Arhitecturi Paralele

Memoria Cache

Dr. Ing. Cristian Chilipirea – cristian.chilipirea@gmail.ro

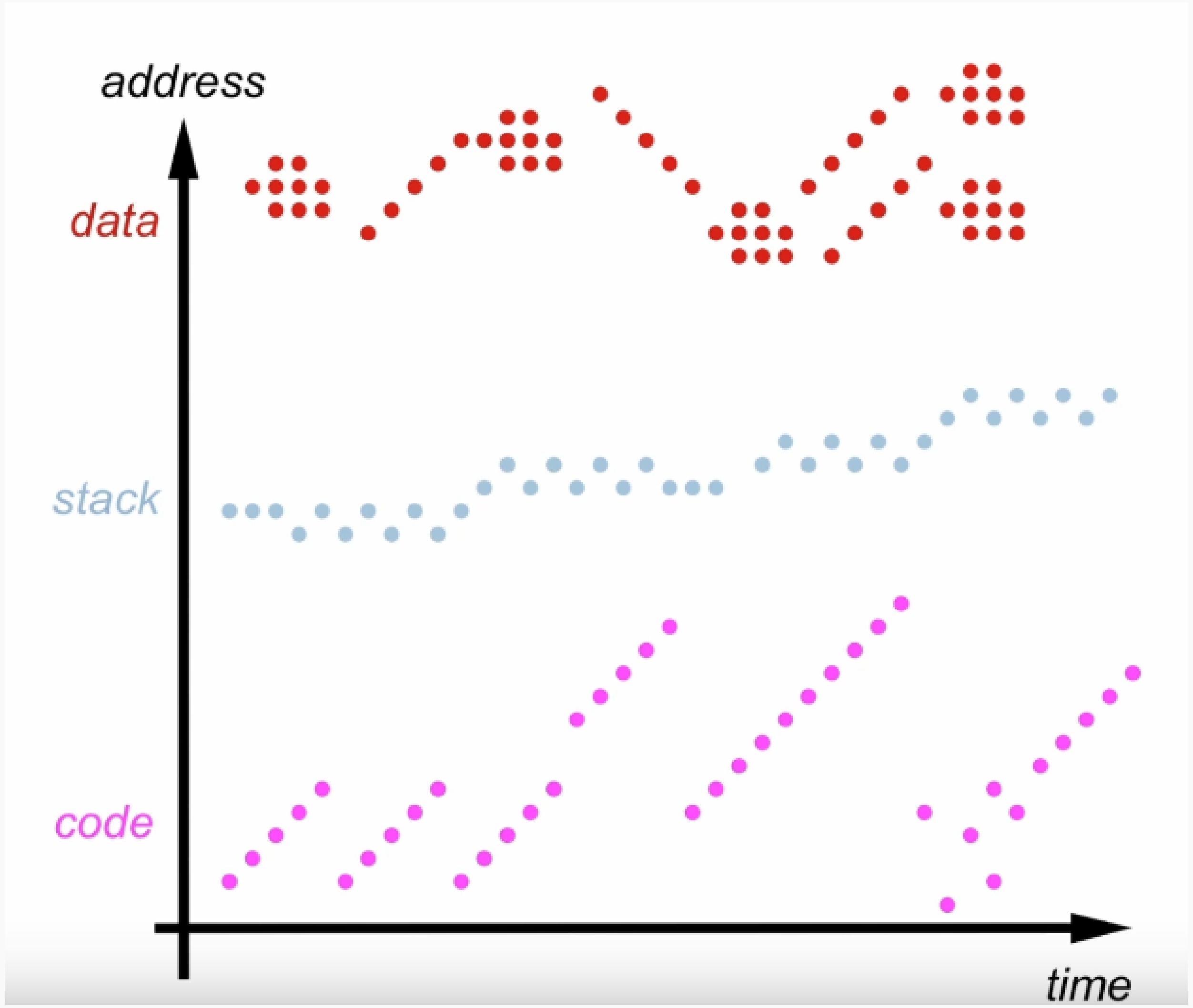
Ierarhia de memorii

	Capacitate	Latență	Cost/GB	Controlat de
Regiștri	x 1 B	1ns <	-	Compilator
Cache (SRAM)	x 10MB	1-10ns	$5000	Hardware
RAM (DRAM)	x 100GB	70-100ns	$50	Hardware/Kernel
SSD (Flash)	x 1TB	7-150µs	$1	Kernel
HDD (Magnetic)	x 19TB	1-10ms	$0.1	Kernel

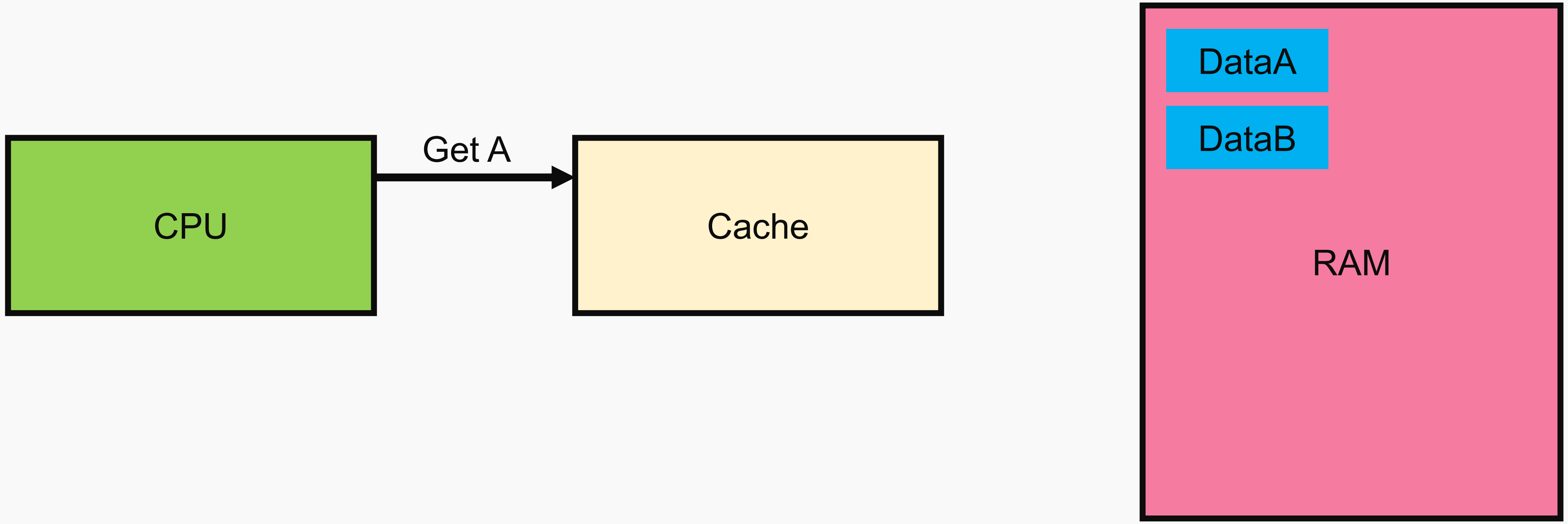
Principiul “locality”

- Dacă avem un acces asupra unei date de la locația X, un acces la locația X+ΔX la un moment de timp t+Δt devine mai probabil cu cât ΔX și Δt se apropie de 0.
- **Localitate spațială**: Două zone de memorie apropiate vor fi accesate la un interval de timp apropiat.
- **Localitate temporală**: Dacă o zonă de memorie e accesată, sunt șanse mari să fie accesată din nou.

