Caches and Virtual Memory

Suvinay Subramanian
(adapted from prior 6.823 offerings)

Soul of a Computer Architect

- Abstractions
- Tradeoffs

Caches

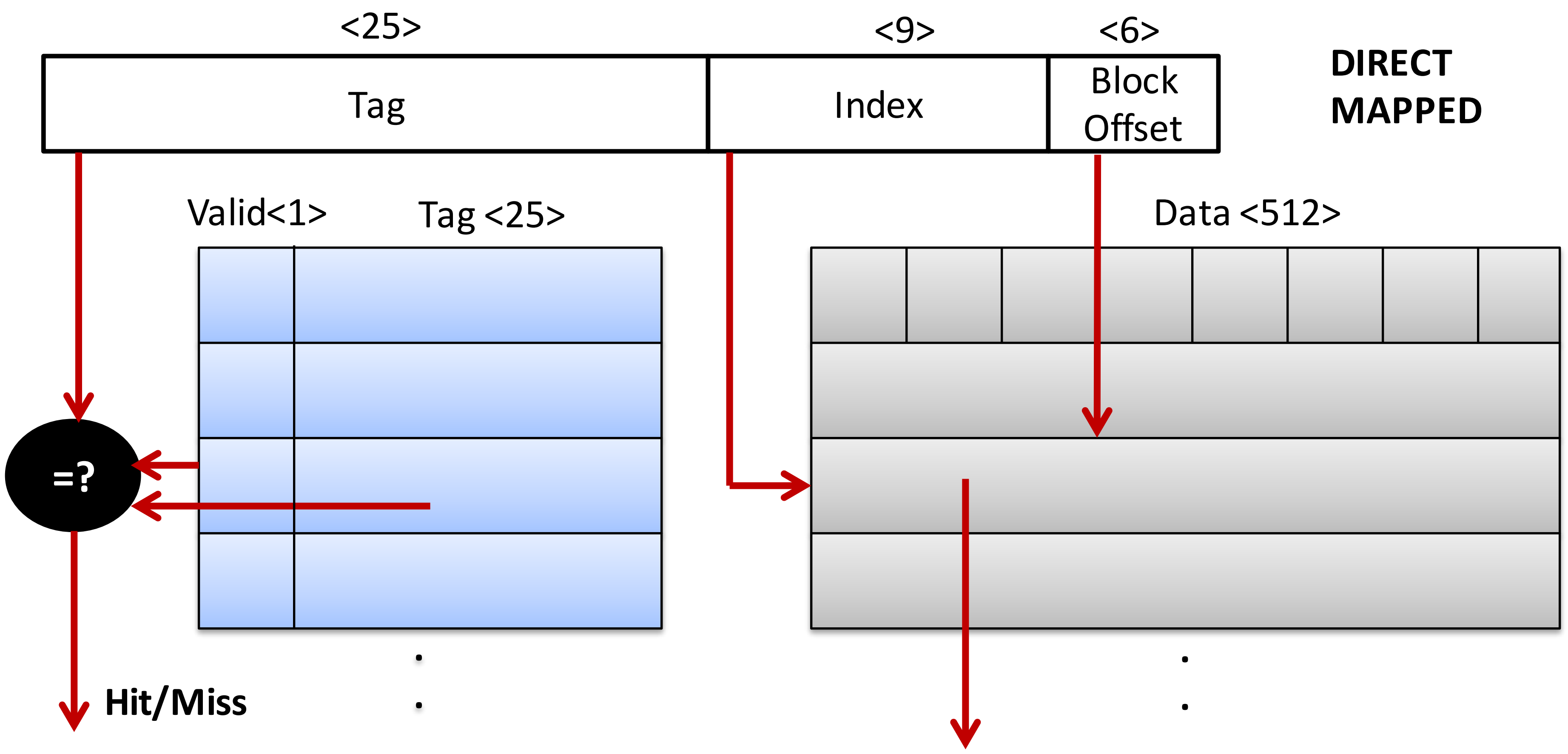
- Response to the processor—memory latency gap
- Basic Idea: Principle of locality
  - Spatial
  - Temporal

