Lecture 09

Virtual Memory

이재진

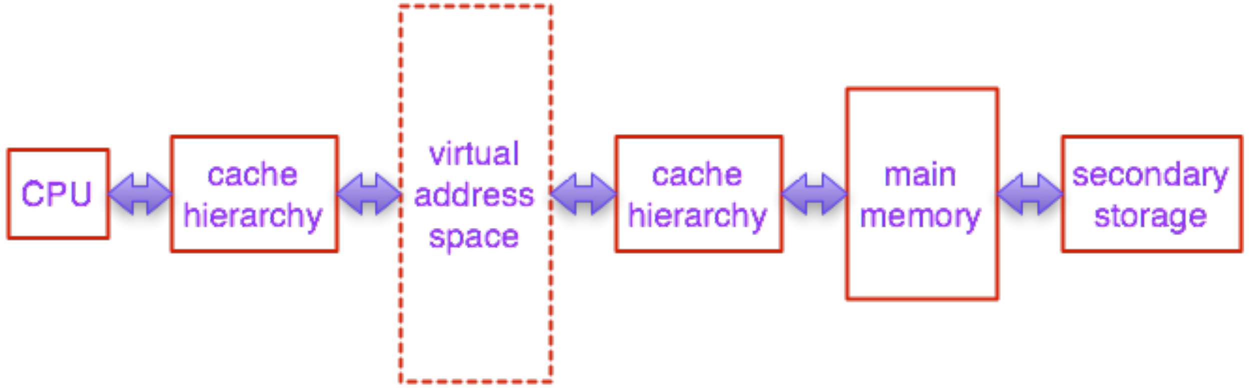
서울대학교 컴퓨터공학부

http://aces.snu.ac.kr

THUNDER Research Group
Seoul National University
서울대학교 천둥 연구실

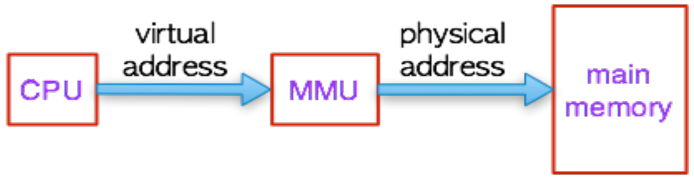
Virtual Memory

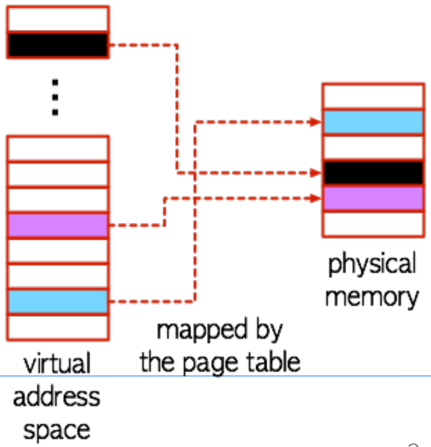
- An abstraction of main memory by the operating system
  - Provide each process with a large and uniform address space
    - The size of the address space is bigger than that of main memory
  - Protect the address space of each process from corruption by other processes
  - Treat main memory as a cache of the permanent secondary storage (hard disk)



MMU and Pages

- Each byte in main memory has a unique physical address (PA)
- The CPU generates a virtual address (VA) to access the main memory
- The memory management unit (MMU) translates the virtual address to the corresponding physical address using a look-up table (page table) stored in main memory
- The virtual address space is divided into uniform virtual pages
  - Each page is indexed by its virtual page number
- The physical memory is divided into uniform physical pages (page frames)
  - Each frame is indexed by its page frame number



