CASPER

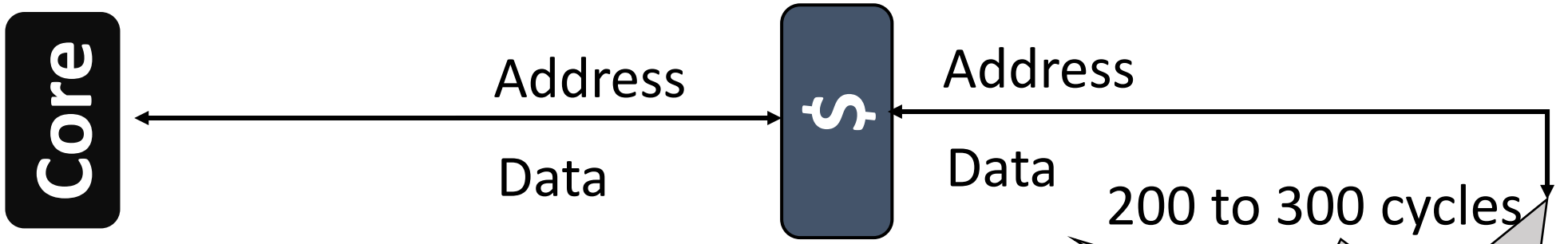
CS230: Digital Logic Design and Computer Architecture

Lecture 17: Caches

https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html

Caching: 10K Feet View

North pole ☺

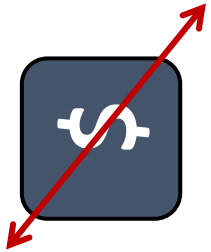


Costly DRAM accesses☹

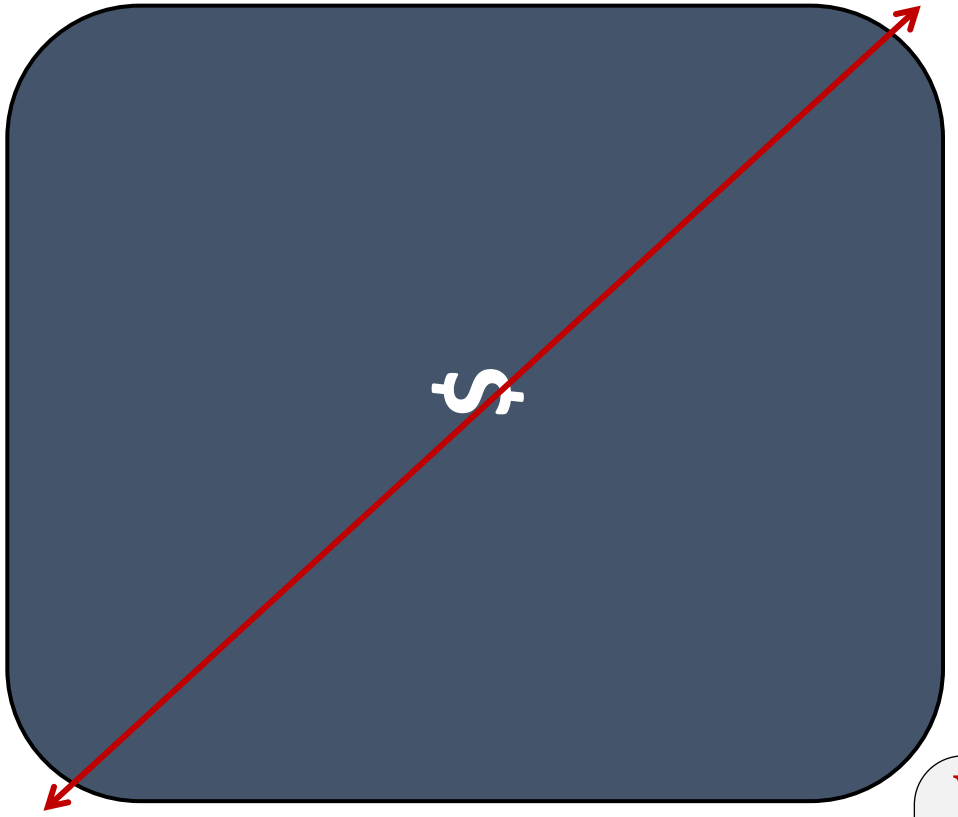
Caching is a *speculation* technique ☺
Works – if locality

How big/small?

Core

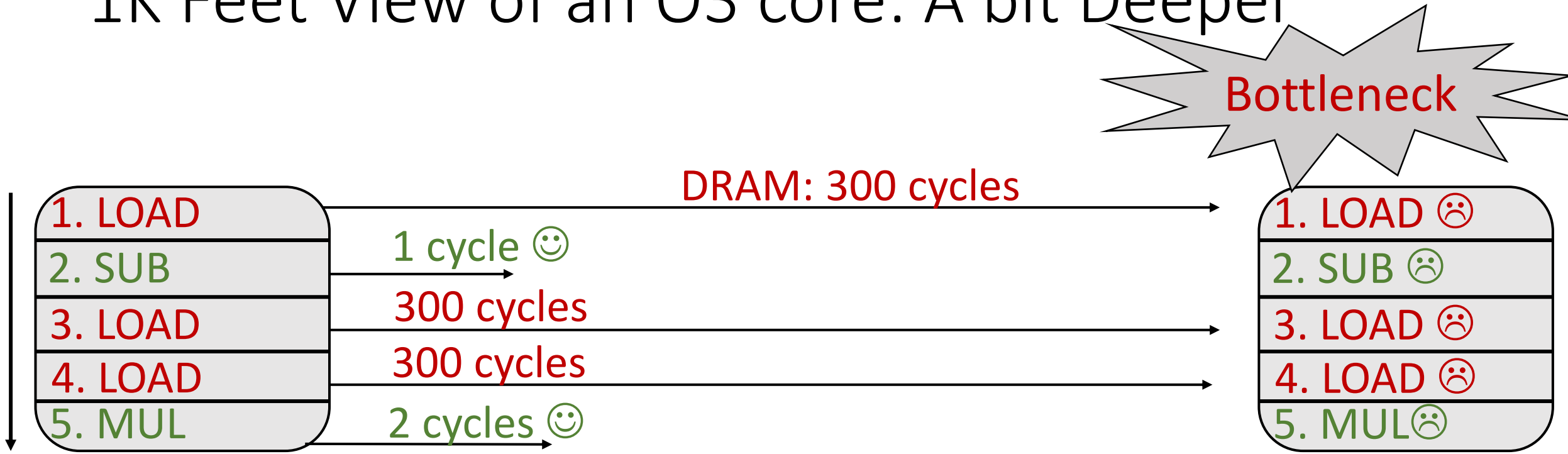


Latency: low
Area: low
Capacity: low



Latency: high
Area: high
Capacity: high

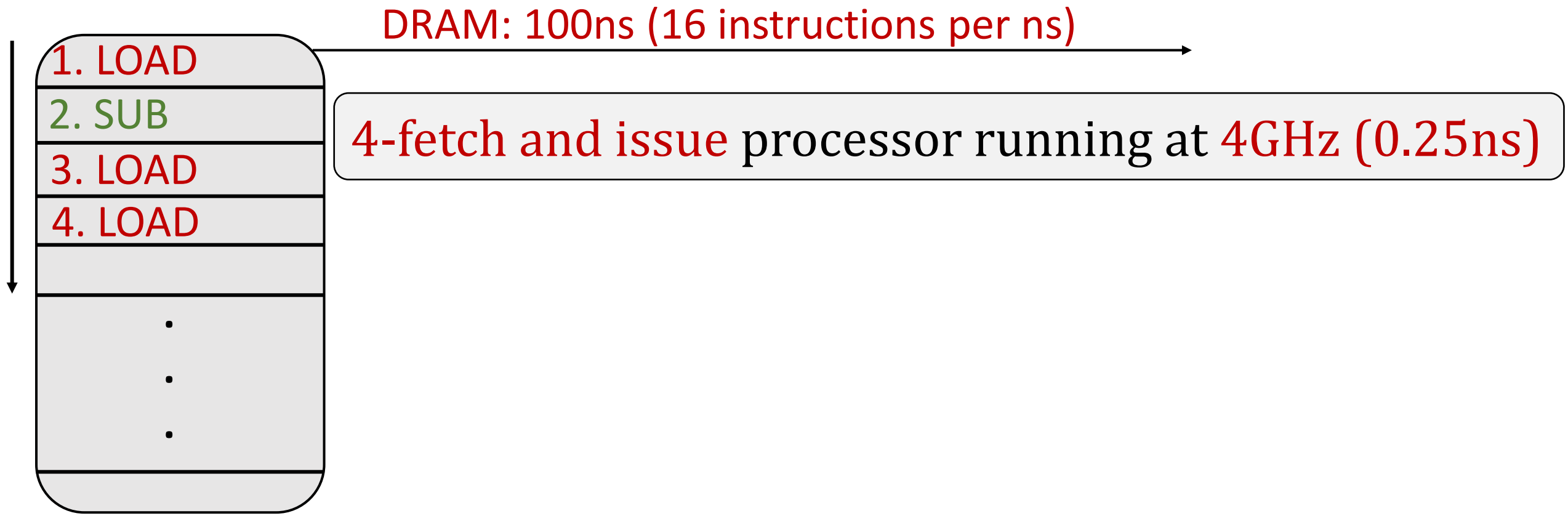
1K Feet View of an O3 core: A bit Deeper



