Course Load in a Nutshell

Term Schedule

Dates are a guideline and are subject to change.

Assignment/Exam	Weight	Due Date
Tutorials	10%	1% each
Assignment 1	8%	January 31
Term Test 1	12%	February 3
Assignment 2	15%	February 28
Term Test 2	12%	March 2
Assignment 3	18%	April 3
Final Exam	25%	To be scheduled

Text: readings and forum questions (will be part of assignments!)

Silberschatz, Galvin, and Gagne 7th edition or later…

Tutorials

YES!!!! THEY ARE WORTH GOING TO!!! 1% of your mark each, good practice for tests/exam

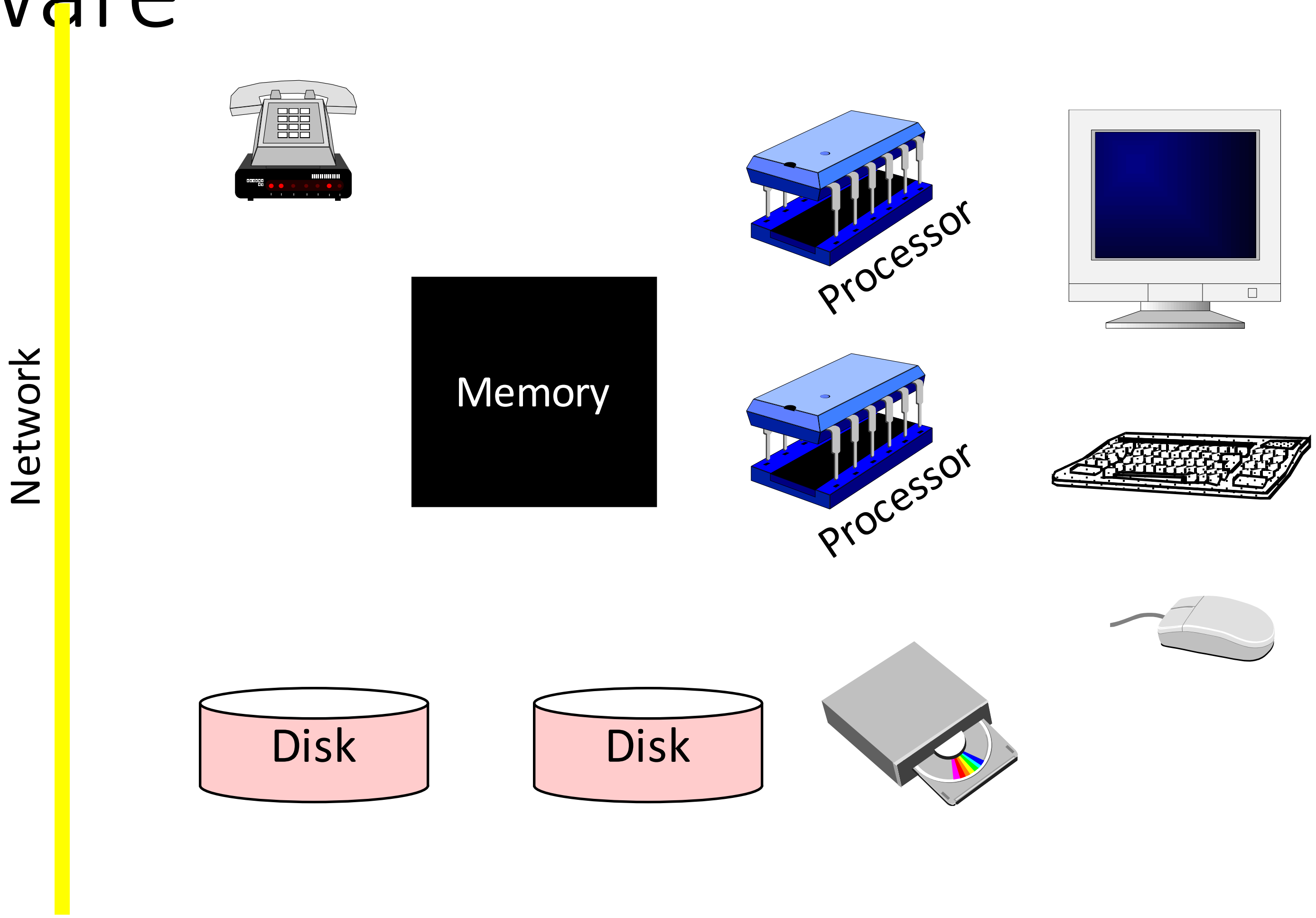
T01	ELL 061	2020-01-13	2020-04-03	W	10:30-11:20
T02	ECS 108	2020-01-13	2020-04-03	W	13:30-14:20
T03	DSB C116	2020-01-13	2020-04-03	R	14:30-15:20

In the Beginning …

- There was hardware
  - processor
  - storage
  - *card reader…*

- And not much else
  - no operating system
  - no libraries
  - no compilers

Hardware



MAGICAL *Abstractions*

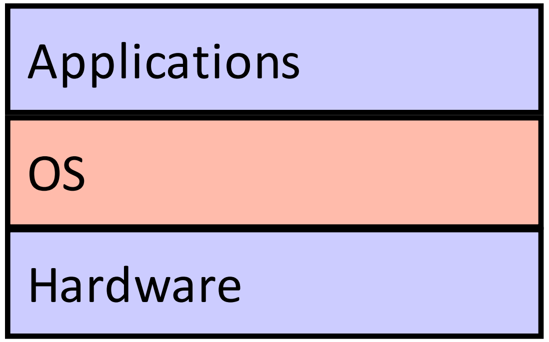
Hardware

- disks
- memory
- processors
- network
- monitor
- keyboard
- mouse

Operating system

- files
- programs
- threads of control
- communication
- windows, graphics
- input
- locator

