CSC 112: Computer Operating Systems
Lecture 1

What is an Operating System?

Department of Computer Science,
Hofstra University

Acknowledgement: Lecture slides based on UC Berkeley CS 162: Operating Systems and System Programming

Syllabus

- Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
  - Not strictly required
  - PPT files contain all relevant materials
- Topics covered
  - OS Concepts: How to Navigate as a Systems Programmer!
    - » Process, I/O, Networks and Virtual Machines
  - 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
    - » Protocols, N-Tiers, RPC, NFS, DHTs, Consistency, Scalability, multicast
  - Reliability & Security
    - » Fault tolerance, protection, security
  - Cloud Infrastructure

Group Projects

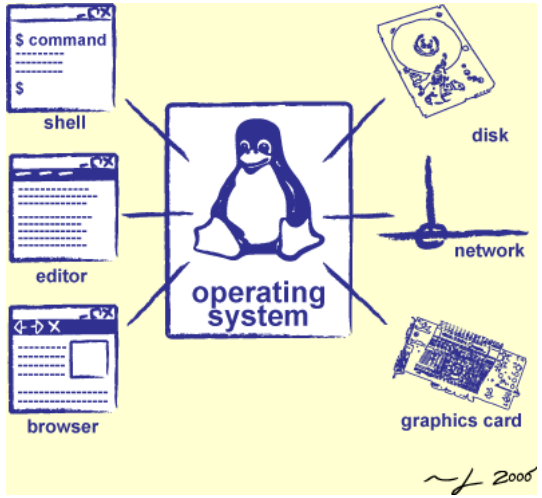
- Project teams have 1-4 members
- Everyone should do work and have clear responsibilities

Grading (Tentative breakdown)

- TODO
- 36% three midterms (12% each)
  - Thursday, 2/17, TBA, time set after survey
  - Thursday, 3/17, TBA, time set after survey
  - Thursday, 4/28, TBA, time set after survey
  - These will be IN-PERSON!
- 36% projects
- 18% homework
- 10% participation (Sections, Lecture, …)
- *No final exam*
- Projects
  - Initial design document, Design review, Code, Final design
  - Submission via *git push* triggers autograder

Goals for Today

- What is an Operating System?
  - And – what is it not?
- What makes Operating Systems so exciting?
- Oh, and "How does this class operate?"



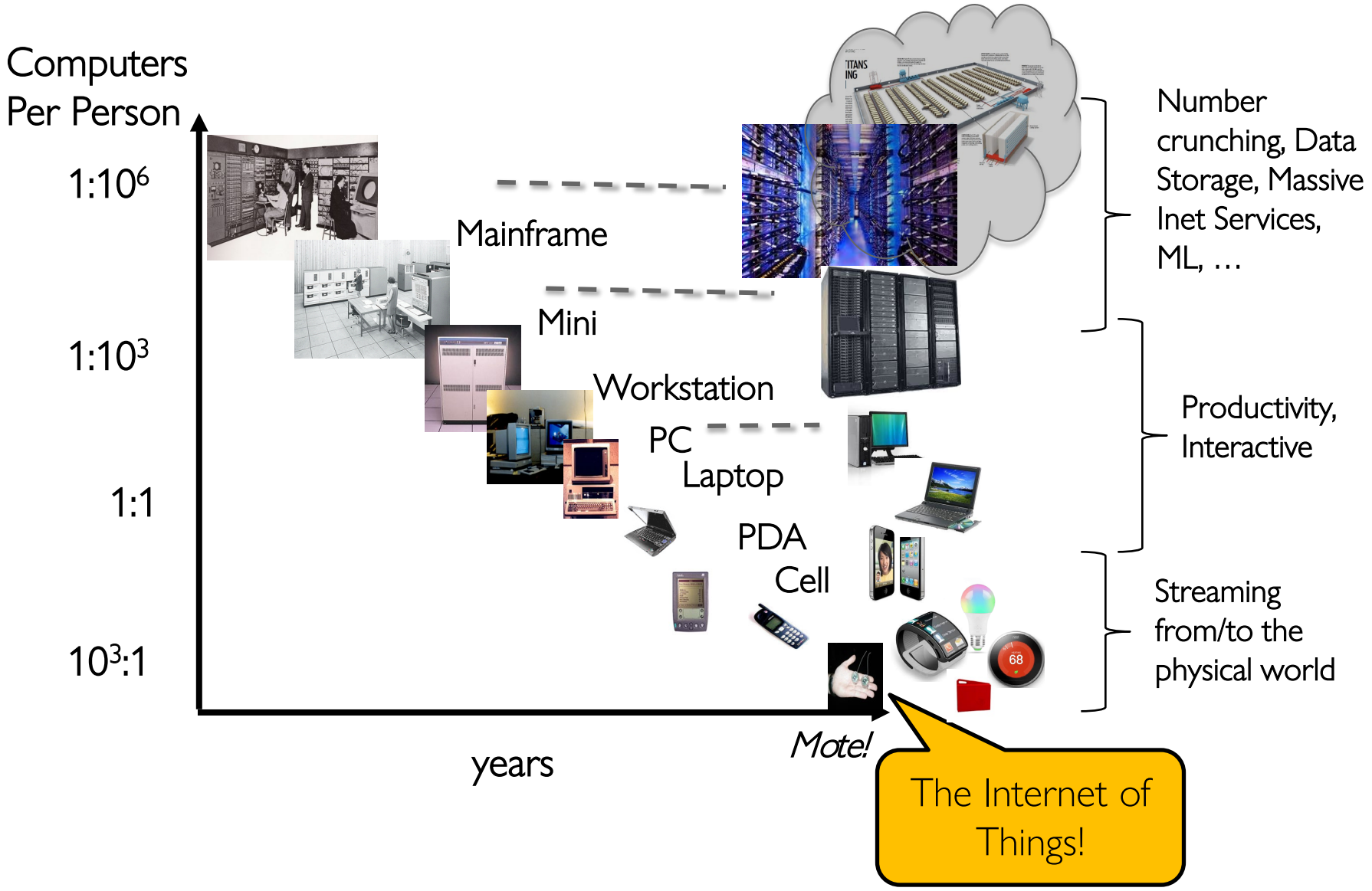
Interactive is important!

Ask Questions!

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

Across Incredible Diversity

Bell's Law: New computer class every 10 years



And Range of Timescales

Jeff Dean: "Numbers Everyone Should Know"

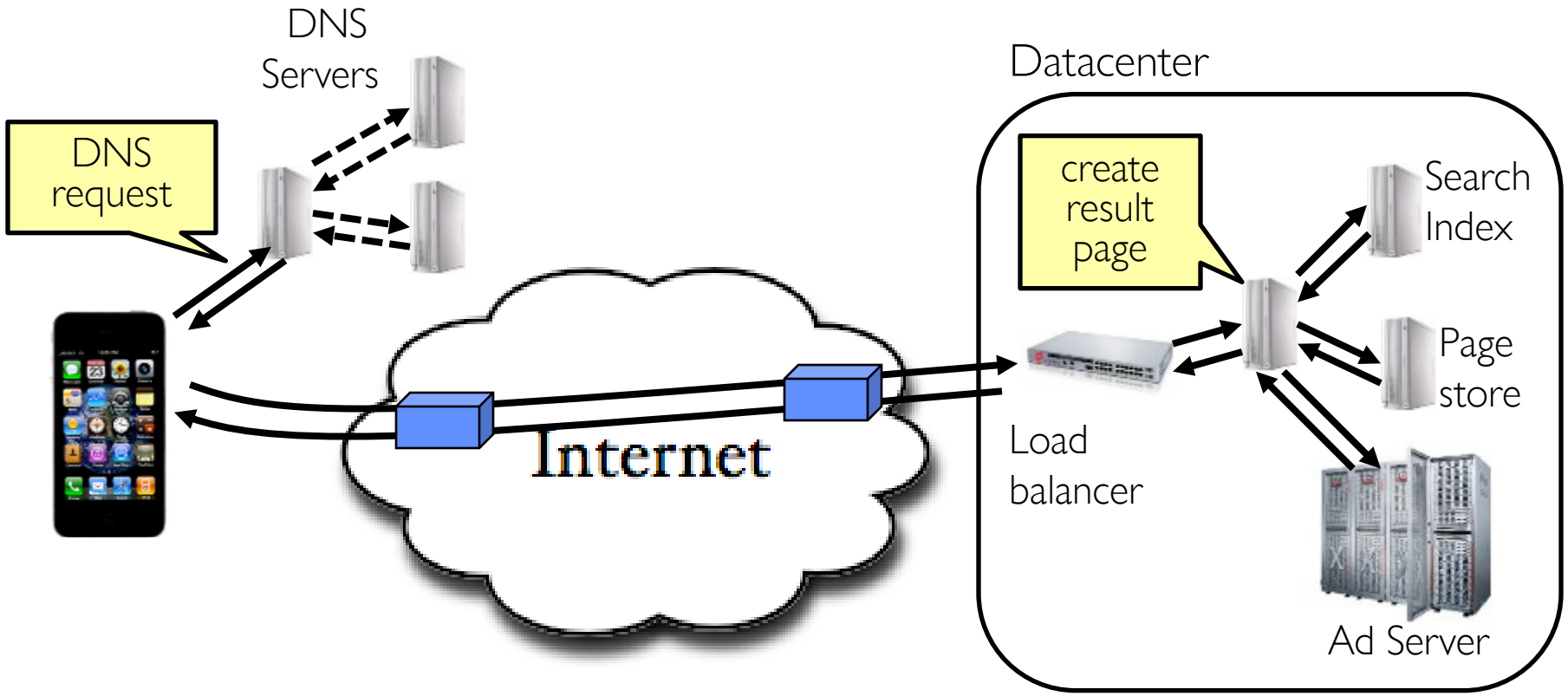
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Operating Systems are at the Heart of it All!

- Make the incredible advance in the underlying technology available to a rapidly evolving body of applications
  - Provide **consistent abstractions** to applications, even on different hardware
  - Manage **sharing of resources** among multiple applications

- The key building blocks:
  - Processes
  - Threads, Concurrency, Scheduling, Coordination
  - Address Spaces
  - Protection, Isolation, Sharing, Security
  - Communication, Protocols
  - 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, …

But: What is an operating system?

What does an Operating System *do?*

- 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??

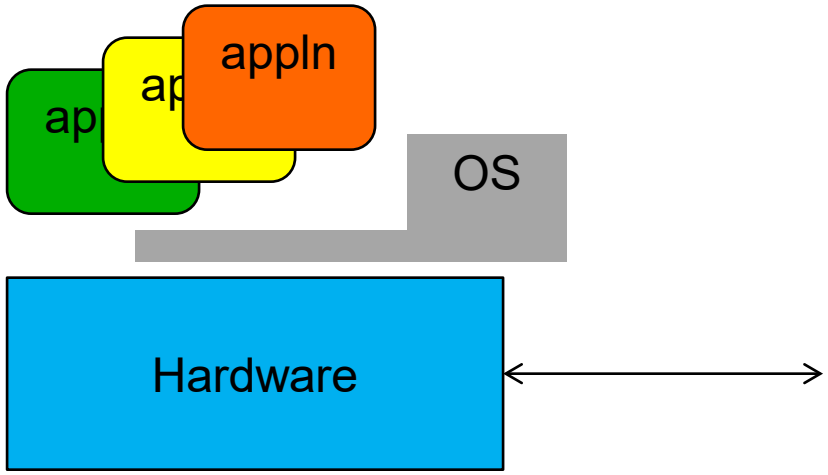
Definition of an Operating System

- 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

One Definition of an Operating System

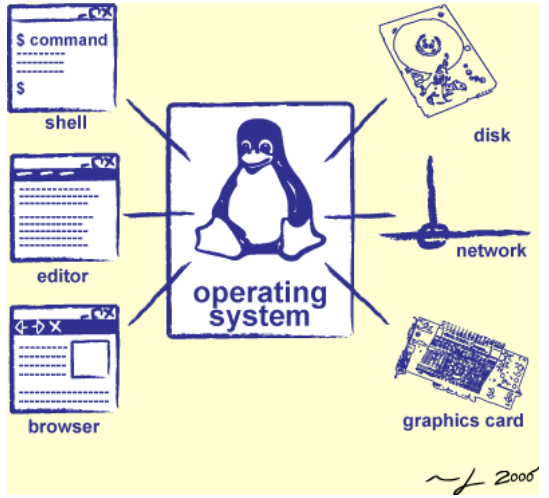
- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication



Operating System

Switchboard Operator

Computer Operators



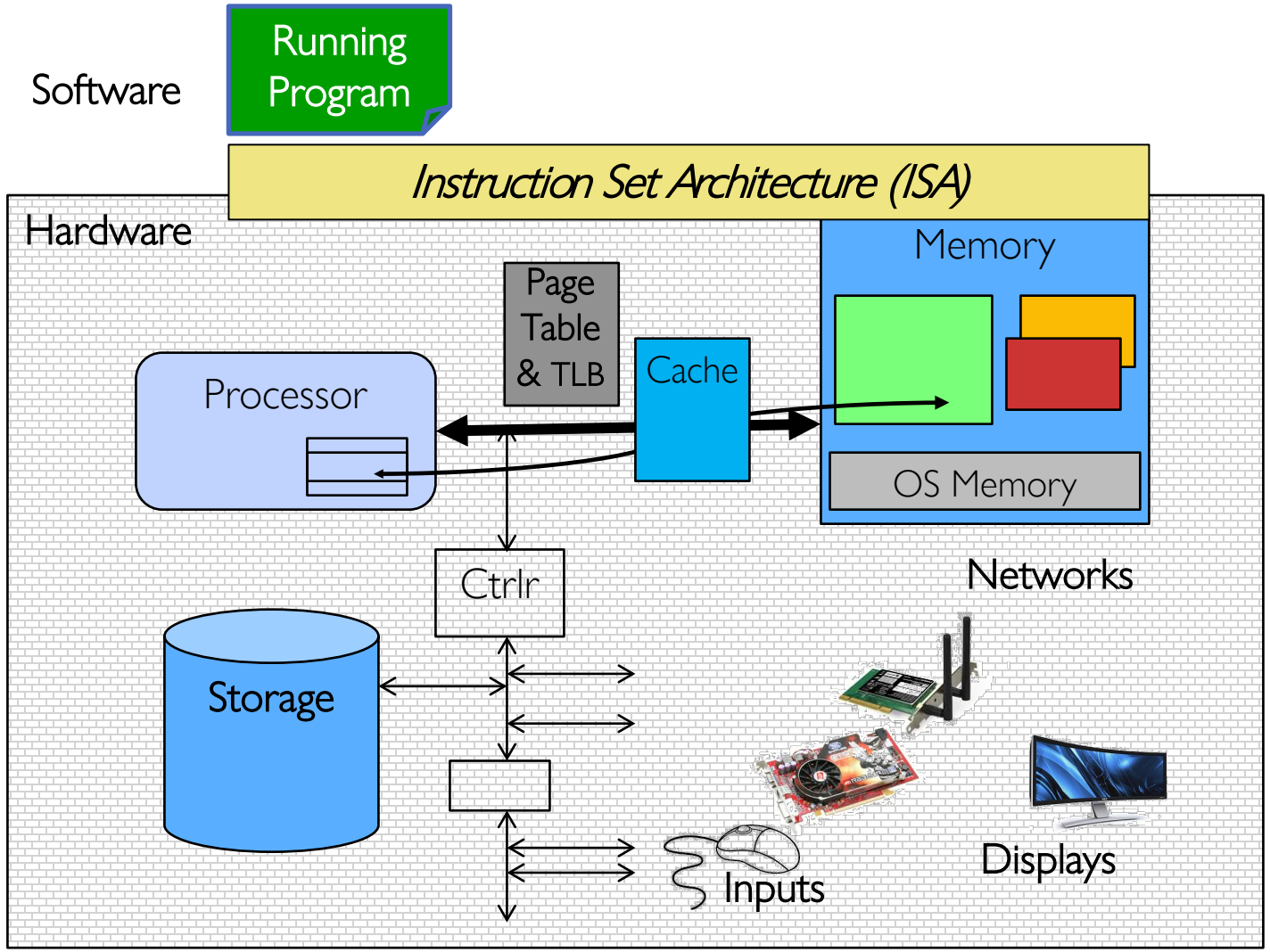
Operating System

What makes something a *system?*

- Multiple interrelated parts
  - Each potentially interacts with the others
- Robustness requires an *engineering mindset*
  - Meticulous error handling, defending against malicious careless users
  - Treating the computer as a concrete machine, with all of its limitations and possible failure cases

System programming is an important part of this class!

