CSC 112: Computer Operating Systems Lecture 1

What is an Operating System?

Department of Computer Science,

Hofstra University

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

Goals for Today

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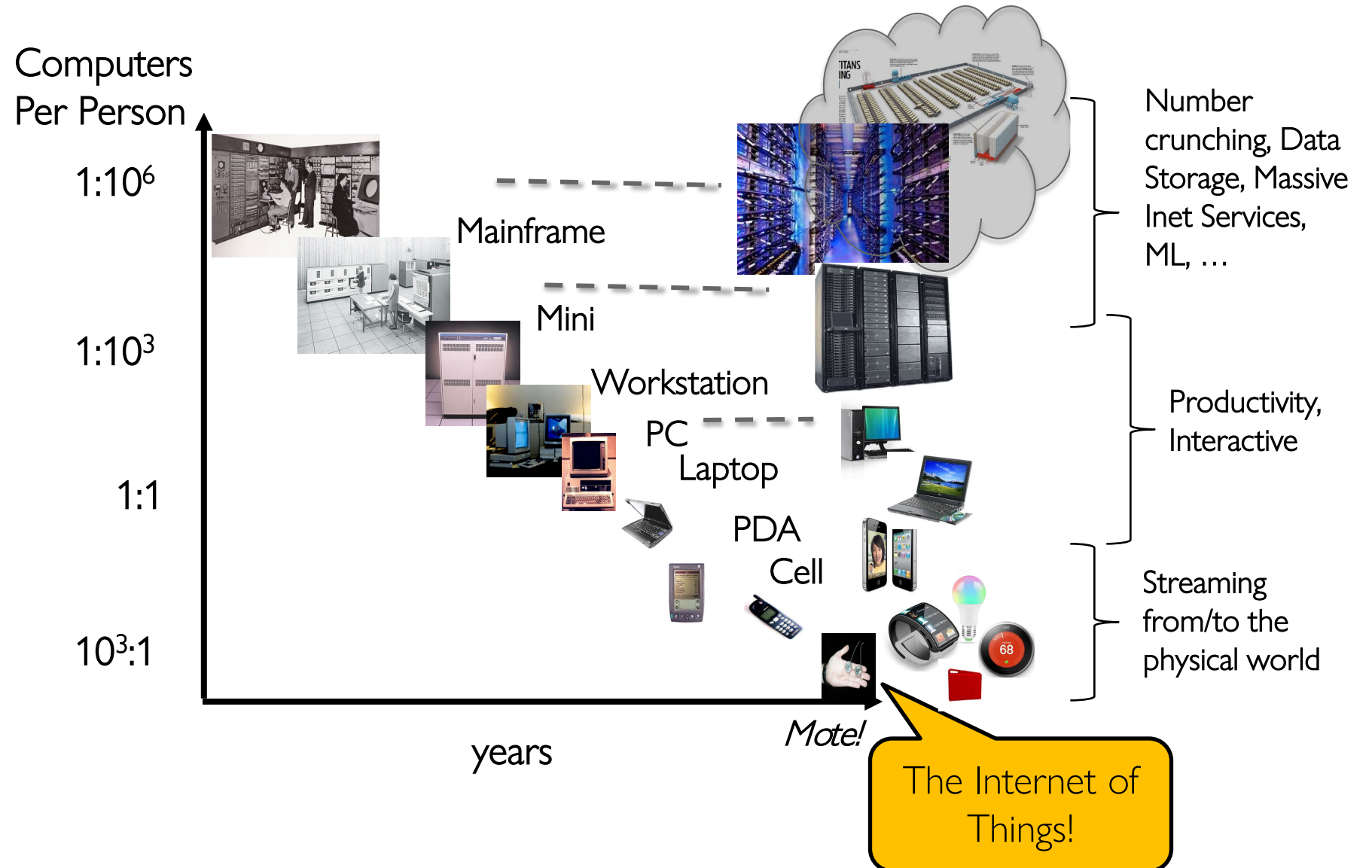- ## Oh, and "How does this class operate?"
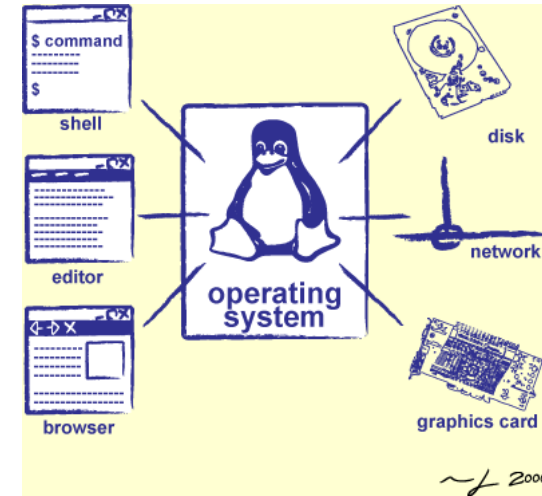
Interactive is important!

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

Computers Per Person

Bell's Law: New computer class every 10 years

$1:10^6$

$1:10^3$

$1:1$

$10^3:1$

Mainframe

Mini

Workstation

PC

Laptop

PDA

Cell

Mote!

years

Number crunching, Data Storage, Massive Inet Services, ML, …

Productivity, Interactive

Streaming from/to the physical world

The Internet of Things!

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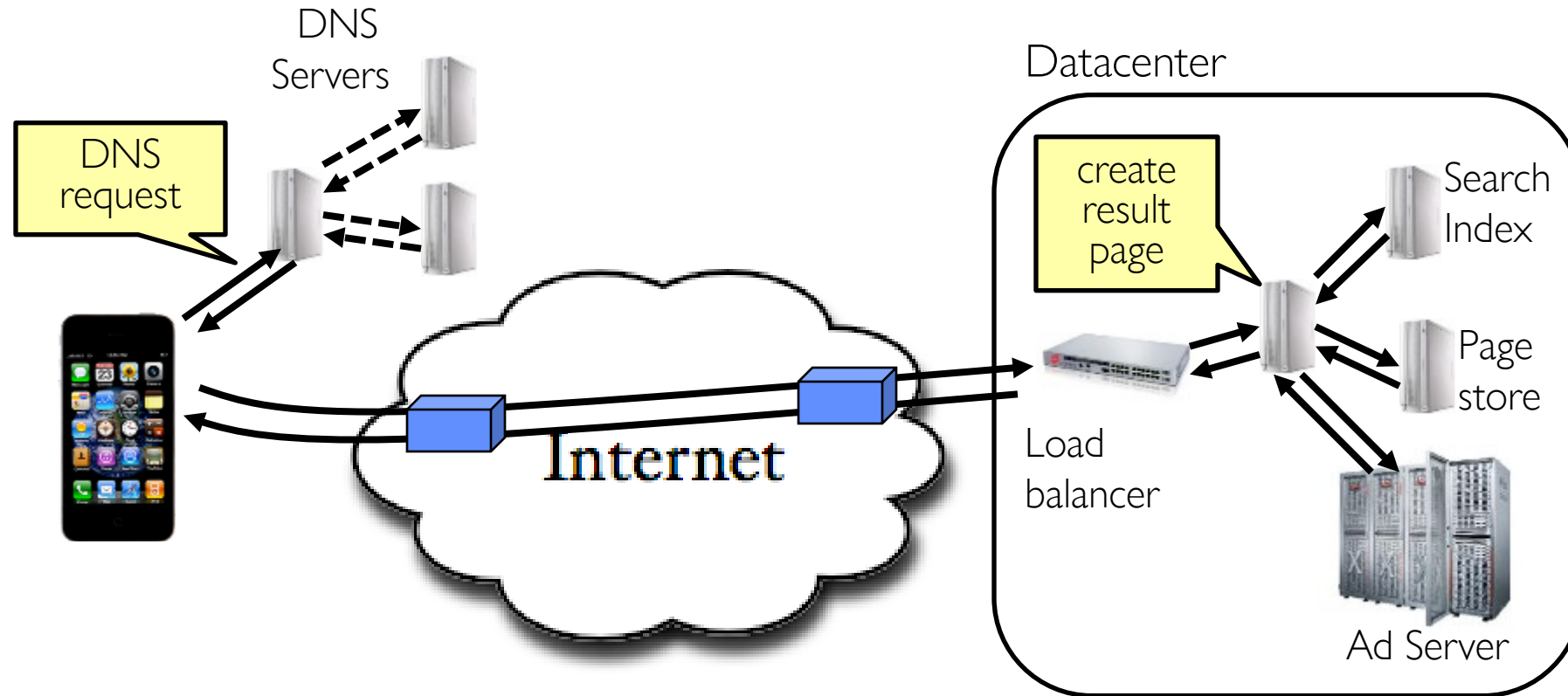
- Complex interaction of multiple components in multiple administrative domains
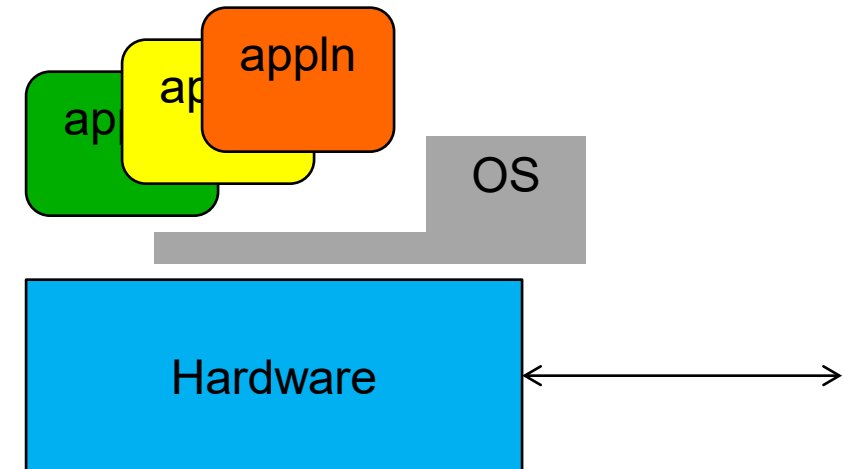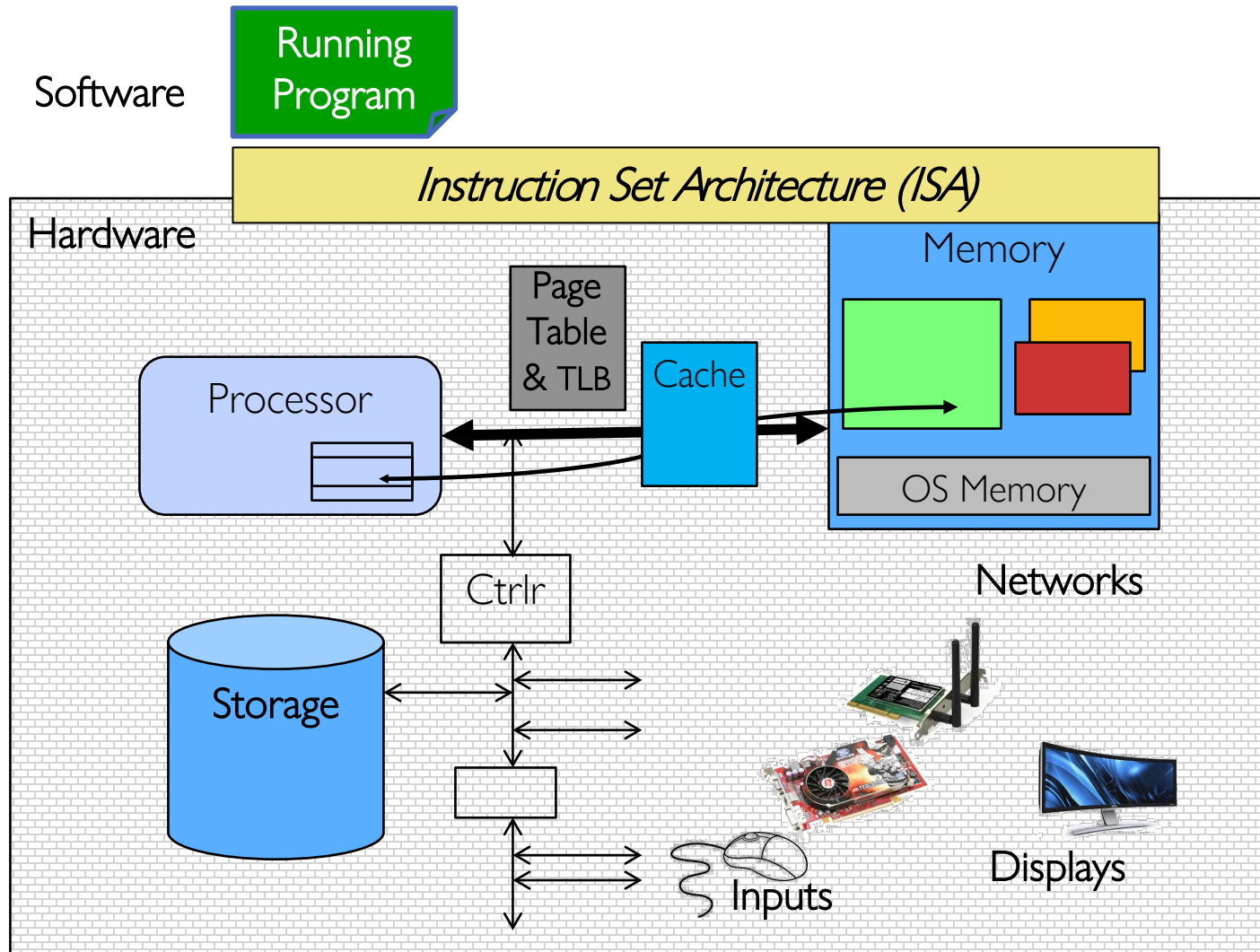 - Systems, services, protocols, …

But: What is an operating system?