The traditional picture



- "The OS is everything you don't need to write in order to run your application"
  - In some ways, it is:
    - all operations on I/O devices require OS calls (*syscalls)*
  - In other ways, it isn't:
    - you use the CPU/memory without OS calls
    - it intervenes without having been explicitly called

The OS and hardware

- An OS mediates programs' access to hardware resources (*sharing* and *protection*)
  - computation (CPU)
  - volatile storage (memory) and persistent storage (disk, etc.)
  - network communications (TCP/IP stacks, Ethernet cards, etc.)
  - input/output devices (keyboard, display, sound card, etc.)
- The OS abstracts hardware into logical resources and well-defined interfaces to those resources (*ease of use*)
  - processes (CPU, memory)
  - files (disk)
  - programs (sequences of instructions)
  - sockets (network)

Privileged instructions

- some instructions are restricted to the OS
  - known as privileged instructions
- e.g., only the OS can:
  - directly access I/O devices (disks, network cards)
    - why?
  - manipulate memory state management
    - page table pointers, TLB loads, etc.
    - why?
  - manipulate special ‘mode bits’
    - interrupt priority level
    - why?

OS protection

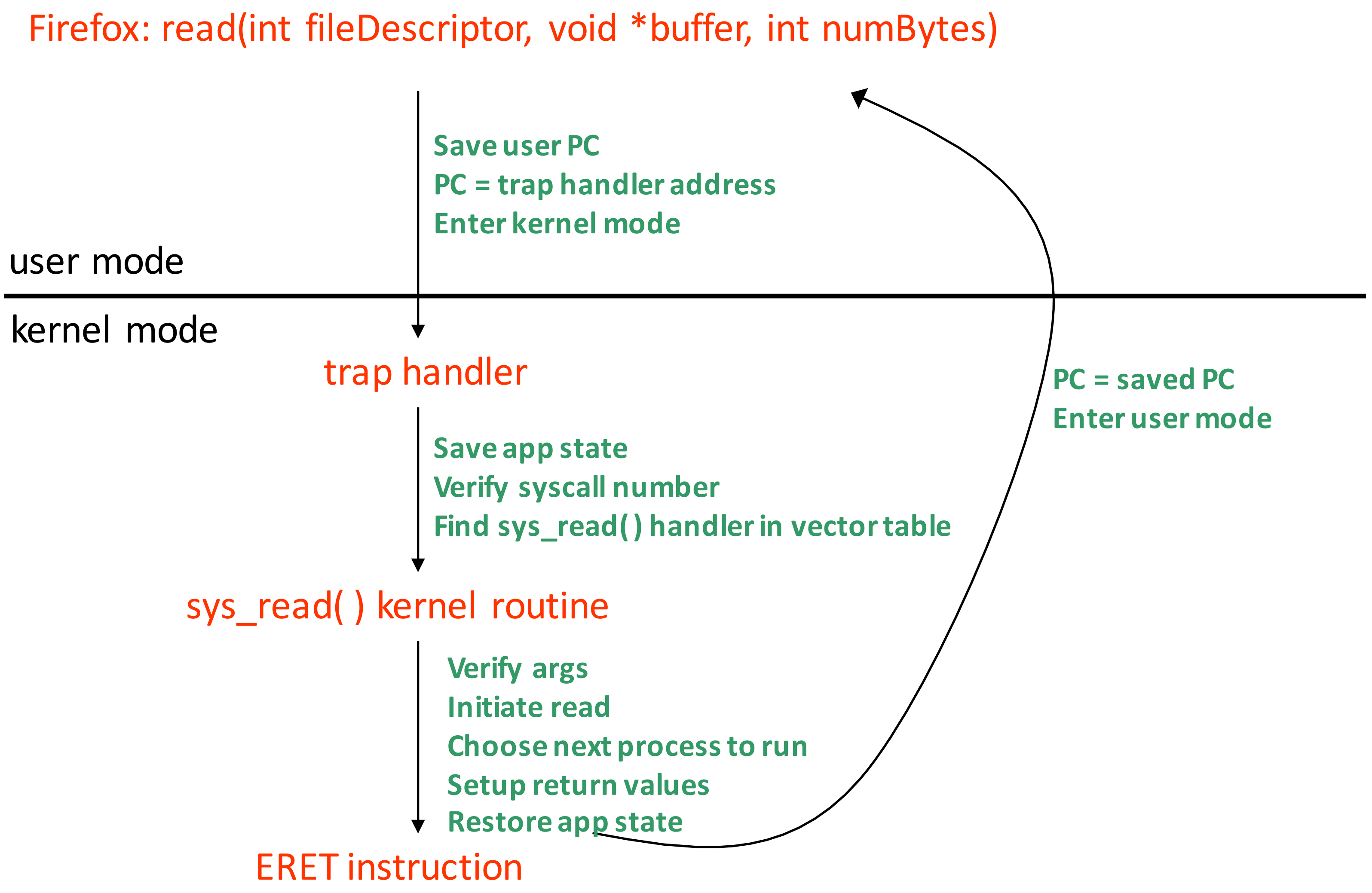
- So how does the processor know if a privileged instruction should be executed?
  - the architecture must support at least two modes of operation: kernel mode and user mode
    - VAX, x86 support 4 protection modes
  - mode is set by status bit in a protected processor register
    - user programs execute in user mode
    - OS executes in kernel (privileged) mode (OS == kernel)
- Privileged instructions can only be executed in kernel (privileged) mode

