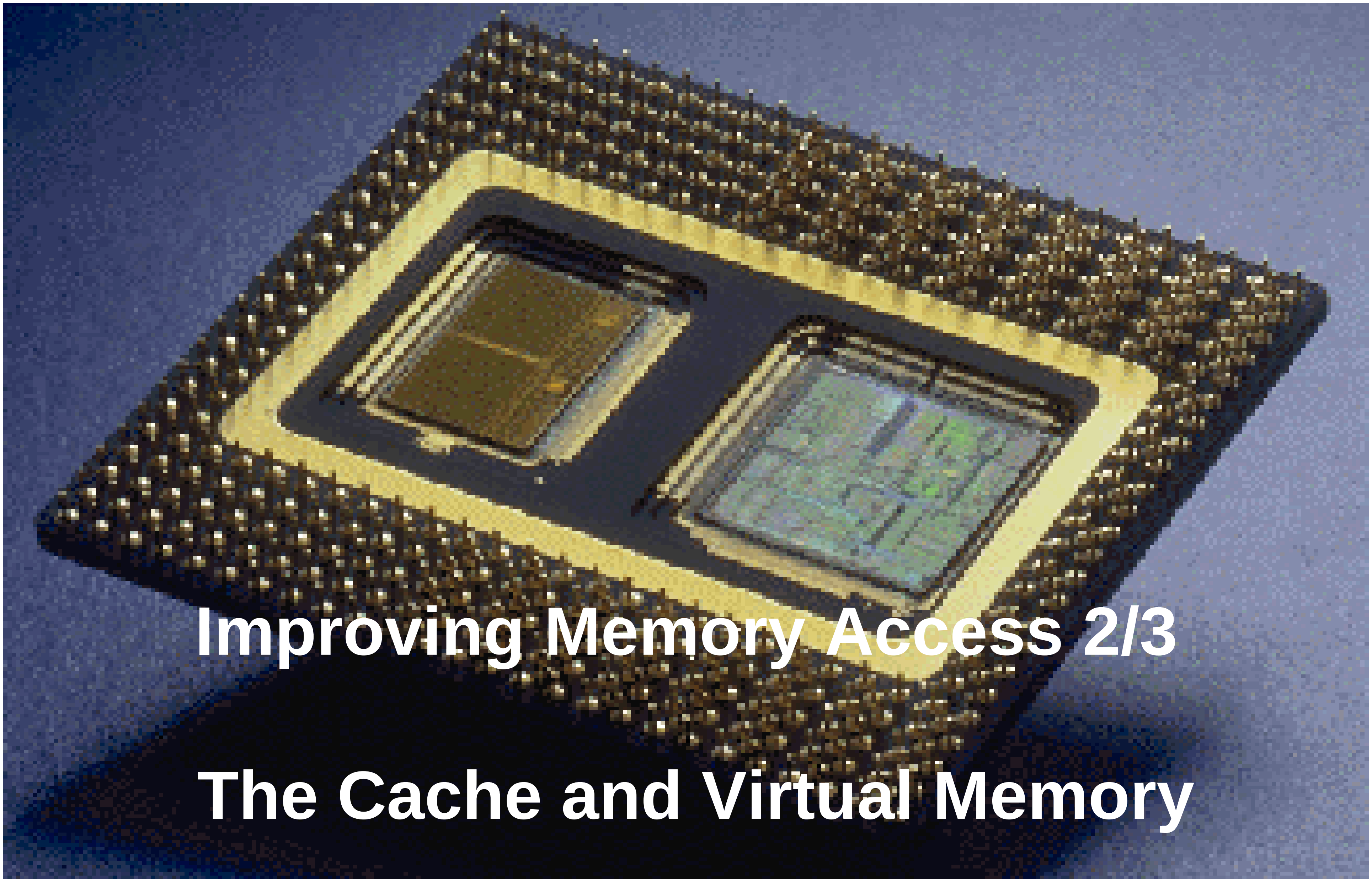
EECS 322 Computer Architecture

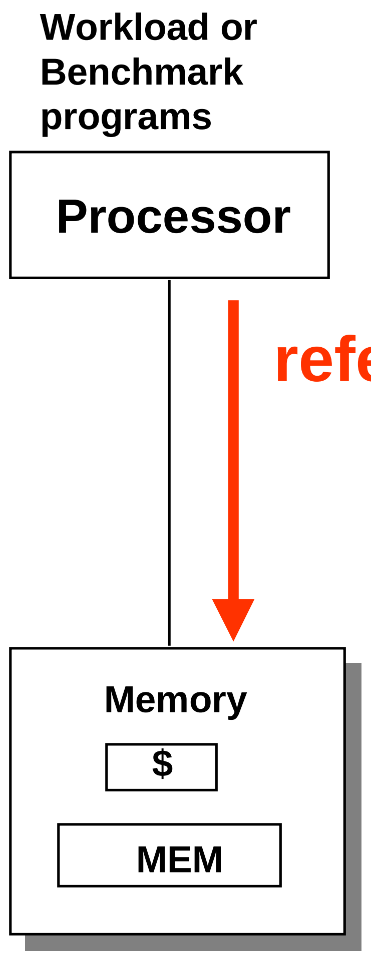


Improving Memory Access 2/3

The Cache and Virtual Memory

The Art of Memory System Design

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



reference stream

<op,addr>, <op,addr>,<op,addr>,<op,addr>, . . .

op: i-fetch, read, write

Principle of Locality

- Principle of Locality

  states that programs access a relatively small portion of their address space at <u>any instance of time</u>

- <u>Two types of locality</u>

  - <u>Temporal locality</u> (locality in time)

    If an item is referenced, then

    <u>the same</u> item will tend to be referenced soon

    "the tendency to reuse recently accessed data items"

  - <u>Spatial locality</u> (locality in space)

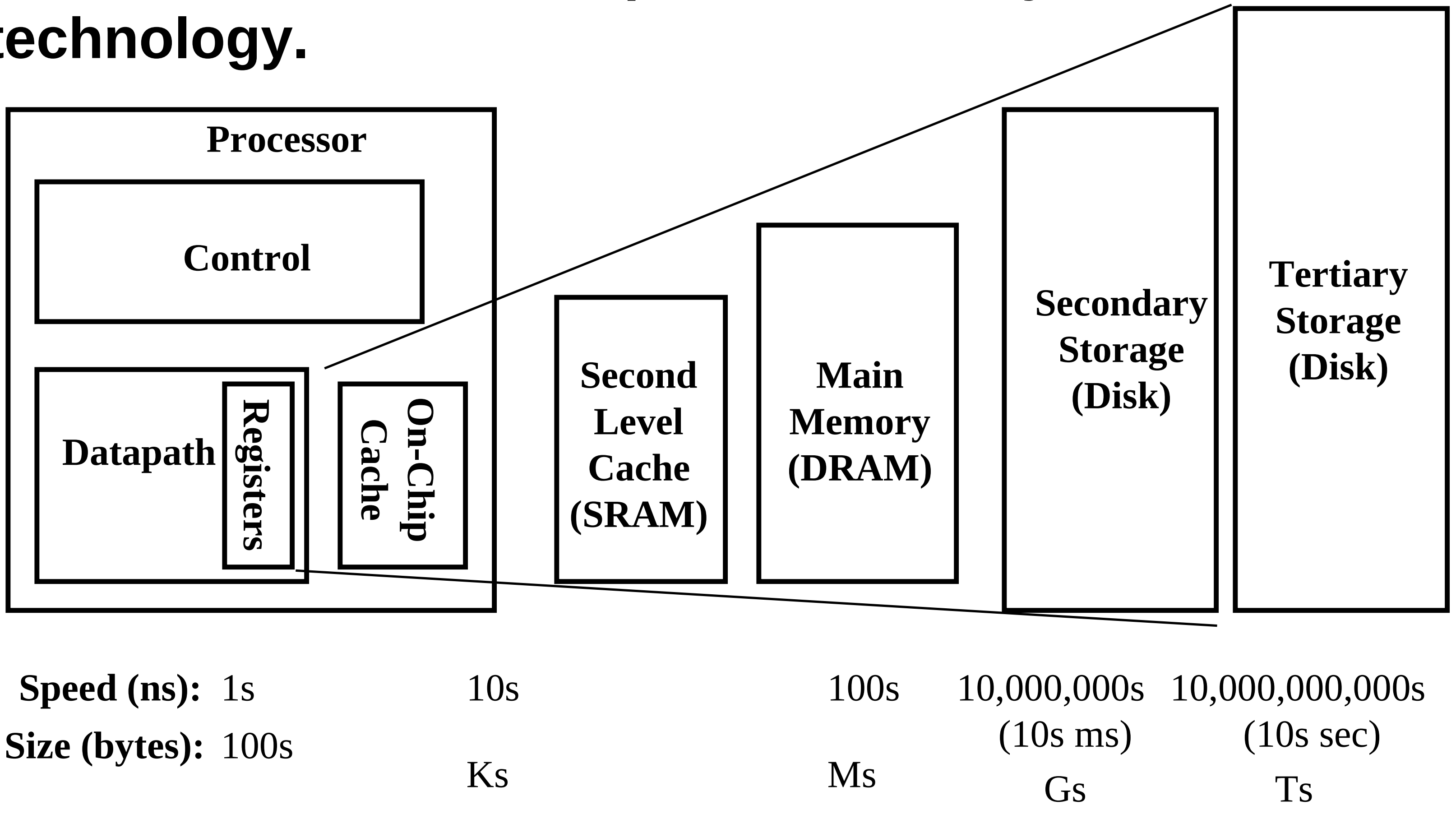
    If an item is referenced, then

    <u>nearby</u> items will be referenced soon

    "the tendency to reference nearby data items"

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.**



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.
- 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.

Spatial Locality

- Temporal only cache

  cache block contains only one word (No spatial locality).

- Spatial locality

  Cache block contains multiple words.

