Operating Systems Concepts

Session Schedule - Monday (8am to 9am),

Wednesday (5pm to 6pm),

Saturday (8am to 9am)

4th August,2021

Process - A Process is a Program under execution.

Users/Programmers -> Executing Programs (Process which is an software application) -> -> Hardware

Hardware - CPU, I/O Devices(Mouse, Keyboard, Printer,Monitor,Scanners etc..), RAM (Random Access Memory) & Secondary Storage(Hard disk), SSD

Operating system is a collection of system programs that act as an interface between application programs and underlying computer hardware which is managed by Kernel.

Kernel is the innermost layer of Operating system that has direct access to system hardware.

Operating system provides a layer of abstraction to the software applications by hiding the internal aspects of managing the hardware for performing program execution and I/O operations.

Most of the machine critical applications are hosted on Cloud, this means that multiple applications on running on a single instance of the hardware. What does this mean ?

You have only one CPU and only One printer(assume) and there are multiple programs trying to access these resources simultaneously.

What is the primary aim of an Operating system from a Programmer or User's point of View ?

Answer - Provide convenience.

- Popular OS is Windows - Up until 2015 it had 95% of market share,
- Linux oS had only about 2% and
- Rest was by Macintosh.

As per latest stats, windows has roughly about 82% of market share.

As the complexity of the business problems became more and more complex, programmers time became more valuable, which means that the programs written by the programmer should run efficiently and deliver more value to the customer. This has directly reflected on improvising the efficiency of underlying Operating system.

The underlying OS should execute more number of transactions in a given unit of time. This measure is called as "Throughput".

This resulted in hugh demand for Multi User Operating systems and hence Linux started to get traction in the IT world.

<<Date>>

When single CPU/Processor is being used by multiple users simultenously to execture their jobs, there comes the concept of "Time Sharing".

1. What is meant by Multi User (it means the jobs submitted by each user) OSystem
2. What is meant by Multi-Tasking OSystem
3. What is meant by Multi Programming OS.
4. What is meant by Multi Processor OS.

OS Functionalities:

a) **Resource Management** -

b) **Process Management** - Process management is all about deciding on how to execute multiple processes simultaneously. Process management is done by using C.P.U/Processor scheduling algorithms (FIFO, LFU, SJF, Round Robin CPU scheduling algorithms)

c) **Storage Management** - Deals with Persistent storage aspects. OS interacts with Kernel for accessing various I/O devices thru System Calls. A system call is nothing but an API(Application Programming Interface)

How we manage our data on Secondary storage(HDD)

Different File systems that different OS's use and few popular file systems are:

a. CIFS - Common Interface File System

b. NFS - Network File System.

Data on HDD in the form data blocks gets stored inside a Track and Sector.

d) **Memory Management** - Deals with how OS manages the RAM (Random Access Memory)

Eg. Calculator.c is the program - This will be stored as a file on HDD (Source Code file)

cc Calcuatlor.c - This will create Calculator.exe and will be stored on HDD as Executable file.

c:>d:\cPrograms\Calculator.exe (Enter)

CPU connects to Kernel thru system call called "read “and passes the path of the executable file.

Kernel will connect to HDD and will get the file content(in binary format) and hands over to OS.

Binary file will be placed on Memory and CPU will execute the program that is placed on Memory.

e) **Security & Privacy** - There are different layers of security in Application life cycle.

a) Security at OS Level(Authentication of Users & Authorization on every file being accessed) - Underlying O.S

b) Security at Network Level (Fire walls, Proxy Servers) - Infrastructure team

c) Security at Programmng Level (Encription and Decription of credentials and/or data) - Programmer or System Architects are responsible

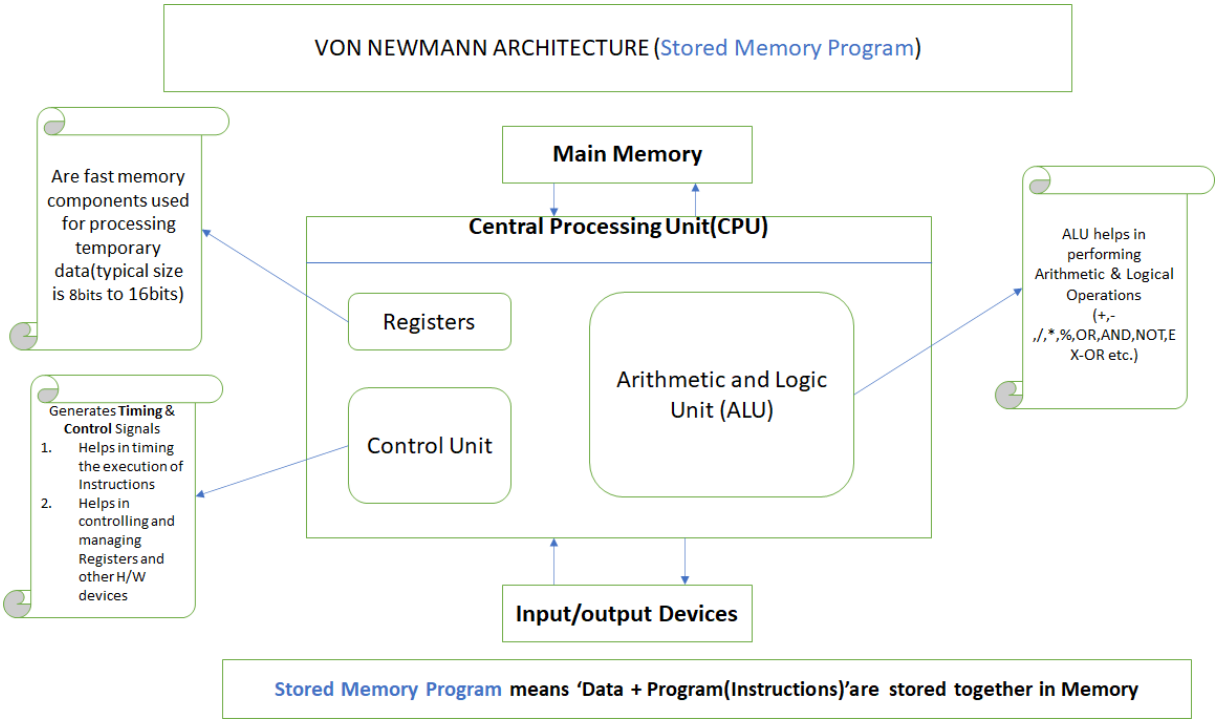
Security of File System - User-Group-Others