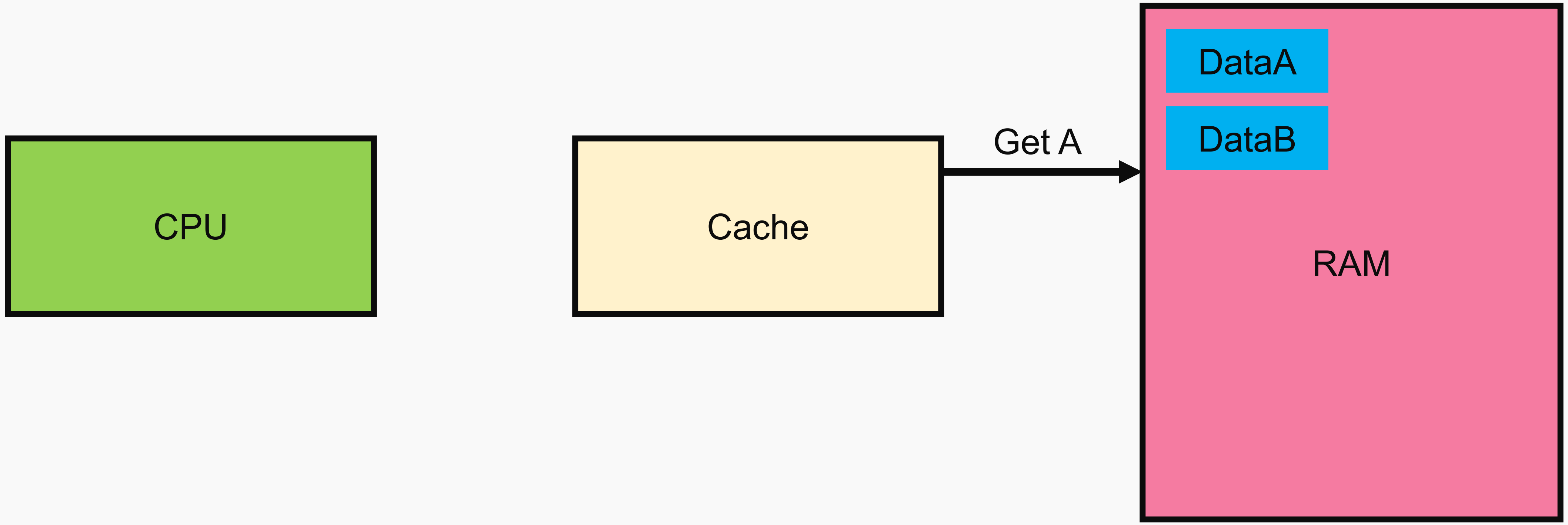
Pattern acces la memorie



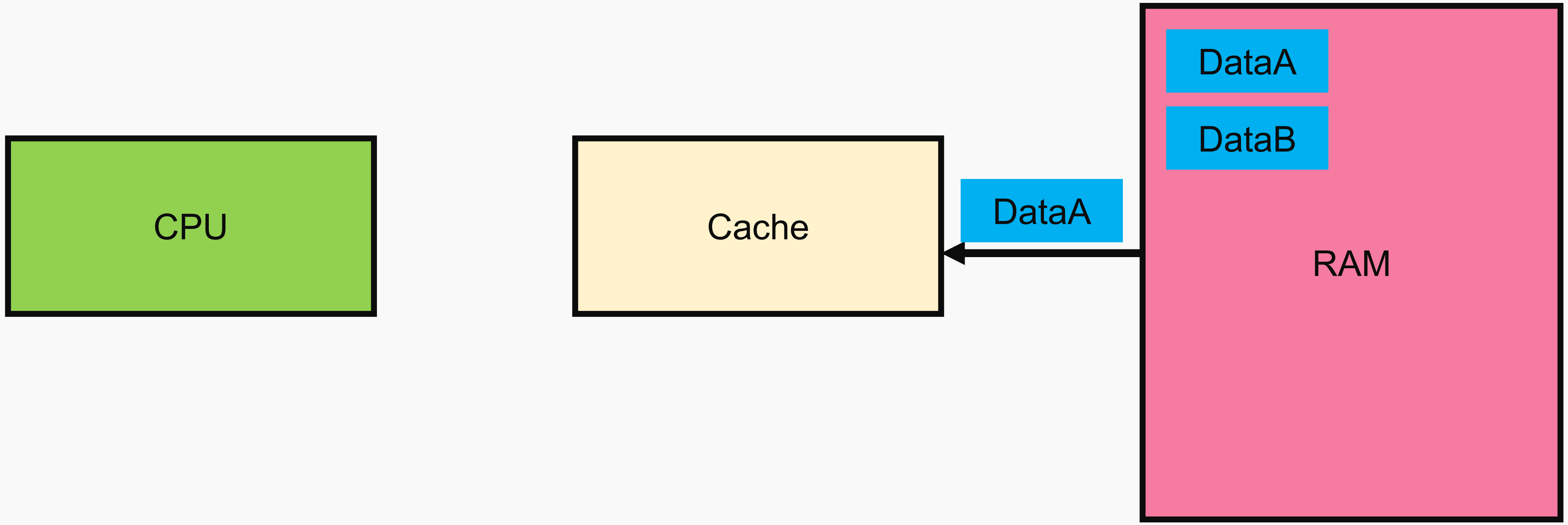
Utilizare



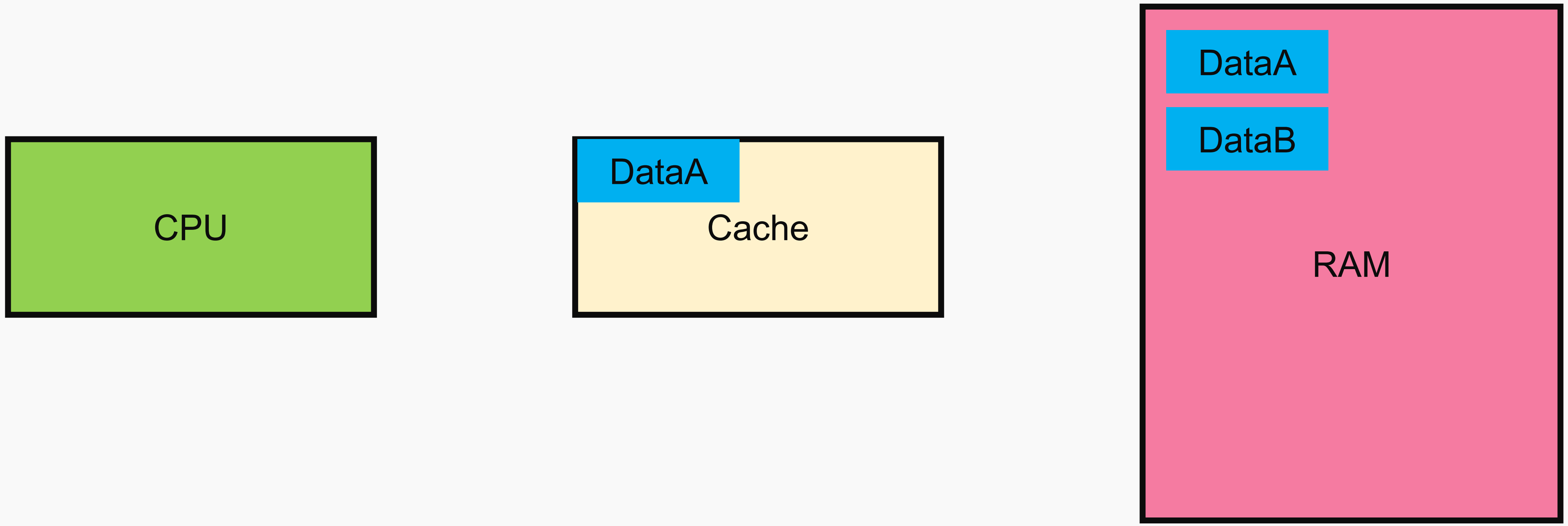
Utilizare



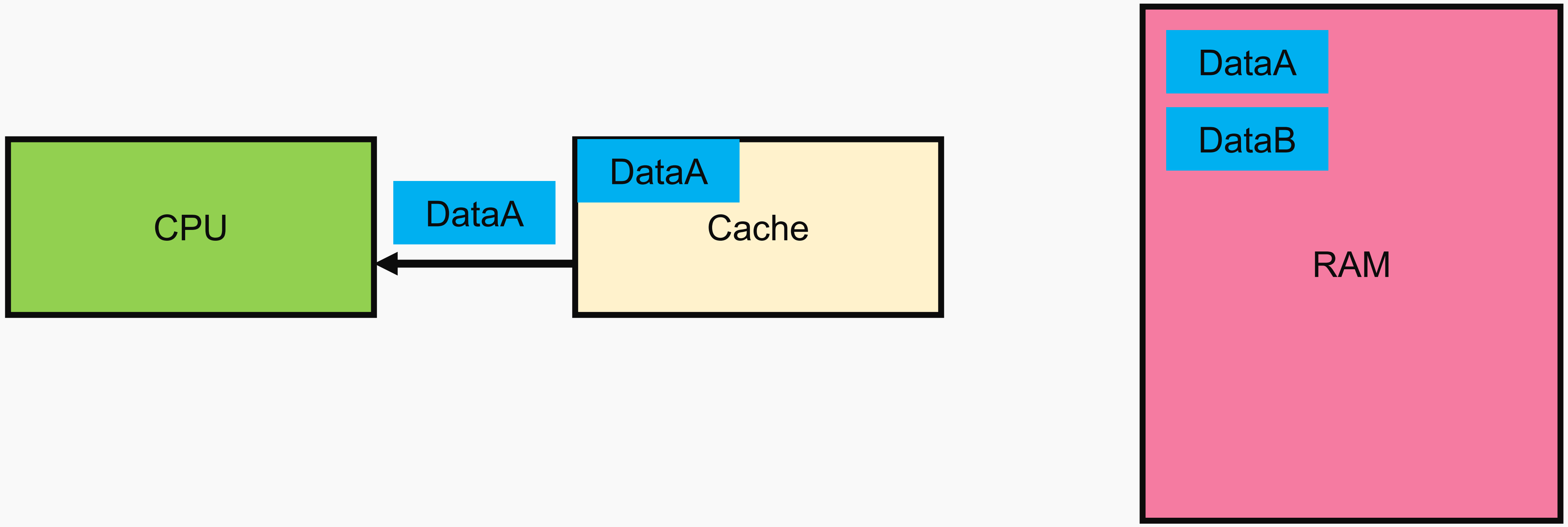
Utilizare



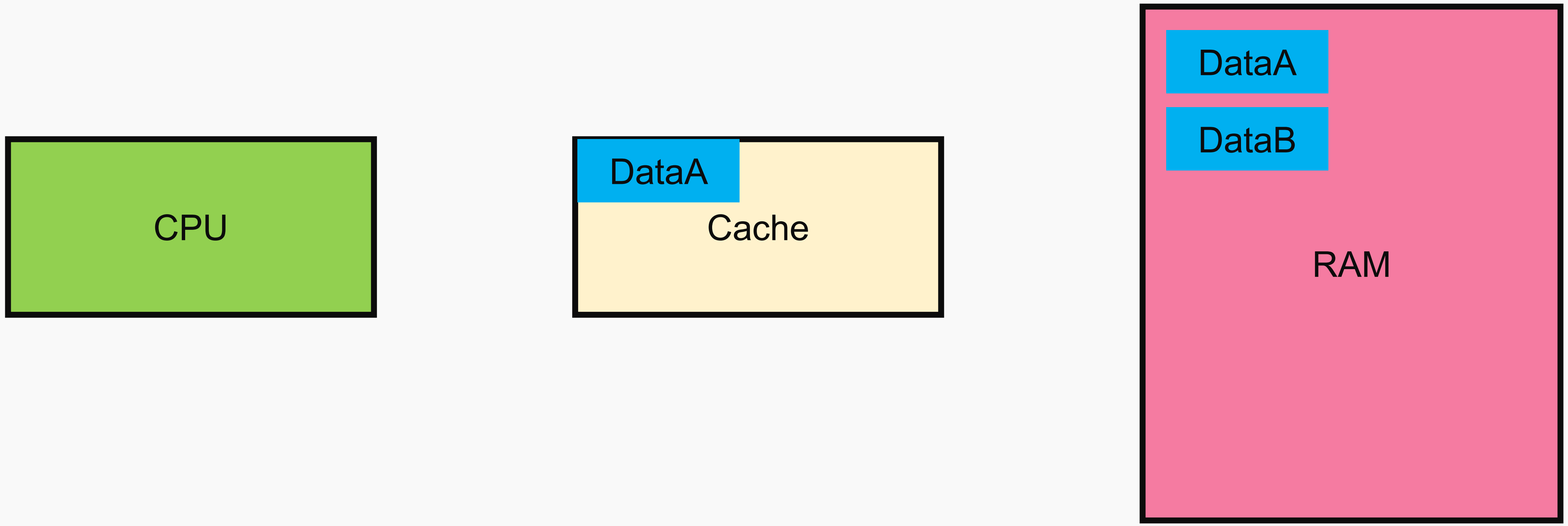
