Lecture 01

Introduction

What Operating Systems Do

- Computer system divided into 4 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
- **Hardware**, consisting out of: Central Processing Unit (CPU); Memory; Input/Output (I/O) devices, provides the basic computing resources for the system.
- **Application programs** define the ways in which these resources are used to solve users' computing problems.
- The **operating system** controls the hardware and coordinates its use among the various application programs.
- 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.
- The operating system from two view points:
  - User View
  - System View

User View

- Users view varies according to interface used.
- Some operating systems are designed for **ease of use** with some attention paid to performance and none paid to resource allocation.

- ☐ These systems are designed for the single-user experience.

- Some operating systems are designed to maximize resource utilization to assure that all available CPU time, memory, and I/O are used efficiently and no individual user takes more than his share.
  - ☐ These are multi-user systems where terminals are connected to mainframe or minicomputers.
  - ☐ users share resources and may exchange information.
- In some cases users sit at workstations connected to networks of other workstations and servers.
  - ☐ These systems have dedicated resources such as networking and servers.
  - ☐ These operating systems compromise between individual usability and resource utilization.

System View

- The program that is most intimately involved with the hardware.
- The operating system is a resource allocator.
- The following resources may be required to solve a problem:
  - ☐ CPU time
  - ☐ memory space
  - ☐ file-storage space
  - ☐ I/O devices
  - ☐ etc.
- The operating system acts as the manager of these resources.
- A different view of an operating system emphasizes the need to control the various I/O devices and user programs. The operating system as a control program.
  - ☐ A control program manages the execution of user programs to prevent errors and improper use of the computer.
  - ☐ It is especially concerned with the operation and control of I/O devices.

Defining Operating Systems

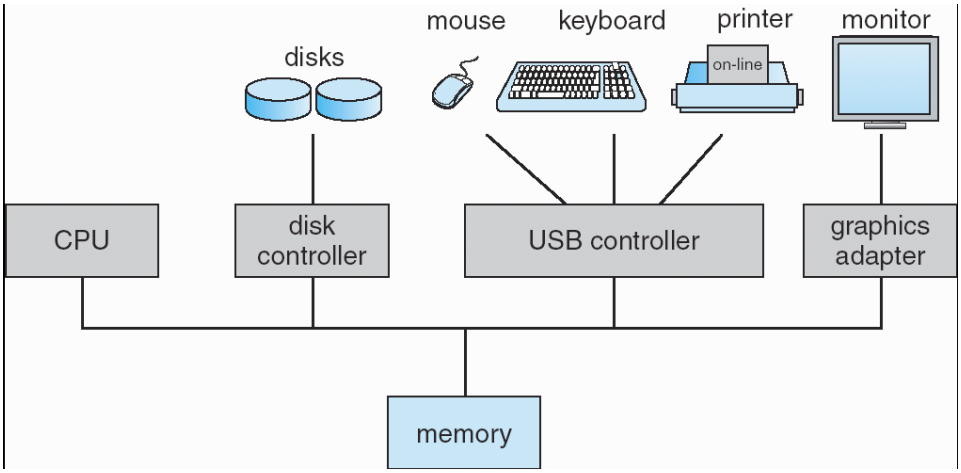
- There is no real definition for an Operating System.
- The goal of an operating system is to execute programs and to make solving user problems easier.
- The computer hardware is constructed toward this goal.
- Because hardware alone is not easy to use, application programs are developed.
- These programs require common operations, such as controlling I/O.
- These common functions of controlling and allocating resources are then brought together into one piece of software: the operating system.

- The definition we use here is as follows:
  - 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:** associated with the operating system but not part of the kernel.
  - **Application programs:** include all programs not associated with the operation of the system.

