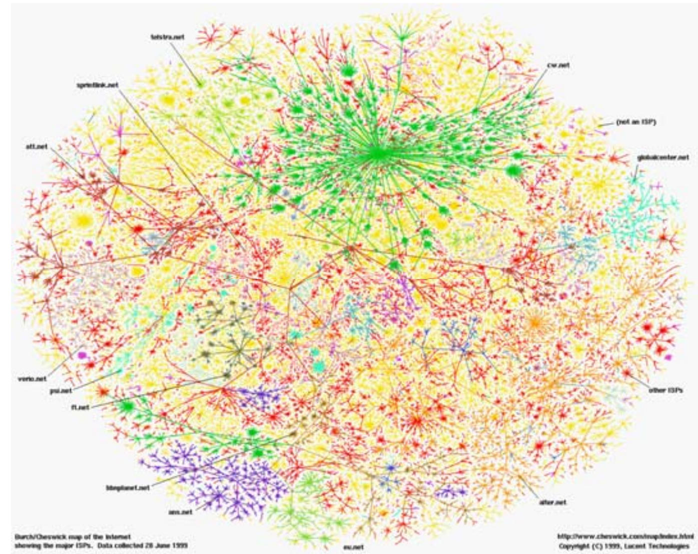
CS162
Operating Systems and Systems Programming
Lecture 1

What is an Operating System?

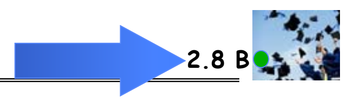
August 26th, 2015

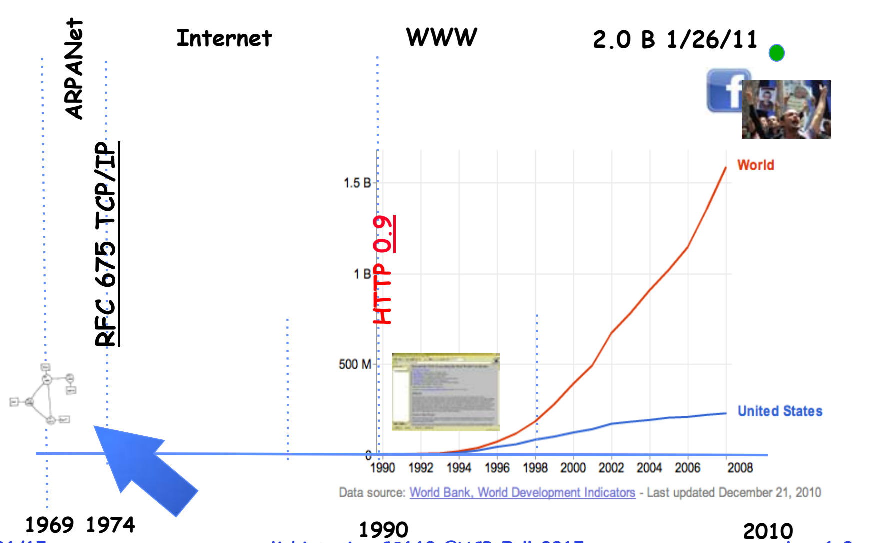
Prof. John Kubiatowicz
http://cs162.eecs.Berkeley.edu

Greatest Artifact of Human Civilization…



3 Billion Internet Users by …

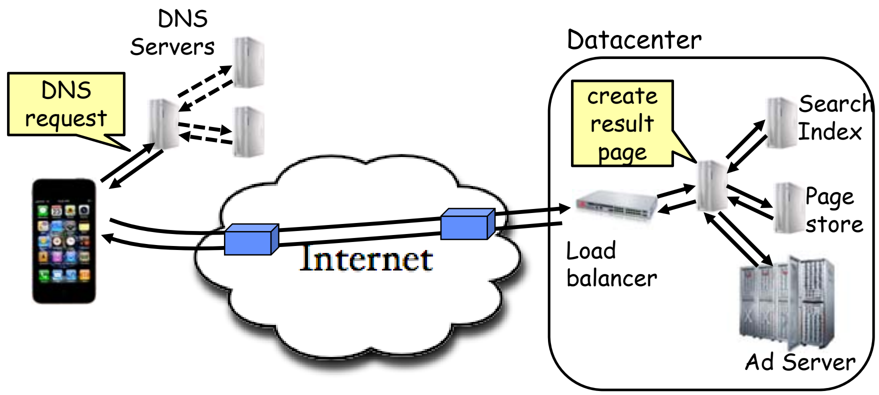




Operating Systems at the heart of it all …

- Make the incredible advance in the underlying hardware available to a rapid evolving body of applications.
  - Processing, Communications, Storage, Interaction
- The key building blocks
  - Scheduling
  - Concurrency
  - Address spaces
  - Protection, Isolation, Security
  - Networking, distributed systems
  - Persistent storage, transactions, consistency, resilience
  - Interfaces to all devices

Example: What's in a Search Query?



- Complex interaction of multiple components in multiple administrative domains
  - Systems, services, protocols, …

Why take CS162?

- Some of you will actually design and build operating systems or components of them.
  - Perhaps more now than ever
- Many of you will create systems that utilize the core concepts in operating systems.
  - Whether you build software or hardware
  - The concepts and design patterns appear at many levels
- All of you will build applications, etc. that utilize operating systems
  - The better you understand their design and implementation, the better use you'll make of them.

Goals for Today

- What is an Operating System?
  - And – what is it not?
- Examples of Operating Systems design
- What makes Operating Systems So Exciting?
- Oh, and "How does this class operate?"

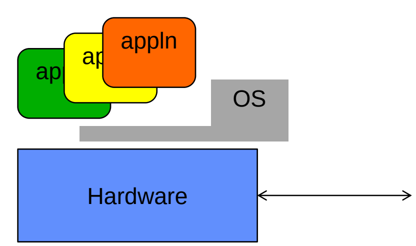
Interactive is important!

Ask Questions!

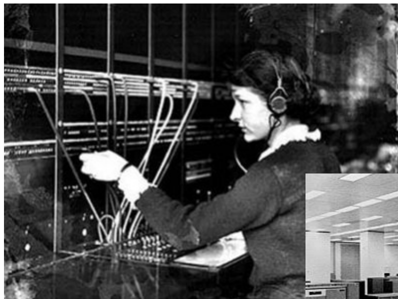
Slides courtesy of David Culler, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

What is an operating system?

- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities



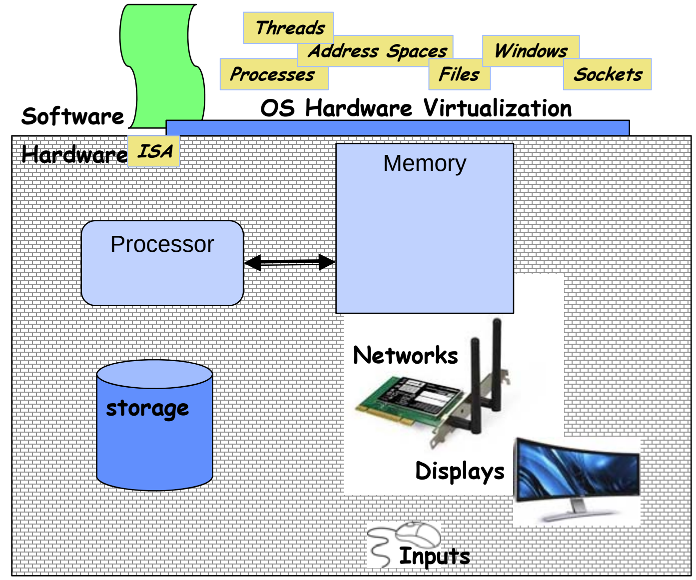
Operator …



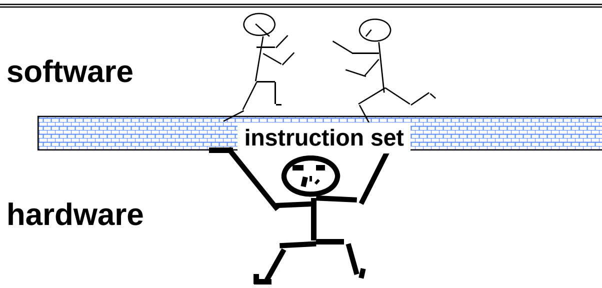
Switchboard Operator

Computer Operators

OS Basics: "Virtual Machine" Boundary

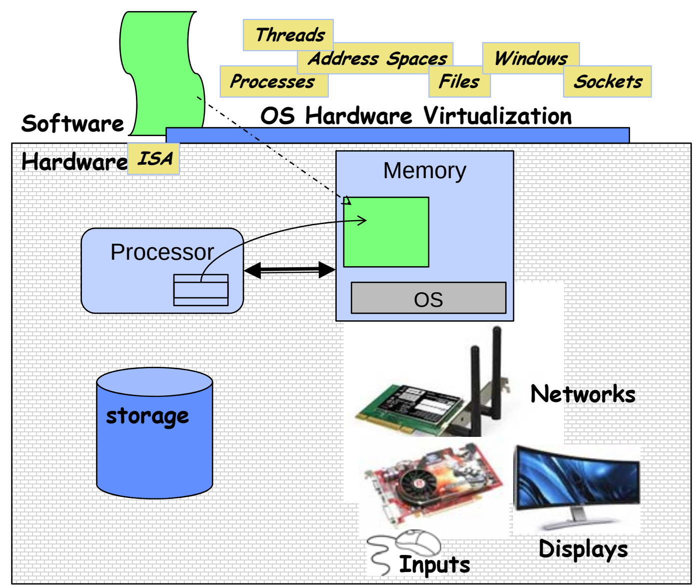


Interfaces Provide Essential Boundaries

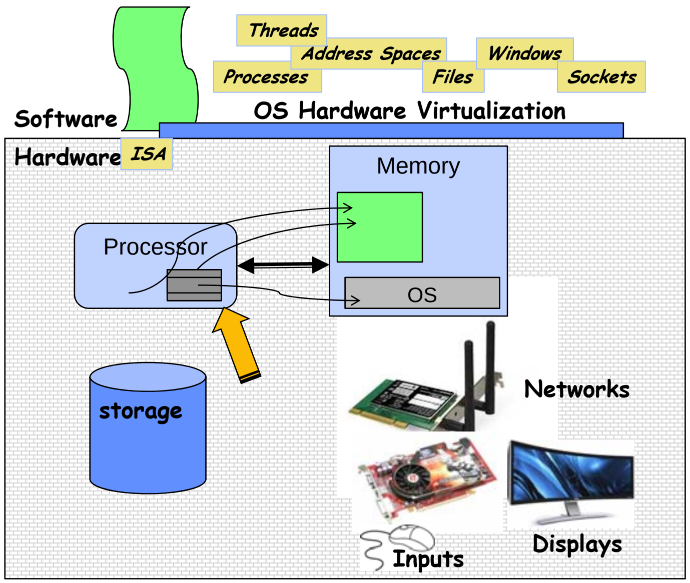


- Why do interfaces look the way that they do?
  - History, Functionality, Stupidity, Bugs, Management
  - CS152 ⇒ Machine interface
  - CS160 ⇒ Human interface
  - CS169 ⇒ Software engineering/management
- Should responsibilities be pushed across boundaries?
  - RISC architectures, Graphical Pipeline Architectures