  - When a miss occurs, then fetch multiple words.
  - Advantage

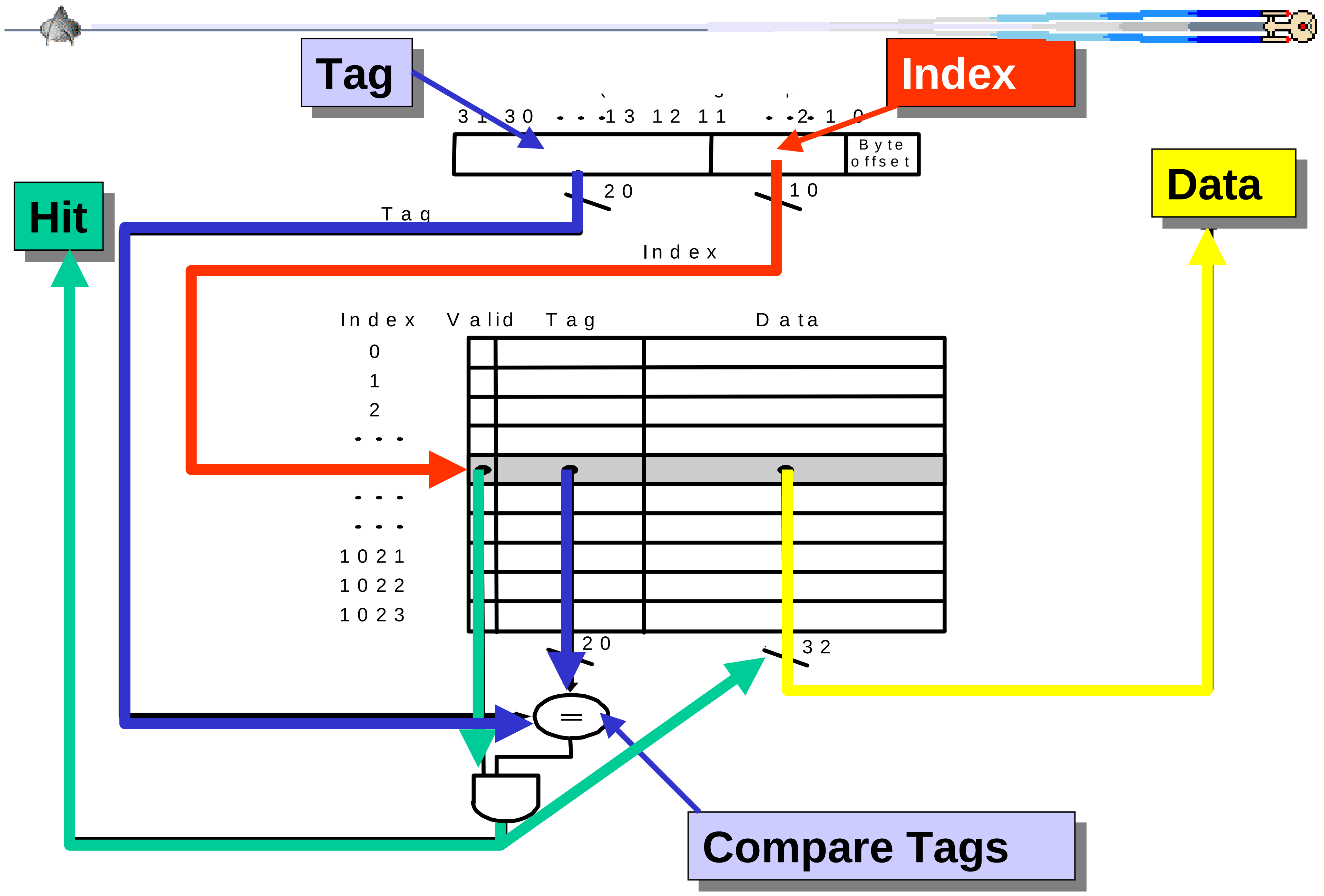
    Hit ratio increases because there is a high probability that the adjacent words will be needed shortly.

  - Disadvantage

    Miss penalty increases with block size

Figure 7.7

Direct Mapped Cache: Mips Architecture



Cache schemes

write-through cache

Always write the data into both the cache and memory and then wait for memory.

write buffer

write data into cache and write buffer.
If write buffer full processor must stall.

No amount of buffering can help if writes are being generated faster than the memory system can accept them.

write-back cache

Write data into the cache block and only write to memory when block is modified but complex to implement in hardware.

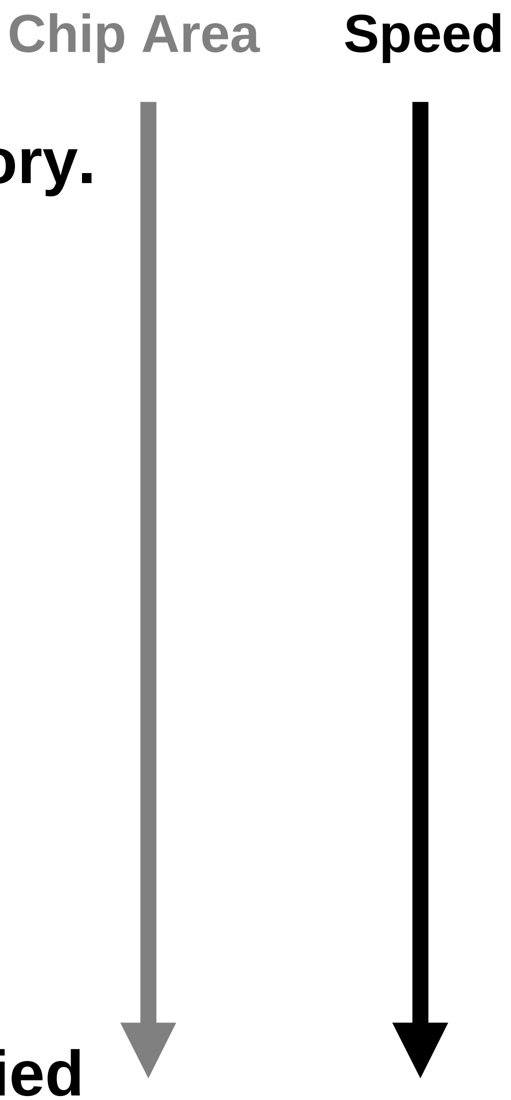
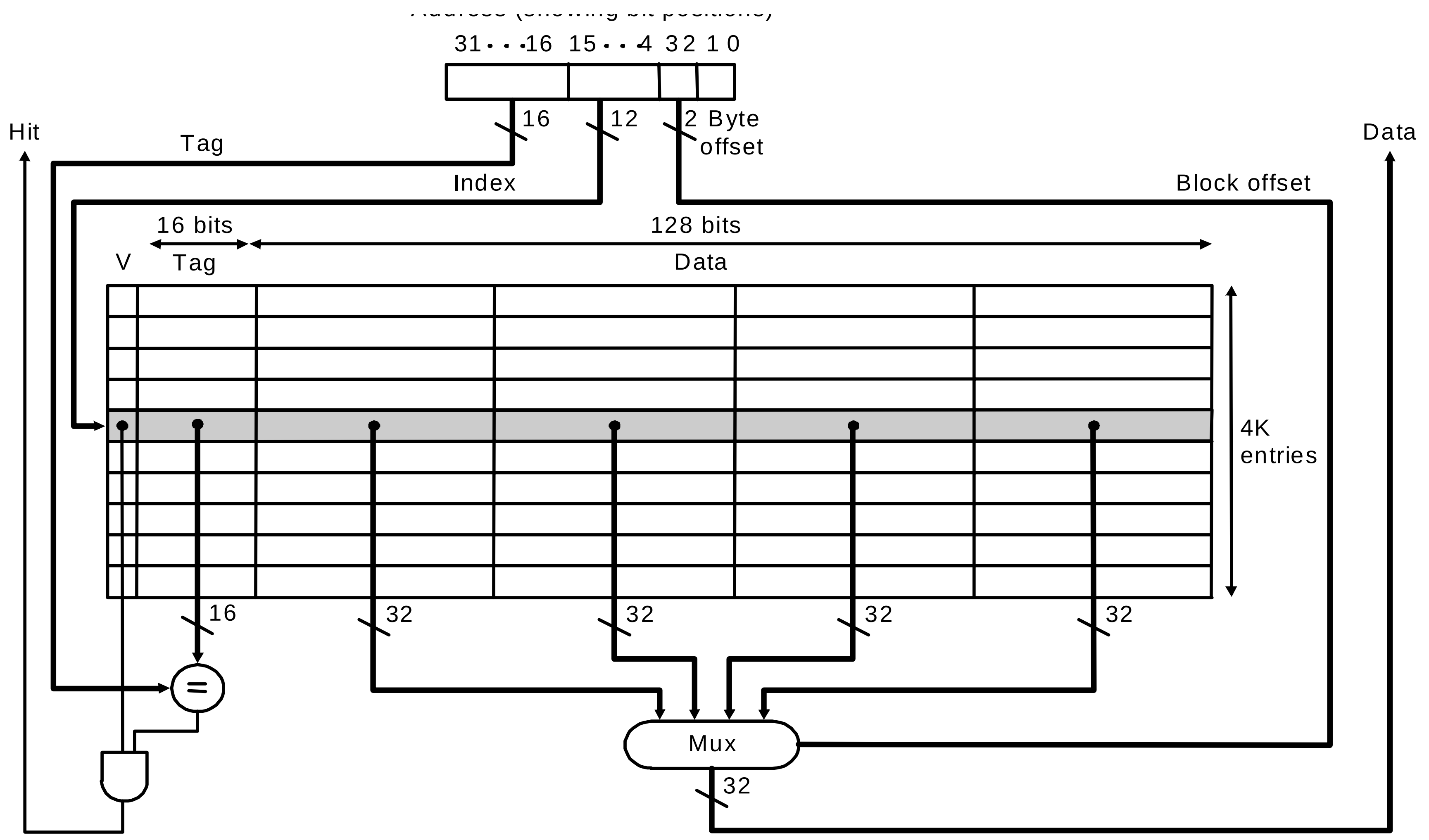


Figure 7.10

Spatial Locality: 64 KB cache, 4 words

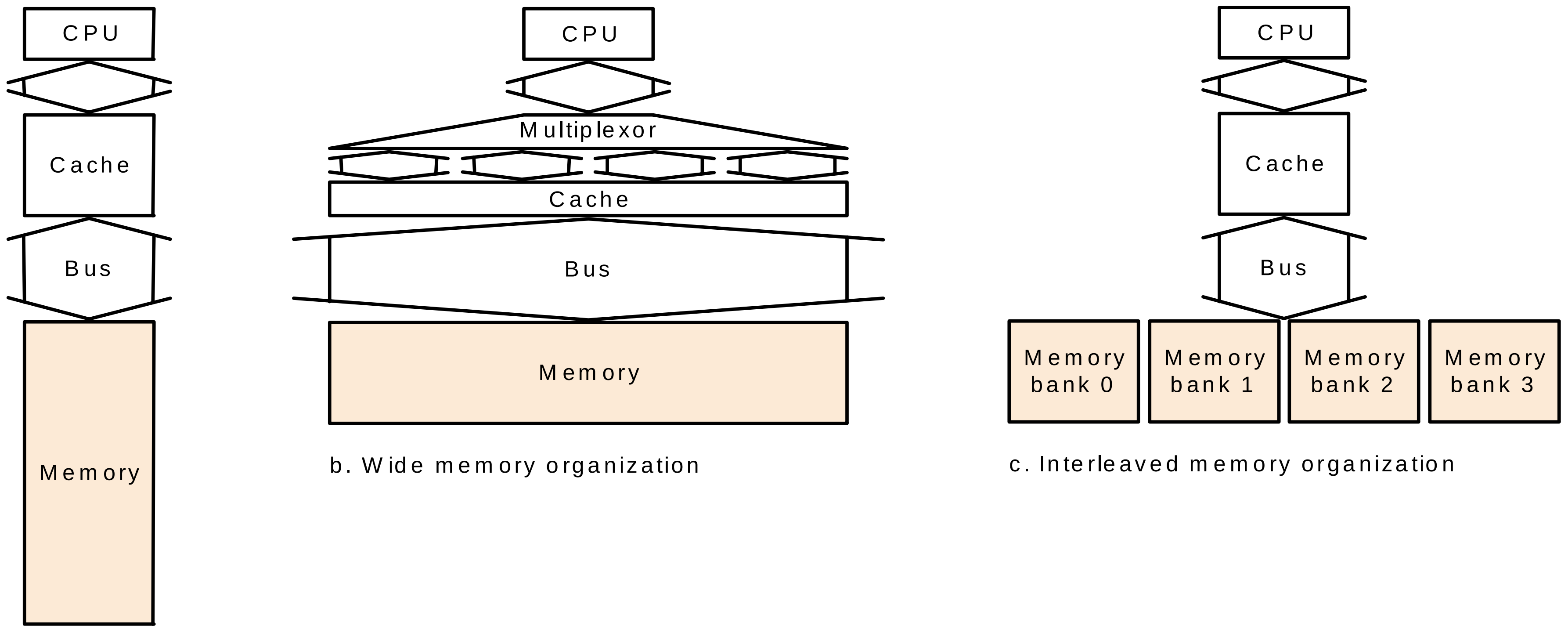
- 64KB cache using four-word (16-byte word)
- 16 bit tag, 12 bit index, 2 bit block offset, 2 bit byte offset.