Hardware/Software Interface

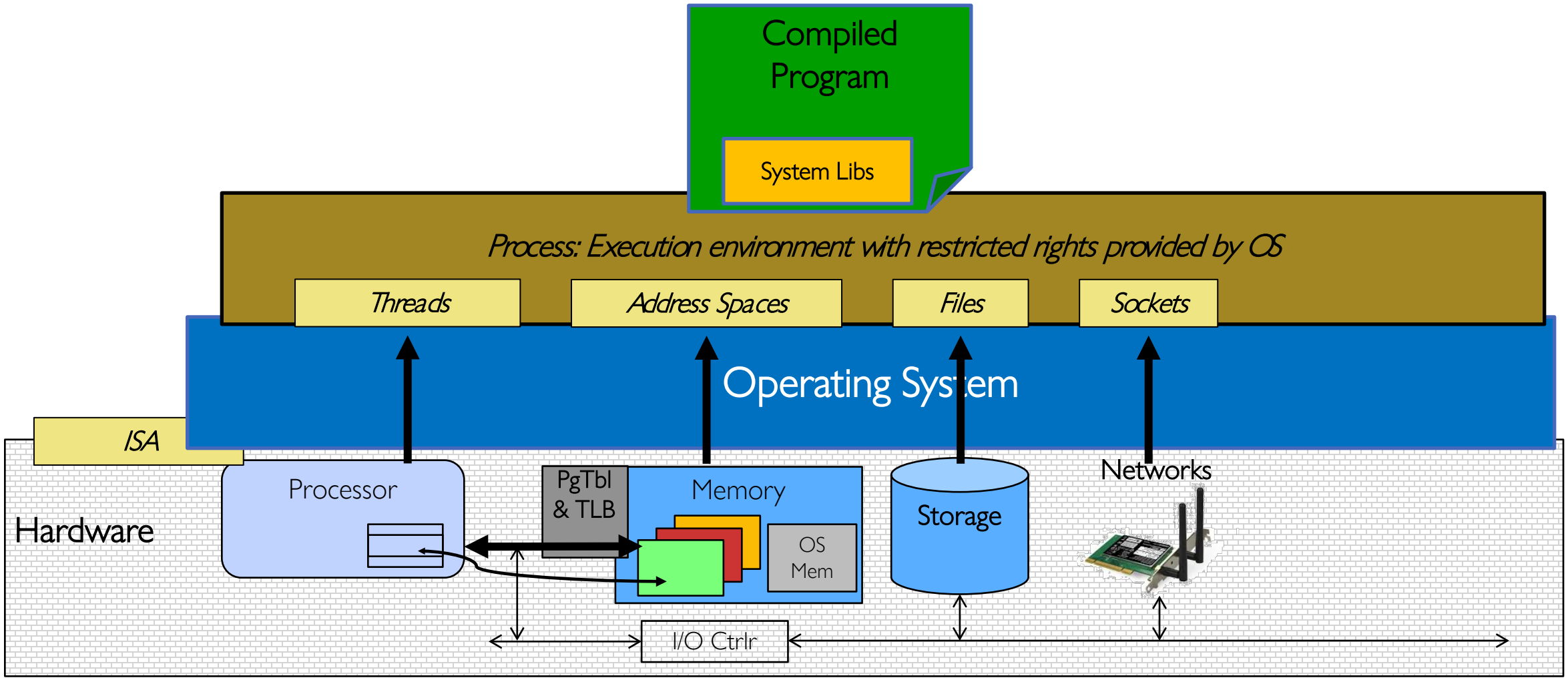


The OS *abstracts* these hardware details from the application

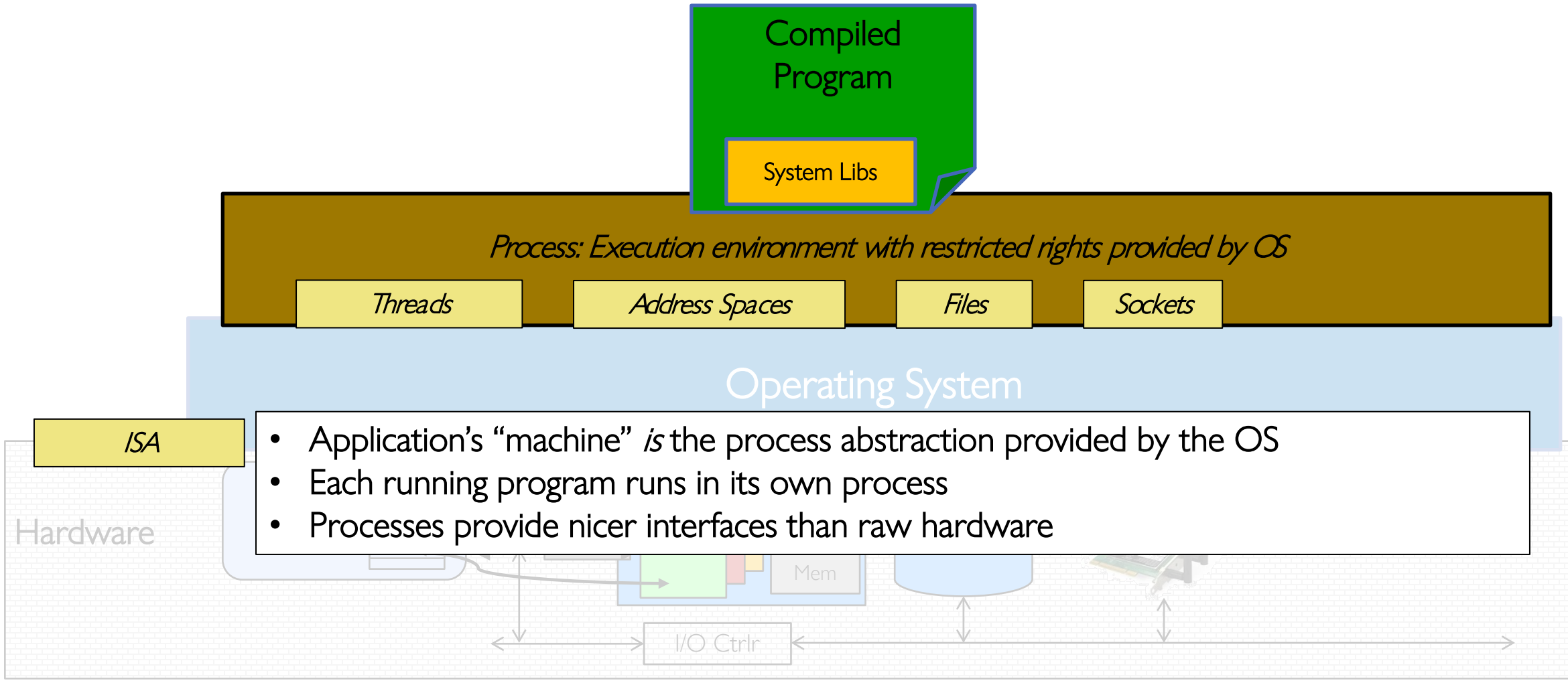
What is an Operating System?

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

OS Basics: Virtualizing the Machine

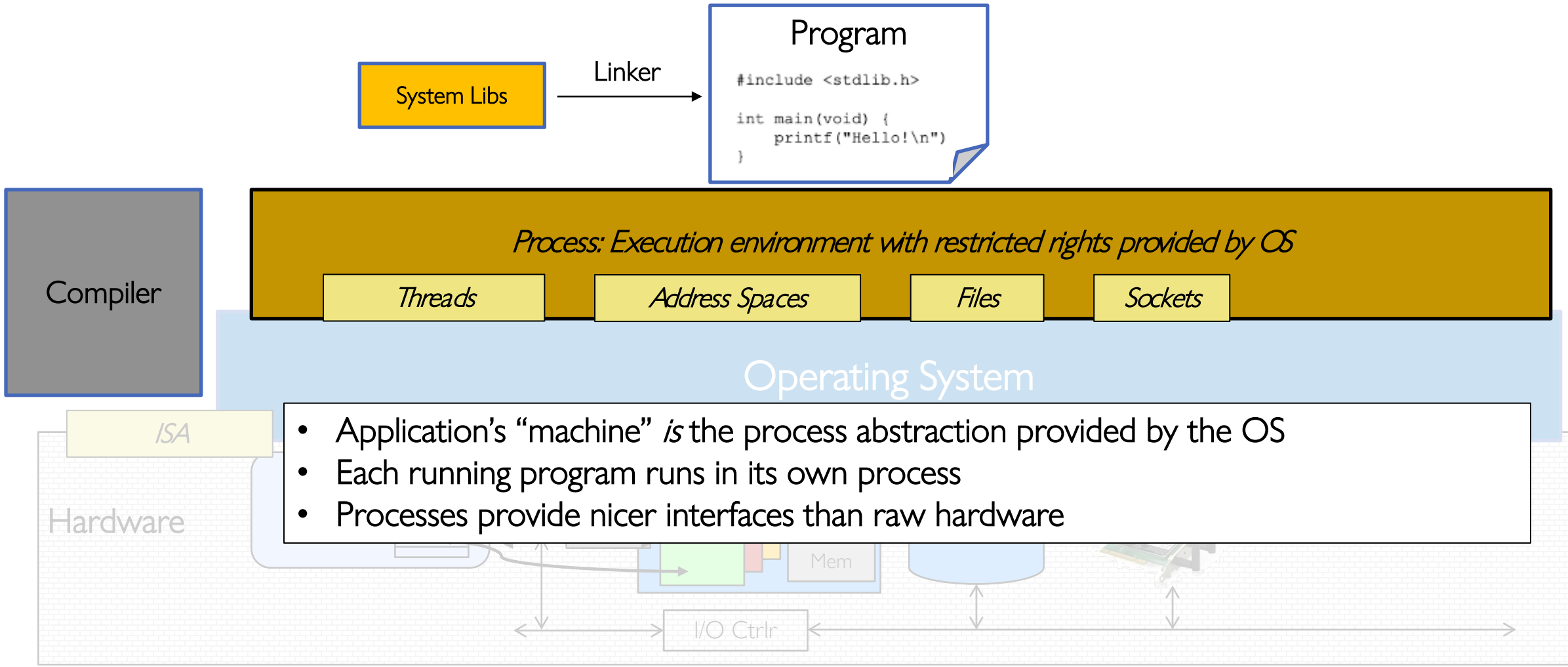


Compiled Program's View of the World



- Application's "machine" *is* the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware

System Programmer's View of the World



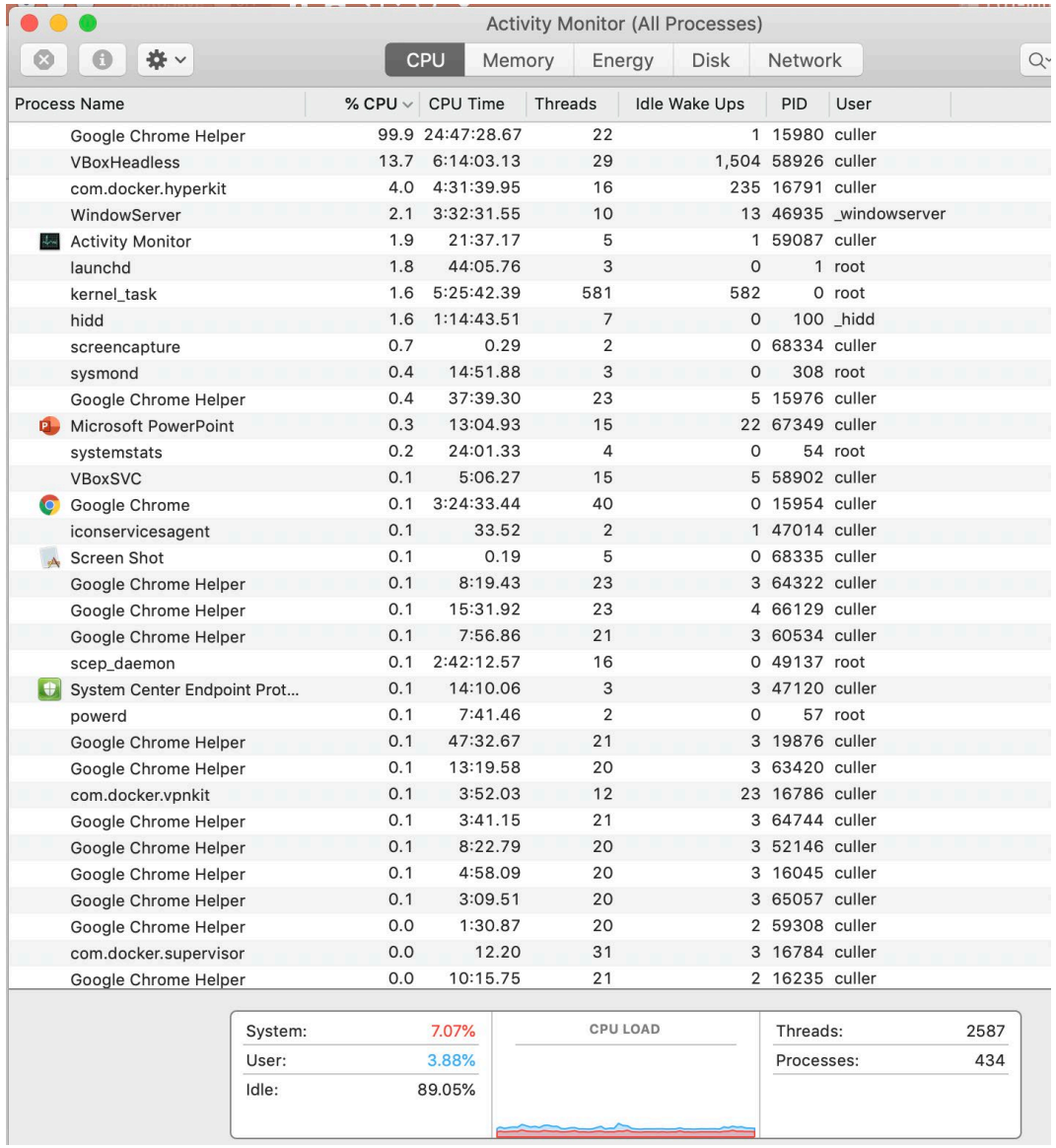
- Application's "machine" *is* the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware

What's in a Process?

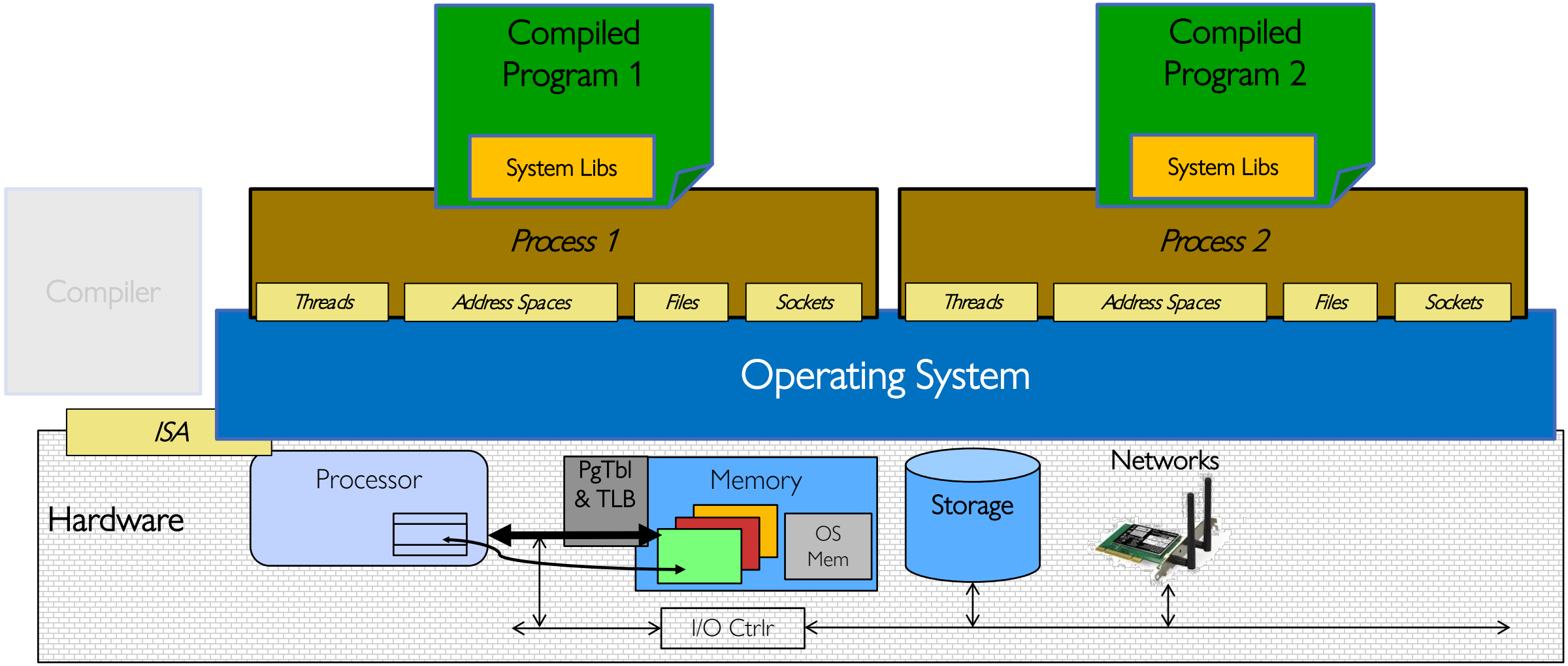
A process consists of:

- Address Space
- One or more threads of control executing in that address space
- Additional system state associated with it
  - Open files
  - Open sockets (network connections)
  - …

