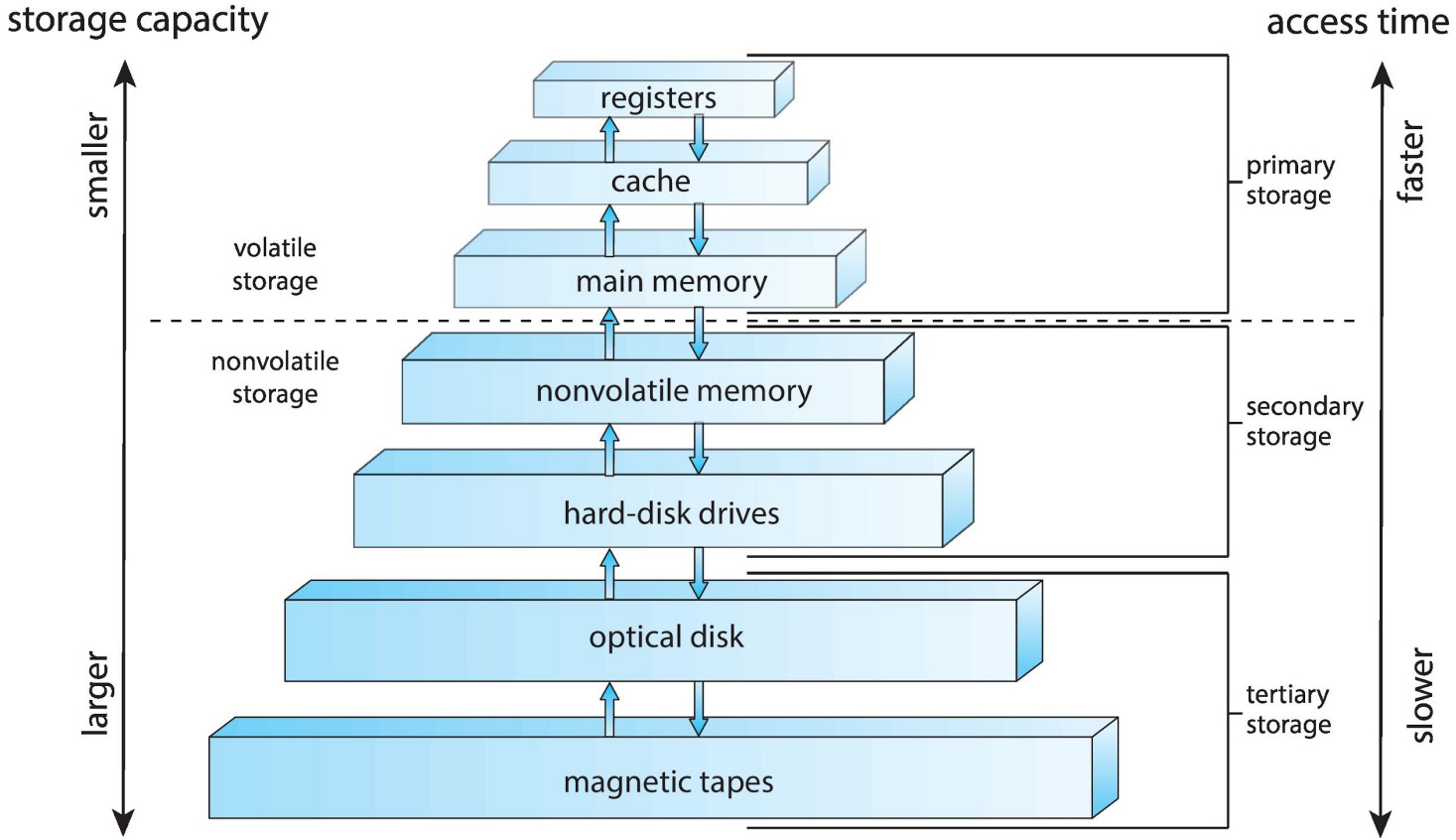
- To execute a program, all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when

- 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

Secondary-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
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

