

Virtual memory

Hung-Wei Tseng

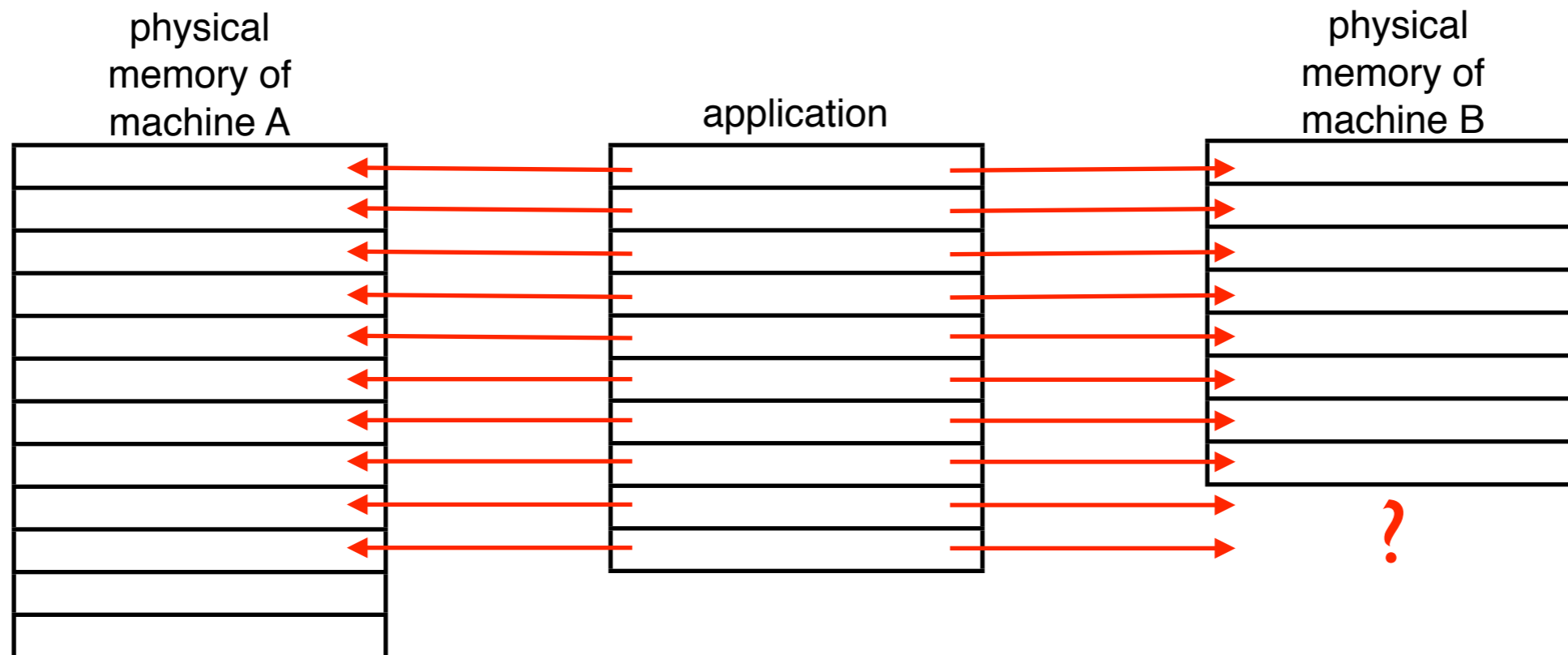
Outline

- Why virtual memory
- How VM works
- VM and cache

Virtual memory

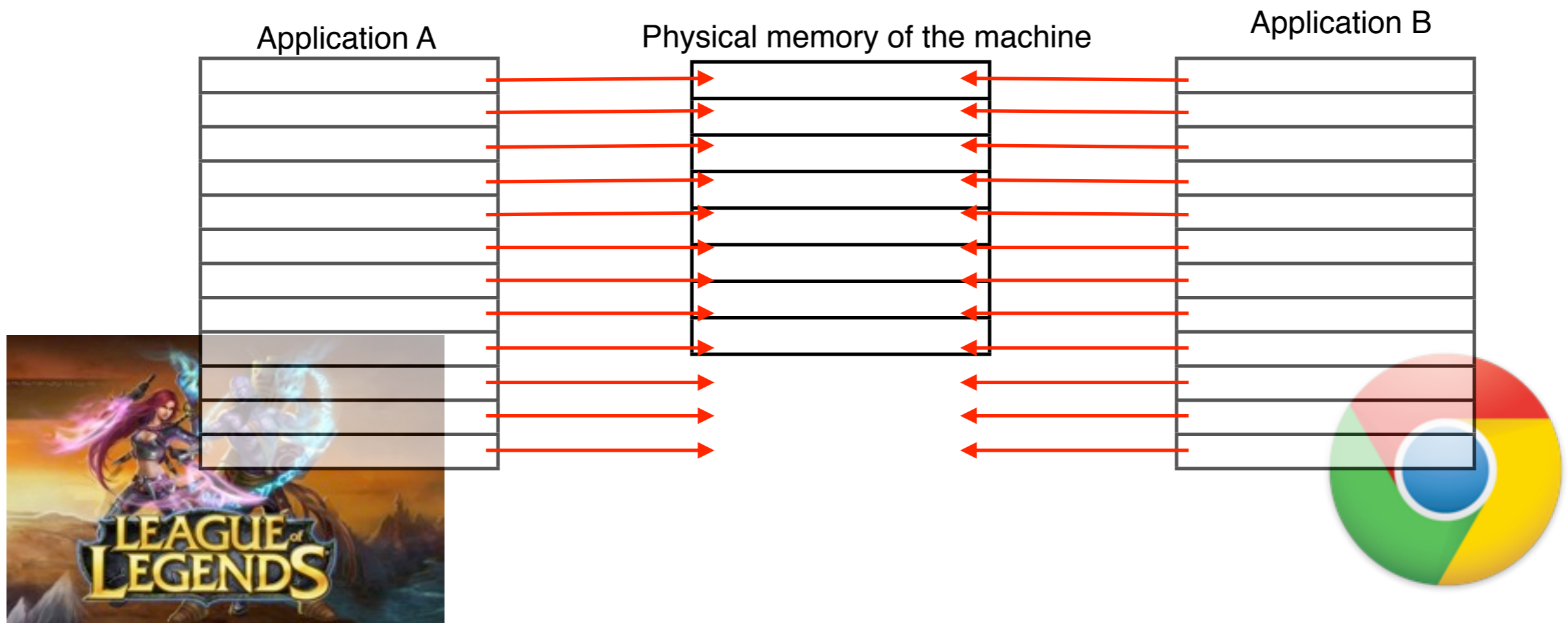
Scenario I

- An application is design on machine A with memory size X. Can we safely execute the same application on another machine B with memory size Y?



