CS 352 – Lecture 17

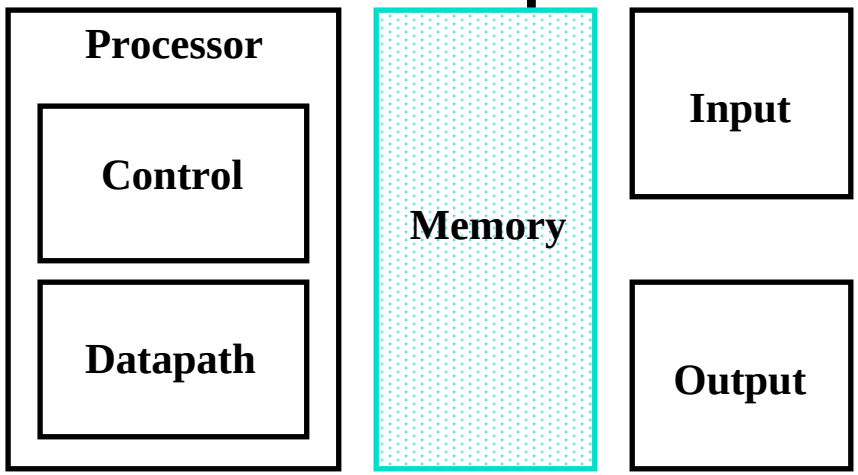
Caches and Virtual Memory Review

Today

- **Last time**
  - **Virtual Memory II – page tables and TLB's**
- **This time**
  - **Caching and virtual memory review**
- **Administrivia**
  - **Homework #5 is online, and due on April 6.**
  - **Final project will be cache simulator, in either Java or C/C++**

The Big Picture: Where are We Now?

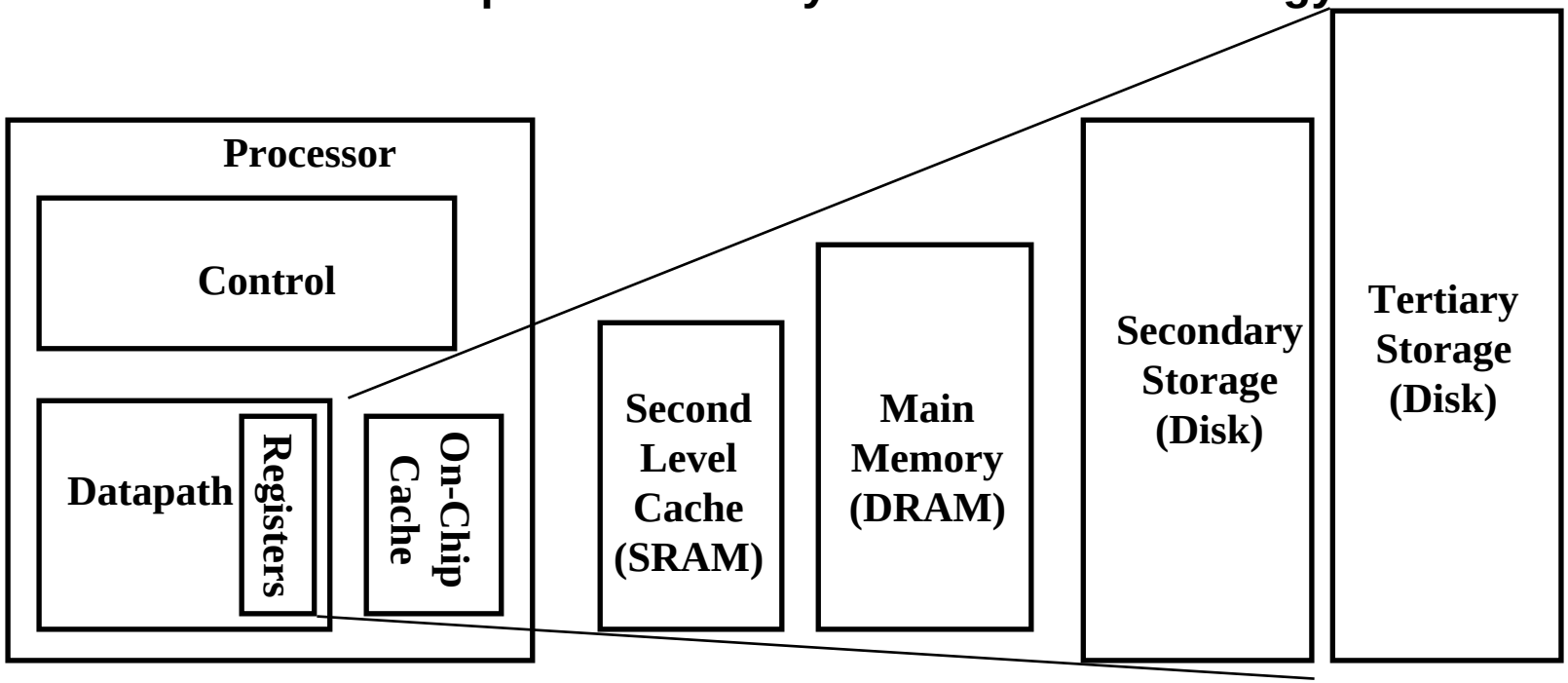
° The Five Classic Components of a Computer



Recap: Memory Hierarchy of a Modern Computer System

° **By taking advantage of the principle of locality:**

- **Present the user with as much memory as is available in the cheapest technology.**
- **Provide access at the speed offered by the fastest technology.**



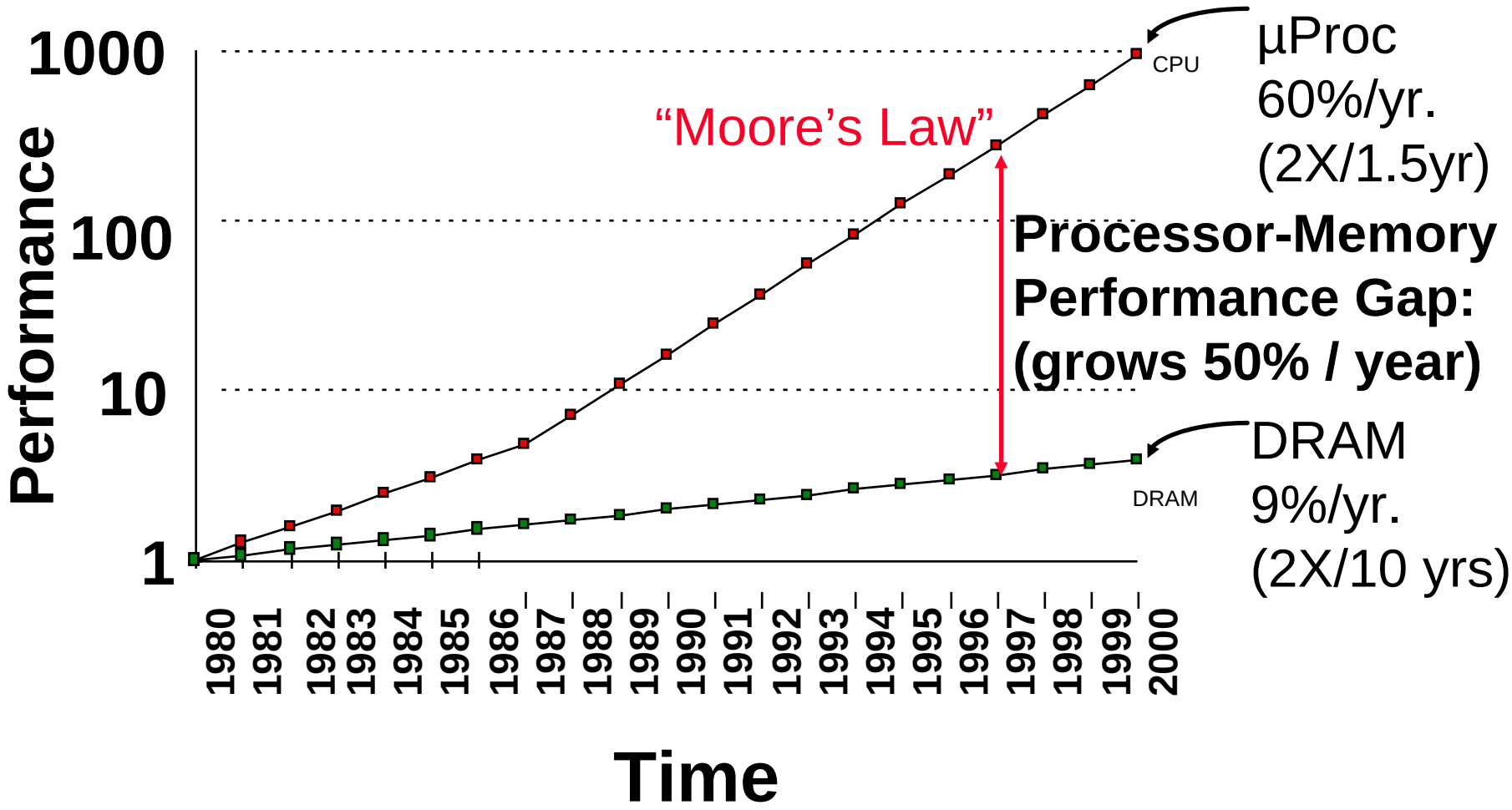
Speed (ns):	1s	10s	100s	10,000,000s (10s ms)	10,000,000,000s (10s sec)
Size (bytes):	100s	Ks	Ms	Gs	Ts

