Multiprocessors

Erik Hagersten
Uppsala University

UPPSALA UNIVERSITET

Outline of these lectures

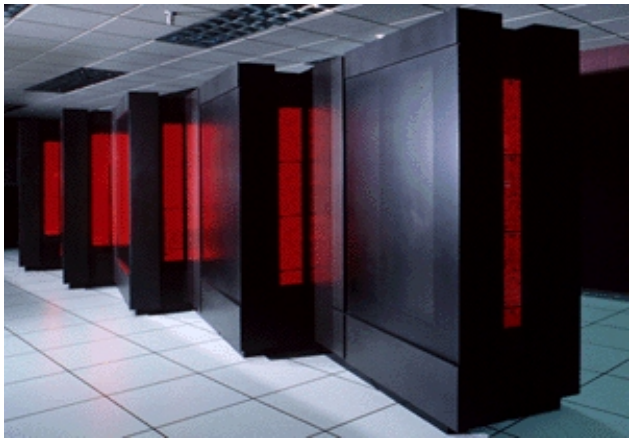
1. Uniprocessors
2. Multiprocessors
   - Coherence
   - Memory ordering
   - Vector instructions, GPUs
3. Multicores & Manycores
4. Optimizing for speed

Parallel Comp 2012

UPPSALA UNIVERSITET

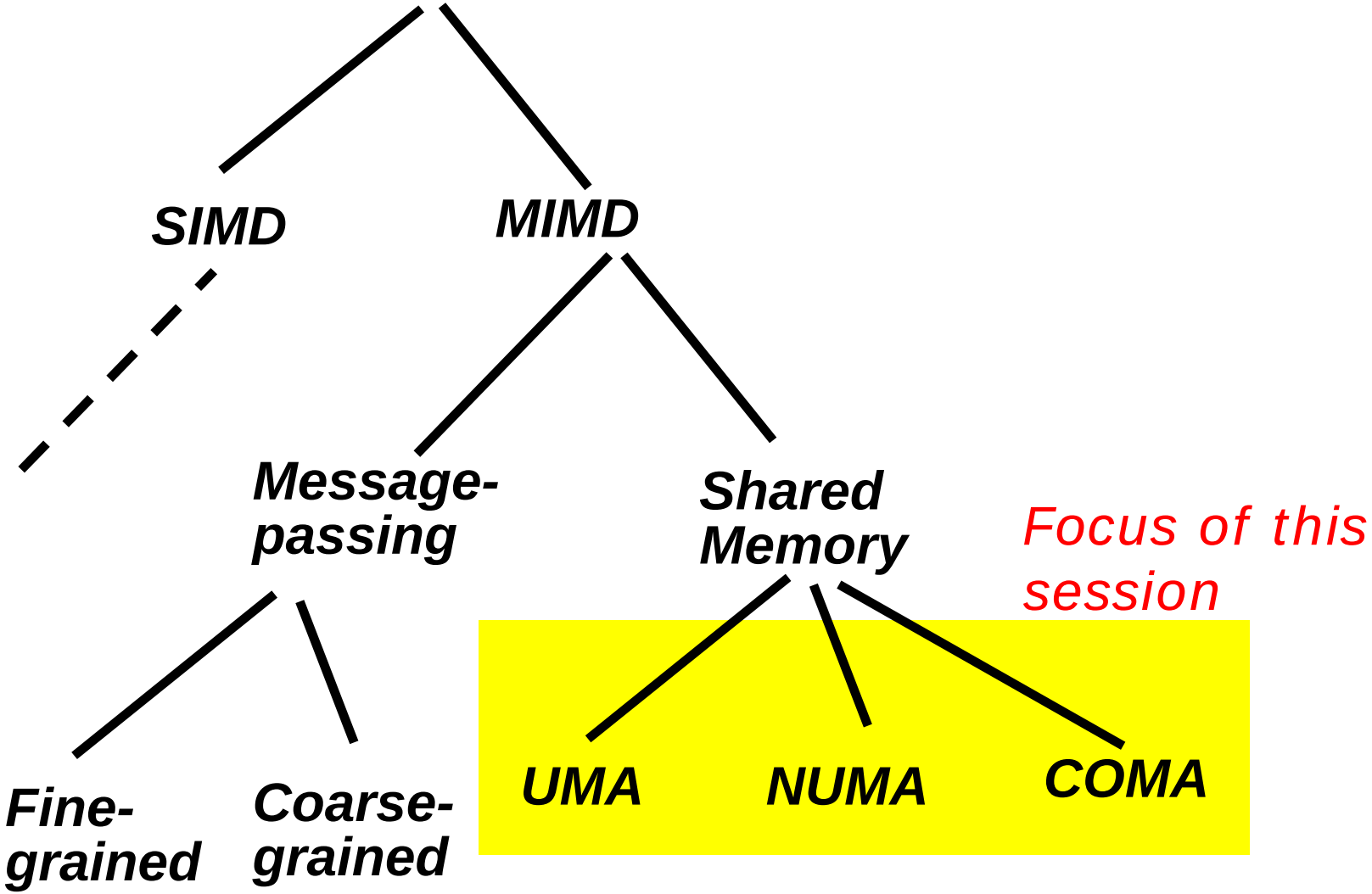
The era of the “supercomputer” multiprocessors

