CASPER

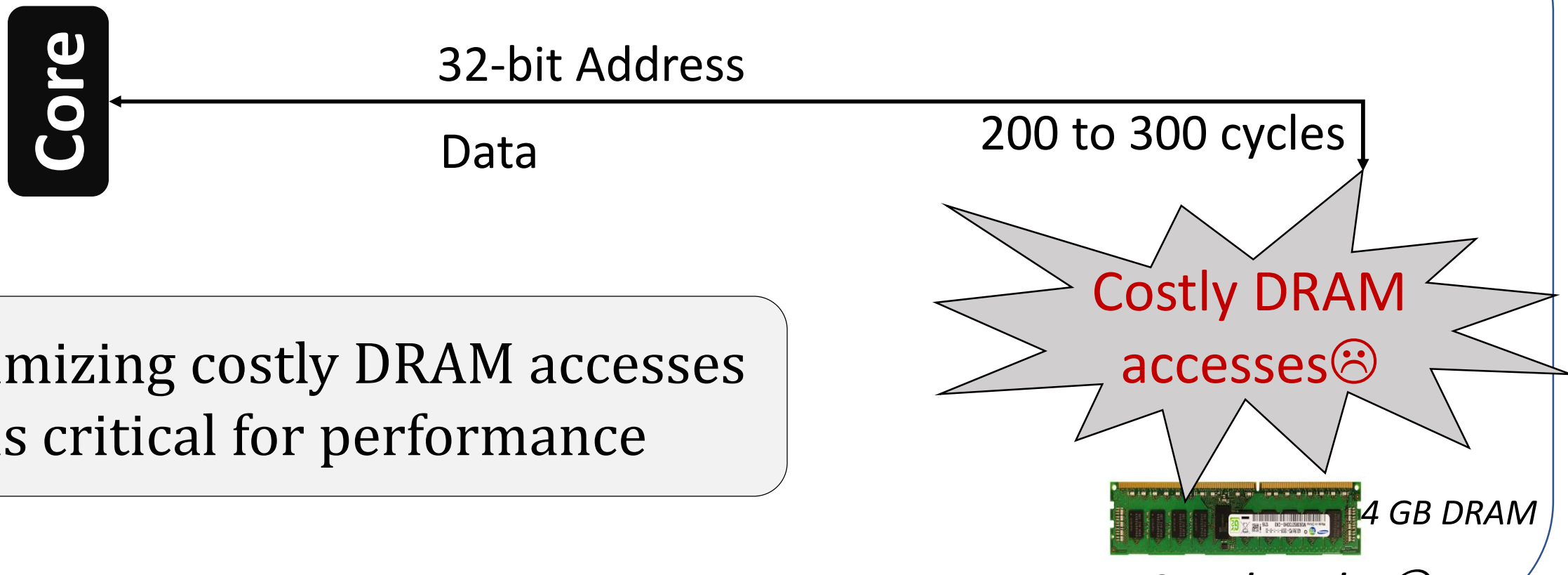
CS230: Digital Logic Design and Computer Architecture

Lecture 18: Cash, no; Caches

https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html

World with no caches

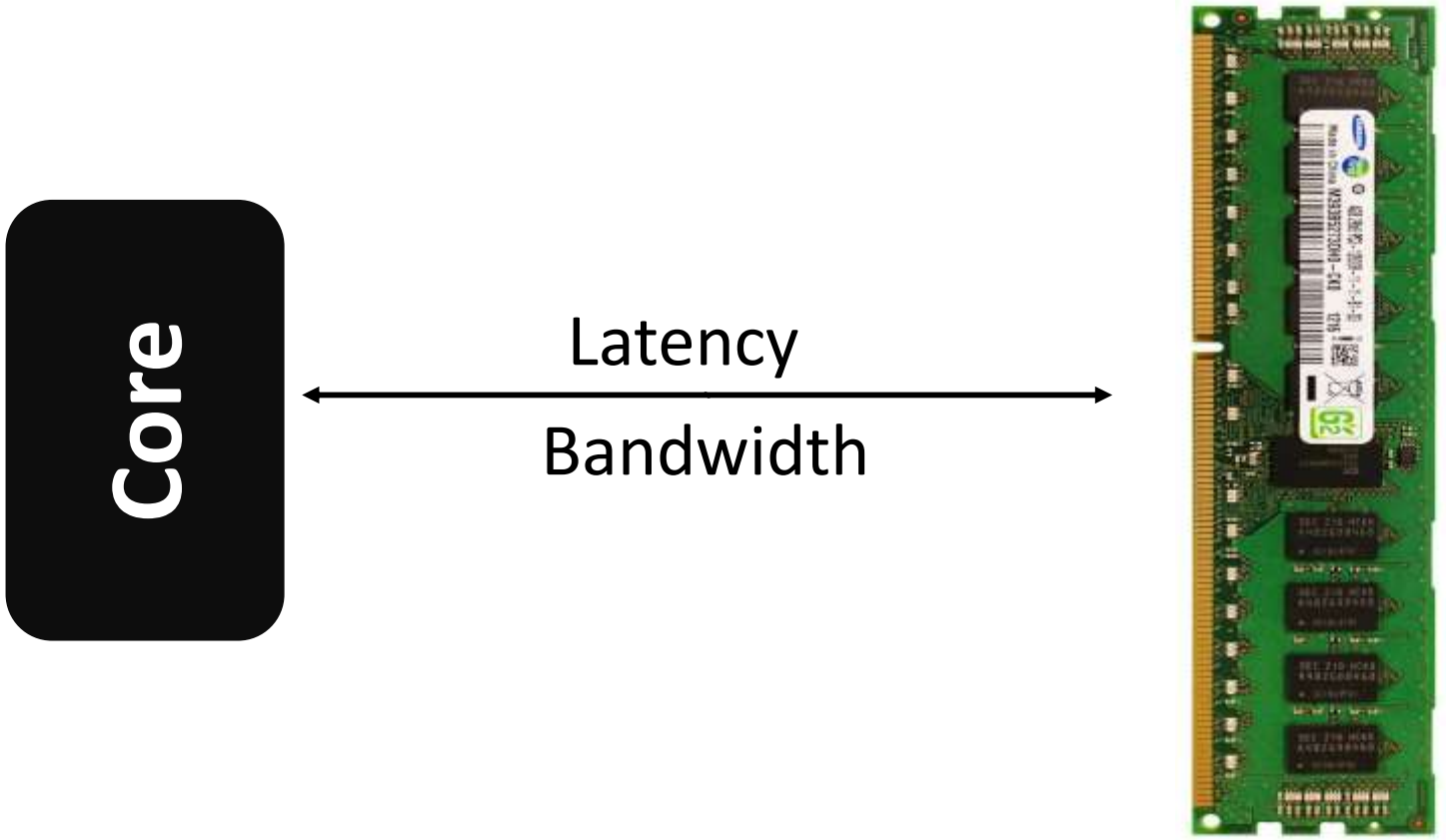
North pole ☹



Minimizing costly DRAM accesses is critical for performance

South pole ☹

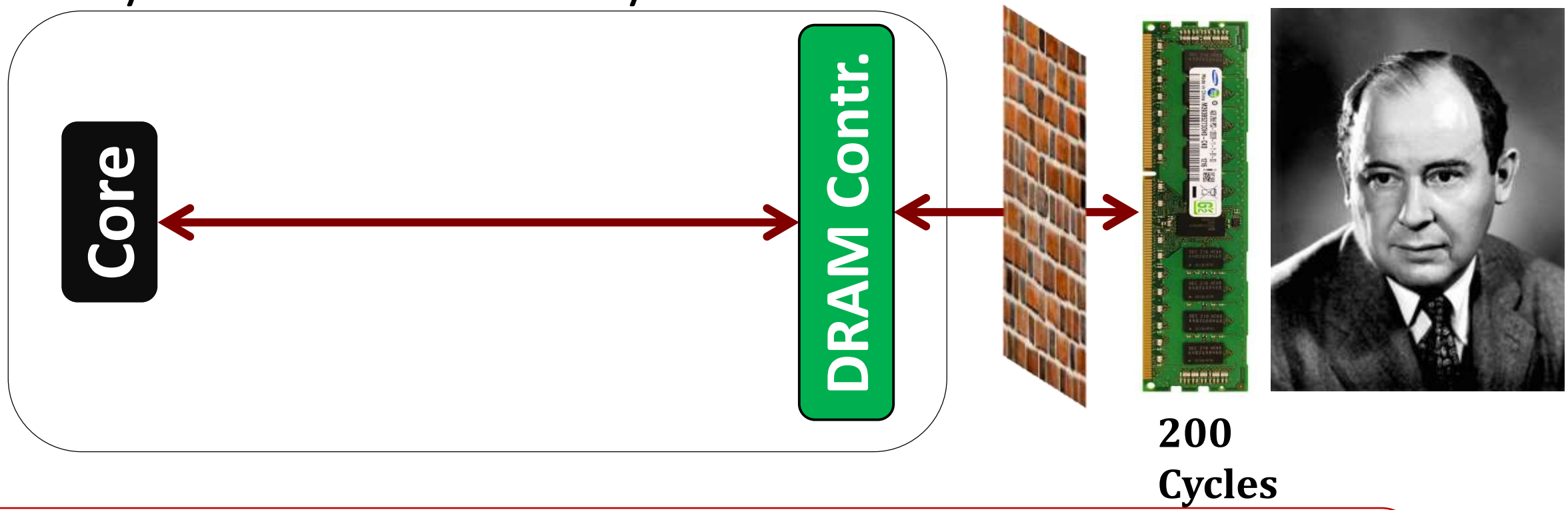
Remember Latency and Bandwidth



Latency ☹

Bandwidth problems can be cured with money.
Latency problems are harder because the speed of light is fixed – you can't bribe God

Why access memory?