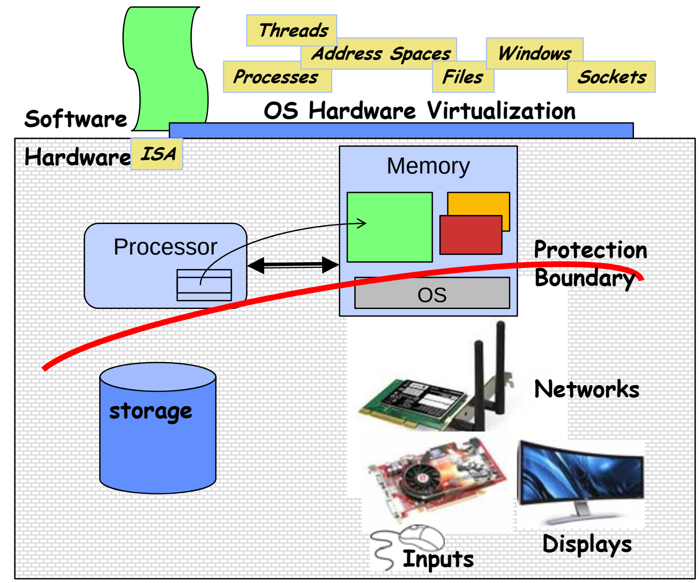
OS Basics: Program => Process



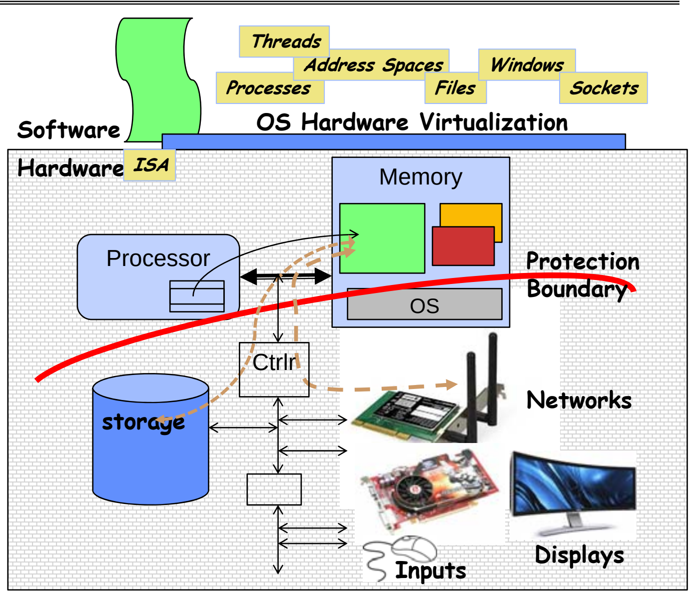
OS Basics: Context Switch



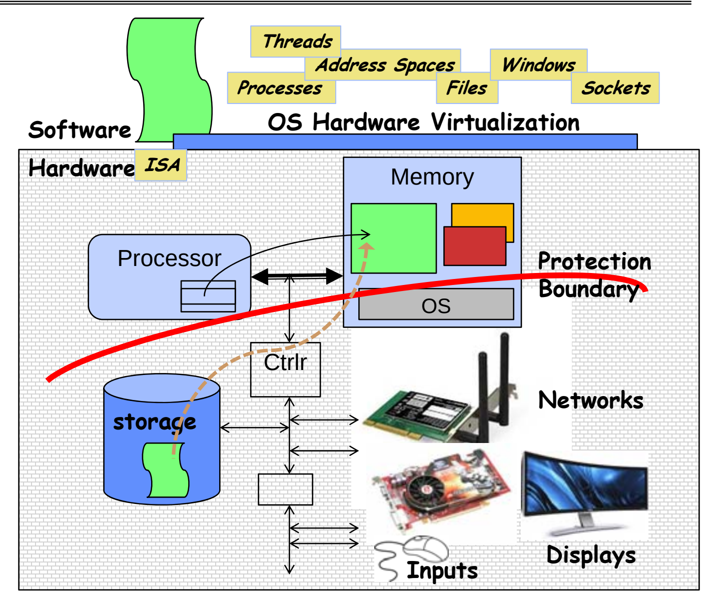
OS Basics: Scheduling, Protection



OS Basics: I/O

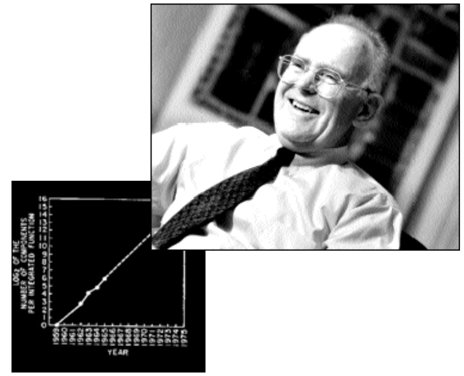


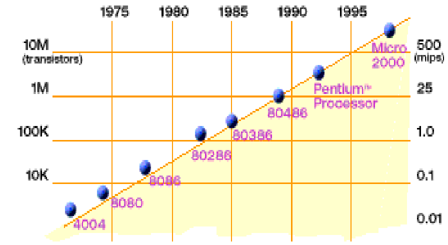
OS Basics: Loading



What makes Operating Systems exciting and Challenging?

Technology Trends: Moore's Law





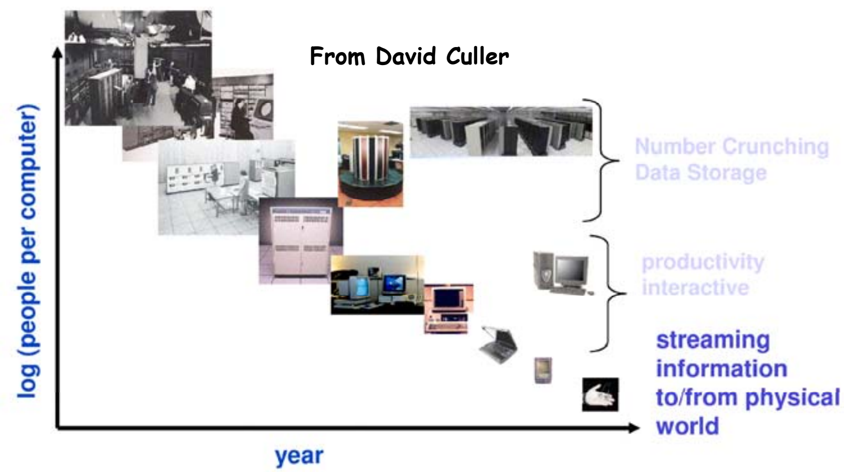
2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

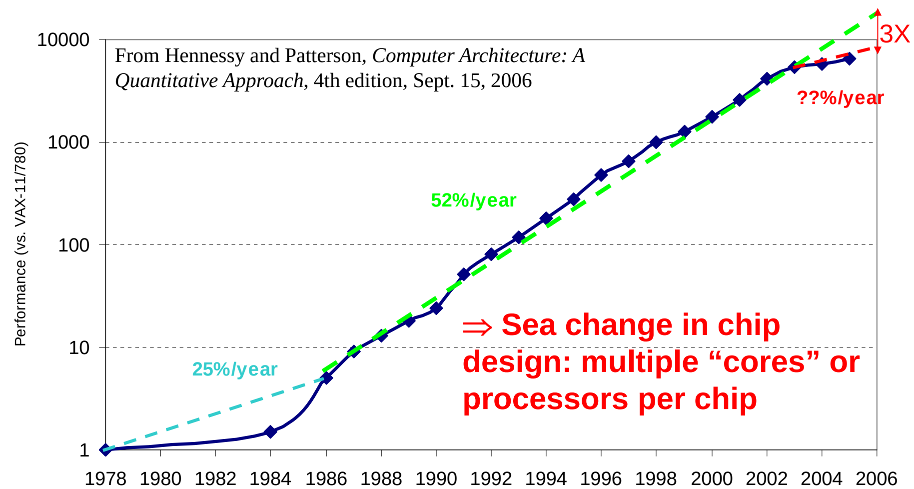
Microprocessors have become smaller, denser, and more powerful.

People-to-Computer Ratio Over Time



- Today: Multiple CPUs/person!
  - Approaching 100s?