Utilizare



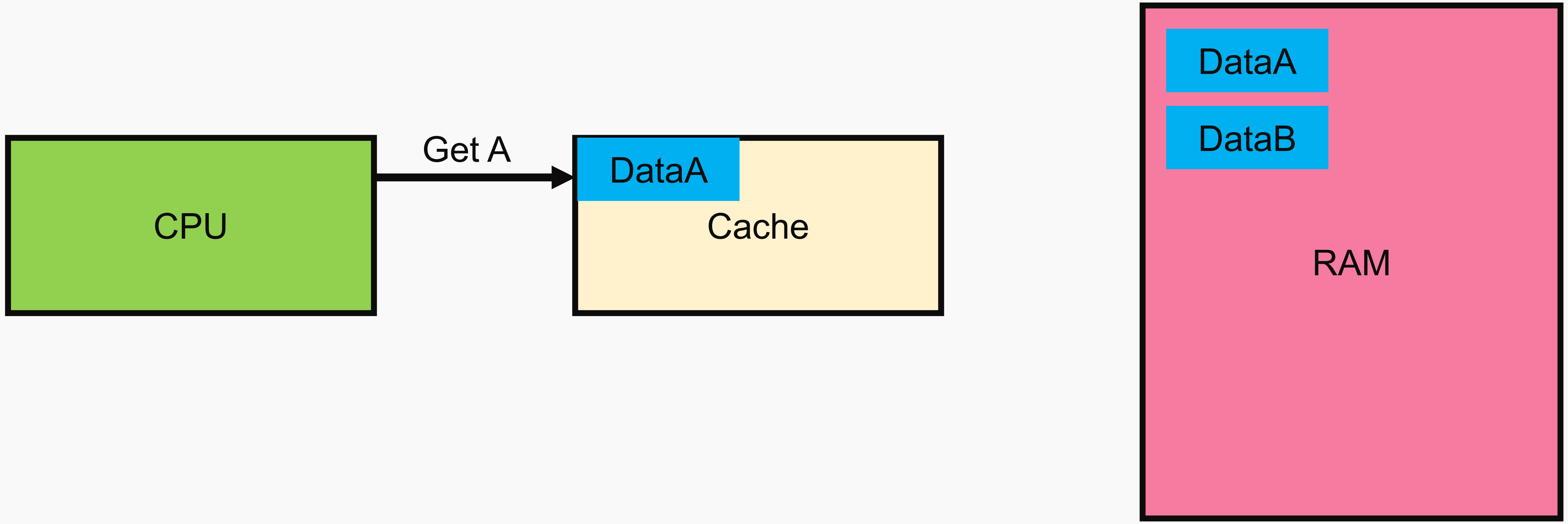
Utilizare



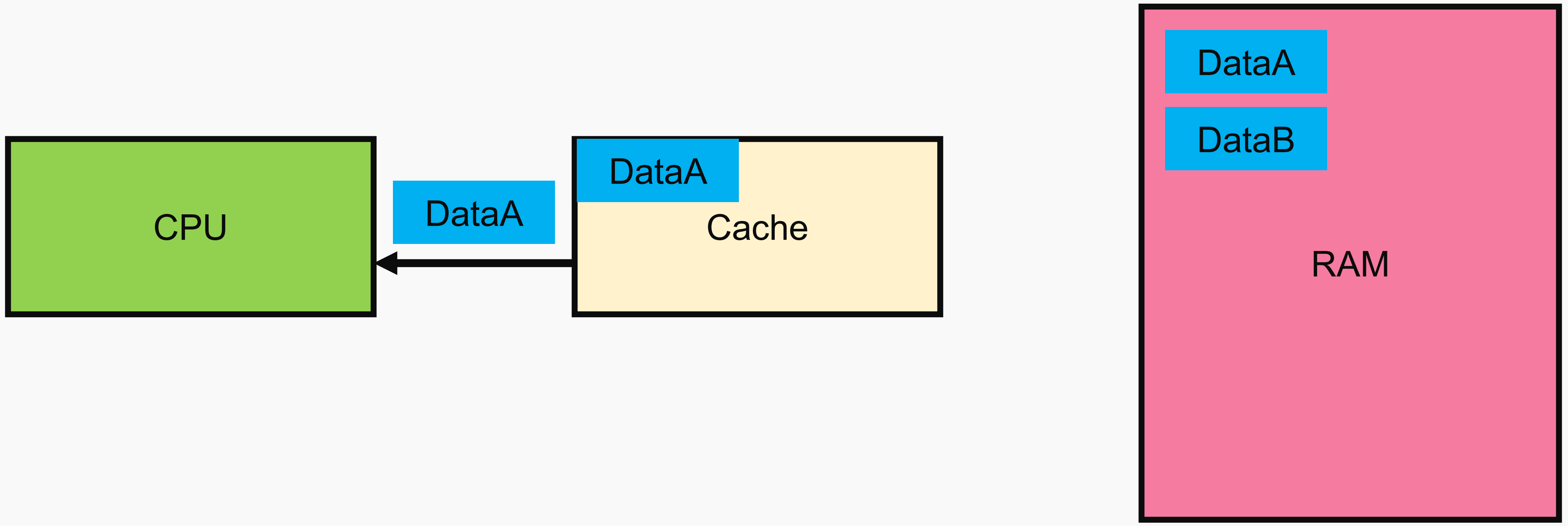
Utilizare



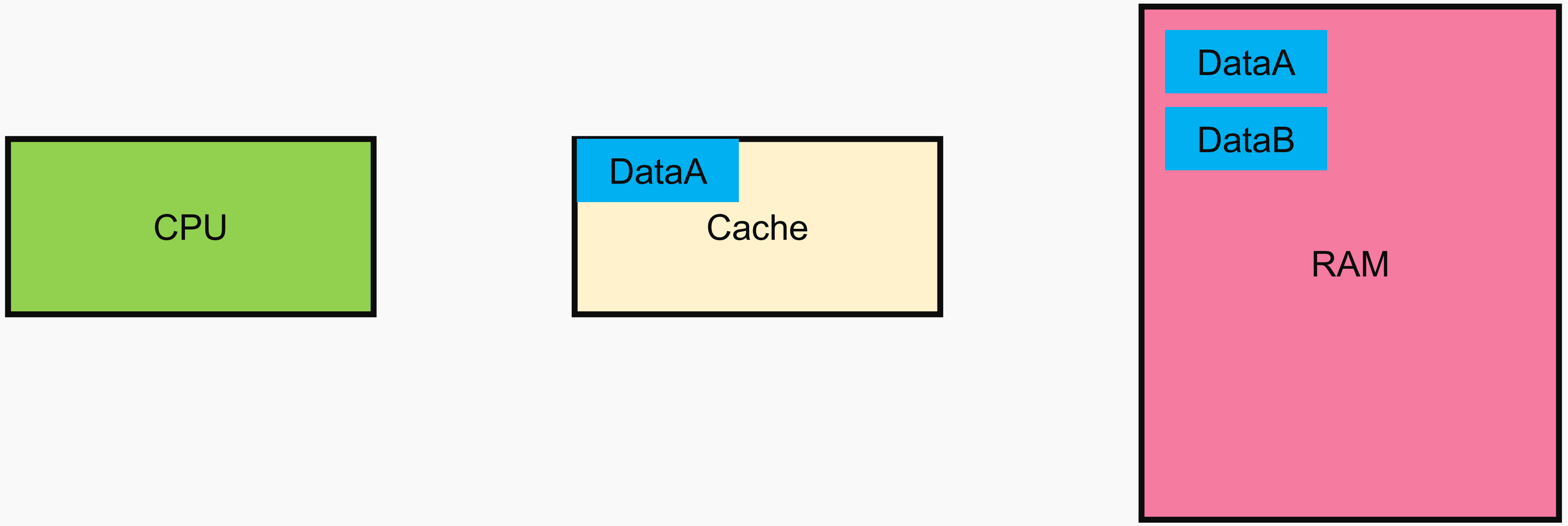
Utilizare – Cache Hit



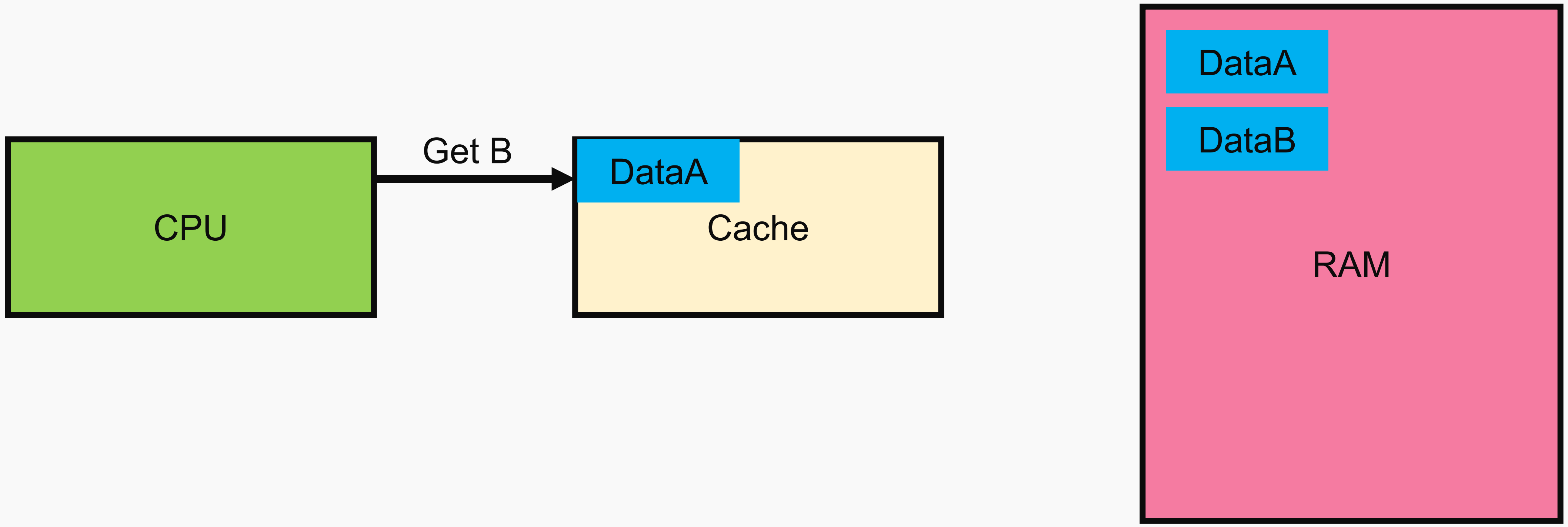
Utilizare – Cache Hit



