CS 31: Intro to Systems Course Recap

Kevin Webb
Swarthmore College
April 26, 2016

Reading Quiz

Reading Quiz

Just kidding. Did I scare you?

Final Exam

- Thursday, May 12, 2:00 PM. SCI 199

- Similar format to the midterm

- You get ~100% more time
- Exam is ~15% longer

- ~2/3 post-midterm material

Course Recap

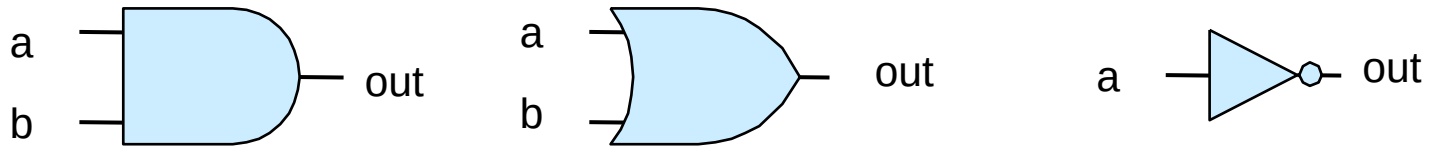
- This course was a vertical slice of computer
  - From lowest level: simple logic
  - To high level: large, complex programs run on OS

- Big goal: make complex machine easier to use
  - Hide details with the right **abstractions**
  - Improve performance when possible

Lowest Level

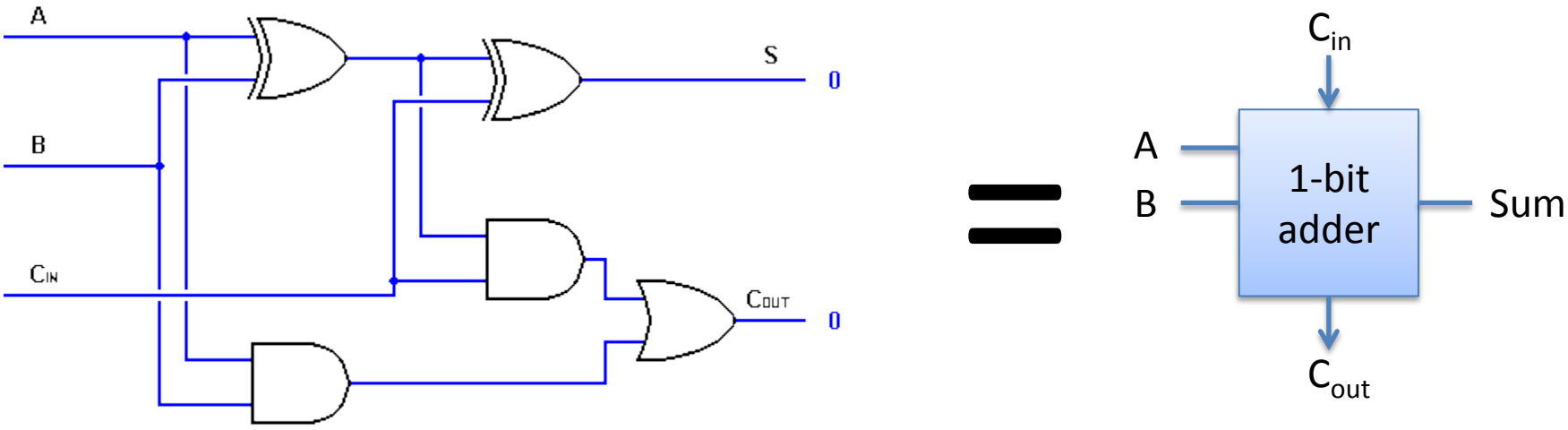
- Storing and representing data
  - 2's complement integers, floating point, etc.
  - Arithmetic using bits

