CISC 7310X

C08a Virtual Memory

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Outline

- Background and basic concepts

- Demand Paging
- Copy-on-Write
- Page Replacement
- Allocation of Frames
- Thrashing
- Memory-Mapped Files
- Allocating Kernel Memory
- Other Considerations
- Operating-System Examples

Problems

- A program may become too large to fit in memory
- Collectively in a multiprogramming system, the programs exceed memory although each fits.

Observations

- Code needs to be in memory to execute, but entire program rarely used
  - Error code, unusual routines, large data structures
- Entire program code not needed at same time

Addressing the Problems

- Consider ability to execute partially-loaded program
  - Program no longer constrained by limits of physical memory
  - Each program takes less memory while running
    - more programs run at the same time (increased degree of multiprogramming)
    - Increased CPU utilization and throughput with no increase in response time or turnaround time
  - Less I/O needed to load or swap programs into memory
    - Each user program runs faster

Introducing Virtual Memory

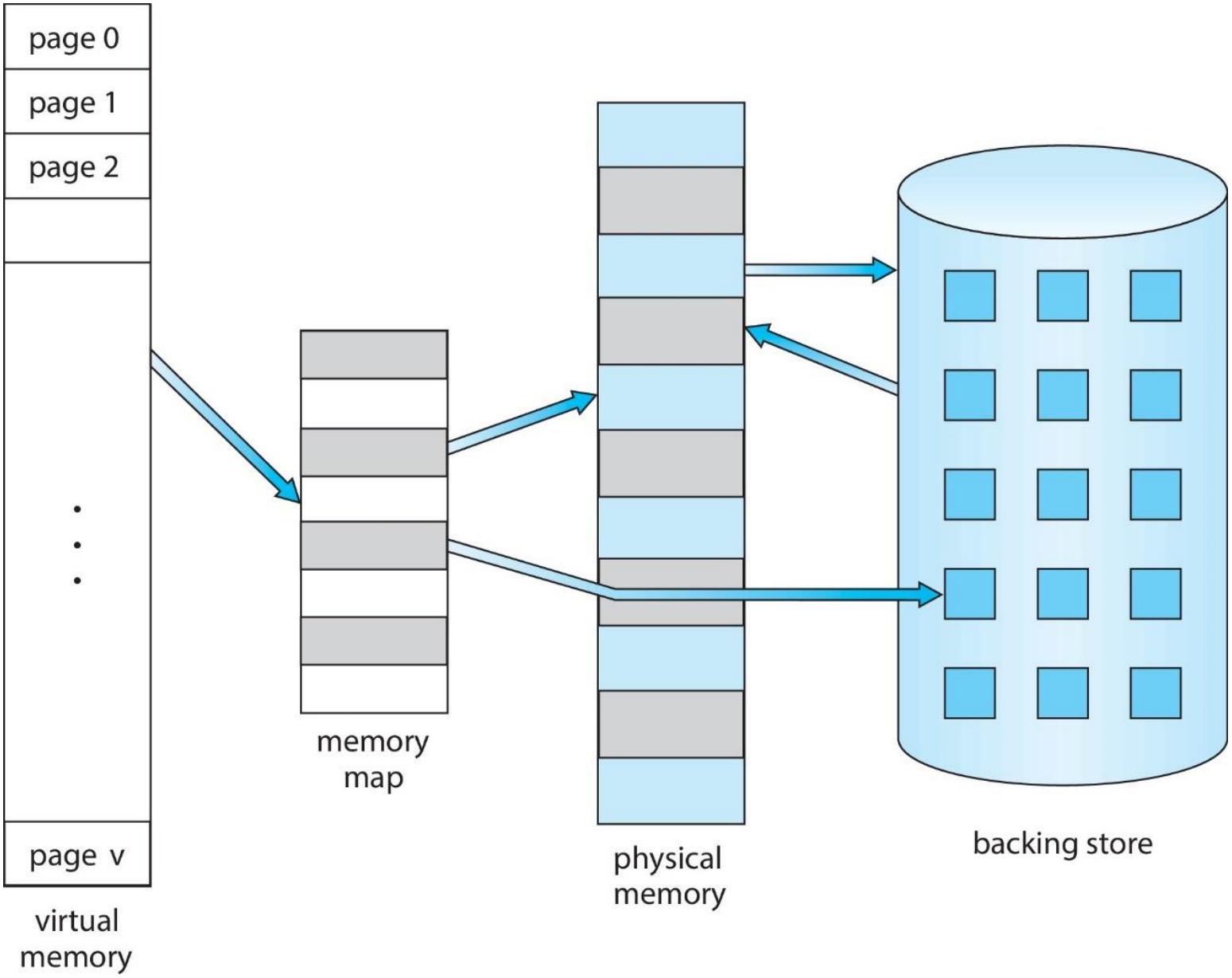
- Separation of user logical memory from physical memory
- Only part of the program needs to be in memory for execution

Logical Address and Physical Address Spaces

- Logical address space can therefore be much larger than physical address space
- Often called virtual address spaces

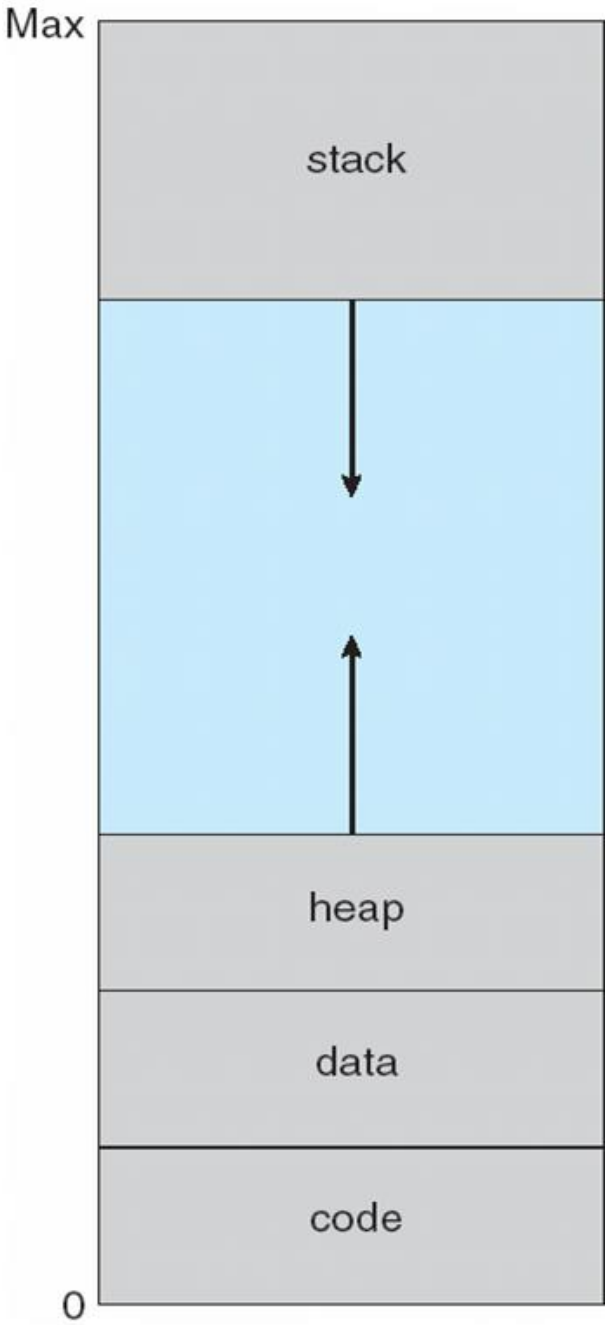
Virtual Memory

- Virtual address space/logical address space
  - logical view of how process is stored in memory
  - Usually start at address 0, contiguous addresses until end of space
  - Meanwhile, physical memory organized in page frames
  - MMU must map logical to physical


page 0
page 1
page 2
page v
virtual memory
memory map
physical memory
backing store

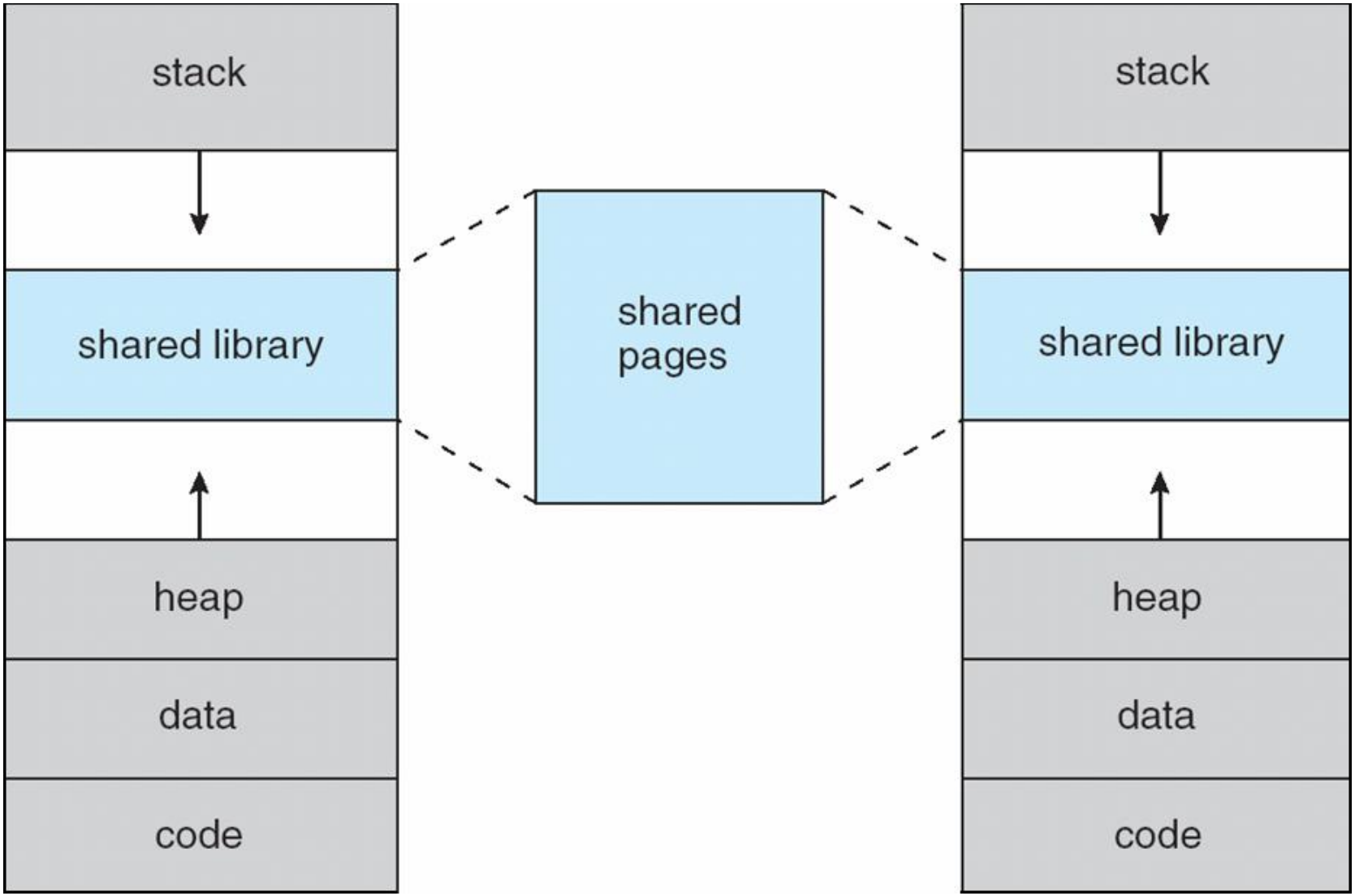
Logical Address Space: Common Design

- Usually design logical address space for stack to start at Max logical address and grow "down" while heap grows "up"
  - Maximizes address space use
  - Unused address space between the two is hole
- No physical memory needed until heap or stack grows to a given new page
- Enables sparse address spaces with holes left for growth, dynamically linked libraries, etc


Max
stack
heap
data
code
0

Sharing

- System libraries shared via mapping into virtual address space
- Shared memory by mapping pages read-write into virtual address space
- Pages can be shared during process creation (e.g., fork()), speeding process creation


stack
shared library
heap
data
code
shared pages
stack
shared library
heap
data
code

Implementing Virtual Memory

- To discuss
  - Demand paging
  - Demand segmentation

Questions?

- Concept of virtual/logical address space and physical address space
- Benefit of separating virtual/physical address space
- Concept of paging
- Benefits of paging