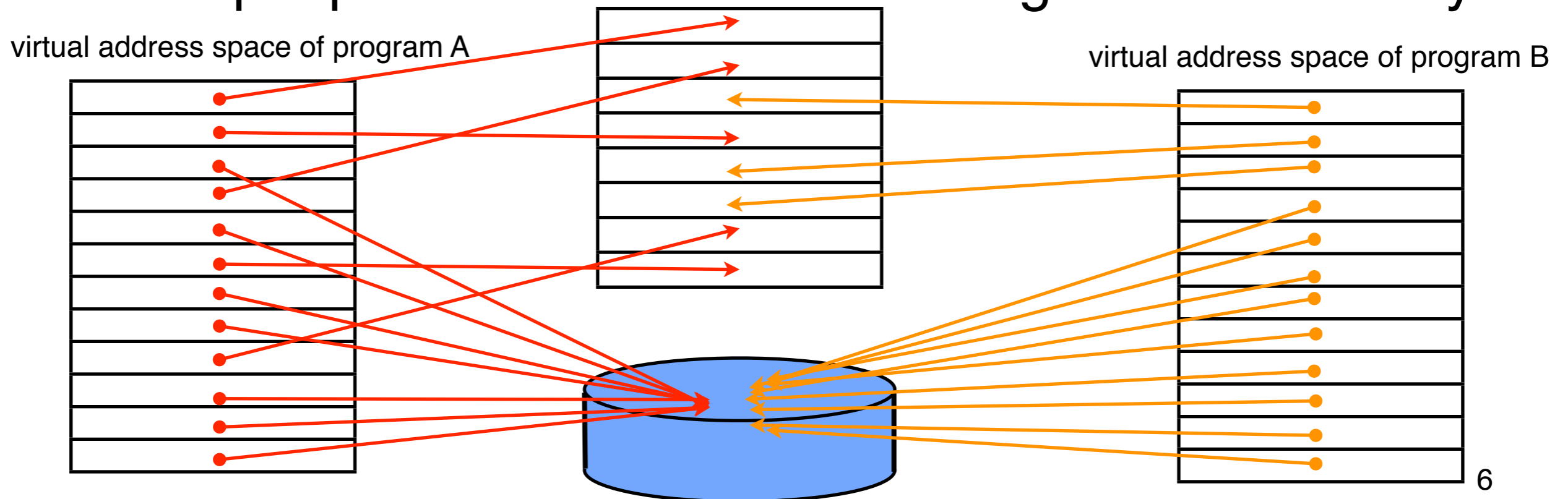
Scenario II

- Both application A and application B would like to use the same machine
 - Each application should not touch data of the other. This is called “protection” in OS



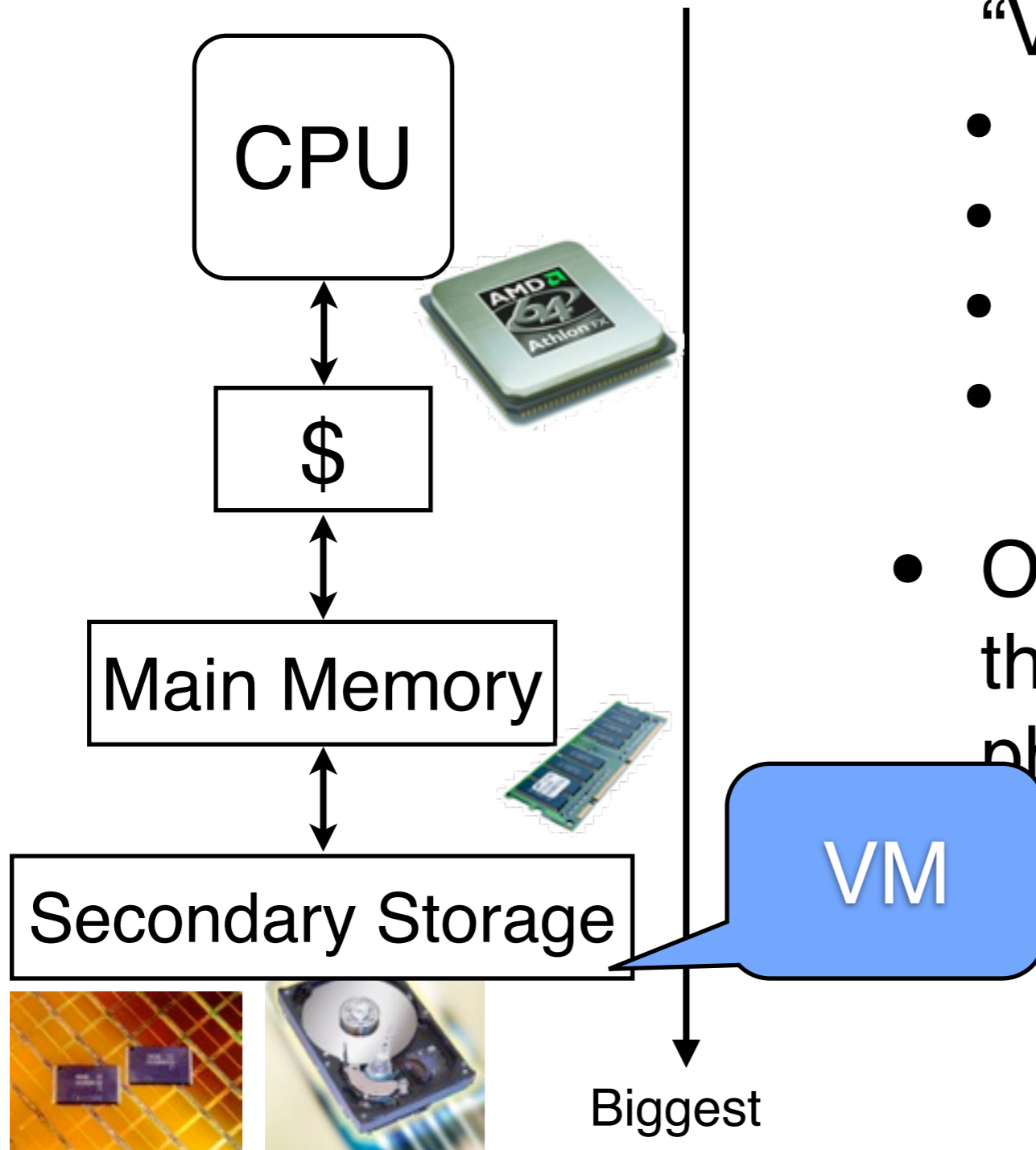
Why Virtual Memory?