Caches: Basics

- Block / Line: Unit of storage in cache
- Data references:
  - Hit or miss

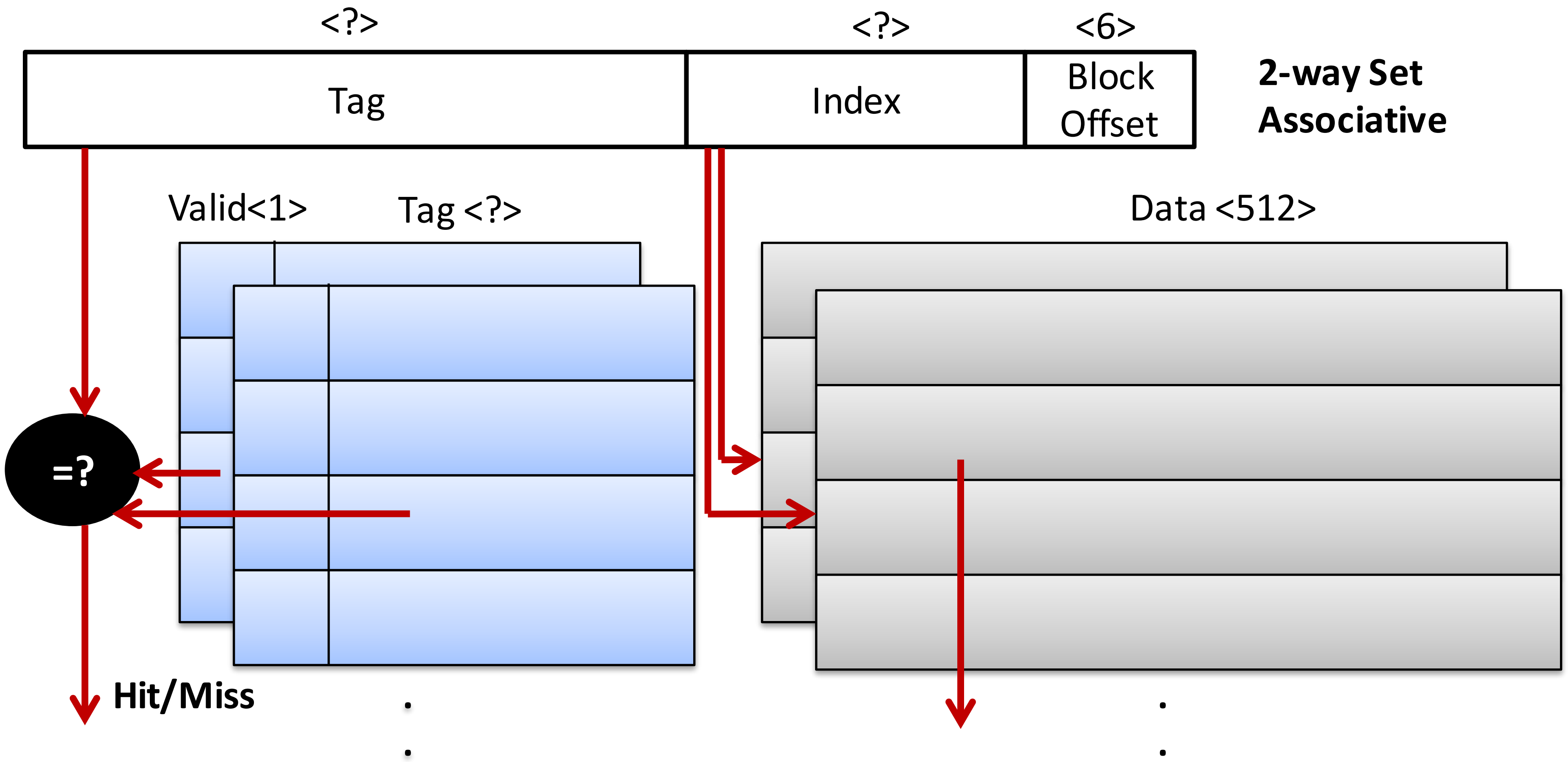
- Important cache design decisions:
  - Placement: Where, how to find/place block?
  - Replacement: How to remove line?
  - Others…

Direct Mapped Cache



Metadata? (2^9)*26b = 13kb **Cache Size? (2^9)*64B = 32kB**

Set Associative Cache



Index? 8

Tag? 26

Same Cache Size 32kB

Set Associative Cache: Decisions

- Each block has a “priority”
  How to determine / adjust priority?
- Insertion, Promotion, Eviction

- LRU Replacement Policy
  - Evict the least-recently used block
  - What information do you need?
    - In hardware, how many bits?
    - How does it affect access time for cache?

