Address Translation

Chapter 8 OSPP

Part I: **Advanced**

Sparse Address Spaces

- What if virtual address space is large?
  - 32-bits, 4KB pages => 500K page table entries
  - 64-bits => 4 quadrillion page table entries

  - Famous quote:
  - “Any programming problem can be solved by adding a level of indirection”

- Today’s OS allocate page tables on the fly, even on the backing store!
  - Allocate/fill only page table entries that are in use
  - STILL, can be really big

Multi-level Translation

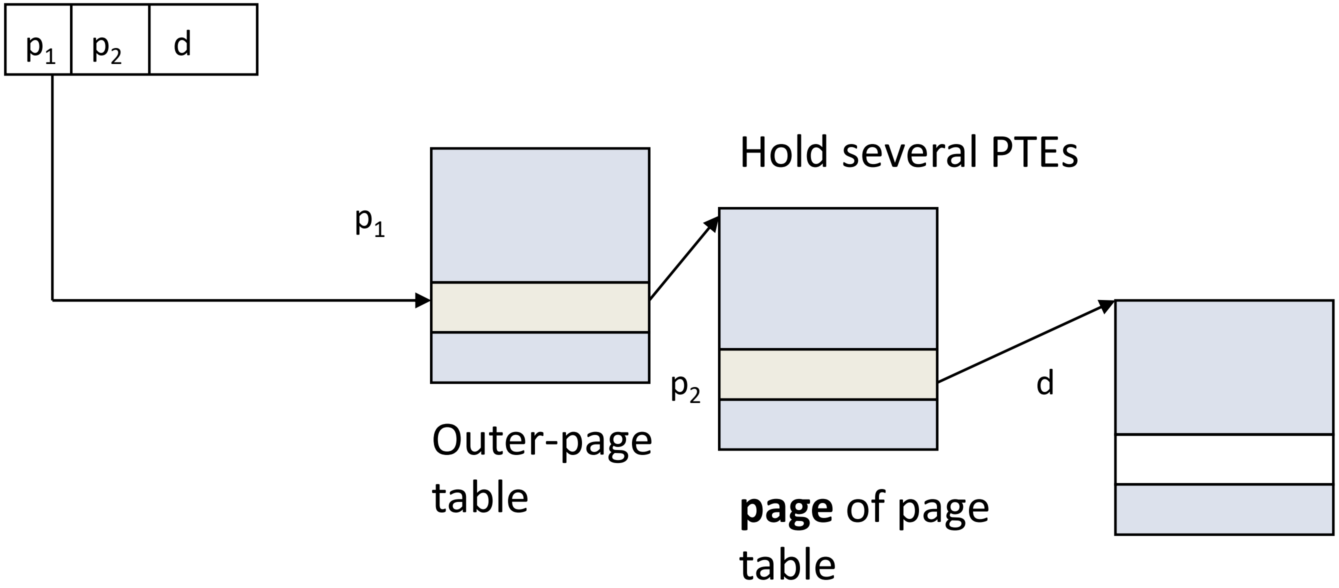
- Tree of translation tables
  - Multi-level page tables
  - Paged segmentation
  - Multi-level paged segmentation

- Stress: hardware is doing the translation!

- Page the page table or the segments! … or both

Address-Translation Scheme

- Address-translation scheme for a two-level 32-bit paging architecture



<board>

Two-Level Paging Example

- A VA on a 32-bit machine with 4K page size is divided into:
  - a page number consisting of 20 bits
  - a page offset consisting of 12 bits (set by hardware/OS)
  - assume trivial PTE of 4 bytes (just frame #)

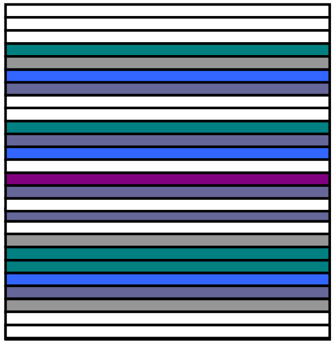
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page number
  - a 10-bit page offset (to each PTE)
- Thus, a VA is as follows:

page number		page offset
p_i	p_2	d
10	10	12

- where p_i is an index into the outer page table, and p_2 is the displacement within the page of the outer page table (i.e the PTE entry).

Multi-level Page Tables

- How big should the outer-page table be?
  - Size of the page table for process (PTE is 4): 2^{20}x4=2^{22}
  - Page this (divide by page size): $2^{22}/2^{12} = 2^{10}$
  - Answer: 2^{10} x4=2^{12}
- How big is the virtual address space now?
  - Still 2^{32}
- Have we reduced the amount of memory required for paging?
  - Nope – just reduced the amount of contiguous memory required: outer page table is smaller by factor of page size



