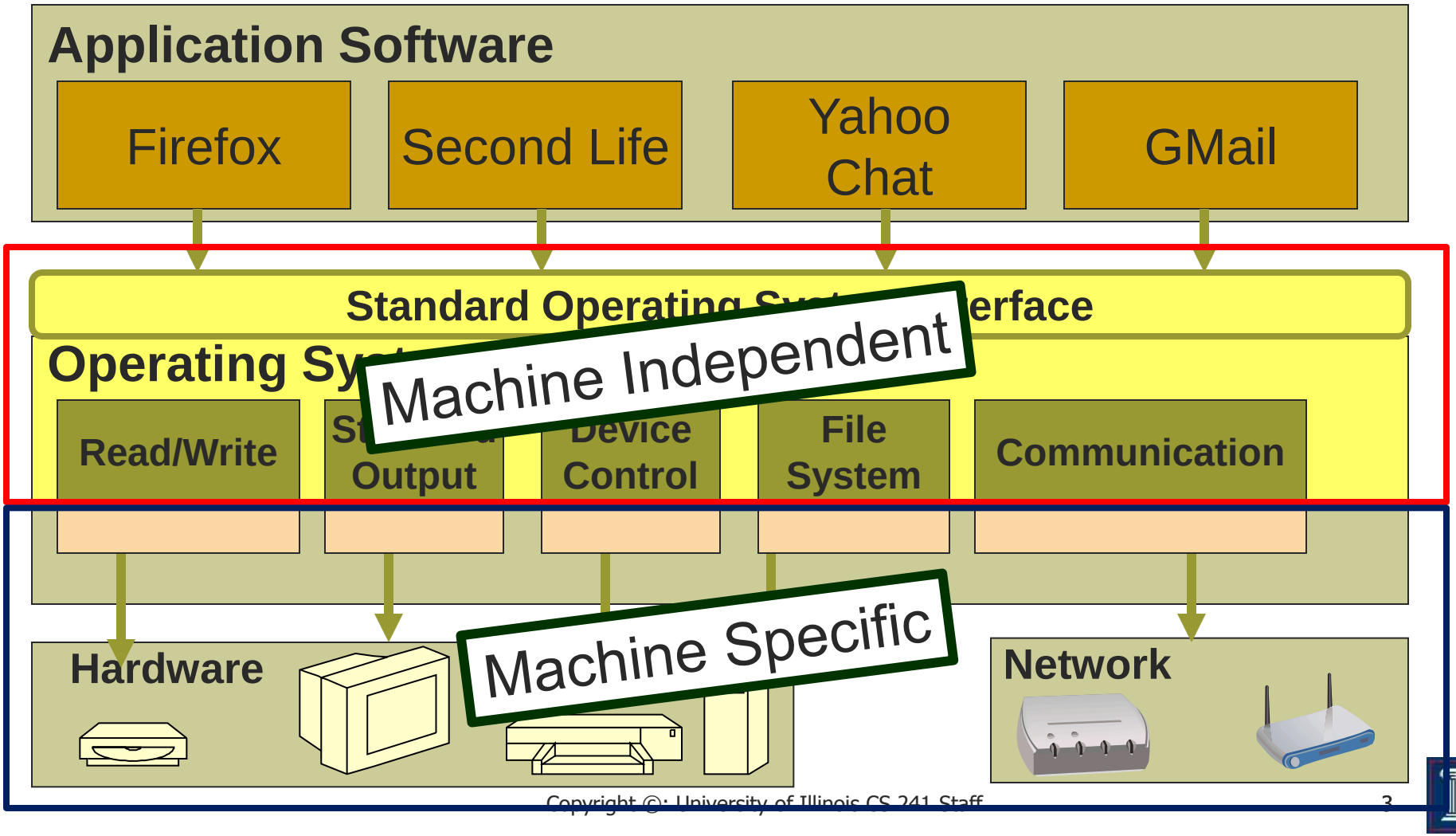
Operating Systems Orientation

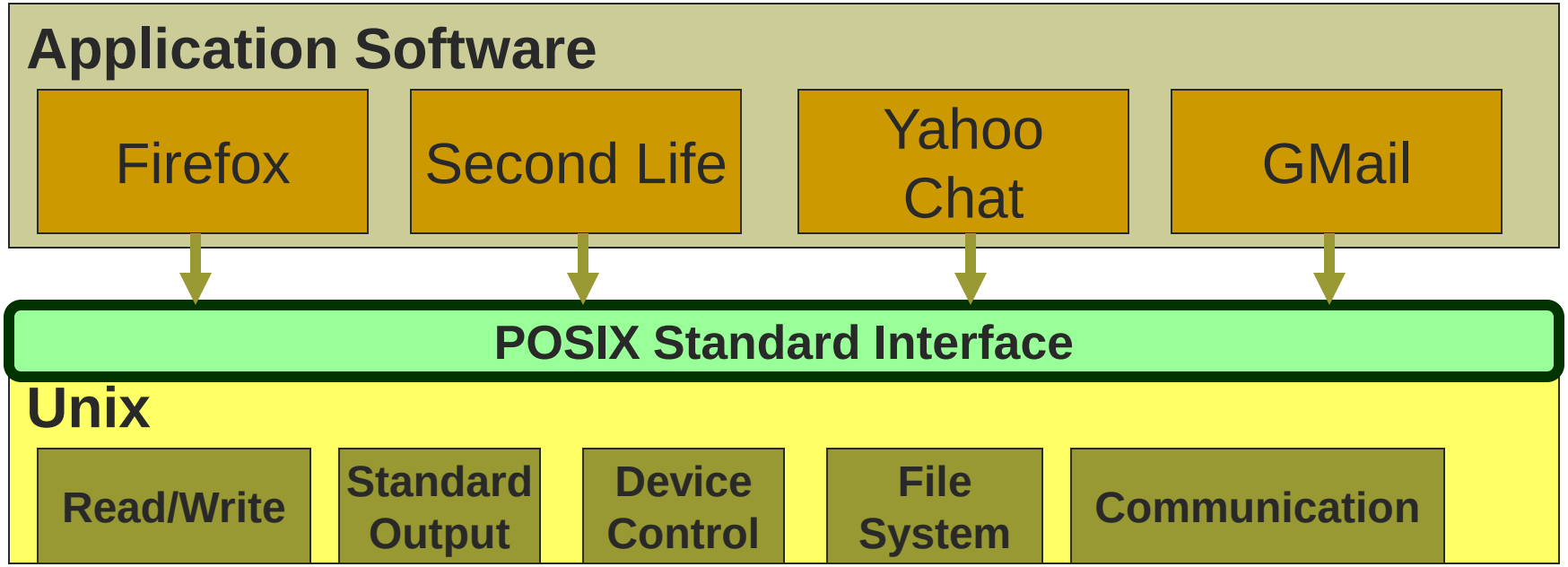
Objectives

- Explain the main purpose of operating systems and describe milestones of OS evolution
- Explain fundamental machine concepts
  - Instruction processing
  - Memory hierarchy
  - Interrupts
  - I/O
- Explain fundamental OS concepts
  - System calls
  - Processes
  - Synchronization
  - Files
- Explain the POSIX standard (UNIX specification)