Designing the Memory System

Figure 7.13

- **Make reading multiple words easier by using banks of memory**



b. Wide memory organization

c. Interleaved memory organization

a. One-word-wide memory organization

Memory organizations

Figure 7.13

One word wide memory organization

Advantage

Easy to implement, low hardware overhead

Disadvantage

Slow: 0.25 bytes/clock transfer rate

Interleave memory organization

Advantage

Better: 0.80 bytes/clock transfer rate

Banks are valuable on writes: independently

Disadvantage

more complex bus hardware

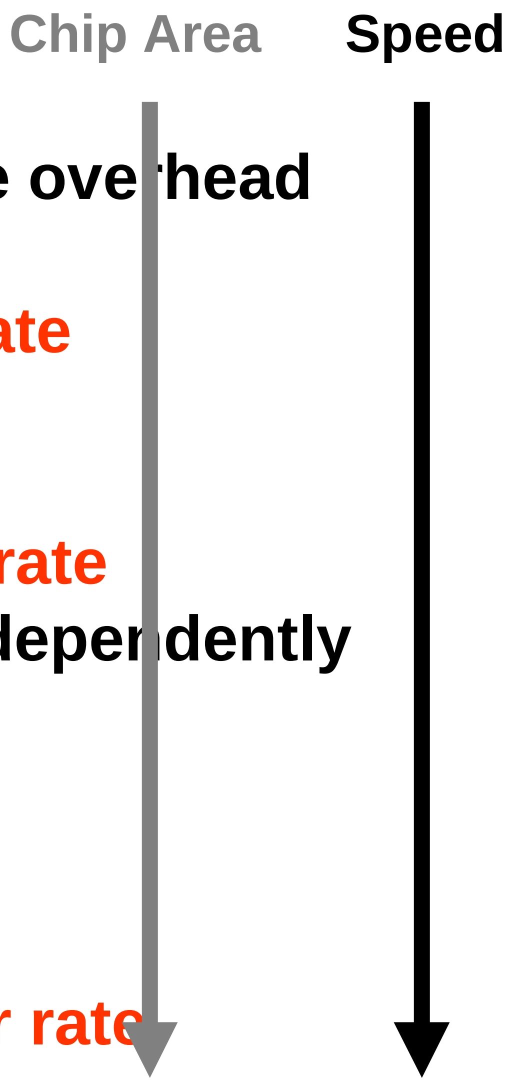
Wide memory organization

Advantage

Fastest: 0.94 bytes/clock transfer rate

Disadvantage

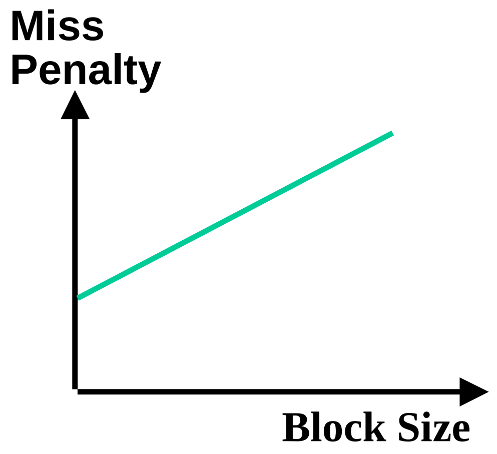
Wider bus and increase in cache access time

