CSCI-2510 Principles of Computer Systems

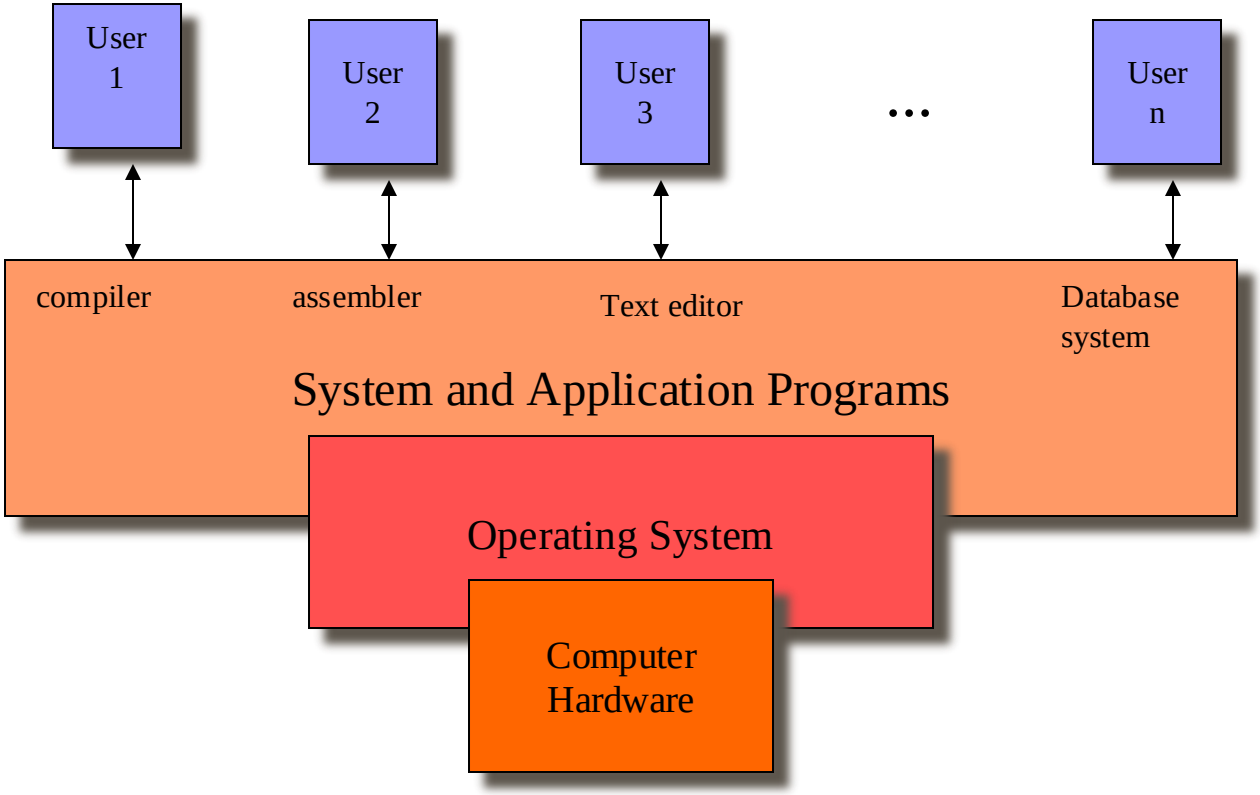
What is an Operating System?

- An OS is a program that acts an intermediary between the user of a computer and computer hardware.
- Major cost of general purpose computing is software.
  - OS simplifies and manages the complexity of running application programs efficiently.

Computer System Components

- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices).
- Operating System
  - Controls and coordinates the use of hardware among application programs.
- Application Programs
  - Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).
- Users
  - People, machines, other computers

Abstract View of System



Operating System Views

- Resource allocator
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program
  - Controls execution of user programs and operation of I/O devices.
- Kernel
  - The program that executes forever (everything else is an application with respect to the kernel).

Operating system roles

- **Referee**
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications
- **Illusionist**
  - Each application appears to have the entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport
- **Glue**
  - Libraries, user interface widgets, …
  - Reduces cost of developing software

Example: file systems

- Referee
  - Prevent users from accessing each other's files without permission
- Illusionist
  - Files can grow (nearly) arbitrarily large
  - Files persist even when the machine crashes in the middle of a save
- Glue
  - Named directories, printf, …

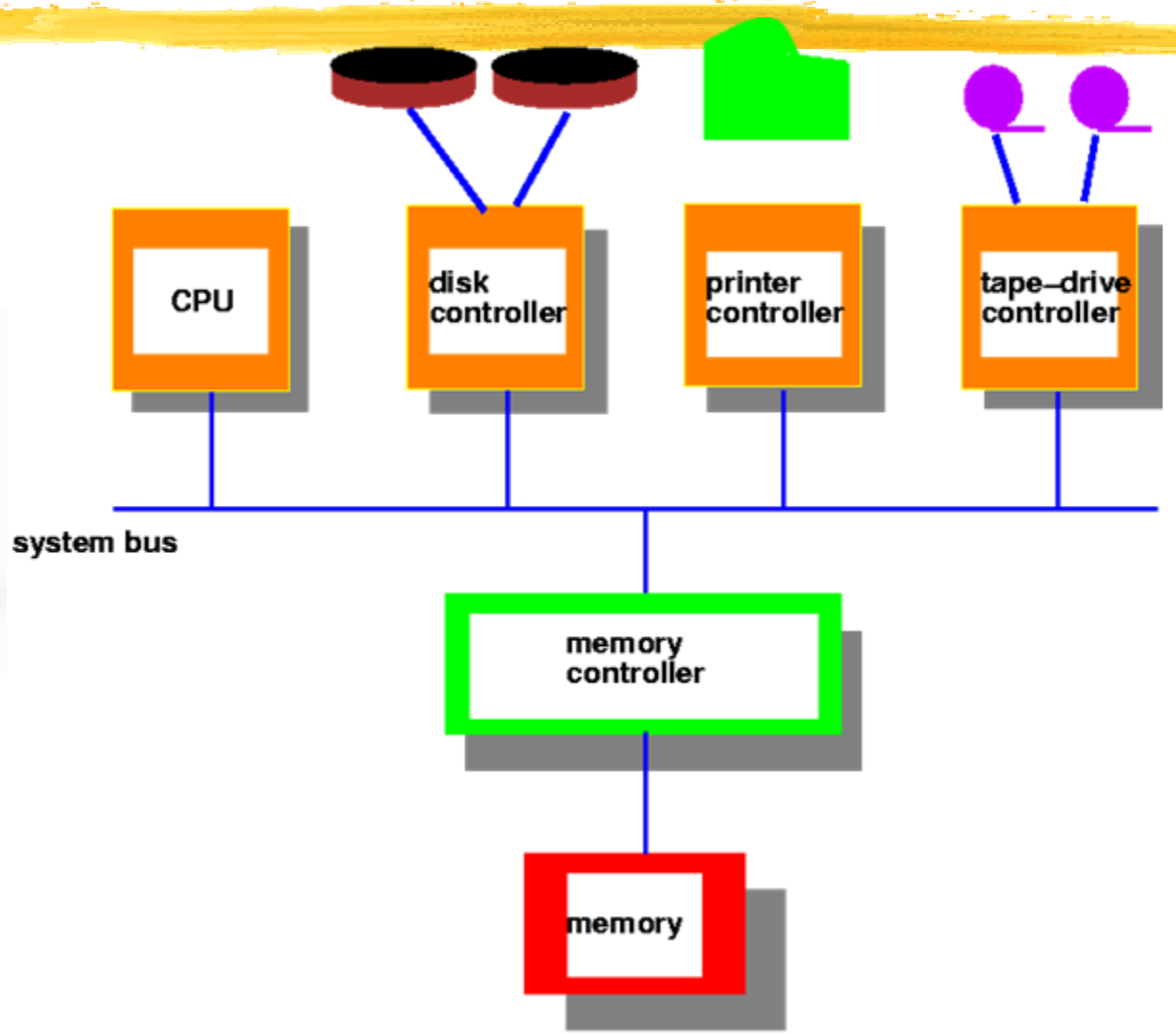
Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
  - Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability
    - error confinement, fault tolerance, reconfiguration.

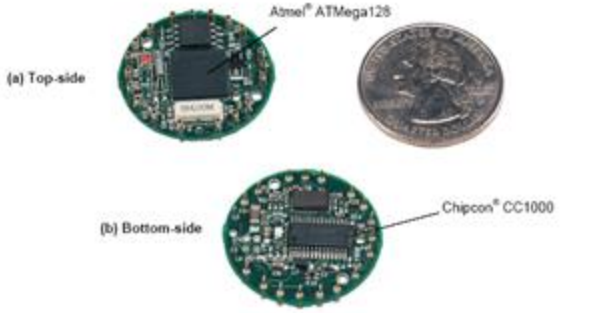
Why should I study Operating Systems?

