

Virtual Memory 3

Hakim Weatherspoon

CS 3410, Spring 2011

Computer Science

Cornell University

Announcements

PA3 available. Due Tuesday, April 19th

- Work with **pairs**
- Be responsible with new knowledge
- **Scheduling a games night, possibly Friday, April 22nd**

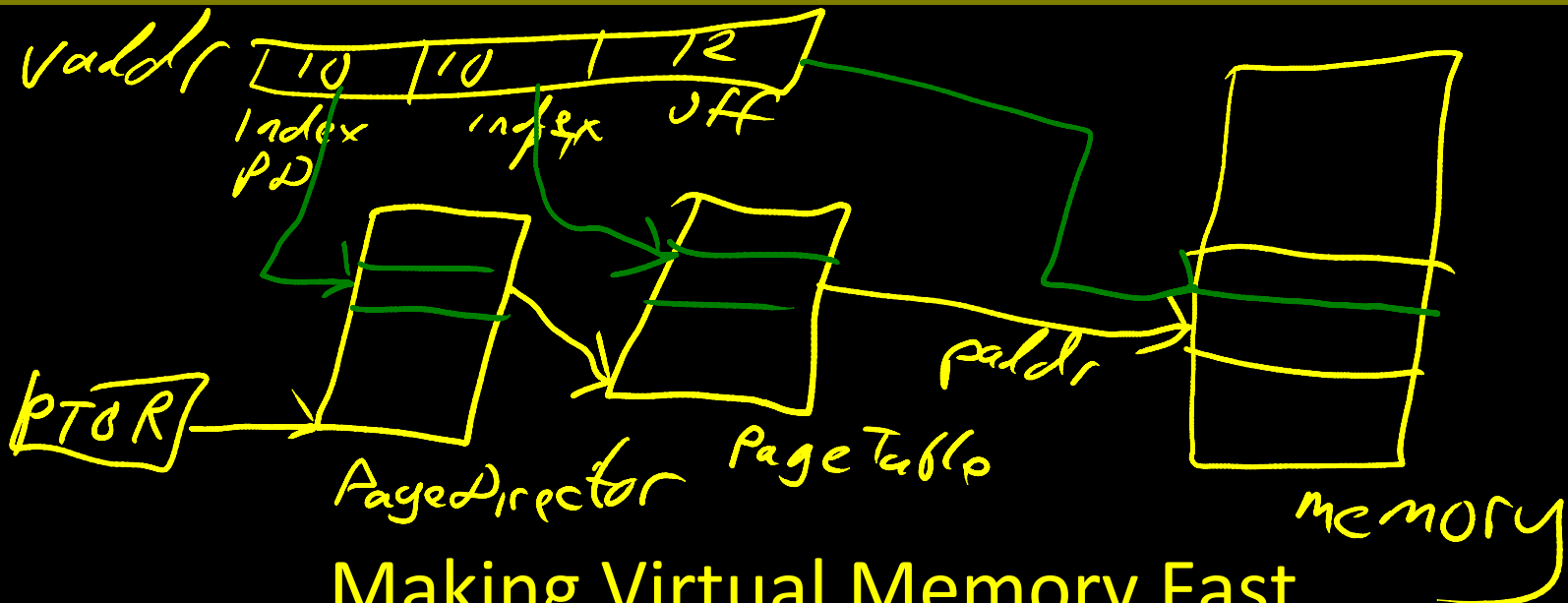
Next four weeks

- Two projects and one homeworks
- Prelim2 will be Thursday, April 28th
- PA4 will be final project (no final exam)
 - ***Will not be able to use slip days***

Goals for Today

Virtual Memory

- Address Translation
 - Pages, page tables, and memory mgmt unit
- Paging
- Role of Operating System
 - Context switches, working set, shared memory
- Performance
 - How slow is it
 - Making virtual memory fast
 - Translation lookaside buffer (TLB)
- Virtual Memory Meets Caching



