Lecture-19 (Virtual Memory and Caches) CS422-Spring 2018

Biswa@CSE-IITK

Summary

- **Reducing hit time**

1. Small and simple caches
2. Way prediction
3. Trace caches

- **Increasing cache bandwidth**

4. Pipelined caches
5. Multibanked caches
6. Nonblocking caches

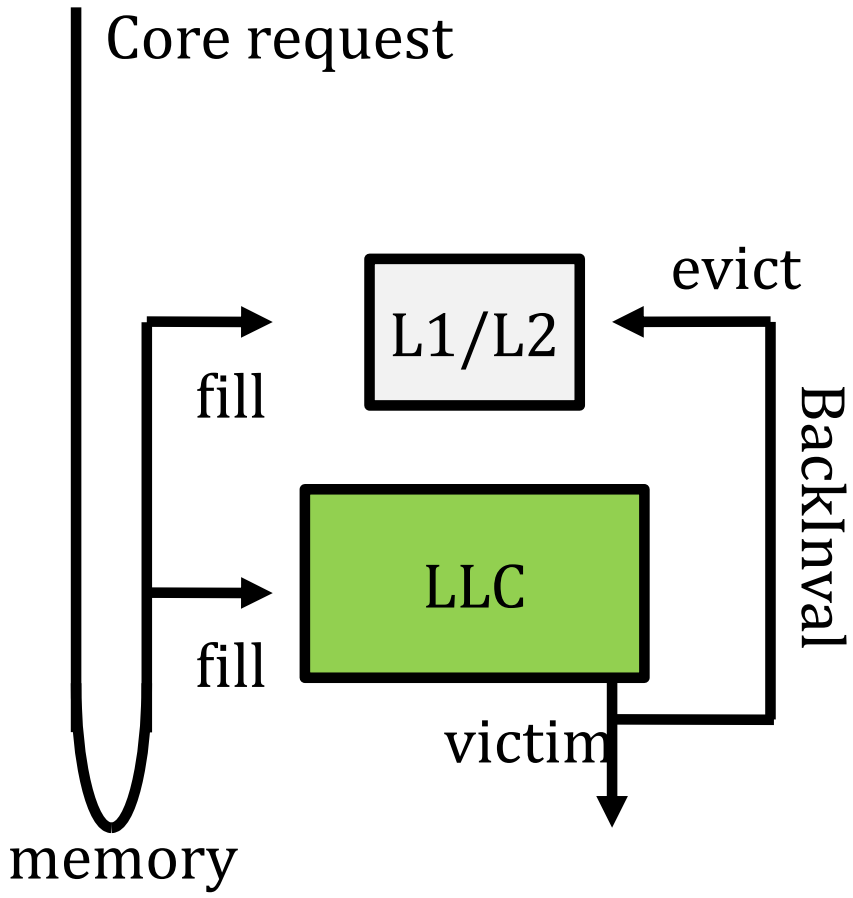