Goals of memory system design

- **Provide illusion of:**
  - **A private memory for each process**
  - **Referenced using a single “namespace” – virtual addresses**
  - **Nearly infinite in size**
  - **Ability to forbid certain operations (e.g. memory writes) to certain addresses**
- **Make it as fast as possible**
  - **Ideally, average access time is nearly that of fastest (but smallest) memory**
  - **Assume that programs exhibit spatial and temporal locality**

Recap: Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)

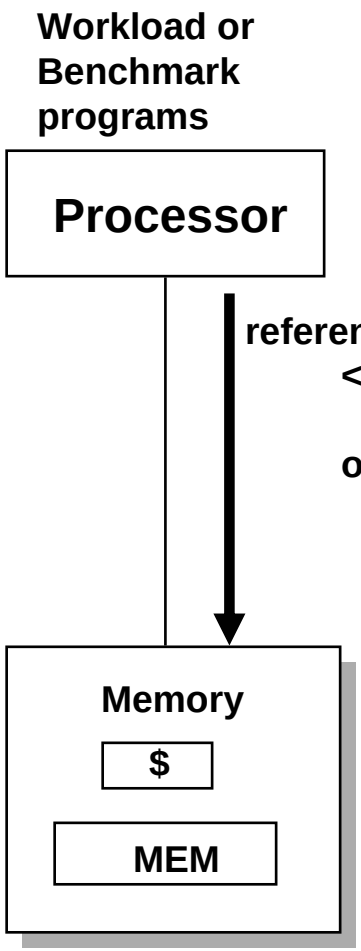


WRM, from DAP

Recap:

- **Two Different Types of Locality:**
  - **Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.**
  - **Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.**
- **By taking advantage of the principle of locality:**
  - **Present the user with as much memory as is available in the cheapest technology.**
  - **Provide access at the speed offered by the fastest technology.**
- **DRAM is slow but cheap and dense:**
  - **Good choice for presenting the user with a BIG memory system**
- **SRAM is fast but expensive and not very dense:**
  - **Good choice for providing the user FAST access time.**

The Art of Memory System Design

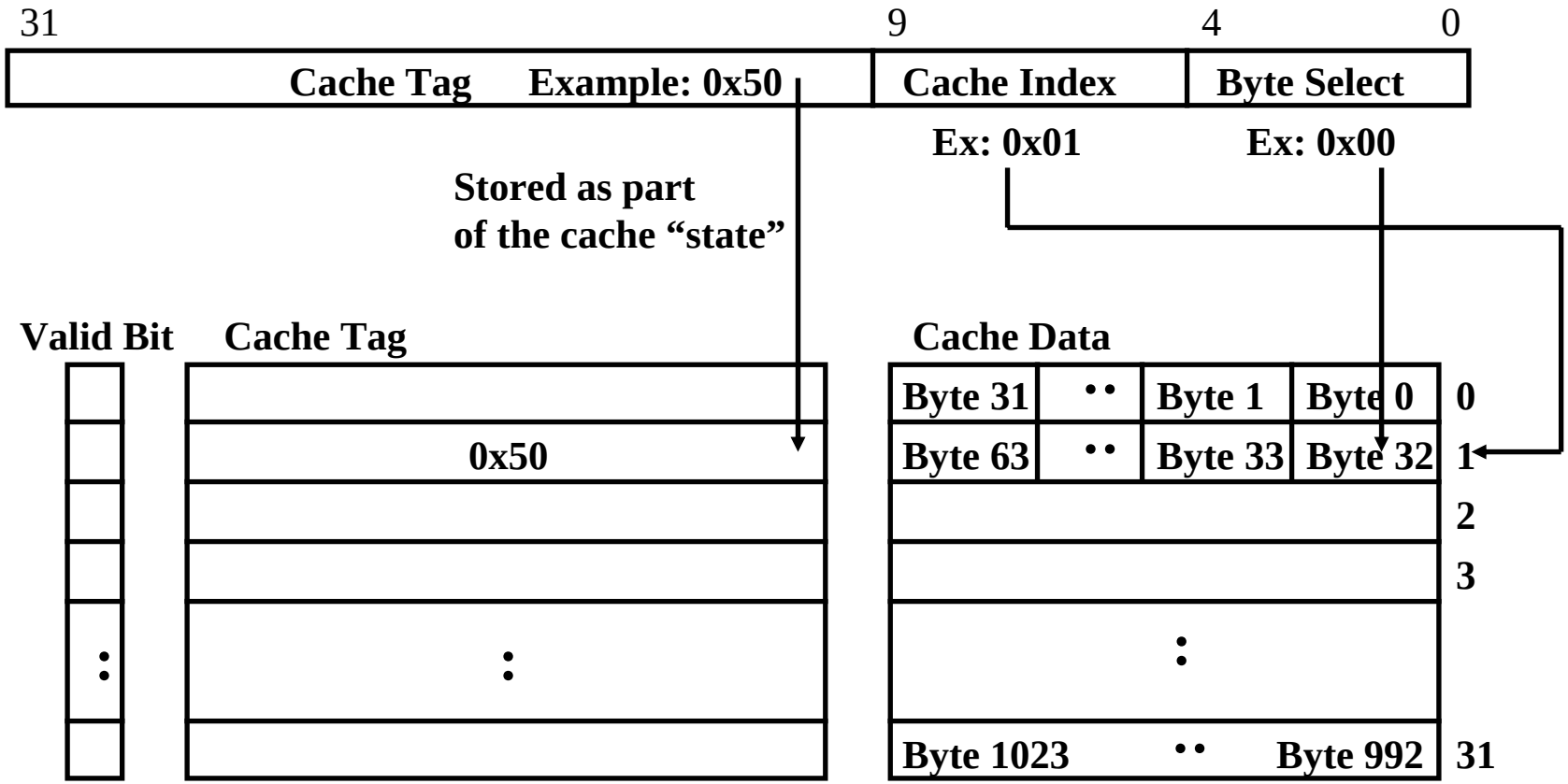


Optimize the memory system organization to minimize the average memory access time for typical workloads

Example: 1 KB Direct Mapped Cache with 32 B Blocks

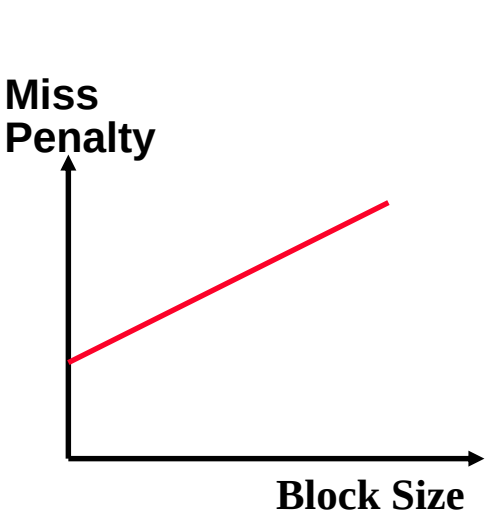
° For a 2 ** N byte cache:

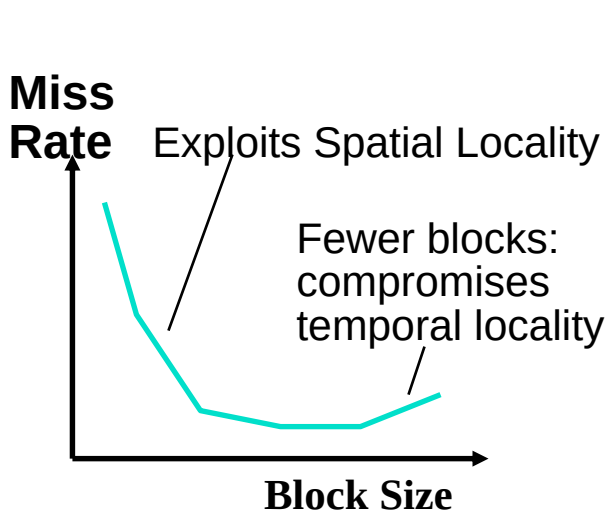
- The uppermost (32 - N) bits are always the Cache Tag
- The lowest M bits are the Byte Select (Block Size = 2 ** M)

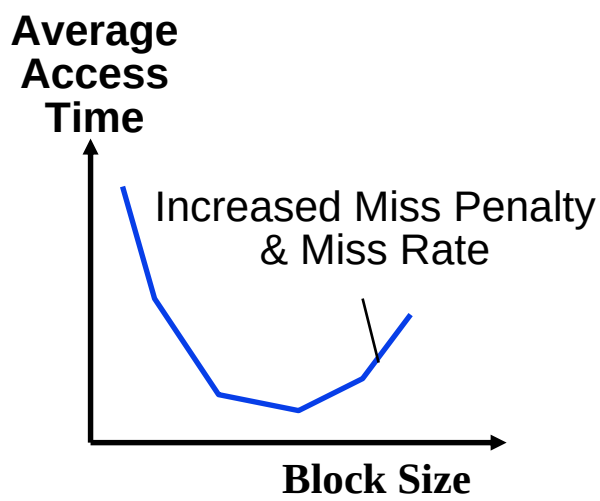


Block Size Tradeoff

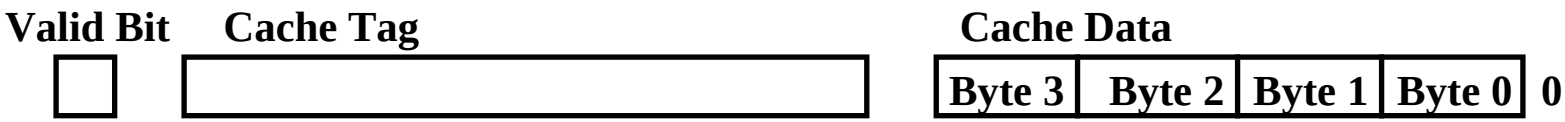
- **In general, larger block size take advantage of spatial locality BUT:**
  - **Larger block size means larger miss penalty:**
    - **Takes longer time to fill up the block**
  - **If block size is too big relative to cache size, miss rate will go up**
    - **Too few cache blocks**
- **In gerneral, Average Access Time:**
  - **= Hit Time x (1 - Miss Rate) + Miss Penalty x Miss Rate**







Extreme Example: single big line



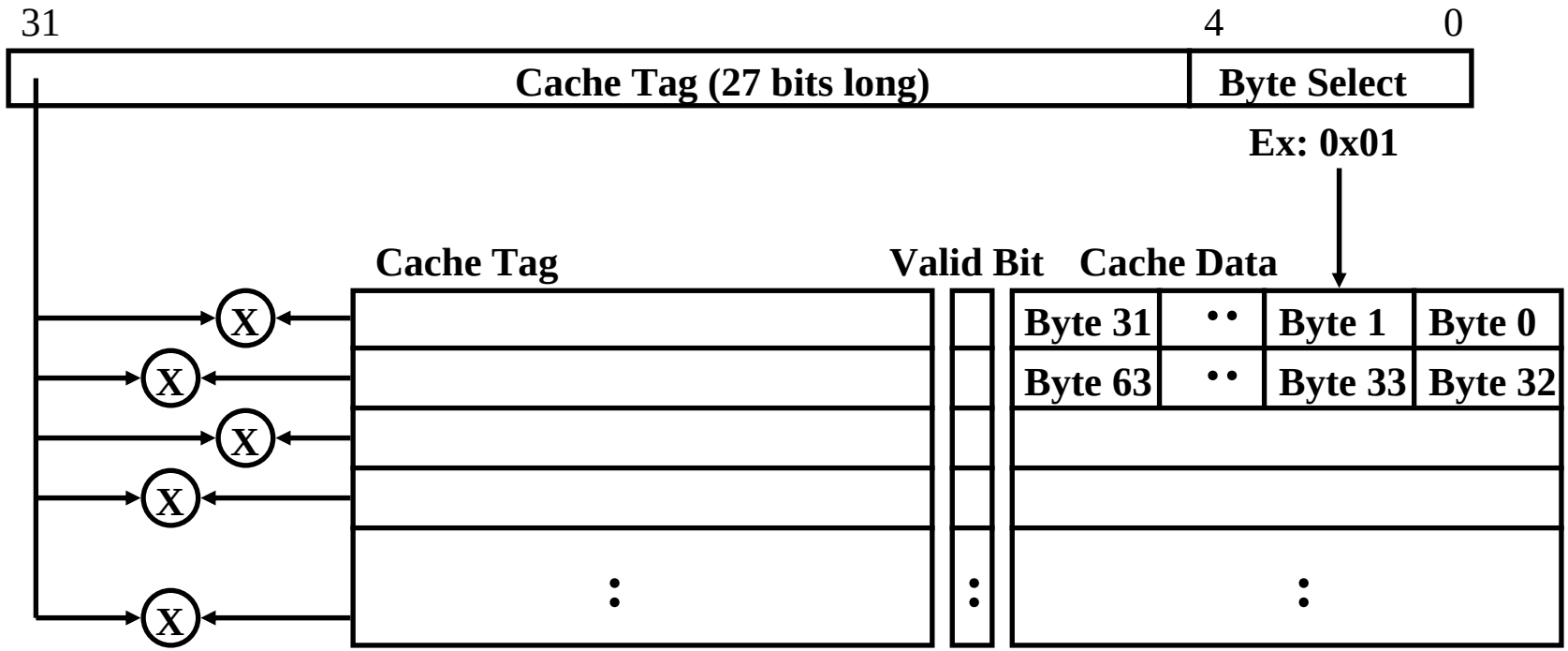
- Cache Size = 4 bytes Block Size = 4 bytes
  - Only ONE entry in the cache
- If an item is accessed, likely that it will be accessed again soon
  - But it is unlikely that it will be accessed again immediately!!!
  - The next access will likely to be a miss again
    - Continually loading data into the cache but discard (force out) them before they are used again
    - Worst nightmare of a cache designer: Ping Pong Effect
- Conflict Misses are misses caused by:
  - Different memory locations mapped to the same cache index
    - Solution 1: make the cache size bigger
    - Solution 2: Multiple entries for the same Cache Index

Another Extreme Example: Fully Associative

° **Fully Associative Cache**

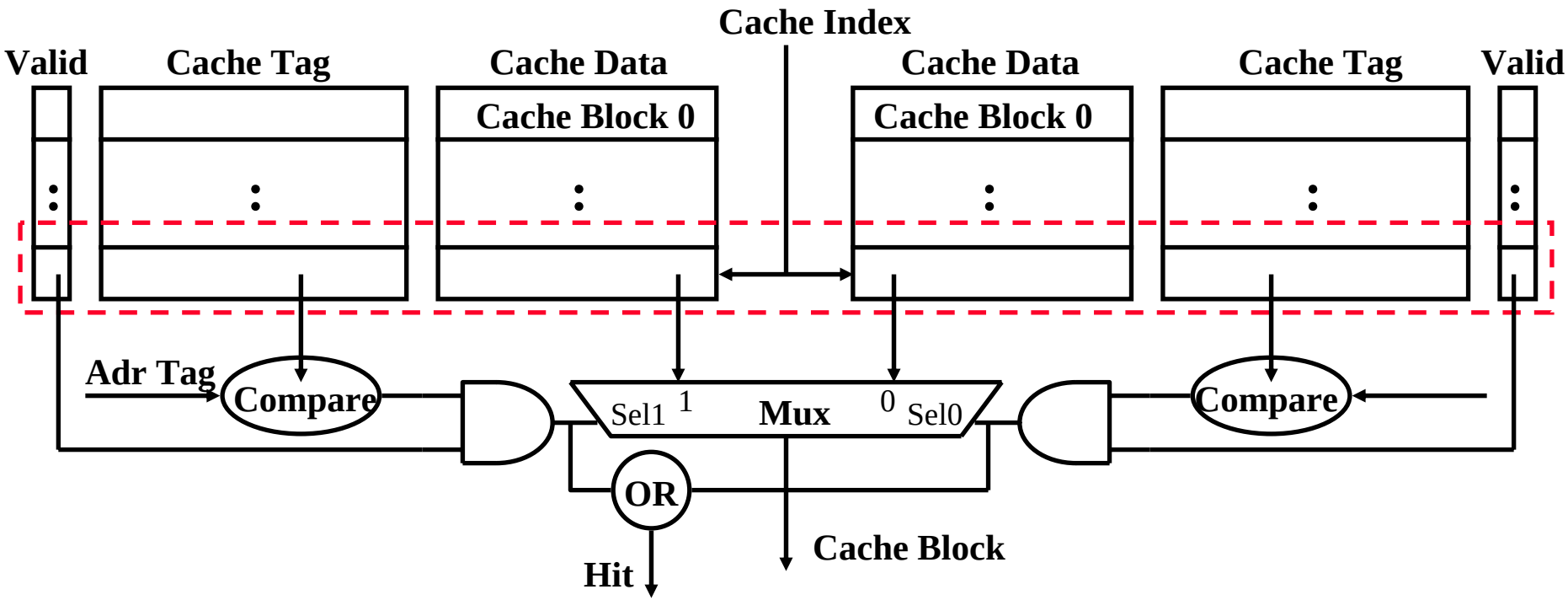
- **Forget about the Cache Index**
- **Compare the Cache Tags of all cache entries in parallel**
- **Example: Block Size = 2 B blocks, we need N 27-bit comparators**