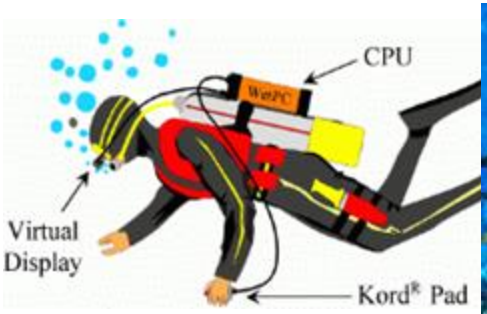
- Need to understand interaction between the hardware and applications
  - New applications, new hardware..
  - Inherent aspect of society today
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Increasing need for specialized operating systems
  - e.g. embedded operating systems for devices - cell phones, sensors and controllers
  - real-time operating systems - aircraft control, multimedia services

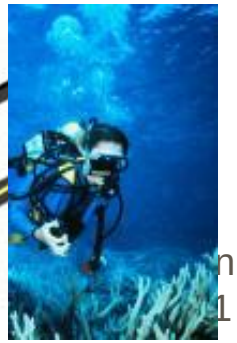
Computer System Architecture (traditional)



Systems Today

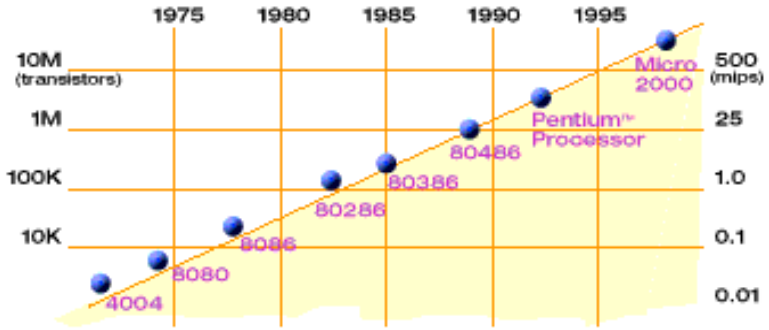


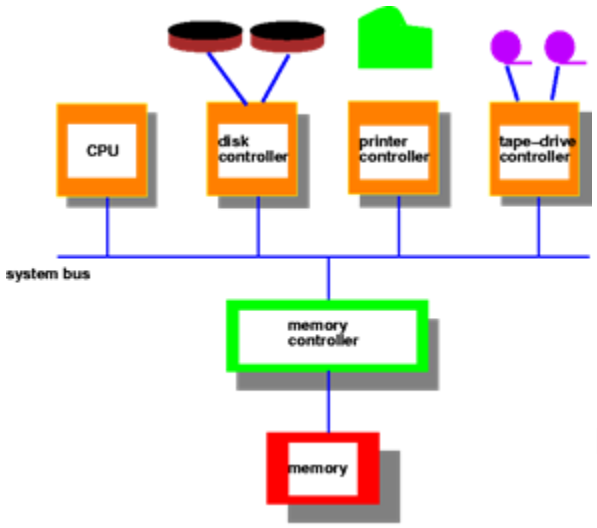




Hardware Complexity Increases

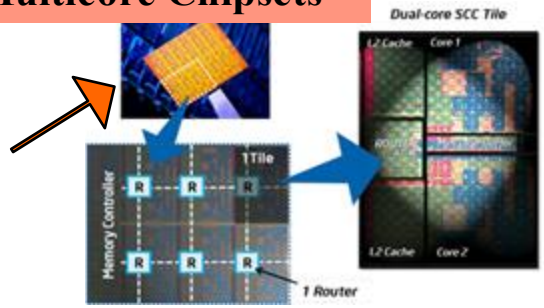
Moore's Law: 2X transistors/Chip Every 1.5 years