- **Reducing Miss Penalty**

7. Critical word first
8. Early Restart

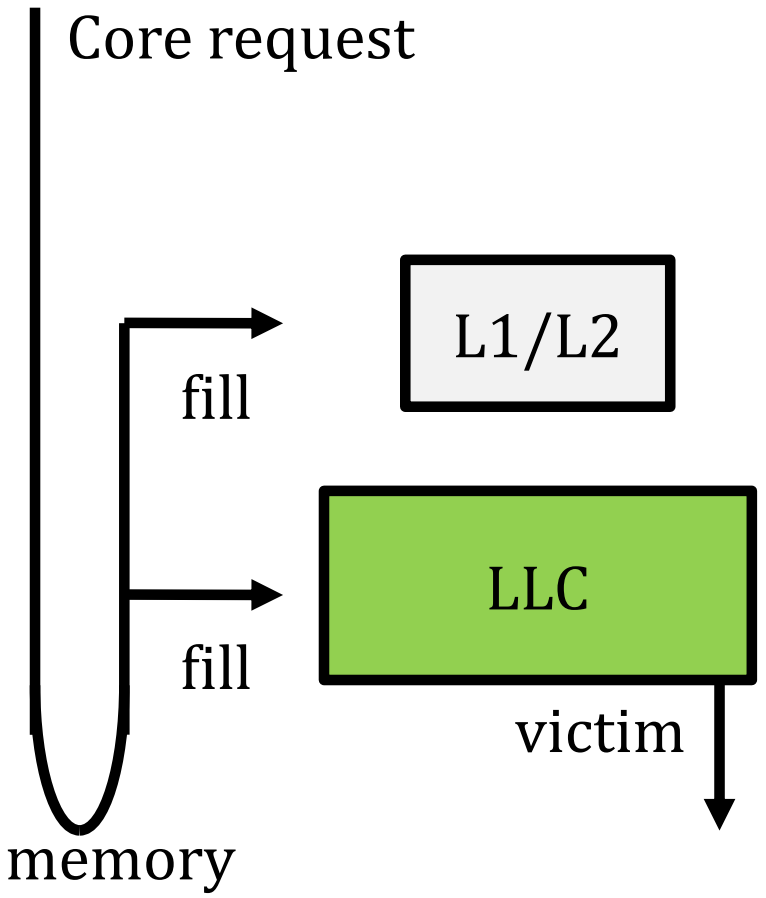
- **Reducing Miss Rate**

9. Victim Cache
10. Hardware prefetching
11. Compiler prefetching
12. Compiler Optimizations

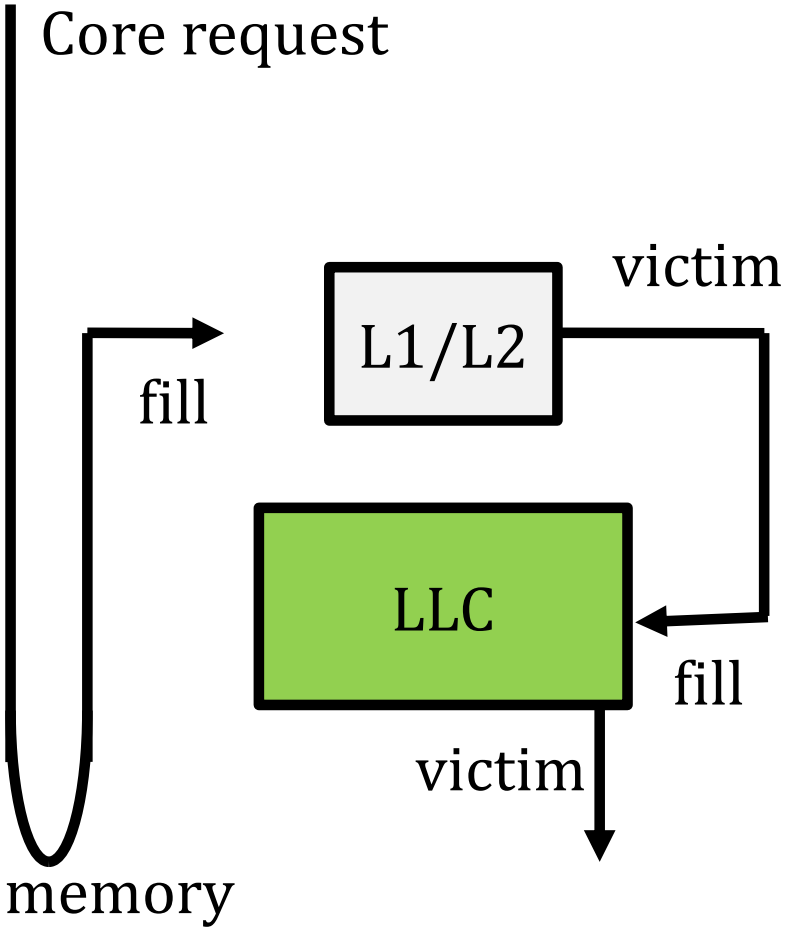
