Virtual Memory

Slides courtesy of Mary Jane Irwin (www.cse.psu.edu/~mji)

www.cse.psu.edu/~cg431

Modifications by Nathan Sprague

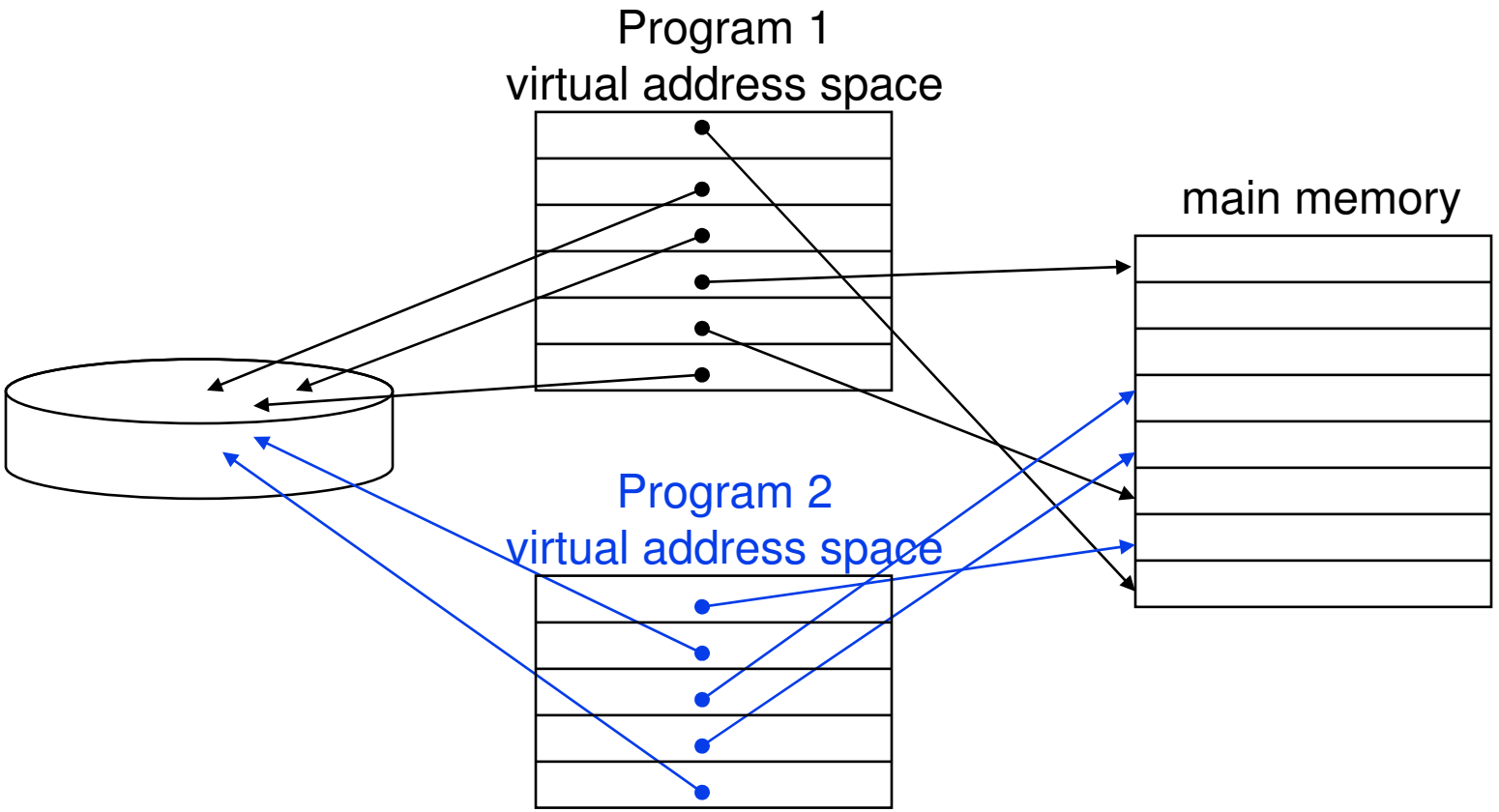
[Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005]