Memory stores CODE and DATA

Processor accesses for LOADs (reads) and STOREs(writes)

Memory Wall: Grandmother of all the walls ☺

Do not ignore the common case mantra

Reduction in DRAM accesses ~ Improvement in execution time

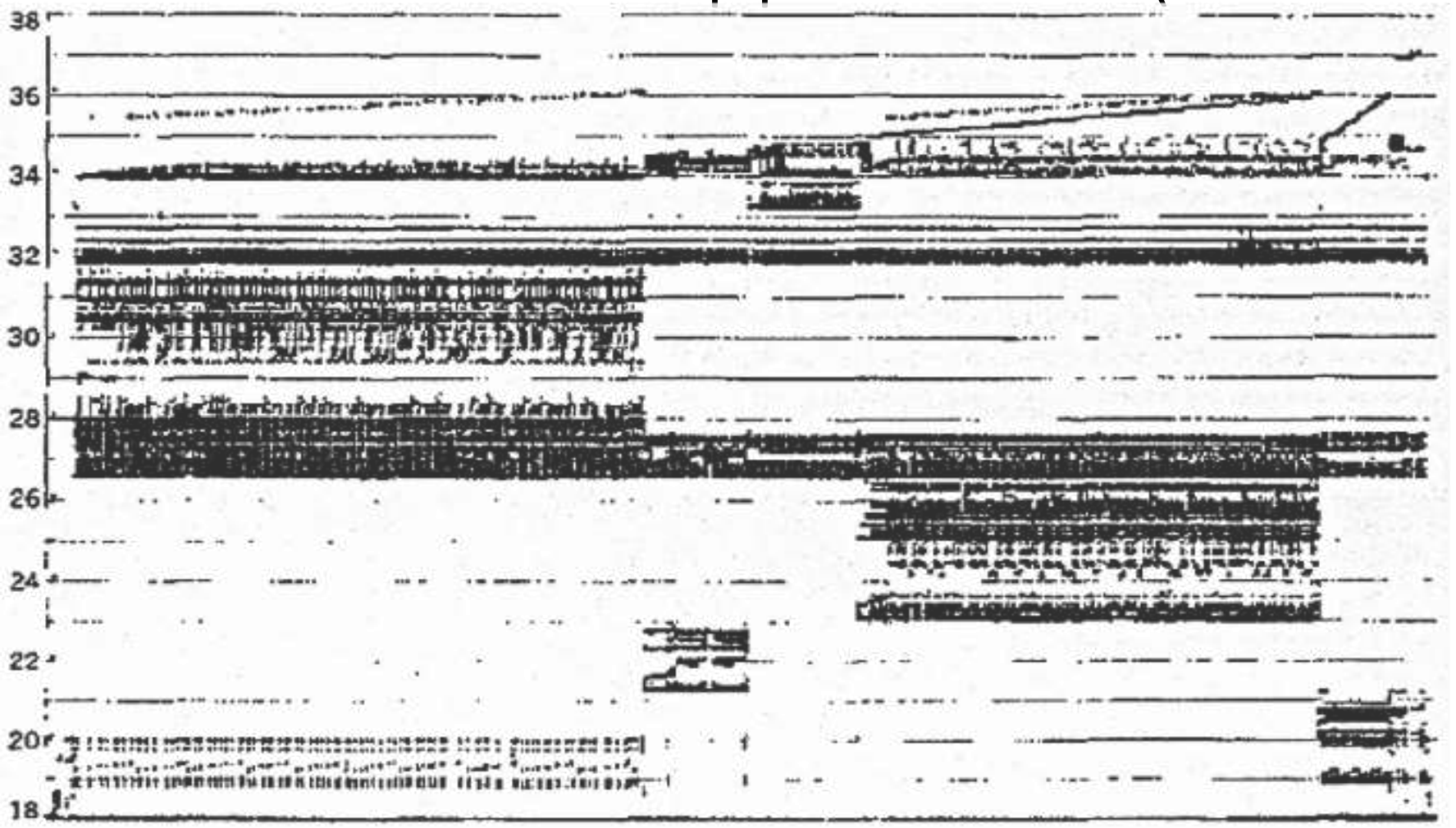
WRONG!

- First Law of Performance:

  Make the common case fast

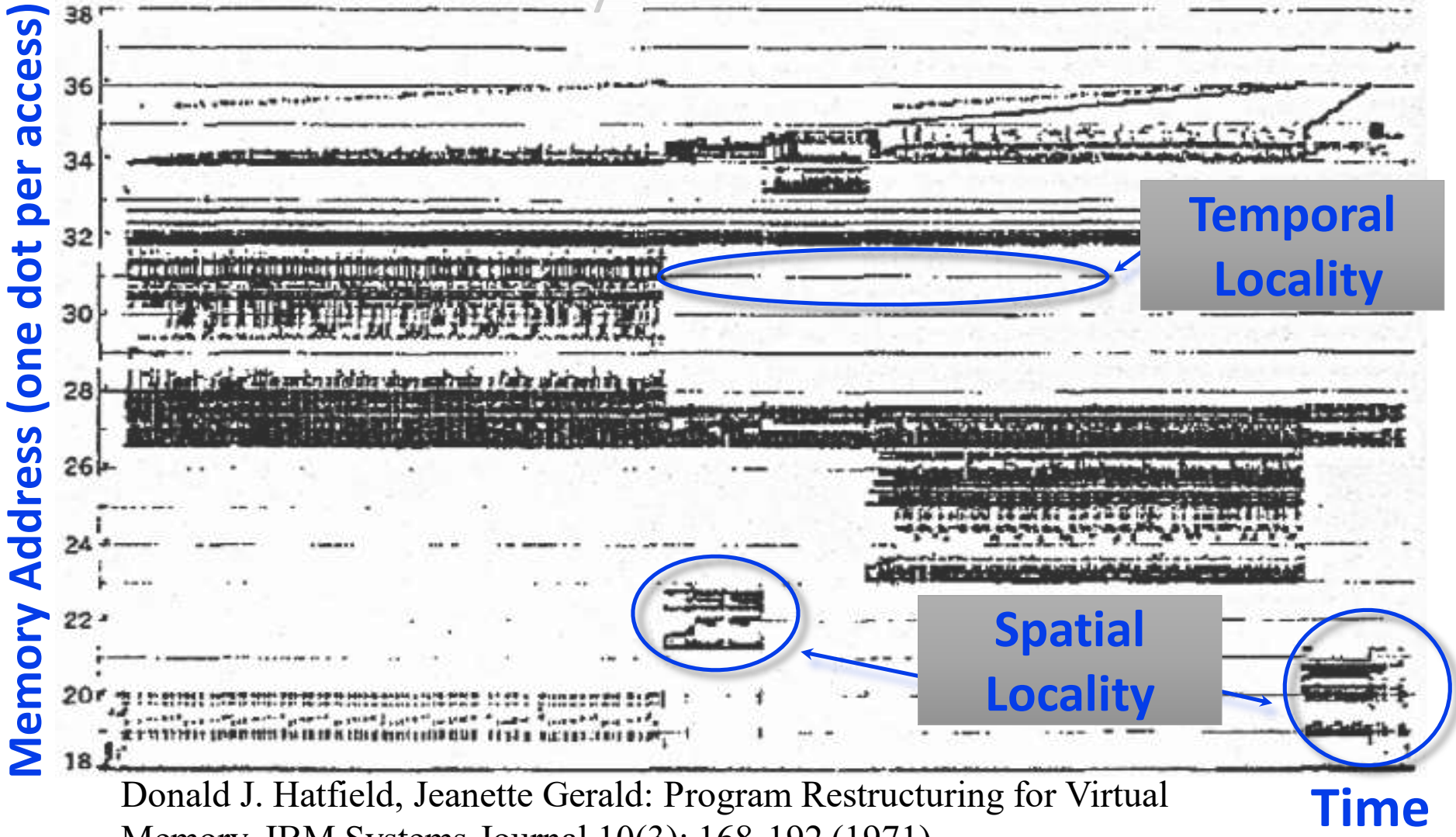
What if your program is not memory intensive

Let's look at the Applications (benchmarks)



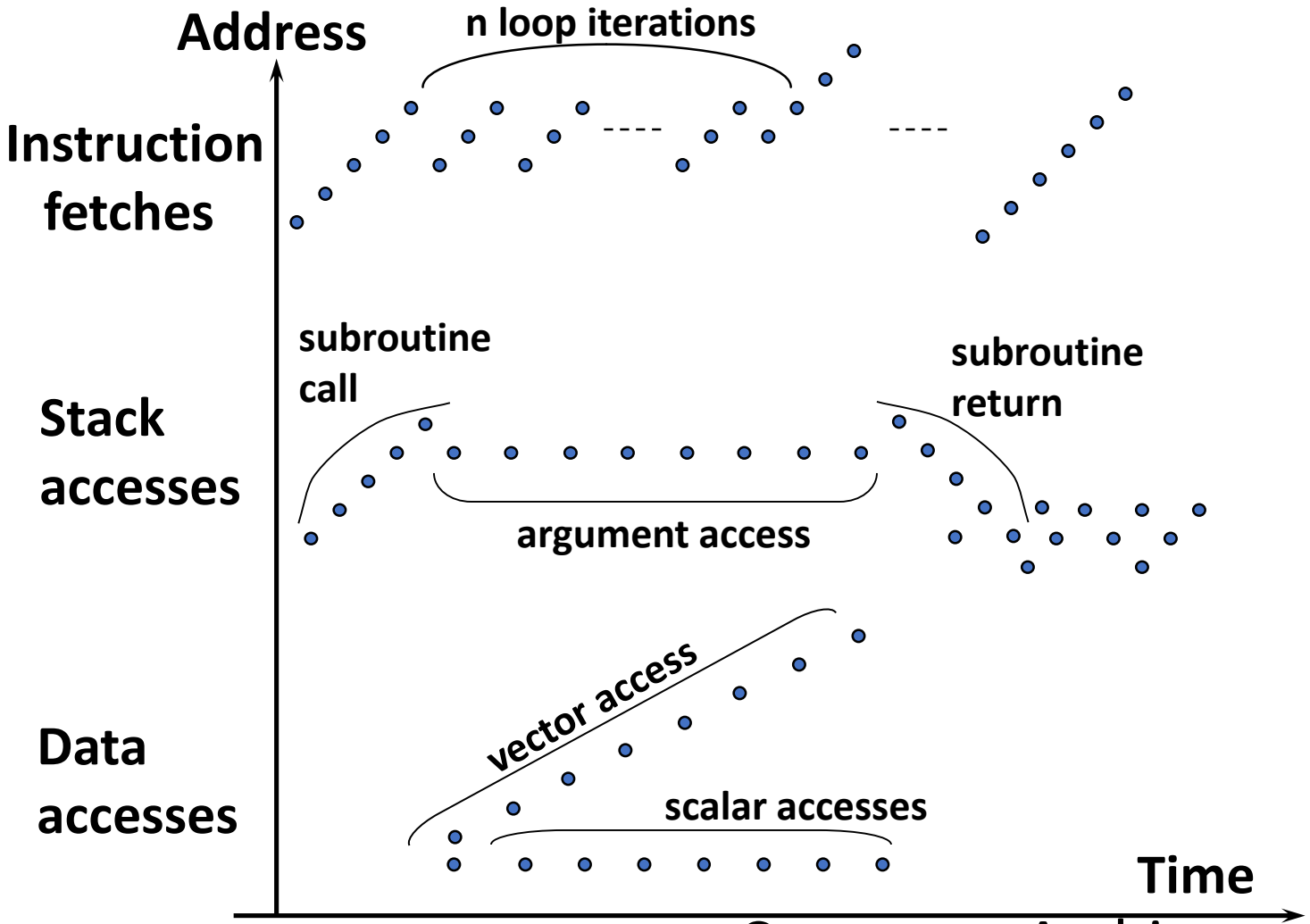
Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Oh Yes locality