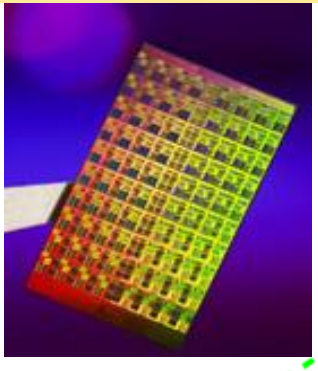


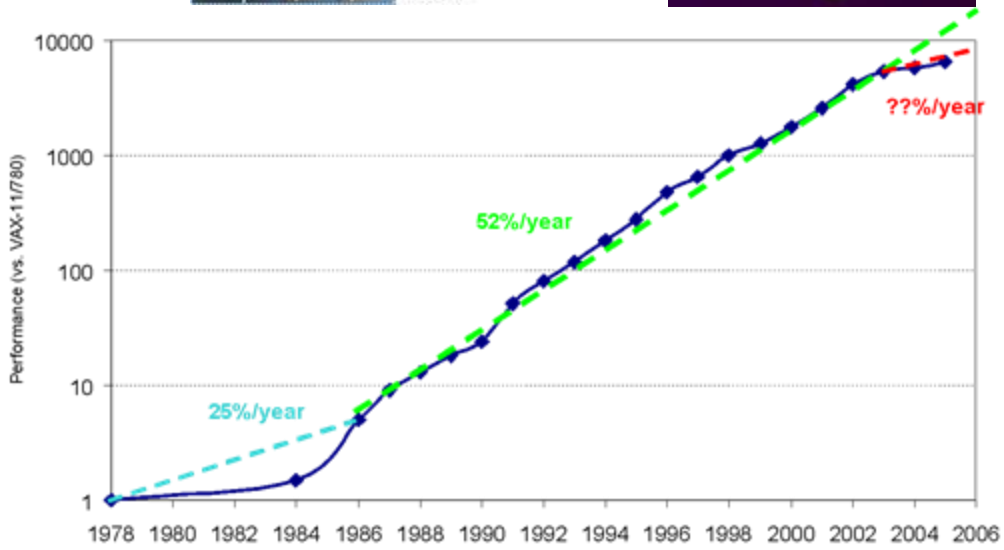


From Berkeley OS course

Intel Multicore Chipsets





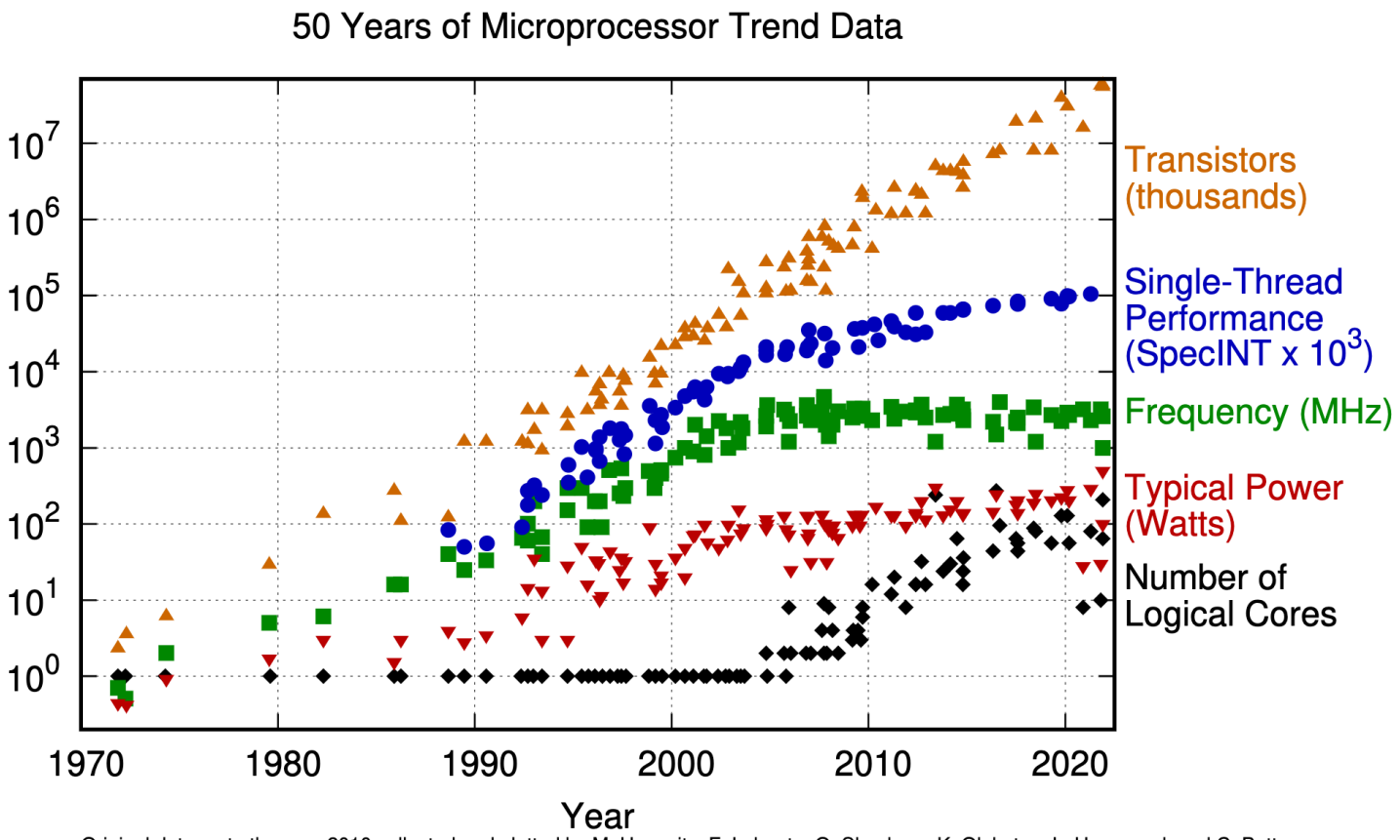


From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

OS needs to keep pace with hardware improvements

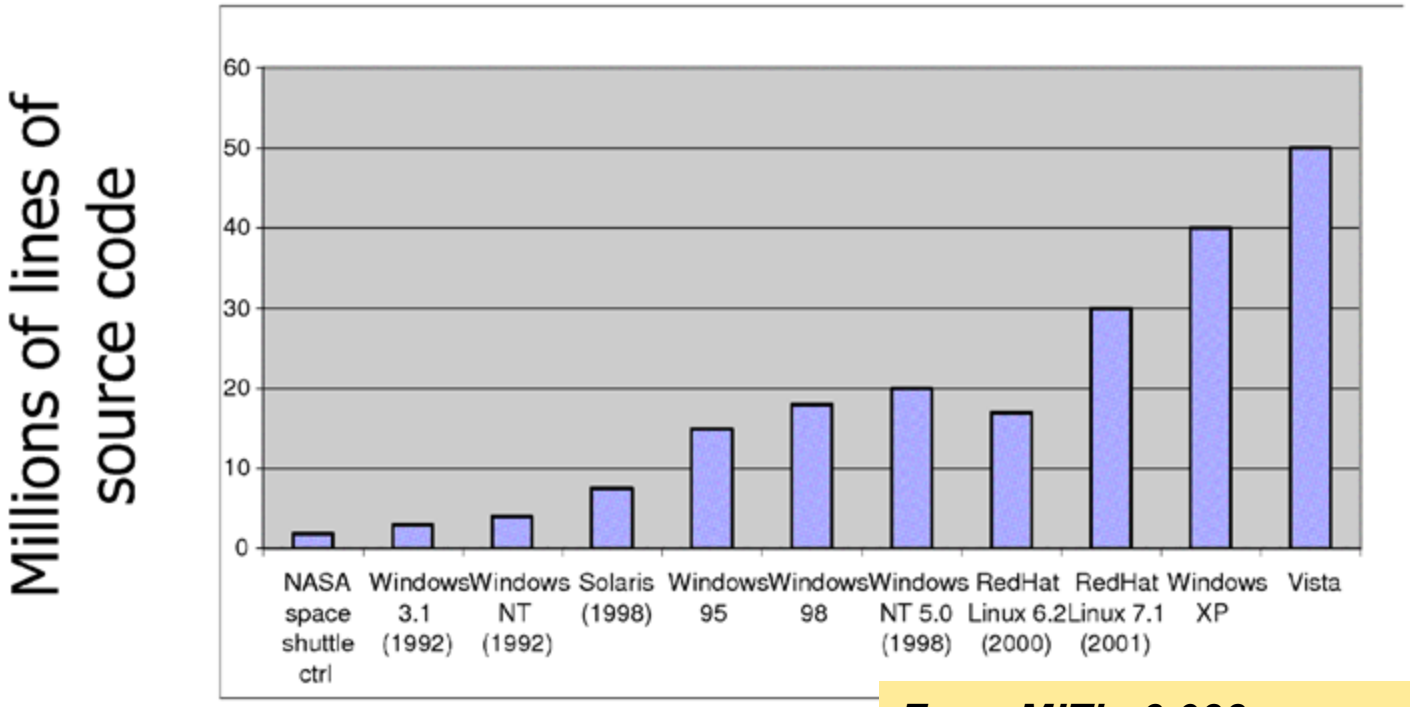
	1981	1997	2014	Factor (2014/1981)
Uniprocessor speed (MIPS)	1	200	2500	2.5K
CPUs per computer	1	1	10+	10+
$/Processor MIPS	$100K	$25	$0.20	500K
DRAM Capacity (MiB)/$	0.002	2	1K	500K
Disk Capacity (GiB)/$	0.003	7	25K	10M
Home Internet	300 bps	256 Kbps	20 Mbps	100K
Machine room network	10 Mbps (shared)	100 Mbps (switched)	10 Gbps (switched)	1000
Ratio of users to computers	100:1	1:1	1:several	100+

50 Years of Trend



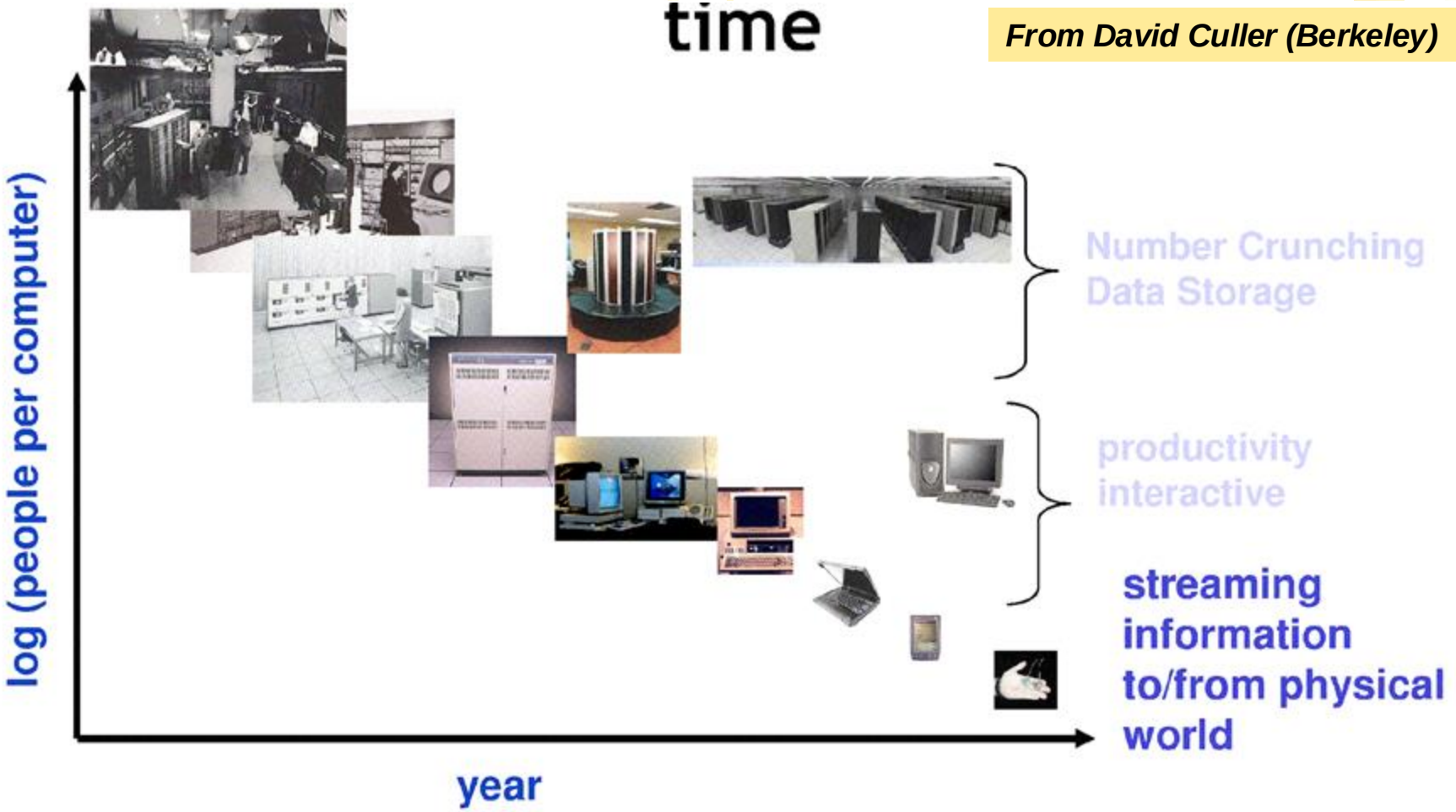
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

Software Complexity Increases



From MIT's 6.033 course

People-to-Computer Ratio Over Time



Operating System Spectrum

- Monitors and Small Kernels
  - special purpose and embedded systems, real-time systems
- Batch and multiprogramming
- Timesharing
  - workstations, servers, minicomputers, timeframes
- Transaction systems
- Personal Computing Systems
- Mobile Platforms, devices (of all sizes)

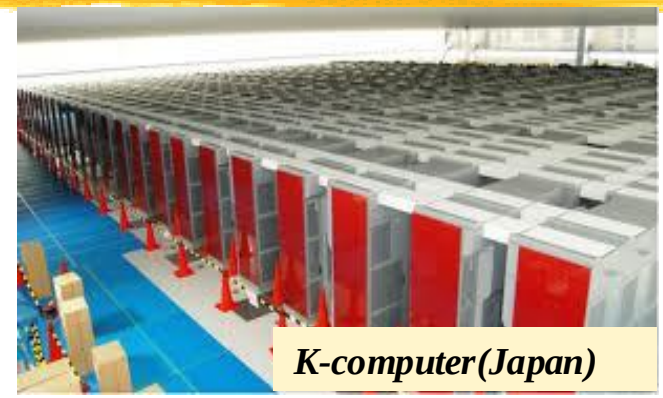
Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- Improved Throughput, economical, increased reliability.
- Kinds:
  - Vector and pipelined
  - Symmetric and asymmetric multiprocessing
  - Distributed memory vs. shared memory
- Programming models:
  - Tightly coupled vs. loosely coupled ,message-based vs. shared variable

Parallel Computing Systems

ILLIAC 2 (UIllinois)

Climate modeling, earthquake simulations, genome analysis, protein folding, nuclear fusion research, …..