rwx-rwx-rwx

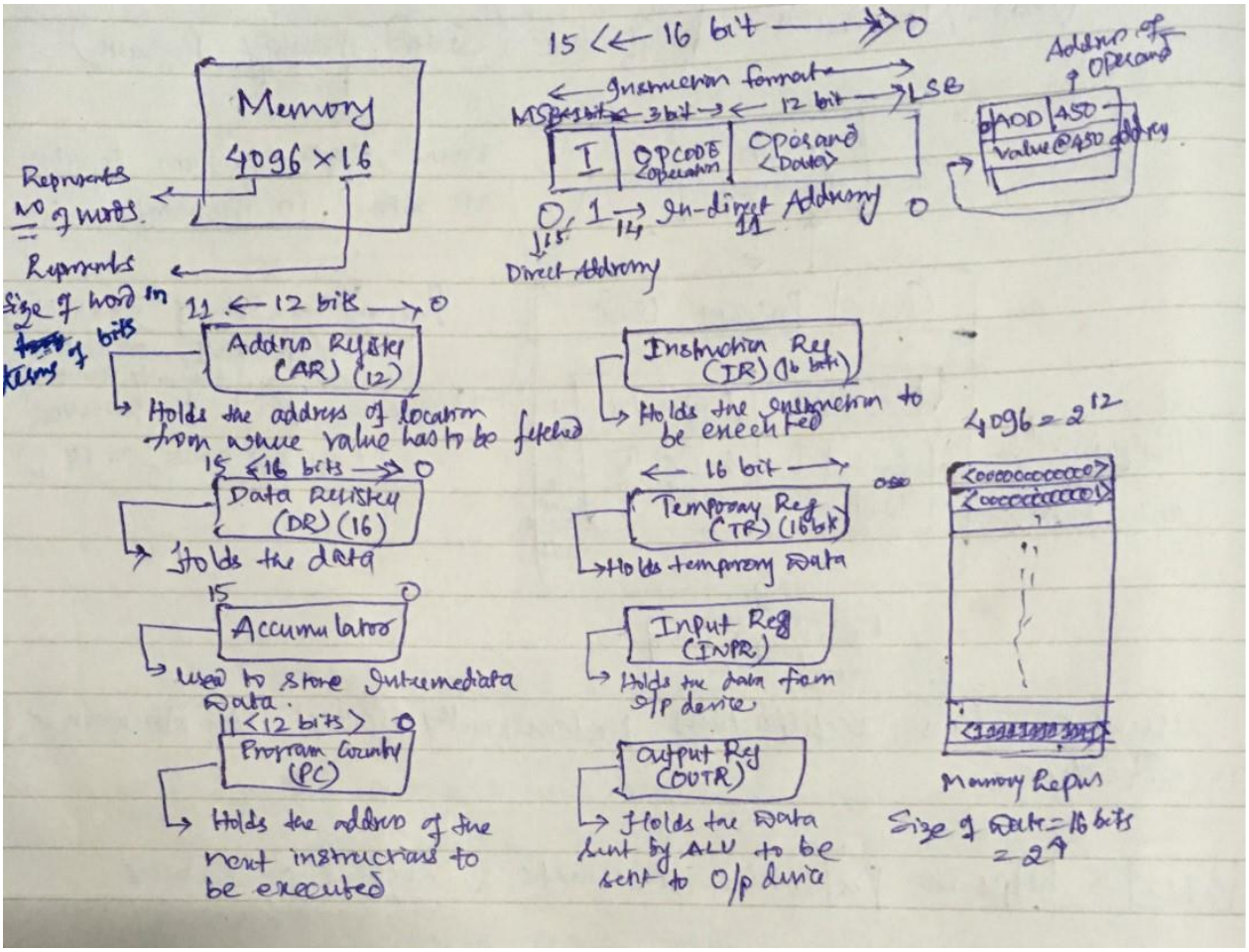
999

< <Date>>

All the modern computer hardware Architectures adopt Von-Neuman Architecture.
Von-Neuman architecture is an Achitype. An Architype is something link a style.



Various General-Purpose Registers in Computer

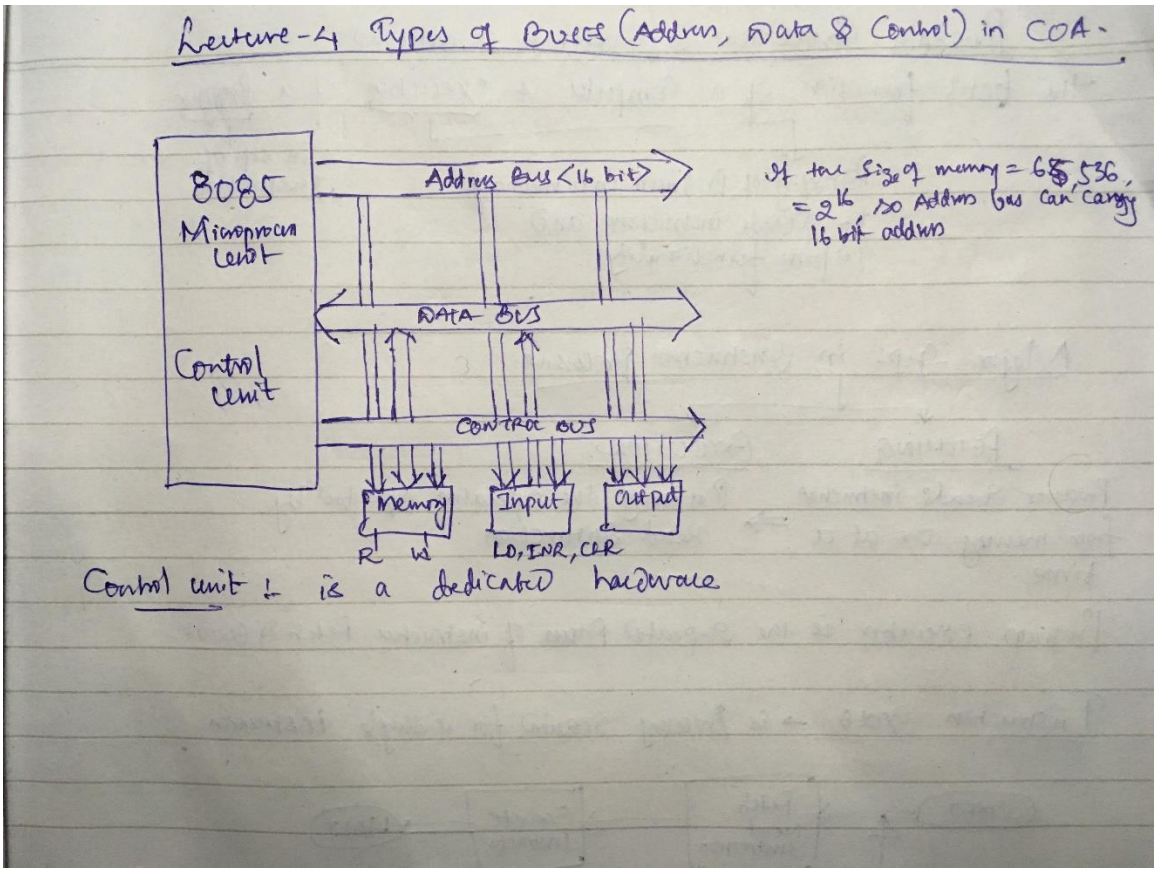


Instruction format is combination of <AddressingMode-1bit><3bit OpCode><12bit Operands>

Add.c

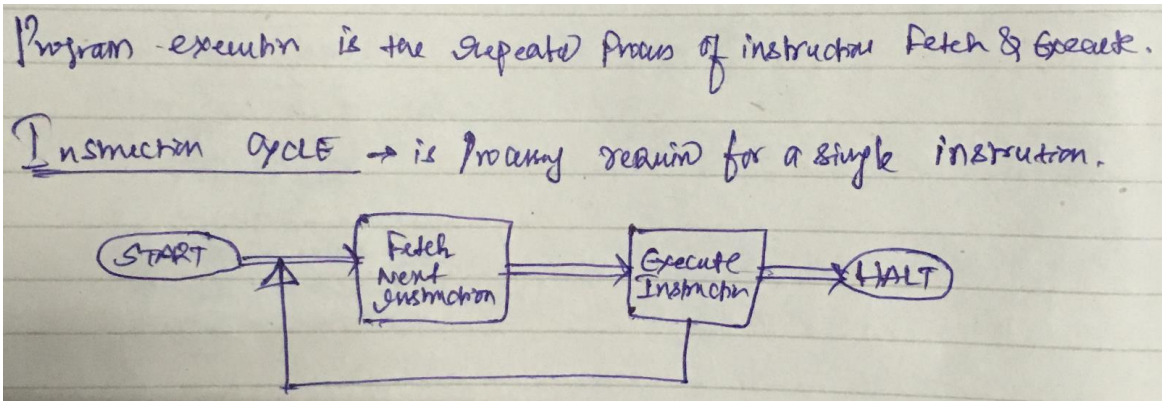
```
Void main()
{
  Int a,b,c;        //Declaration – 3 memory locations will be allocated in RAM (in Stack Area)

  Scanf(a,b);       // I/O operation which is reading the data from key-board and upon reading(when user
                    enters “Enter Button”), and interrupt will be generated to OS. OS will read the data from
                    keyboard and pass it on to memory
  C=a+b;            // Computation which leads to generation of instruction in the Instruction format.
<0><ADD><a,b>
                      ALU executes the operation and stores the result into Accumulator.
                      From Accumulator(Register), the value of ACC will then be passed on to variable Ç’in
the memory.
  Print(“sum = “,c); // Value of ‘C’ will be pushed to STDIO
}
```

Types of BUSES in Computer



Program Execution and Instruction Execution

1. To execute a program, the program has to be brought into Memory
2. Each statement has to decoded into Instruction
3. Execute each instruction.



Key activities in processing the instruction are:

a) Fetching the instruction
b) Executing the instruction

The above discussion is made to understand how an OS plays an important role in manging the Process. Between fetching and execution of instructions, there is something called as waiting stage.