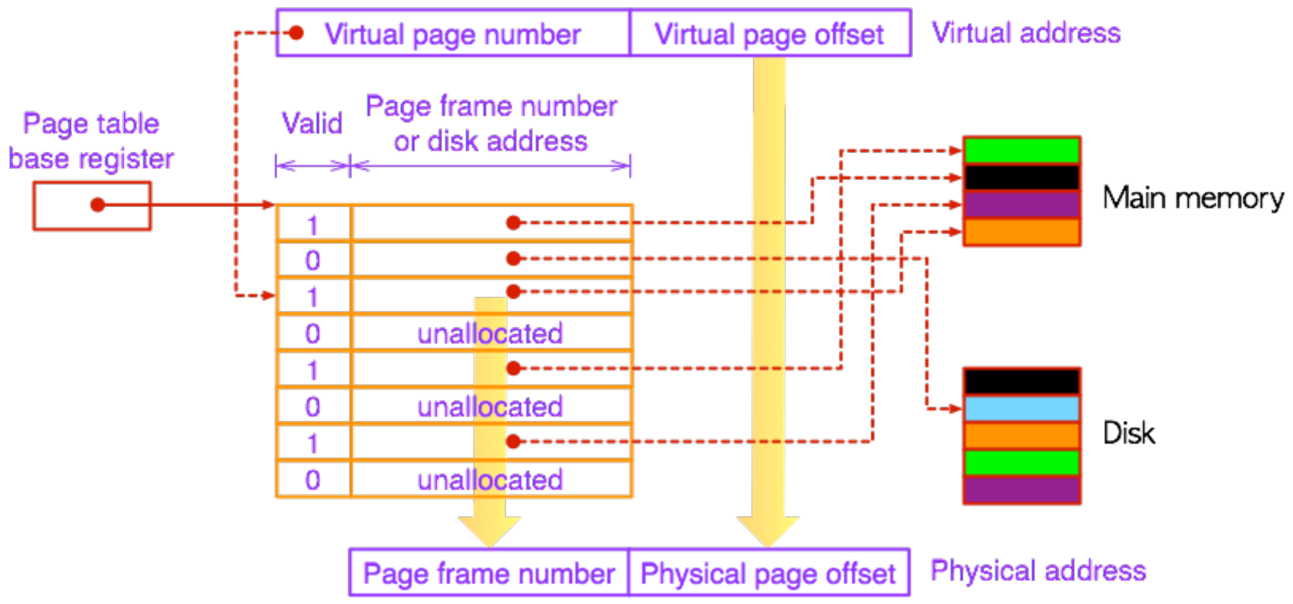


Page Tables

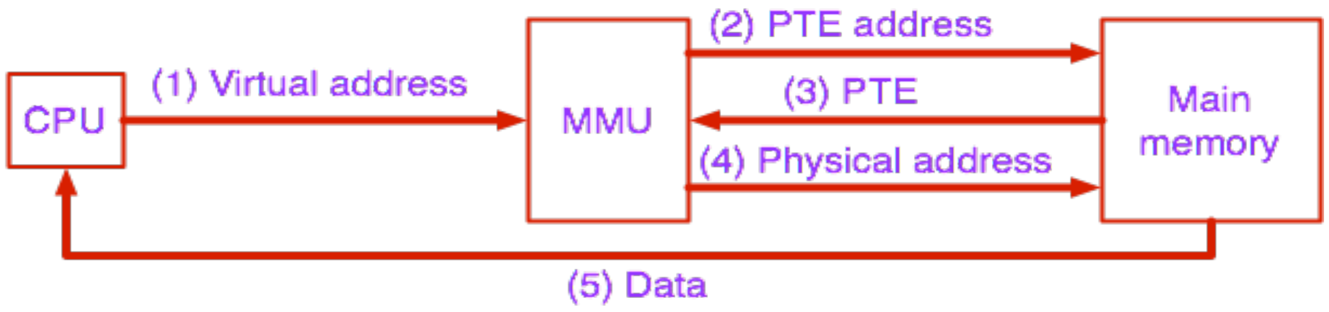
- Map virtual pages to physical pages
  - An array of page table entries (PTE)
  - A PTE consists of a valid bit and an n-bit address field (physical page frame number or secondary storage address) in addition to other page attributes
- The MMU reads the page table when it converts VA to PA
- The OS (page fault handler) takes care of maintaining the contents of the page table and transferring pages between main memory and secondary storage
- Swapping (paging)
  - The activity of transferring a page between the secondary storage and main memory
- Demand paging
  - Wait until the last moment to swap in a page when a miss (page fault) occurs