- Every process runs on “virtual memory space”
 - Process: a running program in the operating system
- Physical memory “caches” memory “pages” from virtual memory
- Single program can exceed the size of physical memory.
- Multiple processes can share a single main memory.



Memory Hierarchy

Fastest,
Most Expensive



Biggest

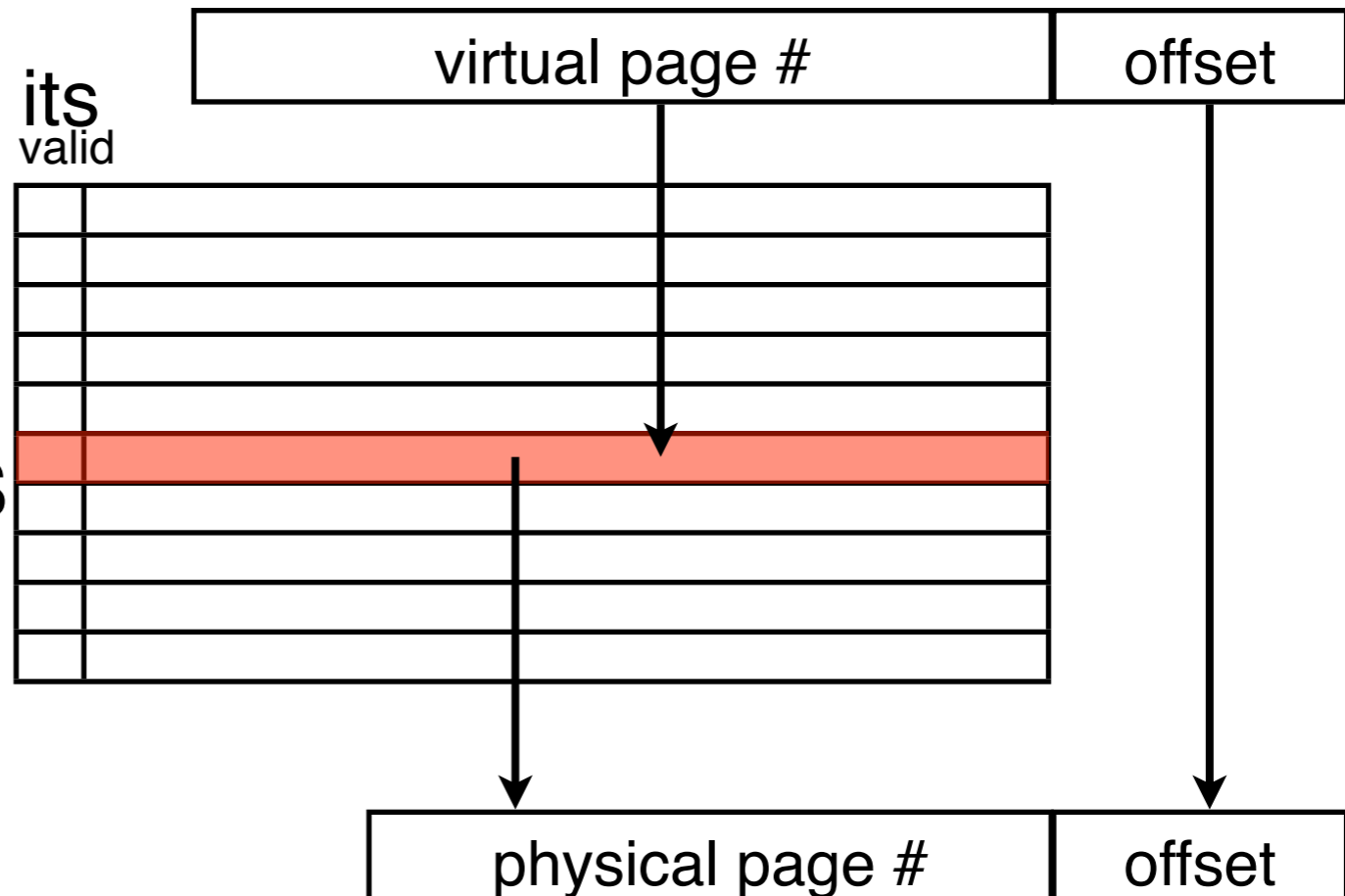
- Main memory is a cache for “Virtual Memory”
 - A: Fully Associative
 - B: page size!
 - S: 1 (Since it’s FA)
 - Replacement policies?
 - LRU, random...
- Operating system manages the mapping between physical and virtual addresses

