CS 354
Machine Organization and Programming

Michael Doescher
Spring 2021

Intro to Operating Systems II
- CPU Virtualization
- Memory Virtualization
- Concurrency / Threads

Week 13 Readings

- Introduction_to_OS.pdf
- Processes.pdf
- Direct_Execution.pdf
- Address_Space.pdf
- Address_Translation.pdf
- Paging.pdf
- Threads.pdf

The Process

- 10000 processes at a time?
- With 1 CPU?

- “Virtualizing the CPU” emulating many CPU’s
- OS creates the illusion of an endless supply of CPUs by time sharing

- Hardware Resources required for the “context switching”
- OS Resources to make decisions about scheduling

State

- List of all machine resources used by a process
  - Memory (instructions, data, heap, stack) -> The Address Space
  - Registers
  - I/O
    - files
    - standard in, standard out, error stream
    - network connections

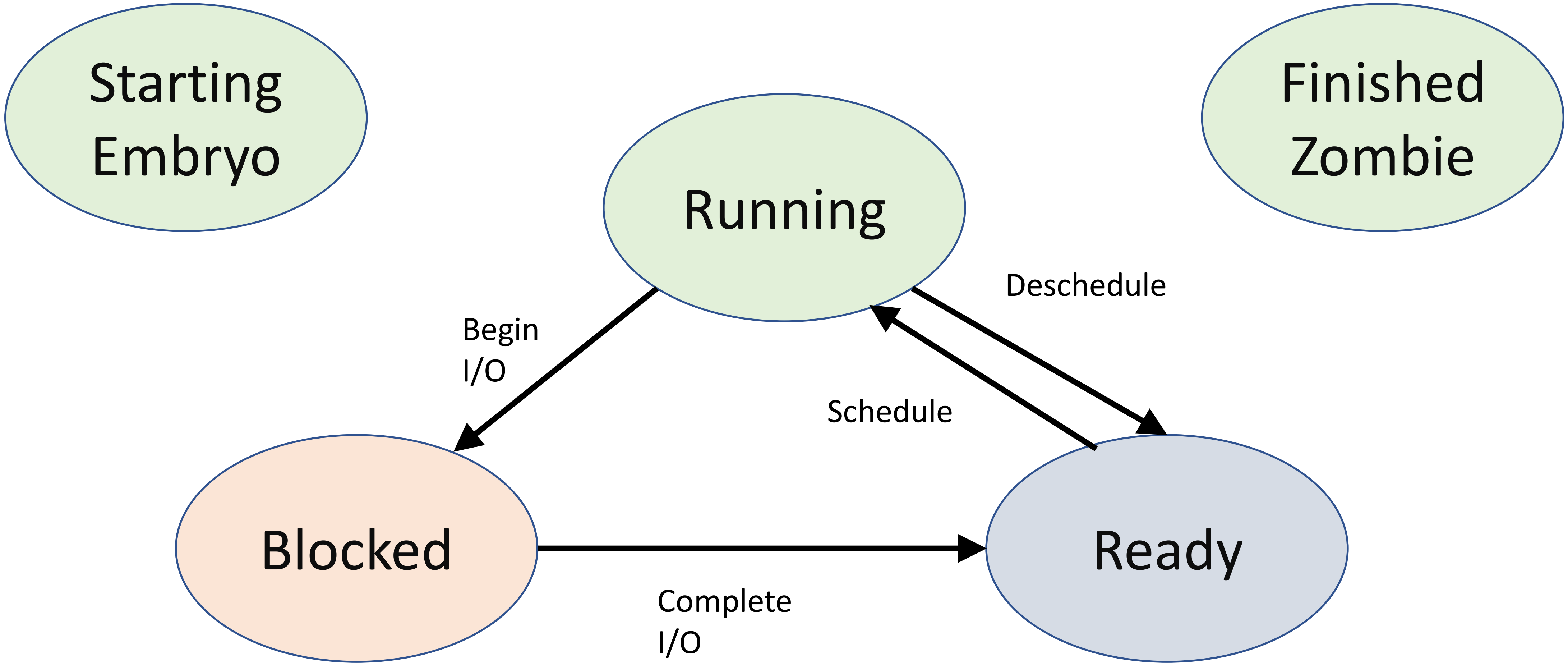
System Calls

- Create
- Destroy
- Wait
- Control
- Status

Create

- Add entry to the process list
- Copy Code and Data to Memory (load)
- Allocate Memory for Heap and Stack
- Push Command Line Parameters onto the Stack
- Setup I/O – standard in , standard out, error
- Clear Registers
- Transfer Control to 1st instruction of main

Process State



Direct Execution

- OS Creates Process
- Transfer Control to Process
  - Run Main
  - Return 0
- Destroy
- Free Process Memory
- Remove from Process List