Virtual Memory

- Use main memory as a “cache” for secondary memory
  - Allows efficient and safe sharing of memory among multiple programs
  - Provides the ability to easily run programs larger than the size of physical memory
  - Simplifies loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)
- What makes it work? – again the Principle of Locality
  - A program is likely to access a relatively small portion of its address space during any period of time
- Each program is compiled into its own address space – a “virtual” address space
  - During run-time each virtual address must be translated to a physical address (an address in main memory)

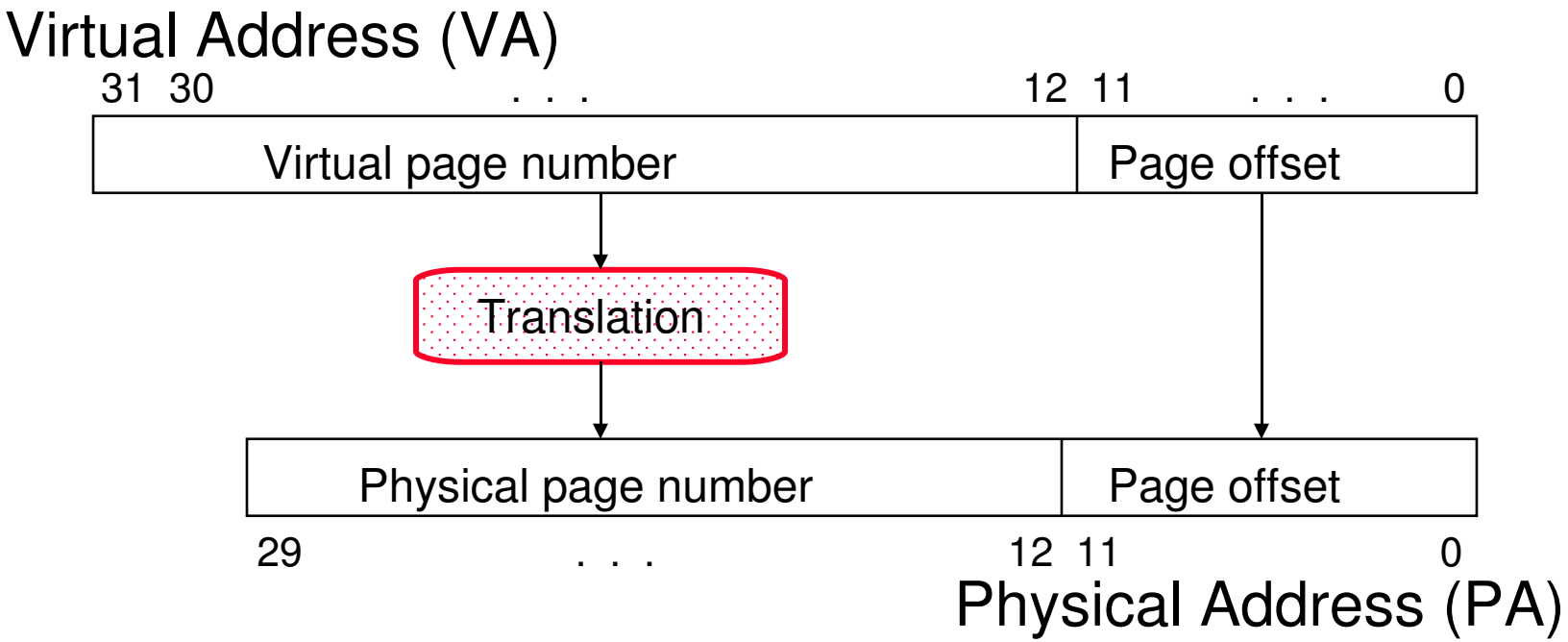
Two Programs Sharing Physical Memory

- A program's address space is divided into pages.
  - The starting location of each page (either in main memory or in secondary memory) is contained in the program's page table



