西安交通大學
XI'AN JIAOTONG UNIVERSITY

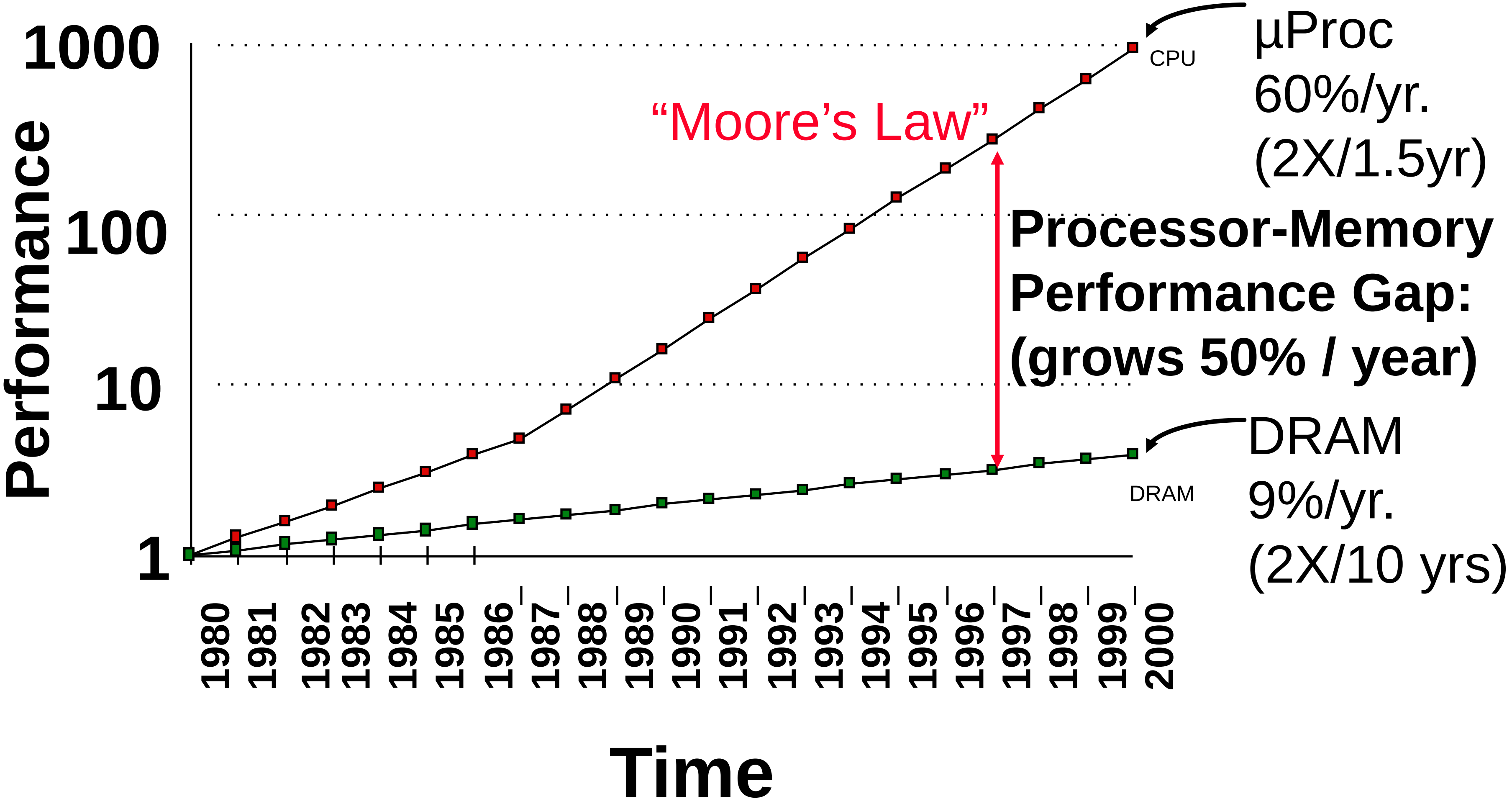
# Modern Computer Architecture

## Lecture3 Review of Memory Hierarchy

**Hongbin Sun**
国家集成电路人才培养基地
**Xi'an Jiaotong University**

# Recap: Who Cares About the Memory Hierarchy?

**Processor-DRAM Memory Gap (latency)**



µProc 60%/yr. (2X/1.5yr)

"Moore's Law"

**Processor-Memory Performance Gap: (grows 50% / year)**

DRAM 9%/yr. (2X/10 yrs)

# Levels of the Memory Hierarchy

*Capacity*
*Access Time*
*Cost*

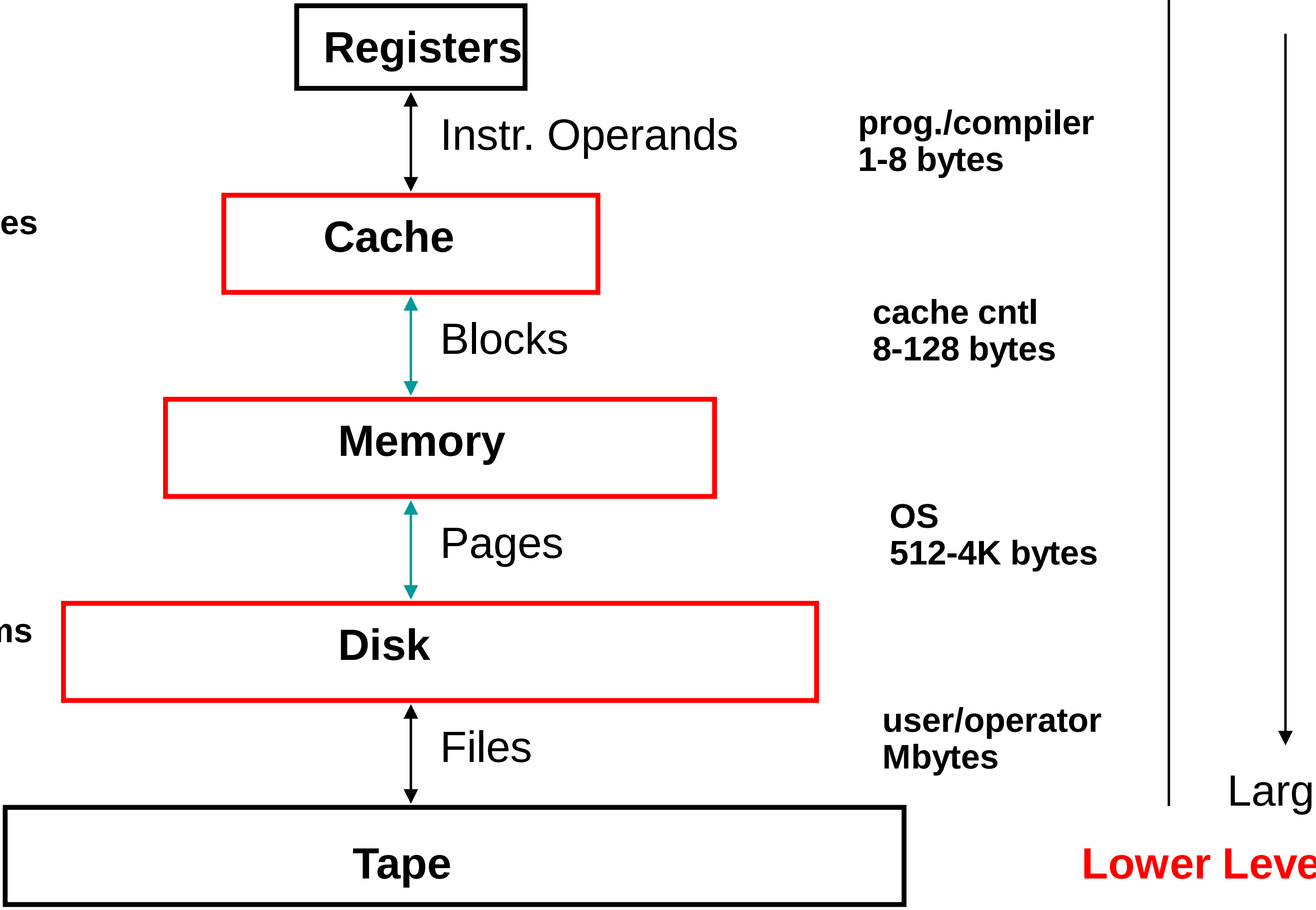
*CPU Registers*
**100s Bytes**
**<1s ns**