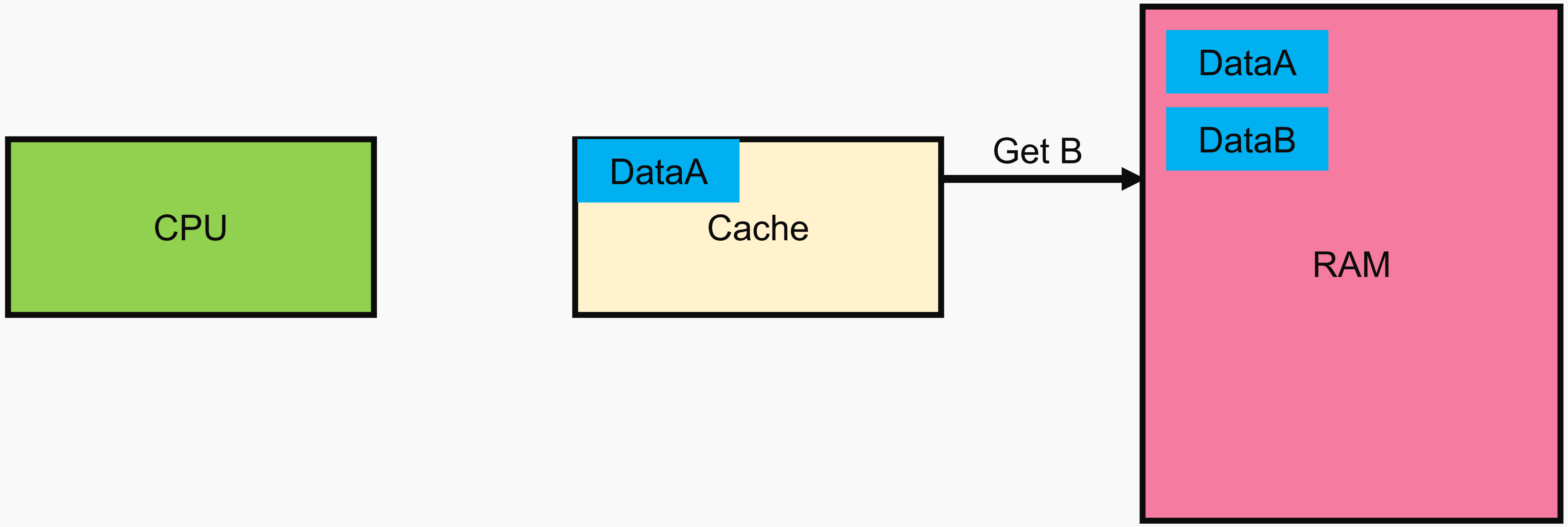
Utilizare



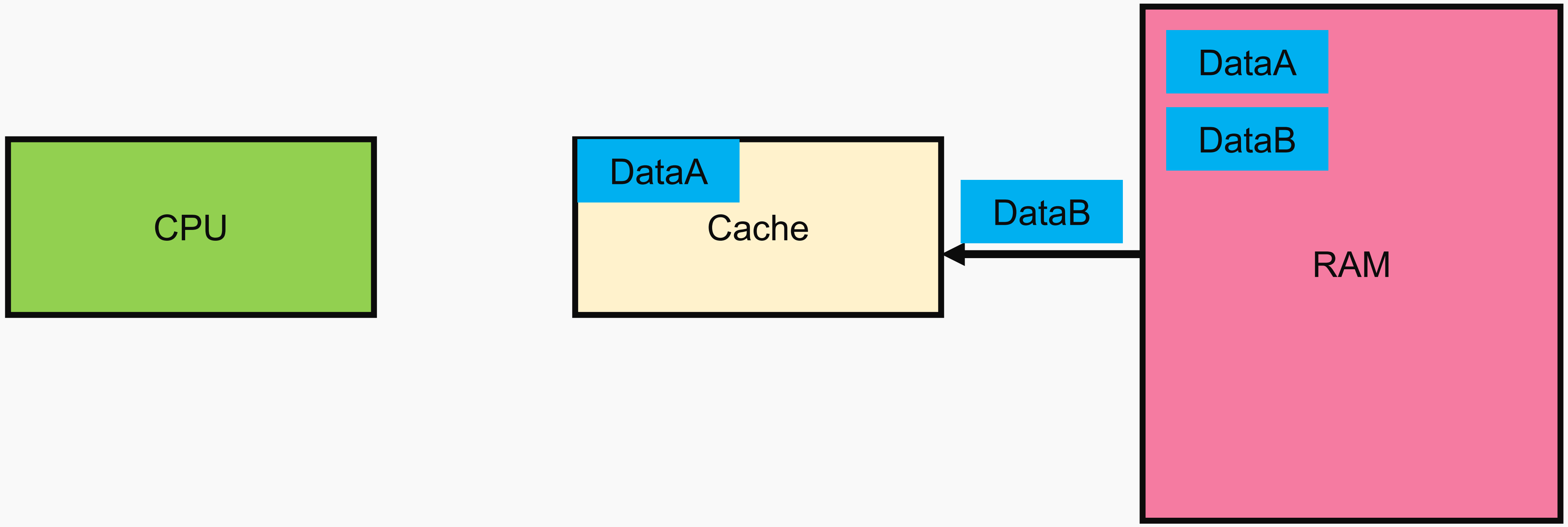
Utilizare – Cache Miss



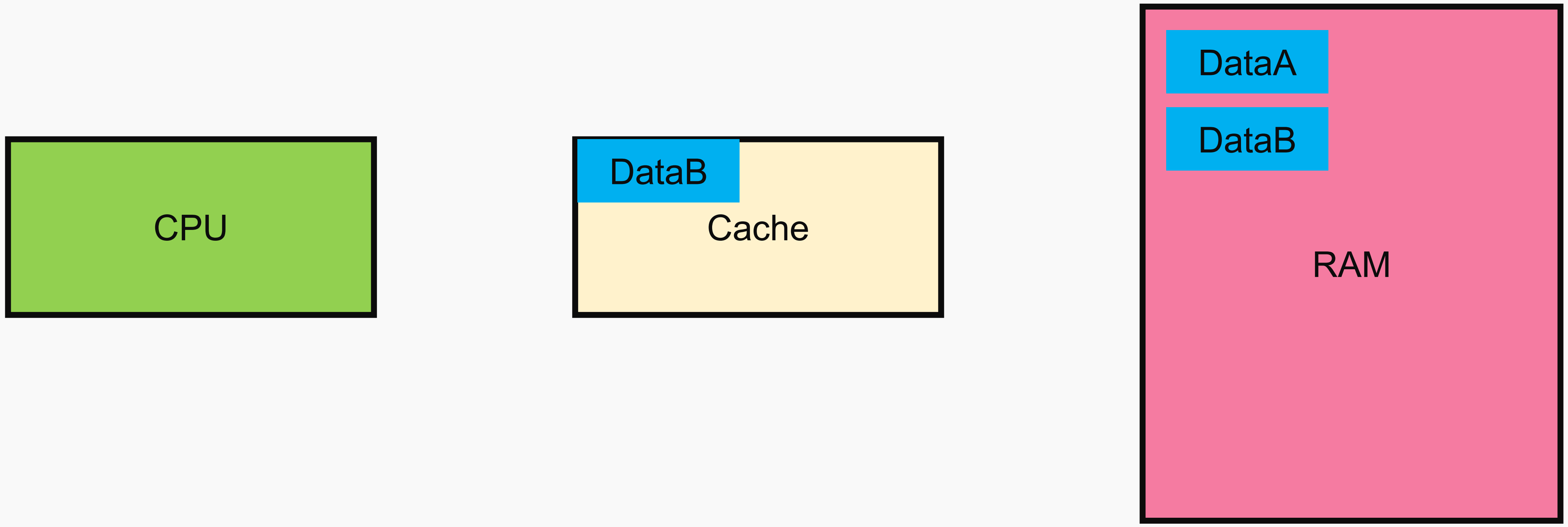
Utilizare – Cache Miss



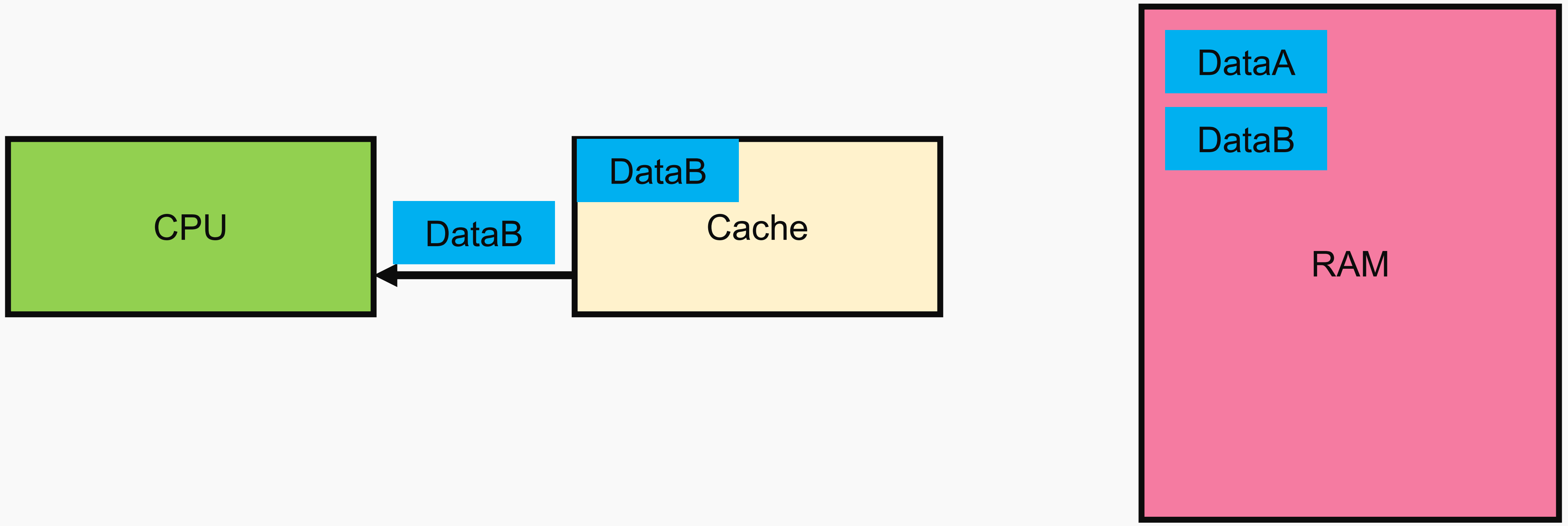
Utilizare – Cache Miss



Utilizare – Cache **Miss**