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

Hardware/Software Interface

Software

Running Program

Instruction Set Architecture (ISA)

Hardware

Processor

Page Table & TLB

Cache

Memory

OS Memory

Ctrlr

Storage

Networks

Inputs

Displays

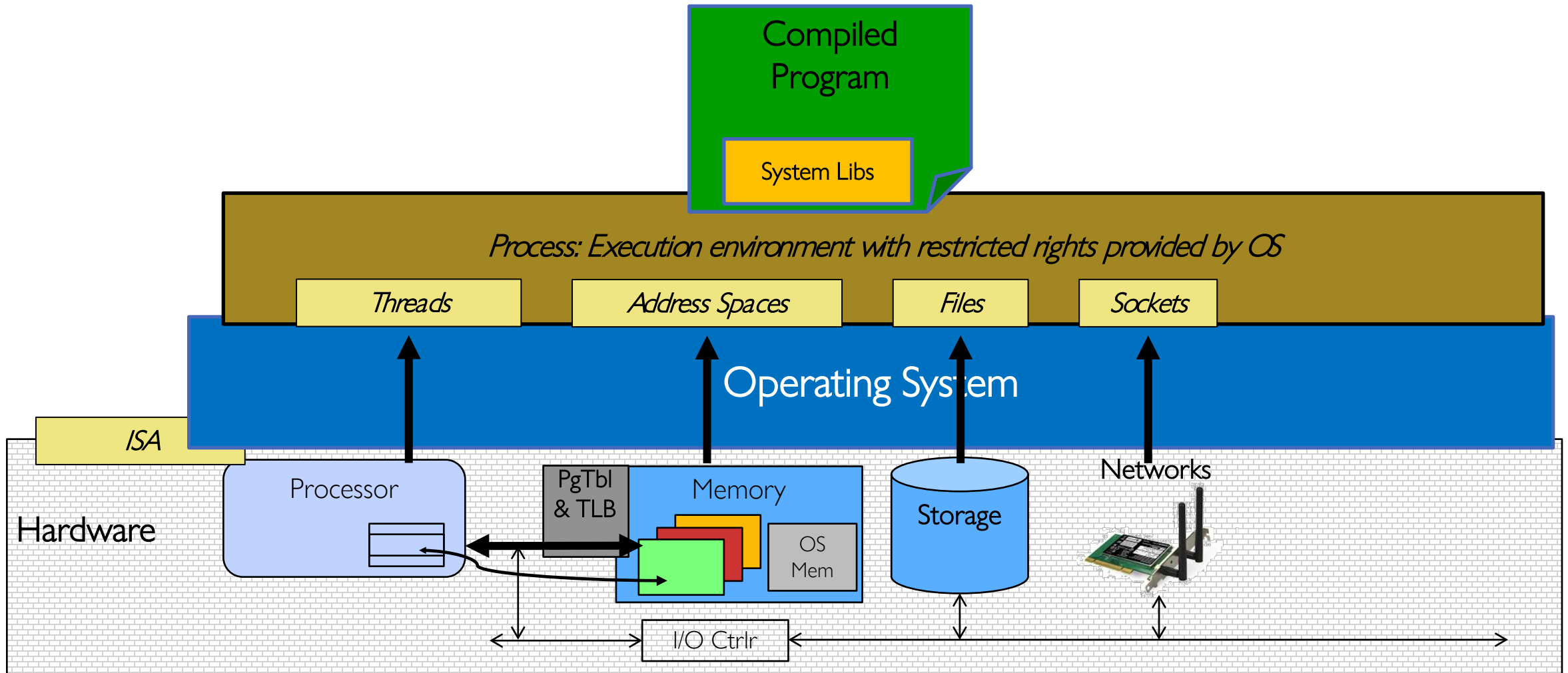The OS *abstracts* these hardware details from the application
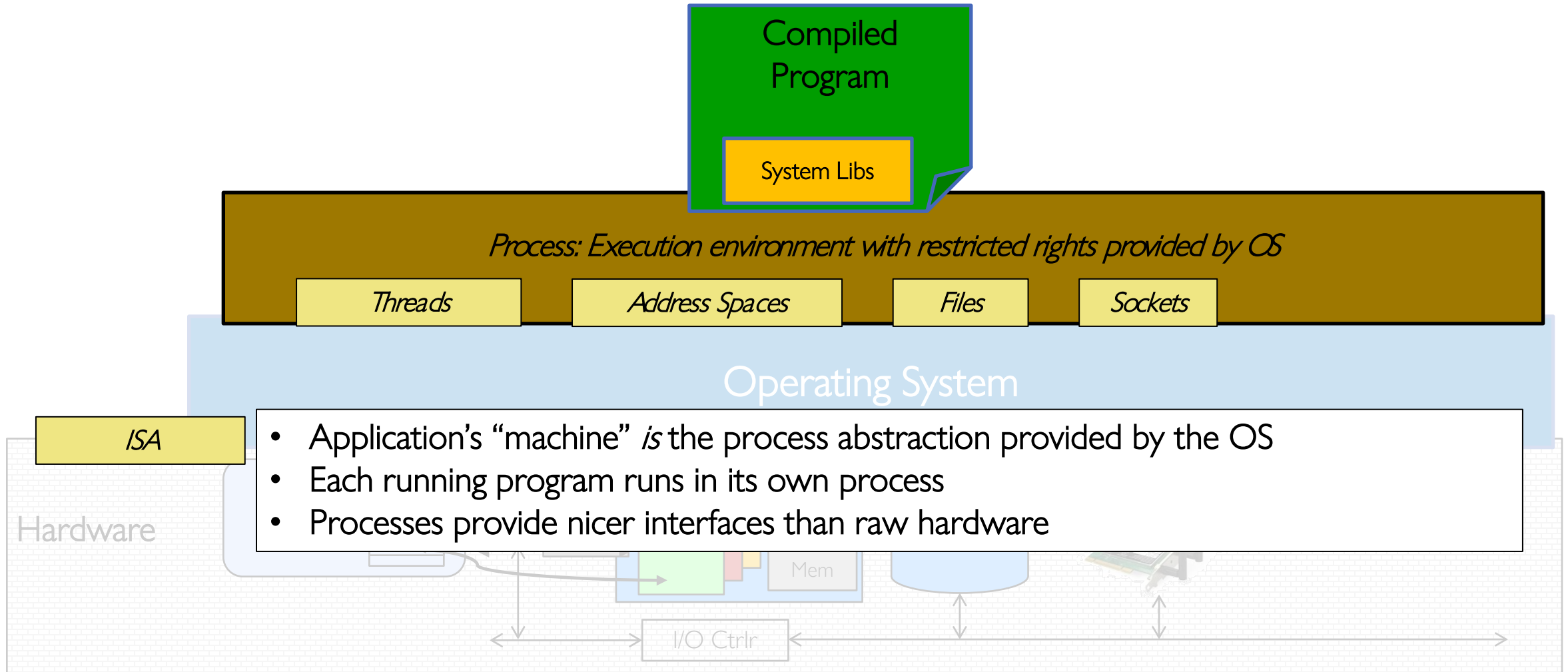
What is an Operating System?

- 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

System Libs

Process: Execution environment with restricted rights provided by OS

Threads | Address Spaces | Files | Sockets

Operating System

ISA

Hardware

Processor

PgTbl & TLB

Memory

OS Mem

Storage

Networks

I/O Ctrlr

Compiled Program's View of the World

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

| Threads | Address Spaces | Files | Sockets |

Operating System

ISA

- 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

Hardware

Mem

I/O Ctrlr

System Programmer's View of the World

Program

```
#include <stdlib.h>

int main(void) {
    printf("Hello!\n")
}
```

Linker

System Libs

Compiler

Process: Execution environment with restricted rights provided by OS

Threads *Address Spaces* *Files* *Sockets*

Operating System

ISA

Hardware

Mem

I/O Ctrlr

- 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
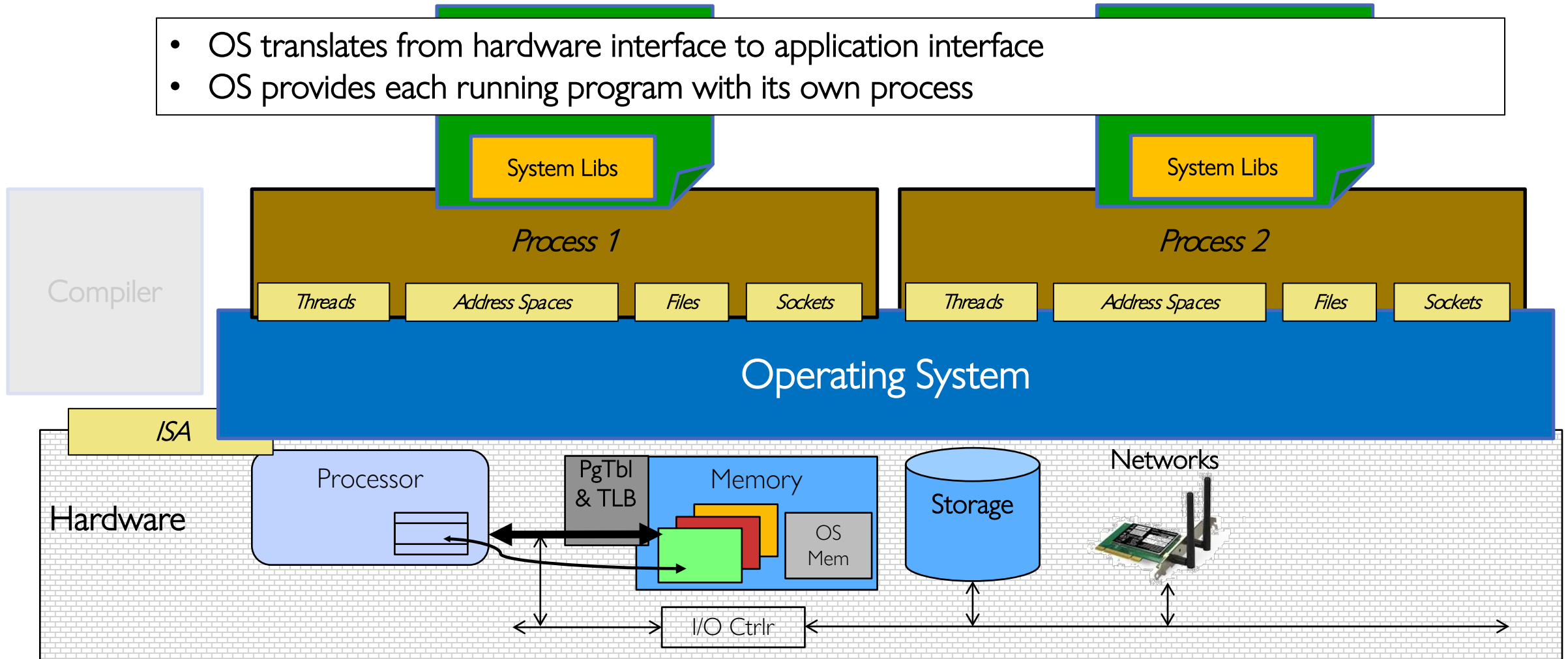 - Open files
  - Open sockets (network connections)
  - …

Operating System's View of the World

Compiled Program 1

System Libs

Compiled Program 2

System Libs

Compiler

Process 1

| Threads | Address Spaces | Files | Sockets |

Process 2

| Threads | Address Spaces | Files | Sockets |

Operating System

ISA

Hardware

Processor

PgTbl & TLB

Memory

OS Mem

Storage

Networks

I/O Ctrlr
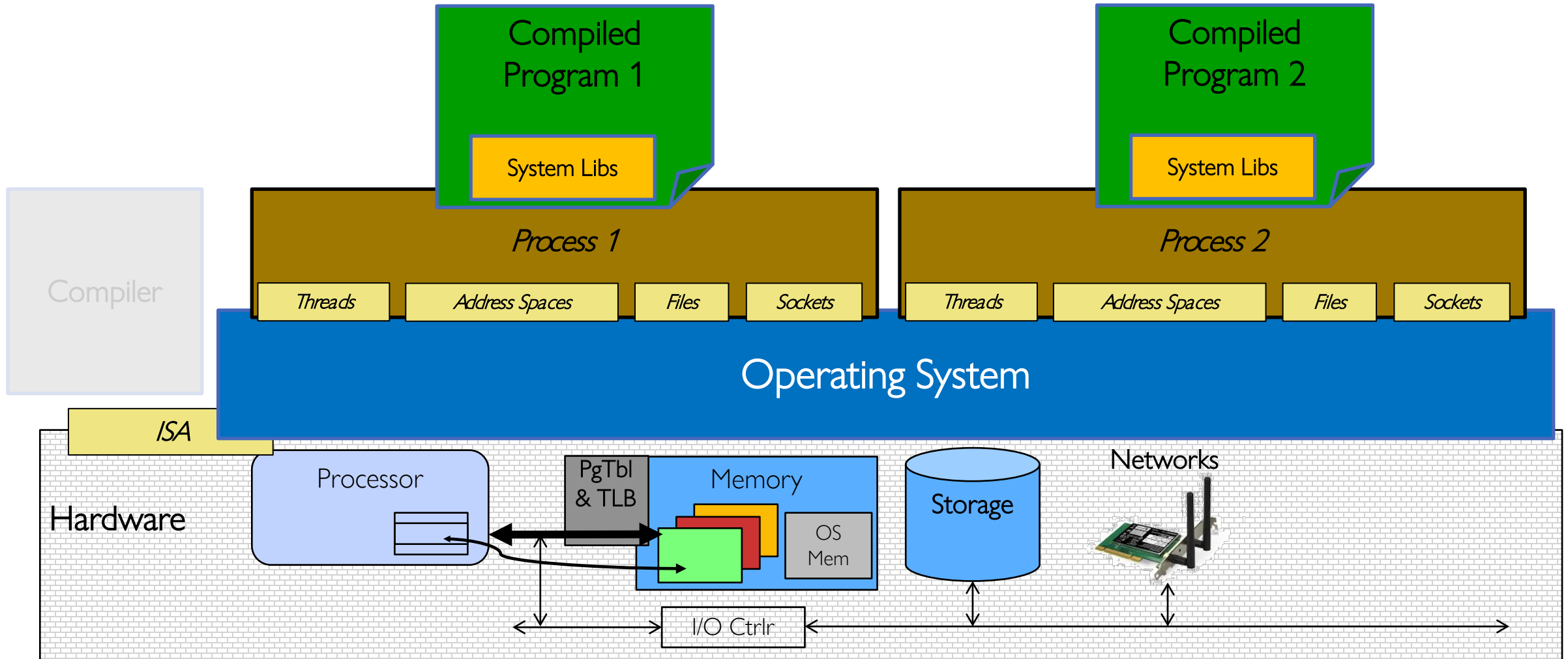
Operating System's View of the World

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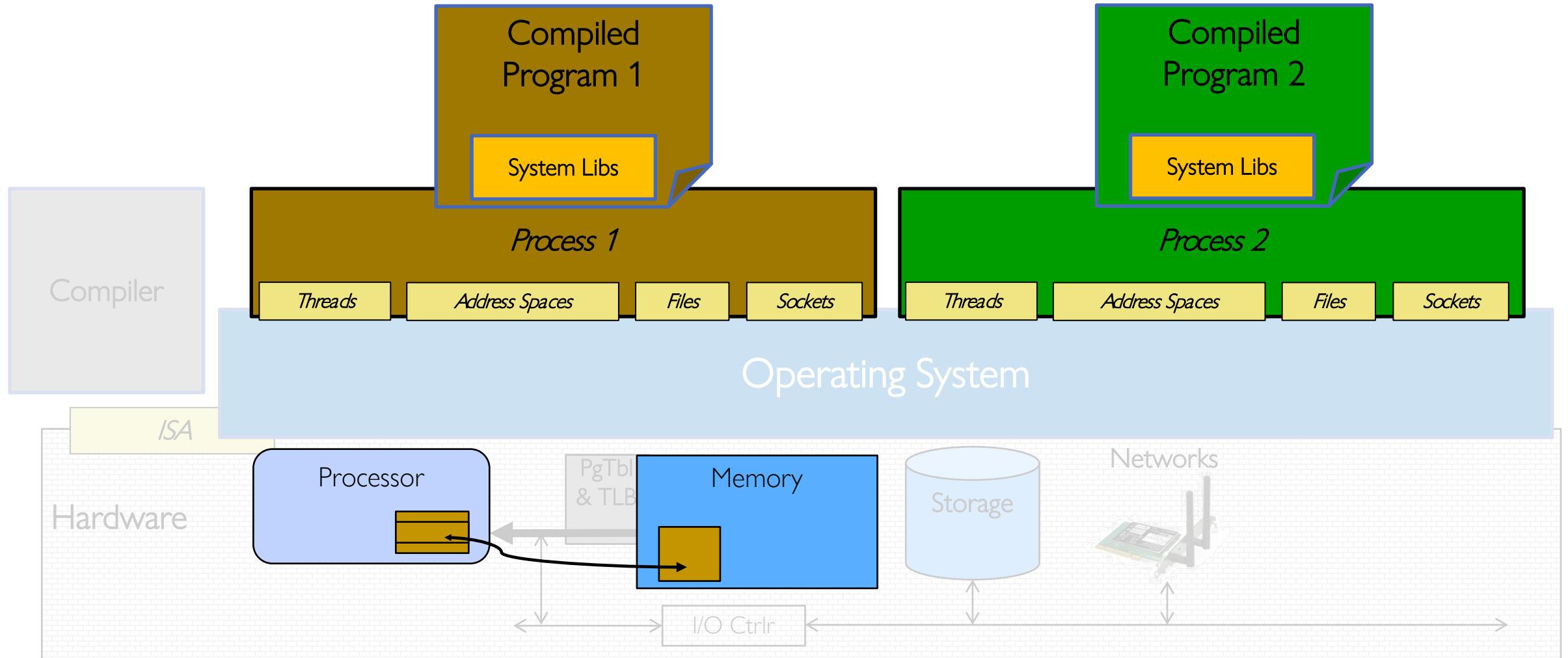- OS provides each running program with its own process

