CS 303: Operating Systems Lecture Set 1

Instructor
Gourinath B.
POD#1D Room#307
IIT Indore

Introduction

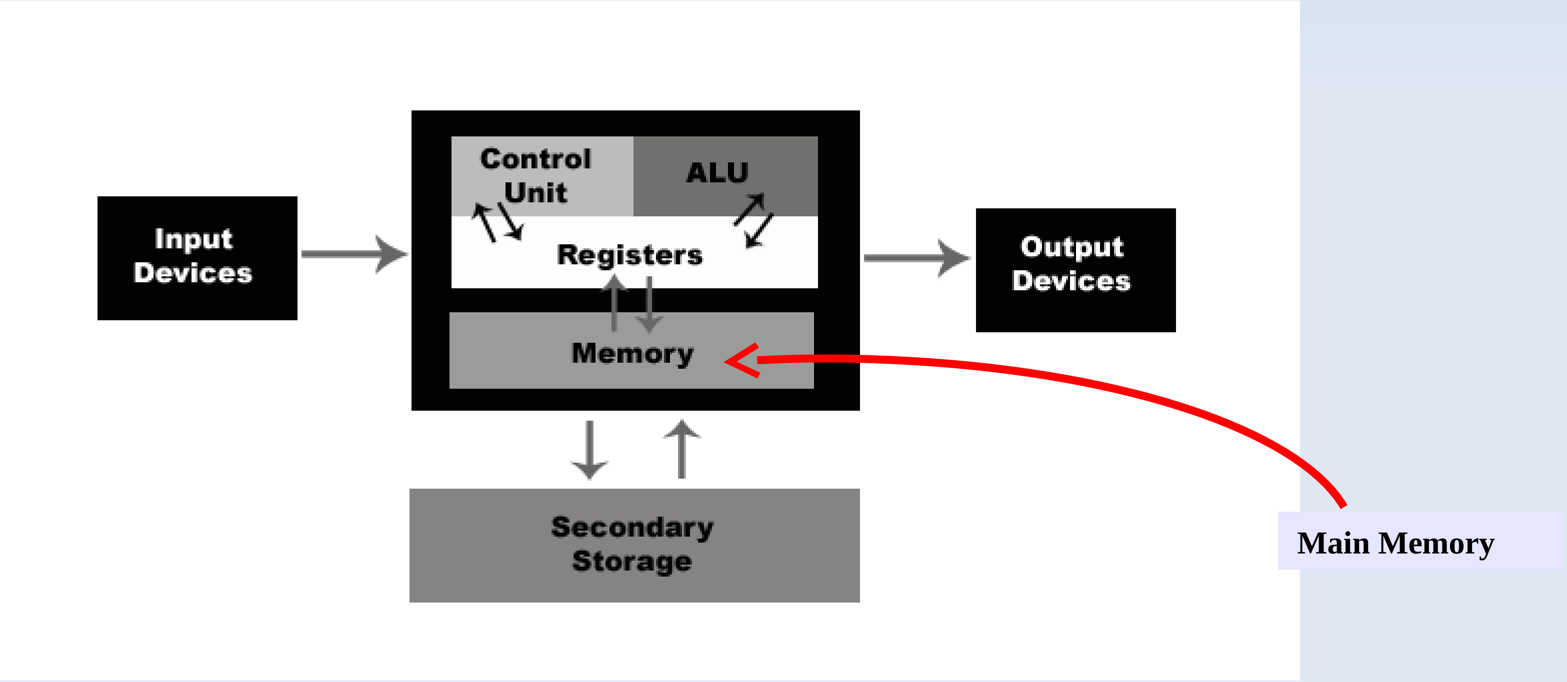
Objective:

- Overview of the Computer system organization
    - processor (CPU), memory, and input/output, architecture and general operations.

- Understand its role and principal functions.

Computer System Organization
(Abstract view)

Typically, a common bus for all!



Requires: Dev. Controller HW for delegation → Dev. Driver (SW) is required

Program and types

- What is a (software) program?
    - Many definitions …..