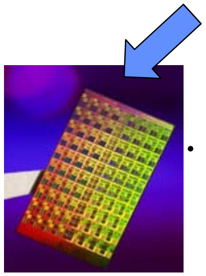
New Challenge: Slowdown in Joy's law of Performance



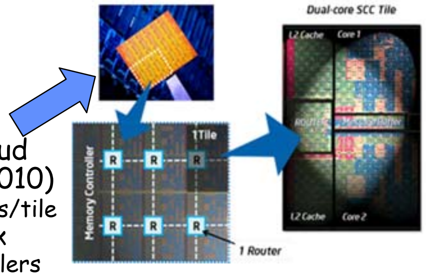
⇒ Sea change in chip design: multiple "cores" or processors per chip

- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

ManyCore Chips: The future is here

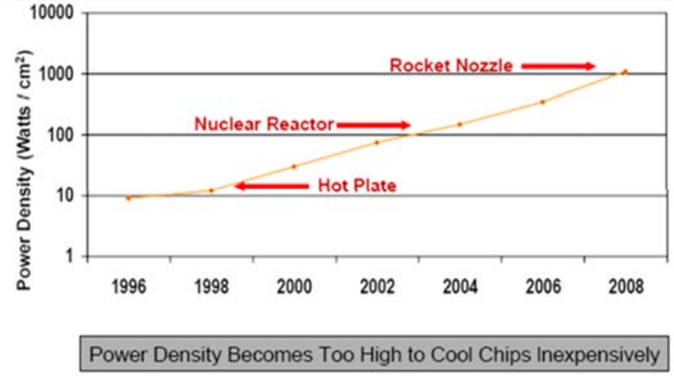


- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers
  - Hardware support for message-passing



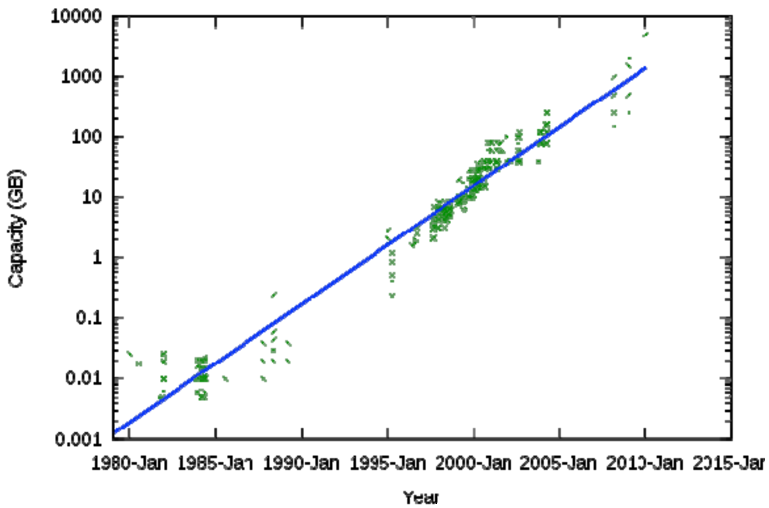
- "ManyCore" refers to many processors/chip
  - 64? 128? Hard to say exact boundary
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- **Parallelism must be exploited at all levels**

Another Challenge: Power Density



- Moore's Law Extrapolation
  - Potential power density reaching amazing levels!
- Flip side: Battery life very important
  - Moore's law can yield more functionality at equivalent (or less) total energy consumption

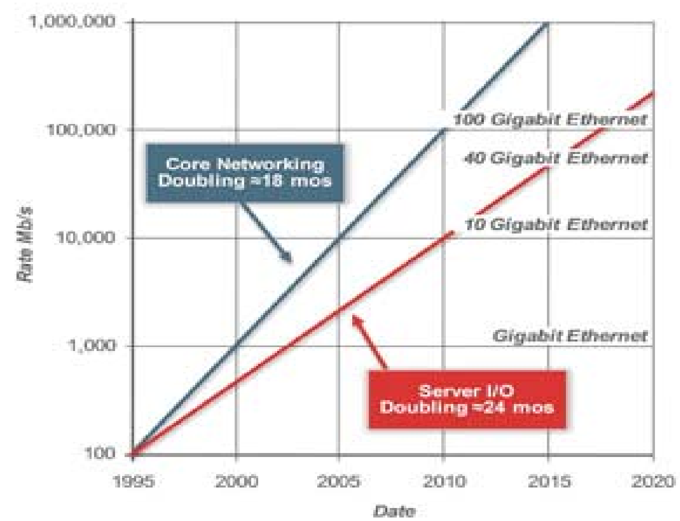
Storage Capacity



- *Retail* hard disk capacity in GB

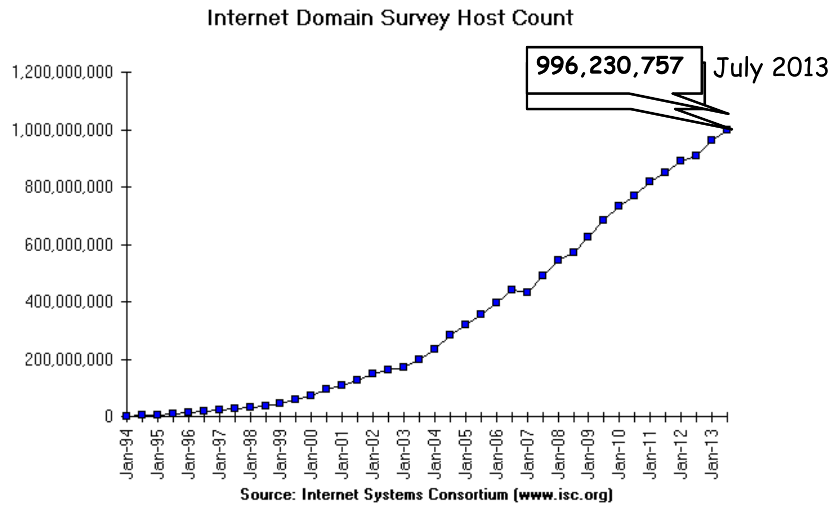
(source: http://www.digitaltonto.com/2011/our-emergent-digital-future/)

Network Capacity



(source: http://www.ospmag.com/issue/article/Time-Is-Not-Always-On-Our-Side)

Internet Scale: .96 Billion Hosts



https://www.isc.org/solutions/survey

Internet Scale: Almost 2.5 Billion Users!