How VM works

Address translation

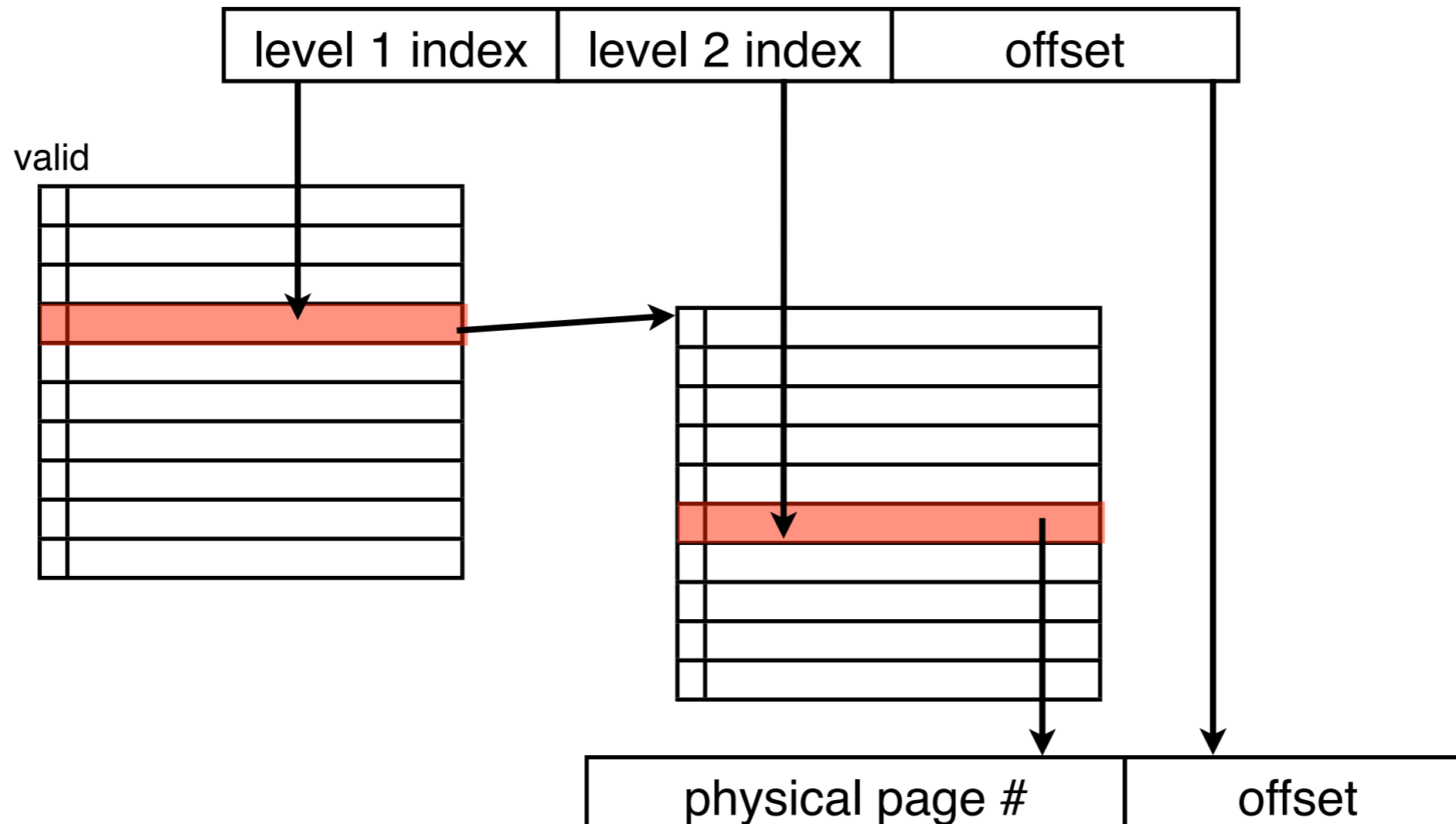
- Processor uses **virtual addresses**, main memory uses physical memory addresses
- Virtual address space is organized into “pages”
- The system references the “**page table**” to translate addresses

- each process has its own page table
- the page table content is maintained by OS

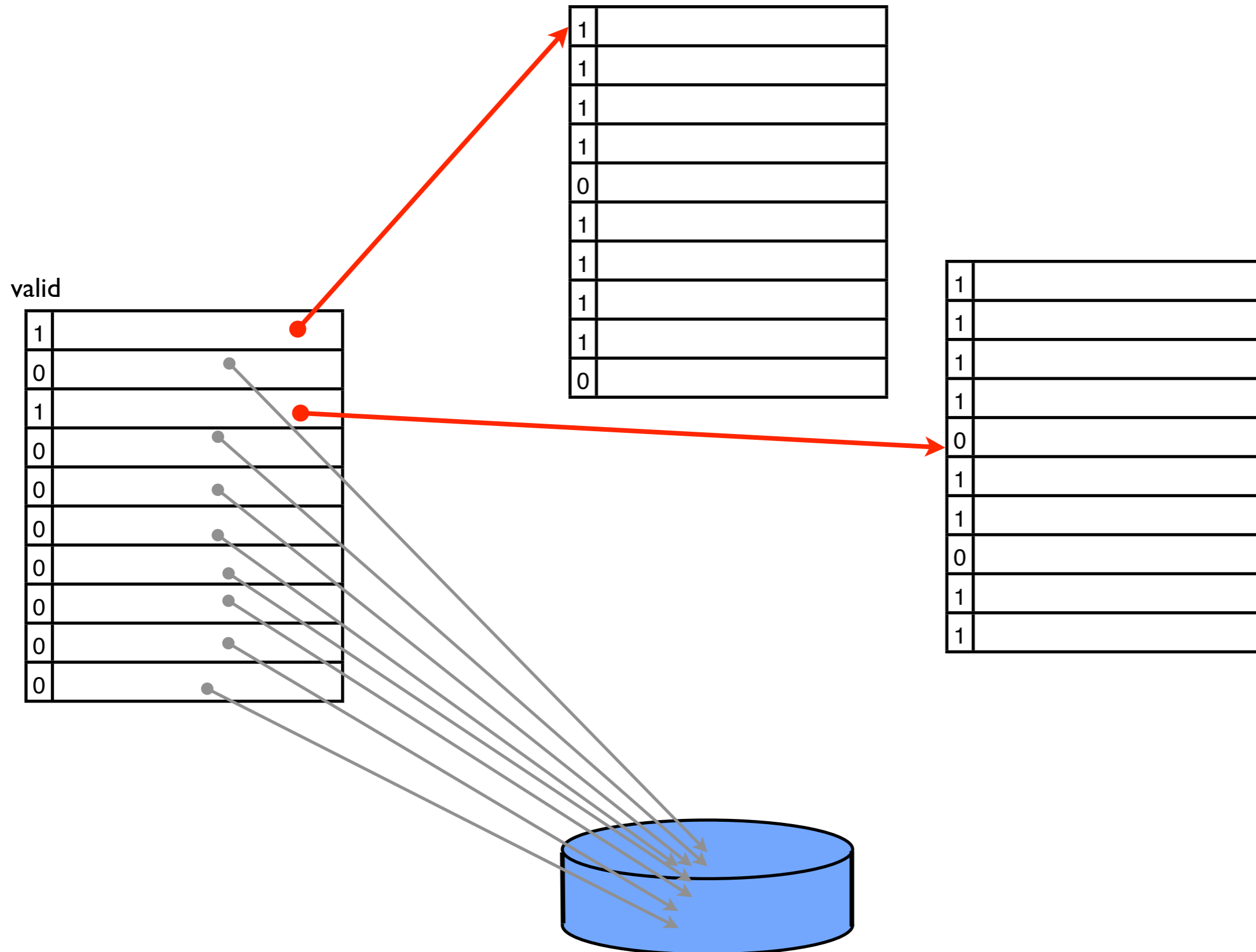


Hierarchical page table

- Break the virtual page number into several pieces
- If one piece has N bits, build an 2^N -ary tree
- Only store the part of the tree that contain valid pages
- Walk down the tree to translate the virtual address



