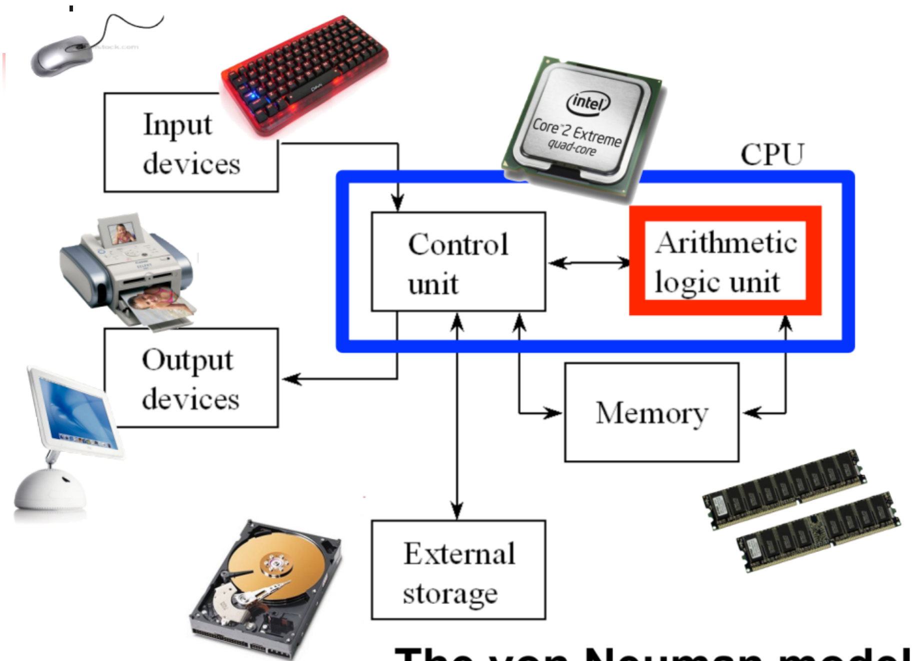
Chapter 2

COMPUTER SYSTEM & OS STRUCTURES

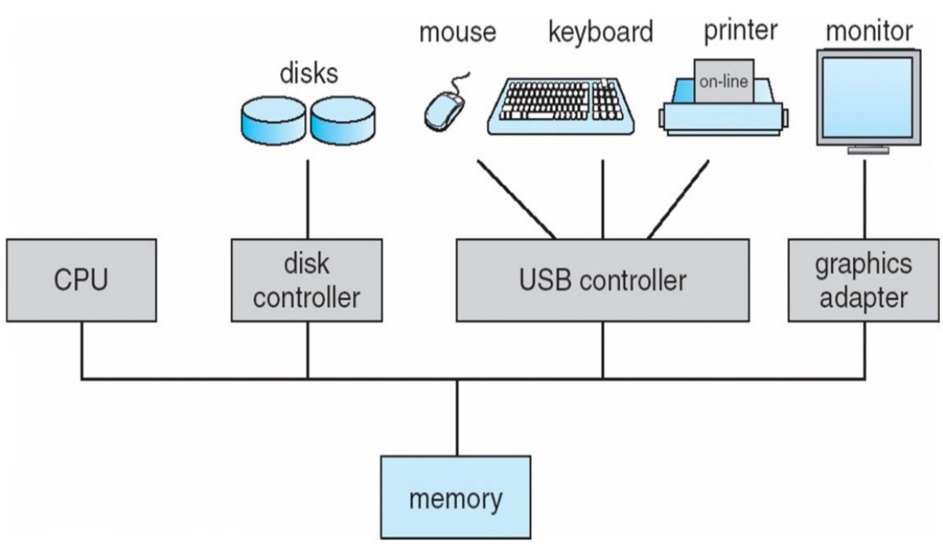
Content

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



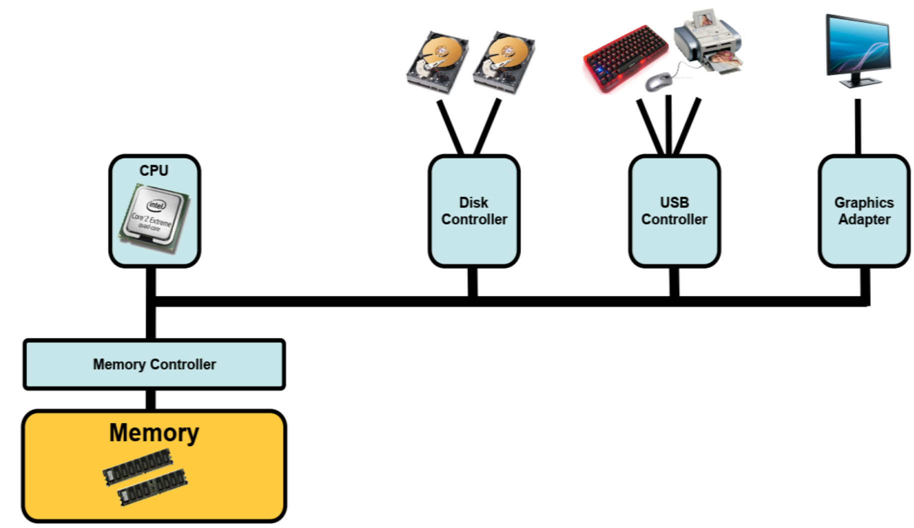
The von Neuman modell

Computer System Organization

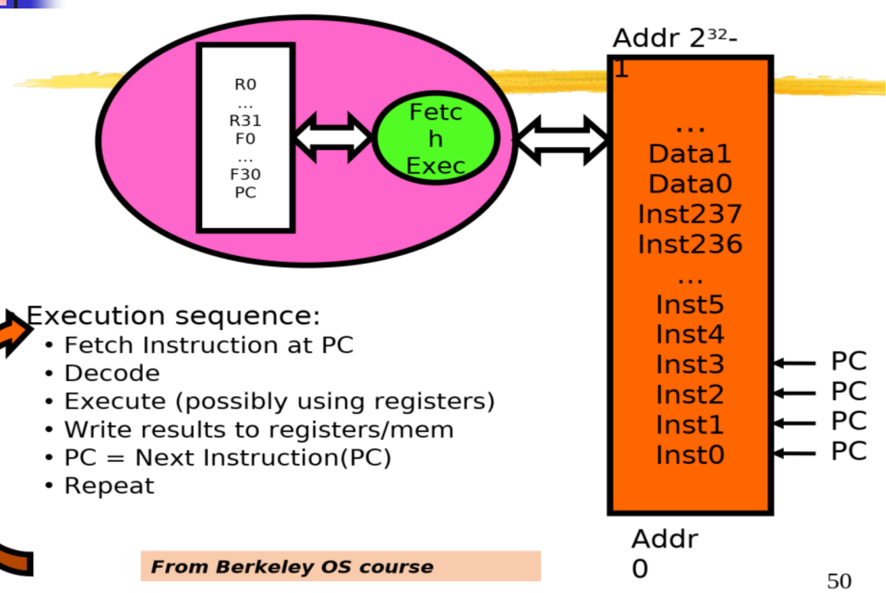


Đinh Công Đoan 4

A typical modern computer system



CPU execution



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

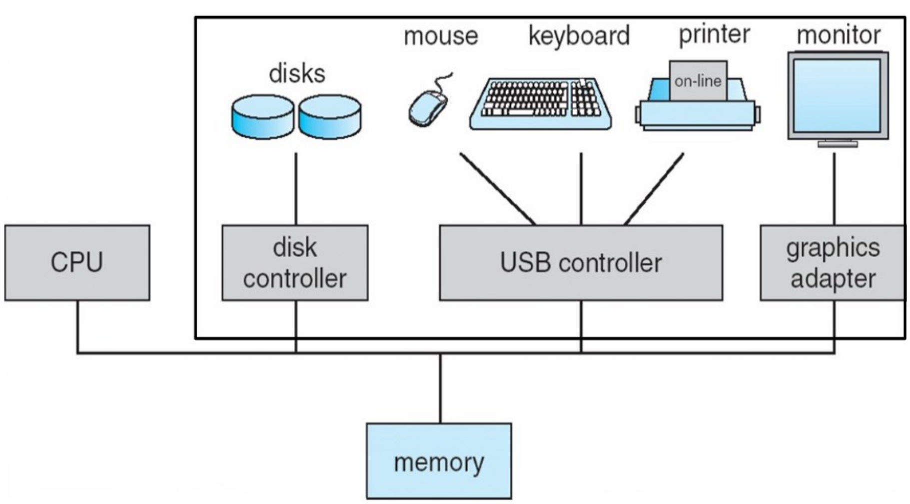
From Berkeley OS course

Computer – system 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

Computer System Organization

I/O devices



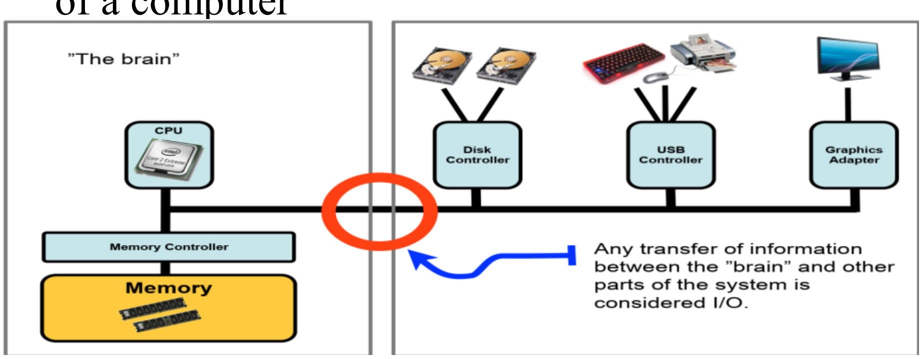
I/O devices

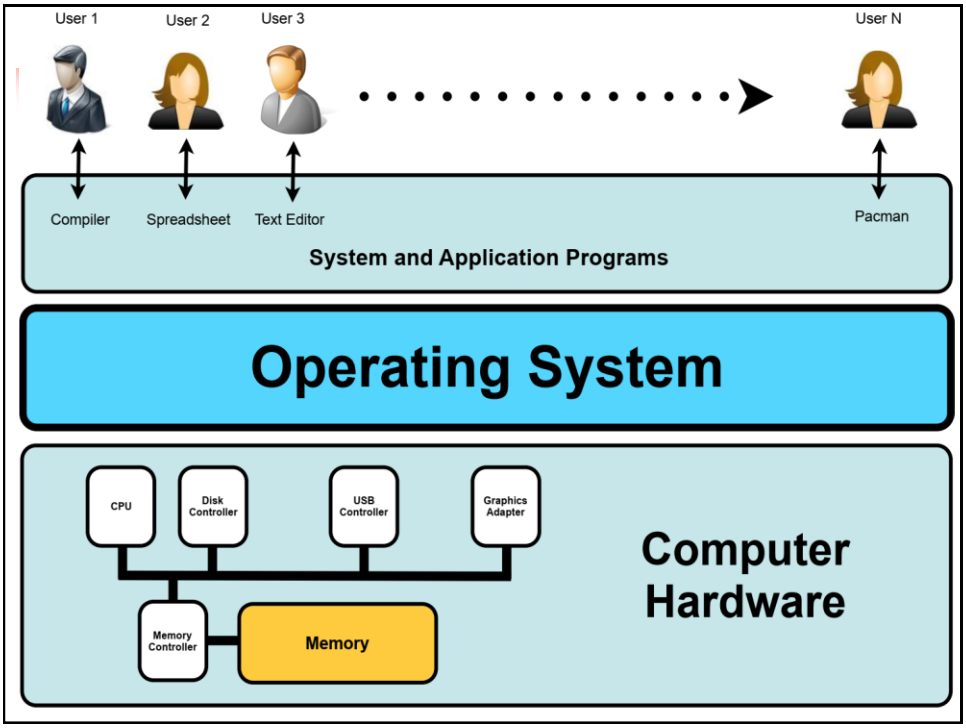
- I/O devices and the CPU execute concurrently.
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer. I/O is from the device to local buffer of controller
- CPU moves data from/to main memory to/from the local buffers

Đinh Công Đoan 9

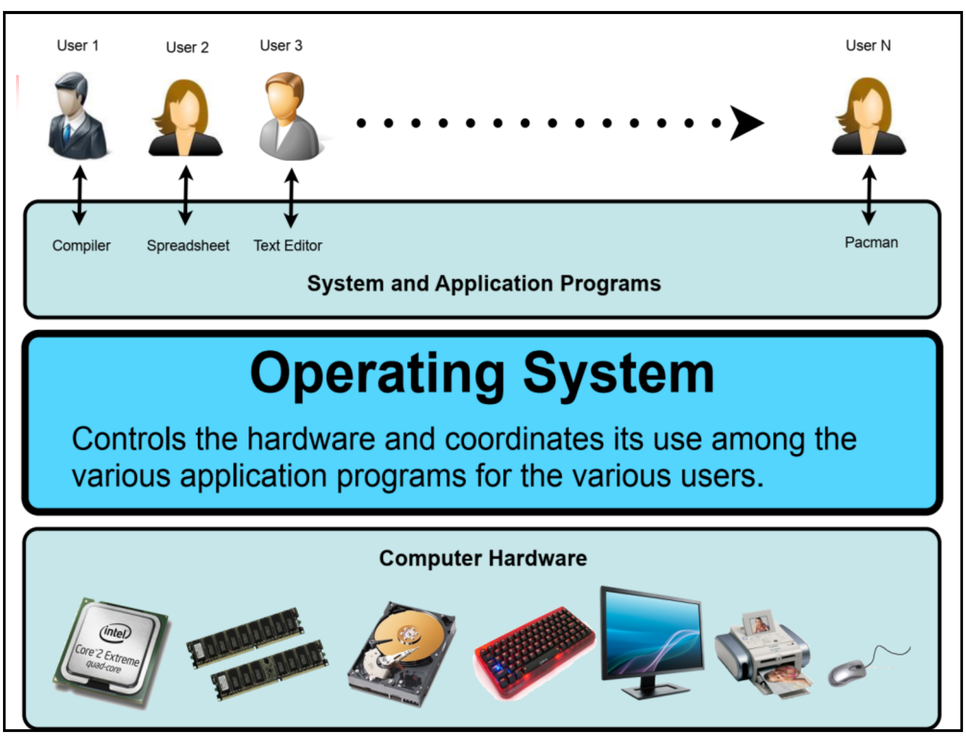
Input and Output (I/O)

