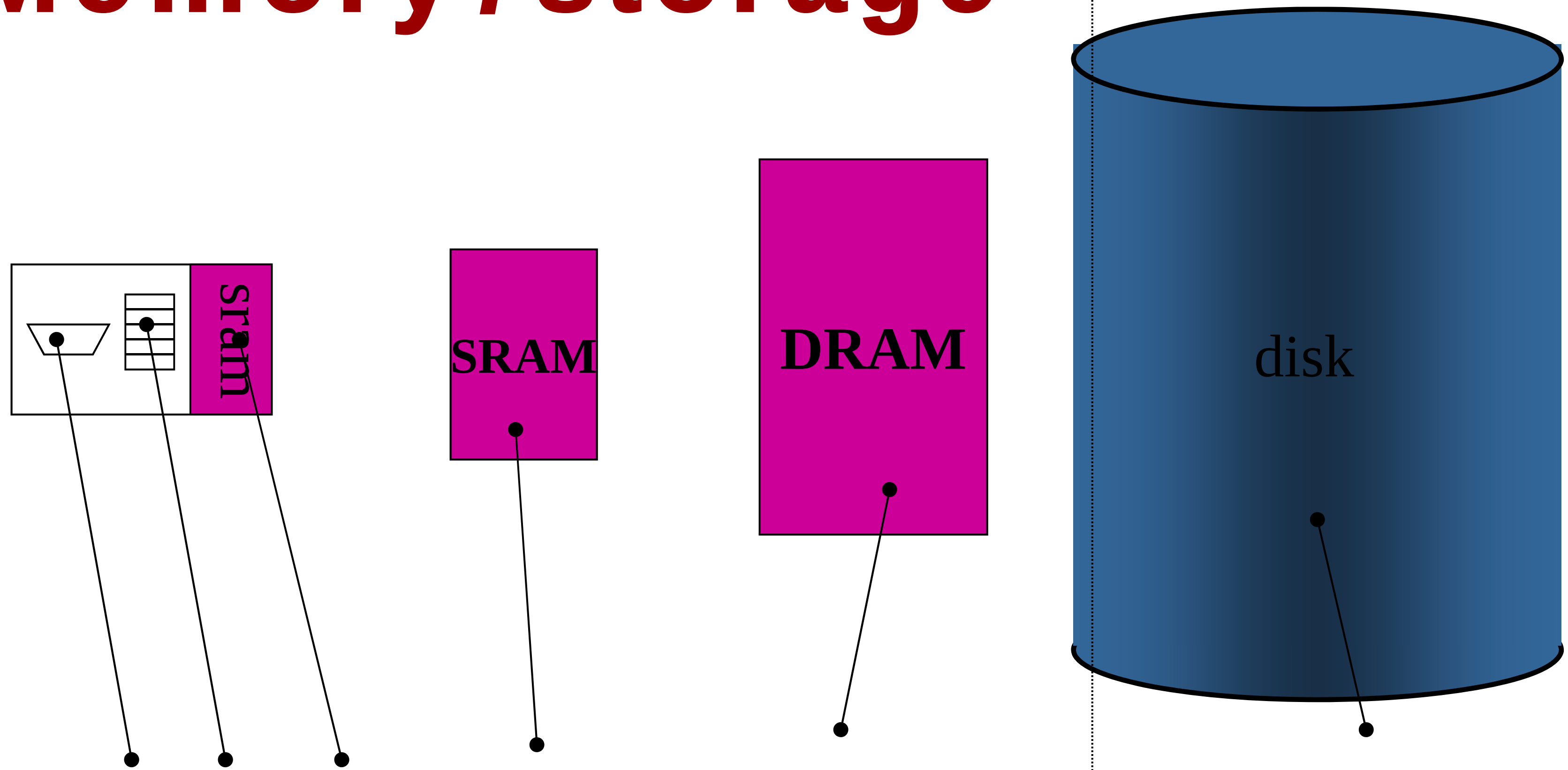
Caches and Memory System

Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

UPPSALA UNIVERSITET

Memory / storage

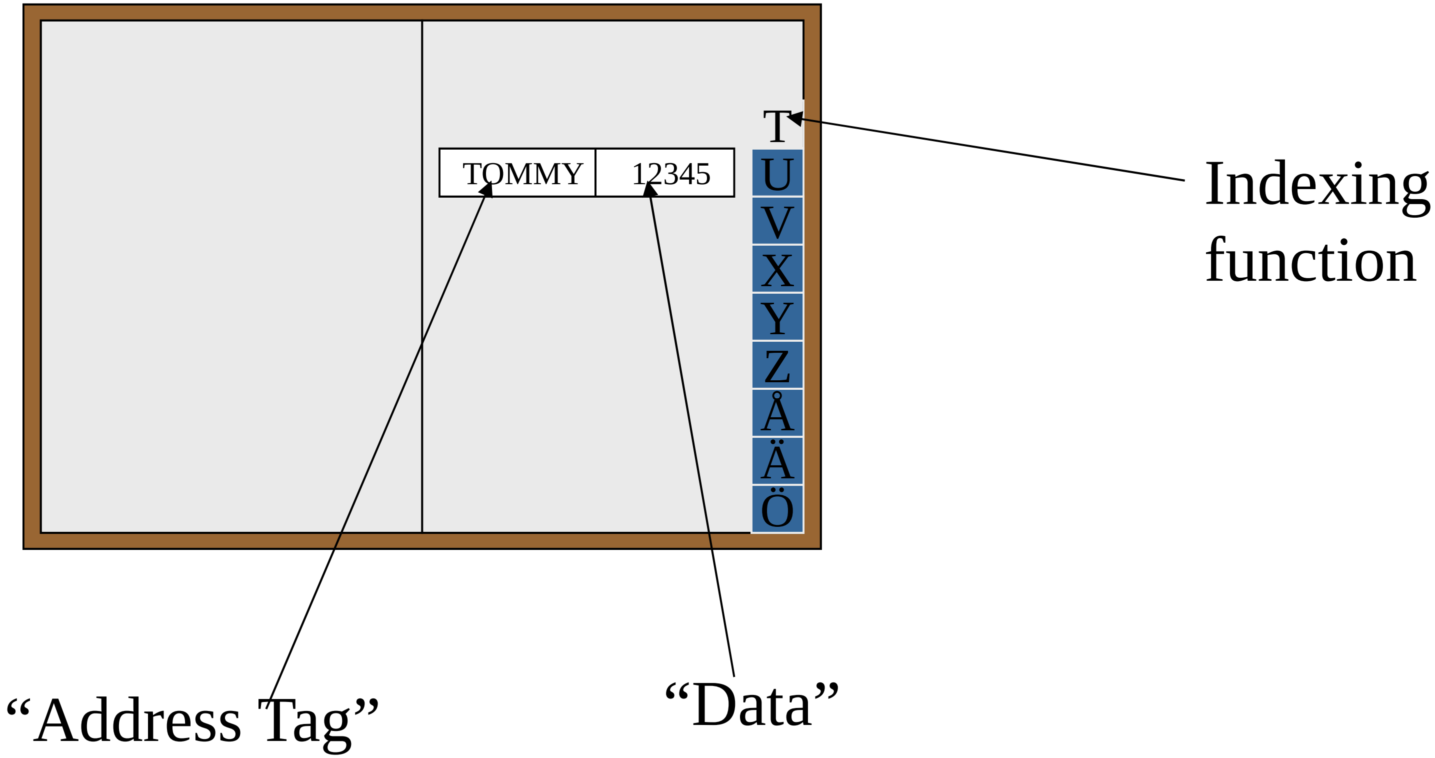


2000: 1ns 1ns 3ns 10ns 150ns 5 000 000ns

1kB 64k 4MB 1GB 1 TB

Address Book Cache

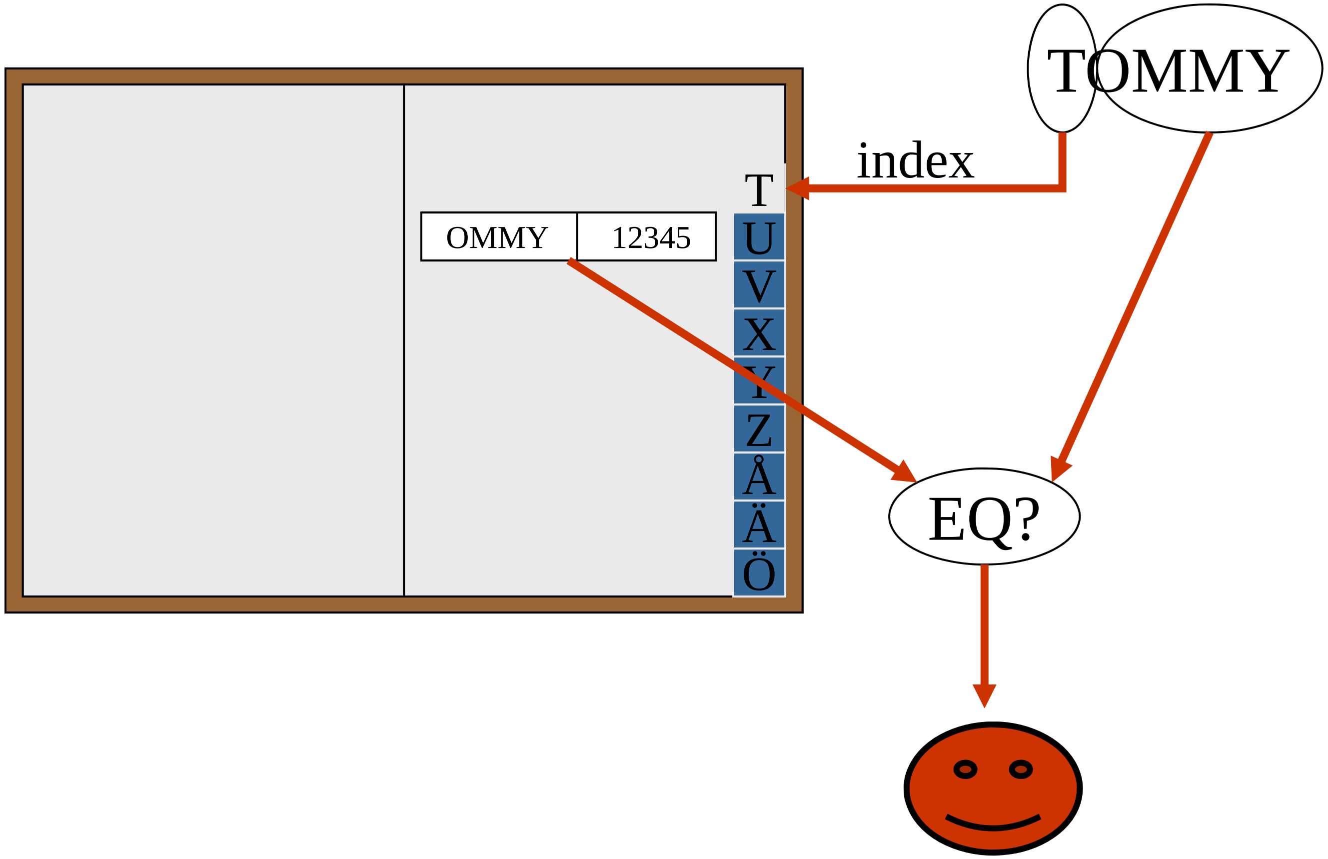
Looking for Tommy's Telephone Number



One entry per page =>
Direct-mapped caches with 28 entries

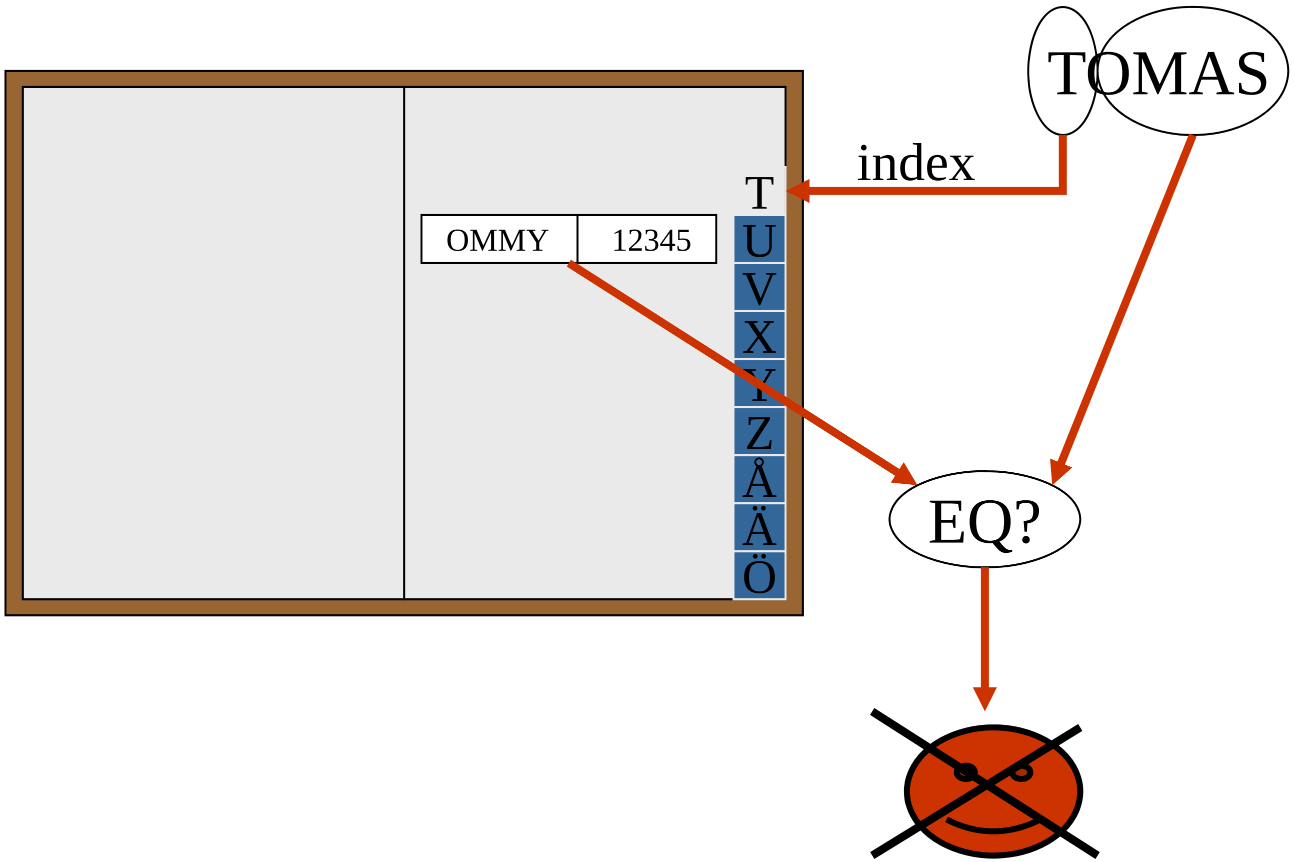