Issues

- No Limitations
- Infinite Loop
- Security
  - restricted operations

Limited Direct Execution

- Time – Sharing
- User Mode and Kernel Mode
- Trap Instructions + Return from Trap

- Trap and Interrupt Tables set up at boot time
  - Where is the code to run each system call
  - Address of the SysCall Handler is stored in a register at boot time

System Call Protocol

1) OS
- Create Process
- Push Registers to Kernel Stack
- Push PC to Kernel Stack
- Return from Trap

2) Hardware
- Restore Registers (from Kernel Stack)
- User Mode
- Restore PC (from Kernel Stack)

3) Process
- Run Main
- System Call – Trap to OS
- With System Call Number

4) Hardware
- Push Registers to Kernel Stack
- Kernel Mode
- Jump to Trap Handler (PC)

5) OS
- Handle Trap
- Return from Trap

6) Hardware
- Go to Step 2

7) OS
- If Trap is exit / return from main
- Free memory
- Remove from Process List

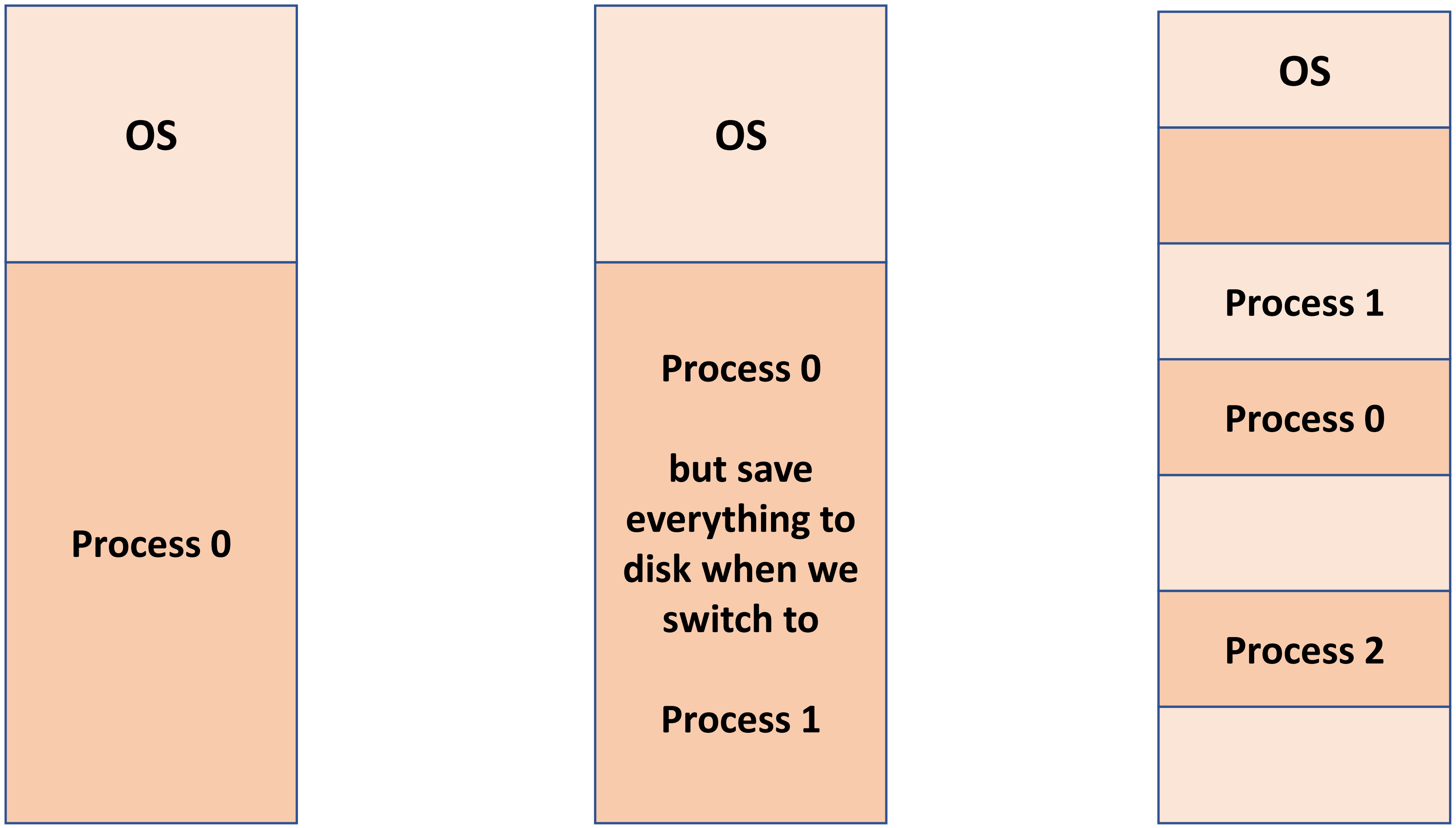
CS 354
Machine Organization and Programming

