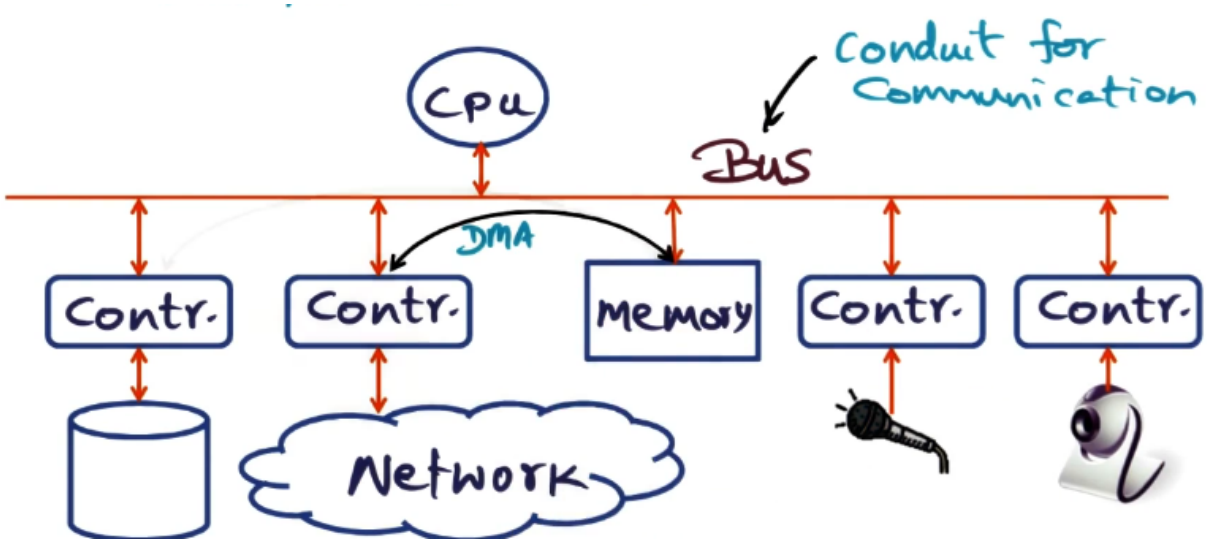
Introduction to AOS

Principle of Abstraction

1. Abstraction: Well-understood interface that hides all the details within a subsystem
   - Examples of abstractions:
     - Door knob
     - Instruction set architecture
     - AND logic gate
     - Transistor
     - INT data type
   - Abstractions between Google Earth and the electrons on the silicon
     - Transistors
     - Logic gates
     - ALU, Memory
     - Processor
     - Instruction set architecture
     - Operating system
     - Applications