Address Translation

Chapter 8 OSPP

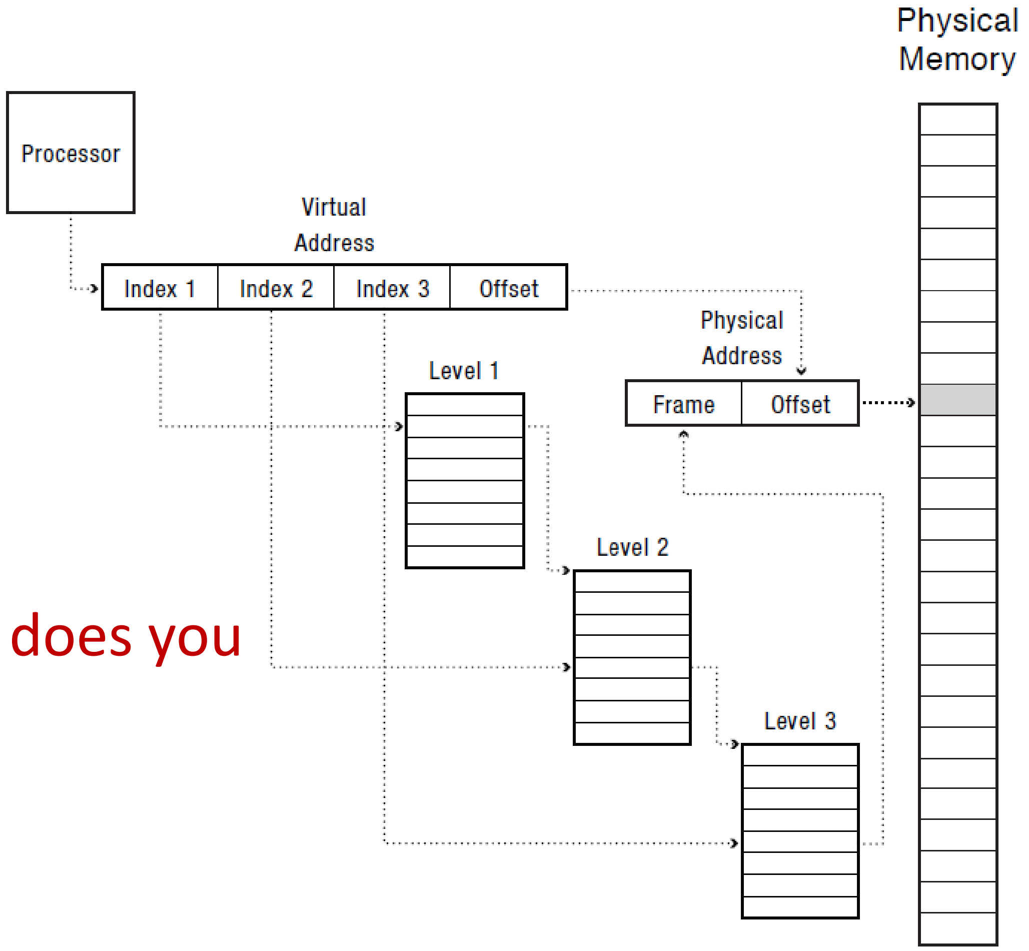
Part I: **Advanced**

Today

- Multi-level translation
- TLB + caching
- Memory hogs paper: much reduced

Multilevel Paging

- Can keep paging!



What CS concept does you remind you of?

Multilevel Paging and Performance

- Can take 3 memory accesses (if TLB miss)
- Suppose TLB access time is 20 ns, 100 ns to memory
- Cache hit rate of 98 percent yields:

  effective access time = 0.98 x 120 + 0.02 x 320 = 124 nanoseconds
  24% slowdown

- Can add more page tables and can show that slowdown grows slowly:

  3-level: 26 %

  4-level: 28%

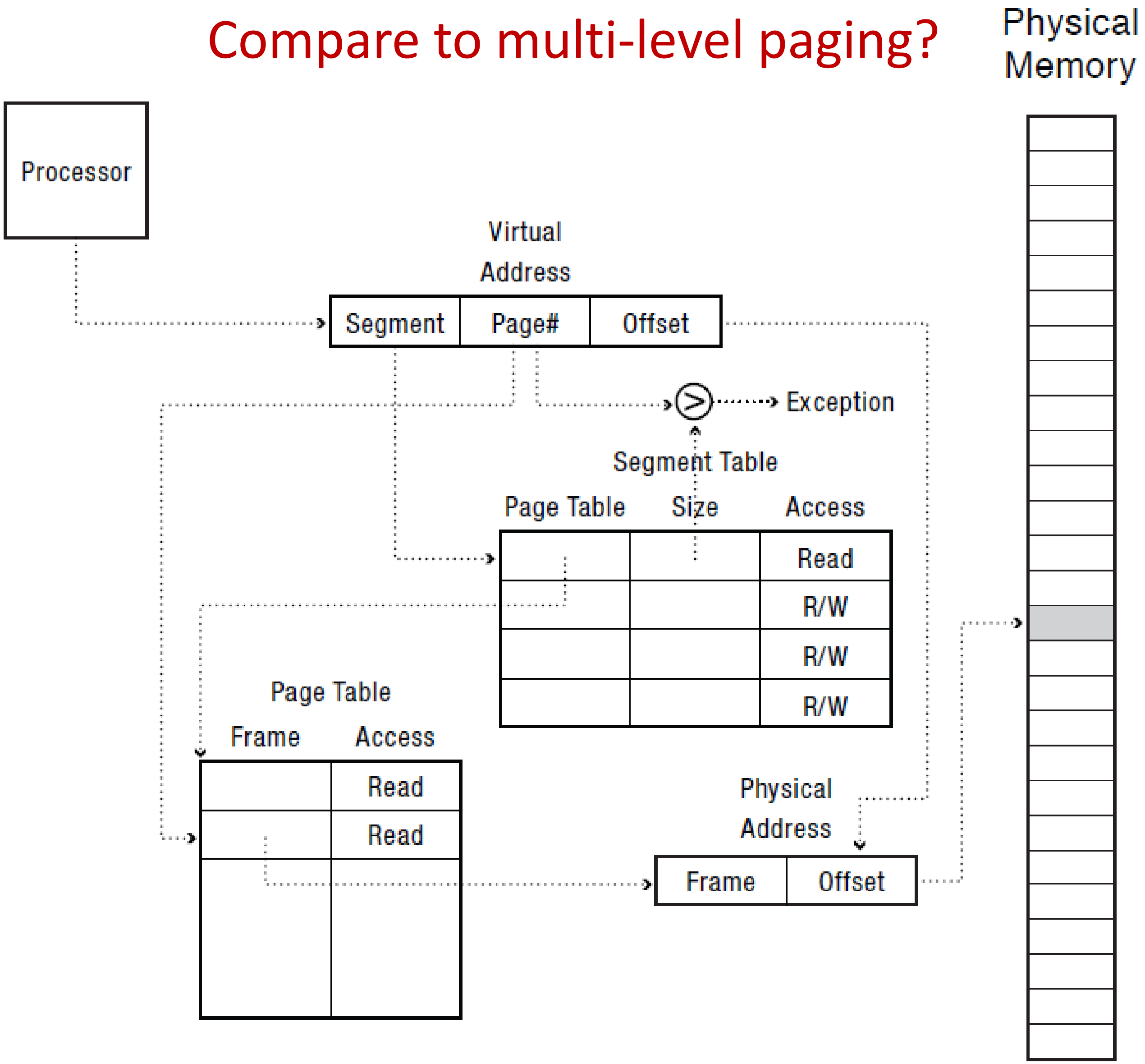
- Q: why would I want to do this!