<<Date>

Chapter 1: Introduction

- What is an Operating System?
- Computer-System Organization
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Open-Source Operating Systems

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

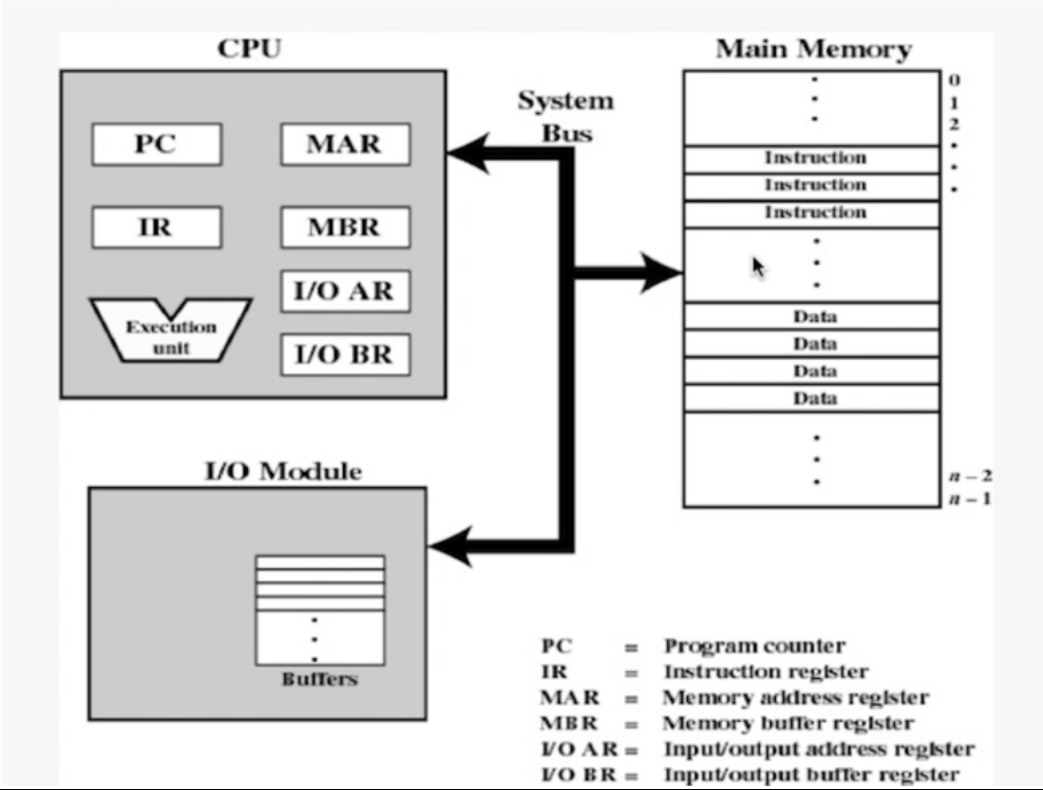
What is an Operating System?

- 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
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

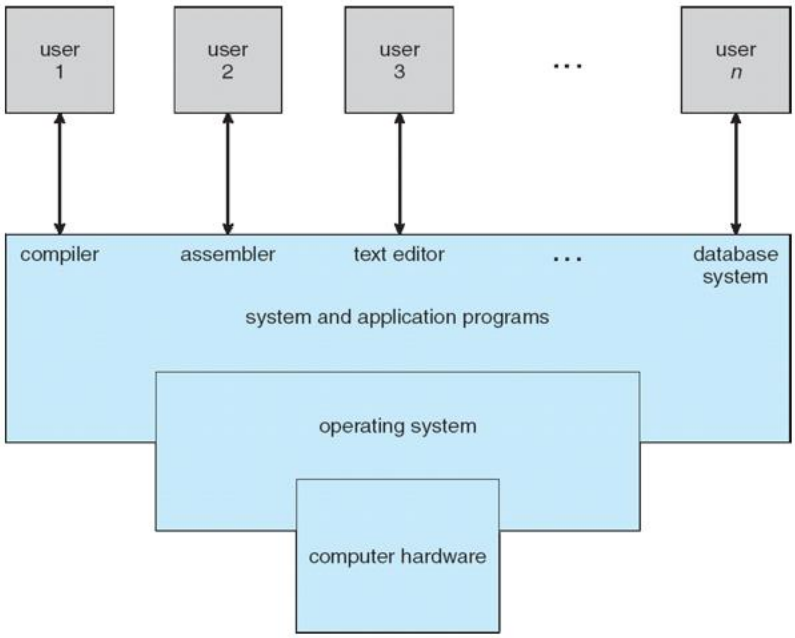
Computer System Structure

- 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

Computer Architecture



Four Components of a Computer System



What Operating Systems Do

- The operating system controls the hardware and coordinates its use among the various application programs for the various users.
- We can also view a computer system as consisting of hardware, software, and data.
- The operating system provides the means for proper use of these resources in the operation of the computer system.
- An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an **environment** within which other programs can do useful work.
- To understand more fully the operating system's role, we explore operating systems from two viewpoints:
  - The user
  - The system.

User View

The user's view of the computer varies according to the interface being used

- **Single user computers** (e.g., PC, workstations). Such systems are designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. the operating system is designed mostly for **ease of use** and **good performance.**
- **Multi user computers** (e.g., mainframes, computing servers). These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization -- to assure that all available CPU time, memory, and I/O are used efficiently and that no individual users takes more than their air share.

User View (Cont.)

- **Handheld computers** (e.g., smartphones and tablets). The user interface for mobile computers generally features a **touch screen.** The systems are resource poor, optimized for usability and battery life.
- **Embedded computers** (e.g., computers in home devices and automobiles) The user interface may have numeric keypads and may turn indicator lights on or off to show status. The operating systems are designed primarily to run without user intervention.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. There are two different views:

- The operating system is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- The operating systems is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Defining Operating System

No universally accepted definition of **what** an OS:

- Operating systems exist to offer a reasonable way to solve the problem of creating a usable computing system.
- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.
- Since bare hardware alone is not particularly easy to use, application programs are developed.
  - These programs require certain common operations, such as those controlling the I/O devices.
  - The common functions of controlling and allocating resources are brought together into one piece of software: the **operating system**.

Defining Operating System (Cont.)

No universally accepted definition of what is **part** of the OS:

- A simple viewpoint is that it includes everything a vendor ships when you order the operating system. The features that are included vary greatly across systems:
  - Some systems take up less than a megabyte of space and lack even a full-screen editor,
  - Some systems require gigabytes of space and are based entirely on graphical windowing systems.

Defining Operating System (Cont.)

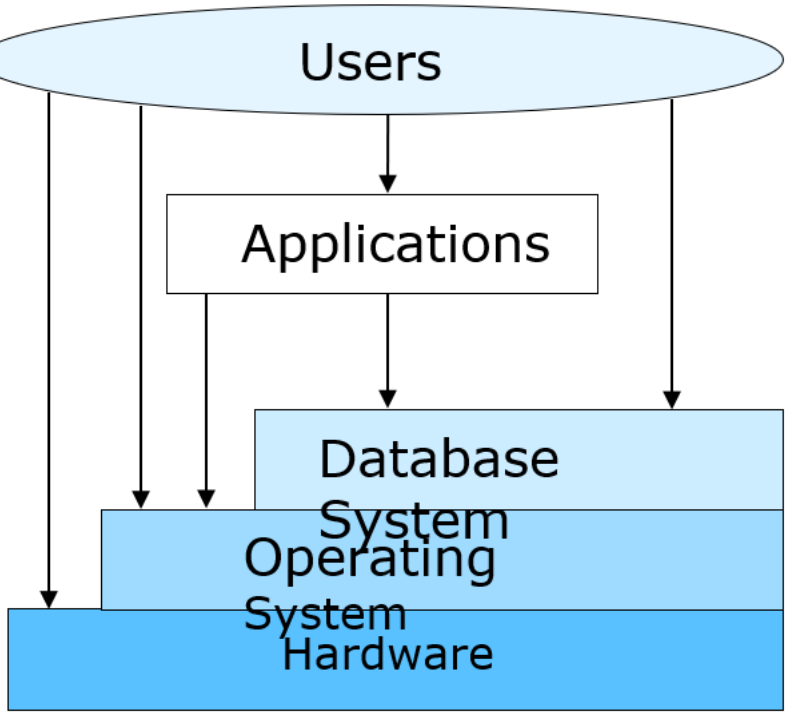
No universally accepted definition of what is **part** of the OS:

- A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer -- usually called the **kernel**.
- Along with the kernel, there are two other types of programs:
  - System programs, which are associated with the operating system but are not necessarily part of the kernel.
  - Application programs, which include all programs not associated with the operation of the system.

Defining Operating System (Cont.)

- The emergence of mobile devices, have resulted in an increase in the number of features that constituting the operating system.
- Mobile operating systems often include not only a core kernel but also **middleware** -- a set of software frameworks that provide additional services to application developers.
- For example, each of the two most prominent mobile operating systems -- Apple's iOS and Google's Android -- feature a core kernel along with middleware that supports databases, multimedia, and graphics (to name only a few).

Evolution of Computer Systems



Computer-System Organization

- A modern general-purpose computer system consists of **one or more CPUs** and **a number of device controllers** connected through **a common bus** that provides access to **shared memory**.
- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays). **Each device controller** has **a local buffer**.
- CPU moves data from/to **main memory** to/from **local buffers**.
- The CPU and the device controllers can execute in parallel, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller **synchronizes** access to the memory.