- Logic gates: simple hardware



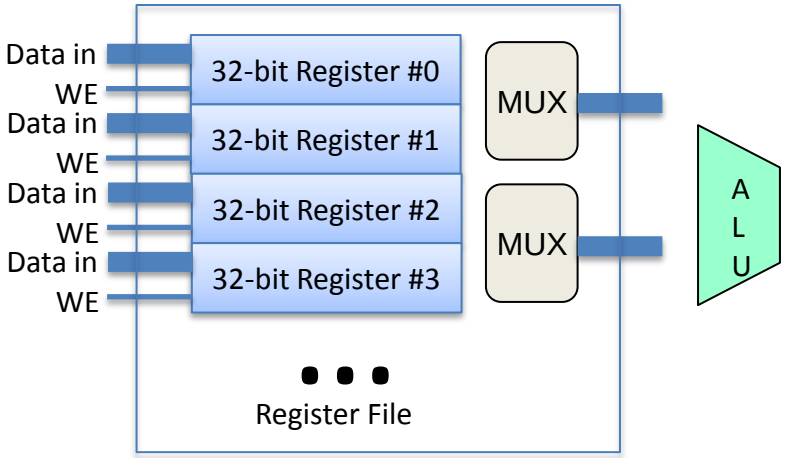
Hardware Abstraction: Circuits

- Combining gates to build specific circuits
  - arithmetic (adders, ALUs)
  - storage (latches, registers)
  - control (fetch, decode, multiplex)



CPU

- Combine circuits to create a CPU
  - Periodic clock: fetch, decode, execute instructions