OS Structure



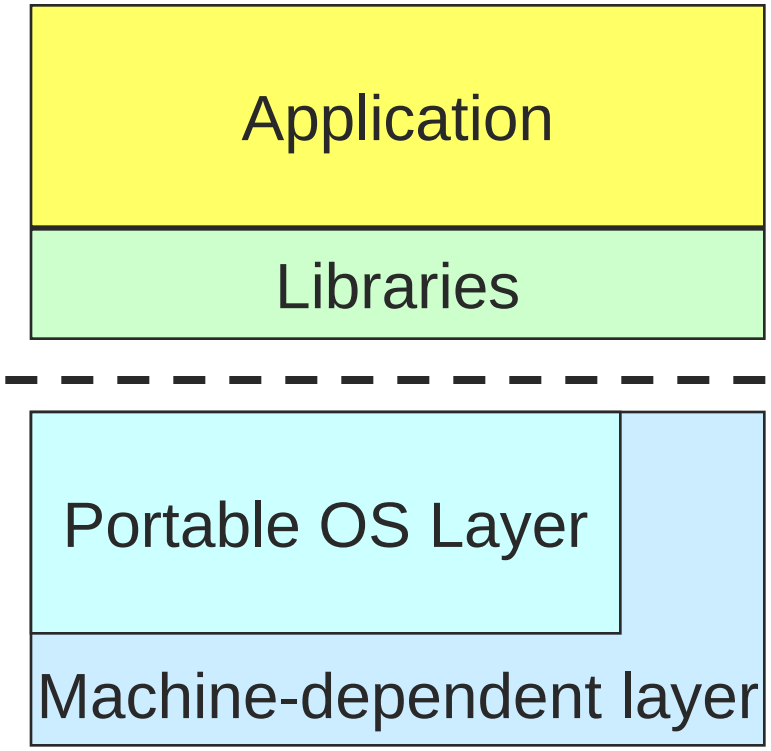
POSIX
The UNIX Interface Standard



What is an Operating System?

- It is an *extended machine*
  - Hides the messy details that must be performed
  - Presents user with a virtualized and simplified abstraction of the machine, easier to use
- It is a *resource manager*
  - Each program gets time with the resource
  - Each program gets space on the resource

