CS 5323 – OS II

Lecture 1 – Course Overview and OS Concepts Review

The Teaching Team

- Instructor:
  - Dr. Sathyanarayanan Aakur
  - Email: saakurn@okstate.edu
  - Office hours (Virtual): Monday/Wednesday 1:00 PM to 2:30 PM, or by appointment.
- Teaching Assistant:
  - TBA
  - Email: TBA
  - Office hours (Virtual): TBD

Course Administration

- Textbook:
  - No official book.
    - *Operating Systems Concepts*, Ninth Edition, by Silberschatz, Galvin, and Gagne published by John Wiley and Sons.
    - *Operating Systems: Three Easy Pieces*, by Remzi H.Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Available online: http://pages.cs.wisc.edu/~remzi/OSTEP/
    - Supplementary material will be posted periodically

Grading Information

Exams (25%)

- One Midterm Exam

Quizzes (25%)

- One quiz at the end of every week on Canvas.
- Open book and open notes

Assignments (25%)

- Five (5) assignments – One every two weeks

Final Project (25%)

- One Final Exam

- Look out for bonus questions on projects and exams.
- Talk to us about any concerns about projects/exams
- This grade distribution is tentative and may be changed
- Note: Canvas weights the assignments automatically but it only considers currently graded assignments. The score on Canvas might not be an accurate reflection of your final score.

More details

Assignments

- Do them to truly understand the material, not to get the grade
- All homework write-ups and quizzes ***must*** be your own work, written up individually and independently
  - Peer discussions are encouraged. Please don't share solutions!
- No late submissions will be accepted

Quizzes and Exams

- Will cover material from each week lectures, reading assignments and supplementary material.
- Only one attempt per quiz

Grading Policies

Percentage	Grade	GPA Quality Points
90 - 100	A	4.0
80 - 89	B	3.0
70 - 79	C	2.0
60 - 69	D	1.0

Tentative Schedule

Week 1	Course introduction and OS Review
Week 2	OS Structure, Process, Inter-process communication
Week 3	Threads, Critical Section
Week 4	Process Synchronization, Semaphore, Mutexes
Week 5	Monitors, CPU Scheduling
Week 6	Classic Problems, Deadlocks
Week 7	Deadlock, Midterm
Week 8	Memory Management, Paging, Segmentation
Week 9	Virtual Memory Management, Files

Week 10	File Systems
Week 11	Disk, Mass Storage
Week 12	I/O Systems
Week 13	Protection, OS Systems and Networks
Week 14	Distributed Systems
Week 15	Review, Make-up, Pre-Finals Week
Week 16	Finals Week

Software Requirements

- We will use C as the major language in this course
  - Intermediate level is a pre-requisite
- CSX server account
  - https://computerscience.okstate.edu/cs-it-services/software/logging-on
- EVERYONE MUST CREATE AN ACCOUNT!
- All assignments must execute on CSX and will be graded based on the execution on the CSX server. It does not matter if it runs on your computer. It must execute on CSX for assignments!
- **You must submit your assignments on Canvas by the due date. No extensions will be permitted.**

Expectations from You

- **Work Hard!**
  - This is a heavy course that covers OS fundamentals.
  - There will be a quiz every week, posted on Canvas on Monday or Wednesday depending on course progress.
  - There will be an assignment posted every two weeks.
  - ***Late work will not be accepted***.
- Attend all lectures (I try to keep it engaging and interactive!)
- Ask questions, participate in discussions
- Exams are comprehensive. Study all materials posted.
- Complete all assignments
  - Try to understand the concepts, usage and implementation

Academic Dishonesty

- **Absolutely no form of cheating will be tolerated**
- See syllabus, OSU Policy, and CS Academic Integrity Policy
- Cheating → Failing grade (no exceptions)

Tips to succeed in the course

- Read any supplementary material posted in addition to the lectures
- Do the assignments and quizzes on your own to understand the concepts
- Don’t be discouraged by errors!
  - Learning is a repetitive process!
  - Mistakes are your keys to success! ☺
- Ask thorough questions to understand the concepts
- Balance your time!
  - I understand that you have several things to do. Allot at least 2-3 hours per week.
  - Remember. Hard work always pays off!

Overview

- What ***is*** an operating system, anyway?
- Operating systems history
- The zoo of modern operating systems
- Review of computer hardware
- Operating system concepts
- Operating system structure
  - User interface to the operating system
  - Anatomy of a system call

What *is* an operating system?