° **By definition: Conflict Miss = 0 for a fully associative cache**



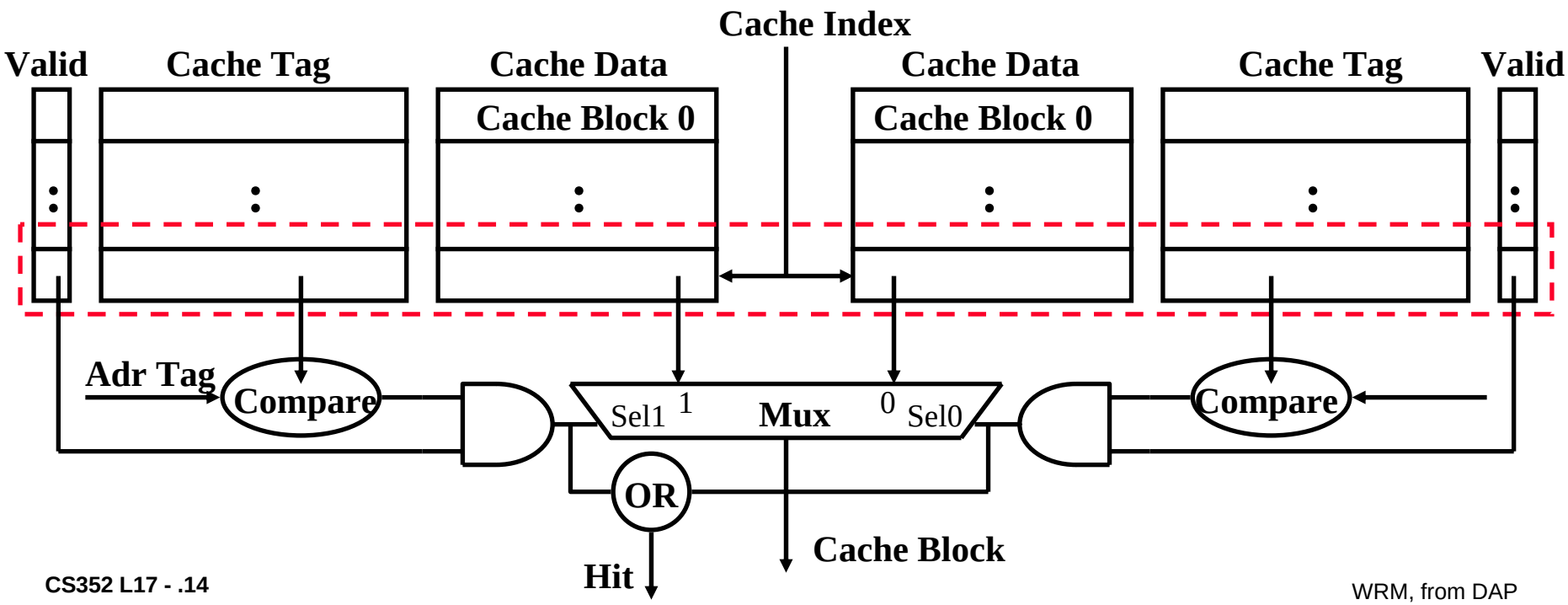
A Two-way Set Associative Cache

- **N-way set associative: N entries for each Cache Index**
  - N direct mapped caches operates in parallel
- **Example: Two-way set associative cache**
  - Cache Index selects a "set" from the cache
  - The two tags in the set are compared in parallel
  - Data is selected based on the tag result



Disadvantage of Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
  - N comparators vs. 1
  - Extra MUX delay for the data
  - Data comes AFTER Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:
  - Possible to assume a hit and continue. Recover later if miss.



Sources of Cache Misses

- **Compulsory (cold start or process migration, first reference): first access to a block**
  - **“Cold” fact of life: not a whole lot you can do about it**
  - **Note: If you are going to run “billions” of instruction, Compulsory Misses are insignificant**
- **Conflict (collision):**
  - **Multiple memory locations mapped to the same cache location**
  - **Solution 1: increase cache size**
  - **Solution 2: increase associativity**
- **Capacity:**
  - **Cache cannot contain all blocks access by the program**
  - **Solution: increase cache size**
- **Invalidation: other process (e.g., I/O) updates memory**

Improving Cache Performance: 3 general options

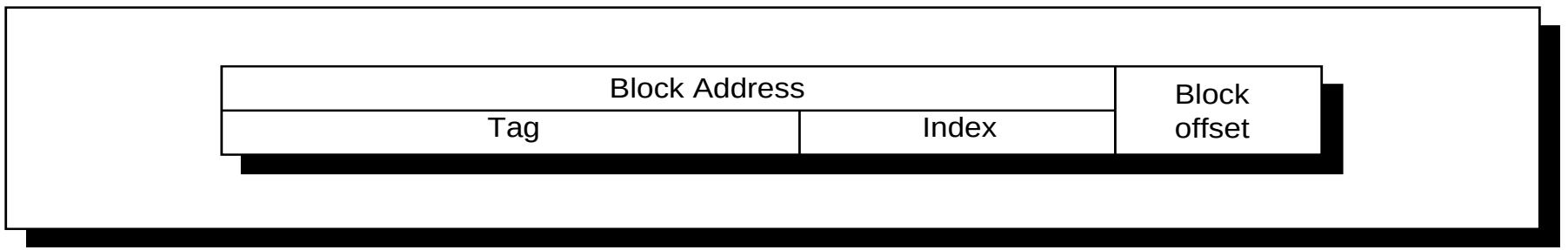
1. Reduce the miss rate,
2. Reduce the miss penalty, or
3. Reduce the hit time.

5 Questions for Memory Hierarchy

- **Q1: Where can a block be placed in the upper level?** ***(Block placement)***
- **Q2: How is a block found if it is in the upper level?** ***(Block identification)***
- **Q3: Which block should be replaced on a miss?** ***(Block replacement)***
- **Q4: What happens on a write?** ***(Write strategy)***
- ***Q5: What is the "bookkeeping strategy"?***
  - ***Does HW or SW handle cache misses?***
  - ***What are the data structures?***
    - ***Especially for address translation***

Q2: How is a block found if it is in the upper level?

- Tag on each block
  - No need to check index or block offset
- Increasing associativity shrinks index, expands tag



Q3: Which block should be replaced on a miss?