*Cache*
**10s-100s K Bytes**
**1-10 ns**
**$10/ MByte**

*Main Memory*
**M Bytes**
**100ns- 300ns**
**$1/ MByte**

*Disk*
**10s G Bytes, 10 ms**
**(10,000,000 ns)**
**$0.0031/ MByte**

*Tape*
**infinite**
**sec-min**
**$0.0014/ MByte**



**Upper Level**

*Staging*
*Xfer Unit*

**prog./compiler**
**1-8 bytes**

**cache cntl**
**8-128 bytes**

**OS**
**512-4K bytes**

**user/operator**
**Mbytes**

**Lower Level**

# The Principle of Locality

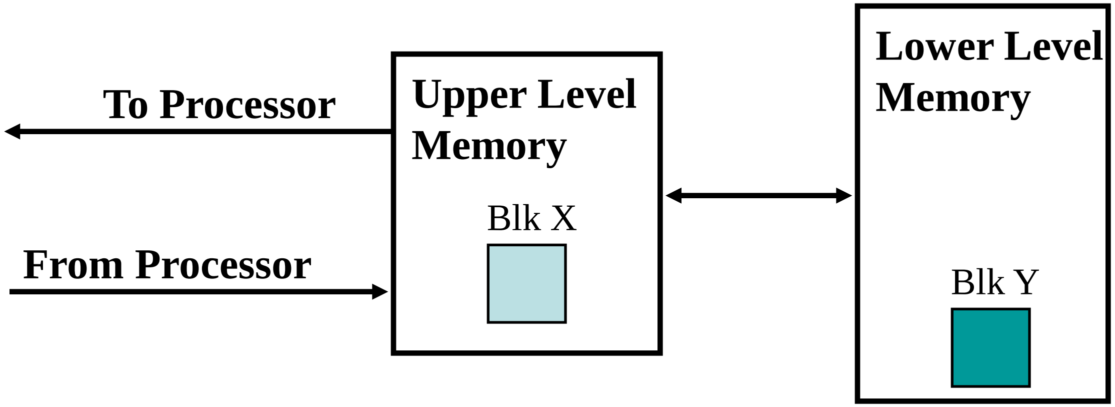
- **The Principle of Locality:**
  - Program access a relatively small portion of the address space at any instant of time.
- **Two Different Types of Locality:**
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon
    (e.g., straightline code, array access)
- **Last 15 years, HW (hardware) relied on locality for speed**

# Memory Hierarchy: Terminology

- **Hit: data appears in some block in the upper level (example: Block X)**
  - **Hit Rate: the fraction of memory access found in the upper level**
  - **Hit Time: Time to access the upper level which consists of**

    RAM access time + Time to determine hit/miss
- **Miss: data needs to be retrieve from a block in the lower level (Block Y)**
  - **Miss Rate = 1 - (Hit Rate)**
  - **Miss Penalty: Time to replace a block in the upper level +**

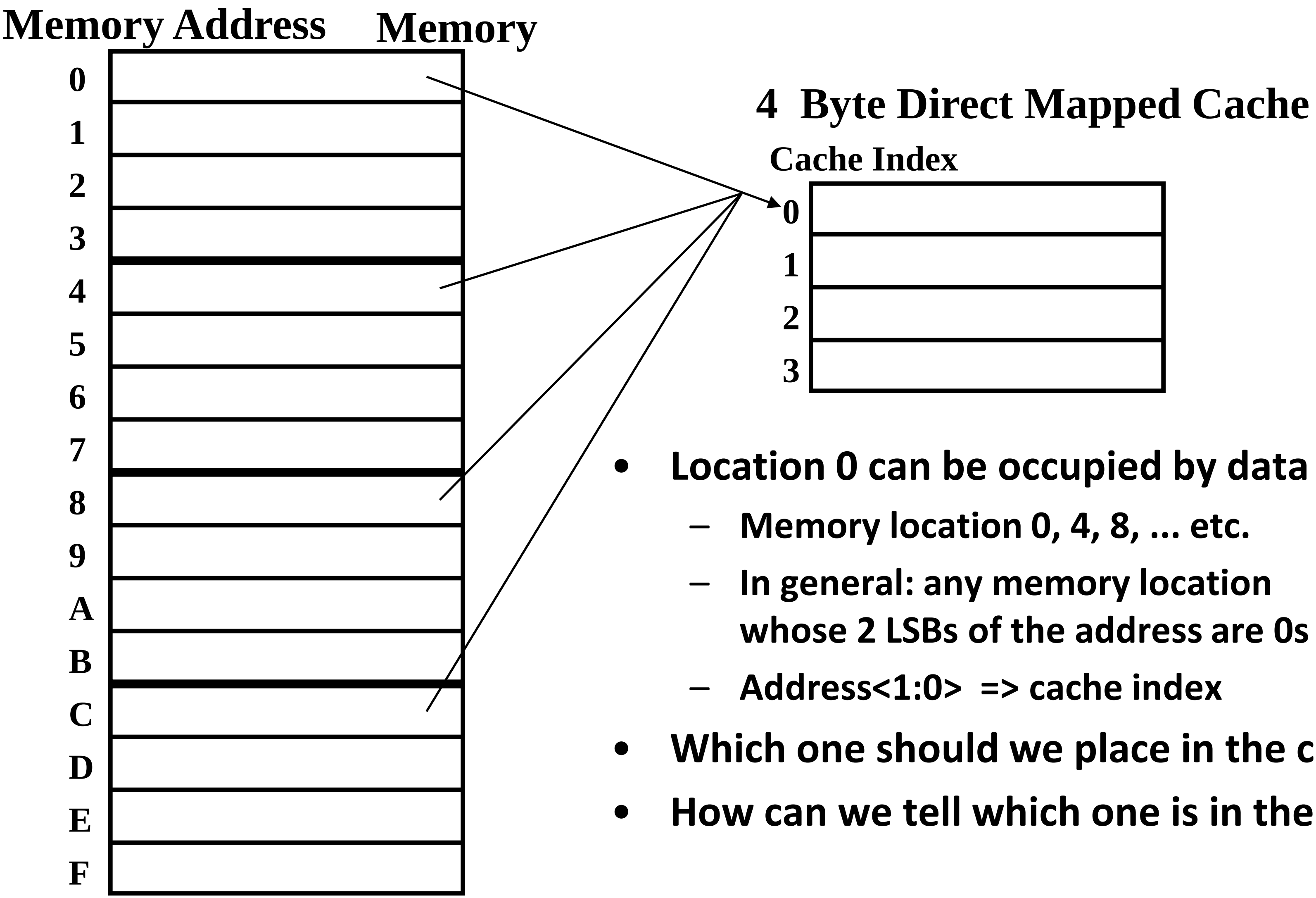
    Time to deliver the block the processor
- **Hit Time << Miss Penalty (500 instructions on 21264!)**



# Cache Measures

- ***Hit rate***: fraction found in that level
  - So high that usually talk about *Miss rate*
  - Miss rate fallacy: as MIPS to CPU performance,
    miss rate to average memory access time in memory
- Average memory-access time
  = Hit time + Miss rate x Miss penalty
  (ns or clocks)
- ***Miss penalty***: time to replace a block from lower level, including time to replace in CPU
  - *access time*: time to lower level
    = f(latency to lower level)
  - *transfer time*: time to transfer block
    =f(BW between upper & lower levels)