- Architects expend considerable effort optimizing cache design
  - Big impact on performance, power

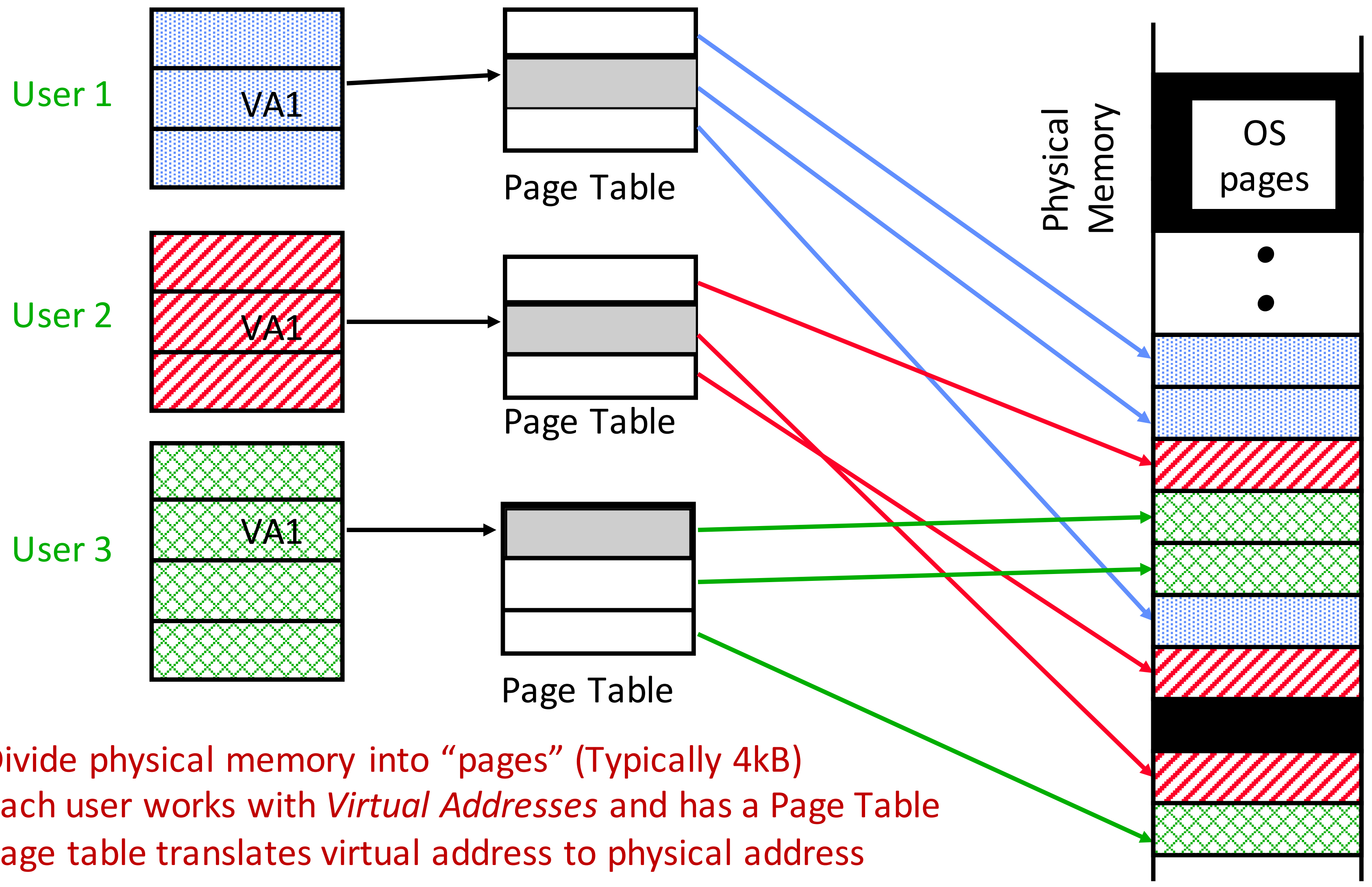
- Large design space
  - Cache size, block size, associativity, replacement policy, write-back/through, write-allocation etc.
  - Tradeoffs: Performance vs Power vs Area vs Complexity vs Cost

Virtual Memory

- Abstraction
  - Programmer sees virtual memory
  - Transparently managed
  - Can support larger virtual memory than physical memory
- Protection

