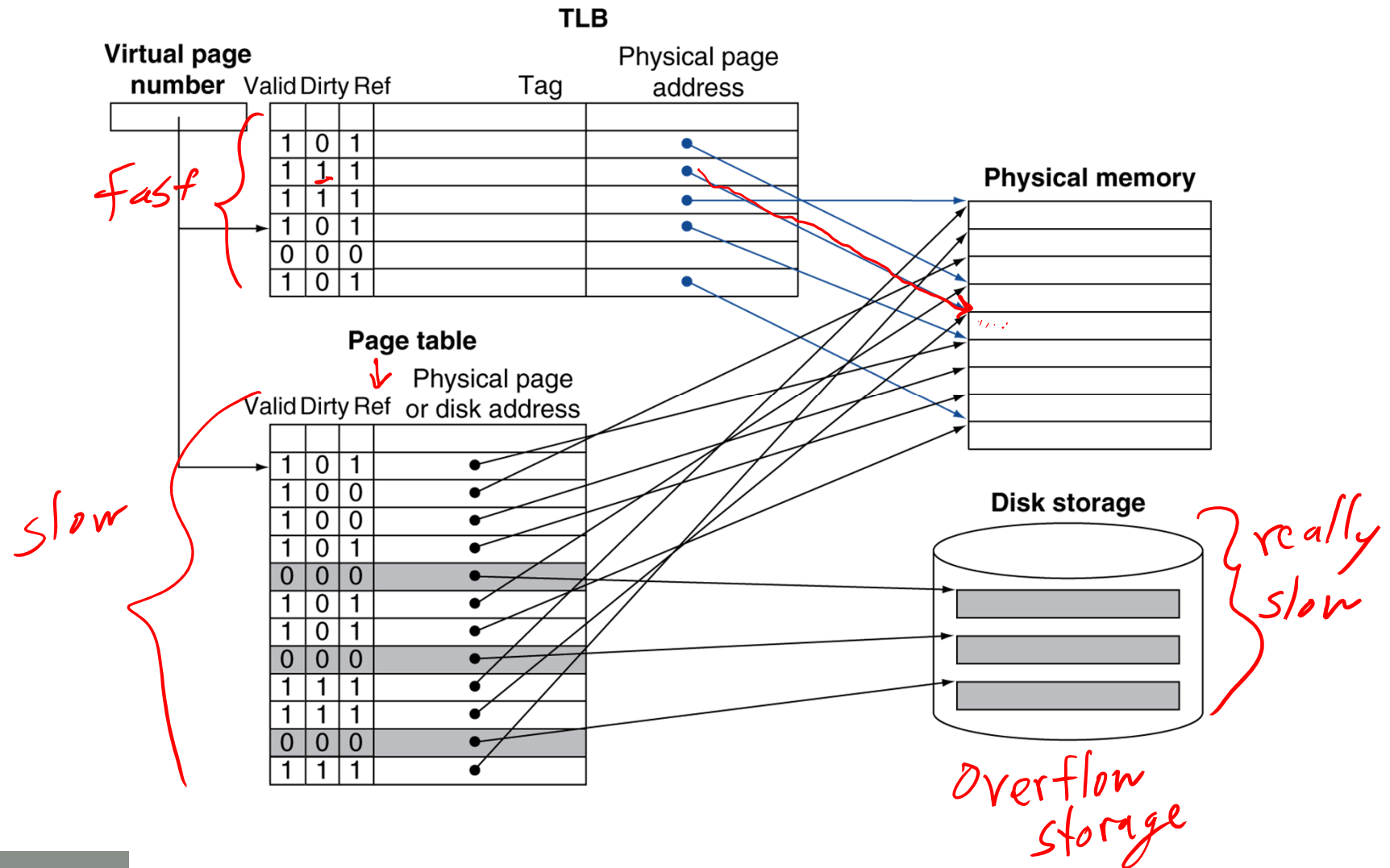# Fast Translation Using a TLB

TLB Misses

- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
 - Then restart the faulting instruction

TLB Miss Handler

- TLB miss indicates
    - Page present, but PTE not in TLB
    - Page not preset
- Must recognize TLB miss before destination register overwritten
    - Raise exception
- Handler copies PTE from memory to TLB
    - Then restarts instruction
    - If page not present, page fault will occur