- A program that runs on the “raw” hardware and supports
  - Resource Abstraction
  - Resource Sharing
- Abstracts and standardizes the interface to the user across different types of hardware
  - Virtual machine hides the messy details which must be performed
- Manages the hardware resources
  - Each program gets time with the resource
  - Each program gets space on the resource
- May have potentially conflicting goals:
  - Use hardware efficiently
  - Give maximum performance to each user

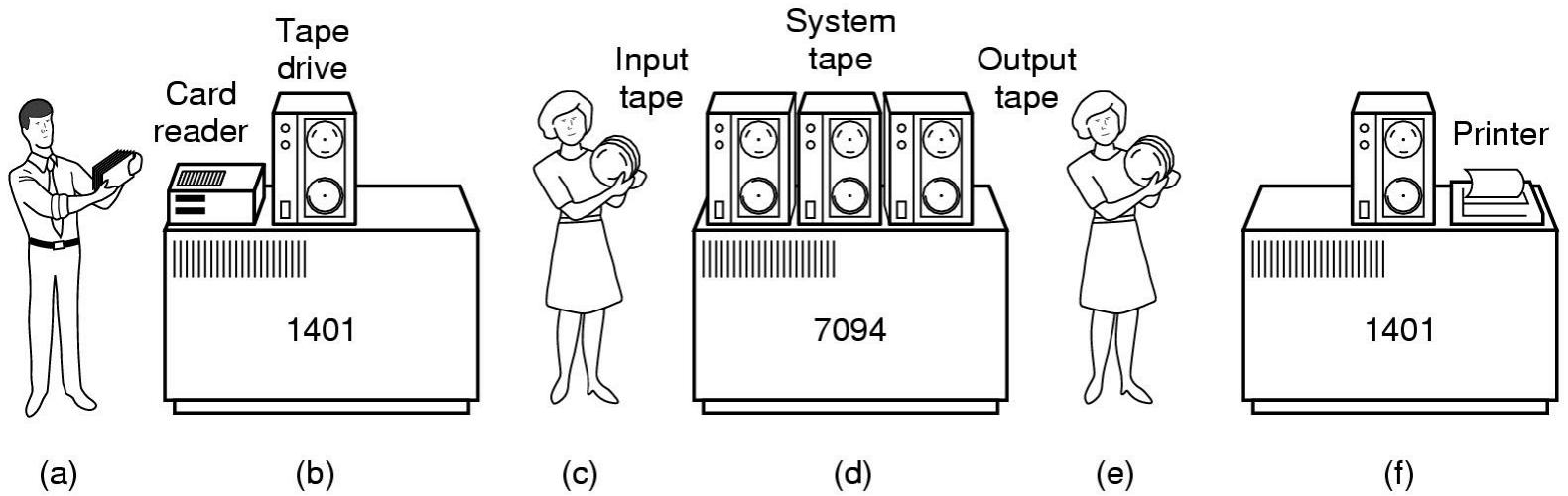
Operating system timeline

- First generation: 1945 – 1955
  - Vacuum tubes
  - Plug boards
- Second generation: 1955 – 1965
  - Transistors
  - Batch systems
- Third generation: 1965 – 1980
  - Integrated circuits
  - Multiprogramming
- Fourth generation: 1980 – present
  - Large scale integration
  - Personal computers
- Next generation: ???
  - Systems connected by high-speed networks?
  - Wide area resource management?

First generation: direct input

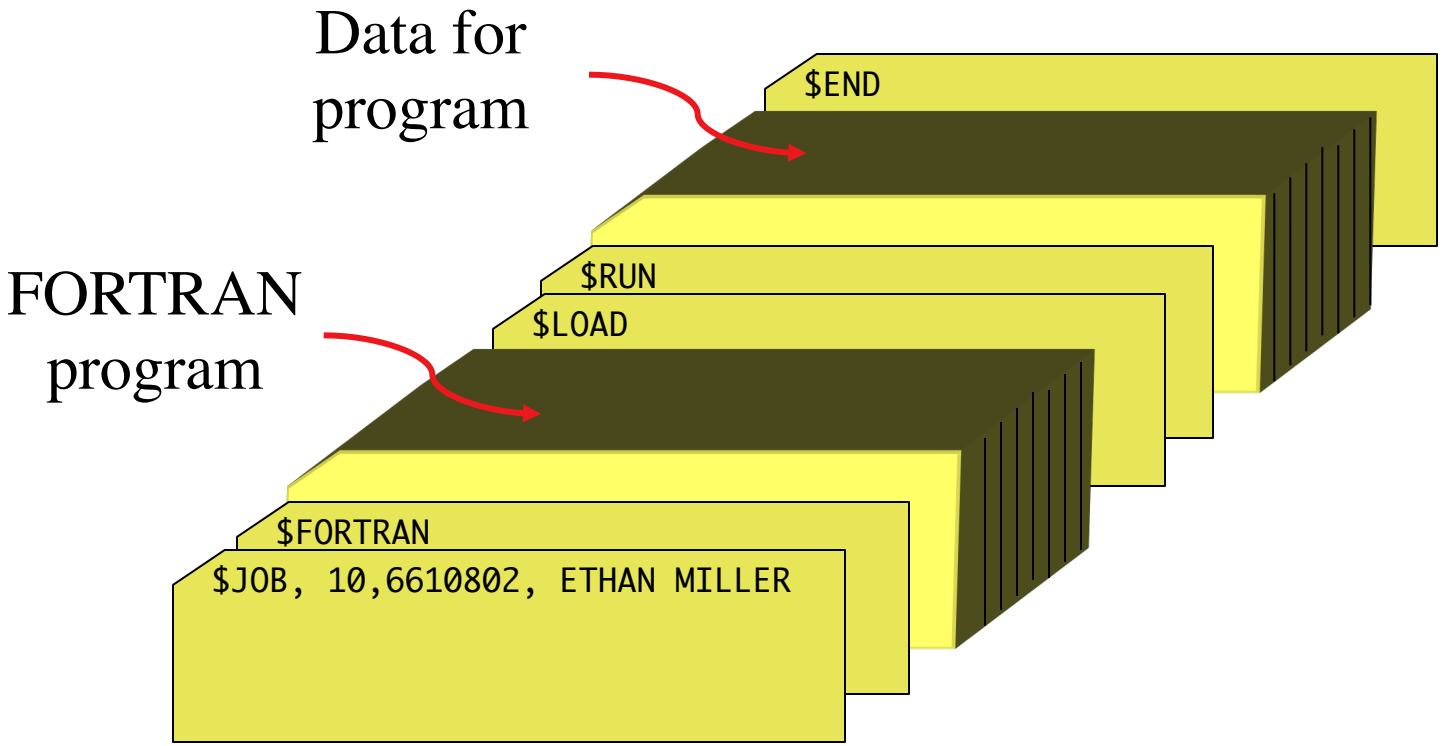
- Run one job at a time
  - Enter it into the computer (might require rewiring!)
  - Run it
  - Record the results
- Problem: lots of wasted computer time!
  - Computer was idle during first and last steps
  - Computers were ***very*** expensive!
- Goal: make better use of an expensive commodity: computer time

