CS 310 Operating Systems

Lecture 2: Operating System Concepts - 1

Ravi Mittal

IIT Goa

Acknowledgements !

- Contents of this class presentation have been taken from various sources. Thanks are due to the original content creators:
  - Class presentation: University of California, Berkeley: David Culler, Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.
- Book: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin, Volume 1

Read the following:

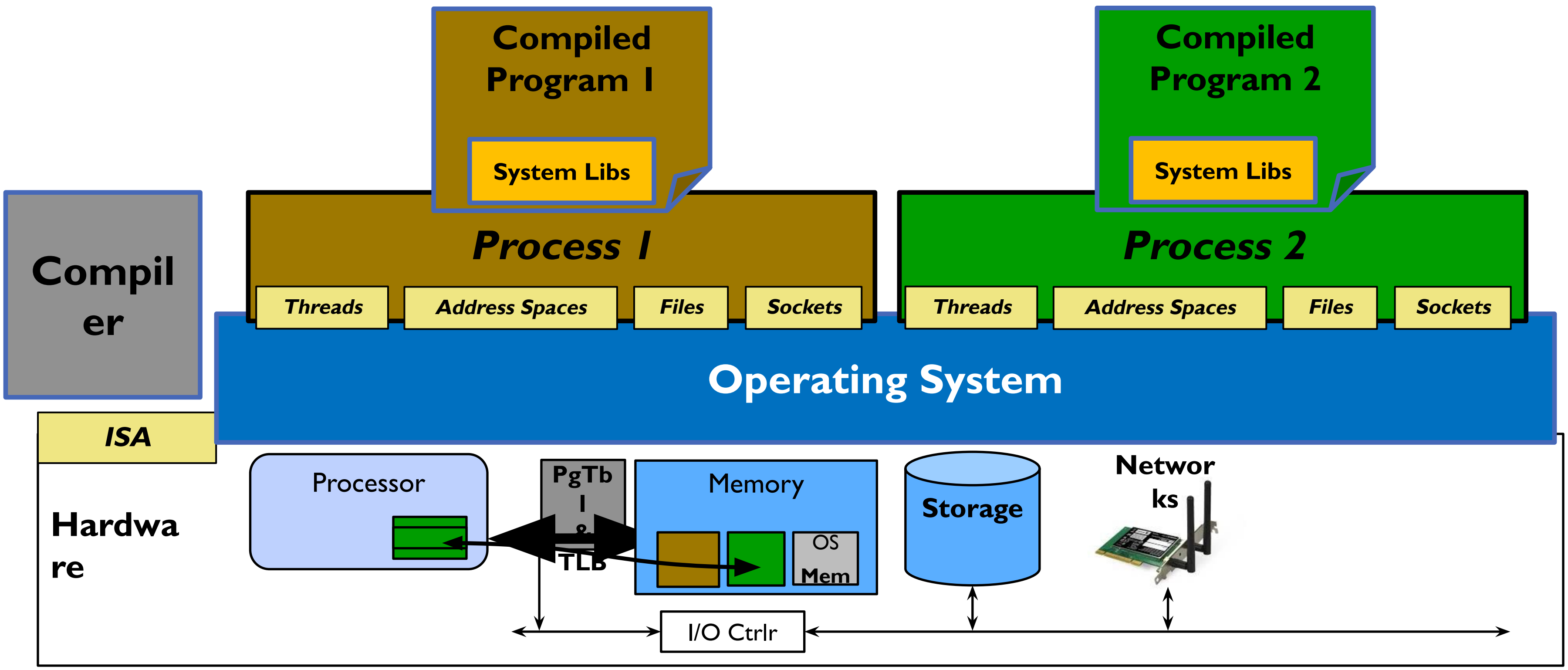
- Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
  - Volume 1, Kernel and Processes
    - Chapter 2

We will study..

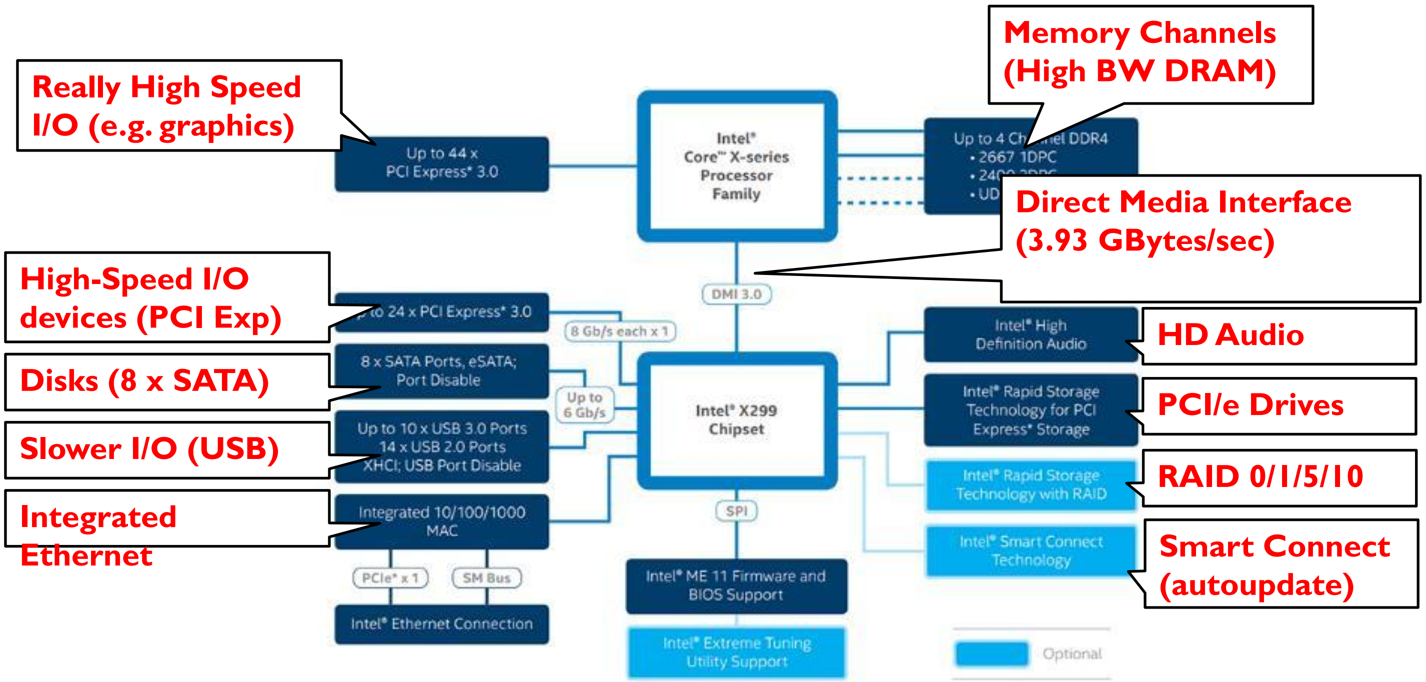
- Introduction
- The first concept: Processes
- The Second Concept: Address Space
- The Third Concept: Threads
- The Fourth Concept: Dual Mode of Operation