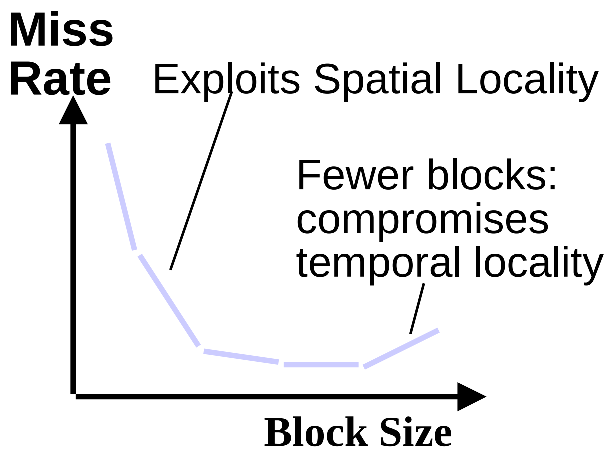


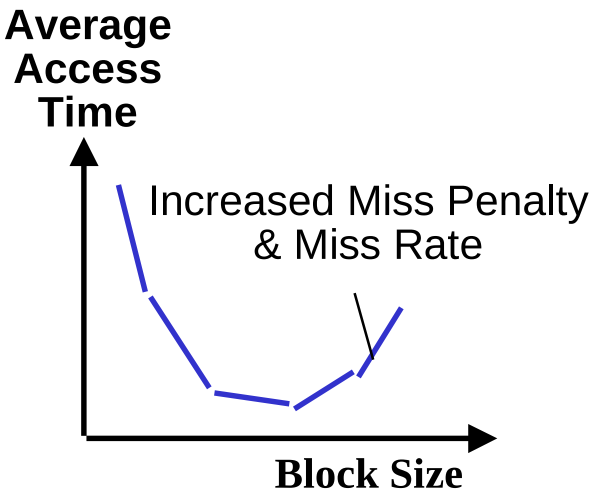
Block Size Tradeoff

- 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

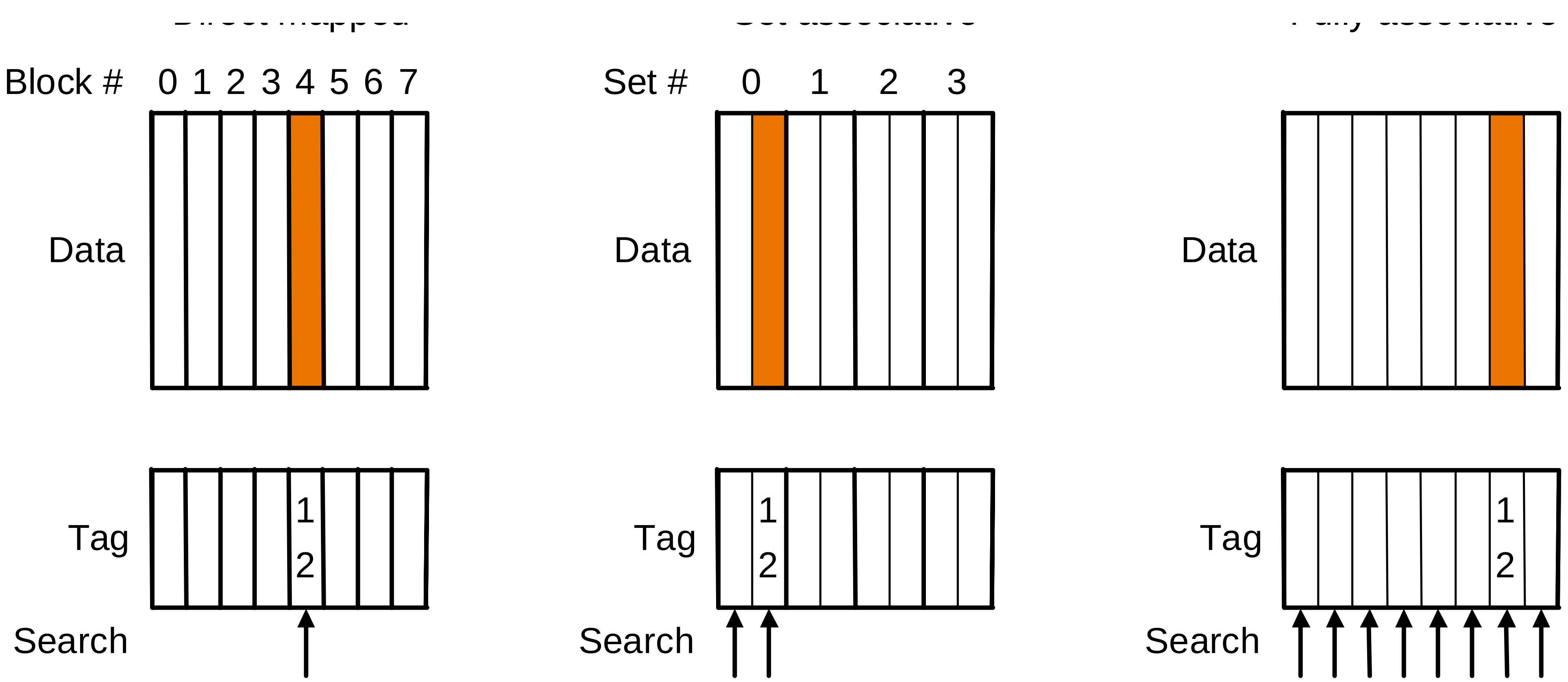






Cache associativity

Figure 7.15



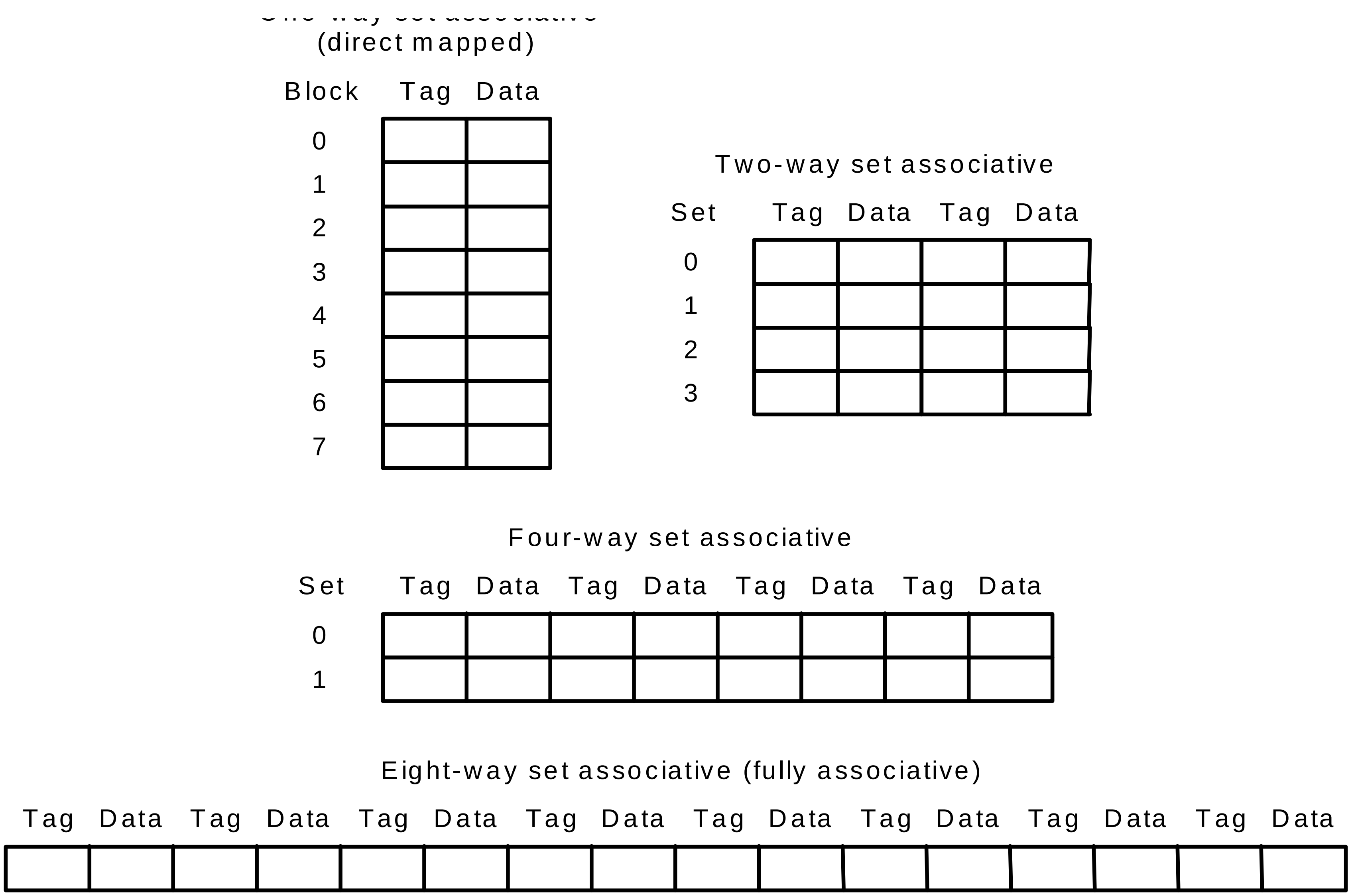
Direct-mapped cache

2-way set associative cache

Fully associative cache

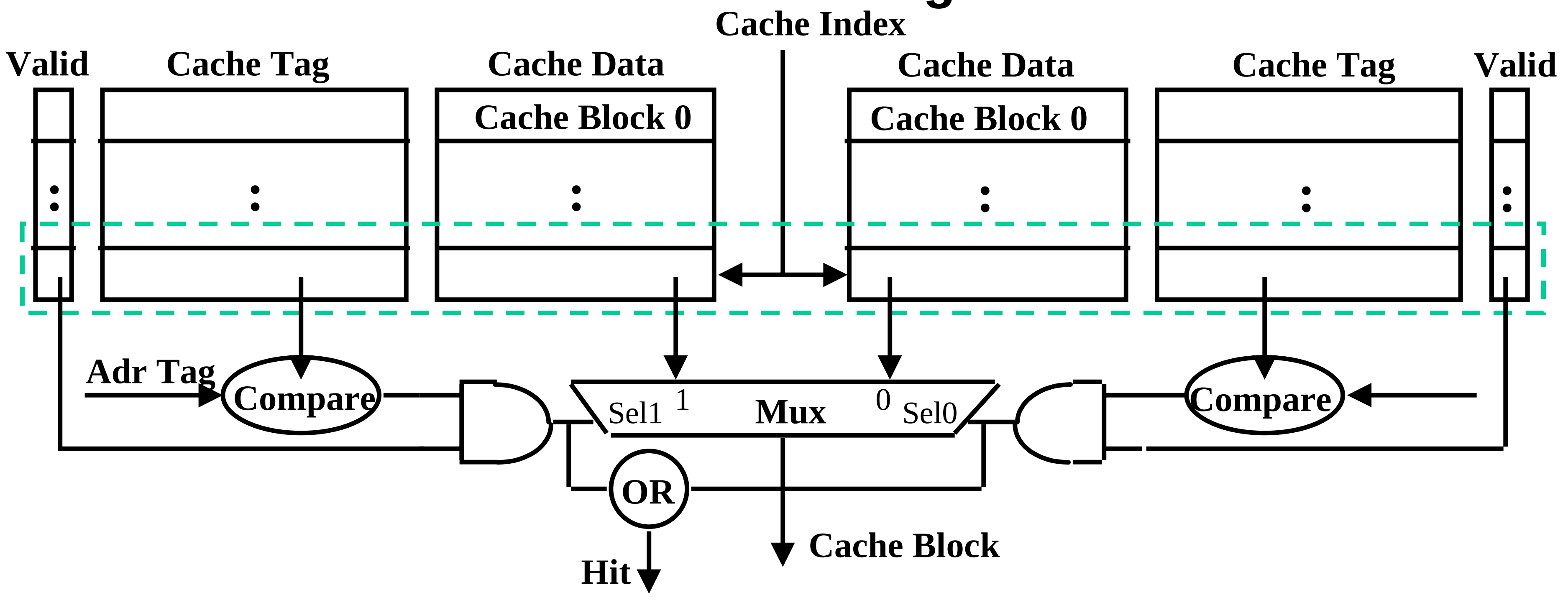
Cache associativity

Figure 7.16



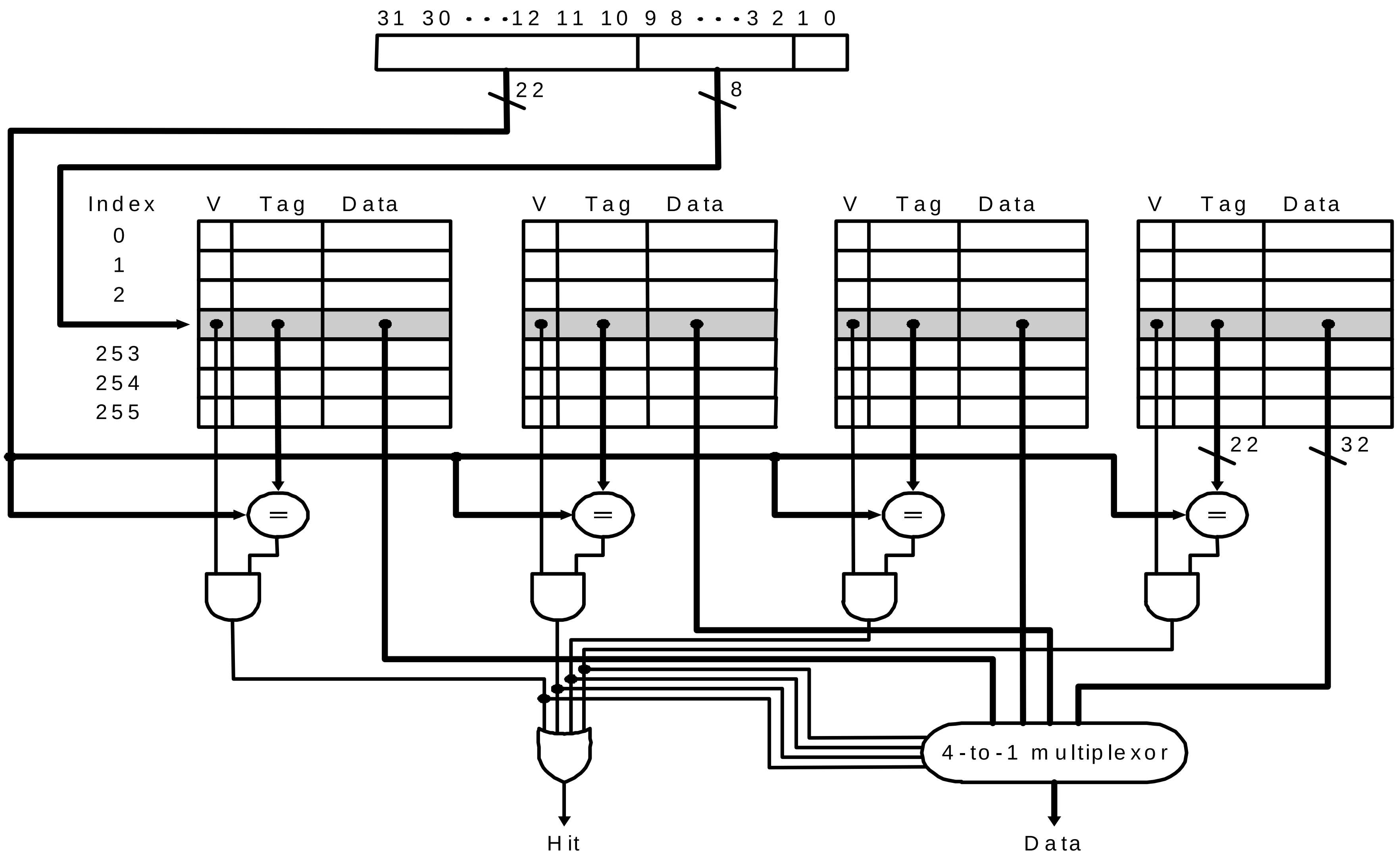
A Two-way Set Associative Cache

- **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



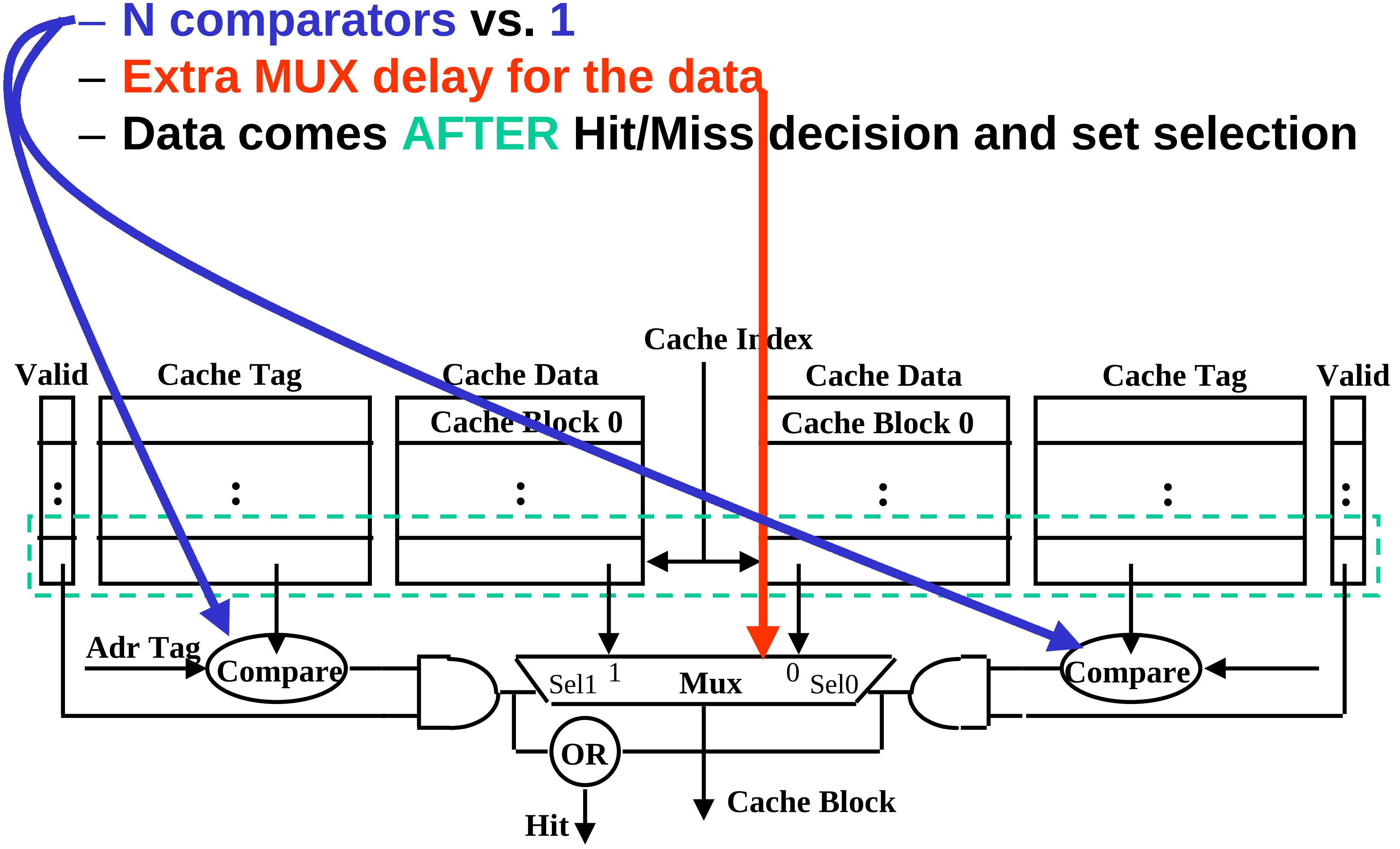
A 4-way set associative implementation

Figure 7.19



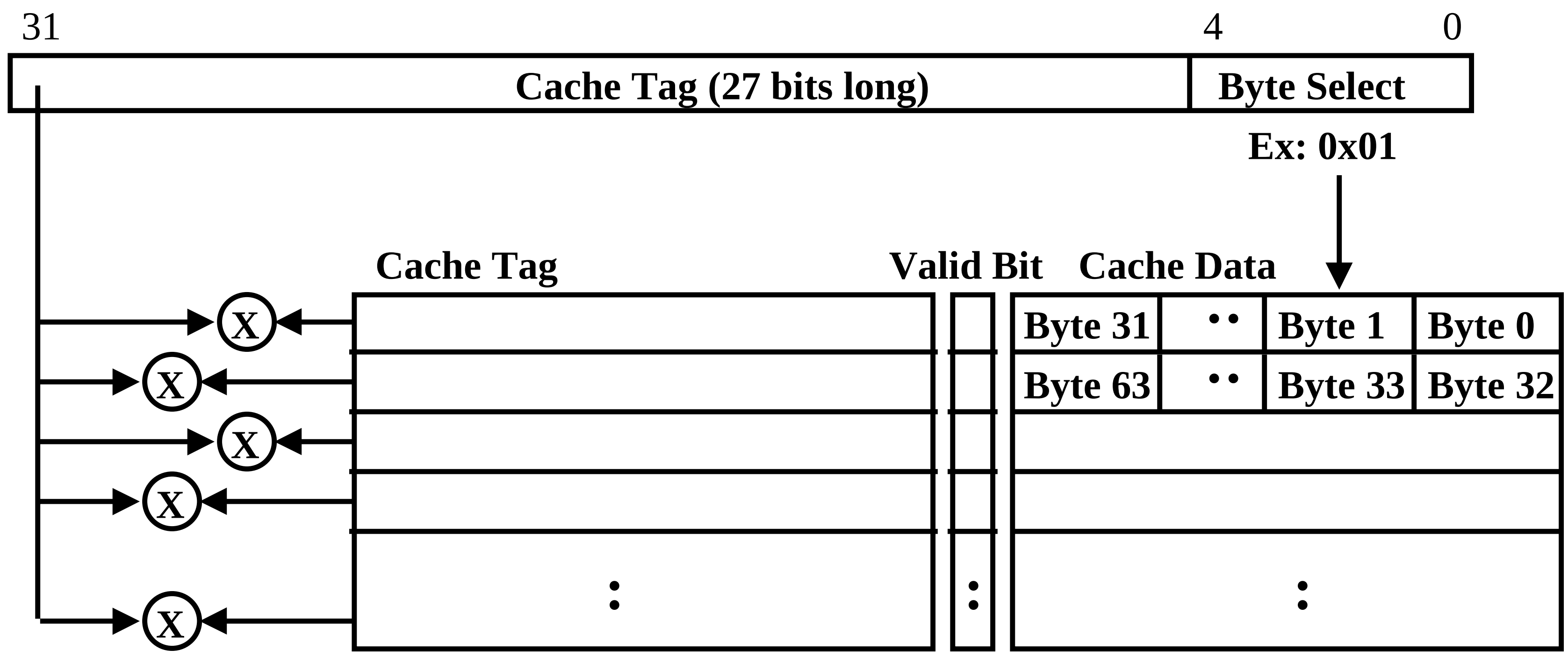
Disadvantage of Set Associative Cache

- 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



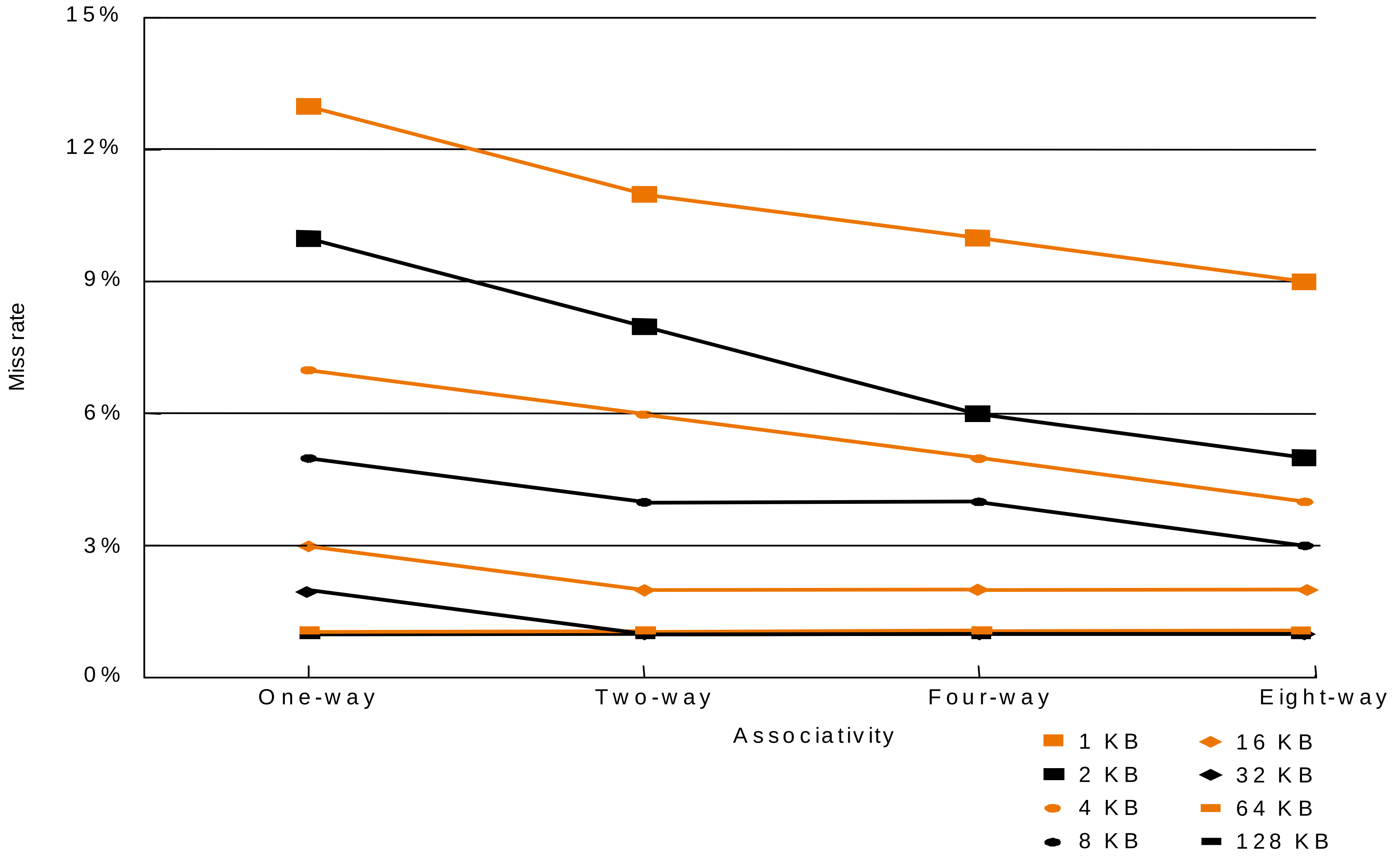
Fully Associative

- **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



Performance

Figure 7.29



Decreasing miss penalty with multilevel caches

- Add a second level cache:
  - often primary cache is on the same chip as the processor
  - use SRAMs to add another cache above primary memory (DRAM)
  - miss penalty goes down if data is in 2nd level cache

- Example:
  - CPI of 1.0 on a 500Mhz machine with a 5% miss rate, 200ns DRAM access
  - Adding 2nd level cache with 20ns access time decreases miss rate to 2%
- Using multilevel caches:
  - try and optimize the hit time on the 1st level cache
  - try and optimize the miss rate on the 2nd level cache

Decreasing miss penalty with multilevel caches

- Add a second level cache:
  - often primary cache is on the same chip as the processor
  - use SRAMs to add another cache above primary memory (DRAM)
  - miss penalty goes down if data is in 2nd level cache