A Peek into Unix

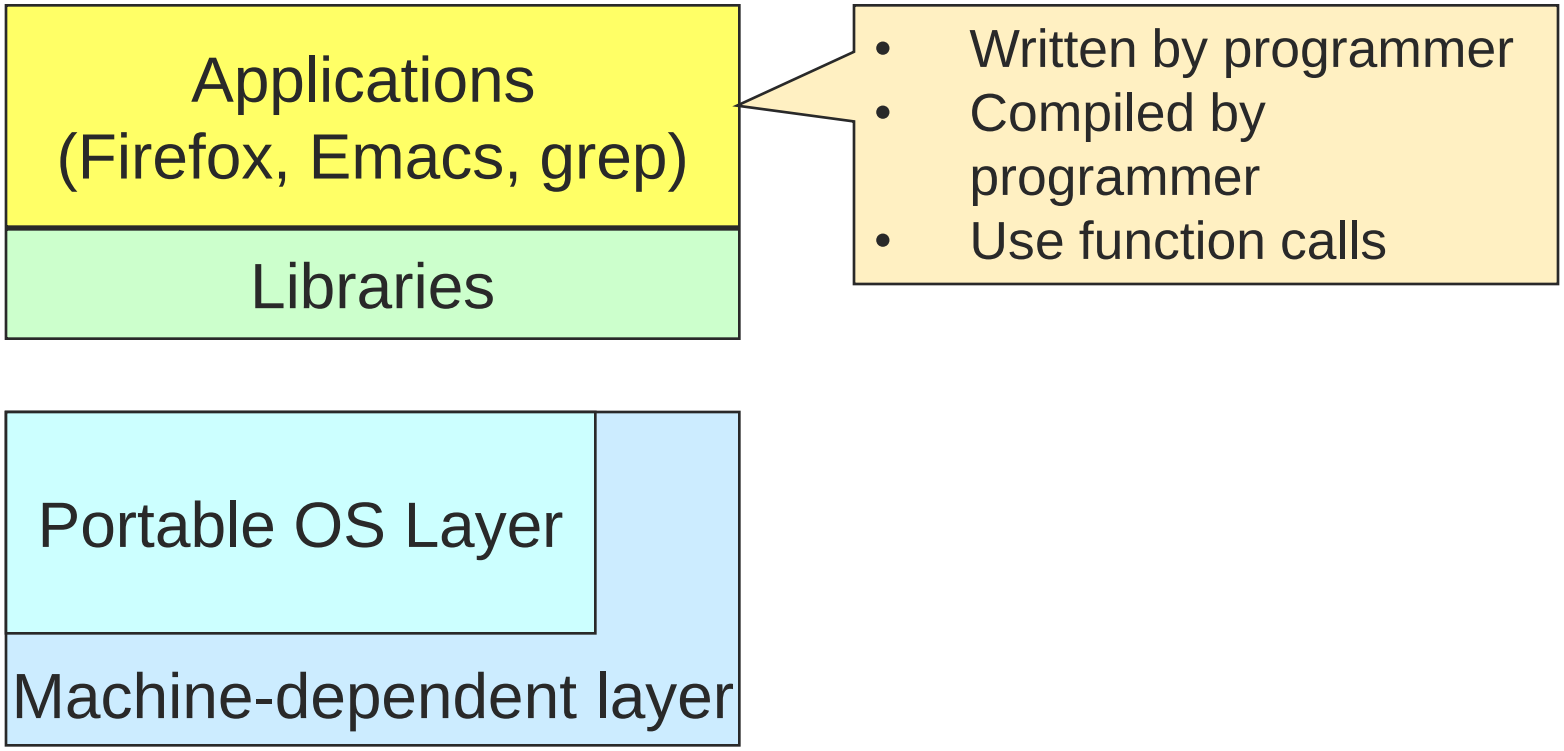


User space/level

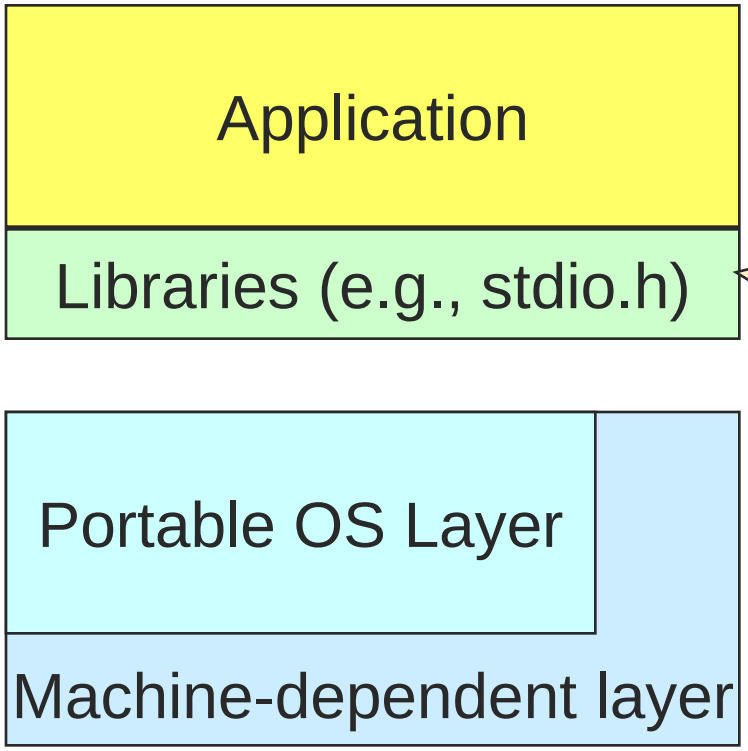
Kernel space/level

- **User/kernel modes are supported by hardware**
- **Some systems do not have clear user-kernel boundary**

Application

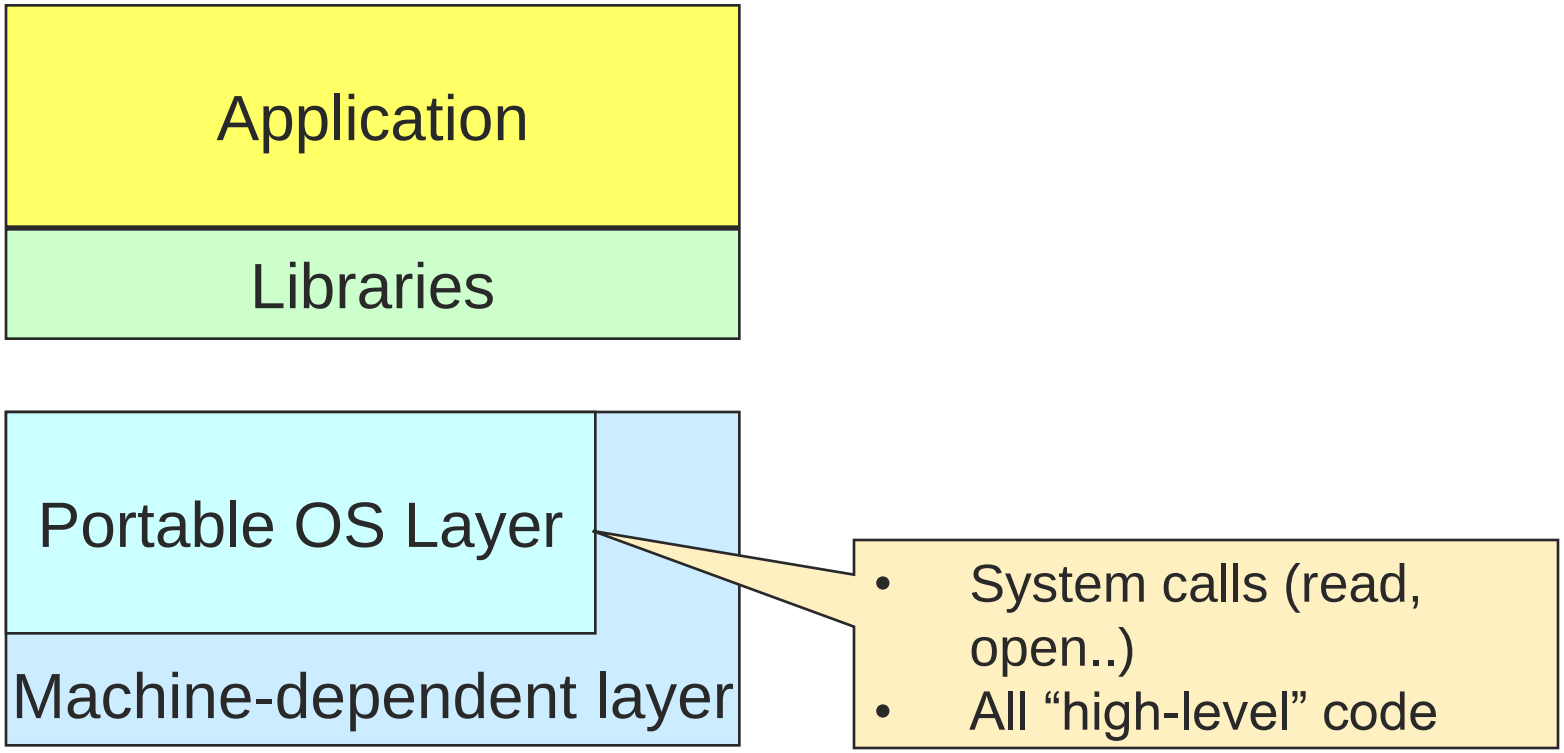


Unix: Libraries

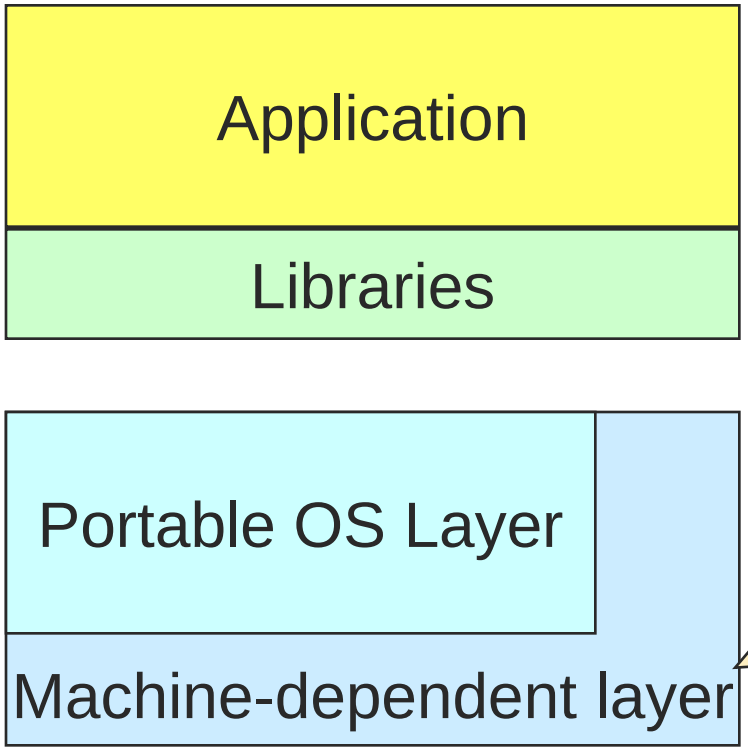


- Provided pre-compiled
- Defined in headers
- Input to linker (compiler)
- Invoked like functions
- May be “resolved” when program is loaded