WORLD INTERNET USAGE AND POPULATION STATISTICS
December 31, 2013

World Regions	Population (2014 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2014	Users % of Table
Africa	1,125,721,038	4,514,400	240,146,482	21.3 %	5,219.6 %	8.6 %
Asia	3,996,408,007	114,304,000	1,265,143,702	31.7 %	1,006.8 %	45.1 %
Europe	825,802,657	105,096,093	566,261,317	68.6 %	438.8 %	20.2 %
Middle East	231,062,860	3,284,800	103,829,614	44.9 %	3,060.9 %	3.7 %
North America	353,860,227	108,096,800	300,287,577	84.9 %	177.8 %	10.7 %
Latin America / Caribbean	612,279,181	18,068,919	302,006,016	49.3 %	1,571.4 %	10.8 %
Oceania / Australia	36,724,649	7,620,480	24,804,226	67.5 %	225.5 %	0.9 %
WORLD TOTAL	7,181,858,619	360,985,492	2,802,478,934	39.0 %	676.3 %	100.0 %

NOTES: (1) Internet Usage and World Population Statistics are for December 31, 2013. (2) CLICK on each world region name for detailed regional usage information. (3) Demographic (Population) numbers are based on data from the US Census Bureau and local census agencies. (4) Internet usage information comes from data published by Nielsen Online, by the International Telecommunications Union, by GfK, local ICT Regulators and other reliable sources. (5) For definitions, disclaimers, navigation help and methodology, please refer to the Site Surfing Guide. (6) Information in this site may be cited, giving the due credit to www.internetworldstats.com. Copyright © 2001 - 2014, Miniwatts Marketing Group. All rights reserved worldwide.

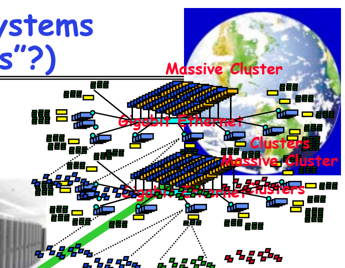
(source: http://www.internetworldstats.com/stats.htm)

Not Only PCs connected to the Internet

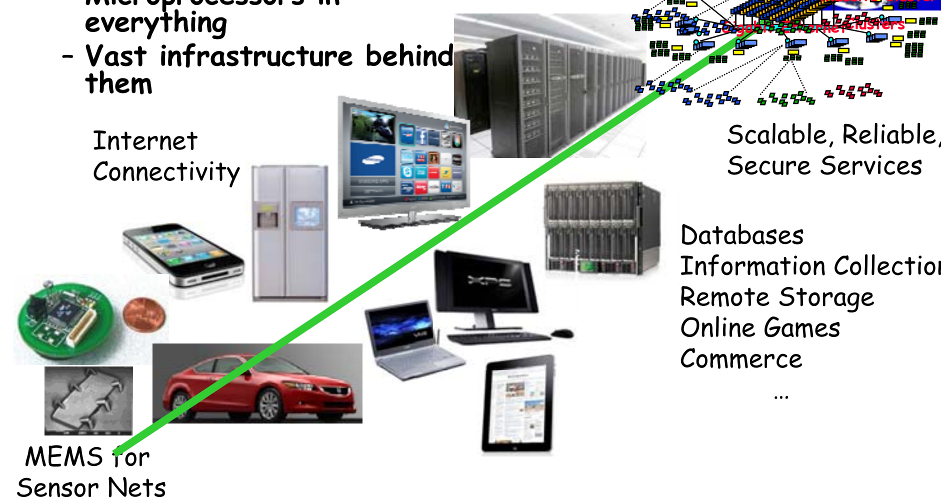
- Smartphone shipments now exceed PC shipments!
- 2011 shipments:
  - 487M smartphones
  - 414M PC clients
    - » 210M notebooks
    - » 112M desktops
    - » 63M tablets
  - 25M smart TVs

- 4 billion phones in the world → smartphone over next decade

Societal Scale Information Systems (Or the "Internet of Things"?)

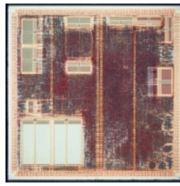


- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them

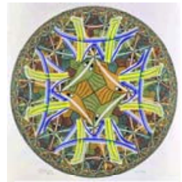


Who am I?

- Professor John Kubiatowicz (Prof "Kubi")
  - Background in Hardware Design
    - » Alewife project at MIT
    - » Designed CMMU, Modified SPAR C processor
    - » Helped to write operating system
  - Background in Operating Systems
    - » Worked for Project Athena (MIT)
    - » OS Developer (device drivers, network file systems)
    - » Worked on Clustered High-Availability systems (CLAM Associates)
    - » OS lead researcher for Tessellation OS
  - Peer-to-Peer
    - » OceanStore project – Store your data for 1000 years
    - » Tapestry and Bamboo – Find your data around globe
    - » SwarmLab Global DataPlane for the Internet of Things (IoT)
  - Quantum Computing
    - » Well, this is just cool, but probably not apropos



Alewife



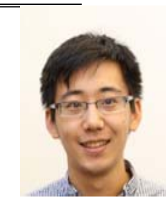
Tessellation

OceanStore

CS162 Team - GSIs:

- William Liu
  - Head GSI
  - Section 109, F 2-3 @ 6 Evans

- Andrew Chen
  - Section 110, F 10-11 @ 205 Dwinelle



- Roger Chen
  - Section 105, F 11-12 @ 102 Latimer
  - Section 107, F 12-1 @ 179 Stanley

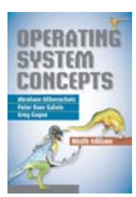
- Aleks Kamko
  - Section 104, F 10-11 @ 385 LeConte
  - Section 106, F 11-12 @ B0051 Hildebrand

- Alec Mouri
  - Section 103, T 4-5 @ 102 Wurster
  - Section 108, F 1-2 @ 87 Evans

- Jackson Chang
  - Section 101, T 2-3 @ 310 Hearst Mining
  - Section 102, T 3-4 @ 3 Evans

Infrastructure, Textbook & Readings

- Infrastructure
  - Website: http://cs162.eecs.berkeley.edu
  - Piazza: https://piazza.com/berkeley/fall2015/cs162
  - Webcast: Yes! Will post link when available
- Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
- Recommend: Operating Systems Concepts, 9th Edition Silbershatz, Galvin, Gagne
  - Copies in Bechtel
- Online supplements
  - See course website
  - Includes Appendices, sample problems, etc.
  - Networking, Databases, Software Eng, Security
  - Some Research Papers!