Note: Device Controller is a part of I/O Interface of a specific I/O device which acts as Interface between CPU for data get(Input)/put(Output) and takes care of data part.

Input/Output Device and CPU Communications thru I/O Interface having Device Controller and Device Electronics as sub systems:

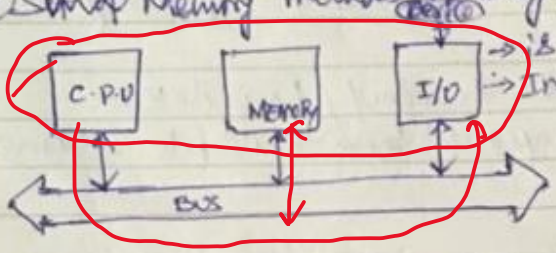
Finally, O.S acts as interface between "System calls".

Part-3 Input / output : I/O

Part-1 : Processor
Part-2 : Memory
Part-3 : I/O

(*) CPU is essentially the master of the whole situation.

(*) Storage Memory includes hierarchy (HDD - RAM - BUFFERS - REGISTERS)



is external memory.
→ interface.

I/O Constitutes set of devices & transducers.
Bus is a set of signal lines if can be standardized, then we can any type of Processor and we can also accomodate different types of memory systems also CPU memory subsystem to be specific.

The interaction between CPU & memory is ① CPU is the master of the system and address the memory. Also, it is the master of the bus.
② Memory as a slave responds and it sends/receives the data from/to CPU.
③ Likewise I/O interacts with CPU either for Input or output of data.
Memory - holds frozen data
I/O - holds live data

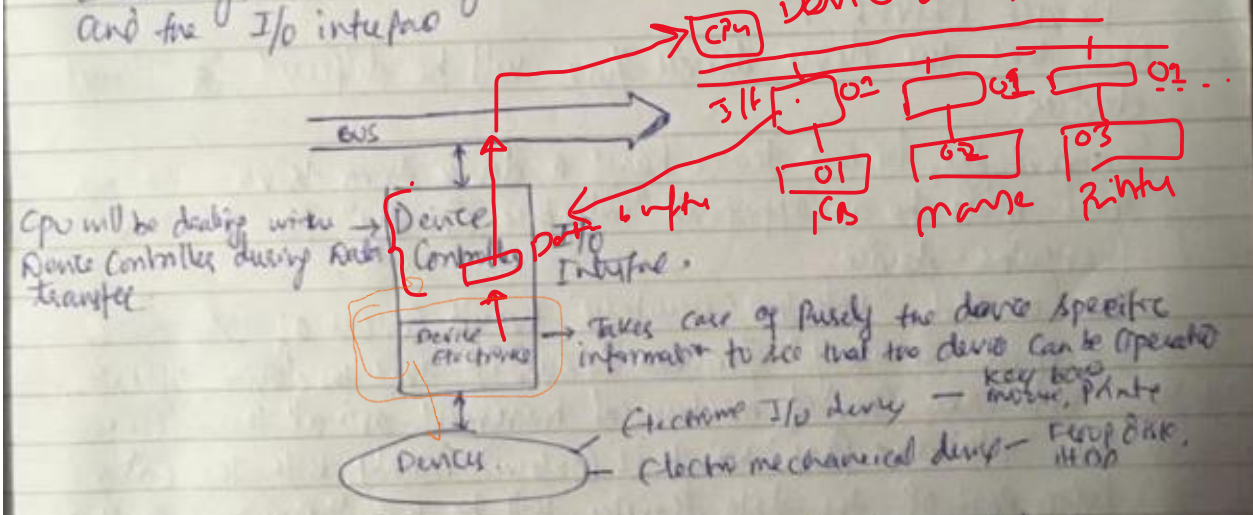
Like how CPU can address the memory for R/W of data, same CPU can address I/O devices for get(I/P)/put(O/P) data.
The problem with interaction between CPU & I/O devices is that CPU will not be efficient utilized incase it has to deal with s/w I/O devices and most of the I/O devices are slow in fact.
Therefore, it is better that C.P.U should not get directly involved in this.

So, the I/o adapter as component acts as Buffer and interface between the device & system Bus on which the CPU exists

(F) On the Bus, the data transmitted between Device & CPU should be standardized.

(I) While we say that interaction between CPU & memory, CPU & I/O are similar, the key difference is that the speed of I/o devices are generally slow.

Let us now get into the details of I/o.
Essentially, we can say that there is an interface from the Bus end and the I/o interface



Device Controller :- Acts as interface between CPU for data get/put
Takes care of data part

Device Electronics :- Deals with different electronic and electro mechanical I/o devices by interfacing with them

(S) CPU will address the Device Controller and then the data will have to be transferred In (or) out

(I) The Device Controller will identify each device with a unique code which is the actual address of the device

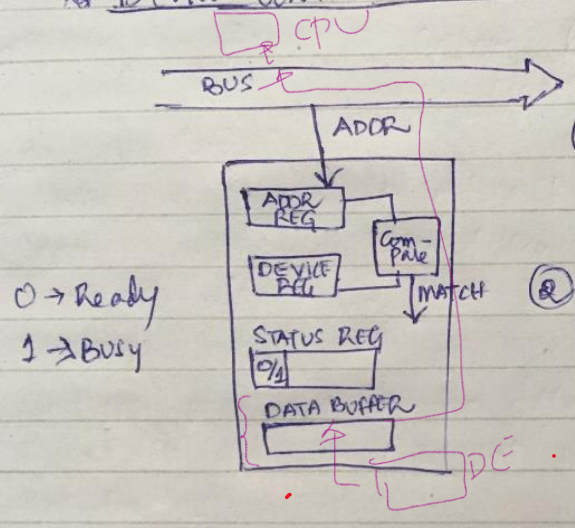
(2) If there is a device that generates 1 bit of data at a time and let us say that, the Bus is capable of transmitting 16 bit of data, then if device controller transmits every time 1 bit of data generated by the device, then unnecessarily the Bus bandwidth is wasted.
So, it is the job of the device controller to accumulate 16 bit of data and then interface send it to CPU over Bus.

Now, the piece of software/program running on CPU to make a particular device operable (or) addressable is called as "DEVICE DRIVER".
Note that the device drivers S/W's will be different for different devices.
Correspondingly, for each device have a device ~~driver~~ drivers on CPU, there must be an appropriate "interface circuitry" for that particular device.
For each "interface circuitry" would consist of specific "Electronic Circuitary" for ensuring that the device operates, which will take care of mechanical movement and all those things and the "Electronic part", which is concerned mainly with the data (or) logic part of the data transfer will be the "Device Controller".

So, it is actually the device drivers, which when run, will generate a code, which will be placed on the bus that helps a particular device to be identified and if the data is ready, if it is an input device that data will be passed over the bus and the device drivers will receive and route it to the necessary registers.

A Device Controller – Detailed view

memory mapped I/O interface

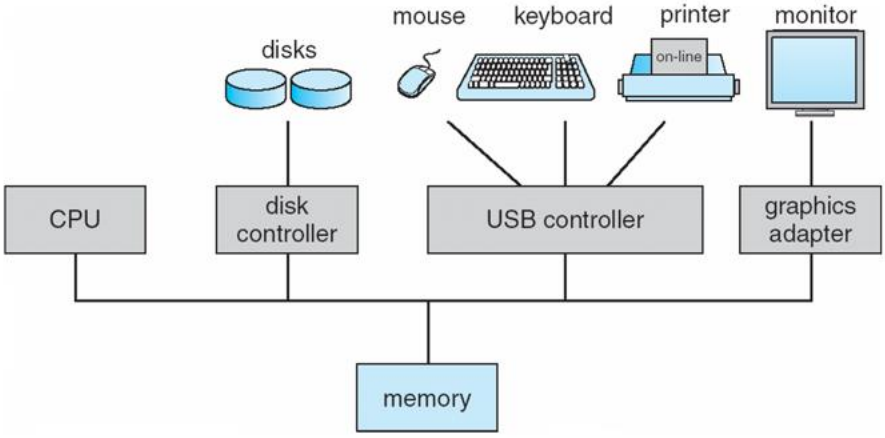


0 → Ready
1 → Busy

① Device driver generates the code in CPU with Device Address (to be accessed) and places it in "ADDRESS REGISTER" in device controller of device over BUS.

② Value in ADDR REGISTER is compared with value in "DEVICE REGISTER" and in match occurs, then data generated by I/p device in Data Buffer will be placed in System BUS.