Basic Idea: Indirection

Page-based Virtual Memory

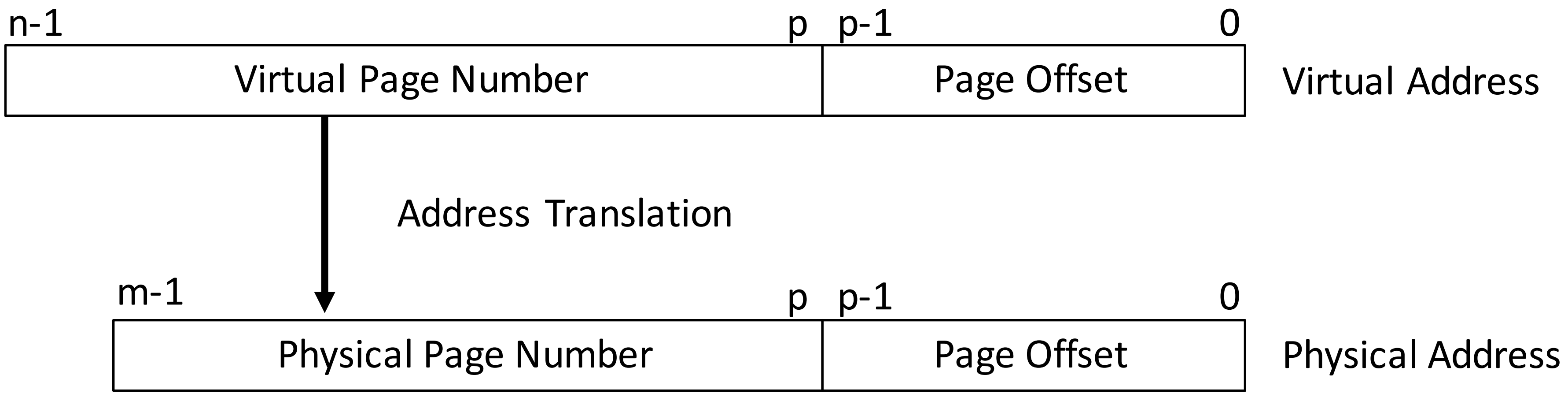


- Divide physical memory into "pages" (Typically 4kB)
- Each user works with *Virtual Addresses* and has a Page Table
- Page table translates virtual address to physical address

Address Translation

Parameters

- $P = 2^p$ = page size (bytes).
- $N = 2^n$ = Virtual-address limit
- $M = 2^m$ = Physical-address limit



Page offset bits do not change with translation

Page Table Issues

(A): How large is the page table? How do we store and access it?

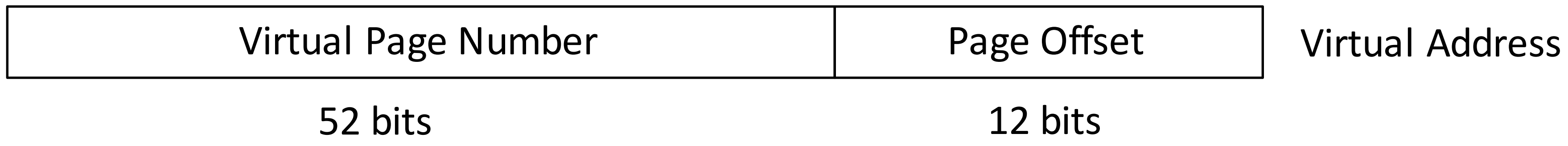
(B): How much time does it take for translation?

(C): When do we do the address translation?