()

- Niklaus Wirth [Turing Award Winner 1984]

  ALGORITHM/S + DATA STRUCTURE/S = PROGRAM/S [1976]

- Kowalski [published in 1979]

  ALGORITHM = LOGIC + CONTROL

- PROGRAM = LOGIC + CONTROL + DATA STRUCTURES
- Software means several programs
- To execute, it needs a processing platform i.e. Hardware
- Software (S/W) : System S/W, Application S/W

Program

- Typically, programs are written in High-level languages
  - Then compiled into binary executable
  - This executable is run
- Compilation involves the steps of:

  1. Pre-processing 2. Compilation 3. Assembly and 4. linking
- Consider compiling the C program for adding two numbers
  - `:~$gcc -Wall addTwoNums.c -o addTwoNums`
- Consider compiling to preserve the intermediate temporary files generated
  - `$ gcc -Wall --save-temps addTwoNums.c -o addTwoNums`

Intermediate Program Files

1. Pre-processing: .i `addTwoNums.i`
2. Compilation: .s `addTwoNums.s`
3. Assembly: .o `addTwoNums.o`
4. linking: exe `addTwoNums`

- Contents of .o and executables are not in standard text as they are machine codes
- Can be read with `readelf`
- `$ readelf –a addTwoNums.o` (to view the object/assembled file)

Program Execution

Executes on the intended microprocessor

- Illustrate
- Machine code is executed

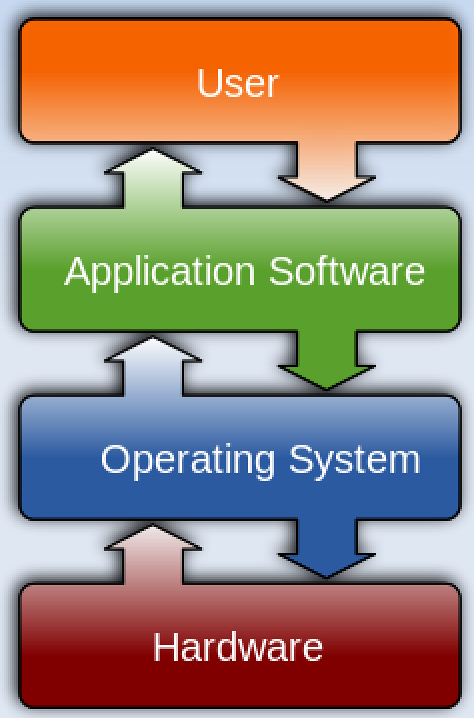
Program and types

- Software types: (based on functionality)

  System Software and Application Software (+3rd type)

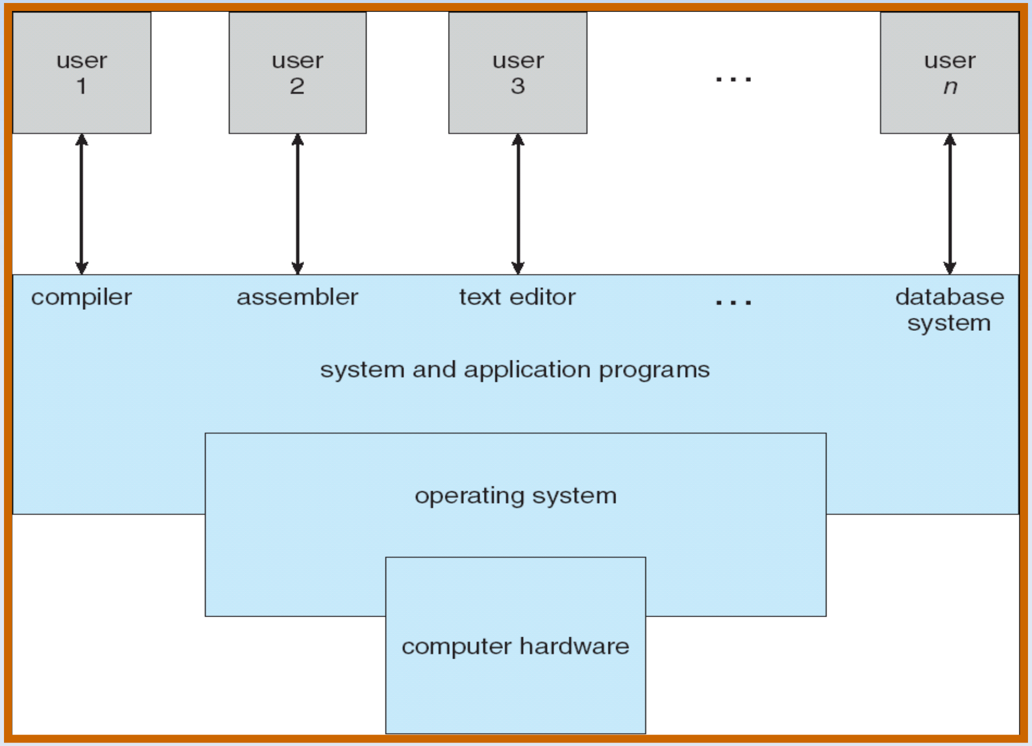
- Application Software (serves the user)
  - Uses/makes the computer system to perform special functions
    - Target platform: Either HW or SW
    - Ex: Image editors, Browsers, Doc editors, etc.
- System software (facilitates the usage of H/W)
    - directly operates the computer hardware, to provide basic functionality needed by other appli. software
    - Provide a platform for running Appli. Software
    - Ex: OS, DD, System Maintenance utilities, etc.