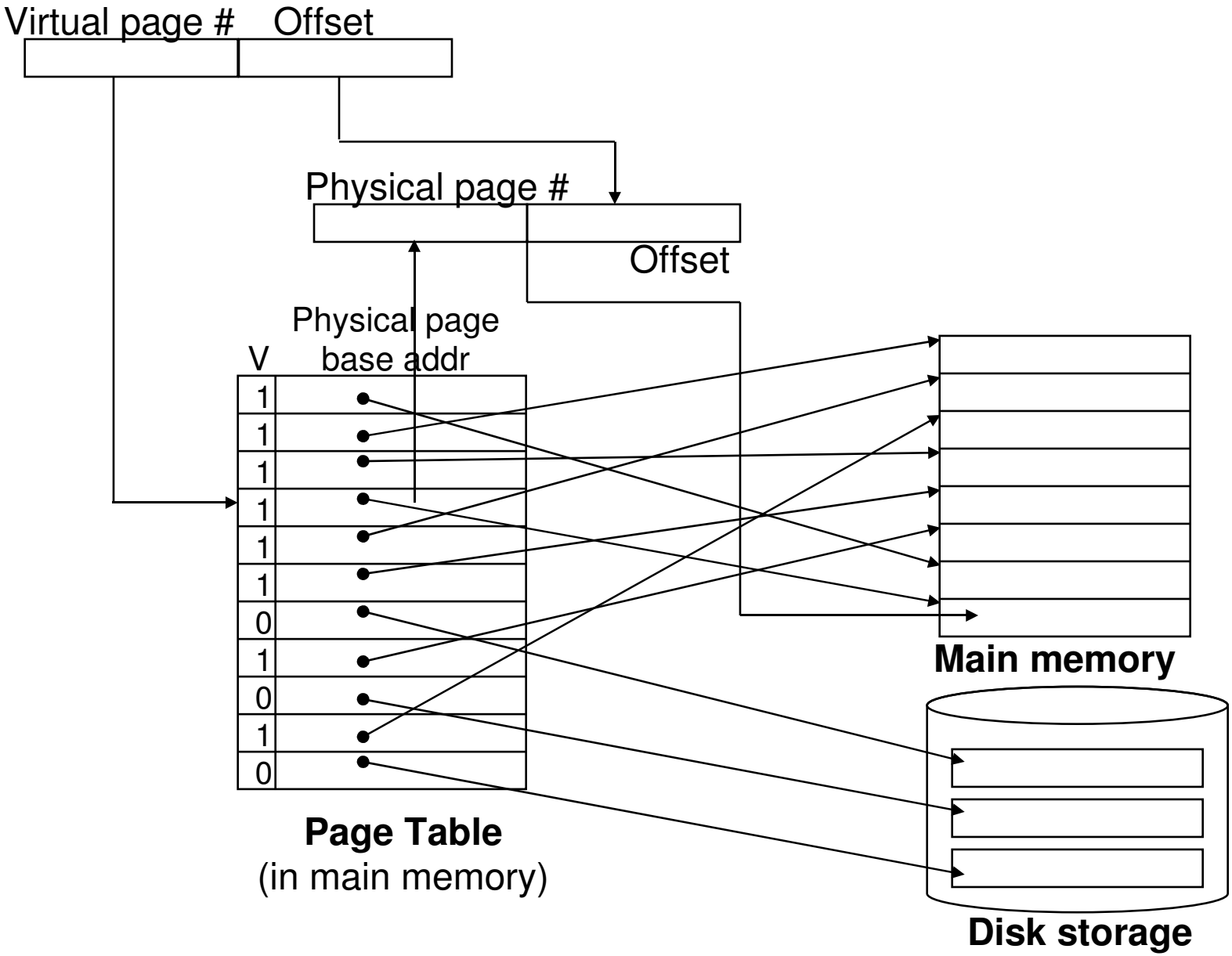
Address Translation

- A virtual address is translated to a physical address by a combination of hardware and software



- So each memory request *first* requires an address translation from the virtual space to the physical space
  - A virtual memory miss (i.e., when the page is not in physical memory) is called a page fault

