Lecture.1.Overview.txt

- Operating System
    - Is a collection of programs that manage hardware resources
    - Is a program that controls the execution of application programs
    - Is an interface between applications and hardware
    - Operating Systems should be convenient to use, efficient when dealing with computations of any kind, and able to evolve/adapt to changing requirements

- OS Definition
    - By definition, all operating systems should contain:
        - Kernel
            - It is the main part of an operating system
            - Runs all the time
            - Allocates hardware resources to software
        - System Programs
            - Sits ideally until invoked
            - i.e. ls, rm, format, etc.
        - Middleware
            - Computer software that provides services to software applications beyond those available from the OS
                - i.e. Multi-media, special GUI, special relational database, etc.
                - i.e. Game Development Engine (Unity), Java Data-Base Connectivity (JDBC)
        - Note: Application programs (i.e. Word, Keynote, etc.) are not associated with operating systems, and are not part of the middleware
    - Everything listed above should be included when you buy an OS

- Computer System Operation
    - I/O devices, and the CPU can execute in parallel
    - Each device controller has a local buffer and is in charge of a particular device type
    - Each device controller type has an operating system device driver to manage it
    - CPU moves data from/to main memory to/from local buffers
    - Device controller informs CPU that it has finished its operation by causing an interrupt
        - Interrupts are very important because operating systems are driven by interrupts

- Key Aspects Of The System
    - Interrupts
    - Storage Structure
    - I/O Structure

- Interrupts
    - An interrupt transfers control to the interrupt service routine

        - Generally this occurs through the interrupt vector
            - The interrupt vector contains the addresses of all the service routines
    - The interrupt architecture must save the address of the interrupted instruction
        - The stack is usually used to store this information
    - A trap or exception is a software generated interrupt caused by an error or a user request
        - i.e. Error: Division By Zero,
               Error: Array Out Of Bounds Index
    - An operating system is interrupt driven

- I/O Interrupts
    - The interrupt handler signals the user process upon the completion of I/O
    - There are two kinds of interrupts: Synchronous & Asynchronous
        - Synchronous
            - The user process makes an I/O request and waits until the I/O is completed
                - This is very costly, because the process may wait for a long time
                - Typically, I/O devices consume a lot of time
            - The user process must know the I/O latency, so it knows how long it should wait
                - This is required because if the I/O fails, then the system will be waiting forever and fail to function
        - Asynchronous
            - An I/O request returns without waiting for the I/O completion
                - The interrupt is scheduled at the completion of I/O
                - The interrupt handler signals the user process when the I/O is done
            - This allows for concurrent I/O operations on several devices
                - Asynchronous is how most systems are implemented

- Interrupt Handling
    - The operating system preserves the state of the CPU by storing registers, and the program counter system
        - The OS determines the type of interrupt that has occured
            1. Polling
                - During a polling interrupt, the registers are examined to determine what type of interrupt has happened/occured
            2. Vectored Interrupt System
                - The Interrupt Vector Table contains addresses of interrupt service routines
                - Much faster than Polling
    - The CPU can enable or disable certain kinds of interrupts
        - Maskable Interrupts can be turned off by the CPU

        - Non-Maskable Interrupts cannot be turned off by the CPU, and they will always happen
            - i.e. Unrecoverable memory errors; Writing to memory you don't have access to
    - The Interrupt Response Time is the time that elapses between the interrupt signal and the execution of the first statement in the corresponding interrupt handler
        - This time needs to be as small as possible
        - This time can be used to measure Interrupt Efficiency

- Classes Of Interrupts
    - Program
        - Generated/caused by:
            - Arithmetic overflow
            - Division by zero
            - Referencing memory you don't have access to
            - Etc.
    - Timer
        - Generated by a timer within the processor
            - Each CPU core has a timer associated with it
    - I/O
        - Generated by an I/O controller
            - Called when an I/O completes an operation
    - Hardware
        - Caused by power failure or memory parity error
    - Program interrupts and hardware interrupts are not good interrupts
        - These interrupts should be avoided when the system is running
    - It is possible to have nested interrupts
        - i.e. You tell Keynote to print a file ->
          The printer starts but runs out of paper ->
          The operation times out so it starts again ->
          Return to user program (Keynote)

- Storage Structure
    - Main Memory
        - Random Access Memory (RAM)
            1. DRAM
                - This stands for Dynamic RAM
                - Needs to be refreshed every period in order to get information saved
            2. SRAM
                - This stands for Static RAM
                - Much faster than DRAM, but more expensive
        - Main memory is also referred to as primary storage
    - Secondary Storage
        - Extension of main memory that provides large non-volatile storage capacity
            - It retains stored data even when powered off

        - i.e.
            - Hard Disk Drives (HDD)
                - An electro-mechanical data storage device
            - Non-volatile memory (NVM)
                - i.e. Solid-State drive (SSD) is a type of non-volatile storage
                - The information is kept even after the system is turned off
        - On the other hand, in volatile-storage, the information is lost once the system is turned off
            - i.e. Random Access Memory (RAM)
    - The Registers in a CPU are the fastest memory in a computer
        - However, registers are very limited, and you cannot put all your data into them
        - Before cache, working data was primarily stored in main memory (RAM)
            - RAM is much slower than Registers
        - Cache memory is faster than main memory (RAM), but slower than Registers