What is an Operating System?



OS is SW managing computer hardware to provide computer programs with necessary execution environment.

What is an Operating System?



OS is SW managing computer hardware to provide computer programs with necessary execution environment.

Why Operating Systems?

- Single user case (Ex. desktop/PDA)?
  - hardware abstraction
    - Helps to write code/applications that is NOT hardware specific!
  - reusable part implemented in OS
    - While application does the essential
- - Need to run several applications concurrently

- If there are several users/applications sharing the computer- (Ex: Mainframes, servers, etc.)
  - something has to manage:
  - who should get which resources? when?
    - what is each user/application allowed to do?

What Operating Systems Do?

Several perspectives -

- OS is program most involved with the hardware
  - hardware abstraction
- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System

- The low-level software on a computer which provides user-programs (i.e. applications) with necessary execution environment by:

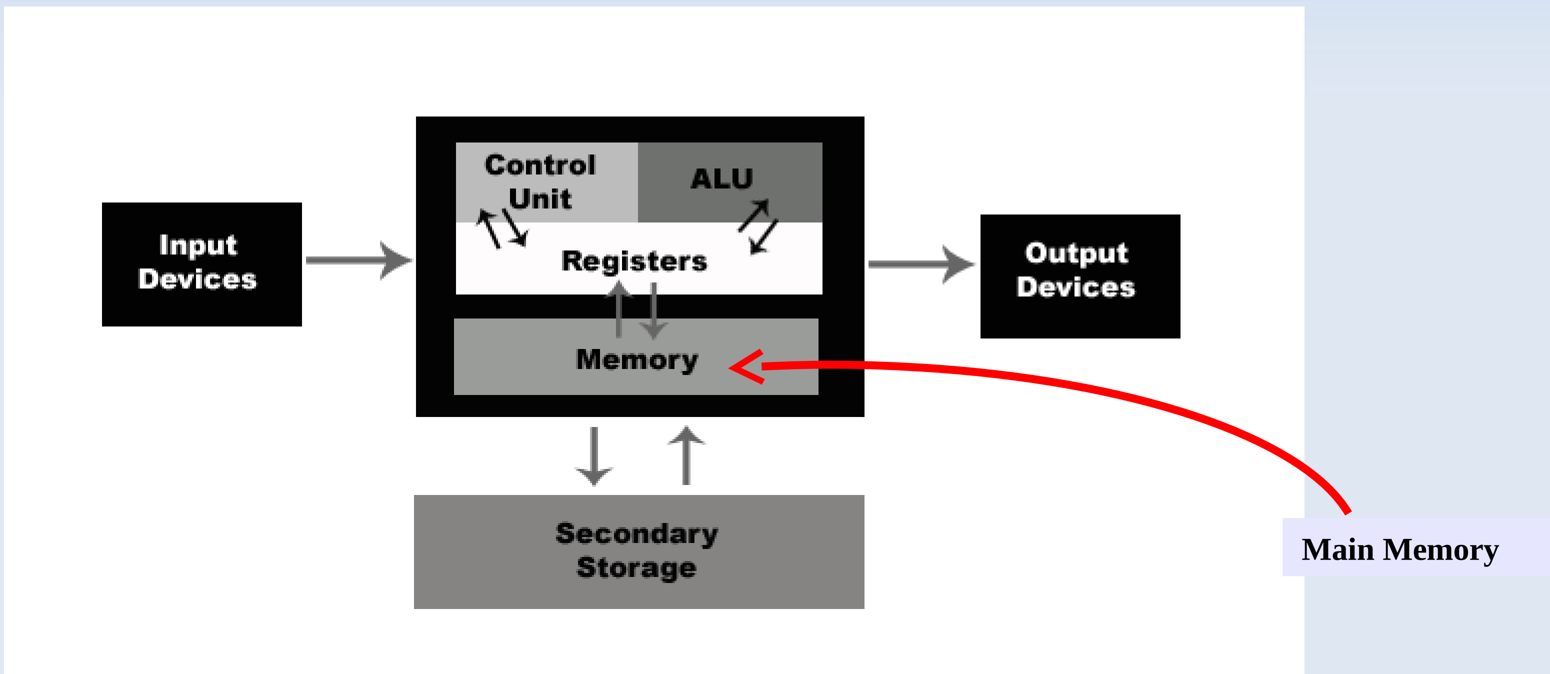
  (a) defining a framework for program execution and