Address Translation Mechanisms

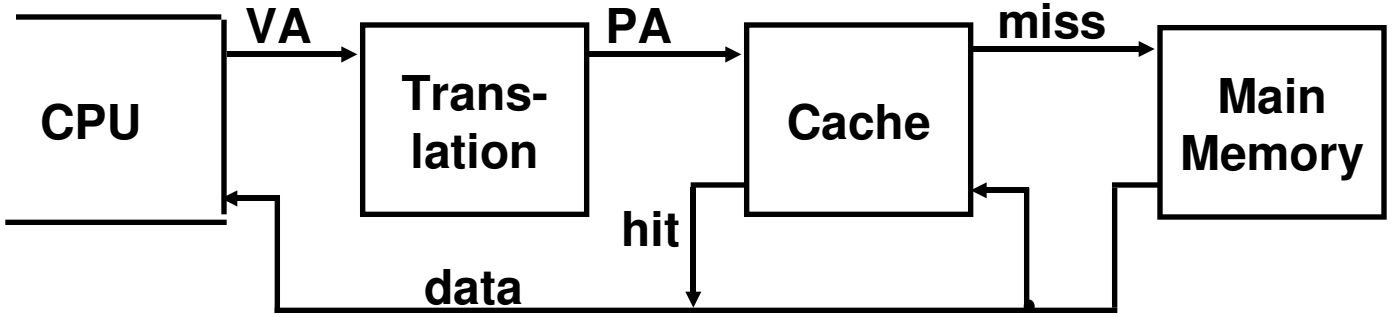


Page Table Organization

- The page table is fully associative
- OS is responsible for the page replacement policy
- We can afford a complex page replacement policy, because the cost for a page fault is SOOO high
- Do you foresee any difficulties with the page table system we've seen so far?