Making Virtual Memory Fast

The Translation Lookaside Buffer (TLB)

cache for translation

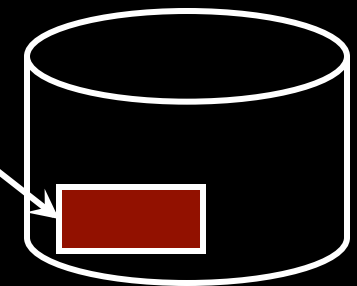
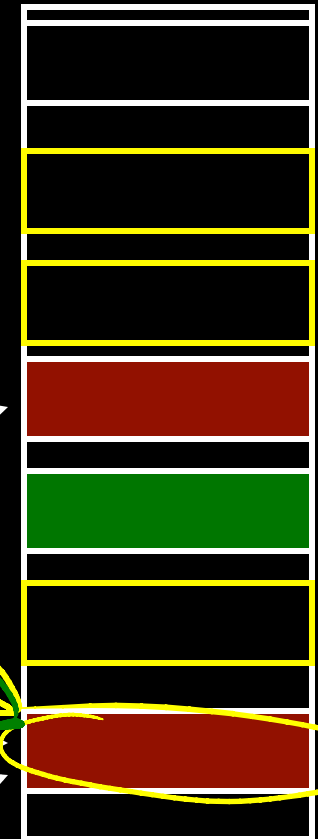
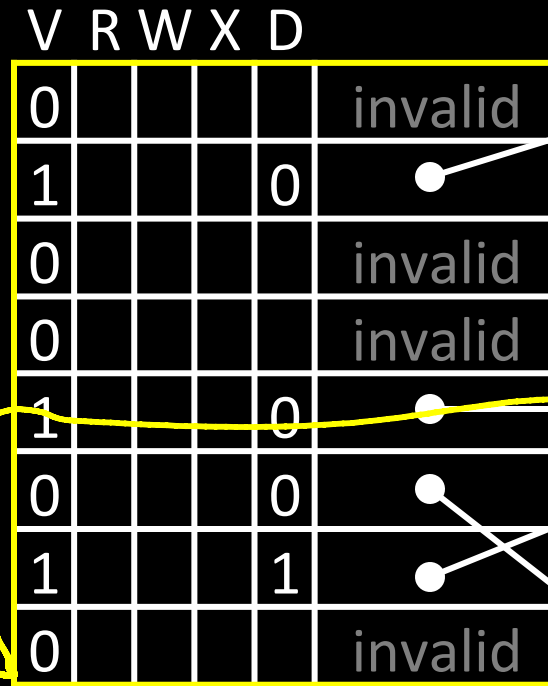
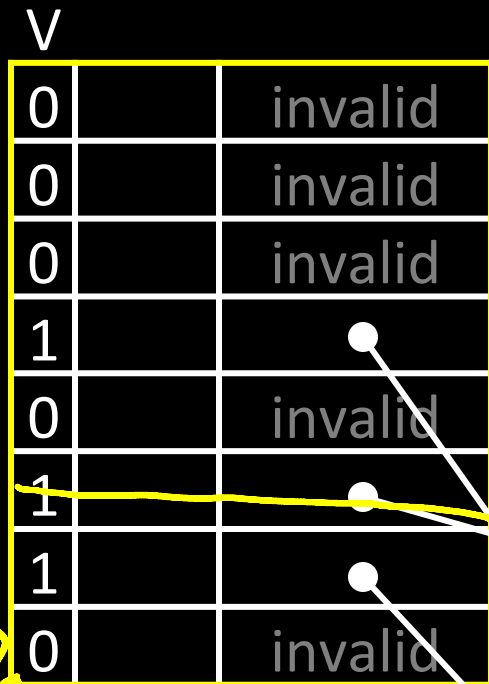
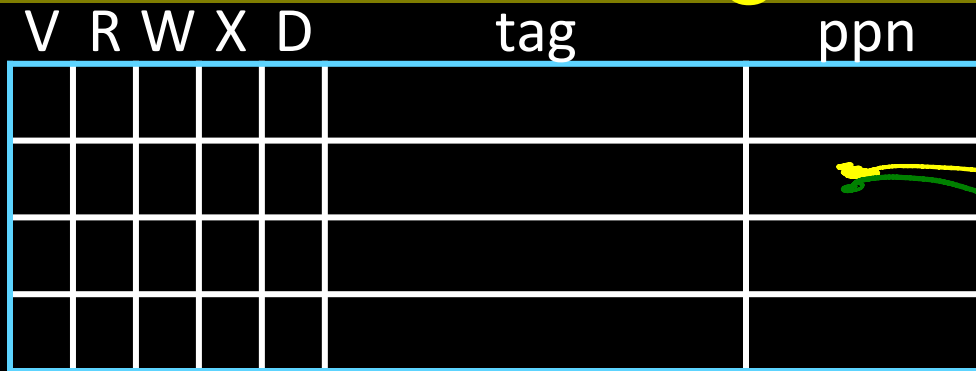
Translation Lookaside Buffer (TLB)