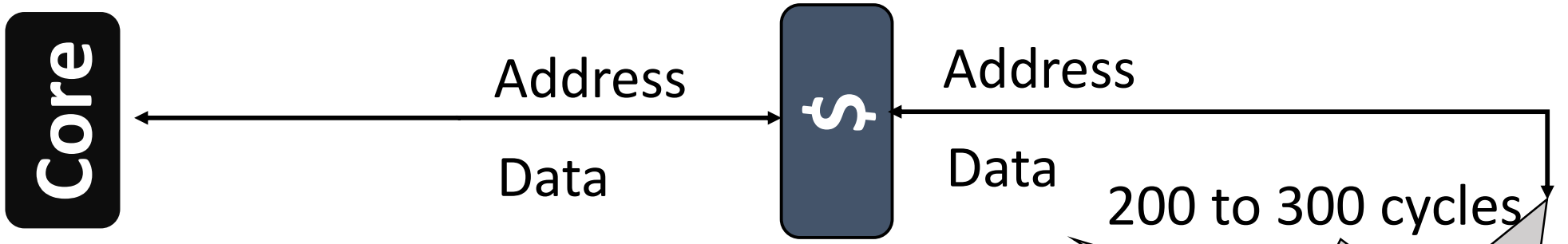
Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Few Examples



Caching: 10K Feet View

North pole ☺

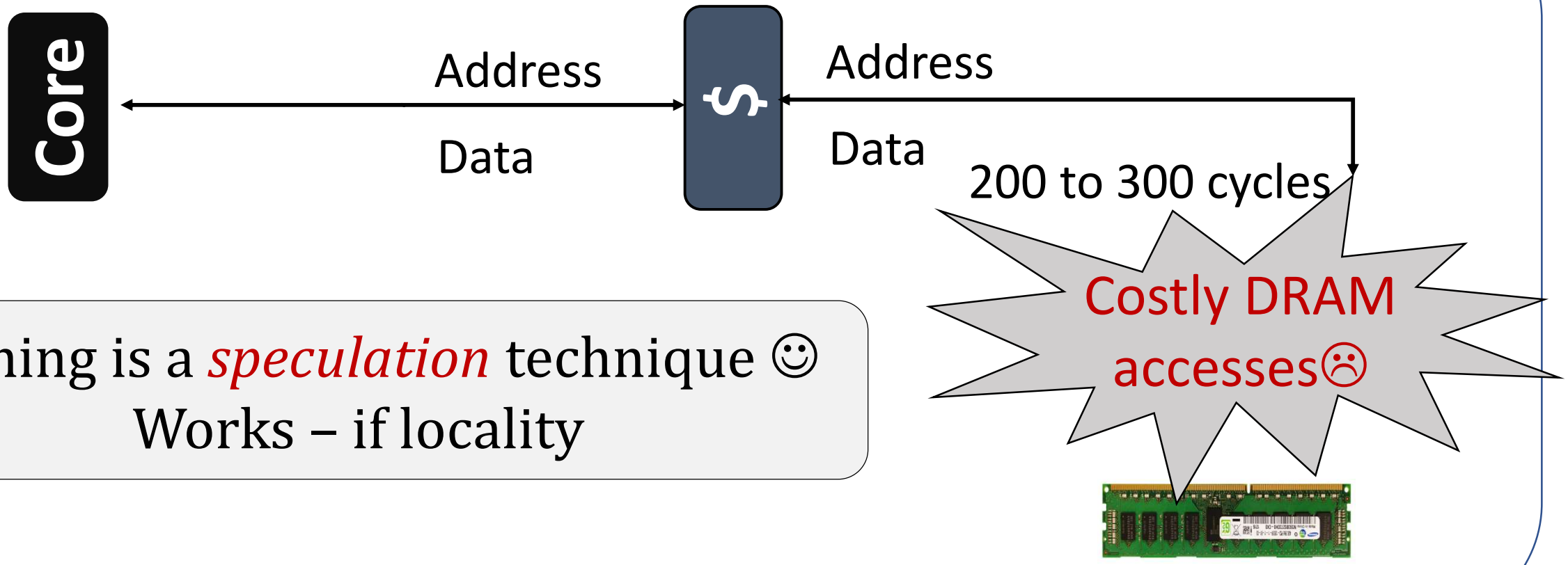


Costly DRAM accesses☹

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

Caching: 10K Feet View

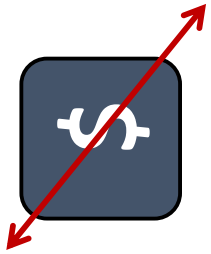
North pole ☺



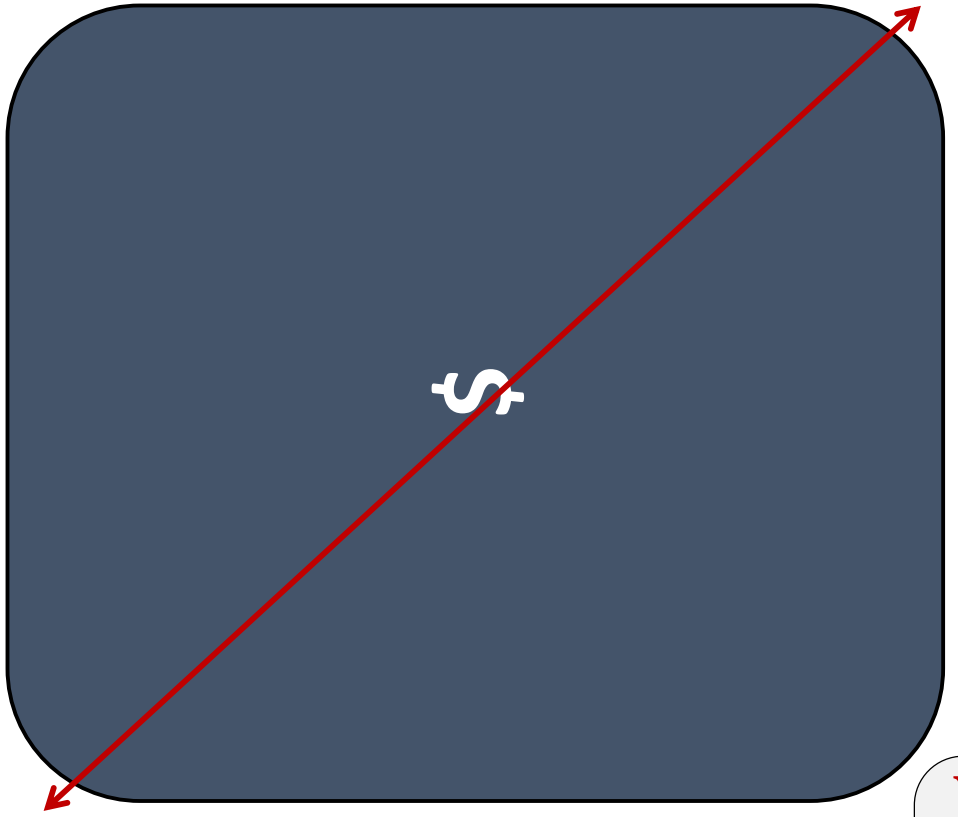
Caching is a *speculation* technique ☺
Works – if locality

How big/small?

Core

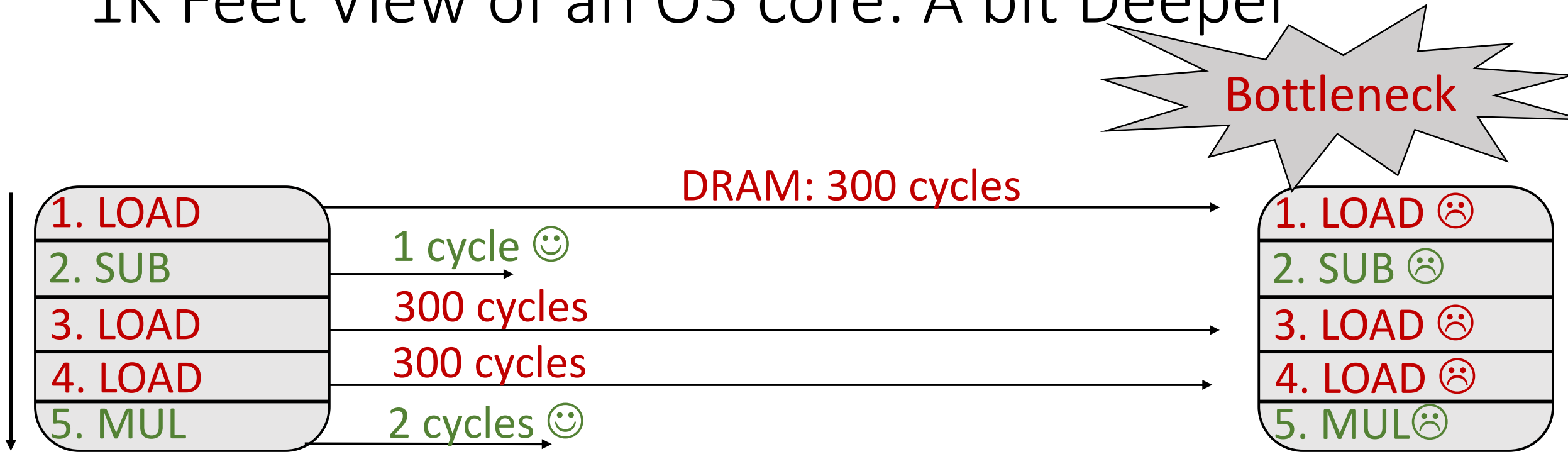


Latency: low
Area: low
Capacity: low



Latency: high
Area: high
Capacity: high

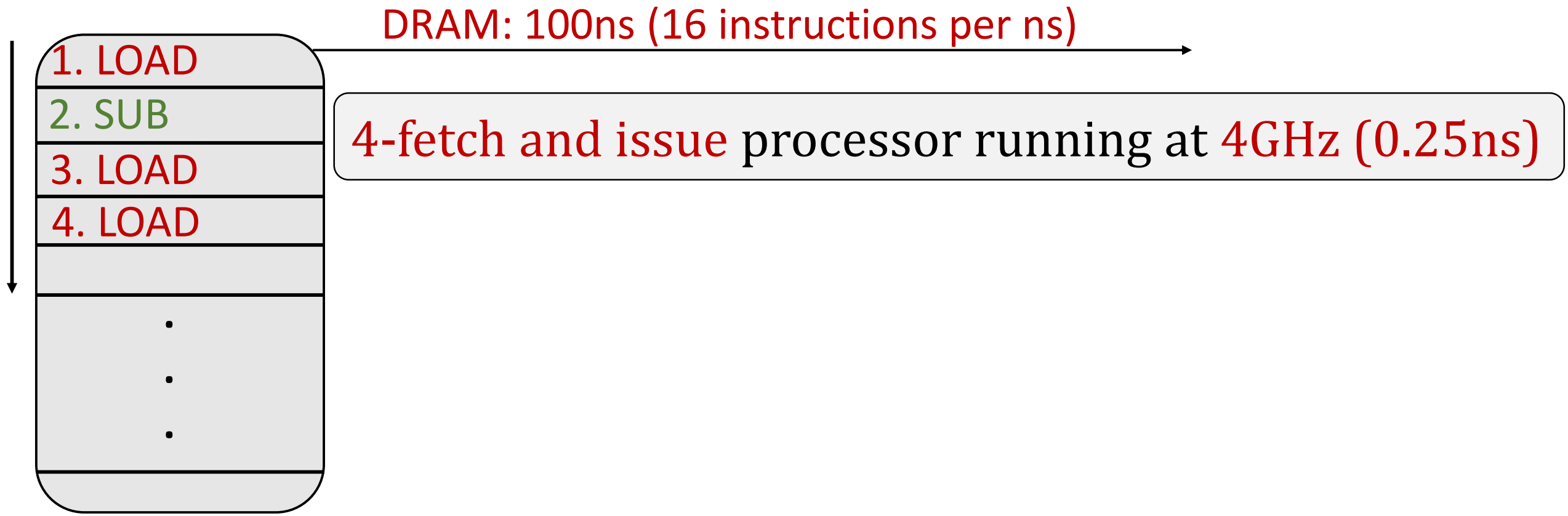
1K Feet View of an O3 core: A bit Deeper