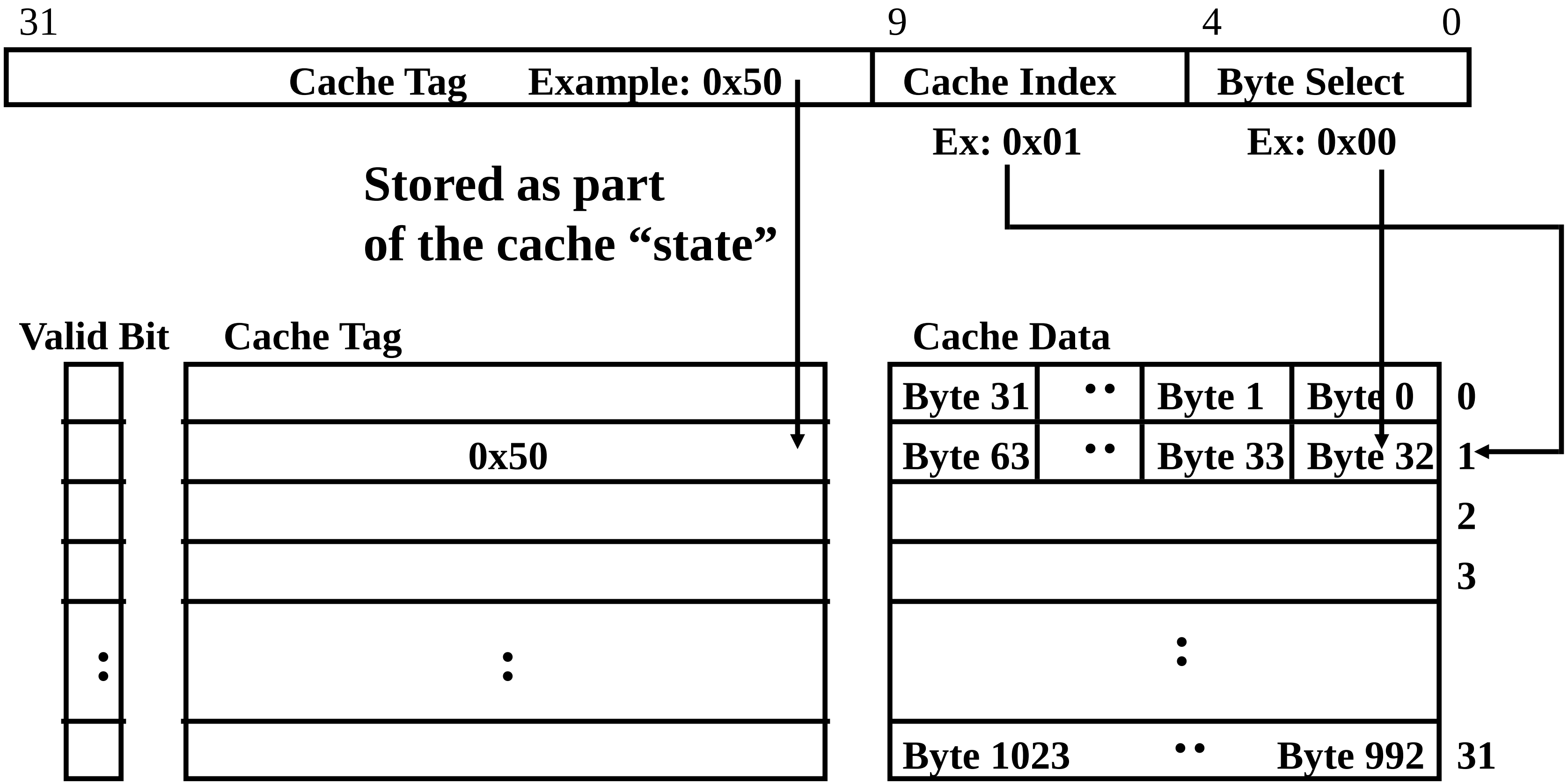
# Simplest Cache: Direct Mapped



- Location 0 can be occupied by data from:
  - Memory location 0, 4, 8, ... etc.
  - In general: any memory location whose 2 LSBs of the address are 0s
  - Address<1:0> => cache index
- Which one should we place in the cache?
- How can we tell which one is in the cache?

# 1 KB Direct Mapped Cache, 32B 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)



31 9 4 0

Cache Tag Example: 0x50

Cache Index

Byte Select

Ex: 0x01

Ex: 0x00

Stored as part
of the cache "state"

Valid Bit

Cache Tag

0x50

Cache Data

Byte 31 .. Byte 1 Byte 0 0

Byte 63 .. Byte 33 Byte 32 1

2

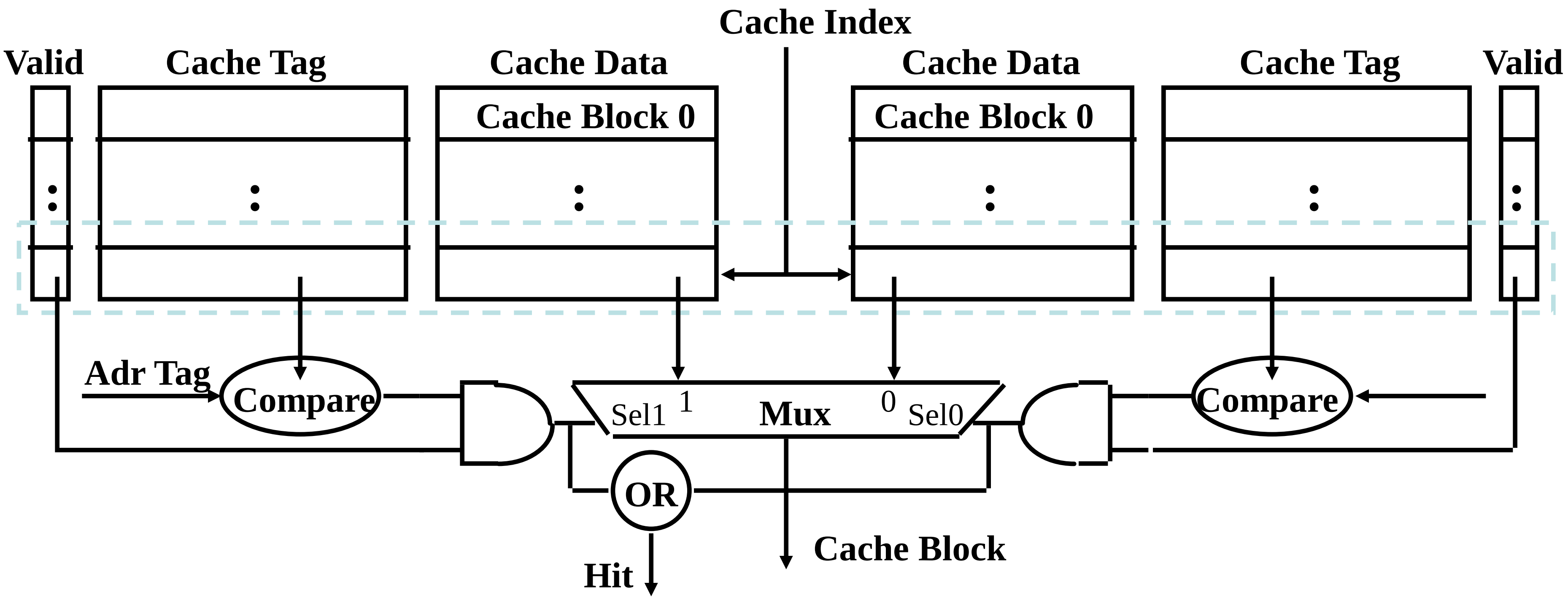
3

:

Byte 1023 .. Byte 992 31

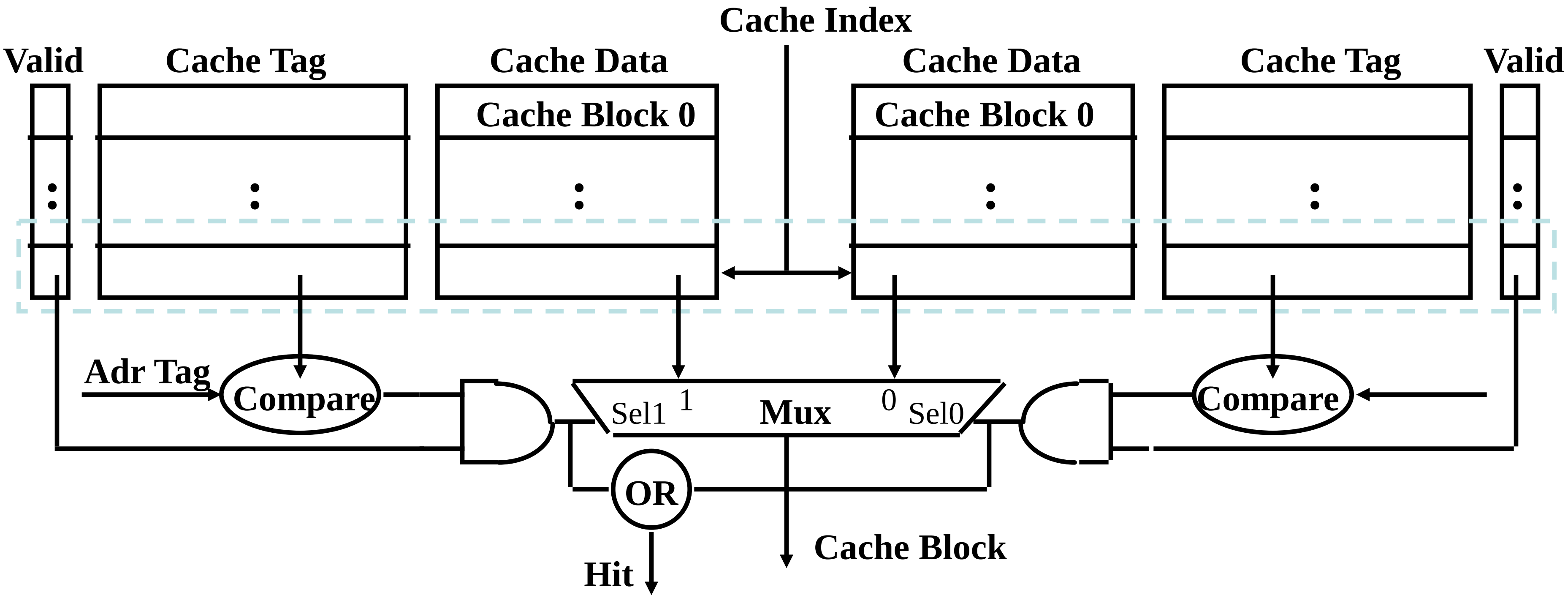
# Two-way Set Associative Cache

- **N-way set associative: N entries for each Cache Index**
  - **N direct mapped caches operates in parallel (N typically 2 to 4)**
- **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 v. Direct Mapped Cache:**
  - **N comparators vs. 1**
  - **Extra MUX delay for the data**
  - **Data comes AFTER Hit/Miss**
- **In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:**
  - **Possible to assume a hit and continue. Recover later if miss.**

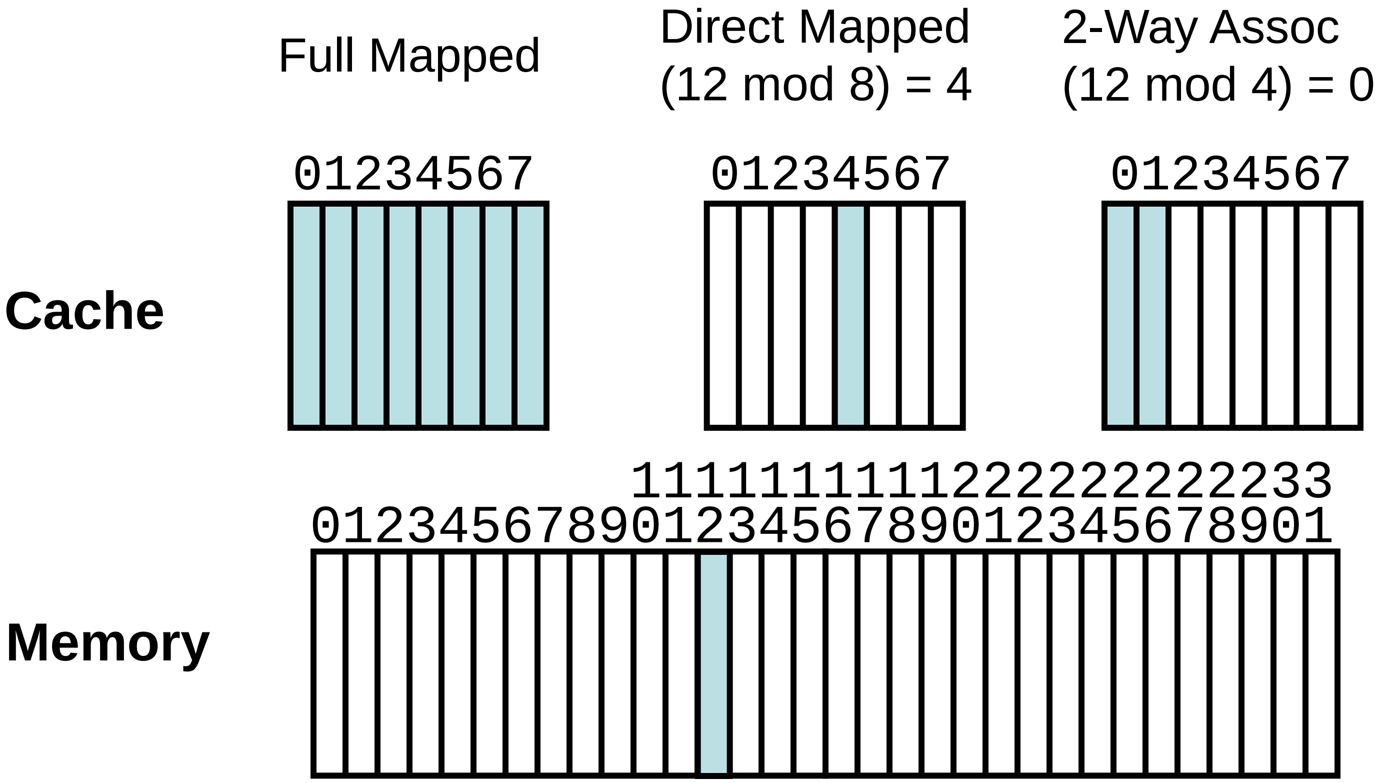


# 4 Questions for Memory Hierarchy

- 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)*

# Q1: Where can a block be placed in the upper level?

- **Block 12 placed in 8 block cache:**
  - **Fully associative, direct mapped, 2-way set associative**
  - **S.A. Mapping = Block Number Modulo Number Sets**



Full Mapped

Direct Mapped
(12 mod 8) = 4

2-Way Assoc
(12 mod 4) = 0

**Cache**

**Memory**

# 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 →

| | | |
|---|---|---|
| Block Address | | Block Offset |
| Tag | Index | |

# Q3: Which block should be replaced on a miss?

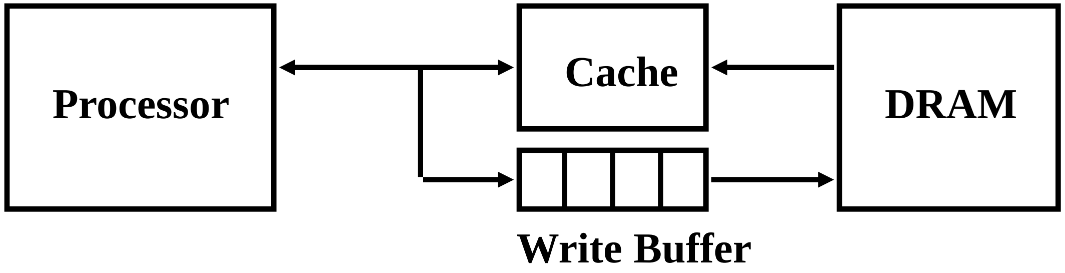
- Easy for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least Recently Used)

| Assoc: | 2-way | | 4-way | | 8-way | |
|---|---|---|---|---|---|---|
| Size | LRU | Ran | LRU | Ran | LRU | Ran |
| 16 KB | 5.2% | 5.7% | 4.7% | 5.3% | 4.4% | 5.0% |
| 64 KB | 1.9% | 2.0% | 1.5% | 1.7% | 1.4% | 1.5% |
| 256 KB | 1.15% | 1.17% | 1.13% | 1.13% | 1.12% | 1.12% |

