

Introduction to AOS

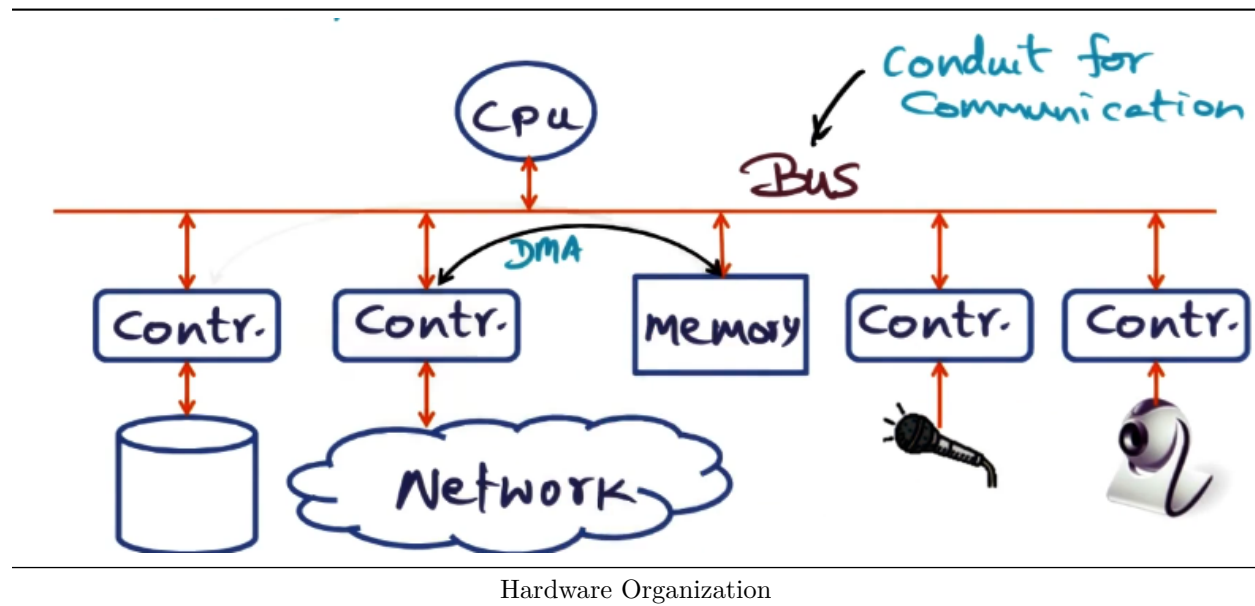
Principle of Abstraction

* Abstraction: Well-understood interface that hides all the details within a subsystem

- Examples of abstractions:
 1. Door knob
 2. Instruction set architecture
 3. AND logic gate
 4. Transistor
 5. 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

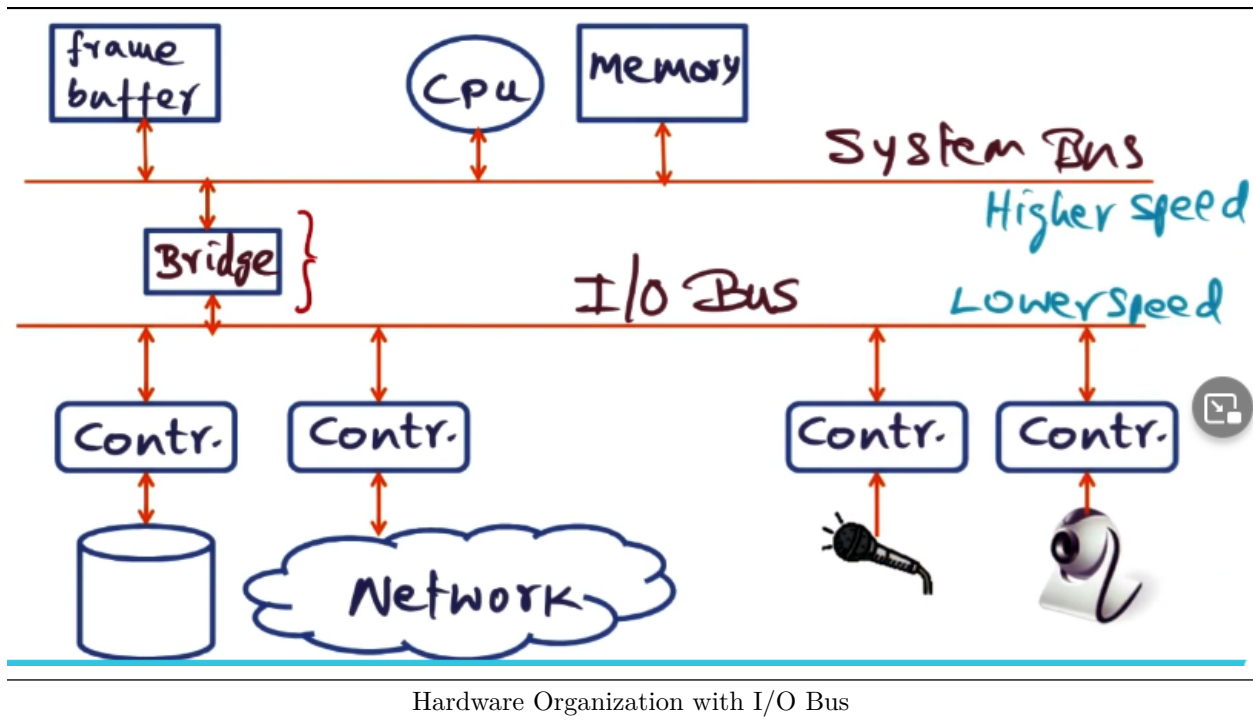
Hardware Resources

- * Hardware consists of processor, memory, and IO devices
 - Organization of these components is fairly constant across devices
- * 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



- * 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



* 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

- * Functionality required of operating systems
 1. OS is a resource manager
 2. OS provides a consistent interface to the hardware resources
 3. OS schedules applications on the CPU
- * Definition of OS
 - Protected access to hardware resources
 - Arbitrate among competing requests
 - Well-defined APIs for OS services
- * What happens when a mouse is clicked?
 1. 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
 2. OS fields the interrupt
 - + OS schedules itself to run on the CPU in addition to applications
 3. OS passes it to the program for appropriate action for this particular interrupt

Managing the CPU and Memory

- * 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
- * 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
- * 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
- * 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
- * Process vs Thread
 - A process is a program plus the state of all the threads executing in the program
- * Memory-Related OS Abstraction
 - Address space for each process is distinct from one another
 - Implement the abstraction using hardware capabilities