Michael Doescher
Spring 2021

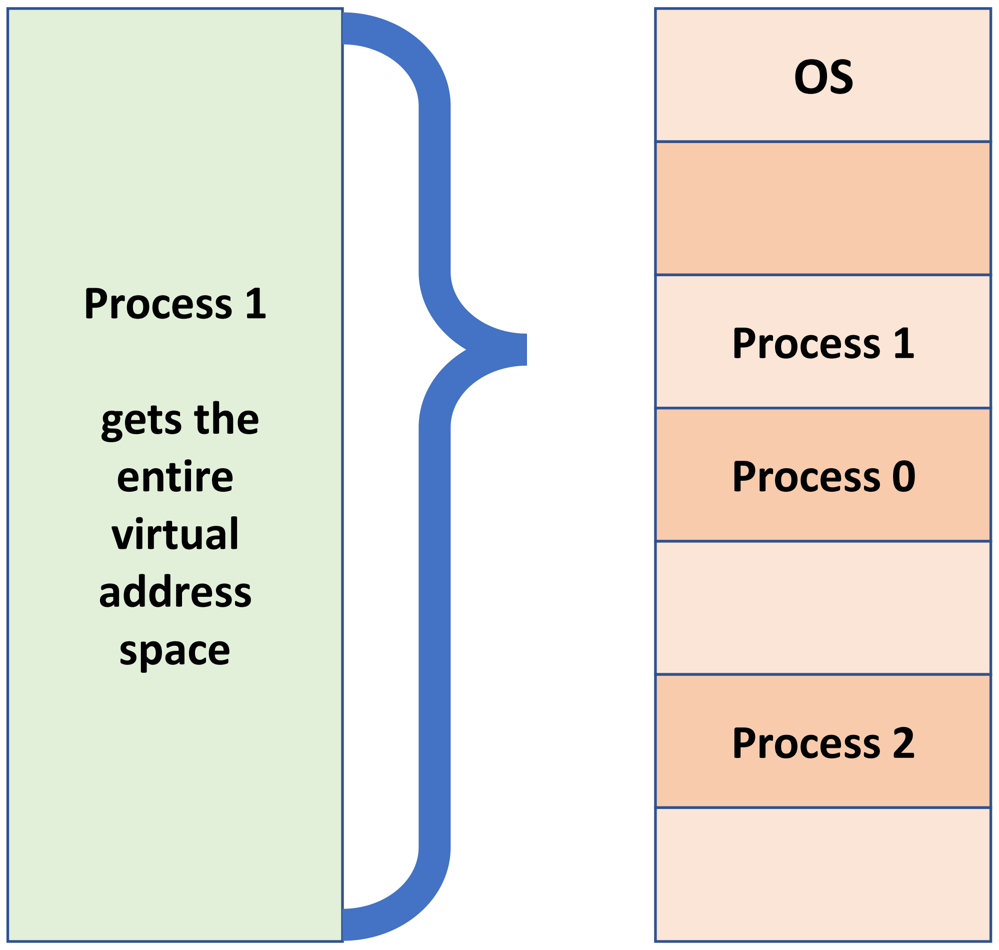
Intro to Operating Systems II

- CPU Virtualization
- Memory Virtualization
- Concurrency / Threads

Address Space Models



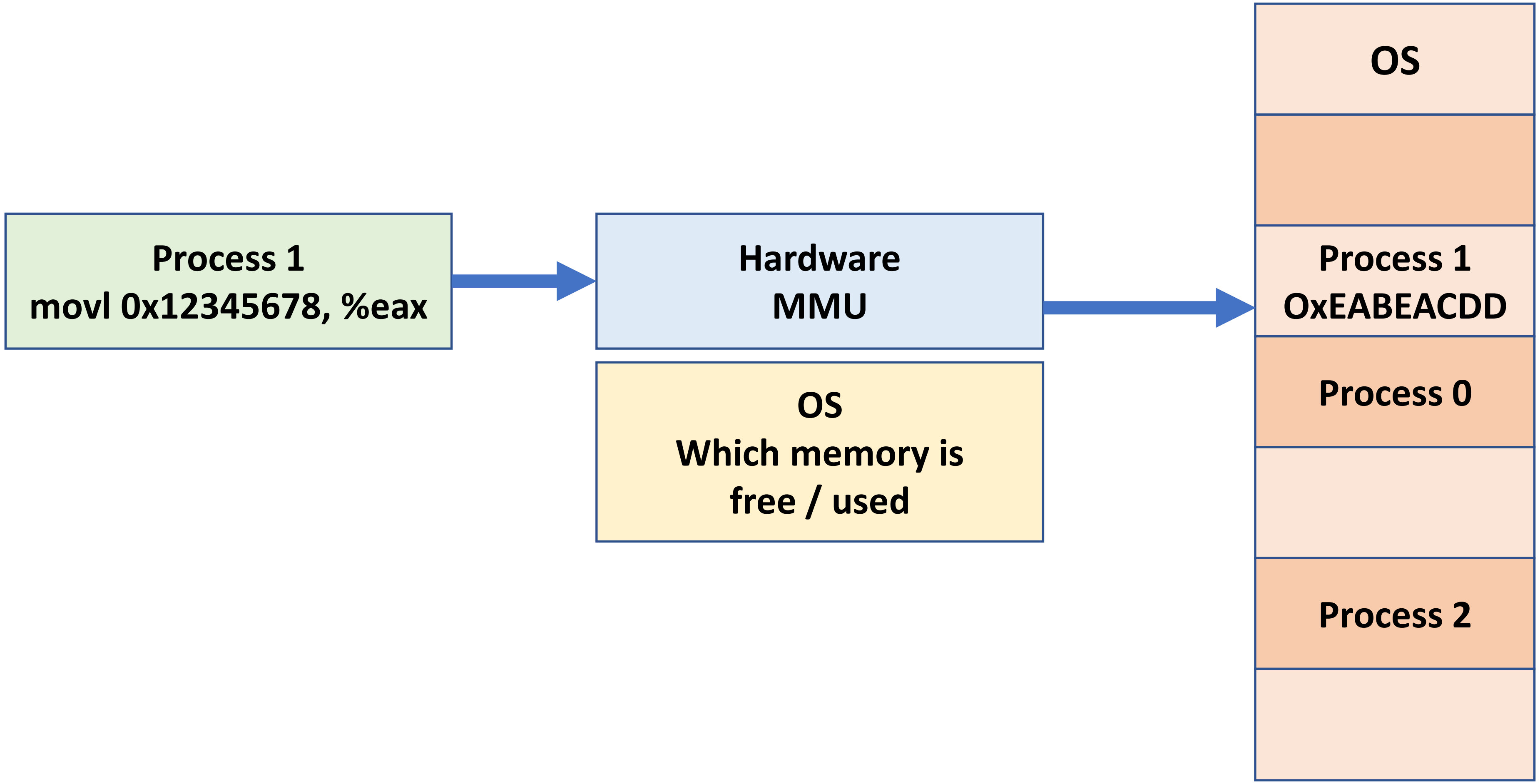
Address Space Virtualization



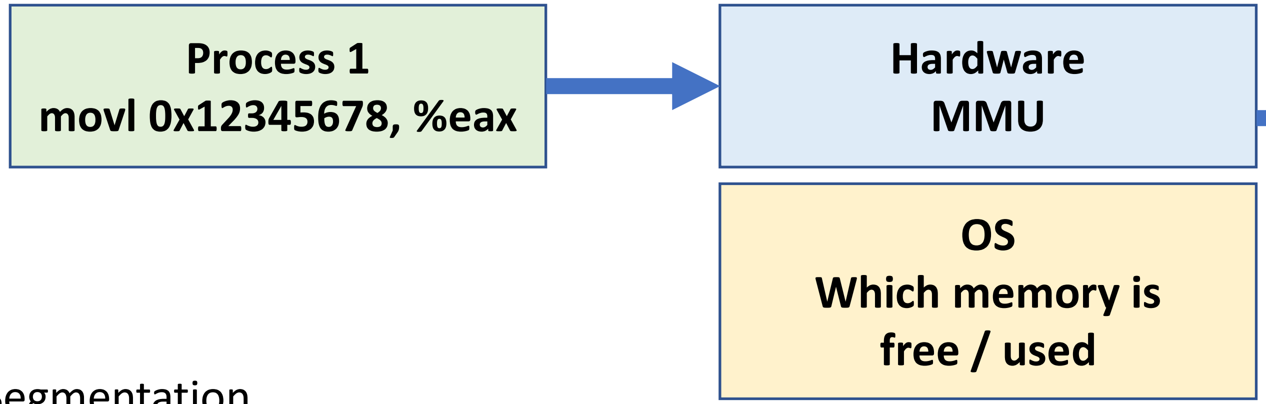
Goals

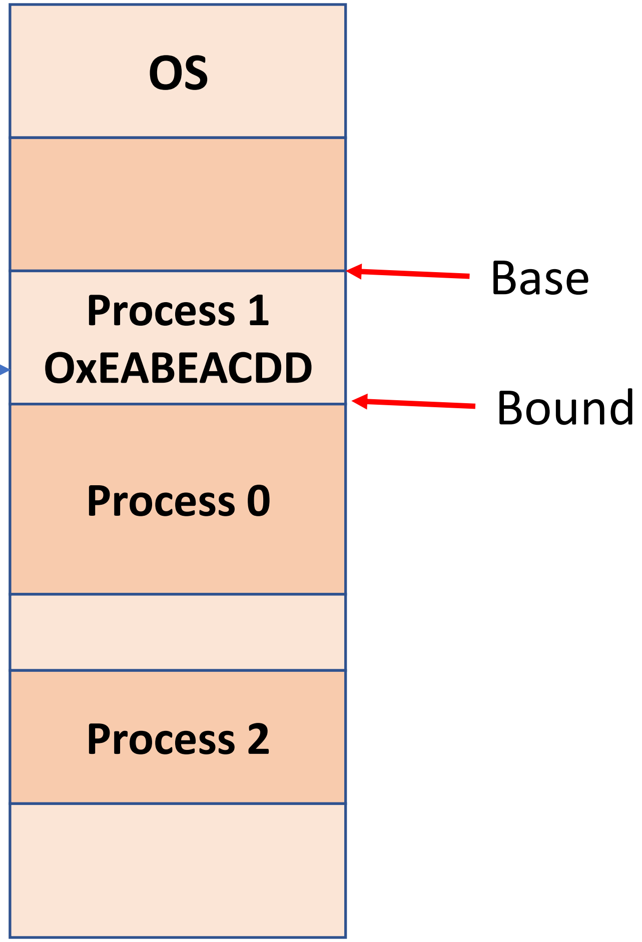
- Hide all details from the user / programmer
- Efficiency
- Protection / Isolation
- Allocation Policies – OS takes care of this
- Easy to Program

Address Space Virtualization



Segmentation

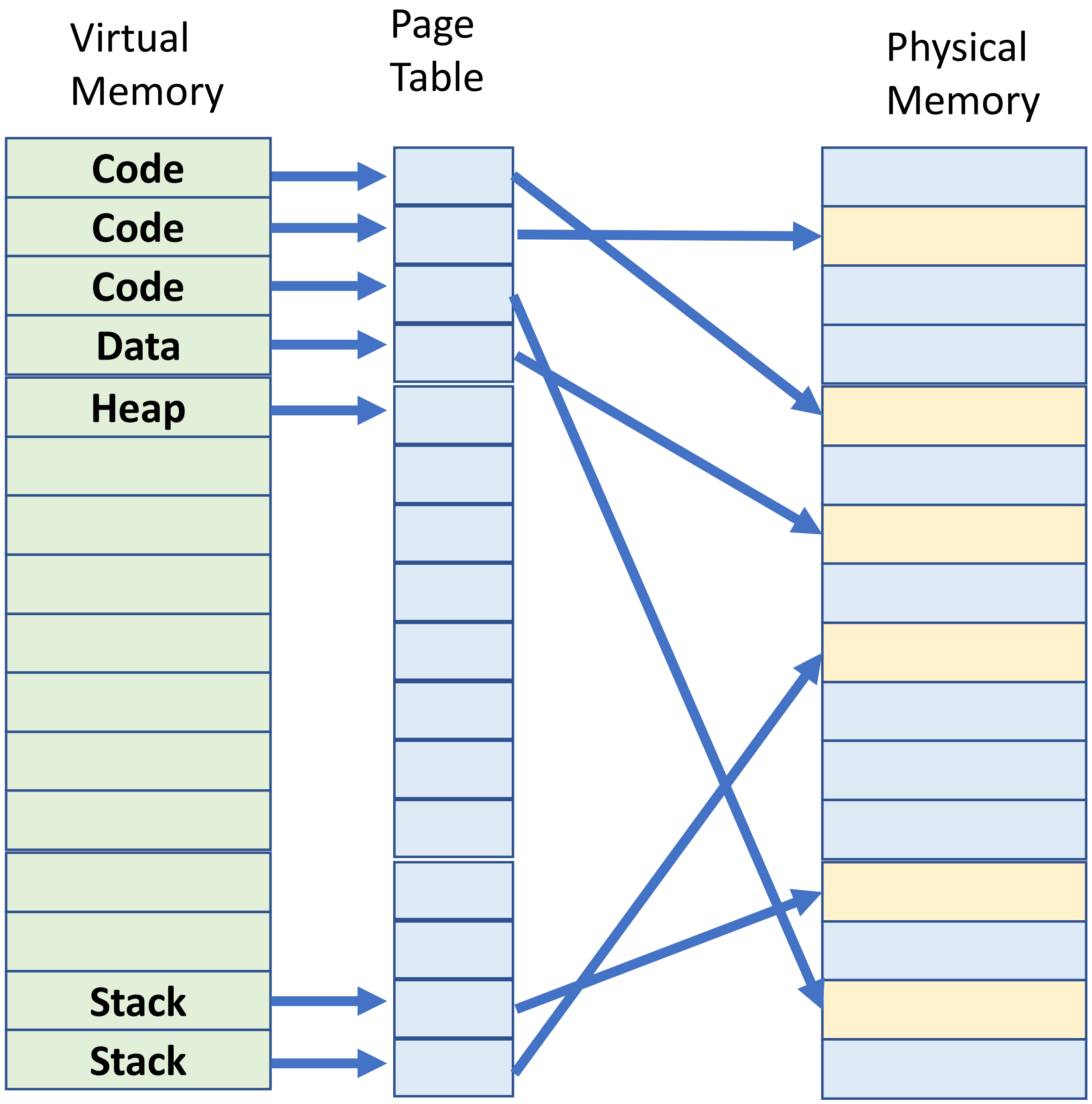


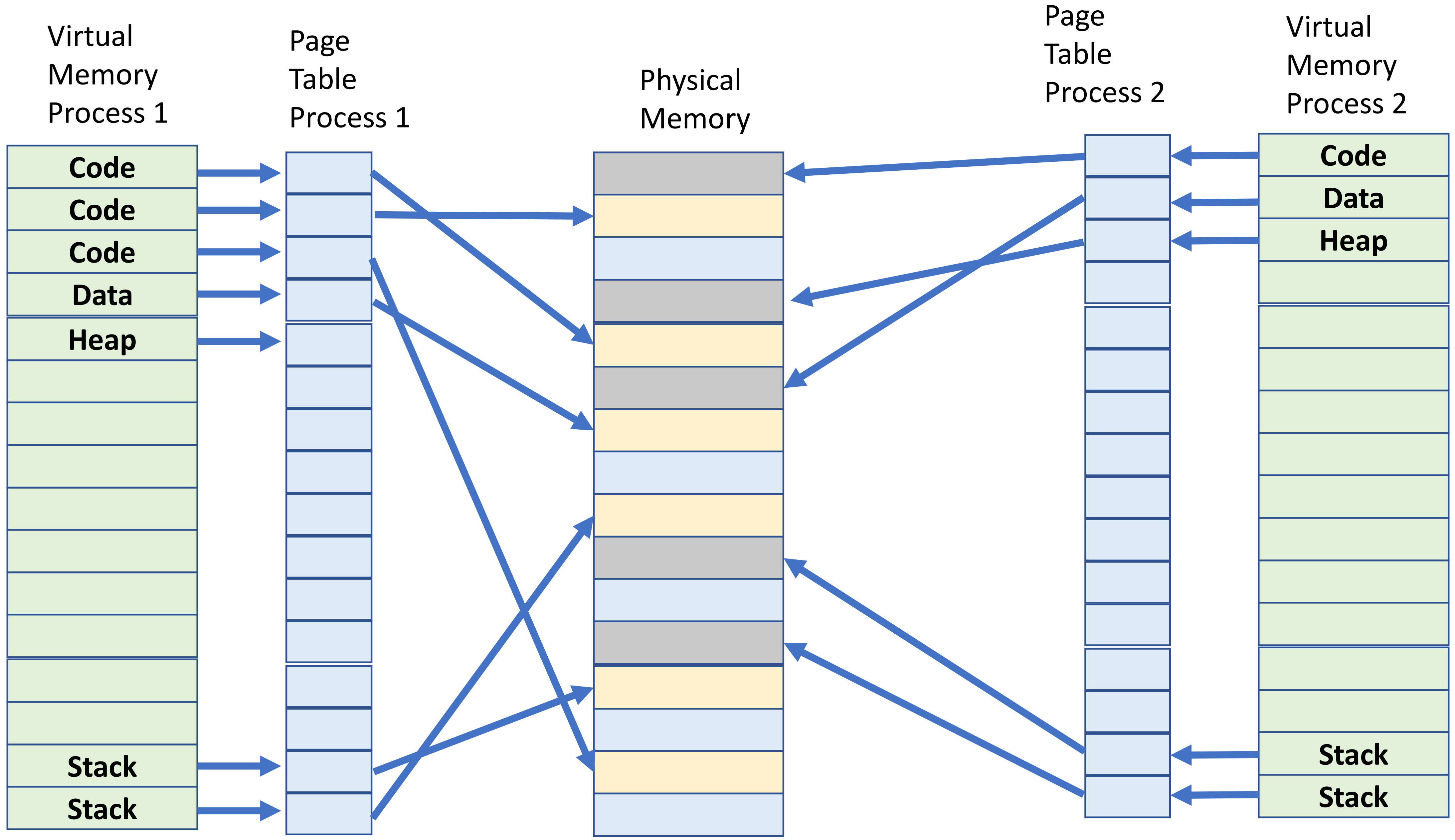


Segmentation

- Contiguous physical address space for each process
- Base and Bound Registers
- Physical Address = Virtual Address + Base
- (Must be less than bound)
- Relocatable
- External Fragmentation

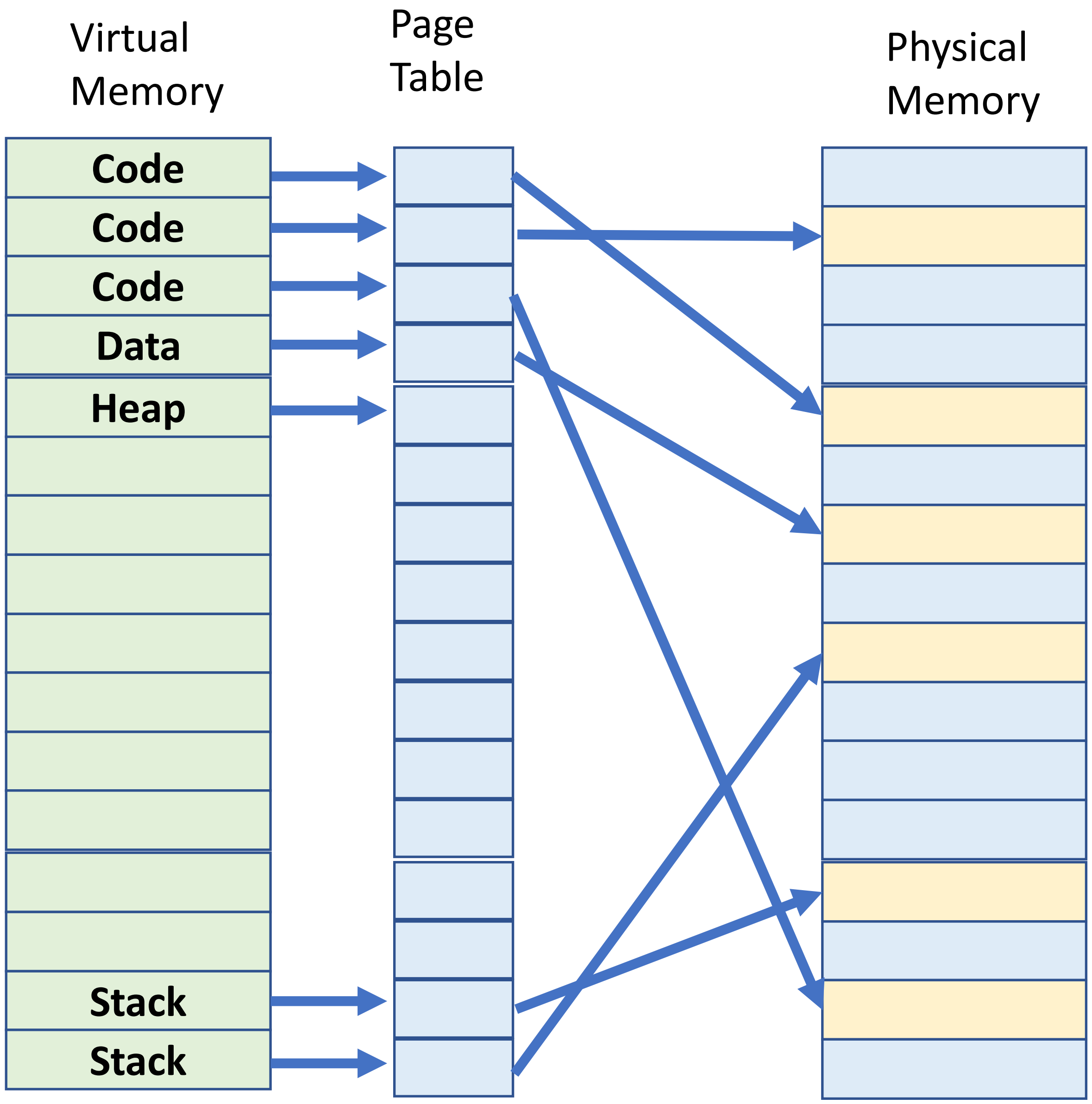
Paging


Virtual Memory
Page Table
Physical Memory
Code
Code
Code
Data
Heap
Stack
Stack



Virtual Memory Process 1
Page Table Process 1
Physical Memory
Page Table Process 2
Virtual Memory Process 2
Code
Code
Code
Data
Heap
Stack
Stack
Code
Data
Heap
Stack
Stack

Paging



- Memory divided into fixed size blocks called pages
- Example
  - page = 4096 bytes
  - 32 bit machine
  - address 0x12345678
  - page = 12345
  - offset = 678
- Each Process gets it's own page table
- Page table entries = 4 bytes

- Size of individual page table
- 2^{20} entries * 4 bytes = 4 MB

- 100 processes = 400 MB!!!

CS 354
Machine Organization and Programming

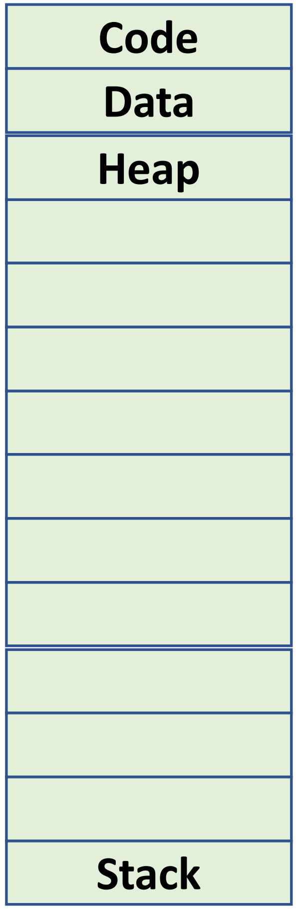
Michael Doescher
Spring 2021

Intro to Operating Systems II
- CPU Virtualization
- Memory Virtualization
- Concurrency / Threads

Multithreading

Run multiple copies of the same code at one time

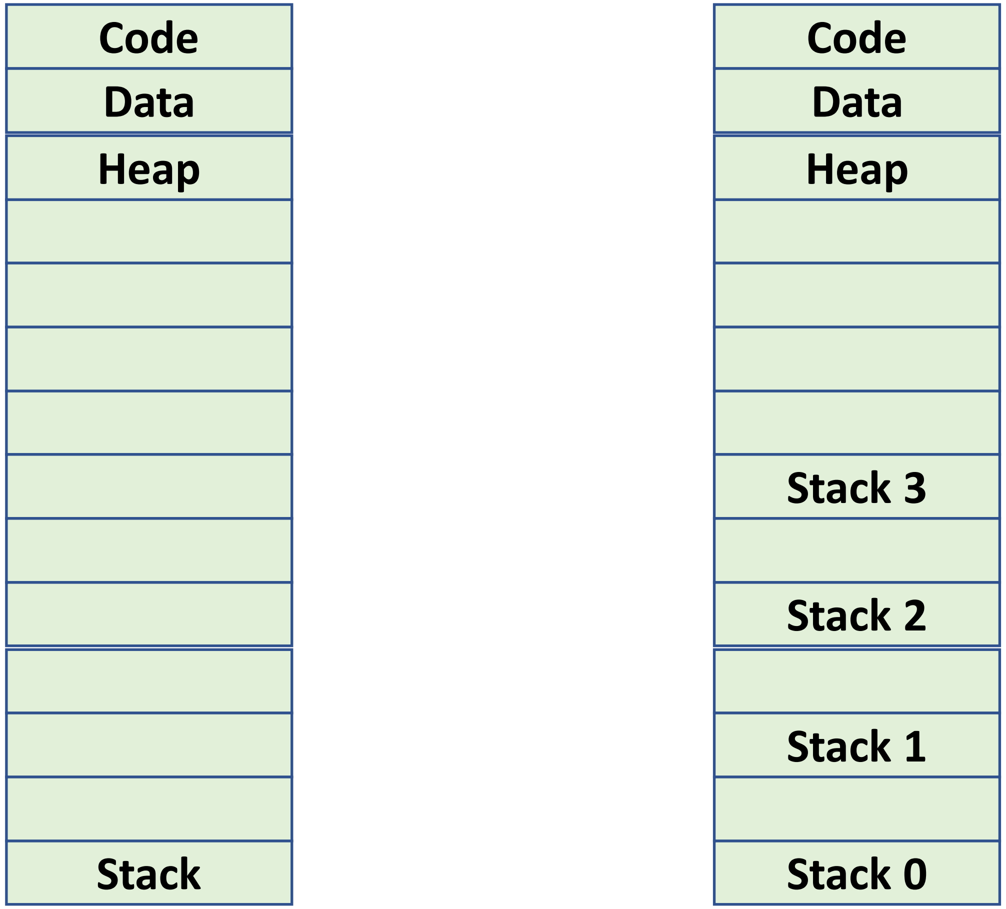
Example: Parallel processing to perform same operation on different segments of data set



Multithreading

Run multiple copies of the same code at one time

Example: Parallel processing to perform same operation on different segments of data set



Multithreading

Run multiple copies of the same code at one time

Example: Parallel processing to perform same operation on different segments of data set

Each thread gets a thread control block – and is managed (scheduled) by the OS

May execute in any order
Writing to the same memory can be a problem
"Race condition"

Atomic Instructions
Synchronization Primitives to force one thread to wait for another to finish writing