Modern Computer System



Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in **ROM or EPROM**, generally known as **firmware**
  - Initializes all aspects of system
  - Loads **operating system kernel** and starts execution

Firmware – Is a tangible electronic component with embedded software instructions such as BIOS. Firmware is held in non-volatile memory devices such as ROM, EPROM or flash memory

---

<<Date >>

Computer-System Operation

- Once the kernel is loaded and executing, it can start providing services to the system and its users.
- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**, or **system daemons** that run the entire time the kernel is running.
- On UNIX, the first system process is **init** and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.
- The occurrence of an event is usually signaled by an **interrupt.**

About the Unix Boot Process

Bootstrapping is the full name for the process of bringing a computer system to life and making it ready for use.

The name comes from the fact that a computer needs its operating system to be able to do anything, but it must services normally provided by the operating system to do so. Hence, it must "pull itself up by its own bootstraps."

The basic boot process is very similar for all Unix systems, although the mechanisms used to accomplish it vary qui

The normal Unix boot process has these main phases:

Basic hardware detection (memory, disk, keyboard, mouse, and the like).

Executing the firmware system initialization program (happens automatically).

Locating and running the initial boot program (by the firmware boot program), usually from a predetermined location

Locating and starting the Unix kernel.

The kernel initializes itself and then performs final, high-level hardware checks, loading device drivers and/or kernel

The kernel starts the init process, which in turn starts system processes (daemons) and initializes all active subsyst

The *kernel* is the part of the Unix operating system that remains running at all times when the system is up.

The kernel executable image itself, conventionally named *unix* (System V-based systems), *vmunix* (BSD-based sys

It is traditionally stored in or linked to the root directory.

Here are typical kernel names and directory locations for the various operating systems we are considering:

AIX	*/unix* (actually a link to a file in */usr/lib/boot*)
FreeBSD	*/kernel*
HP-UX	*/stand/vmunix*
Linux	*/boot/vmlinuz*
Tru64	*/vmunix*
Solaris	*/kernel/genunix*

```
[vobugari@webminal.org ~]$uname
Linux
[vobugari@webminal.org ~]$
```

```
[vobugari@webminal.org ~]$pwd
/home/vobugari
[vobugari@webminal.org ~]$cd /
[vobugari@webminal.org /]$pwd
/
[vobugari@webminal.org /]$cd boot
```

```
[vobugari@webminal.org boot]$pwd
/boot
[vobugari@webminal.org boot]$ls -l
```

```
-rwxr-xr-x. 1 root root  5156528 Oct 25  2016 vmlinuz-0-rescue-41227fcee63d4
c579f21e742e4225972
-rwxr-xr-x. 1 root root  5157936 Oct 11  2016 vmlinuz-3.10.0-327.36.2.el7.x8
6_64
-rwxr-xr-x. 1 root root  5156528 Nov 19  2015 vmlinuz-3.10.0-327.el7.x86_64
-rwxr-xr-x. 1 root root  5396240 Apr 12  2017 vmlinuz-3.10.0-514.16.1.el7.x8
6_64
[vobugari@webminal.org boot]$
```

Unix Daemon Processes

A **daemon** (also known as background processes) is a Linux or UNIX program that runs in the background.

Almost all daemons have names that end with the letter "**d**".

For example, httpd the daemon that handles the Apache server

sshd which handles SSH remote access connections.

Linux often start daemons at boot time.

Shell scripts stored in /etc/init.d directory are used to start and stop daemons.

```
[vobugari@webminal.org ~]$cd /
[vobugari@webminal.org /]$cd etc
[vobugari@webminal.org etc]$pwd
/etc
[vobugari@webminal.org etc]$ls -l ini*
lrwxrwxrwx. 1 root root  11 May 25  2017 init.d -> rc.d/init.d
-rw-r--r--. 1 root root 511 Apr 12  2017 inittab
```

```
[vobugari@webminal.org etc]$cd init.d
[vobugari@webminal.org init.d]$pwd
/etc/init.d
```

```
[vobugari@webminal.org init.d]$ls -l
total 16
-rw-r--r--. 1 root root 15131 Sep 12  2016 functions
-????????? ? ?    ?        ?            ? init_siab
-????????? ? ?    ?        ?            ? netconsole
-????????? ? ?    ?        ?            ? network
-????????? ? ?    ?        ?            ? README
-????????? ? ?    ?        ?            ? siab
[vobugari@webminal.org init.d]$
```

Typical functions of Daemon Processes

- Open network port (such as port 80) and respond to network requests.
- Monitor system such as hard disk health or RAID array.
- Run scheduled tasks such as cron.

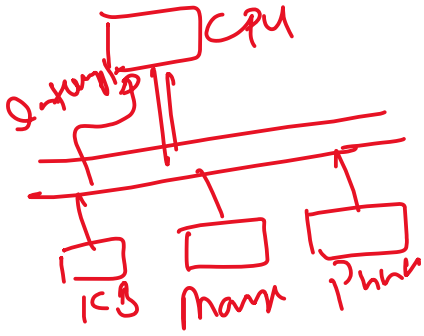
Following are some of the Service Daemons for Linux:

- amd - Auto Mount Daemon
- anacron - Executed delayed cron tasks at boot time
- apmd - Advanced Power Management Daemon
- atd - Runs jobs queued using the at tool
- autofs - Supports the automounter daemon allowing mount and unmount of devices on demand
- crond - The task scheduler daemon
- cupsd - CUPS printer daemon
- dhcpd - Dynamic Host Configuration Protocol and Internet Bootstrap Protocol Server
- ftpd - FTP Server Daemon

- gated - routing daemon that handles multiple routing protocols and replaces routed and egpup
- httpd - Web Server Daemon
- inetd - Internet Superserver Daemon
- imapd - An imap server daemon
- lpd - Line Printer Daemon
- memcached - In-memory distributed object caching daemon
- mountd - mount daemon
- mysql - Database server daemon

Interrupts

- There are two types of interrupts:
  - **Hardware** -- a device may trigger an interrupt by sending a signal to the CPU, usually by way of the system bus.
  - **Software** -- a program may trigger an interrupt by executing a special operation called a **system call.**
- A software-generated interrupt (sometimes called **trap** or **exception**) is caused either by an error (e.g., divide by zero) or a user request (e.g., an I/O request).
- An operating system is **interrupt driven.**