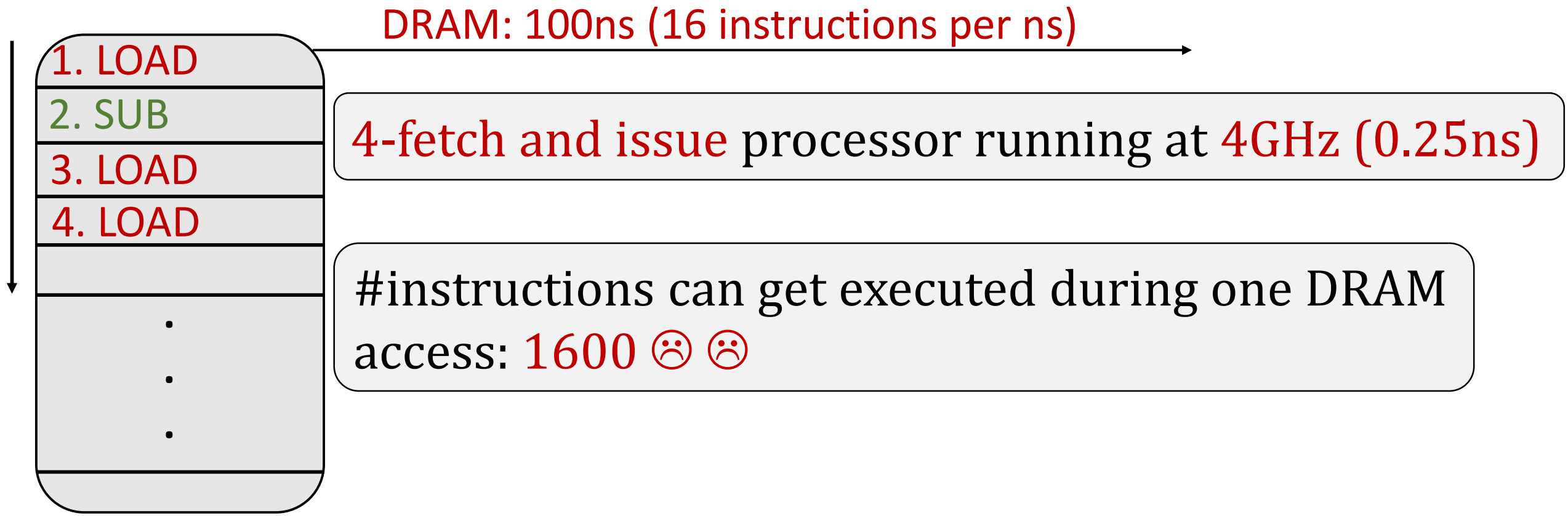
In-order Instruction Fetch
(Multiple fetch in one cycle)