- In computer architecture, the combination of the CPU and main memory (i.e. memory that the CPU can read and write to directly, with individual instructions) is considered the "brain" of a computer





User 1
User 2
User 3
User N
Compiler
Spreadsheet
Text Editor
Pacman
System and Application Programs
Operating System
CPU
Disk Controller
USB Controller
Graphics Adapter
Memory Controller
Memory
Computer Hardware



User 1
User 2
User 3
User N
Compiler
Spreadsheet
Text Editor
Pacman
System and Application Programs
Operating System
Controls the hardware and coordinates its use among the various application programs for the various users.
Computer Hardware

Important definitions

- **CPU**
  The central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.
- **Register**
  A processor register is a quickly accessible location available to a computer's central processing unit (CPU). Registers usually consist of a small amount of fast storage. A CPU only has a small number of registers.
- **Memory**
  Memory refers to the computer hardware integrated circuits that store information for immediate use in a computer; it is synonymous with the term "primary storage". The memory is much slower than the CPU register but much larger in size

- **CPU context** : At any point in time, the values of all the registers in the CPU defines the CPU context. Sometimes CPU state is used instead of CPU context.
- **Program** : A set of instructions which is in human readable format. A passive entity stored on secondary storage.
- **Executable** : A compiled form of a program including machine instructions and static data that a computer can load and execute. A passive entity stored on secondary storage

- **Process:** A program loaded into memory and executing or waiting. A process typically executes for only a short time before it either finishes or needs to perform I/O (waiting). A process is an active entity and needs resources such as CPU time, memory etc to execute.
- Kernel : The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system

System and Application Programs

Operating System

Controls the hardware and coordinates its use among the various application programs for the various users.

Bootstrap program

Kept on chip (ROM or EEPROM), aka **firmware**.