Address Translation

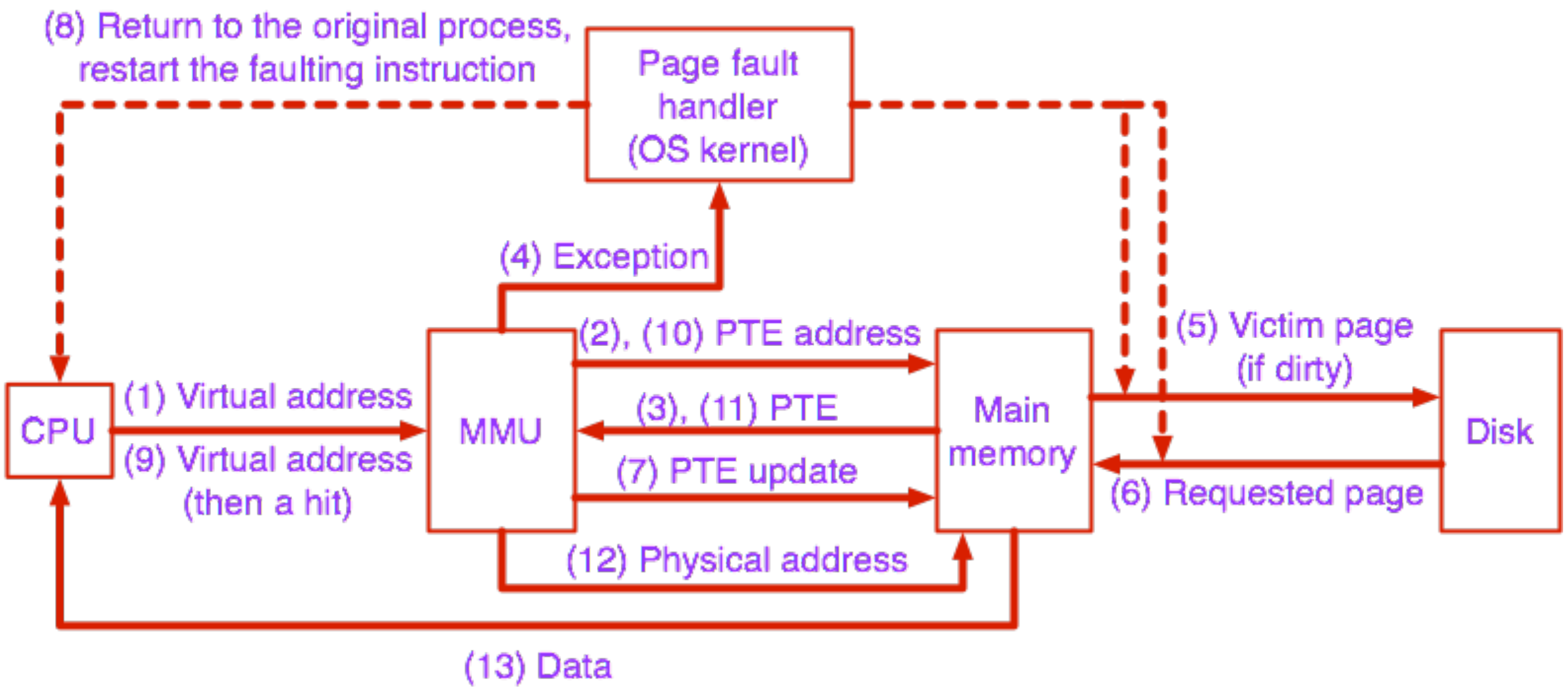
- Three types of virtual pages
  - Unallocated: Pages that have not yet been allocated by the VM (no space on secondary storage)
  - Cached: Allocated pages that are currently cached in main memory
  - Uncached: Allocated pages that are not cached in main memory (reside on secondary storage)
- A single page table for the entire address space is large
  - 32-bit address space, 4KB pages, and 4B PTEs result in 4MB page table resident in main memory
  - Use a hierarchy of page tables and demand paging for the tables



Page Hit

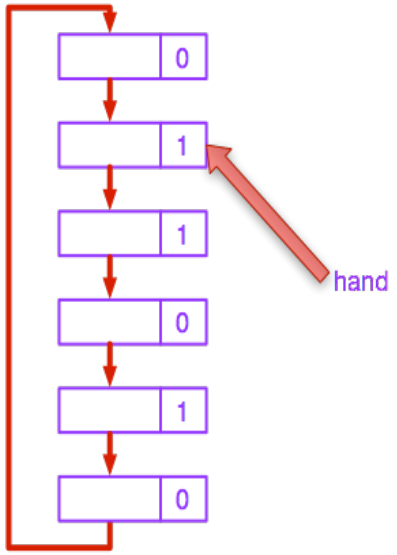

CPU
(1) Virtual address
MMU
(2) PTE address
(3) PTE
(4) Physical address
Main memory
(5) Data