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

Impact of one DRAM access

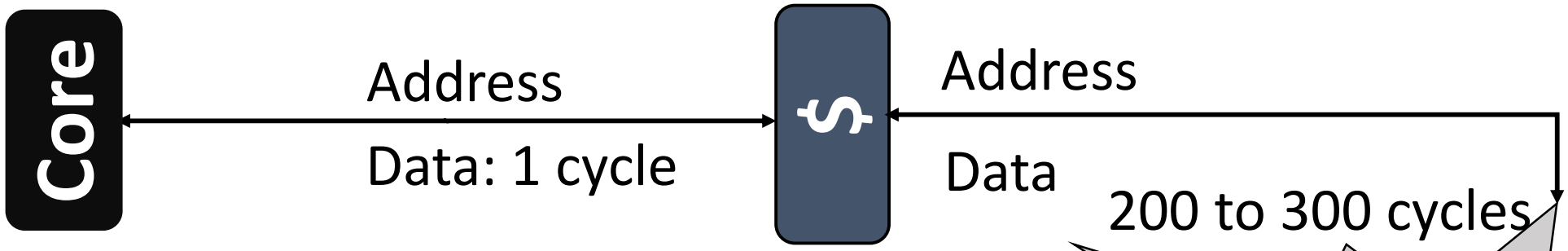


Impact of one DRAM access



Cache with latency

North pole ☺

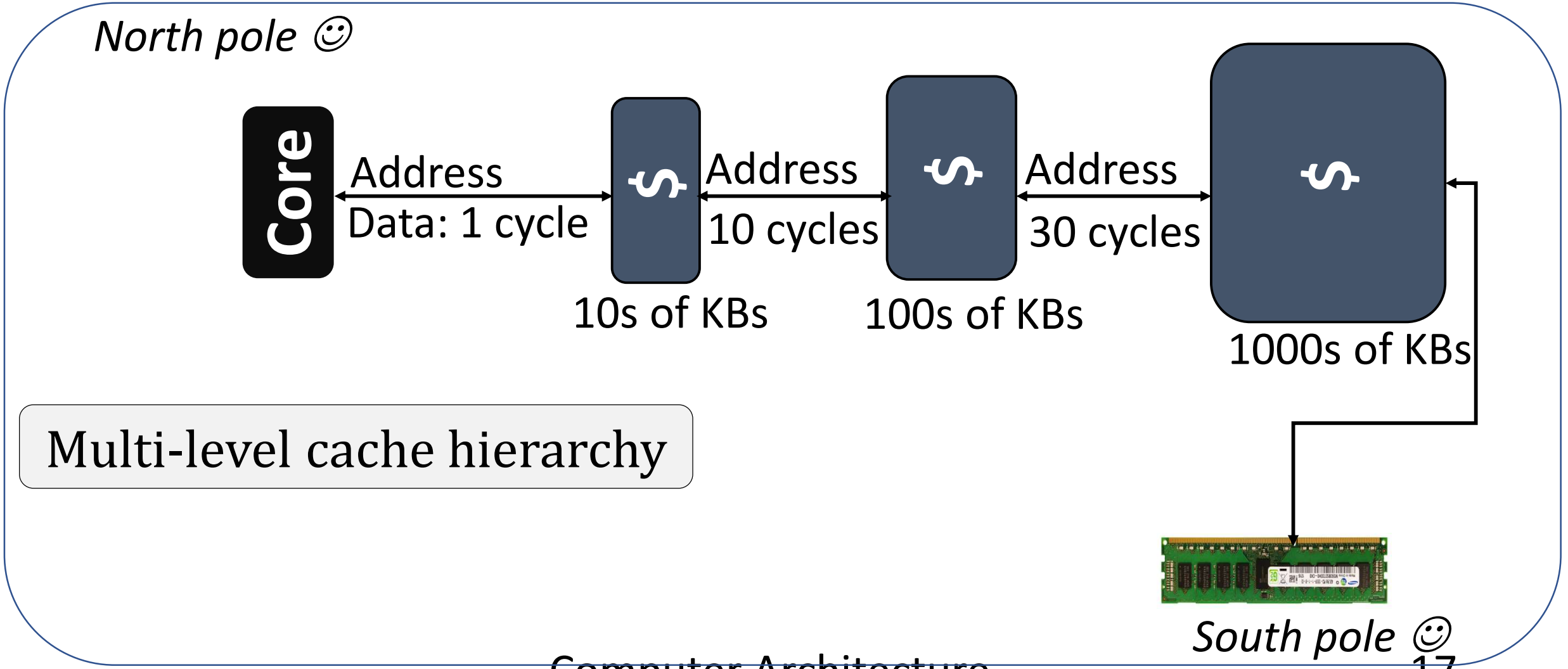


32 to 64KB $ will be available in one to four cycles ☹

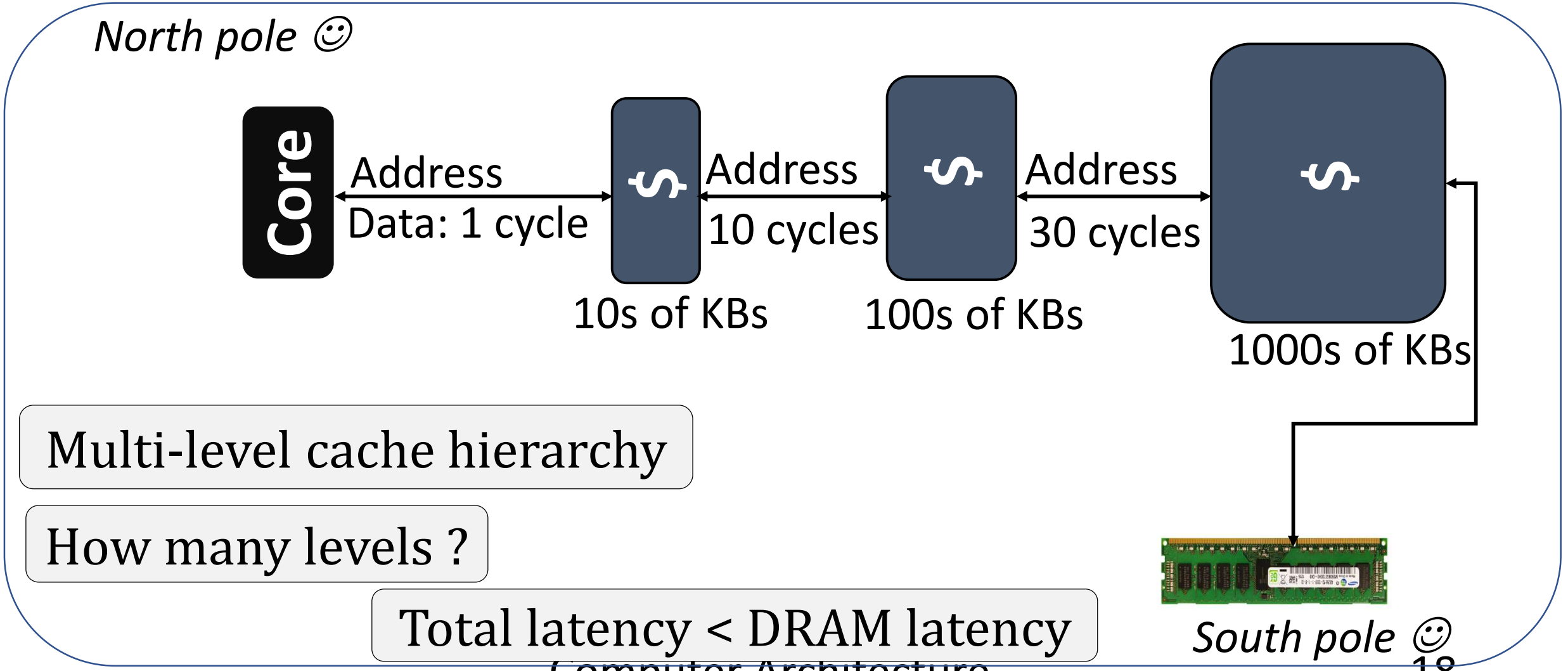
Costly DRAM accesses☹

South pole ☺

Cache hierarchy with latency



Cache hierarchy with latency

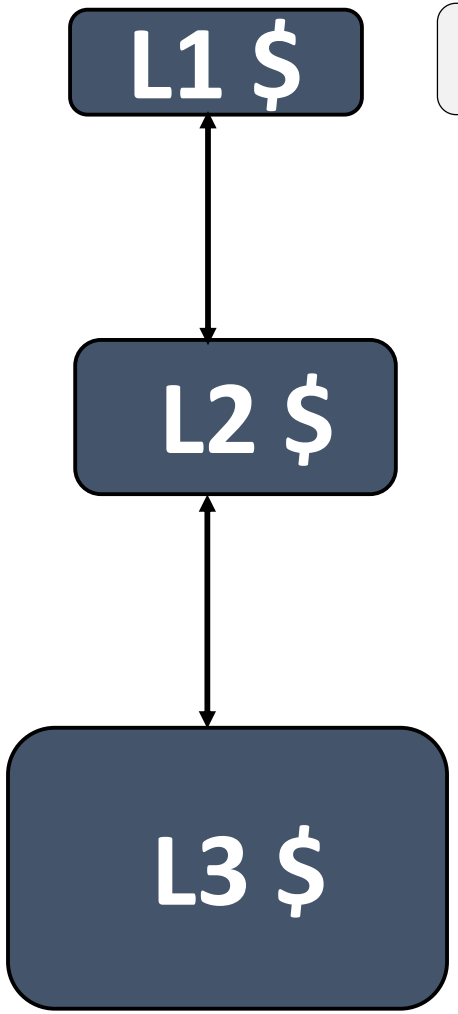


Multi-level cache hierarchy

How many levels ?

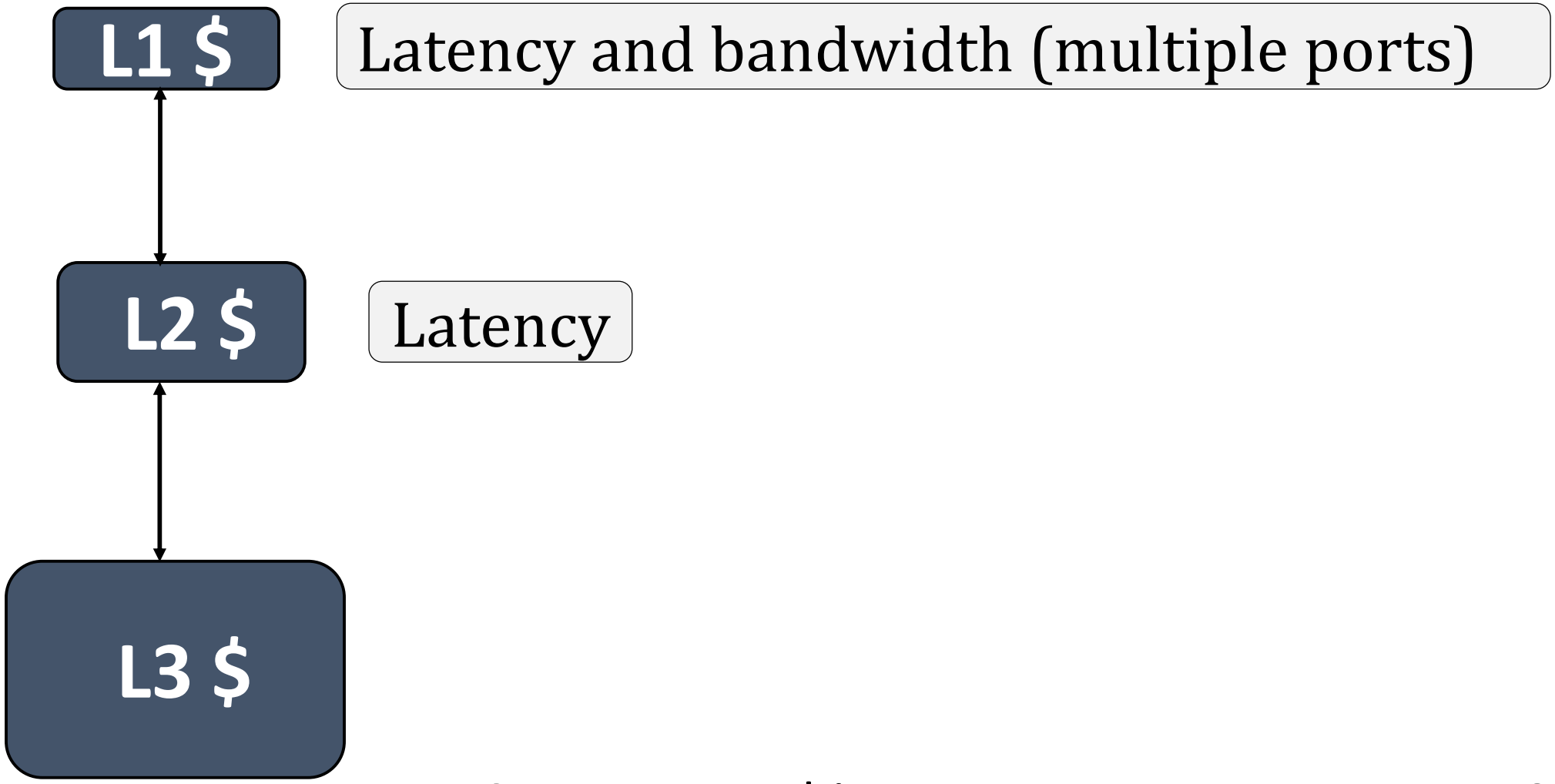
Total latency < DRAM latency

Takeaway

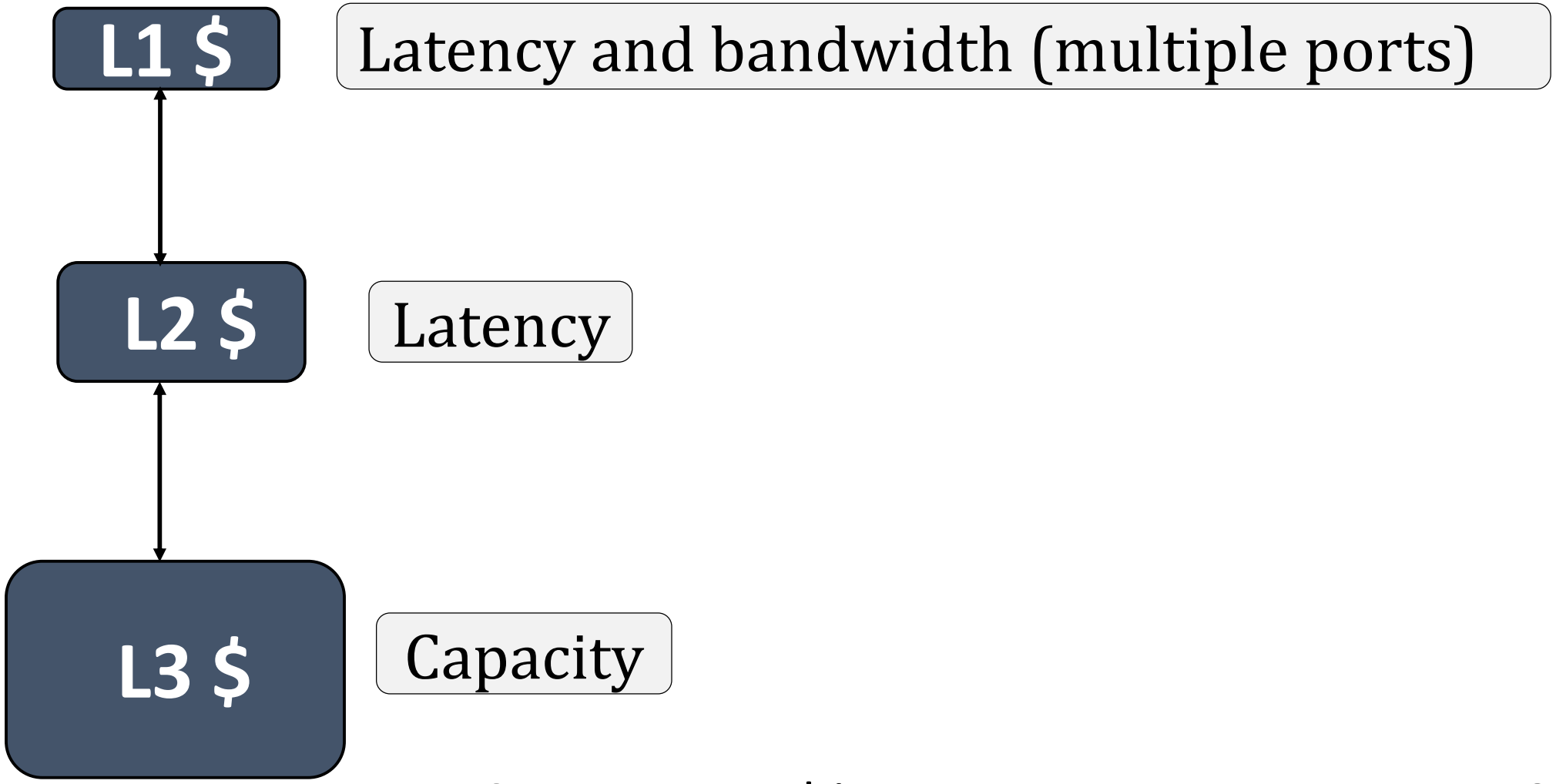


Latency and bandwidth (multiple ports)