Syllabus

- OS Concepts: How to Navigate as a Systems Programmer!
  - Process, I/O, Networks and VM
- Concurrency
  - Threads, scheduling, locks, deadlock, scalability, fairness
- Address Space
  - Virtual memory, address translation, protection, sharing
- File Systems
  - i/o devices, file objects, storage, naming, caching, performance, paging, transactions, databases
- Distributed Systems (8)
  - Protocols, N-Tiers, RPC, NFS, DHTs, Consistency, Scalability, multicast
- Reliability & Security
  - Fault tolerance, protection, security
- Cloud Infrastructure

Learning by Doing

- Individual Homework (1-2 weeks): Learn Systems Programming
  - 0. Tools, Autograding, recall C, executable
  - 1. Simple Shell
  - 2. Web server
  - …
- Three Group Projects (Pintos in C)
  - 1. Threads & Scheduling
  - 2. User-programs
  - 3. File Systems

Getting started

- Start homework 0 immediately
  - Gets cs162-xx@cory.eecs.berkeley.edu (and other inst m/c)
  - Github account
  - Registration survey
  - Vagrant virtualbox – VM environment for the course
    - » Consistent, managed environment on your machine
  - Get familiar with all the cs162 tools
  - Submit to autograder via git
- Go to section this week (starting tomorrow!)
  - Bring laptop
  - Also, watch for us to post various small help-sessions
- Waitlist ???
  - Drop Deadline: September 4th
  - If you are not serious about taking, please drop early

Group Project Simulates Industrial Environment

- Project teams have 4 members (try really hard to get 4 members – 3 members requires serious justification)
  - Must work in groups in "the real world"
  - Same section much preferred
- Communicate with colleagues (team members)
  - Communication problems are natural
  - What have you done?
  - What answers you need from others?
  - You must document your work!!!
- Communicate with supervisor (TAs)
  - What is the team's plan?
  - What is each member's responsibility?
  - Short progress reports are required
  - Design Documents: High-level description for a manager!

Grading

- 40% midterms/Final
- 40% projects
- 15% homework
- 5% participation
- Project grading
  - [10 pts] Initial design
  - [10 pts] Design review
  - [10 pts] Design document
  - [60 pts] Code (3 checkpoints)
  - [10 pts] Final design
- Submission via *git push* to release branch
  - Triggers autograder
- Regular *git push* so TA sees your progress

Personal Integrity

- UCB Academic Honor Code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

http://asuc.org/honorcode/resources/HC%20Guide%20for%20Syllabi.pdf

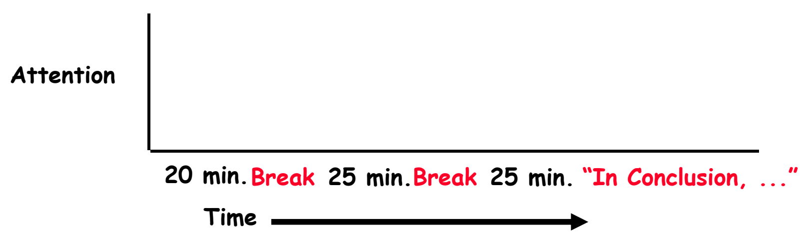
CS 162 Collaboration Policy

Explaining a concept to someone in another group
Discussing algorithms/testing strategies with other groups
Helping debug someone else's code (in another group)
Searching online for generic algorithms (e.g., hash table)

Sharing code or test cases with another group
Copying OR reading another group's code or test cases
Copying OR reading online code or test cases from from prior years

We compare all project submissions against prior year submissions and online solutions and will take actions (described on the course overview page) against offenders

Typical Lecture Format



- 1-Minute Review
- 20-Minute Lecture
- 5- Minute Administrative Matters
- 25-Minute Lecture
- 5-Minute Break (water, stretch)
- 25-Minute Lecture
- Instructor will come to class early & stay after to answer questions

Lecture Goal

Interactive!!!

What is an Operating System?

- Referee
  - Manage sharing of resources, Protection, Isolation
    - » Resource allocation, isolation, communication

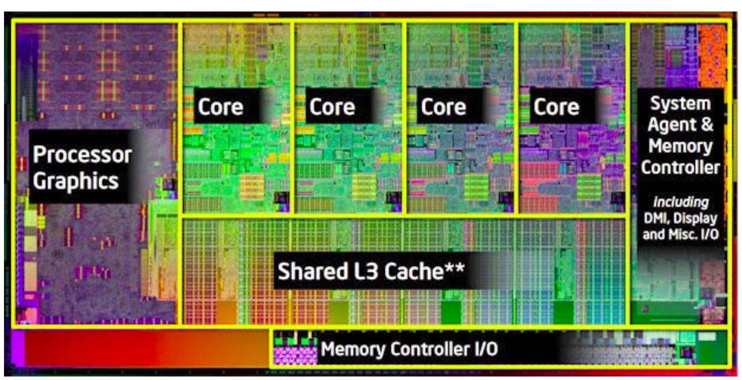
- Illusionist
  - Provide clean, easy to use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization

- Glue
  - Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

Challenge: Complexity

- Applications consisting of…
  - … a variety of software modules that …
  - … run on a variety of devices (machines) that
    - » … implement different hardware architectures
    - » … run competing applications
    - » … fail in unexpected ways
    - » … can be under a variety of attacks
- Not feasible to test software for all possible environments and combinations of components and devices
  - The question is not whether there are bugs but how serious are the bugs!

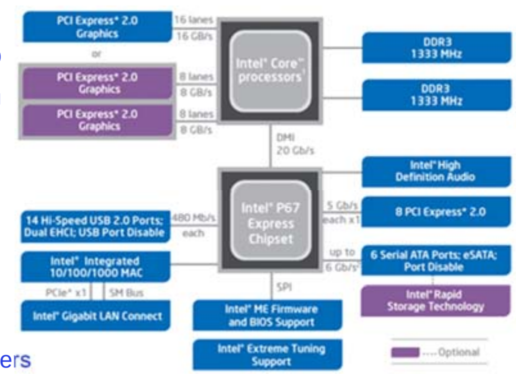
A modern processor: SandyBridge

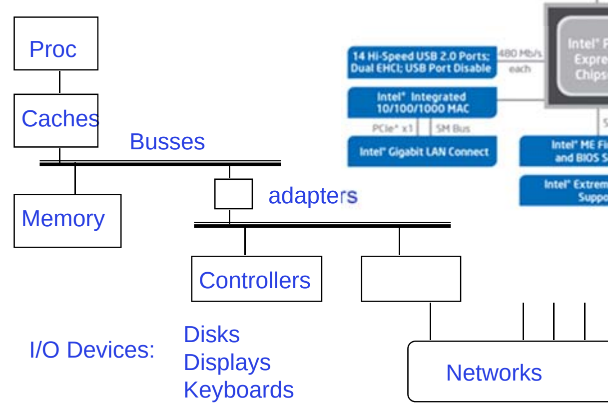


- Package: LGA 1155
  - 1155 pins
  - 95W design envelope
- Cache:
  - L1: 32K Inst, 32K Data (3 clock access)
  - L2: 256K (8 clock access)
  - Shared L3: 3MB – 20MB (not out yet)
- Transistor count:
  - 504 Million (2 cores, 3MB L3)
  - 2.27 Billion (8 cores, 20MB L3)
- Note that ring bus is on high metal layers – above the Shared L3 Cache