Crossing protection boundaries

- So how do user programs do something privileged?
  - e.g., how can you write to a disk if you can't execute an I/O instructions?
- User programs must call an OS procedure – that is, get the OS to do it for them
  - OS defines a set of system calls
  - User-mode program executes system call instruction
- Syscall instruction
  - Like a protected procedure call

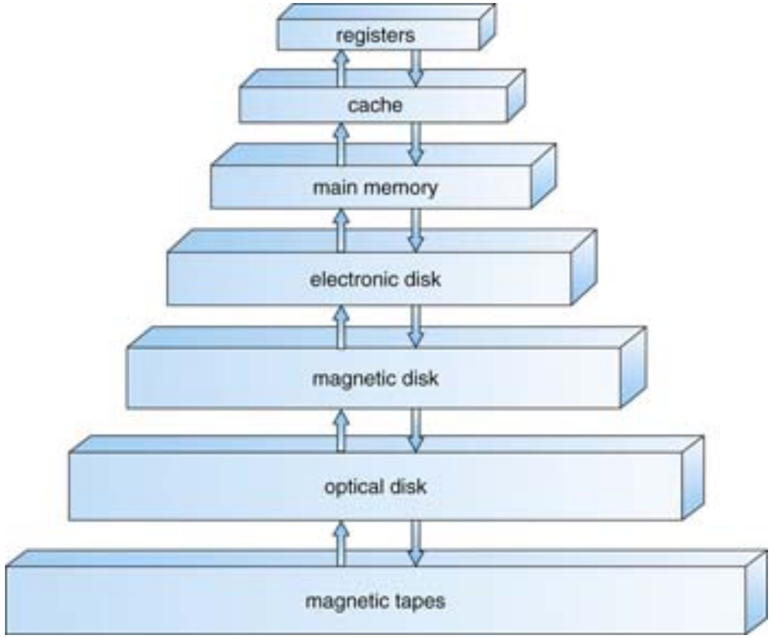
- The syscall instruction atomically:
  - Saves the current PC
  - Sets the execution mode to privileged
  - Sets the PC to a handler address
- With that, it's a lot like a local procedure call
  - Caller puts arguments in a place callee expects (registers or stack)
    - One of the args is a syscall number, indicating which OS function to invoke
  - Callee (OS) saves caller's state (registers, other control state) so it can use the CPU
  - OS function code runs
    - OS must verify caller's arguments (e.g., pointers)
  - OS returns using a special instruction
    - Automatically sets PC to return address and sets execution mode to user

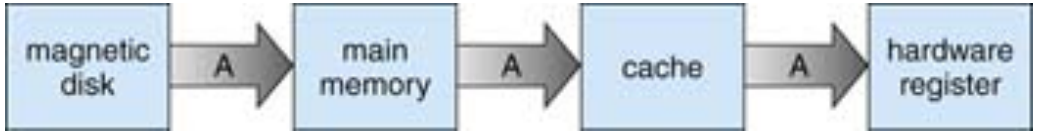
A kernel crossing illustrated

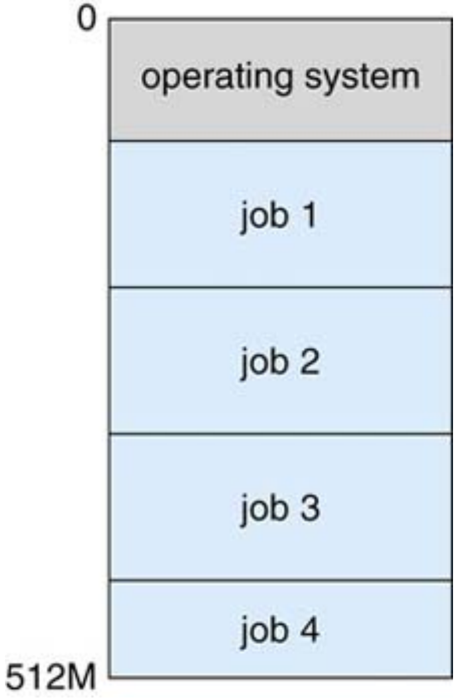


Memory?