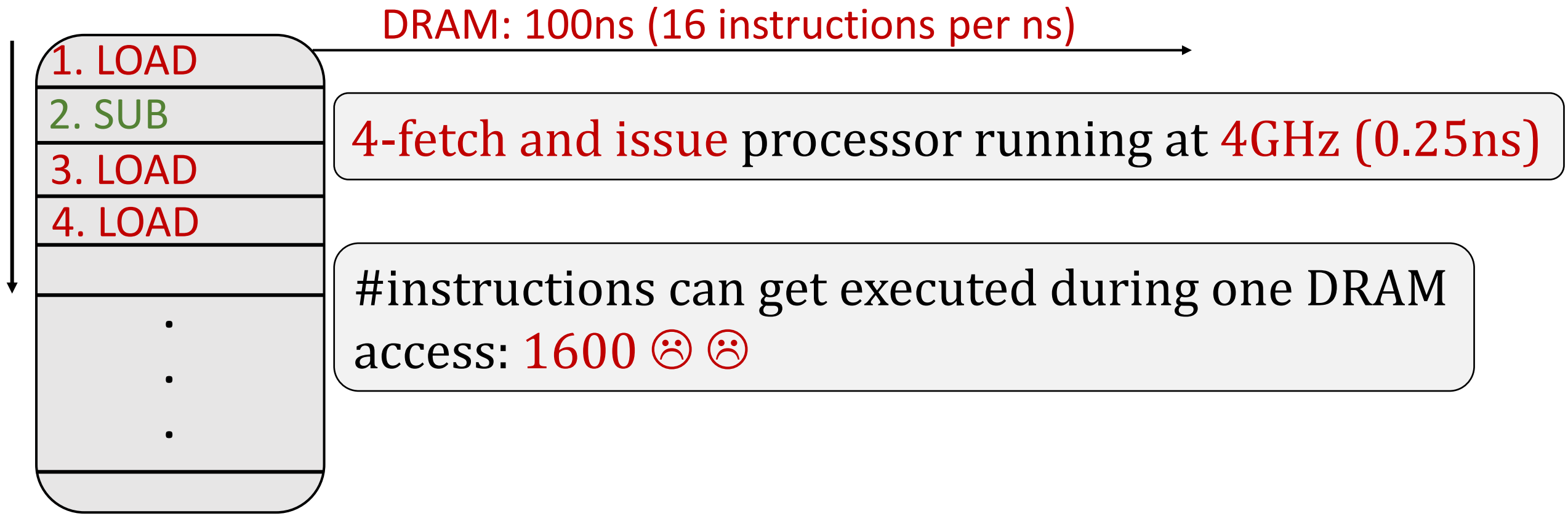
In-order Instruction Fetch
(Multiple fetch in one cycle)

Processor core says all LOADs should take one cycle. Ehh!