  (b) defining a set of

  - a default interface to the user when no application program is running and
- OS: kernel together with
  - set of system programs which use facilities provided by the kernel to perform higher-level functionalities (house-keeping) tasks
- KERNEL: is the core (irreducible core from which entire OS func. Gets delivered)

Computer System Organization
(Abstract view)

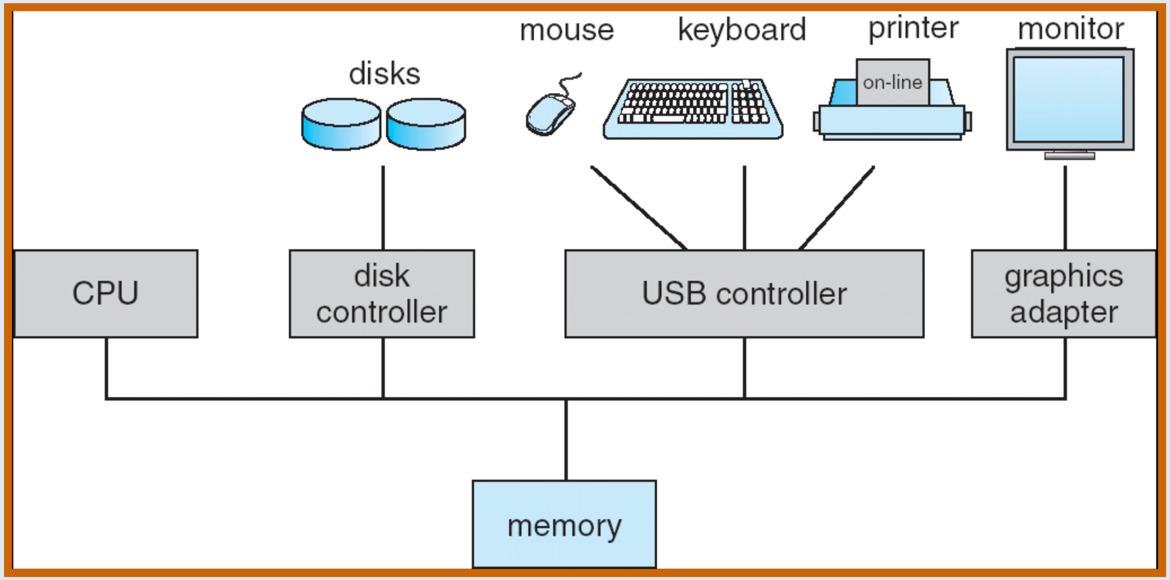
Typically, a common bus for all!