For Example…

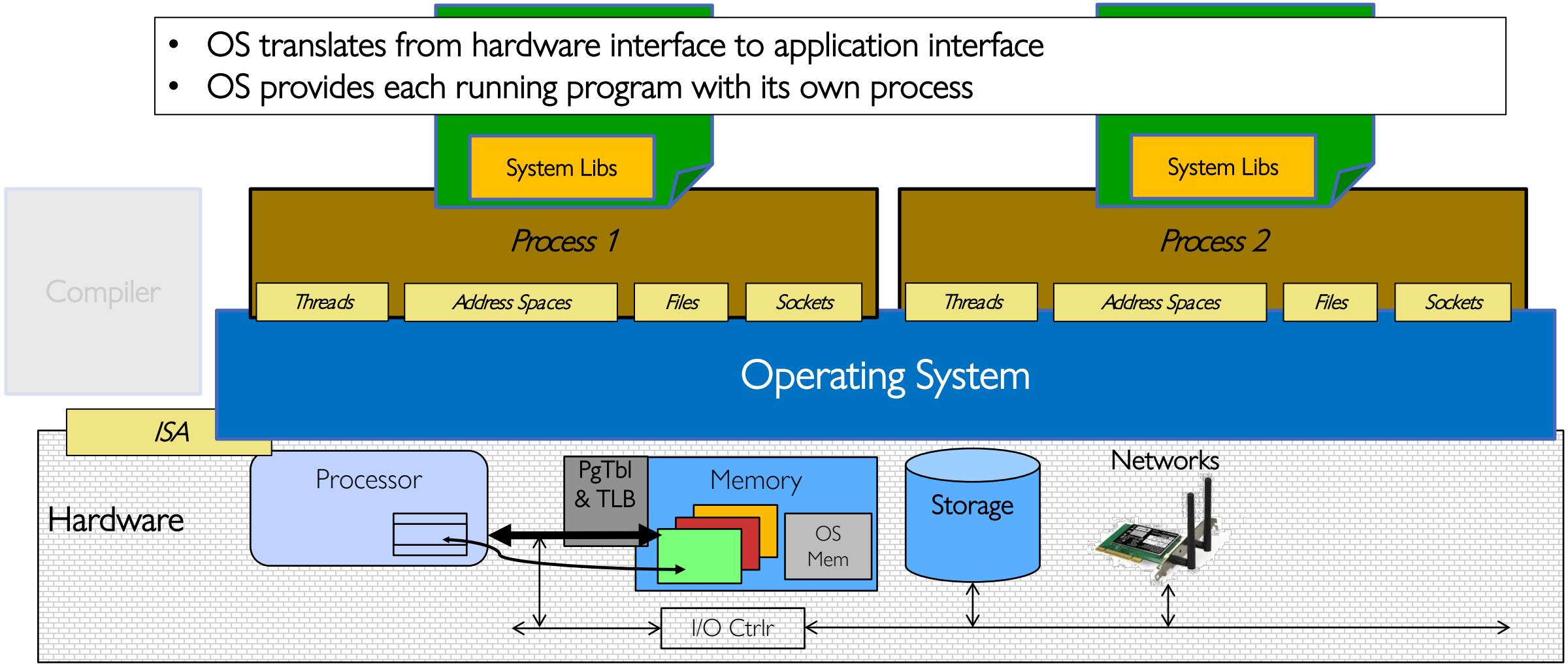

Activity Monitor (All Processes)
CPU
Memory
Energy
Disk
Network
Process Name % CPU CPU Time Threads Idle Wake Ups PID User
Google Chrome Helper 99.9 24:47:28.67 22 1 15980 culler
VBoxHeadless 13.7 6:14:03.13 29 1,504 58926 culler
com.docker.hyperkit 4.0 4:31:39.95 16 235 16791 culler
WindowServer 2.1 3:32:31.55 10 13 46935 _windowserver
Activity Monitor 1.9 21:37.17 5 1 59087 culler
launchd 1.8 44:05.76 3 0 1 root
kernel_task 1.6 5:25:42.39 581 582 0 root
hidd 1.6 1:14:43.51 7 0 100 _hidd
screencapture 0.7 0.29 2 0 68334 culler
sysmond 0.4 14:51.88 3 0 308 root
Google Chrome Helper 0.4 37:39.30 23 5 15976 culler
Microsoft PowerPoint 0.3 13:04.93 15 22 67349 culler
systemstats 0.2 24:01.33 4 0 54 root
VBoxSVC 0.1 5:06.27 15 5 58902 culler
Google Chrome 0.1 3:24:33.44 40 0 15954 culler
iconservicesagent 0.1 33.52 2 1 47014 culler
Screen Shot 0.1 0.19 5 0 68335 culler
Google Chrome Helper 0.1 8:19.43 23 3 64322 culler
Google Chrome Helper 0.1 15:31.92 23 4 66129 culler
Google Chrome Helper 0.1 7:56.86 21 3 60534 culler
scep_daemon 0.1 2:42:12.57 16 0 49137 root
System Center Endpoint Prot... 0.1 14:10.06 3 3 47120 culler
powerd 0.1 7:41.46 2 0 57 root
Google Chrome Helper 0.1 47:32.67 21 3 19876 culler
Google Chrome Helper 0.1 13:19.58 20 3 63420 culler
com.docker.vpnkit 0.1 3:52.03 12 23 16786 culler
Google Chrome Helper 0.1 3:41.15 21 3 64744 culler
Google Chrome Helper 0.1 8:22.79 20 3 52146 culler
Google Chrome Helper 0.1 4:58.09 20 3 16045 culler
Google Chrome Helper 0.1 3:09.51 20 3 65057 culler
Google Chrome Helper 0.0 1:30.87 20 2 59308 culler
com.docker.supervisor 0.0 12.20 31 3 16784 culler
Google Chrome Helper 0.0 10:15.75 21 2 16235 culler
System: 7.07%
User: 3.88%
Idle: 89.05%
CPU LOAD
Threads: 2587
Processes: 434

Operating System's View of the World



Operating System's View of the World

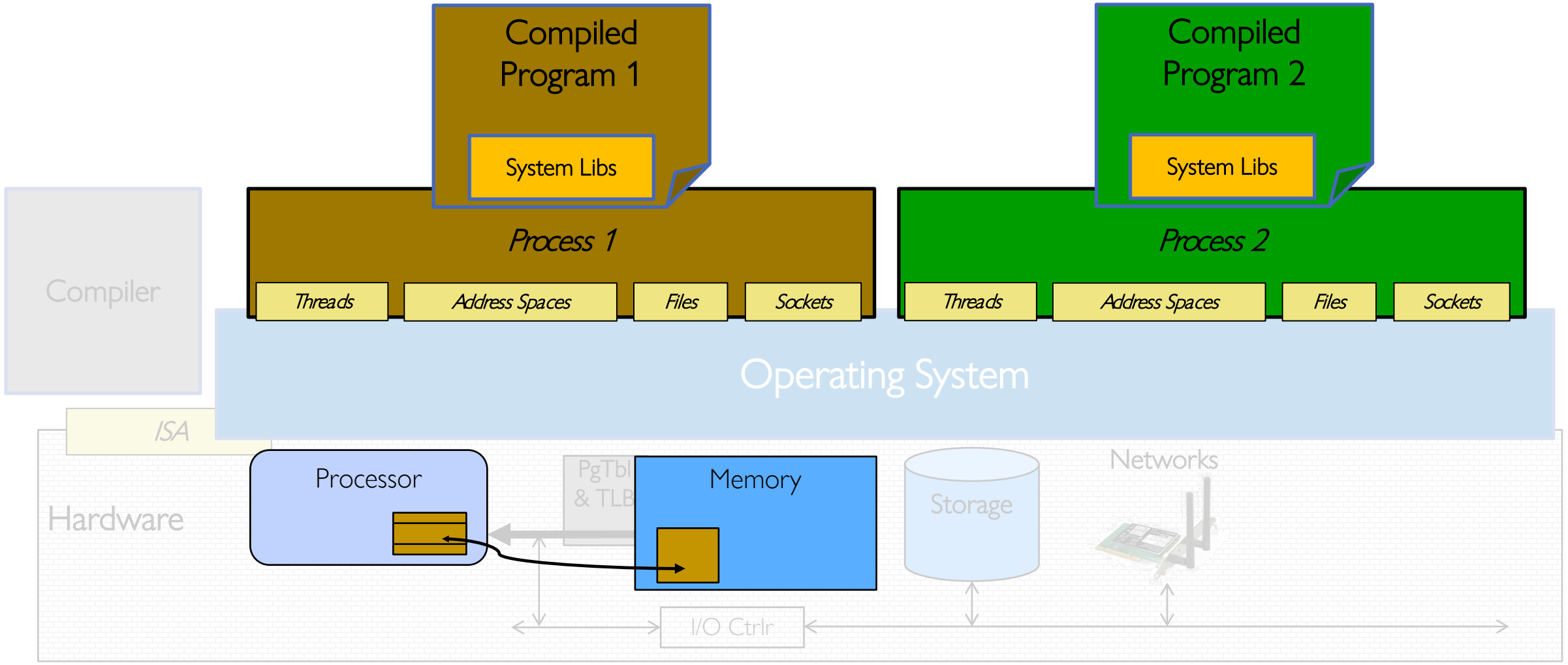
- OS translates from hardware interface to application interface
- OS provides each running program with its own process



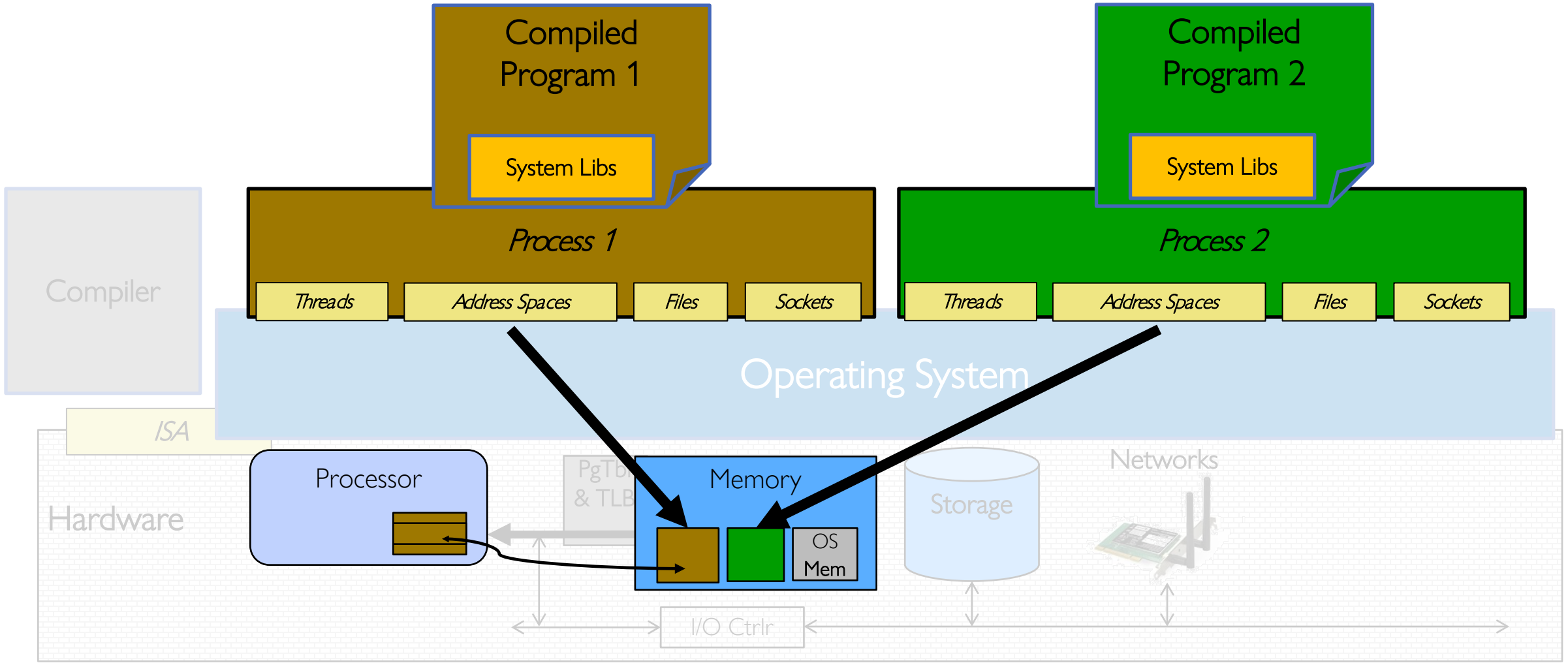
What is an Operating System?

- Referee
  - Manage protection, isolation, and sharing of resources
    - » Resource allocation and communication
- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization

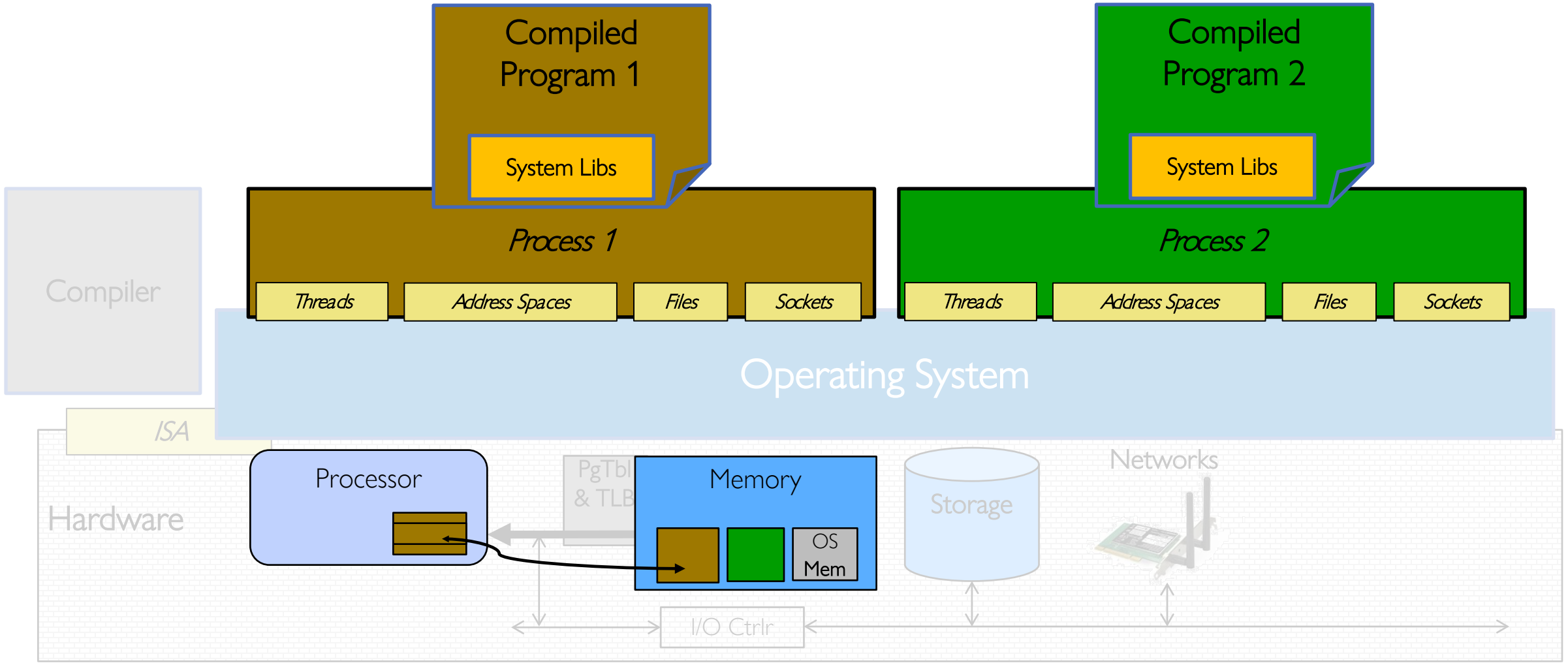
OS Basics: Running a Process



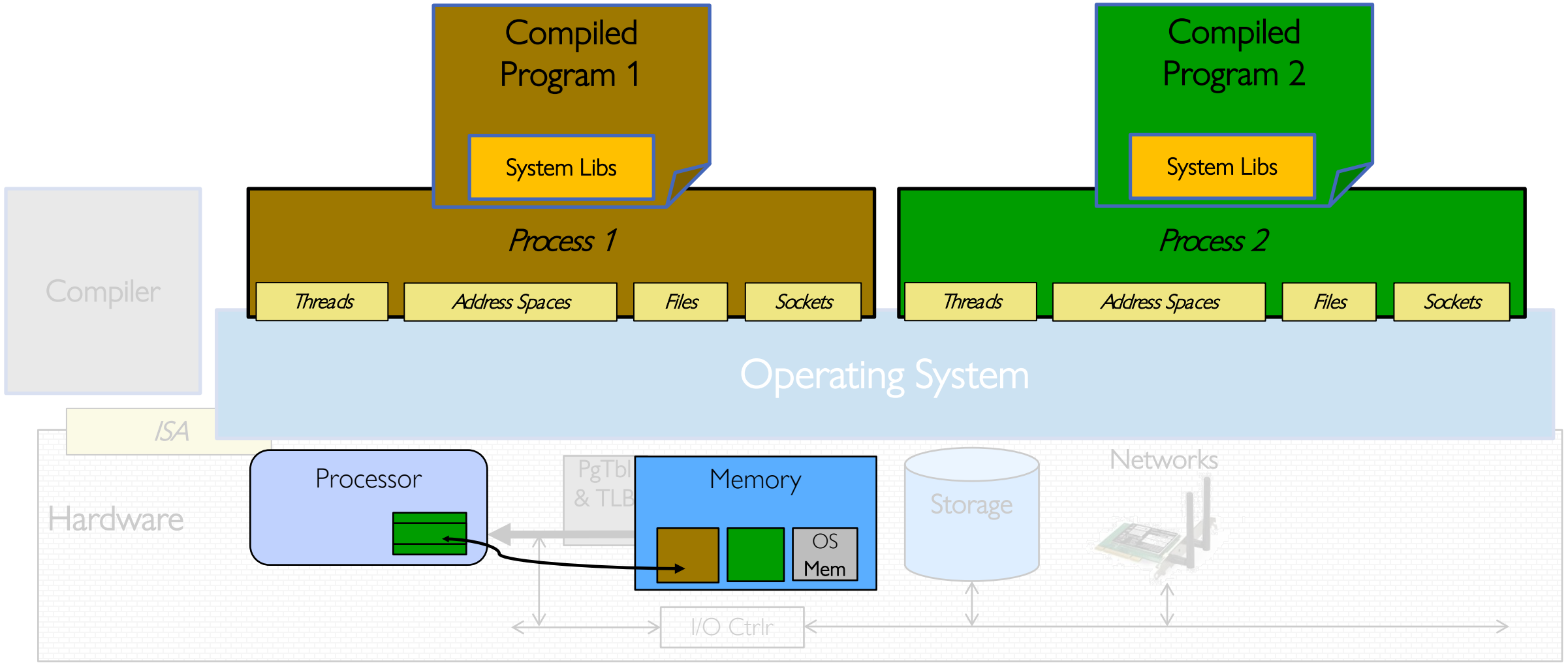
OS Basics: Switching Processes



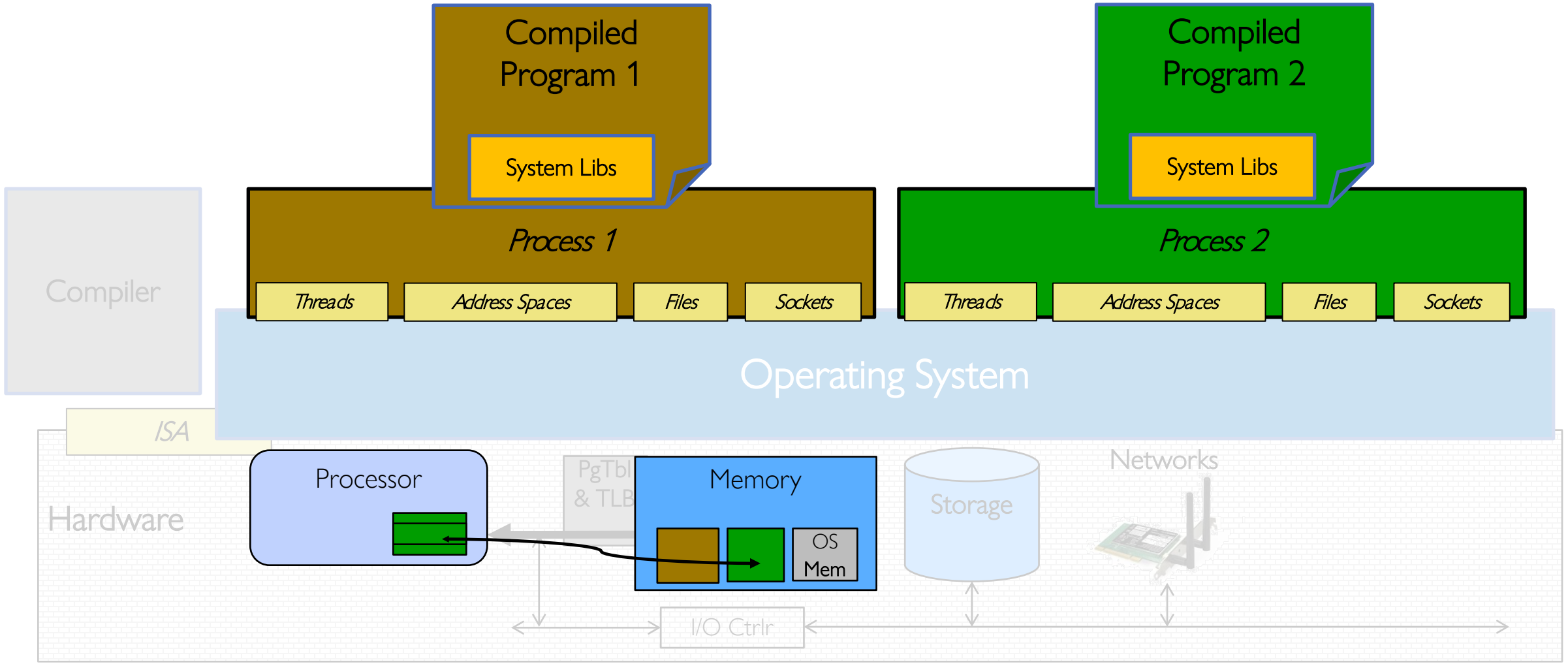
OS Basics: Switching Processes



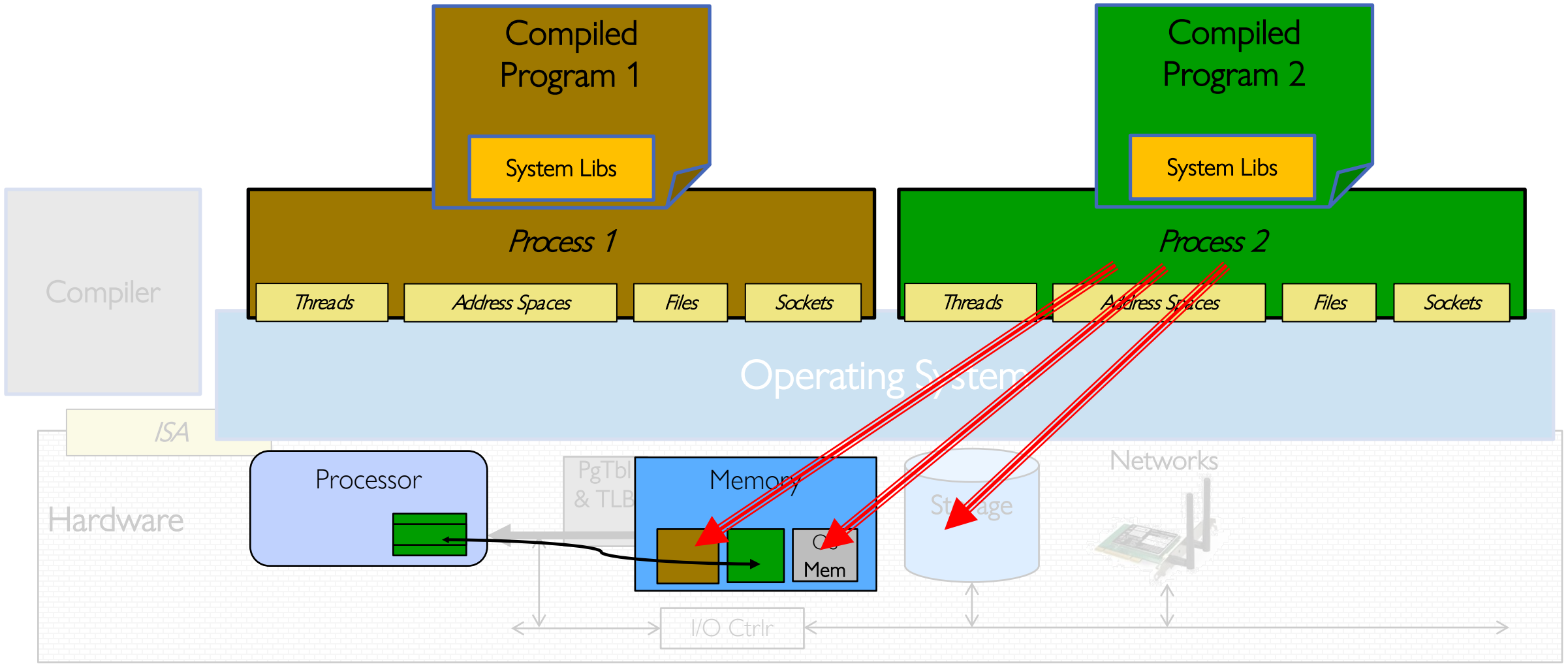
OS Basics: Switching Processes



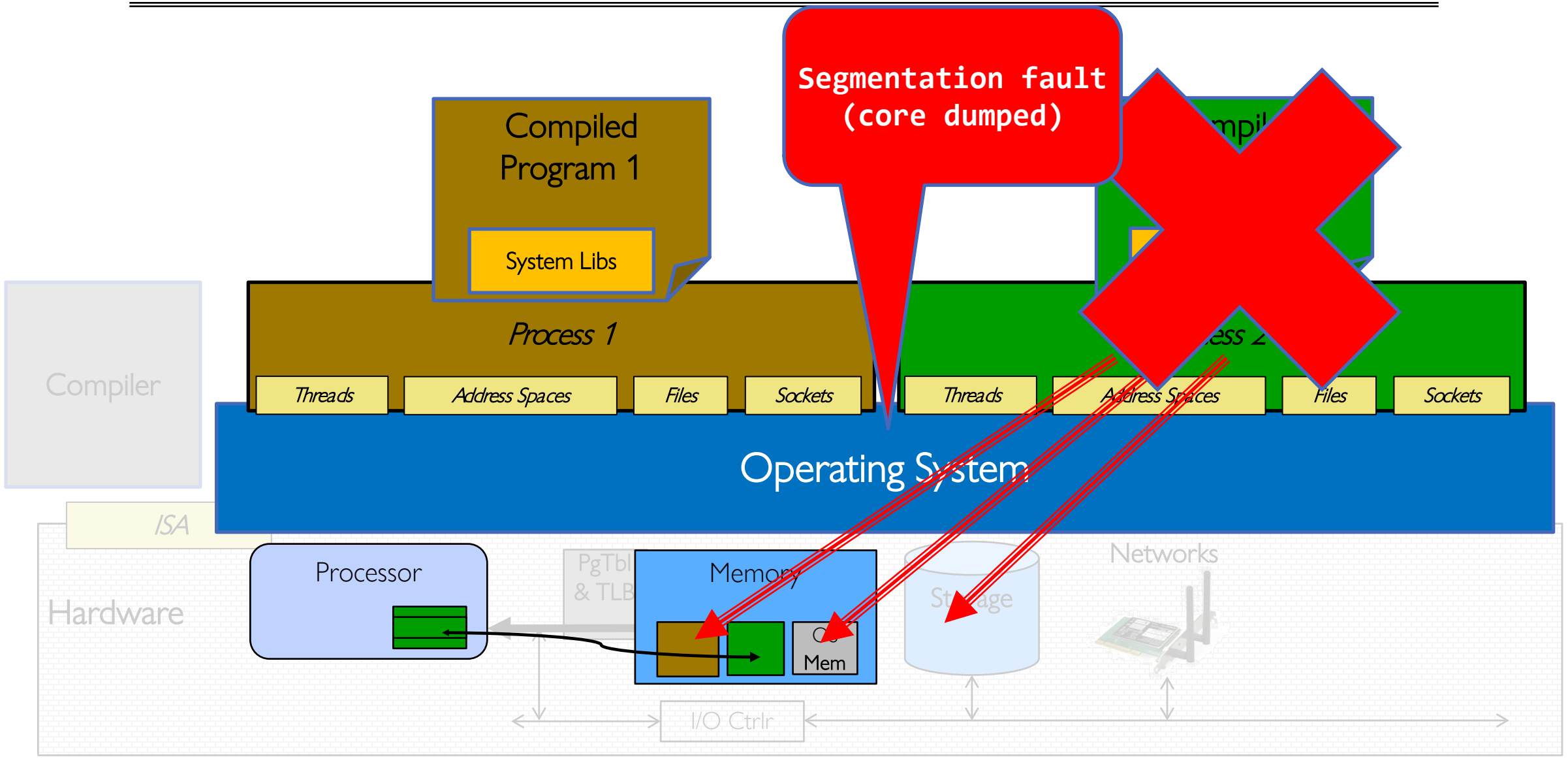
OS Basics: Switching Processes



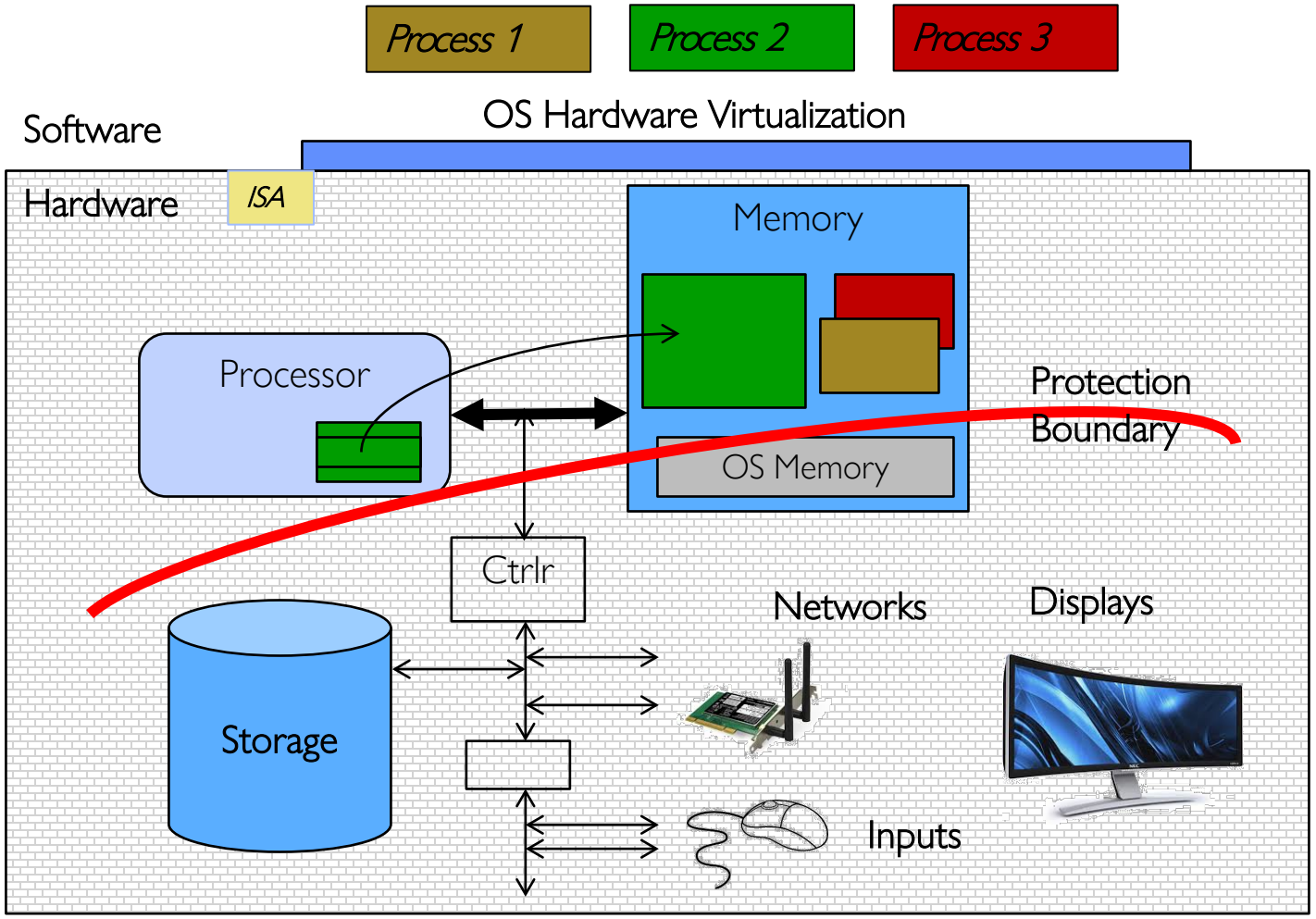
OS Basics: Protection



OS Basics: Protection

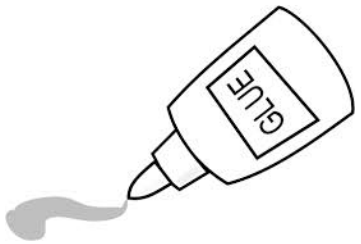


OS Basics: Protection



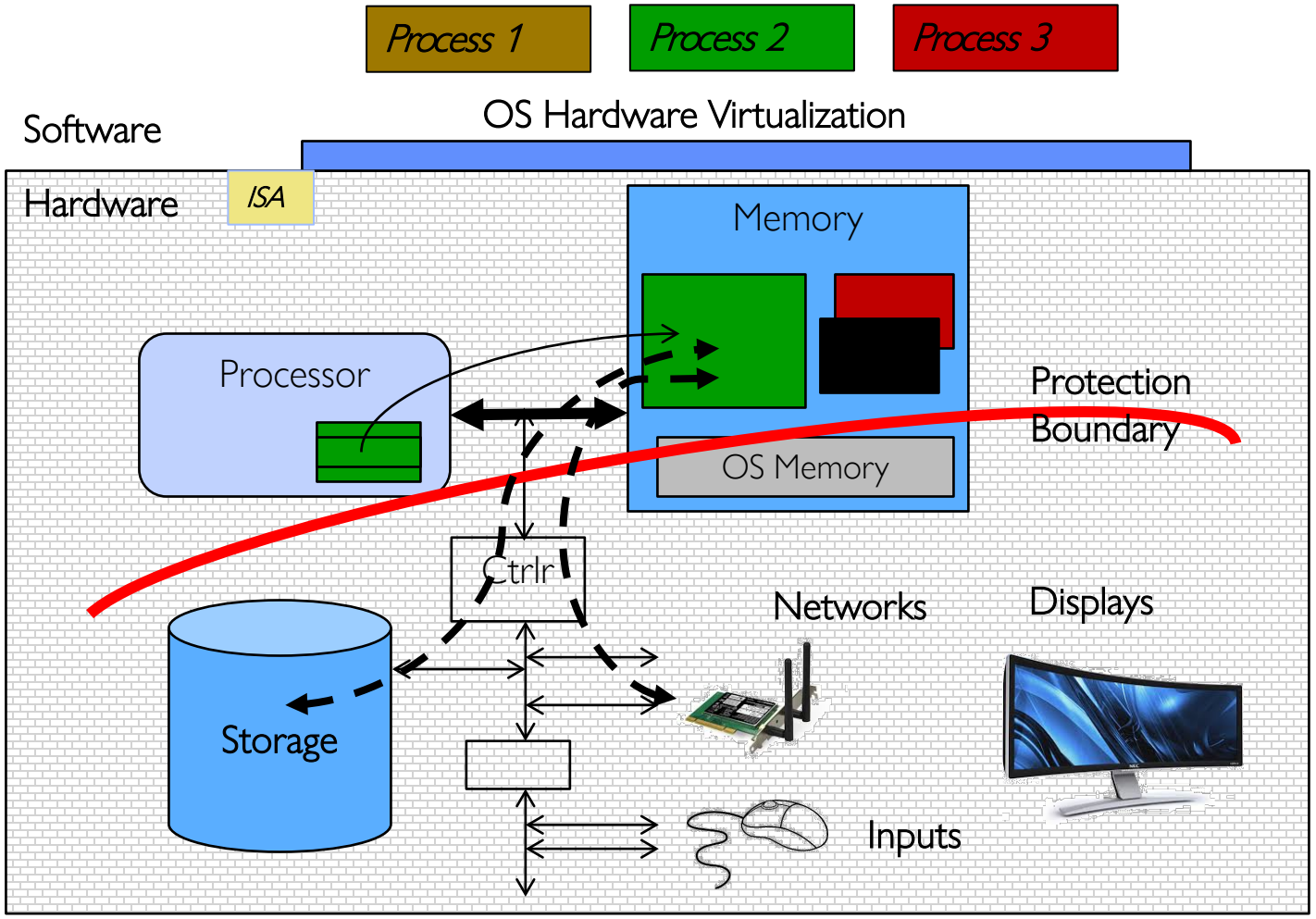
- OS *isolates* processes from each other
- OS *isolates* itself from other processes
- … even though they are actually running on the same hardware!

What is an Operating System?