K-computer(Japan)

IBM Blue Gene

Connection Machine (MIT)

Tianhe-1(China)

Distributed Systems

Hardware – *very cheap* ; Human – *very expensive*

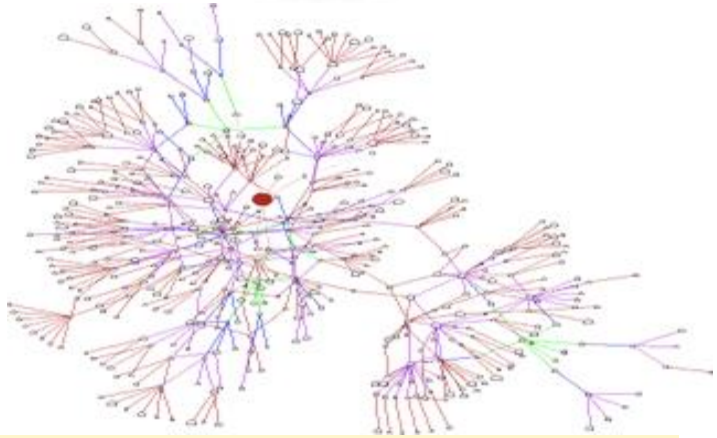
- Distribute computation among many processors.
- Loosely coupled -
  - no shared memory, various communication lines
- client/server architectures
- Advantages:
  - resource sharing
  - computation speed-up
  - reliability
  - communication - e.g. email
- Applications - digital libraries, digital multimedia

Distributed Computing Systems

Globus Grid Computing Toolkit

PlanetLab

Cloud Computing Offerings



Gnutella P2P Network

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too long.
  - Useful in applications such as multimedia, virtual reality.

Operating systems are everywhere

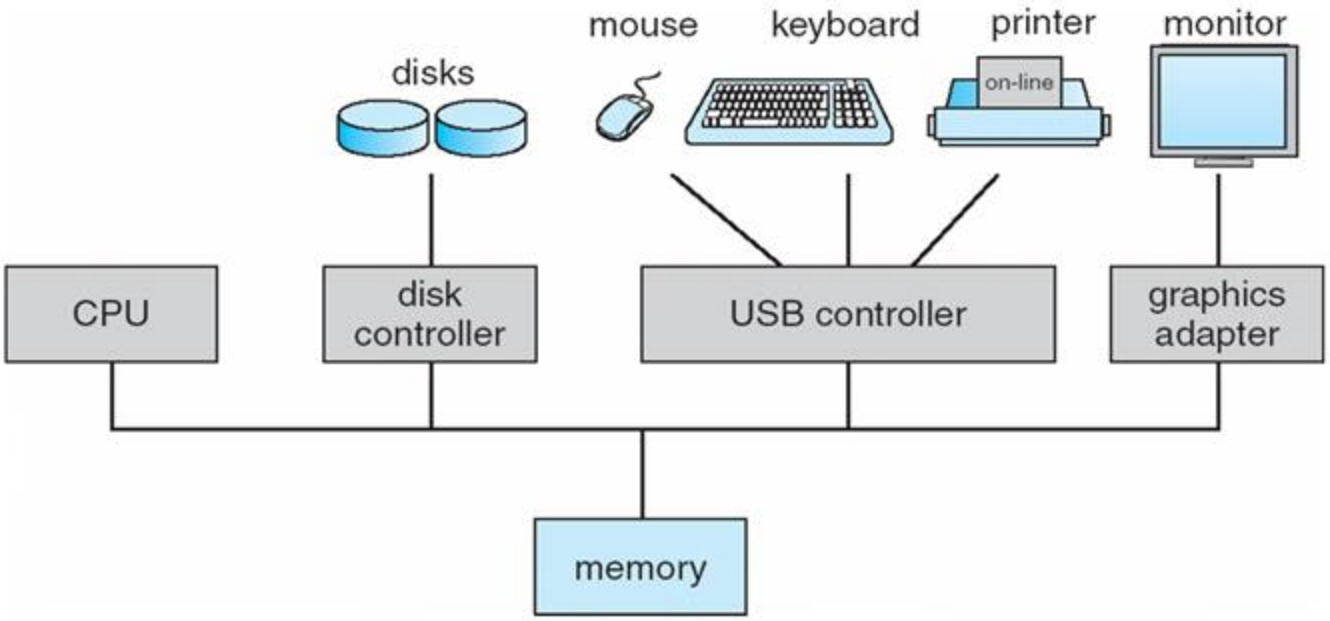
Summary of lecture

- What is an operating system?
- Early Operating Systems
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems

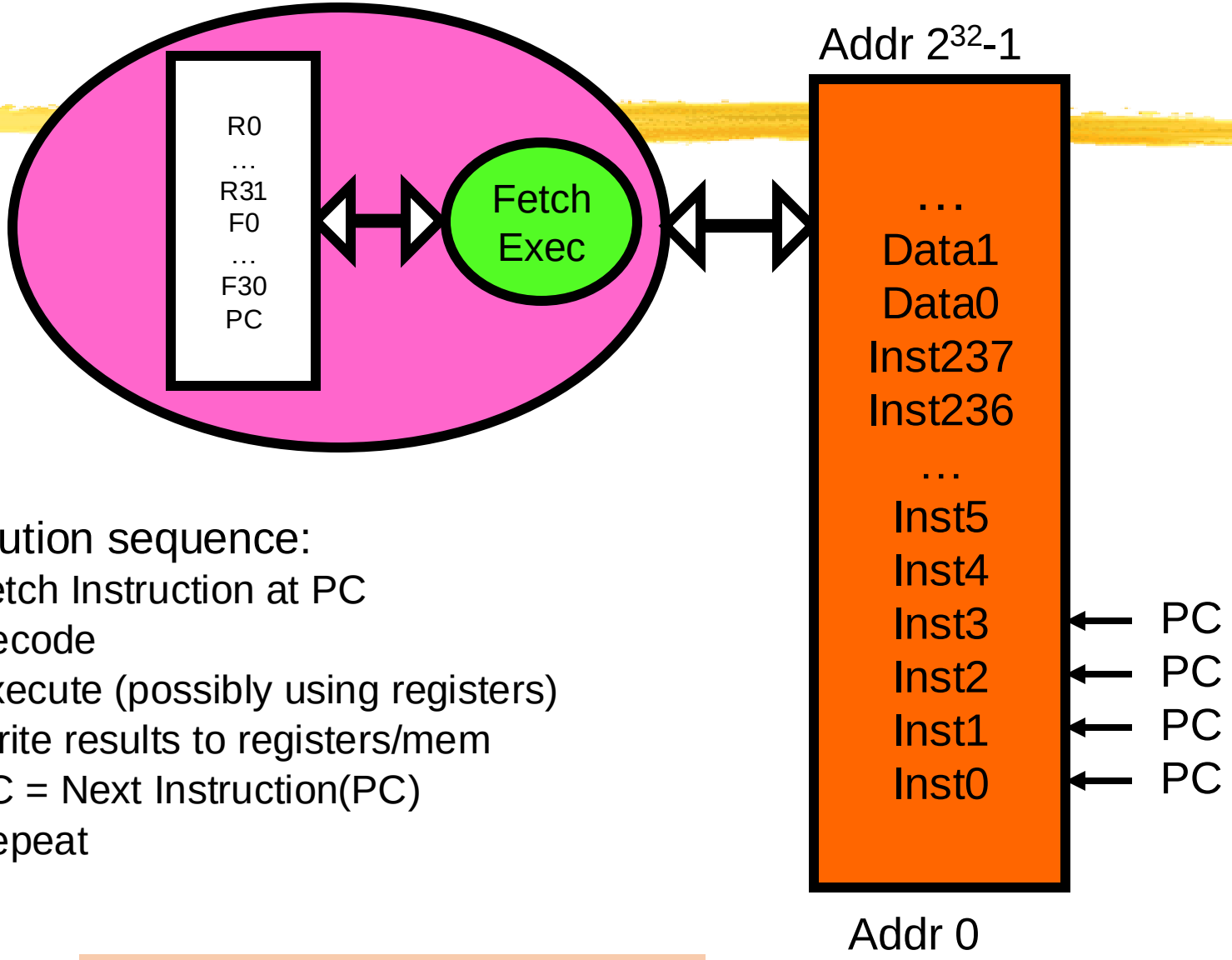
Computer System & OS Structures

- Computer System Organization
- Operational Flow and hardware protection
- System call and OS services
- Storage architecture
- OS organization
- OS tasks
- Virtual Machines