Typical Unix OS Structure



Typical Unix OS Structure



- Bootstrap
- System initialization
- Interrupt and exception
- I/O device driver
- Memory management
- Kernel/user mode switching
- Processor management

History of Computer Generations

- Pre-computing generation 1792 - 1871
  - Charles Babbage's "Analytical Engine"
  - Purely mechanical
  - Designed, but never actually built
    - Required high-precision gears/cogs that didn't exist yet
  - A man before his time
    - When this works, we'll need software!
    - First programming language
    - World's first programmer: Ada Lovelace

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
  - Vacuum tubes, relays, plug boards
  - Seconds per operation!
    - Focus on numerical calculations
  - No programming language
    - Everything done using pure machine language or wiring electrical circuits!
  - No operating system
    - Sign up for your time slot!
  - Progress: Punch cards!

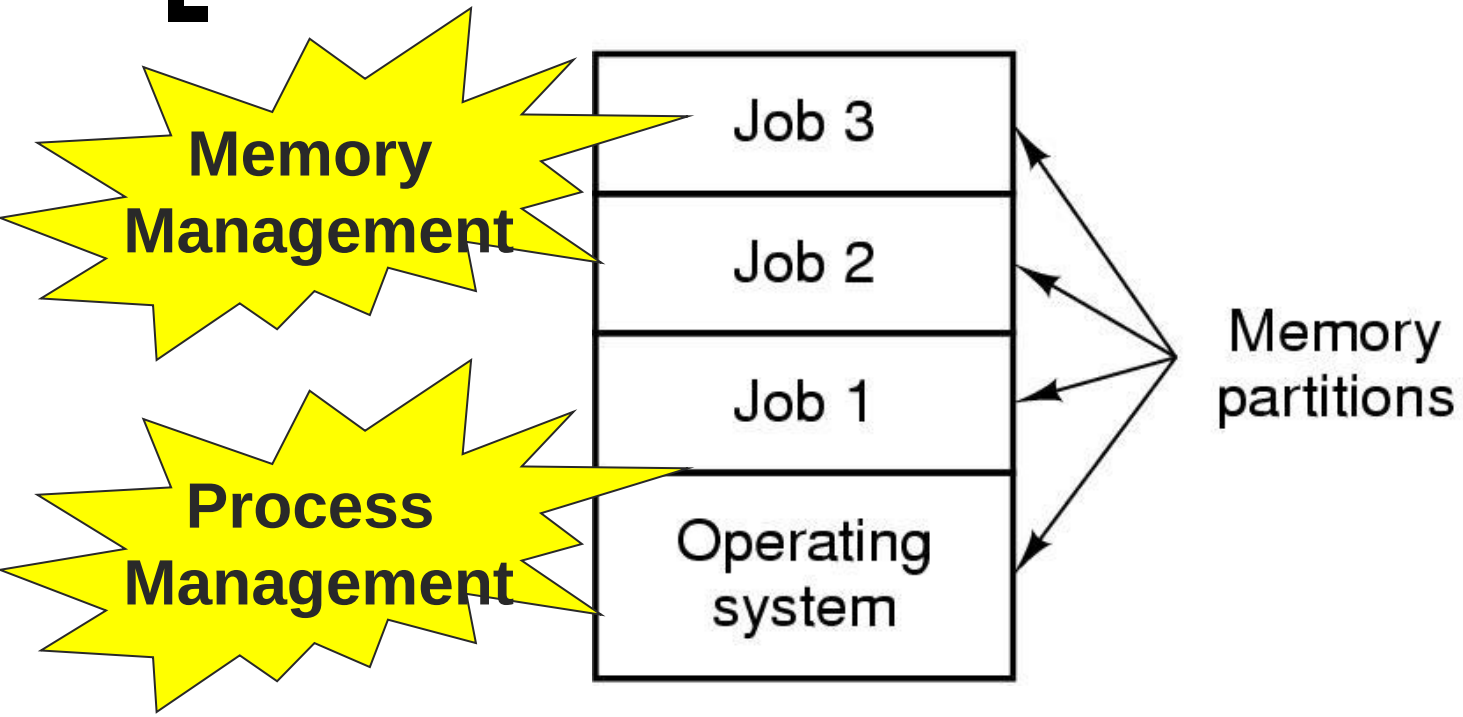
History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 - 1965
  - Transistors, mainframes
  - Large human component

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
  - Integrated circuits and multiprogramming
  - IBM's New model: all software and OS must work on all platforms
    - A beast!
  - Progress: Multiprogramming
    - Keep the CPU busy