Computer-System Organization

- Computer-system operation

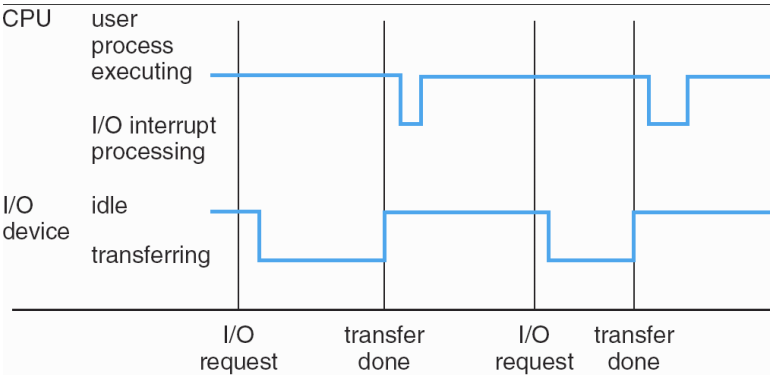
- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory



cycles

Computer-System Operation

- For a computer to start running it needs an initial program to run at boot time.
  - This initial program or **bootstrap** program tends to be simple.
  - It is stored in ROM or EEPROM and is known as firmware within the computer hardware.
  - It initializes all aspects of the system.
  - The bootstrap must know how to load the operating system. To accomplish this the bootstrap program must locate and load the operating system kernel into memory.
- The occurrence of an event is usually signaled by an **interrupt** from either hardware or software.
  - Hardware trigger an interrupt by sending a **signal** to the CPU.
  - Software may trigger an interrupt by executing a special operation called a **system call or monitor call**.
  - Look at fig 1.3 p.9 for a timeline of the interrupt operation.



  - Since only a predefined number of interrupts are possible, a table of pointers to interrupt routines is used to increase speed.
  - The table of interrupt pointers is stored in low memory.
  - These locations keep the addresses of the interrupt service routines for

the various devices.

- This array or interrupt vector is then indexed by a unique device number. This number is given with the interrupt request to provide the address of the interrupt service routine for the interrupting device.

- The CPU and device controllers (each in charge of a certain type of device) are connected to shared memory through a common bus
- The CPU and device controllers can execute concurrently, competing for memory cycles
- A memory controller synchronizes access to the memory
- Bootstrap program = a simple initial program that runs when the computer is powered up, and transfers control to the OS
- Modern OSs are interrupt driven: If there is nothing for the OS to do, it will wait for something to happen
- Events are almost always signaled by an interrupt or a trap:
- Hardware interrupts usually occur by sending a signal to the CPU
- Software interrupts usually occur by executing a system call
- Trap = a software-generated interrupt caused by an error / a request from the program that an OS service be performed