Inclusion- Cache Hierarchy



Inclusive



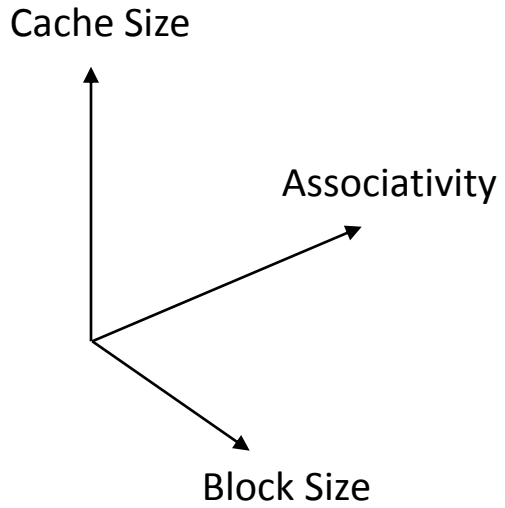
Non-Inclusive

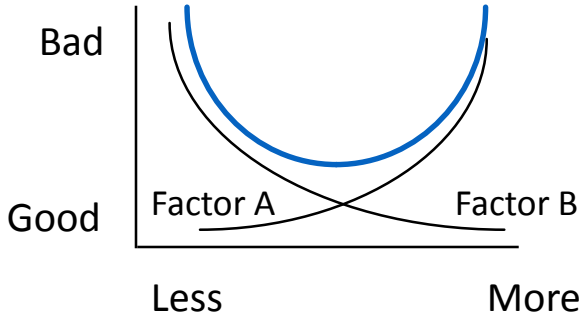


Exclusive

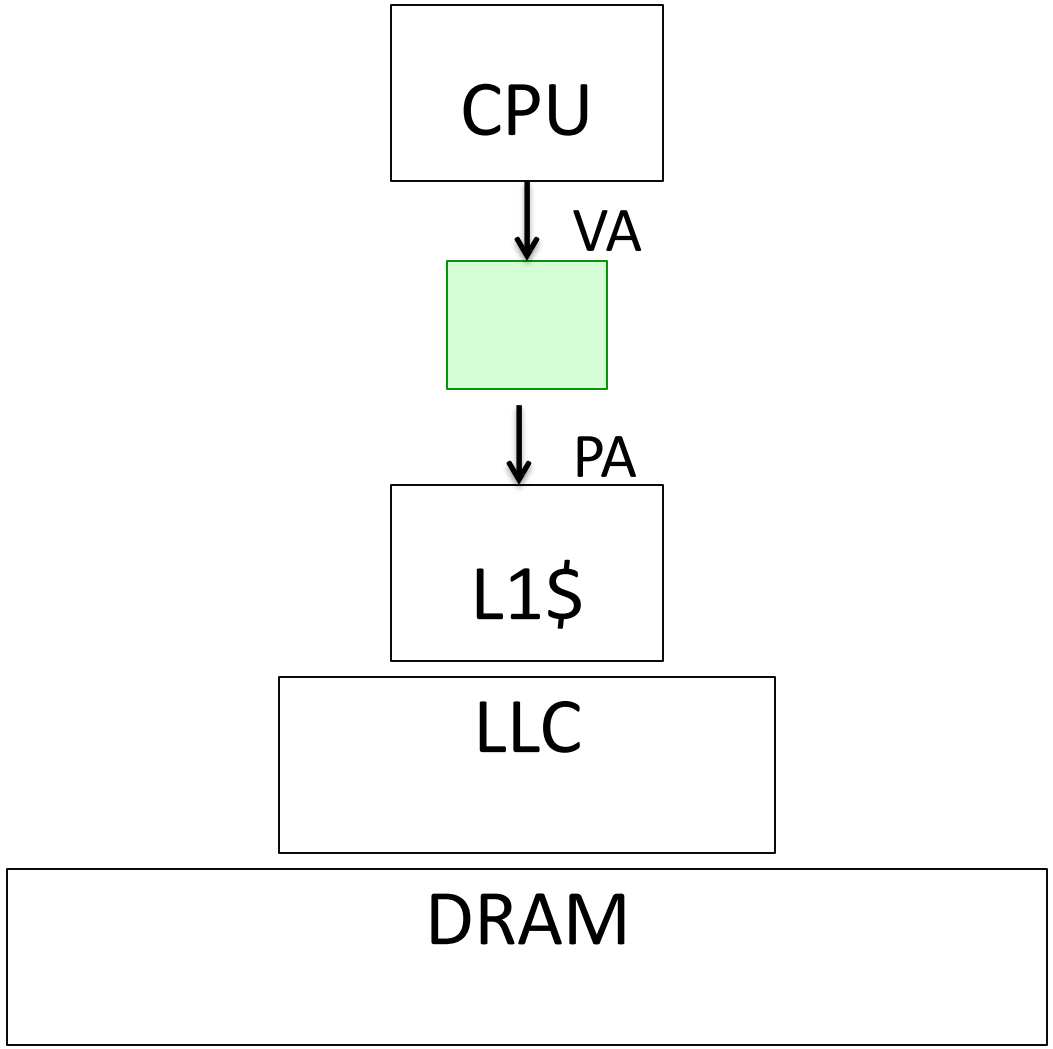
Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocation
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins

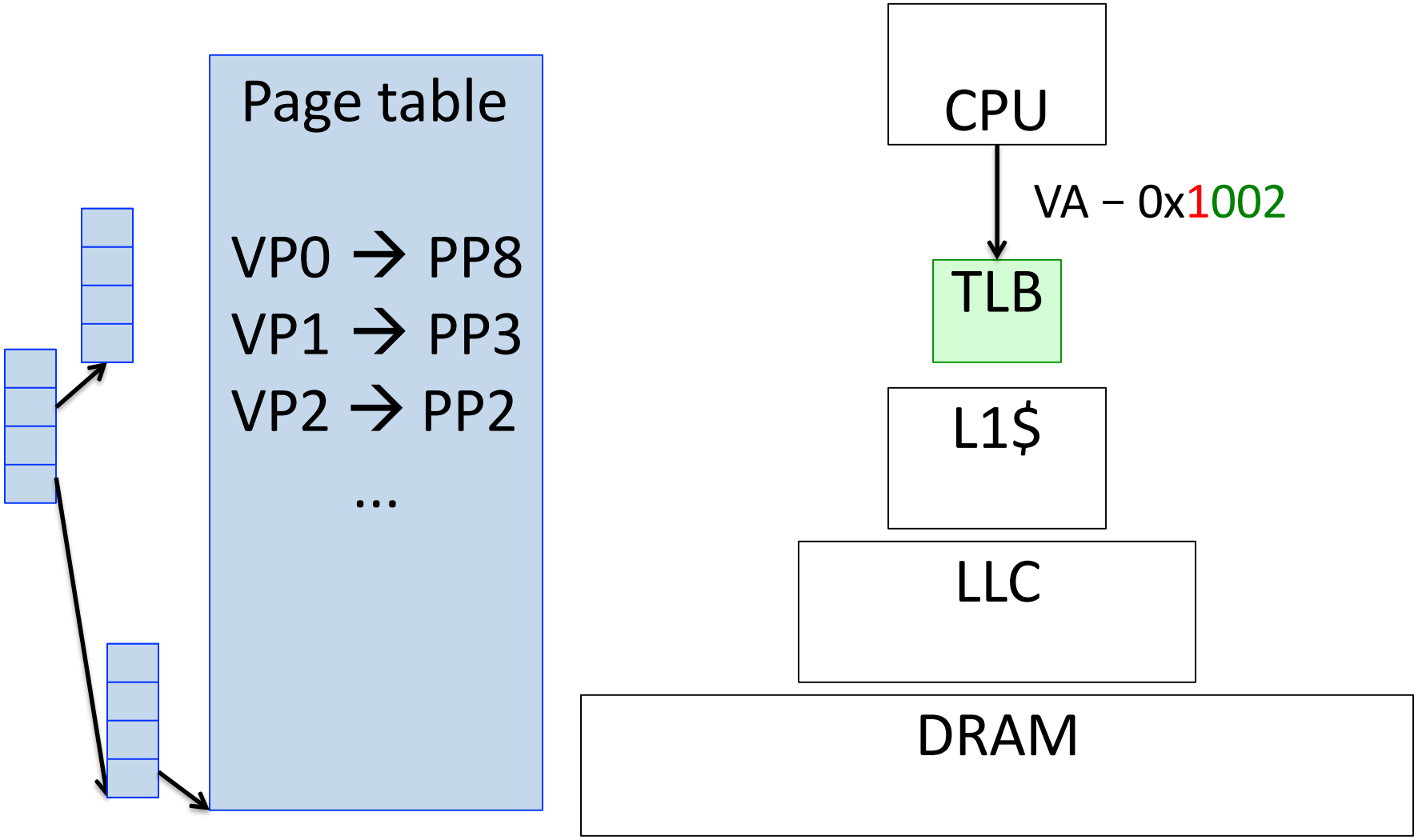




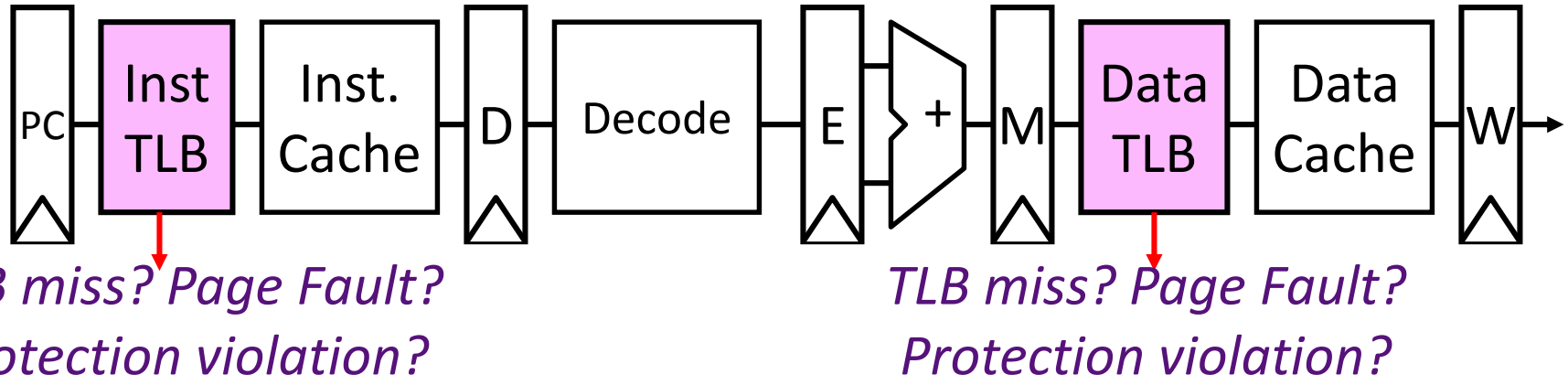
Address Translation – Welcome to the Virtual World



Address Translation



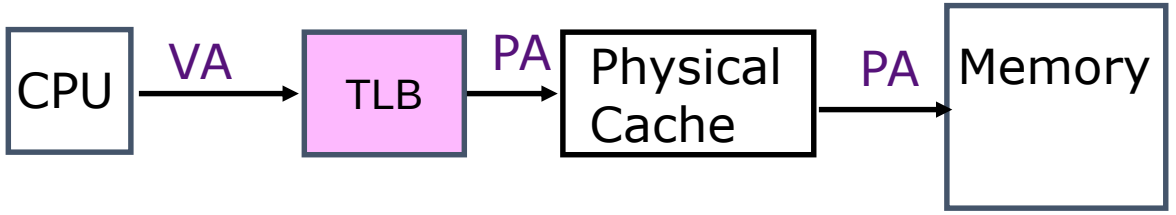
Virtual World?



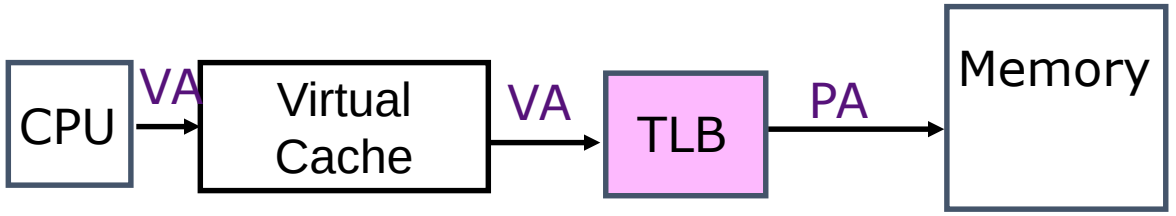
- Need to cope with additional latency of TLB:
  - slow down the clock?
  - pipeline the TLB and cache access?
  - virtual address caches
  - parallel TLB/cache access

Why Virtual? (CS330?)