Impact of one DRAM access

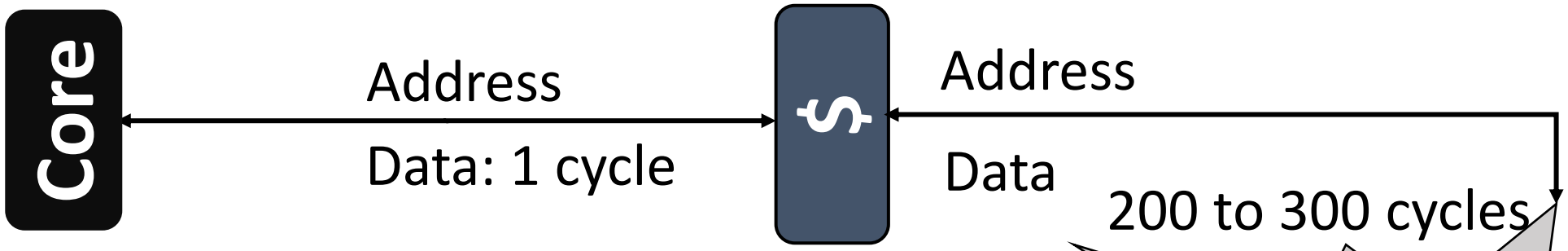


Impact of one DRAM access



Cache with latency

North pole ☺

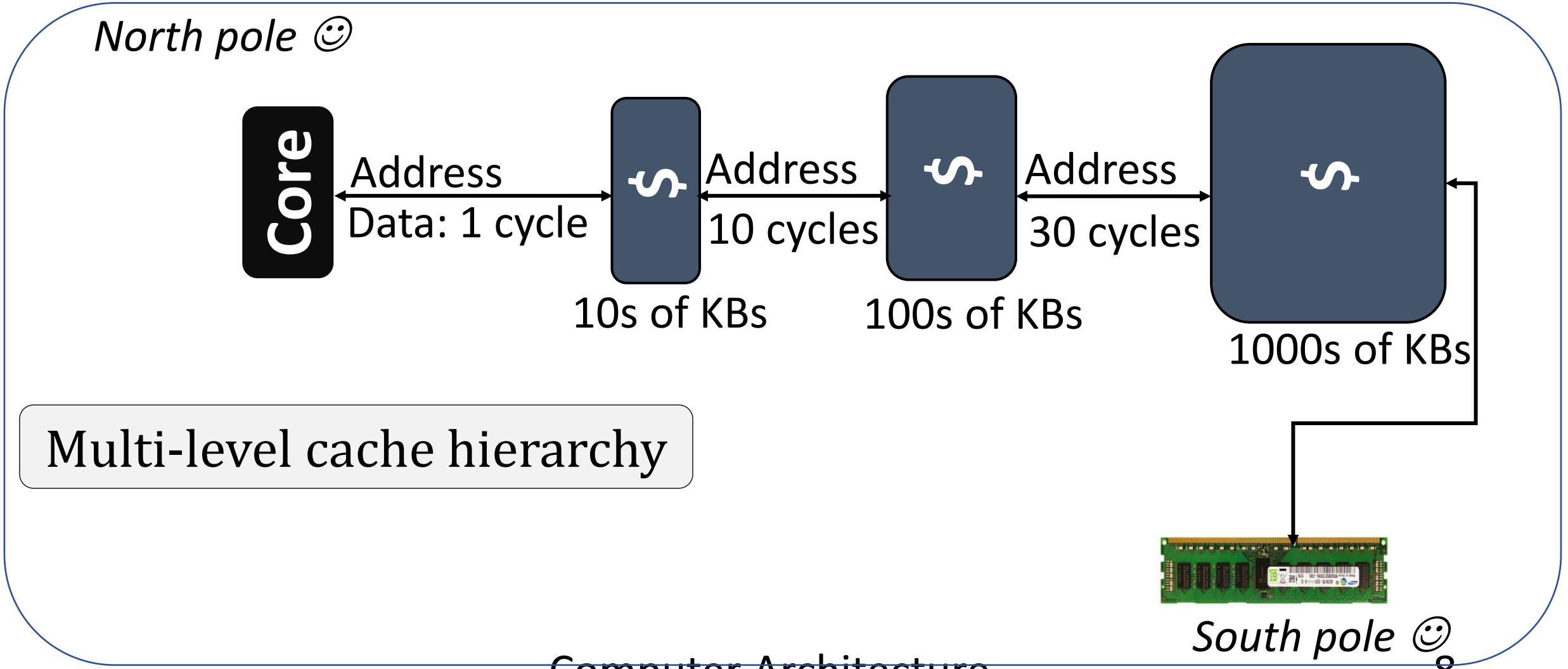


32 to 64KB $ will be available in one to four cycles ☹

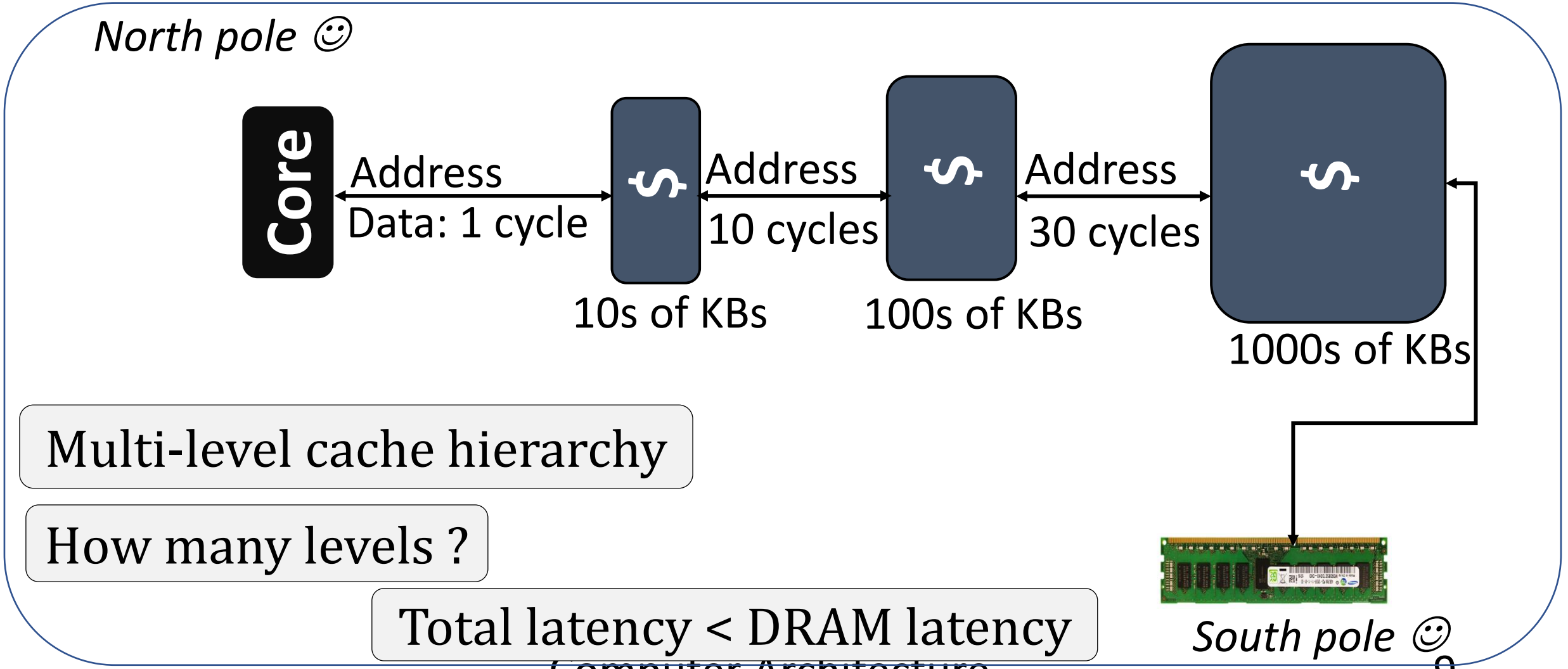
Costly DRAM accesses☹

South pole ☺

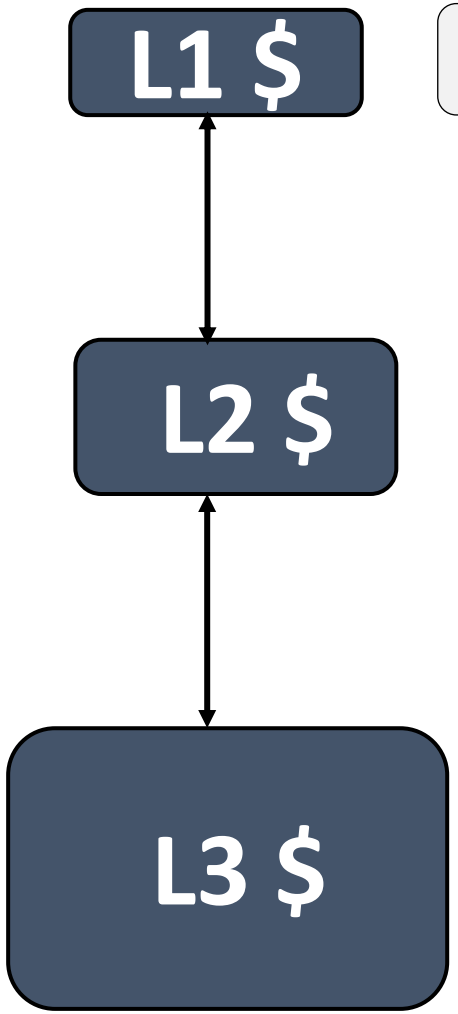
Cache hierarchy with latency



Cache hierarchy with latency

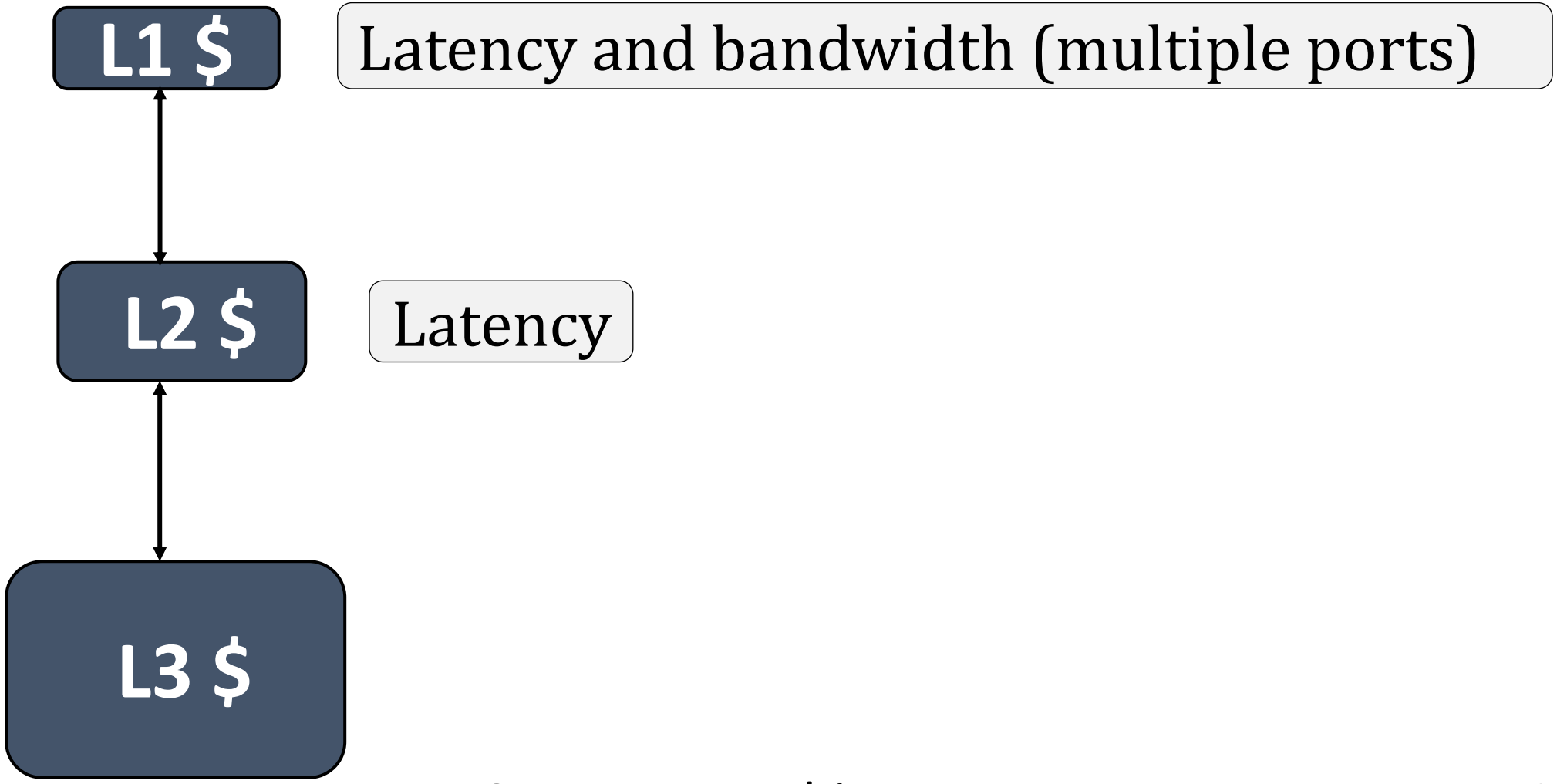


Takeaway

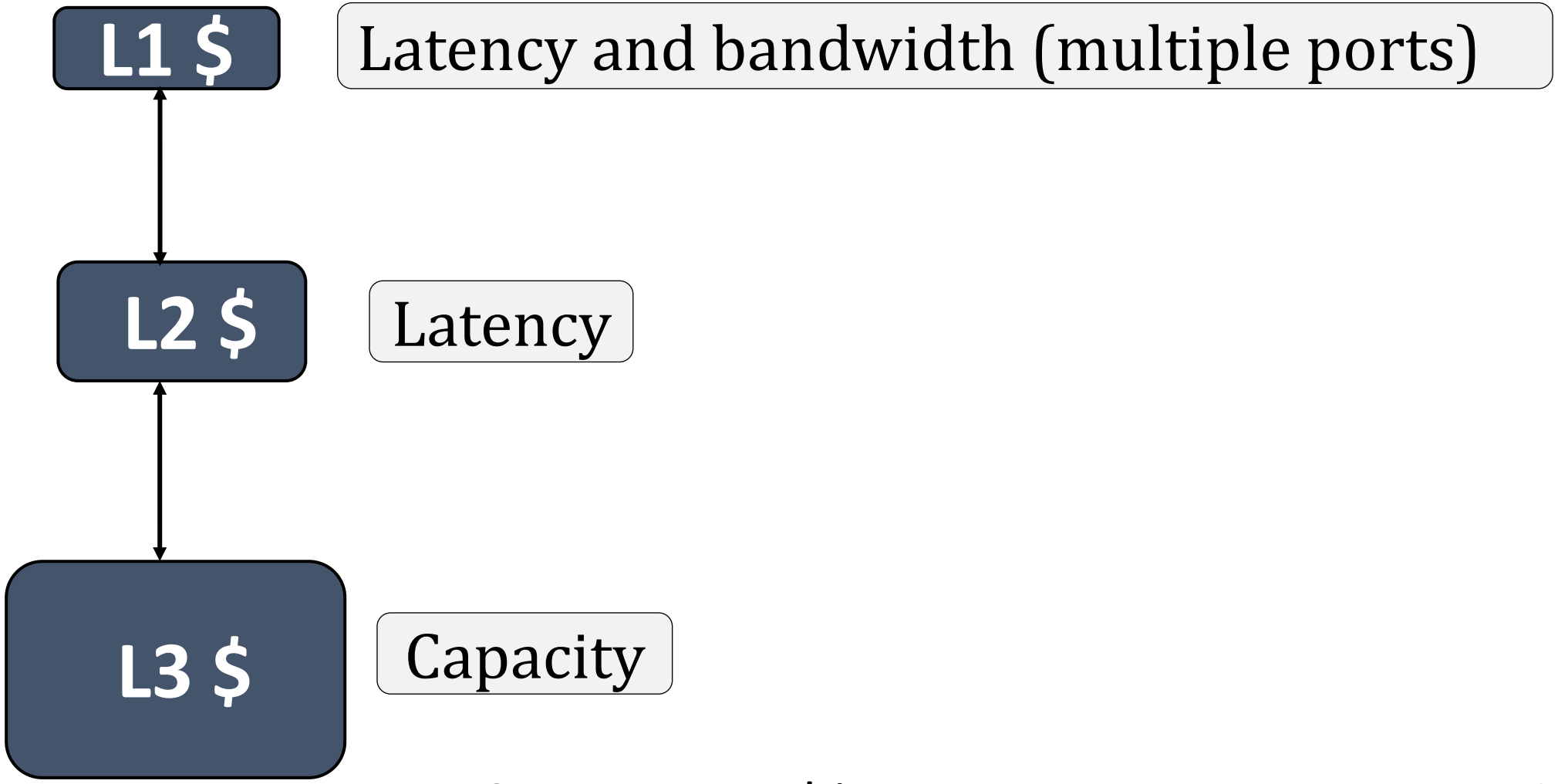


Latency and bandwidth (multiple ports)