Computer System **Organization**



CPU execution



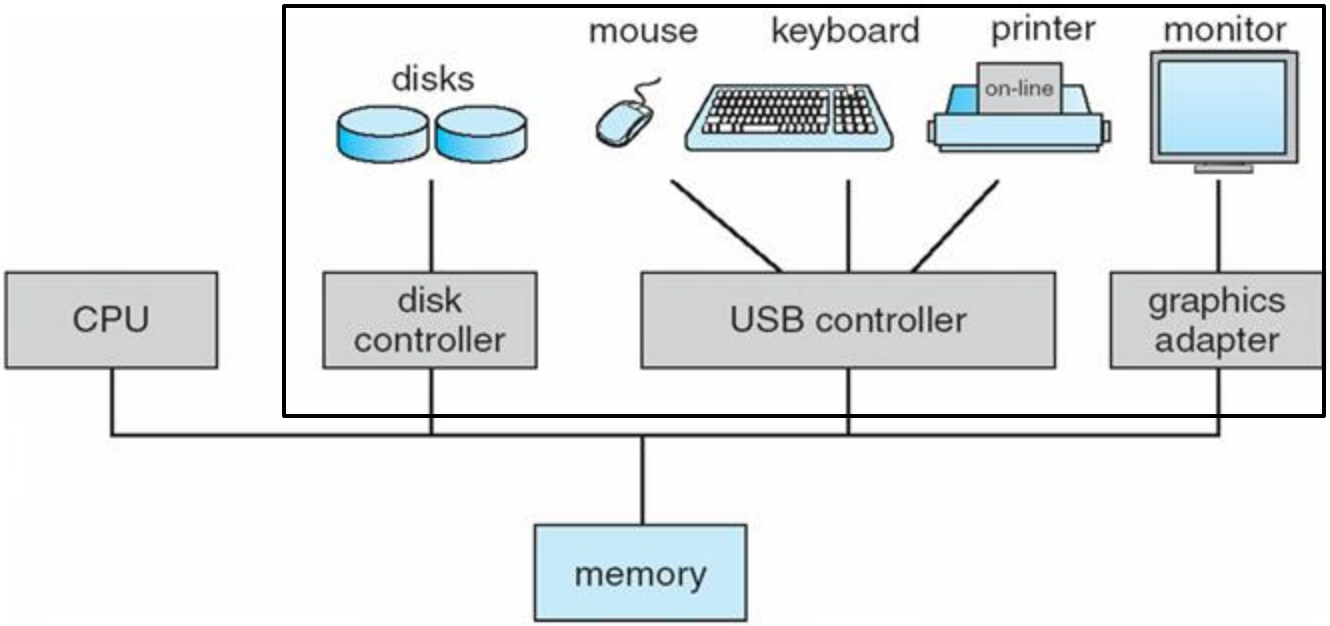
Execution sequence:

- Fetch Instruction at PC
- Decode
- Execute (possibly using registers)
- Write results to registers/mem
- PC = Next Instruction(PC)
- Repeat

From Berkeley OS course

Computer System **Organization**

I/O devices



I/O devices

- I/O devices and the CPU execute concurrently.
- Each device controller is in charge of a particular device type
  - Each device controller has a local buffer. I/O is from the device to local buffer of controller
- CPU moves data from/to main memory to/from the local buffers

Interrupts

- **Interrupt** transfers control to the **interrupt service routine**
  - Interrupt Service Routine: Segments of code that determine action to be taken for interrupt.
- Determining the type of interrupt
  - Polling: same interrupt handler called for all interrupts, which then polls all devices to figure out the reason for the interrupt
  - Interrupt Vector Table: different interrupt handlers will be executed for different interrupts

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

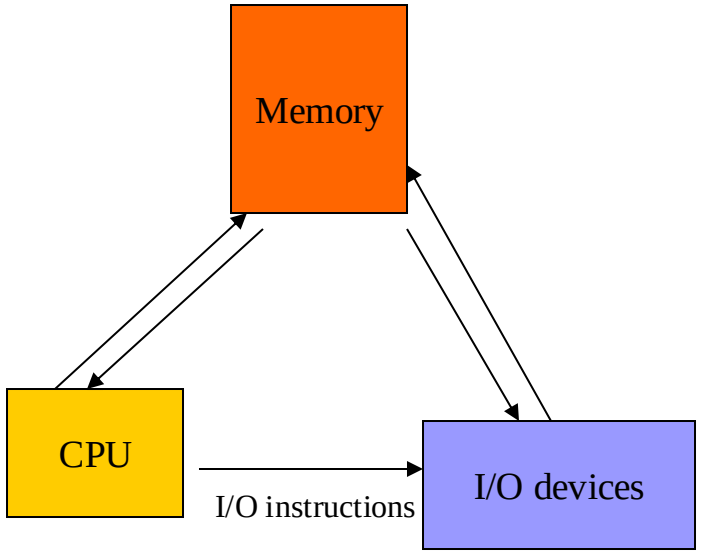
Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

Interrupt **handling**

- OS preserves the state of the CPU
  - stores registers and the program counter (address of interrupted instruction).
- What happens to a new interrupt when the CPU is handling one interrupt?
  - Incoming interrupts can be disabled while another interrupt is being processed. In this case, incoming interrupts may be lost or may be buffered until they can be delivered.
  - Incoming interrupts can be masked (i.e., ignored) by software.
  - Incoming interrupts are delivered, i.e., nested interrupts.

Direct Memory Access (DMA)

- Typically used for I/O devices with a lot of data to transfer (in order to reduce load on CPU).
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Device controller interrupts CPU on completion of I/O



Process Abstraction and rights

- Process: an *instance* of a program, running with limited rights
- Address space: set of rights of a process
  - Memory that the process can access
- Other permissions the process has (e.g., which system calls it can make, what files it can access)

Hardware Protection

- CPU Protection:
  - Dual Mode Operation
  - Timer interrupts
- Memory Protection
- I/O Protection

Should a process be able to execute any instructions?

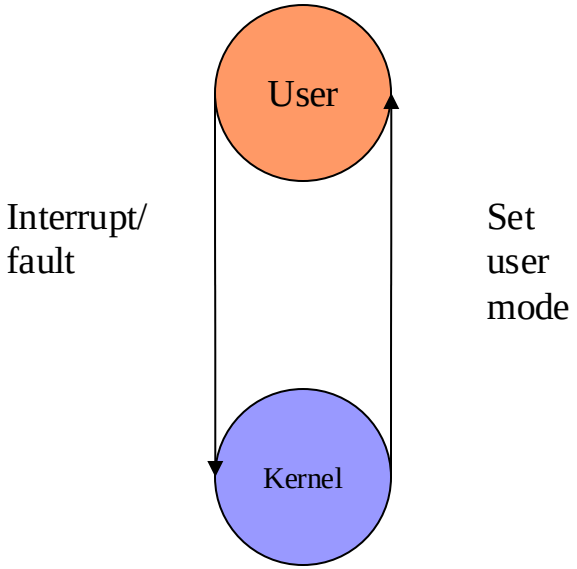
- No
  - Can alter system configuration
  - Can access unauthorized memory
  - Can access unauthorized I/O
  - etc.
- How to prevent?

Dual-mode operation

- Provide hardware support to differentiate between at least two modes of operation:
  1. User mode -- execution done on behalf of a user.
  2. Kernel mode (monitor/supervisor/system mode) -- execution done on behalf of operating system.
- "Privileged" instructions are only executable in the kernel mode
- Executing privileged instructions in the user mode "traps" into the kernel mode

Dual-mode operation(cont.)

- Mode bit added to computer hardware to indicate the current mode: kernel(0) or user(1).
- When an interrupt or trap occurs, hardware switches to kernel mode.



CPU Protection

- How to prevent a process from executing indefinitely?

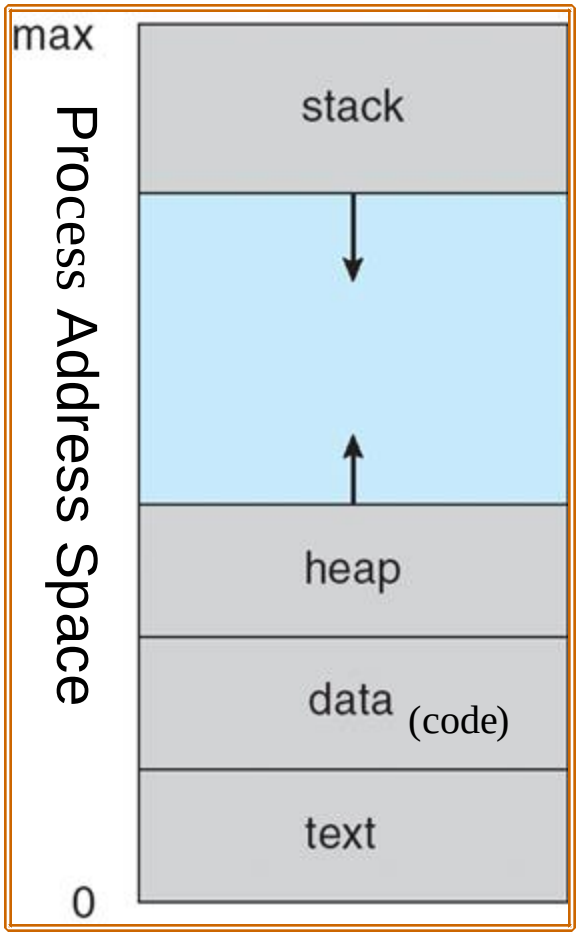
CPU Protection

- Timer - interrupts computer after specified period to ensure that OS maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches a value of 0, an interrupt occurs.
- Timer is commonly used to implement time sharing.
- Timer is also used to compute the current time.
- Programming the timer can only be done in the kernel since it requires privileged instructions.