Hardware Resources

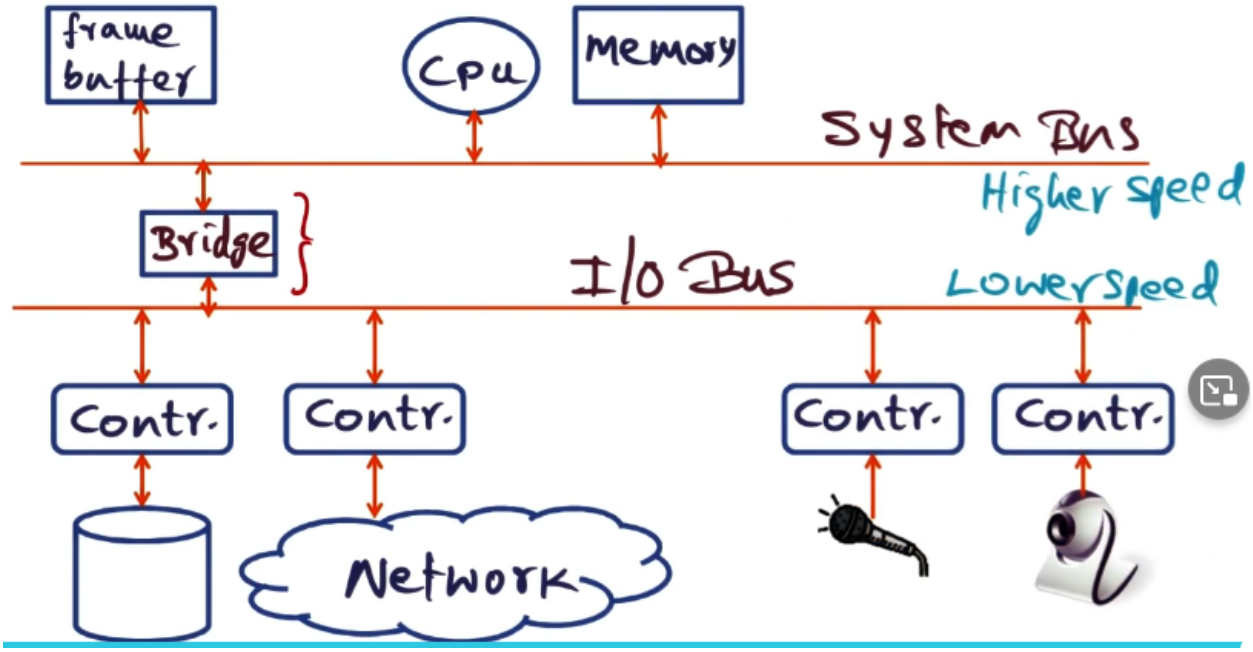
1. Hardware consists of processor, memory, and IO devices
   - Organization of these components is fairly constant across devices
2. Internal organization
   - CPU connected to a bus
   - Memory connected to same bus
   - Storage and other peripheral devices accessible through bus
   - Network interface through a controller
   - DMA can allow controllers to interface with memory directly instead of through the shared bus



Hardware Organization

3. Organization with I/O Bus
   - Separate bus for I/O from system with bridge connecting them
   - Intent is that cumulative need for bandwidth of individual devices is less than the bandwidth needed by the memory/CPU

- System bus is higher speed, I/O bus is lower speed
- Bridge arbitrates DMA or communication with CPU



Hardware Organization with I/O Bus

4. Specifics of computational power, memory capacity, and number/types of I/O devices may vary from one computer system to the next, but the underlying organization is similar

OS Functionality

1. Functionality required of operating systems
   - OS is a resource manager
   - OS provides a consistent interface to the hardware resources
   - OS schedules applications on the CPU
2. Definition of OS
   - Protected acccess to hardware resources
   - Arbitrate among competing requests
   - Well-defined APIs for OS services
3. What happens when a mouse is clicked?
   - Hardware controller interfacing with mouse raises an interrupt on the interrupt line (dedicated line on bus)
     - Interrupt is a hardware mechanism that alerts the OS that some piece of hardware needs attention
   - OS fields the interrupt
     - OS schedules itself to run on the CPU in addition to applications
   - OS passes it to the program for appropriate action for this particular interrupt

Managing the CPU and Memory

1. Different programs all need access to hardware without hogging the CPU or overwriting code/data of other programs
   - OS needs to manage this process by scheduling itself on the CPU, but needs to minimize its time spent consuming cycles

2. How can one CPU run several programs in parallel with only one CPU?
    - Applications share the CPU through the operating system, but only one is running at a time. OS multiplexes the CPU
3. Resource needs of applications include CPU, memory, peripheral devices
    - At the time of launch, OS knows enough about the application to create a memory footprint (stack, heap, global data, code)
    - Application asks for additional resources at runtime
    - OS only interferes only to arbitrate resources needed by application and gets out of the way as quickly as possible
4. Processor-related OS abstractions
    - Program: Static image loaded into memory of an application
    - Process: A program in execution (process = program + state)
        - State is continuously evolving as the process executes
5. Process vs Thread
    - A process is a program plus the state of all the threads executing in the program
6. Memory-Related OS Abstraction
    - Address space for each process is distinct from one another
    - Implement the abstraction using hardware capabilities