System Libs

System Libs
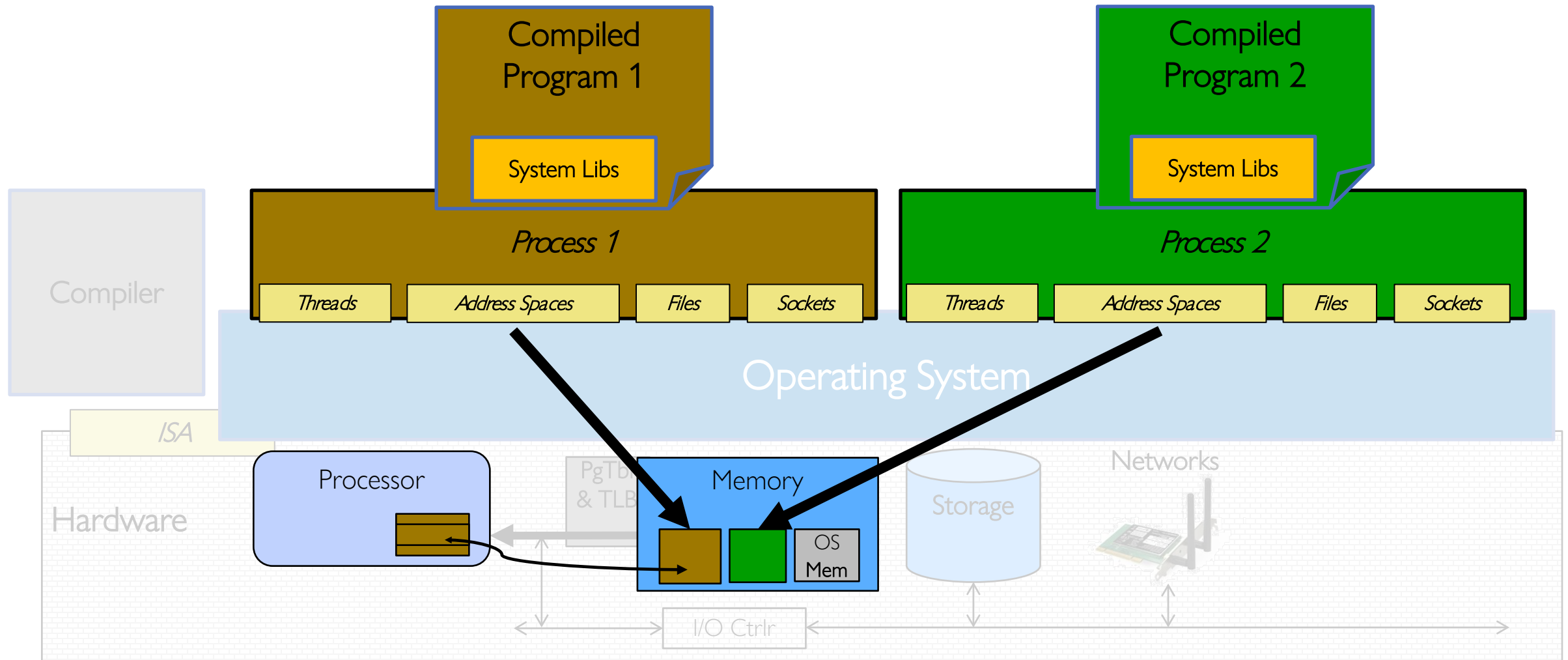
Process 1

Process 2

| *Threads* | *Address Spaces* | *Files* | *Sockets* | *Threads* | *Address Spaces* | *Files* | *Sockets* |

Operating System

Compiler

ISA

Hardware

Processor

PgTbl & TLB

Memory

OS Mem

Storage

Networks

I/O Ctrlr

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
- Referee
  - Manage protection, isolation, and sharing of resources
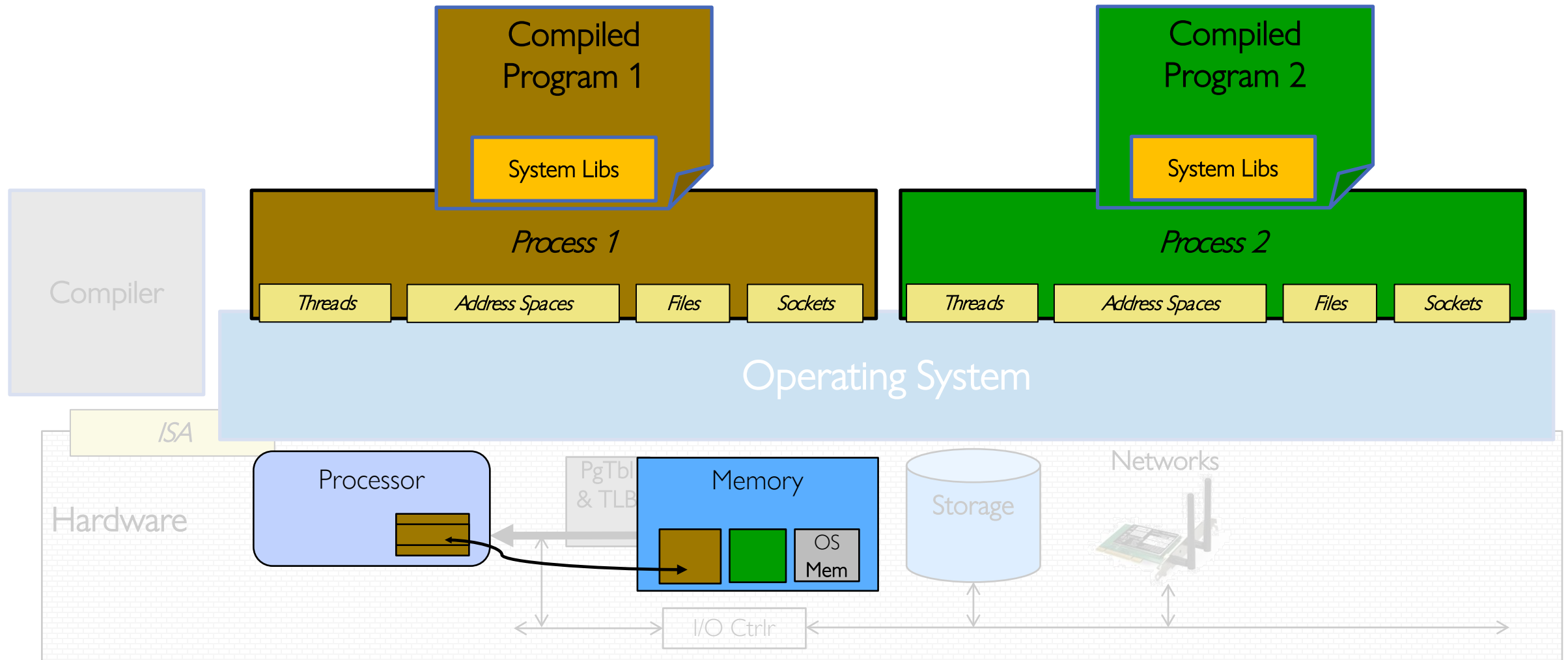   - » Resource allocation and communication
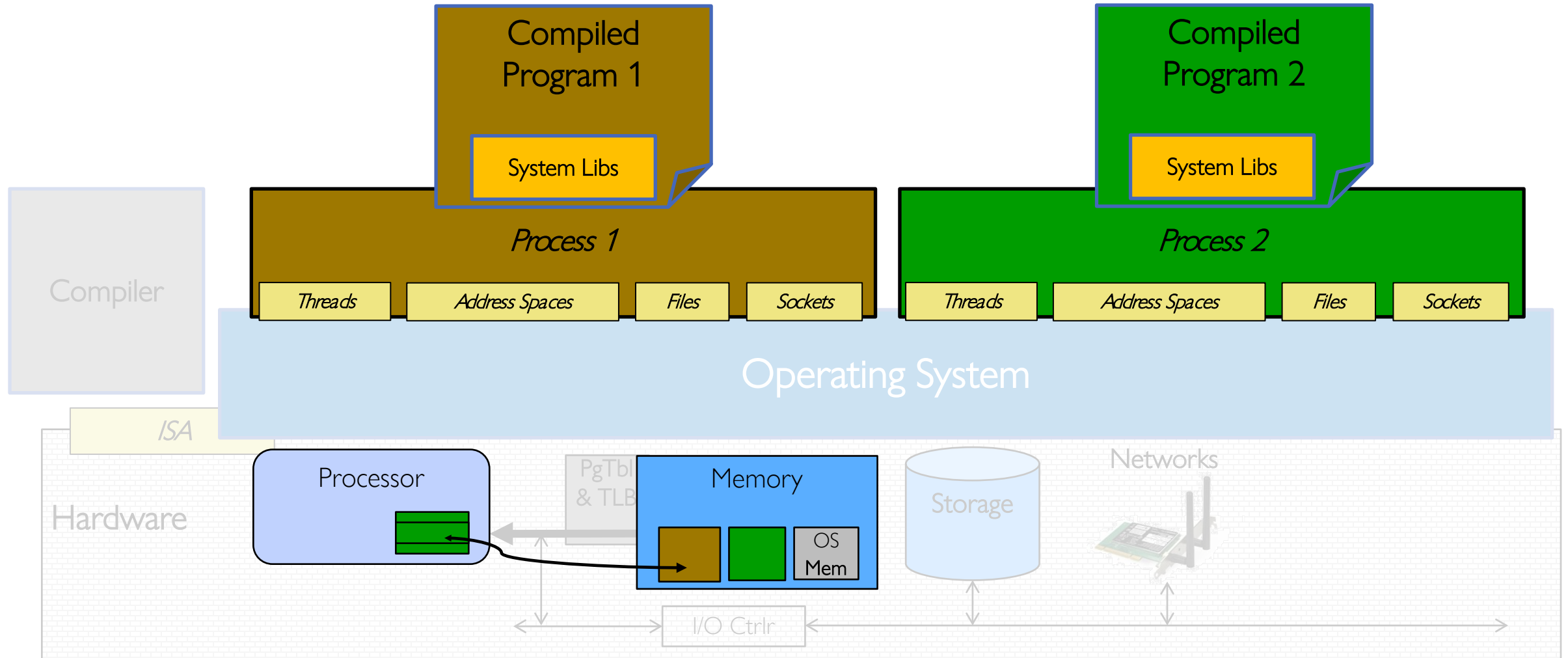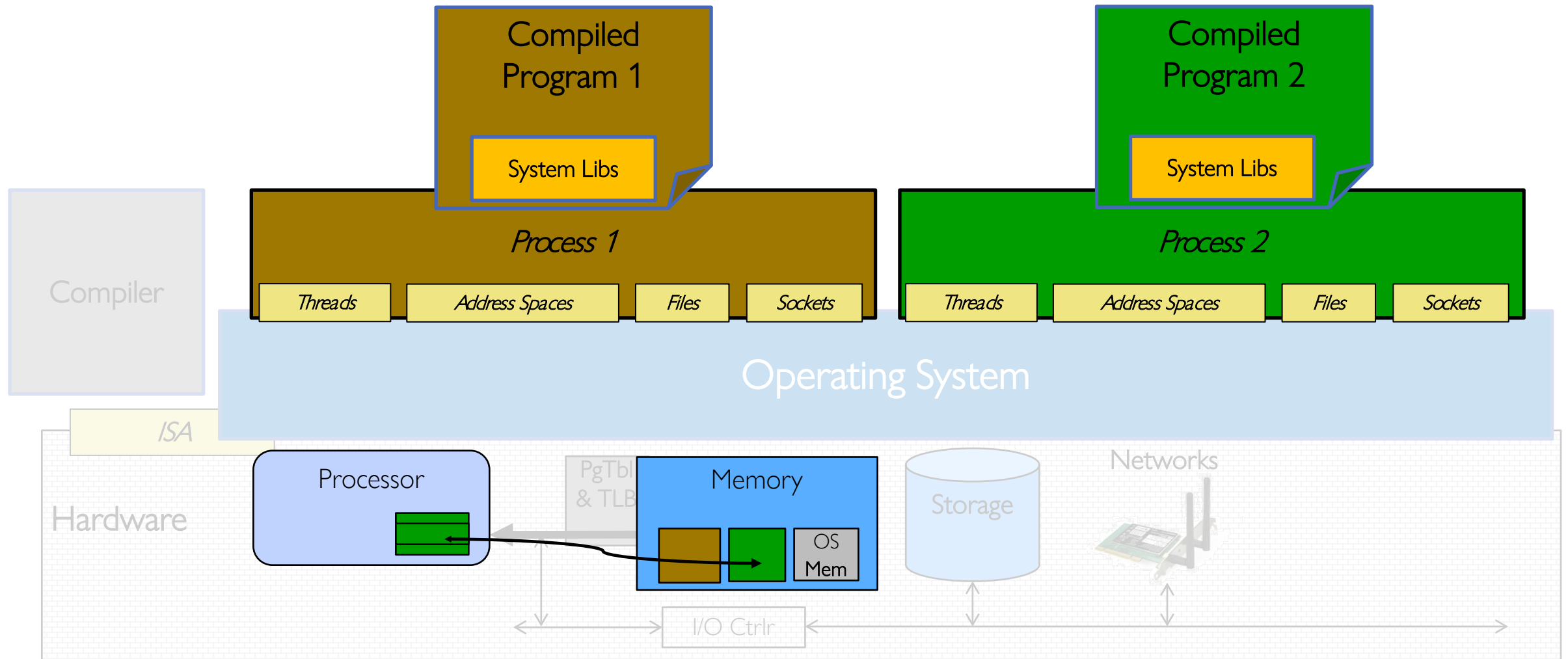
OS Basics: Running a Process