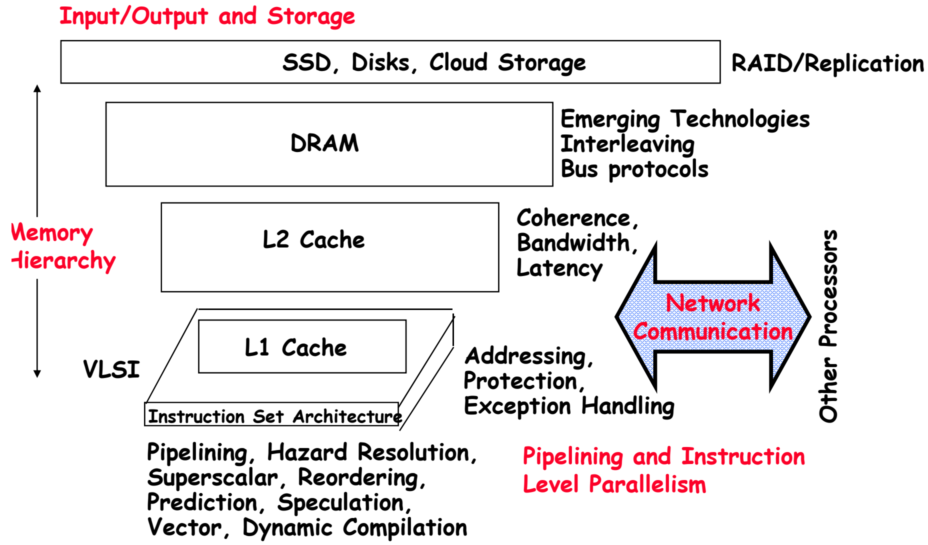
Functionality comes with great complexity!

SandyBridge I/O Configuration

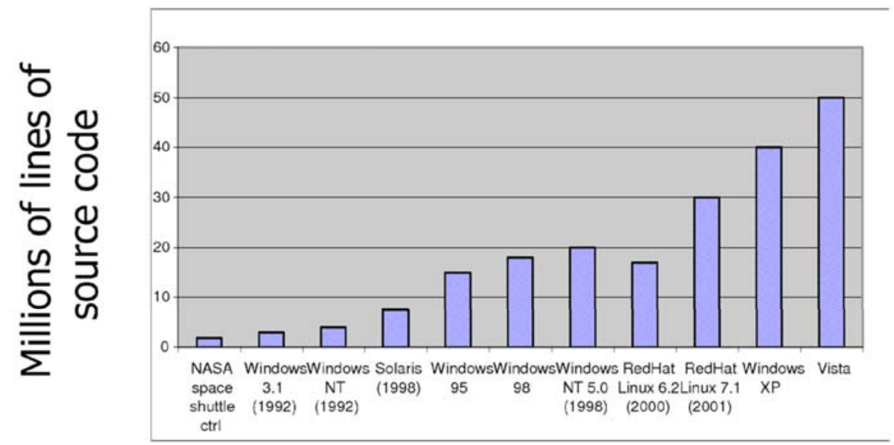




Sample of Computer Architecture Topics

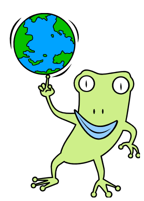


Increasing Software Complexity

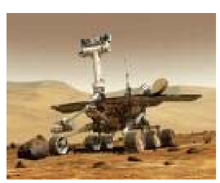


From MIT's 6.033 course

Example: Some Mars Rover ("Pathfinder") Requirements



- Pathfinder hardware limitations/complexity:
  - 20Mhz processor, 128MB of DRAM, VxWorks OS
  - cameras, scientific instruments, batteries, solar panels, and locomotion equipment
  - Many independent processes work together
- Can't hit reset button very easily!
  - Must reboot itself if necessary
  - Must always be able to receive commands from Earth
- Individual Programs must not interfere
  - Suppose the MUT (Martian Universal Translator Module) buggy
  - Better not crash antenna positioning software!
- Further, all software may crash occasionally
  - Automatic restart with diagnostics sent to Earth
  - Periodic checkpoint of results saved?
- Certain functions time critical:
  - Need to stop before hitting something
  - Must track orbit of Earth for communication
- A lot of similarity with the Internet of Things?
  - *Complexity,* QoS, Inaccessbility, Power limitations … ?



How do we tame complexity?

- Every piece of computer hardware different
  - Different CPU
    - » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - » Cable, DSL, Wireless, Firewalls,…
- Questions:
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?

OS Tool: Virtual Machine Abstraction

Application

Virtual Machine Interface

Operating System

Physical Machine Interface

Hardware

- Software Engineering Problem:
  - Turn hardware/software quirks ⇒ what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc…
- For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What's the hardware interface? (physical reality)
  - What's the application interface? (nicer abstraction)

Virtual Machines

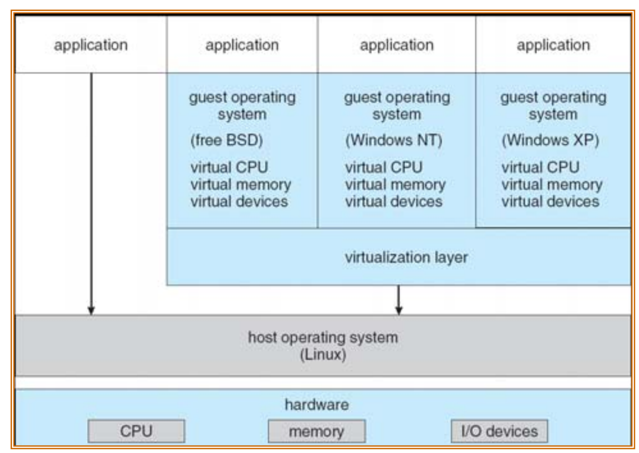
- Software emulation of an abstract machine
  - Give programs illusion they own the machine
  - Make it look like hardware has features you want
- Two types of "Virtual Machine"s
  - Process VM: supports the execution of a single program; this functionality typically provided by OS
  - System VM: supports the execution of an entire OS and its applications (e.g., VMWare Fusion, Virtual box, Parallels Desktop, Xen)

Process VMs

- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different devices appear to have same high level interface
  - Device interfaces more powerful than raw hardware
    - » Bitmapped display ⇒ windowing system
    - » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)
- Fault Isolation
  - Processes unable to directly impact other processes
  - Bugs cannot crash whole machine
- Protection and Portability
  - Java interface safe and stable across many platforms

System Virtual Machines: Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs



What is an Operating System,… Really?

- Most Likely:
  - Memory Management
  - I/O Management
  - CPU Scheduling
  - Communications? (Does Email belong in OS?)
  - Multitasking/multiprogramming?
- What about?
  - File System?
  - Multimedia Support?
  - User Interface?
  - Internet Browser? ☺
- Is this only interesting to Academics??