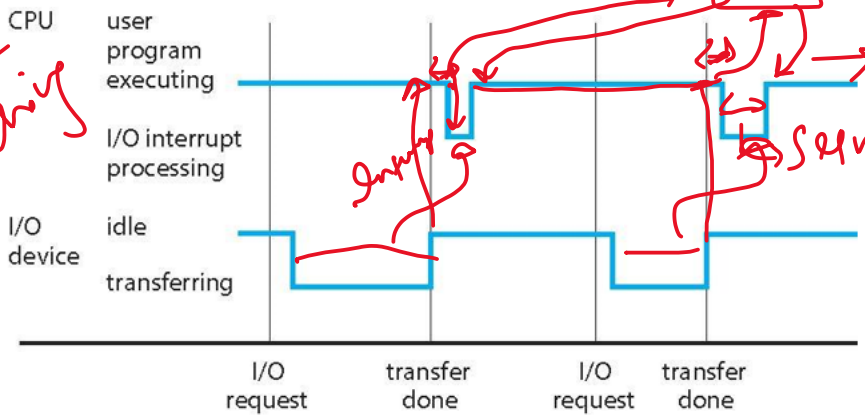


Common Functions of Interrupts

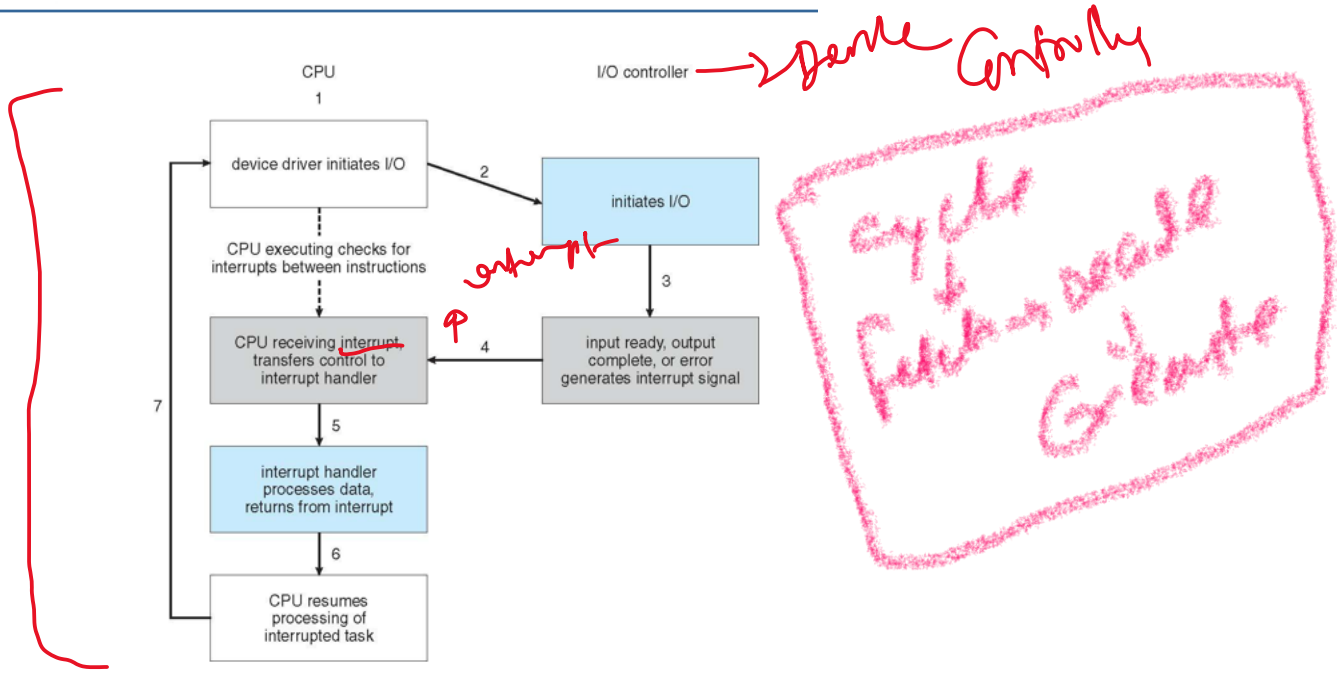
- When an interrupt occurs, the operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred and transfers control to the interrupt-service routine. Interrupt Service routine is also known as System Programs.
- An interrupt-service routine is a collection of routines (modules), each of which is responsible for handling one particular interrupt (e.g., from a printer, from a disk)
- The transfer is generally through the **interrupt vector**, which contains the addresses of all the service routines

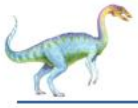
  Interrupt Vector – is a Look up table (like Hash Table) that maintains Name(Interrupt Service Routine(**Key**) and its address(**Value**)
- Interrupt architecture must save the address of the interrupted instruction.

Interrupt Timeline



Interrupt-driven I/O cycle.

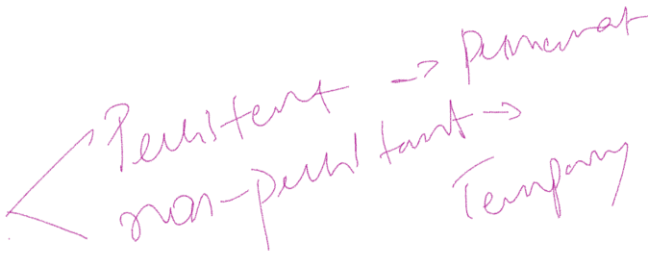




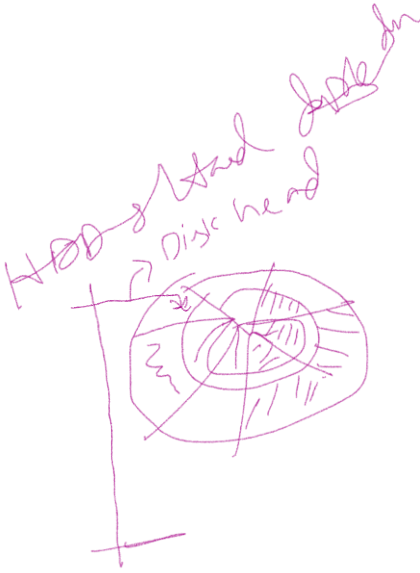
Intel Pentium processor event-vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Storage Structure



- Main memory – the only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard 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
  - **Solid-state disks** – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular
- Tertiary storage

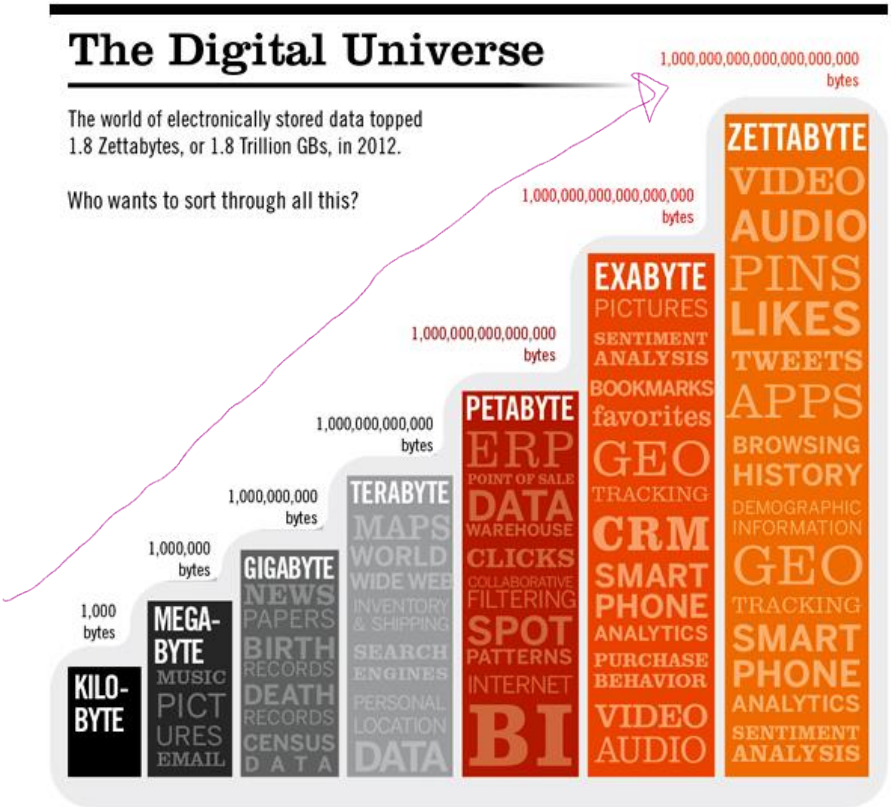


Storage Definition

- The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits.
- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.
- A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.

Storage Definition (Cont.)

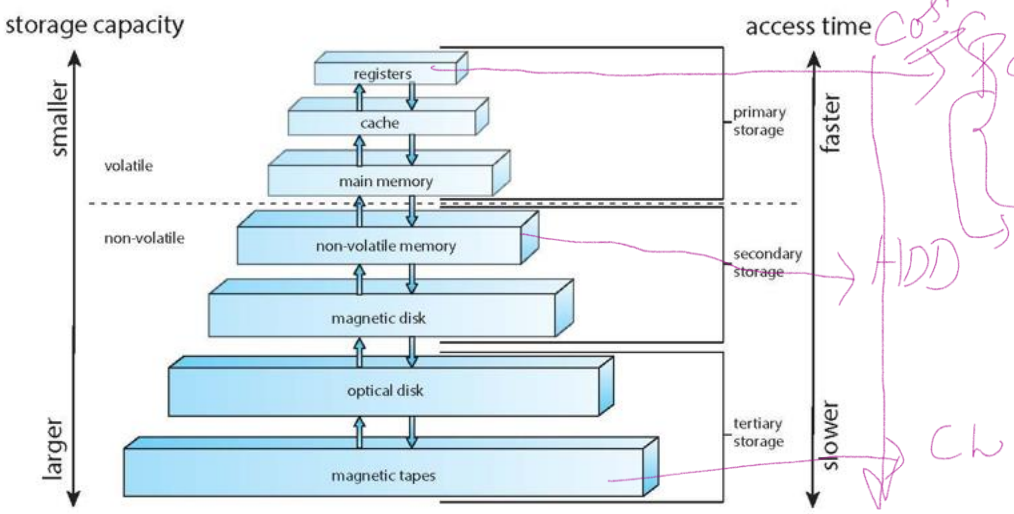
- Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
  - A **kilobyte**, or **KB**, is 1,024 bytes
  - a **megabyte**, or **MB**, is $1{,}024^2$ bytes
  - a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
  - a **terabyte**, or **TB**, is $1{,}024^4$ bytes
  - a **petabyte**, or **PB**, is $1{,}024^5$ bytes
  - exabyte, zettabyte, yottabyte
- Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).



Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information from "slow" storage 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



I/O Structure

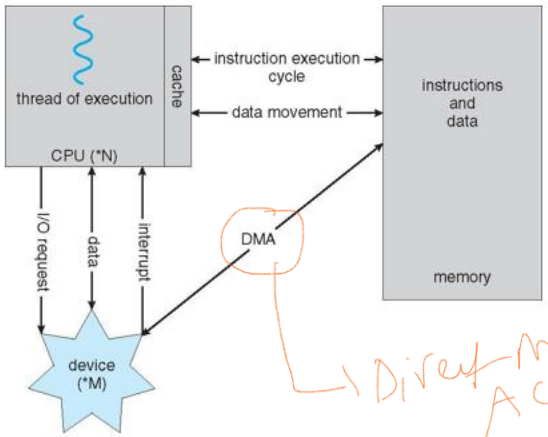
- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device. More than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface** (SCSI) controller.
- A device controller maintains some local buffer storage and a set of special-purpose registers.
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Typically, operating systems have a **device driver** for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

I/O Structure (Cont.)

- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as "read" a character from the keyboard).
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.
- For other operations, the device driver returns status information.

How a Modern Computer Works

A von Neumann architecture and a depiction of the interplay of all components of a computer system.



<<Date>>

Computer-System Architecture

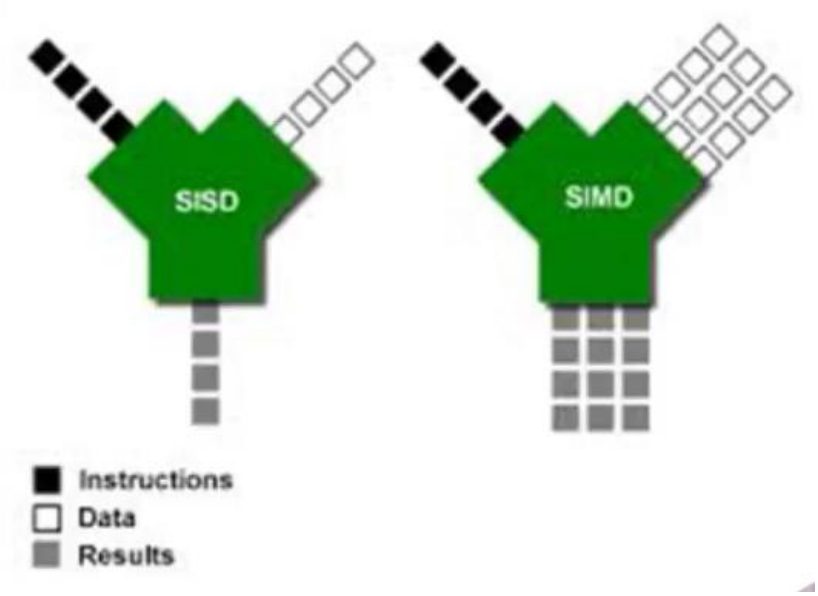
- **Single general-purpose processor**
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    - **Increased throughput**
    - **Economy of scale**
    - **Increased reliability**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

Components of a Microprocessor are:

a. GPU - Graphic Processing Unit
b. DSP - Digital Signal Processor
c. FPU - Floating Point Unit
d. SOC - System On Chip

Special Purpose Processors

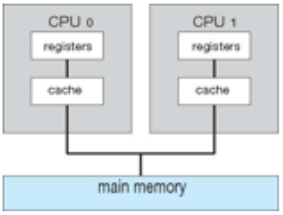
1. Will have more ALU's and GPU's(Graphic Processing Units) devote more transistors for Data Processing.
2. Very efficient for "Fast parallel floating point processing"
3. Supports "Single Instructions Multiple Data Operations" (**Flynn's Classification)**



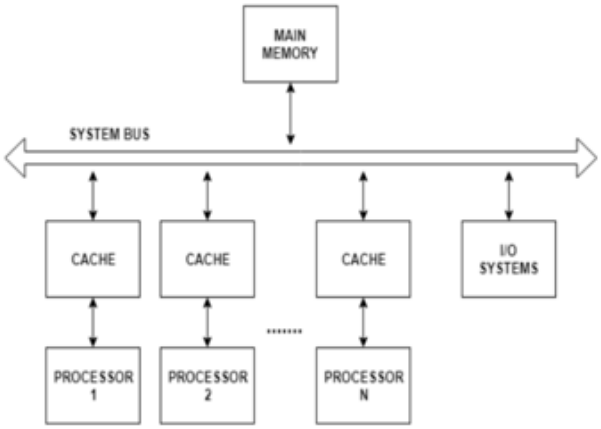
- Performs same instruction on multiple data elements
- Also called, "sub-world parallel instructions" meaning perform same operation
- SIMD Architectures can provide substantial speed for:
    - 3D Graphics
    - 3D Physics
    - Image processing
    - Signal processing
    - Numerical processing &
    - Video encoding and decoding

4. High Computation per Memory Access

Symmetric Multiprocessing Architecture



1. Multiple Processors share a common memory and OS
2. All of these processors work in tandem to execute their processes
3. OS treats all processors equally and no processor is reserved for special purpose



Symmetric Multiprocessing – Key Points

- Is also known as tightly coupled multiprocessing as all the CPU's are connected to same bus and have access to a shared memory
- Each processor has its own private cache memory to decrease system bus traffic and to reduce data access time
- Allows processor to execute any process no matter where its data is located in memory. Only condition is that no process should be executing on more than one processor
- Do not exceed more than 16 processors keeping view of OS and other device constraints such as Memory and BUS etc..

Symmetric Multiprocessing - Uses

- Useful for Timesharing systems as they have multiple processors running in parallel
- Relevant on personal computers only when multithreaded programming is taken into account
- Time sharing systems that use multithreading programming can also take advantage of Symmetric Multiprocessing architecture
- Throughput of the system is increased as there are multiple processors facilitating for execution of multiple processes
- Reliable and dependable in terms of providing fail-safe feature since even if a single processor fails, the system still endures.

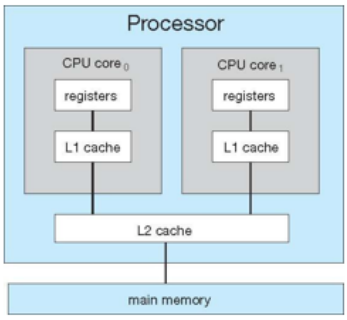
Symmetric Multiprocessing - Disadvantages

- Leads to complicated OS design since the OS has to simultaneously handle multiple processors
- Need for large main memory to accommodate multiple processors as they are all connected to same memory via common BUS.

Multicore Systems

- Most CPU design now includes multiple computing cores on a single chip. Such multiprocessor systems are termed **multicore**.
- **Multicore systems** can be more **efficient** than **multiple chips with single cores** because:
  - **On-chip communication** is **faster** than **between-chip** communication.
  - One chip with multiple cores uses significantly less **power** than multiple single-core chips, an important issue for **laptops** as well as **mobile devices**.
- Note -- while **multicore systems** are **multiprocessor systems**, not all **multiprocessor systems** are **multicore**.

A dual-core with two cores placed on the same chip



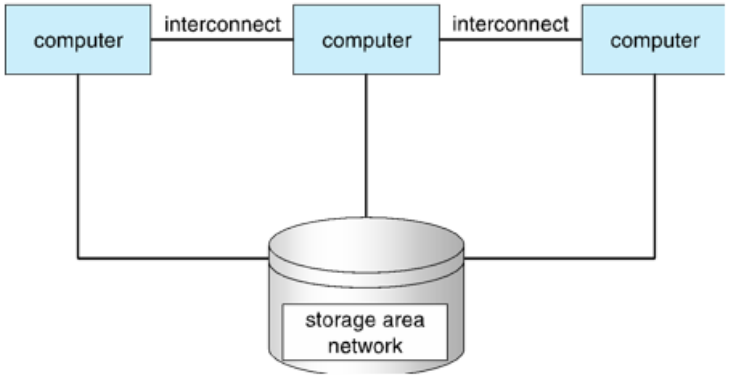
- L1 Cache memory is embedded on Microprocessor chip it self.
- L2 Cache memory is on a separate chip (possibly on expansion card) that can be accessed more quickly than the larger “main” memory.