Compiled Program 1

System Libs

Process 1

| Threads | Address Spaces | Files | Sockets |

Compiled Program 2

System Libs

Process 2

| Threads | Address Spaces | Files | Sockets |

Operating System

Compiler

ISA

Hardware

Processor

PgTbl & TLB

Memory

Storage

Networks

I/O Ctrlr

OS Basics: Switching Processes

OS Basics: Switching Processes

Compiled Program 1

System Libs

Process 1

| Threads | Address Spaces | Files | Sockets |

Compiled Program 2

System Libs

Process 2

| Threads | Address Spaces | Files | Sockets |

Compiler

Operating System

ISA

Hardware

Processor

PgTbl & TLB

Memory

OS Mem

Storage

Networks

I/O Ctrlr

OS Basics: Switching Processes

Compiled
Program 1

System Libs

Process 1

| Threads | Address Spaces | Files | Sockets |

Compiled
Program 2

System Libs

Process 2

| Threads | Address Spaces | Files | Sockets |

Operating System

Compiler

ISA

Hardware

Processor

PgTbl
& TLB

Memory

OS
Mem

Storage

Networks

I/O Ctrlr

OS Basics: Switching Processes

OS Basics: Protection

Compiled Program 1

System Libs

Process 1

Threads | Address Spaces | Files | Sockets

Compiled Program 2

System Libs

Process 2

Threads | Address Spaces | Files | Sockets

Compiler

Operating System

ISA

Hardware

Processor

PgTbl & TLB

Memory

Old Mem

Storage

Networks

I/O Ctrlr
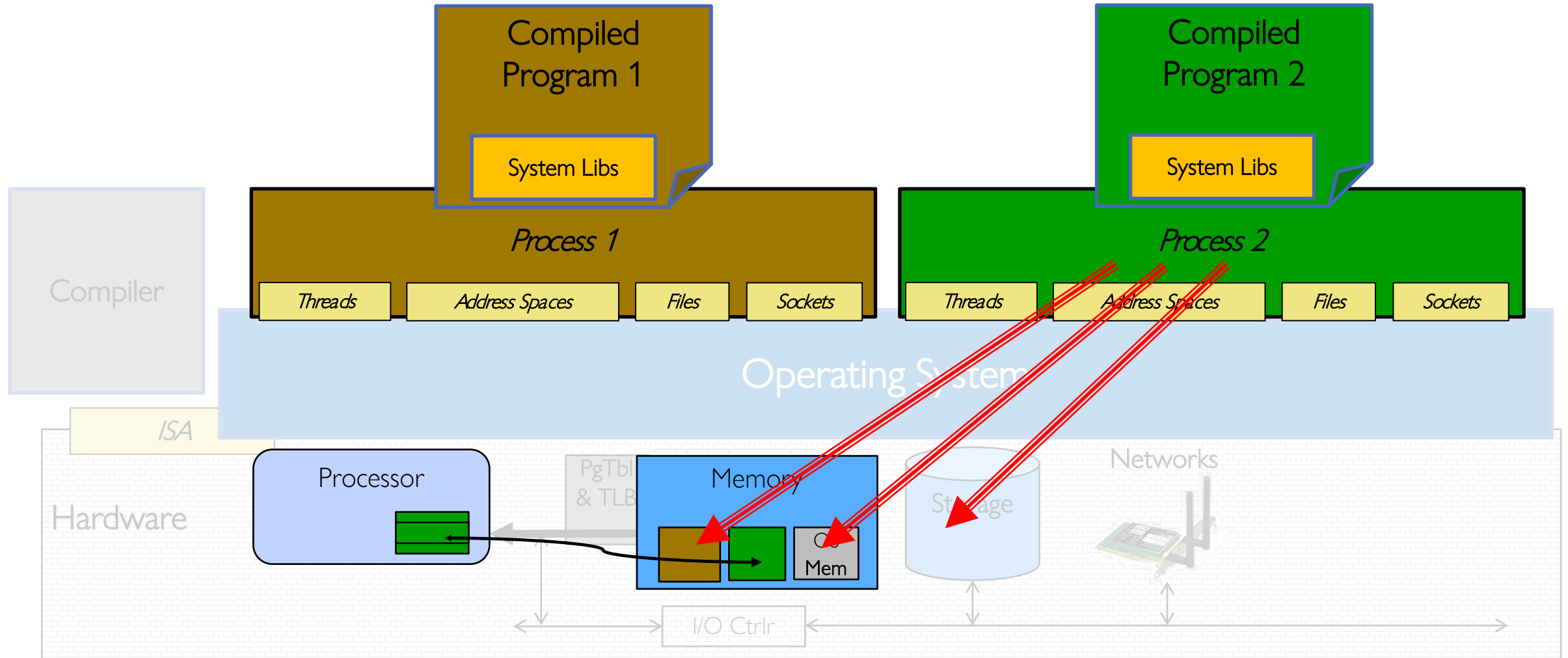
OS Basics: Protection

OS Basics: Protection

Process 1   Process 2   Process 3

Software
OS Hardware Virtualization

Hardware   ISA

Memory

Processor

Protection
Boundary

OS Memory

Ctrlr

Storage   Networks   Displays

Inputs
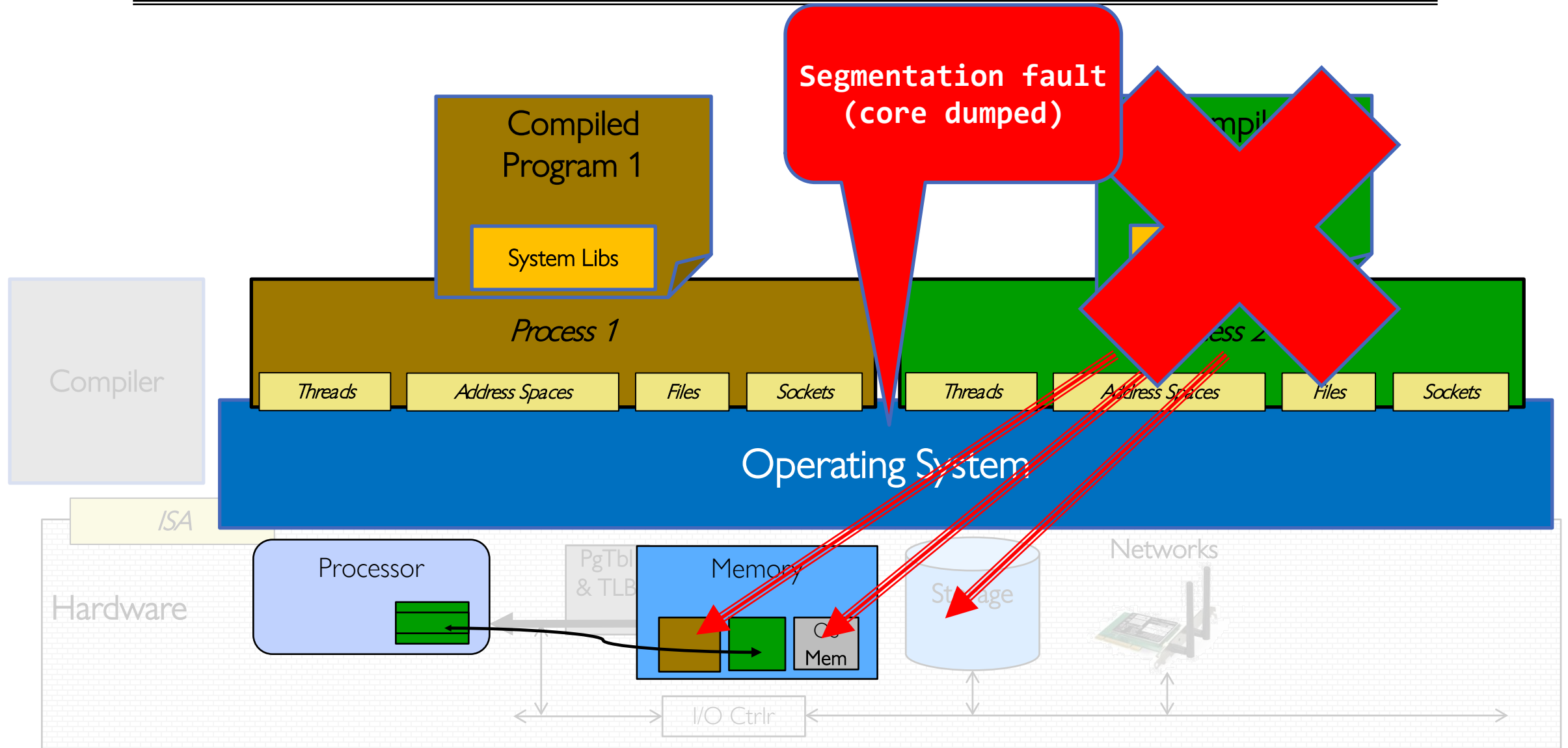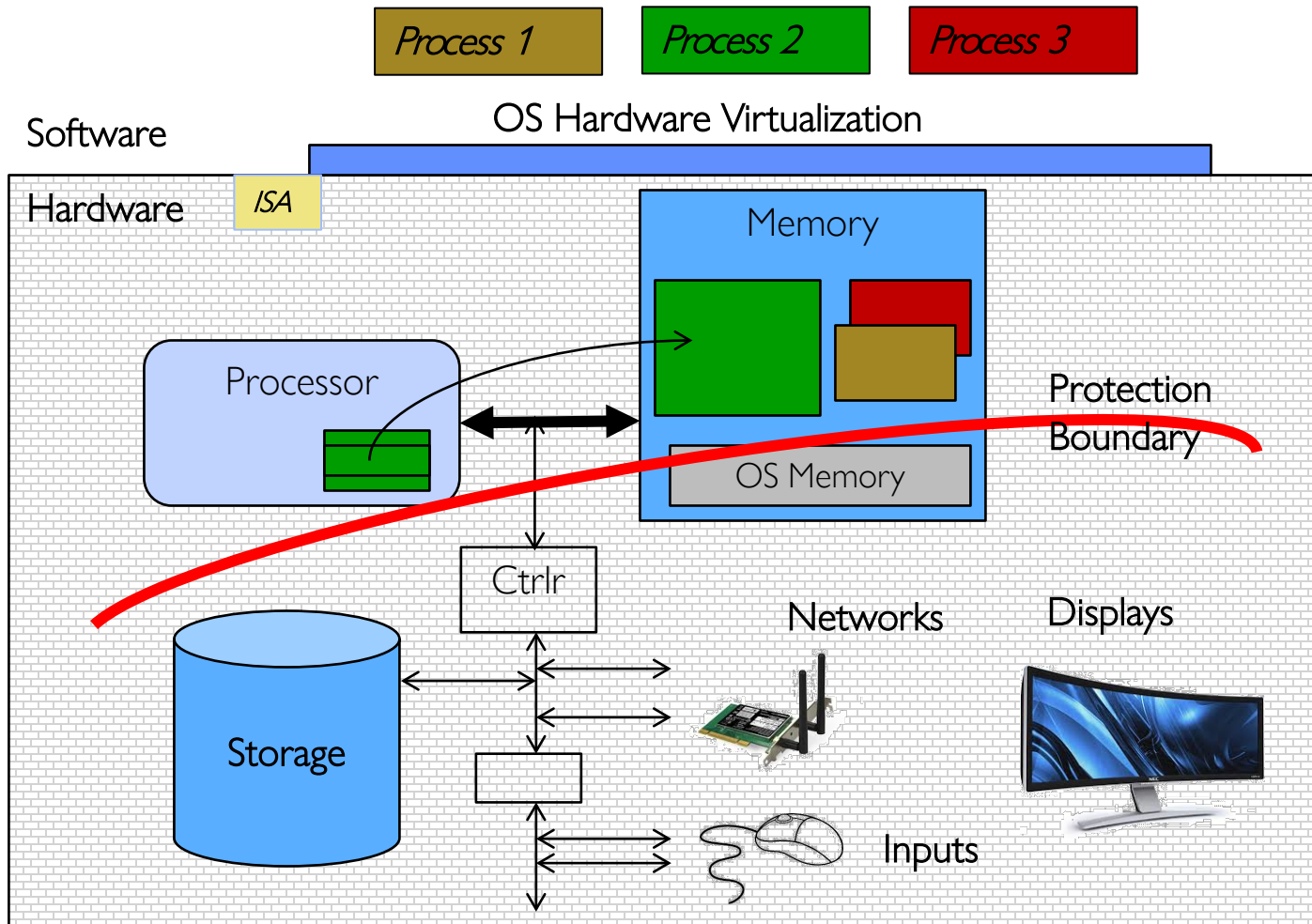
- OS *isolates* processes from each other

- OS *isolates* itself from other processes

- … even though they are actually running on the same hardware!

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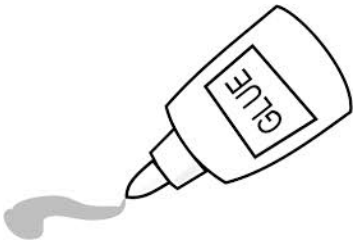
- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    » Infinite memory, dedicated machine
    » Higher level objects: files, users, messages
    » Masking limitations, virtualization
- Referee
  - Manage protection, isolation, and sharing of resources
    » Resource allocation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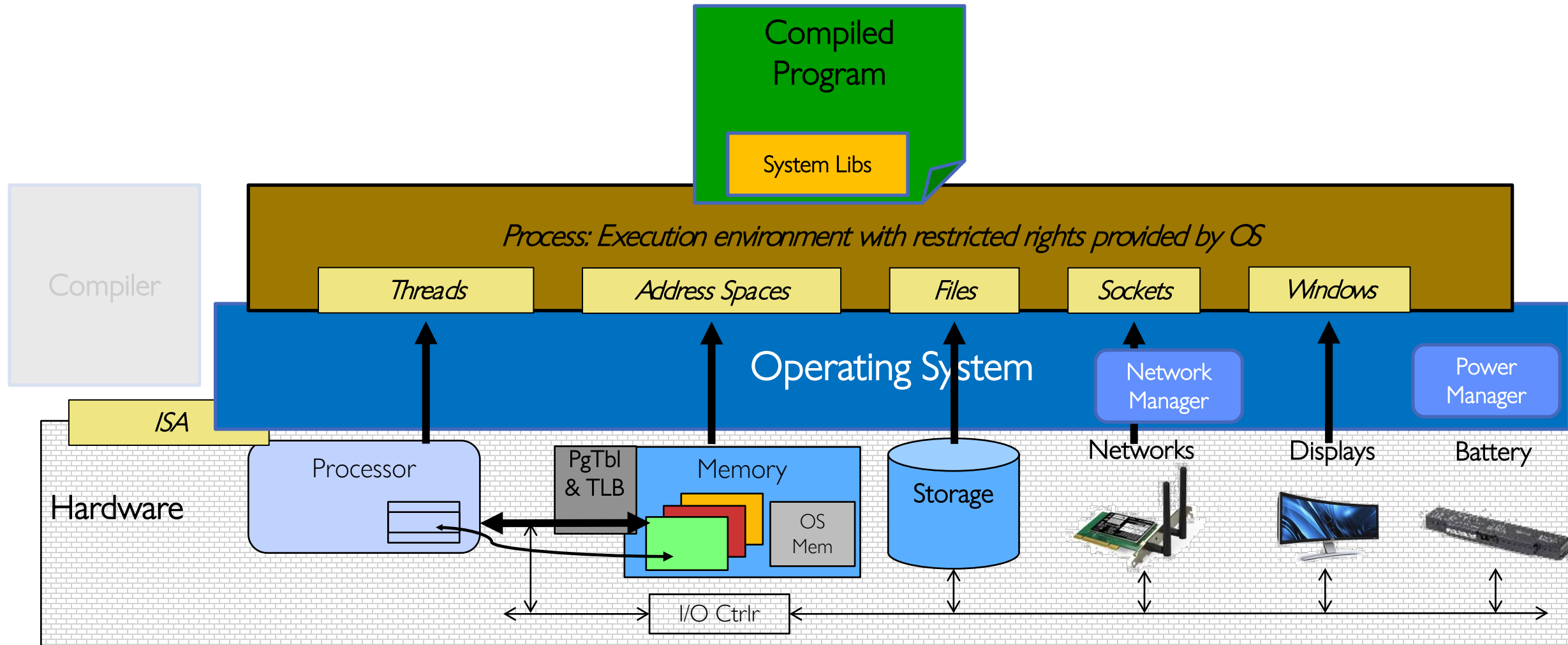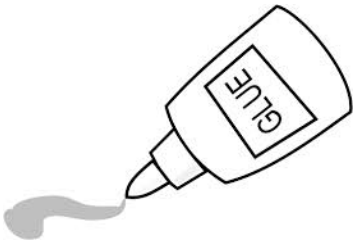   » Storage, Window system, Networking
    » Sharing, Authorization
    » Look and feel