Many others…

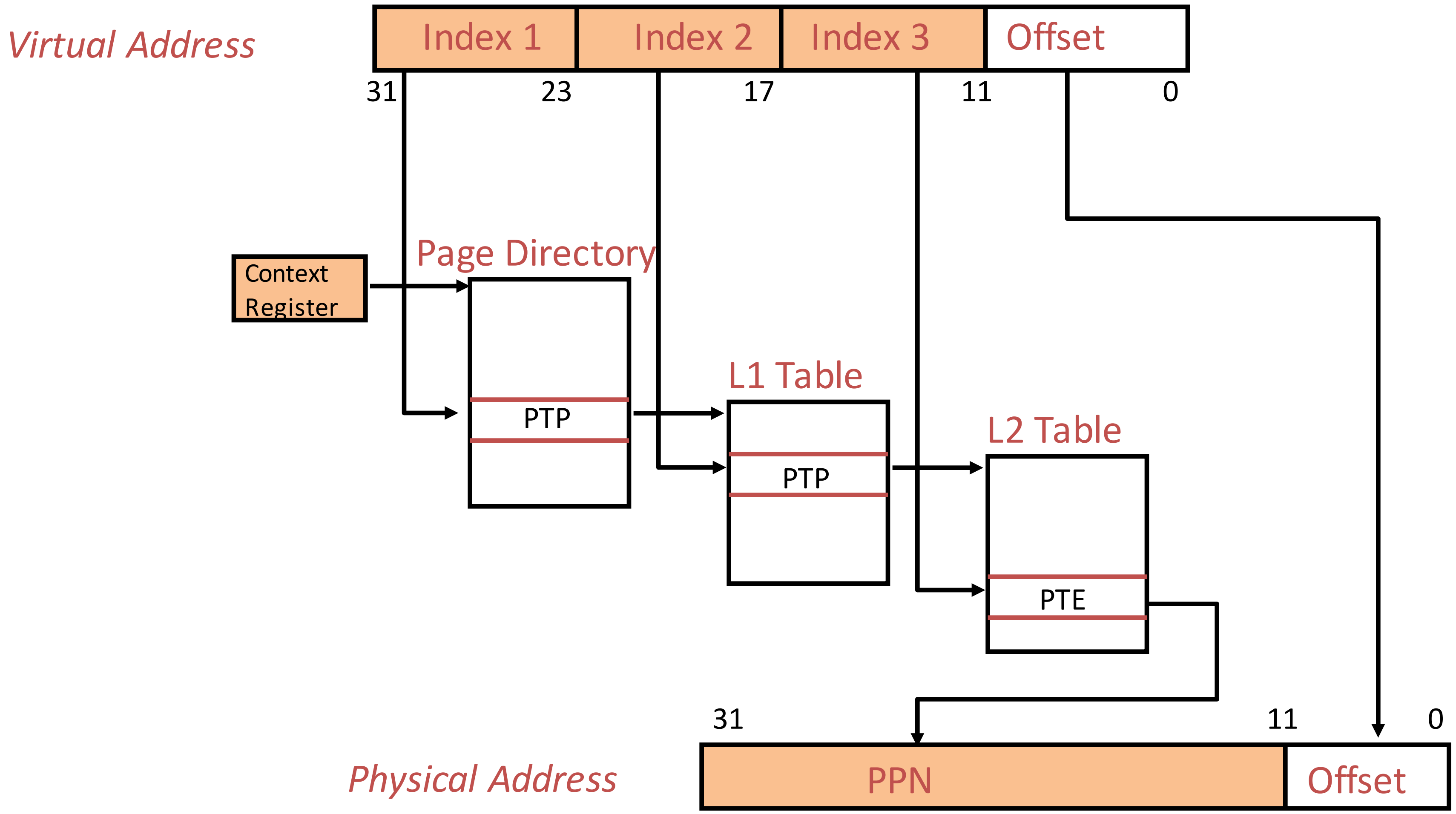
(A): Hierarchical Page Tables

- Issue (A): Page Table Size



How many entries in the page table? 2^{52}

(A): Hierarchical Page Tables



Page Table Issues

(A): How large is the page table? How do we store and access it?

(B): How much time does it take for translation?

(C): When do we do the address translation?

Many others…

(B): Speeding up the Common Case

- Page Tables stored in Memory
  - 1st Memory Access: obtain physical address
  - 2nd Memory Access: get data
- **Translation Lookaside Buffer (TLB)**
  - Fully-associative cache containing PPNs for VPNs
  - Is TLB miss same as a Page Fault?

    No.