Utilizare – Cache Miss



Hit/Miss Ratio

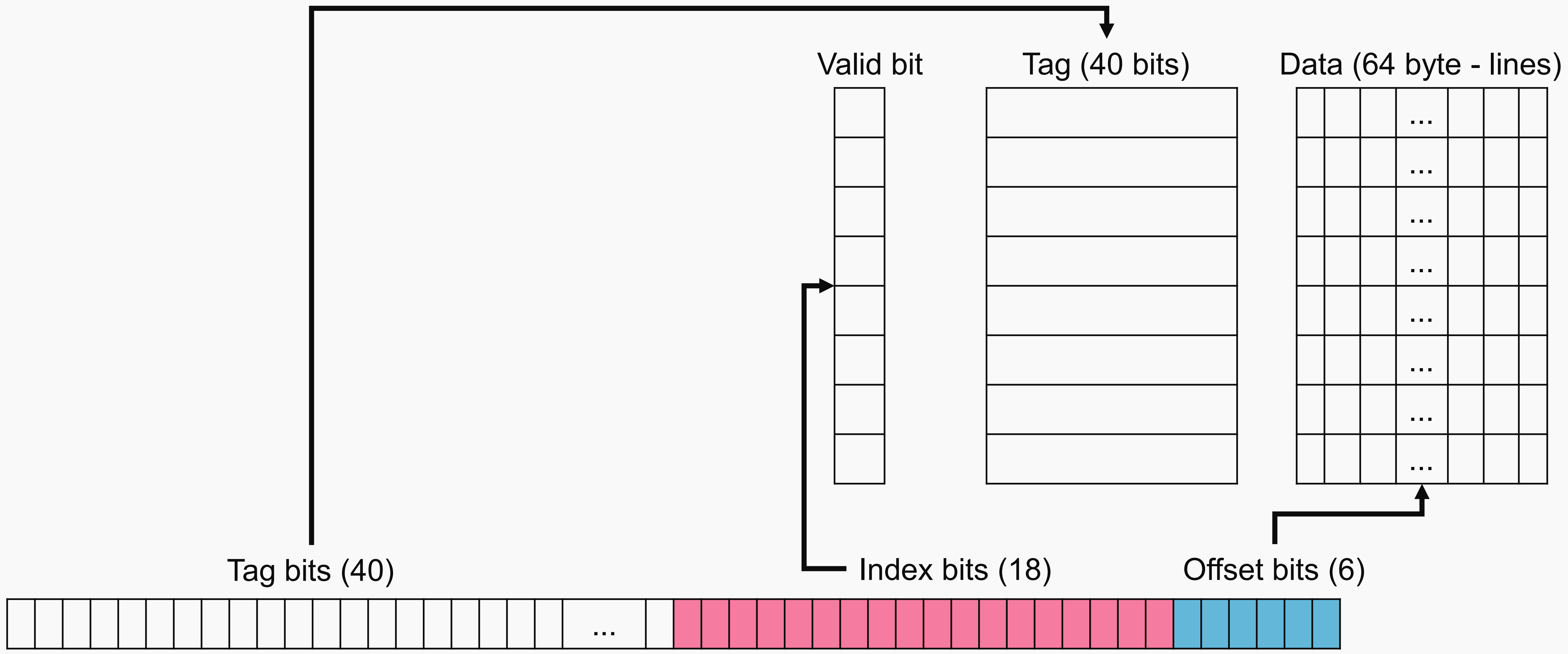
- Hit Ratio: $HR = \frac{hits}{hits+misses} = 1 - MR$

- Miss Ratio: $MR = \frac{misses}{hits+misses} = 1 - HR$

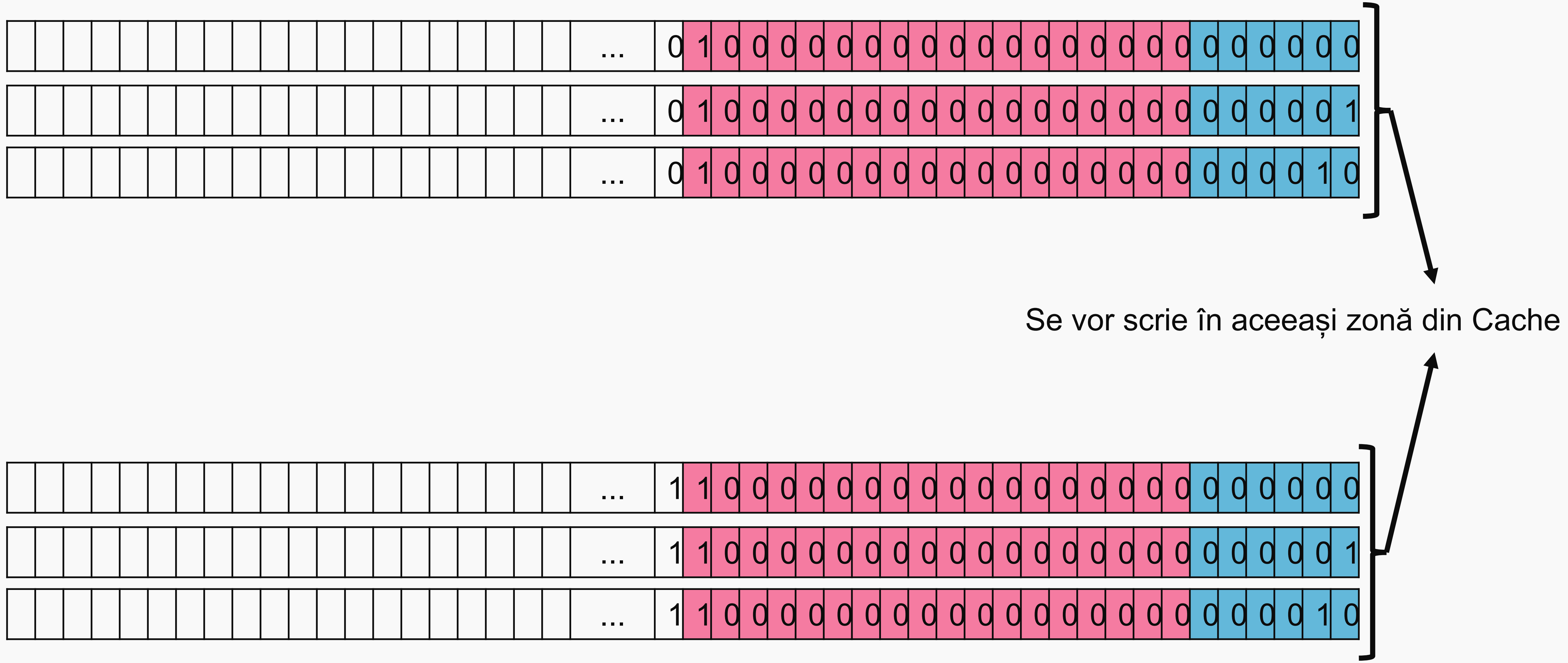
- Average Memory Access Time:

$$AMAT = HitTime + MissRatio * MissPenalty$$

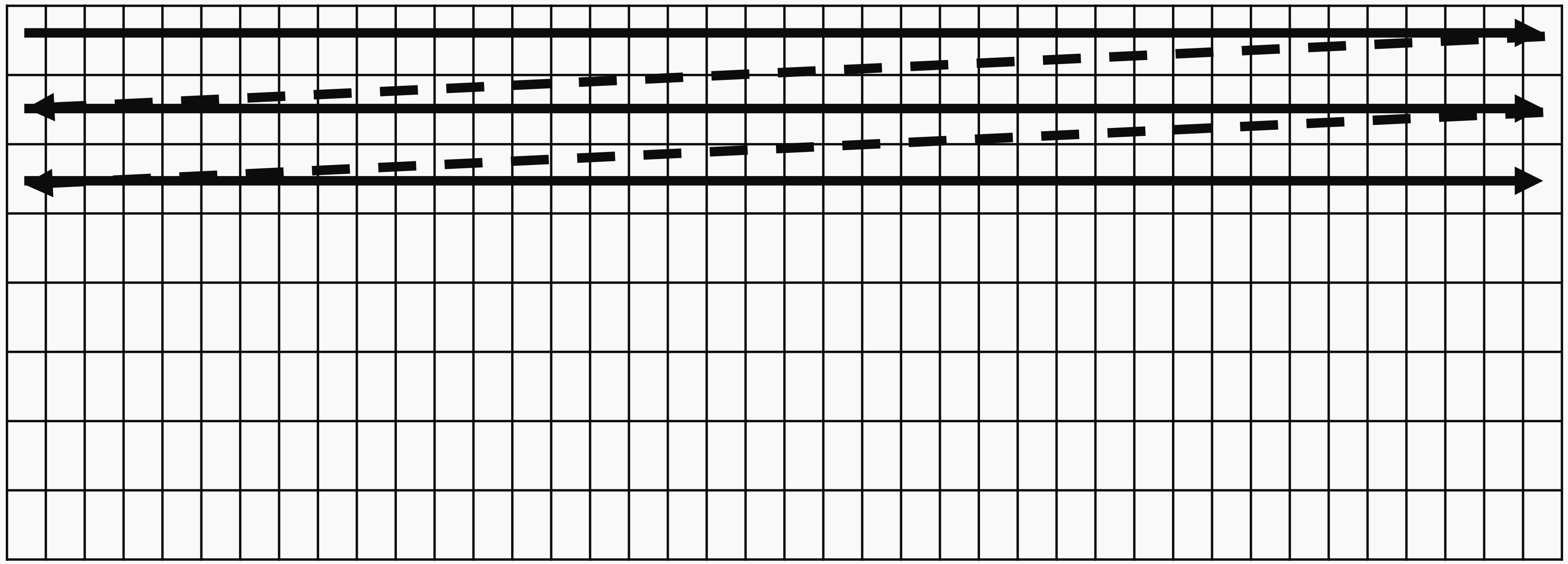
Direct-Mapped Cache



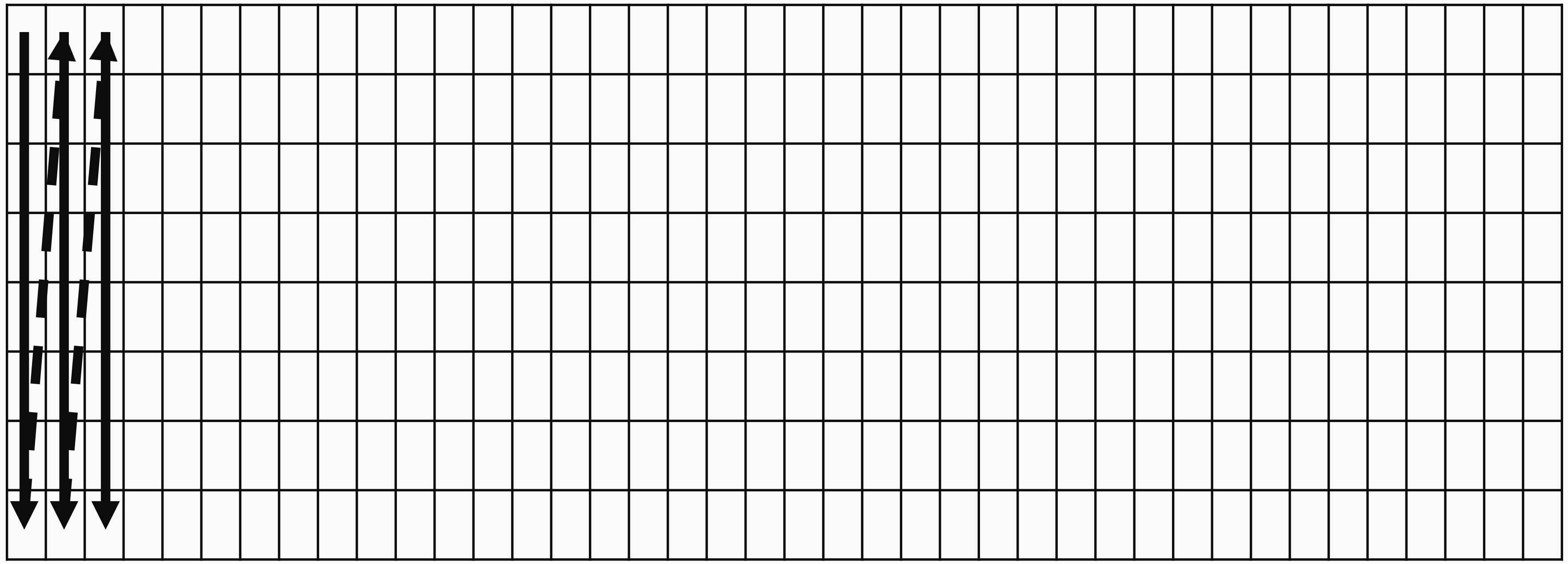
Problemă: Cache Thrashing



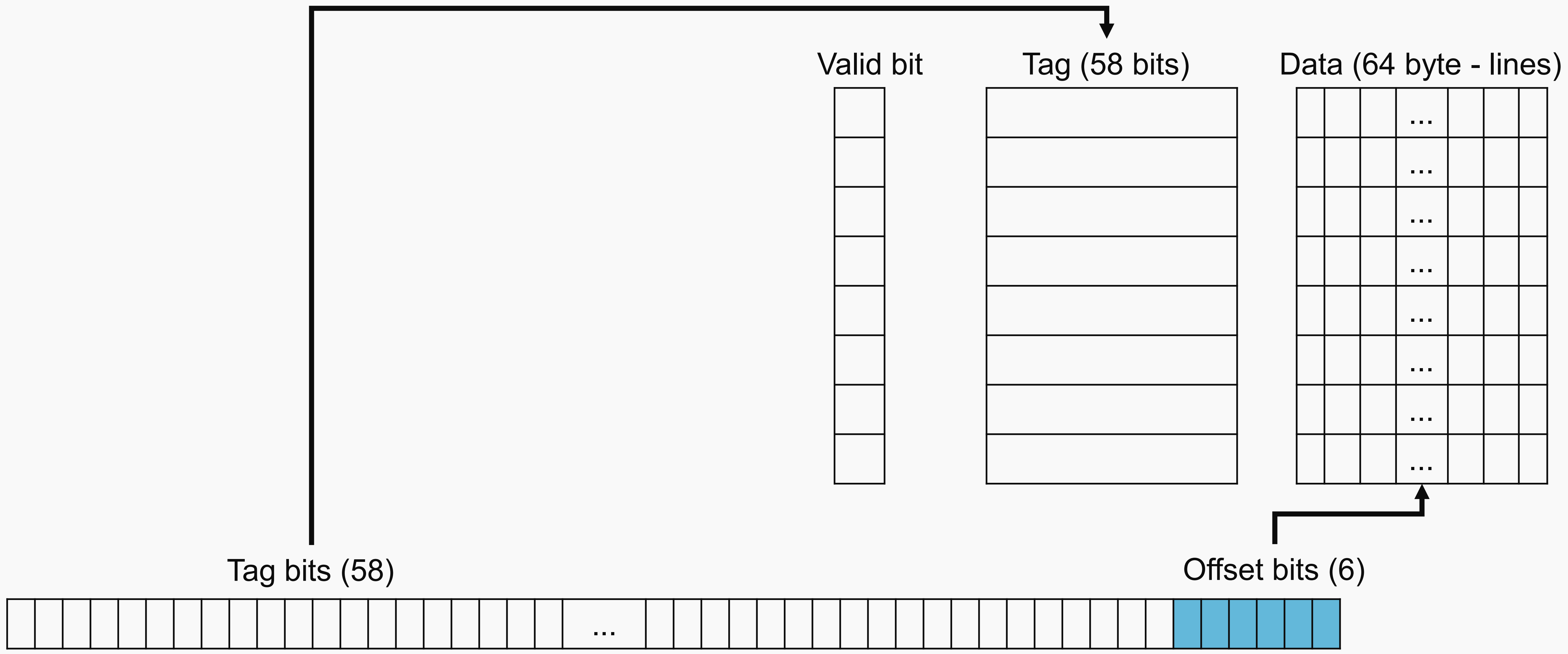
Matrici - Utilizare Cache Bună



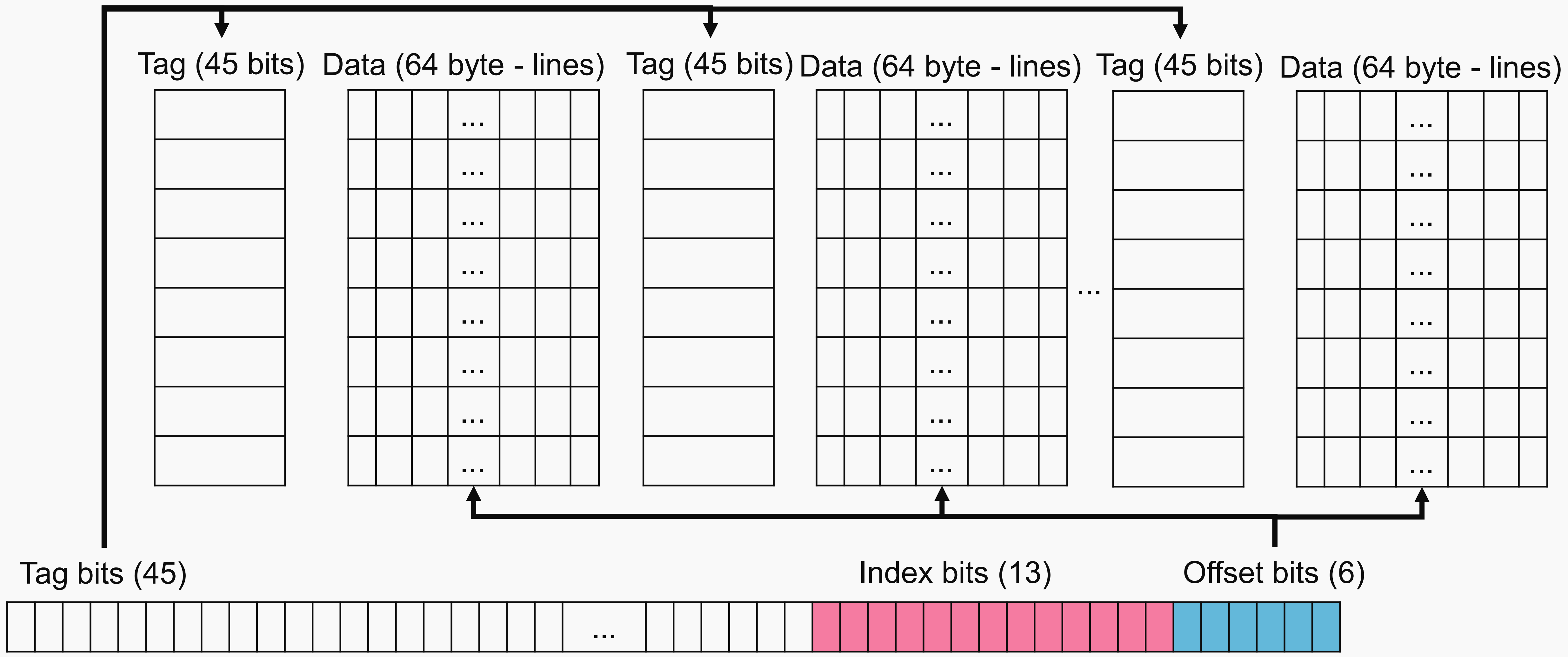
Matrici - Utilizare Cache Rea



Fully-Associative Cache

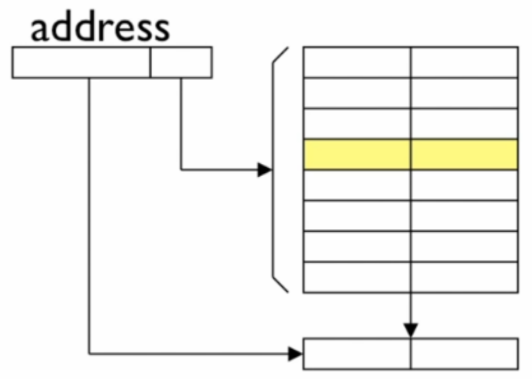


N (20) Way-Associative Cache



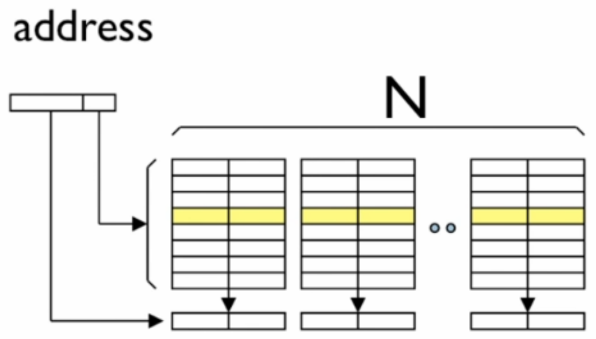
Associativity Implies Choices

Direct-mapped



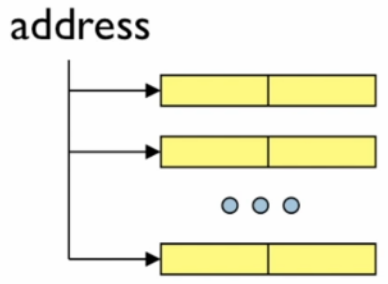
- Compare addr with only one tag
- Location A can be stored in exactly one cache line