OS Basics: I/O

Process 1   Process 2   Process 3

Software

OS Hardware Virtualization

Hardware   ISA

Memory

Processor

Protection
Boundary

OS Memory

Ctrlr

Storage

Networks   Displays

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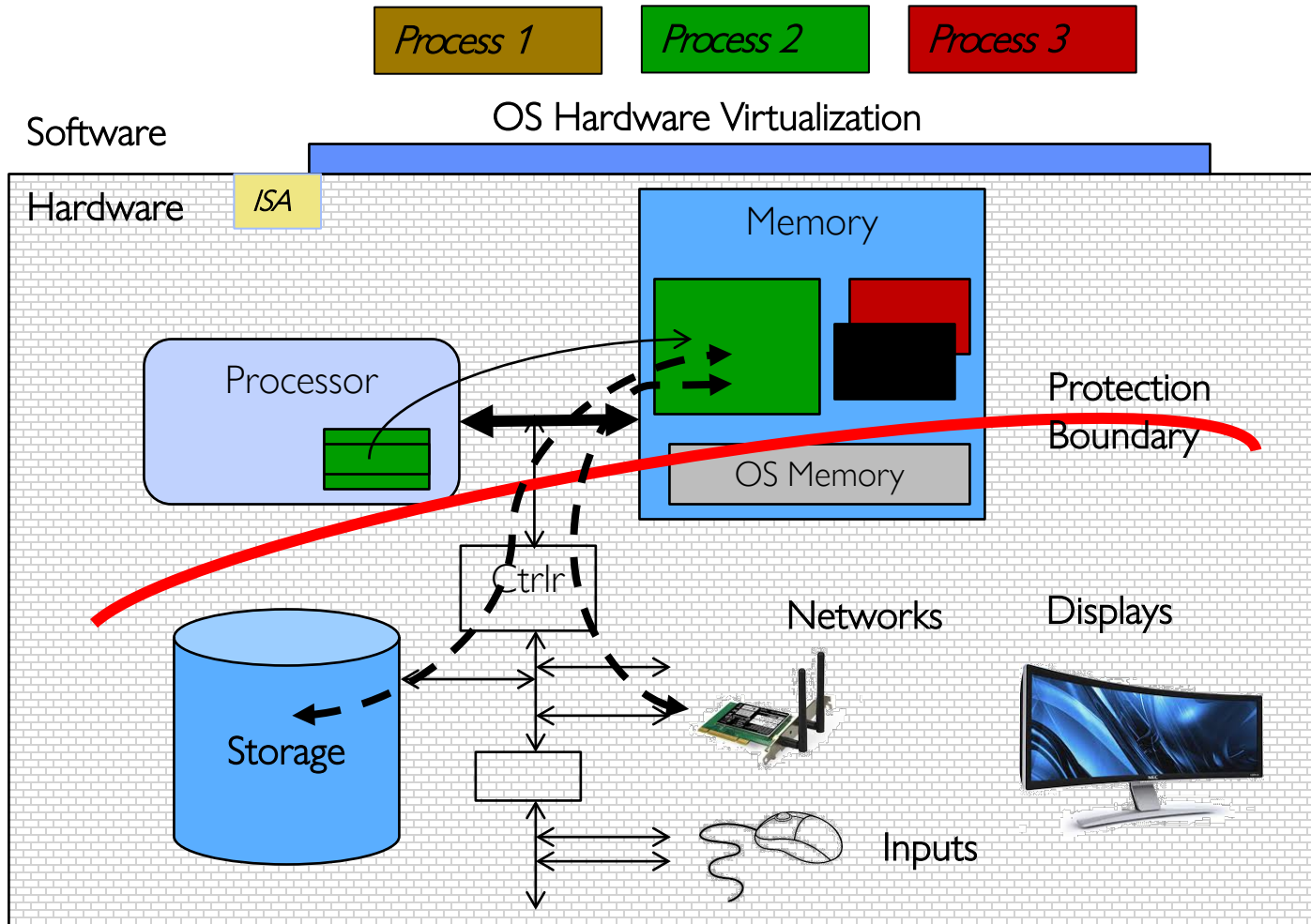
- OS provides common services in the form of I/O

OS Basics: Look and Feel

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Threads | Address Spaces | Files | Sockets | Windows

Operating System

Compiler

ISA

Hardware

Processor

PgTbl & TLB | Memory | OS Mem

Storage

Networks

Displays

I/O Ctrlr
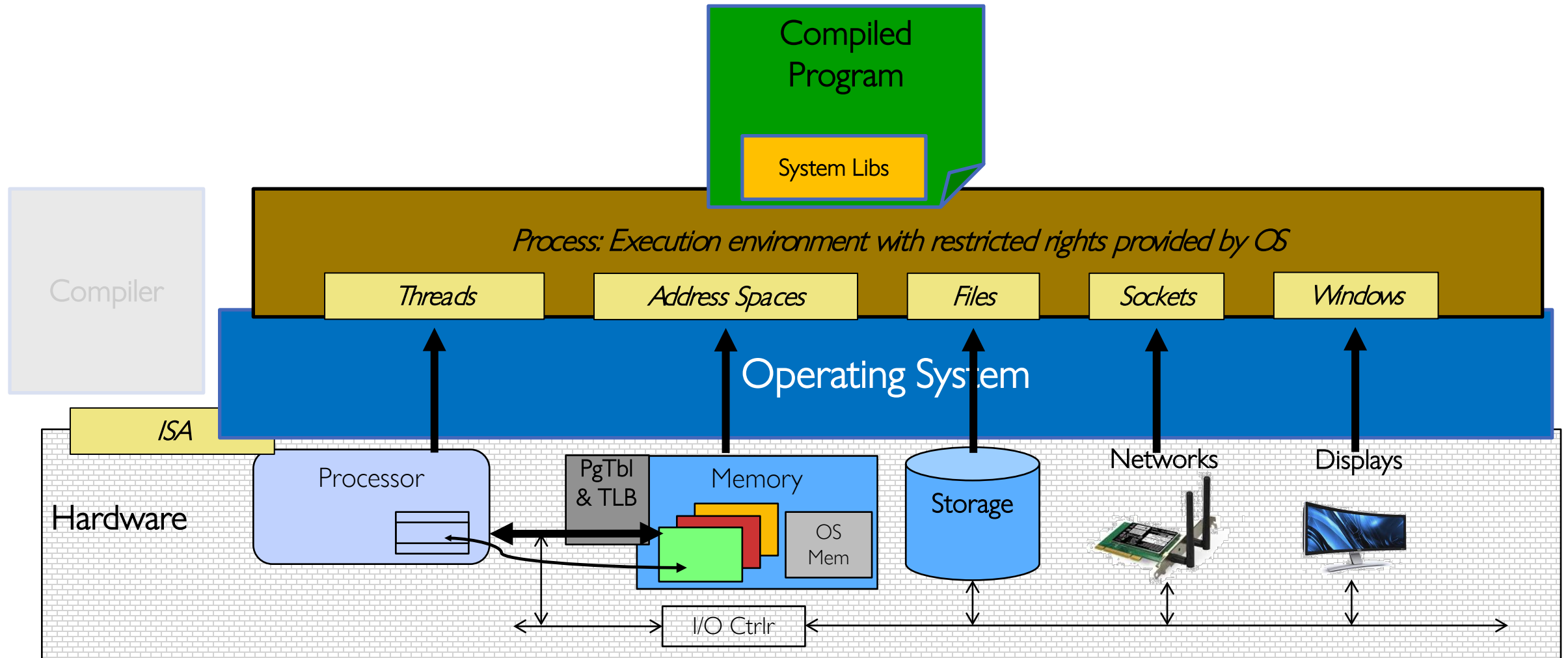
OS Basics: Background Management

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiler

Threads | Address Spaces | Files | Sockets | Windows

Operating System

ISA

Network Manager

Power Manager

Hardware

Processor

PgTbl & TLB | Memory | OS Mem

Storage

Networks | Displays | Battery

I/O Ctrlr

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 CS112?

- 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?

Massive Cluster

Gigabit Ethernet

Clusters

Massive Cluster

Gigabit Ethernet

Clusters

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

Internet
Connectivity

Scalable, Reliable,
Secure Services

Databases
Information Collection
Remote Storage
Online Games
Commerce
…

MEMS for
Sensor Nets
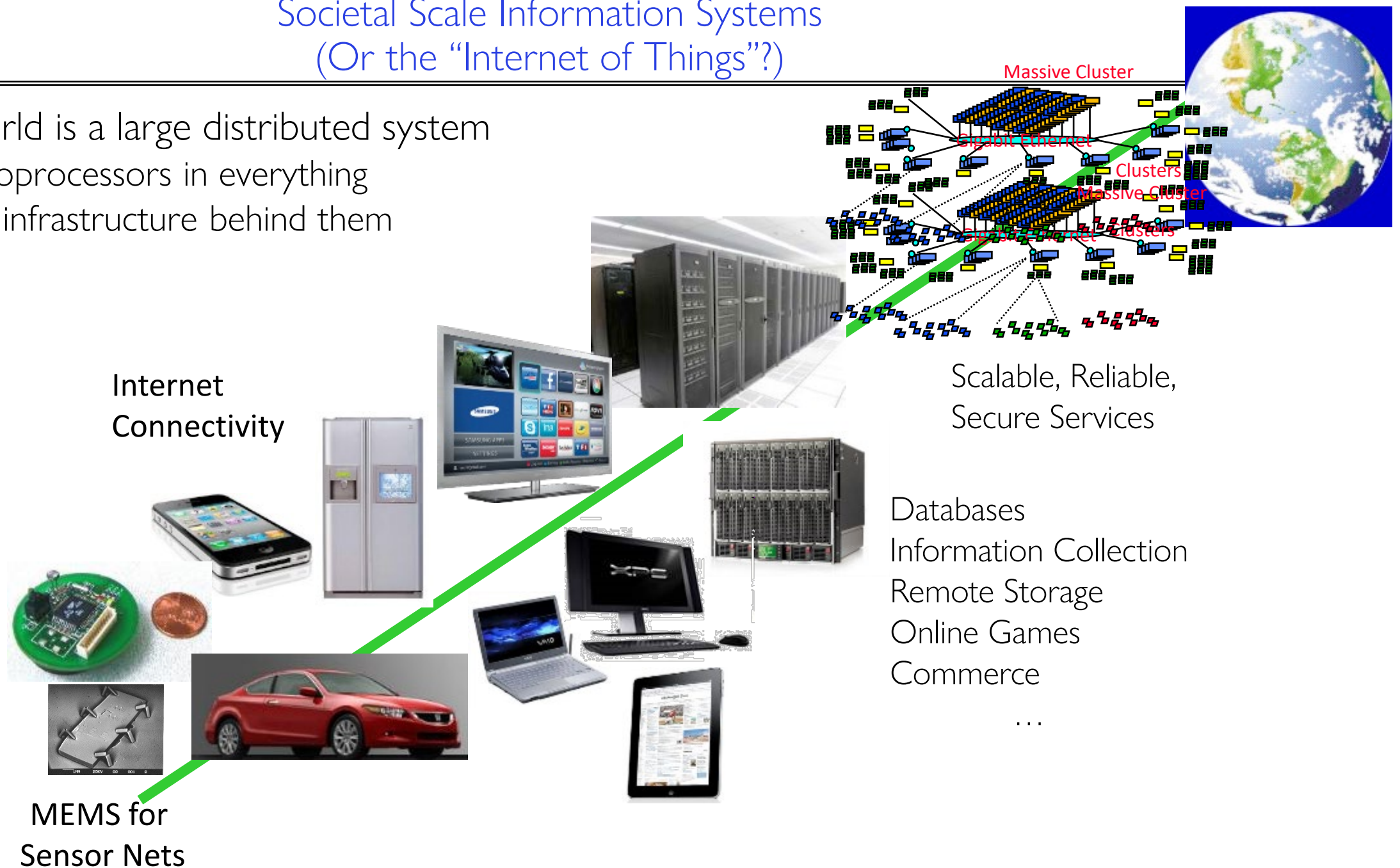
Technology Trends: Moore's Law

2X transistors/Chip Every 1.5 years
Called "Moore's Law"
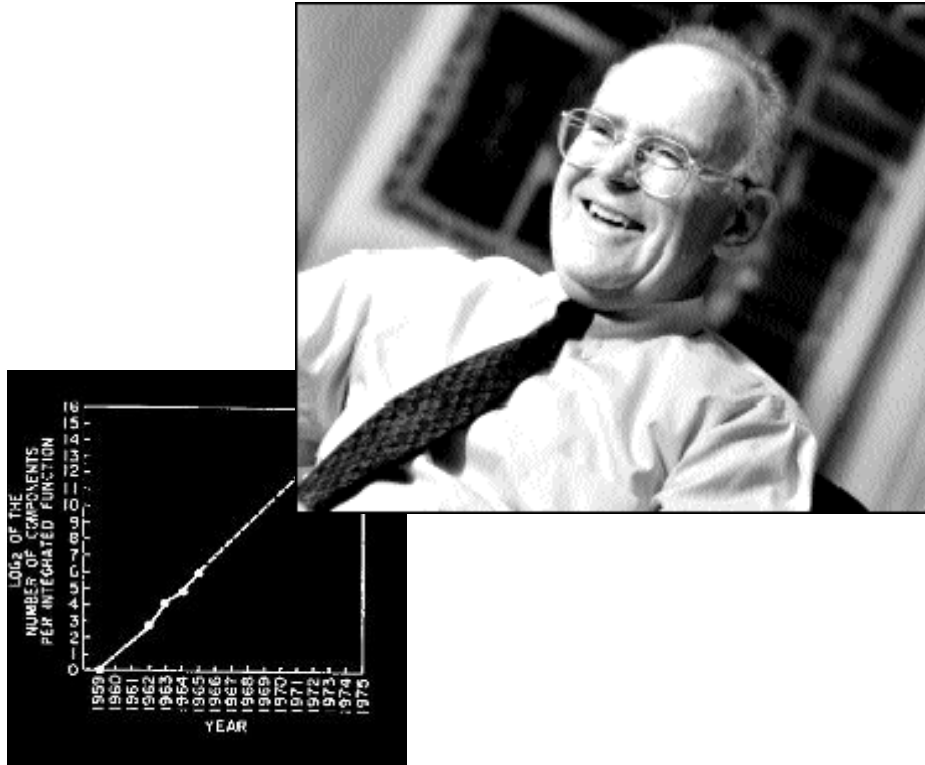
Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months

Microprocessors have become smaller, denser, and more powerful

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

Performance (vs. VAX-11/780)

10000

1000

100

10

1

25%/year

52%/year

3X

??%/year

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

1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006
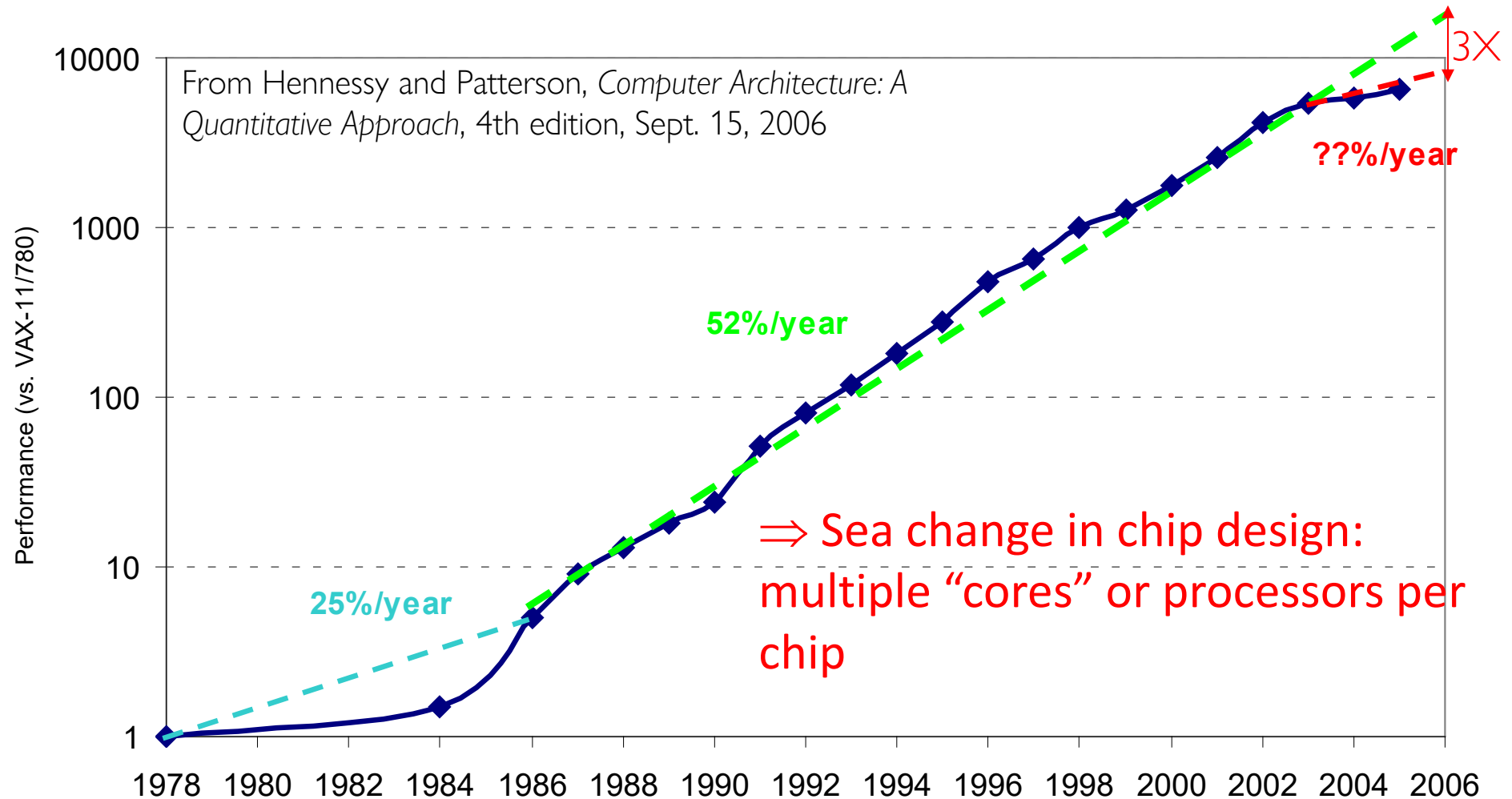
- VAX   : 25%/year 1978 to 1986
- RISC + x86 : 52%/year 1986 to 2002
- RISC + x86 : ??%/year 2002 to present

