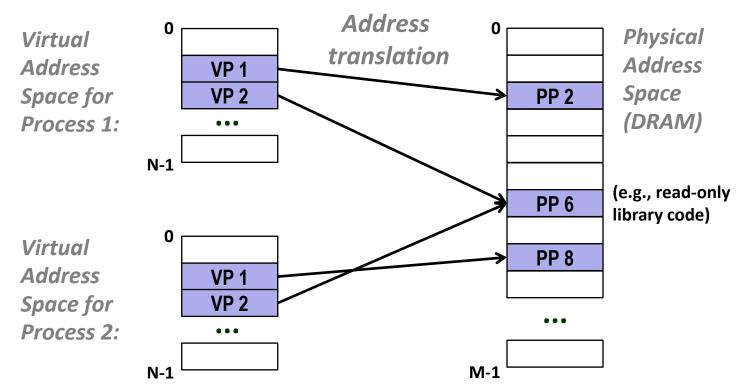
Virtual Memory: ... as a tool for memory management

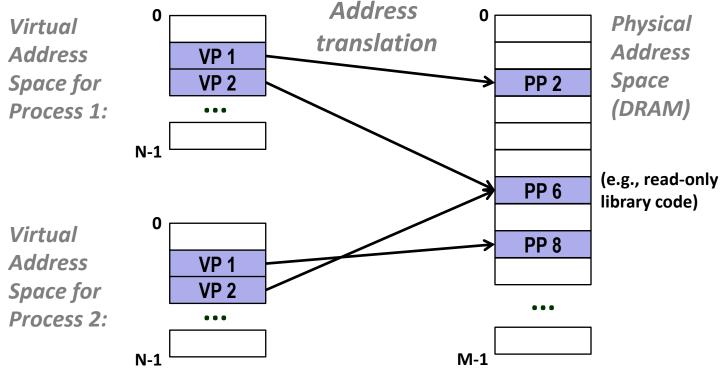
VM as a Tool for Memory Management

- Key idea: each process has its own virtual address space
 - It can view memory as a simple linear array
 - Mapping function scatters addresses through physical memory
 - Well-chosen mappings can improve locality

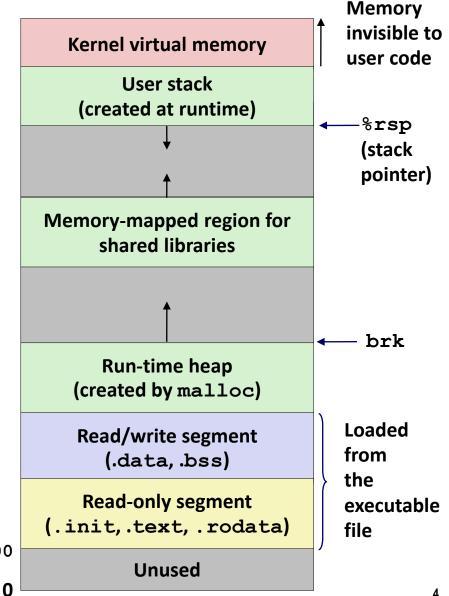


VM as a Tool for Memory Management

- Simplifying memory allocation
 - Each virtual page can be mapped to any physical page
 - A virtual page can be stored in different physical pages at different times
- Sharing code and data among processes
 - Map virtual pages to the same physical page (here: PP 6)



Simplifying Linking and Loading

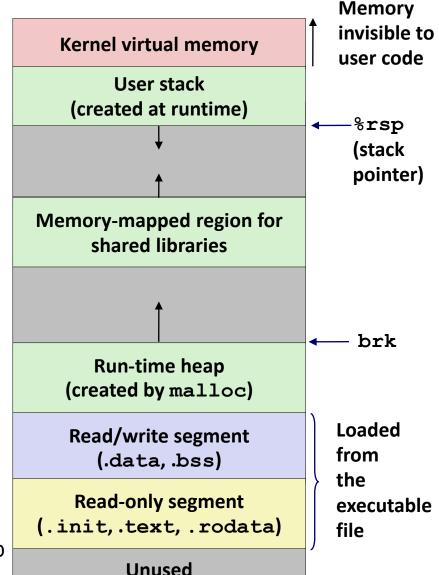


 0×400000

Simplifying Linking and Loading

Linking

- Each program has similar virtual address space
- Code, data, and heap always start at the same addresses.



 0×400000

0

Memory

Simplifying Linking and Loading

Linking

- Each program has similar virtual address space
- Code, data, and heap always start at the same addresses.

Loading

- execve allocates virtual pages for .text and .data sections & creates PTEs marked as invalid
- The .text and .data sections are copied, page by page, on demand by the virtual memory system

invisible to **Kernel virtual memory** user code User stack (created at runtime) %rsp (stack pointer) Memory-mapped region for shared libraries brk **Run-time heap** (created by malloc) Loaded Read/write segment from (.data, .bss) the **Read-only segment** executable (.init,.text,.rodata) file Unused

0x400000