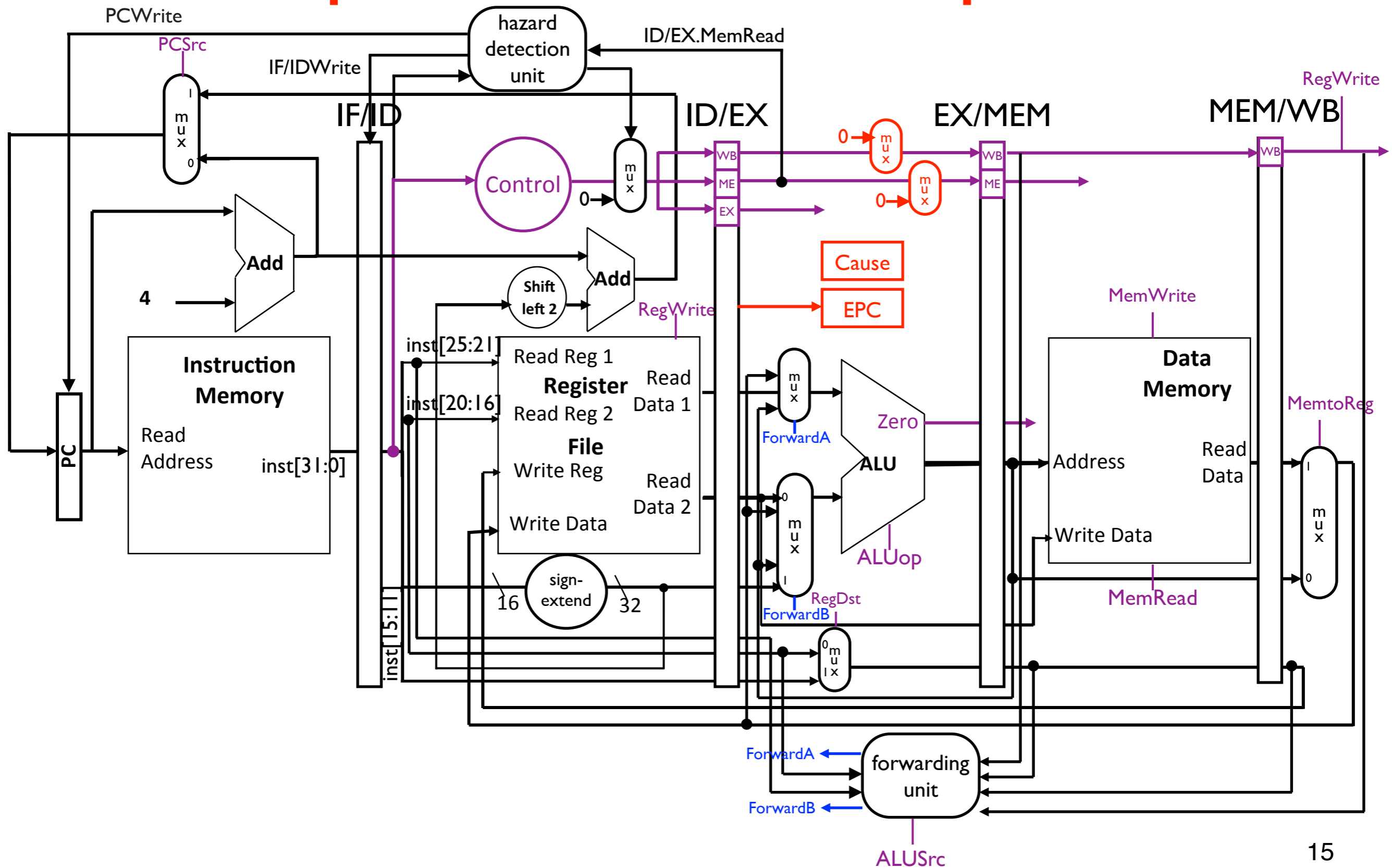
Hierarchical page table



Page fault

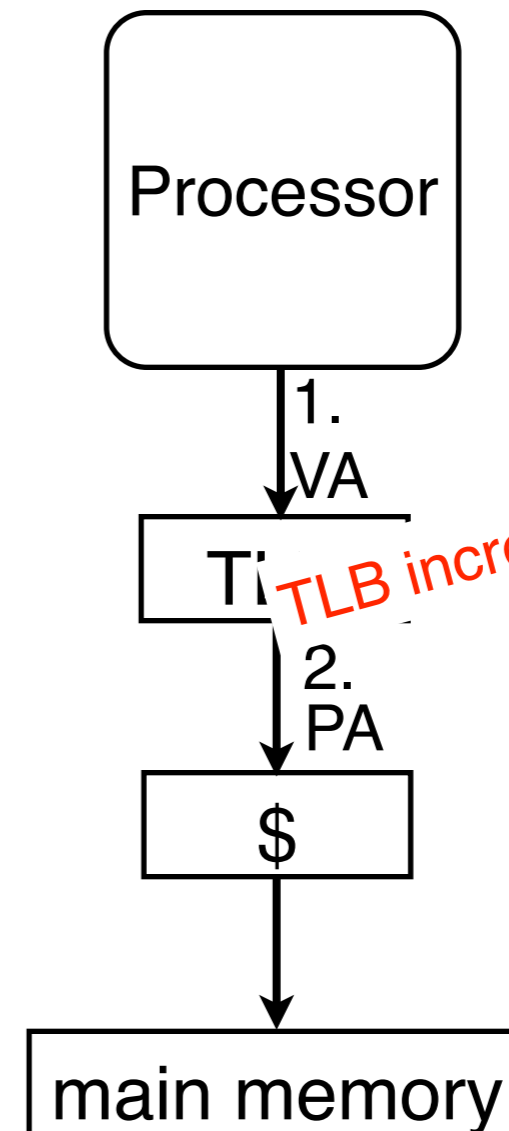
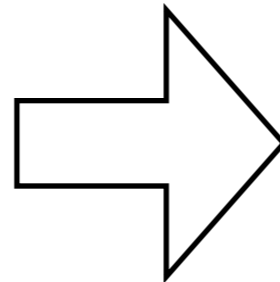
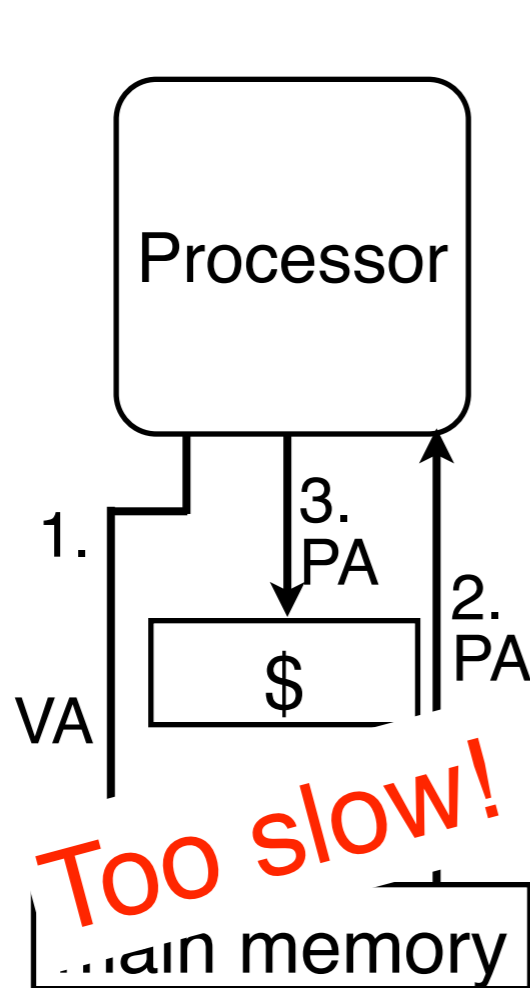
- When the referenced memory page is not mapped to physical memory
- The processor raises an “exception” and transfers the control (change the PC) to operating system code that handles the page fault
 - LRU or other page replacement policies to decide the evicting page