Second generation: batch systems



- Bring cards to 1401
- Read cards onto input tape
- Put input tape on 7094
- Perform the computation, writing results to output tape
- Put output tape on 1401, which prints output

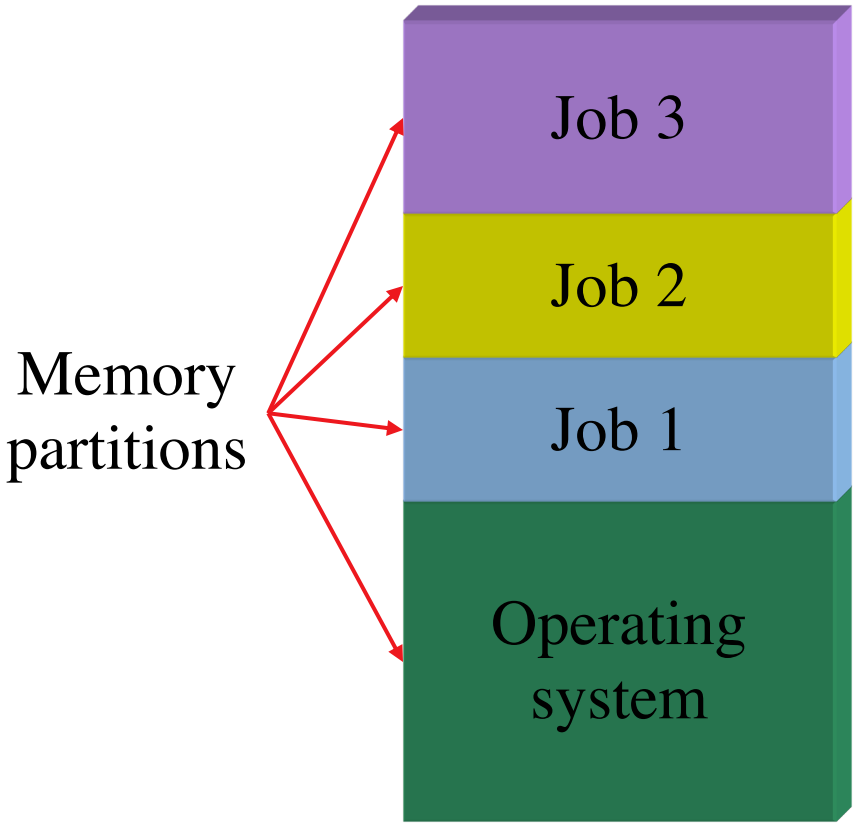
Structure of a typical 2nd generation job



Spooling

- Original batch systems used tape drives
- Later batch systems used disks for buffering
  - Operator read cards onto disk attached to the computer
  - Computer read jobs from disk
  - Computer wrote job results to disk
  - Operator directed that job results be printed from disk
- Disks enabled simultaneous peripheral operation on-line (spooling)
  - Computer overlapped I/O of one job with execution of another
  - Better utilization of the expensive CPU
  - Still only one job active at any given time