Decreasing miss penalty with multilevel caches

- Example:
  - CPI of 1.0 on a 500Mhz machine with a 5% miss rate, 200ns DRAM access
  - Adding 2nd level cache with 20ns access time decreases miss rate to 2%

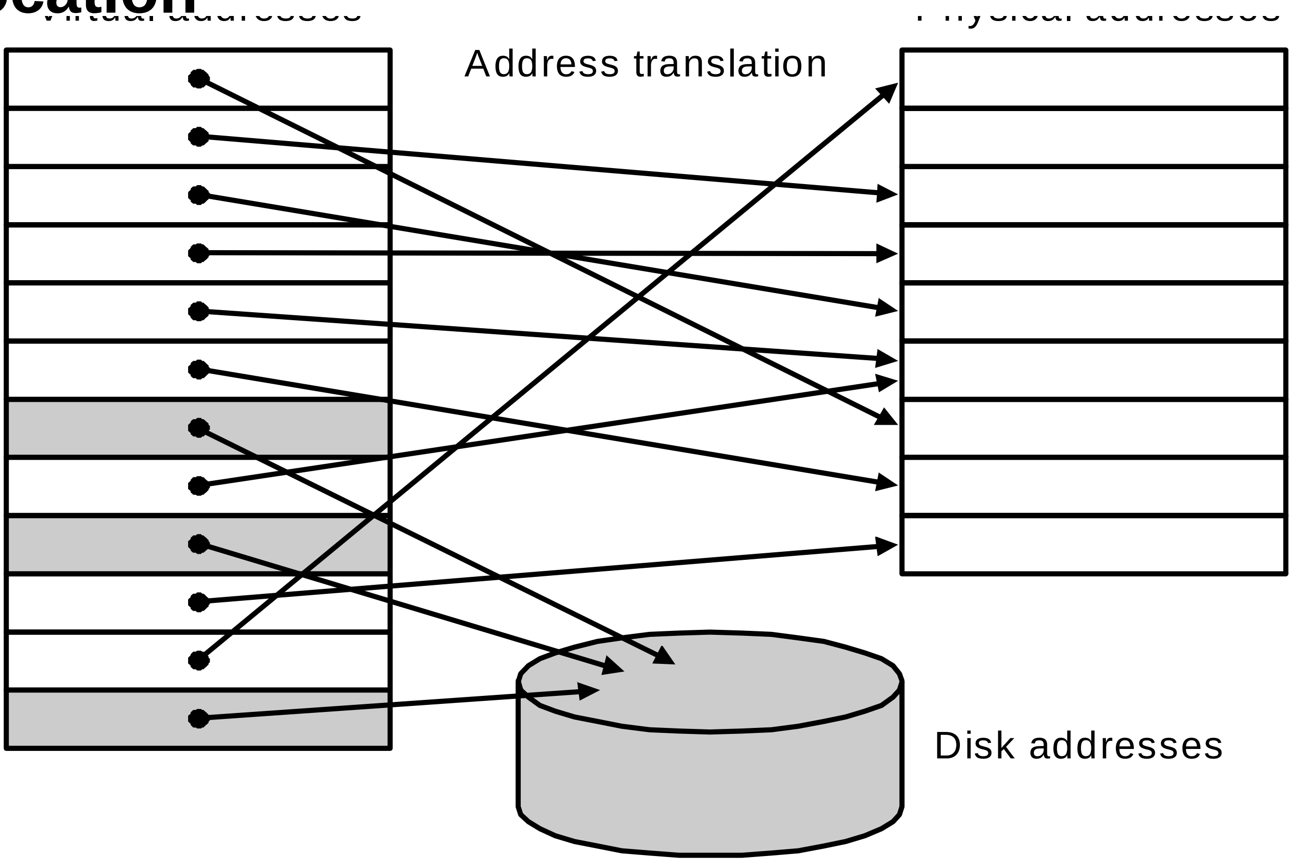
- Using multilevel caches:
  - try and optimize the hit time on the 1st level cache
  - try and optimize the miss rate on the 2nd level cache

A Summary on 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

Virtual Memory

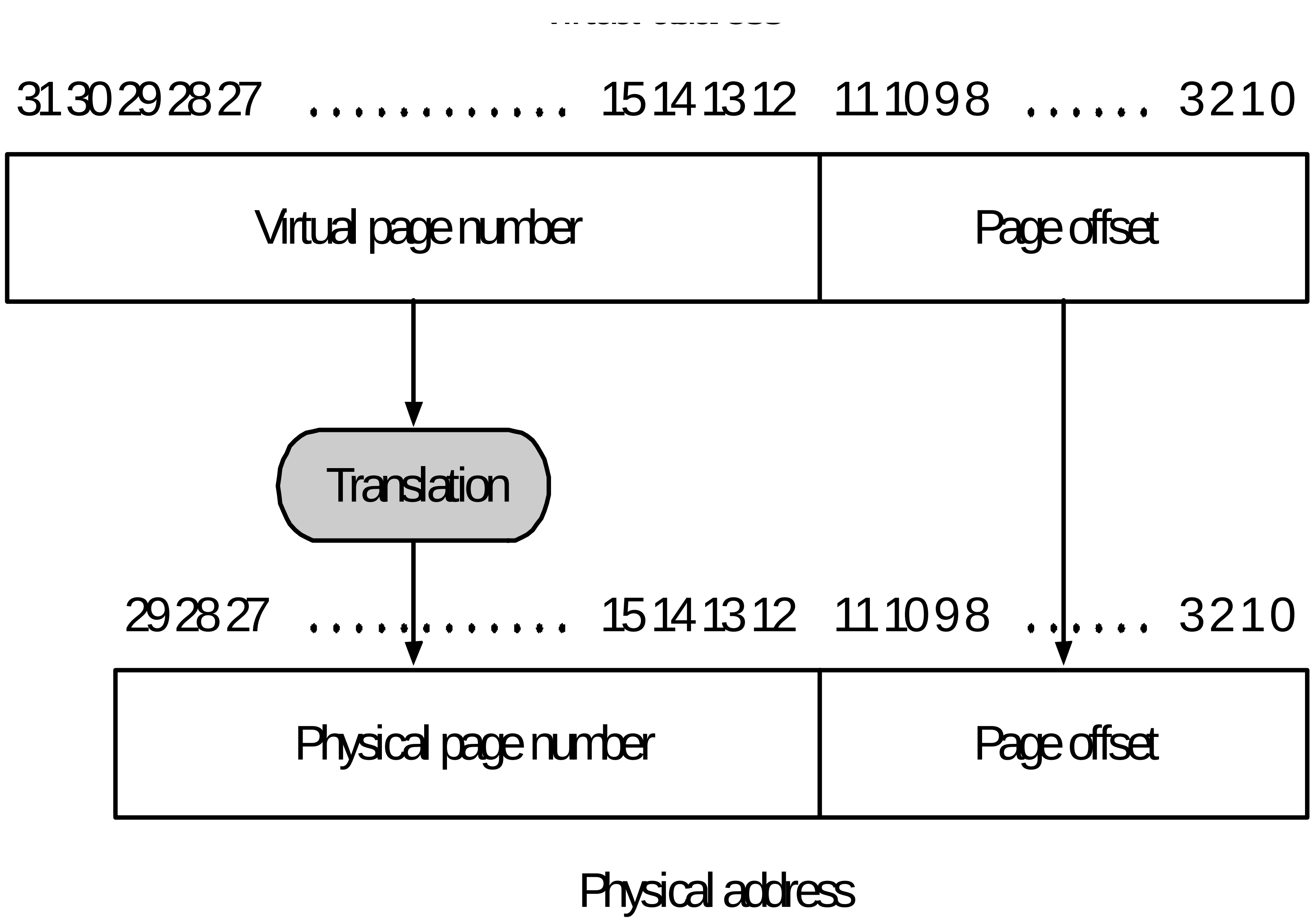
- Main memory can act as a cache for the secondary storage (disk) Advantages:
  - illusion of having more physical memory
  - program relocation
  - protection



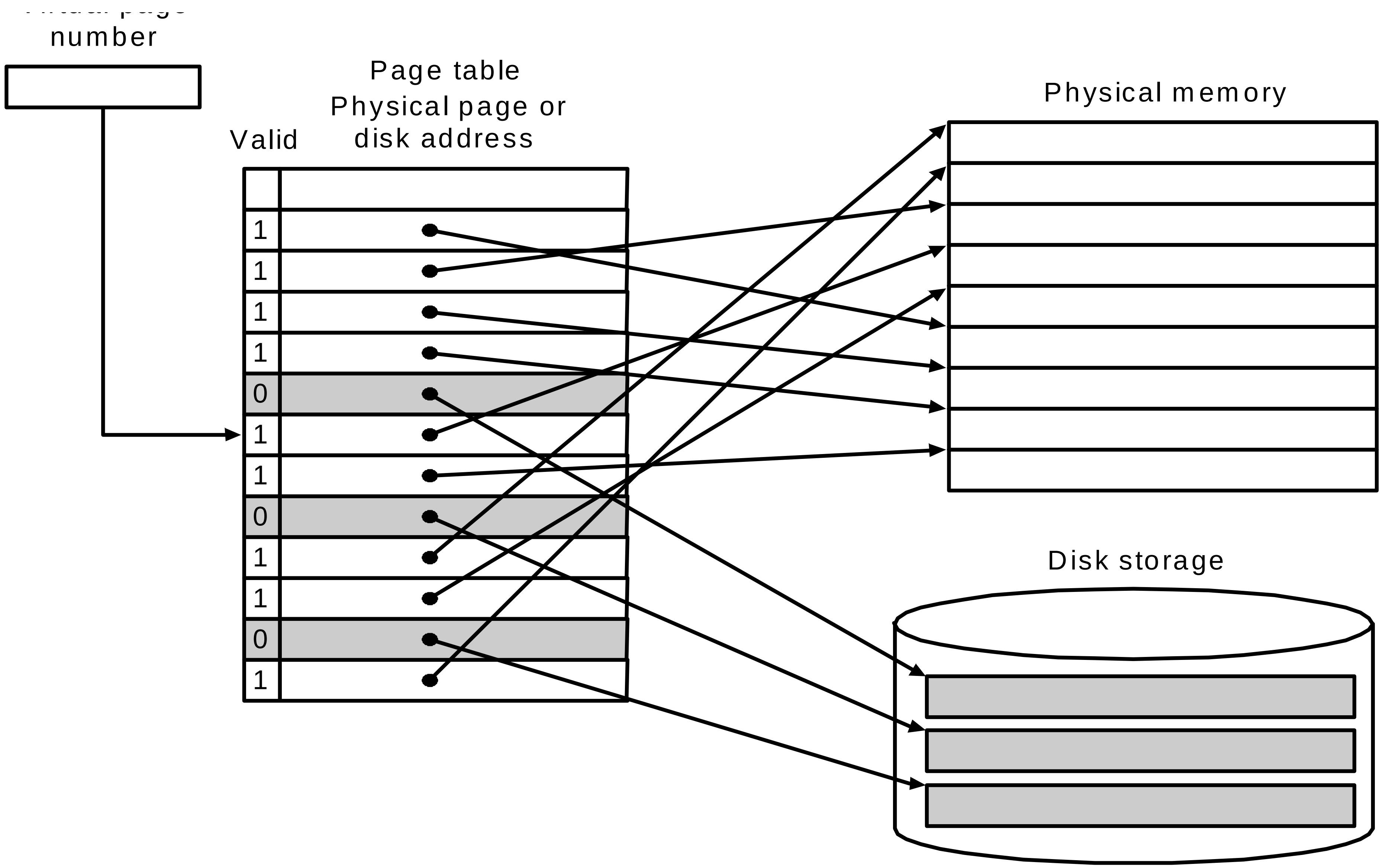
Pages: virtual memory blocks

- **Page faults:** the data is not in memory, retrieve it from disk
  - huge miss penalty, thus pages should be fairly large (e.g., 4KB)
  - reducing page faults is important (LRU is worth the price)
  - can handle the faults in software instead of hardware
  - using write-through is too expensive so we use writeback