Last Class

Recall: OS Protection

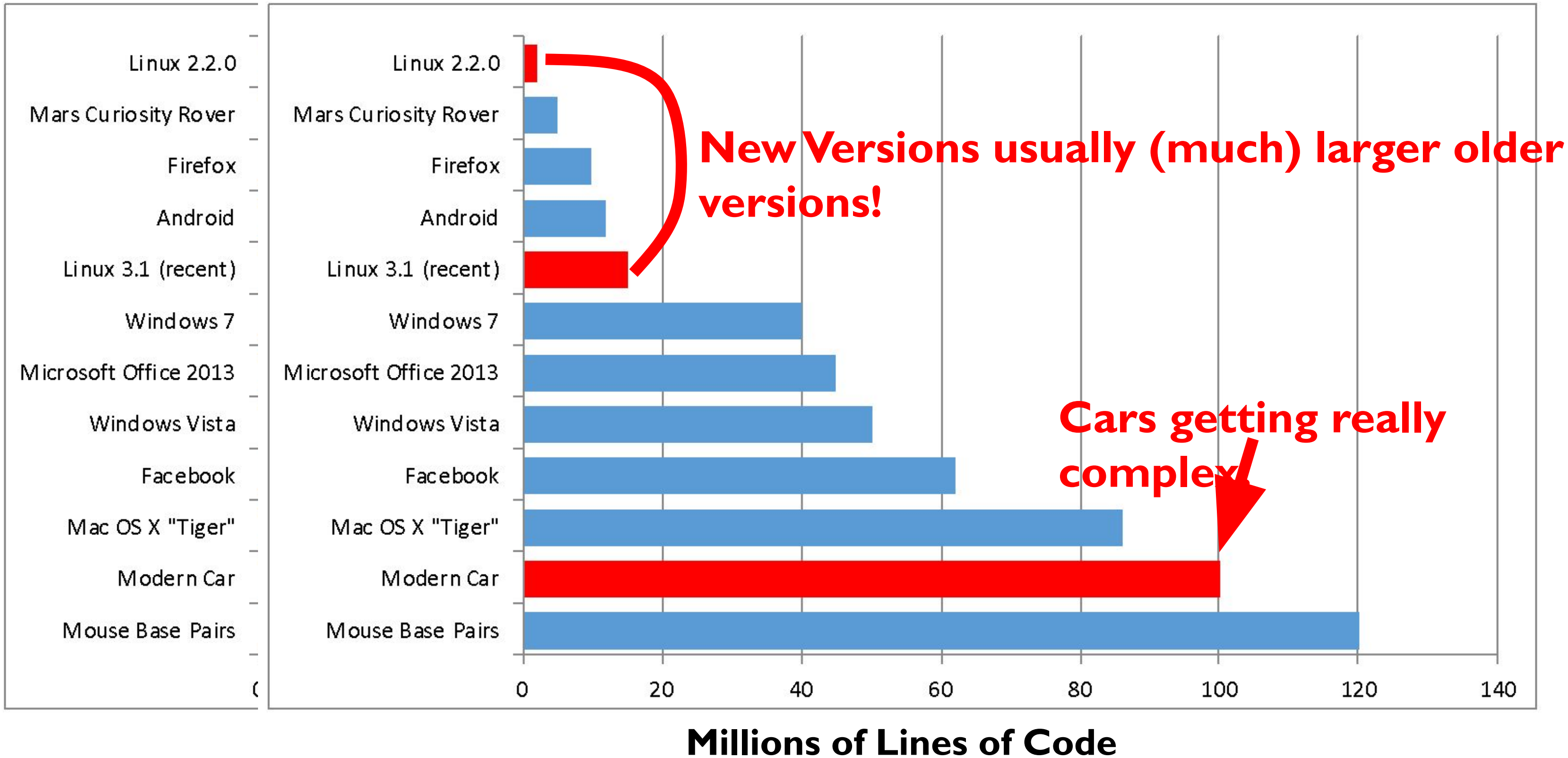


Recall: HW Functionality ⇒ great complexity!



