Virtual Memory Part 2

Arkaprava Basu

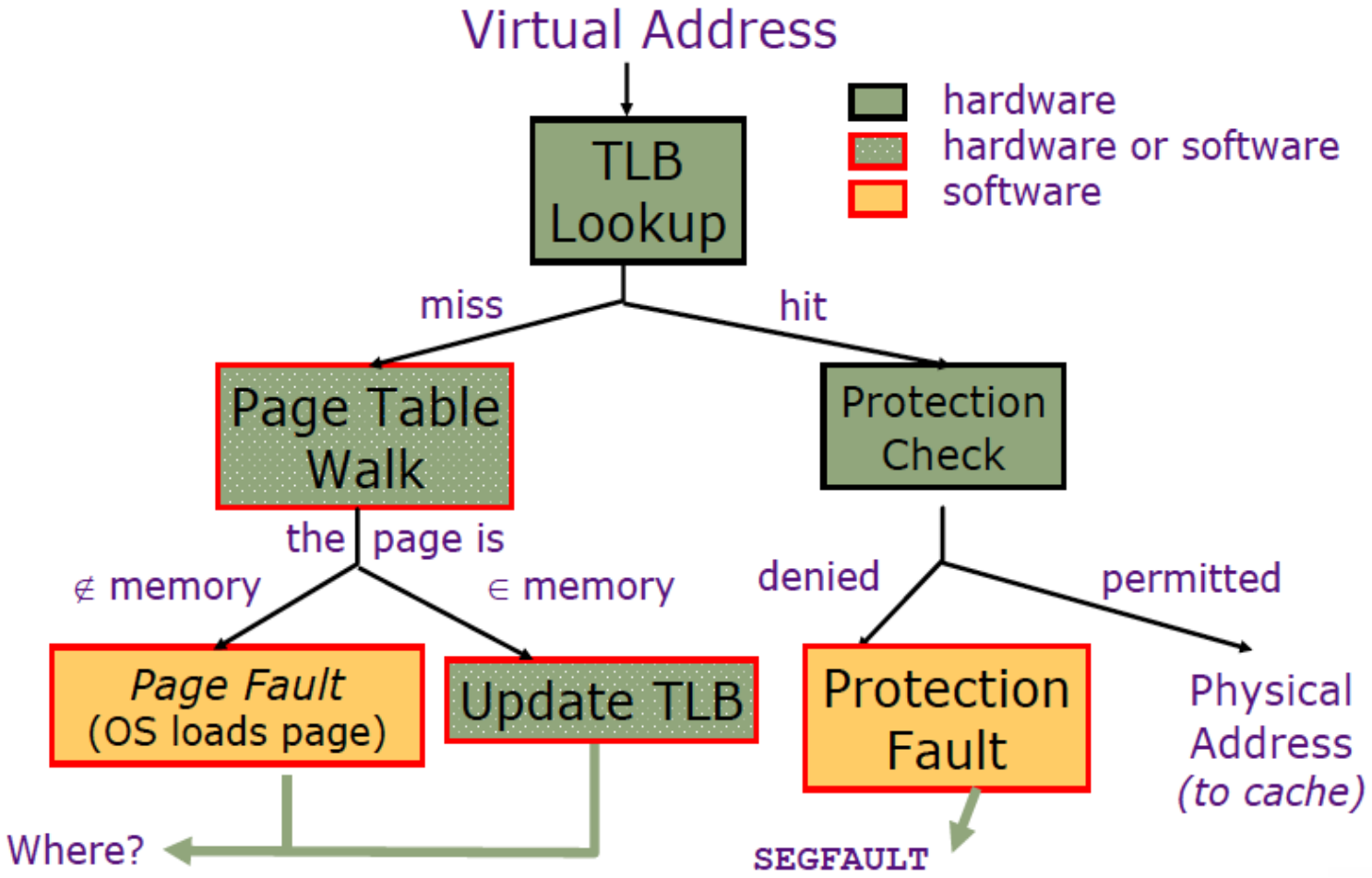
Dept. of Computer Science and Automation
Indian Institute of Science
www.csa.iisc.ac.in/~arkapravab/

Acknowledgements

- Some of the slides in the deck were provided by Profs. Luis Ceze (Washington), Nima Horanmand (Stony Brook), Mark Hill, David Wood, Karu Sankaralingam (Wisconsin), Abhishek Bhattacharjee(Yale).

- Development of this course is partially supported by funding from Western Digital corporations.

Putting it all Together



Interactions between Caching and Virtual memory

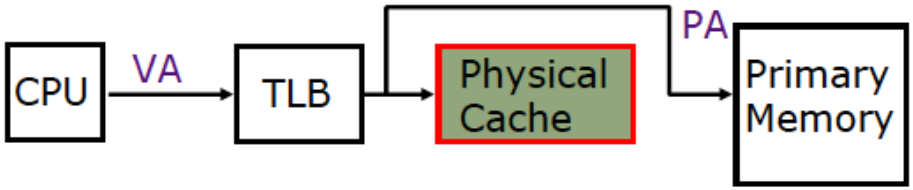
What type of address a cache use?