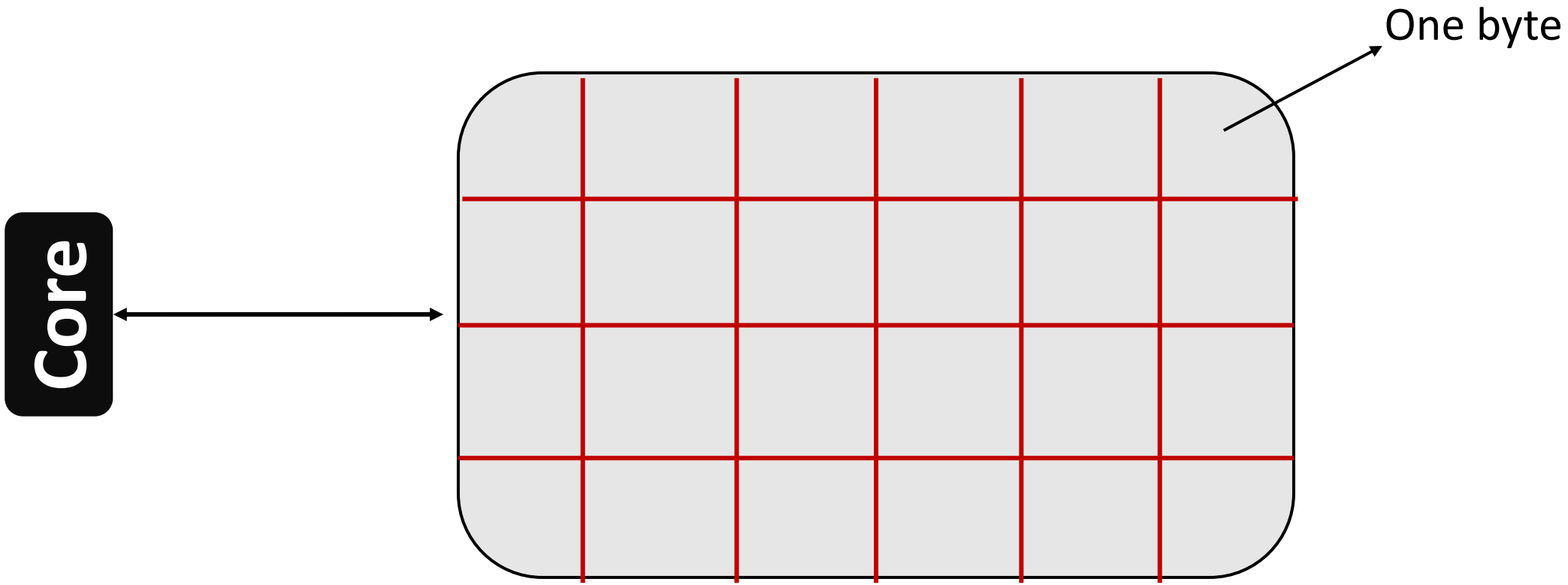
Takeaway



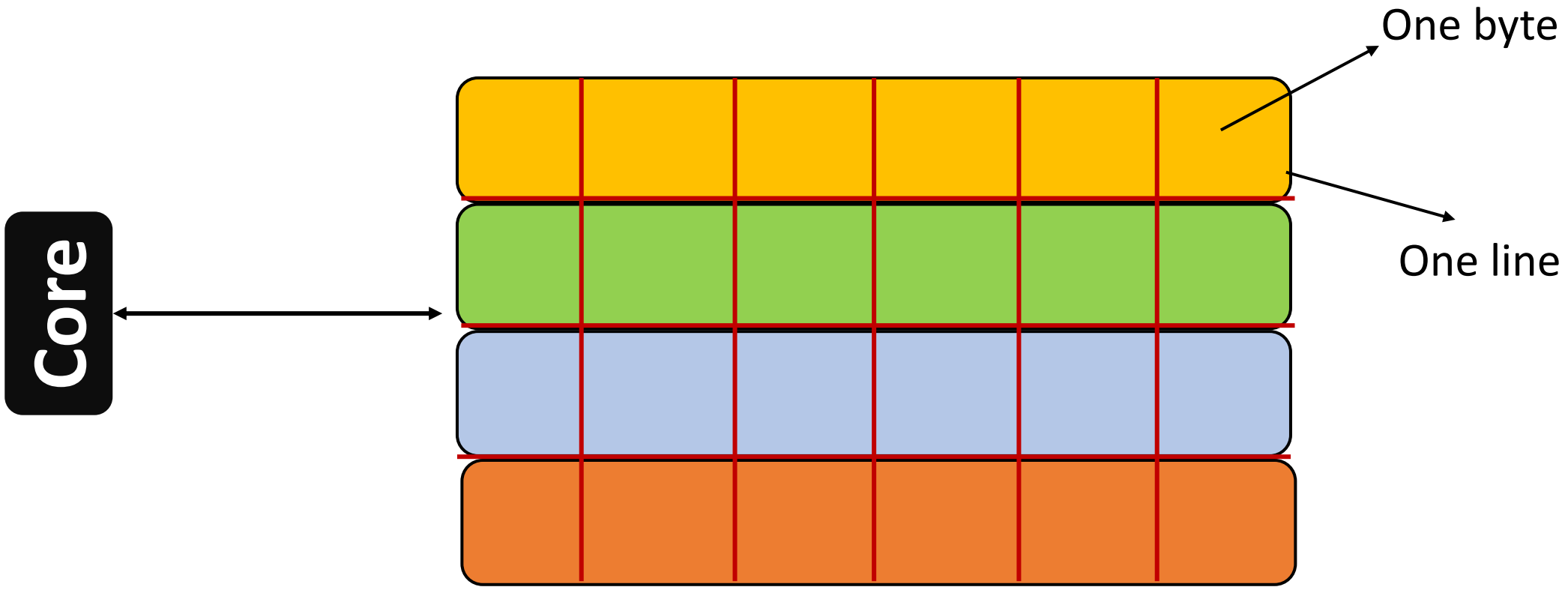
Takeaway


L1 $
Latency and bandwidth (multiple ports)
L2 $
Latency
L3 $
Capacity

Accessing a cache

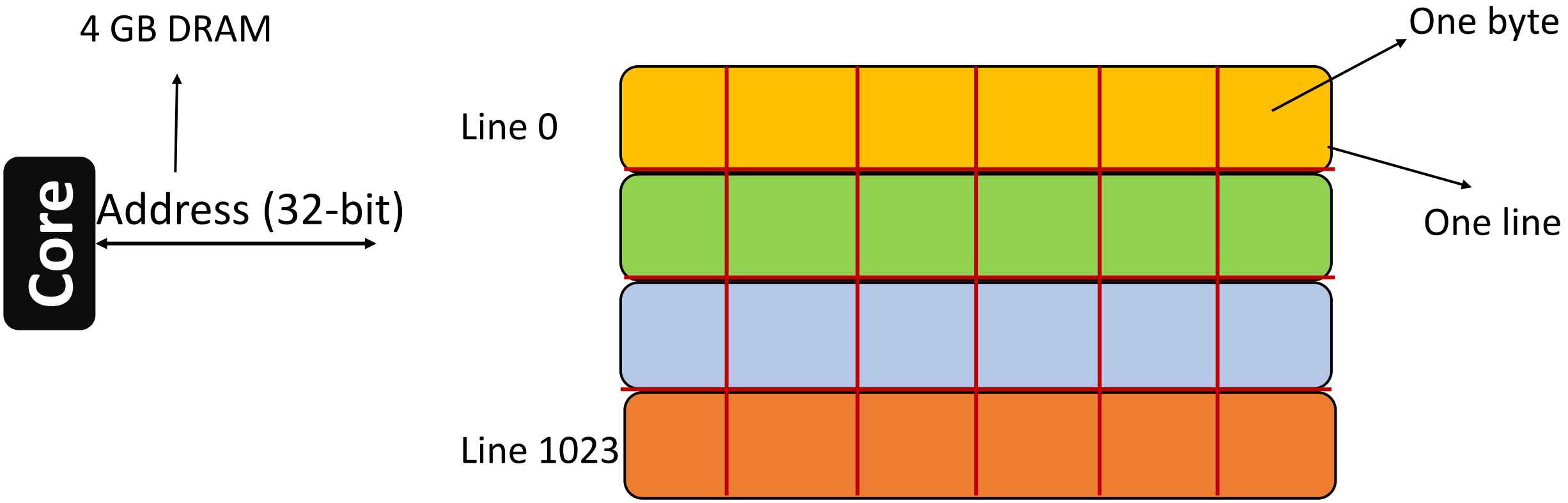


Bytes to blocks (lines)