Requires: Dev. Controller HW for delegation → Dev. Driver (SW) is required

Computer System Organization (refined)

- One or more CPUs, device controllers connected through common bus providing access to shared memory
- Concurrent execution (CPUs and devices) competing for memory access (cycles)



Storage Structure

Primary storage

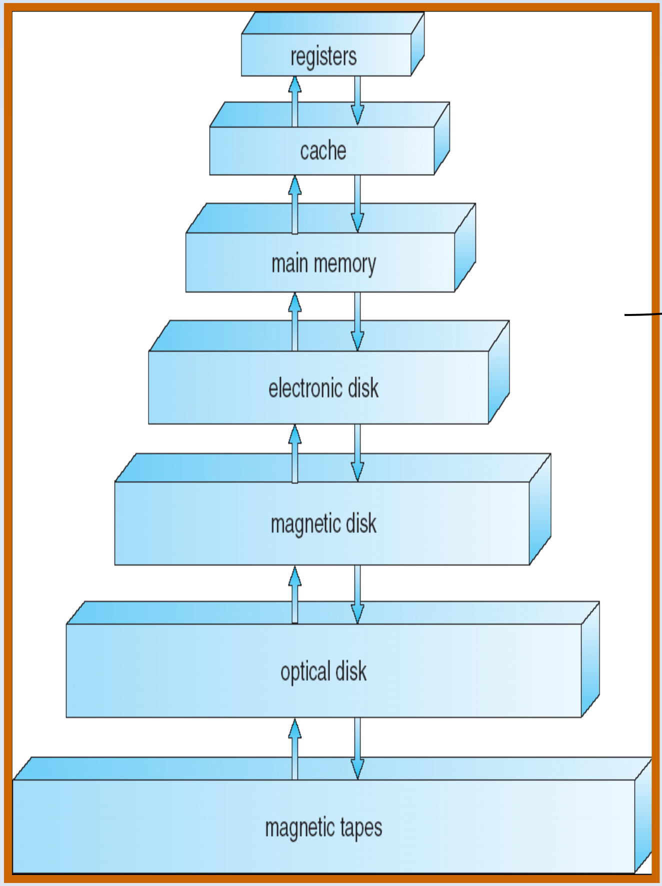
- Main memory, only mem. directly accessible by CPU
  - Program must be in main memory in order to be executed
  - Main memory usually not large enough to hold all programs and data (paging)
  - Main memory is *volatile*

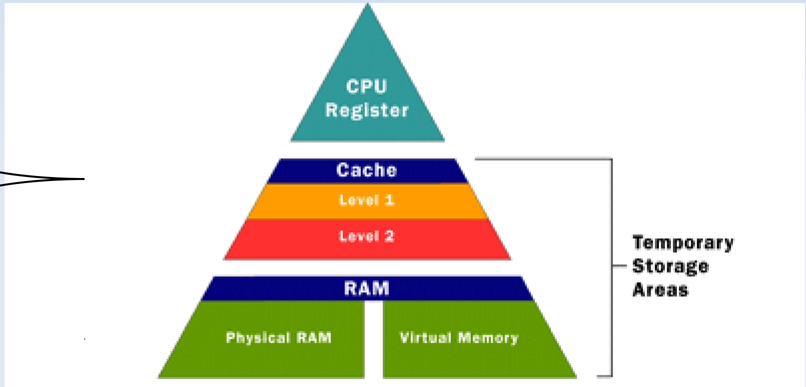
Secondary storage:

- large quantities of data, permanently

In general, we have a hierarchy of storage devices varying by speed, cost, size and volatility

Storage-Device Hierarchy





Computer-System Architecture

- Single-processor system
  - From PDAs to mainframes
  - Almost all have special-purpose processors for graphics, I/O
    - Not considered multiprocessor
- Multi-processor systems
  - Increase throughput
  - Economy of scale
  - Increased reliability
  - Asymmetric multiprocessing
    - Each processor assigned a specific task (master-slave)
  - Symmetric multiprocessing (SMP) most common

    All processors perform tasks within the OS
- Clusters, distributed systems
- Multiple cores, blade servers

User View of a Computer: Kinds of Operating Systems

- PCs: This is my box, only I am using it (user after login)
  - i.e. desktop system with one user monopolizing its resources
  - OS maximizes the work (or PLAY) user is performing
  - OS designed mostly for ease of use, not for resource utilization
  - Handheld systems – usability + low hardware demands
- MF/MC: This is The Big Holy Computer, I am blessed to get some CPU time
  - i.e. mainframe or minicomputer (?)
  - OS is designed to maximize resource use (CPU, memory, I/O)
- Networked clusters: Communism in practice - Let’s share our computers
  - i.e. workstations connected to networks of servers
  - Dedicated and shared resources
  - OS compromises between individual usability and resource utilization
- Embedded systems: What? There is a computer inside?!
  - Safety critical: RTOS real-time guarantee