Address Book Cache

Looking for Tommy's Number



Address Book Cache

Looking for Tomas' Number



Miss!
Lookup Tomas' number in the telephone directory

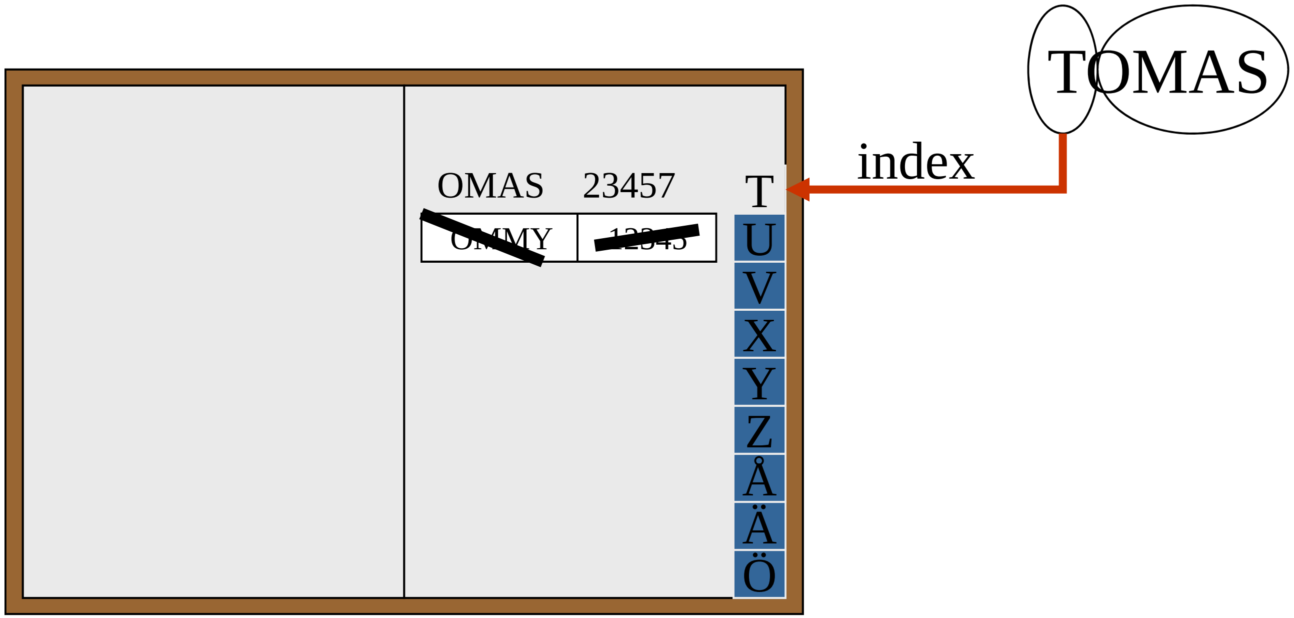
UPPSALA UNIVERSITET

PDC Summer School 2012

UPPSALA UNIVERSITET

Address Book Cache

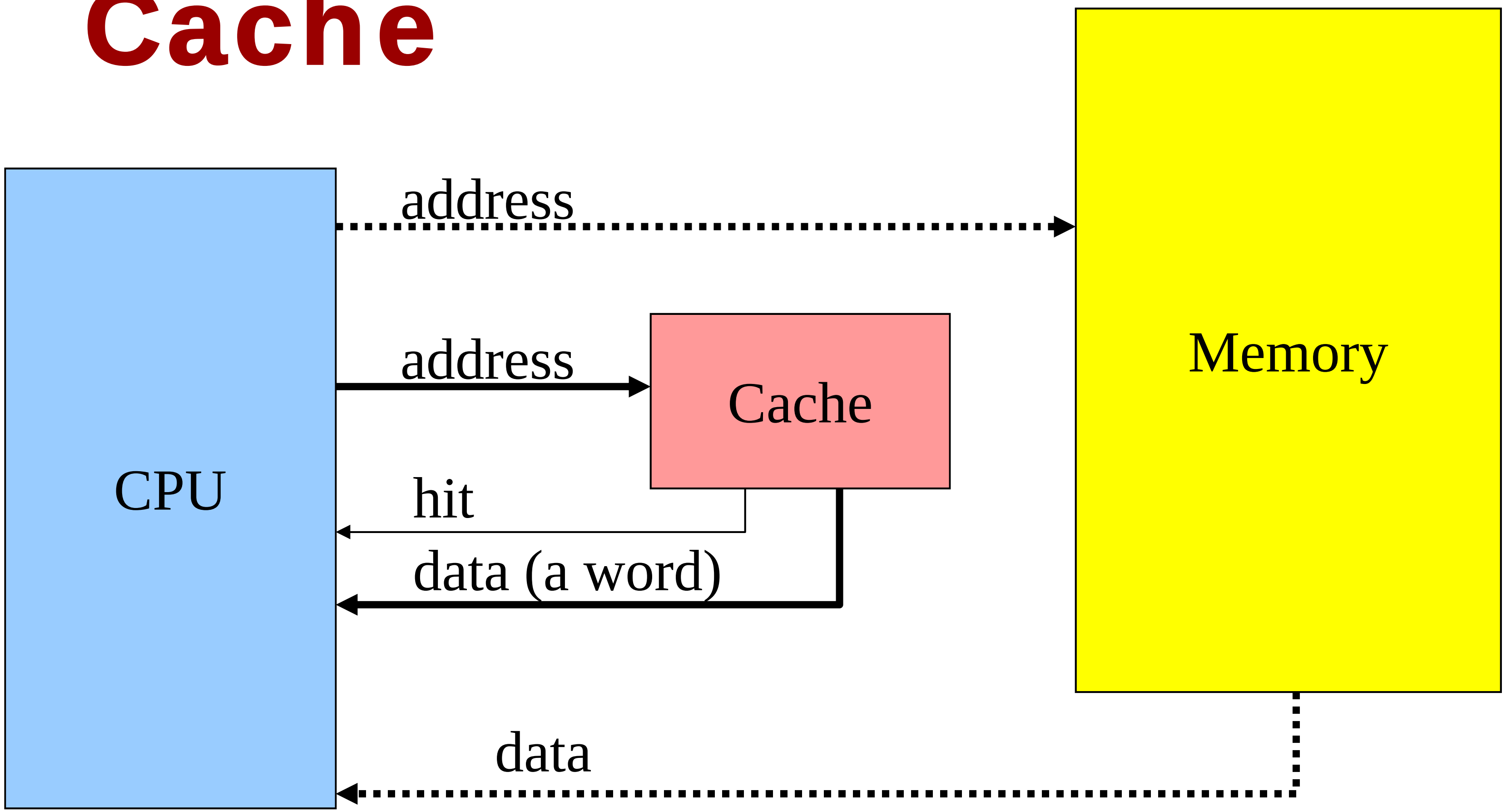
Looking for Tomas' Number



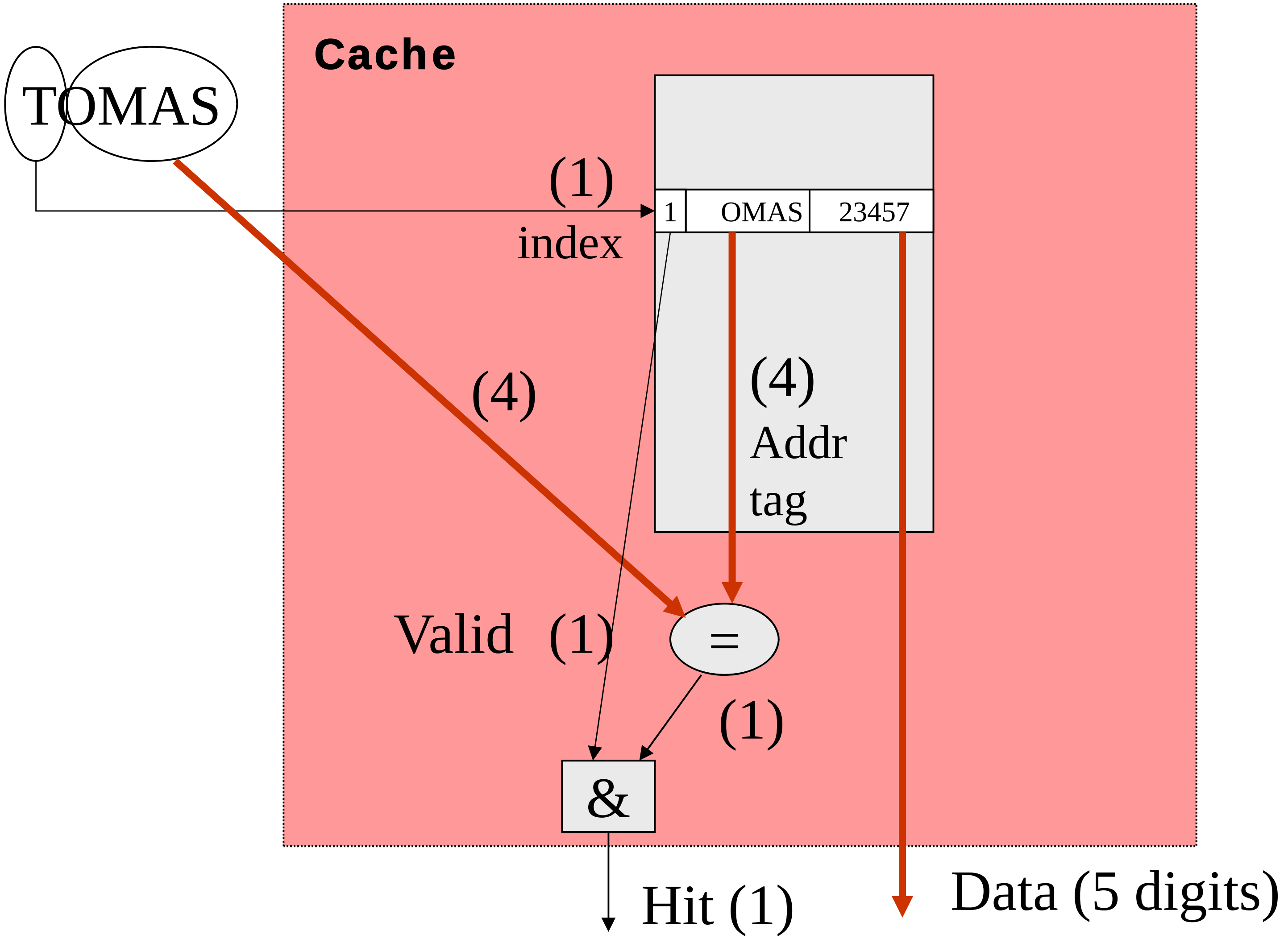
Replace TOMMY's data with TOMAS' data.
(Only one person per page = direct mapped cache)