    TLB miss => VPN to PPN mapping not found in TLB. $\nearrow$ Page walk

    Page Fault => Page not found in memory

Page Table Issues

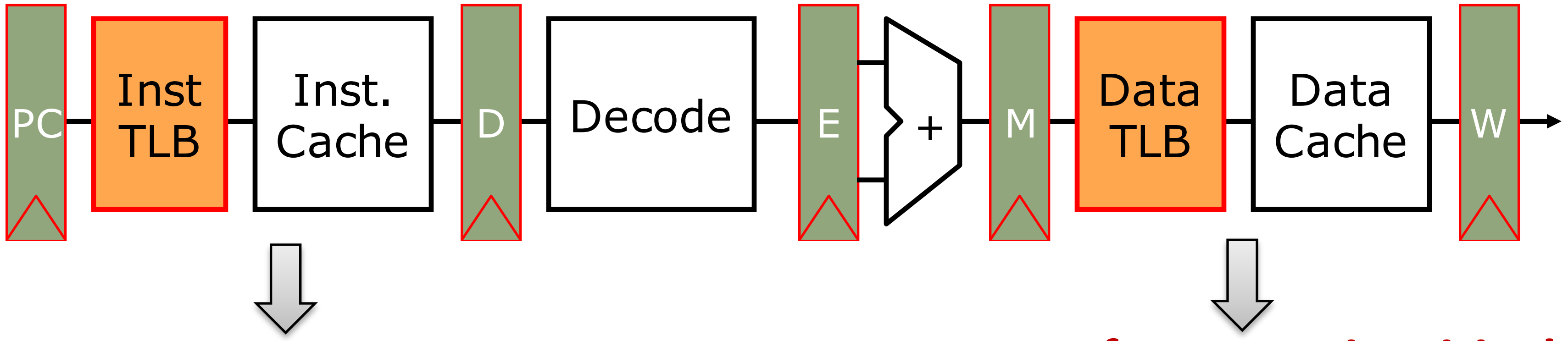
(A): How large is the page table? How do we store and access it?

(B): How much time does it take for translation?

(C): When do we do the address translation?

Many others…

(C): Address Translation in CPU Pipeline

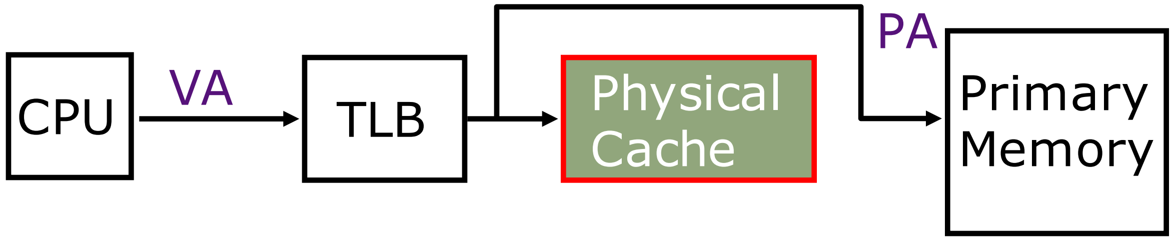


L1 Performance is critical. Needs to be 1 cycle!

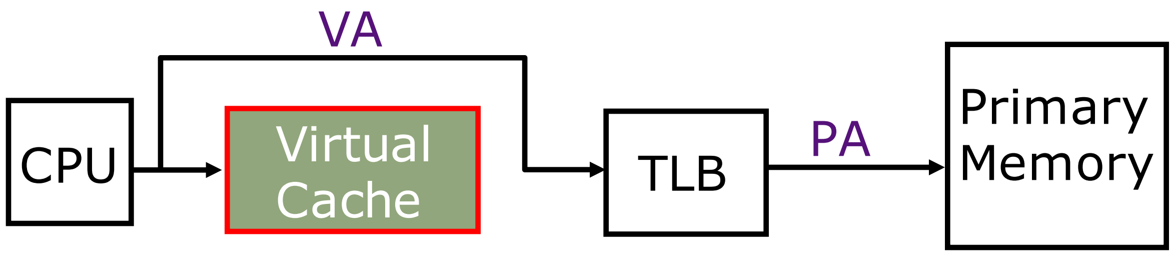
L1 Performance is critical. Needs to be 1 cycle!

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

(C): Physical or Virtual Address Caches?