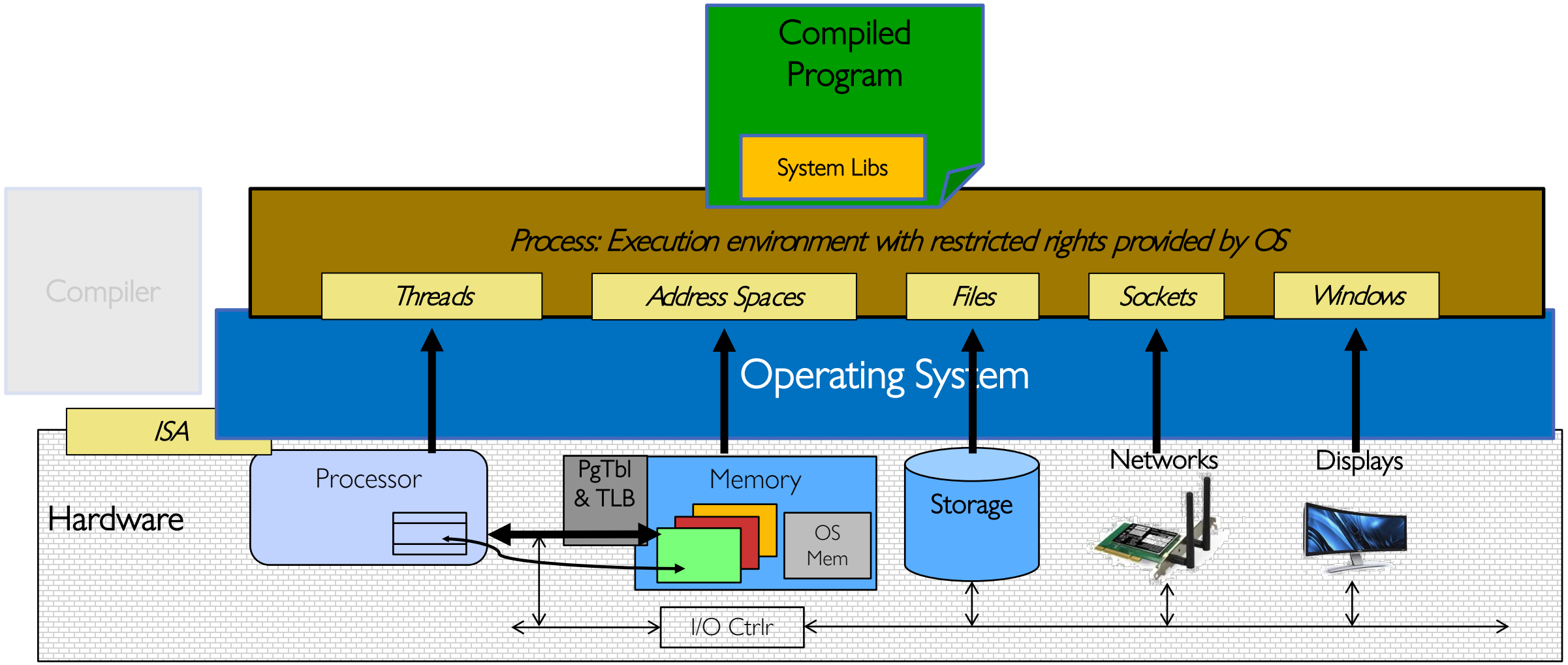
- Referee
  - Manage protection, isolation, and sharing of resources
    - » Resource allocation and communication
- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization
- Glue
  - Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

OS Basics: I/O

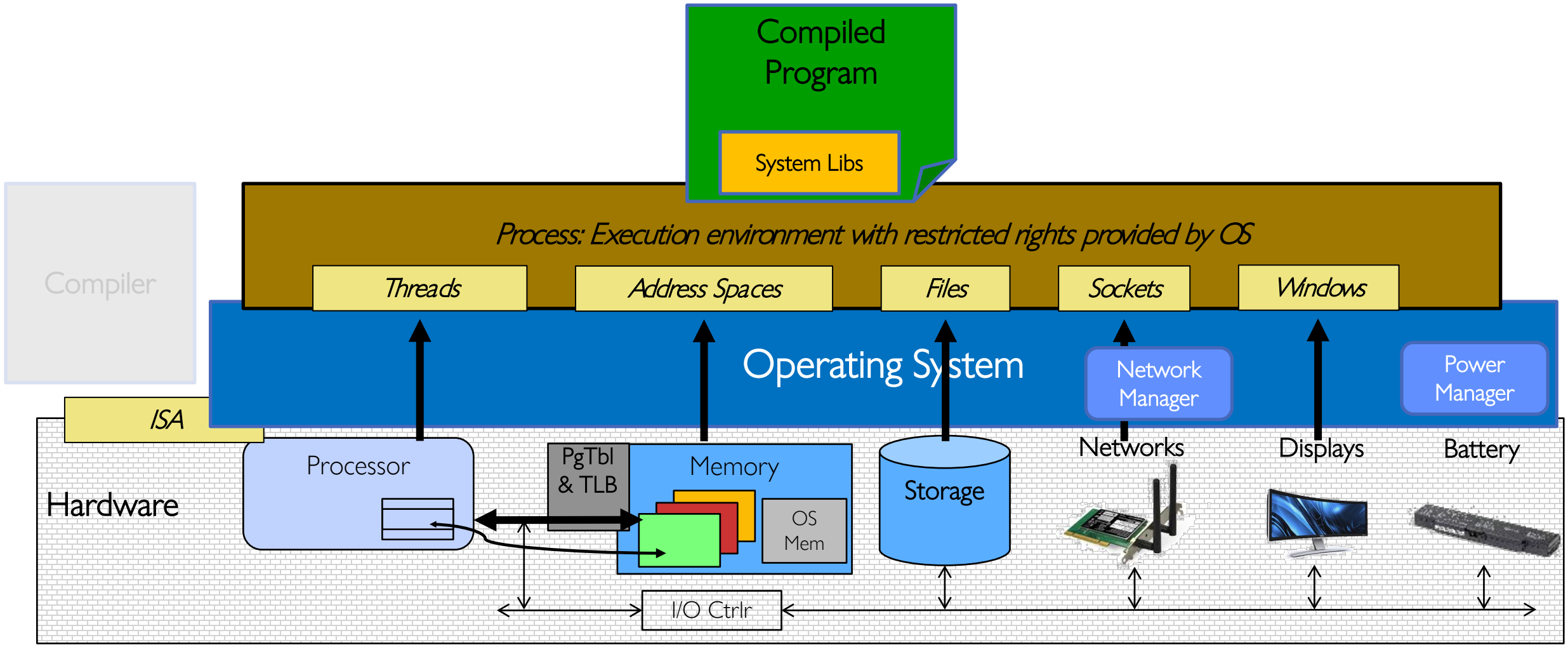


- OS provides common services in the form of I/O

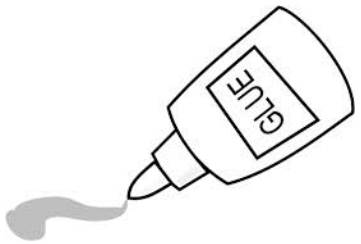
OS Basics: Look and Feel



OS Basics: Background Management



What is an Operating System?



- Referee
  - Manage protection, isolation, and sharing of resources
    - » Resource allocation and 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

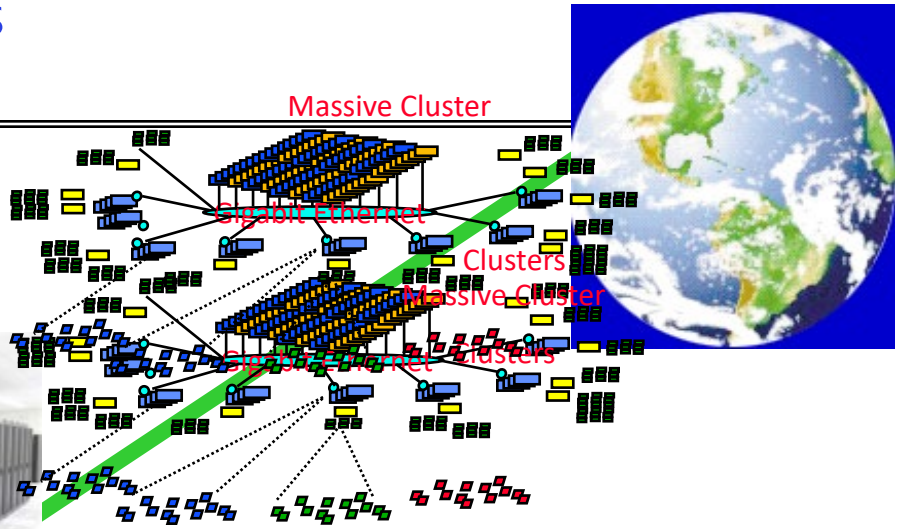
Why take CS162?

- 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.

What makes Operating Systems Exciting and Challenging?

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

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



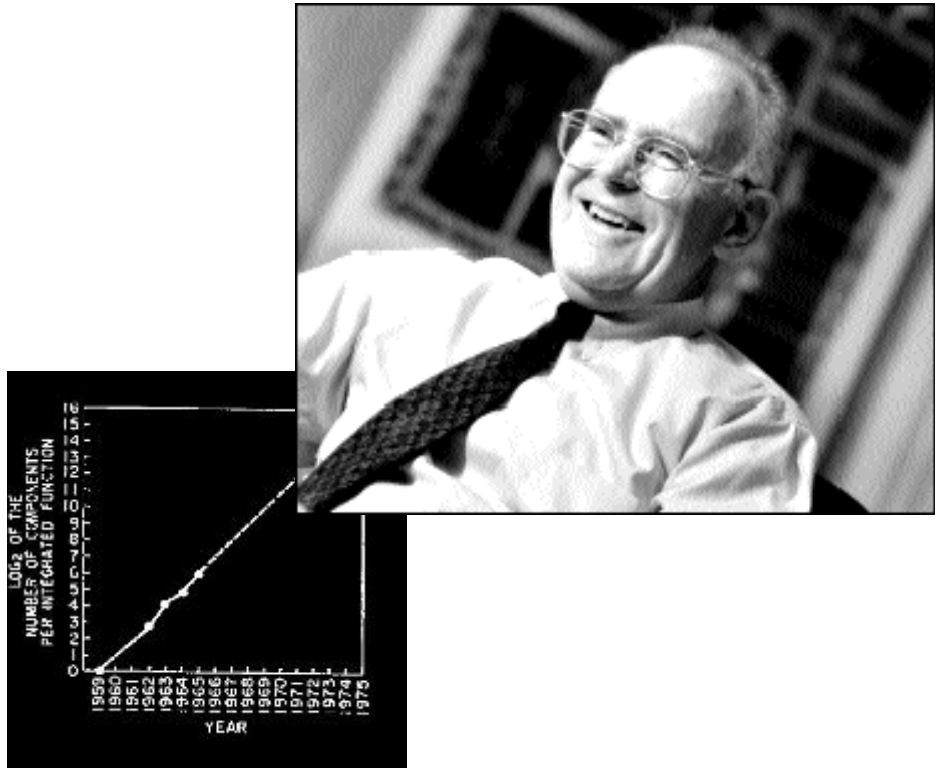
Scalable, Reliable, Secure Services

Internet Connectivity

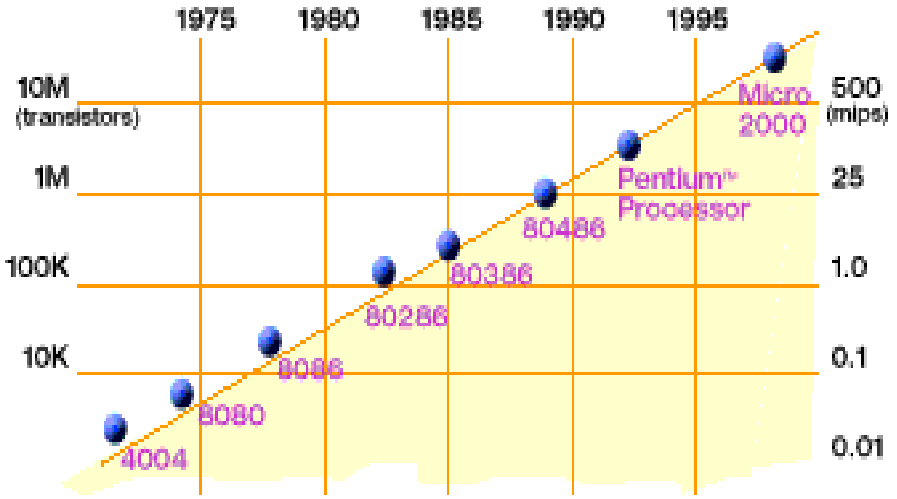
Databases
Information Collection
Remote Storage
Online Games
Commerce
…

MEMS for Sensor Nets

Technology Trends: 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

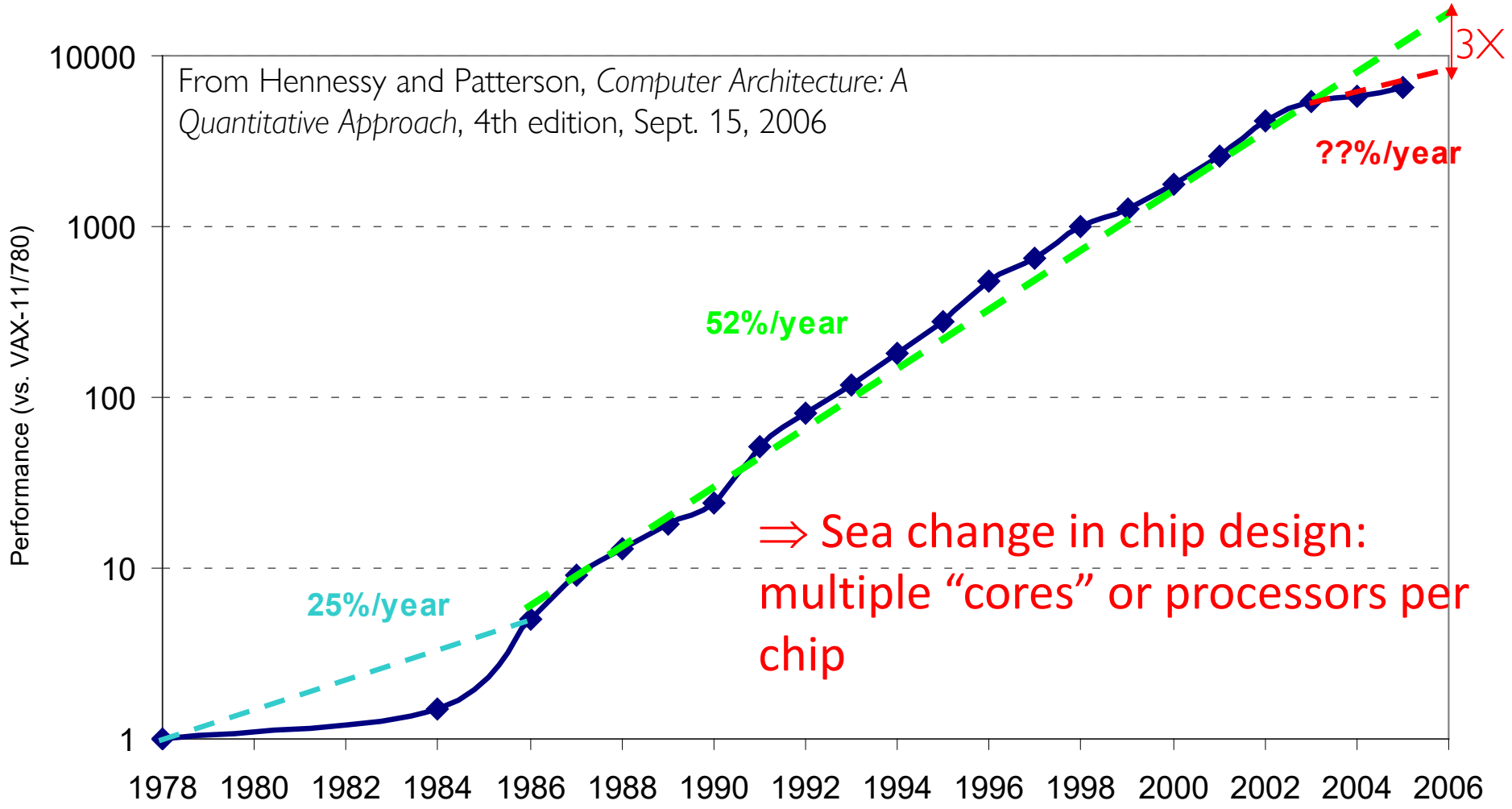


2X transistors/Chip Every 1.5 years

Called "Moore's Law"

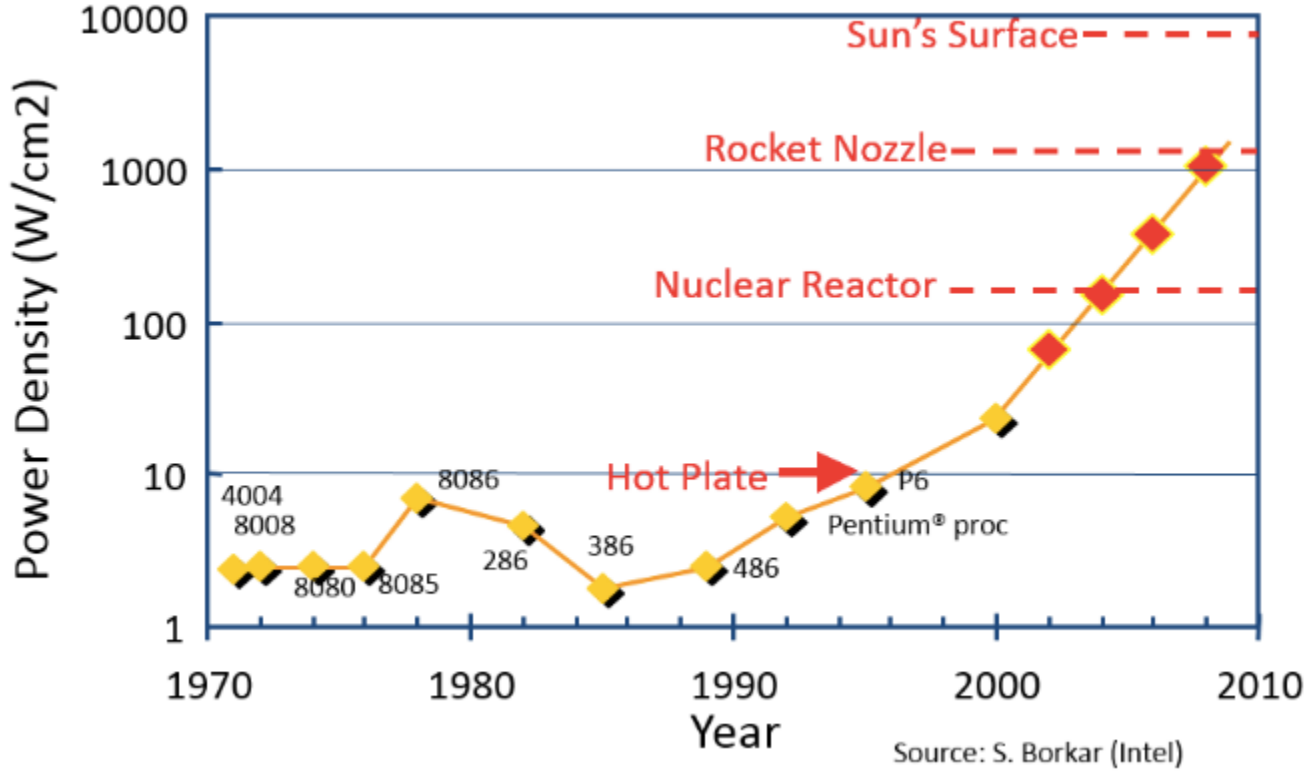
Microprocessors have become smaller, denser, and more powerful