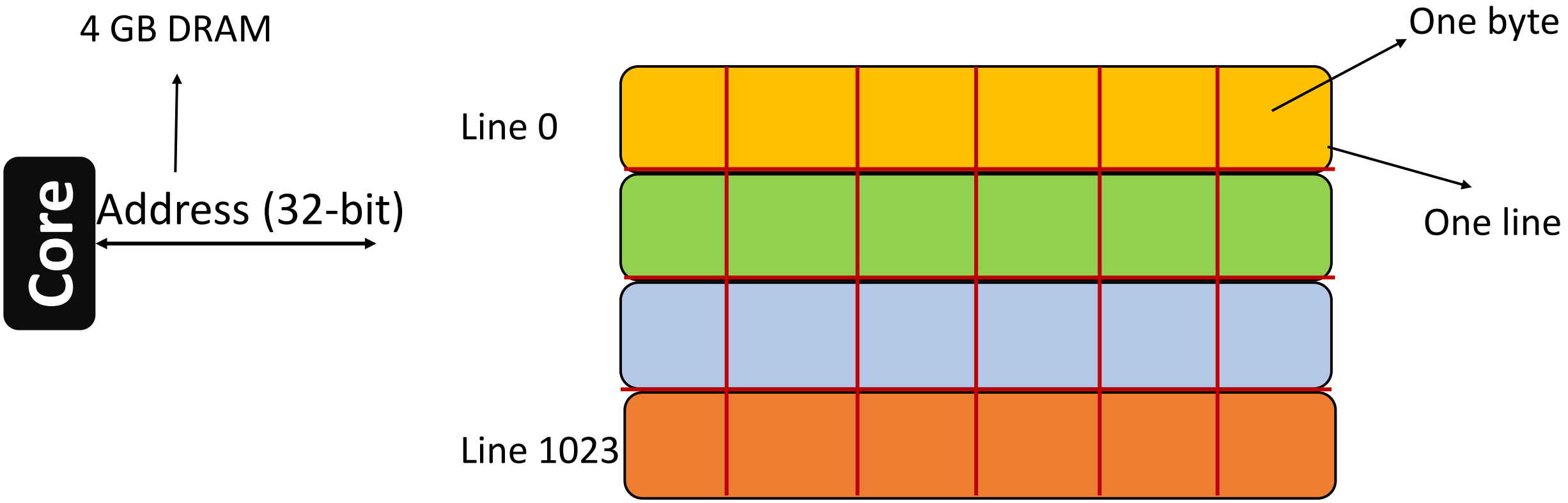


Typical line size: 64 to 128 Bytes

A bit deeper: 1024 lines each of 32B



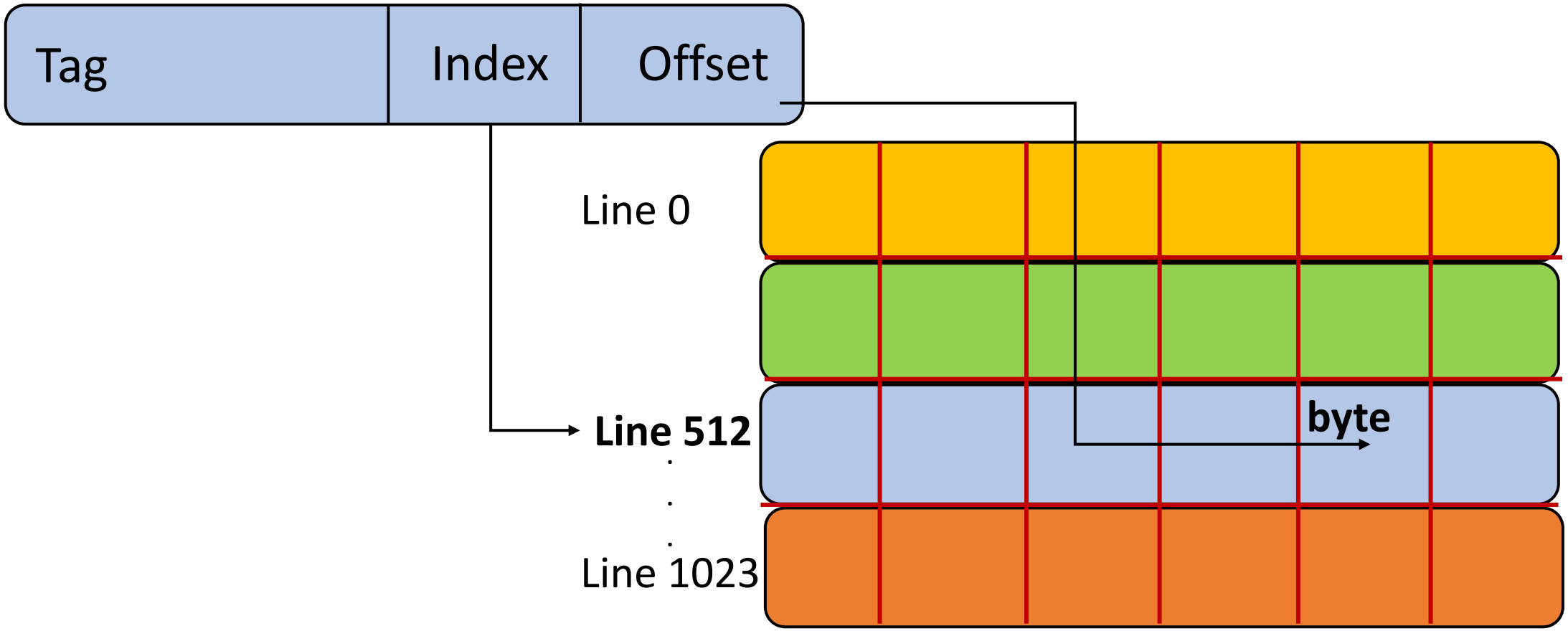
A bit deeper: 1024 lines each of 32B



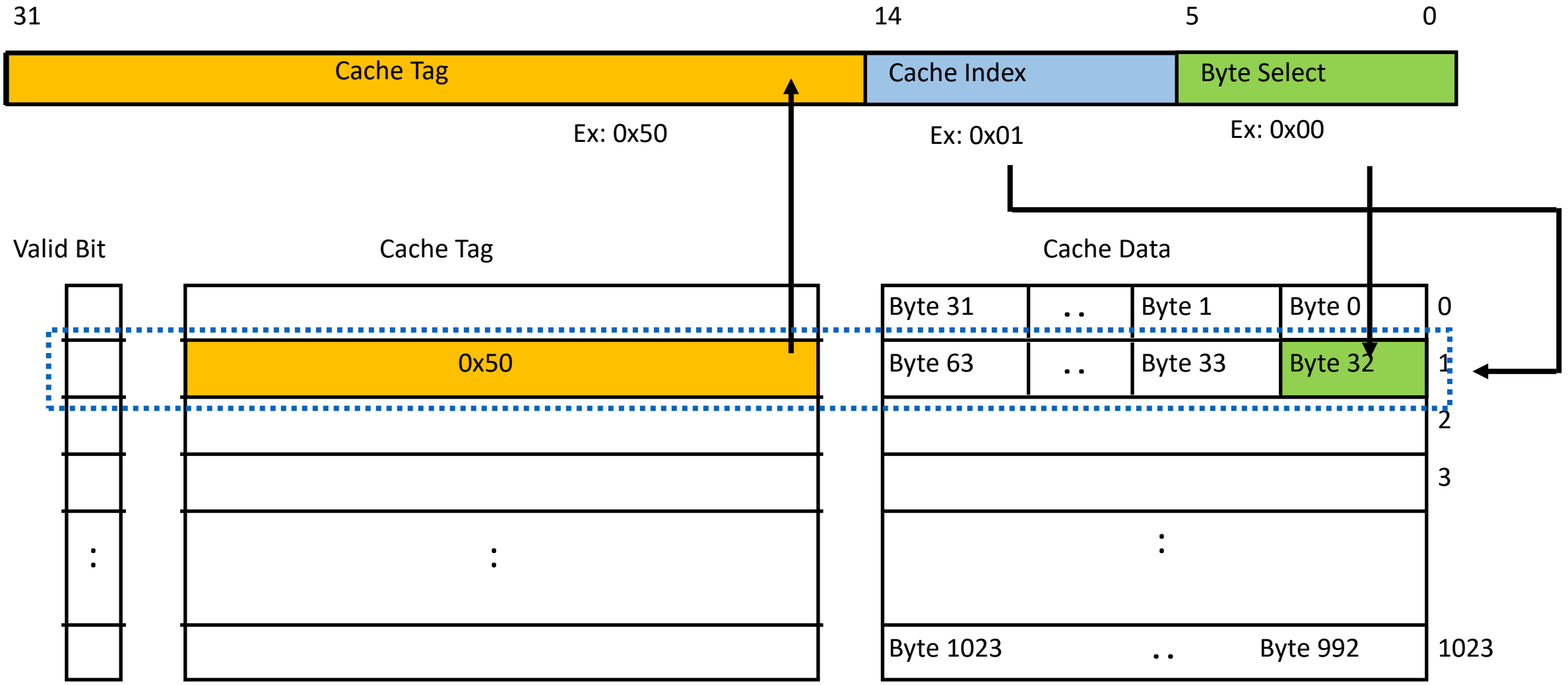
Line number (index): 10 bits

Byte offset (offset): 5 bits

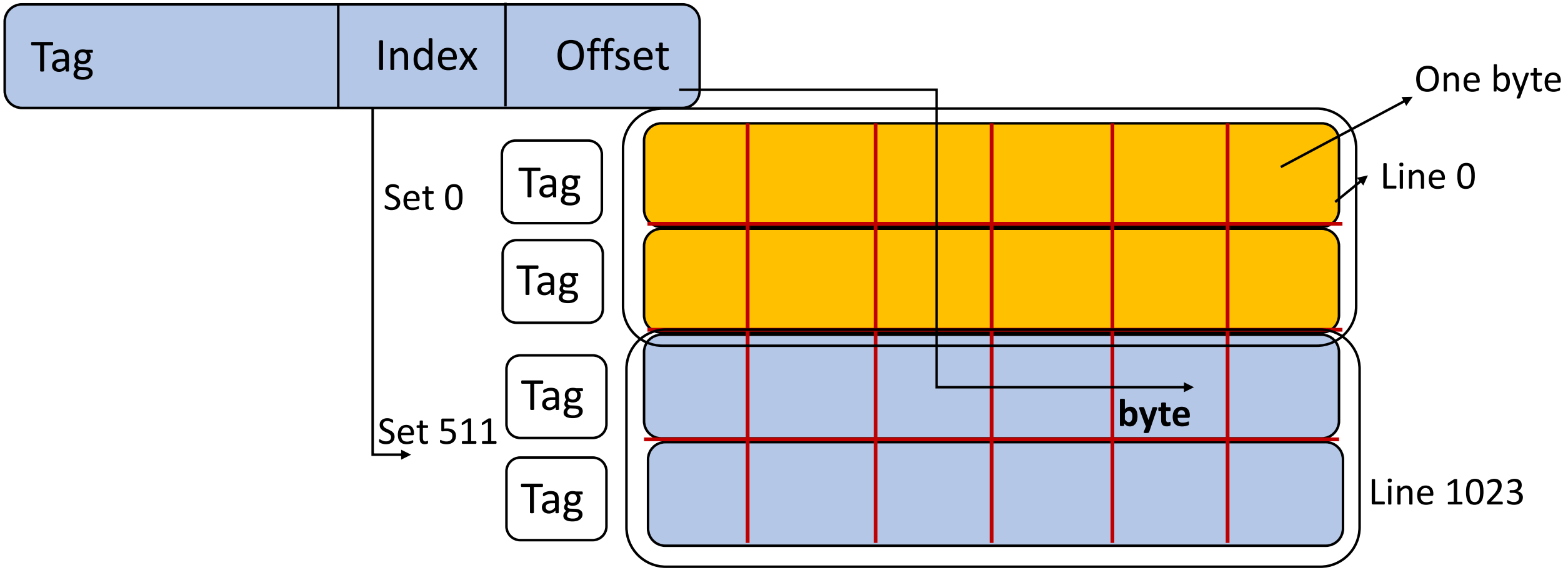
Direct Mapped Cache



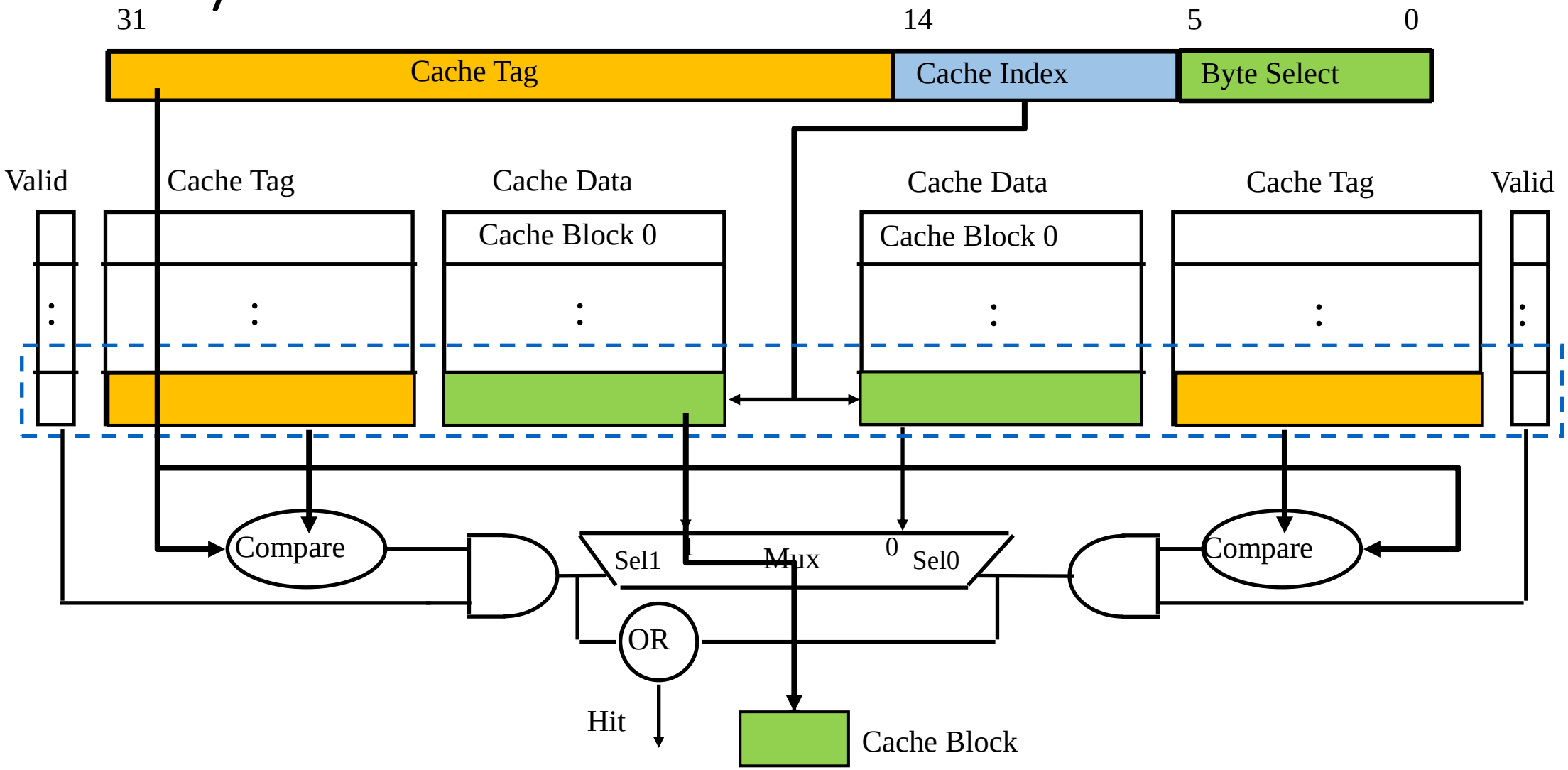
Direct Mapped in Action



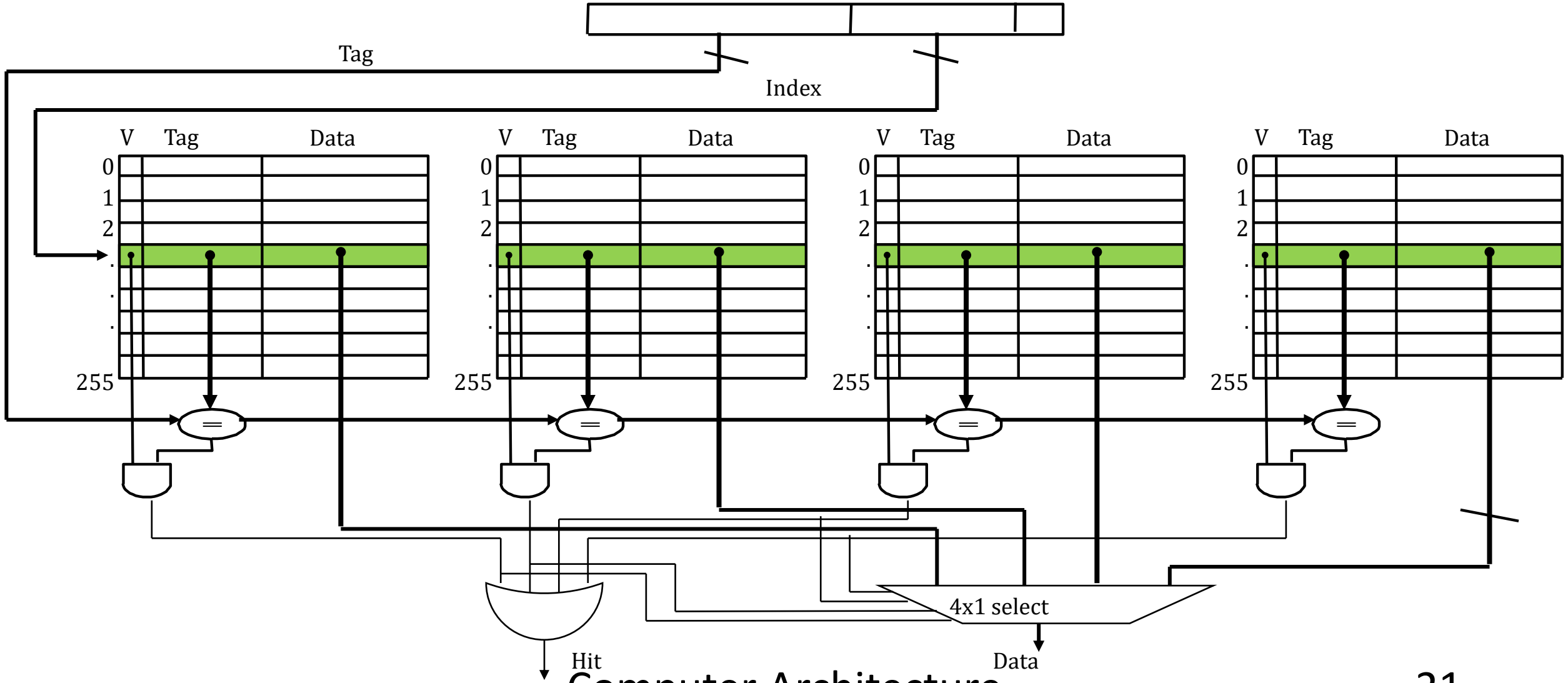
What if we have multiple ways?



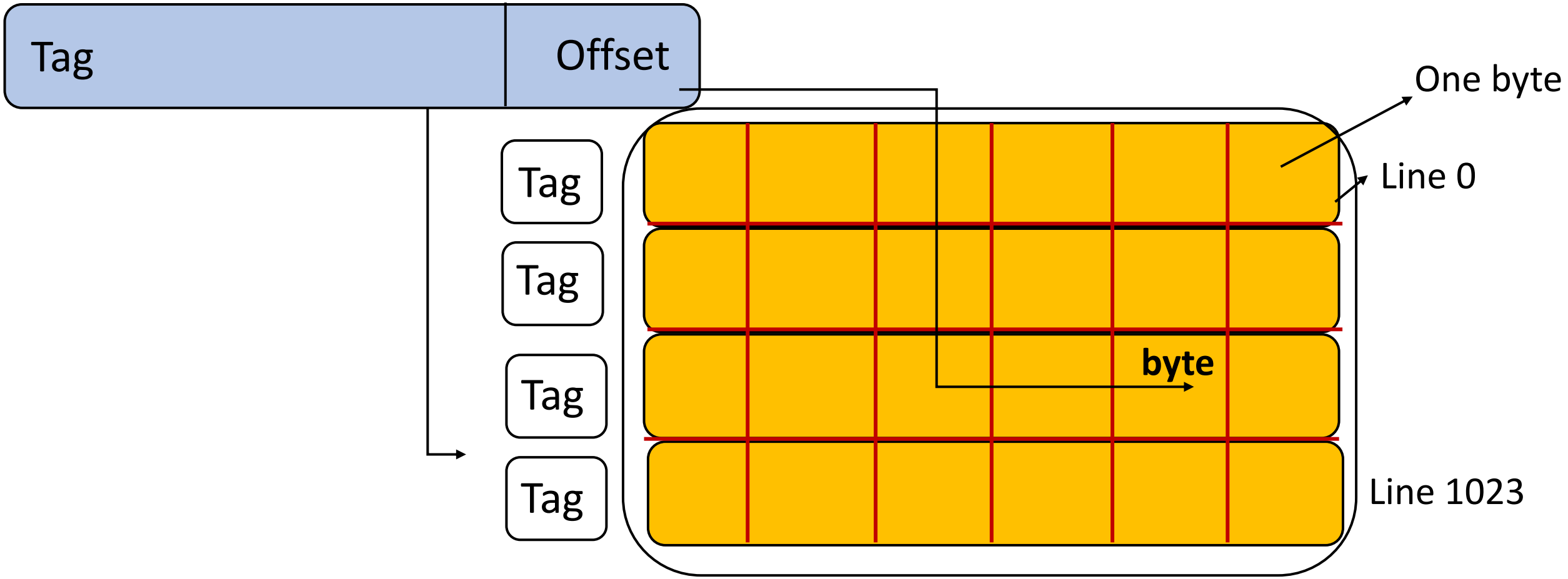