Program Counter (PC): **Memory address of next instr**

Instruction Register (IR): **Instruction contents (bits)**

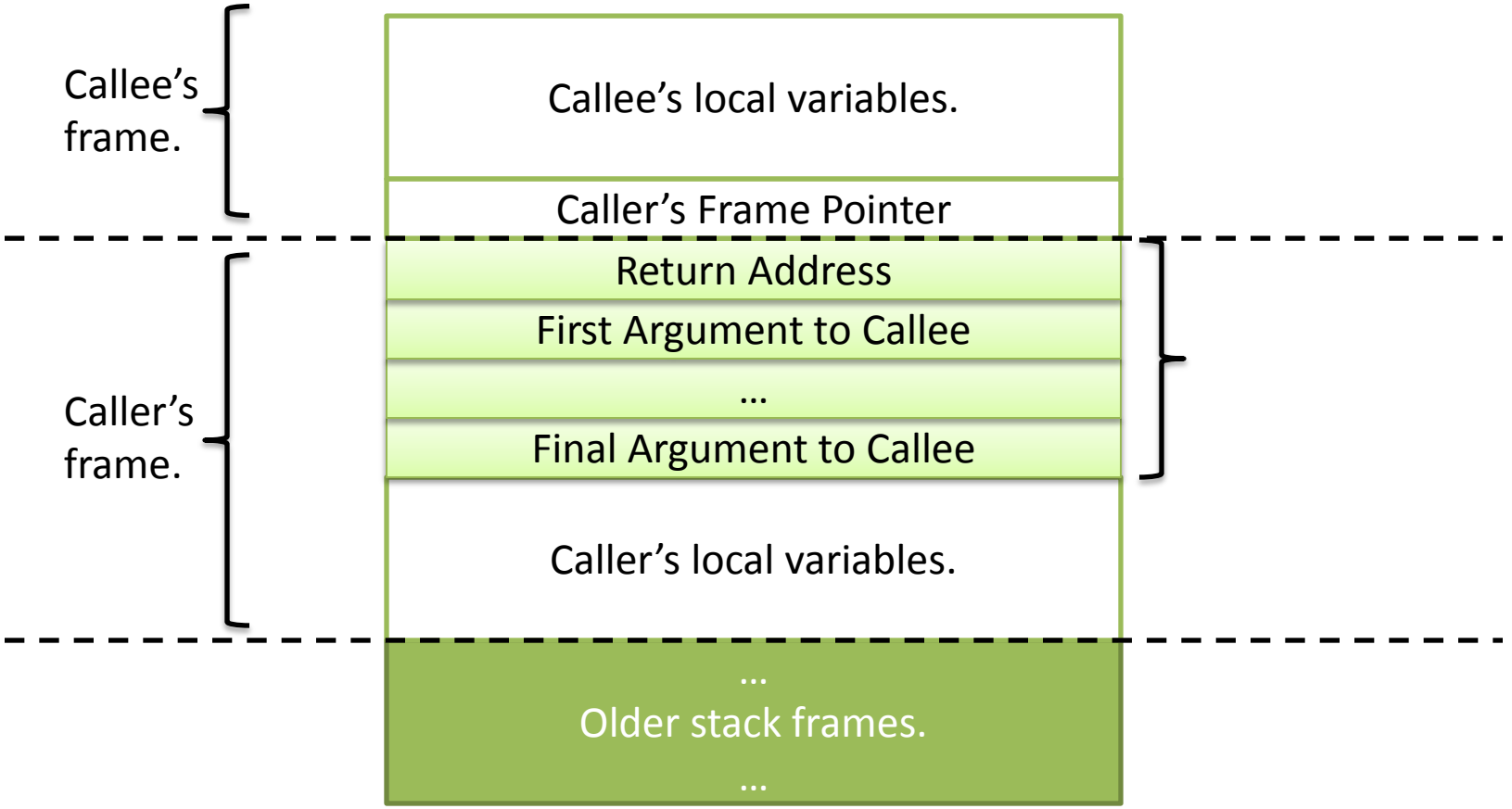
Instruction Set Architecture

- ISA defines CPU / software interaction
  - Machine properties (# registers, address modes)
  - Method for controlling hardware (assembly lang)

```
x = y >> 3 | x * 8
      movl -8(%ebp), %eax      # R[%eax] ← x
      imull $8, %eax           # R[%eax] ← x*8
      movl -12(%ebp), %edx     # R[%edx] ← y
      rshl $3, %edx            # R[%edx] ← y >> 3
      orl  %eax, %edx          # R[%edx] ← y>>3 | x*8
      movl %edx, -8(%ebp)      # M[R[%ebp-8]] ← result
```

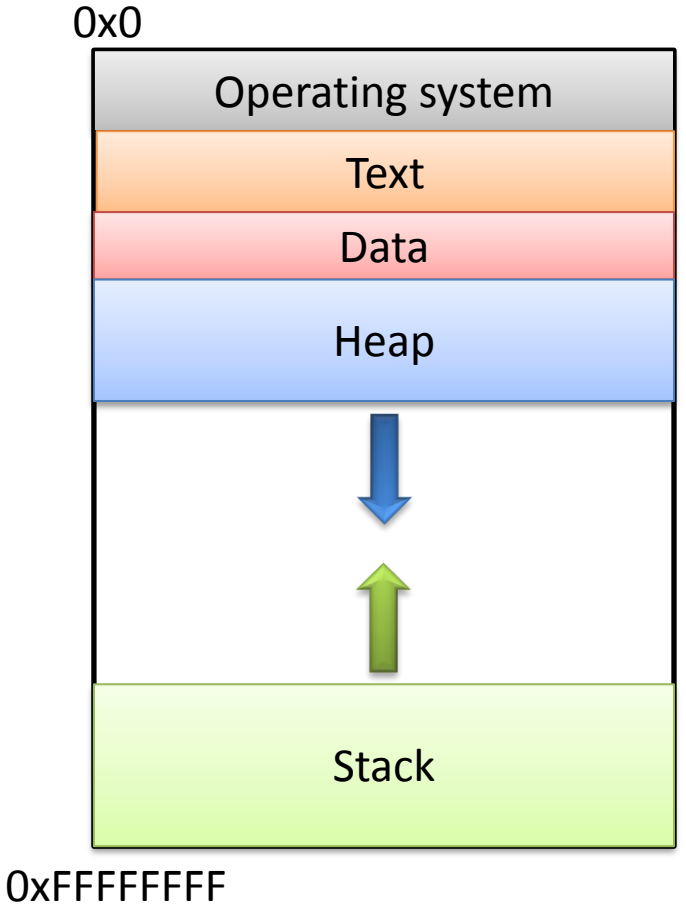
Conventions

- Agreed upon system for using ISA
  - e.g., manipulating the stack, register meaning