Big Challenge: Slowdown in Joy's law of Performance



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

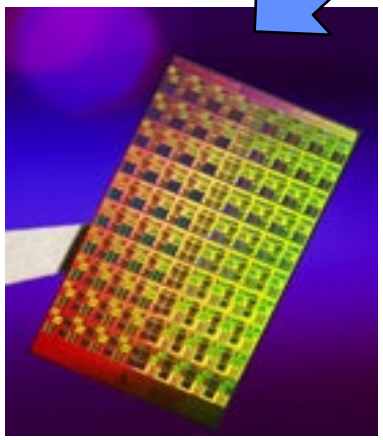
Another Challenge: Power Density

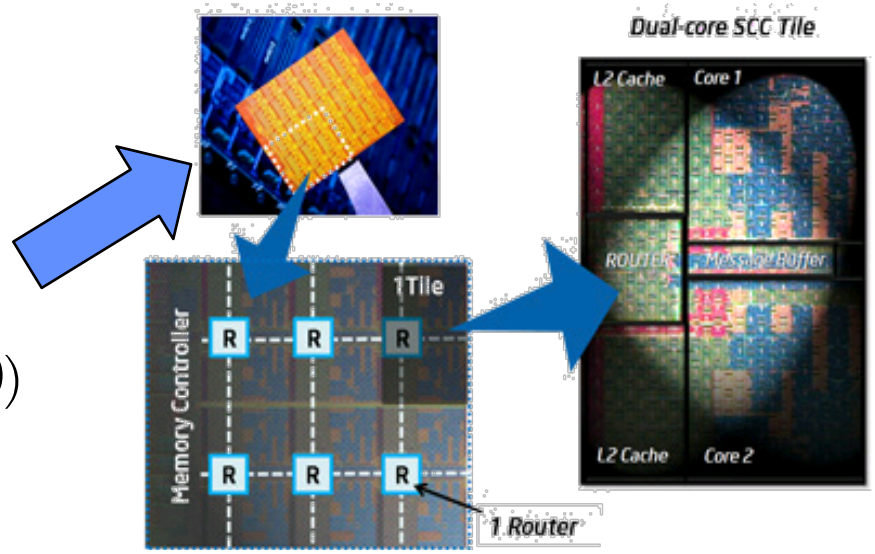


- Moore's law extrapolation
  - Potential power density reaching amazing levels!
- Flip side: battery life very important
  - Moore's law yielded more functionality at equivalent (or less) total energy consumption

ManyCore Chips: The future arrived in 2007

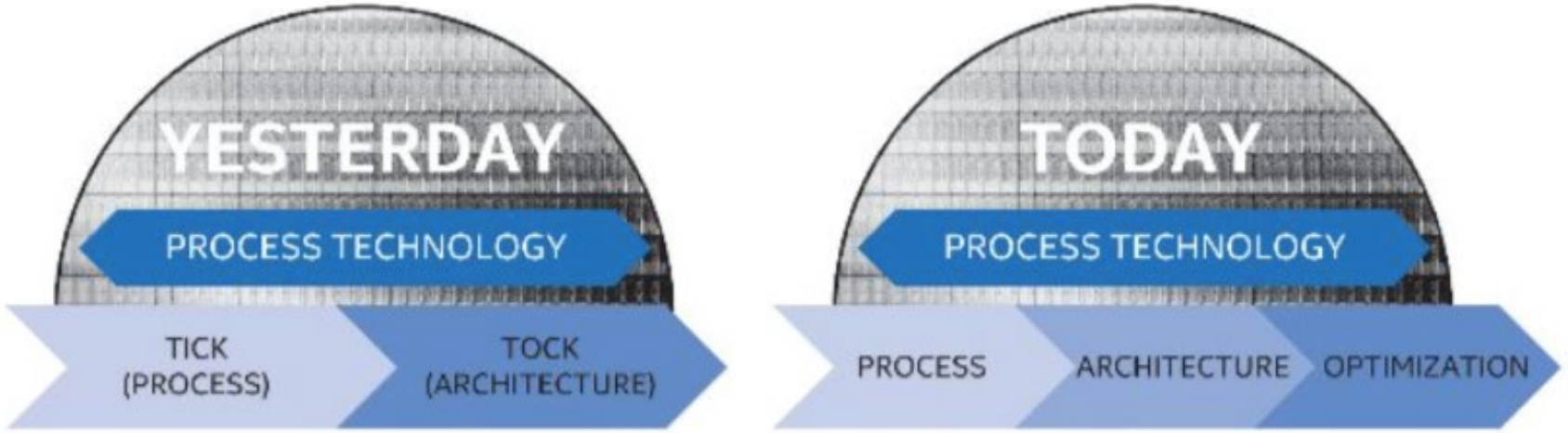
- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers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





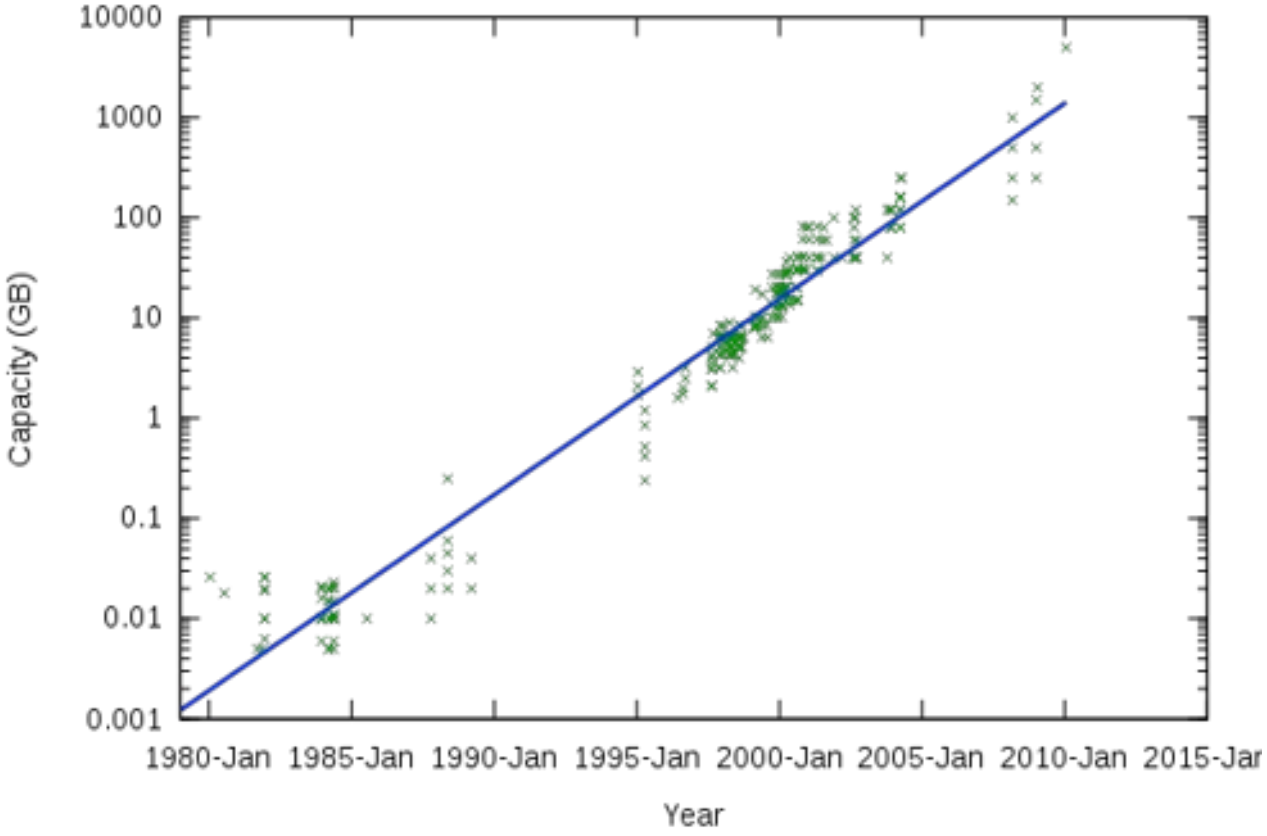
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels
- Amazon X1 instances (2016)
  - 128 virtual cores, 2 TB RAM

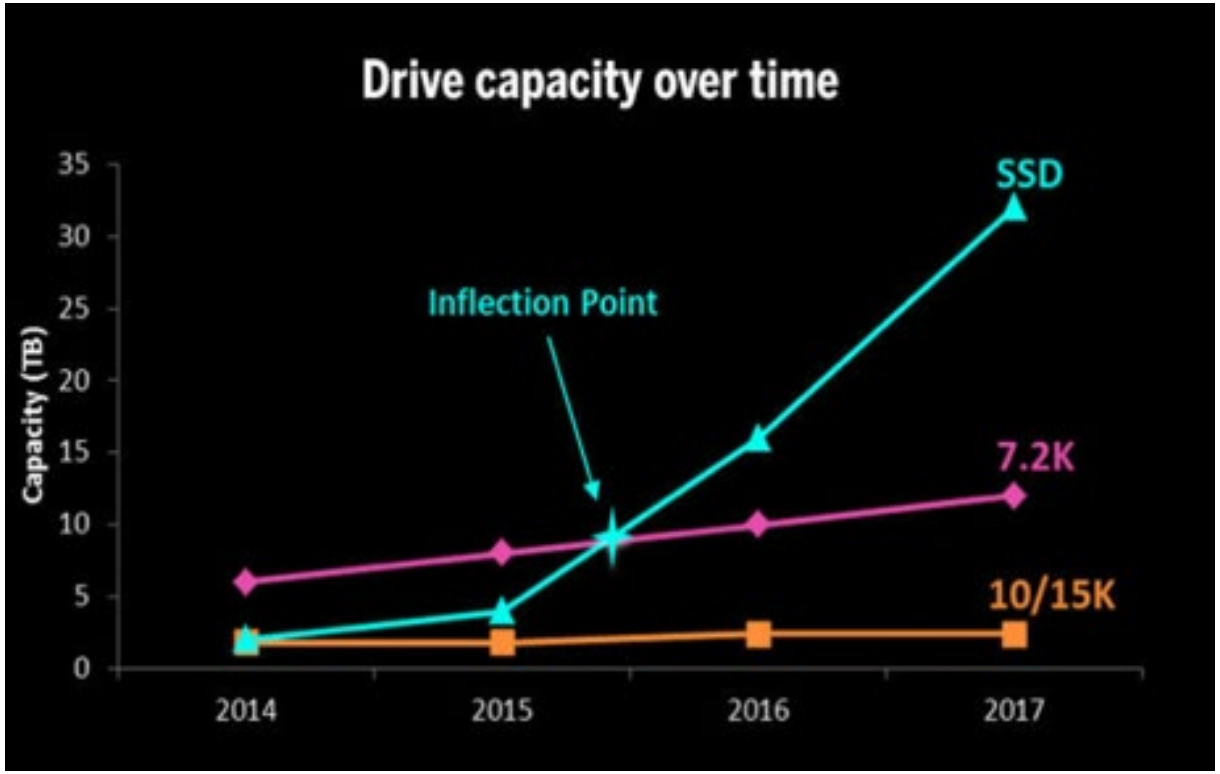
But then Moore's Law Ended…



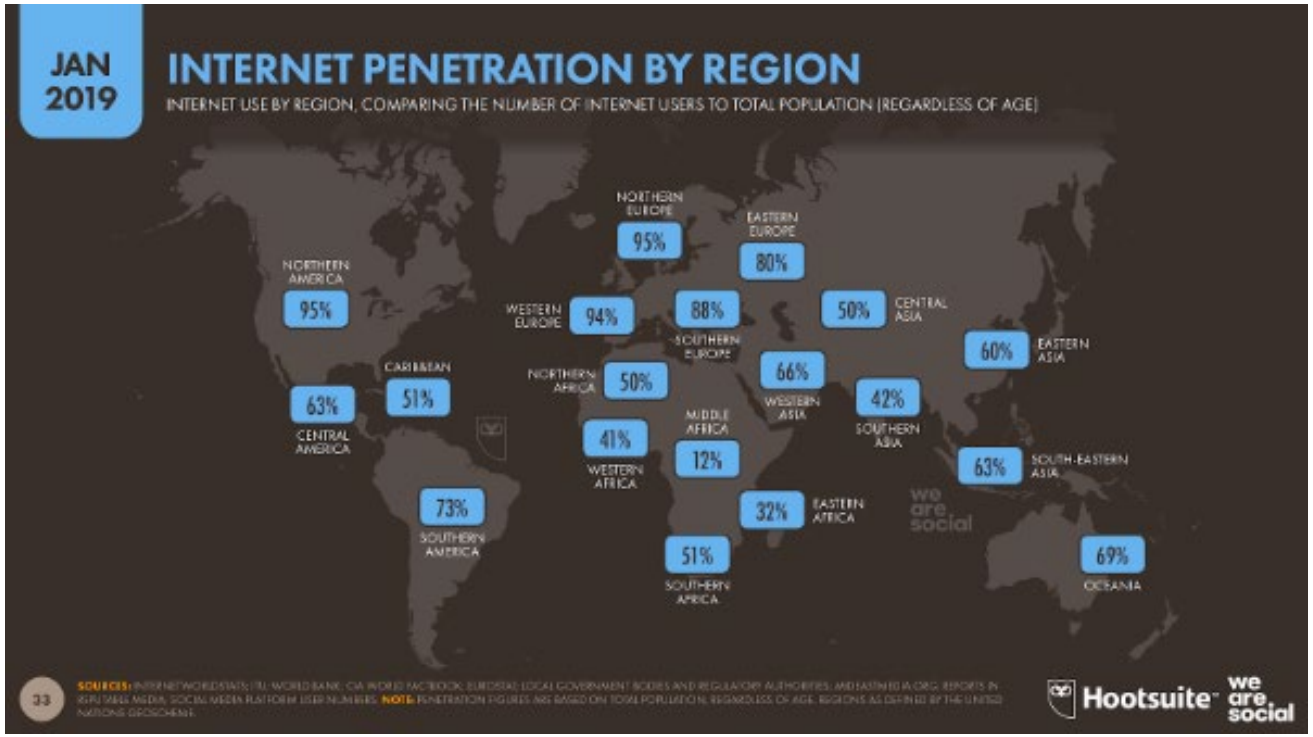
- Moore's Law has (officially) ended -- Feb 2016
  - No longer getting 2 x transistors/chip every 18 months…
  - or even every 24 months
- May have only 2-3 smallest geometry fabrication plants left:
  - Intel and Samsung and/or TSMC
- Vendors moving to 3D stacked chips
  - More layers in old geometries

Storage Capacity is Still Growing!