2-way associative in action



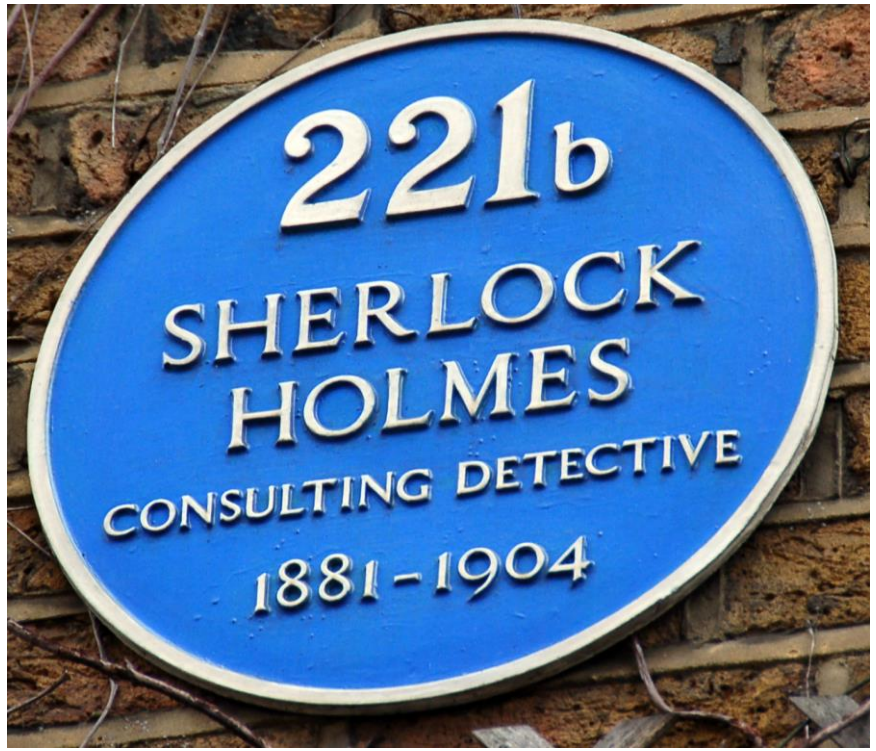
4-way associative: Just a better picture



Extreme: One cache, one set, fully associative



A bit different way



Baker Street: Cache Index ☺

221b: Tag bits ☺

Sherlock Holmes: Byte offset ☺ ☺

Knobs of interest

Line size, associativity, cache size

Tradeoff: latency, complexity, energy/power

Tips: Think about the extremes:
Line size = one byte or cache size
Associativity = one or #lines
Cache size = Goal oriented: latency/bandwidth or capacity