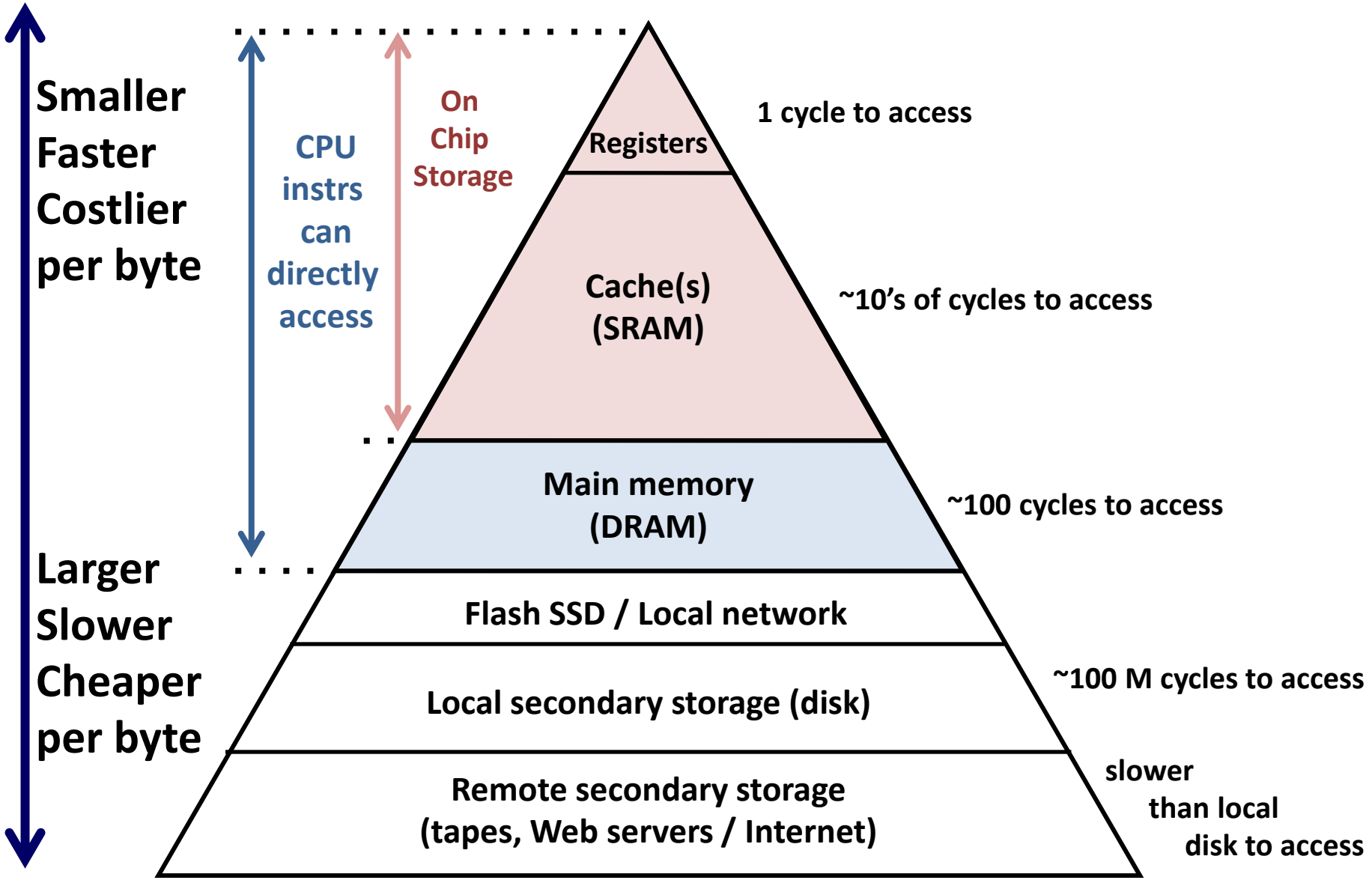


Storage and Memory

- Allocating memory (stack vs. heap)
- Pointers
  - malloc() / free()
  - address of (&)
  - dereferencing
  - arrays, 2D arrays



The Memory Hierarchy



Caching

- Improve performance by keeping a small memory for frequently-used data
  - Many parameters inform address division (tag, idx)
    - direct map vs. associative
    - block size
- Exploit major idea: **Locality**
  - temporal / spatial