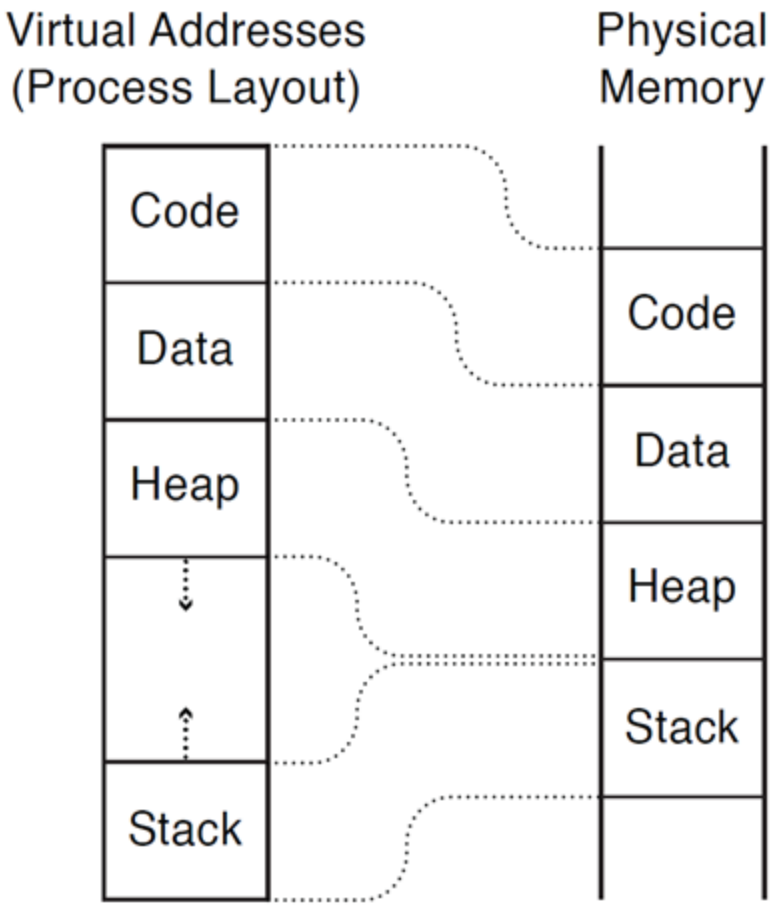
How to isolate memory access?

Process address space

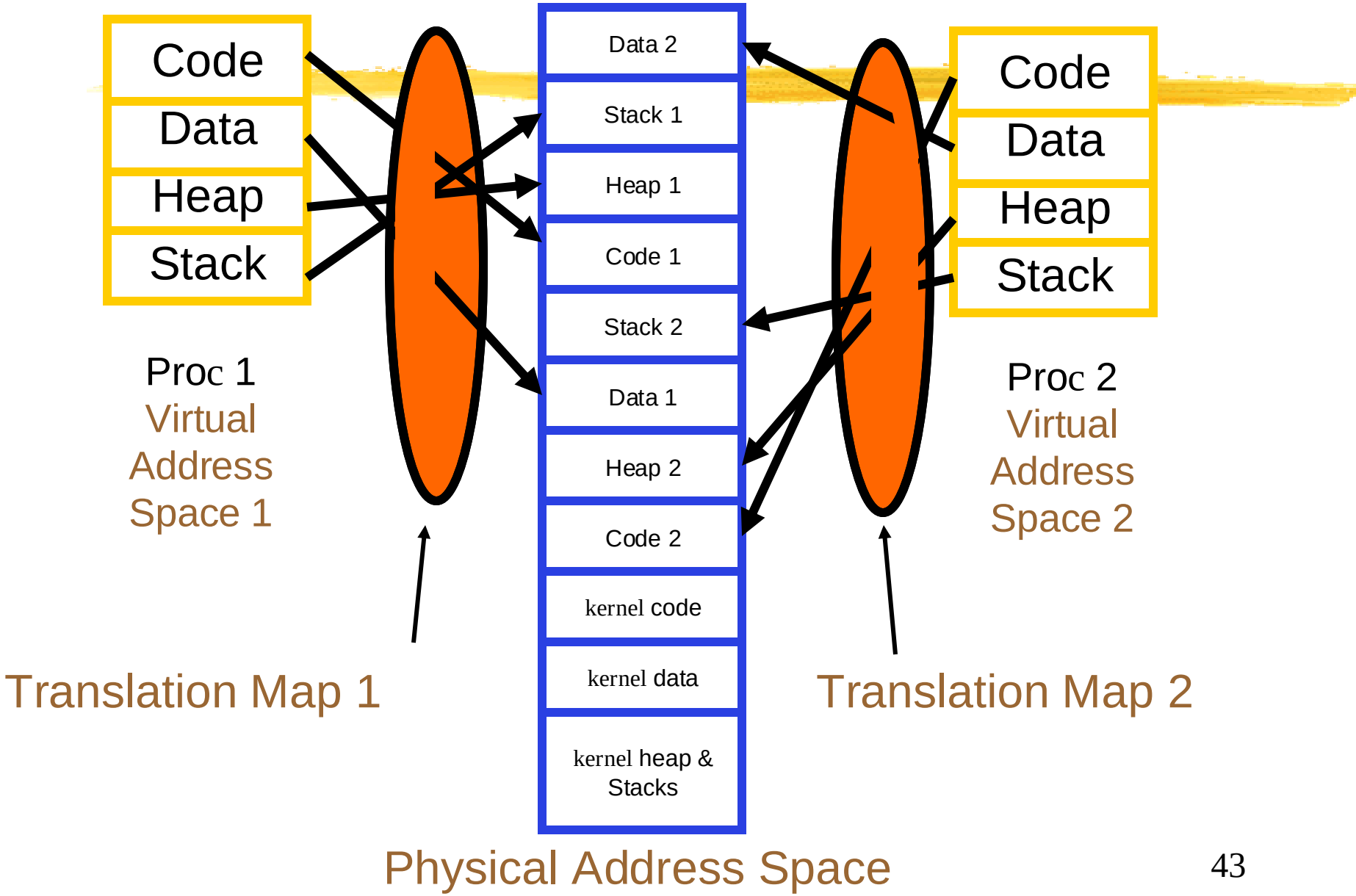
- Address space ⇒ the set of accessible addresses :
  - For a 32-bit processor there are 2^{32} = 4 billion addresses
  - For a 64-bit processor there are 2^{64} = many many addresses