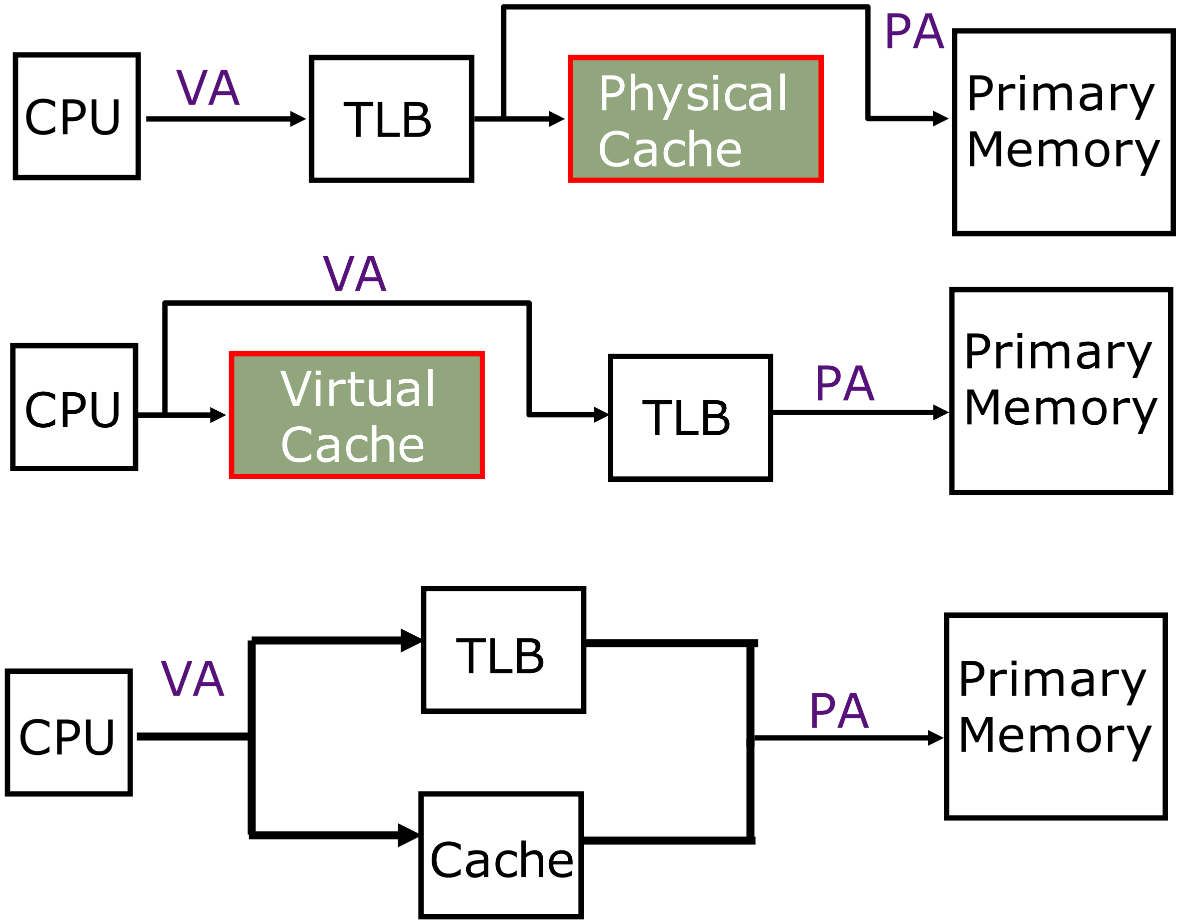
Alternative: place the cache before the TLB