# 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 repeated writes to same location
- **WT always combined with write buffers so that don't wait for lower level memory**

# Write Buffer for Write Through



- A Write Buffer is needed between the Cache and Memory
  - Processor: writes data into the cache and the write buffer
  - Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO:
  - Typical number of entries: 4
  - Works fine if: Store frequency (w.r.t. time) << 1 / DRAM write cycle
- Memory system designer's nightmare:
  - Store frequency (w.r.t. time) -> 1 / DRAM write cycle
  - Write buffer saturation

# 6 Basic Cache Optimizations

- **Reducing Miss Rate**

**1. Larger Block size (compulsory misses)**

**2. Larger Cache size (capacity misses)**

**3. Higher Associativity (conflict misses)**

- **Reducing Miss Penalty**

**4. Multilevel Caches**

- **Reducing hit time**

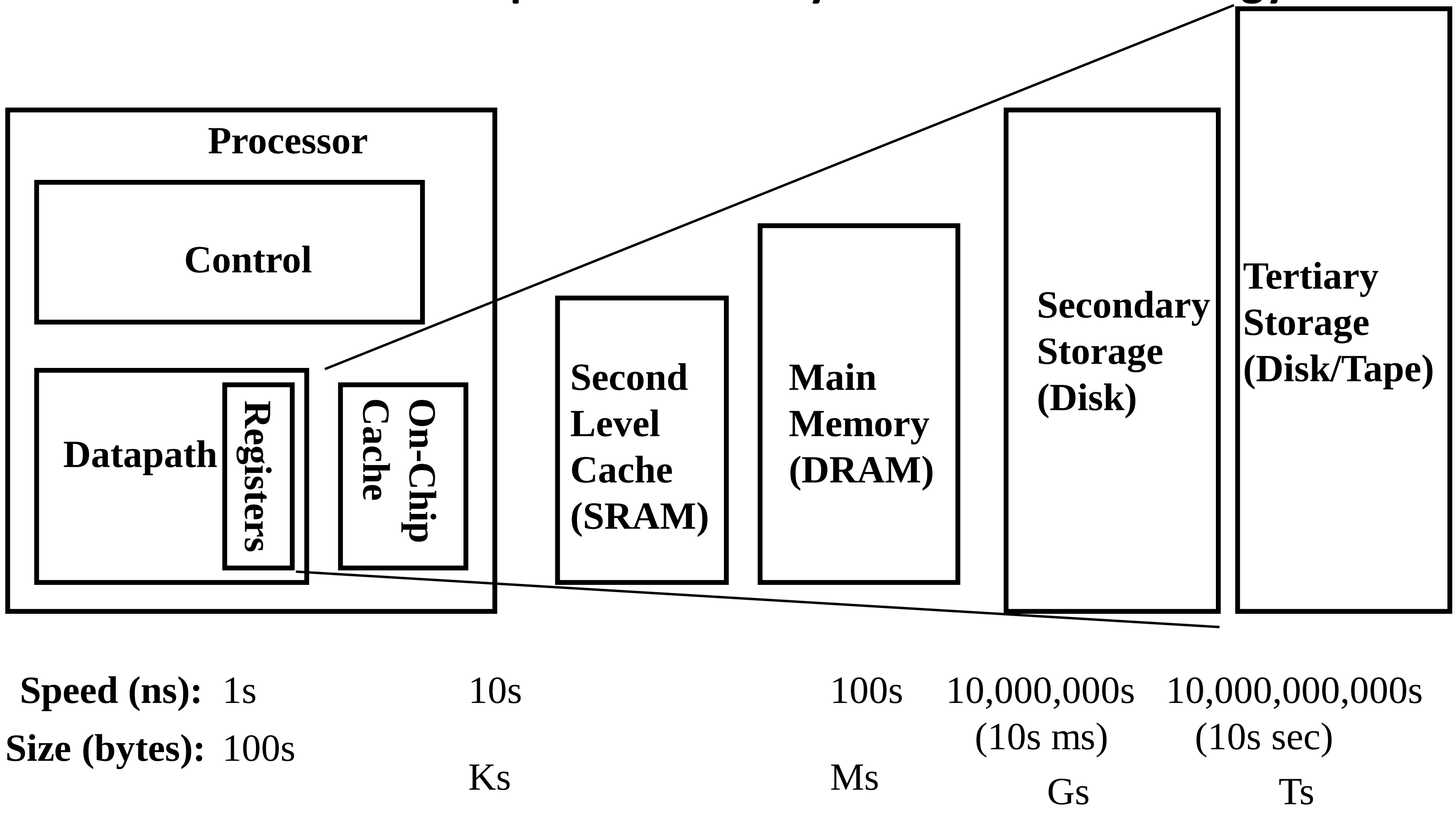
**5. Giving Reads Priority over Writes**

- **E.g., Read complete before earlier writes in write buffer**

**6. Avoid address translation**

# A Modern Memory Hierarchy

- 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.



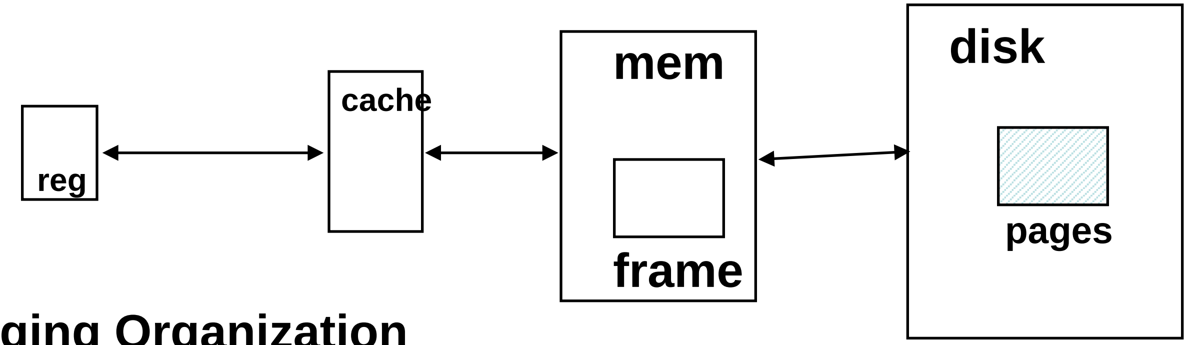
# Basic Issues in VM System Design

- 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*

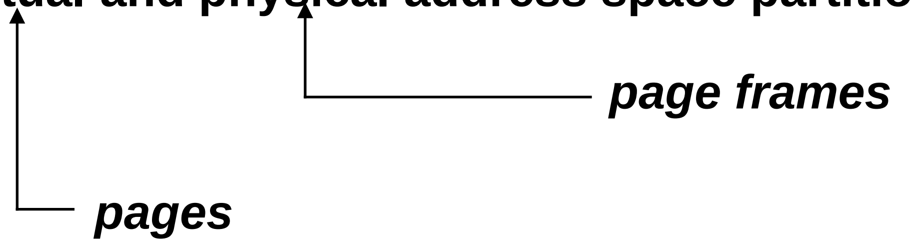
- which region of M is to hold the new block --> *placement policy*