Operating System

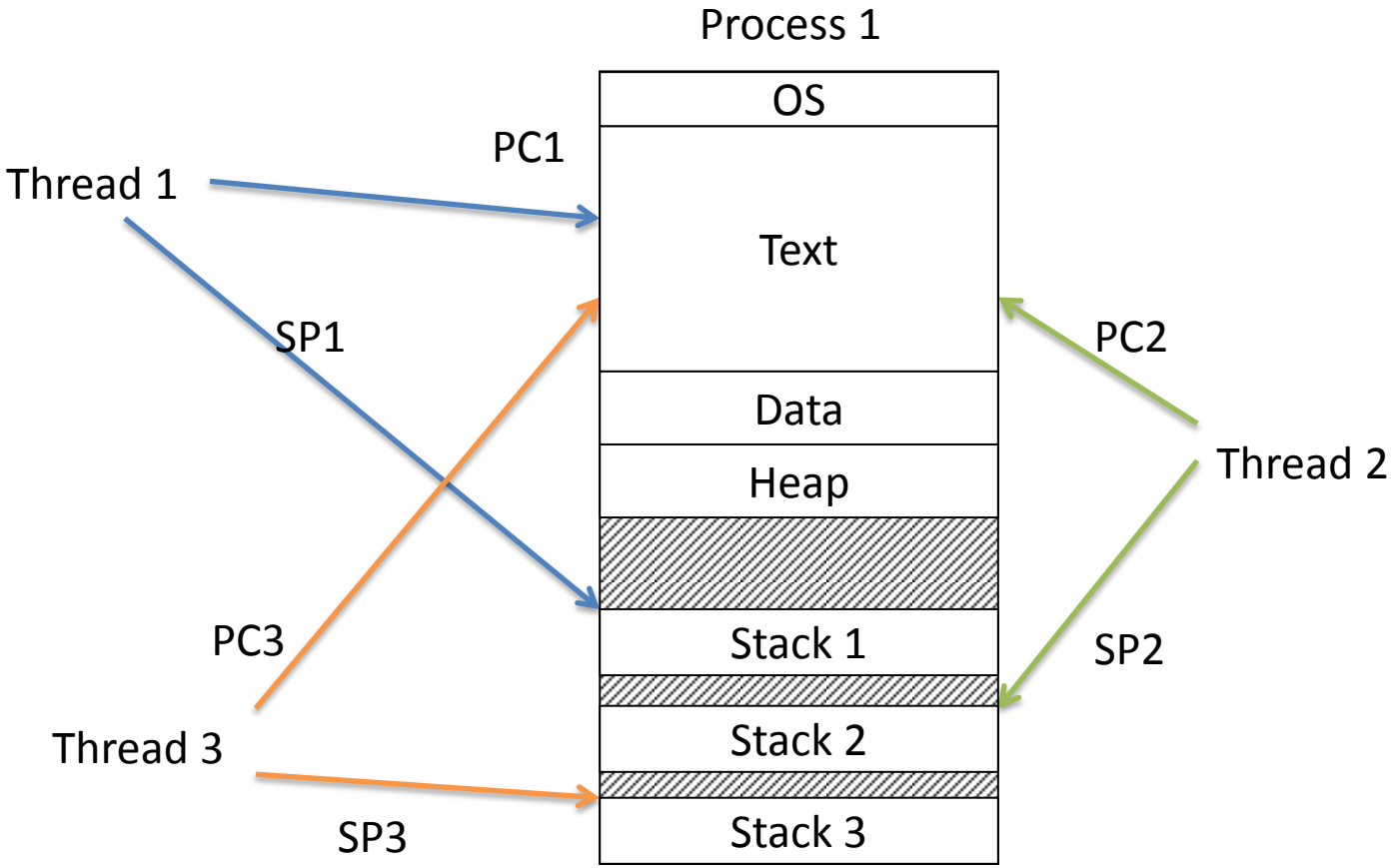
- Software supports: making programs easy/fast
- Three major abstractions:
  1. Process
  2. Thread
  3. Virtual memory
- Mechanism vs. policy

Processes

- Program in execution
  - fork() / exit() to create / terminate
- Represents all of the resources of a task
  - virtual address space (process memory)
  - open files
  - process ID, other accounting info
- One or more threads of execution