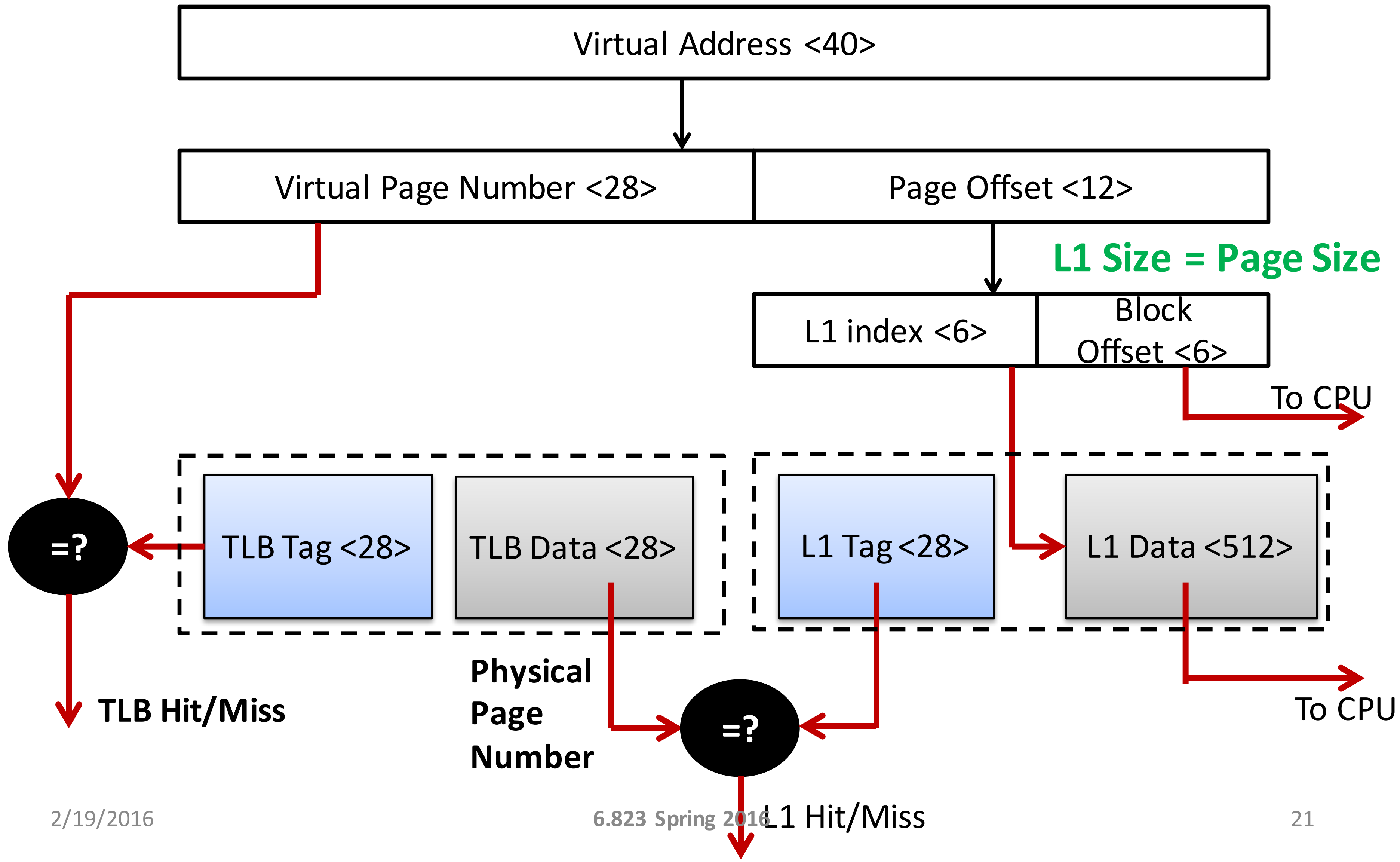
(StrongARM)

- 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 (-)

(C): Physical or Virtual Address Caches?



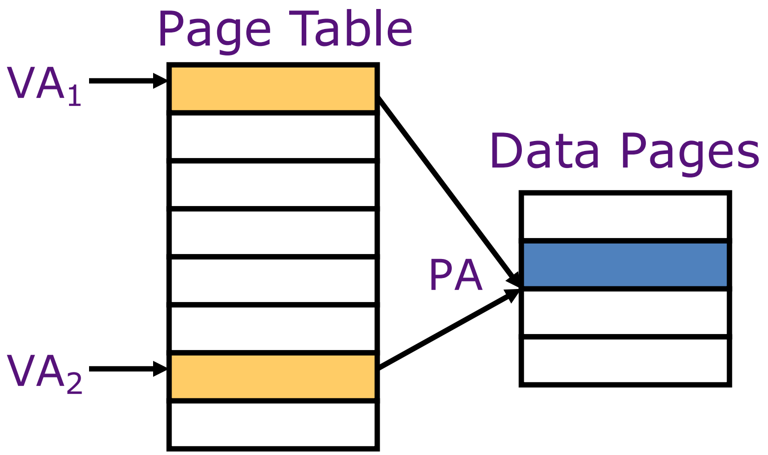
Concurrent Accesses to TLB and Cache



Concurrent Accesses to TLB and Cache

- Cache Size ≤ Page Size for direct-mapped
- Cache Size ≤ A*(Page Size) for associativity A

Aliasing in Virtual-Address 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!

How to avoid aliasing?

- Direct Mapped Caches
  - VAs of shared pages must agree in cache index bits
  - All VAs accessing same PA will map in same location in cache

- Exploit inclusive L2
  - L1 is virtually addressed (for speed), L2 is physically addressed (since address translation ready by then)
  - Suppose VA1 and VA2 both map to PA
  - VA1 is already in both L1 and L2
  - After VA2 is resolved to PA, collision will be detected in L2 (mapped to same index)
  - Purge VA1 from L1 and L2, and load VA2

Pages vs. Cache Blocks

- Pages == Cache Blocks/Lines
- Page Number == Cache Index + Tag
- Page Offset == Block Offset

Summary

- Caches, Virtual Memory: Two important components of modern computer systems

- Principles: Locality, Indirection

- Interaction between cache and VM
  - Virtual vs Physical Cache vs Combination
  - Aliasing