Intel Skylake-X I/O Configuration

Recall: Increasing Software Complexity



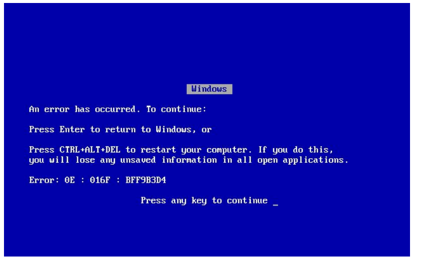
(source https://informationisbeautiful.net/visualizations/million-lines-of-code/)

Linux 5.3 has approx. 27 million lines of code

Complexity leaks into OS if not properly designed

- Buggy device drivers

- Holes in security model or bugs in OS lead to instability and privacy breaches
  - Meltdown (2017)
  - Spectre (2017)



OS Abstracts Underlying Hardware to help Tame Complexity

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

Application

Abstract Machine Interface

Operating System

Physical Machine Interface

Hardware

Design Goals of an Operating System

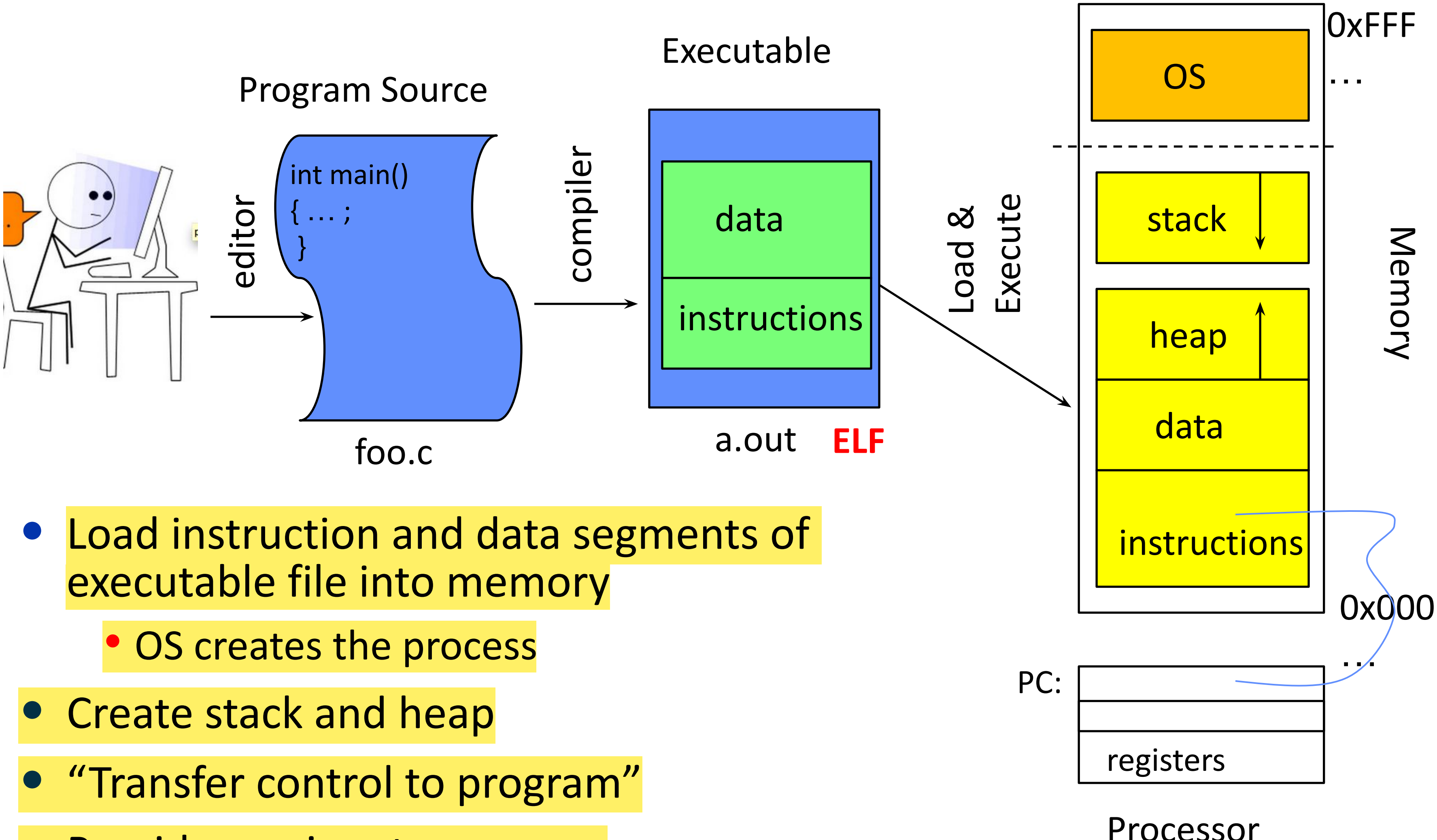
- Abstraction of the hardware resources
- Convenience of users
- Efficiency of usage of CPU, Memory etc
- Isolation between multiple processes

Four important concepts..