Storage-Device Hierarchy



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

Characteristics of Various Types 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

File Systems

- 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

I/O Subsystem

- One purpose of OS is to hide peculiarities of I/O 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)
  - General device-driver interface
  - Drivers for specific hardware devices

Pondering Moment 1

- The kernel has full access to all hardware and CPU instructions, e.g.,
  - Directly access I/O devices
  - Write anywhere in memory
  - Read content from any memory address
- User processes (like Chrome or Firefox) should not be allowed full access to all hardware nor should they be able to execute all CPU instructions
  - Otherwise, anarchy!!!!!
- But they are all running programs -- how can we enforce the above things?

Dual Modes of Operation

- User mode
  - Limited privileges
  - Only has access to those resources granted by the OS
- Kernel mode
  - Execution with the full privileges of the hardware
  - Read/write to any memory, access any I/O device, read/write any disk sector, send/read any packet, etc.
- To implement user mode and kernel mode, we need hardware support!
  - A flag in a CPU register determines the mode
    - On the x86 it's called the CPL (current privilege level)
      - The bottom two bits of the CS register
        - CPL = 0 means kernel mode: privileged
        - CPL = 3 means user mode: no privilege

Privileged Instructions

- A subset of instructions restricted to use only by the OS
- E.g.,

INVD—Invalidate Internal Caches

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F 08	INVD	ZO	Valid	Valid	Flush internal caches; initiate flushing of external caches.

NOTES:

* See the IA-32 Architecture Compatibility section below.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
ZO	NA	NA	NA	NA

Description

Invalidates (flushes) the processor's internal caches and issues a special-function bus cycle that directs external caches to also flush themselves. Data held in internal caches is not written back to main memory.

After executing this instruction, the processor does not wait for the external caches to complete their flushing operation before proceeding with instruction execution. It is the responsibility of hardware to respond to the cache flush signal.

The INVD instruction is a privileged instruction. When the processor is running in protected mode, the CPL of a program or procedure must be 0 to execute this instruction.

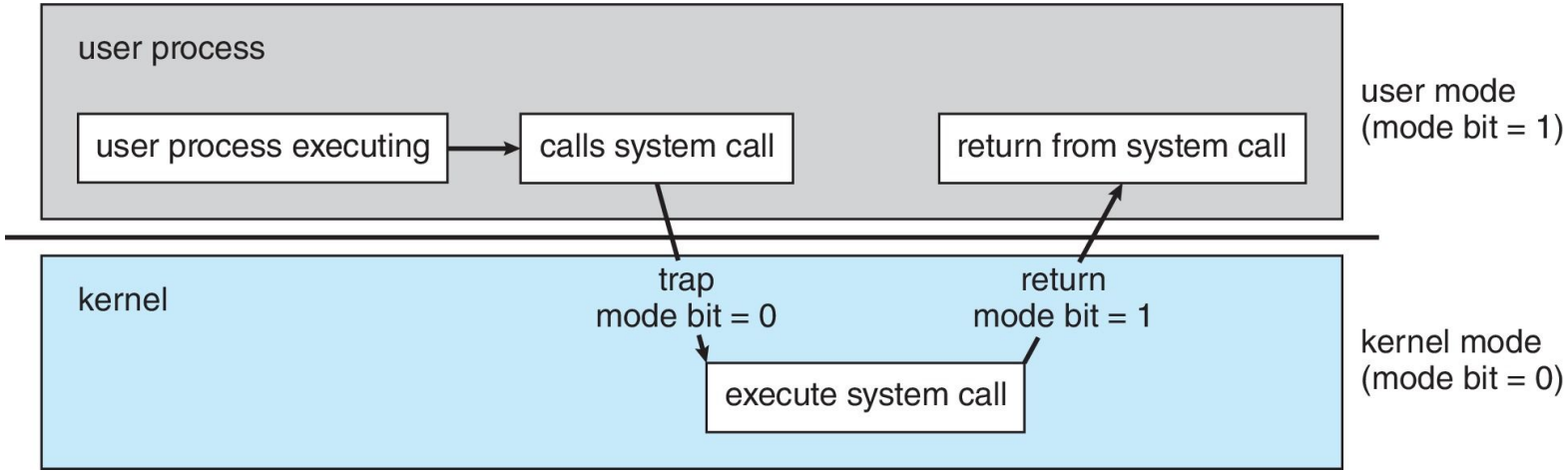
The INVD instruction may be used when the cache is used as temporary memory and the cache contents need to be invalidated rather than written back to memory. When the cache is used as temporary memory, no external device should be actively writing data to main memory.