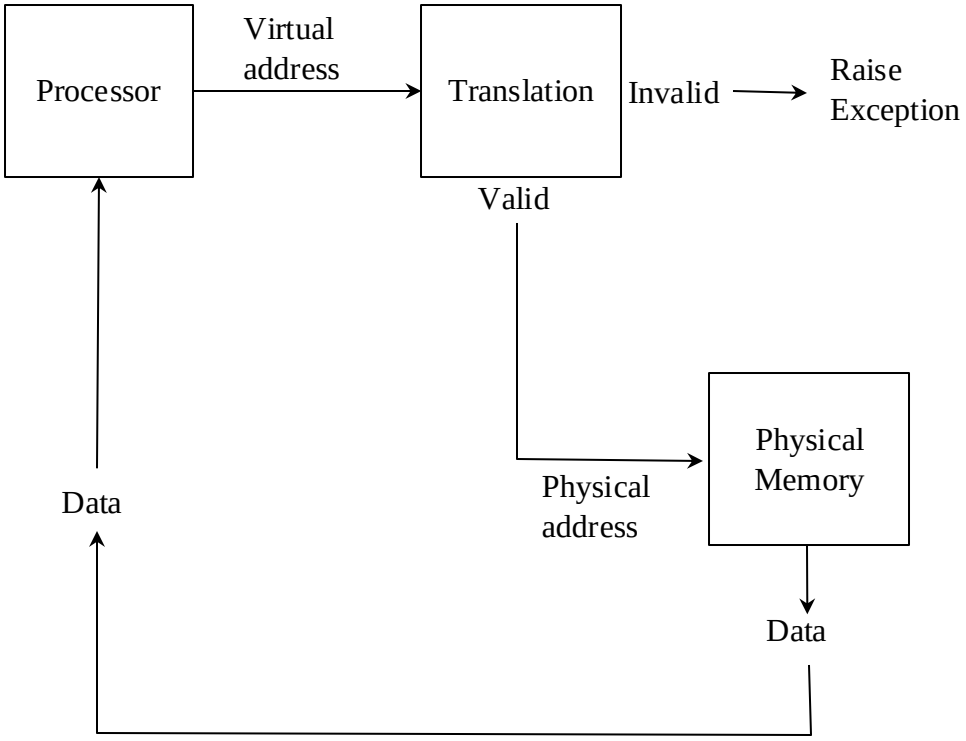
Virtual **Address**



Providing the Illusion of Separate Address Spaces



Address translation and memory protection

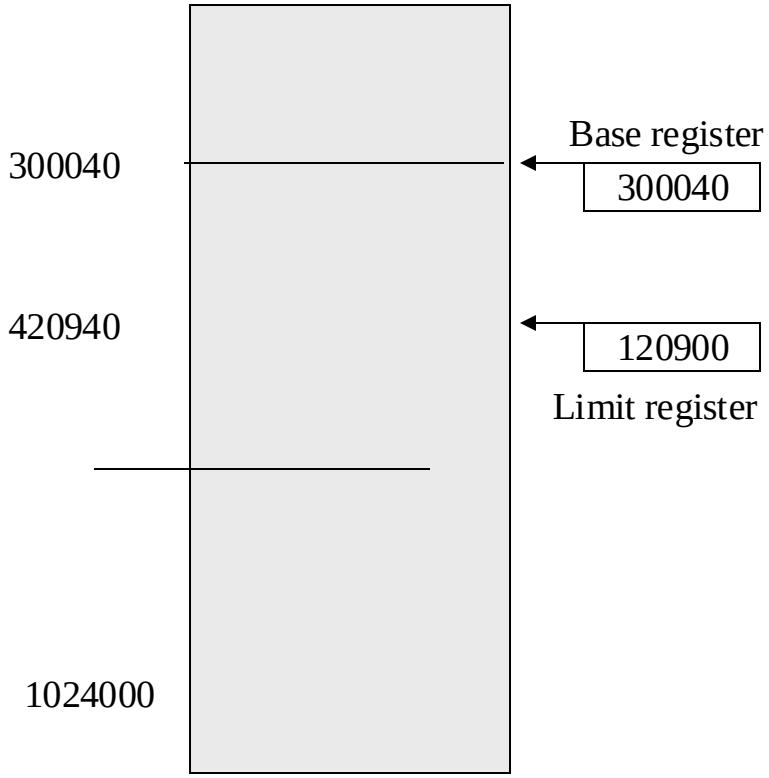


Memory Protection

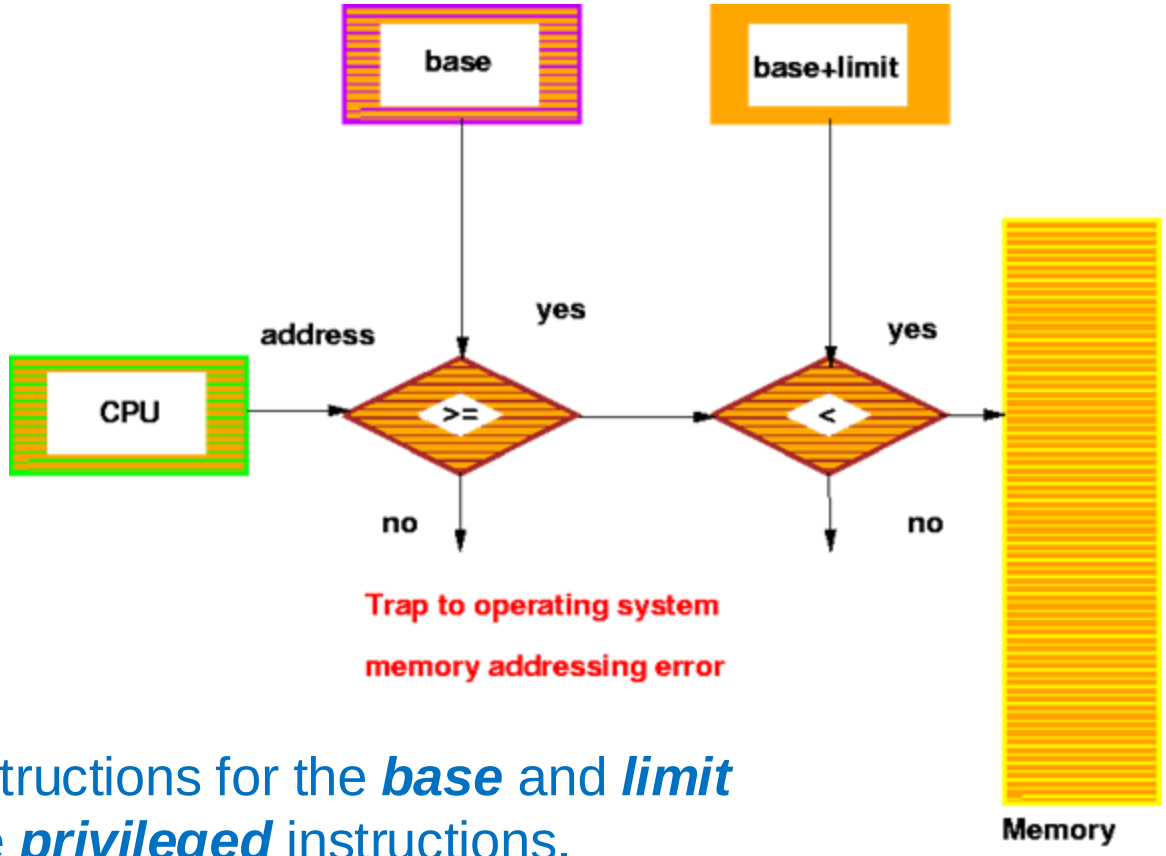
- When a process is running, only memory in that process address space must be accessible.
- When executing in kernel mode, the kernel has unrestricted access to all memory.

Memory Protection: **base and limit**

- To provide memory protection, add two registers that determine the range of legal addresses a program may address.
  - **Base Register** - holds smallest legal physical memory address.
  - **Limit register** - contains the size of the range.
- Memory outside the defined range is protected.
- Sometimes called ***Base and Bounds*** method

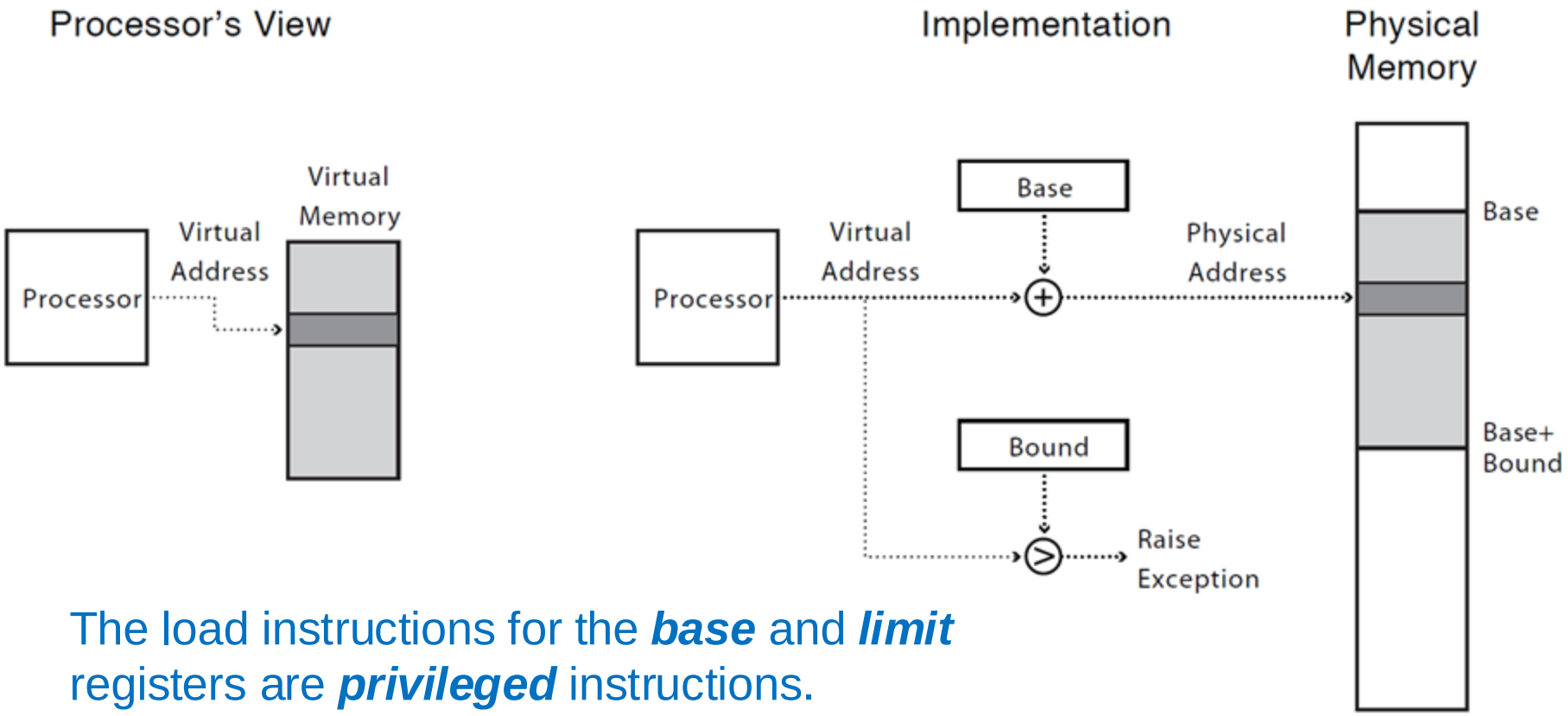


Hardware Address Protection



The load instructions for the ***base*** and ***limit*** registers are ***privileged*** instructions.

Virtual Address translation using the Base and Bounds method



The load instructions for the ***base*** and ***limit*** registers are ***privileged*** instructions.

I/O Protection

- All I/O instructions are privileged instructions.