Four Fundamental OS Concepts

- **Process: an instance of a running program**
  - Protected Address Space + One or more Threads
- **Address space**
  - Set of memory addresses accessible to program (for read or write)
- **Thread: Execution Context**
  - Fully describes program state
- **Dual mode operation / Protection**
  - User / Kernel mode

OS Bottom Line: Run Programs



- Load instruction and data segments of executable file into memory
  - OS creates the process
- Create stack and heap
- "Transfer control to program"
- Provide services to program
- While protecting OS and program

ELF: Executable and Linking Format

Operating System Needs are changing (1)

- Network Speeds:
  - 1980: 300 bps /$
  - 2000: ~ 256 Kbps /$
  - 2020: ~ 20 Mbps/$
- So, in the past 40 years, the speed of networking has increased by a factor of ~ 67000 times
- We now have Internet connection of 1 Gbps

Operating System Needs are changing (2)

- Number of Cores / CPU
  - 1980: 1 core / CPU
  - 2000: 1 core / CPU
  - 2020: 8+ cores / CPU 64 cores / Server-CPU
- So, in the past 20 years, the number of cores have become 10 times or more
- How OS handles tasks on so many cores?
  - Process/job management
  - Power Management
  - Scheduling, Availability, Security

Operating System Needs are changing (3)

- **Cost per megaflop/sec:**
  - 1980: ~ $100,000 /megaflop/sec
  - 2000: ~ $25 /megaflop/sec
  - 2020: ~ $0.20 /megaflop/sec
- So, in the past 40 years, the cost of per million operations has decreased by a factor of ~ 500,000 times

Operating System Needs are changing (4)

- **RAM Capacity B/$:**
  - 1980: ~ 2 KiB /$
  - 2000: ~ 2 MiB /$
  - 2020: ~ 2 GiB /$
- So, in the past 40 years, the cost per byte of RAM has decreased by a factor of ~ 1,000,000 times

Operating System Needs are changing (5)

- **Ratio of Computers to Users**
  - 1980: ~ 100 users : 1 computer
  - 2000: ~ 1 user : 1 computer
  - 2020: ~ 1 user : many computers
- So, in the past 40 years, the number of users to computer ratio has increased by a factor of at least 200 times
- Paradigm shift in which OS are designed for evolving computing technology

Operating System Challenges

- Reliability
- Availability
- Security
- Privacy
- Portability
- Performance

OS System needs are changing

Legacy Needs

- Runs one application at a time
- Manage time quotas for many users
- Uses submit jobs and get results days later

Modern Needs

- Multiprogramming across many cores and many concurrent users
- Interactive jobs; completing all jobs asap
- Optimize for users not for computer's resource time

Future Needs

- Manage and use an ever-increasing number of processors/computers
- Peta-scale storage, data centers, cloud
- Optimize for diversity

The first OS concepts - Process

What is a process ?

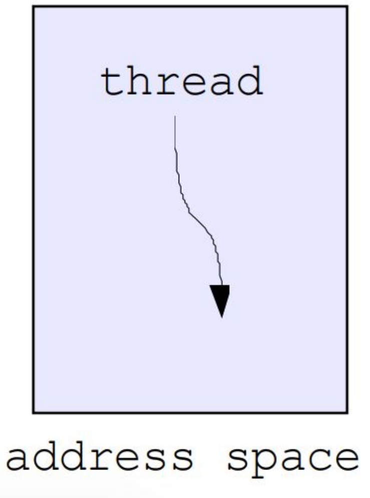
- A process is a program in execution
- The process is the OS's abstraction for execution
- Execution environment with restricted rights
  - (Protected) Address Space with One or More Threads
  - Owns memory (address space), file descriptors, sockets
  - Encapsulate one or more threads sharing process resources
- Simplest (classic) case: a sequential process
  - A single thread of execution (an abstraction of the CPU)
- A sequential process is
  - The unit of execution
  - The unit of scheduling
  - The dynamic (active) execution context
    - vs. the program – static, just a bunch of bytes

Process – Two key abstractions

- Process is not the binary source code
  - It is an instance of running program
- Process provides two key abstractions
  - Memory
    - Each process assumes entire system memory to itself
  - Execution
    - Provides abstraction of continued operation
      - There may be hundreds of processes in a system
      - Each process gets impression of continuous operation

What is a Process?

- Why processes?
  - Protected from each other! OS Protected from them
  - Processes provides memory protection



Protection and Isolation

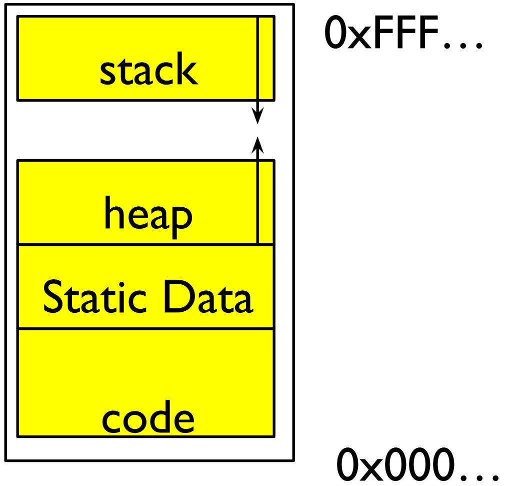
- Processes provide protection and isolation
  - Reliability: bugs can only overwrite memory of process they are in
  - Security and privacy: malicious or compromised process can't read or write other process' data