Third generation: multiprogramming



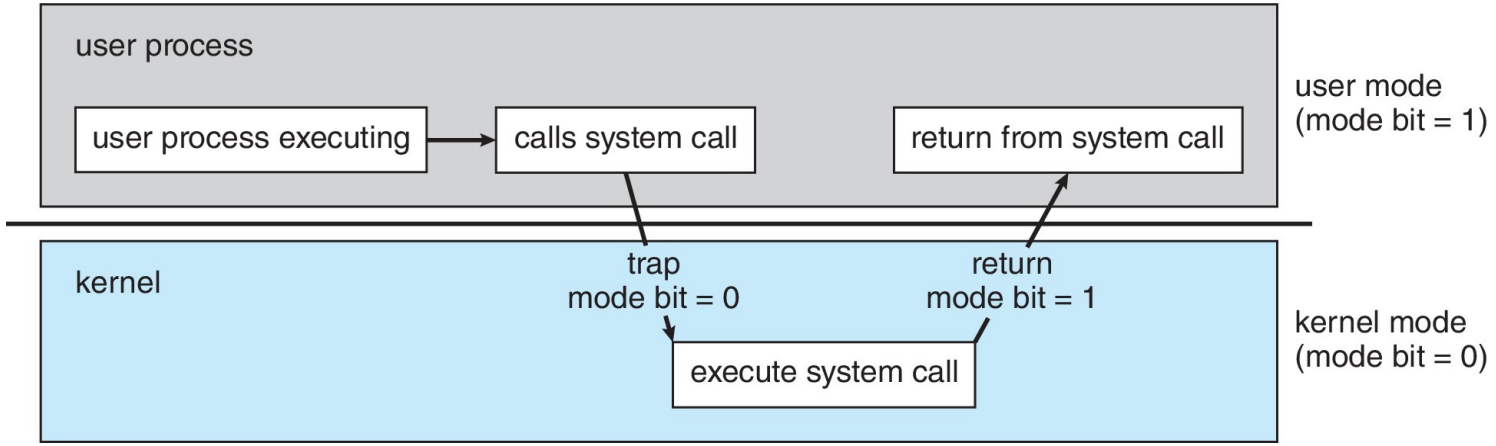
- Multiple jobs in memory
  - Protected from one another
- Operating system protected from each job as well
- Resources (time, hardware) split between jobs
- Still not interactive
  - User submits job
  - Computer runs it
  - User gets results minutes (hours, days) later

Dual-mode and Multimode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode**
    - The executing code has no ability to directly access hardware or reference memory.
    - Code running in user mode must delegate to system APIs to access hardware or memory.
    - Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable.
    - Most of the code running on your computer will execute in user mode.
  - **Kernel mode**
    - The executing code has *complete and unrestricted access* to the underlying hardware.
    - It can execute any CPU instruction and reference any memory address.
    - Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system.
    - Crashes in kernel mode are catastrophic; they will halt the entire PC.
  - **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

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



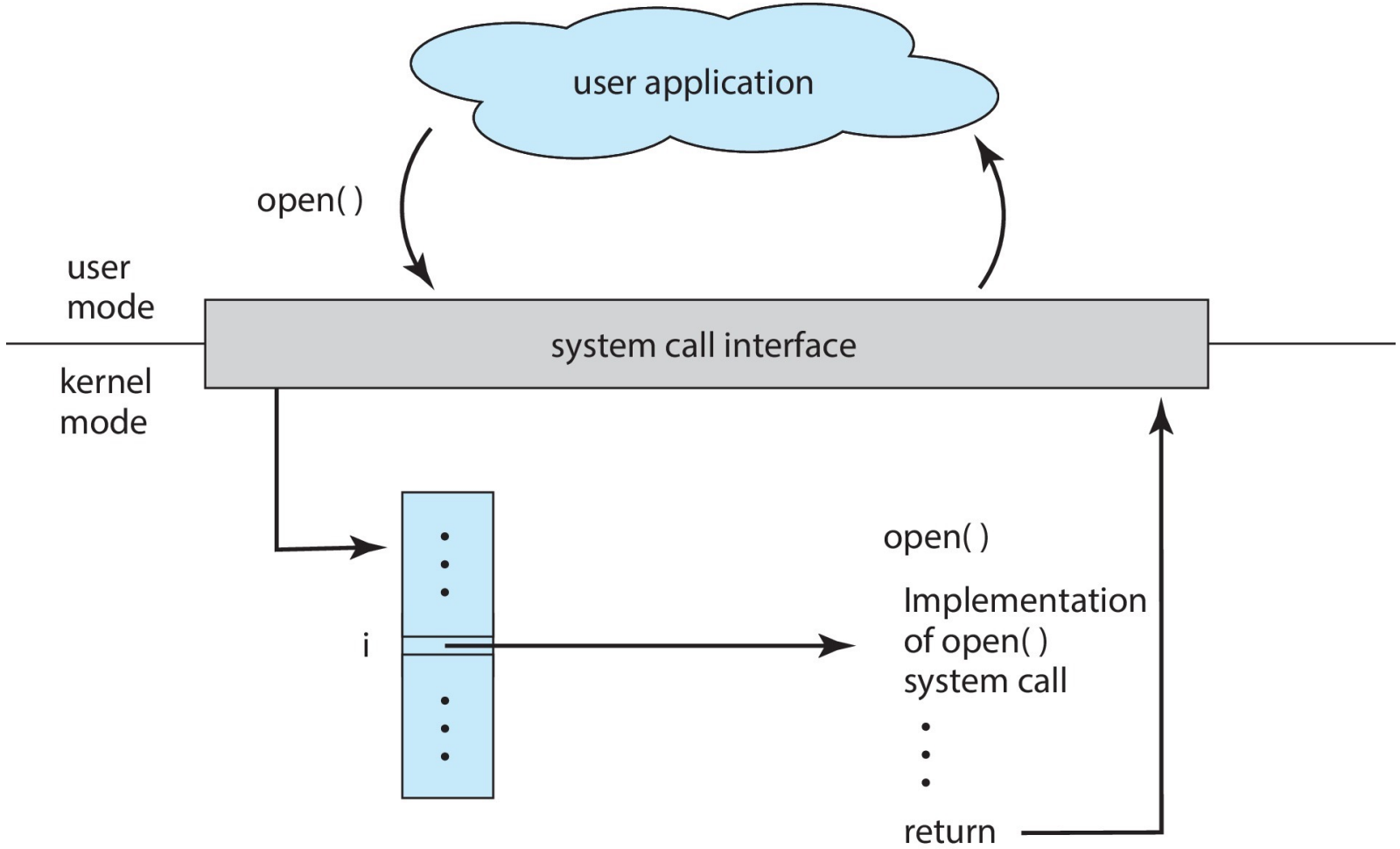
System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship



Timesharing

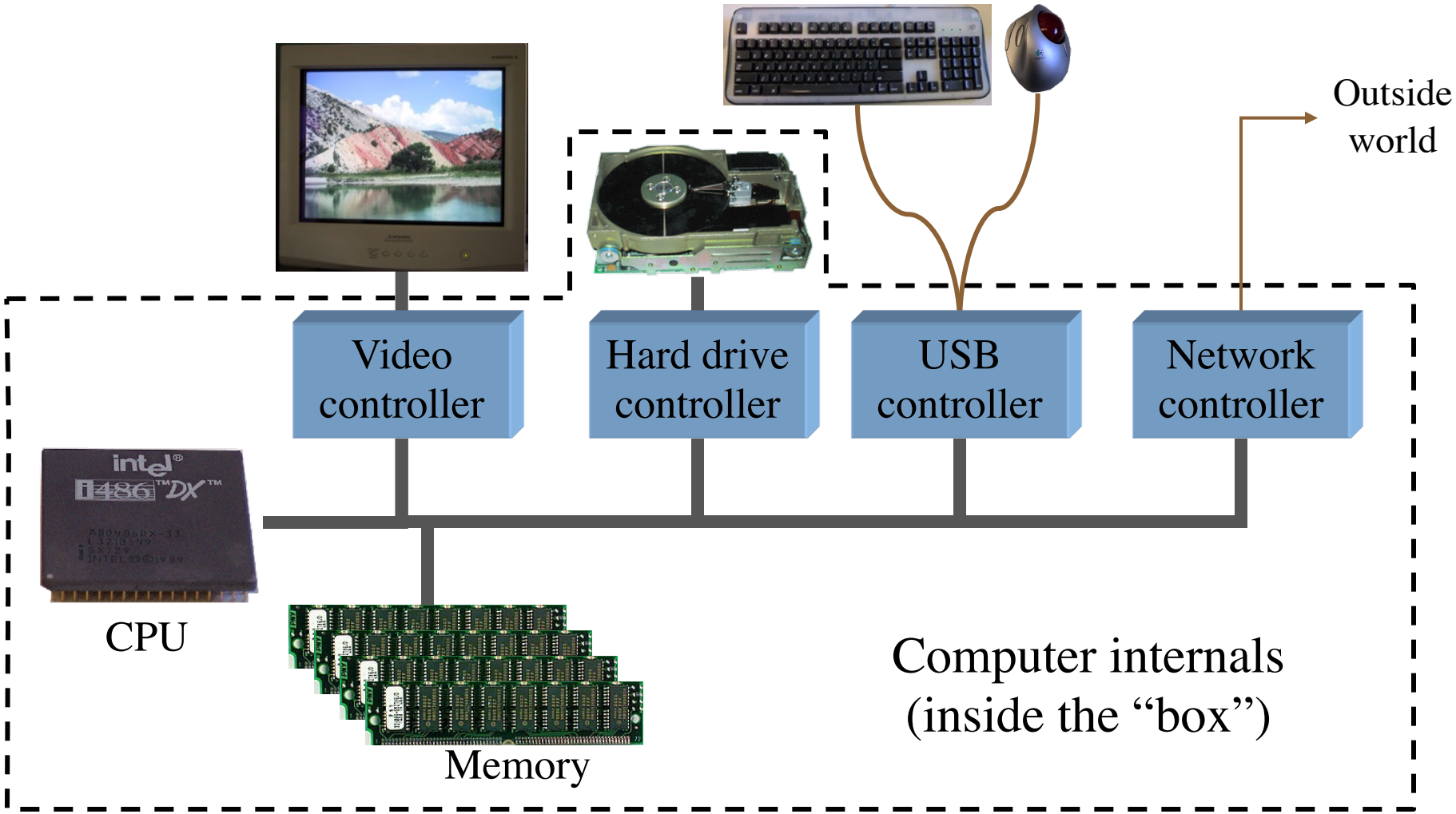
- Multiprogramming allowed several jobs to be active at one time
  - Initially used for batch systems
  - Cheaper hardware terminals -> interactive use
- Computer use got much cheaper and easier
  - No more “priesthood”
  - Quick turnaround meant quick fixes for problems