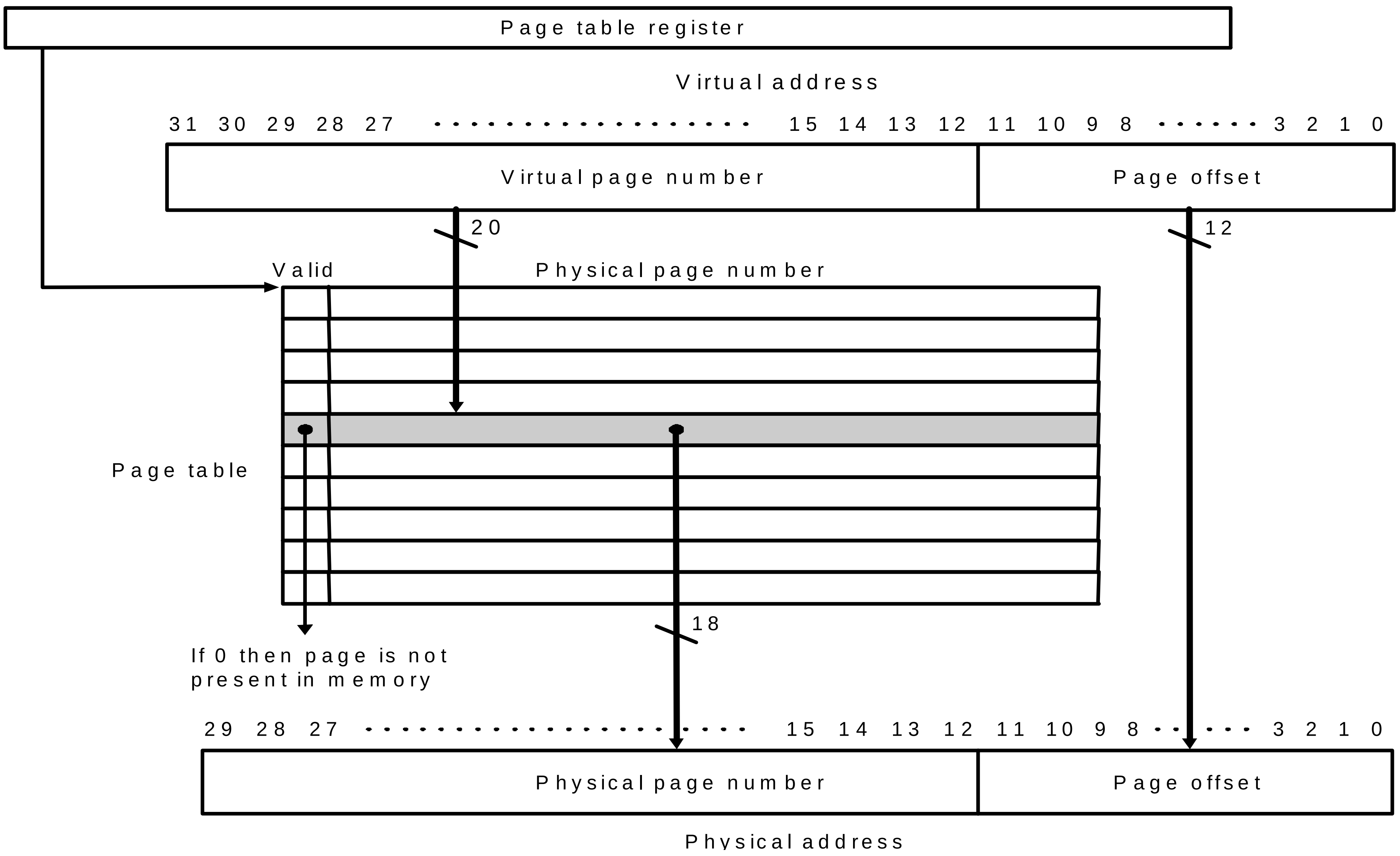
Pages: virtual memory blocks



Page Tables


number
Page table
Physical page or
disk address
Valid
1
1
1
1
0
1
1
0
1
1
0
1
Physical memory
Disk storage

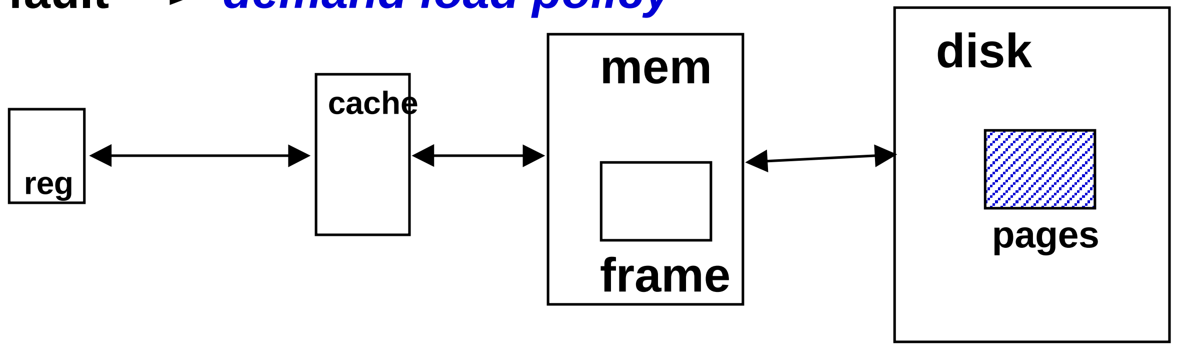
Page Tables



Page table register
Virtual address
31 30 29 28 27 .... 15 14 13 12 11 10 9 8 .... 3 2 1 0
Virtual page number
Page offset
20
12
Valid
Physical page number
Page table
18
If 0 then page is not present in memory
29 28 27 .... 15 14 13 12 11 10 9 8 .... 3 2 1 0
Physical page number
Page offset
Physical address

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*



<u>Paging Organization</u>

virtual and physical address space partitioned into blocks of equal size

page frames

pages

TLBs: Translation Look-Aside Buffers

A way to speed up translation is to use a special cache of recently used page table entries

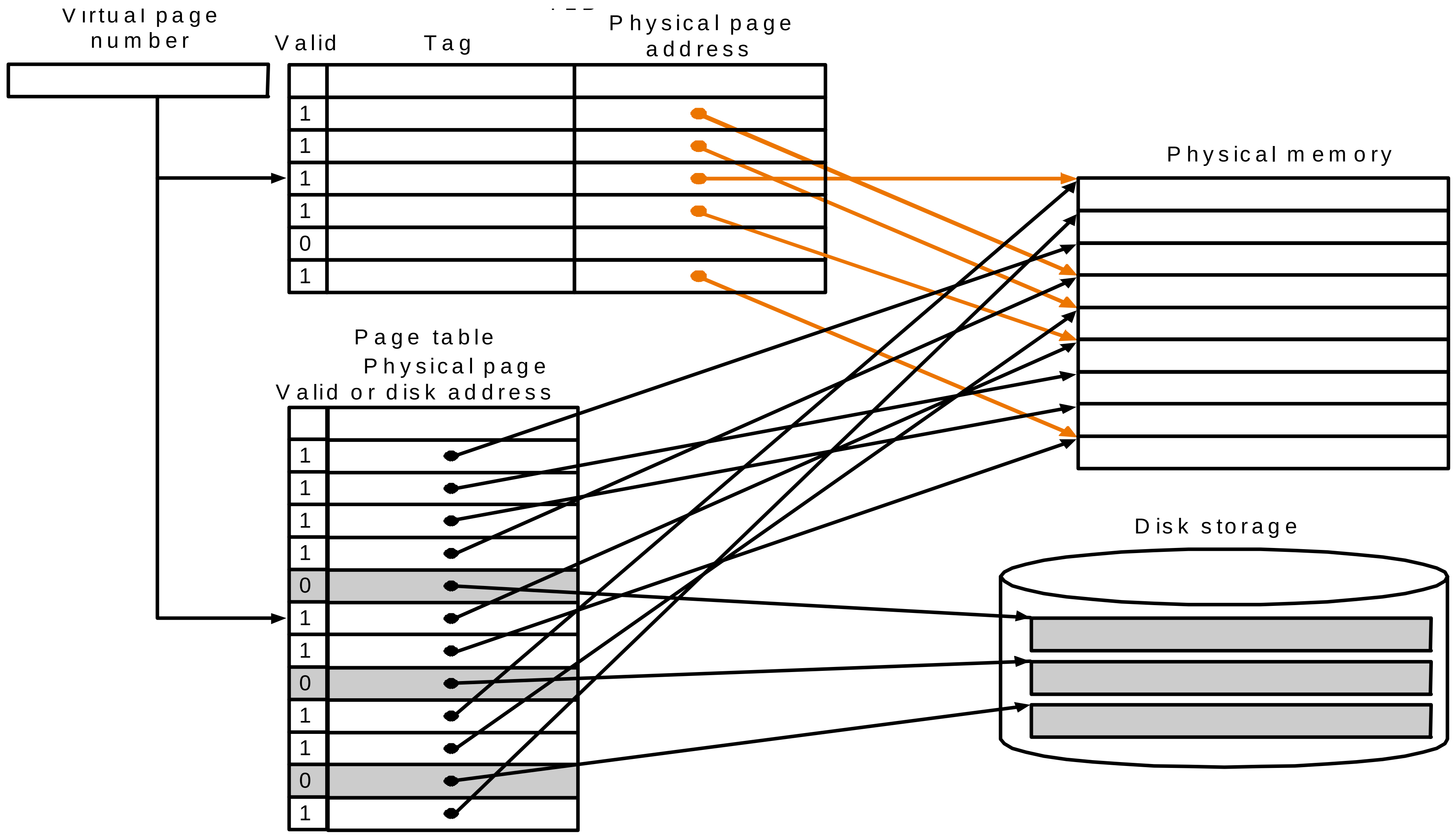
-- this has many names, but the most frequently used is *Translation Lookaside Buffer* or *TLB*