- Mechanisms:
  - Address translation: address space only contains its own data

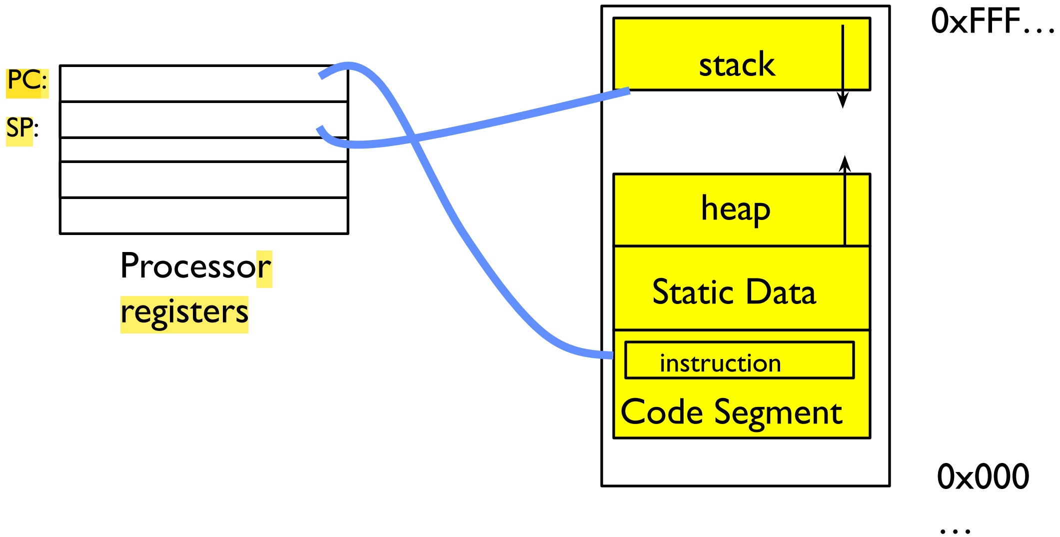
The Second OS concepts – Address Space

Second OS Concept: Address Space

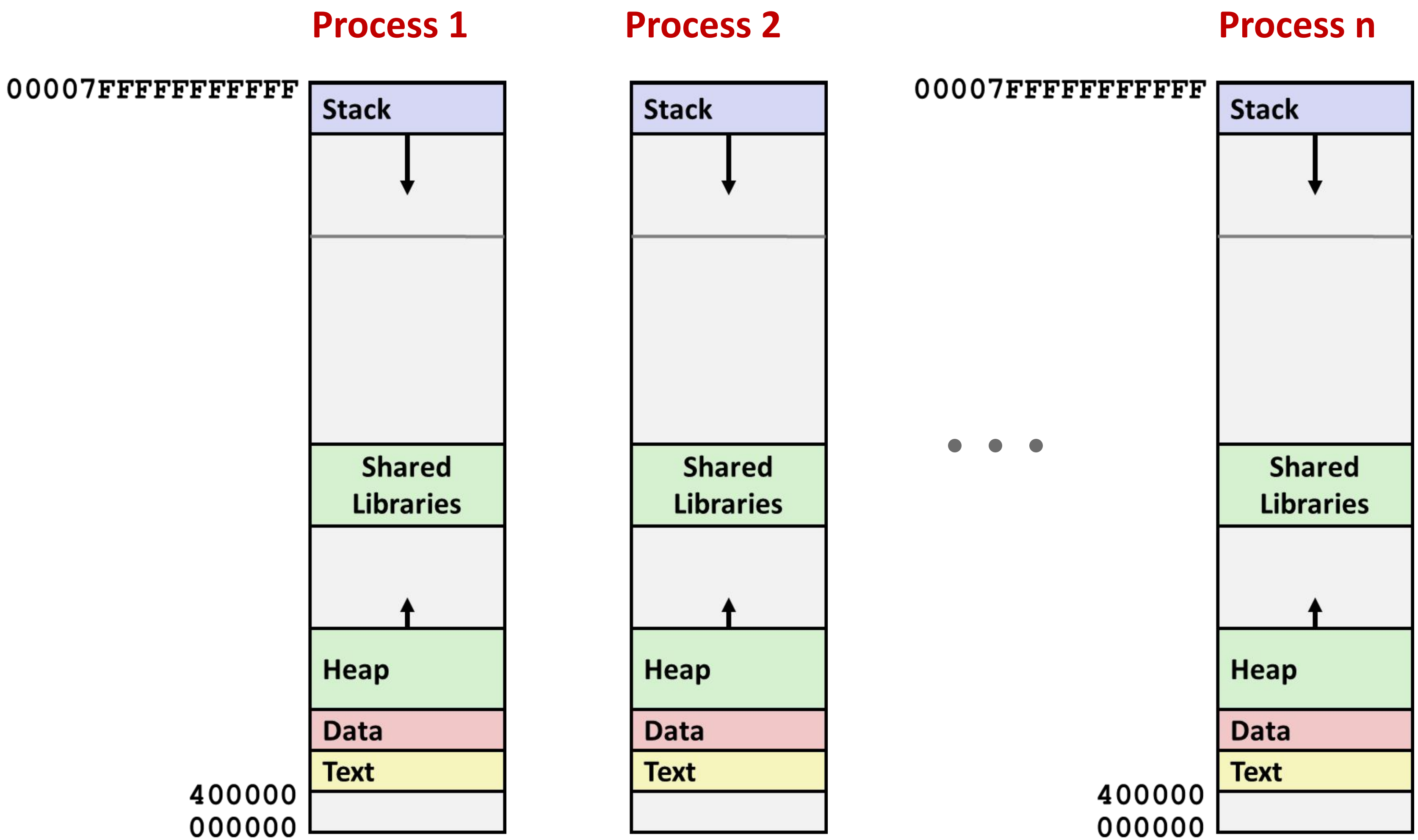
- Address space ⇒ the set of accessible addresses + state associated with them:
  - For 32-bit processor: 2^{32} = 4 billion (10^9) addresses
  - For 64-bit processor: 2^{64} = 18 quintillion (10^{18}) addresses
- The address space of a process contains all of the memory state of the running program
- What happens when you read or write to an address?
  - Perhaps acts like regular memory
  - Perhaps causes I/O operation
    - (Memory-mapped I/O)
  - Perhaps causes exception (fault)
  - Communicates with another program