Another Challenge: Power Density

Source: S. Borkar (Intel)
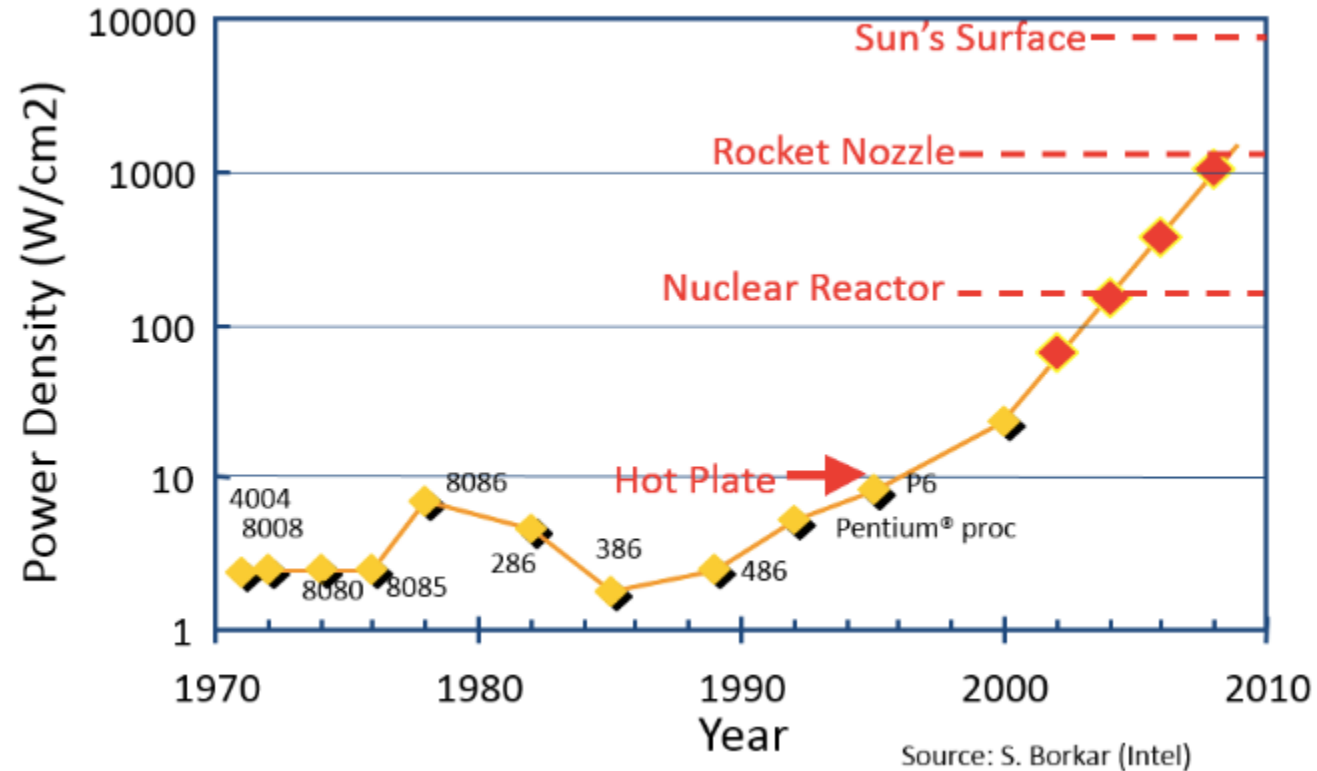
- Moore's law extrapolation

  – 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
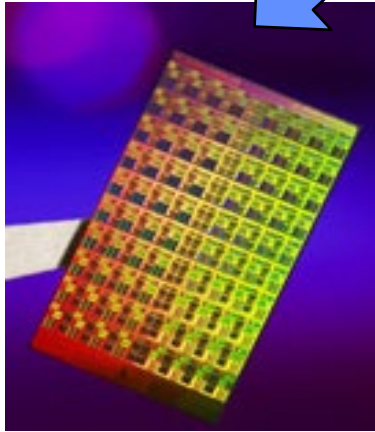
- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers
  - Hardware support for message-passing
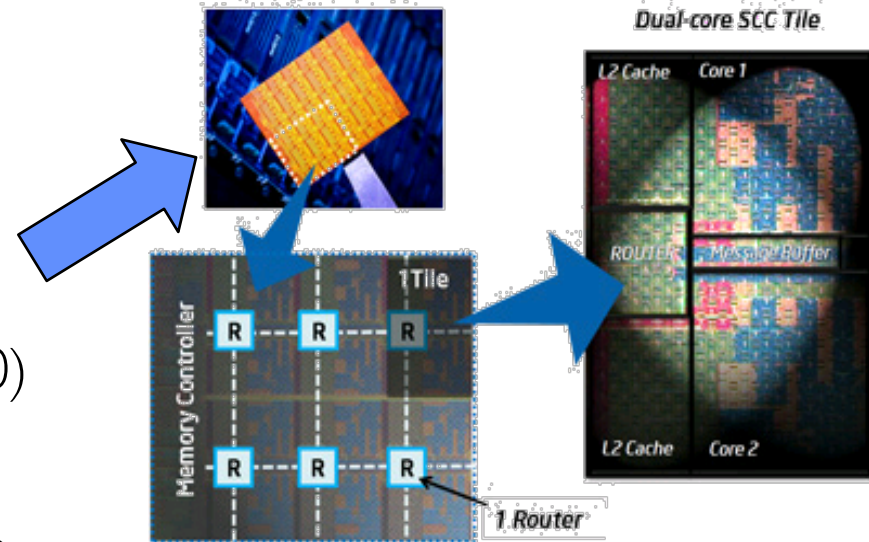
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels
- Amazon X1 instances (2016)
  - 128 virtual cores, 2 TB RAM
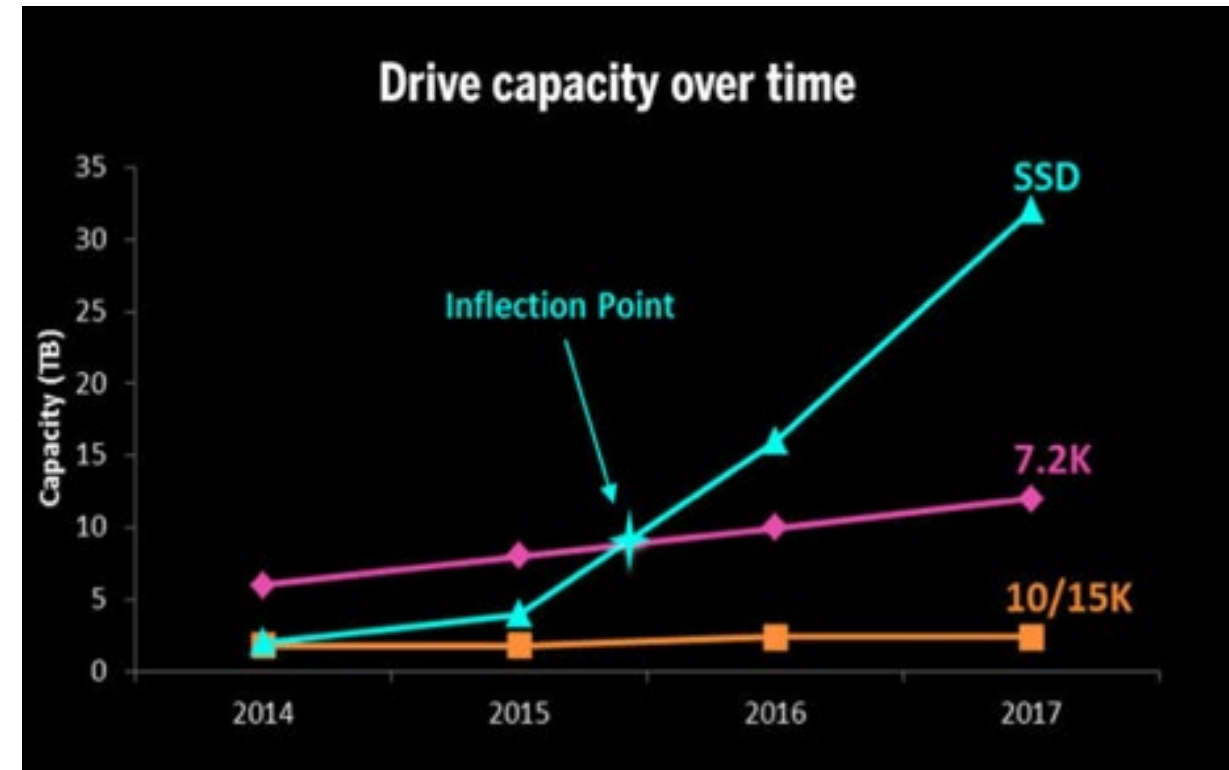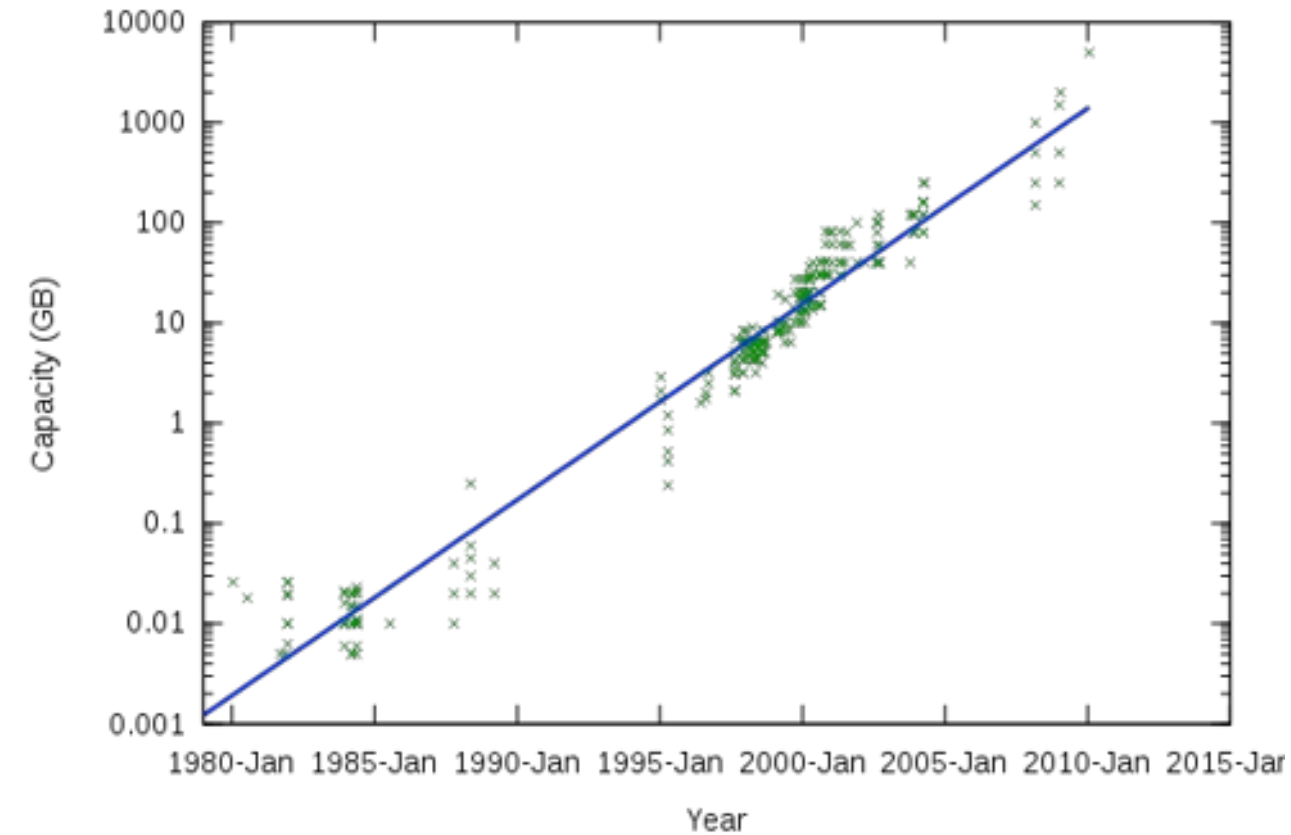
But then Moore's Law Ended...
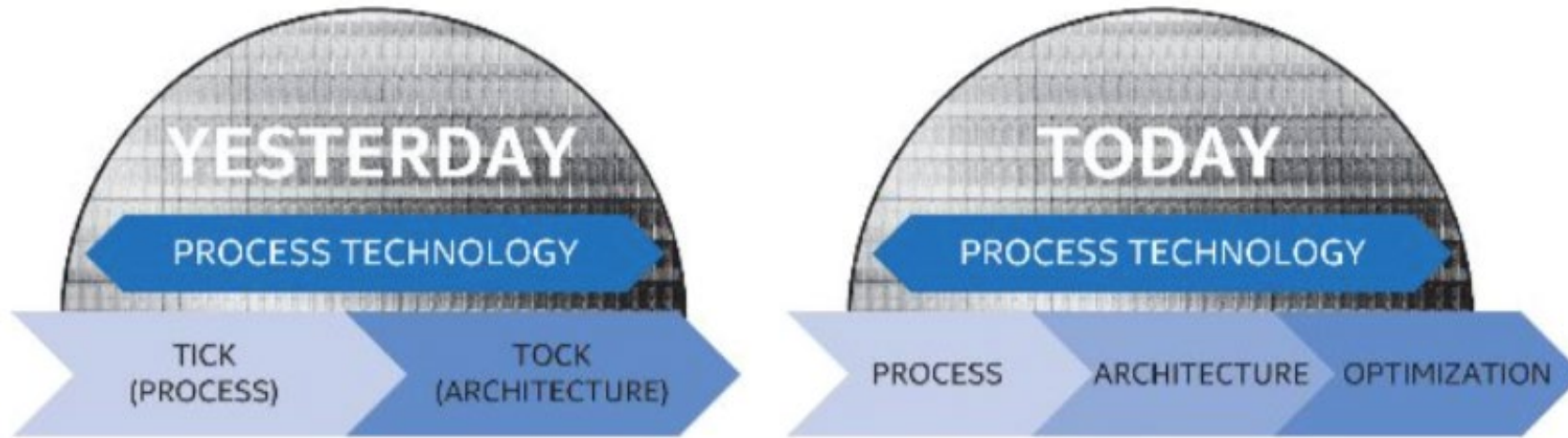
- Moore's Law has (officially) ended -- Feb 2016
  - No longer getting 2 x transistors/chip every 18 months…
  - or even every 24 months
- May have only 2-3 smallest geometry fabrication plants left:
  - Intel and Samsung and/or TSMC
- Vendors moving to 3D stacked chips
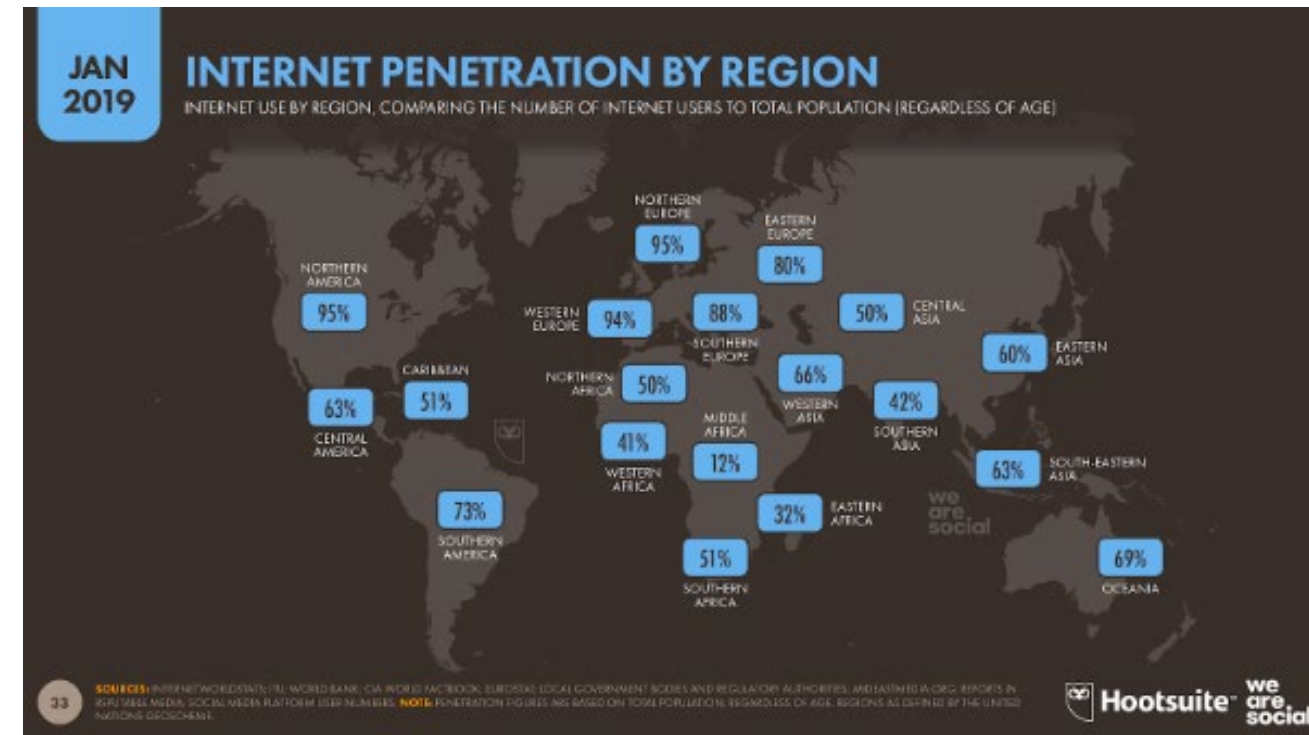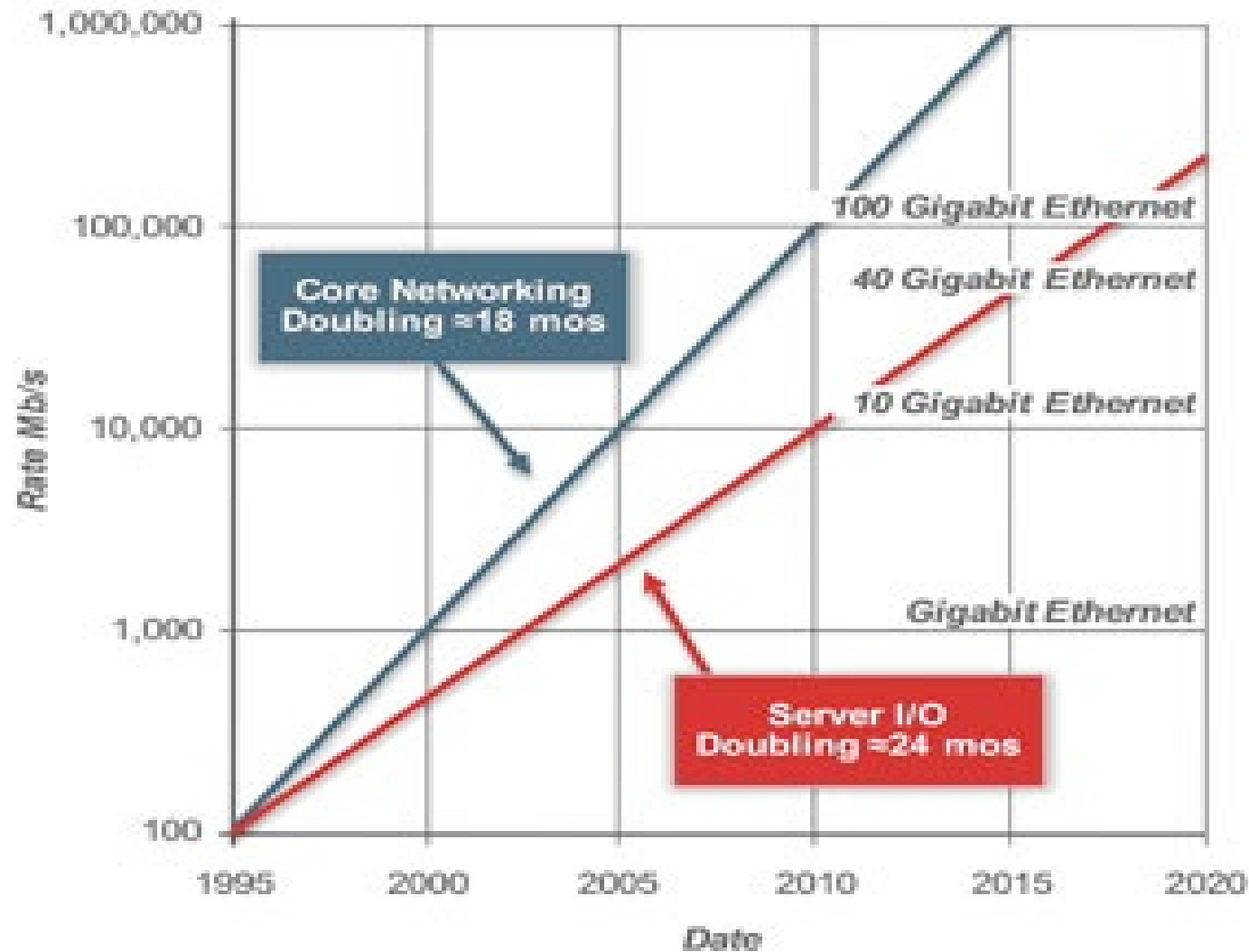 - More layers in old geometries