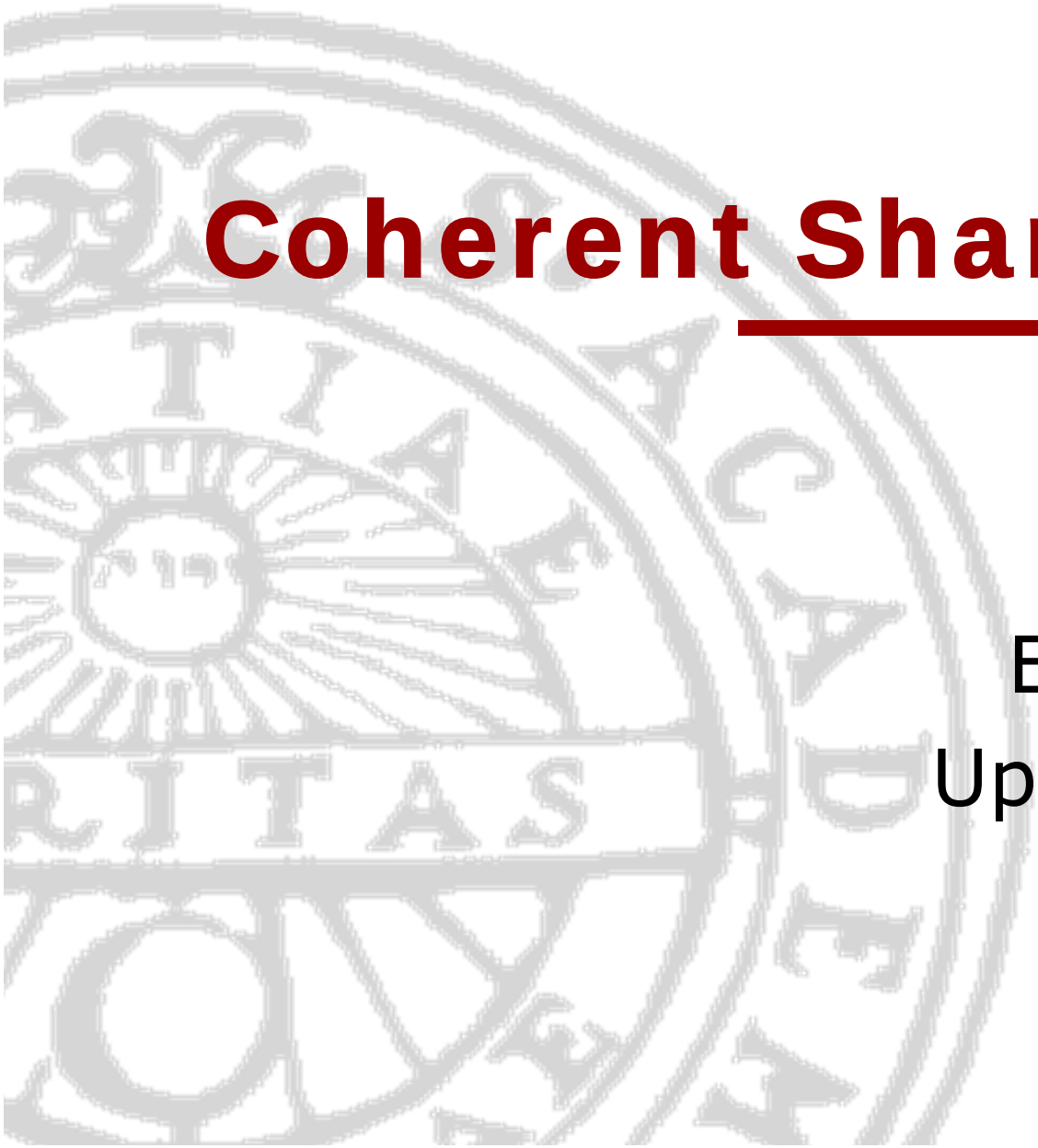
- The one with the most blinking lights wins
- The one with the strangest languages wins
- The niftier the better!