Small program executed on power up or reboot.

Initializes all aspects of the system, from CPU register to device controllers to memory content.

Locates and **loads the kernel into memory** for execution.

Kernel

The part of the operating system that is running at all times.

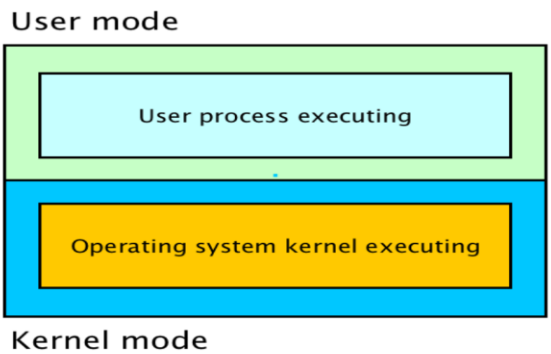
On boot, starts executing the first process such as **init**.

Waits for some **event** to occur...

Computer Hardware

Dual mode operation

- In order to protect the operating system from user processes and protect user processes from each other, two modes are provided by the hardware: **user mode** and **kernel mode**.



Dual mode operation place restrictions on the type and scope of operations that can be executed by the CPU. This design allows the operating system kernel to execute with more privileges than user application processes.

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

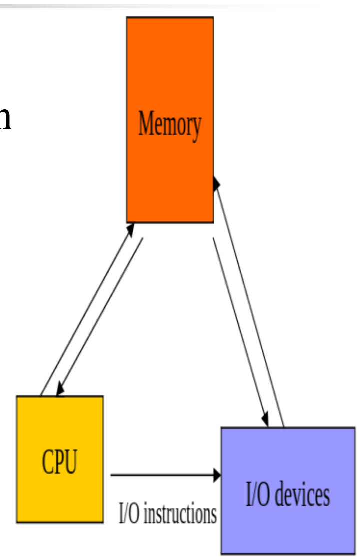
Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

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

- 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

How to limit process rights?

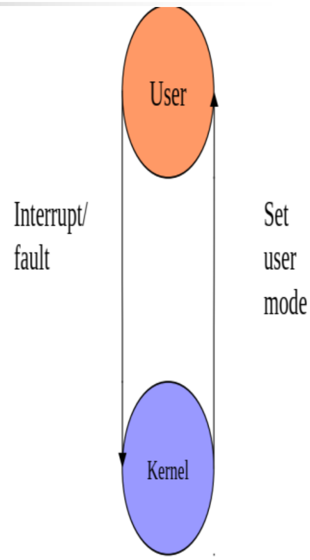
- Should a process be able to execute any instructions?
- No
  - Can alter system configuration
  - Can access unauthorized memory
  - Can access unauthorized I/O
  - etc.
- How to prevent?

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

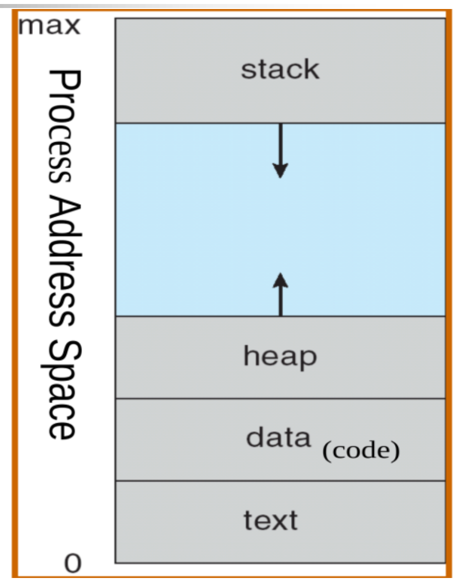


CPU Protection

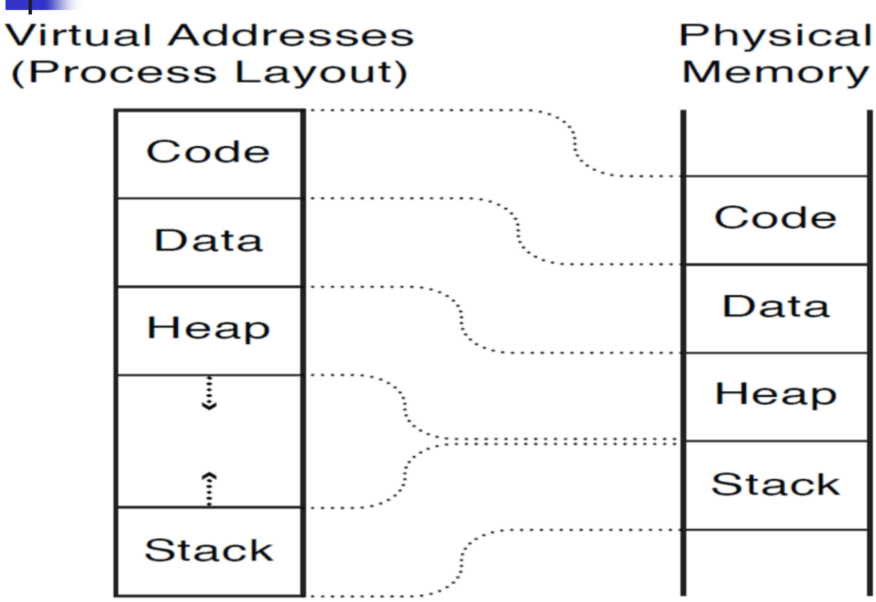
- How to prevent a process from executing indefinitely?
- Timer - interrupts computer after specified period to ensure that OS maintains control.
- Timer is decremented every clock tick.
- When timer reaches a value of 0, an interrupt occurs.
- Timer is commonly used to implement time sharing.
- Timer 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?