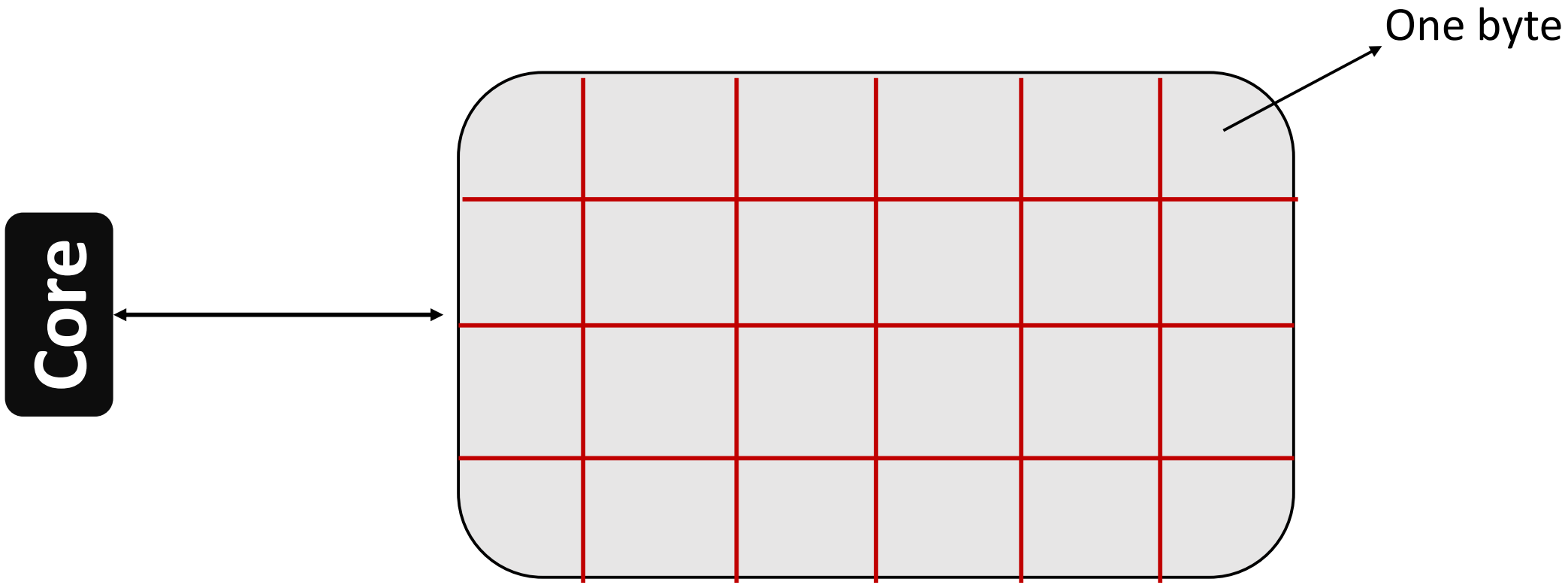
Takeaway



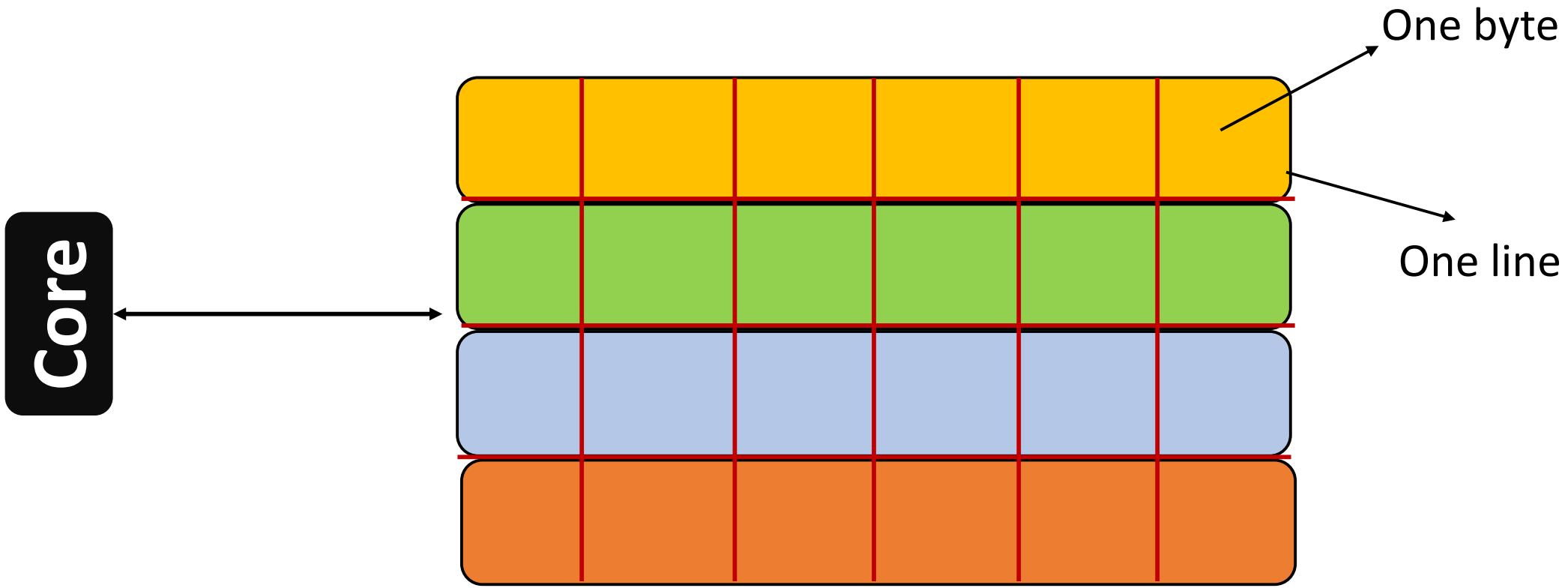
Takeaway


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

Accessing a cache

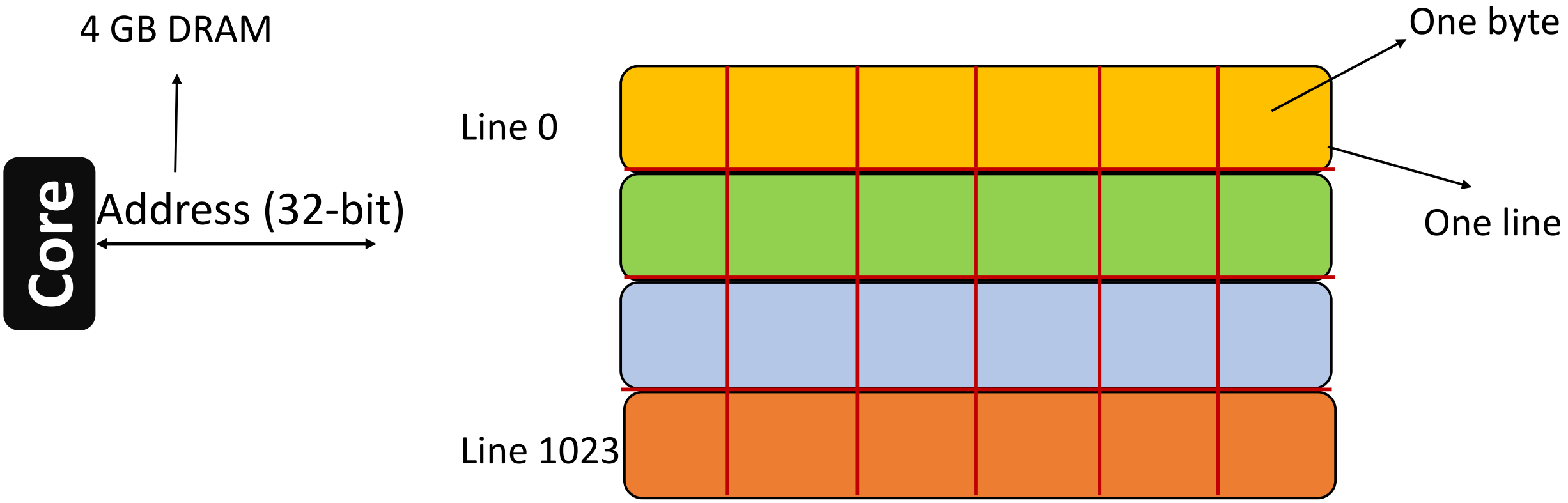


Bytes to blocks (lines)