- missing item fetched from secondary memory only on the occurrence of a fault --> *demand load policy*



<u>Paging Organization</u>

virtual and physical address space partitioned into blocks of equal size



# Address Map

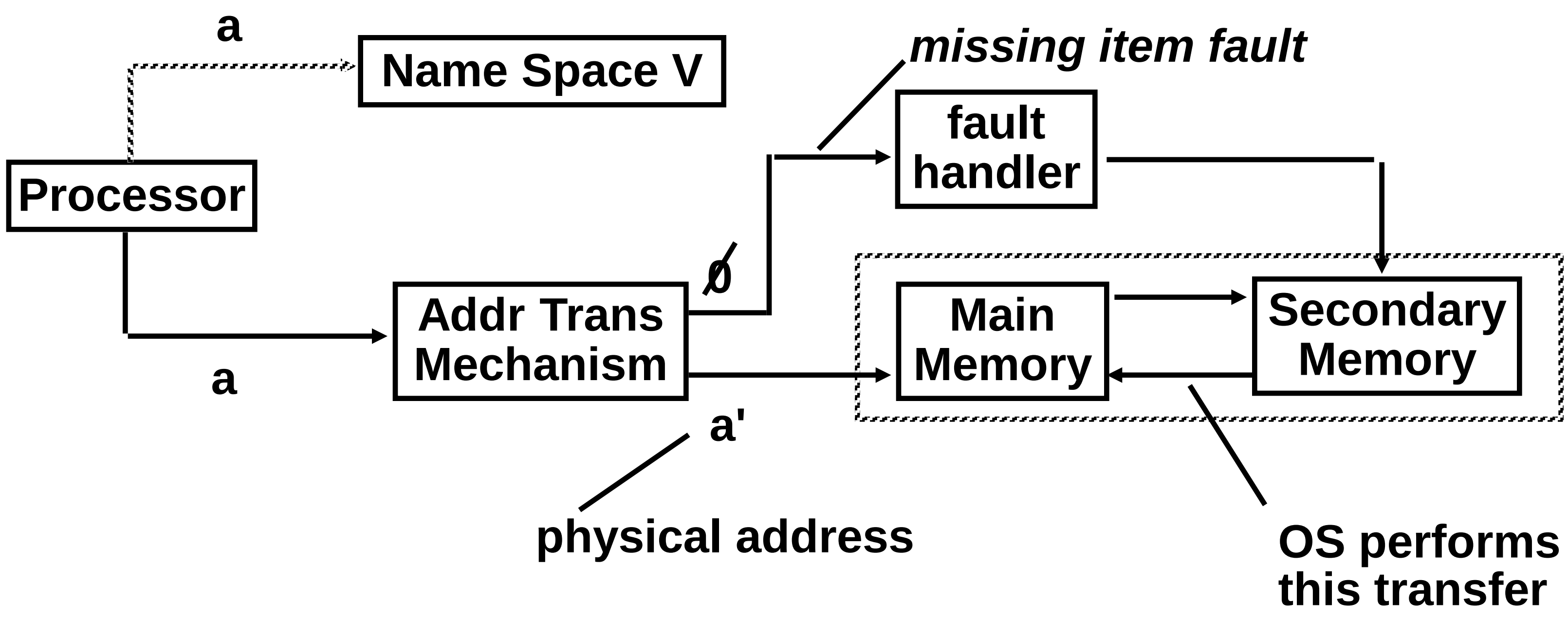
V = {0, 1, . . . , n - 1} virtual address space
M = {0, 1, . . . , m - 1} physical address space

n > m

MAP: V --> M U {0} address mapping function

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

= 0 if data at virtual address a is not present in M



a

Name Space V

Processor

a

Addr Trans Mechanism

0

a'