Parallel Comp 2012

Taxomy for Architectures [Flynn]

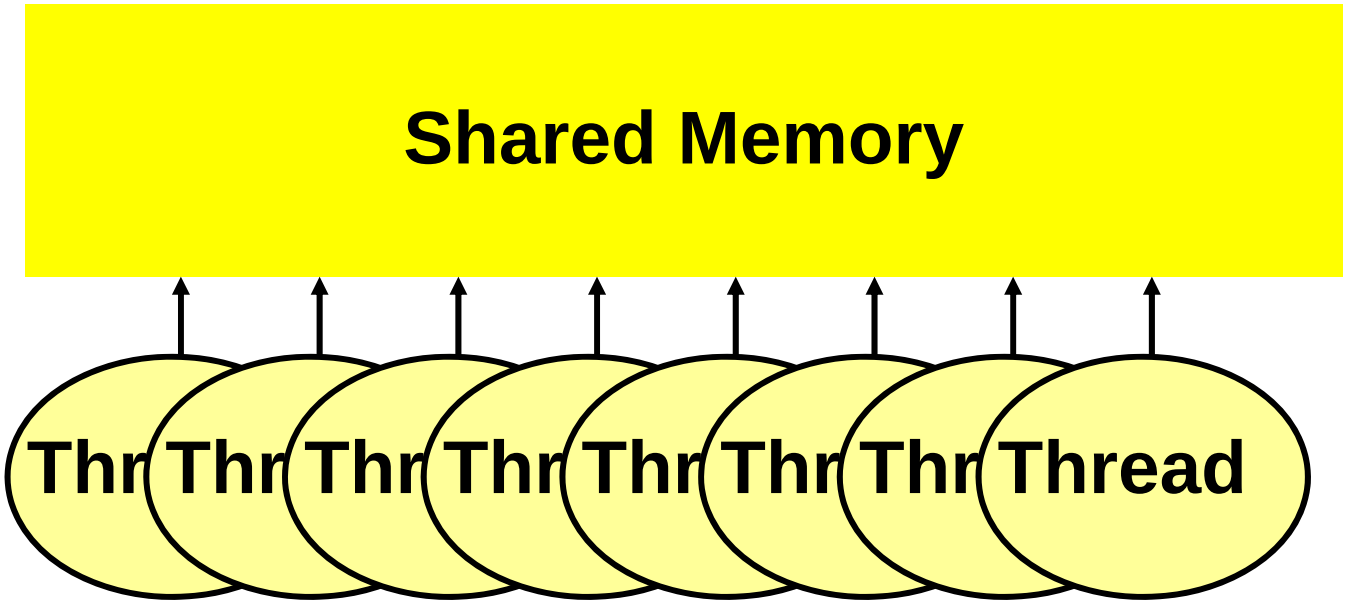




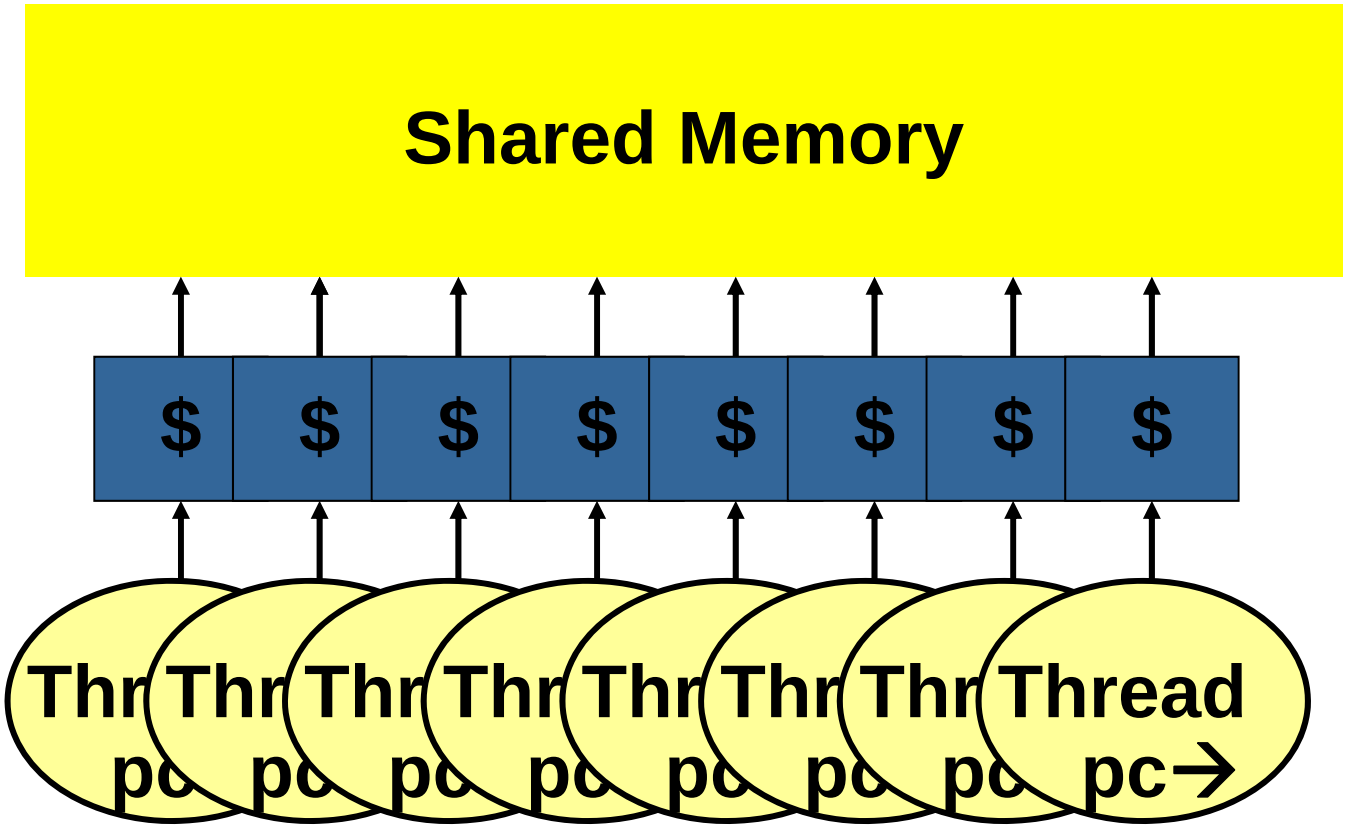