- **Easy for Direct Mapped**
- **Set Associative or Fully Associative:**
  - **Random**
  - **LRU (Least Recently Used)**

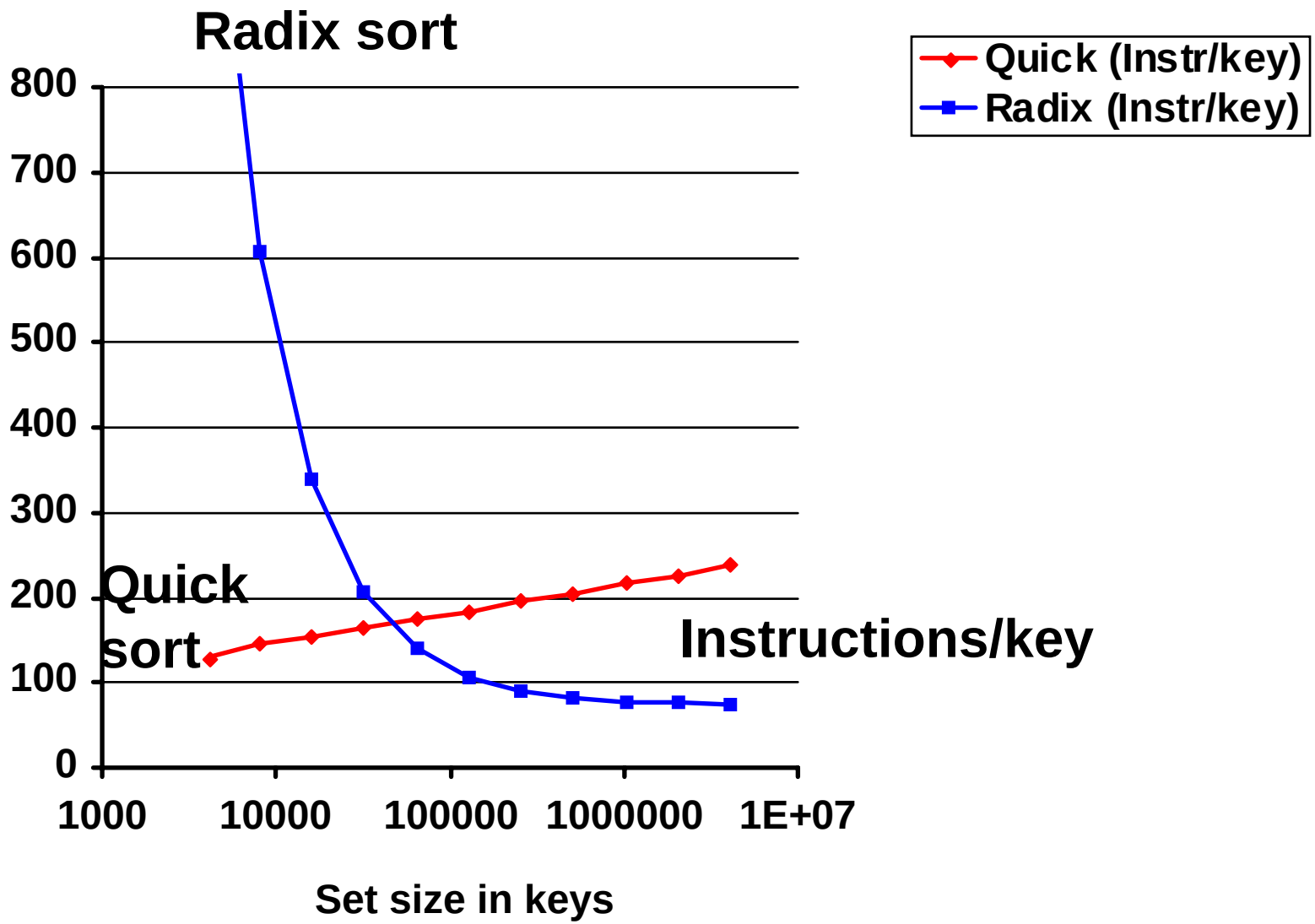
Q4: What happens on a write?

- ***Write through*—The information is written to both the block in the cache and to the block in the lower-level memory.**
- ***Write back*—The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.**
  - **is block clean or dirty?**
- **Pros and Cons of each?**
  - **WT: read misses cannot result in writes**
  - **WB: no writes of repeated writes**
- **WT always combined with write buffers so that don't wait for lower level memory**

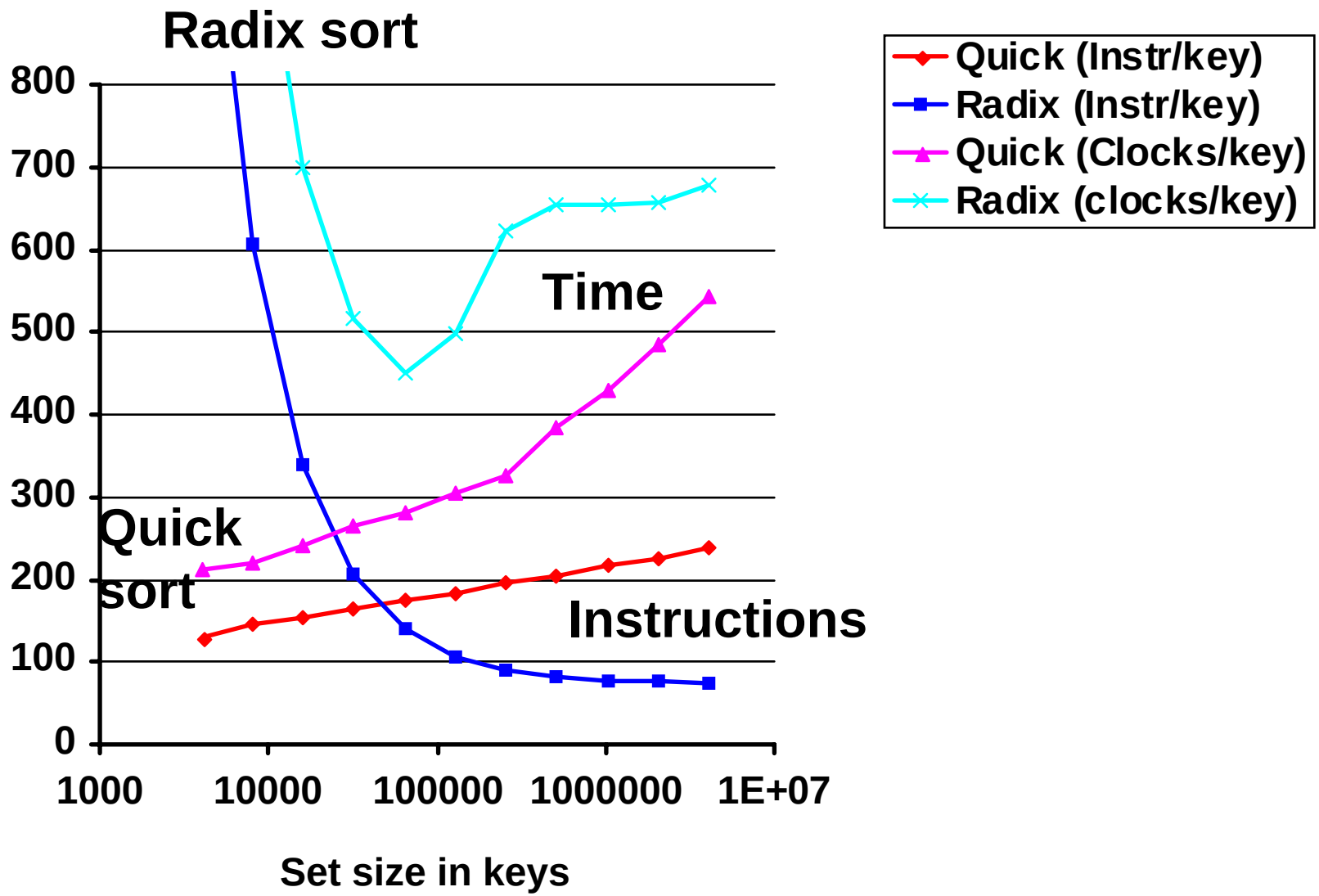
Impact of Memory Hierarchy on Algorithms

- Today CPU time is a function of (ops, cache misses) vs. just f(ops):
  What does this mean to Compilers, Data structures, Algorithms?
- "The Influence of Caches on the Performance of Sorting" by A. LaMarca and R.E. Ladner. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, January, 1997, 370-379.
- Quicksort: fastest comparison based sorting algorithm when all keys fit in memory
- Radix sort: also called "linear time" sort because for keys of fixed length and fixed radix a constant number of passes over the data is sufficient independent of the number of keys
- For Alphastation 250, 32 byte blocks, direct mapped L2 2MB cache, 8 byte keys, from 4000 to 4000000