History of Operating Systems



- Multiprogramming/timesharing system
  - Three jobs in memory – 3rd generation

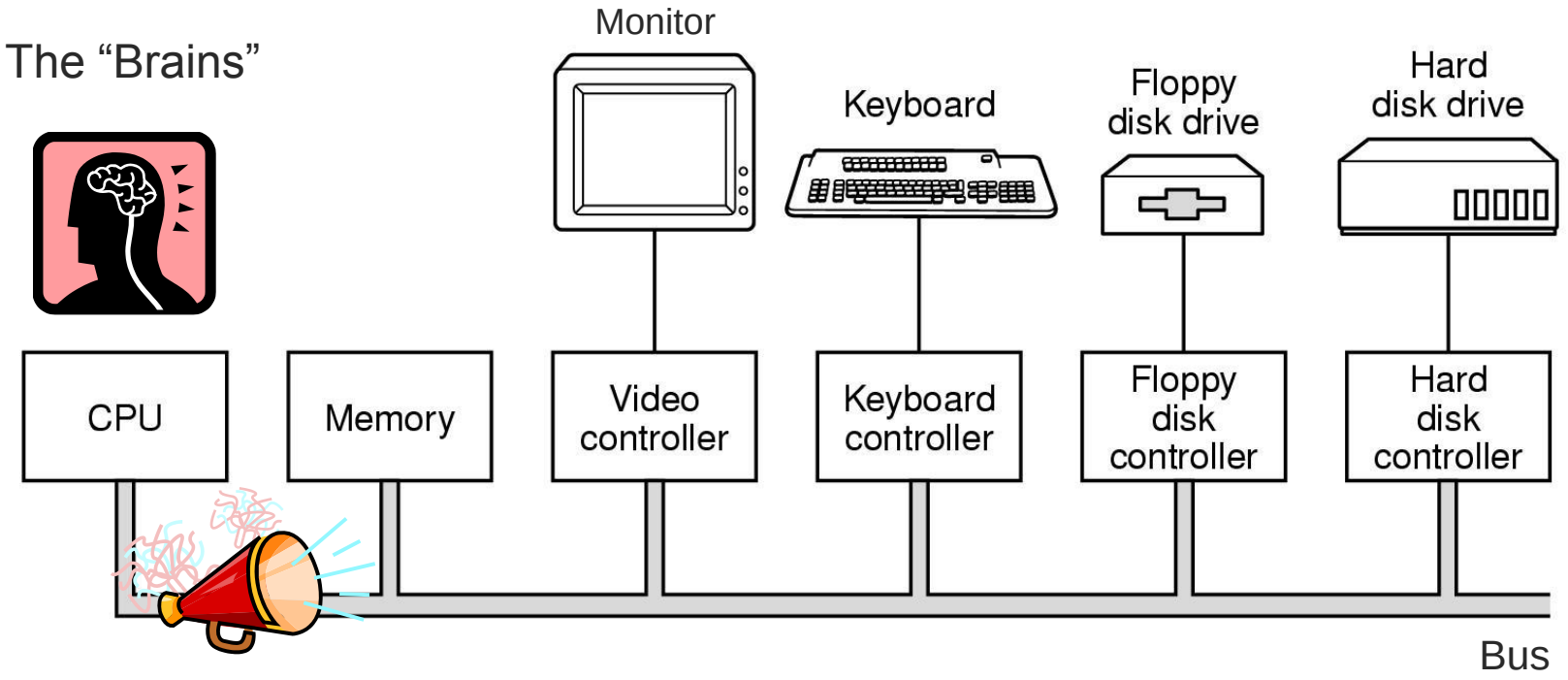
History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
  - Integrated circuits and multiprogramming
  - IBM's New model: all software and OS must work on all platforms
  - Progress: Multiprogramming and timesharing
  - Progress: Spooling
    - Always have something ready to run
  - MULTICS + minicomputers == UNIX!

History of Computer Generations

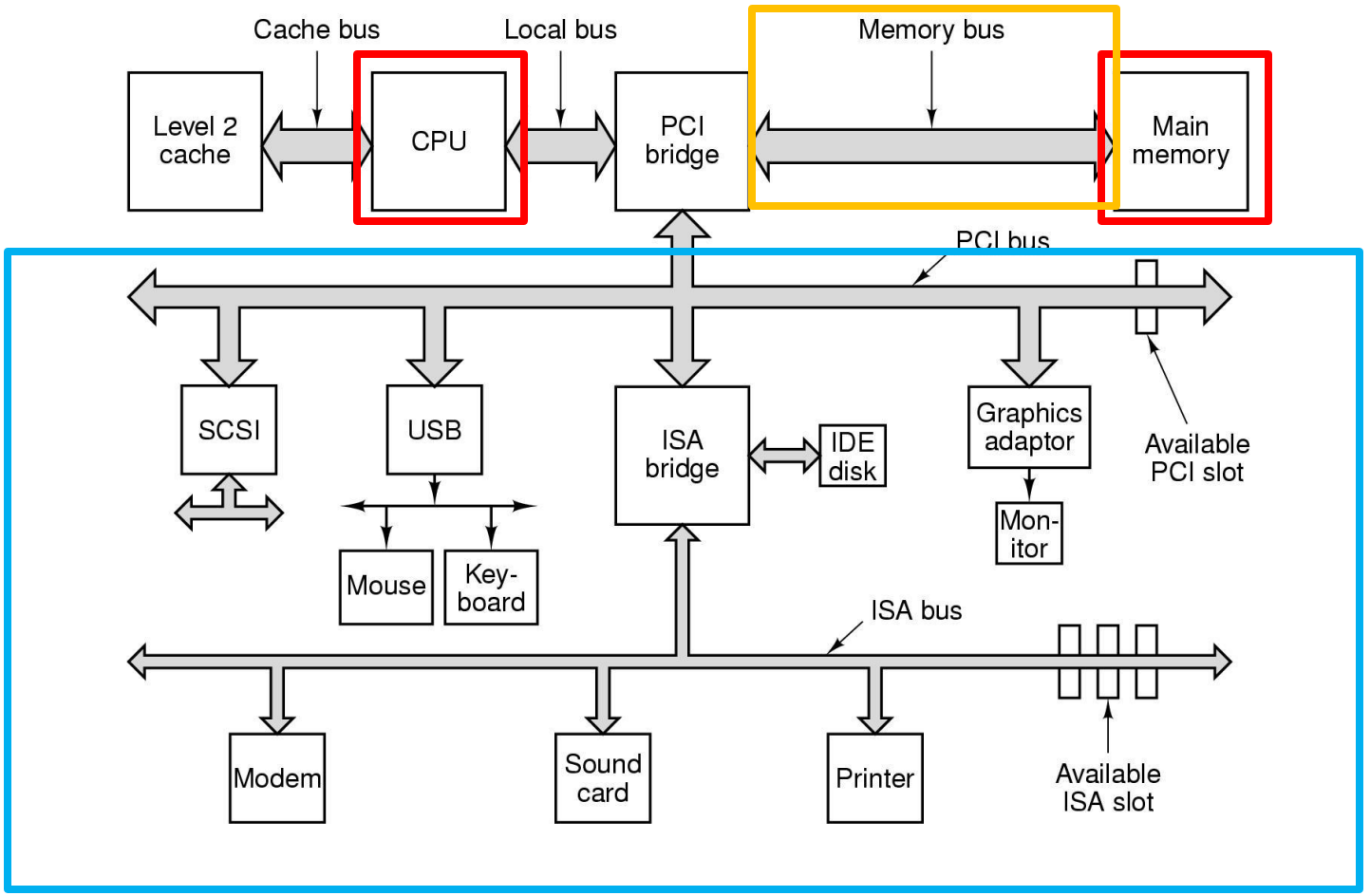
- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
- Fourth generation 1980 – present
  - Personal computers
  - Multi-processors
  - Phones
  - …