PDC Summer School 2012

Cache

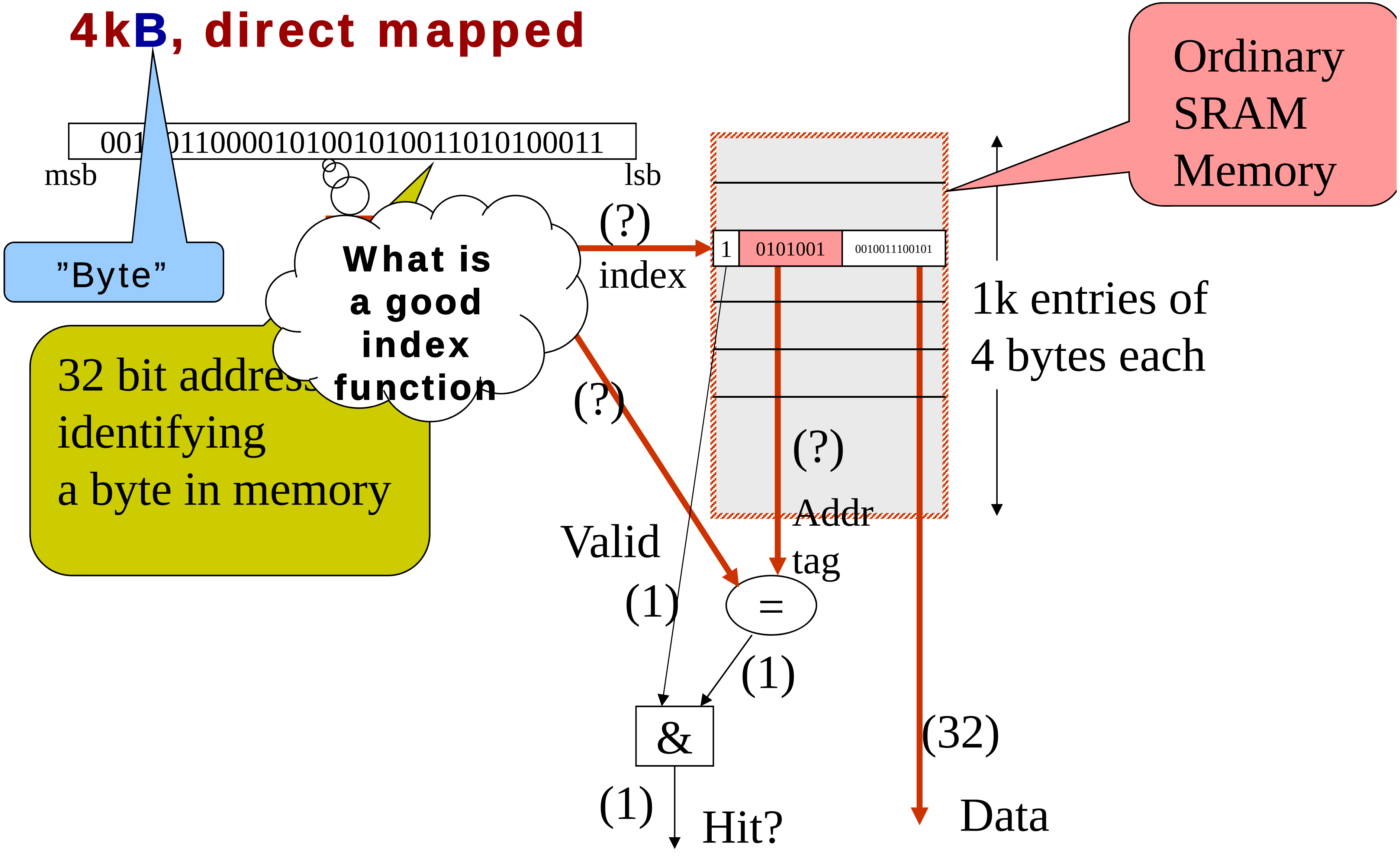


Cache Organization

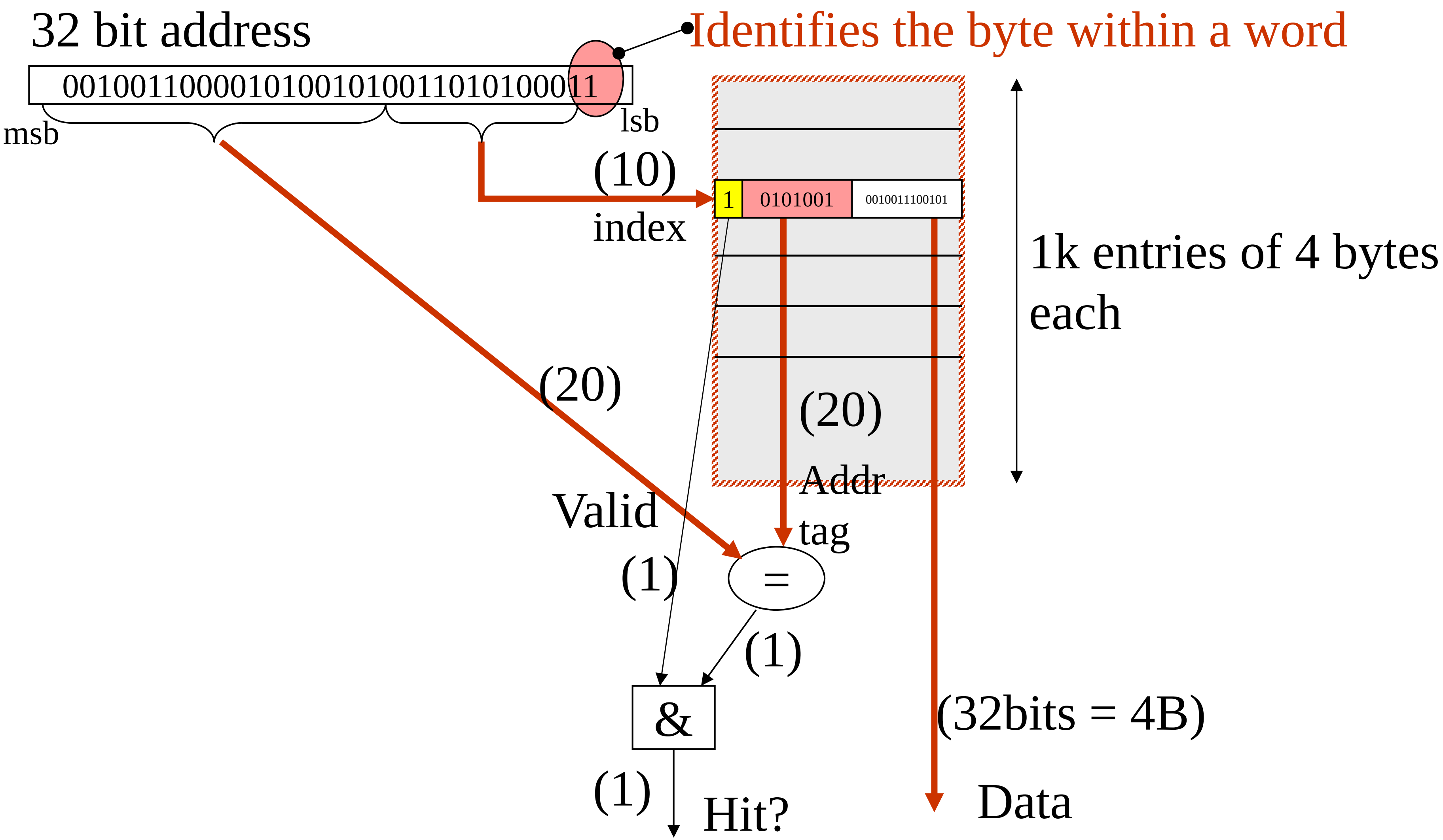


Cache Organization (really)

4kB, direct mapped



Cache Organization
4kB, direct mapped

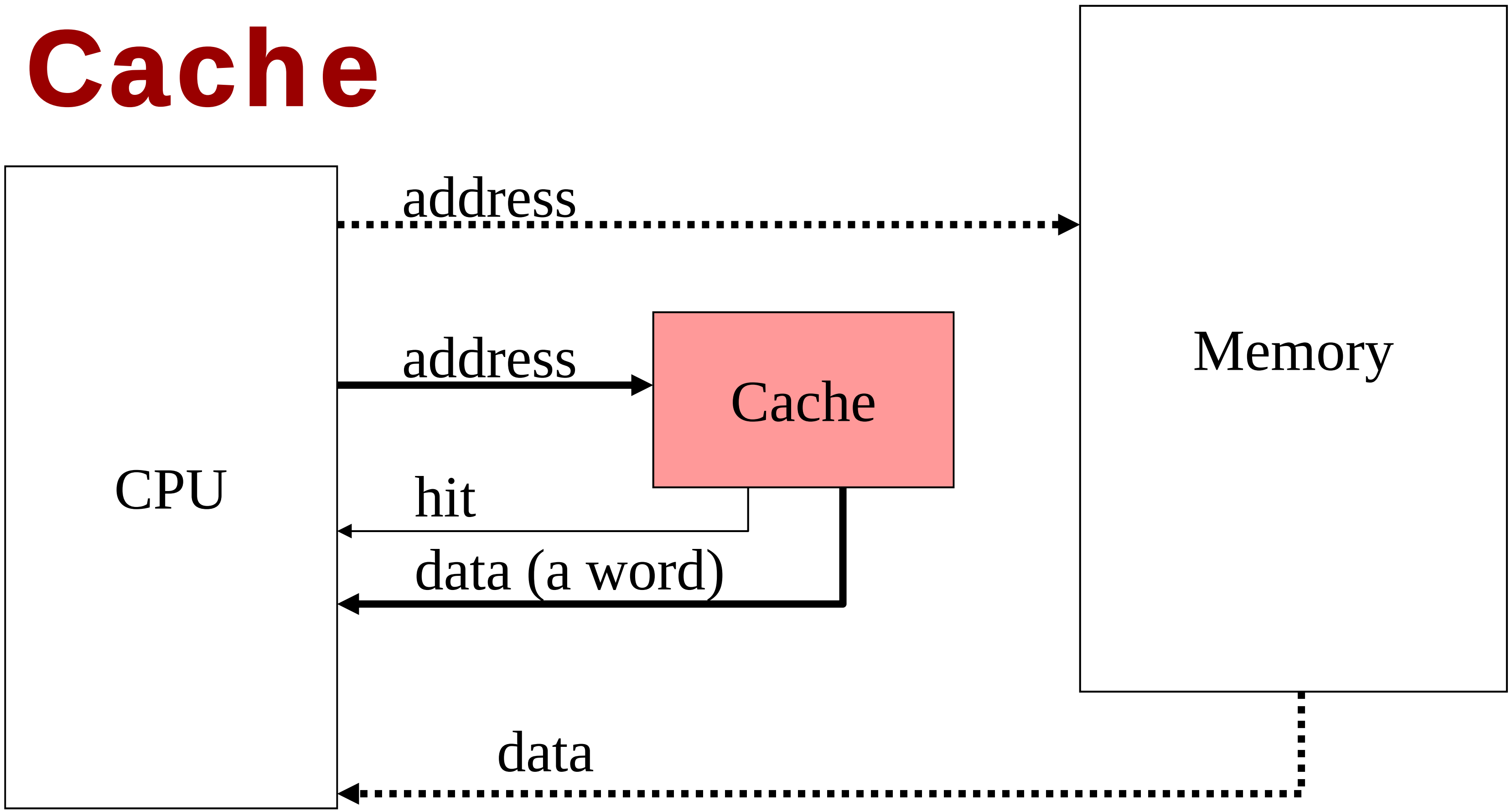


UPPSALA UNIVERSITET

PDC Summer School 2012

UPPSALA UNIVERSITET

Cache



Hit: Use the data provided by the cache

~Hit: Use data from memory and also store it in the cache