- Software generates load/stores with virtual address
- Memory is addressed by physical addresses
- TLB/PTWs does the address translation

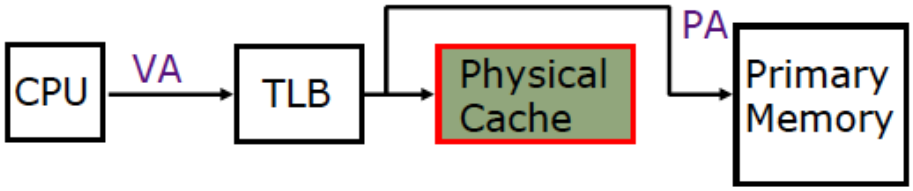
What type of address a cache use?

- Software generates load/stores with virtual address
- Memory is addressed by physical addresses
- TLB/PTWs does the address translation
- **But, what type of addresses caches use?**
  - **Virtual or physical addresses?**

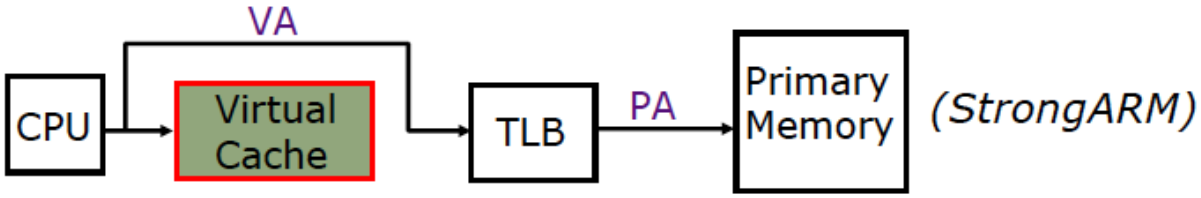
Virtually -Addressed and Physically -Addressed Caches



Virtually -Addressed and Physically -Addressed Caches

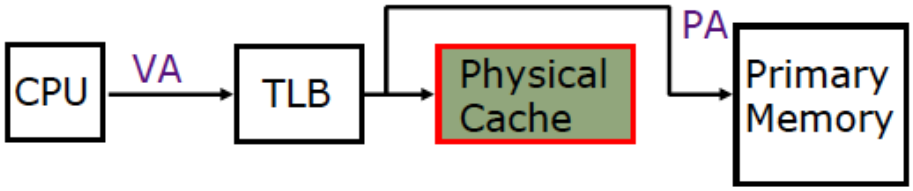


Alternative: place the cache before the TLB

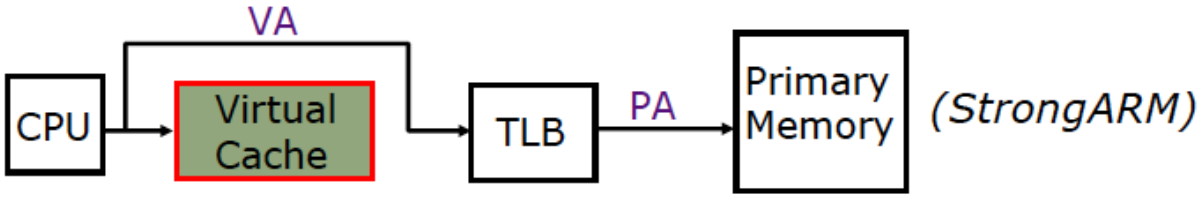


- No need to do address translation on a cache hit (good)

Virtually -Addressed and Physically -Addressed Caches

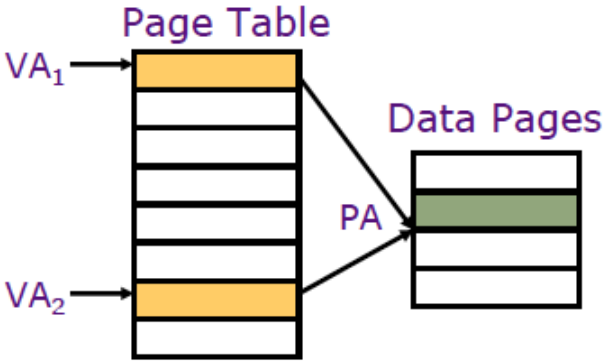


Alternative: place the cache before the TLB



- No need to do address translation on a cache hit (good)
- Cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (bad).
- ***Aliasing (synomym and homonym)*** problems due to sharing of pages (bad).

Aliasing in Virtually -Addressed Caches

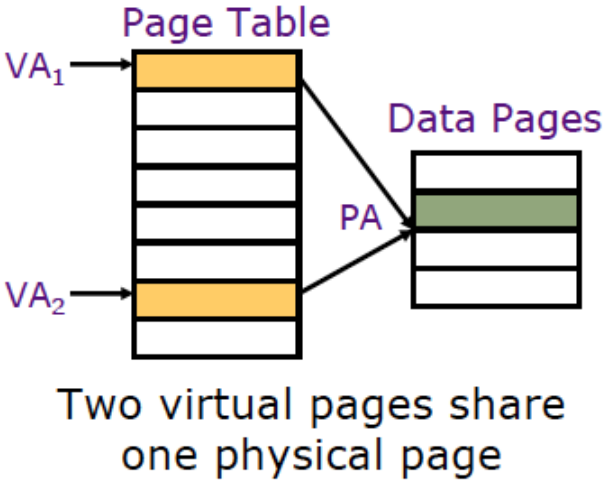


Two virtual pages share one physical page

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!