Memory

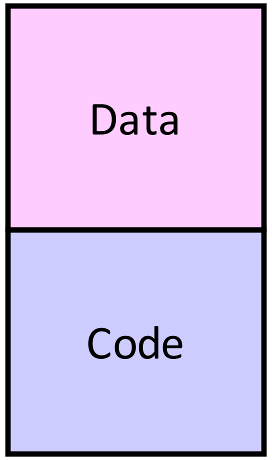




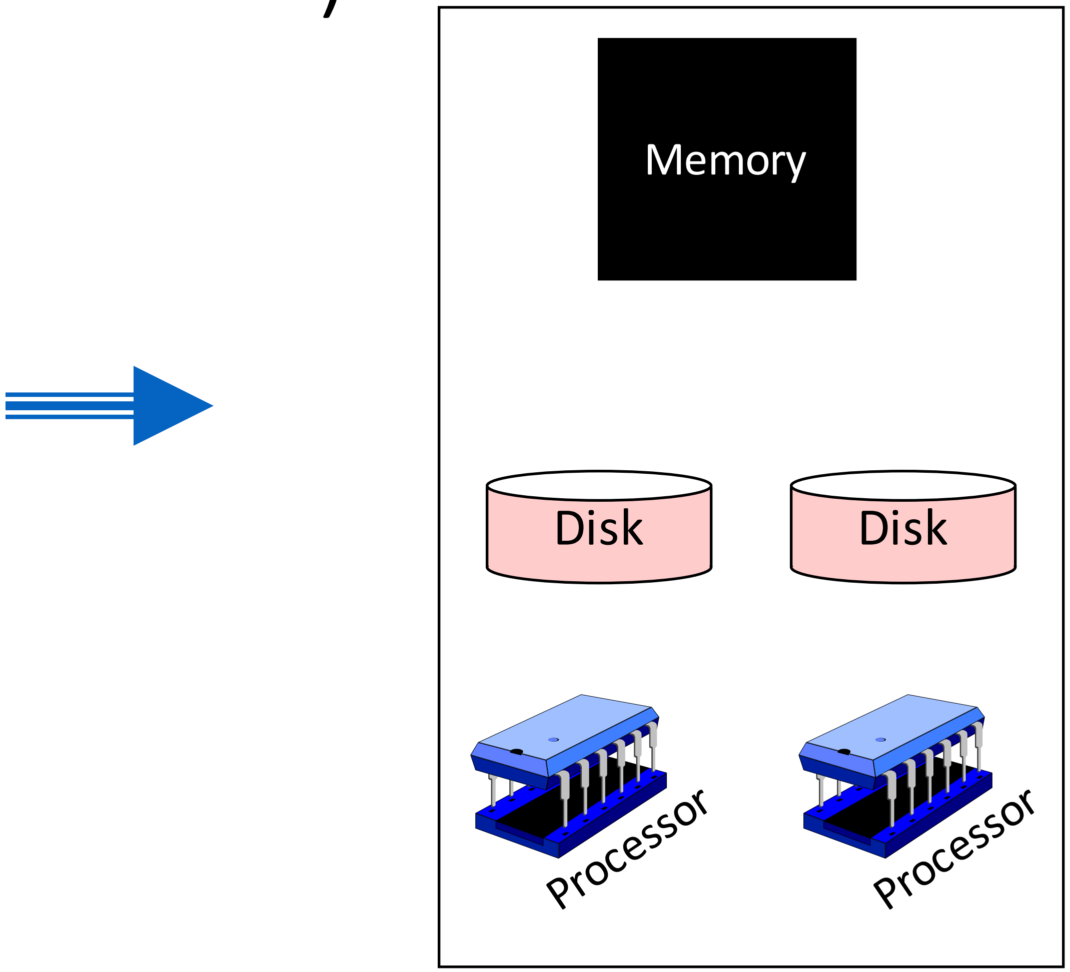


Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	< 16 MB	< 64 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25– 0.5	0.5– 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000– 100,000	5000– 10,000	1000– 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

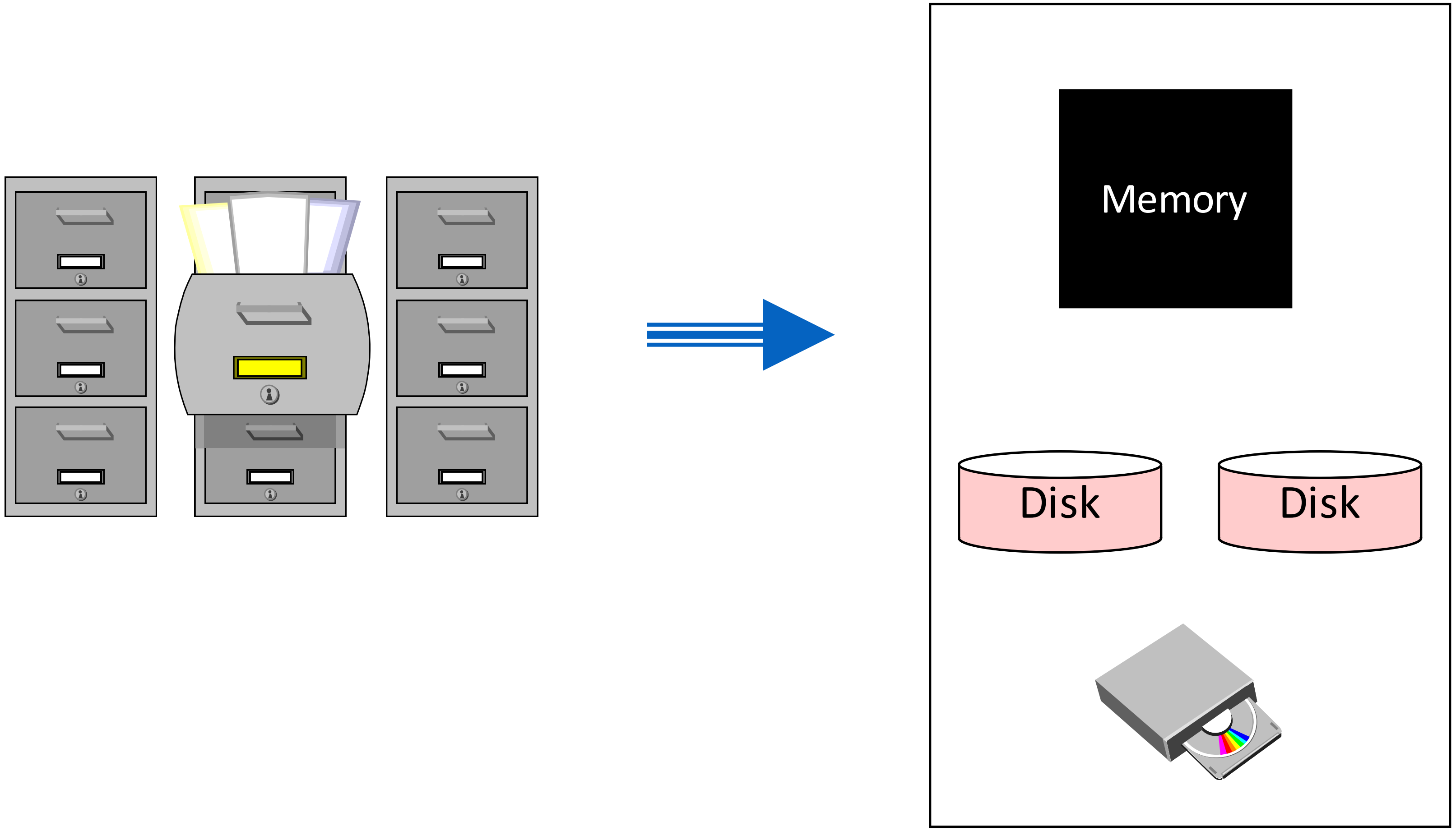
Programs (Assignment 2)