Types of modern operating systems

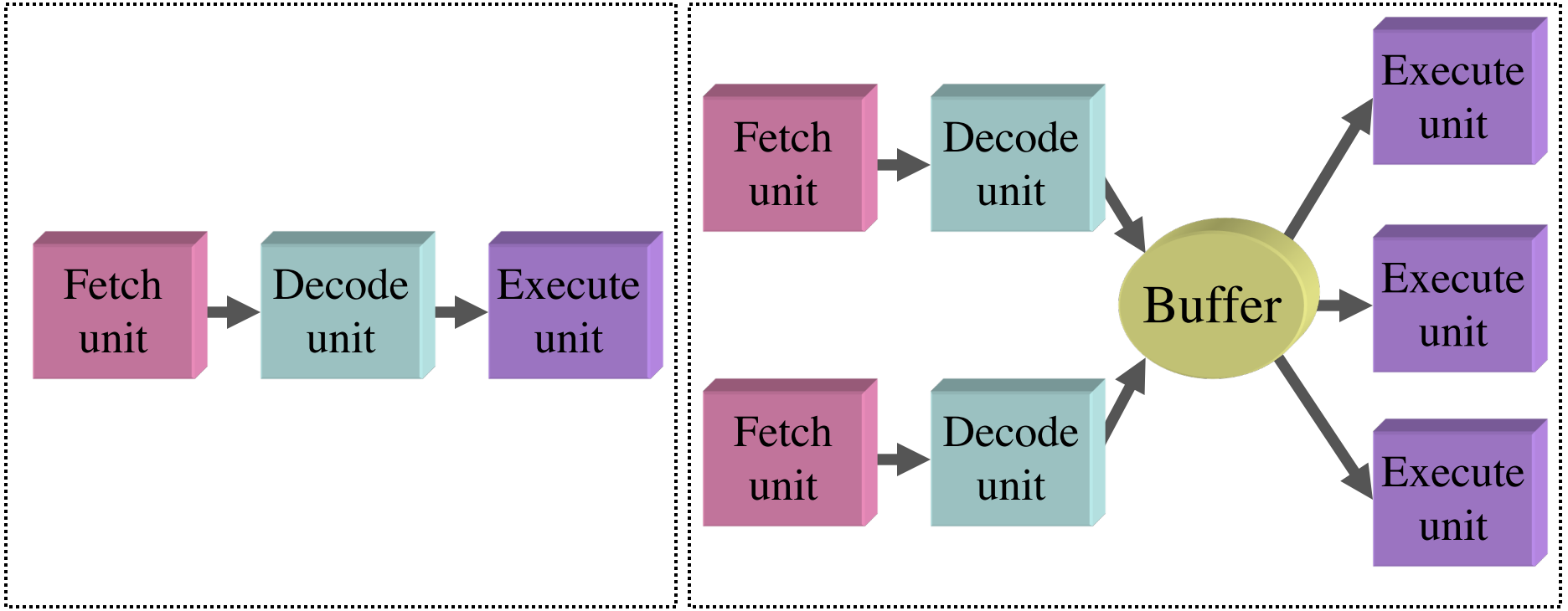
- Mainframe operating systems: MVS
- Server operating systems: FreeBSD, Solaris
- Multiprocessor operating systems: Cellular IRIX
- Personal computer operating systems: Windows, Unix
- Real-time operating systems: VxWorks
- Embedded operating systems
- Smart card operating systems