Virtual Address	Physical Address	Dirty	Ref	Valid	Access

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

Making Address Translation Fast

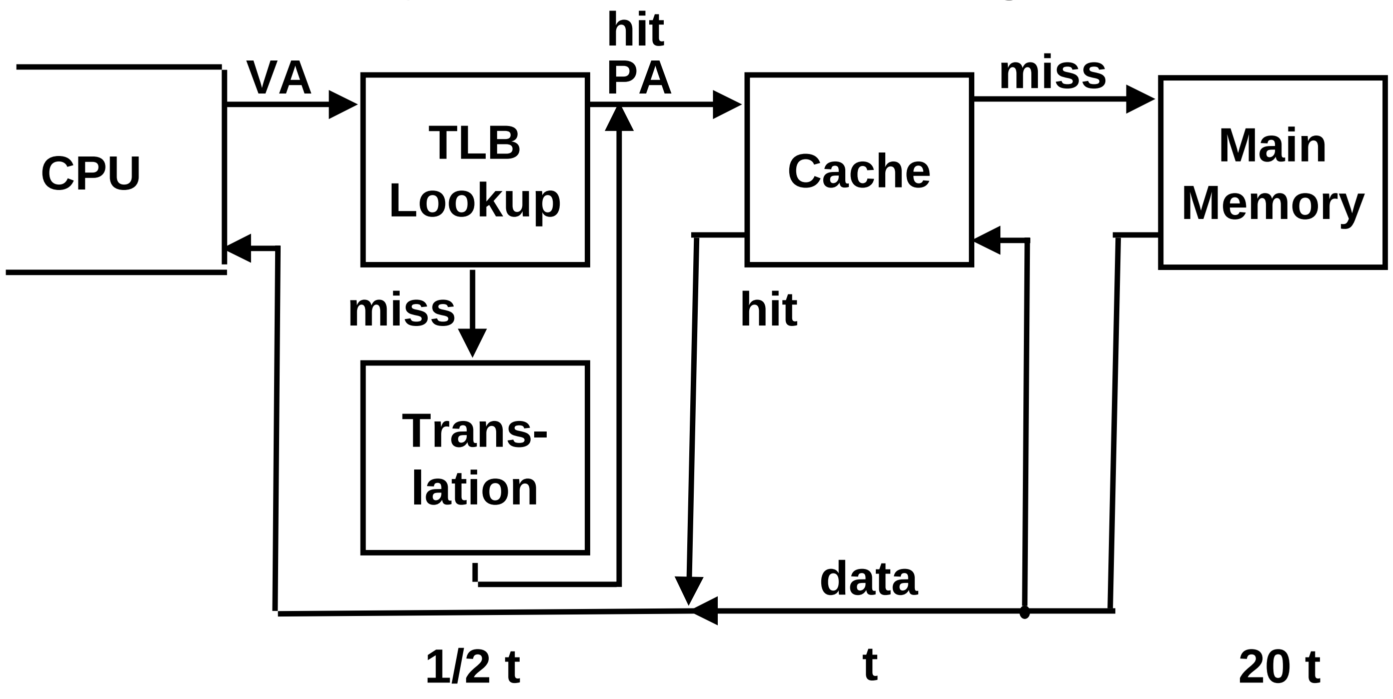
- A cache for address translations: translation lookaside buffer



Translation Look-Aside Buffers

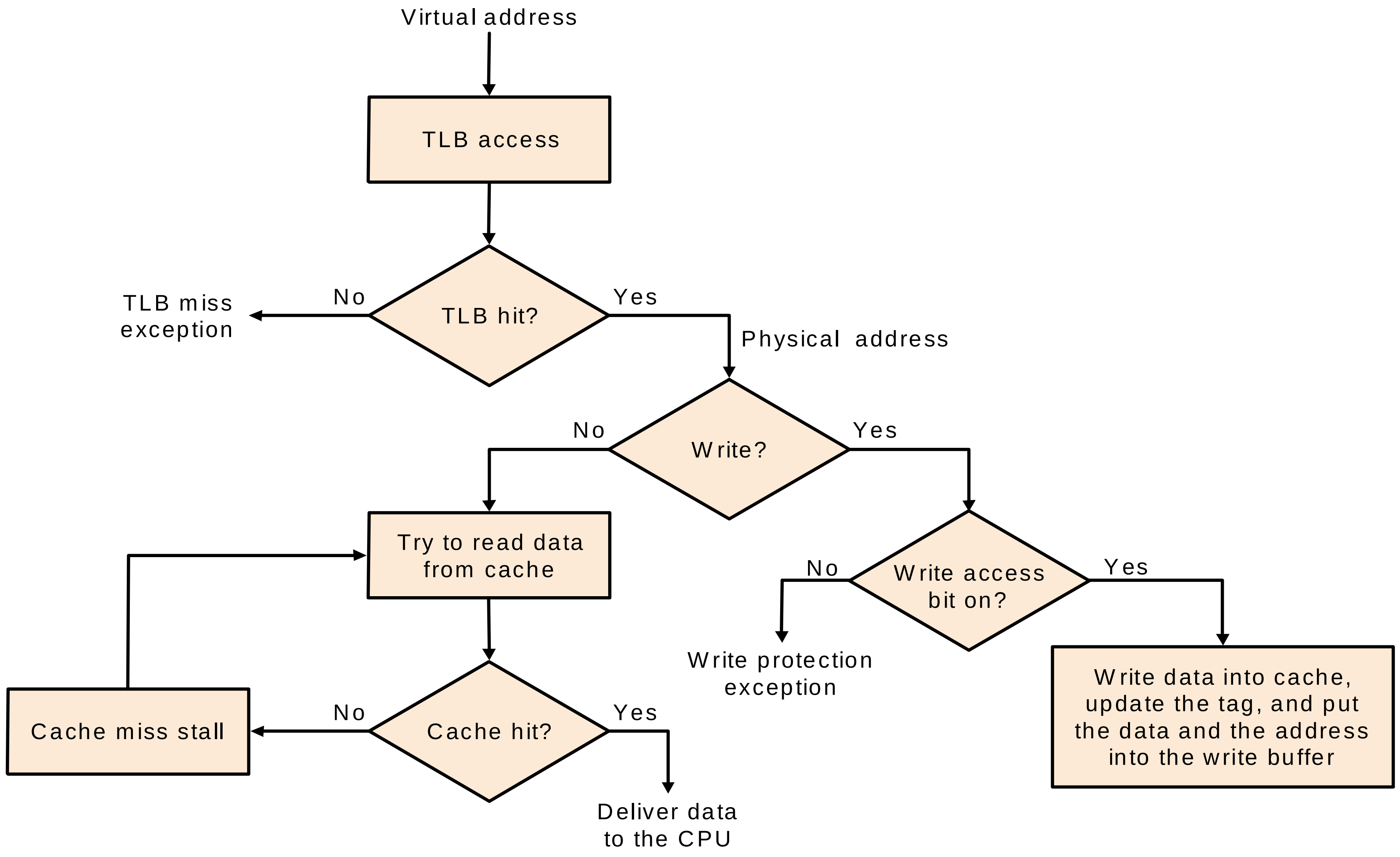
Just like any other cache, the TLB can be organized as
fully associative, set associative, or direct mapped

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

TLBs and caches

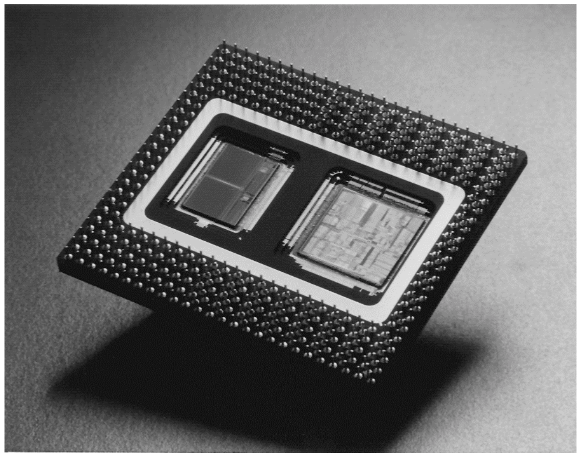


Modern Systems

Figure 7.32

- **Very complicated memory systems:**

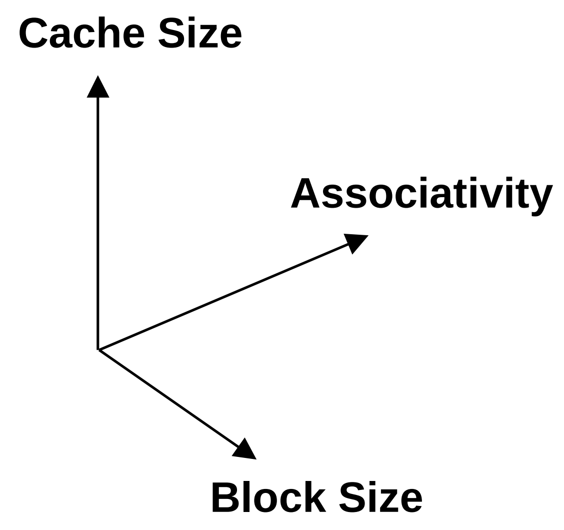
Characteristic	Intel Pentium Pro	PowerPC 604
Virtual address	32 bits	52 bits
Physical address	32 bits	32 bits
Page size	4 KB, 4 MB	4 KB, selectable, and 256 MB
TLB organization	A TLB for instructions and a TLB for data Both four-way set associative Pseudo-LRU replacement Instruction TLB: 32 entries Data TLB: 64 entries TLB misses handled in hardware	A TLB for instructions and a TLB for data Both two-way set associative LRU replacement Instruction TLB: 128 entries Data TLB: 128 entries TLB misses handled in hardware

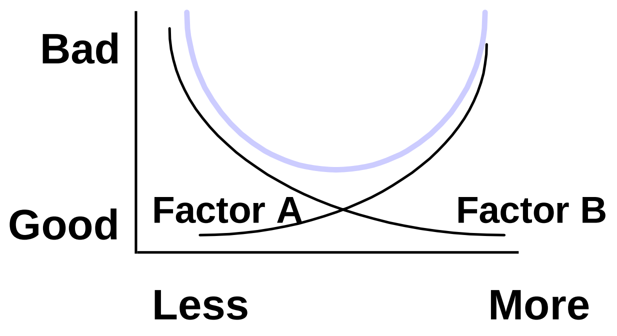


Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split intruction and data caches
Cache size	8 KB each for instructions/data	16 KB each for instructions/data
Cache associativity	Four-way set associative	Four-way set associative
Replacement	Approximated LRU replacement	LRU replacement
Block size	32 bytes	32 bytes
Write policy	Write-back	Write-back or write-through

Summary: 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: 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 are important for fast translation
- TLB misses are significant in processor performance: (funny times, as most systems can’t access all of 2nd level cache without TLB misses!)

Summary: 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 (ops, cache misses) vs. just f(ops):**

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