Operating System Definition (Cont.)

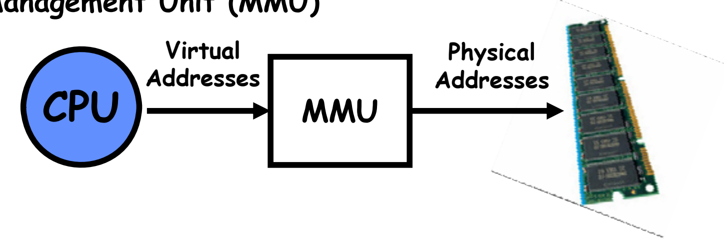
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the kernel.
  - Everything else is either a system program (ships with the operating system) or an application program

Example: Protecting Processes from Each Other

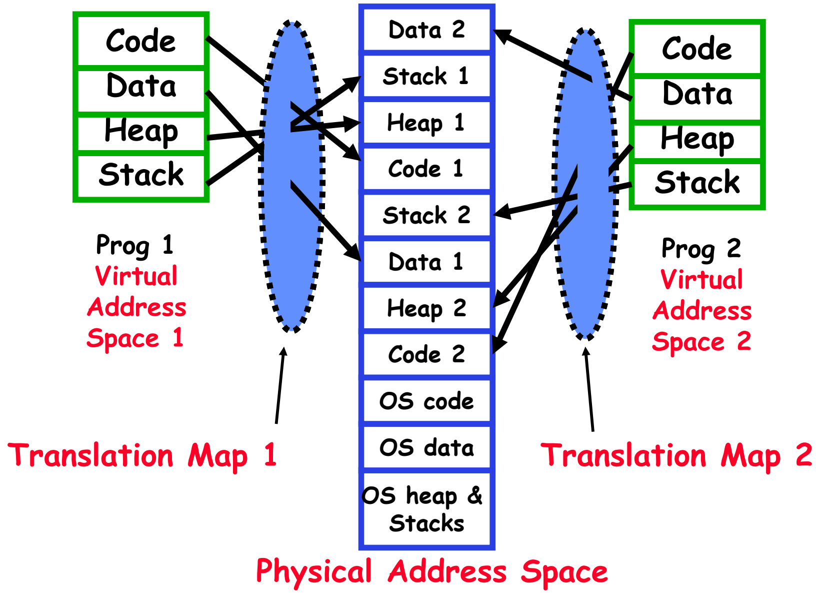
- Problem: Run multiple applications in such a way that they are protected from one another
- Goal:
  - Keep User Programs from Crashing OS
  - Keep User Programs from Crashing each other
  - [Keep Parts of OS from crashing other parts?]
- (Some of the required) Mechanisms:
  - Address Translation
  - Dual Mode Operation
- Simple Policy:
  - Programs are not allowed to read/write memory of other Programs or of Operating System

Address Translation

- Address Space
  - A group of memory addresses usable by something
  - Each program (process) and kernel has potentially different address spaces.
- Address Translation:
  - Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  - Mapping *often* performed in Hardware by Memory Management Unit (MMU)

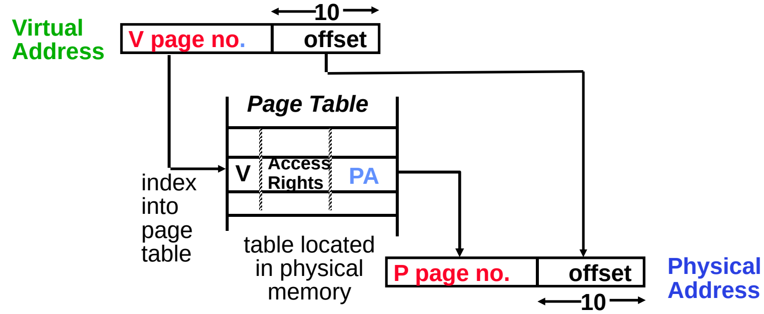


Example of Address Translation



Address Translation Details

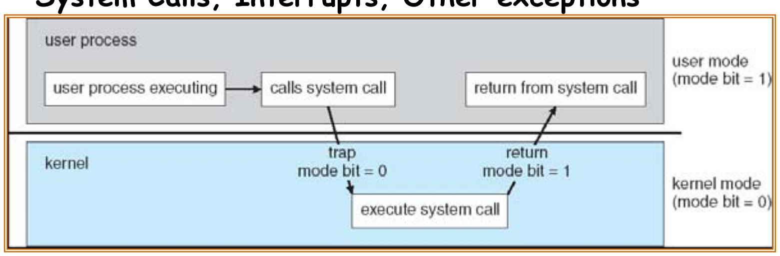
- For now, assume translation happens with table (called a Page Table):



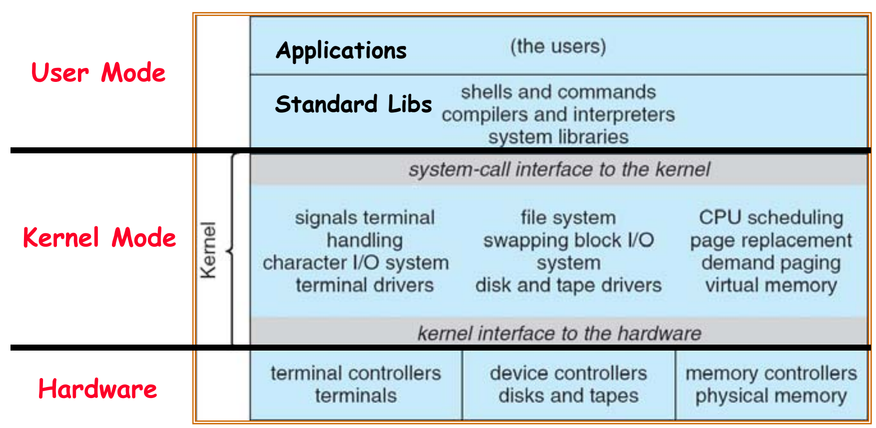
- Translation helps protection:
  - Control translations, control access
  - Should Users be able to change Page Table???

Dual Mode Operation

- Hardware provides at least two modes:
  - "Kernel" mode (or "supervisor" or "protected")
  - "User" mode: Normal programs executed
- Some instructions/ops prohibited in user mode:
  - Example: cannot modify page tables in user mode
    - » Attempt to modify ⇒ Exception generated
- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions



UNIX System Structure



“In conclusion…”

- Operating systems provide a virtual machine abstraction to handle diverse hardware
- Operating systems coordinate resources and protect users from each other
- Operating systems simplify application development by providing standard services
- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery

- CS162 combines things from many other areas of computer science –
  - Languages, data structures, hardware, and algorithms