Aliasing in Virtually -Addressed Caches



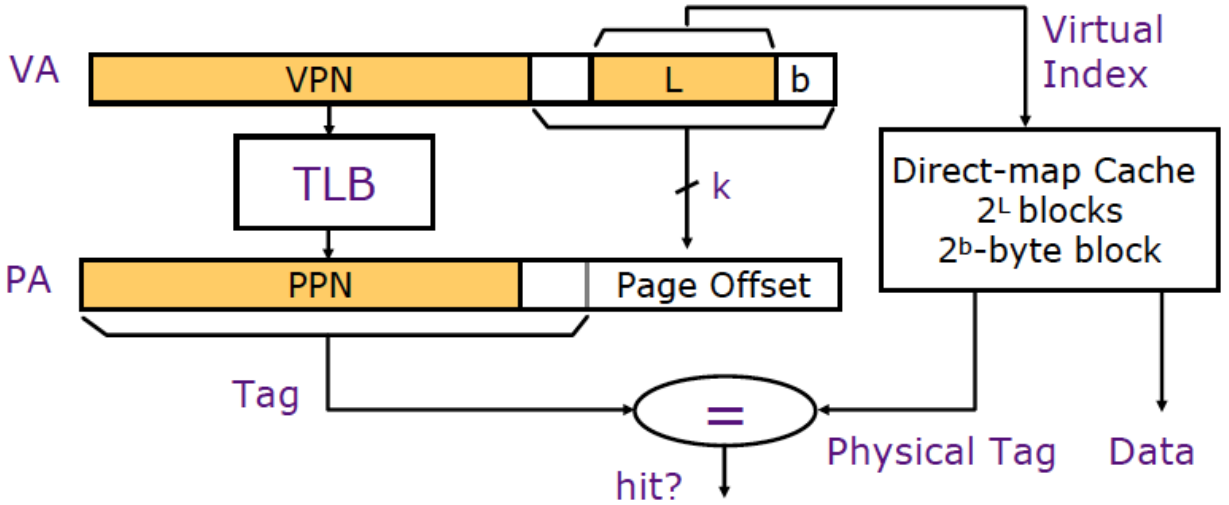
Two virtual pages share one physical page

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!

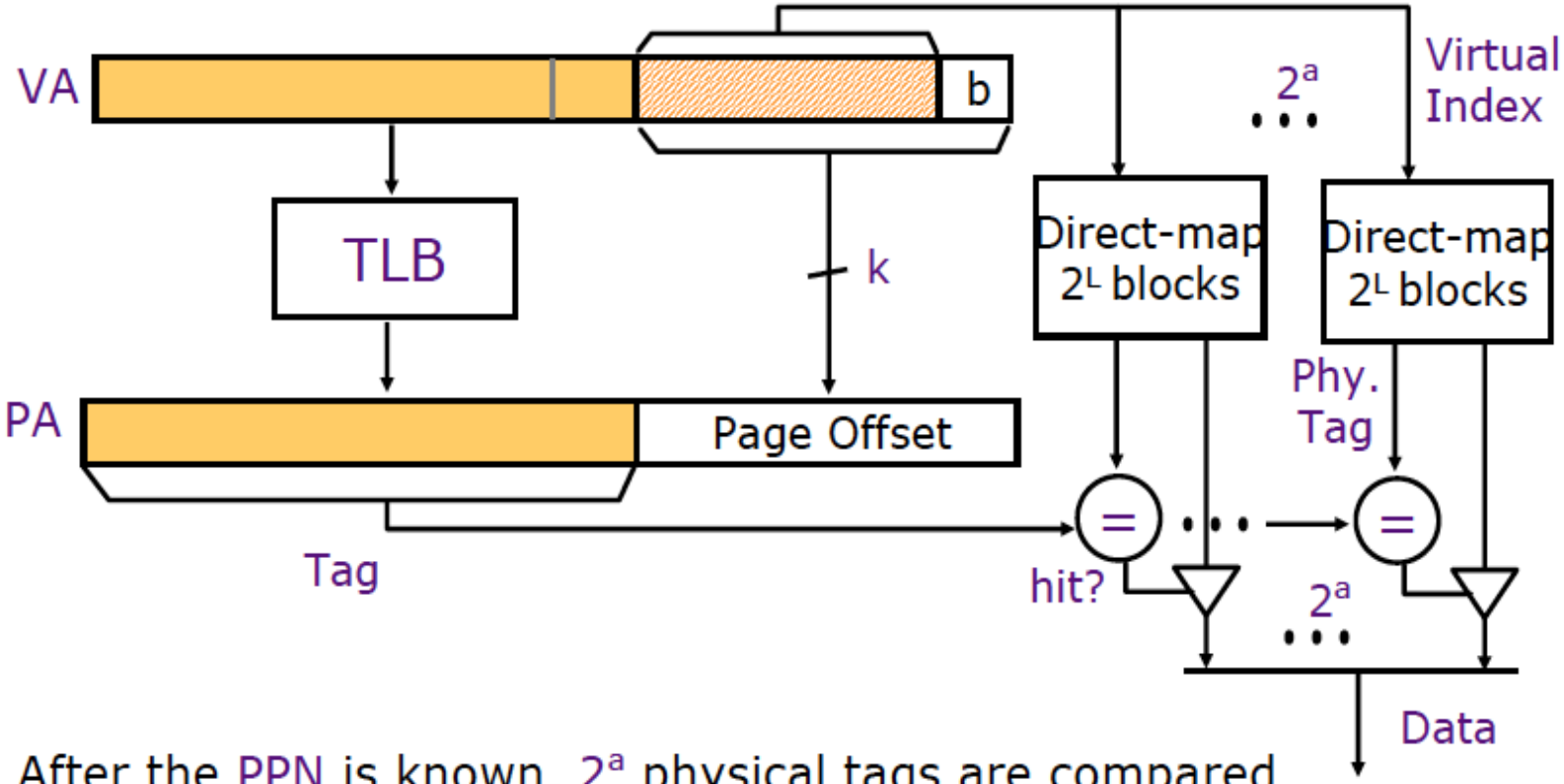
- General solution: disallow aliases to coexist in caches.
- OS solution for direct-mapped cache: VAs of shared pages must agree in cache index bits; this ensures that all VAs matching the same PAs conflict in the same direct-mapped cache (early SPARC approach).