Page Fault


(8) Return to the original process, restart the faulting instruction
Page fault handler (OS kernel)
(4) Exception
(2), (10) PTE address
(3), (11) PTE
(7) PTE update
(5) Victim page (if dirty)
(6) Requested page
CPU
(1) Virtual address
(9) Virtual address (then a hit)
MMU
Main memory
Disk
(12) Physical address
(13) Data

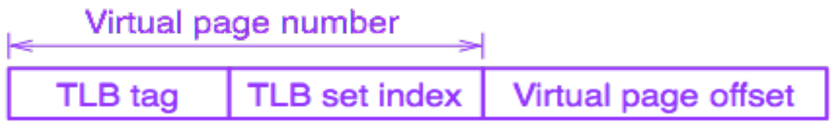
Page Replacement Policies

- LRU
- FIFO
- Second chance
- Clock
  - A bit (R) that indicates whether the page is referenced or not
    - When a page is first loaded in memory, R = 0
    - When the page is referenced, R = 1
  - Maintain a circular list of pages in memory
    - The hand points to the current page in the list
    - When it is time to replace a page, the first frame with R = 0 encountered is replaced
    - During the search for replacement, each reference bit set to 1 is changed to 0

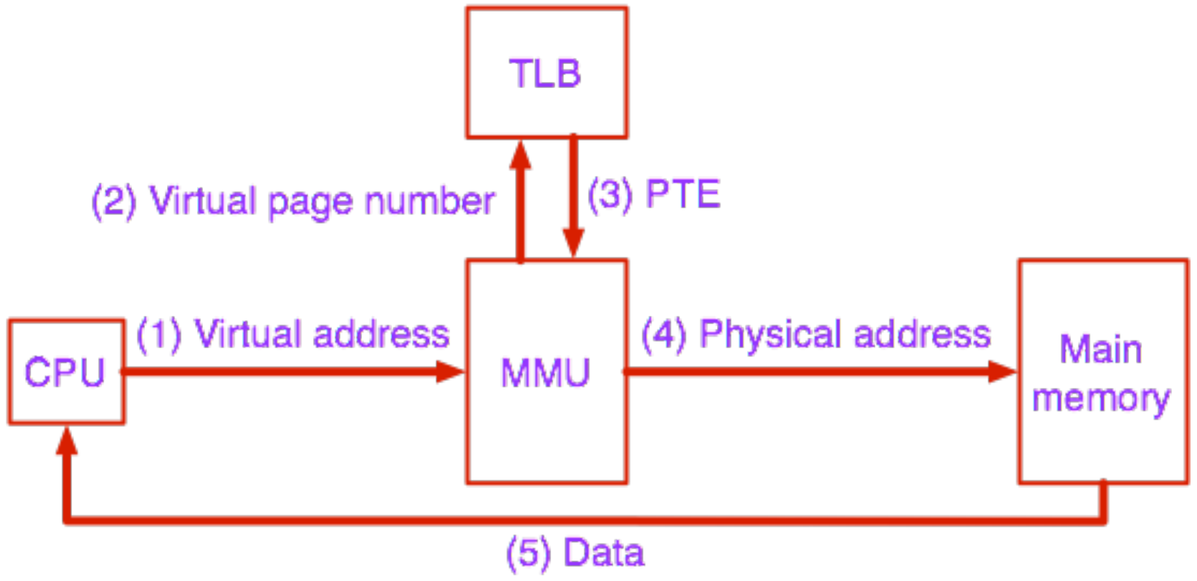


Translation Lookaside Buffer (TLB)