Page Fault Handler

- Use faulting virtual address to find PTE

- Locate page on disk

- Choose page to replace
  - If dirty, write to disk first

- Read page into memory and update page table

- Make process runnable again
  - Restart from faulting instruction

TLB and Cache Interaction
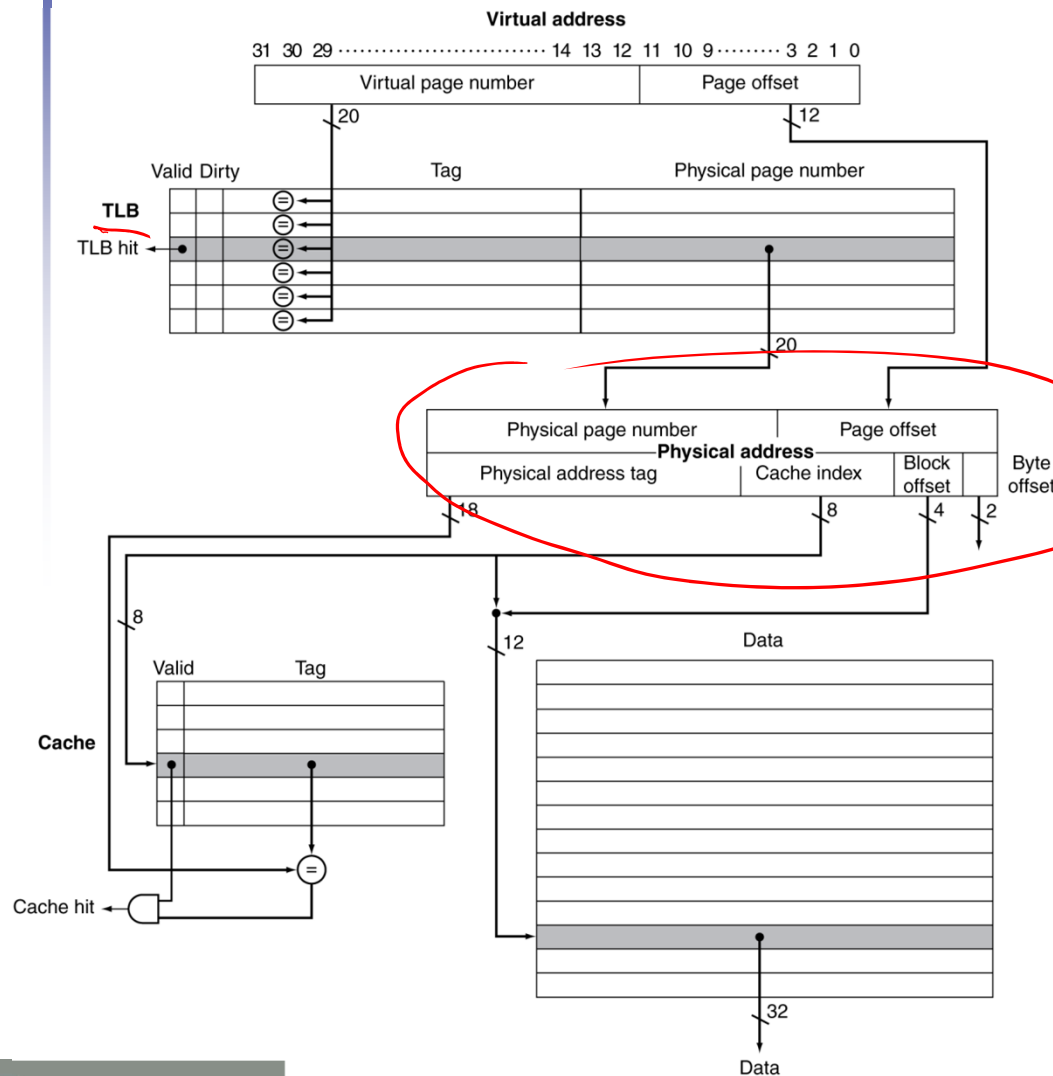
- If cache tag uses physical address
  - Need to translate before cache lookup
- Alternative: use virtual address tag
  - Complications due to aliasing
    - Different virtual addresses for shared physical address

Memory Protection

- Different tasks can share parts of their virtual address spaces
  - But need to protect against errant access
  - Requires OS assistance
- Hardware support for OS protection
  - Privileged supervisor mode (aka kernel mode)
  - Privileged instructions
  - Page tables and other state information only accessible in supervisor mode
  - System call exception (e.g., syscall in MIPS)

The Memory Hierarchy

The BIG Picture

- ## Common principles apply at all levels of the memory hierarchy
    - Based on notions of caching
- ## At each level in the hierarchy
    - Block placement
    - Finding a block
    - Replacement on a miss
    - Write policy

Block Placement

- Determined by associativity
  - Direct mapped (1-way associative)
    - One choice for placement
  - n-way set associative
    - n choices within a set
  - Fully associative
    - Any location

- Higher associativity reduces miss rate
  - Increases complexity, cost, and access time

Finding a Block

Associativity	Location method	Tag comparisons
Direct mapped	Index	1
n-way set associative	Set index, then search entries within the set	n
Fully associative	Search all entries	#entries
	Full lookup table	0

- ## Hardware caches
  - Reduce comparisons to reduce cost
- ## Virtual memory
  - Full table lookup makes full associativity feasible
  - Benefit in reduced miss rate

Replacement

- Choice of entry to replace on a miss
  - Least recently used (LRU)
    - Complex and costly hardware for high associativity
  - Random
    - Close to LRU, easier to implement
- Virtual memory
  - LRU approximation with hardware support

Write Policy

- ## Write-through
    - Update both upper and lower levels
    - Simplifies replacement, but may require write buffer
- ## Write-back
    - Update upper level only
    - Update lower level when block is replaced
    - Need to keep more state
- ## Virtual memory
    - Only write-back is feasible, given disk write latency