Paged Segmentation

- Process memory is segmented
- Segment table entry:
  - Pointer to page table
  - Page table length (# of pages in segment)
  - Access permissions
- Page table entry:
  - Page frame
  - Access permissions
- Share/protection at either page or segment-level
- <board>

Paged Segmentation (Implementation)

Compare to multi-level paging?



Multilevel Translation

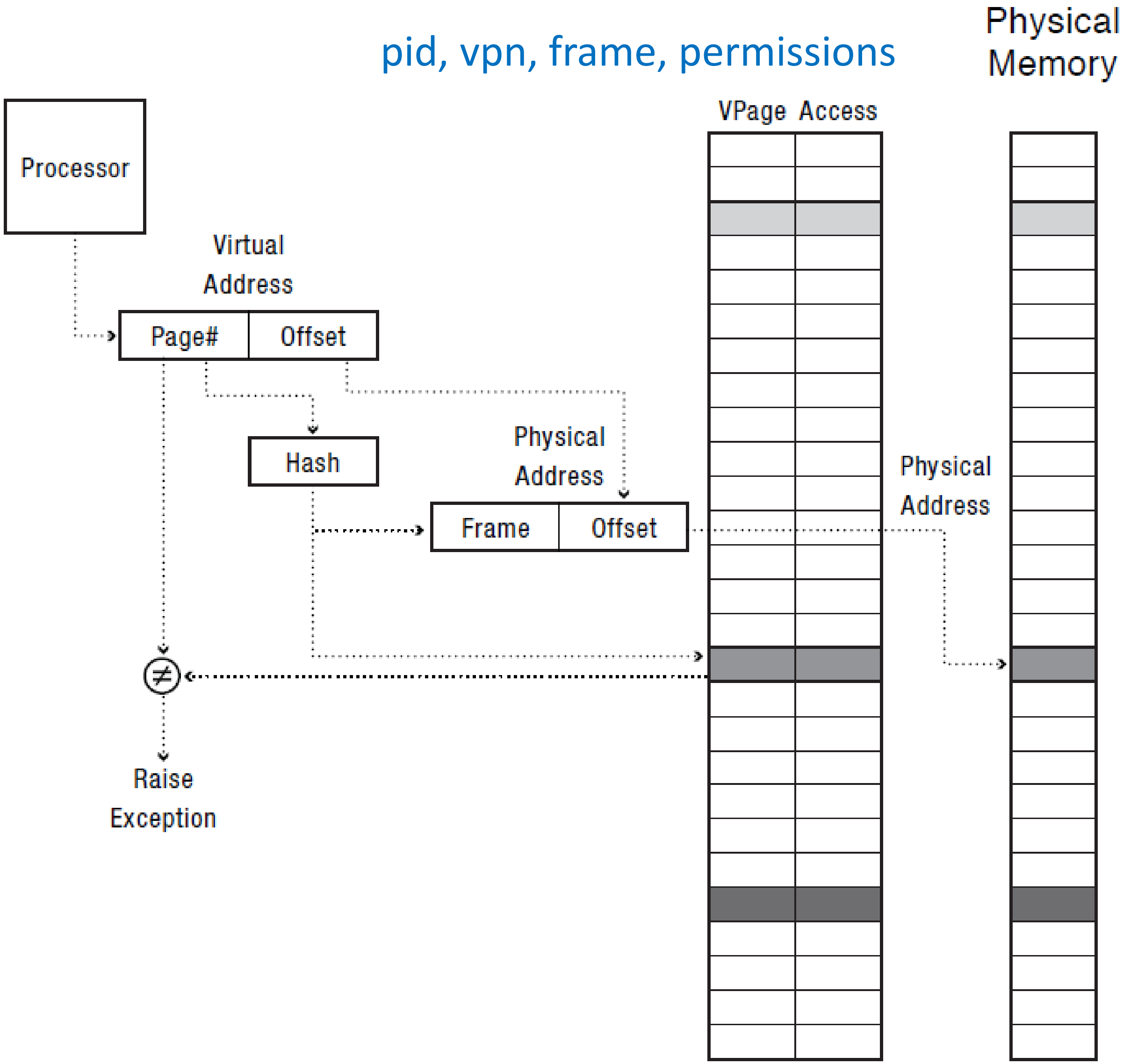
- Pros:
  - Simple and flexible memory allocation (i.e. pages)
  - Share at segment or page level
  - Reduced fragmentation
- Cons:
  - Space overhead: extra pointers
  - Two (or more) lookups per memory reference, but TLB

Portability

- Many operating systems keep their own memory translation data structures for portability, e.g.
    - List of memory objects (segments), e.g. fill-from location
    - Virtual page -> physical page frame (shadow page table)
        - Different from h/w: extra bits (C-on-Write, Z-on-Ref, clock bits)
    - Physical page frame -> set of virtual pages
        - Why?
- Inverted page table : replace all page tables; solve
    - Hash from virtual page -> physical page
    - Space proportional to # of physical ~~pages~~ frames – sort of
    - <board>