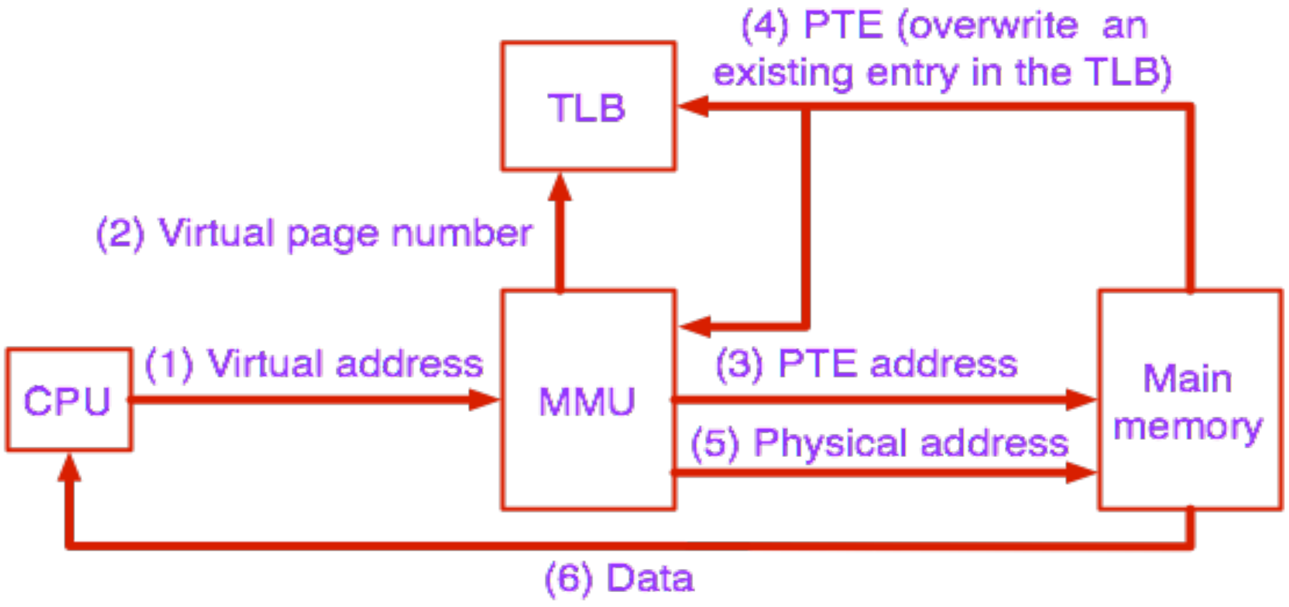
- Every time the CPU generates a virtual address, the MMU must refer to the page table for address translation
    - High overhead
- A small, virtually addressed cache where each line holds a block consisting of a single PTE
    - Has a high degree of associativity
- Micro-TLB
    - A small TLB placed over the main TLB to boost the speed of address translation for cache accesses
    - The main TLB handles micro-TLB misses
    - Smaller number of entries than the main TLB



TLB Hit


TLB
(2) Virtual page number
(3) PTE
CPU
(1) Virtual address
MMU
(4) Physical address
Main memory
(5) Data

TLB Miss

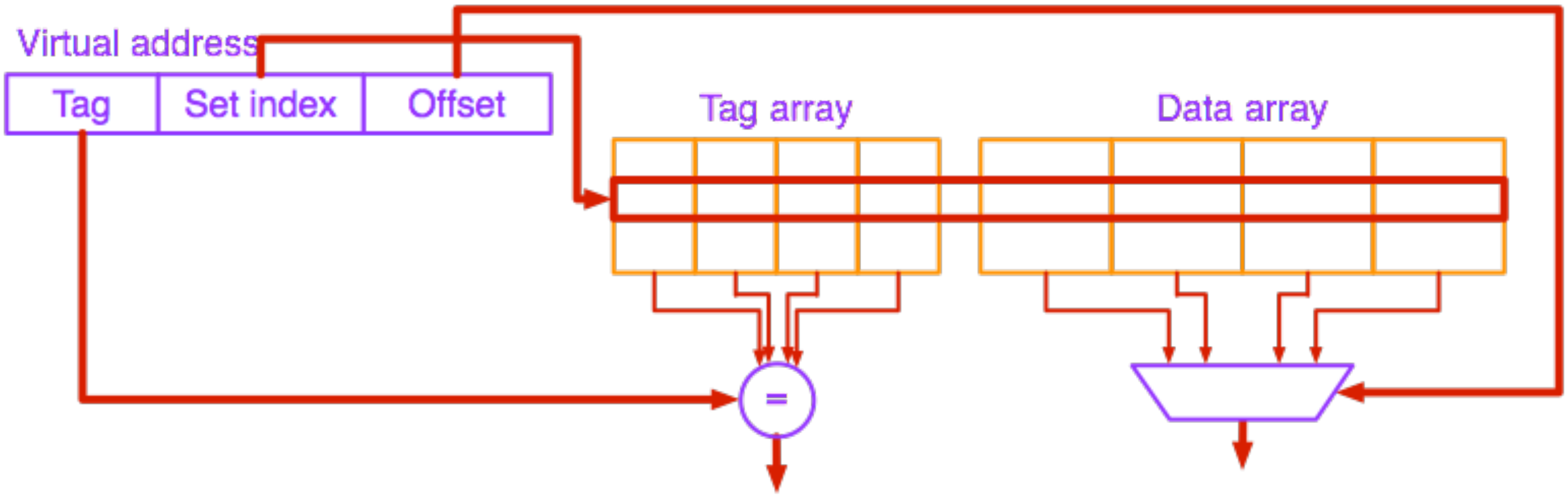

TLB
(4) PTE (overwrite an existing entry in the TLB)
(2) Virtual page number
CPU
(1) Virtual address
MMU
(3) PTE address
(5) Physical address
Main memory
(6) Data

Caches and Virtual Memory

- Virtually-addressed caches vs. physically-addressed caches
  - Which address do we send to the cache?
  - Virtually-addressed cache: faster (no address translation) but security issues (requires cache flushing by the OS on context switching)
  - Physically-addressed cache: slower but no security issues (no OS intervention)
- Four possible combinations
  - Physically indexed, physically tagged
  - Physically indexed, virtually tagged
  - Virtually indexed, physically tagged
  - Virtually indexed, virtually tagged

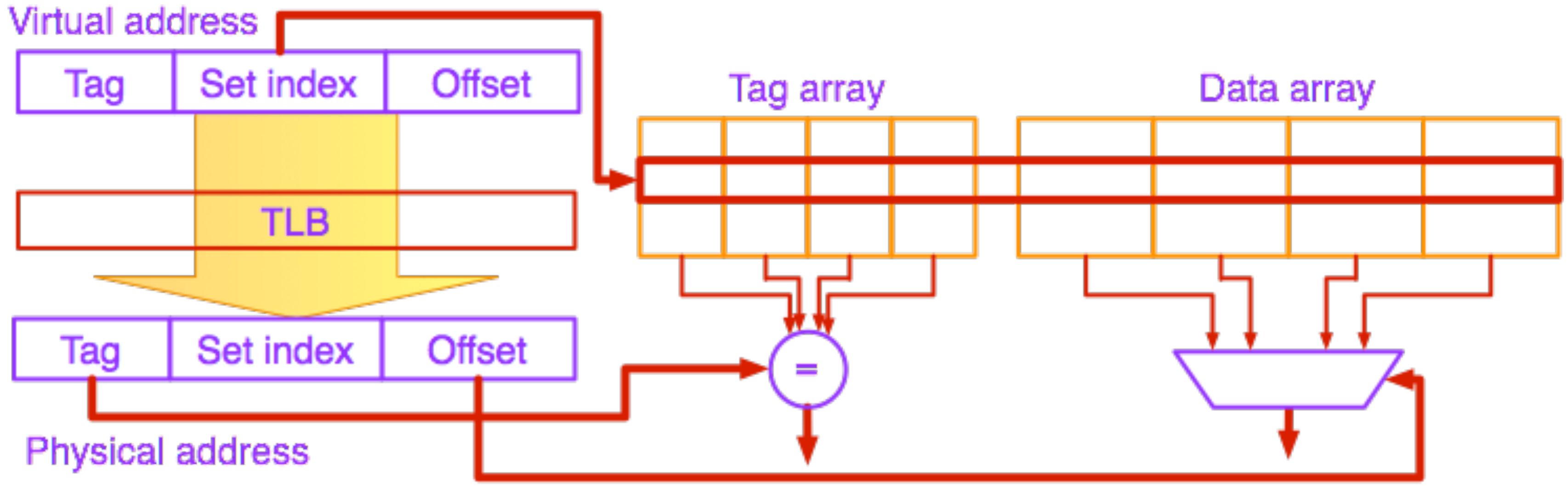
Virtually Indexed, Virtually Tagged

- Address translation occurs on a cache miss
- TLB (address translation) is not in the critical path