 - Fetch the demanding page from secondary storage (hard drive, SSD)
 - Transfer the control back to the original program
 - Could be slow! The access time of hard drive is more than 6ms

Pipeline with exception



VM & Cache

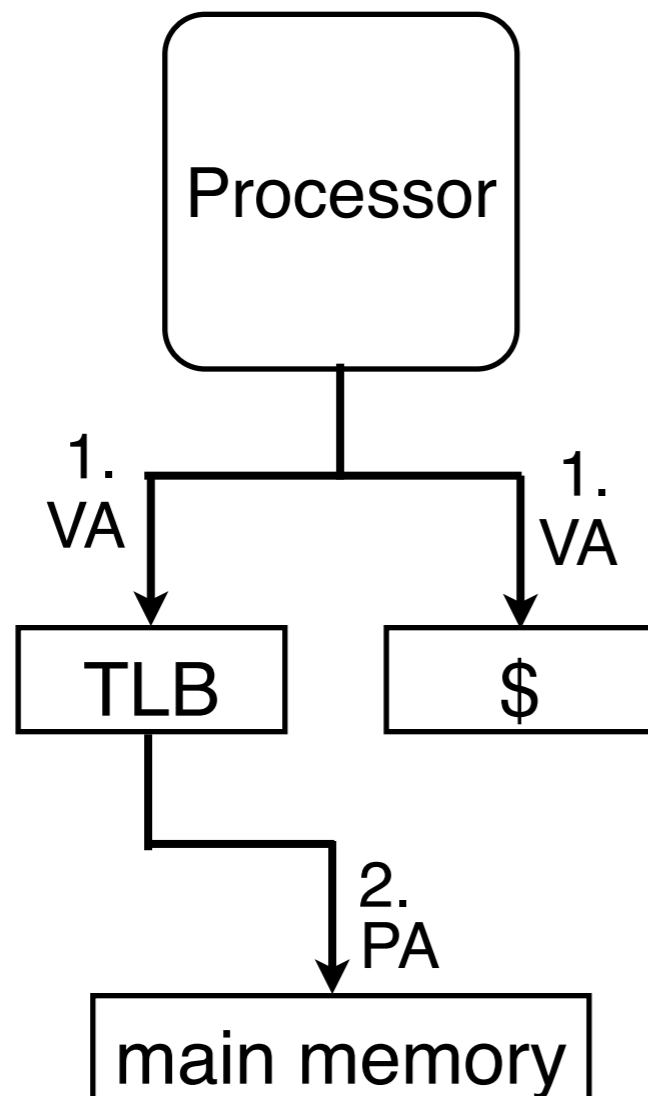
Cache + Virtual Memory



- TLB: Translation Look-aside Buffer
 - a cache of page table
 - small, high-associativity
 - miss penalty: access to page table in main memory