Computer Hardware Review

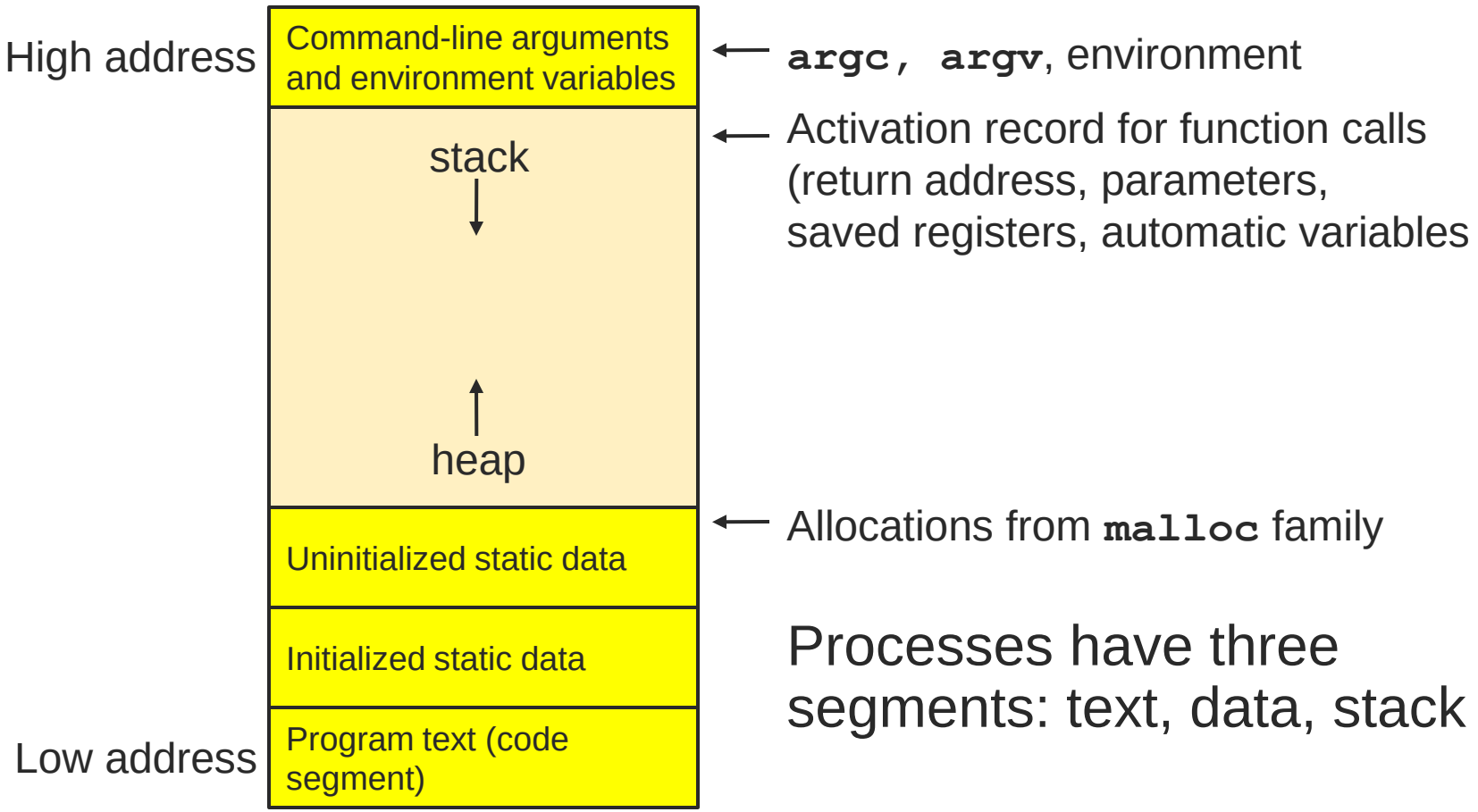


- Components of a simple personal computer

Early Pentium system



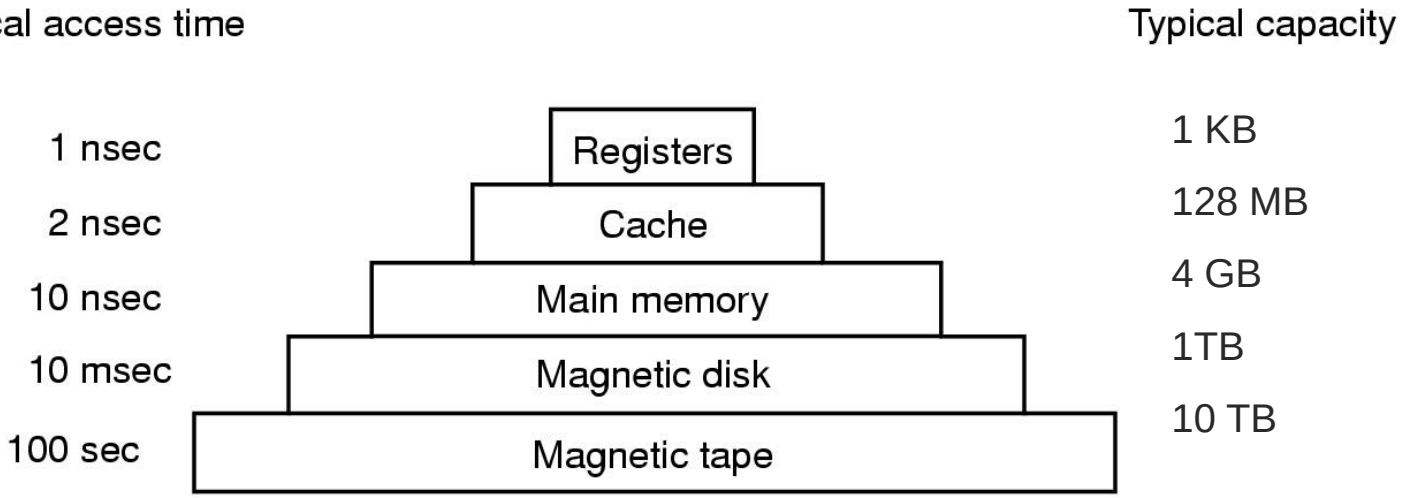
Sample Layout for program image in main memory