Operating System Services

Services provided to user programs:

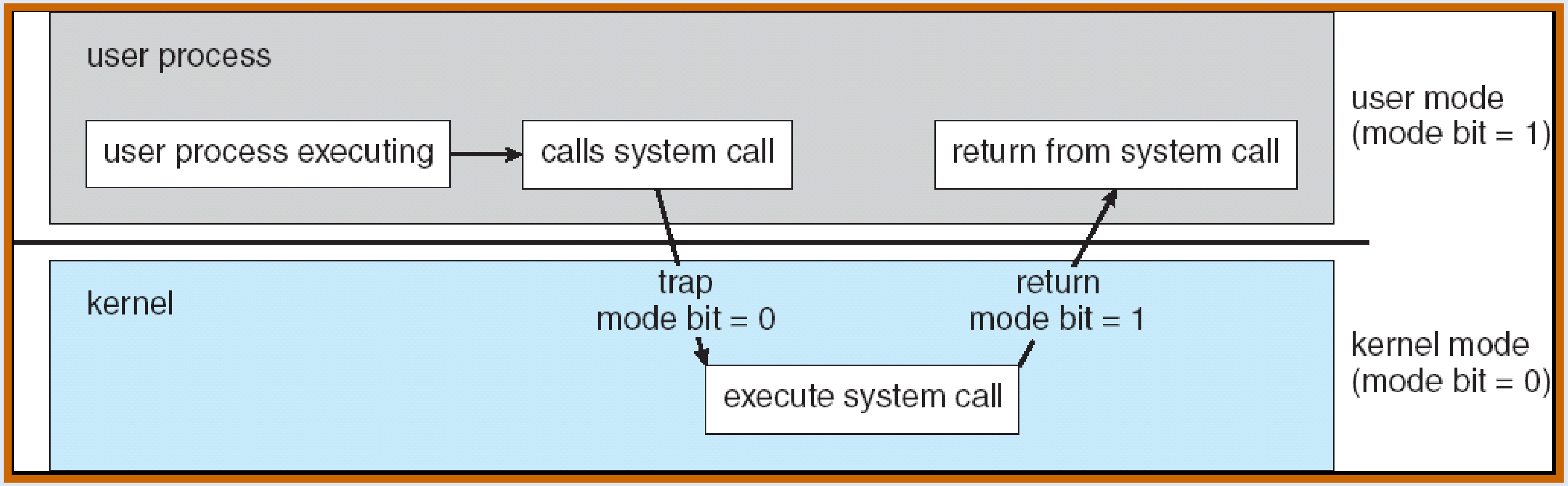
- I/O operations
  - User program cannot directly access I/O hardware, OS does the low level part for them
- Communications
  - Both inter-process on the same computer, and between computers over a network
  - via shared memory or through message passing
- Error detection
  - Errors do occur: in the CPU and memory hardware, in I/O devices, in user programs
  - OS should handle them appropriately to ensure correct and consistent computing
  - Low level debugging tools really help

Operating-System Operations

- OS is interrupt driven
- Interrupts raised by hardware and software
  - Mouse click, division by zero, request for operating system service
  - Timer interrupt (i.e. process in an infinite loop), memory access violation (processes trying to modify each other or the operating system)
- Some operations should be performed only by a trusted party
  - Accessing hardware, memory-management registers
  - A rogue user program could damage other programs, steal the system for itself, …
  - Solution: dual-mode operation

Transition from User to Kernel Mode

- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user



Processes

- Introduced to obtain a systematic way of monitoring and controlling program execution
- A process is an executable program with:
    - associated data (variables, buffers…)
    - execution context: i.e. all the information that (the CPU needs to execute the process + content of the processor registers)
- the OS manages:
    - Creation and deletion of user and system processes
    - Suspending and resuming processes
    - Process synchronization
    - Process communication
    - Deadlocks (priorities)

Memory Management

- OS keeps track of which part of memoryis currently being used
- Deciding which process to move in or out of the memory
- Allocating and deallocating memory

Storage Management

- Creating and deleting files/directories
- File/directory organization
- Mapping files into secondary storage
- Making back-ups