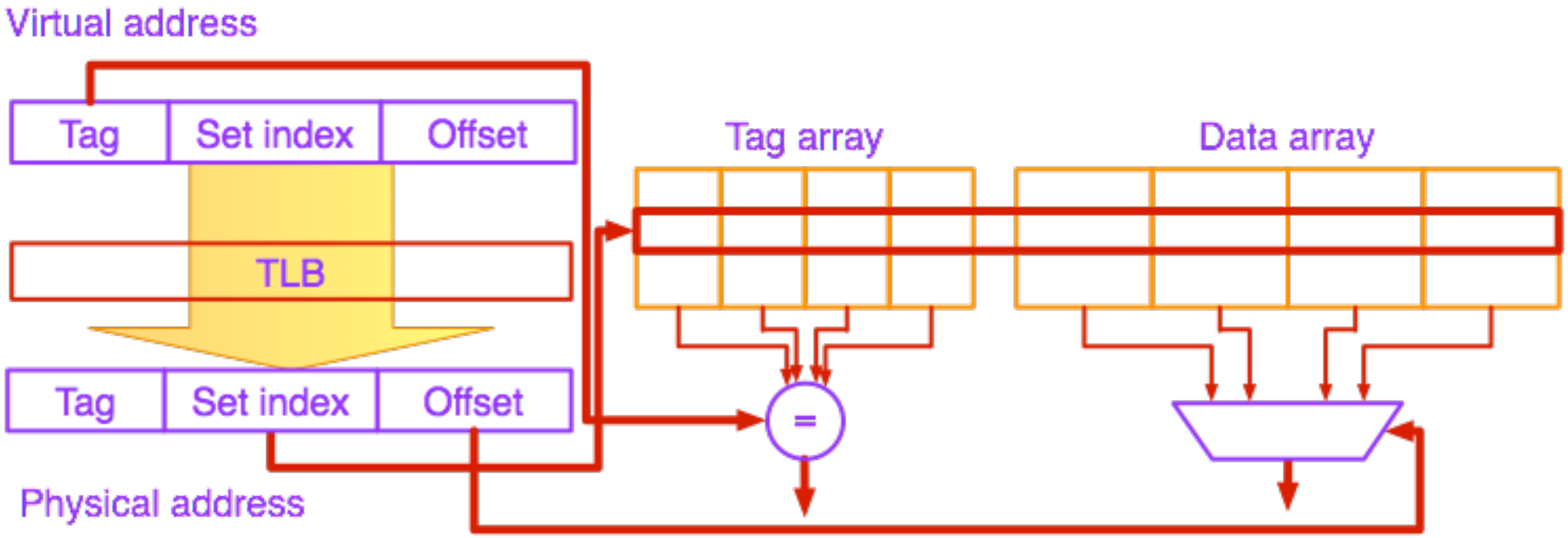
Virtually Indexed, Physically Tagged

- Common in real systems
- The address translation can happen at the same time as the cache indexing
- TLB is not in the critical path
- Much faster than physically-indexed caches



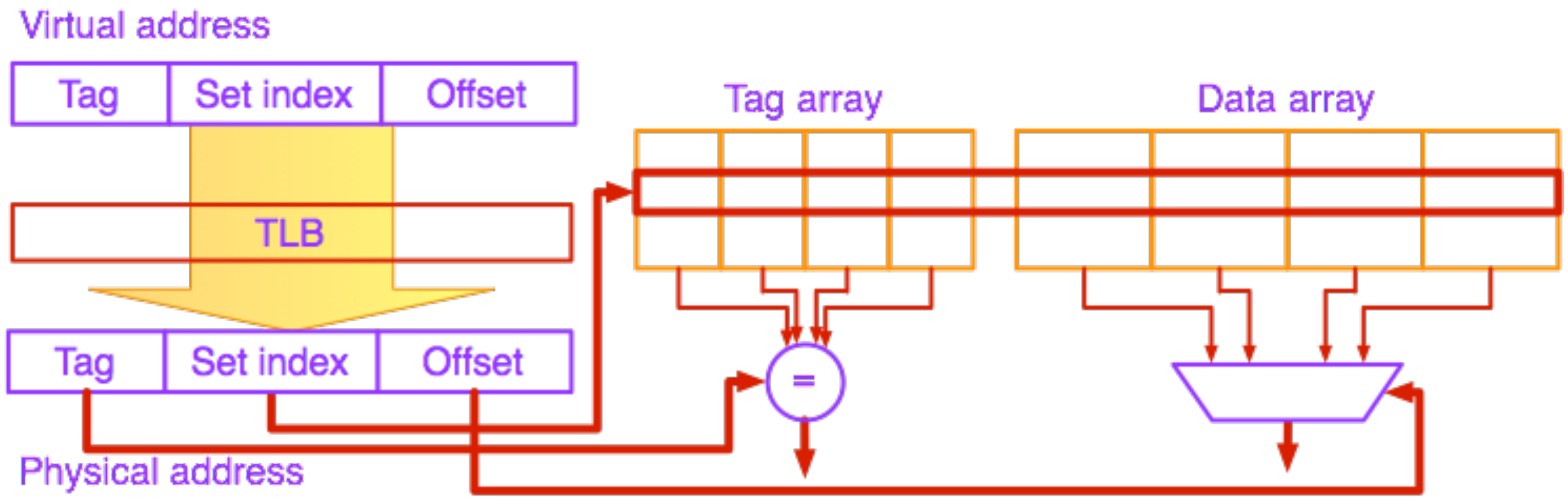
Physically Indexed, Virtually Tagged

- Never used
- No OS intervention for cache management
- TLB is in the critical path