https://github.com/HewlettPackard/cacti/

Cache misses

Cold Miss: cache starts empty and this is the first reference

Conflict Miss: Many mapped to the same index bits

Capacity Miss: Cache size is not sufficient

Coherence Miss: in Multi-core systems, only [not I/O coherence]

On a Miss, Replace a block, which block?

Think of each block in a set having a “priority”
Indicating how important it is to keep the block in the cache

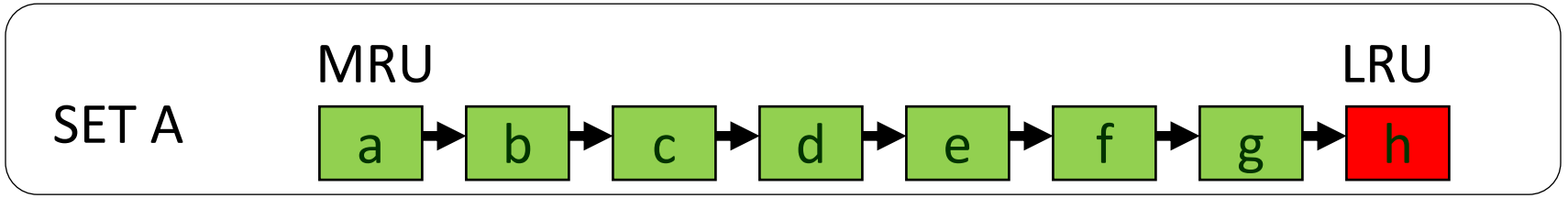
Key issue: How do you determine/adjust block priorities?