(maybe threads!)



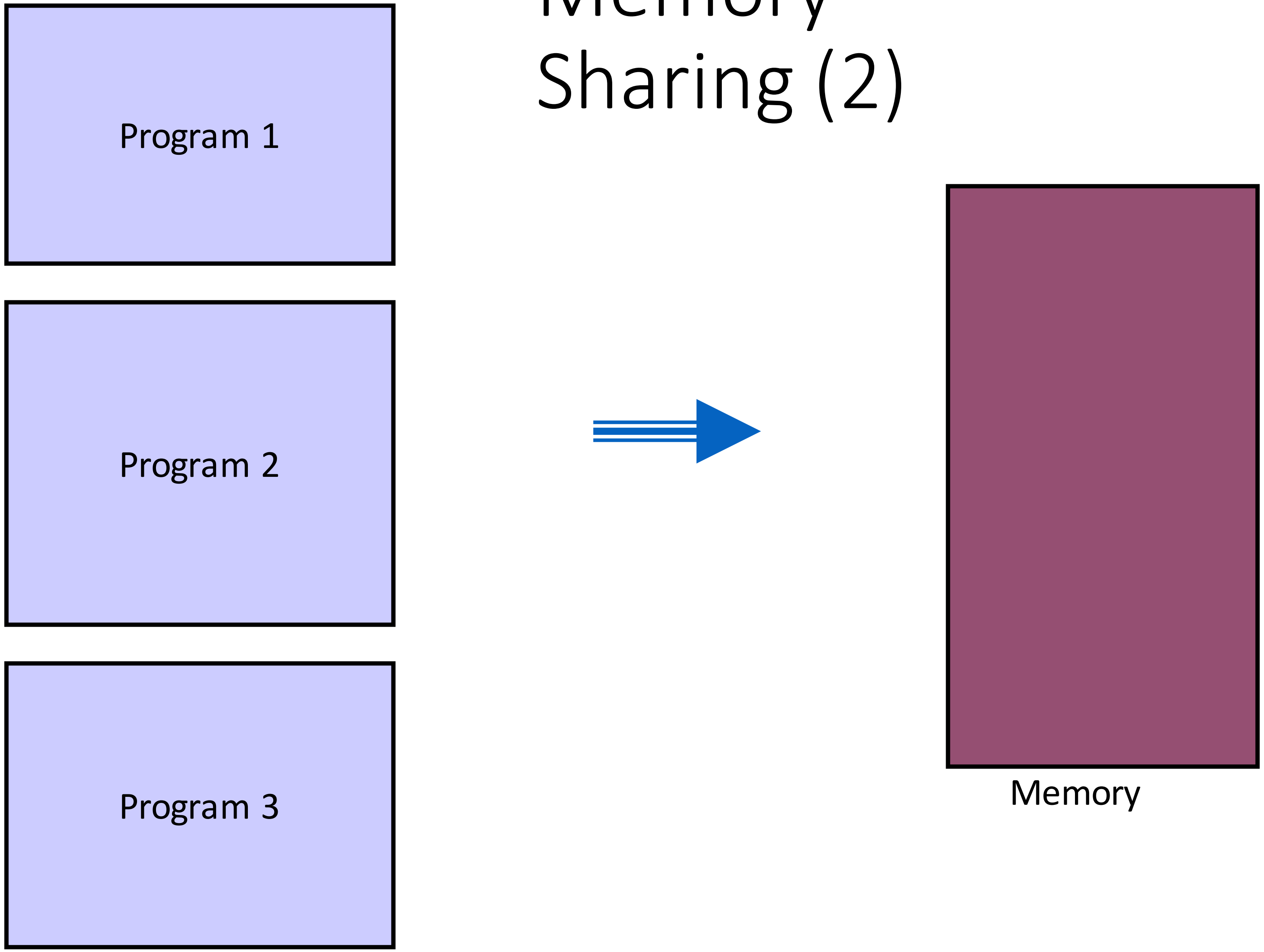
Files (Assignment 3)