- Storage Hierarchy
    - Memory is that closer to the processor is faster to access and process
        - It is also more expensive and smaller in size

- I/O Techniques
    - When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module
        - i.e.
            - Programmed I/O
            - Interrupt Drive I/O
            - Direct Memory Access (DMA)

- Programmed I/O
    - How it works
        1. The I/O module performs the requested action then sets the appropriate bits in the I/O status register
        2. The processor periodically checks the status of the I/O module until it determines that the instruction has completed
    - With programmed I/O the performance level of the entire system is severely degraded
        - This is because the checks are not instant, they are done in intervals
    - i.e. Microsoft's Windows checking for updates in the background
    - i.e. Mail App on iOS checking for new mail every X minutes in the background

- Interrupt-Driven I/O

    - How it works:
        1. Processor issues an I/O command to module
        2. I/O module will interrupt the processor service when it is ready to exchange data
        3. The processor executes the data transfer
    - Interrupt-Driven I/O is much faster than programmed I/O, but still requires active intervention of the processor to transfer data between I/O module and memory
        - Most implementation of I/O is interrupt-driven
            - Interrupt-Driven I/O is the best choice
                - However, it is not fast enough for a huge amount of data/information
    - i.e. Turning on a microphone causes the OS to dedicate more resources to it
    - i.e. Manually checking for updates OR manually checking for new mail/messages

- Direct Memory Acess (DMA)
    - Performed by a separate module on the system bus or incorporated into an I/O module
    - Transfers the entire block of data directly to and from memory without going through the processor
        - The processor is involved only at the beginning and end of the transfer
        - The processor executes more slowly during a transfer when processor access to the bus is required
        - DMA transfers memory from I/O device to main memory, or vice versa
    - DMA is more efficient than interrupt-driven or programmed I/O

- Modern Computer Architecture
    - Von Neumann
        - In this architecture, one bus is used for both data transfers and instruction fetches
    - Harvard
        - The data buses are separated from the instruction buses
    - Modern CPUs use multi-processor and multi-core architecture

- Computer System Architecture
    - Single general-purpose processor system
    - Multiprocessor systems
        - Advantages: Increased throughput, economy of scale, increased reliability, etc.
        - Issues: Data consistency, Load balancing, I/O Bottleneck, Cache coherency, etc.
            - Load balancing: One processor cannot do all the work or the bulk of it
            - Cache coherency: Occurs when two processors are using the same cache, and one processor modifies the data. The first processor needs to alert the other processor

that the data has been changed, before it starts working with the old data
- Types of multiprocessor systems:
1. Asymmetric multiprocessing
- Each processor is assigned a specific task
2. Symmetric multiprocessing
- Each processor performs all tasks

- Non-Unified Memory Access System (NUMA)
- Adding additional CPUs does not scale very well
- The system bus becomes a bottleneck
- To solve this issue, each CPU, or group of CPUs, are provided with their own memory
- All CPUs share one physical address space

- Clustered Systems
- Contains multiple nodes with multiple micro-processors
- All of them share storage via a storage-area network (SAN)
- Types:
- Asymmetric
- Clustering has one machine in hot-standby mode
- A node is standing by to take over if another node fails
- i.e. Cellular Phone Service
- If one tower is bust, another nearby tower will take its place
- Symmetric
- Cluster has 'n' multiple nodes running applications, monitoring each other

- Operating System Operations
- Bootstrap Program
- Simple code to initialize the system, and load the kernel
- Kernel
- Starts services provided outside of the kernel
- i.e. System daemons like syslogd, sshd, etc.
- The kernel is interrupt driven
- Hardware interrupt
- i.e. By an I/O device
- Software interrupt
- i.e. Exception or trap
- i.e. Accessing memory you do not have permission for (i.e. Segmentation fault)
- The kernel is always running and transitions from user mode to kernel mode if there is an interrupt

- Caching
- Information being used is temporarily copied from slower to faster storage
- i.e. Hard Drive to RAM to Cache

    - The faster storage is first checked to see if the information is already there
        - If it is, use the information
            - This is a cache hit
        - If not, copy the data from the primary/secondary memory to the cache
            - This is a cache miss
    - Multiprocessor environments must provide cache coherency in hardware
        - This ensures that all CPUs have the most recent value in their memory

- Characteristics Of Different Storage
    - Registers are managed by the compiler
    - Cache is managed by hardware
    - The OS manages main memory and secondary memory
    - The implementation technology for Registers, Cache, and RAM is CMOS (SRAM)

- Modern Computer Motherboard
    - Even the cheapest modern general purpose computer CPU contains multiple cores
        - Multicore programming is the only way forward for future software
            - i.e. Simulating the universe (or a galaxy)
            - i.e. Performing extremely large and complex computations
                - i.e. Protein folding