Inverted Page Table

pid, vpn, frame, permissions



Address Translation

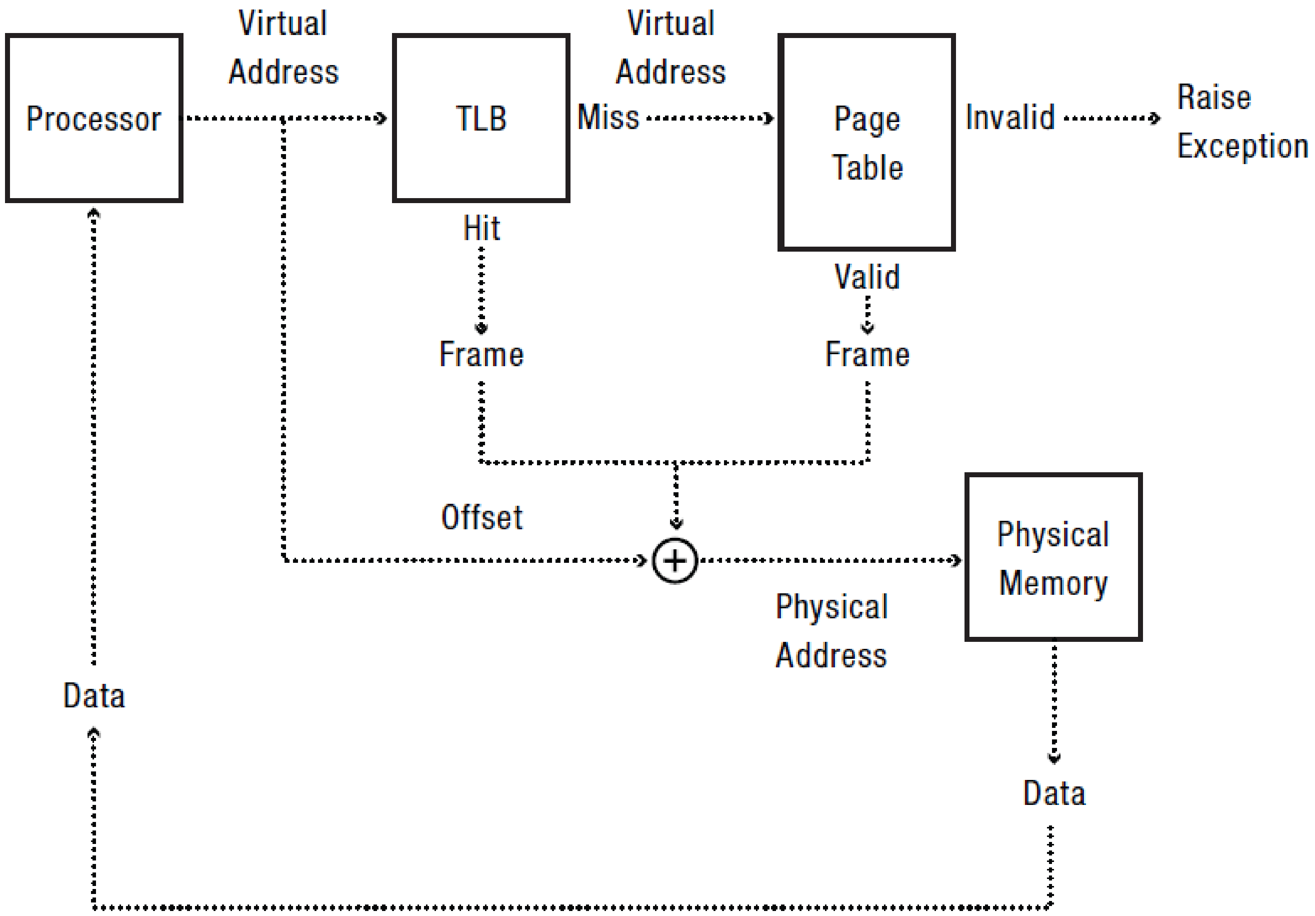
Chapter 8 OSPP

Part I: **Advanced**

Back to TLBs

Pr(TLB hit) * cost of TLB lookup +
  Pr(TLB miss) * cost of page table lookup

TLB and Page Table Translation



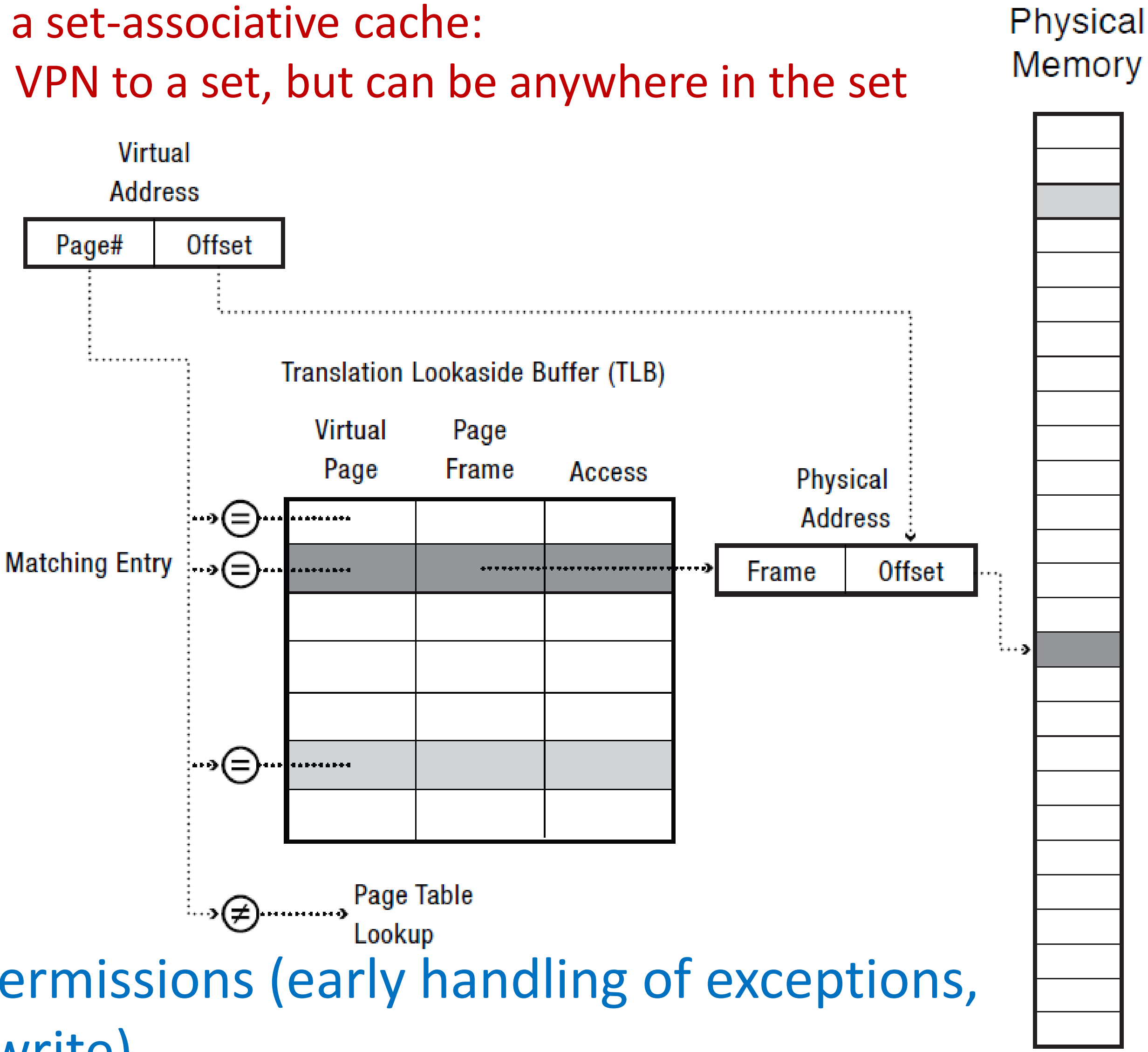
TLB Miss

- Done all in hardware

- Or in software (software-loaded TLB)
  - Since TLB miss is rare …