Caches: Virtual or Physical?

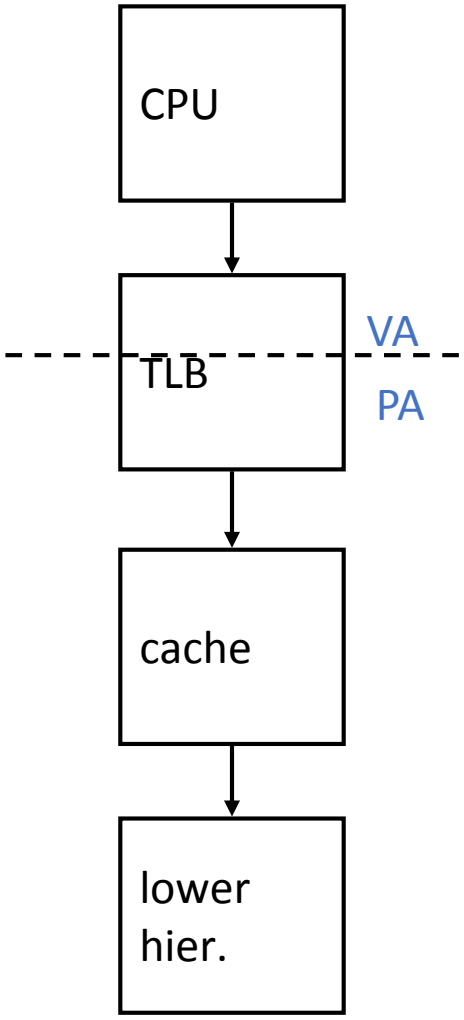


Alternative: place the cache before the TLB

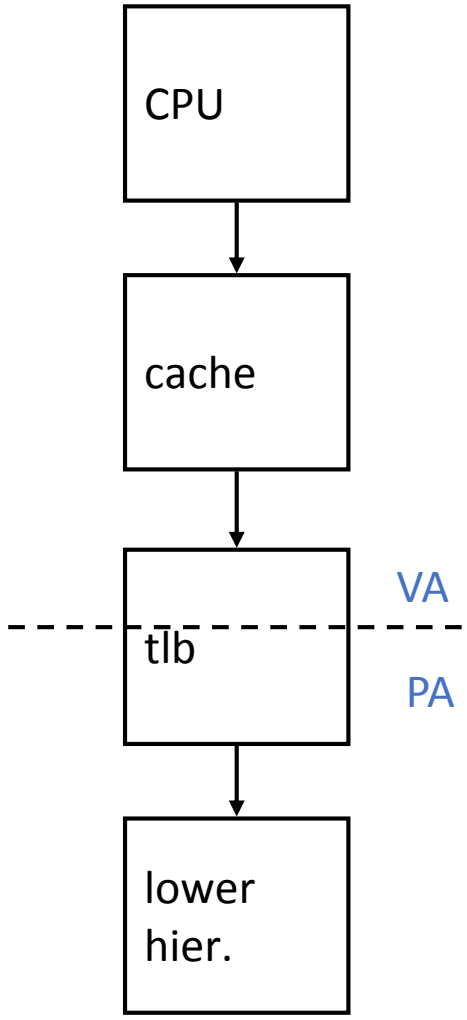


- one-step process in case of a hit (+)
- cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- *aliasing problems* due to the sharing of pages (-)
- maintaining cache coherence (-)

