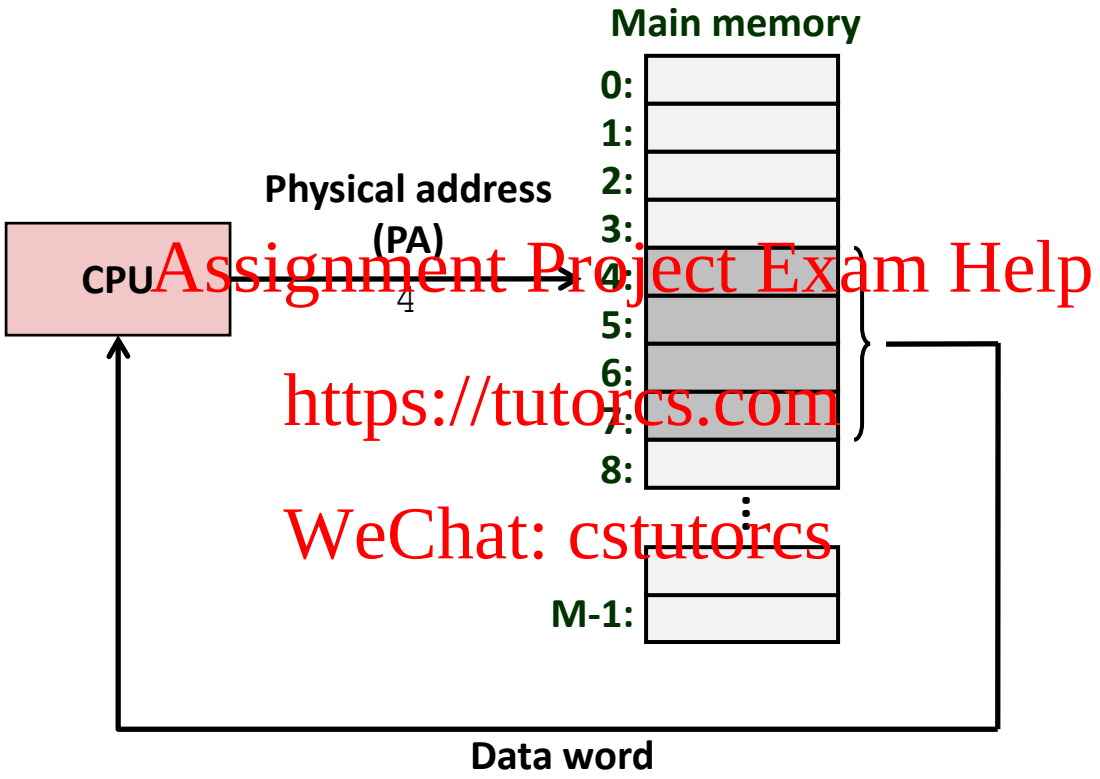
Chapter 9: Virtual Memory

Today

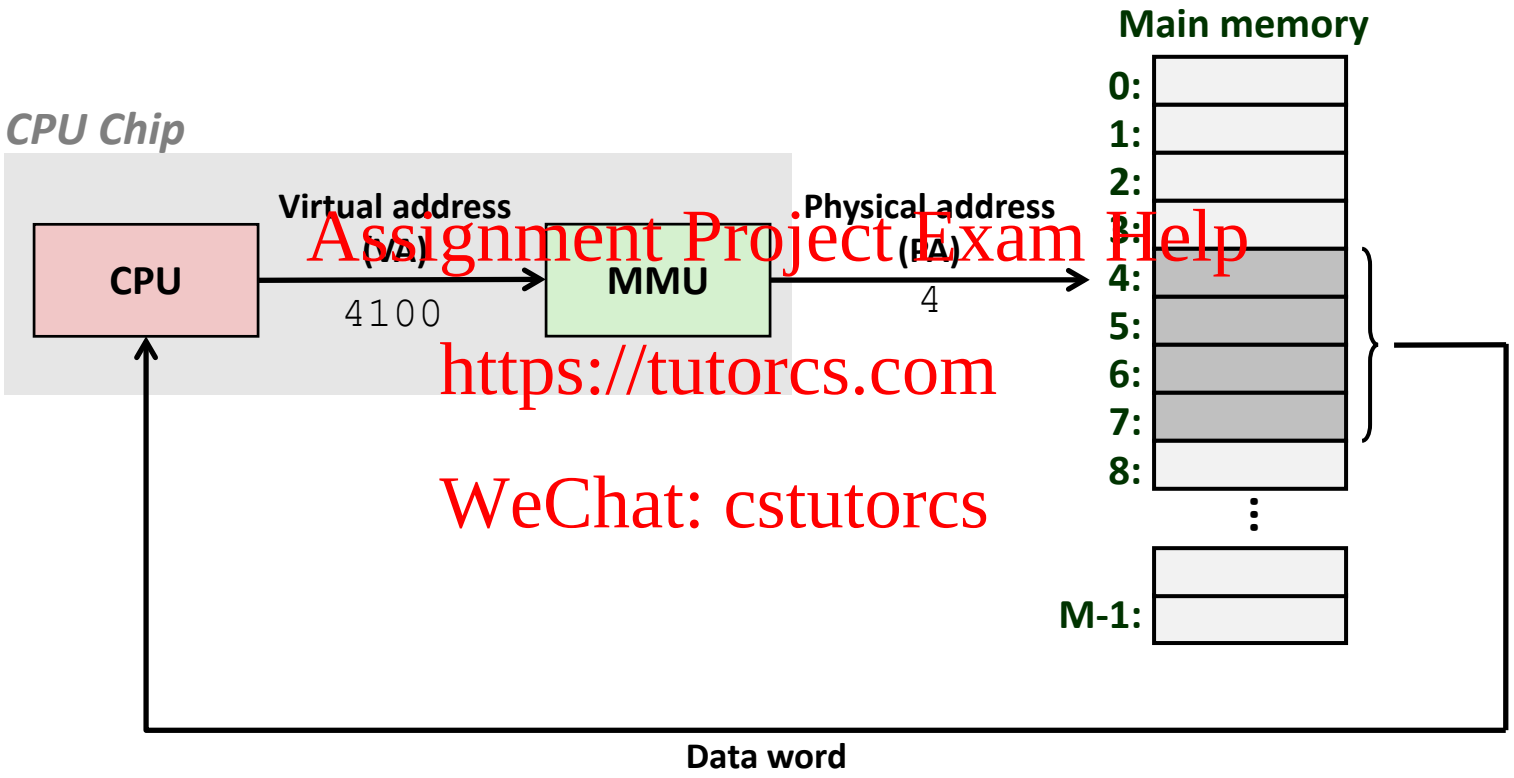
- **Address spaces**
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection
- Address translation

A System Using Physical Addressing



- Used in “simple” systems like embedded microcontrollers in devices like cars, elevators, and digital picture frames

A System Using Virtual Addressing



- Used in all modern servers, desktops, and laptops
- One of the great ideas in computer science

Address Spaces

- **Linear address space:** Ordered set of contiguous non-negative integer addresses:

  {0, 1, 2, 3 … }

- **Virtual address space:** Set of $N = 2^n$ virtual addresses

  {0, 1, 2, 3, …, N-1}

- **Physical address space:** Set of $M = 2^m$ physical addresses

  {0, 1, 2, 3, …, M-1}

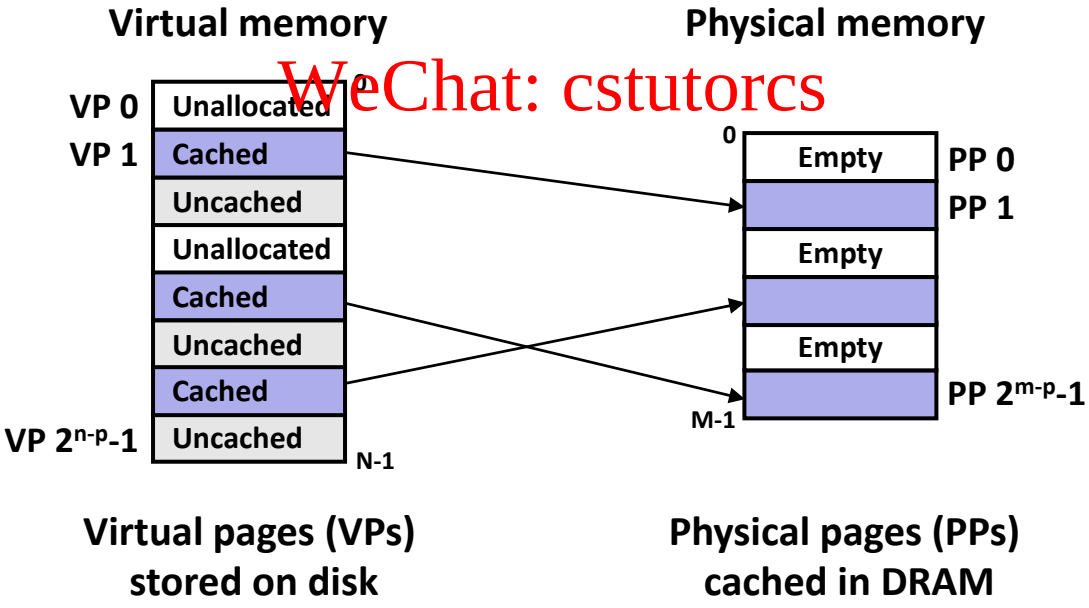
- **Clean distinction between data (bytes) and their attributes (addresses)**
- **Each object can now have multiple addresses**
- **Every byte in main memory:**
  one physical address, one (or more) virtual addresses

Why Virtual Memory (VM)?

- **Uses main memory efficiently**
  - Use DRAM as a cache for the parts of a virtual address space
- **Simplifies memory management**
  - Each process gets the same uniform linear address space
- **Isolates address spaces**
  - One process can't interfere with another's memory
  - User program cannot access privileged kernel information

VM as a Tool for Caching

- ***Virtual memory*** **is an array of N contiguous bytes stored on disk.**
- **The contents of the array on disk are cached in *physical memory* (*DRAM cache*)**
  - These cache blocks are called *pages* (size is P = 2^p bytes)



DRAM Cache Organization

- **DRAM cache organization driven by the enormous miss penalty**
  - DRAM is about ***10x*** slower than SRAM
  - Disk is about ***10,000x*** slower than DRAM

- **Consequences**

  - Large page (block) size: typically 4-8 KB, sometimes 4 MB
  - Fully associative

    - Any VP can be placed in any PP
    - Requires a “large” mapping function – different from CPU caches
  - Highly sophisticated, expensive replacement algorithms
    - Too complicated and open-ended to be implemented in hardware
  - Write-back rather than write-through