  - Trap to the OS on TLB miss
  - Let OS do the lookup and insert into the TLB
  - A little slower … but simpler hardware

TLB Lookup

TLB usually a set-associative cache:
Direct hash VPN to a set, but can be anywhere in the set

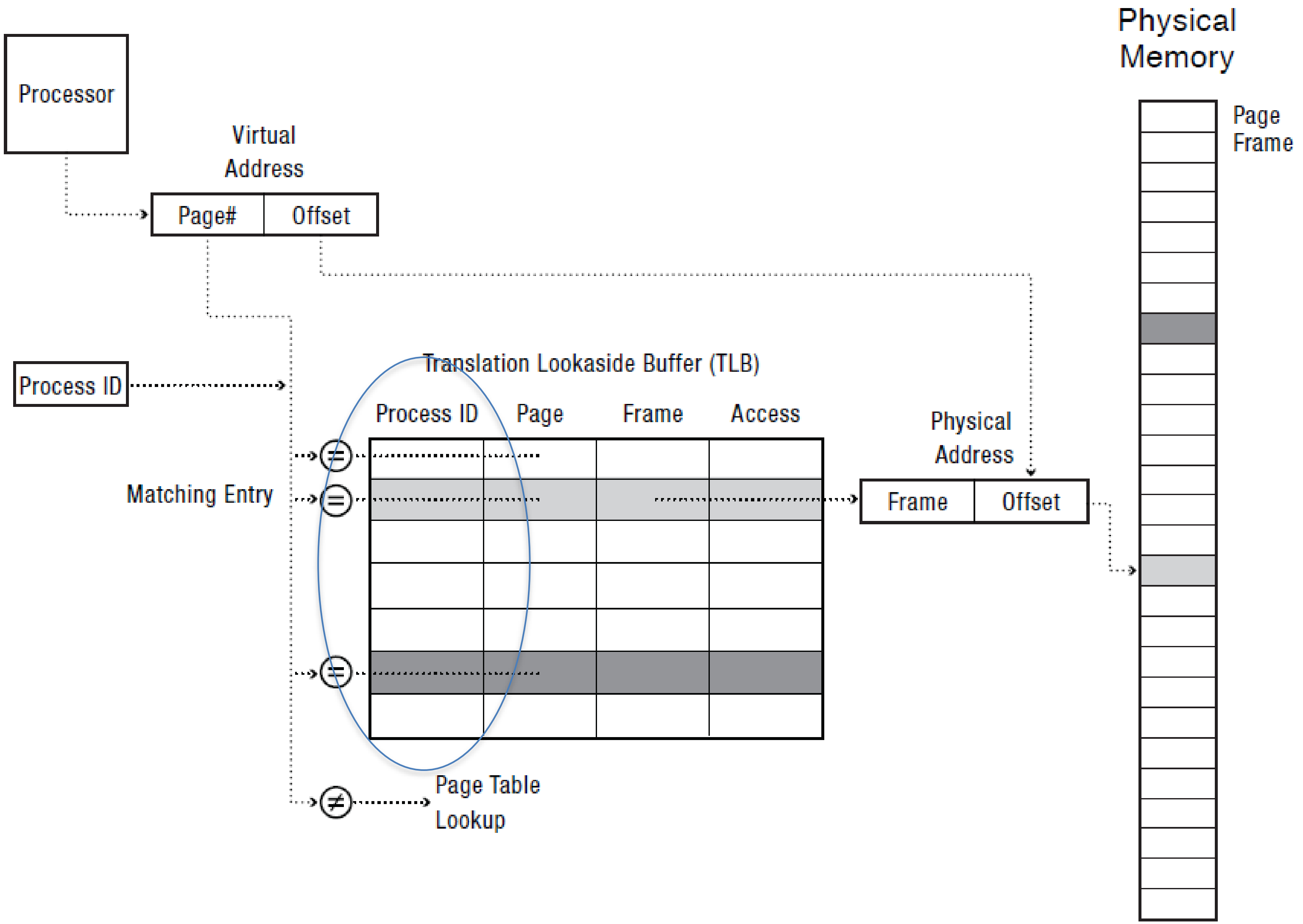


Access: permissions (early handling of exceptions, copy-on-write)

TLB is critical

- What happens on a context switch?
  - Discard TLB? Pros?
  - Reuse TLB? Pros?