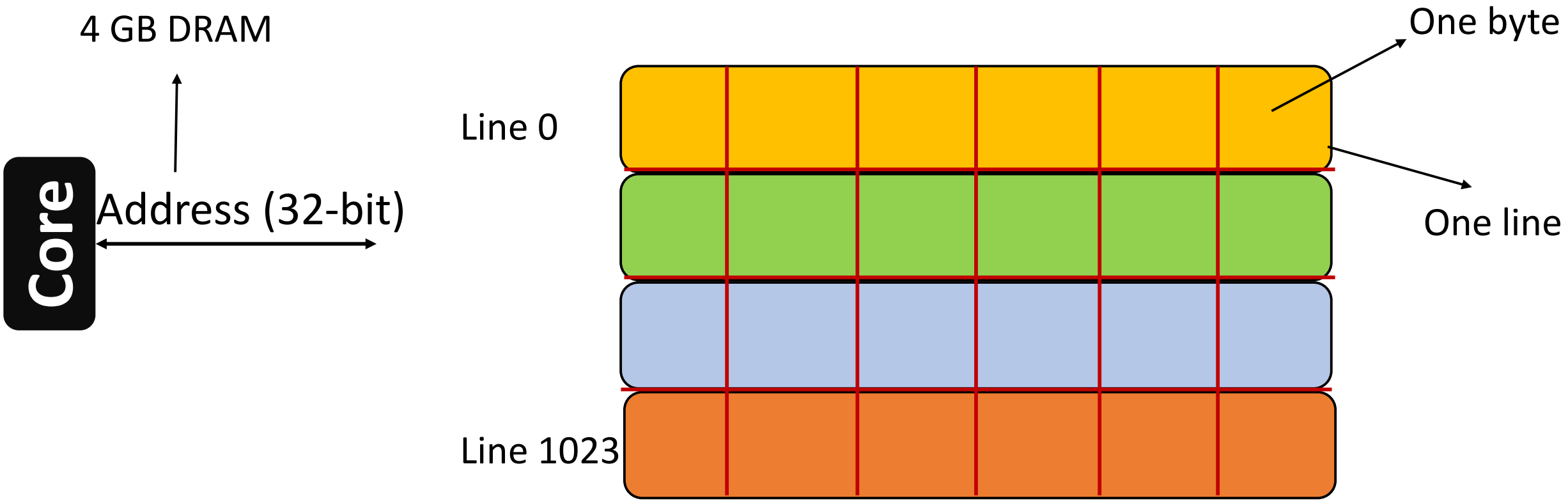


Typical line size: 64 to 128 Bytes

A bit deeper: 1024 lines each of 32B



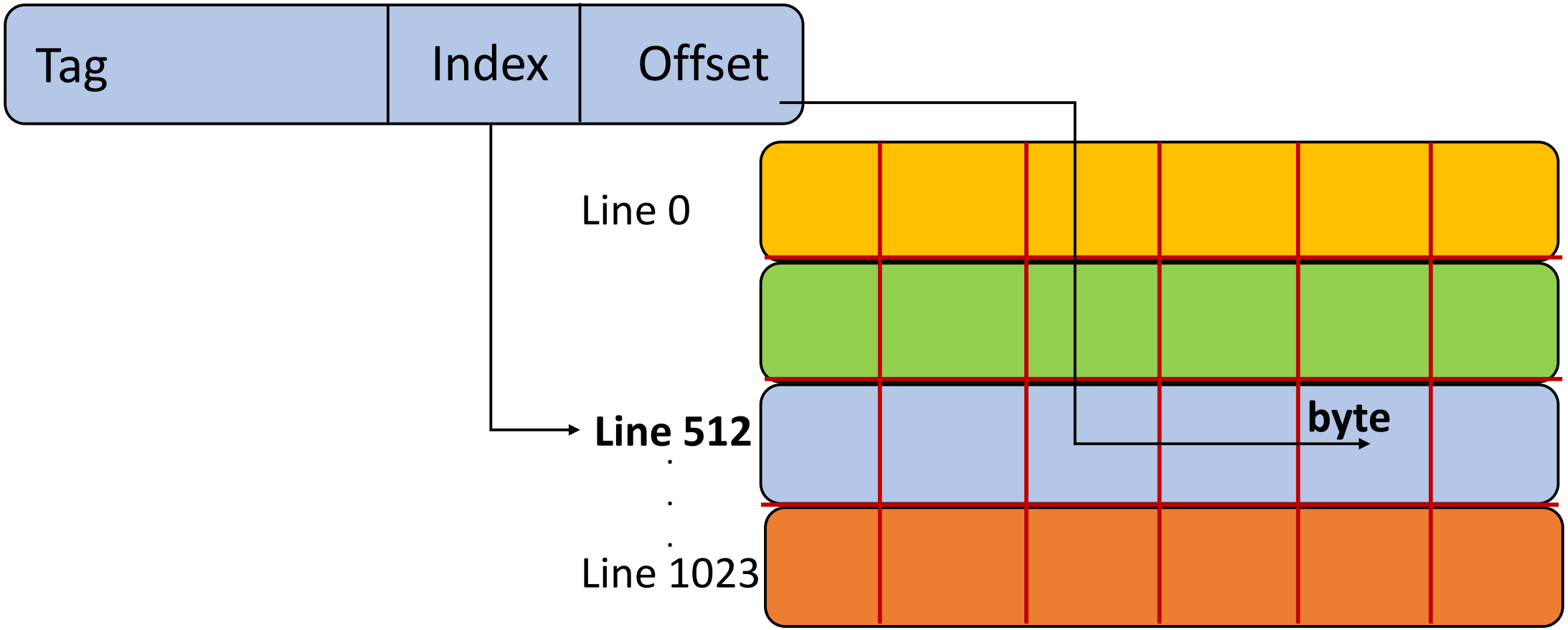
A bit deeper: 1024 lines each of 32B



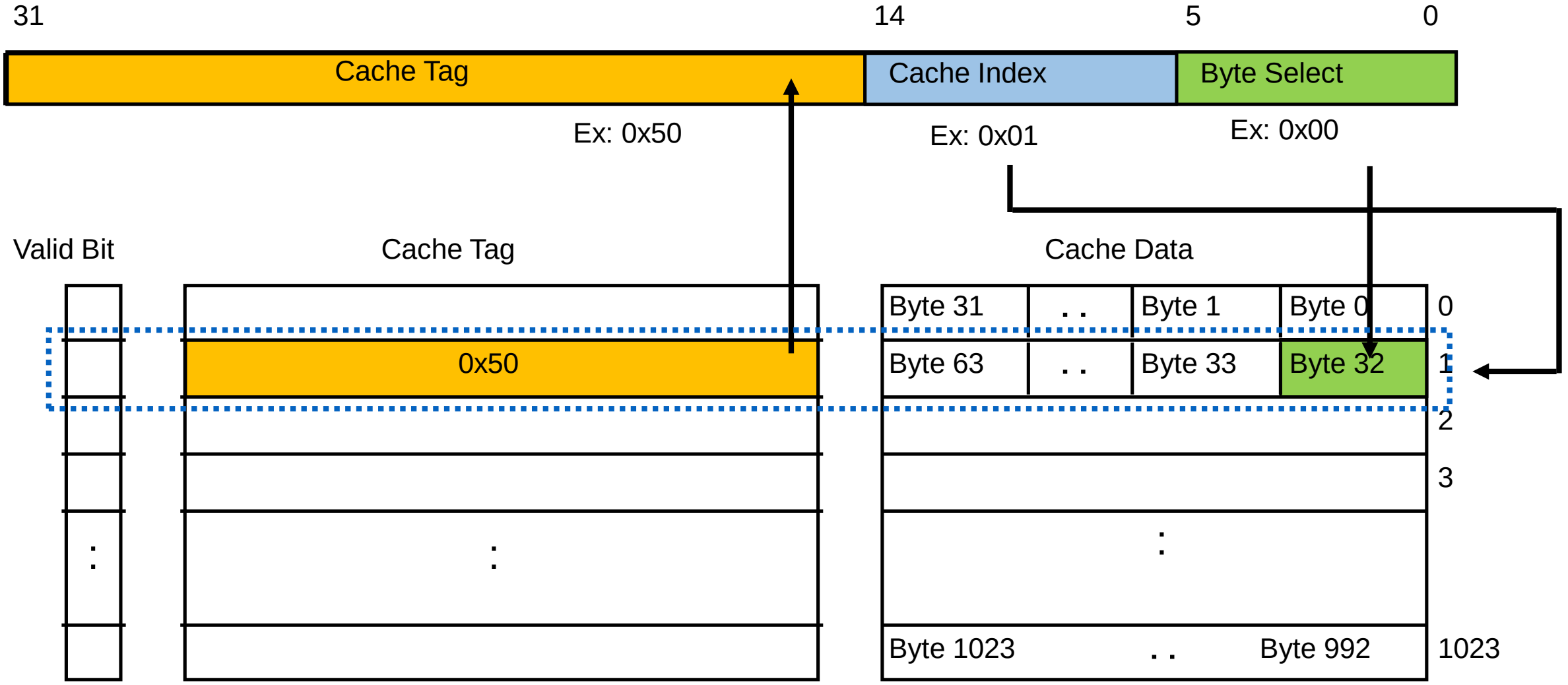
Line number (index): 10 bits

Byte offset (offset): 5 bits

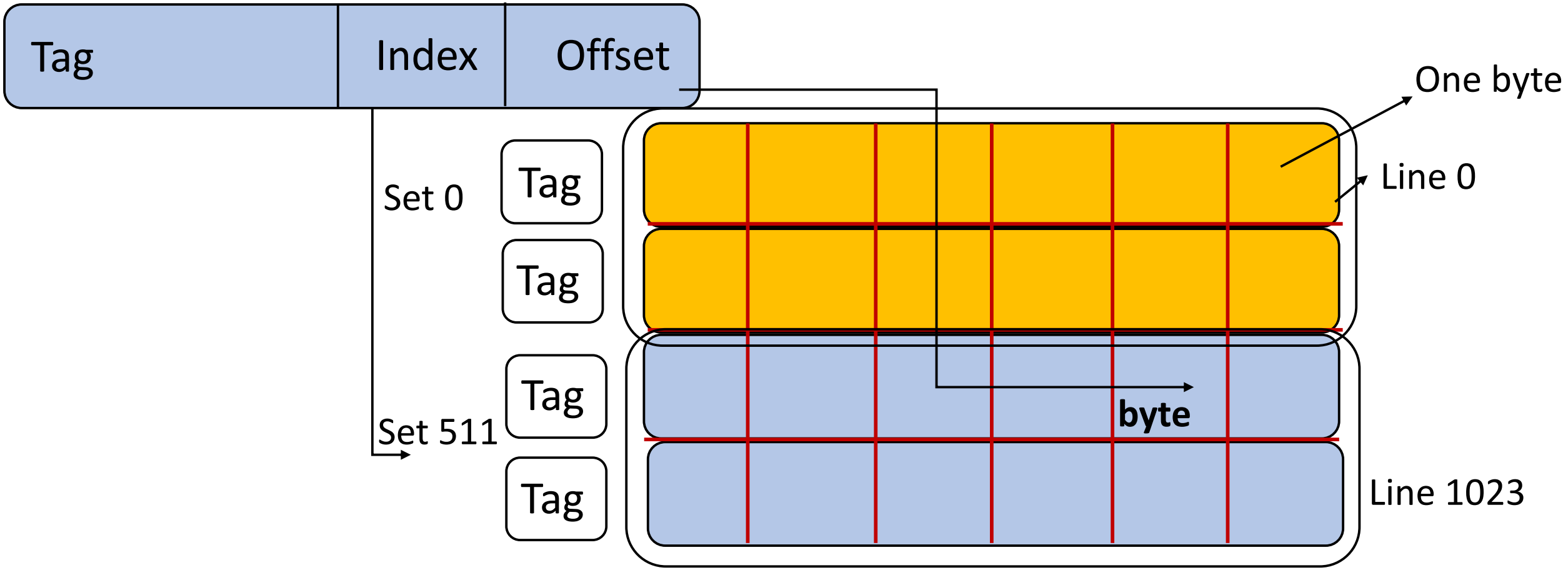
Direct Mapped Cache



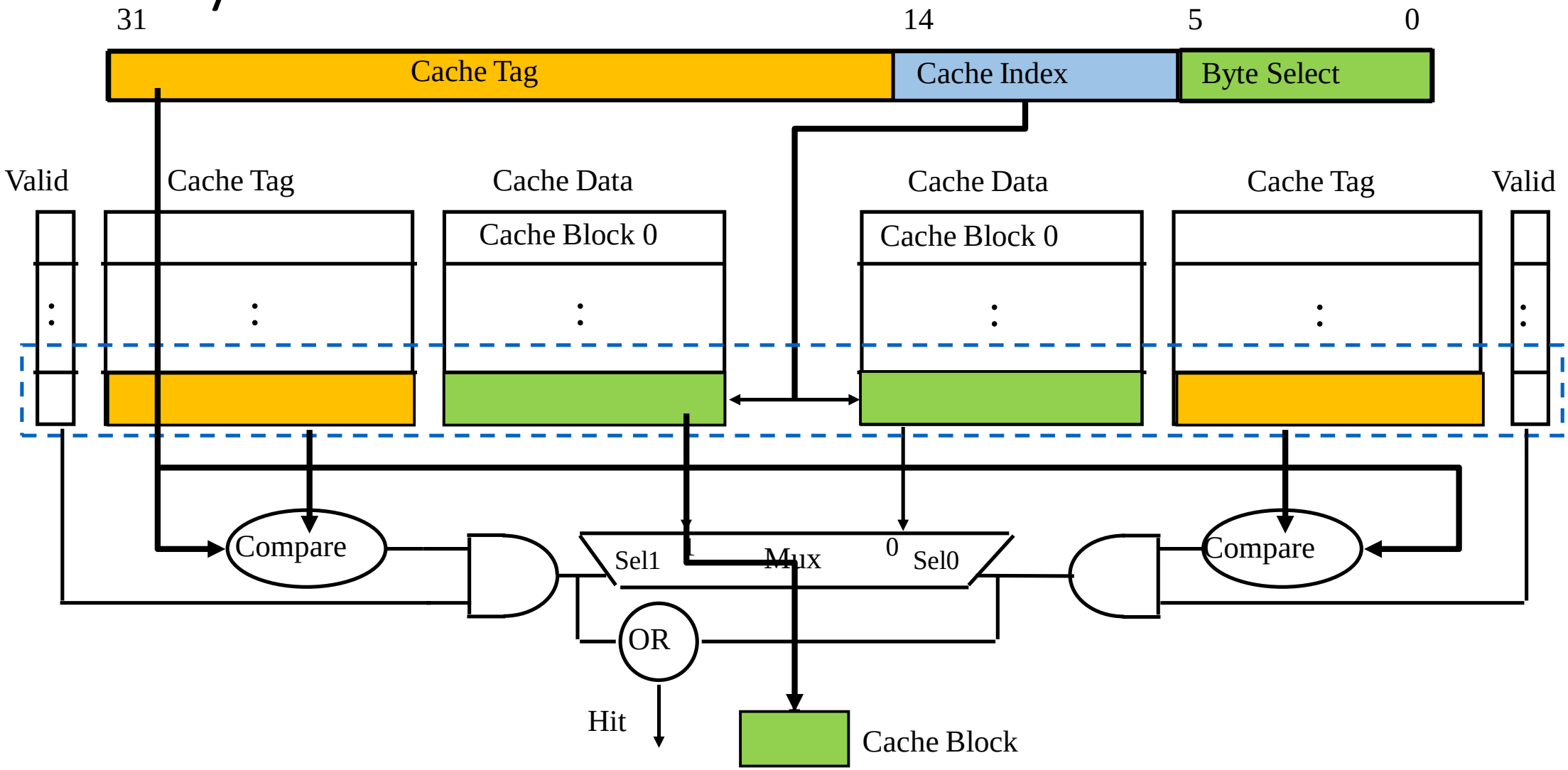