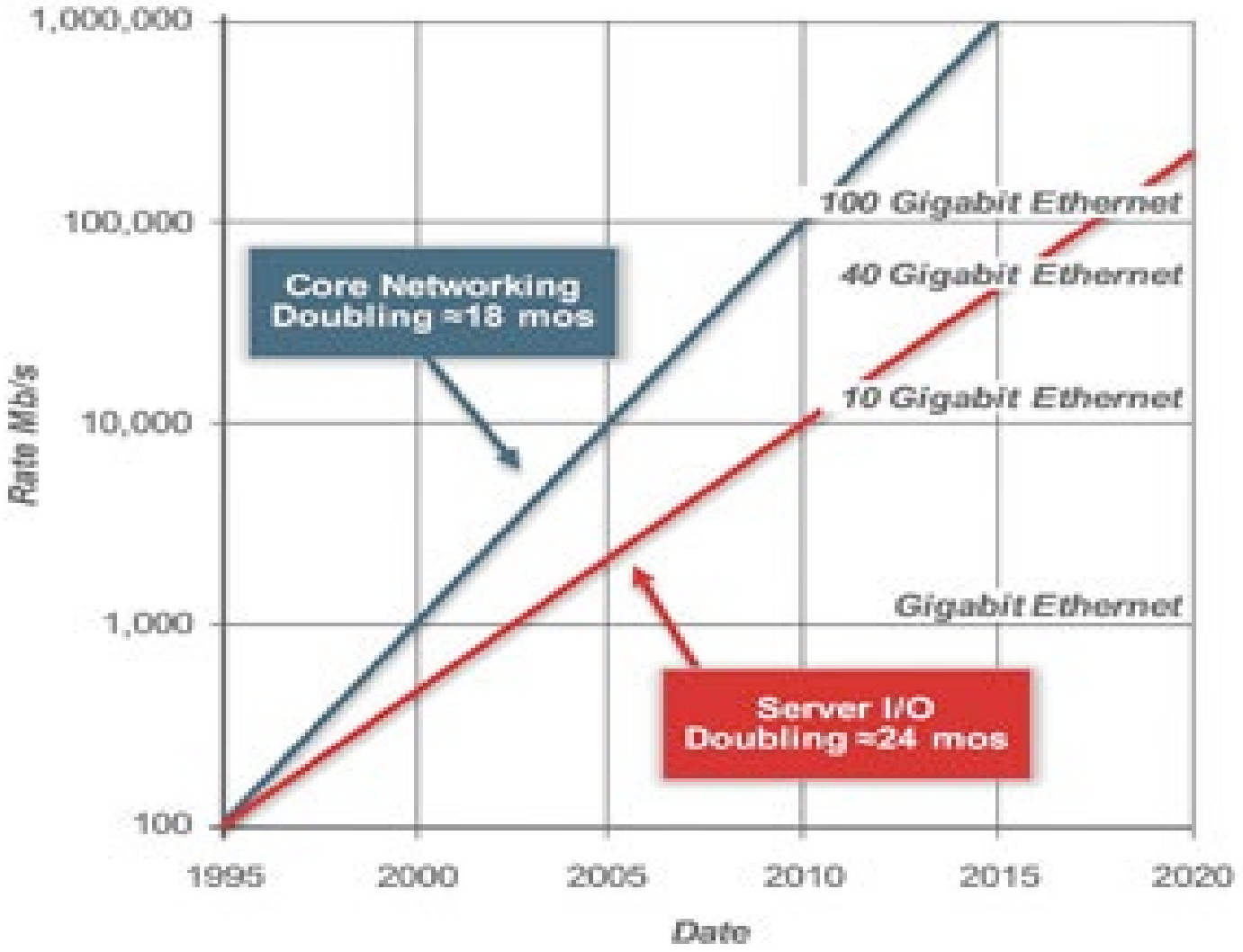




Society is Increasingly Connected…



Network Capacity Still Increasing



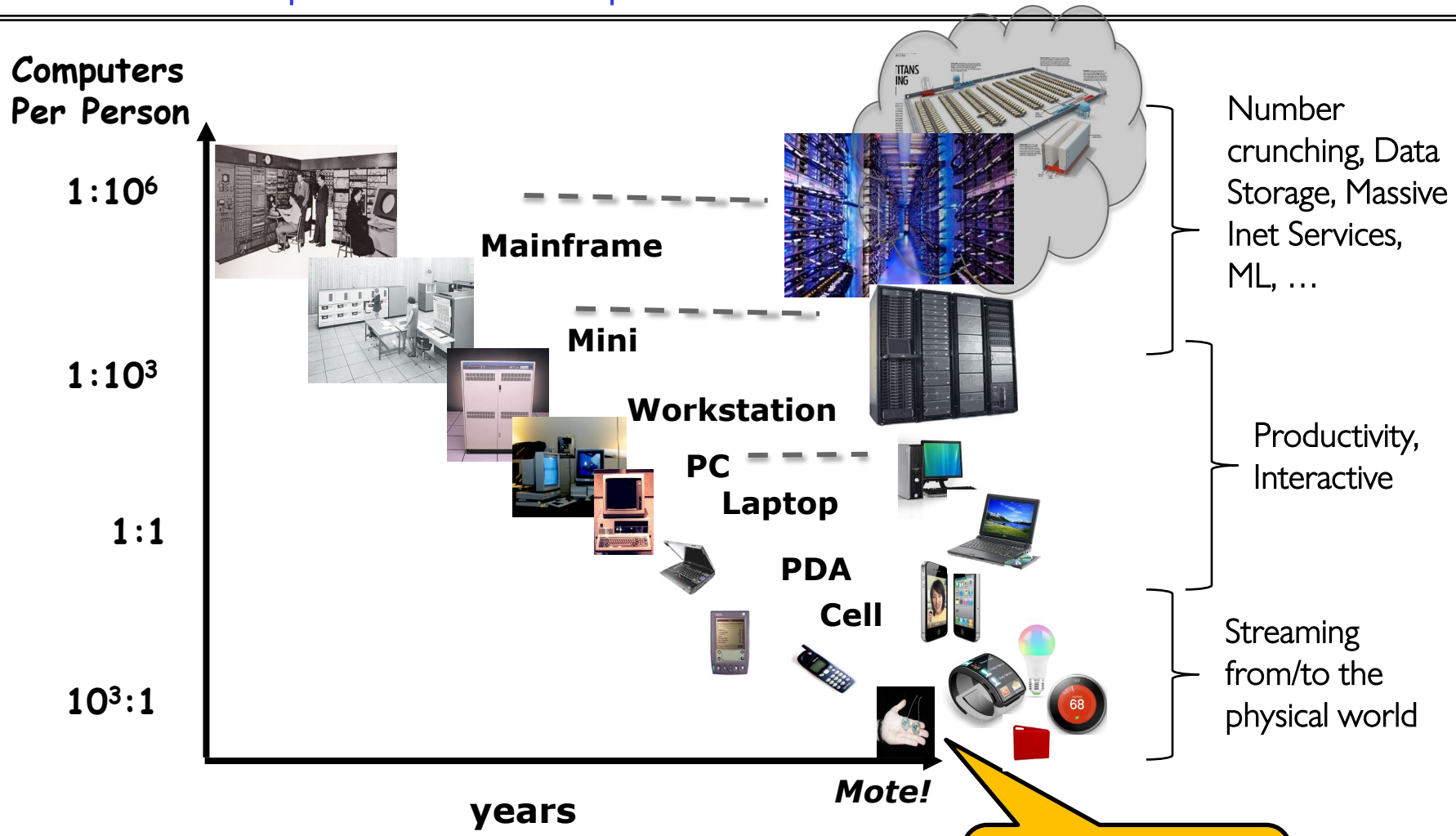
(source: http://www.ospmag.com/issue/article/Time-Is-Not-Always-On-Our-Side)

Not Only PCs connected to the Internet

- In 2011, smartphone shipments exceeded PC shipments!
- 2011 shipments:
  - 487M smartphones
  - 414M PC clients
    - » 210M notebooks
    - » 112M desktops
    - » 63M tablets
  - 25M smart TVs

- 4 billion phones in the world → smartphones over next few years
- Then…

People-to-Computer Ratio Over Time

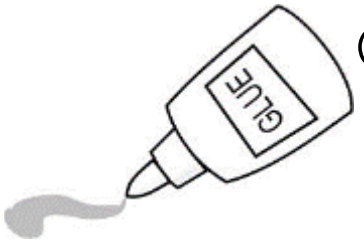


Bell's Law: new computer class per 10 years

What is an Operating System Again?

- 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

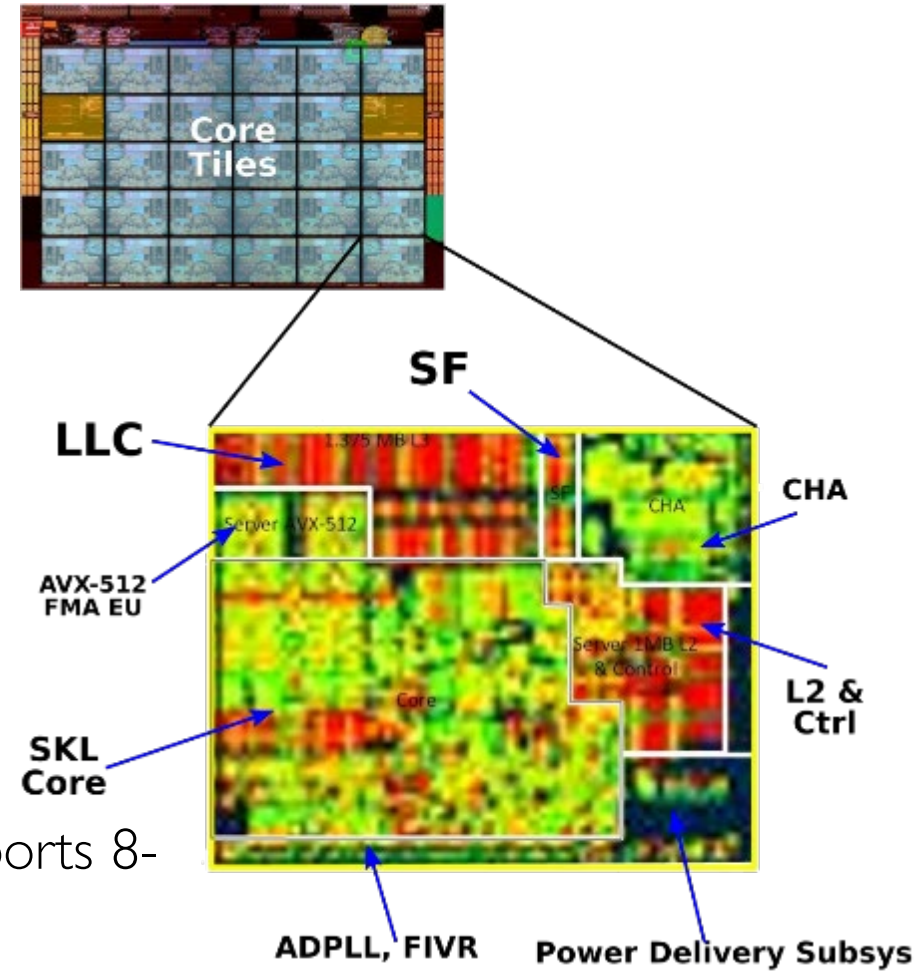
Challenge: Complexity

- 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!

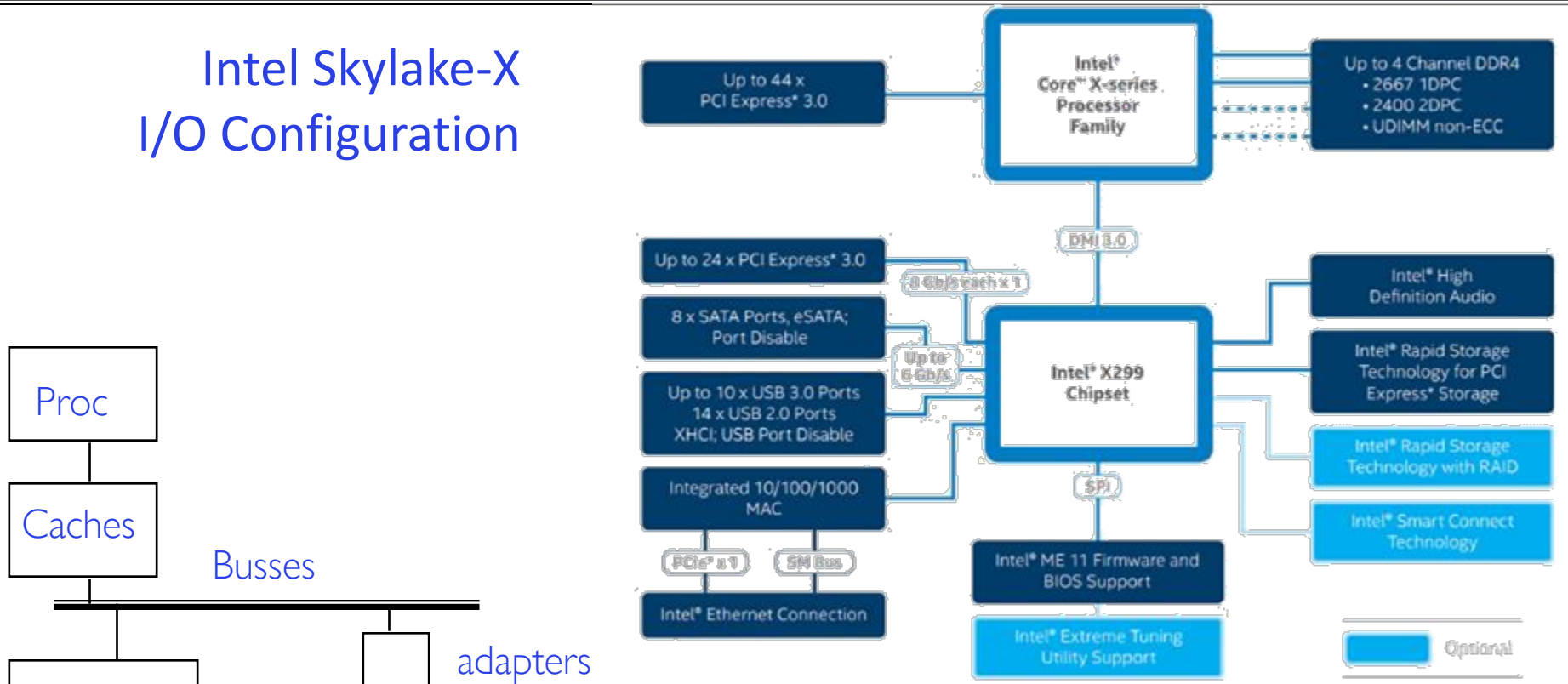
The World Is Parallel: Intel SkyLake (2017)

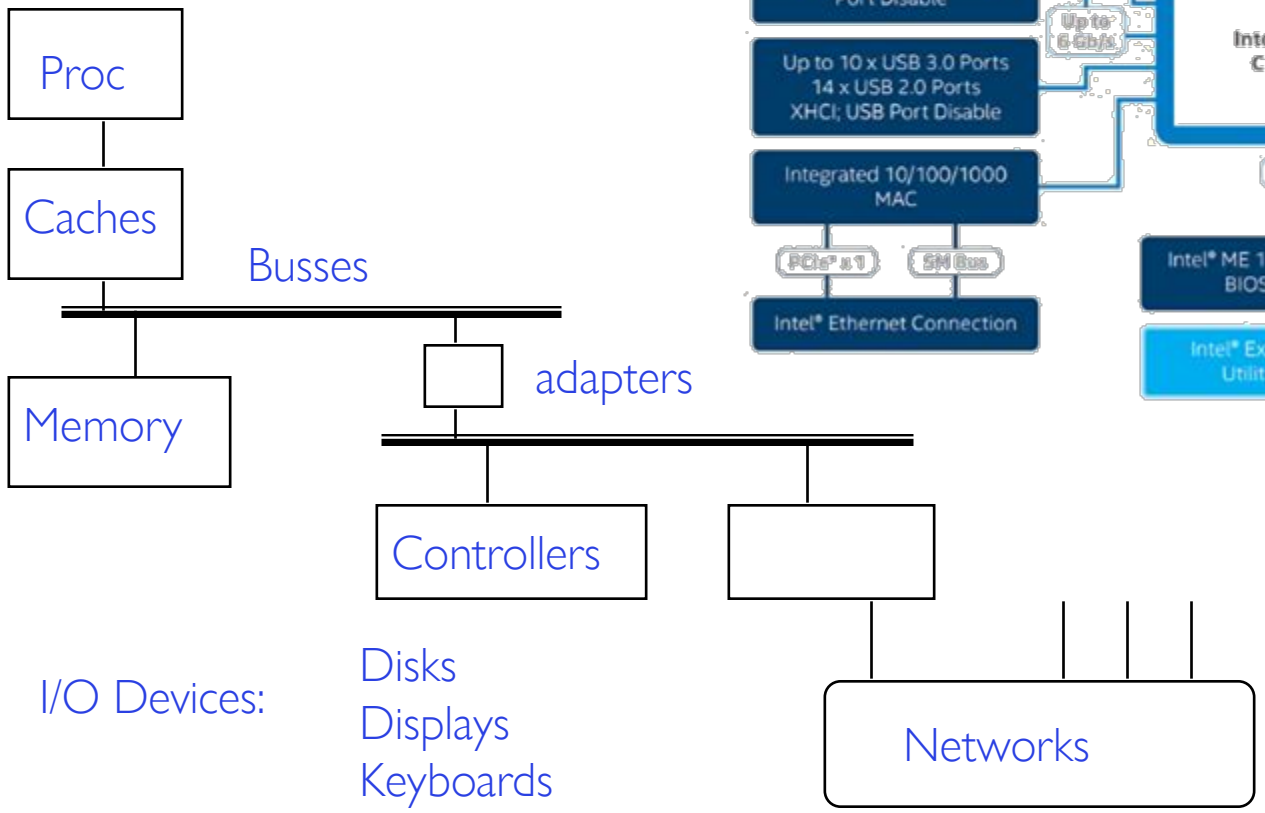
- Up to 28 Cores, 56 Threads
  - 694 mm^2 die size (estimated)
- Many different instructions
  - Security, Graphics
- Caches on chip:
  - L2: 28 MiB
  - Shared L3: 38.5 MiB (non-inclusive)
  - Directory-based cache coherence
- Network:
  - On-chip Mesh Interconnect
  - Fast off-chip network directlry supports 8-chips connected
- DRAM/chips
  - Up to 1.5 TiB
  - DDR4 memory



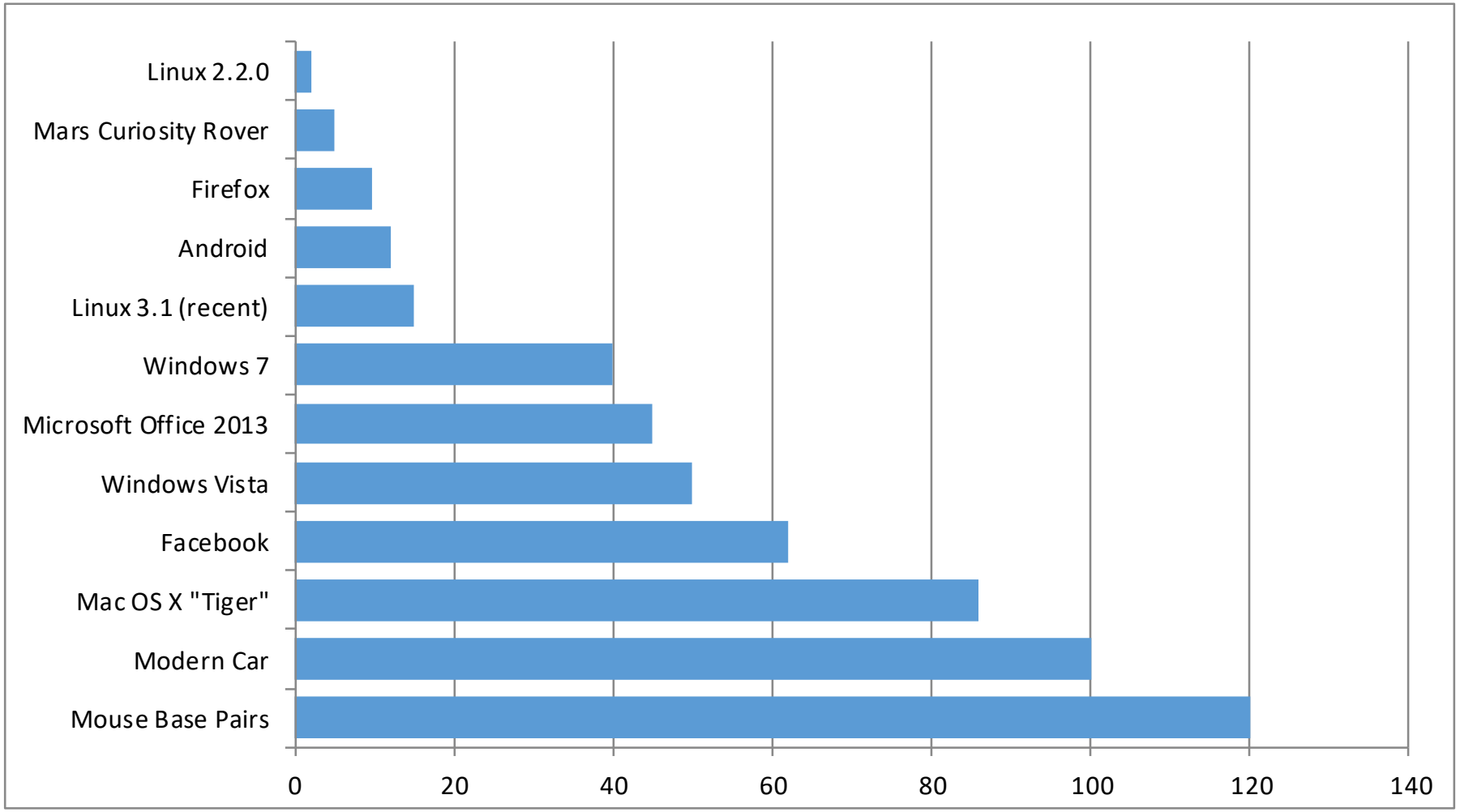
HW Functionality comes with great complexity!