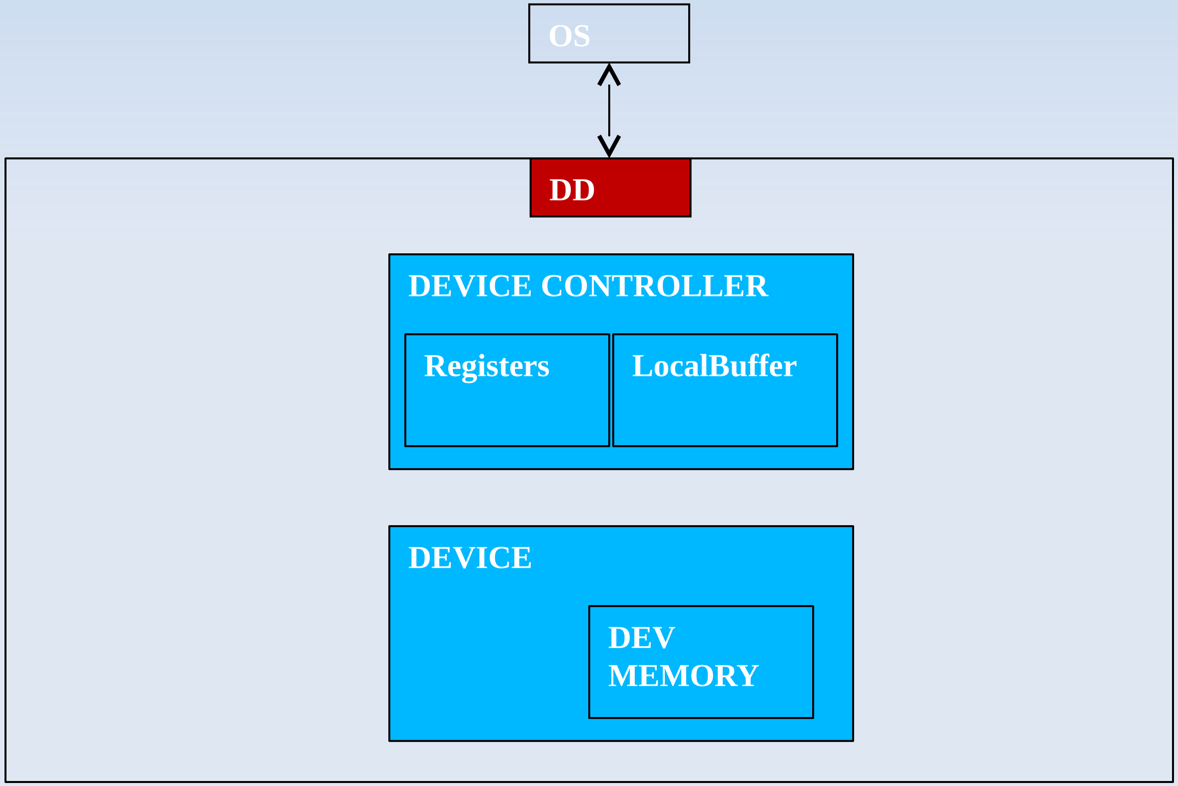
I/O Structure

- Controller has registers (+LocalBuffer) for accepting commands and transferring data (i.e. data-in, data-out, control, status)

- Device driver for that device talks to the controller
  - The driver is the one who knows details of controller
  - Provides uniform interface to device from the kernel/OS

- I/O operation
  - Device driver loads controller registers appropriately
  - Controller examines registers, executes I/O

I/O Illustrated



I/O Illustrated

- I/O Operation is:

```
{
    DD: Load DC_Register /* d_i, d_o, status, control*/
    DC: Check Registers
        {
          if  (d_i) transfer DM -> LB;
          if  (d_o) transfer LB  -> DM;
              …
              …
        }
    DC_INTERRUPT;
    return(); /*or return(MEM_ADD)*/
}
```

I/O Structure

- How does the driver know when the I/O completes?
  - Check the *status* register
    - Called direct I/O
    - Low overhead if I/O is fast
    - If I/O is slow, lots of *busy waiting*
- Any idea how to deal with slow I/O?
  - Do something else and let the controller signal device driver (raising and interrupt) that I/O has completed
    - Called interrupt-driver I/O
    - More overhead, but gets rid of busy waiting