Why do you miss in a cache

- Mark Hill's three "Cs"
  - Compulsory miss (touching data for the first time)
  - Capacity miss (the cache is too small)
  - Conflict misses (non-optimal cache implementation)
- (Multiprocessors)
  - Communication (imposed by coherence)
  - False sharing (side-effect from large cache blocks)

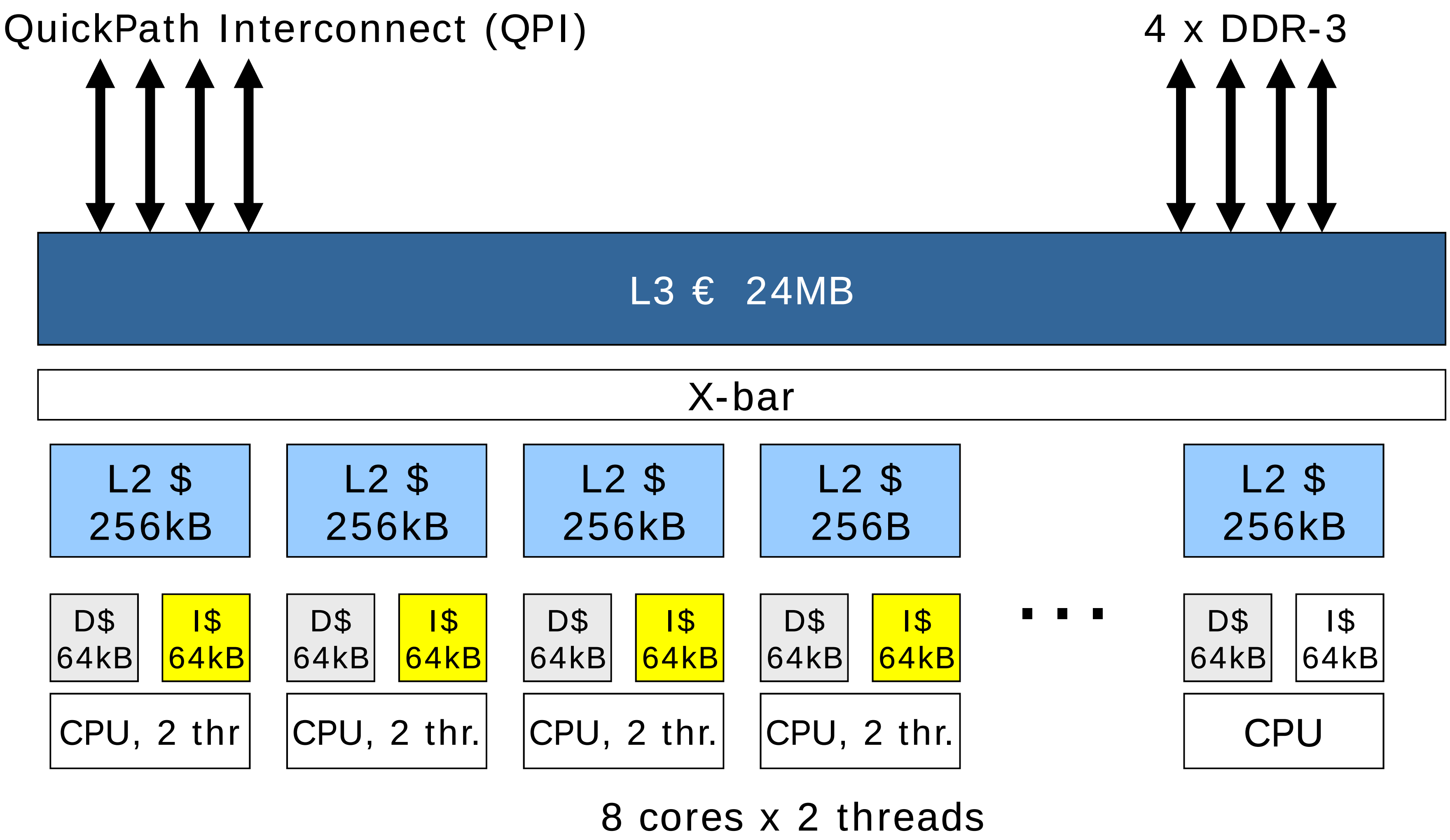
How to get more effective caches?