Concurrent TLB/Cache Access



- Observation: page offset remains same between VA and PA
  - Use page offset bit to index into the cache
- Index L is available without consulting the TLB.
  - Cache and TLB accesses can begin simultaneously.
- Tag comparison is made after both accesses are completed.
  - Cases: L + b = k; L + b < K; L + b > k
- Overlap cache indexing time with TLB lookup time
  - But cache access cannot complete until address translation completes

Virtually -Indexed Physically - Tagged Caches



After the PPN is known, 2^a physical tags are compared

Typical cache hierarchy today

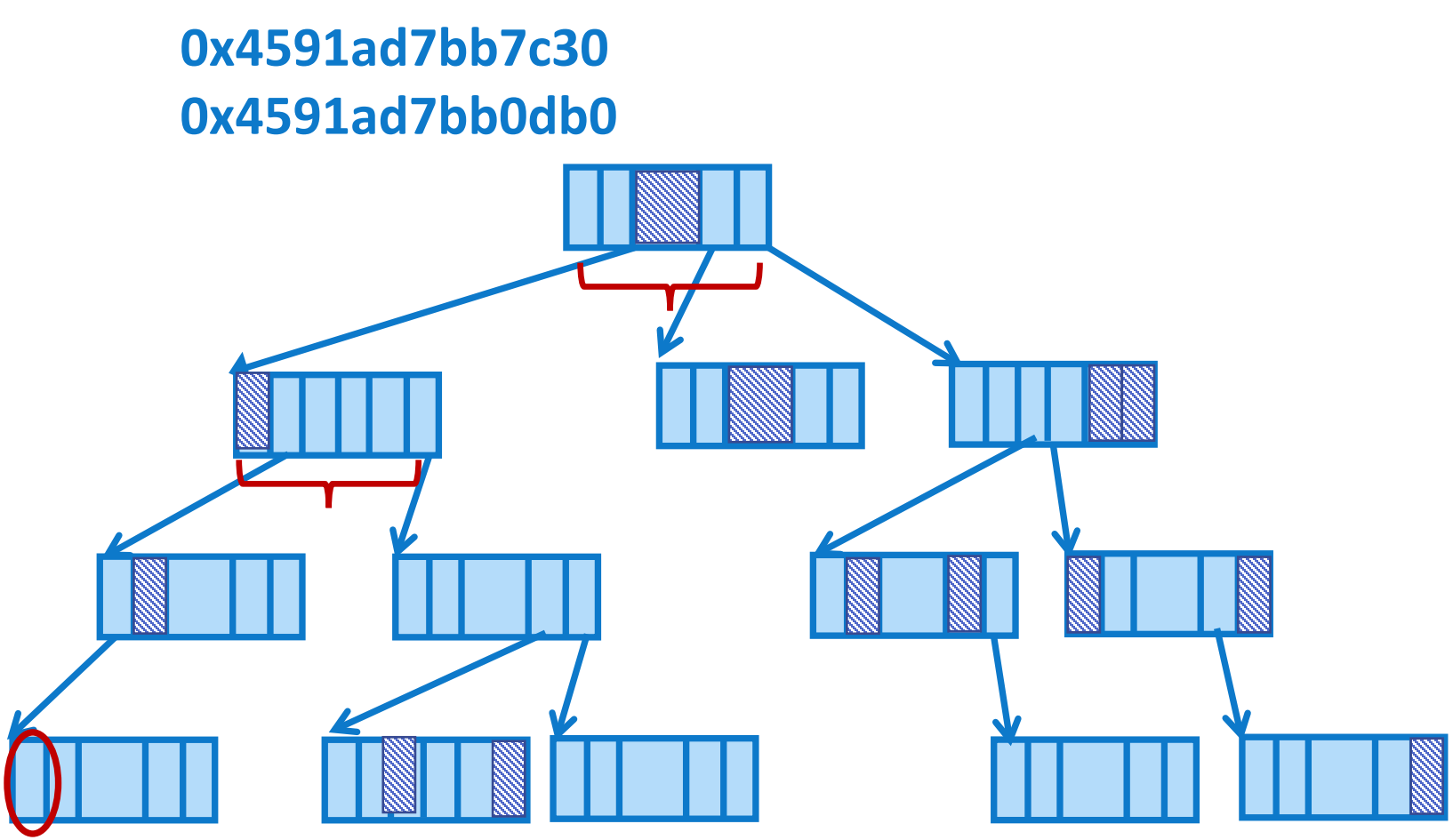
- L1 cache is **virtually-indexed and physically tagged**
  - Low latency access for the good case: L1 TLB hit and L1 cache hit
    - Helps to achieve higher clock frequency of a processor
- L2-L3 caches are physically indexed and physically tagged