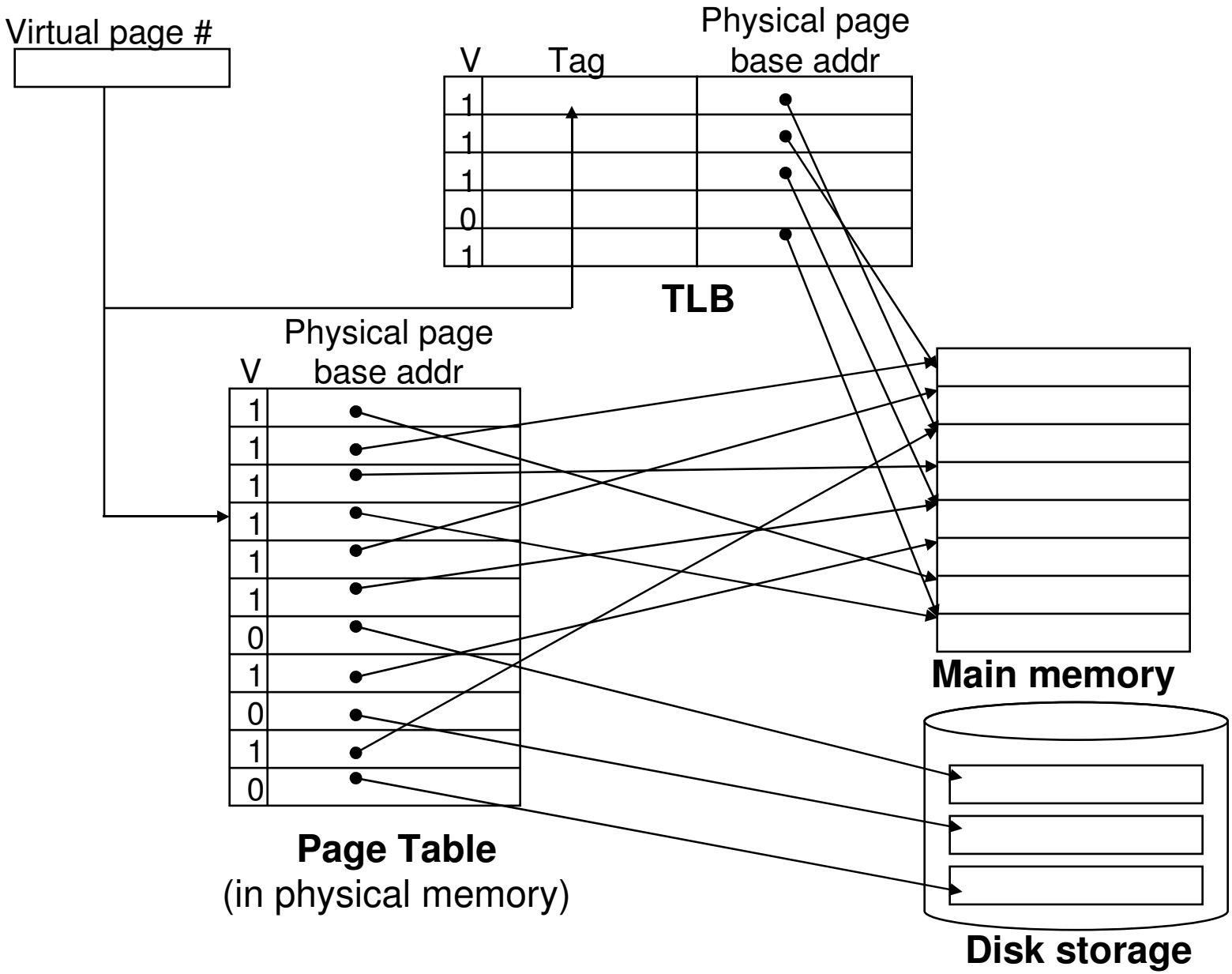
Virtual Addressing with a Cache

- Thus it takes an *extra* memory access to translate a VA to a PA



- This makes memory accesses very expensive
- The hardware fix is to use a Translation Lookaside Buffer (TLB) – a small cache that keeps track of recently used address mappings

Making Address Translation Fast

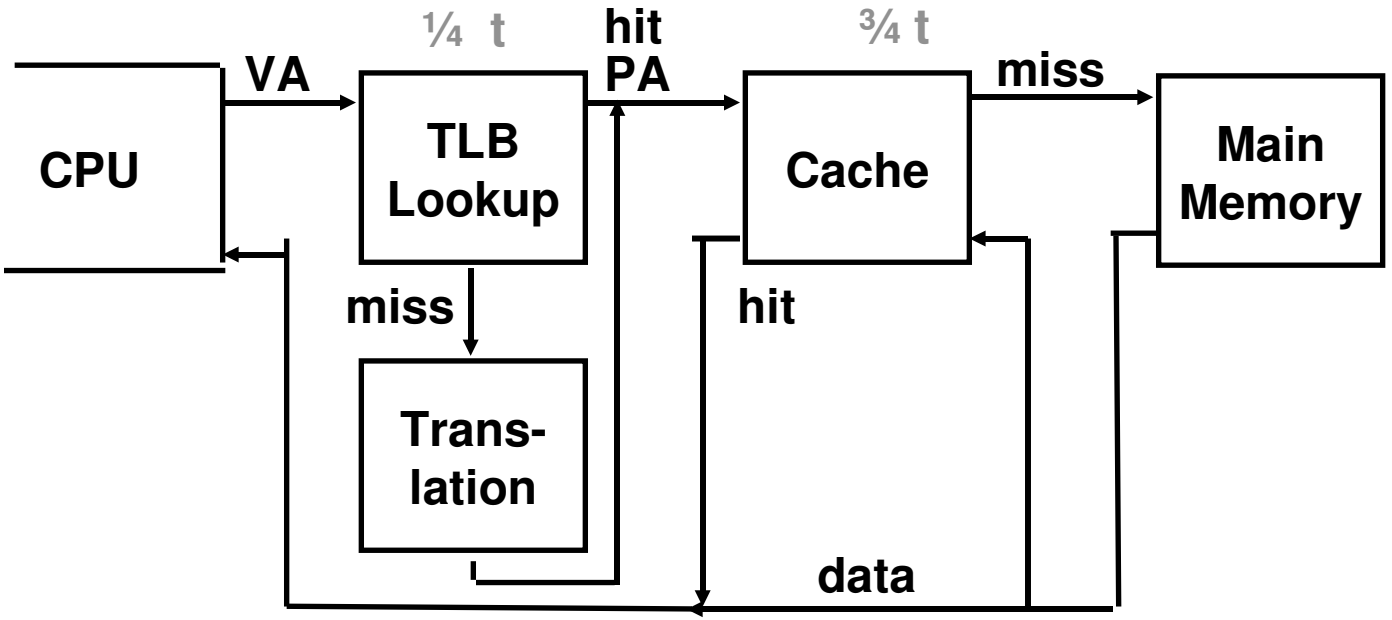


Translation Lookaside Buffers (TLBs)