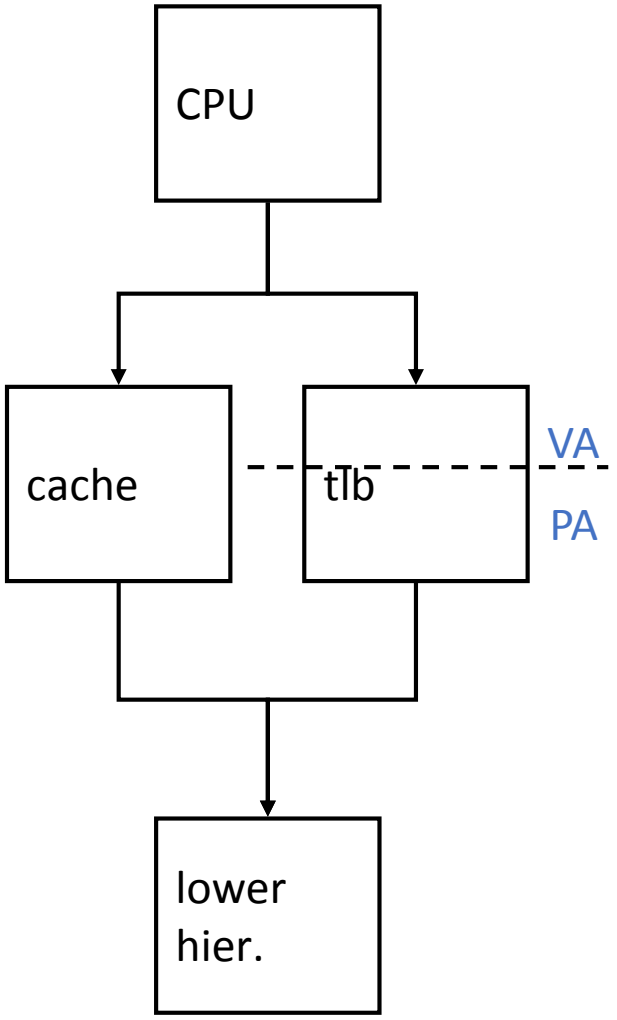
Cache and Virtual Memory



physical cache

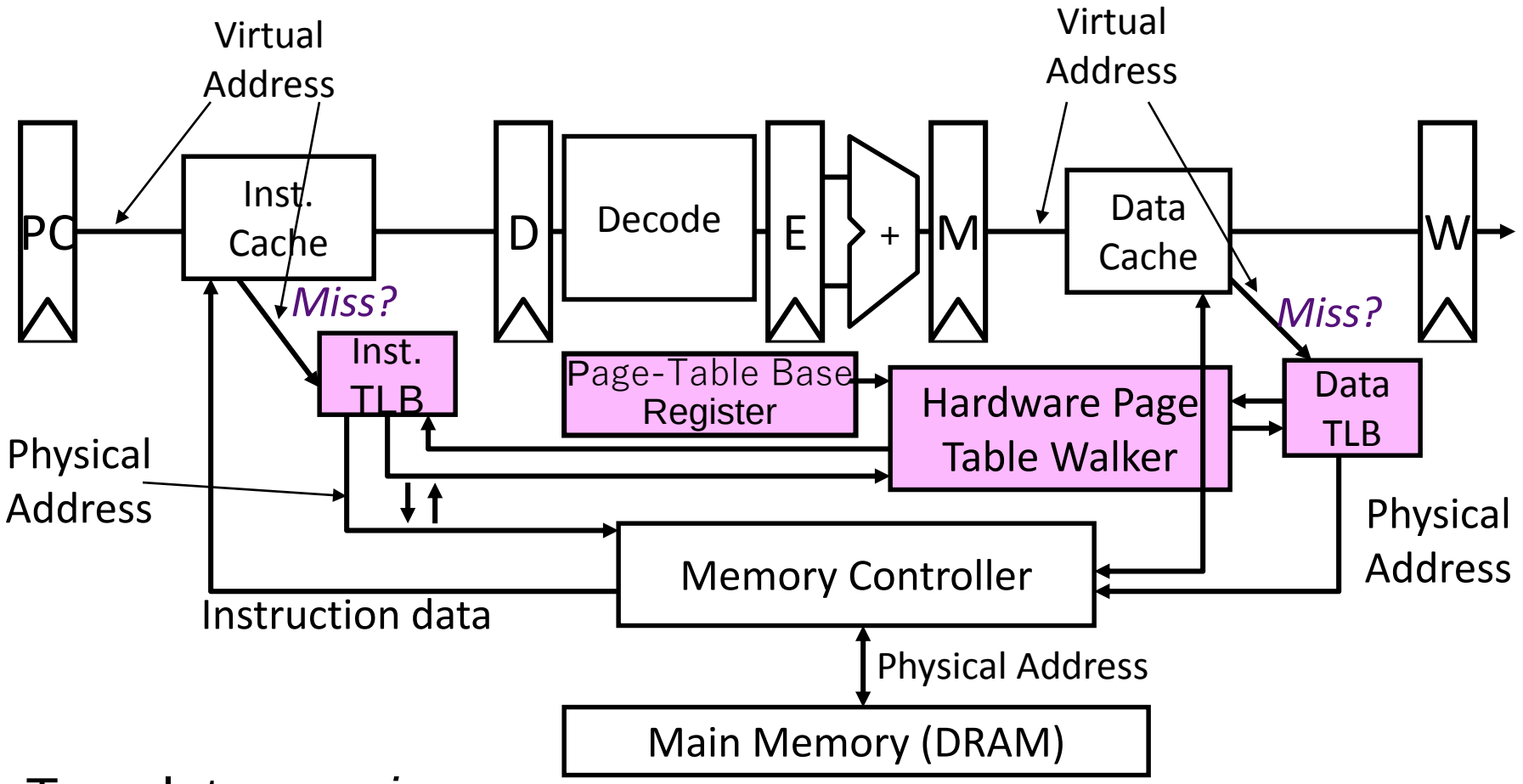


virtual (L1) cache



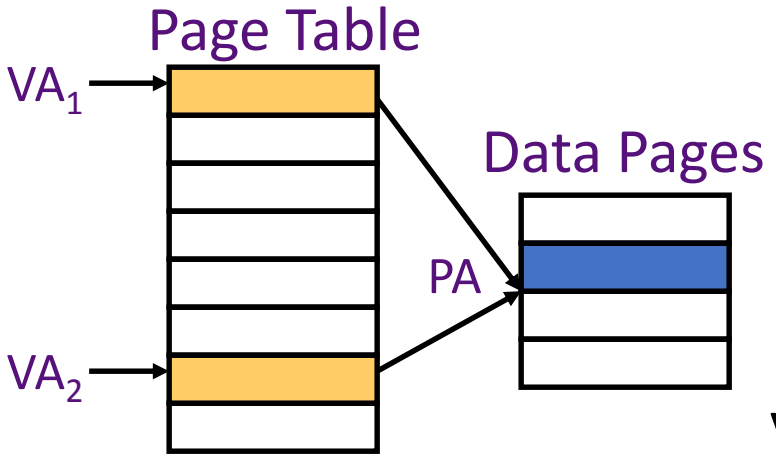
virtual-physical cache

Virtual Caches



Translate on *miss*

Problem: Aliasing (Synonyms, Homonyms?)



Two virtual pages share one physical page

Tag	Data
VA_1	1st Copy of Data at PA
VA_2	2nd Copy of Data at PA