Hardware Translation Lookaside Buffer (TLB)

A small, very fast cache of recent address mappings

- TLB hit: avoids PageTable lookup
- TLB miss: do PageTable lookup, cache result for later

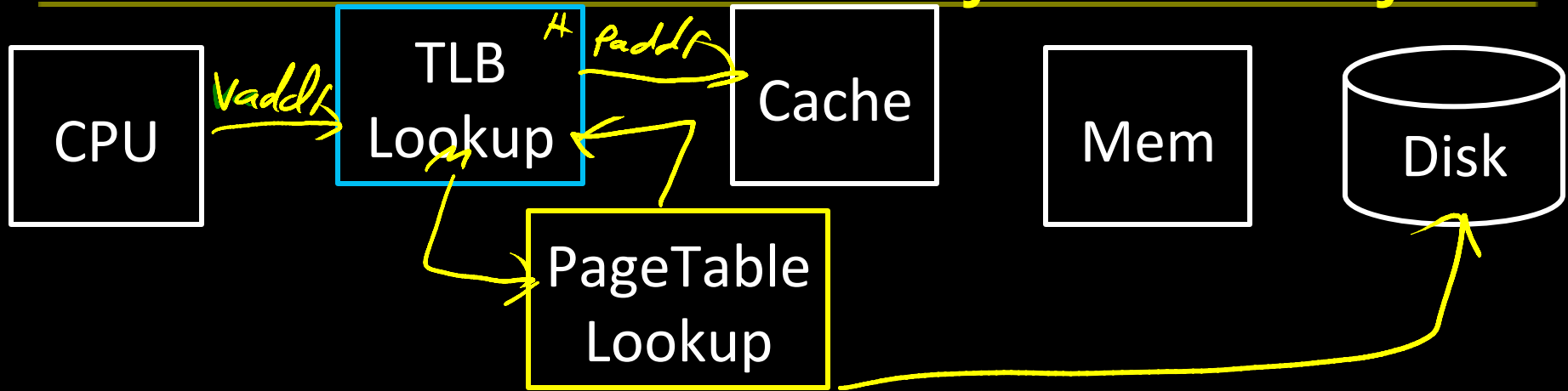
TLB Diagram



Handwritten: #10R, PD

Handwritten: PT

A TLB in the Memory Hierarchy



(1) Check TLB for vaddr (~ 1 cycle)

(2) TLB Hit

- compute paddr, send to cache

(2) TLB Miss: traverse PageTables for vaddr

(3a) PageTable has valid entry for in-memory page

- Load PageTable entry into TLB; try again (tens of cycles)

(3b) PageTable has entry for swapped-out (on-disk) page

- Page Fault: load from disk, fix PageTable, try again (millions of cycles)

(3c) PageTable has invalid entry

- Page Fault: kill process

TLB Coherency

TLB Coherency: What can go wrong?

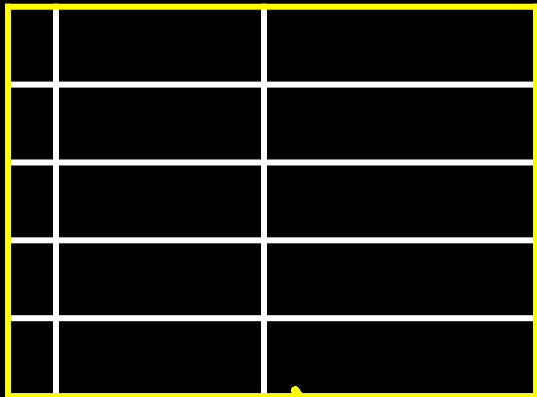
A: PageTable or PageDir contents change

- swapping/paging activity, new shared pages, ...