- TLB can be organized as fully associative, set associative, or direct mapped

- TLB access time is typically smaller than cache access time (because TLBs are much smaller than caches)
  - TLBs are typically not more than 128 to 256 entries even on high end machines

A TLB in the Memory Hierarchy



- A TLB miss
  - If the page is in main memory, then the TLB miss can be handled by loading the translation information from the page table into the TLB
    - Takes 10's of cycles to find and load the translation info into the TLB
  - If the page is not in main memory, then it's a true page fault
    - Takes 1,000,000's of cycles to service a page fault
- TLB misses are much more frequent than true page faults

TLB Event Combinations

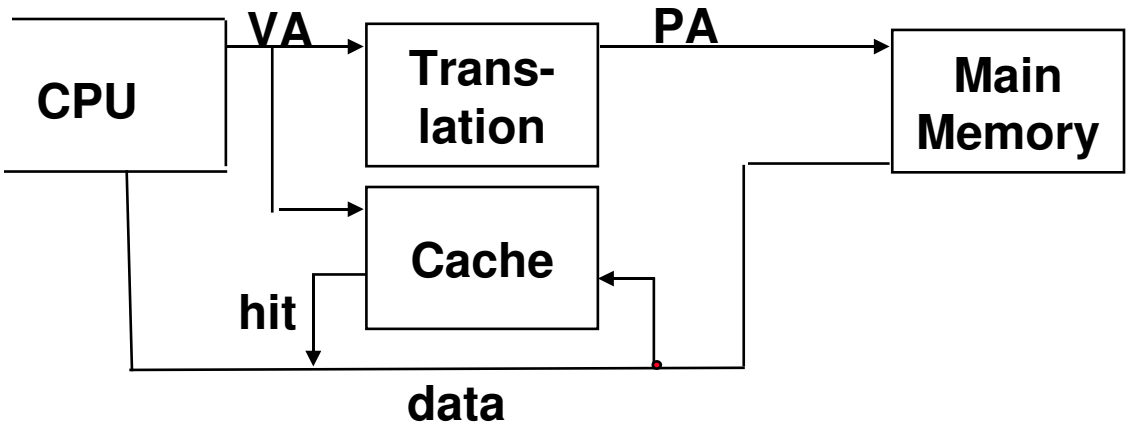
TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	
Hit	Hit	Miss	
Miss	Hit	Hit	
Miss	Hit	Miss	
Miss	Miss	Miss	
Hit	Miss	Miss/ Hit	
Miss	Miss	Hit	

TLB Event Combinations

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes – what we want!
Hit	Hit	Miss	Yes – although the page table is not checked if the TLB hits
Miss	Hit	Hit	Yes – TLB miss, PA in page table
Miss	Hit	Miss	Yes – TLB miss, PA in page table, but data not in cache
Miss	Miss	Miss	Yes – page fault
Hit	Miss	Miss/ Hit	Impossible – TLB translation not possible if page is not present in memory
Miss	Miss	Hit	Impossible – data not allowed in cache if page is not in memory

Why Not a Virtually Addressed Cache?

- A virtually addressed cache would only require address translation on cache misses



but

- Two different virtual addresses can map to the same physical address (when processes are sharing data), i.e., two different cache entries hold data for the same physical address – synonyms
  - Must update all cache entries with the same physical address or the memory becomes inconsistent

Summary

- The Principle of Locality:
  - Program likely to access a relatively small portion of the address space at any instant of time.
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- Caches, TLBs, Virtual Memory all understood by examining how they deal with the four questions
  1. Where can block be placed?
  2. How is block found?
  3. What block is replaced on miss?
  4. How are writes handled?
- Page tables map virtual address to physical address
  - TLBs are important for fast translation

Understanding The Behavior of Memory

- Three types of cache misses:
  - Compulsory – the fist time an item is accessed
  - Capacity – misses that result from a full cache
  - Conflict – misses that result from two items being mapped to the same location.