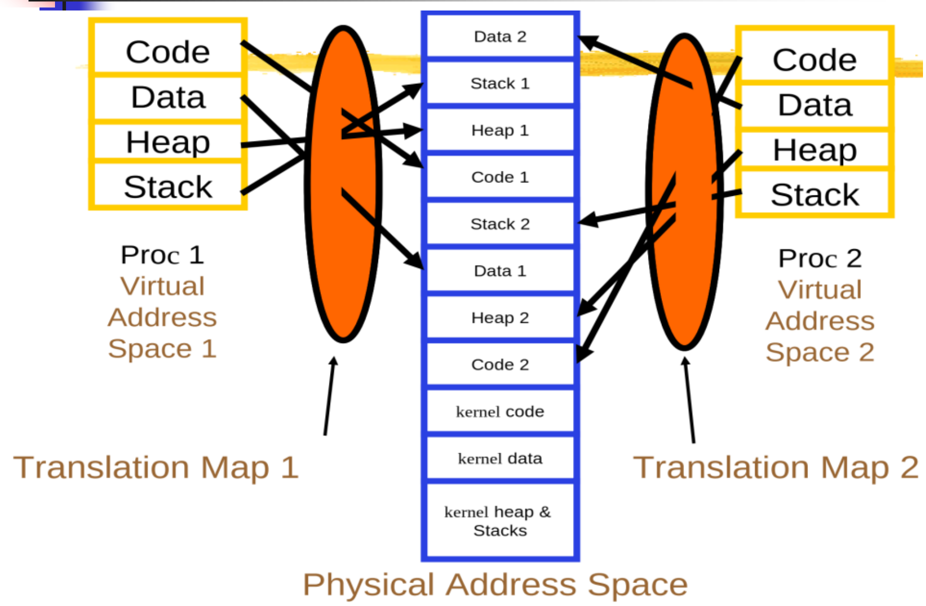
- Process address space
  - Address space ⇒ the set of accessible addresses + state associated with them:
  - For a 32-bit processor there are 2^{32} = 4 billion addresses



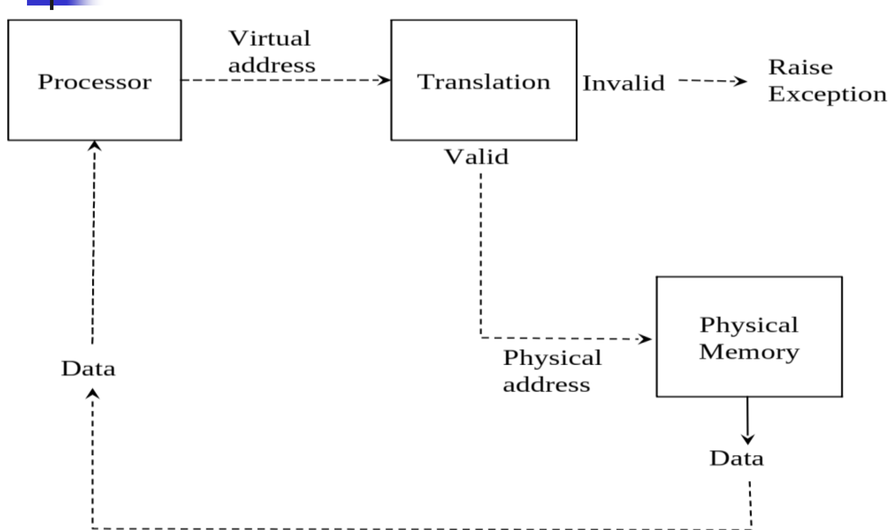
Virtual Address



Providing the Illusion of Separate Address Spaces



Address translation and memory protection

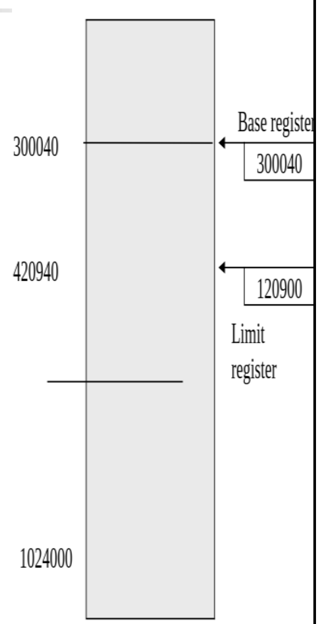


Memory Protection

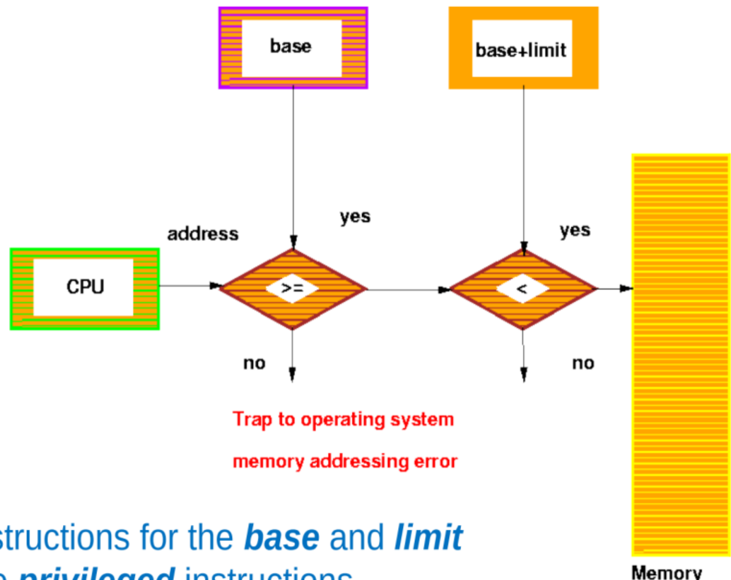
- When a process is running, only memory in that process address space must be accessible.
- When executing in kernel mode, the kernel has unrestricted access to all memory.