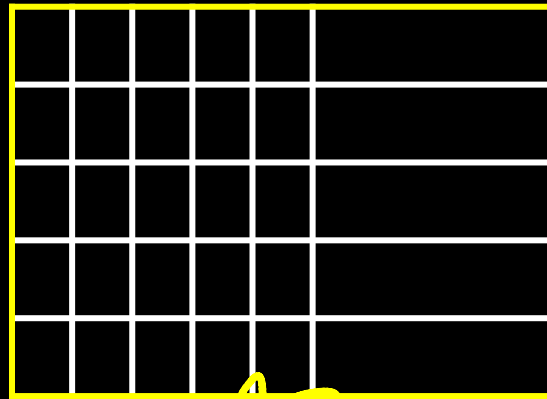
A: Page Table Base Register changes

- context switch between processes

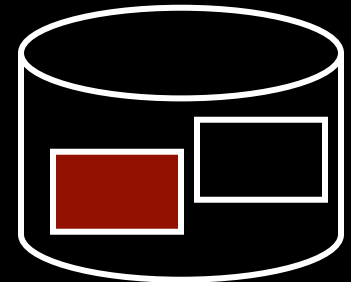
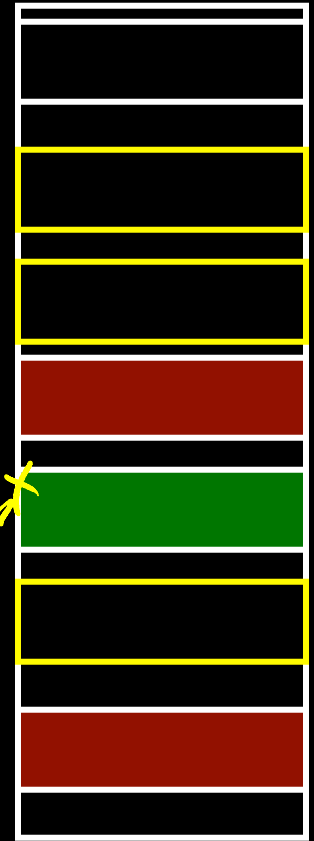
How to keep
PTBR, PT,
TLB consistent?
TLB ?



PD



PT



Translation Lookaside Buffers (TLBs)

When PTE changes, PDE changes, PTBR changes....

Full Transparency: **TLB coherency in hardware**

- Flush TLB whenever PTBR register changes
[easy – why?] – *easy, but expensive (port)*
- Invalidate entries whenever PTE or PDE changes
[hard – why?] – *better port, but harder
PID bookkeeping*

TLB coherency in software

If TLB has a no-write policy...

- OS invalidates entry after OS modifies page tables
- OS flushes TLB whenever OS does context switch

TLB Parameters

TLB parameters (typical)

- very small (64 – 256 entries), so very fast
- fully associative, or at least set associative
- tiny block size: why?

*misses
are
expensive*

Intel Nehalem TLB (example)

- 128-entry L1 Instruction TLB, 4-way LRU
- 64-entry L1 Data TLB, 4-way LRU
- 512-entry L2 Unified TLB, 4-way LRU

Virtual Memory meets Caching

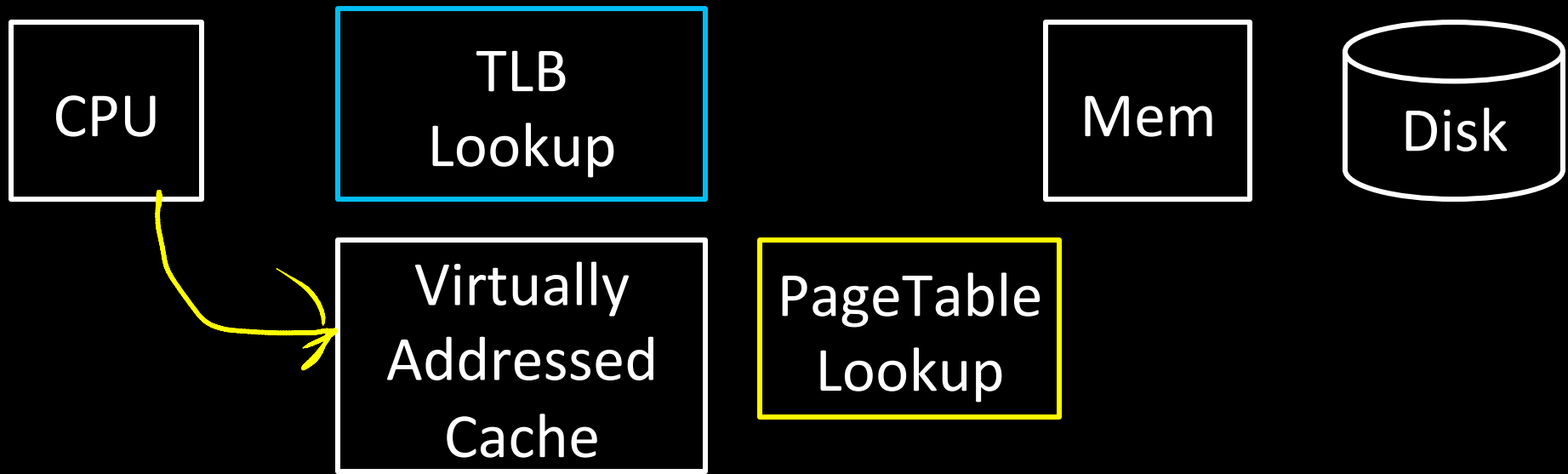
Virtually vs. physically addressed caches