Coherent Shared Memory

Erik Hagersten
Uppsala University

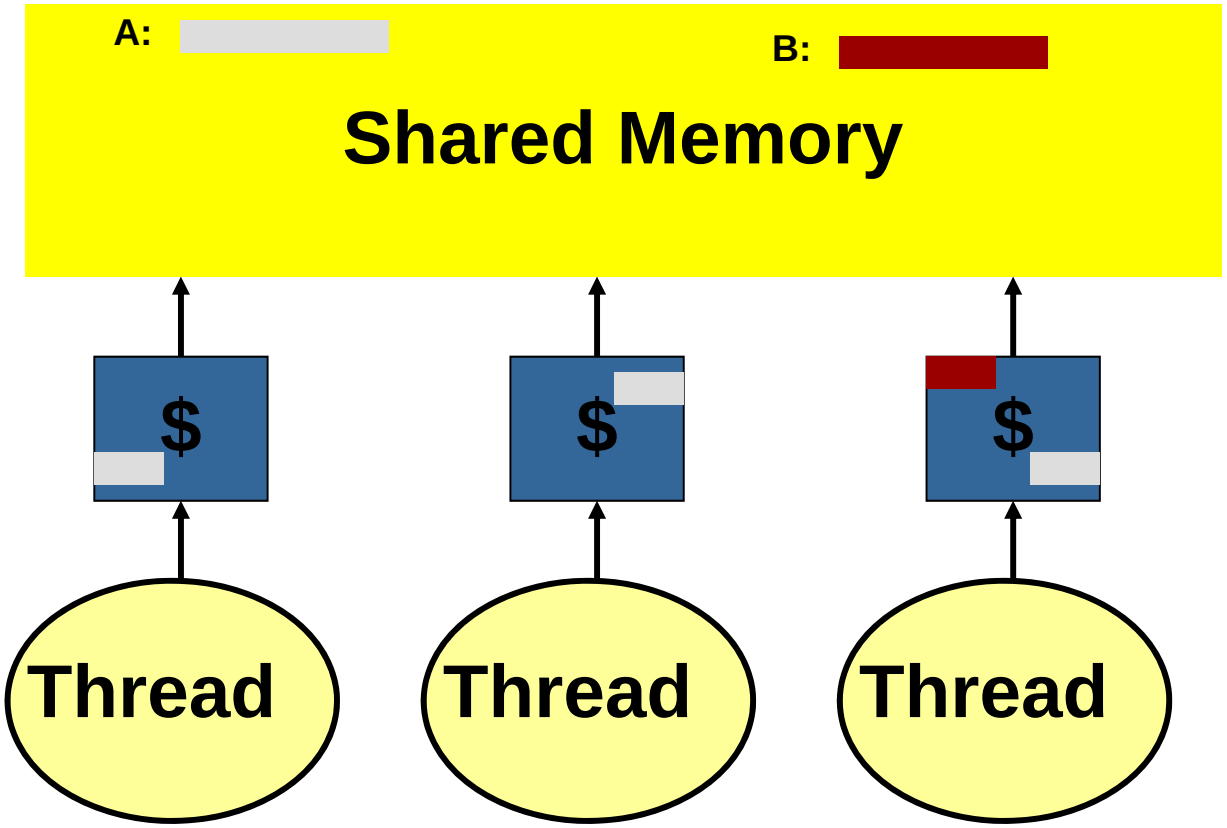
Programming Model:



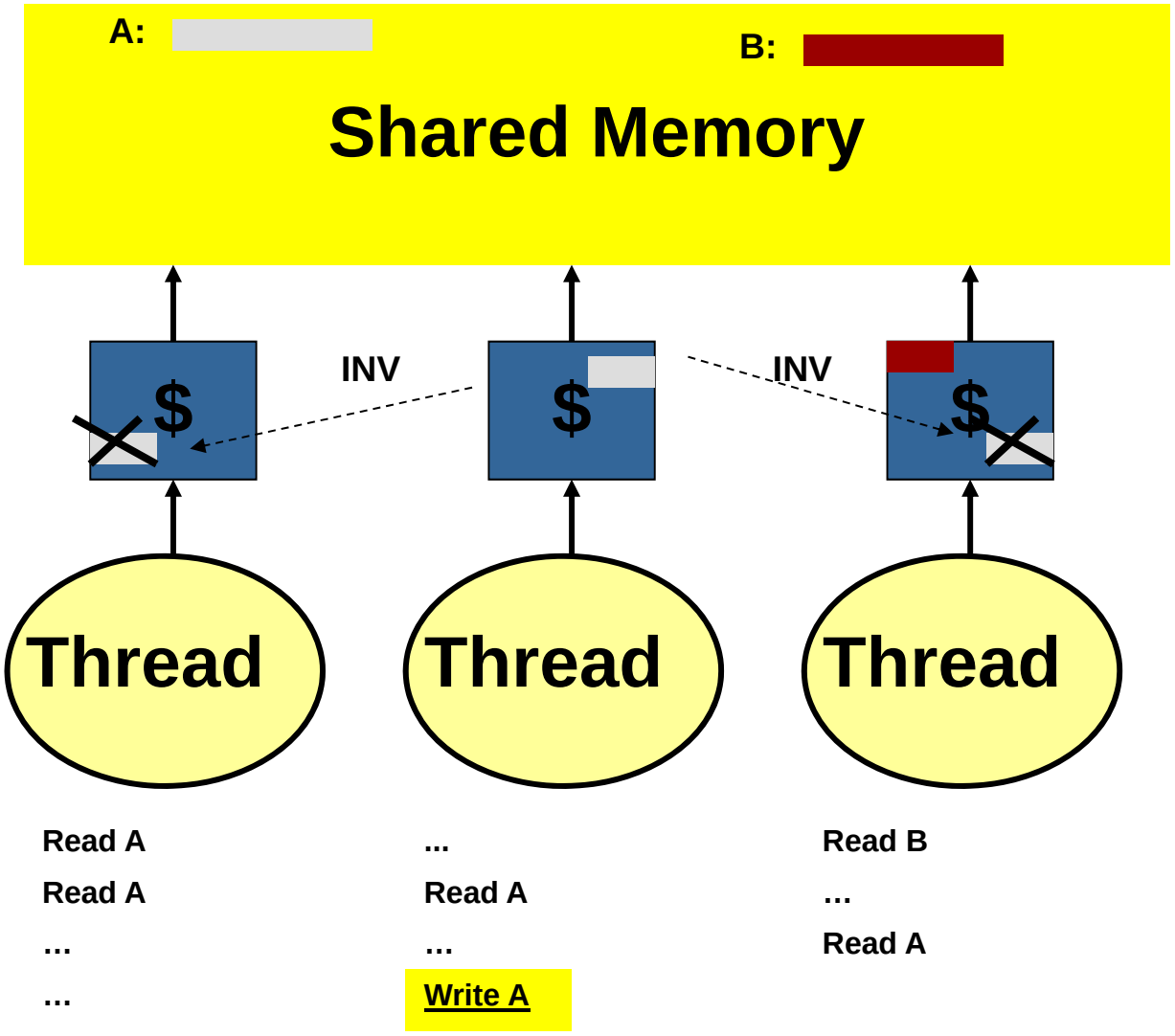
Adding Caches: More Concurrency