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

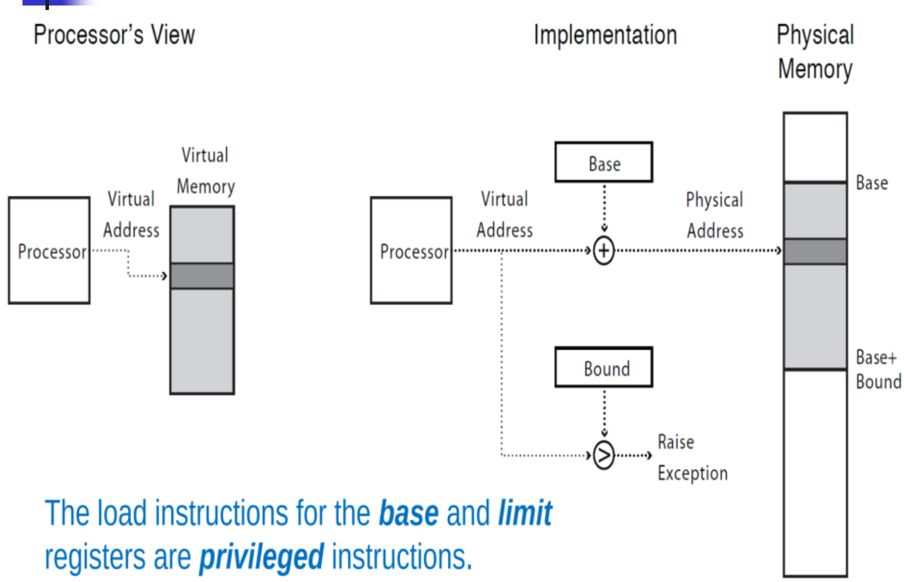


Hardware Address Protection



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

Virtual Address translation using the Base and Bounds method



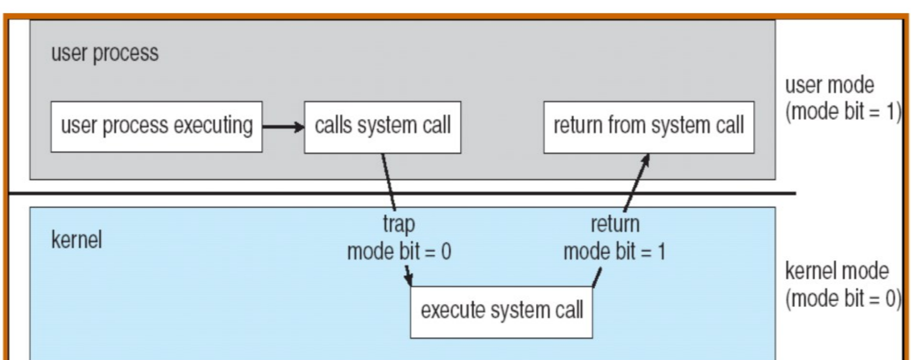
The load instructions for the ***base*** and ***limit*** registers are ***privileged*** instructions.

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

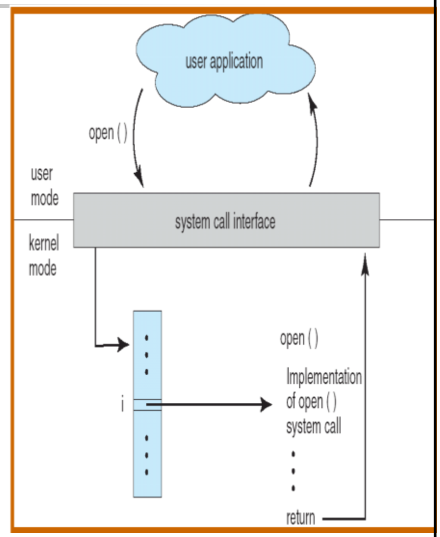
System Calls

- 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)
- System calls pass parameters between an and OS via registers or memory, memory tables or stack.
- Linux has about 300 system calls
  - read(), write(), open(), close(), fork(), exec(), ioctl(),…..



System services or system programs

- Convenient environment for program development and execution.
  - Command Interpreter (i.e., shell) parses/executes other system programs
  - Window management
  - System libraries, e.g., libc

Command Interpreter System

- Commands that are given to the operating system via command statements that execute
  - ✓ Process creation and deletion, I/O handling, secondary storage management, main memory Management, file system access, protection, networking, etc.
- Obtains the next command and executes it.
- Programs that read and interpret control statements also called –
  - ✓ Command-line interpreter, shell (in UNIX)