- Reuse Solution: Tagged TLB
  - Each TLB entry has process ID
  - TLB hit only if process ID matches current process

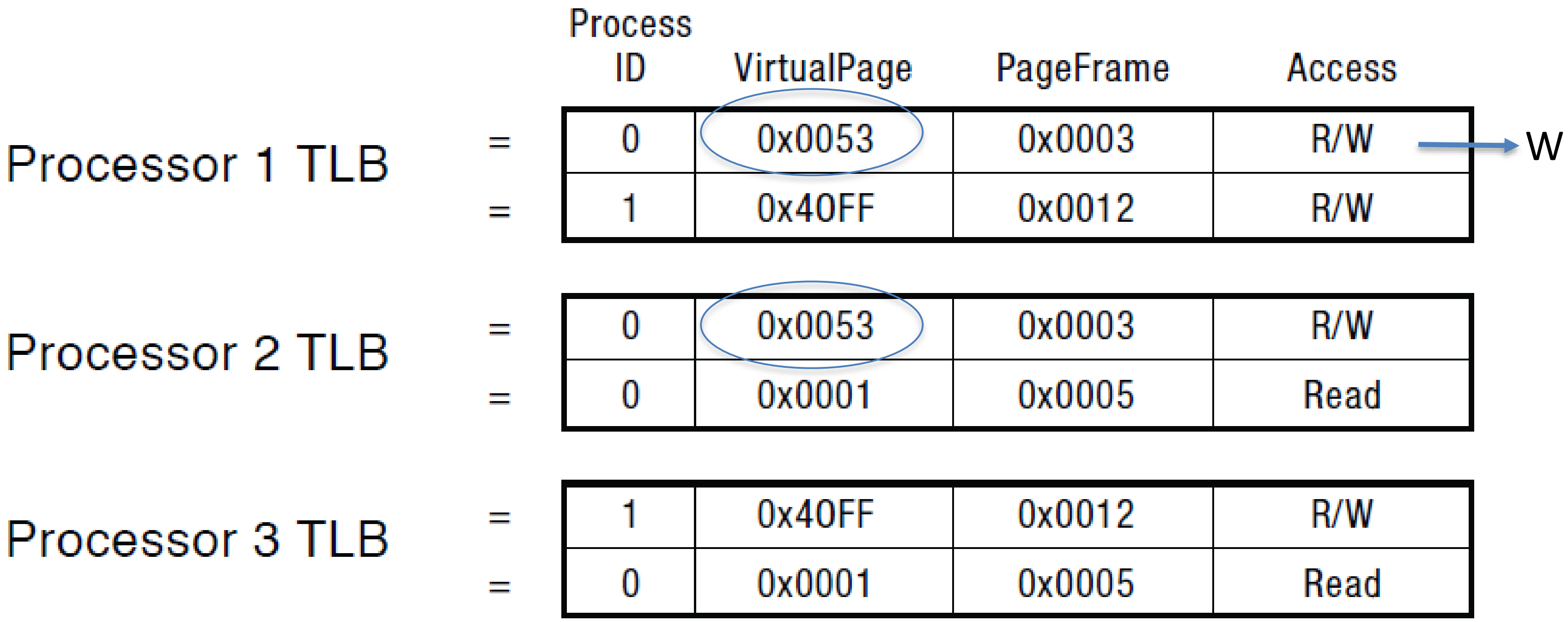


Avoid flushing the TLB on a context-switch

TLB consistency

- What happens when the OS changes the permissions on a page?
  - For demand paging, copy on write, zero on reference, … or is marked invalid!
- TLB may contain old translation or permissions
  - OS must ask hardware to purge TLB entry
- On a multicore: TLB shootdown
  - OS must ask each CPU to purge TLB entry
  - Similar to above

TLB Shootdown



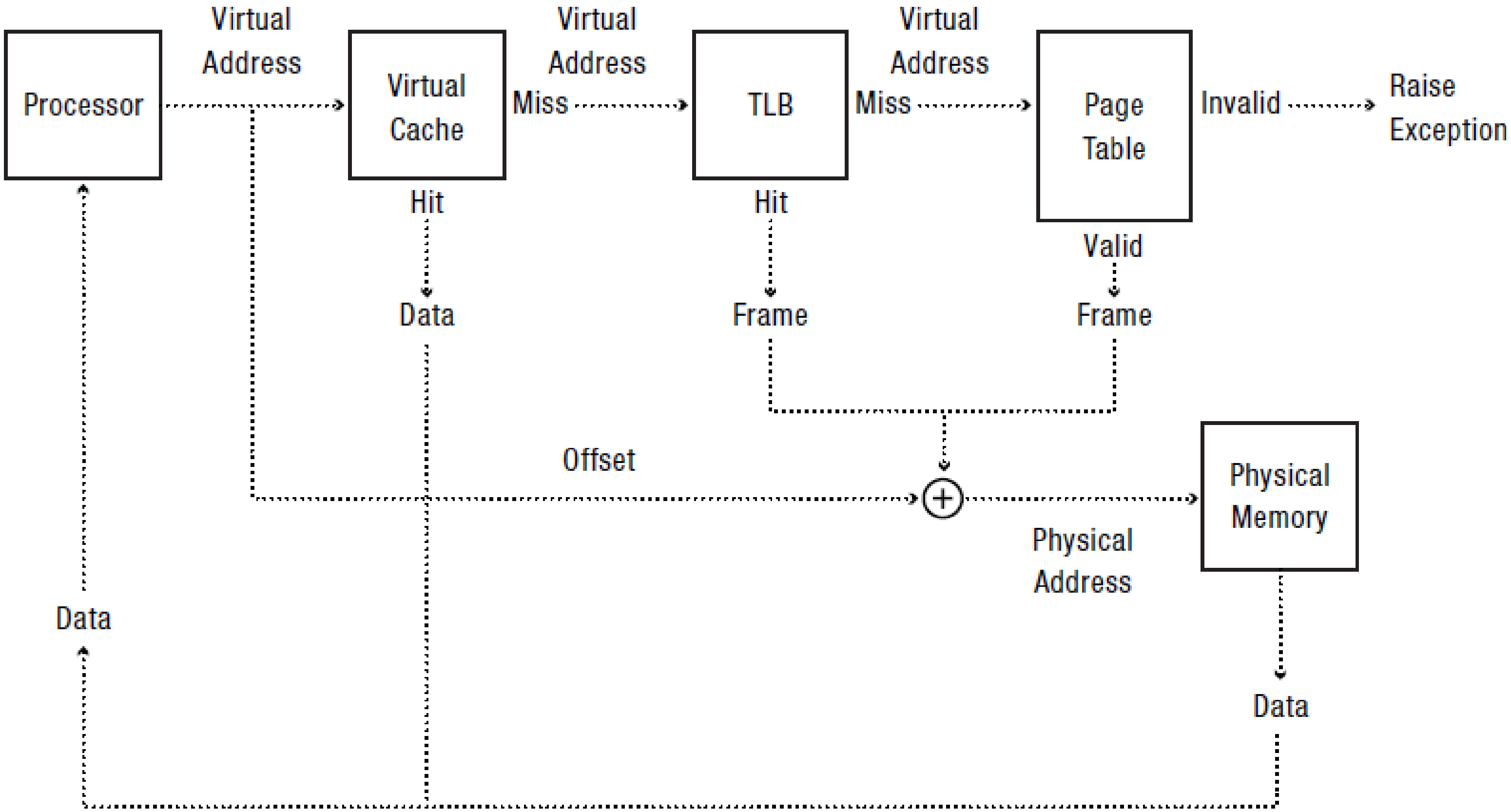
		Process ID	VirtualPage	PageFrame	Access
Processor 1 TLB	=	0	0x0053	0x0003	R/W
	=	1	0x40FF	0x0012	R/W
Processor 2 TLB	=	0	0x0053	0x0003	R/W
	=	0	0x0001	0x0005	Read
Processor 3 TLB	=	1	0x40FF	0x0012	R/W
	=	0	0x0001	0x0005	Read