Address Space: In a Picture

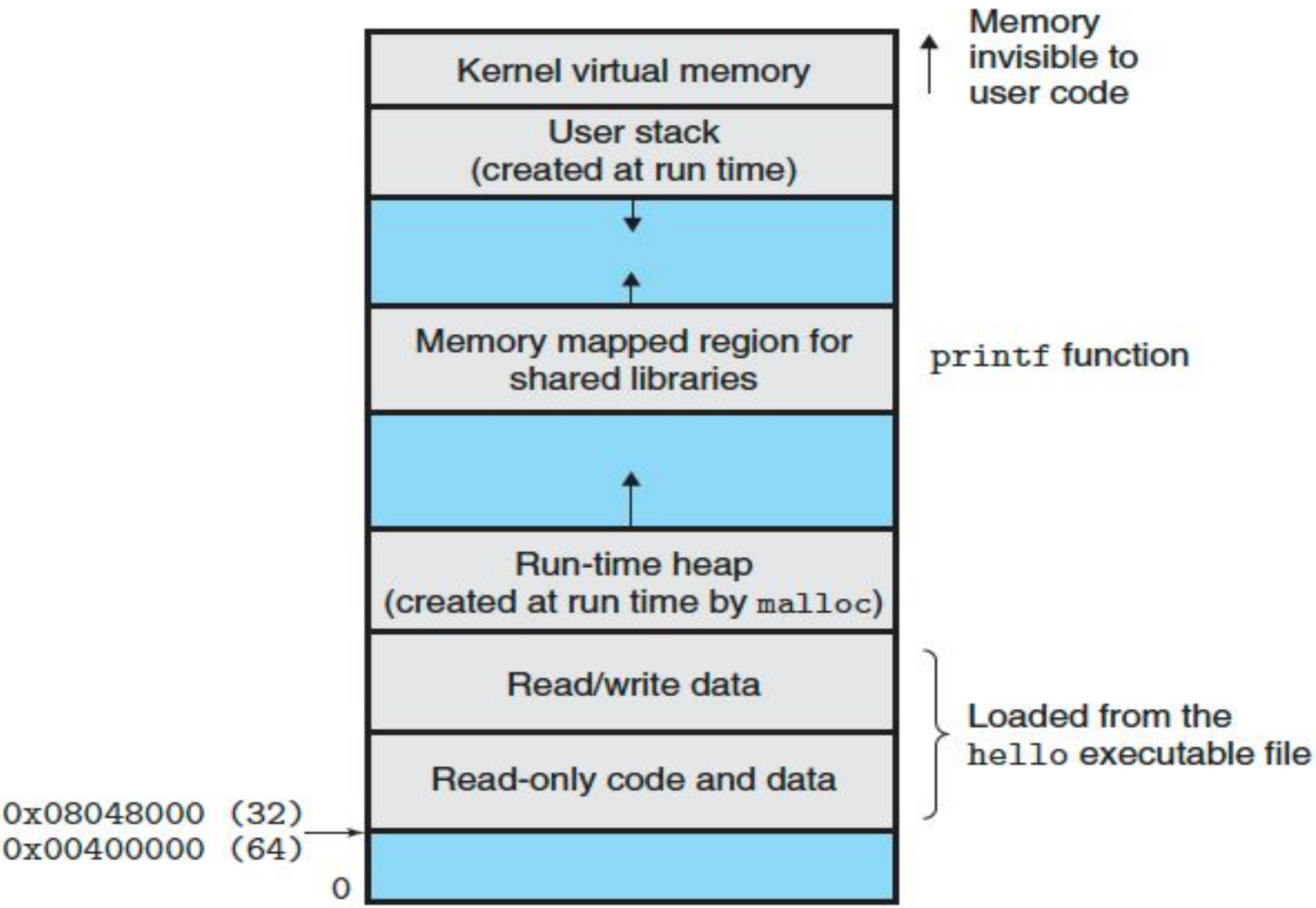


Recall: Each process can have the same address space



Solution: Virtual Memory

Address Space



Linux Memory layout of a process

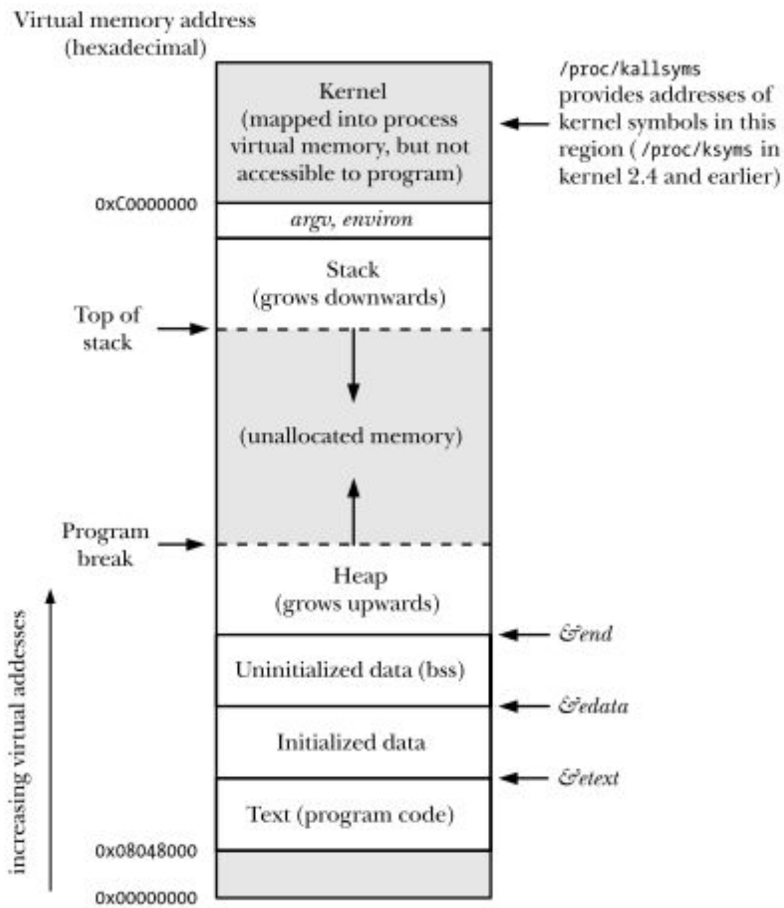