- Larger cache (more capacity)
- Cache block size (larger cache lines)
- More placement choice (more associativity)
- Innovative caches (victim, skewed, ...)
- Cache hierarchies (L1, L2, L3, CMR)
- Latency-hiding (weaker memory models)
- Latency-avoiding (prefetching)
- Cache avoiding (cache bypass)

Who to replace?
Picking a “victim”

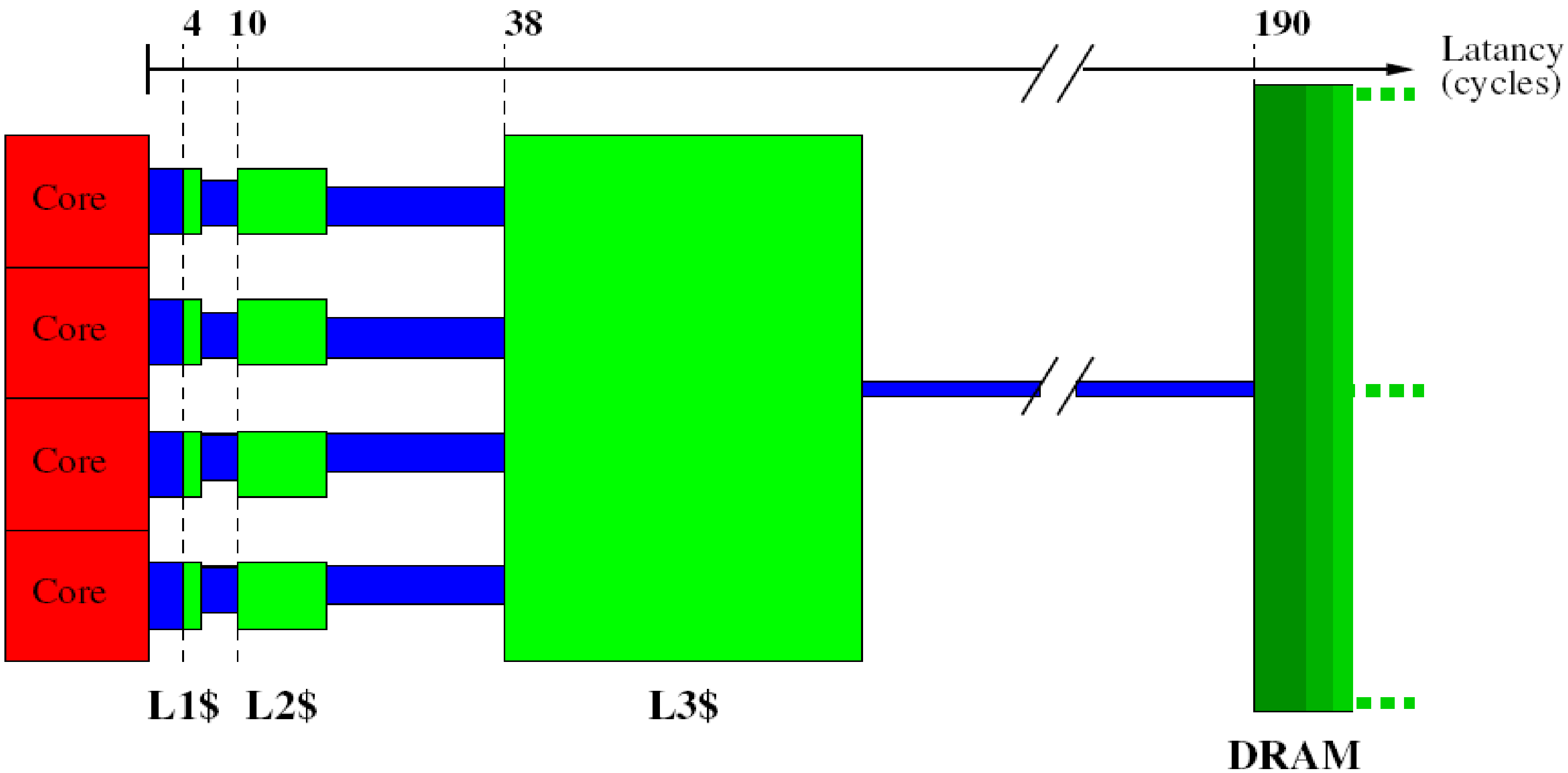
- Least-recently used (LRU)
  - Considered the best algorithm
  - Only practical up to 4-way (16 bits/CL)
- Not most recently used
  - Remember who used it last: 8-way -> 3 bits/CL
- Pseudo-LRU
  - Course Time stamps, used in the VM system
- Random replacement
  - Can’t continuously to have “bad luck...

Intel: ”Nehalem-Ex” (i7)



UPPSALA UNIVERSITET

Cache Capacity/Latency/BW



UPPSALA UNIVERSITET

Cache implementation

”8-way set-associative cache”

Cacheline, here 64B:

AT | S | Data = 64B

Caches at all level roughly work like this:

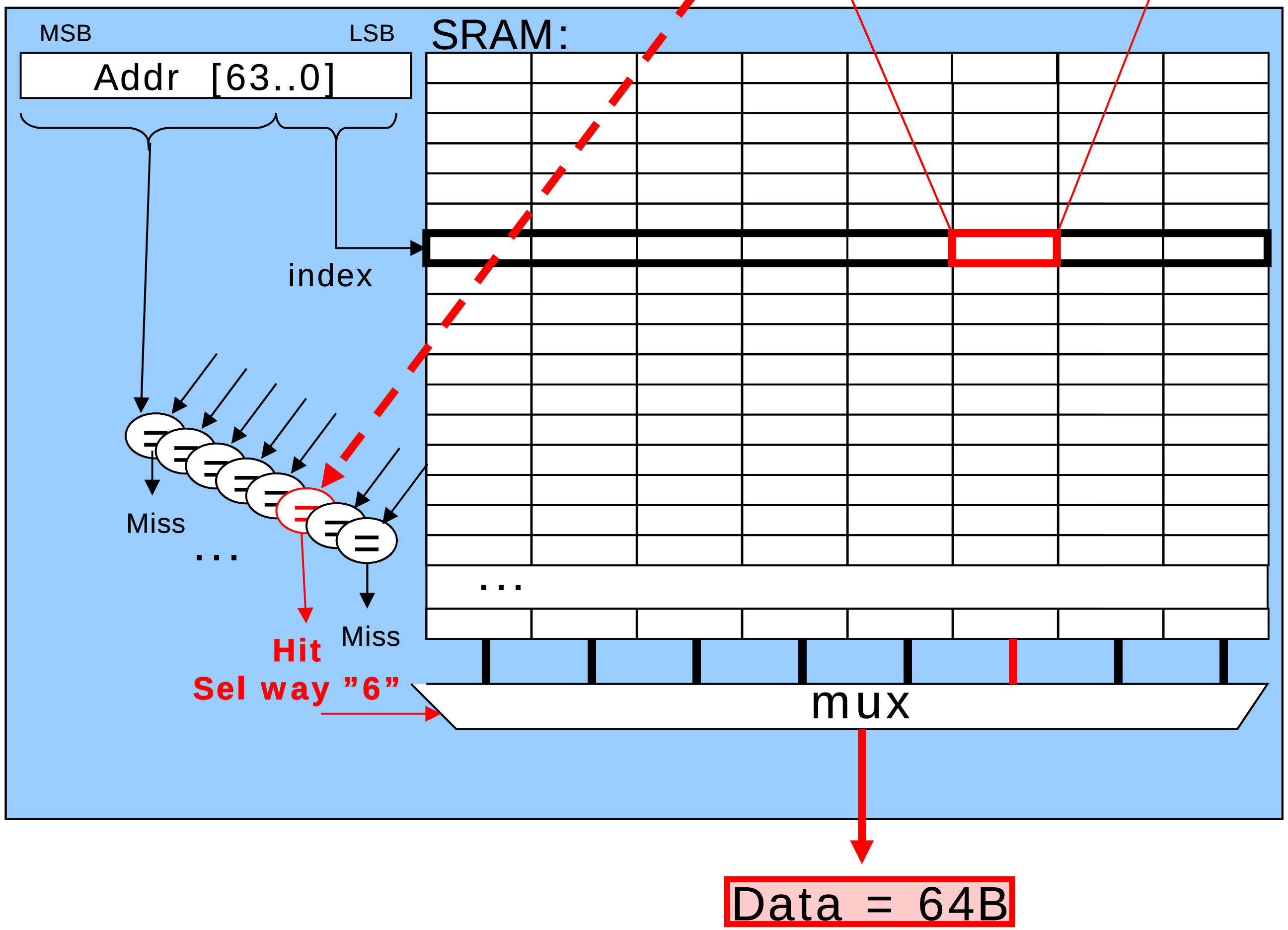
L3 € 24MB

L2 $ 256kB

D1 ¢ 64kB

I1 ¢ 64kB

Generic Cache:

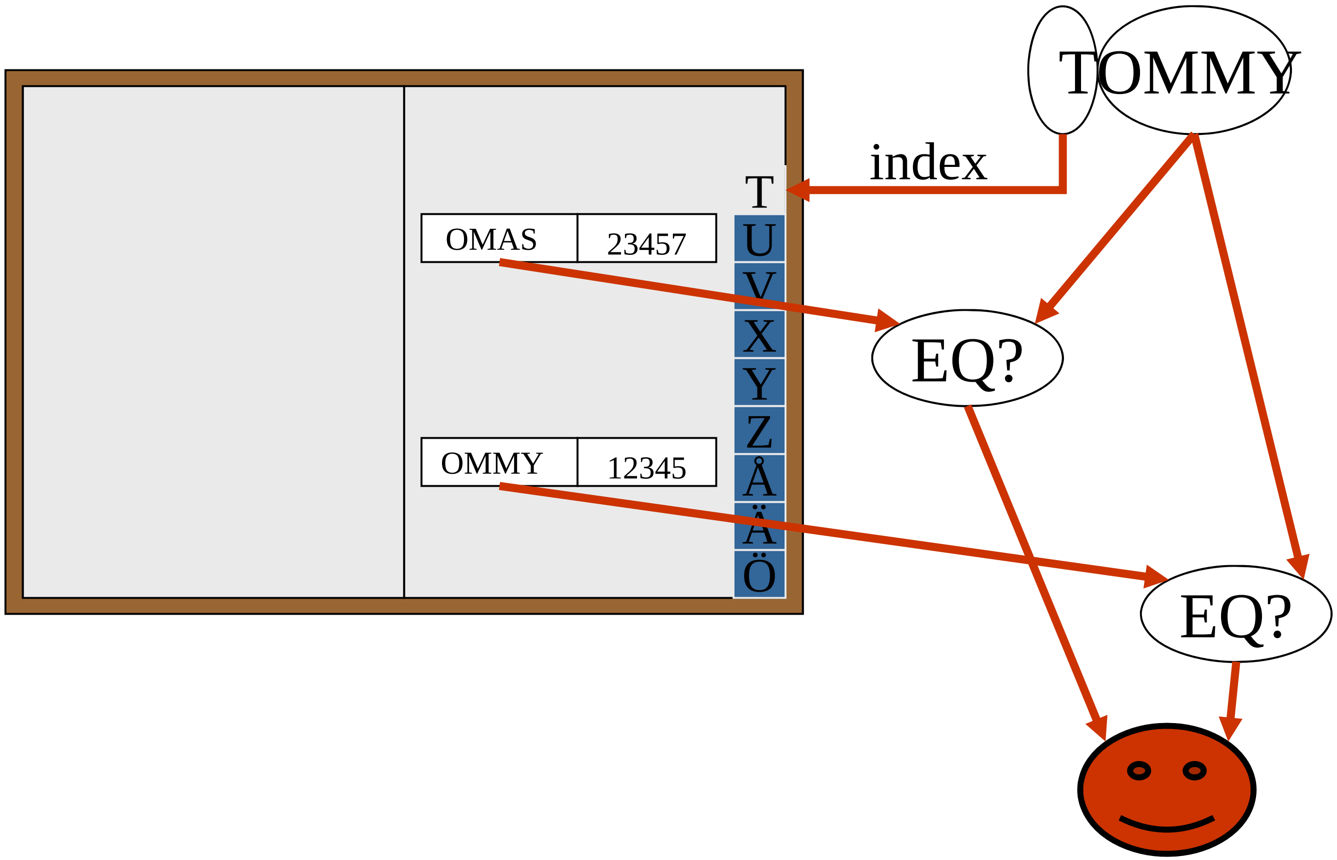


UPPSALA
UNIVERSITET

Address Book Analogy

"2-way set-associative cache"

Two names per page: index first, then search.



Select the second entry!

UPPSALA UNIVERSITET

Cache lingo

Cacheline: Data chunk move to/from a cache

Cache set: Fraction of the cache identified by the index

Associativity: Number of alternative storage places for a cacheline

Replacement policy: picking the victim to throw out from a set (LRU/Random/Nehalem)

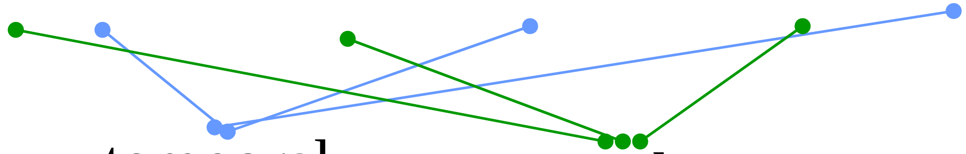
Temporal locality: Likelihood to access the same data again soon

Spatial locality: Likelihood to access nearby data again soon

Typical access pattern:

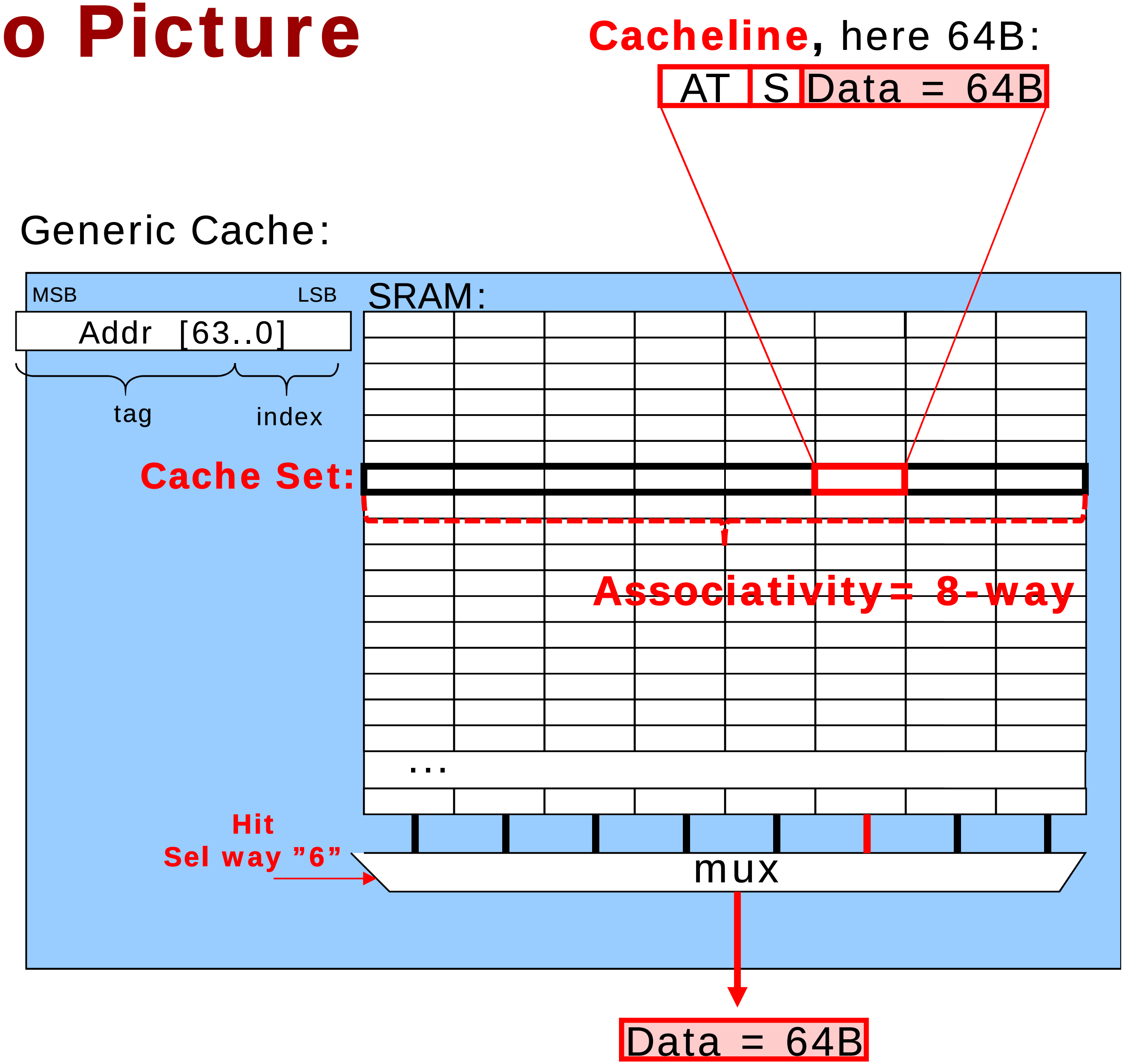
(inner loop stepping through an array)

A, B, C, A+4, B, C, A+8, B, C, ...