TLB Optimizations

Virtually Addressed vs. Physically Addressed Data Caches

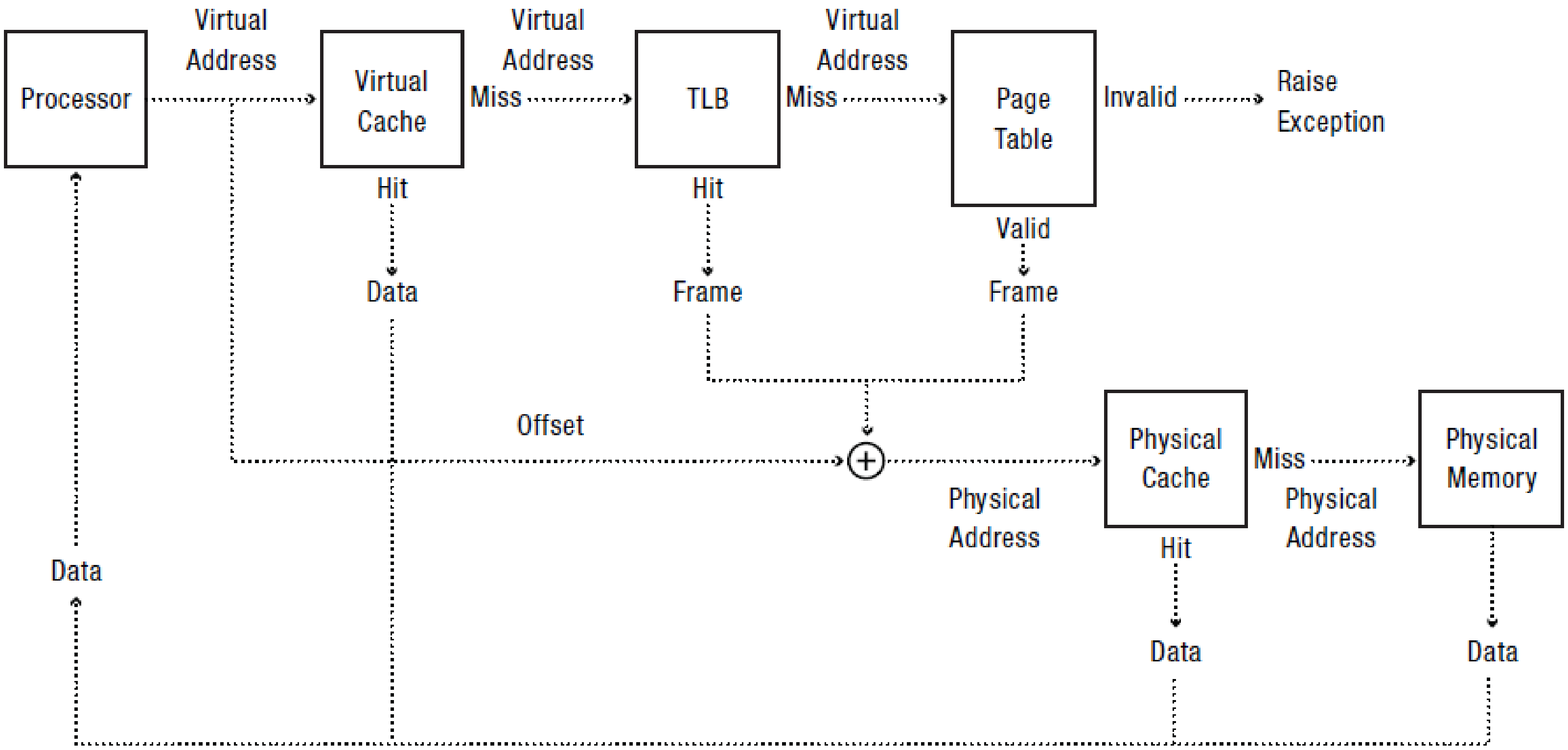
- How about we cache data!
- Too slow to first access TLB to find physical address … particularly for a TLB miss
  - VA -> PA -> data
  - VA -> data
- Instead, first level cache is virtually addressed
- In parallel, access TLB to generate physical address (PA) in case of a cache miss
  - VA -> PA -> data