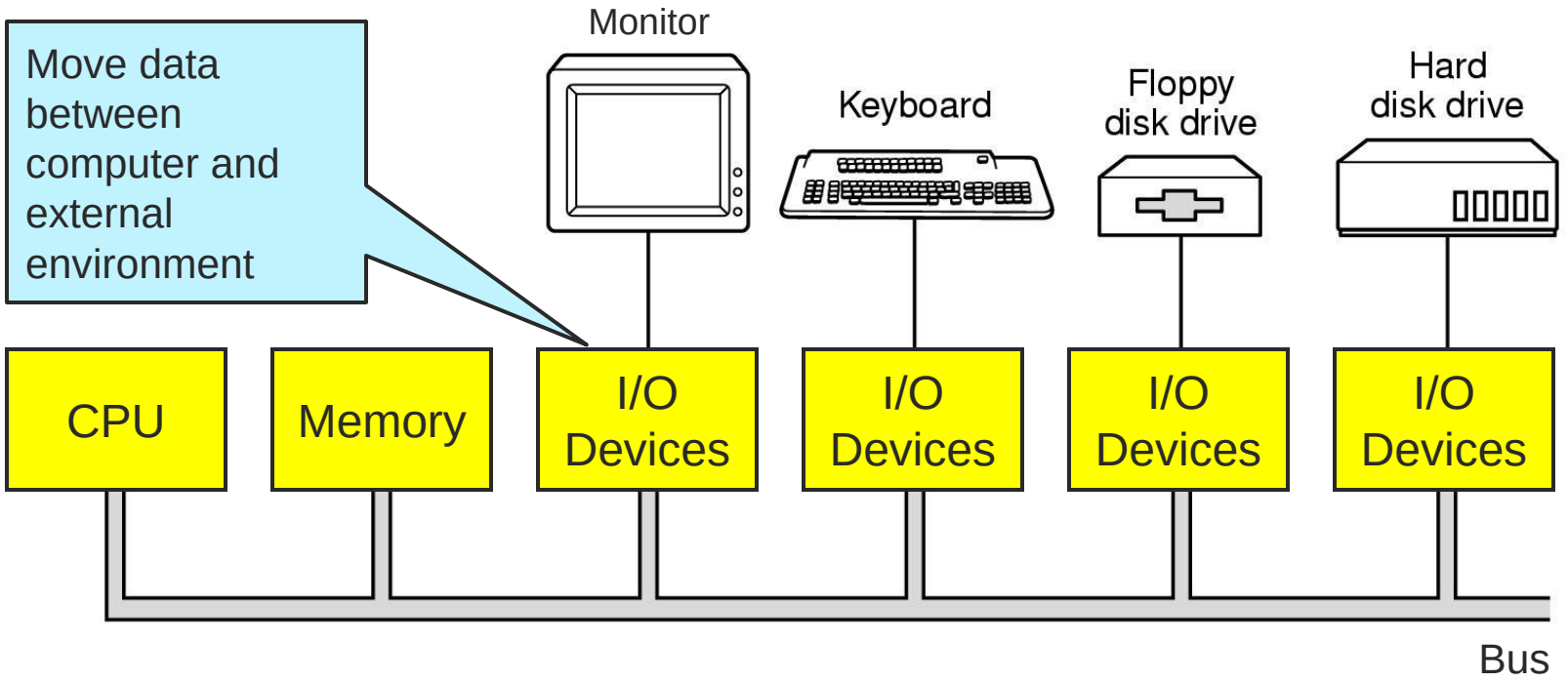
Memory Hierarchy

- Leverage locality of reference

1. Decreasing cost per bit
2. Increasing capacity
3. Increasing access time
4. Decreasing frequency of access



Computer Hardware Review

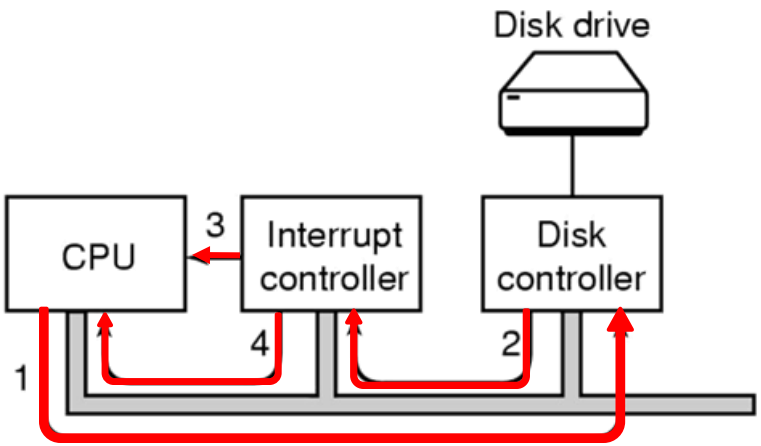


- Components of a simple personal computer

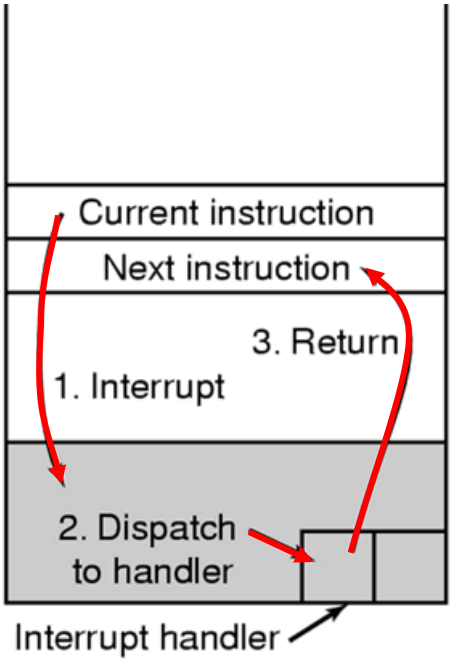
I/O Device Access

- System Calls
  - Application makes a system call
  - Kernel translates to specific driver
  - Driver starts I/O
  - Polls device for completion
- Interrupts
  - Application starts device
  - Asks for an interrupt upon completion
  - OS blocks application
  - Device controller generates interrupt

I/O Interrupt Mechanism



(a)



(b)

1. Application writes into device registers, Controller starts device
2. When done, device controller signals interrupt controller
3. Interrupt controller asserts pin on CPU
4. Interrupt controller puts I/O device number on bus to CPU

Operating System Concepts

- Process
  - An instance of a computer program that is being executed
  - Only one process can use a CPU at a time

Operating System Concepts

- How to go back and forth between processes?
  - How would you switch CPU execution from one process to another?
- Solution: Context Switching
  - Store/restore state on CPU, so execution can be resumed from same point later in time
  - Triggers: multitasking, interrupt handling, user/kernel mode switching
  - Involves: Saving/loading registers and other state into a “process control block” (PCB)
  - PCBs stored in kernel memory

Operating System Concepts

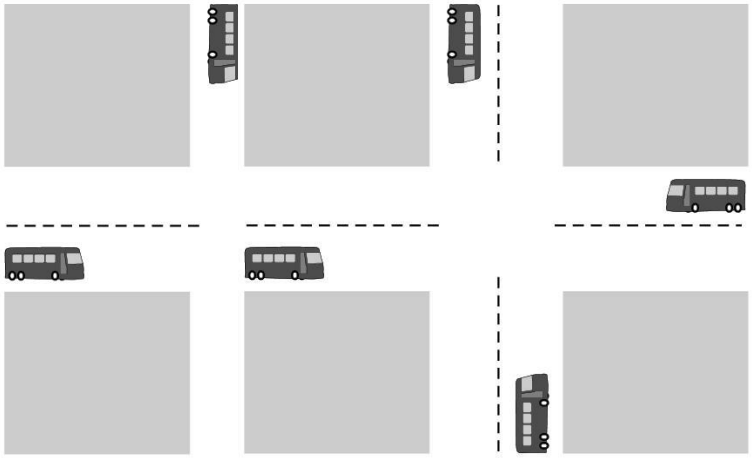
- Context Switching
  - What are the costs involved?