Clustered Systems

Like multiprocessor systems, but **multiple systems** working together

- Usually sharing storage via a **storage-area network (SAN)**
- Provides a **high-availability** service which survives failures
  - **Asymmetric clustering** has one machine in hot-standby mode
  - **Symmetric clustering** has multiple nodes running applications, monitoring each other
- Some clusters are for **high-performance computing (HPC)**
  - Applications must be written to use **parallelization**
- Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems



Multiprogrammed System

- **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
- Batch systems:
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), **OS switches to another job**
- Interactive systems:
  - Logical extension of batch systems -- CPU switches jobs so **frequently** that users can interact with each job while it is running, creating **interactive** computing

---

<<Date>>

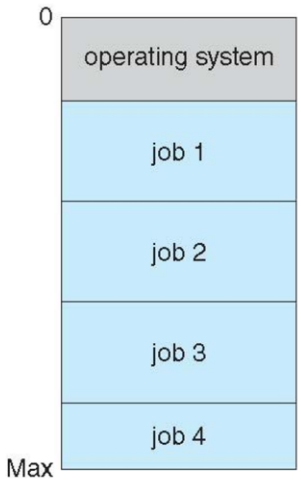
An **emulator** is hardware or software that enables one **computer** system (called the host) to behave like another **computer** system.

Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

Interactive Systems

- **Response time** should be < 1 second
- Each user has at least one program executing in memory. Such a program is referred to as a **process**
- If several processes are ready to run at the same time, we need to have **CPU scheduling.**
- If processes do not fit in memory, **swapping** moves them in and out to run
- **Virtual memory** allows execution of processes not completely in memory

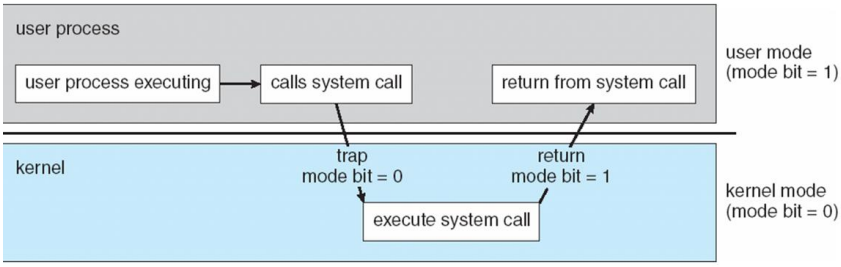
Memory Layout for Multiprogrammed System



Modes of Operation

- A mechanism that allows the OS to protect itself and other system components
- Two modes:
  - **User mode**
  - **Kernel mode**
- Mode bit (0 or 1) 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**
  - Systems call by a user asking the OS to perform some function changes from user mode to kernel mode.
  - Return from a system call resets the mode to user mode.

Transition from User to Kernel Mode



Timer

To prevent process to be in **infinite loop** (process hogging resources), a **timer** is used, which is a hardware device.

- Timer is a counter that is **decremented by the physical clock**.
- Timer is set to interrupt the computer after some time period
- Operating system sets the counter (privileged instruction)
- When **counter** reaches the value **zero**, and interrupt is generated.
- The OS sets up the value of the counter before scheduling a 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, etc.
  - Initialization data
- Process termination requires reclaim of any **reusable resources**
- A **thread** is a basic unit of **CPU utilization** within a process.
  - **Single-threaded process**. Instructions are executed sequentially, one at a time, until completion
  - Process has one **program counter** specifying location of next instruction to execute
- **Multi-threaded process** has one program counter per thread
- Typically, a system has many processes, some user, some operating system running concurrently on one or more CPUs

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

- 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

Storage Management

- OS provides uniform, logical view of information storage
- Abstracts physical properties to logical storage unit - **file**
- Files are stored in a number of different storage medium.
  - Disk
  - Flash Memory
  - Tape
- Each medium is controlled by **device drivers** (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

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 are smaller (size-wise) than storage being cached
  - Cache management important design problem
  - Cache size and **replacement policy**

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

Virtualization

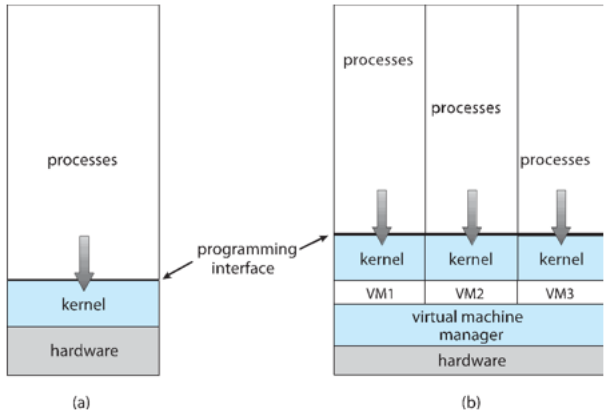
- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when the **source CPU type** is **different** from **the target type** (i.e., **PowerPC** to **Intel x86**)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Virtualization on Laptops and Destops

- A **VMM** allow the user to **install multiple operating systems** to run application written for operating systems other than the **native host**.
  - **Apple laptop** running **Mac OS X** host Windows as a guest
  - Developing apps for multiple OSes **without having multiple systems**
  - Testing applications without having multiple systems
  - Executing and managing compute environments within data centers

<<Date>>

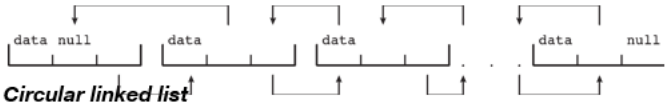
Virtualization Architecture Structure



Kernel Data Structures

- Many -- similar to standard programming data structures
- ***Singly linked list***

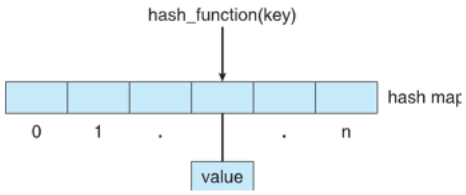
- ***Doubly linked list***



- ***Circular linked list***

Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in

  include files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming **ubiquitous** – even **home systems** use **firewalls** to protect **home computers** from **Internet attacks**

Computing Environments - Mobile

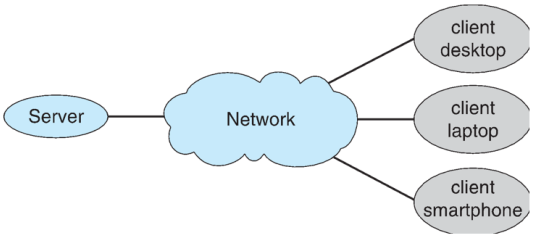
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a "traditional" laptop?
- Extra features – more OS features (GPS -- Waze)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed

- Collection of separate, possibly heterogeneous, systems networked together
  - **Network** is a communications path, **TCP/IP** most common
    - **Local Area Network** (**LAN**)
    - **Wide Area Network** (**WAN**)
    - **Metropolitan Area Network** (**MAN**)
    - **Personal Area Network** (**PAN**)
- **Network Operating System** provides features to allow sharing of data between systems across a network.
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

Computing Environments – Client-Server

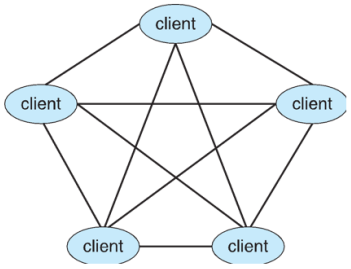
- Dumb terminals supplanted by smart PCs
- Many systems now **servers**, responding to requests generated by **clients**
  - **Compute-server system** provides an interface to client to request services (i.e., **database**)
  - **File-server system** provides interface for clients to **store and retrieve files**



Note – Have clear understanding on the difference between "Thick Client" & "Thin Client"

Computing Environments - Peer-to-Peer

- Another model of distributed system. P2P does not distinguish clients and servers
  - Instead **all nodes** are considered **peers**
  - **Each node** may act as **client**, **server**, or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - **Broadcast request** for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype



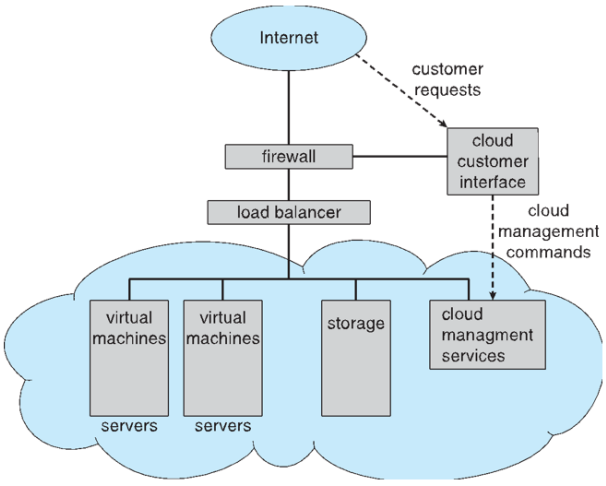
Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Computing Environments – Cloud Computing

Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools

- Internet connectivity requires security like firewalls
- **Load balancers** spread traffic across multiple applications



Computing Environments – Real-Time Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in **source-code format** rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like **VMware Player** (Free on Windows), **Virtualbox** (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

End of Chapter 1