Virtually vs. physically tagged caches

Virtually Addressed Caching

Q: Can we remove the TLB from the critical path?

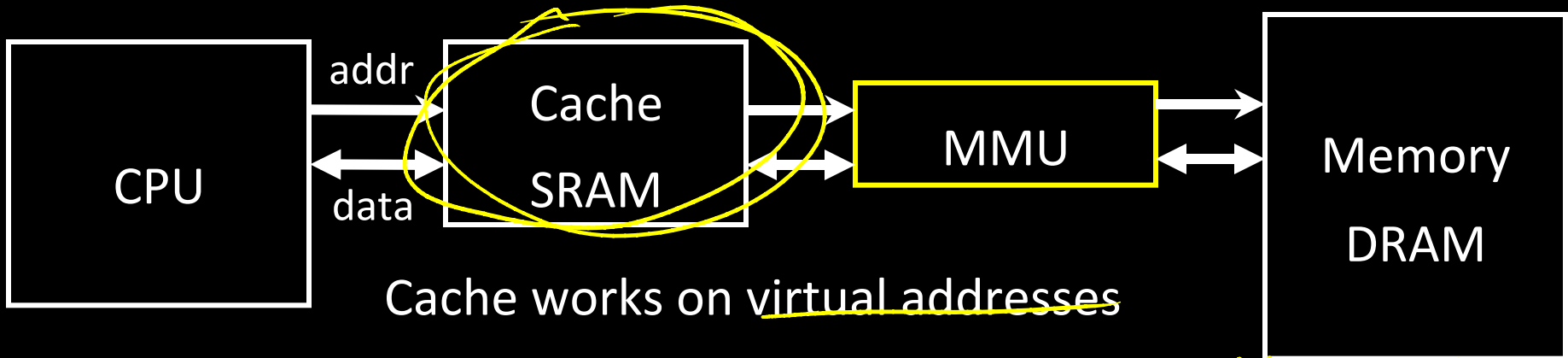
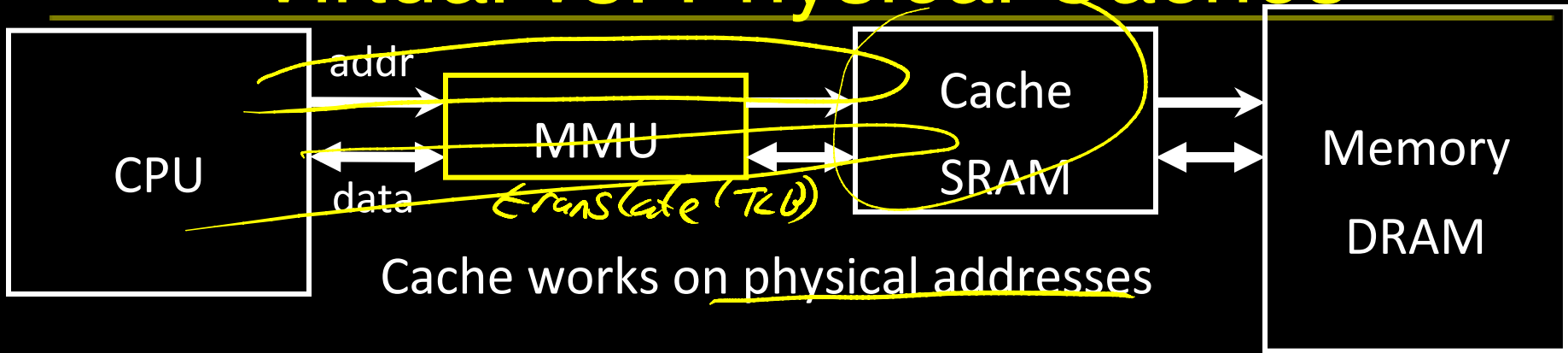
A: Virtually-Addressed Caches