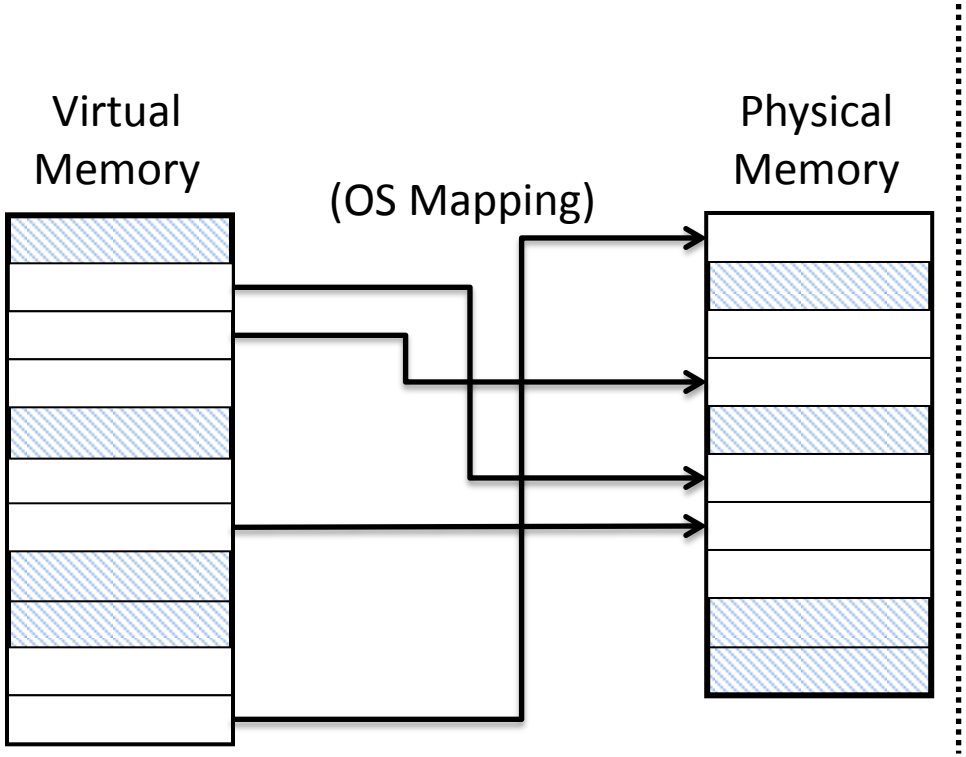
Threads

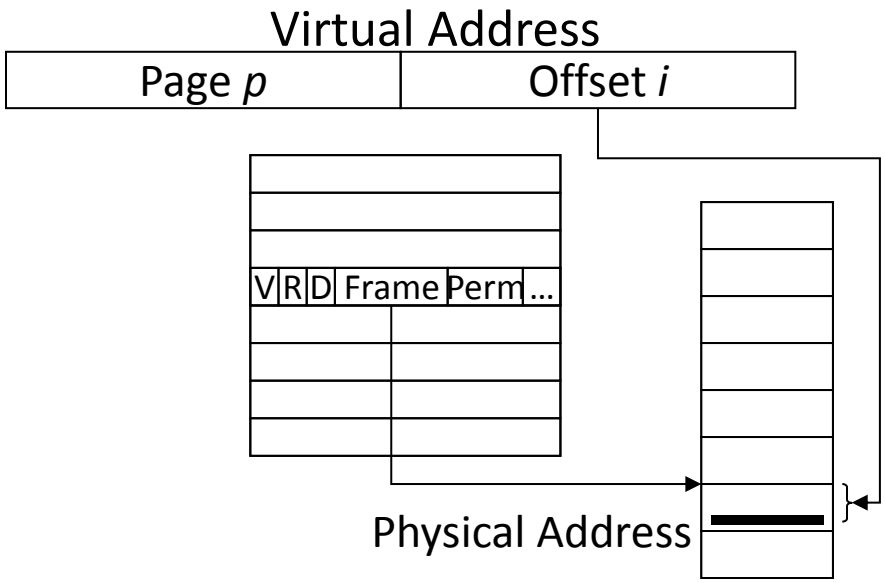
- Execution context within a process
- Independently scheduled