Direct Mapped in Action



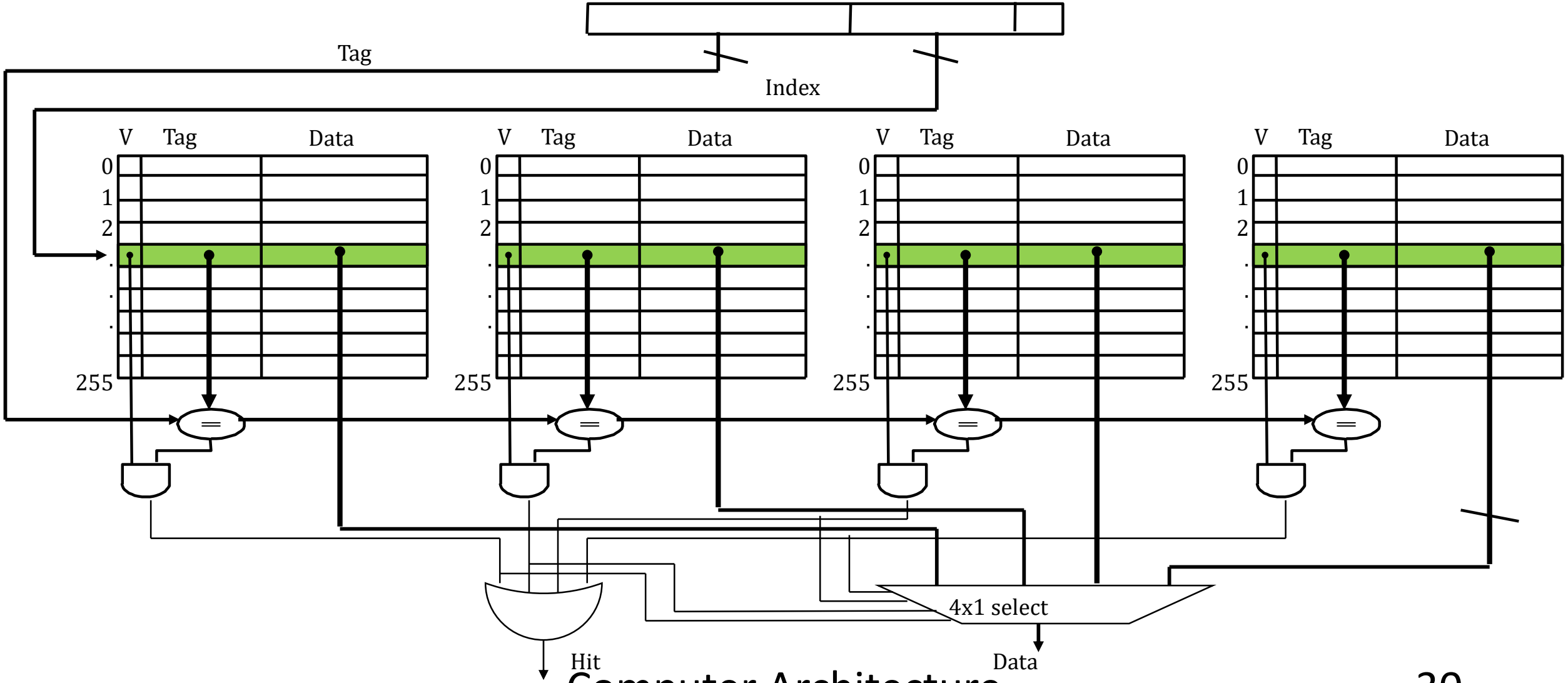
What if we have multiple ways?



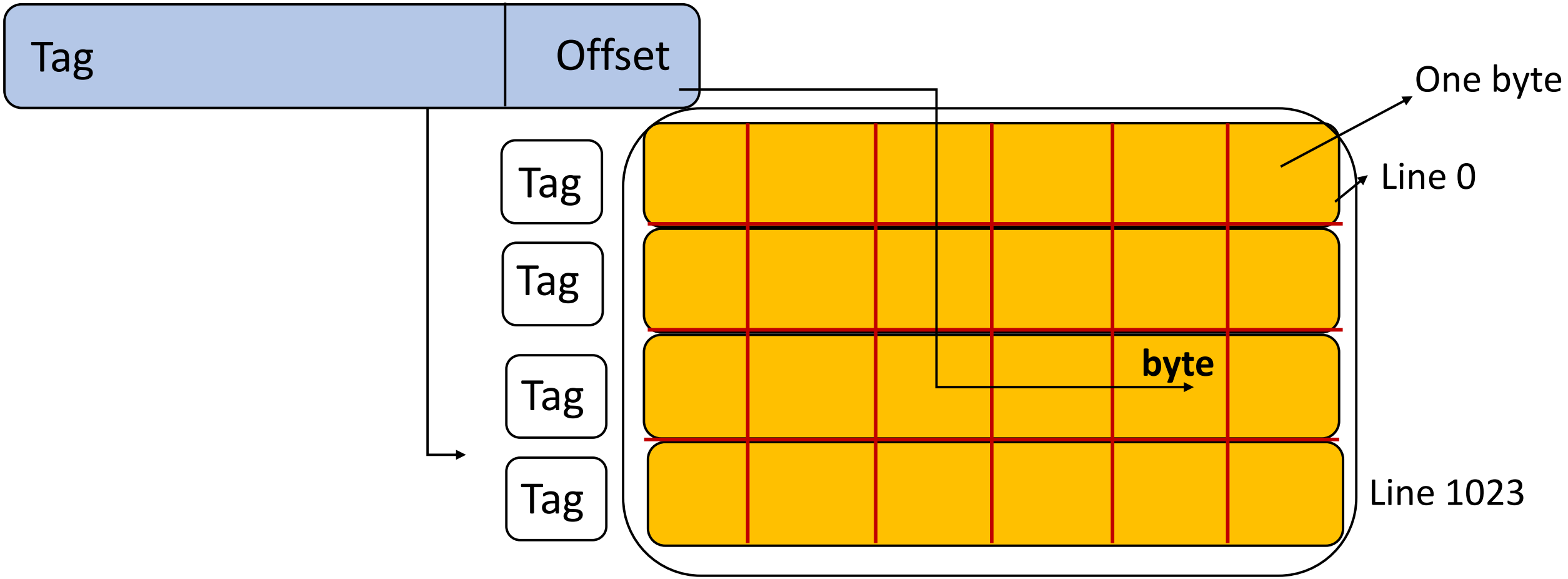
2-way associative in action



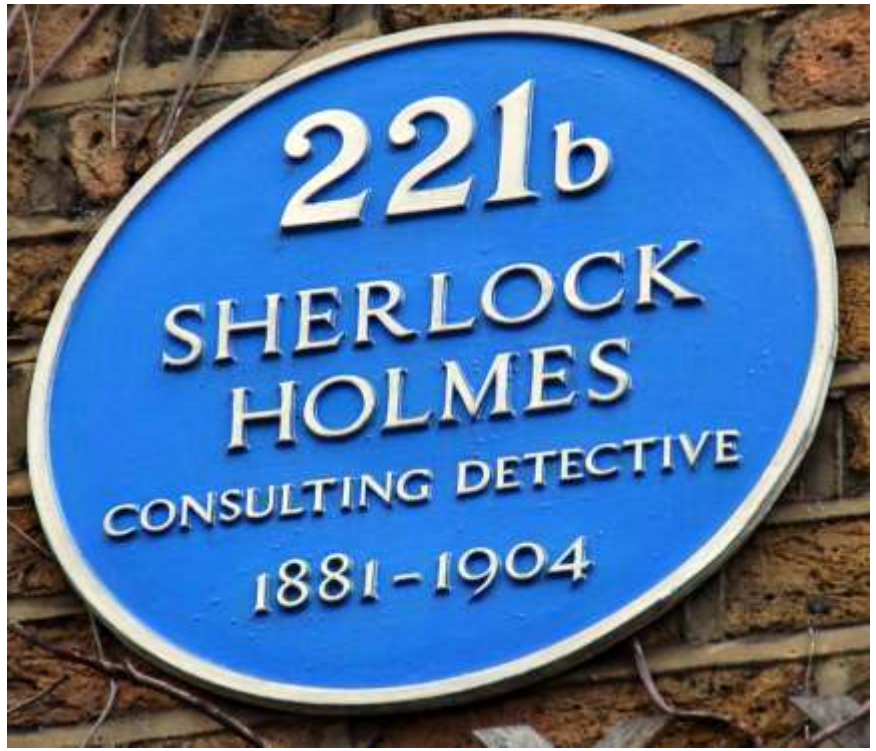
4-way associative: Just a better picture



Extreme: One cache, one set, fully associative



A bit different way



Baker Street: Cache Index ☺

221b: Tag bits ☺

Sherlock Holmes: Byte offset ☺ ☺