Virtually Addressed Caches



Same issues w/r to context-switches and consistency

~~Aliasing Solution 2:~~ Physically Addressed Cache



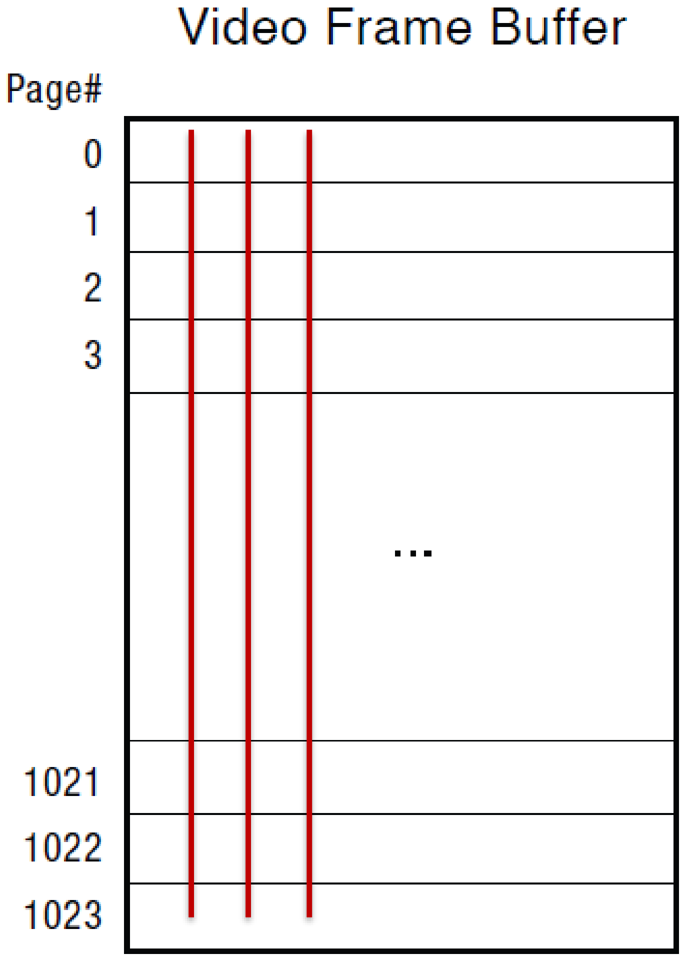
Cache physical translations: at any level! (e.g. frame->data)

Superpages

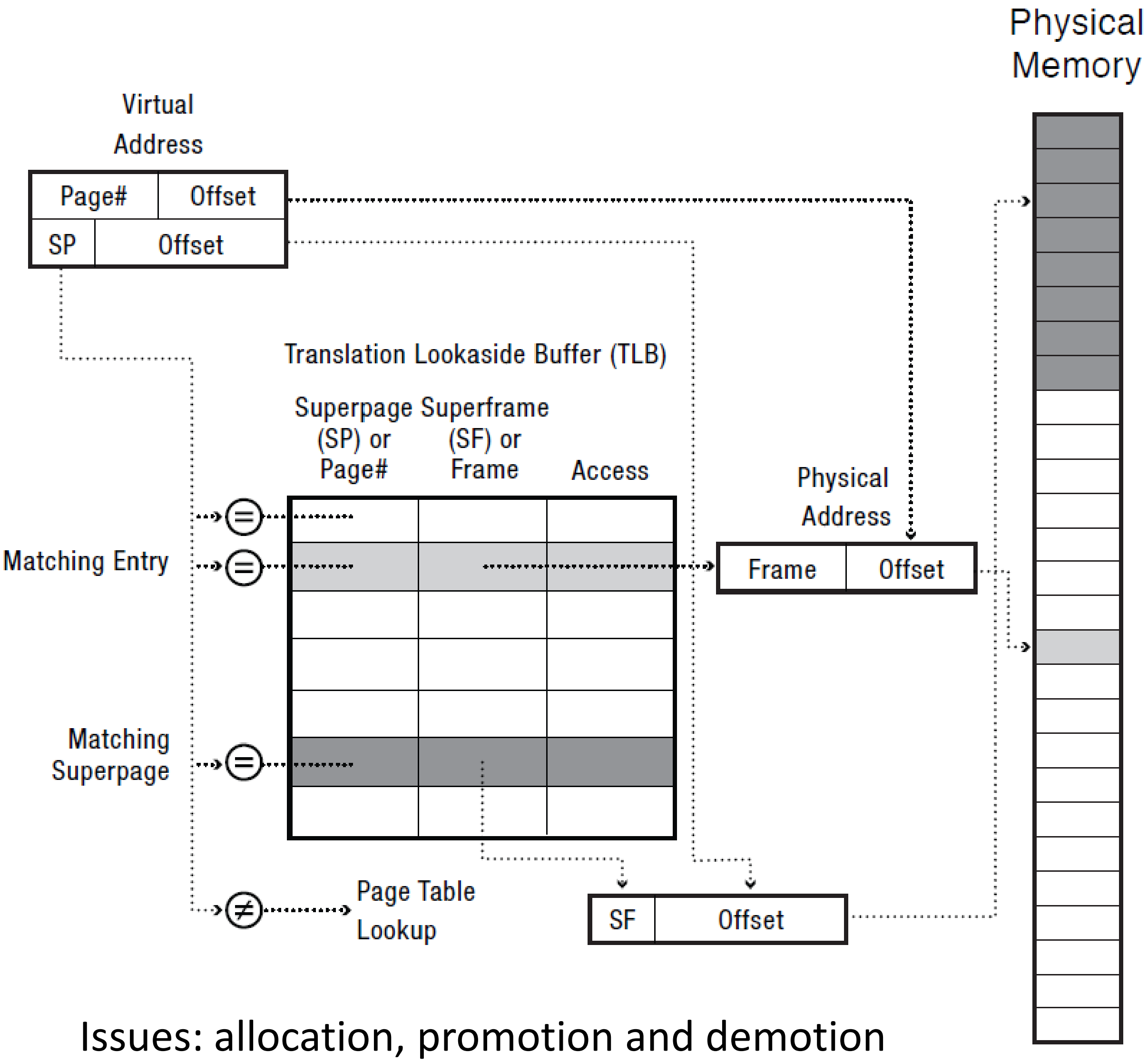
- On many systems, TLB entry can be
  - A page
  - A superpage: a set of contiguous pages
- x86: superpage is set of pages in one page table
  - superpage is memory contiguous
  - x86 also supports a variety of page sizes, OS can choose
    - 4KB
    - 2MB
    - 1GB

Walk an Entire Chunk of Memory

- Video Frame Buffer:
  - 32 bits x 1K x 1K = 4MB
- Very large working set!
  - Draw a ~~horizontal~~ vertical line
  - Lots of TLB misses
- Superpage can reduce this
  - 4MB page



Superpages



Issues: allocation, promotion and demotion

Next Time

- File systems
- Chap 11, OSPP

- Have a great weekend!