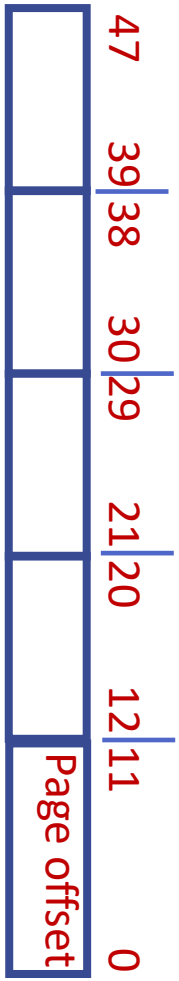
Making address translation faster

Short -circuiting page walking via page walk cache

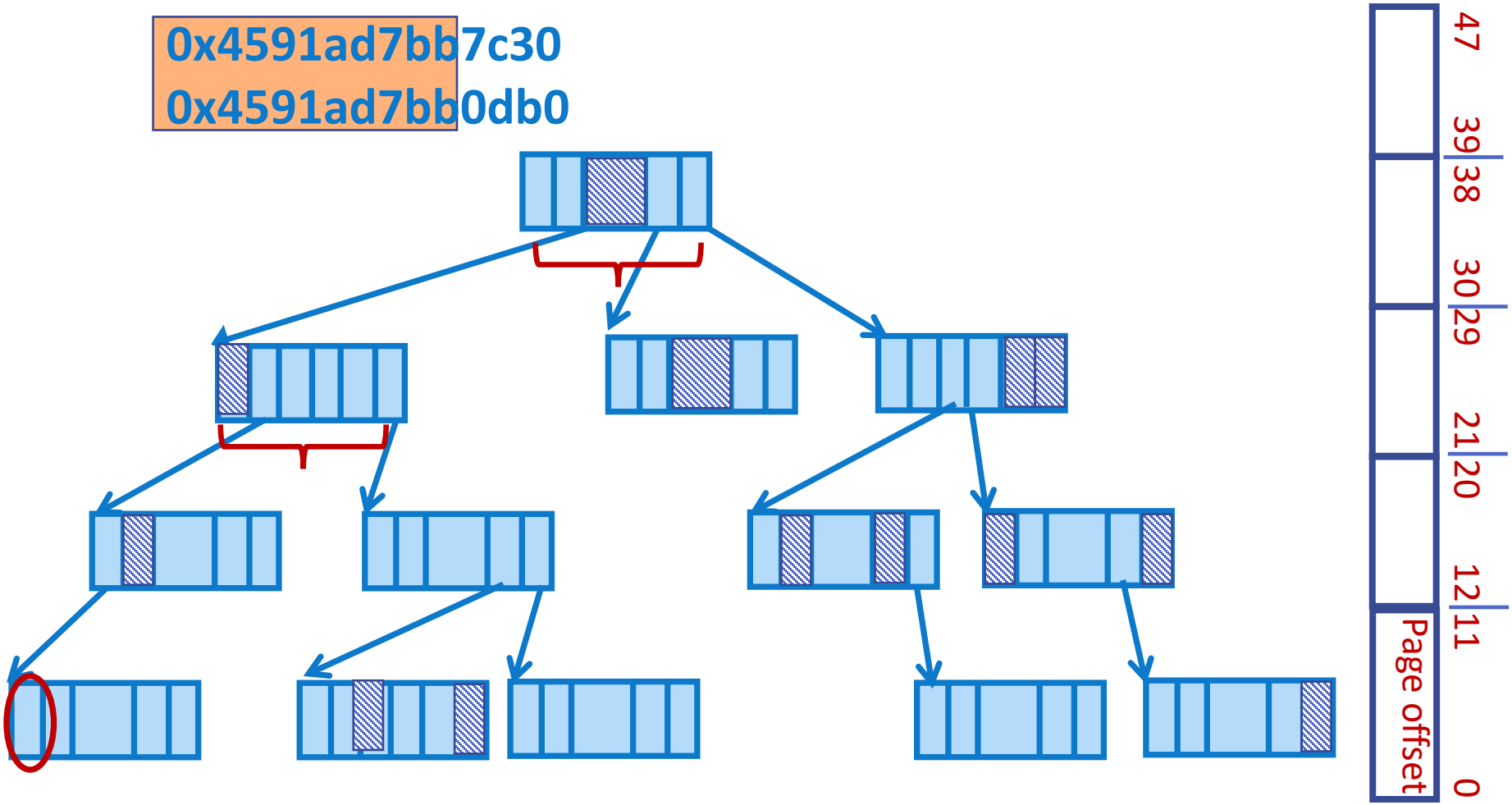
0x4591ad7bb7c30

0x4591ad7bb0db0

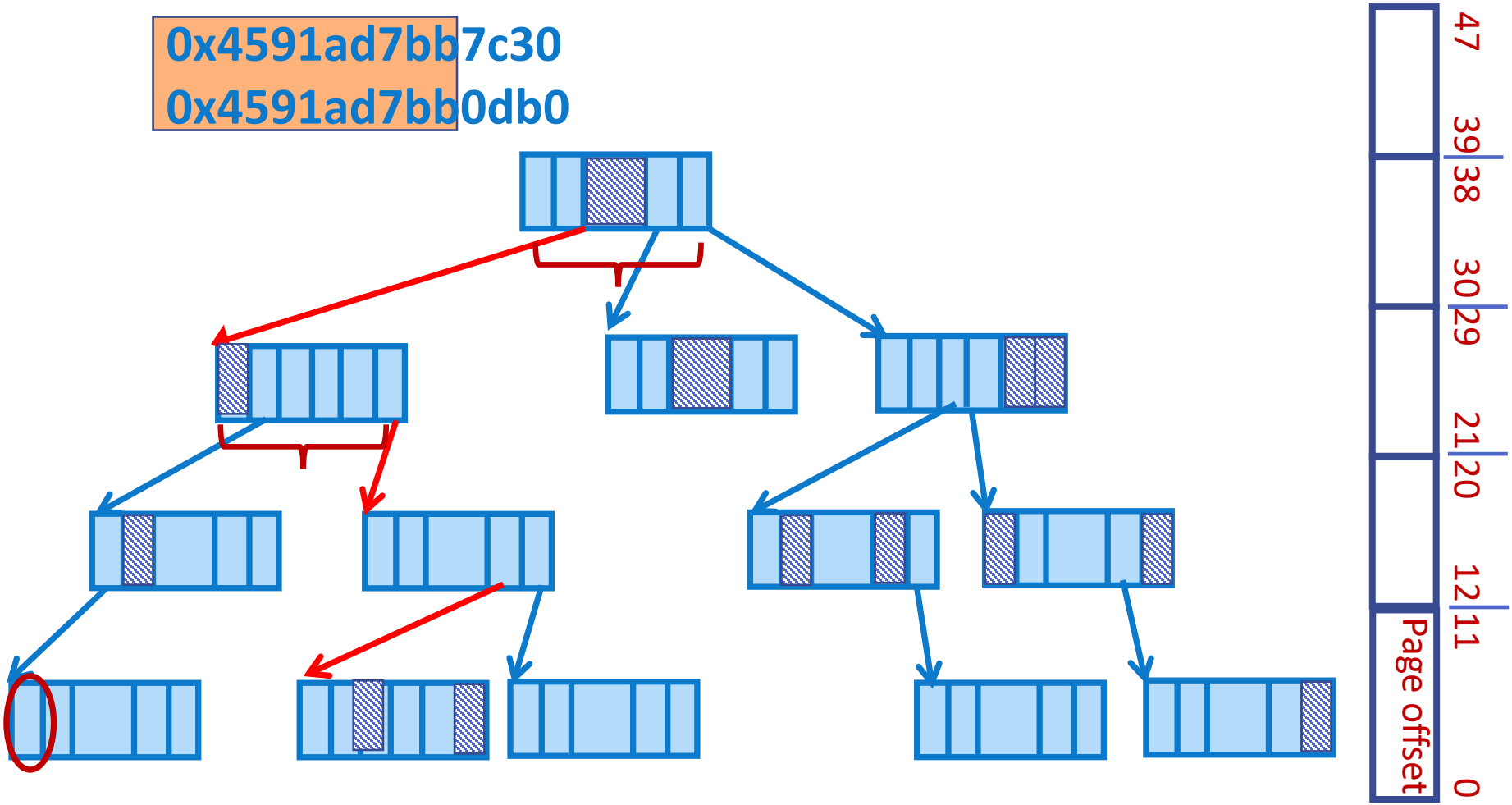