Physically Indexed, Physically Tagged

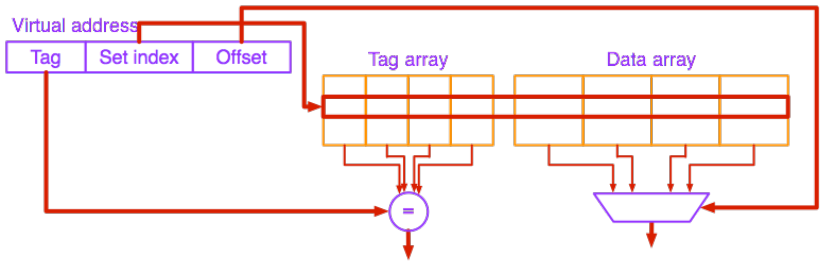
- No OS intervention for cache management
- TLB is in the critical path



Virtual Addressing vs. Physical Addressing

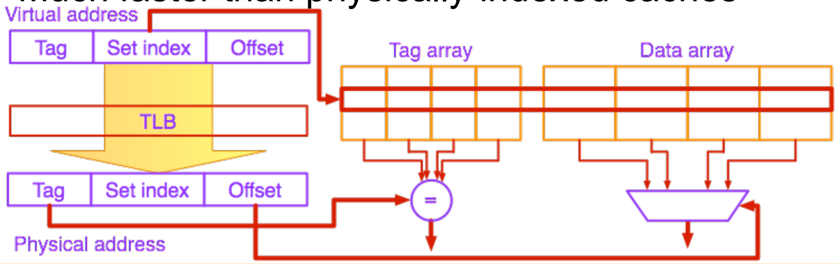
Virtually indexed, virtually tagged

- Address translation occurs on a cache miss
  - TLB (address translation) is not in the critical path



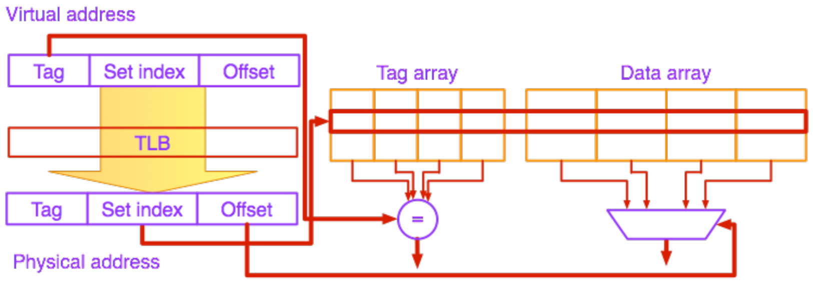
Virtually indexed, physically tagged

- Common in real systems
- The address translation can happen at the same time as the cache indexing
  - TLB is not in the critical path
- Much faster than physically-indexed caches



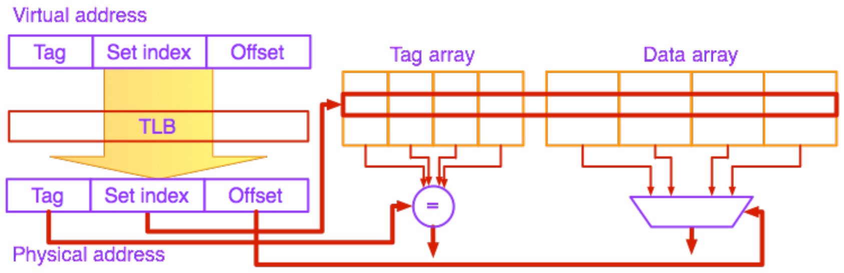
Physically indexed, virtually tagged

- Never used
- No OS intervention for cache management
- TLB is in the critical path



Physically indexed, physically tagged

- No OS intervention for cache management
- TLB is in the critical path


Virtual address
Tag
Set index
Offset
TLB
Tag
Set index
Offset
Physical address
Tag array
Data array
=