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
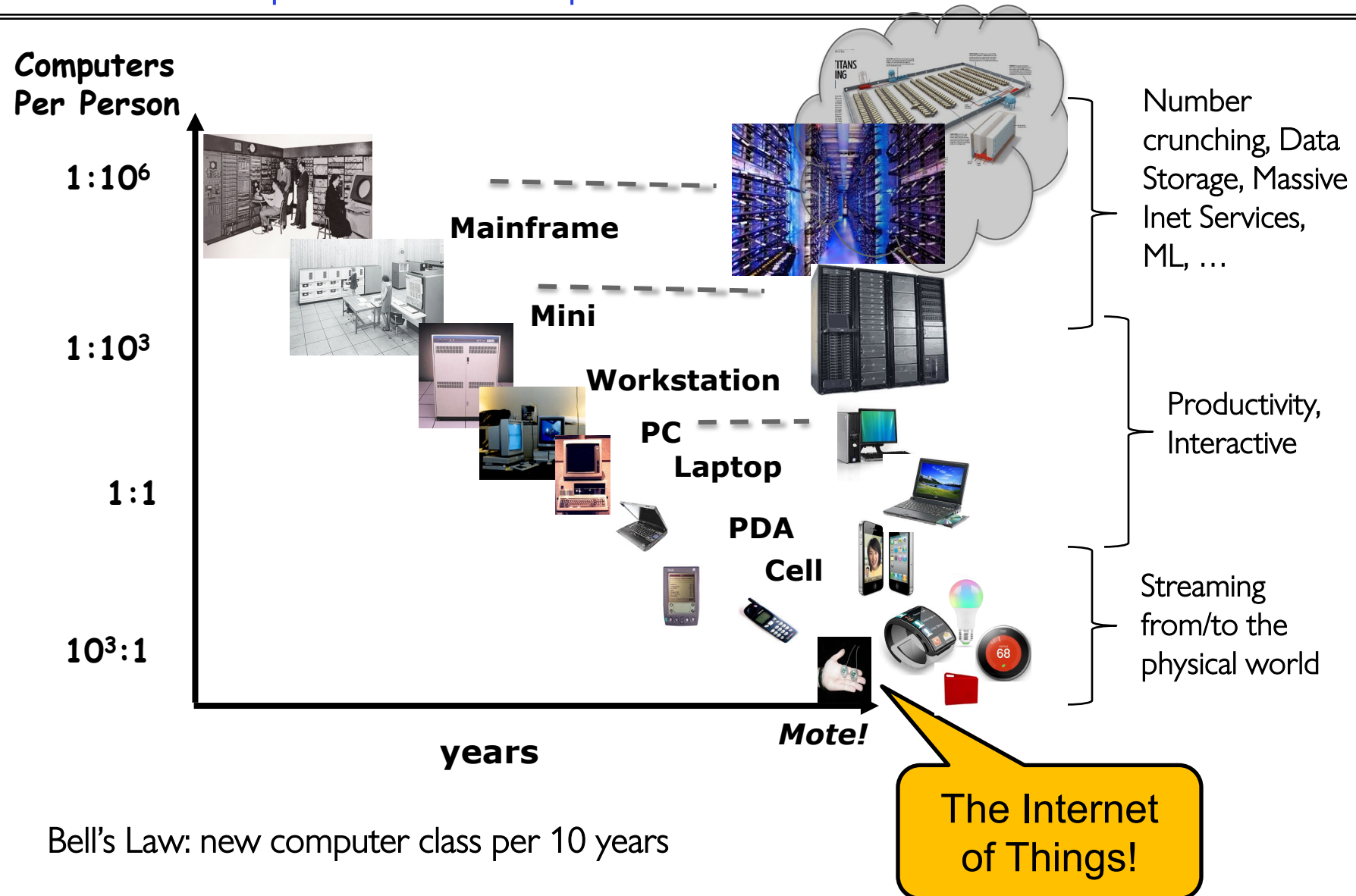   » 63M tablets
  - 25M smart TVs

1.53B in 2017

262.5M in 2017

164M in 2017

39.5M in 2017

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

People-to-Computer Ratio Over Time

Computers Per Person

$1:10^6$

$1:10^3$

$1:1$

$10^3:1$

Mainframe

Mini

Workstation

PC

Laptop

PDA

Cell

Mote!

years

Number crunching, Data Storage, Massive Inet Services, ML, …

Productivity, Interactive

Streaming from/to the physical world

The Internet of Things!

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
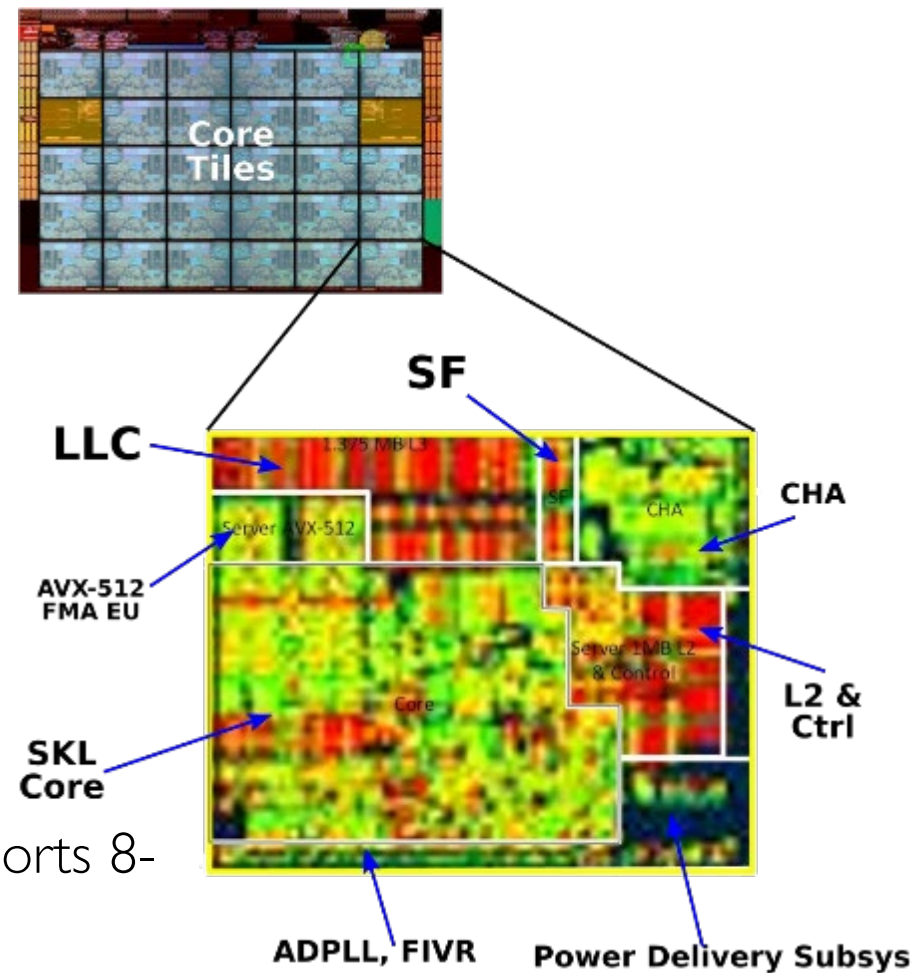   - » Sharing, Authorization
    - » Look and feel

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

Increasing Software Complexity

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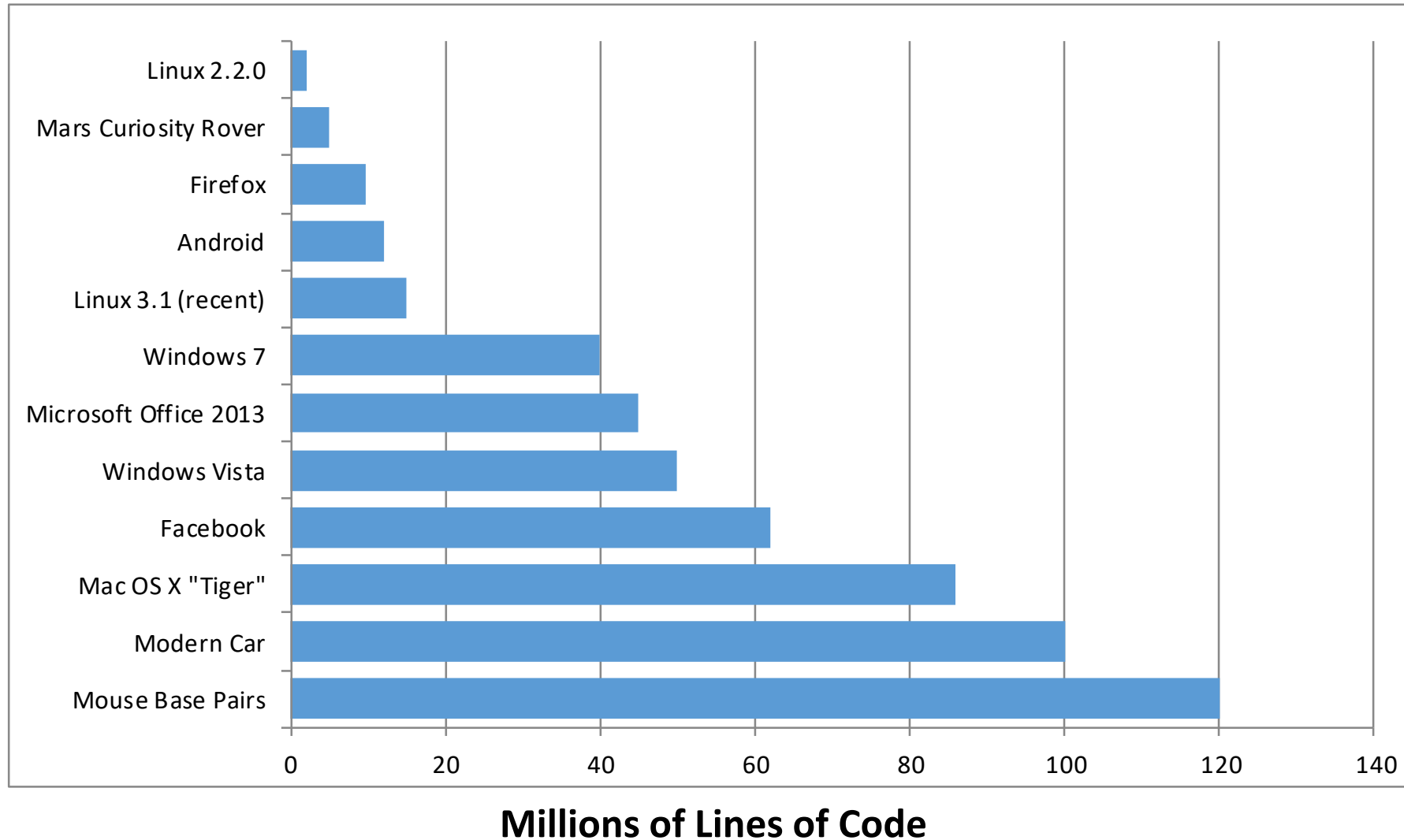
(source https://informationisbeautiful.net/visualizations/million-lines-of-code/)

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
  - One buggy module should not crash critical modules, e.g., 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?