Intel Skylake-X I/O Configuration





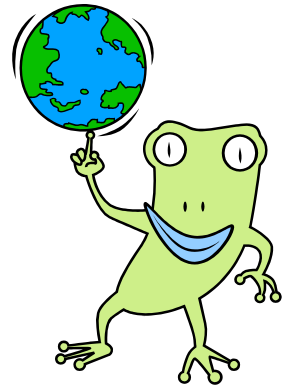
Increasing Software Complexity



Millions of Lines of Code

(source https://informationisbeautiful.net/visualizations/million-lines-of-code/)

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 … ?

Questions

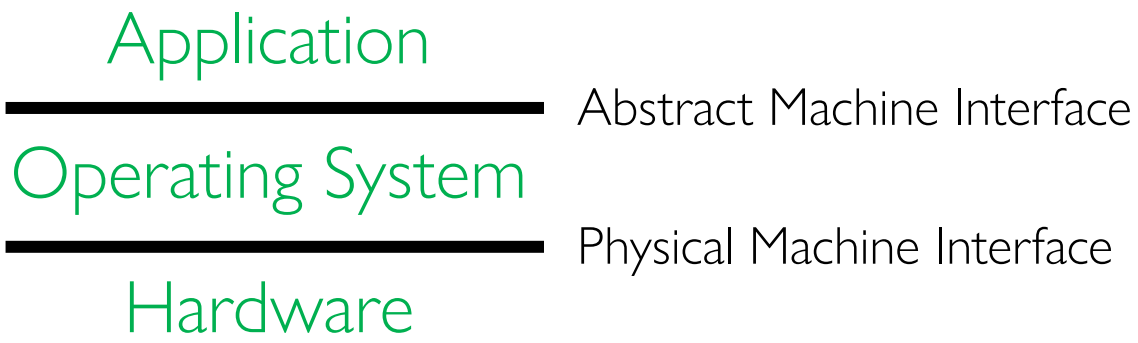
- Does the programmer need to write a single program that performs many independent activities?
- Does every program have to be altered for every piece of hardware?
- Does a faulty program crash everything?
- Does every program have access to all hardware?

Hopefully, no!

Operating Systems help the programmer write robust programs!

OS *Abstracts* the Underlying Hardware

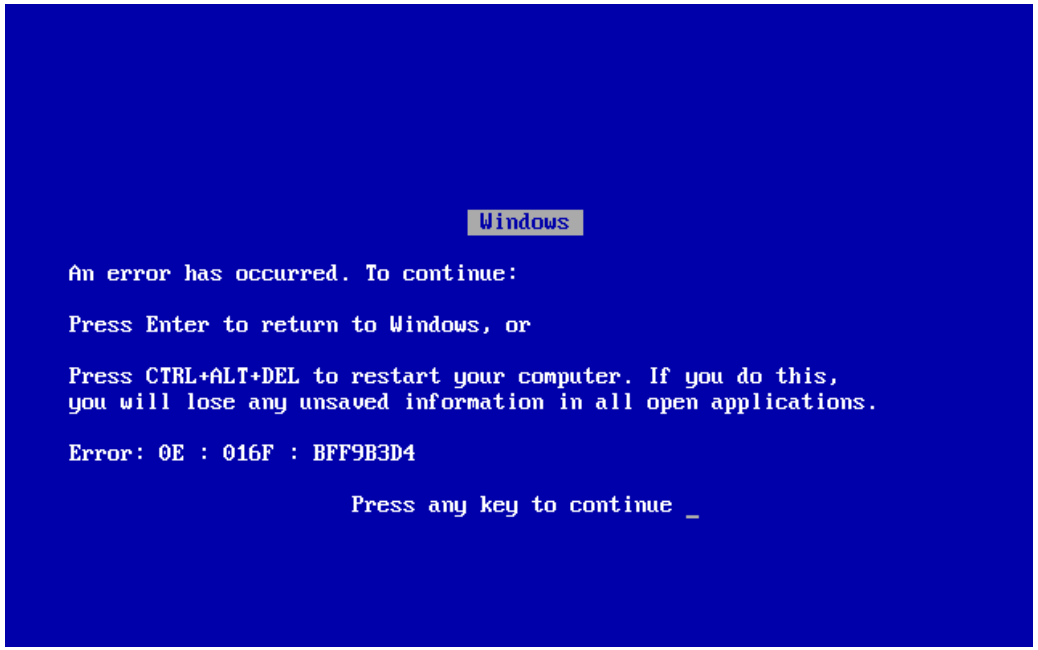
- Processor → Thread
- Memory → Address Space
- Disks, SSDs, … → Files
- Networks → Sockets
- Machines → Processes



- OS as an *Illusionist*:
  - Remove software/hardware quirks (*fight complexity)*
  - Optimize for convenience, utilization, reliability, … *(help the programmer)*
- For any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What hardware interface to handle? (physical reality)
  - What's software interface to provide? (nicer abstraction)

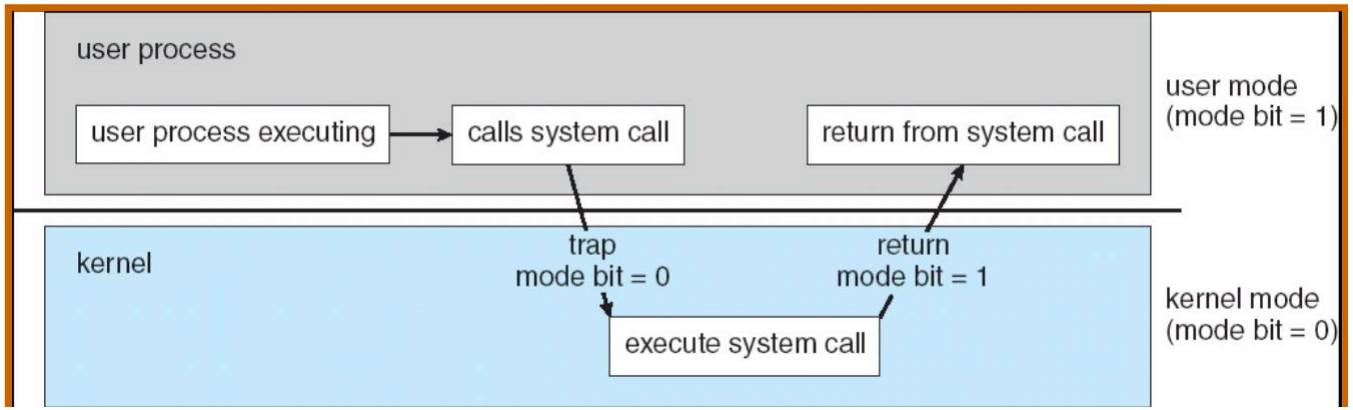
OS *Protects* Processes and the Kernel

- Run multiple applications and:
  - Keep them from interfering with or crashing the operating system
  - Keep them from interfering with or crashing each other

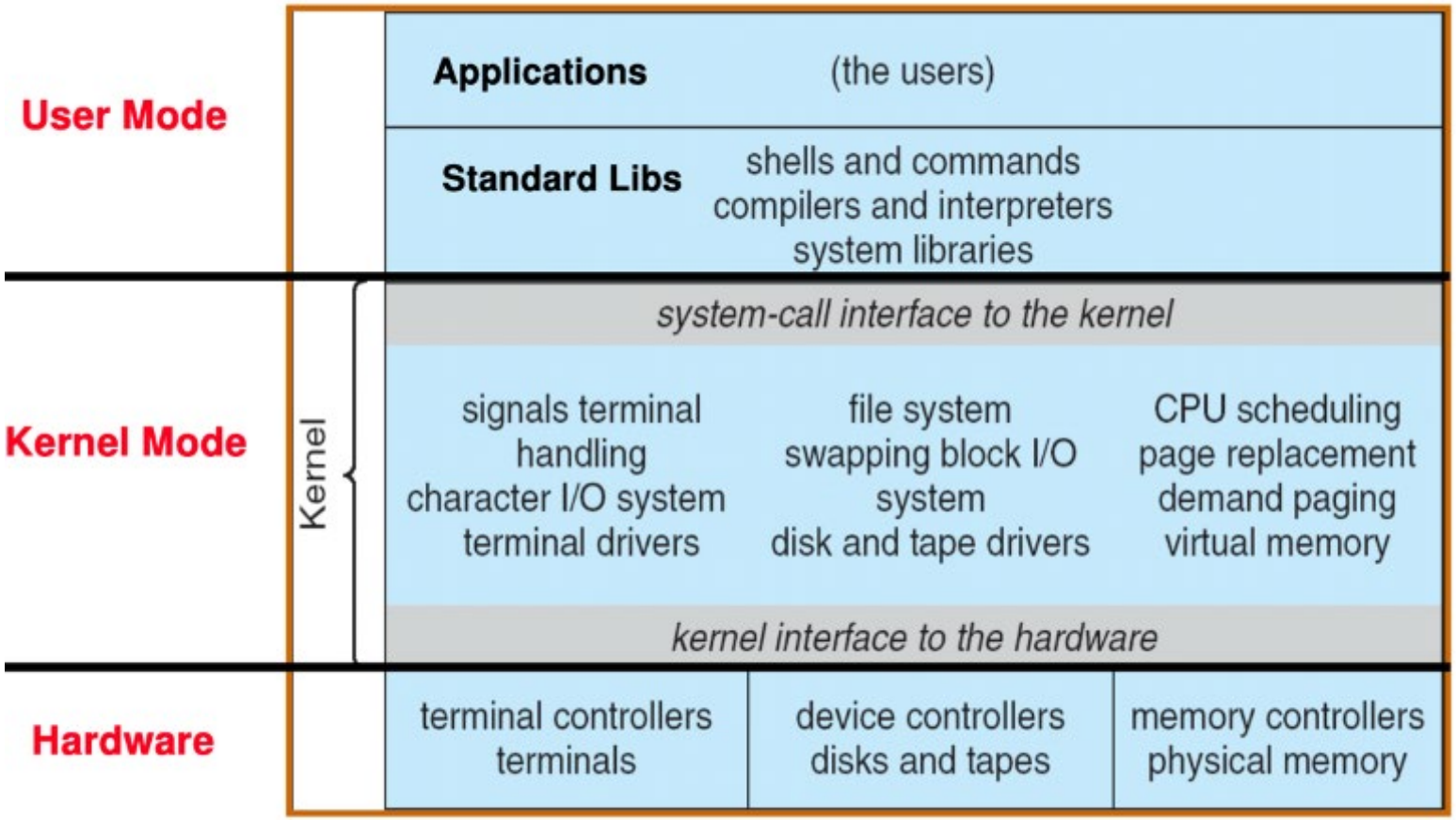


Basic Tool: Dual-Mode Operation

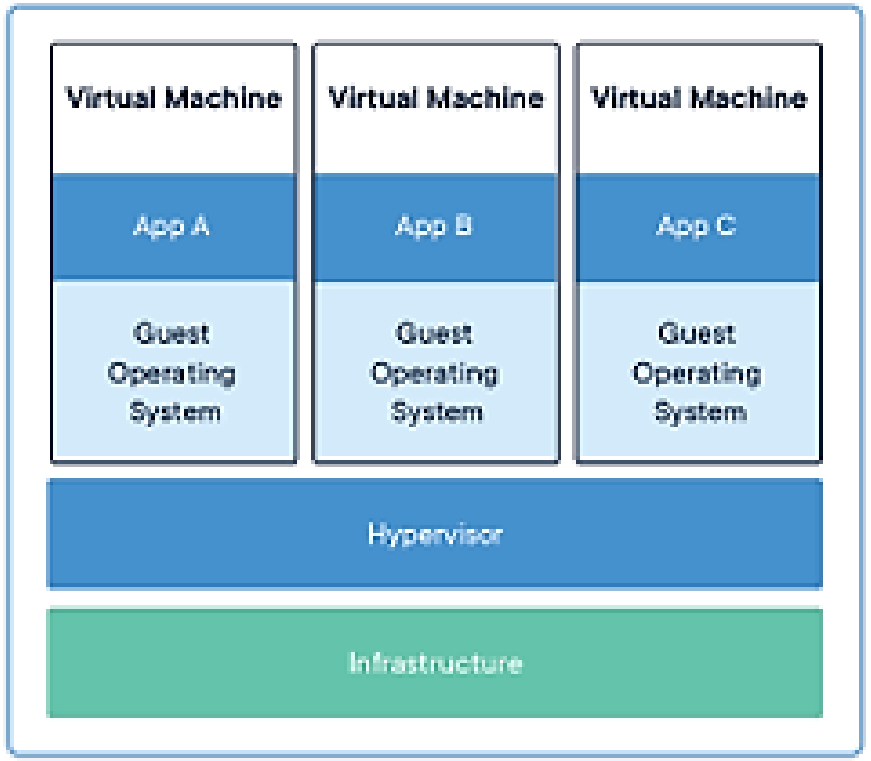
- Hardware provides at least two modes:
  1. Kernel Mode (or "supervisor" mode)
  2. User Mode
- Certain operations are **prohibited** when running in user mode
  - Changing the page table pointer, disabling interrupts, interacting directly w/ hardware, writing to kernel memory
- Carefully controlled transitions between user mode and kernel mode
  - System calls, interrupts, exceptions

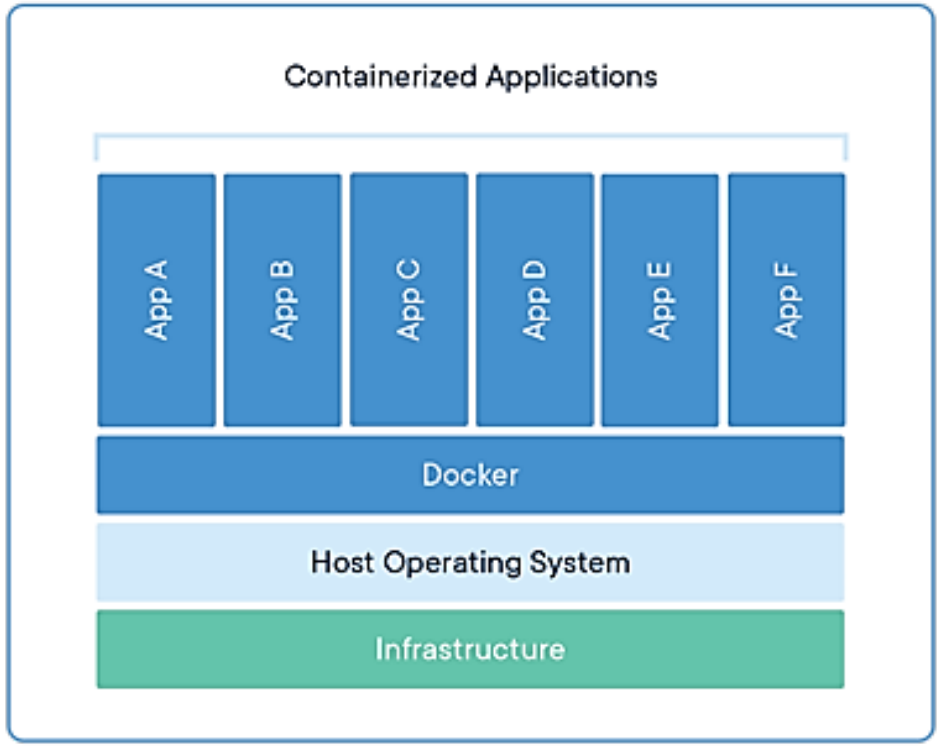


UNIX System Structure



Virtualization: Execution Environments for Systems





Additional layers of protection and isolation can help further manage complexity

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

"In conclusion…Operating Systems:"

- Provide convenient abstractions to handle diverse hardware
  - Convenience, protection, reliability obtained in creating the illusion
- Coordinate resources and protect users from each other
  - Using a few critical hardware mechanisms
- Simplify application development by providing standard services
- Provide fault containment, fault tolerance, and fault recovery