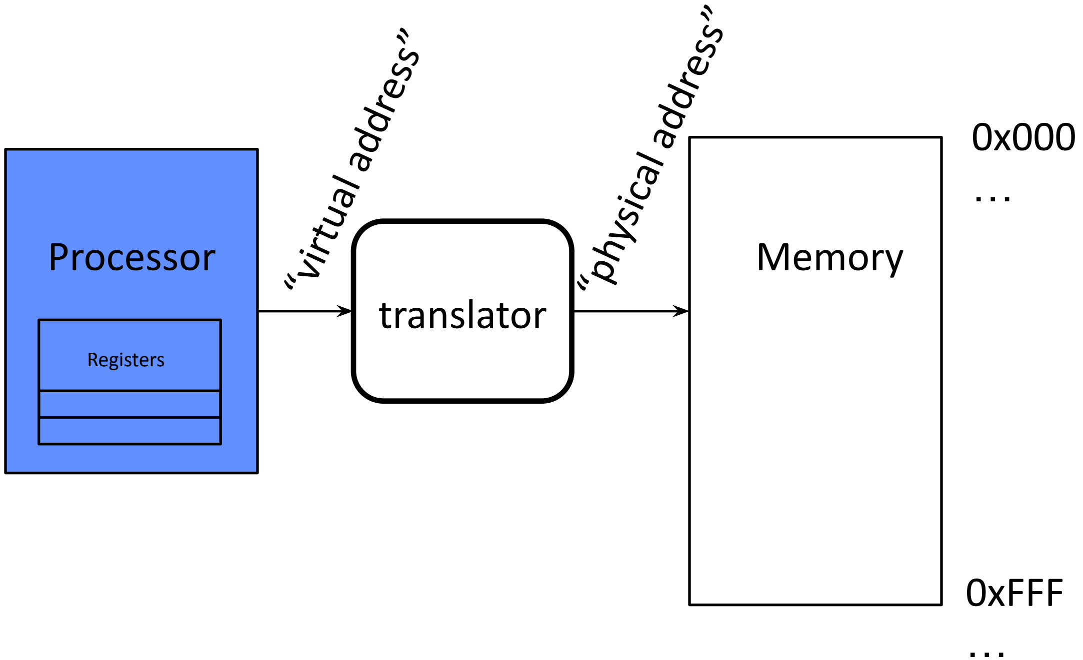


Figure 6-1: Typical memory layout of a process on Linux/x86-32

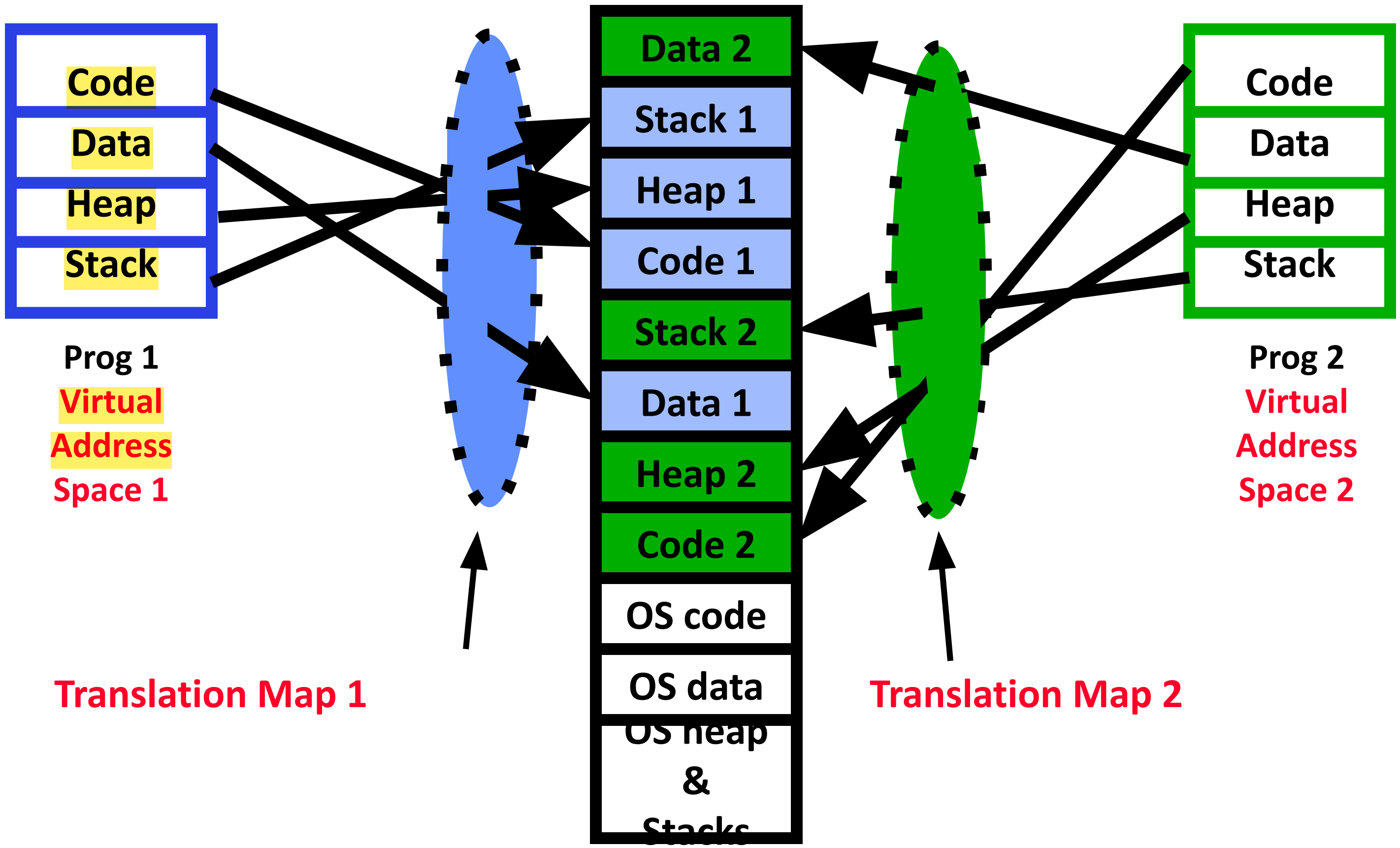
- Unix: Kernel space is mapped in high - but inaccessible to user processes