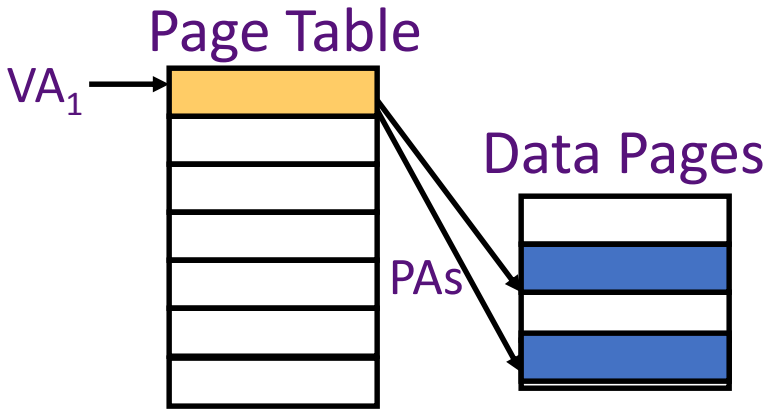
Virtual cache can have two copies of same physical data. Writes to one copy not visible to reads of other!

General Solution: *Prevent aliases coexisting in cache*

Software (i.e., OS) solution for direct-mapped cache

VAs of shared pages must agree in cache index bits; this ensures all VAs accessing same PA will conflict in direct-mapped cache (early SPARCs)

Homonyms (Virtual Tags)