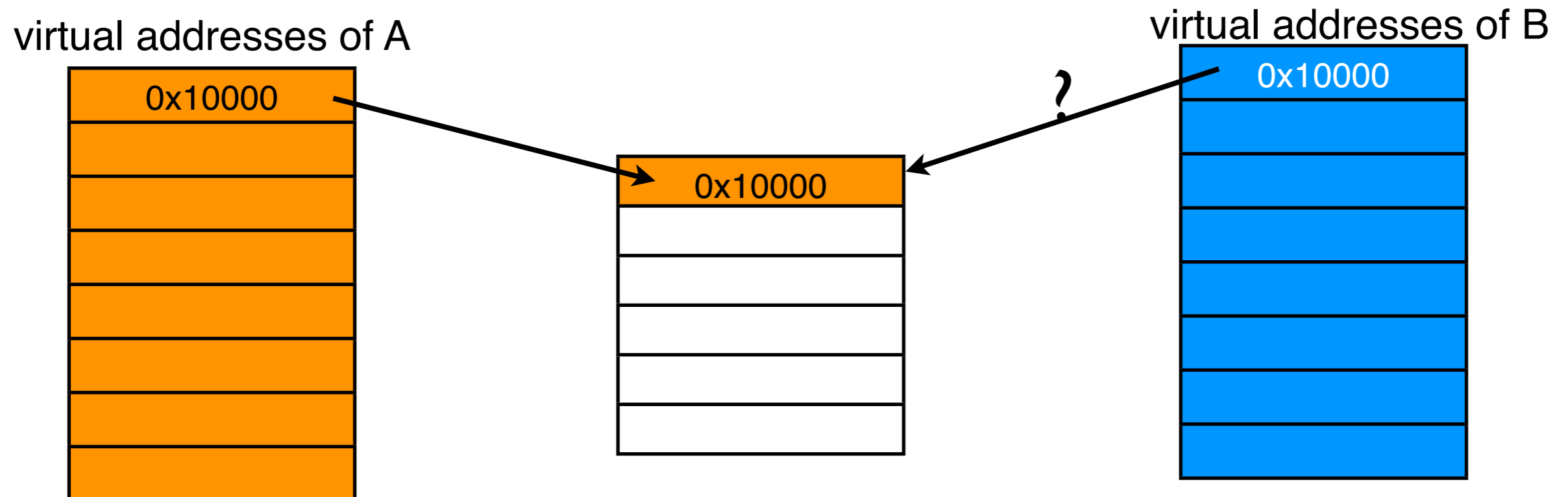
Cache+Virtual Memory

- Virtual Cache
 - The cache also uses virtual addresses
 - Address translation is required only when miss.



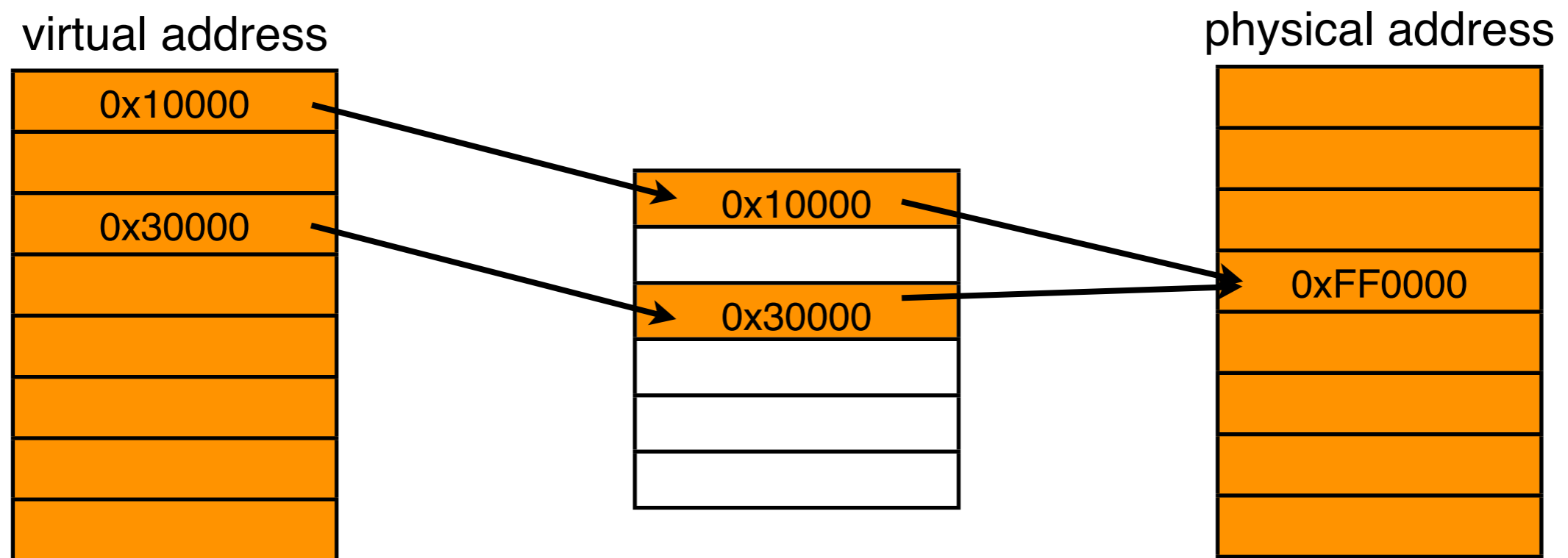
Problems of Virtual Cache

- Multiple processes accessing to the same virtual address
 - Process A accessed 0x10000. Process B also want to access 0x10000
 - Flush the cache when context switch
 - Attach PID to cache