Address Space Translation

- Program operates in an address space that is distinct from the physical memory space of the machine

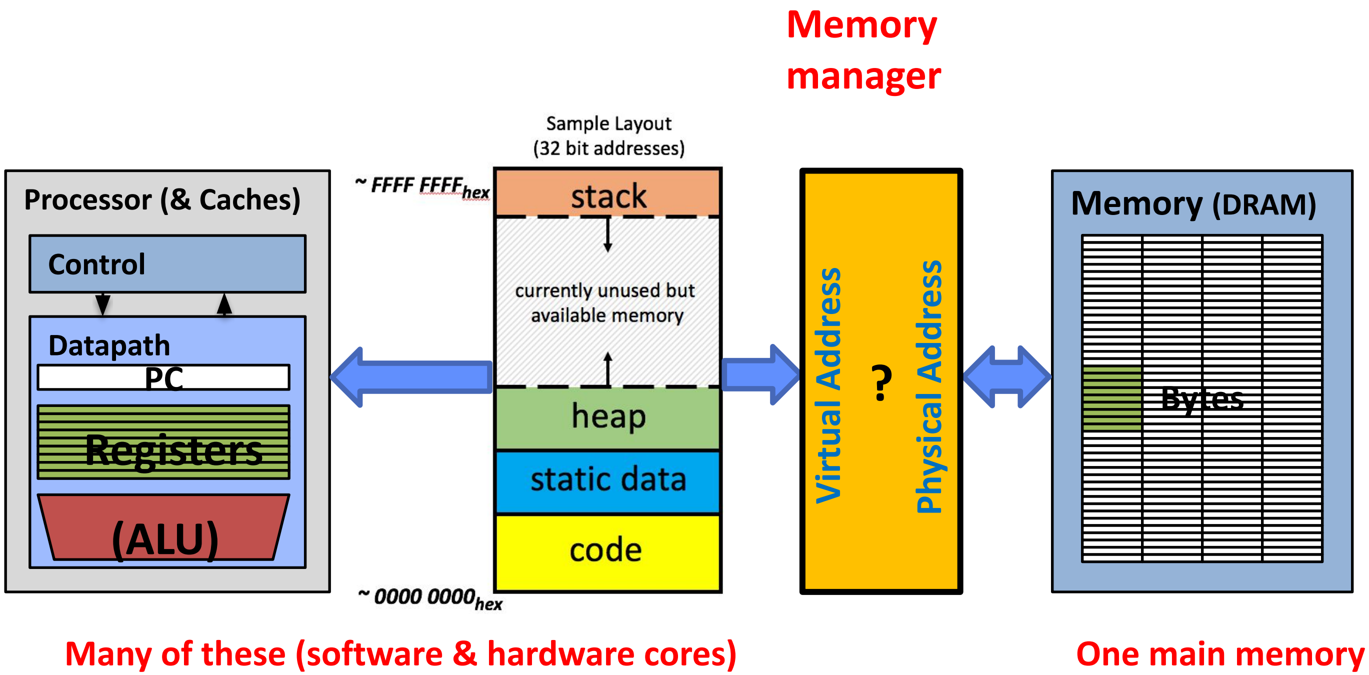


Translation through Page Table



Physical Address Space

Recall: Processors operate in Virtual Address Space



- Each Process uses it's own address space
- Processes use virtual addresses
- Many processes, all using same (conflicting) addresses
- Memory uses physical addresses (also, e.g., 0 ... 0xffff,ffff)
- ***Memory manager maps virtual to physical addresses***

Lecture Summary

- The four important concepts of Operating Systems include
  - Process
  - Address Space
  - Thread
  - Dual mode of operation / Protection
- Process is a program in execution
- To create a process, the OS does the following
  - Creates space in MM to hold
    - Code (Text), Stack, Heap, and Static
  - Starts running a process by executing the first instruction pointed by the PC
- Address space $\Rightarrow$ the set of accessible addresses + state associated with them