N-way set-associative



- Compare addr with N tags simultaneously
- Location A can be stored in exactly one set, but in any of the N cache lines belonging to that set

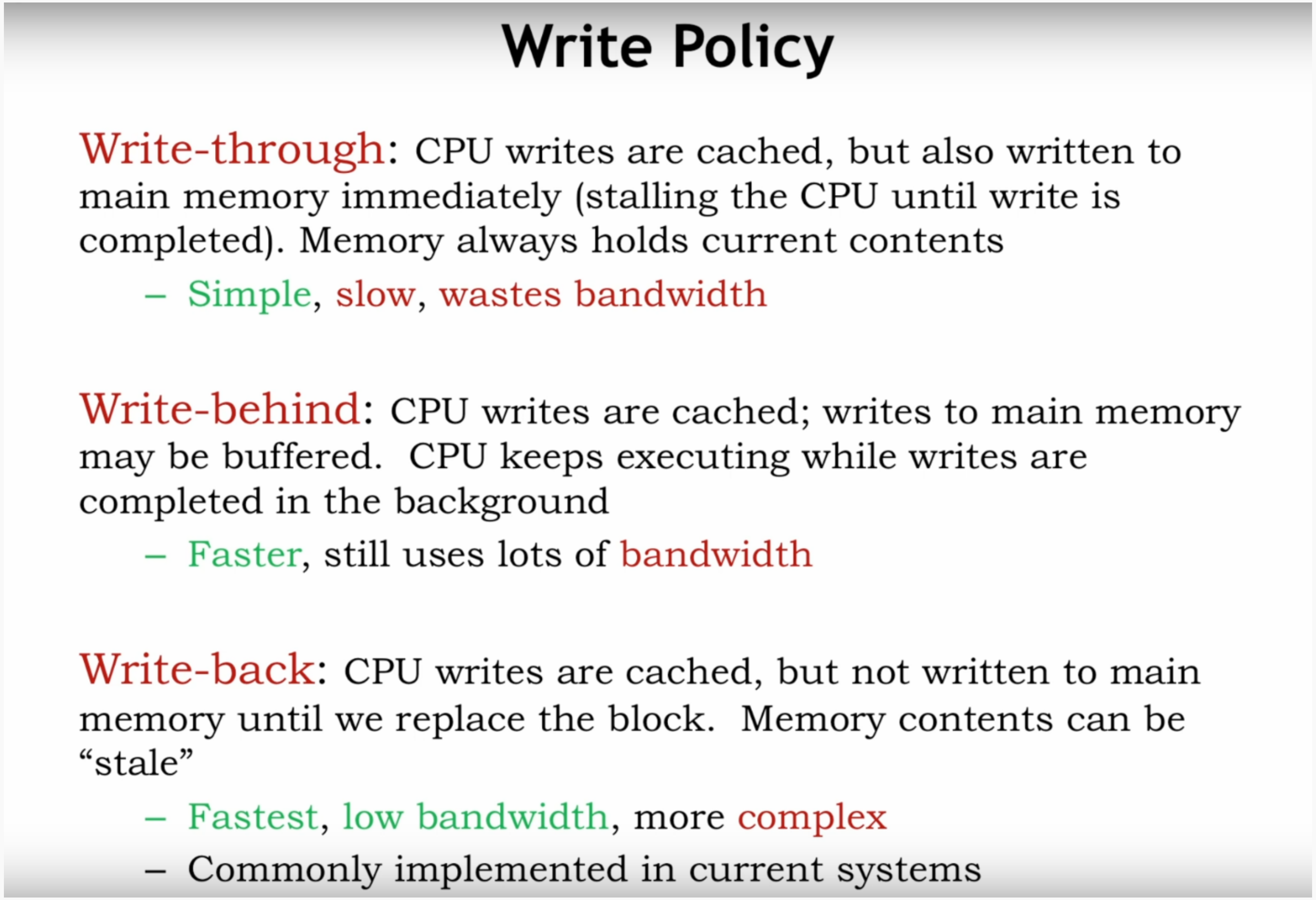
Fully associative



- Compare addr with each tag simultaneously
- Location A can be stored in any cache line

Cache replacement policies

- Random
- FIFO/LIFO
- Least Recently Used/Most recently used
- Least Frequently Used <<<<< LRU approximations
- Bélády (clairvoyant)



Write Policy

Write-through: CPU writes are cached, but also written to main memory immediately (stalling the CPU until write is completed). Memory always holds current contents