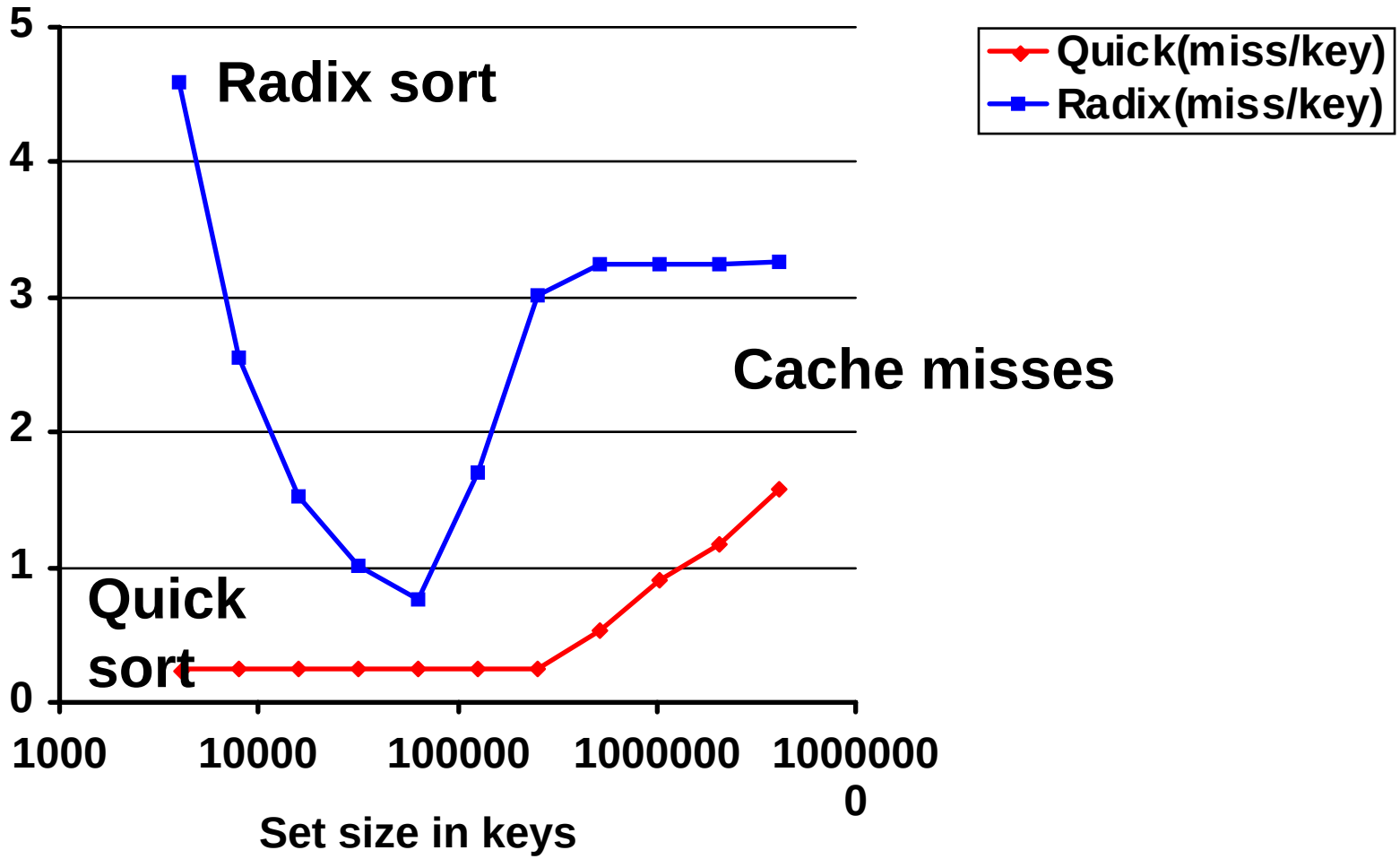
Quicksort vs. Radix as vary number keys: Instructions



Quicksort vs. Radix as vary number keys: Instrs & Time

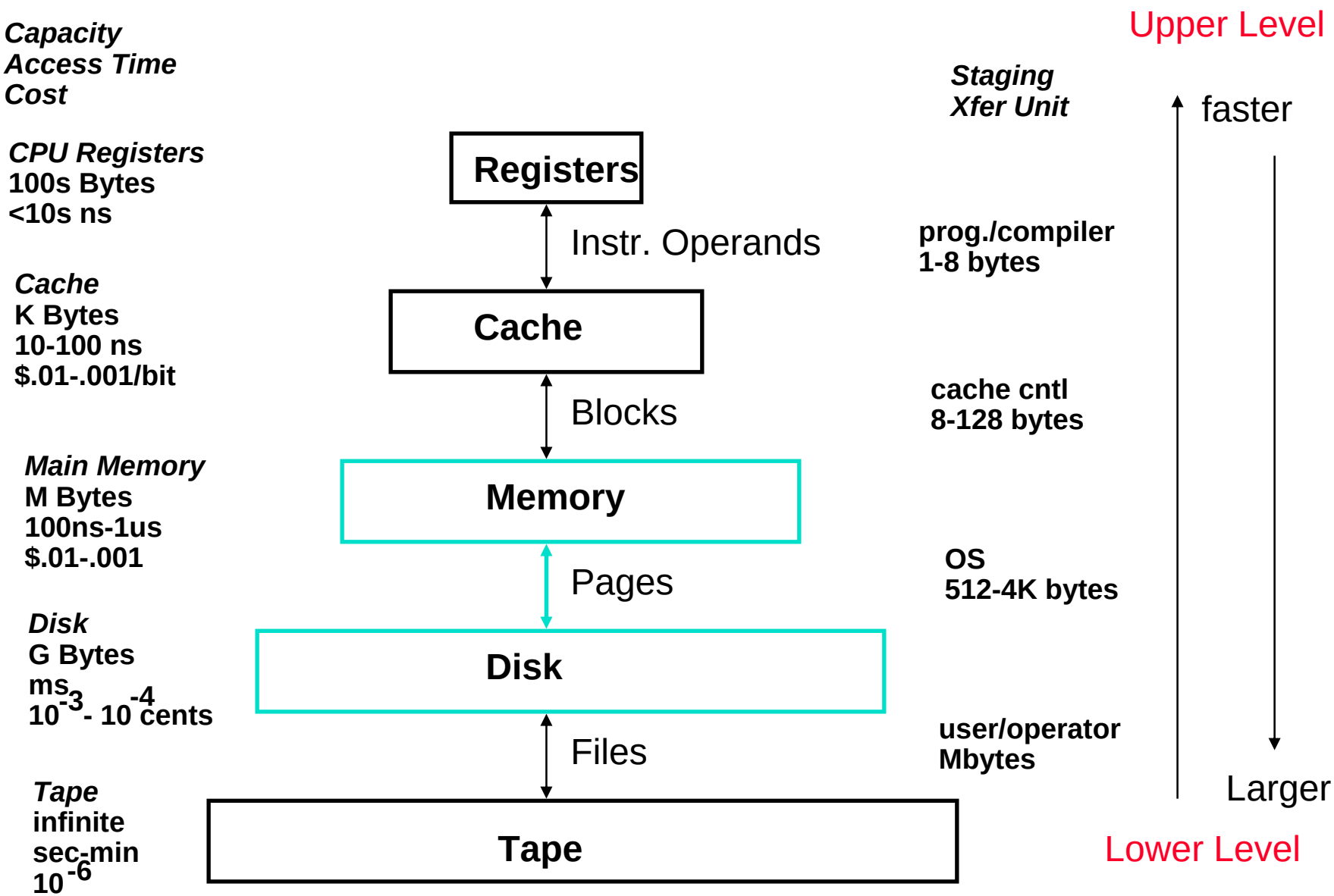


Quicksort vs. Radix as vary number keys: Cache misses



What is proper approach to fast algorithms?