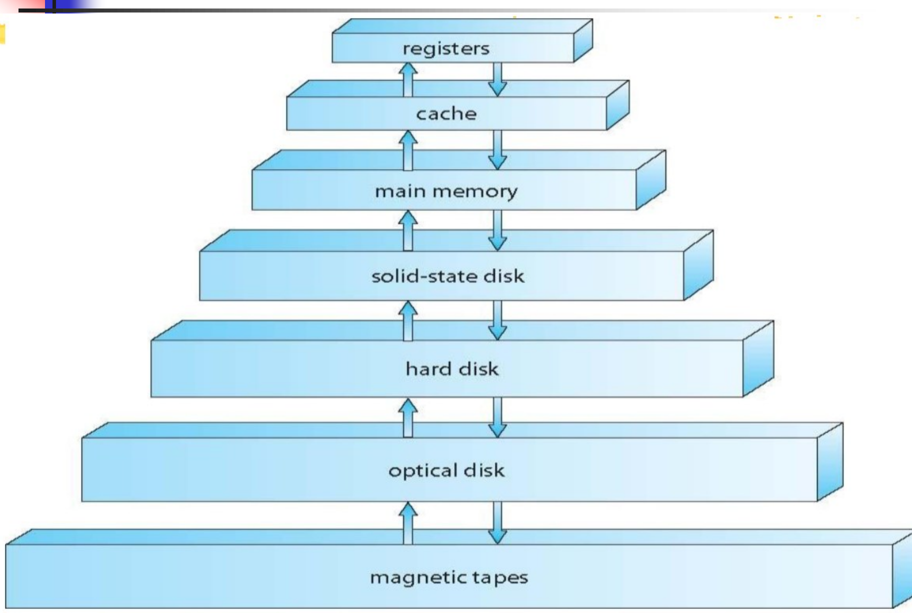
Storage Structure

- 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

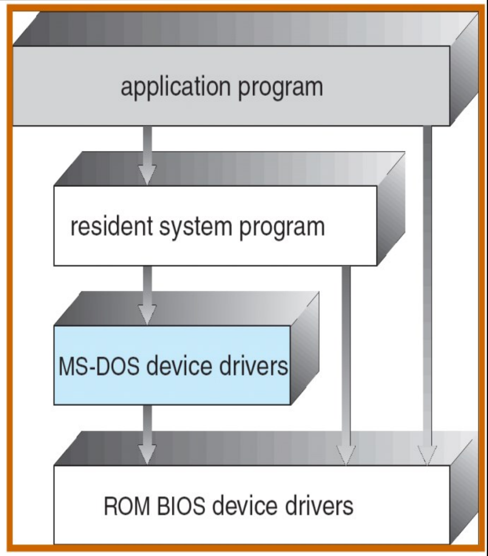
- Storage systems are organized in a hierarchy based on
  - Speed
  - Cost
  - Volatility
- Caching - process of copying information into faster storage system; main memory can be viewed as fast cache for secondary storage.

Storage Device Hierarchy



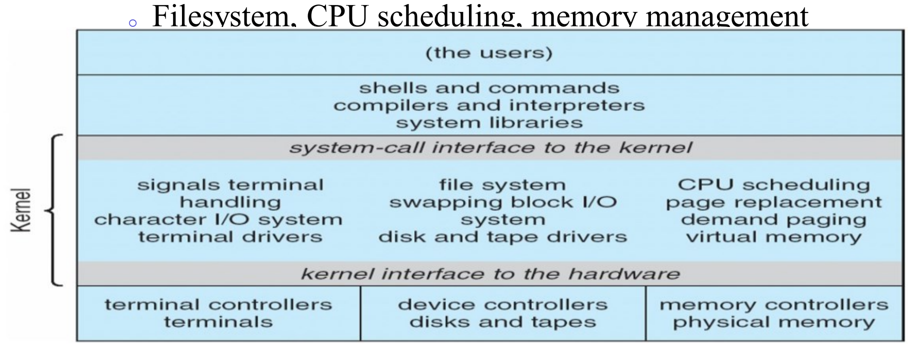
Operating Systems: How are they organized?

- OS Structure - Simple Approach
  - MS-DOS - provides a lot of functionality in little space.
  - Not divided into modules, Interfaces and levels of functionality are not well separated



Original UNIX System Structure

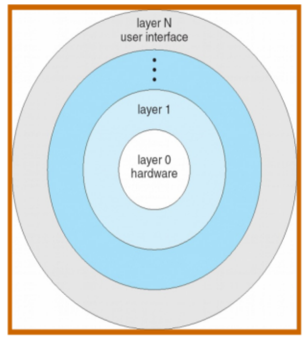
- Limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management



Layered OS Structure

- OS divided into number of layers - bottom layer is hardware, highest layer is the user interface.
- Each layer uses functions and services of only lowerlevel layers.
- THE Operating System and Linux Kernel has successive layers of abstraction

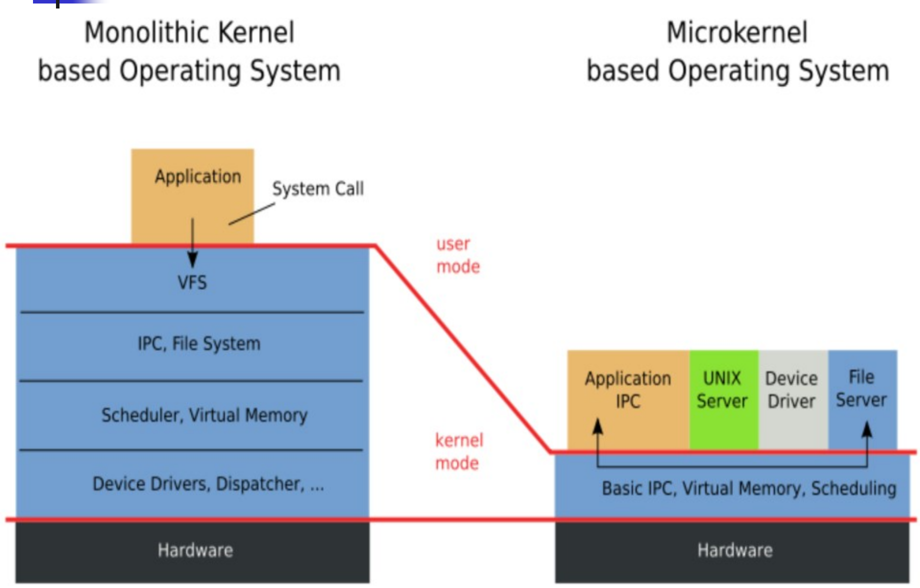
User Programs
Interface Primitives
Device Drivers and Schedulers
Virtual Memory
I/O
CPU Scheduling
Hardware