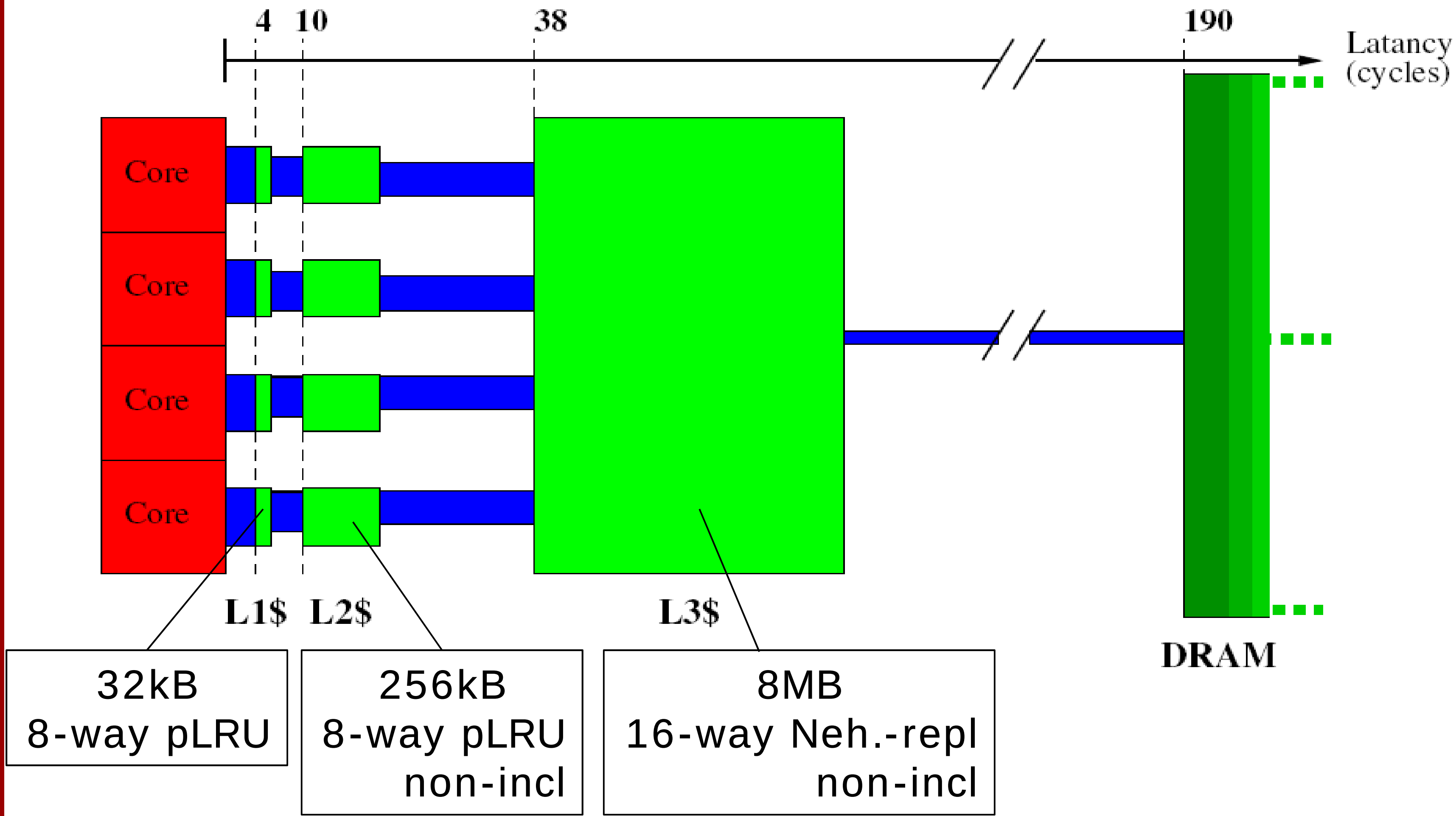
PDC Summer School 2012

Cache Lingo Picture



UPPSALA UNIVERSITET

Exempel Nehalem i7 (one example)



Take-away message: Caches

- Cache are fast but small
- Cache data travels in cache-line chunks (~64bytes)
- Least significant part of the address is used to find the ”set” (aka, indexing)
- There is a limited number of cache lines per set (associativity)
- Typically, several levels of caches

- Caches are most important target for optimizations

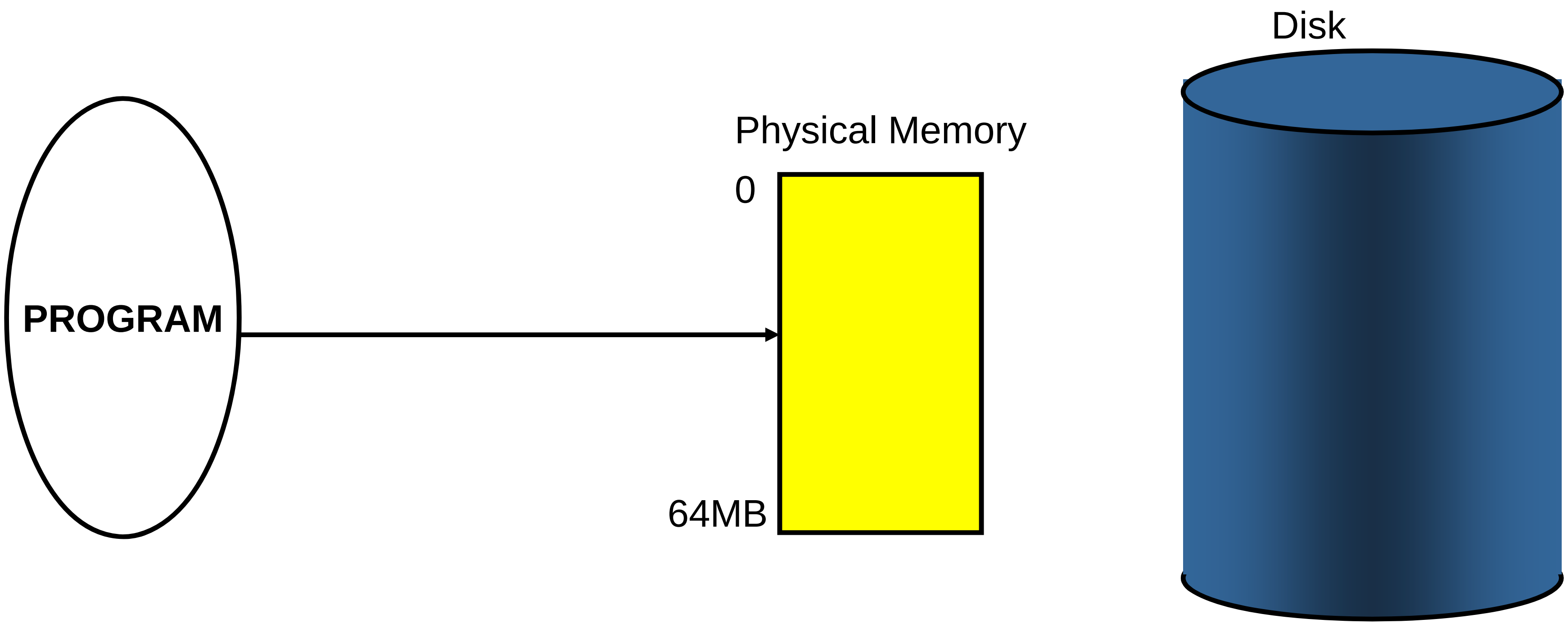
How are we doing?

- Create and explore locality:
  a) Spatial locality
  b) Temporal locality
  c) Geographical locality
- Create and explore parallelism
  a) Instruction level parallelism (ILP)
  b) Thread level parallelism (TLP)
  c) Memory level parallelism (MLP)
- Speculative execution
  a) Out-of-order execution
  b) Branch prediction
  c) Prefetching

Virtual Memory System

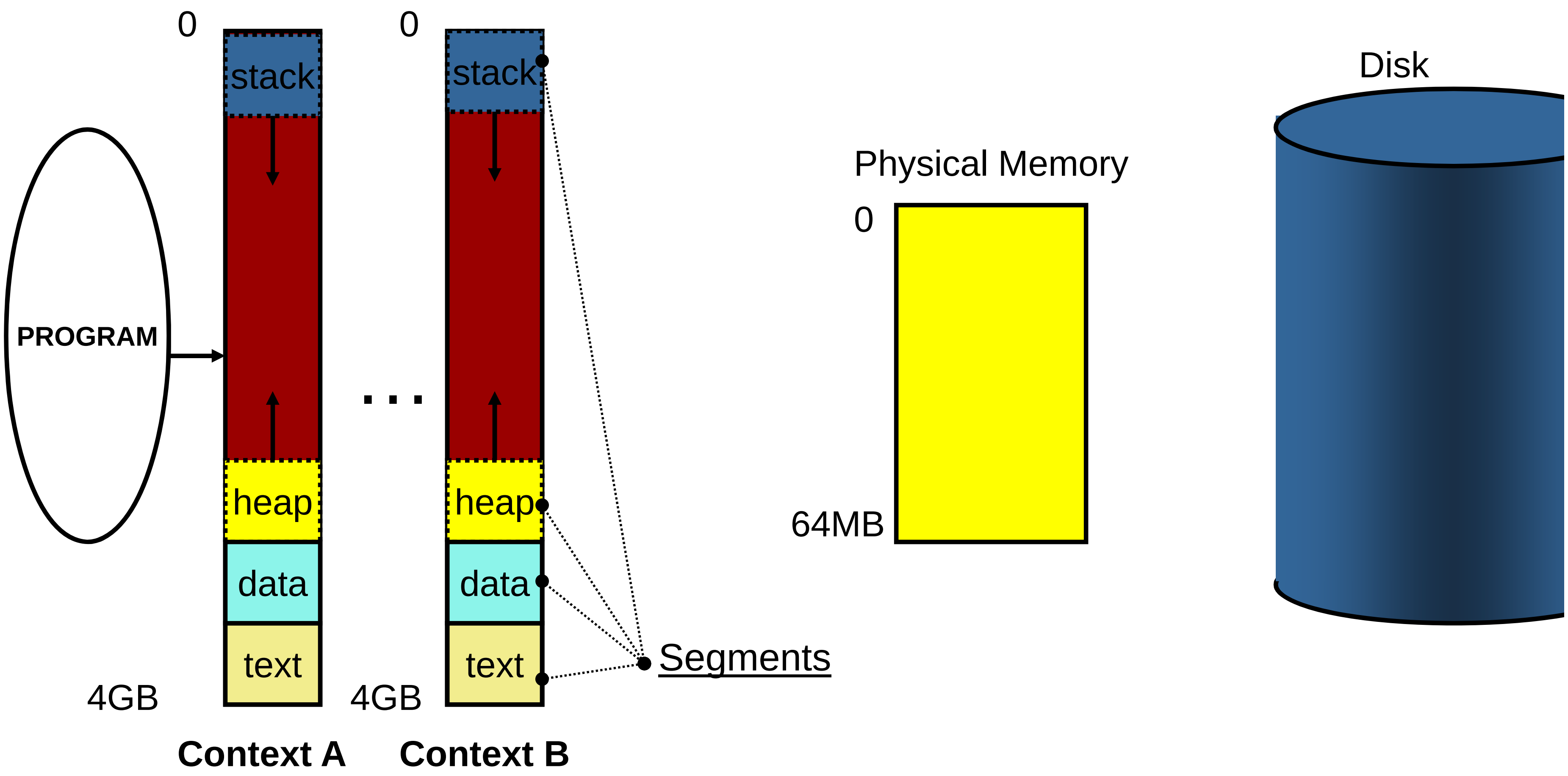
Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