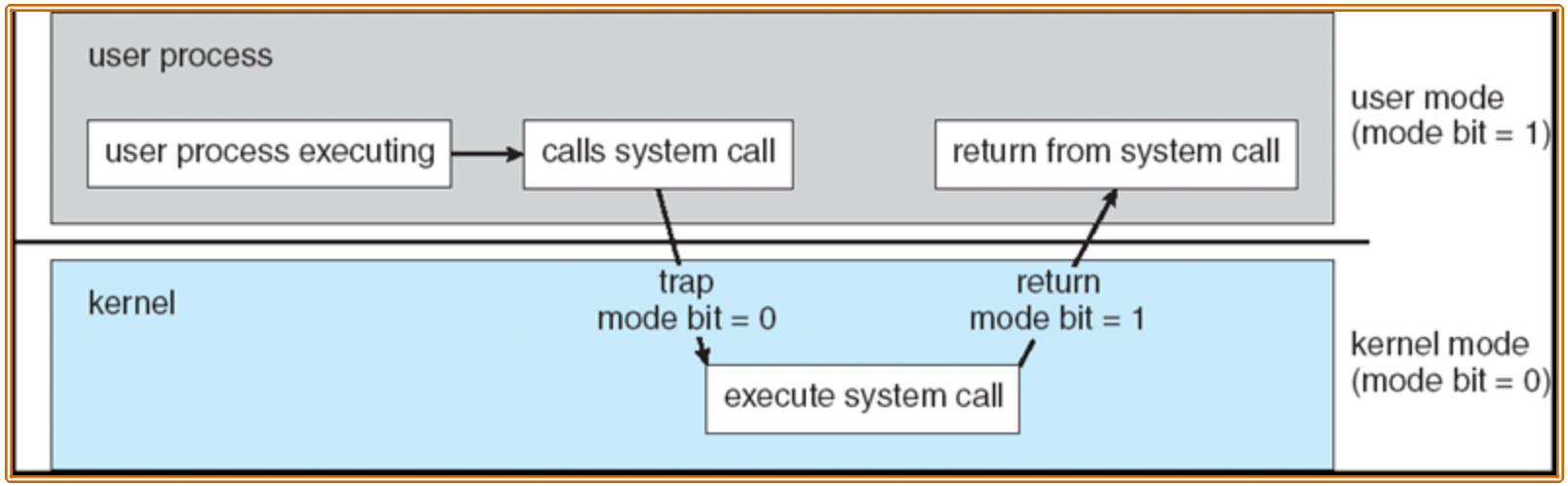
Question

- Given the I/O instructions are privileged, how do users perform I/O?

- Via system calls - the method used by a process to request action by the operating system.

System Calls

- User code can issue a syscall, which causes a trap
- Kernel handles the syscall

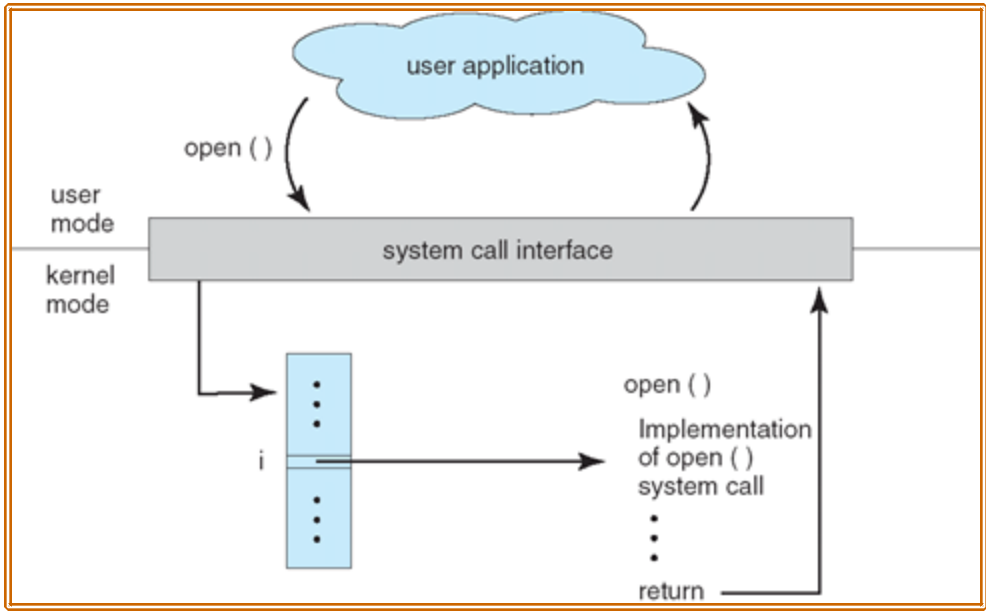


System Calls

- Interface between applications and the OS.
  - Application uses an assembly instruction to trap into the kernel
  - Some higher level languages provide wrappers for system calls (e.g., C)

.

- Linux has about 300 system calls
  - read(), write(), open(), close(), fork(), exec(), ioctl(),…..



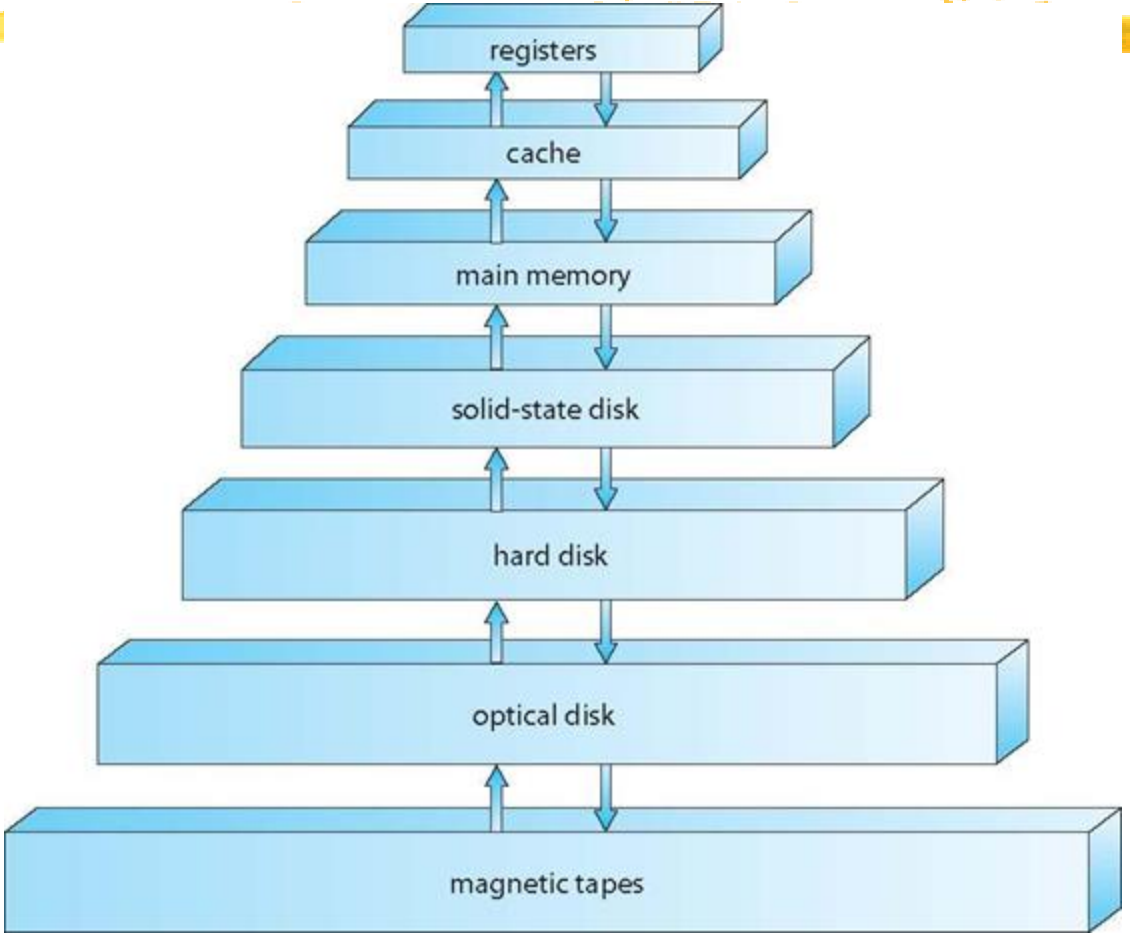
Storage Structure

- Main memory - only large storage media that the CPU can access directly.
- Secondary storage - has large nonvolatile storage capacity.
  - Magnetic disks - rigid metal or glass platters covered with magnetic recording material.
    - Disk surface is logically divided into tracks, subdivided into sectors.
    - Disk controller determines logical interaction between device and computer.

Storage Hierarchy

- Storage systems are organized in a hierarchy based on
  - Speed
  - Cost
  - Volatility
- Caching - process of copying information into faster storage system; main memory can be viewed as fast cache for secondary storage.

Storage Device Hierarchy



OS Task: Process Management

- Process - fundamental concept in OS
  - Process is an instance of a program in execution.
  - Process needs resources - CPU time, memory, files/data and I/O devices.
- OS is responsible for the following process management activities.
  - Process creation and deletion
  - Process suspension and resumption
  - Process synchronization and interprocess communication
  - Process interactions - deadlock detection, avoidance and correction

OS Task: Memory Management

- Main Memory is an array of addressable words or bytes that is quickly accessible.
- Main Memory is volatile.
- OS is responsible for:
  - Allocate and deallocate memory to processes.
  - Managing multiple processes within memory - keep track of which parts of memory are used by which processes. Manage the sharing of memory between processes.
  - Determining which processes to load when memory becomes available.

OS Task: Secondary Storage and I/O Management

- Since primary storage (i.e., main memory) is expensive and volatile, secondary storage is required for backup.
- Disk is the primary form of secondary storage.
  - OS performs storage allocation, free-space management, etc.
- I/O system in the OS consists of
  - Device driver interface that abstracts device details
  - Drivers for specific hardware devices

OS Task: File System Management

- File is a collection of related information - represents programs and data.

- OS is responsible for
  - File creation and deletion
  - Directory creation and deletion
  - Supporting primitives for file/directory manipulation.
  - Mapping files to disks (secondary storage).

OS Task: Protection and Security

- Protection mechanisms control access of processes to user and system resources.
- Protection mechanisms must:
  - Distinguish between authorized and unauthorized use.
  - Specify access controls to be imposed on use.
  - Provide mechanisms for enforcement of access control.

Original UNIX System Structure

- Limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management