Monolithic vs. Microkernel OS

- Monolithic OSes have large kernels with a lot of components
  - ✓ Linux, Windows, Mac
- Microkernels moves as much from the kernel into "*user*" space
  - ✓ Small core OS components running at kernel level
  - ✓ OS Services built from many independent user-level processes
- Communication between modules with message passing
- Benefits:
  - ✓ Easier to extend a microkernel
  - ✓ Easier to port OS to new architectures
  - ✓ More reliable and more secure (less code is running in kernel mode)
  - ✓ Fault Isolation (parts of kernel protected from other par
- Detriments:
  - ✓ Performance overhead severe for naïve implementation

A microkernel OS



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. and disk scheduling.
- 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

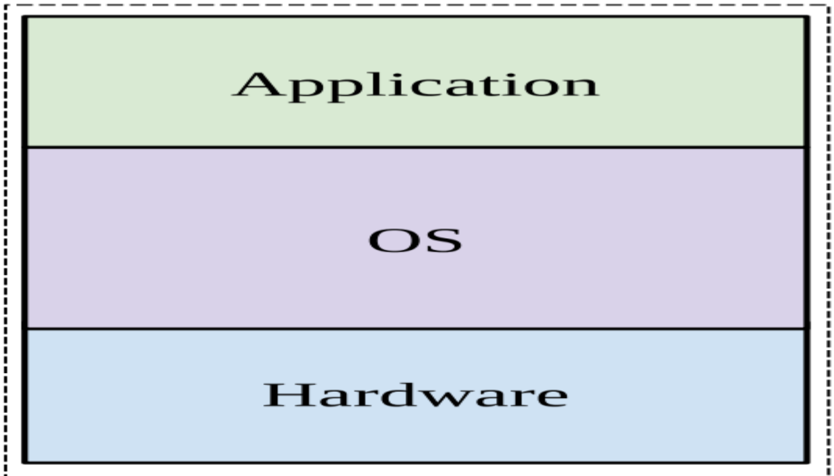
- File is a collection of related information - represents programs and data.
- OS is responsible for
  - ✓ File creation and deletion
  - ✓ Directory creation and deletion
  - ✓ Supporting primitives for file/directory manipulation.
  - ✓ Mapping files to disks (secondary storage)

OS Task: Protection and Security

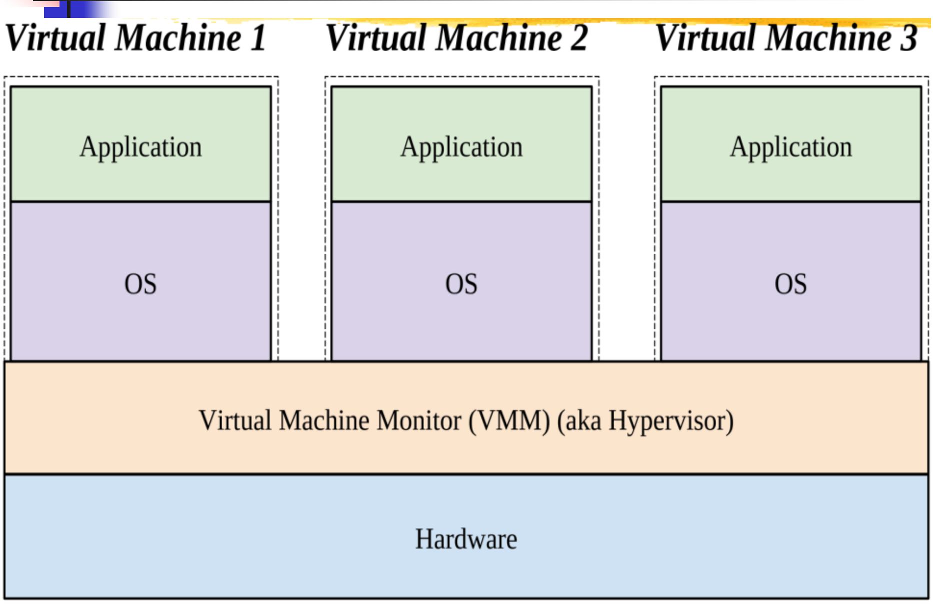
- Protection mechanisms control access of processes to user and system resources.
- Protection mechanisms must:
  - ✓ Distinguish between authorized and unauthorized use.
  - ✓ Specify access controls to be imposed on use.
  - ✓ Provide mechanisms for enforcement of access control.

Virtual Machines

Physical Machine



Virtual Machines



Virtual Machines

- Use cases
  - Resource configuration
  - Running multiple OSes, either the same or different Oses
  - Run existing OS binaries on different architecture

Summary of Lecture

- What is an operating system?
- Operating systems history
- Computer system and operating system structure