**OPERATING SYSTEMS**

**Intro**

OS is the intermediary software between the applications on computer and the computer hardware.

**OS goals:**

* Execute user programs(applications)
* Make solving user problems easier
* Make computers convenient to use
* Utilize computer hardware and resources efficiently

**What OS does?**

Uses all existing resources to perform duties (like a manger).

Overhead: sometimes in order to do some real work it has to take resources from existing work

User view: hides lower level details

System view: resource allocator

**Definitions:**

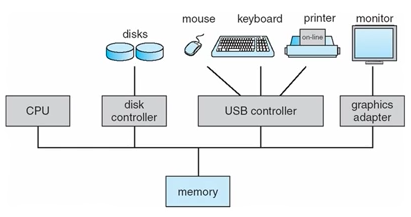
* ***Resource:*** a component needed by a process to complete its task (memory, data, file, CPU, etc.)
* ***Process*** : an active program
* ***Deadlock***: a situation where two or more processes are waiting for resources in use or each other to be completed and neither of them does.
* ***Buffer***: an area of memory to store data between two processes or between a process and a device. It is volatile in nature and may be the only copy available.
* ***Cache***: an area of memory where data can be retrieved fast. Cache coherence problem might arise since you want to work with the most up-to-date data.

**Note**: difference between cache and buffer

* ***OS Kernel***: the one process that is active all the time with some basic functionality. Everything else is either system programs or application programs.
* ***Interrupt***: change in execution. When a process is executing it has to stop so that another process can execute it is called a interrupt. OS has handlers that handle interrupts. Most OS are interrupt driven meaning they wait for something/someone to type, mouse click, etc.
* ***Storage***: they are both volatile (temporary) and persistent/non volatile (permanent).

**Computer System Organization:**

CPUs, device controllers, etc. share access to the memory using a common bus.



**Computer System Operations**

* CPU and I/O devices can operate concurrently
* Each devive in computer has a unique device controller (mostly in-built)
* Each device controller has a local buffer
* CPU moves data from/to main memory to/from local buffer
* I/O is from device to local buffer
* Device controllers inform the CPU of completion of their tasks by interrupts

**Interrupts**

* When interrupt occurs control is transferred to ISR
* OS saves the address of the interrupted process
* So mainly overhead occurs during an interrupt
* A trap or exception is a software generated interrupt caused by either an error or a user request

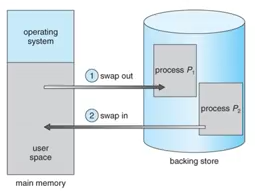
Multiprocessor

* Single-processor system: only one CPU or processing unit.
* Multiprocessor system: many CPUs or a processing unit with many cores.
  + Increased throughout
  + Also means increase sharing of resources > overhead
* Asymmetric and symmetric multiprocessors:
  + Asymmetric multiprocessors have one processor as a master and the other as slaves, hence the master delegates tasks and allocates resources.
  + Symmetric processors have all processors as peers with only OS delegating and allocating.
* Multiprocessor vs clustered systems:
  + Clustered systems are many different systems working together each at a different clock cycle but for a common goal in some sort of a network.
* Multiprocessing and multiprogramming
  + Multiprogramming has many active processes

Multiprogramming

* There are active processes present which all make use of the CPU and I/O devices. Some CPU intensive, while others are less intensive. Multiprogramming all processes make most use of CPU and devices.

Timesharing/multitasking systems

* There are many active processes all ready and waiting. The CPU scheduler decides which one to onboard to the CPU based on priority.
* Sometimes when memory is occupied for a particular process A, another process B is swapped out to make room for process A.
* 

**MIKE MURPHY**

**INTRO TO OS**

OS is a software that provides abstraction and arbitration:

* Abstraction:
  + Hides different hardware configuration details
  + Applications need not be tailored of each individual device present in the system
* Arbitration:
  + Manages access to shared hardware resources
  + Multiple applications to share the same hardware simultaneously

Hardware resources:

* CPU
* Memory
  + Registers and cache (fastest)
  + RAMs (slower)
  + Permanent memory(disk) (slowest)
* I/O devices
* Power and management:
  + Power supply
  + Internal cooling

**BOOK**

SUBDIVISIONS:

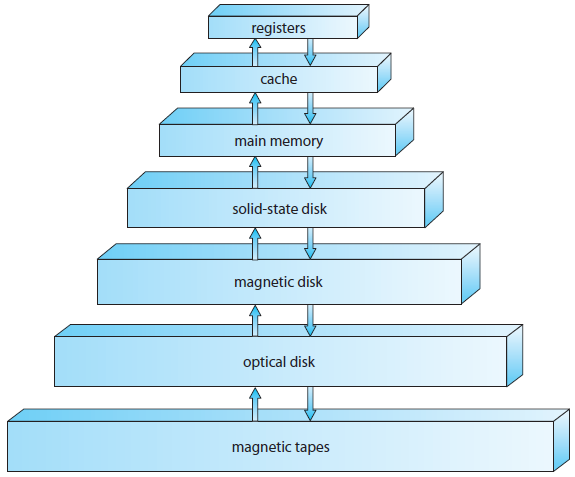
**INTRODUCTION**

**What OS do?**

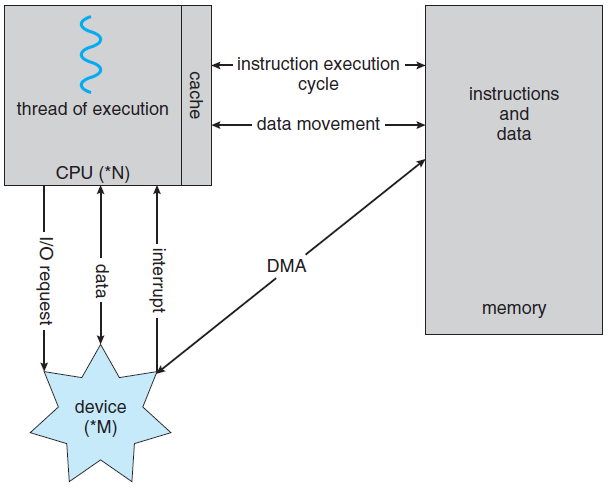
* Computer systems consists of:
  + Hardware (CPU, memory, I/O devices)
  + Applications (Word, Spreadsheets, games, etc.)
  + OS
  + Users
* User’s View:
  + For single user
    - Maximize user’s work
    - Ease of use
    - Resource allocation not a priority
  + For multiple users
    - All the above and resource allocation (CPU time, memory, I/O)
* System’s View:
  + Resource allocator
    - Involves the hardware
    - CPU time, memory space, file-storage space, I/O, etc. involved
    - Faces numerous and conflicting resource requests
  + Control program
    - Executes user programs by preventing errors and improper use of computer
    - Controlling and operating I/O devices
* Defining OS:
  + Computer hardware alone is not sufficient to execute user programs but require application software. These software require resources in memory and I/O which is provided by OS
  + Also called ‘kernel’ a program which runs at all times. There are two other programs:
    - System programs: associated with OS but not necessarily part of kernel.
    - Application programs: all programs that is not associated with operation of system.
    - Middleware: a set of software frameworks that provide additional service to app developers. Eg, iOS and Android have kernel and middlewarethat support databases, graphics and multimedia

**Computer System Organization**

* Computer System Operation
  + Computer system consists of one or more CPUs with many device controllers (each for audio, disk drives, video displays, etc).
  + ***Booting***: The initial program to be run is ***bootstrap program***, stored in ROM or EPROM, known as firmware. It initializes everything in the system and loads the OS by locating the kernel and loading it to memory.
  + Once kernel is loaded it can provided services to users. Some programs (system programs) are loaded outside kernel but run the entire time as long as kernel is running called system processes.
  + ***Interrupts***: on occurrence of interrupt, execution moves to a fixed location containing the service routine for that interrupt. The ISR execute and on completion, the interrupted operation resumes. Since only predefined number of interrupts are present the ISR is called through a pointer to have high speed. The table of pointers are stored in lower memory, which hold addresses of the ISR for the device. After the interrupt is serviced, the saved return address is loaded into the program counter, and the interrupted computation resumes as though the interrupt had not occurred.
* Storage structure
  + General purpose computers run most programs from RAMs
  + ROM used for static programs (bootstrap program and game cartridges) Smartphones have factory installed programs on EEPROM
  + ***Load*** operation from main memory to internal register in CPU
  + ***Store*** operation from register to main memory
  + Execution cycle:
    - fetch instruction from memory and load it into the instruction register
    - Instruction is decoded to cause operands to be fetched from memory to be load in internal register
    - The result is stored back in memory.
  + Secondary storage: for data and programs, more storage and not volatile.



* + - There is trade off in speed, expense per bit and storage, permanency of data from top to bottom.
* I/O Structure:
  + A general computer system has a CPU and many device controllers each for a specific type of device.
  + I/O Operation:
    - Device driver loads appropriate register within the device controller (I/O req)
    - Device controller itself examines the registers to determine what action to take (read a character from the keyboard)
    - Device controller sarts transfer of data to the local buffer
    - Device controller informs driver it has finished the transfer via interrupt
    - 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
    - This will work for small amounts of data, for larger amount we will need DMA.
    - After setting buffer, pointer and counters in I/O device the device controller itself transfers the required data from local buffer to the memory without CPU intervention. Only interrupt per block is generated rather than one per byte.



* **Computer System Architecture**
  + Single-processor systems
    - Computers with a single processor (single CPU).
    - There might be other special-purpose processor (for disk, keyboard, etc.)
    - Sometimes an OS manages them regarding next instruction
    - Sometimes the special-purpose processor relieve OS of their tasks (eg: disk-processor receives a sequence of requests from the main CPU and implements its own disk queue and scheduling algorithm.
  + Multi-processor systems: