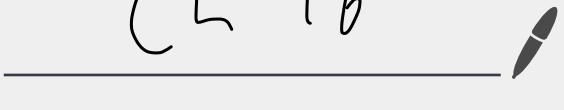
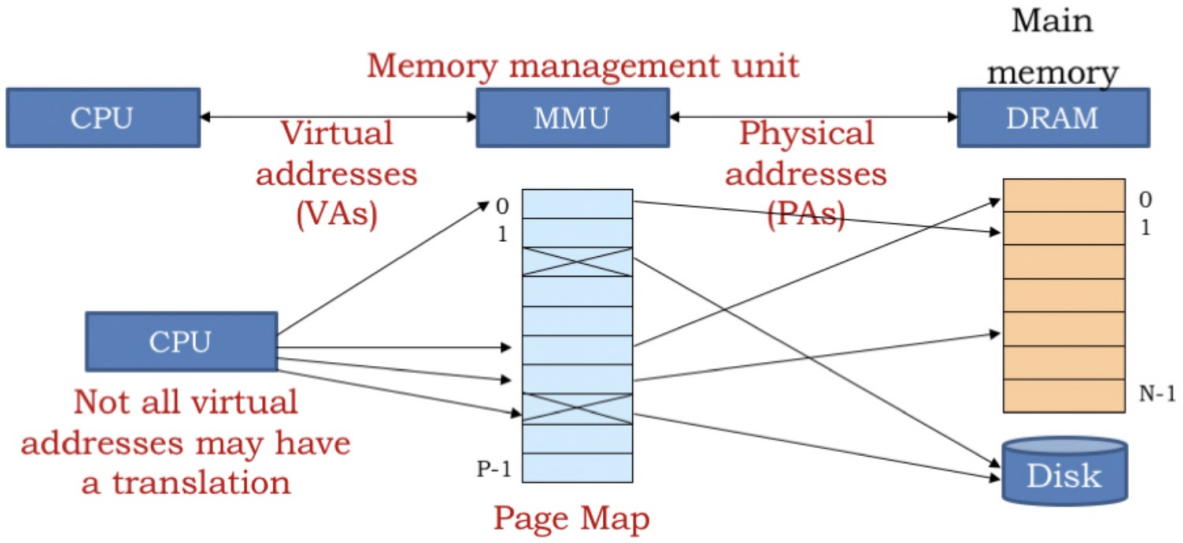

Ch 16

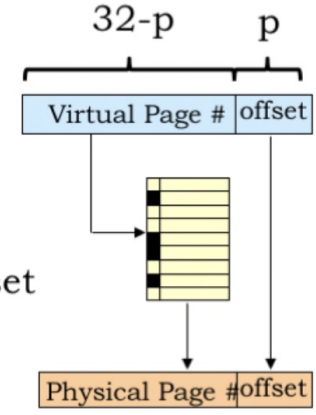
Virtual Memory

- Two kinds of addresses:
  - CPU uses virtual addresses
  - Main memory uses physical addresses
- Hardware translates virtual addresses to physical addresses via an operating system (OS)-managed table, the page map



Virtual Memory Implementation: Paging

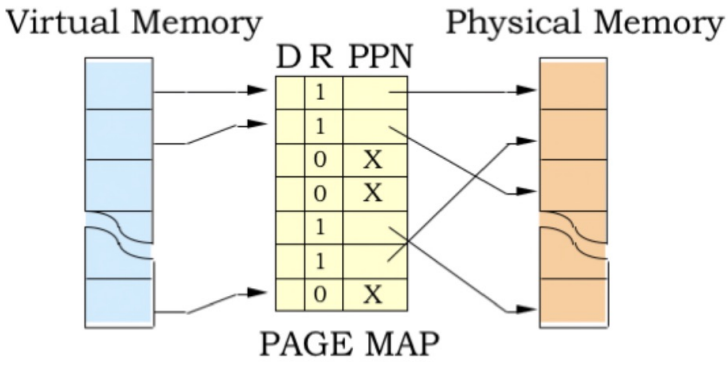
- Divide physical memory in fixed-size blocks, called pages
  - Typical page size (2^p): 4KB -16 KB
  - Virtual address: Virtual page number + offset bits
  - Physical address: Physical page number + offset bits
  - Why use lower bits as offset?



- MMU maps virtual pages to physical pages
  - Use page map to perform translation
  - Cause a page fault (a miss) if virtual page is not resident in physical memory.

Using main memory as a page cache = *paging* or *demand paging*

Simple Page Map Design

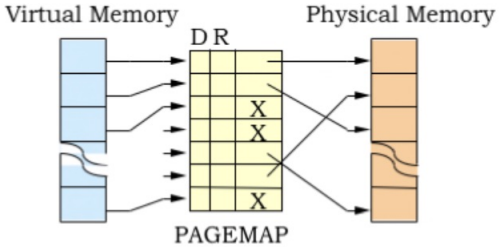


One entry per virtual page

- Resident bit R = 1 for pages stored in RAM, or 0 for non-resident (disk or unallocated). Page fault when R = 0
- Contains physical page number (PPN) of each resident page
- Dirty bit D = 1 if we've changed this page since loading it from disk (and therefore need to write it to disk when it's replaced)

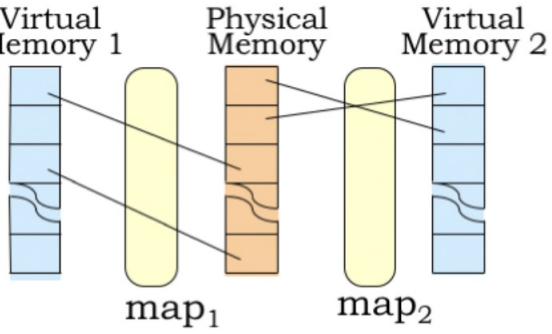
Contexts

A *context* is a mapping of VIRTUAL to PHYSICAL locations, as dictated by contents of the page map:



Several programs may be simultaneously loaded into main memory, each in its separate context:

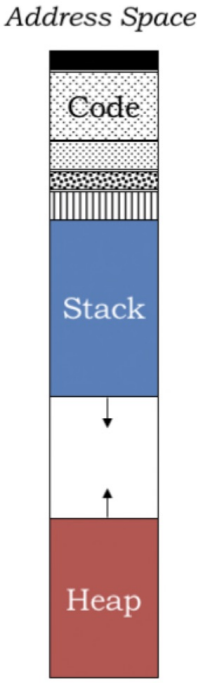
"Context switch": reload the page map?



Memory Management & Protection

- Applications are written as if they have access to the entire virtual address space, without considering where other applications reside
  - Enables fixed conventions (e.g., program starts at 0x1000, stack is contiguous and grows up, …) without worrying about conflicts

- OS Kernel controls all contexts, prevents programs from reading and writing into each other's memory



Summary: Virtual Memory

- Goal 1: Exploit locality on a large scale
  - Programmers want a large, flat address space, but use a small portion!
  - Solution: Cache working set into RAM from disk
  - Basic implementation: MMU with single-level page map
    - Access loaded pages via fast hardware path
    - Load virtual memory on demand: page faults
  - Several optimizations:
    - Moving page map to RAM, for cost reasons
    - Translation Lookaside Buffer (TLB) to regain performance
  - Cache/VM interactions: Can cache physical or virtual locations
- Goals 2 & 3: Ease memory management, protect multiple contexts from each other
  - We'll see these in detail on the next lecture!