- Simple, slow, wastes bandwidth

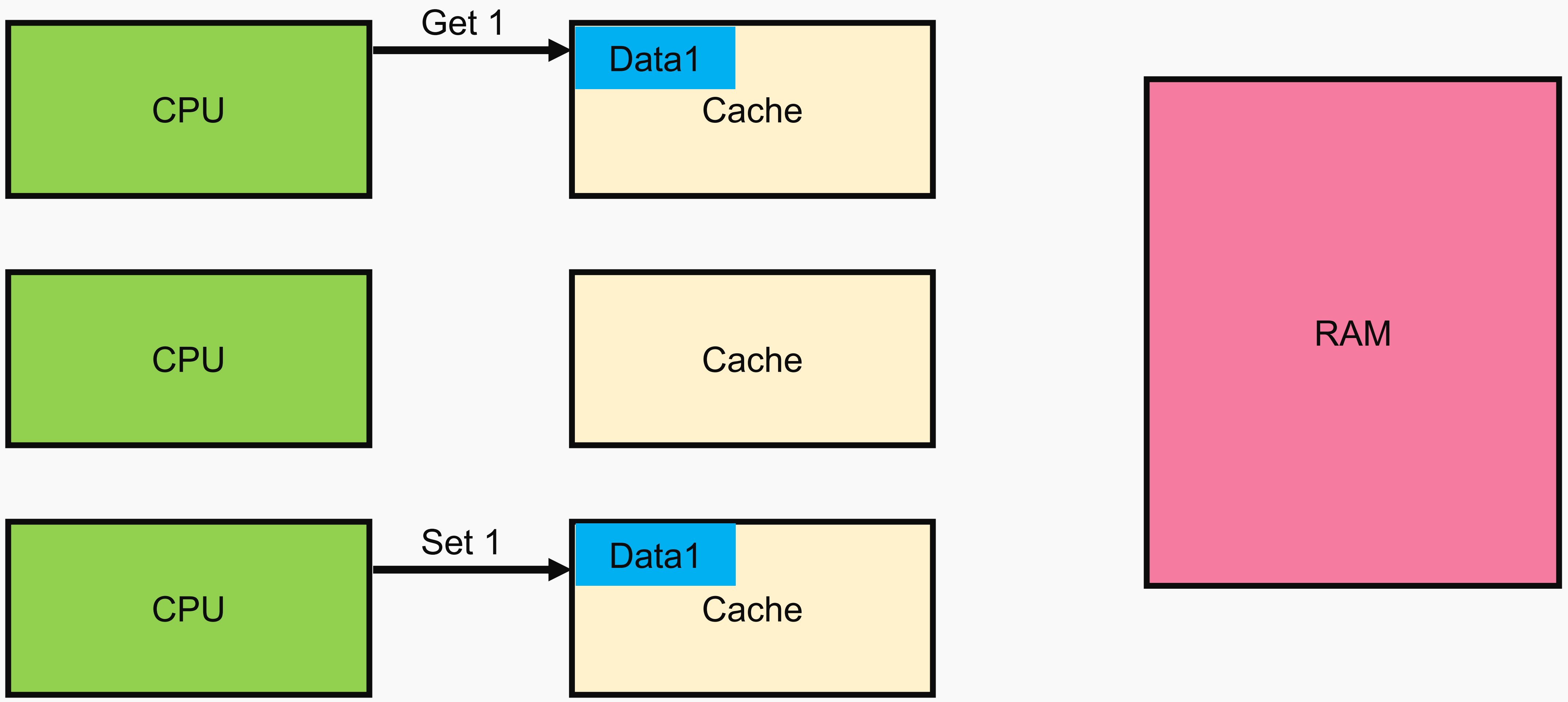
Write-behind: CPU writes are cached; writes to main memory may be buffered. CPU keeps executing while writes are completed in the background

- Faster, still uses lots of bandwidth

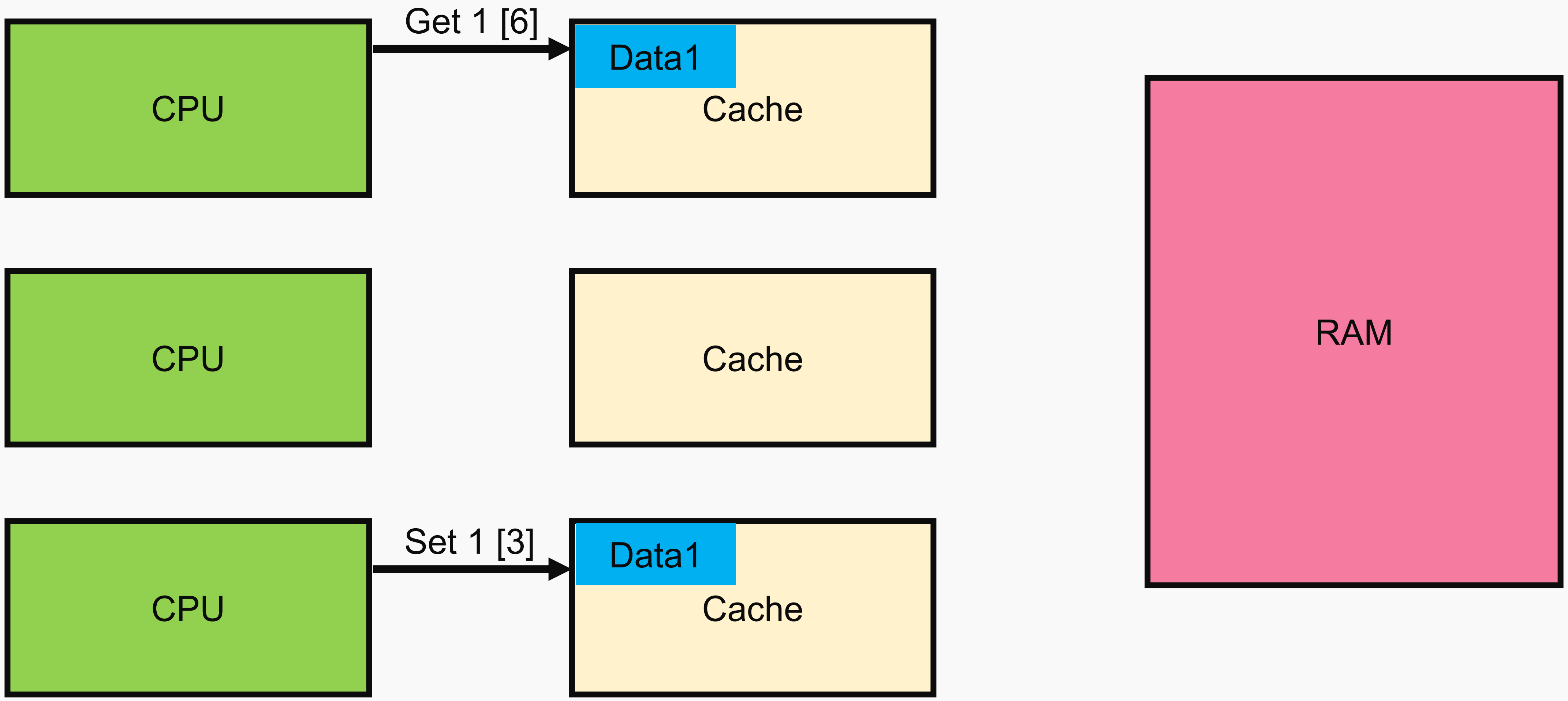
Write-back: CPU writes are cached, but not written to main memory until we replace the block. Memory contents can be “stale”

- Fastest, low bandwidth, more complex
- Commonly implemented in current systems

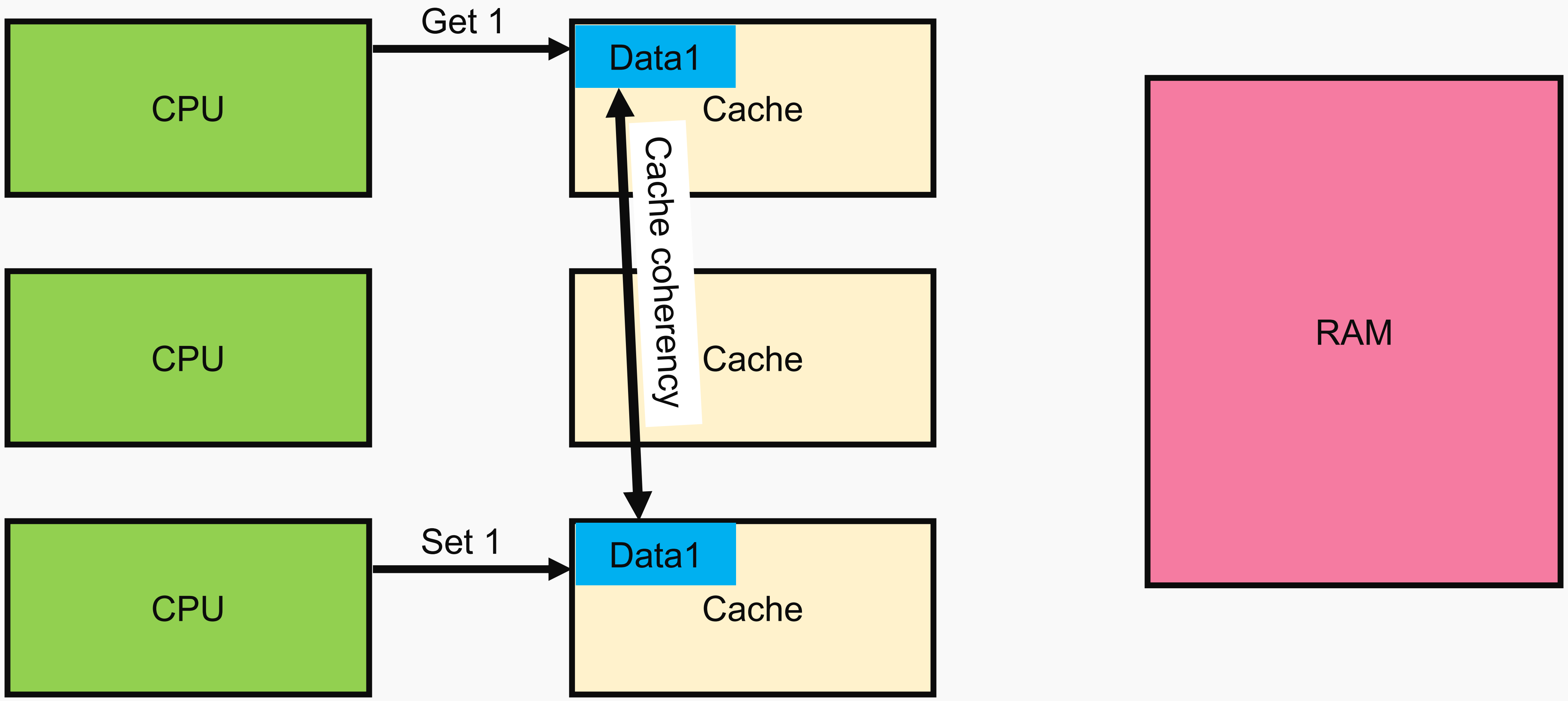
False sharing



False sharing – gets worse – cache lines



False sharing


CPU
Get 1
Data1
Cache
Cache coherency
CPU
Cache
CPU
Set 1
Data1
Cache
RAM