*missing item fault*

fault handler

Main Memory

Secondary Memory

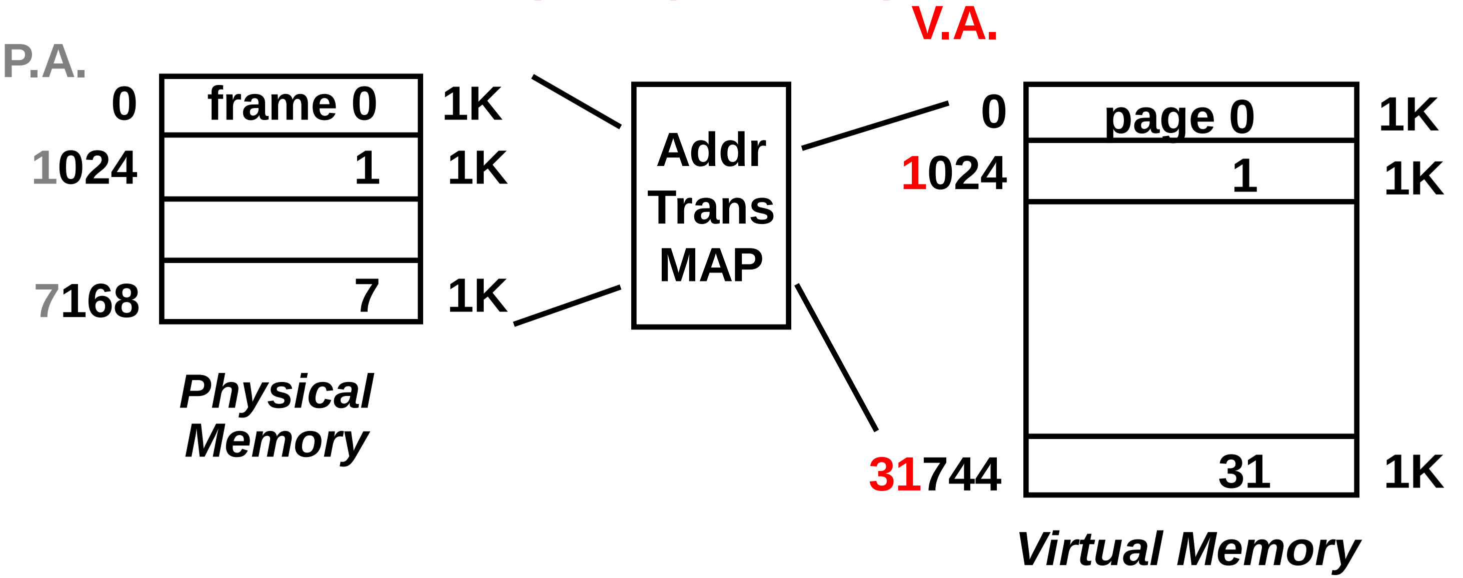
physical address

OS performs this transfer

# Implications of Virtual Memory for Pipeline design

- **Fault?**
- **Address translation?**

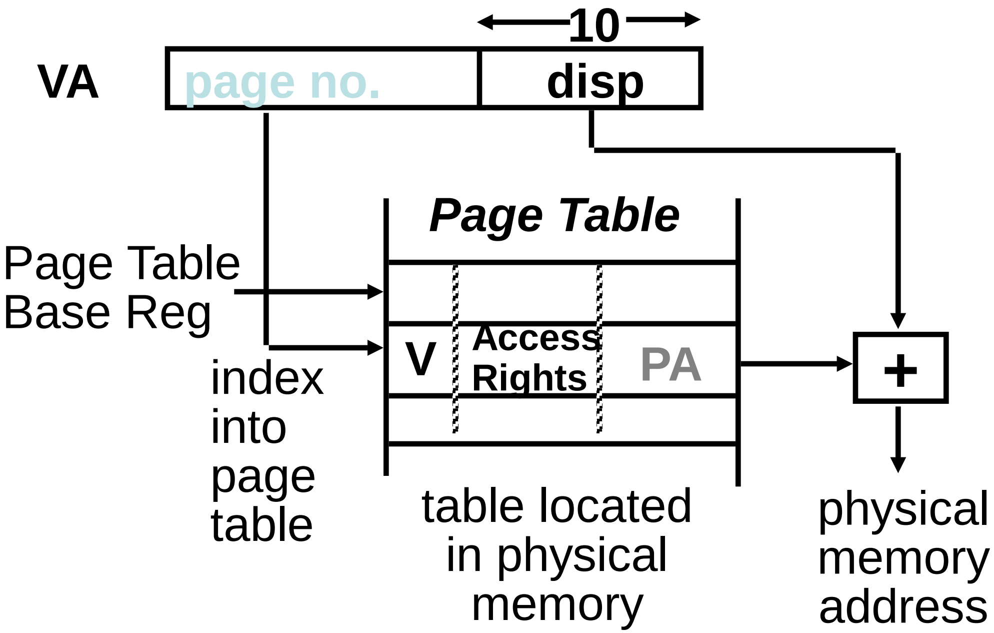
# Paging Organization



unit of mapping

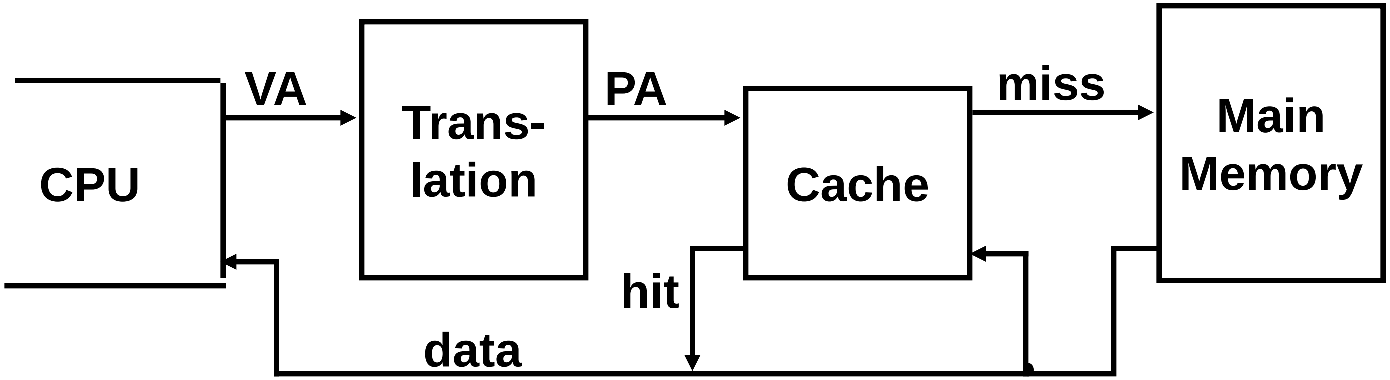
also unit of transfer from virtual to physical memory

## Address Mapping



**actually, concatenation is more likely**

# Address Translation



- Page table is a large data structure in memory
- Two memory accesses for every load, store, or instruction fetch!!!
- Virtually addressed cache?
  - synonym problem
- Cache the address translations?

# 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 |
|---|---|---|---|---|---|
| | | | | | |

Really just a cache on the page table mappings

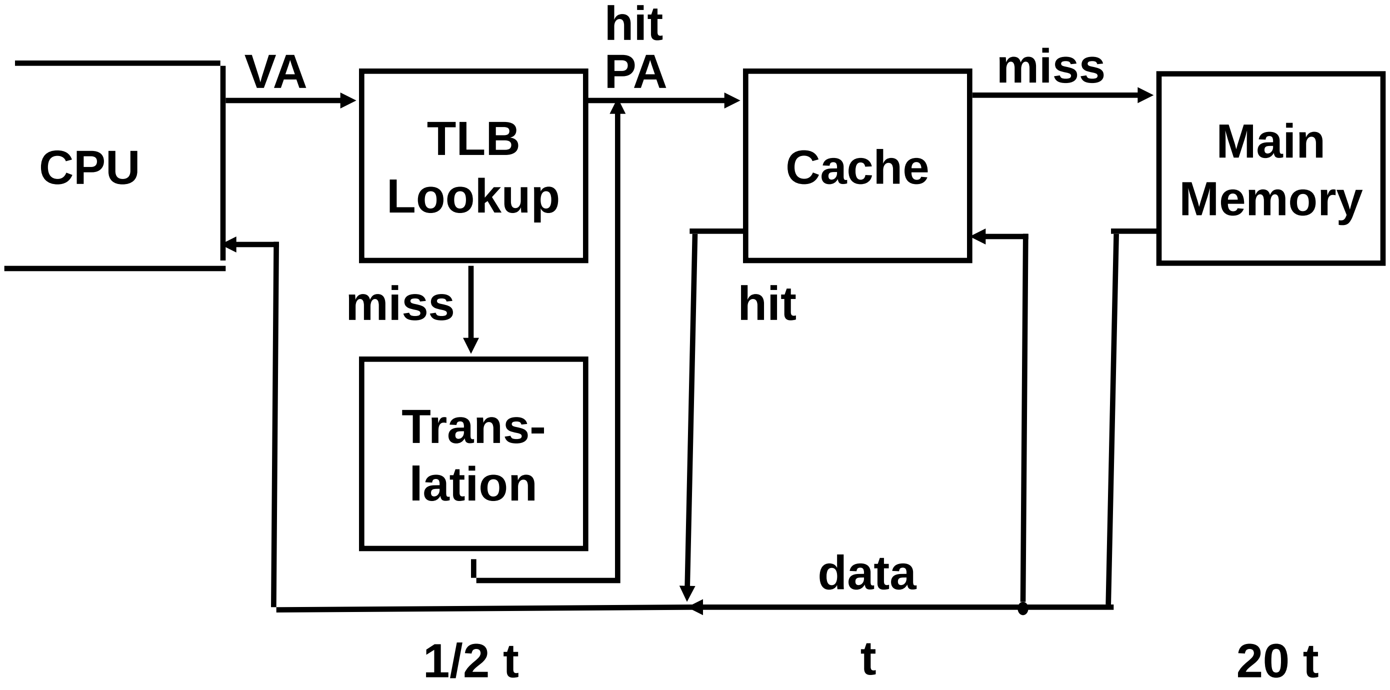
TLB access time comparable to cache access time (much less than main memory access time)