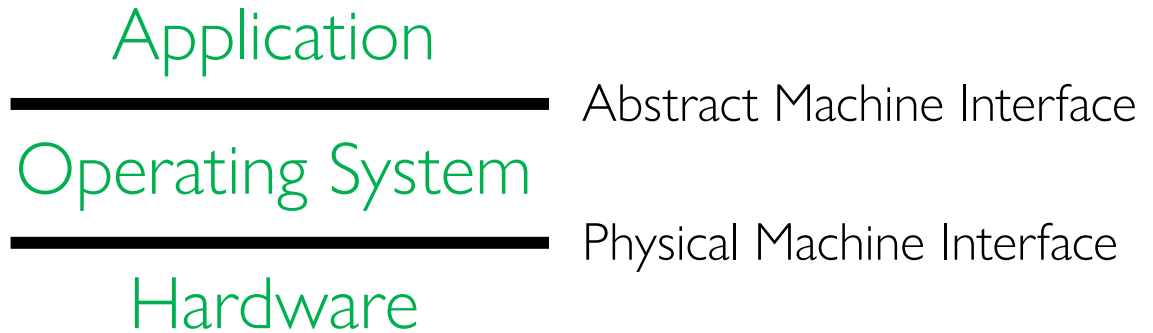- Does every program have access to all hardware?

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

Application

Operating System

Hardware

Abstract Machine Interface

Physical Machine Interface

- 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)
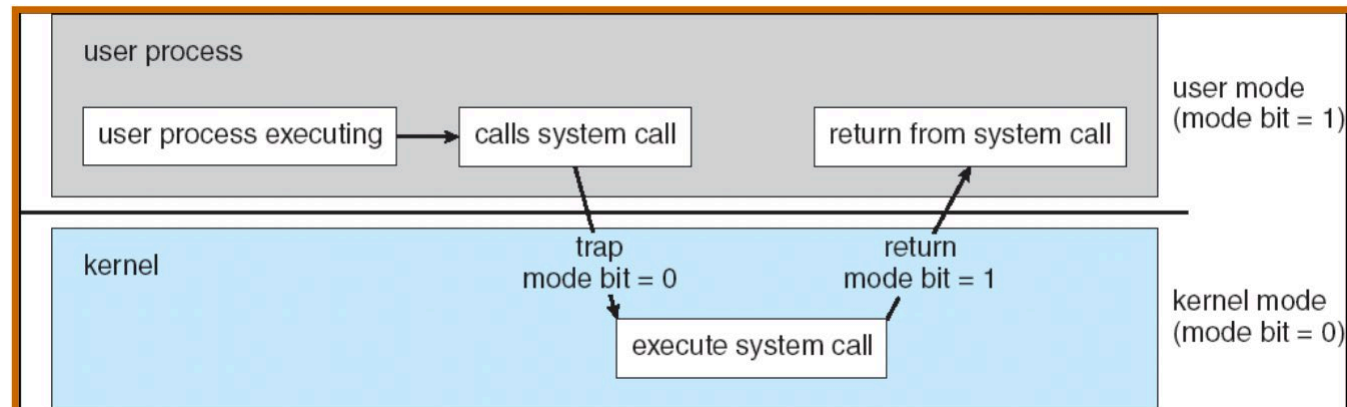
OS *Protects* Processes and the Kernel

- Run multiple applications and:
  - Keep them from interfering with or crashing the operating system
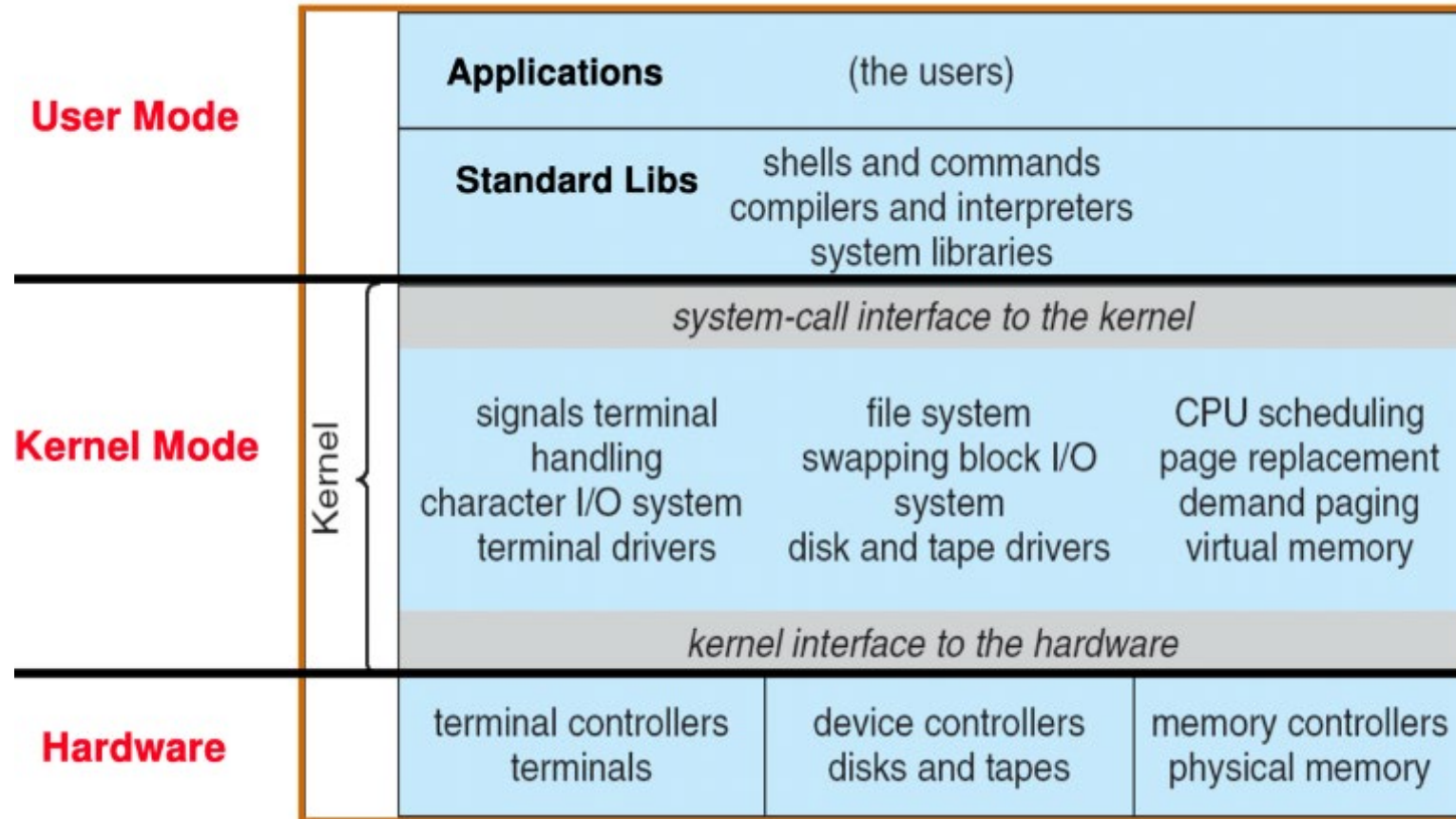 - Keep them from interfering with or crashing each other
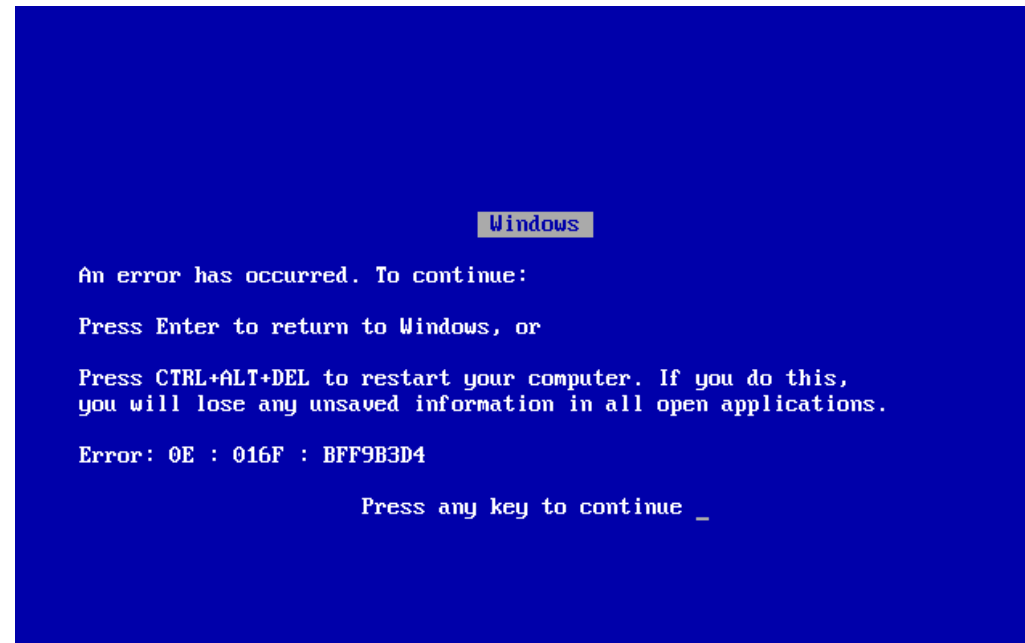
Basic Tool: Dual-Mode Operation
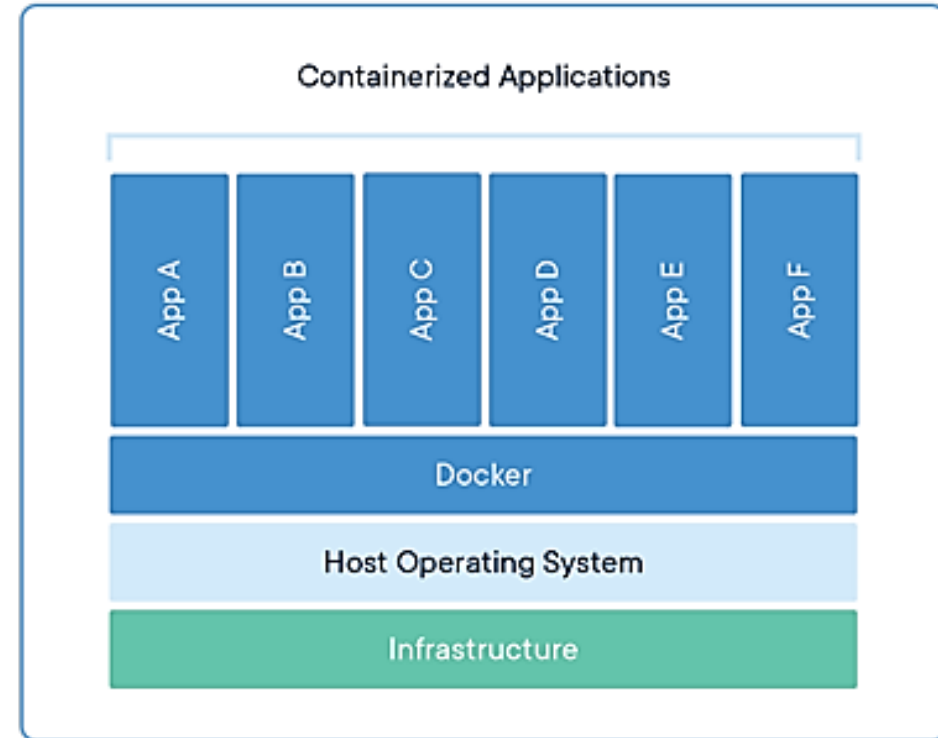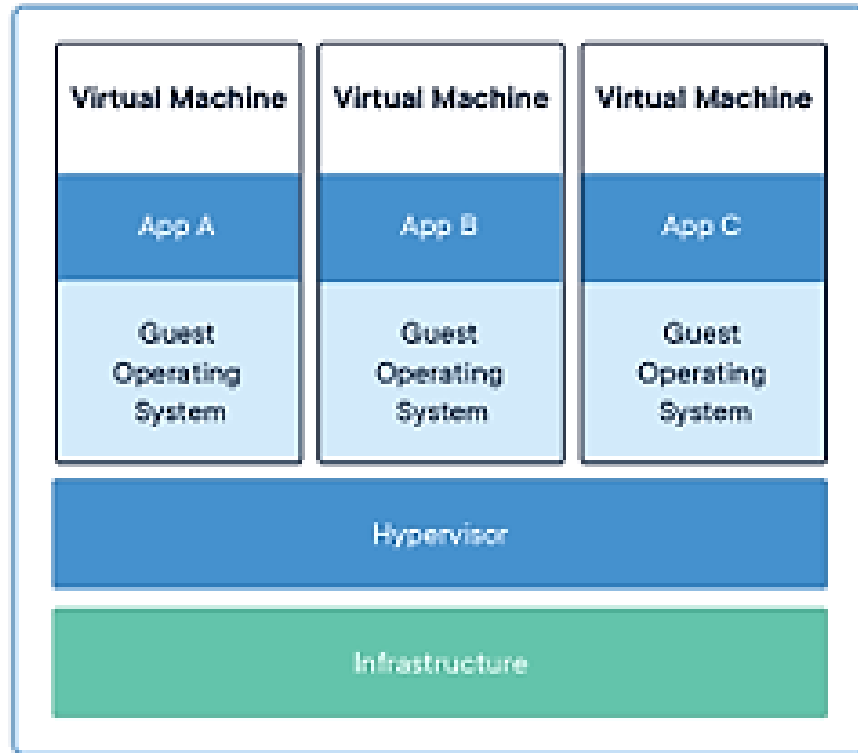
- Hardware provides at least two modes:
  1. Kernel Mode (or "supervisor" mode)
  2. User Mode
- Certain operations are **prohibited** when running in user mode
  – Changing the page table pointer, disabling interrupts, interacting directly w/ hardware, writing to kernel memory
- Carefully controlled transitions between user mode and kernel mode
  – System calls, interrupts, exceptions

UNIX System Structure

User Mode		Applications	(the users)	
		Standard Libs	shells and commands compilers and interpreters system libraries	

Kernel Mode — Kernel:
- system-call interface to the kernel

signals terminal handling character I/O system terminal drivers	file system swapping block I/O system disk and tape drivers	CPU scheduling page replacement demand paging virtual memory

- kernel interface to the hardware

Hardware	terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory

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?

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