Pondering Moment 2

- Processes running in user mode which has no control over resources
  - E.g., they can’t directly read/write disk sectors where files are stored
    - You may say “BS, I read and write files all the time on my computer?!”
- How can processes in user mode operate on resources?
  - They need to require services of the OS kernel

System Calls

- **System calls** form an interface to the OS services
  - A special **trap** instruction is used to switch from **user mode** to **kernel mode**
  - We will discuss it in detail next lecture



Pondering Moment 3

- Resources are scarce (e.g., CPU time), and how should they be used?
  - Run applications one by one?
    - What if there is an infinite loop?
  - Can we rely on each one to relinquish resources voluntarily?
    - Communism never works!
- OS should be able to mediate accesses to resources in a **time sharing** way
  - Better be fair

Multitasking

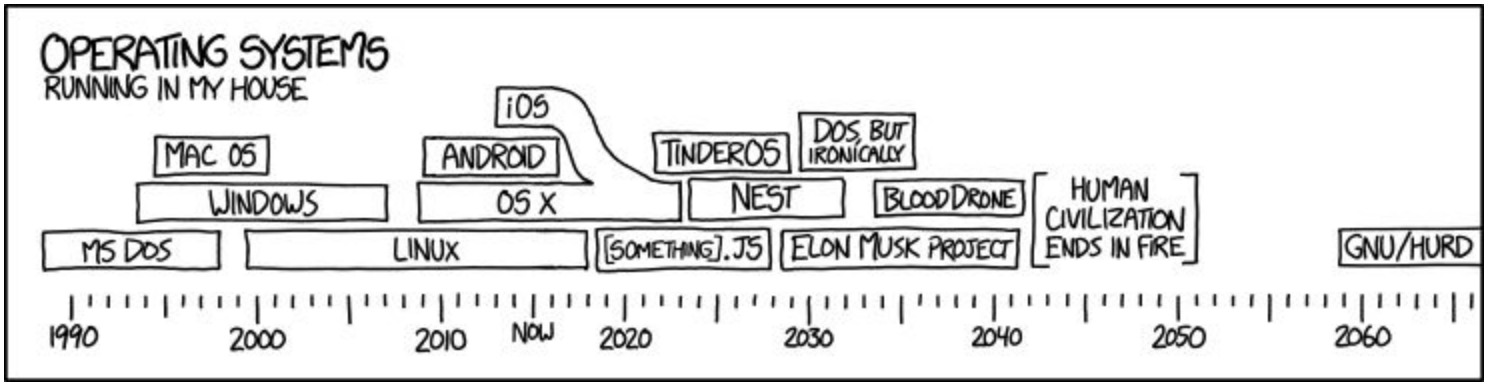
- More than one process can be running at once
  - But some resources can be used by one process at a time
- Mechanism: **context-switch**
  - OS switches processes according to a policy (scheduling policy)
    - The **scheduler** makes the decision with regards to which process will run
  - We will study this in detail later in the course
- But we still don’t know how to prevent infinite loop (or process hogging resources)

Timer

- The timer is critical for an operating system
- It is the fallback mechanism for OS to reclaim control over the machine
  - 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
- There are fundamental connections and differences between **traps** and **interrupts** (and also **faults** and **aborts**)
  - We will cover this next lecture

Next Lecture

Exceptions (interrupts, traps, faults, and aborts), system calls, etc.



Credit: https://xkcd.com/272/