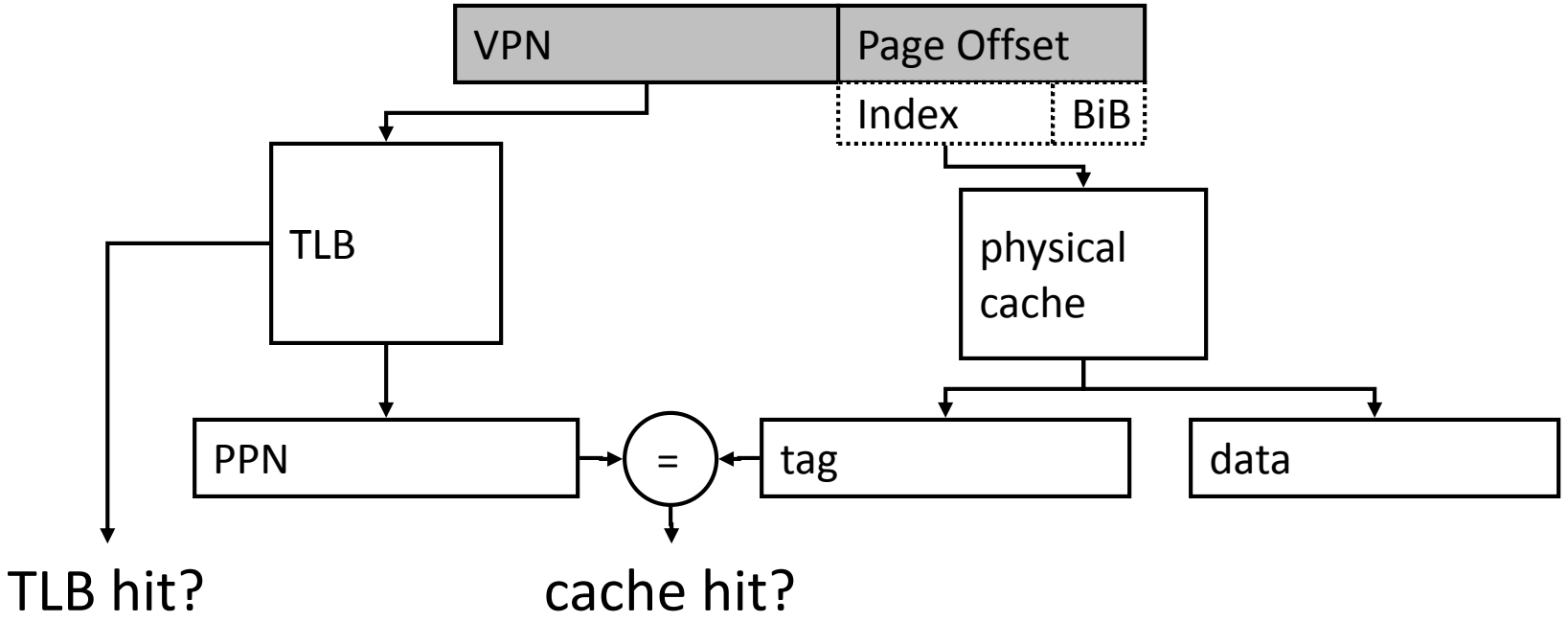


One virtual page maps to two physical pages

Tag may not uniquely identify cache data
Solution: Add ASID with tag
Or
Physical tags
Or
Flush on context switch

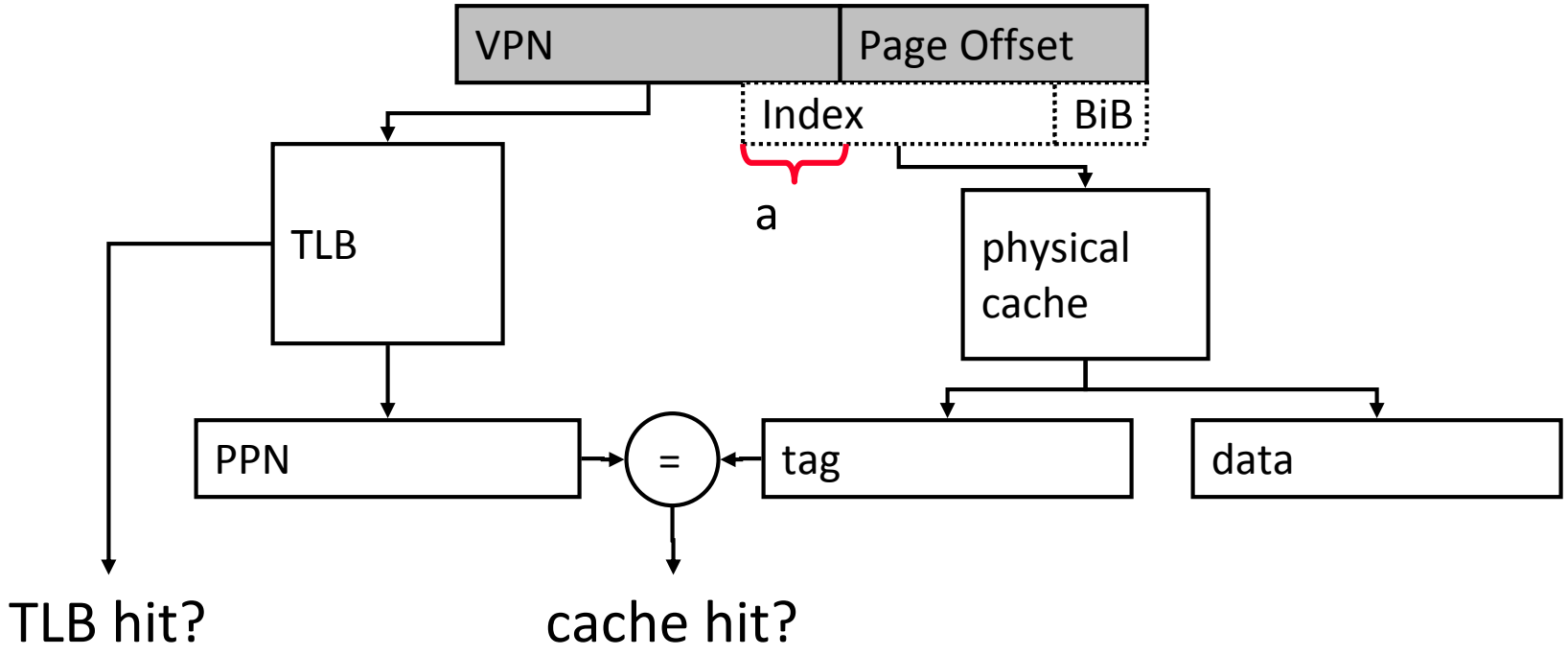
Virtual Memory (Virtually Indexed Physically Tagged)

- If C≤(page_size × associativity), the cache index bits come only from page offset (same in VA and PA)
- If both cache and TLB are on chip
  - index both arrays concurrently using VA bits
  - check cache tag (physical) against TLB output at the end



Virtual Memory (Virtually Indexed Physically Tagged)

- If C>(page_size × associativity), the cache index bits include VPN ⇒ Synonyms can cause problems
  - The same physical address can exist in two locations
- Solutions?



Four Possibilities

- VIVT (Virtual cache)
- VIPT
- PIVT
- PIPT (Physical cache)

Four Possibilities: Cache Performance

- VIVT (Virtual cache): Fastest, Synonyms, Homonyms,
- VIPT: Good enough, No Homonyms, mostly in L1 caches
- PIVT: ??
- PIPT (Physical cache): You know it

Caches+TLB+Page-Table+Prefetcher