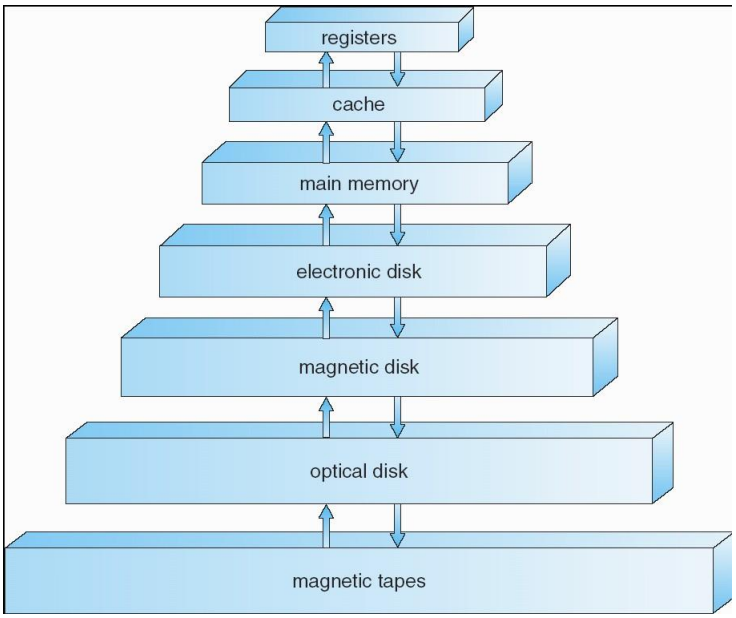
Storage Structure

- General purpose computers run their programs from random-access memory (RAM) called main memory.
  - ☐ Main memory is implemented using **dynamic random-access memory (DRAM)** technology.
- Interaction with memory is achieved through a sequence of load and store instructions to specific memory addresses.
  - ☐ Load instruction moves a word from main memory to an internal register within the CPU.
  - ☐ Store instruction moves content of a register to main memory.
  - ☐ The CPU automatically loads instructions from main memory for execution.
- Instruction-execution cycle as executed by von Neumann architecture system:
  - ☐ Fetch instruction from memory and stores instruction in the instruction register.
  - ☐ Decodes instruction and may cause operands to be fetched from memory and store in some internal register.
  - ☐ After instruction on operands executed, result is stored back in memory.
- The memory unit only sees a stream of memory addresses; it doesn't know they are generated.
  - ☐ We are interested only in the sequence of memory addresses generated by the running program.
- Ideally we want programs and data to reside in main memory permanently, but it is not possible for the following two reasons:
  - ☐ Main memory is to small to store all needed programs and data

permanently.

- Main memory is a **volatile** storage device that loses its contents when power is turned off or otherwise lost.

- For this reason most computer systems provide **secondary storage** as an extension of main memory.
  - The main requirement of **secondary storage** is that it must **hold large quantities** of data.
  - Most common secondary storage device is magnetic disk which provide storage for both programs and data.
  - There are other types of secondary storage systems of which the speed, cost, size, and volatility differ.
  - Look at fig 1.4 p.11 for the storage hierarchy.



- *Caching*–copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

- 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 smaller than storage being cached
  - ☐ Cache management important design problem
  - ☐ Cache size and replacement policy
- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

I/O Structure

- Each device controller is in charge of a specific type of device
- A SCSI (small computer-systems interface) controller can have 7 or more devices attached to it
- A device controller maintains some buffer storage and a set of special-

purpose registers

- It moves data between the peripherals and its buffer storage
- I/O interrupts
  - ☐ Starting an I/O operation:
    - The CPU loads the appropriate registers in the device controller
    - The device controller examines the contents of these registers to see what action to take
    - Once the transfer of data is complete, the device controller informs the CPU that it has finished, by triggering an interrupt
- Synchronous I/O: Control is returned to the user process at I/O completion
  - ☐ To wait for I/O completion, some machines have a 'wait' instruction, while others have a wait loop: 'Loop:jmp Loop'
  - ☐ Advantage: The OS knows which device is interrupting
  - ☐ Disadvantage: No concurrent I/O operations to many devices
  - ☐ Disadvantage: No overlapping useful computation with I/O
- Asynchronous I/O: Control is returned to the user process without waiting for I/O to complete
  - ☐ A device-status table is used to keep track of I/O devices
  - ☐ Each table entry shows the device's type, address, & state
  - ☐ If other processes request a busy device, the OS maintains a wait queue
  - ☐ When an interrupt occurs, the OS determines which I/O device caused the interrupt and indexes the table to determine the status of the device, and modifies it
  - ☐ Advantage: increased system efficiency
- DMA structure
  - ☐ DMA is used for high-speed I/O devicesA program or the OS requests a data transfer
  - ☐ The OS finds a buffer for the transfer
  - ☐ A device driver sets the DMA controller registers to use appropriate source & destination addresses
  - ☐ The DMA controller is instructed to start the I/O operation
  - ☐ During the data transfer, the CPU can perform other tasks
  - ☐ The DMA controller 'steals' memory cycles from the CPU (which slows down CPU execution)
- The DMA controller interrupts the CPU when the transfer has been completed
- The device controller transfers a block of data directly to / from its own buffer storage to memory, with no CPU intervention
- There is no need for causing an interrupt to the CPU
- The basic operation of the CPU is the same:

Computer-System Architecture

Single-Processor Systems

- On a single-processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes

Multiprocessor Systems

- Several processors share the bus, clock, memory, peripherals…

- 3 main advantages:
  - ☐ Increased throughput
    - More processors get more work done in less time
  - ☐ Economy of scale
    - You save money because peripherals, storage, & power are shared
  - ☐ Increased reliability
  - ☐ Failure of one processor won't halt the system
- Graceful degradation = continuing to provide service proportional to the level of surviving hardware
- *Tandem system*
  - ☐ 2 identical processors (primary + backup) are connected by a bus
  - ☐ 2 copies are kept of each process, and the state information of each job is copied to the backup at fixed checkpoints
  - ☐ If a failure is detected, the backup copy is activated and restarted from the most recent checkpoint
  - ☐ Expensive, because of hardware duplication
- *Symmetric multiprocessing (SMP)*
  - ☐ Used by the most common multiple-processor systems
  - ☐ Each processor runs an identical copy of the OS, and these copies communicate with one another as needed
  - ☐ Processes and resources are shared dynamically among processors
  - ☐ Advantage of SMP: many processes can run simultaneously without causing a significant deterioration of performance
  - ☐ Disadvantage of SMP: Since the CPUs are separate, one may be idle while another is overloaded, resulting in inefficiencies
- *Asymmetric multiprocessing*
  - ☐ Each processor is assigned a specific task
  - ☐ A master processor controls the system and the others either look to the master for instruction or have predefined tasks
  - ☐ Master-slave relationship: The master processor schedules & allocates work to the slave processors
  - ☐ As processors become less expensive and more powerful, extra OS functions are off-loaded to slave processors (back ends)
    - E.g. you could add a microprocessor with its own memory to manage a disk system, to relieve the main CPU

Cluster Systems

- Multiple CPUs on two / more individual systems coupled together
- Clustering is usually performed to provide high availability

- A layer of cluster software runs on the cluster nodes
- Each node can monitor the others, so if the monitored machine fails, the monitoring one can take over
- *Asymmetric clustering*
    - ☐ A hot standby host machine and one running the applications

- ☐ The hot standby host just monitors the active server
- ☐ If that server fails, the hot standby host à active server

- *Symmetric mode*
  - ☐ Two / more hosts run applications and monitor each other
  - ☐ More efficient mode, as it uses all the available hardware
  - ☐ Requires that more than one application be available to run
- *Parallel clusters*
  - ☐ Allow multiple hosts to access same data on shared storage
- Most clusters don't allow shared access to data on the disk
- Distributed file systems must provide access control and locking
- DLM = Distributed Lock Manager
- Global clusters: Machines could be anywhere in the world
- Storage Area Networks: Hosts are connected to a shared storage

Operating System Structure

- Multiprogramming
  - ☐ Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute
  - ☐ The OS keeps several jobs in memory and begins to execute one of them until it has to wait for a task (like an I/O operation), when it switches to and executes another job
  - ☐ Job scheduling = deciding which jobs to bring into memory if there is not enough room for all of them
  - ☐ CPU scheduling = deciding which job to choose if several are ready to run at the same time
- Time-sharing (multitasking) systems
  - ☐ Like multiprogramming, but *interactive* instead of batch!
  - ☐ Interactive computer system: direct communication between user & system, where the user expects immediate results
  - ☐ Time-sharing: many users can share the computer simultaneously
  - ☐ The CPU switches among multiple jobs so frequently, that users can interact with each program while it is running
  - ☐ CPU scheduling and multiprogramming provide each user with a small portion of a time-shared computer
  - ☐ Process = a program that's loaded into memory and executing
  - ☐ For good response times, jobs may have to be swapped in & out of main memory to disk (now serving as a backing store for memory)
  - ☐ Virtual memory = a technique that allows the execution of a job that may not be completely in memory
  - ☐ Advantage of VM: programs can be larger than physical memory

- ☐ Time-sharing systems must also provide a file system
- ☐ The file system resides on a collection of disks, so disk management must be provided
- ☐ Concurrent execution needs sophisticated CPU-scheduling schemes

- ☐ The system must provide mechanisms for job synchronization & communication and ensure that jobs don't get stuck in a deadlock