Item	Time	Scaled Time in Human Terms (2 billion times slower)
Processor cycle	0.5 ns (2 GHz)	1 s
Cache access	1 ns (1 GHz)	2 s
Memory access	15 ns	30 s
Context switch	5,000 ns (5 micros)	167 m
Disk access	7,000,000 ns (7 ms)	162 days
System quanta	100,000,000 (100 ms)	6.3 years

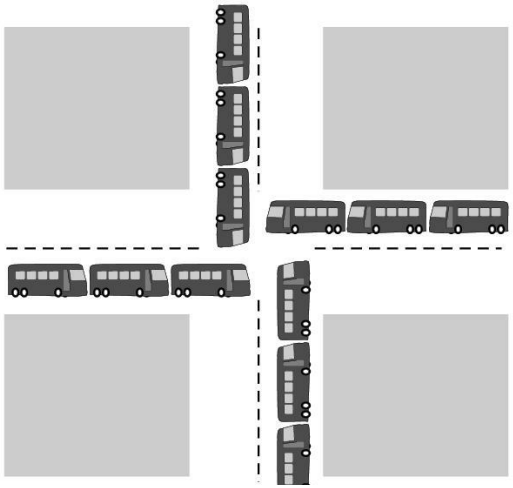
Operating System Concepts

- Shared resources
  - Now I have B KB of memory, but need 2B KB
  - Now I have N processes trying to access the disk
  - How would you control access to resources?
  - What are the challenges?

Operating System Concepts



(a) A potential deadlock

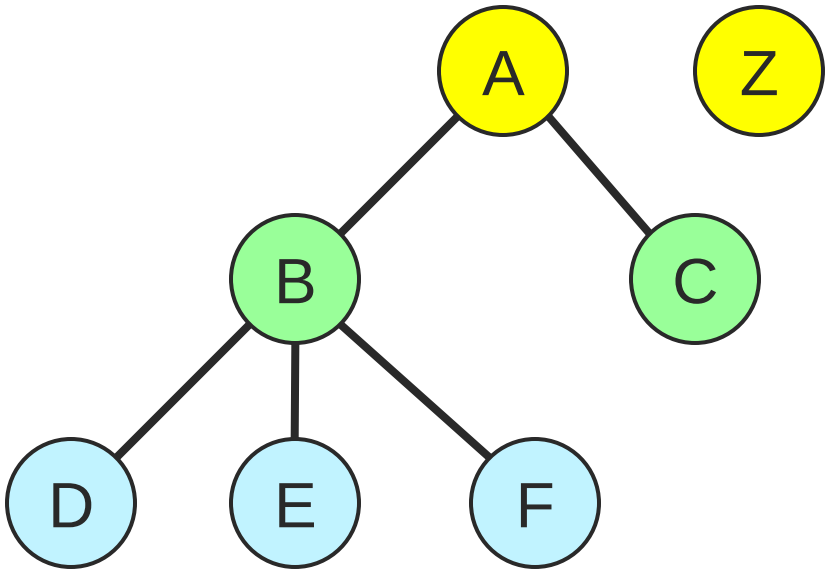


(b) An actual deadlock

- One challenge: Deadlock
  - Set of actions waiting for each other to finish
- Example:
  - Process A has lock on file 1, wants to acquire lock on file 2
  - Process B has a lock on file 2, wants to acquire lock on file 1

Operating System Concepts

- Process
  - An executable instance of a program
  - Only one process can use a (single-core) CPU at a time
- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F



Two simulations accesses to memory…

x++;

x++;

Two simulations accesses to memory…

x++;

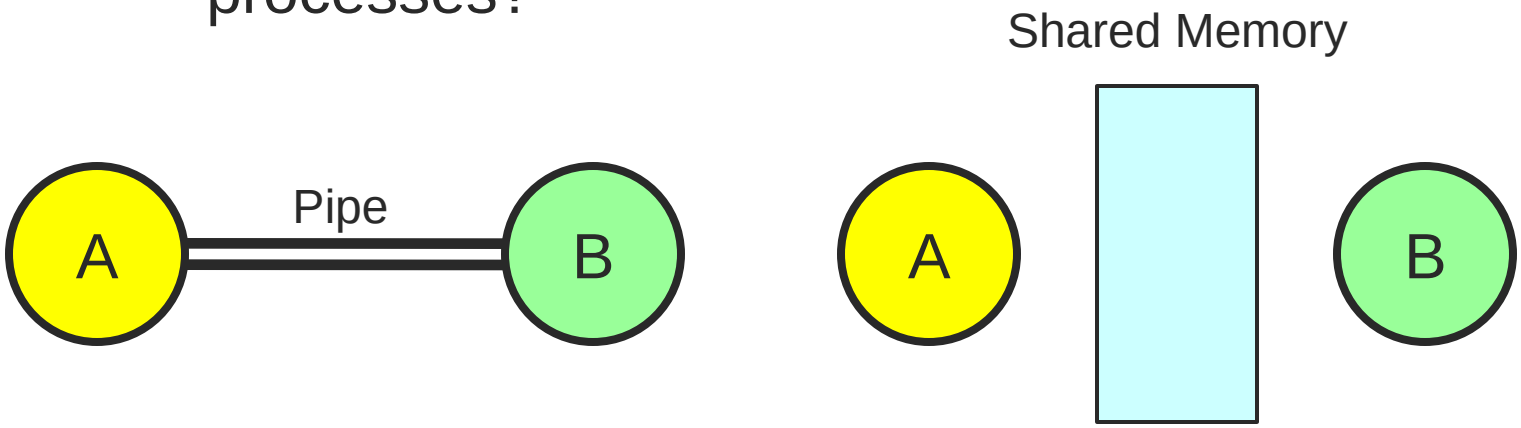
LOAD x
INCR x
STORE x

x++;

LOAD x
INCR x
STORE x

Operating System Concepts

- Inter-process Communication
  - Now process A needs to exchange information with process B
  - How would you enable communication between processes?



Summary

- Resource Manager
- Hardware independence
- Virtual Machine Interface
- POSIX
- Concurrency & Deadlock

Announcements

- Discussion Sections Today
  - MiniMP: Binary Tree library in C
- HW1 Due Tomorrow at 11:59pm
- MP1 Due Sunday at 11:59pm
  - Closely related to the MiniMP
- More Lab Hours Posted
  - Wednesday, Friday, Saturday
- Piazza