Recall: Levels of the Memory Hierarchy



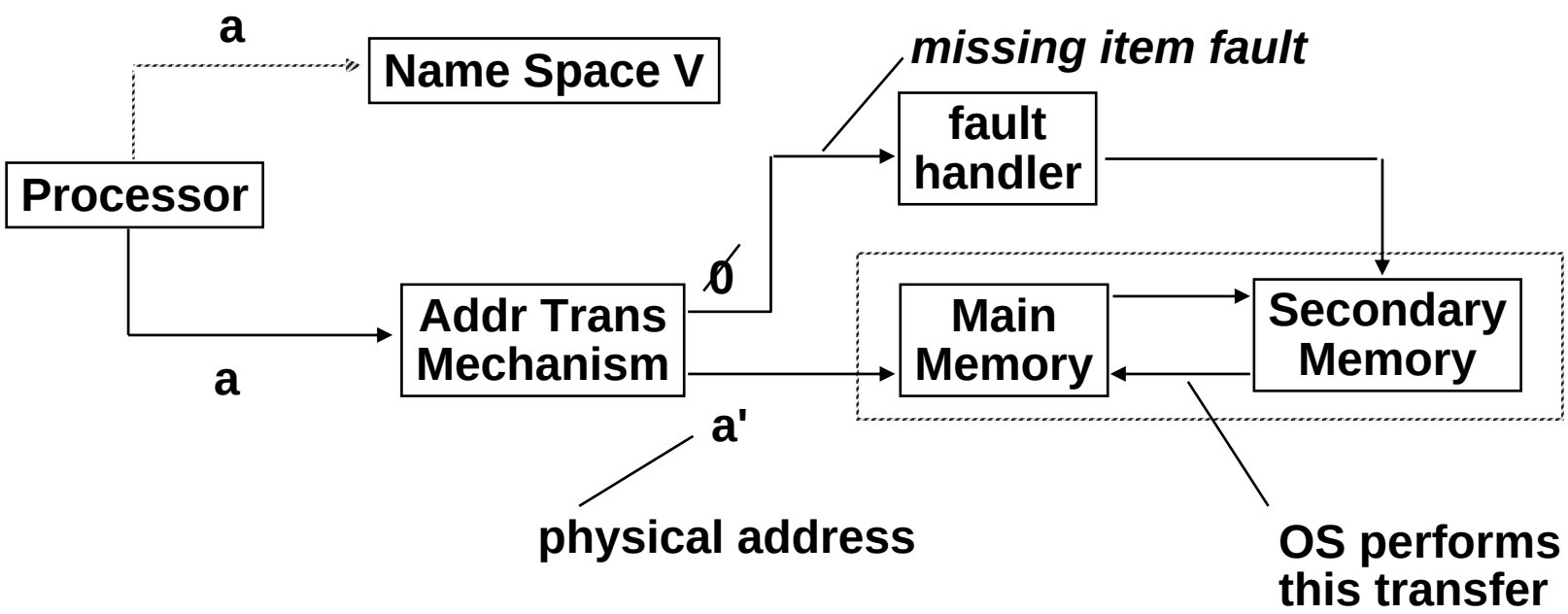
Address Map

V = {0, 1, . . . , n - 1} virtual address space n > m
M = {0, 1, . . . , m - 1} physical address space

MAP: V --> M U {ø} address mapping function

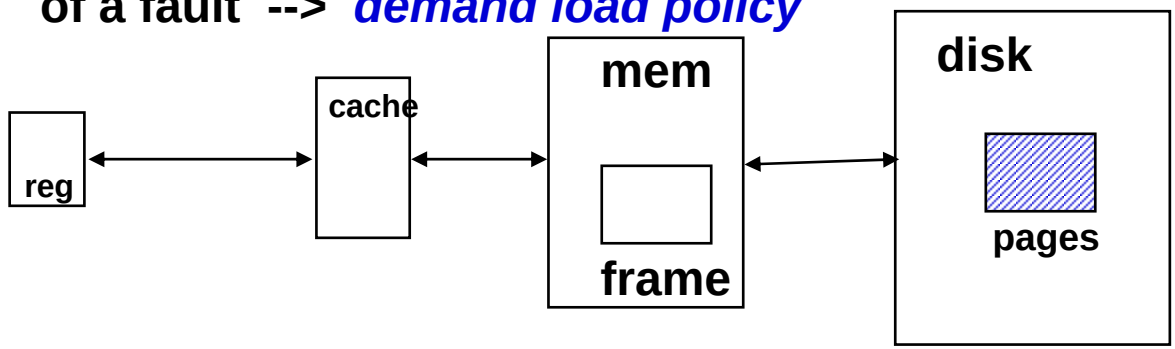
MAP(a) = a' if data at virtual address a is present in physical address a' and a' in M

= ø if data at virtual address a is not present in M



Basic Issues in Virtual Memory System Design

- size of information blocks that are transferred from secondary to main storage (M)
- block of information brought into M, and M is full, then some region of M must be released to make room for the new block --> *replacement policy*
- which region of M is to hold the new block --> *placement policy*
- missing item fetched from secondary memory only on the occurrence of a fault --> *demand load policy*