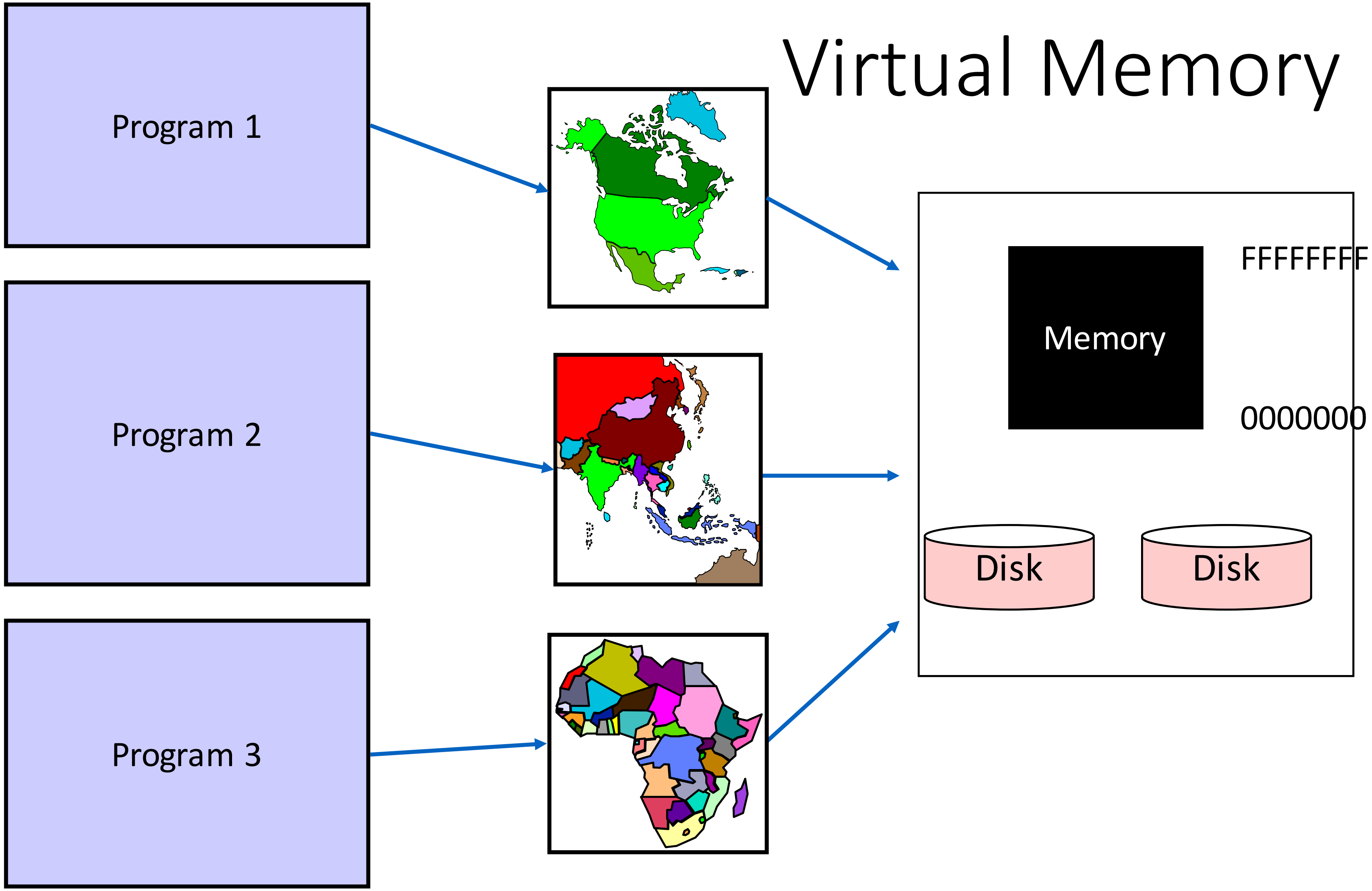
There are a lot of Issues…

- Naming!
- Allocating space on disk (permanent storage)
  - organized for fast access
  - minimize waste
- Shuffling data between disk and memory (high-speed temporary storage)
- Coping with crashes