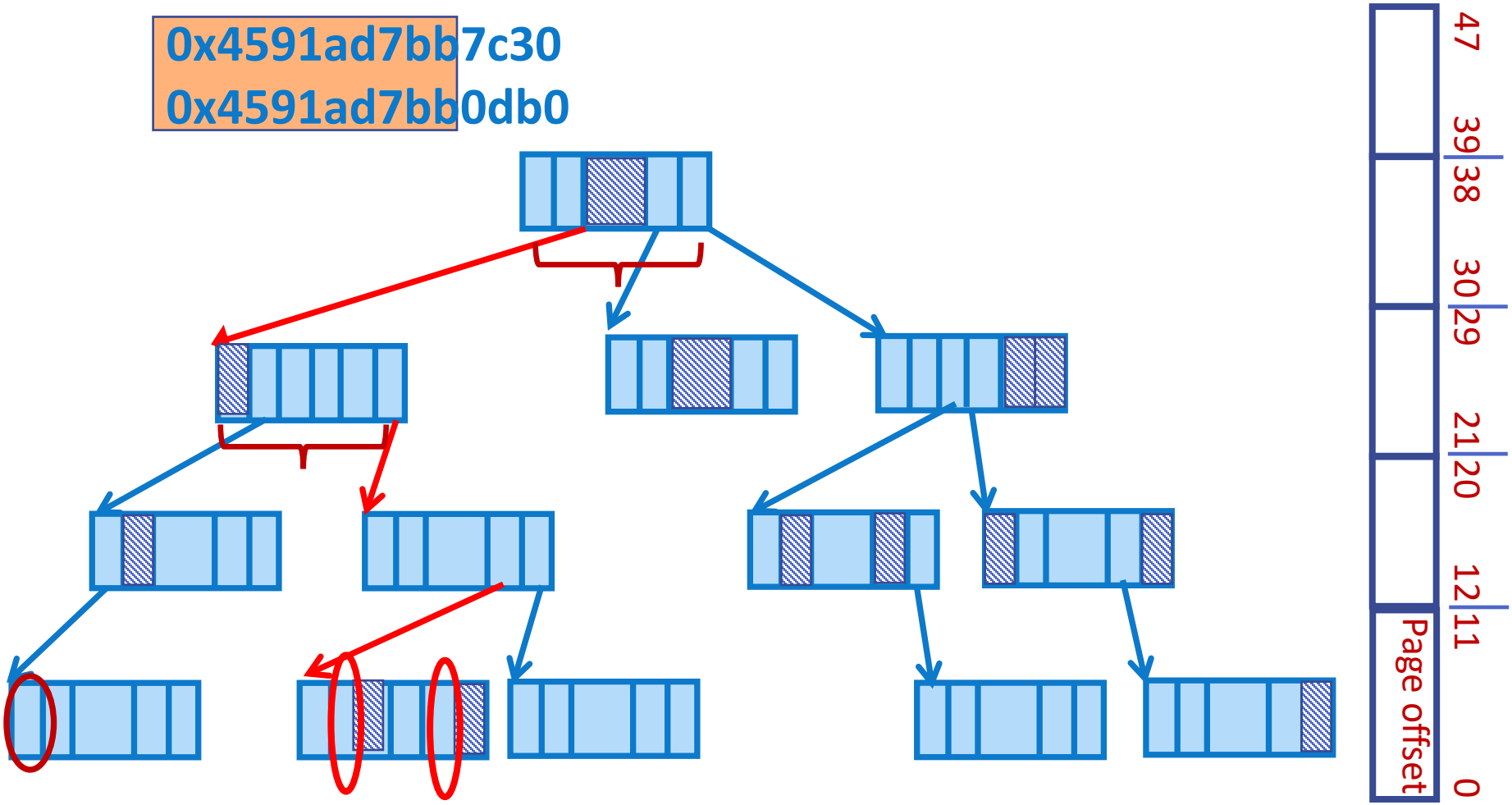
Short -circuiting page walking via page walk cache