Ideally: Belady’s OPT policy, replace the block that will be used furthest in the future. No one knows the future though ☺

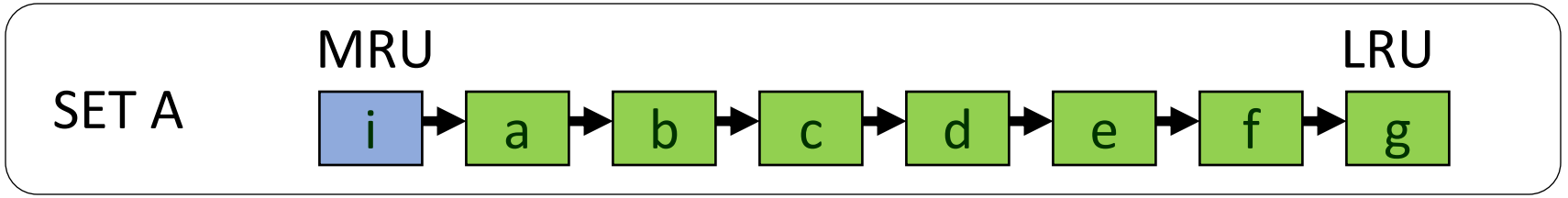
There are three key decisions in a set:
Insertion, promotion, eviction (replacement)

A simple LRU (Least-Recently-Used) Policy

Cache Eviction Policy: On a miss (block *i*), which block to evict (replace) ?



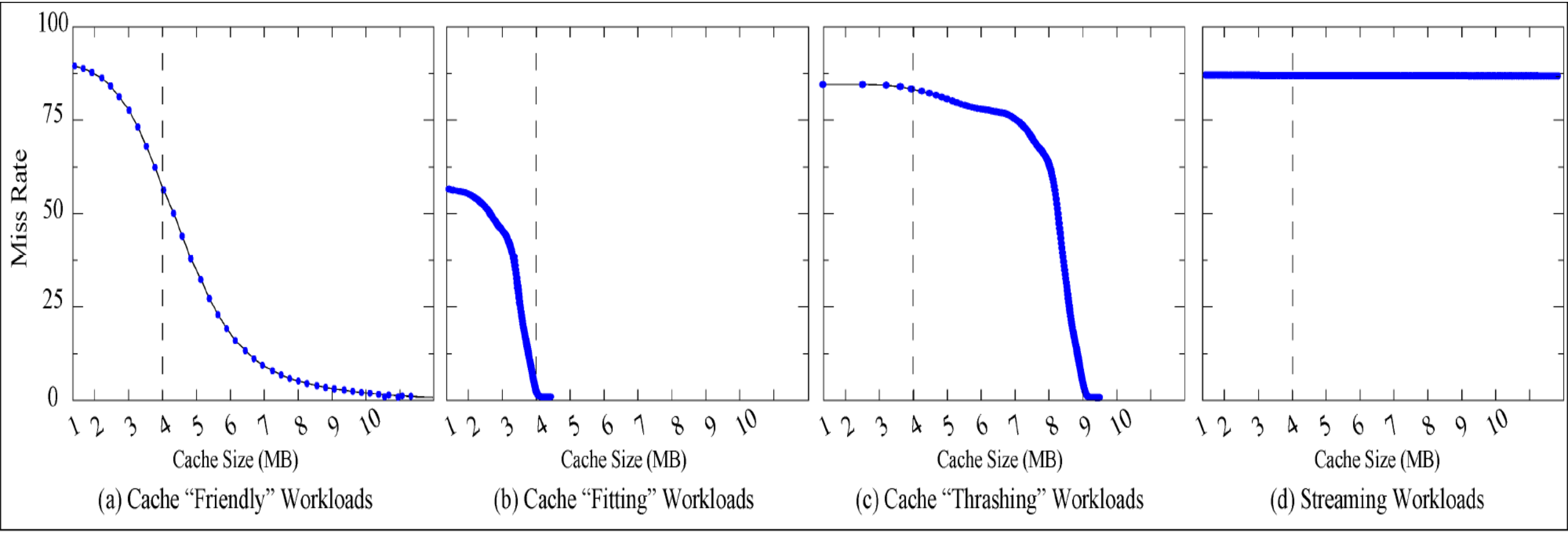
Cache Insertion Policy: New block *i* inserted into MRU.



Cache Promotion Policy: On a future hit (block *i*), promote to MRU

We need priority bits per block. For example, a 16-way cache will need four bit/block LRU causes thrashing when working set > cache size

Types of Applications



Let's redefine cache misses

Compulsory: first reference to a line (a.k.a. cold start misses)

- *misses that would occur even with infinite cache*

Capacity: cache is too small to hold all data

- *misses that would occur even under perfect (Belady's) replacement policy*

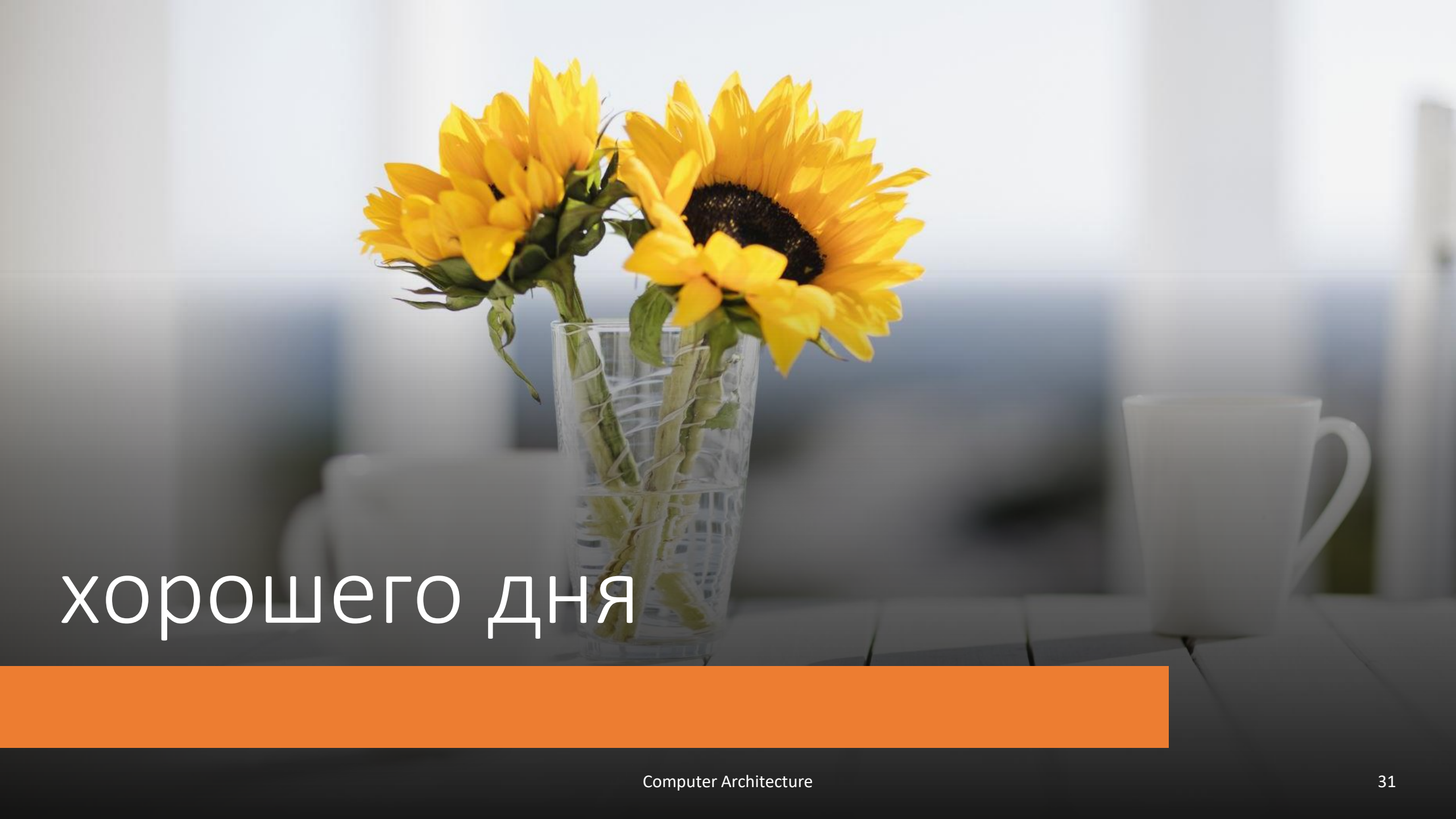
Conflict: misses that occur because of collisions due to line-placement strategy

- *misses that would not occur with ideal full associativity*

Coffee Credits

Karan : + 1

Dhananjay: +1



хорошего дня