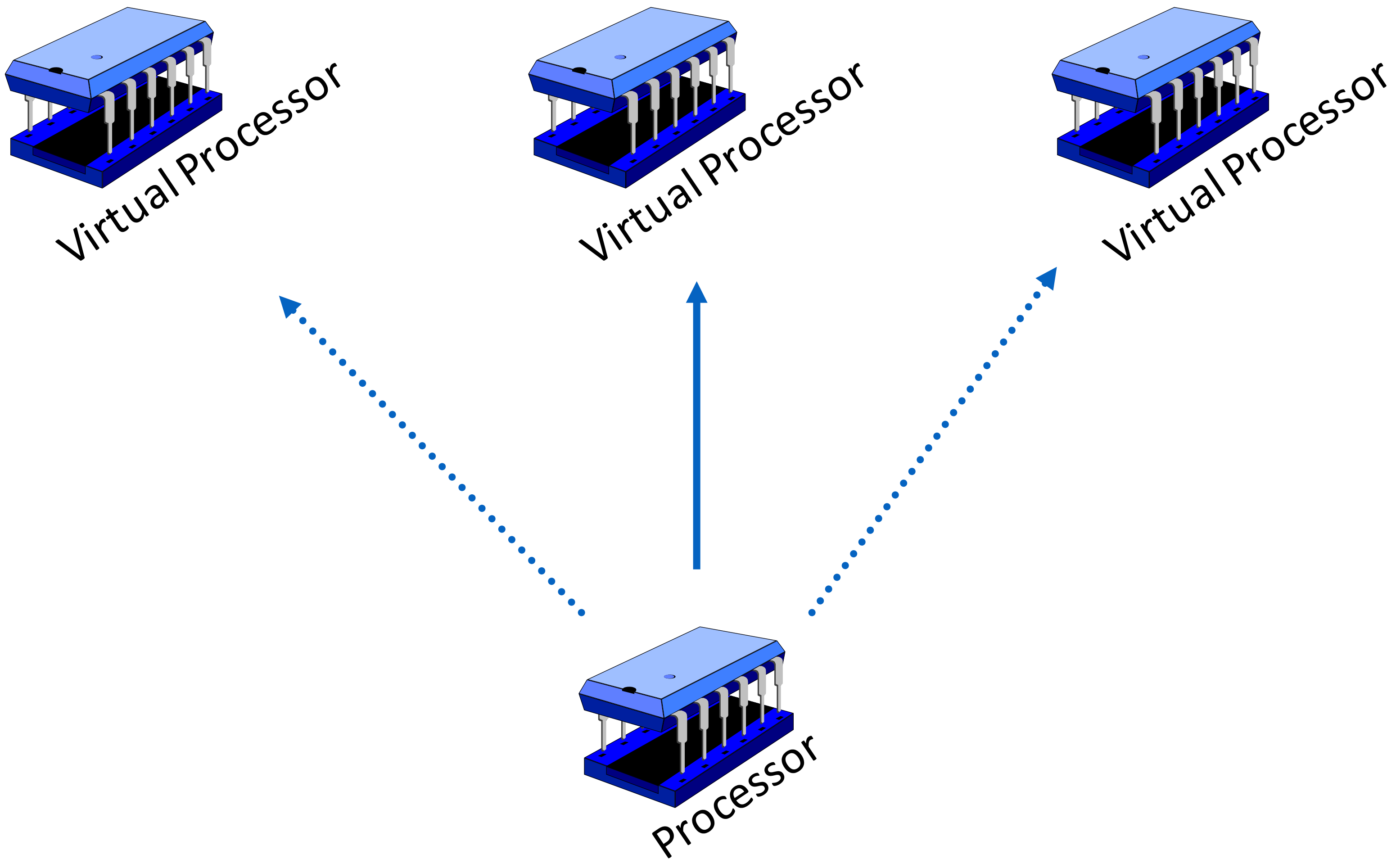
Memory Sharing (2)