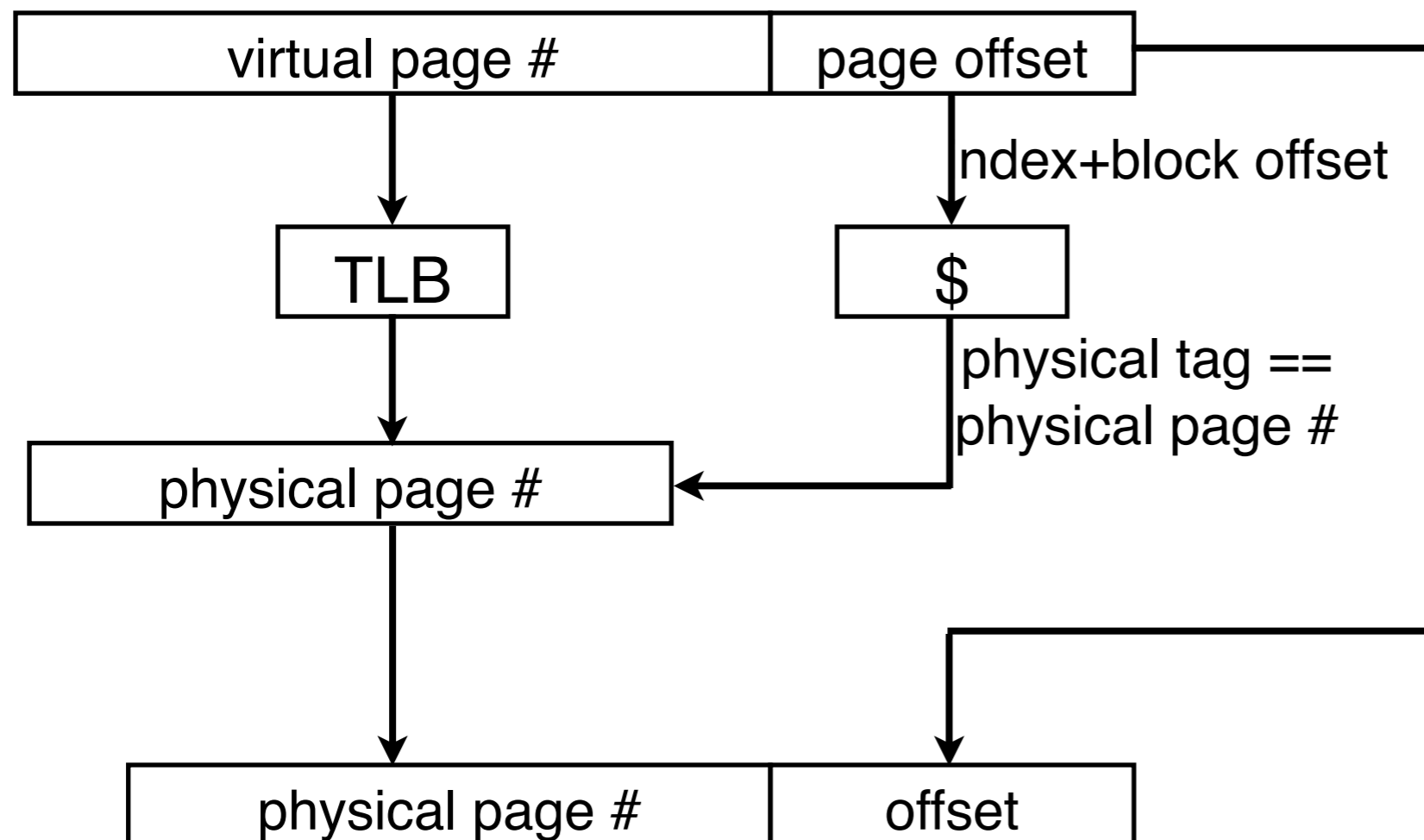
Problems of Virtual Cache

- Alias: A physical address maps to different virtual addresses
 - Two copies of data in cache due to copy on write. One may get the wrong data if the other is modified.



Virtual indexed, physical tagged cache

- Force aliasing virtual addresses mapped to the same cache location.
- Cache stores tag fields of “physical addresses”
 - the physical tag is also the physical page number!



$$C = ABS$$

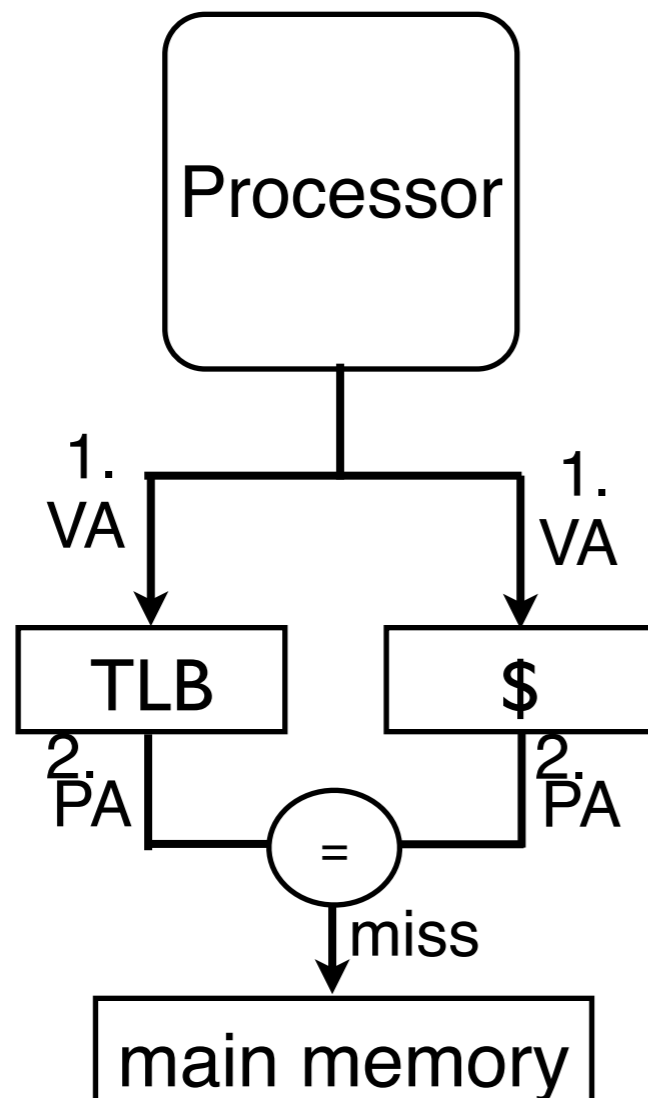
$$\lg(S) + \lg(B) = 12$$

if $A = 1$ (DM cache)

$$C = 1 * (2^{12}) = 4KB$$

Virtual indexed, physical tagged cache

- Force aliasing virtual addresses mapped to the same cache location.
- Cache stores tag fields of “physical addresses”



Cache & Performance

- The 5-stage MIPS processor runs @ 2GHz. 20% are L/S
 - L1 I-cache miss rate: 5%, hit time: 1 cycle
 - L1 D-cache miss rate: 10%, hit time: 1 cycle, 10% evicted blocks are dirty
 - L2 U-Cache miss rate: 20%, hit time: 10 cycles, 20% evicted blocks are dirty
 - L1 TLB miss rate: 1%, hit time < 1 cycle
 - 200 cycles penalty
 - Main memory hit time: 100 cycles
 - All caches are write-back, write-allocate

$CPI_{\text{average}} = 1 +$

$$20\% * (1\% * 200 + 10\% * (1 + 10\%) * (10 + 20\% * (1 + 20\%) * (100))) \\ + 1\% * 200 + (5\% * (10 + 20\% * (1 + 20\%) * (100))) = 5.85$$

Q & A