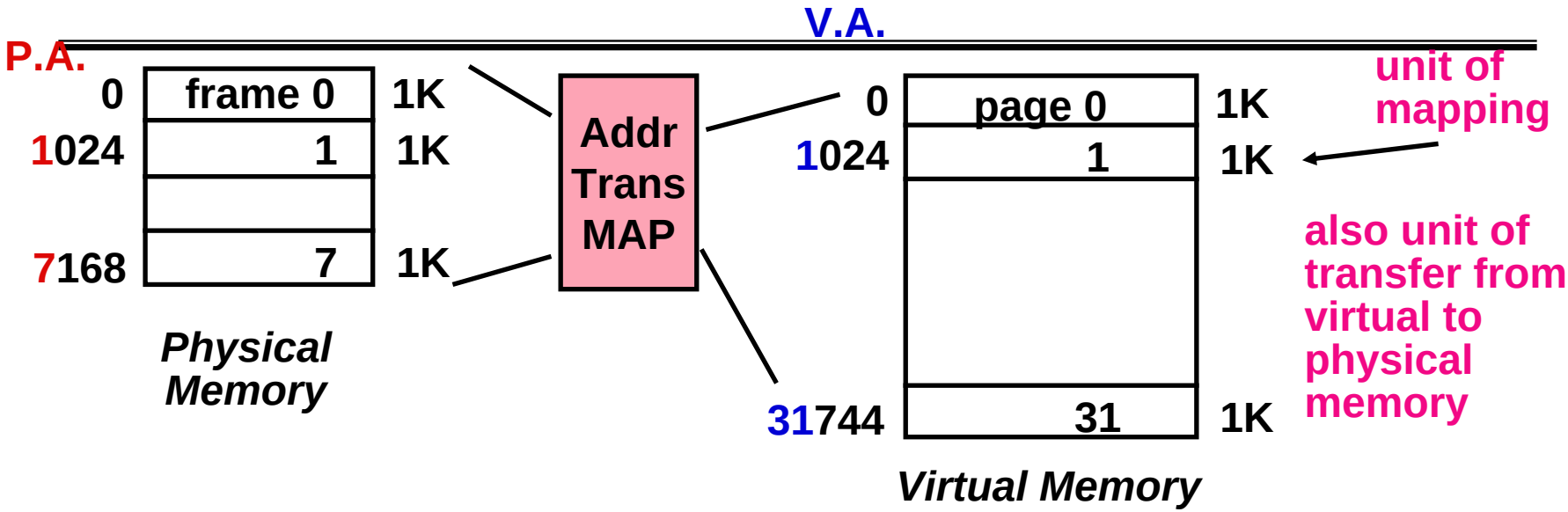
Paging Organization

virtual and physical address space partitioned into blocks of equal size

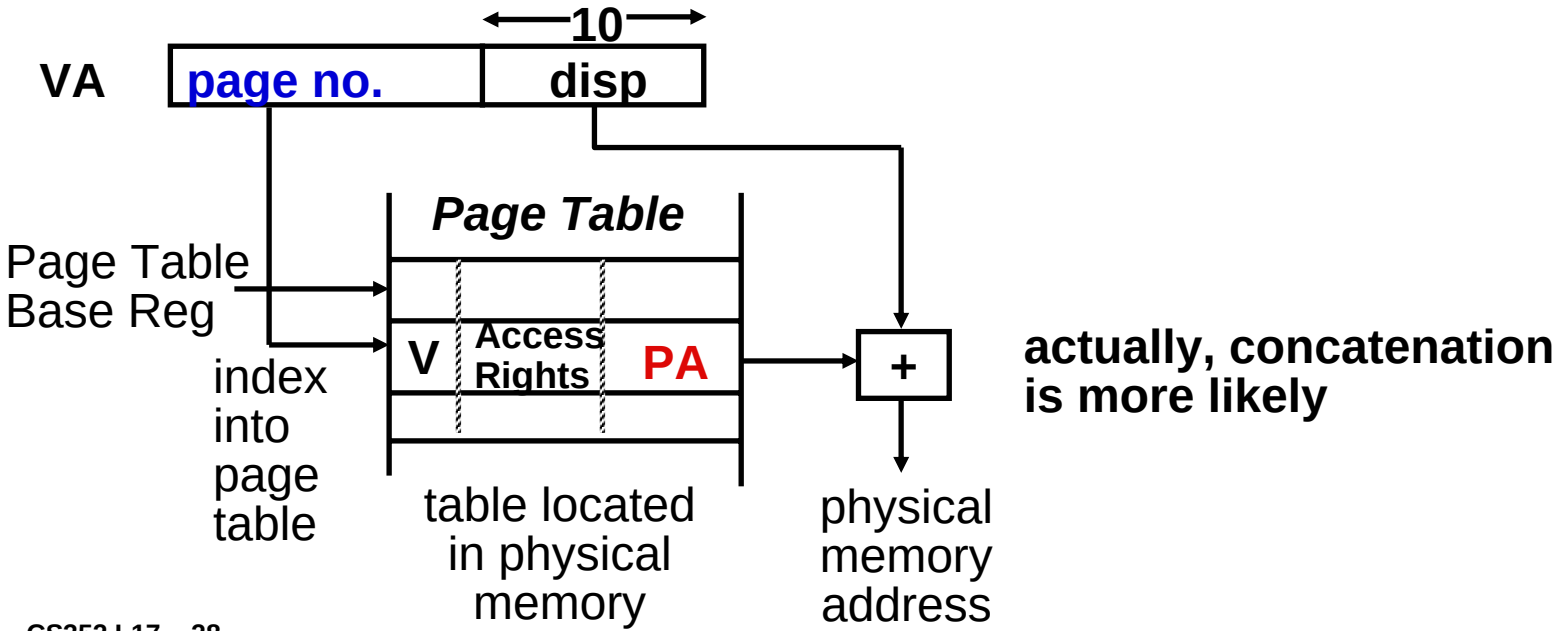
page frames

pages

Paging Organization



Address Mapping



TLBs

A way to speed up translation is to use a special cache of recently used page table entries -- this has many names, but the most frequently used is *Translation Lookaside Buffer* or *TLB*

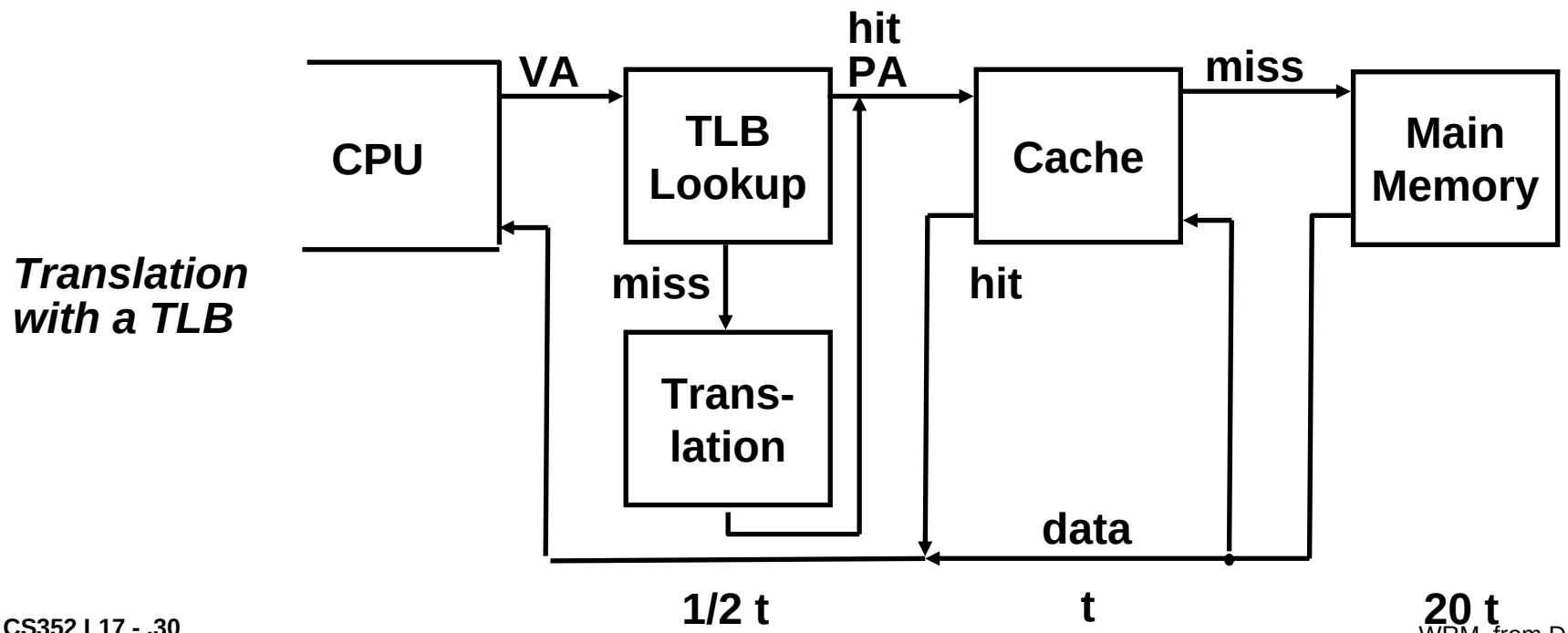
Virtual Address	Physical Address	Dirty	Ref	Valid	Access

TLB access time comparable to cache access time (much less than main memory access time)

Translation Look-Aside Buffers

Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

TLBs are usually small, typically not more than 128 - 256 entries even on high end machines. This permits fully associative lookup on these machines. Most mid-range machines use small n-way set associative organizations.



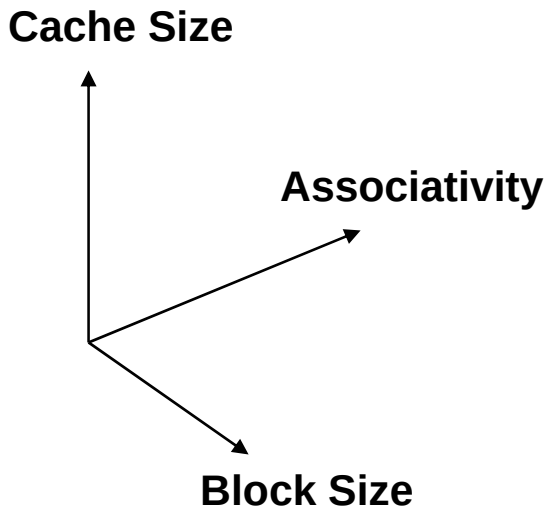
Translation with a TLB

Summary #1/ 4:

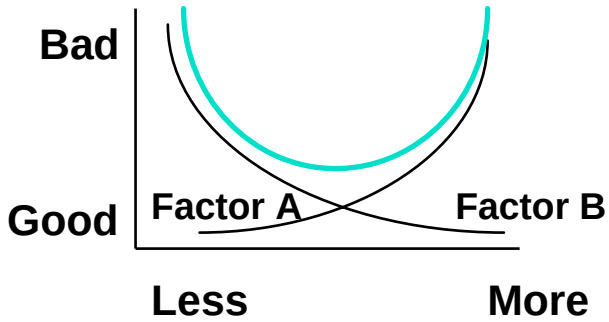
- **The Principle of Locality:**
  - **Program likely to access a relatively small portion of the address space at any instant of time.**
    - **Temporal Locality: Locality in Time**
    - **Spatial Locality: Locality in Space**
- **Three Major Categories of Cache Misses:**
  - **Compulsory Misses: sad facts of life. Example: cold start misses.**
  - **Conflict Misses: increase cache size and/or associativity. Nightmare Scenario: ping pong effect!**
  - **Capacity Misses: increase cache size**
- **Cache Design Space**
  - **total size, block size, associativity**
  - **replacement policy**
  - **write-hit policy (write-through, write-back)**
  - **write-miss policy**

Summary #2 / 4: The 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

Summary #3 / 4 : TLB, Virtual Memory

- Caches, TLBs, Virtual Memory all understood by examining how they deal with 4 questions:
  - 1) Where can block be placed?
  - 2) How is block found?
  - 3) What block is repalced on miss?
  - 4) How are writes handled?
  - 5) How is bookeeping done?
- Page tables map virtual address to physical address
- TLBs are important for fast translation

Summary #4 / 4: Memory Hierachy

- VIrtual memory was controversial at the time: can SW automatically manage 64KB across many programs?
  - 1000X DRAM growth removed the controversy
- Today VM allows many processes to share single memory without having to swap all processes to disk; VM protection is more important than memory hierarchy
- Today CPU time is a function of
  f(ops, cache misses)
  vs. just
  f(ops):

  What does this mean to Compilers, Data structures, Algorithms?

Summary

- **This time**
  - **Caching and virtual memory review**
- **Next time**
  - **DRAM organization**
- **Administrivia**
  - **Homework #5 is online, and due on April 6.**
  - **Final project will be cache simulator, in either Java or C/C++**