⇒Some operating systems can fit into more than one category

Components of a simple PC



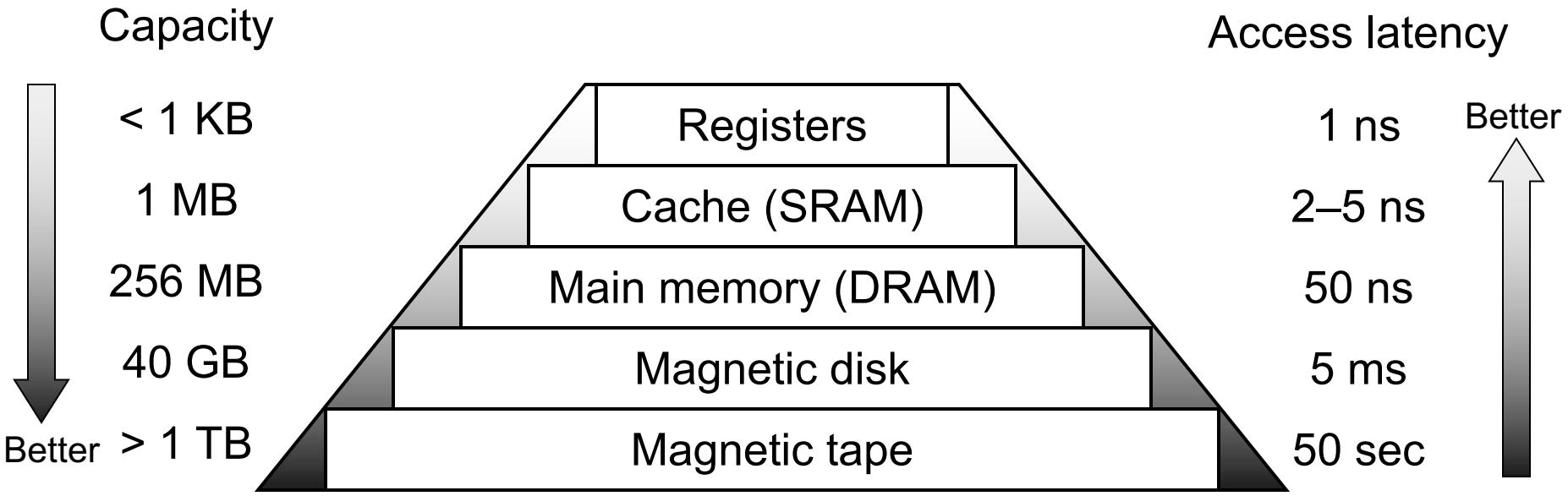
CPU internals



Pipelined CPU

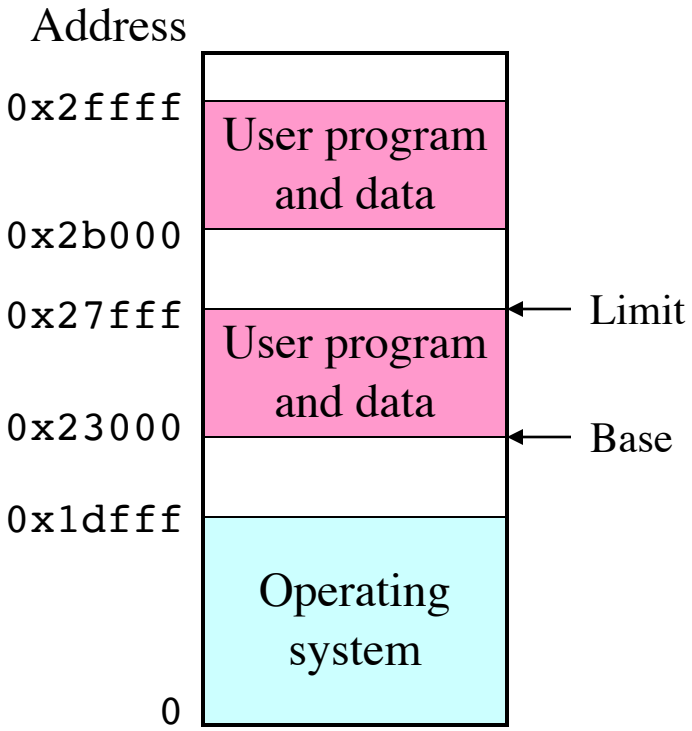
Superscalar CPU

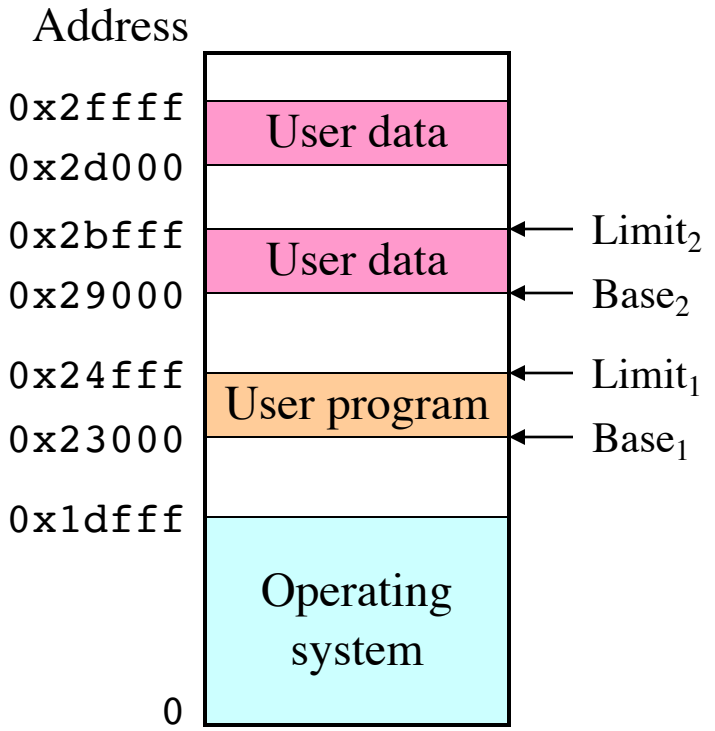
Storage pyramid



- Goal: really large memory with very low latency
  - Latencies are smaller at the top of the hierarchy
  - Capacities are larger at the bottom of the hierarchy
- Solution: move data between levels to create illusion of large memory with low latency

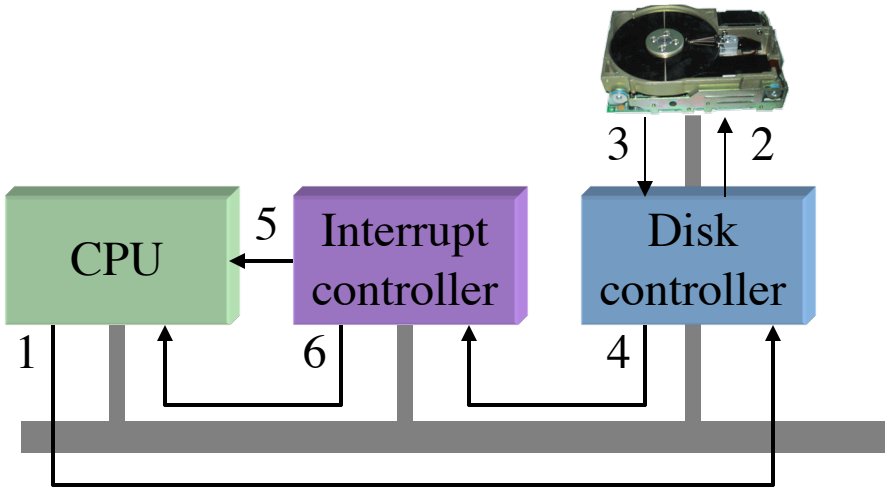
Memory

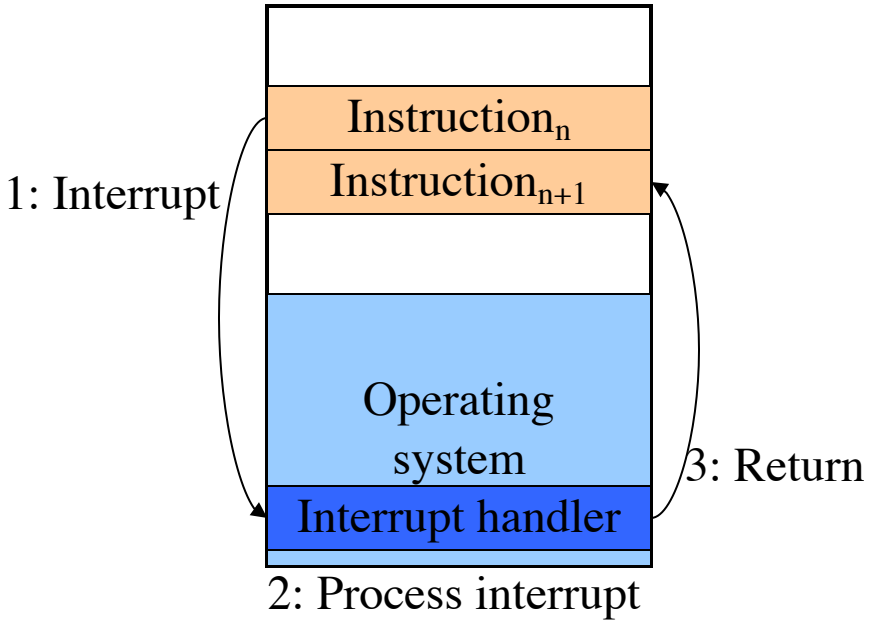




- Single base/limit pair: set for each process
- Two base/limit registers: one for program, one for data

Anatomy of a device request





- Left: sequence as seen by hardware
  - Request sent to controller, then to disk
  - Disk responds, signals disk controller which tells interrupt controller
  - Interrupt controller notifies CPU
- Right: interrupt handling (software point of view)

Operating systems concepts

- Processes (and trees of processes)
- Deadlock
- File systems & directory trees
- Pipes
- We'll cover all of these in more depth later on, but it's useful to have some basic definitions now