# 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*



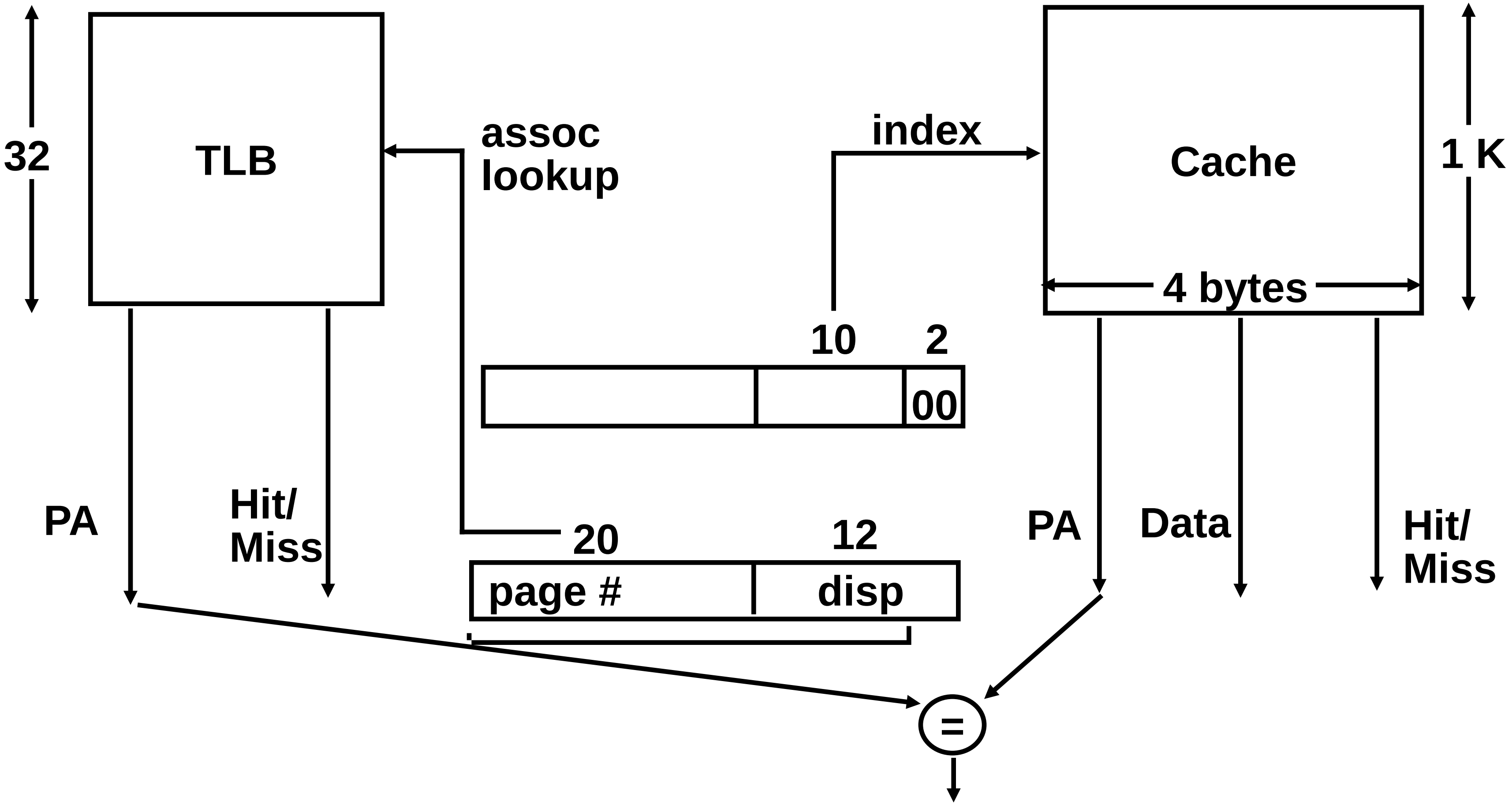
# Reducing Translation Time

Machines with TLBs go one step further to reduce # cycles/cache access

They overlap the cache access with the TLB access:

high order bits of the VA are used to look in the TLB while low order bits are used as index into cache

# Overlapped Cache & TLB Access



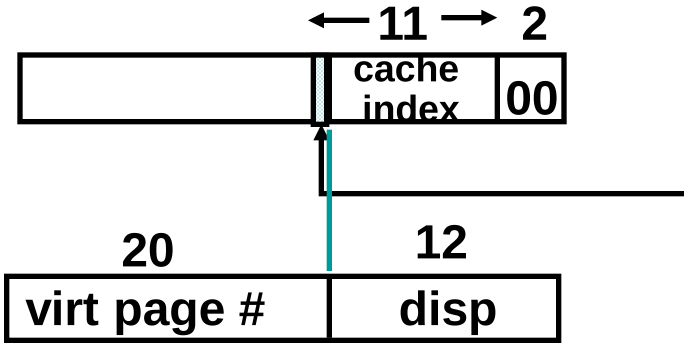
IF cache hit AND (cache tag = PA) then deliver data to CPU
ELSE IF [cache miss OR (cache tag = PA)] and TLB hit THEN
access memory with the PA from the TLB
ELSE do standard VA translation

# Problems With Overlapped TLB Access

Overlapped access only works as long as the address bits used to index into the cache *do not change* as the result of VA translation

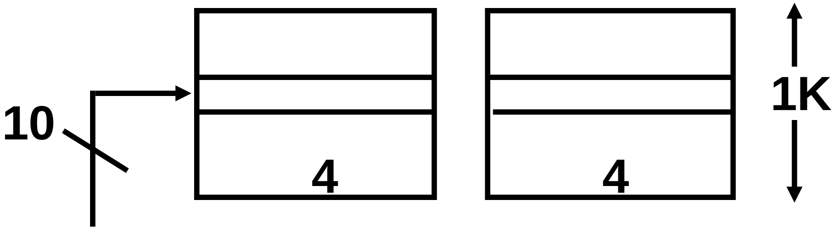
This usually limits things to small caches, large page sizes, or high n-way set associative caches if you want a large cache

Example: suppose everything the same except that the cache is increased to 8 K bytes instead of 4 K:



This bit is changed by VA translation, but is needed for cache lookup

Solutions:
go to 8K byte page sizes;
go to 2 way set associative cache; or
SW guarantee VA[13]=PA[13]



2 way set assoc cache

# Summary #1/4: Pipelining & Performance

- Just overlap tasks; easy if tasks are independent
- Speed Up ≤ Pipeline Depth; if ideal CPI is 1, then:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}}$$

- Hazards limit performance on computers:
  - Structural: need more HW resources
  - Data (RAW,WAR,WAW): need forwarding, compiler scheduling
  - Control: delayed branch, prediction
- Time is measure of performance: latency or throughput
- CPI Law:

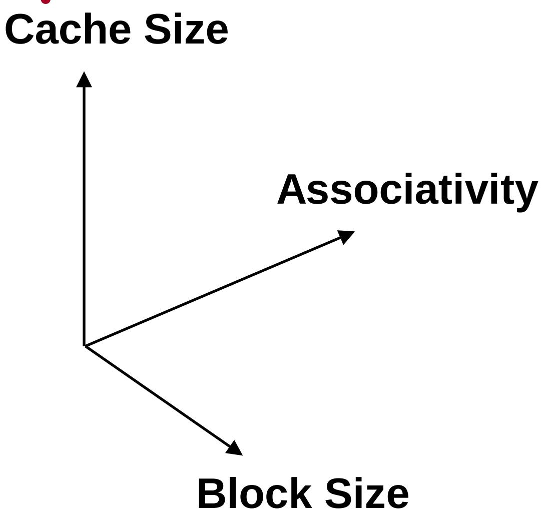
$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

# Summary #2/4: Caches

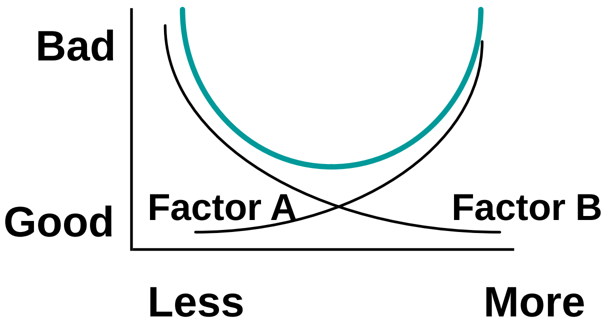
- **The Principle of Locality:**
  - **Program access a relatively small portion of the address space at any instant of time.**
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- **Three Major Categories of Cache Misses:**
  - **Compulsory Misses: sad facts of life. Example: cold start misses.**
  - **Capacity Misses: increase cache size**
  - **Conflict Misses: increase cache size and/or associativity.**
- **Write Policy:**
  - **Write Through: needs a write buffer.**
  - **Write Back: control can be complex**
- **Today CPU time is a function of (ops, cache misses) vs. just f(ops): What does this mean to Compilers, Data structures, Algorithms?**

# Summary #3/4: The Cache Design Space

- **Several interacting dimensions**
  - **cache size**
  - **block size**
  - **associativity**
  - **replacement policy**
  - **write-through vs write-back**



- **The optimal choice is a compromise**
  - **depends on access characteristics**
    - workload
    - use (I-cache, D-cache, TLB)
  - **depends on technology / cost**
- **Simplicity often wins**



# Review #4/4: 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?**
- **Page tables map virtual address to physical address**
- **TLBs make virtual memory practical**
  - **Locality in data => locality in addresses of data, temporal and spatial**
- **TLB misses are significant in processor performance**
  - **funny times, as most systems can't access all of 2nd level cache without TLB misses!**
- **Today VM allows many processes to share single memory without having to swap all processes to disk; today VM protection is more important than memory hierarchy**