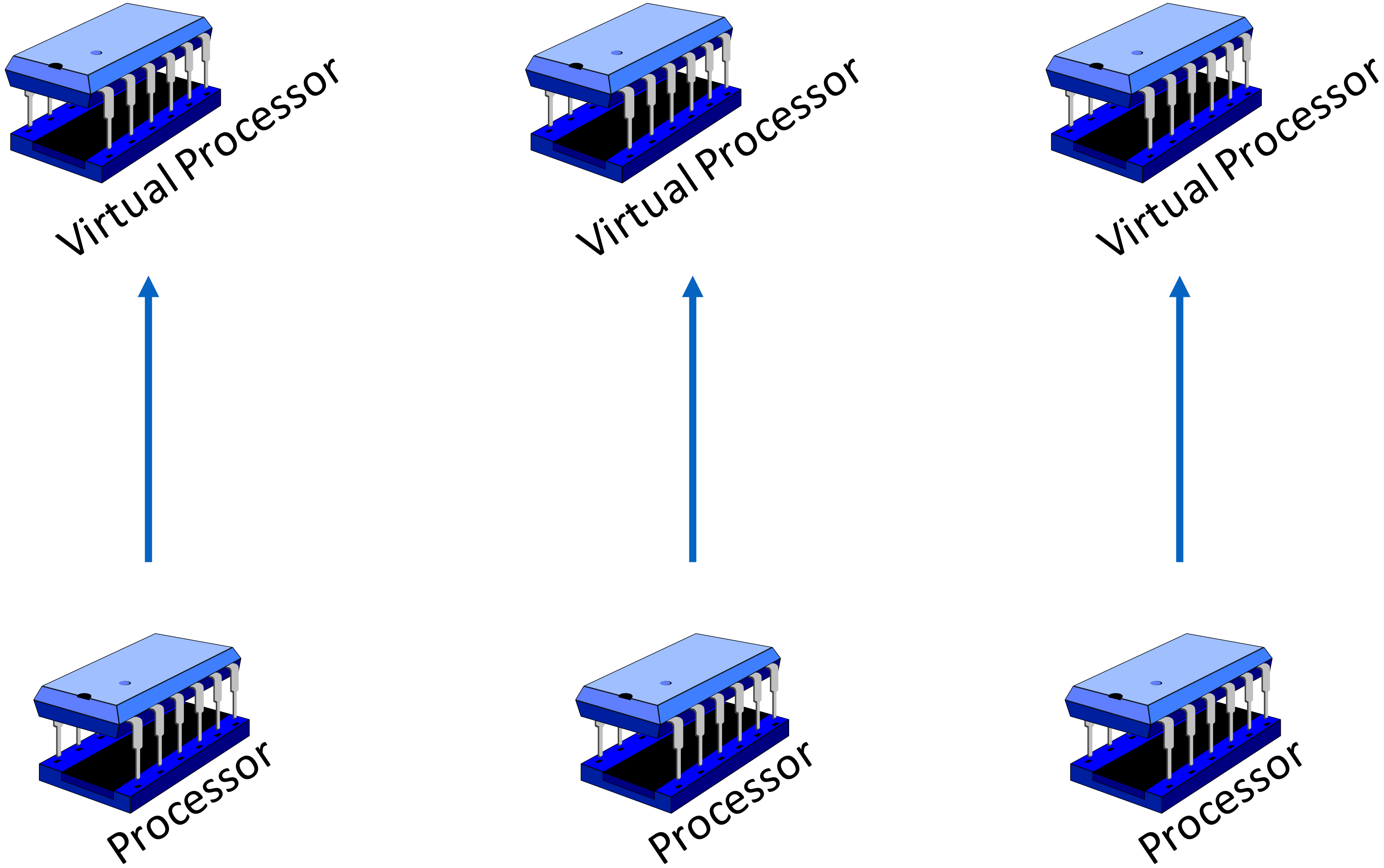
Virtual Memory



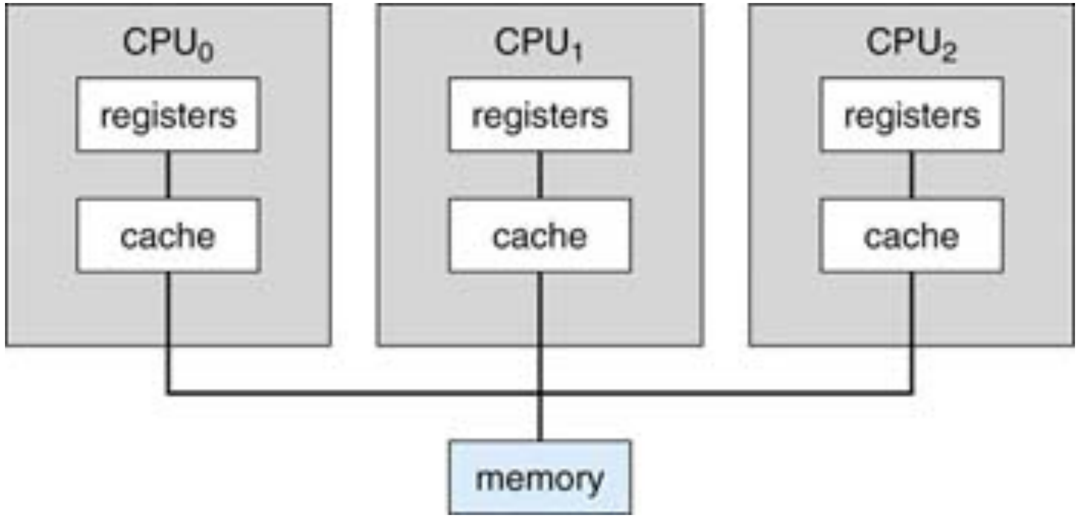
Concurrency (more in Assignment 2)

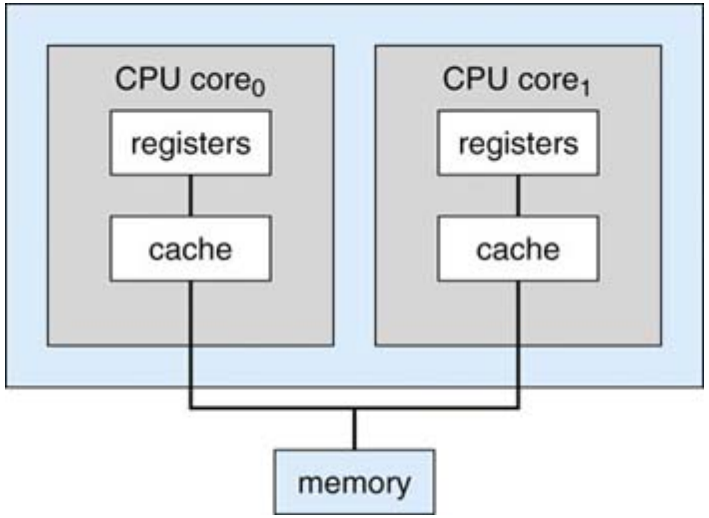


Parallelism



Processors and cores! (Assignment 2)





TO DO!!!!

- Be sure you're on CSC 360 CourseSpaces and upload your "picture"
  - You should have received email by now!

- Please keep up with **the programming**!

- Forum (reading/questions) is posted on the web **now**
  - Due as part of Assignment 1
  - https://www.g2.com/categories/operating-system

- Assignment 0 (parts 1 and 2) posted on CousreSpaces **now**
  - Tutorial question next week!
  - Upload a super duper d-list!!!