Physical Memory

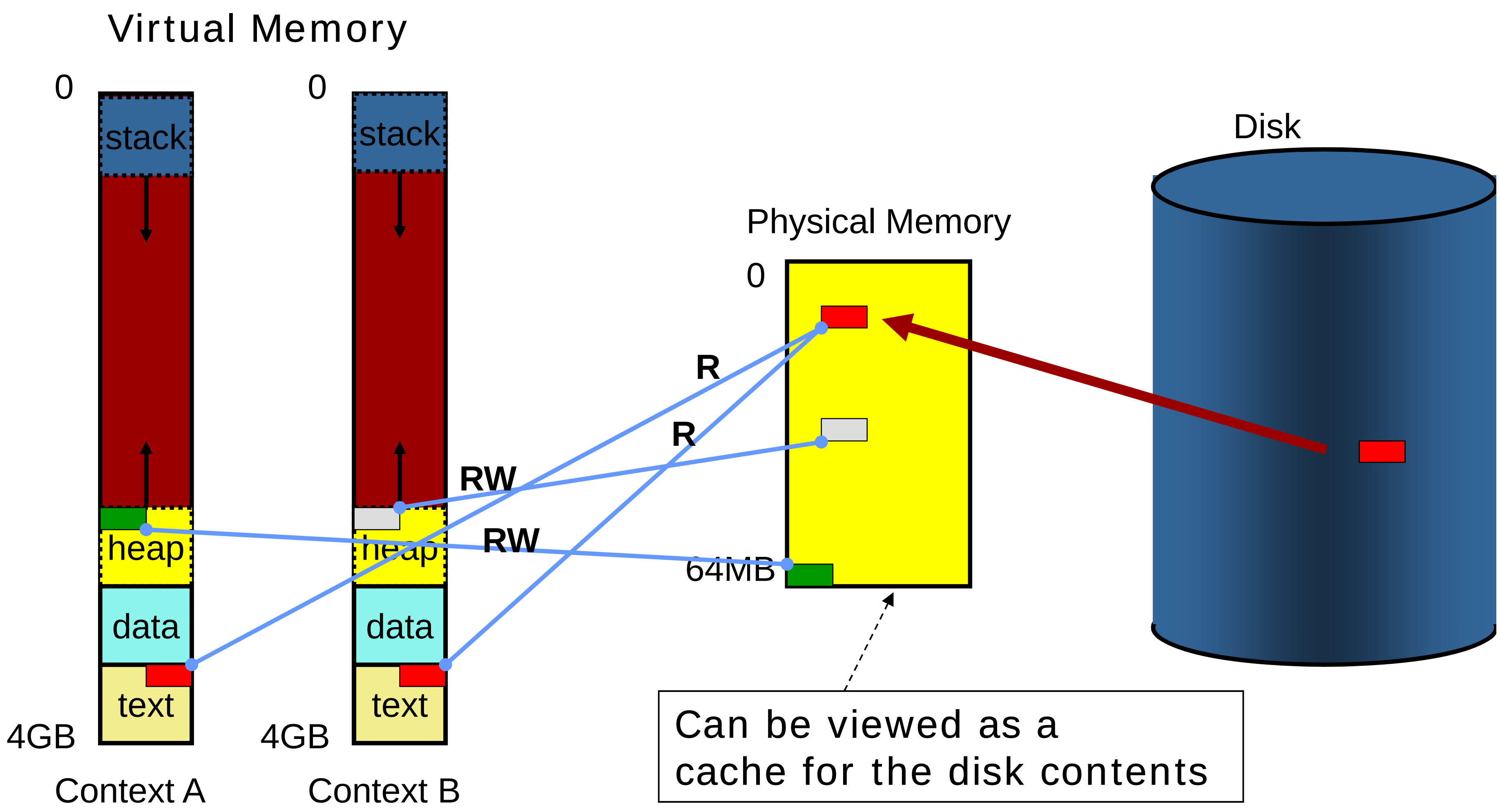


UPPSALA
UNIVERSITET

Virtual and Physical Memory



Translation & Protection

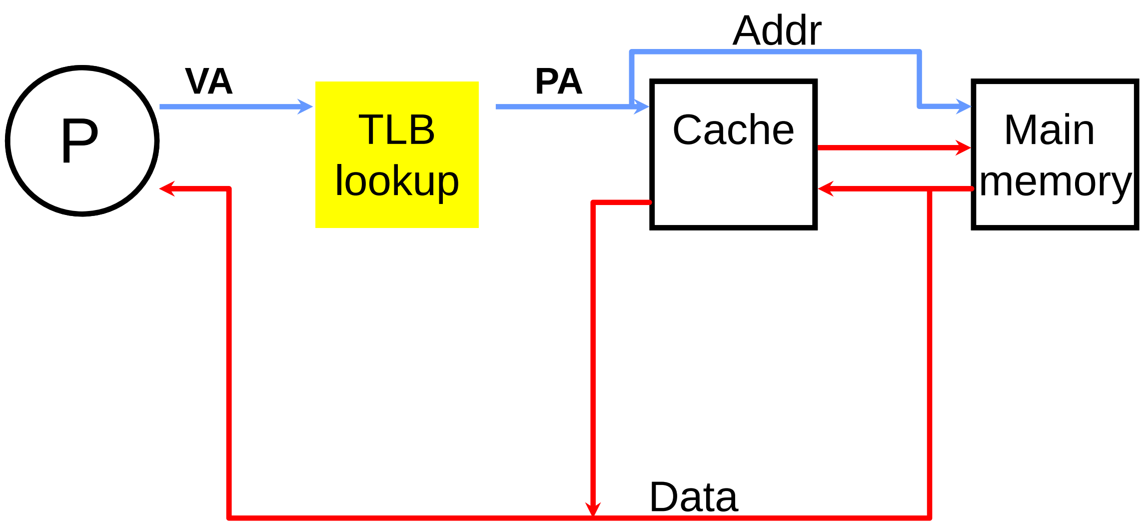


Fast address translation

How to quickly and cheaply find the right physical page in the [fully associativity cache called] physical memory?

- Store the most commonly used address translations in a **cache**—***Translation Look-aside Buffer*** (TLB)

  ==> *The caches rears their ugly faces again!*

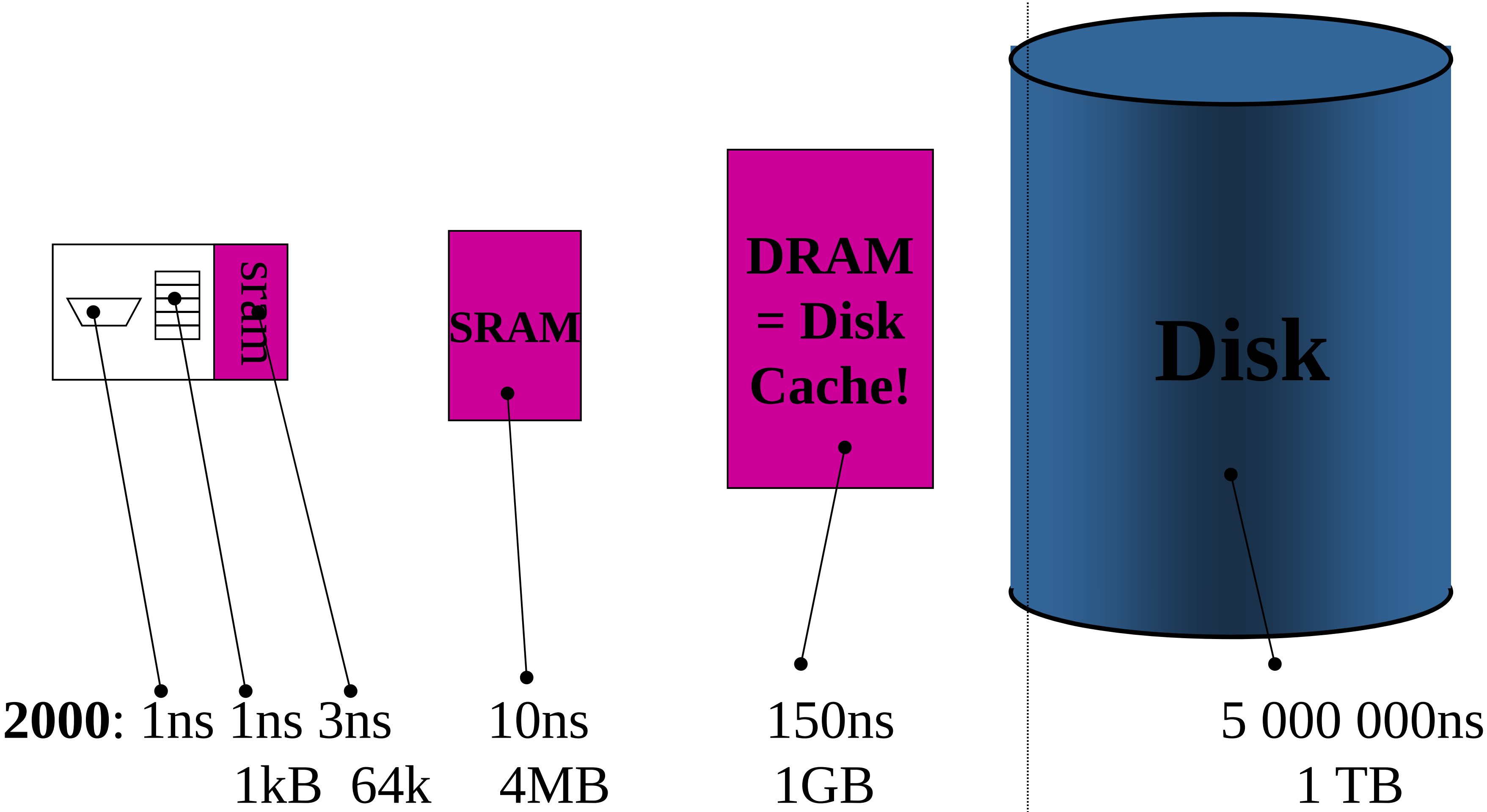


VM dictionary

Virtual Memory System	The "cache" languge
Virtual address	~Cache address
Physical address	~Cache location, "way info"
Page	~Huge cache block
Page fault	~Extremely painfull $miss
Page-fault handler	~The software filling the $
Page-out	Write-back if dirty
Any physical page frame can map any virtual page	Fully associative cache

UPPSALA UNIVERSITET

Memory / storage



UPPSALA UNIVERSITET

Caches Everywhere…

- L1 D cache
- L1 I cache
- L2 cache
- L3 cache
- ITLB
- DTLB
- Virtual memory system
- Branch predictors
- ...