Virtual Memory

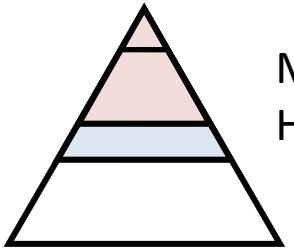
- Allow processes to behave as if they have the entire memory of the machine
- Translate from virtual (fantasy) address to physical

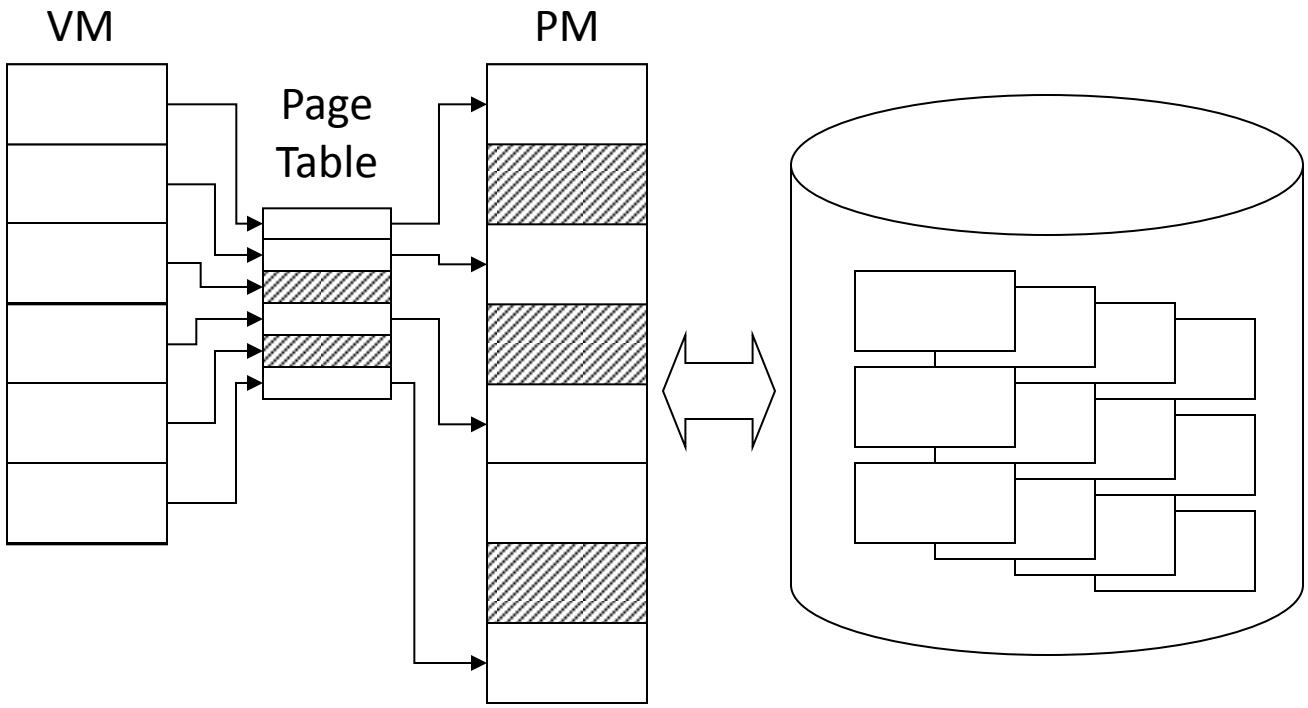




Virtual Memory

- Use disk to store data that hasn't been used lately
  - (Another instance of exploiting locality)





Mechanism & Policy

- Mechanism: the ability to do something
- Policy: rules for governing the mechanism(s)

Mechanism	Policy
Context switching	CPU scheduling
Cache eviction	Cache replacement policy
VM paging to disk	Page replacement policy

- “Best” policy usually varies by workload!

Concurrency & Parallelism

- Single CPU core performance has plateaued
  - Hardware giving us more CPU cores instead
- Programmer's responsibility to use them!
- Big opportunity for performance benefits!

Multi-threading in Practice (pthreads)

- Not always intuitive to reason about…
- Potential problems
  - race conditions
  - deadlock
  - priority inversion, etc.
- Requires careful synchronization

Questions?

- Thank you for a great semester!