① flush TLB on context switch

② Synonyms - Aliasing
make virt addr cache hard
two - Vaddr's map to same phys addr

Virtual vs. Physical Caches



- Q: What happens on context switch? *Phys Addr - nothing*
Virt Addr - flush
- Q: What about virtual memory aliasing? *Phys Addr - no issue*
virt addr - BIG problems
- Q: So what's wrong with physically addressed caches?
perf is bad
want - virt addr + Phys tag (alias issue)

Indexing vs. Tagging

Physically-Addressed Cache

Virtually-Addressed Cache

first

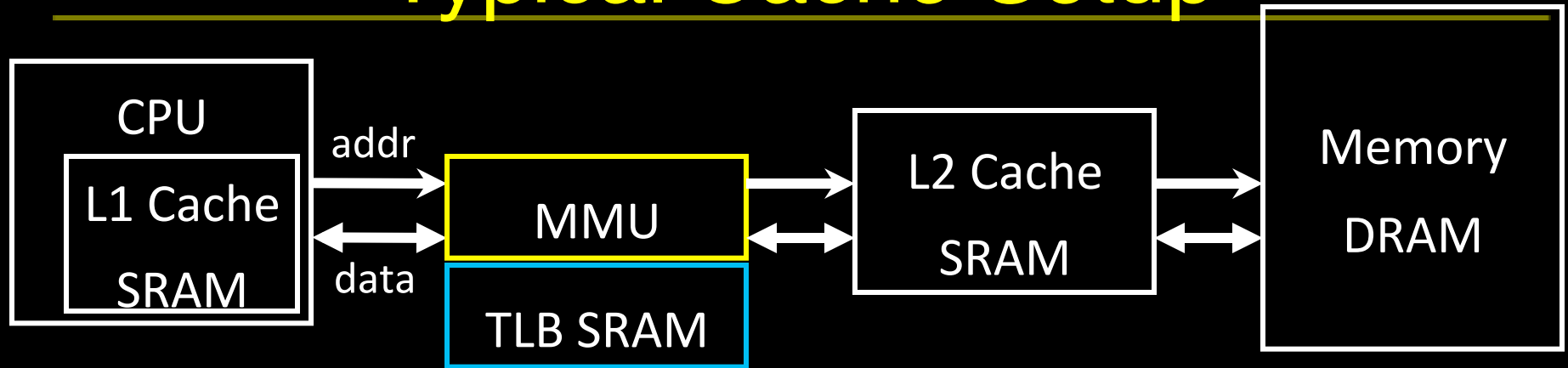
Virtually-Indexed, Virtually Tagged Cache

- fast: start TLB lookup before cache lookup finishes
- PageTable changes (paging, context switch, etc.)
 - need to purge stale cache lines (how?)
- Synonyms (two virtual mappings for one physical page)
 - could end up in cache twice (very bad!)

Virtually-Indexed, Physically Tagged Cache

- ~fast: TLB lookup in parallel with cache lookup
- PageTable changes → no problem: phys. tag mismatch
- Synonyms → search and evict lines with same phys. tag

Typical Cache Setup



Typical L1: On-chip **virtually** addressed, **physically** tagged

Typical L2: On-chip **physically** addressed

Typical L3: On-chip ...

Caches/TLBs/MM

Caches, Virtual Memory, & TLBs

Where can block be placed?

- Direct, n-way, fully associative

What block is replaced on miss?

- LRU, Random, LFU, ...

How are writes handled?

- No-write (w/ or w/o automatic invalidation)
- Write-back (fast, block at time)
- Write-through (simple, reason about consistency)

Summary of Cache Design Parameters

	L1	Paged Memory	TLB
Size (blocks)	1/4k to 4k	16k to 1M	64 to 4k
Size (kB)	16 to 64	1M to 4G	2 to 16
Block size (B)	16-64	4k to 64k	4-32
Miss rates	2%-5%	10^{-4} to $10^{-5}\%$	0.01% to 2%
Miss penalty	10-25	10M-100M	100-1000