Short -circuiting page walking via page walk cache



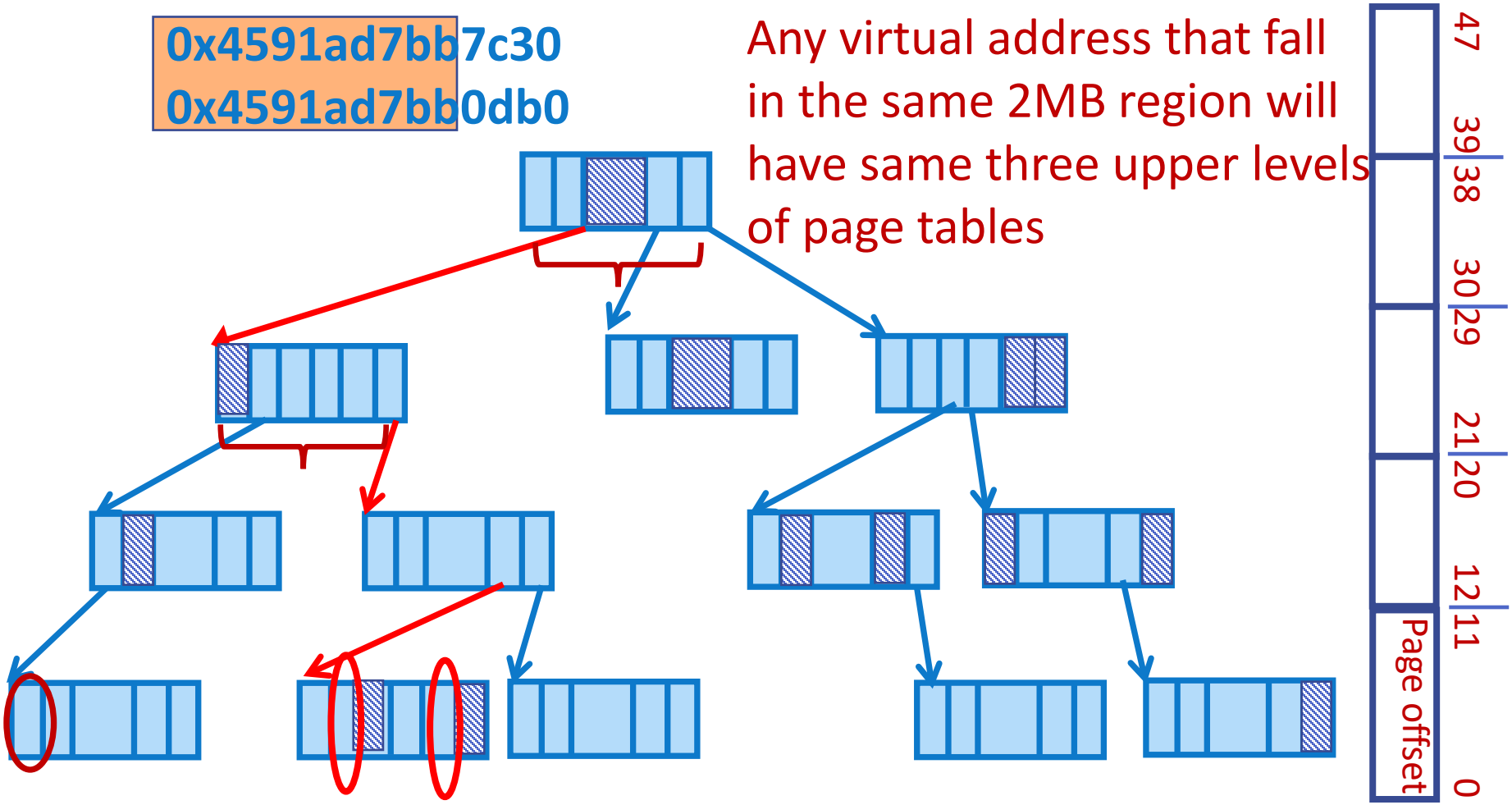
Short -circuiting page walking via page walk cache



Short -circuiting page walking via page walk cache

0x4591ad7bb7c30
0x4591ad7bb0db0

Any virtual address that fall in the same 2MB region will have same three upper levels of page tables



Page walk cache or partial translation cache

- Cache (physical) address of intermediate page table entries
- On a TLB miss lookup the page walk cache first
  - On a hit only one access to page table instead of 4
- The other three upper-level intermediate page table entries
  - A page walk can take one to four memory accesses depending upon which intermediate level hits or a is complete miss in the page walk cache

Superpages : Reducing TLB misses

- TLB miss are long latency operation
  - Address translation overhead == ~ TLB miss overhead
- TLB reach = # of TLB entries ✖ page size
  - Higher TLB reach → (Typically) Lower TLB miss rate

Superpages : Reducing TLB misses

- TLB miss are long latency operation
  - Address translation overhead == ~ TLB miss overhead
- TLB reach = # of TLB entries ✖ page size
  - Higher TLB reach → (Typically) Lower TLB miss rate
- How to increase TLB reach?
  - Increase TLB size.
    - Not always possible due to h/w overhead
  - Increase page size
    - Increases internal fragmentation

Superpages : Reducing TLB misses

- TLB miss are long latency operation
  - Address translation overhead == ~ TLB miss overhead
- TLB reach = # of TLB entries ✖ page size
  - Higher TLB reach → (Typically) Lower TLB miss rate
- How to increase TLB reach?
  - Increase TLB size.
    - Not always possible due to h/w overhead
  - Increase page size
    - Increases internal fragmentation
  - Have multiple page sizes
    - Larger page sizes beyond base page size (e.g., 4KB)

Superpages : Reducing TLB misses

- Larger page sizes called superpages
- Example page sizes in x86-64 (Intel, AMD) machines
  - 4KB (base page size), 2MB, 1GB (superpages)

Superpages : Reducing TLB misses

- Larger page sizes called superpages
- Example page sizes in x86-64 (Intel, AMD) machines
  - 4KB (base page size), 2MB, 1GB (superpages)
- **Challenge**:

Superpages : Reducing TLB misses

- Larger page sizes called superpages
- Example page sizes in x86-64 (Intel, AMD) machines
  - 4KB (base page size), 2MB, 1GB (superpages)
- **Challenge**:
  - Need contiguous physical memory – often scarce

Superpages : Reducing TLB misses

- Larger page sizes called superpages
- Example page sizes in x86-64 (Intel, AMD) machines
  - 4KB (base page size), 2MB, 1GB (superpages)
- **Challenge**:
  - Need contiguous physical memory – often scarce
  - Which page size to use for mapping a given virtual address?
    - Use of larger pages increase internal fragmentation → memory bloat
    - Using smaller page size can increase address translation overhead