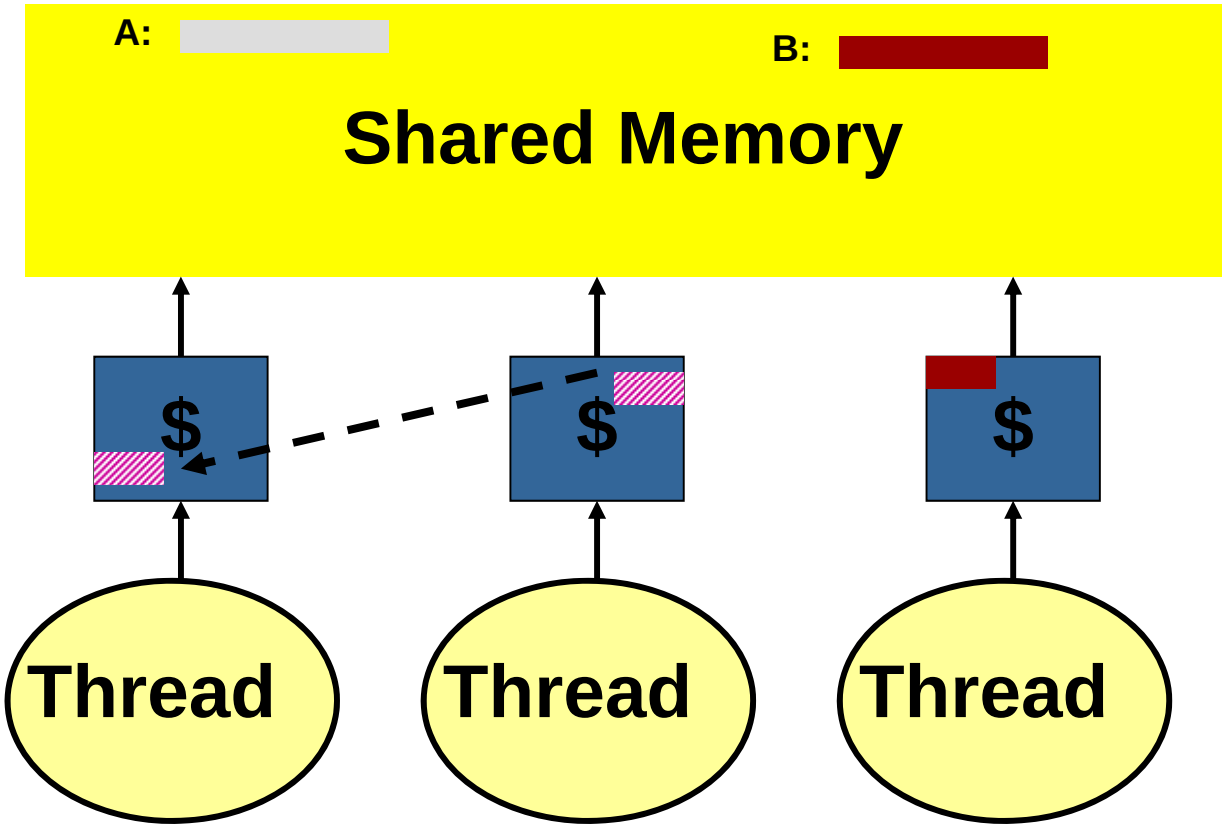
Caches: Automatic Replication of Data



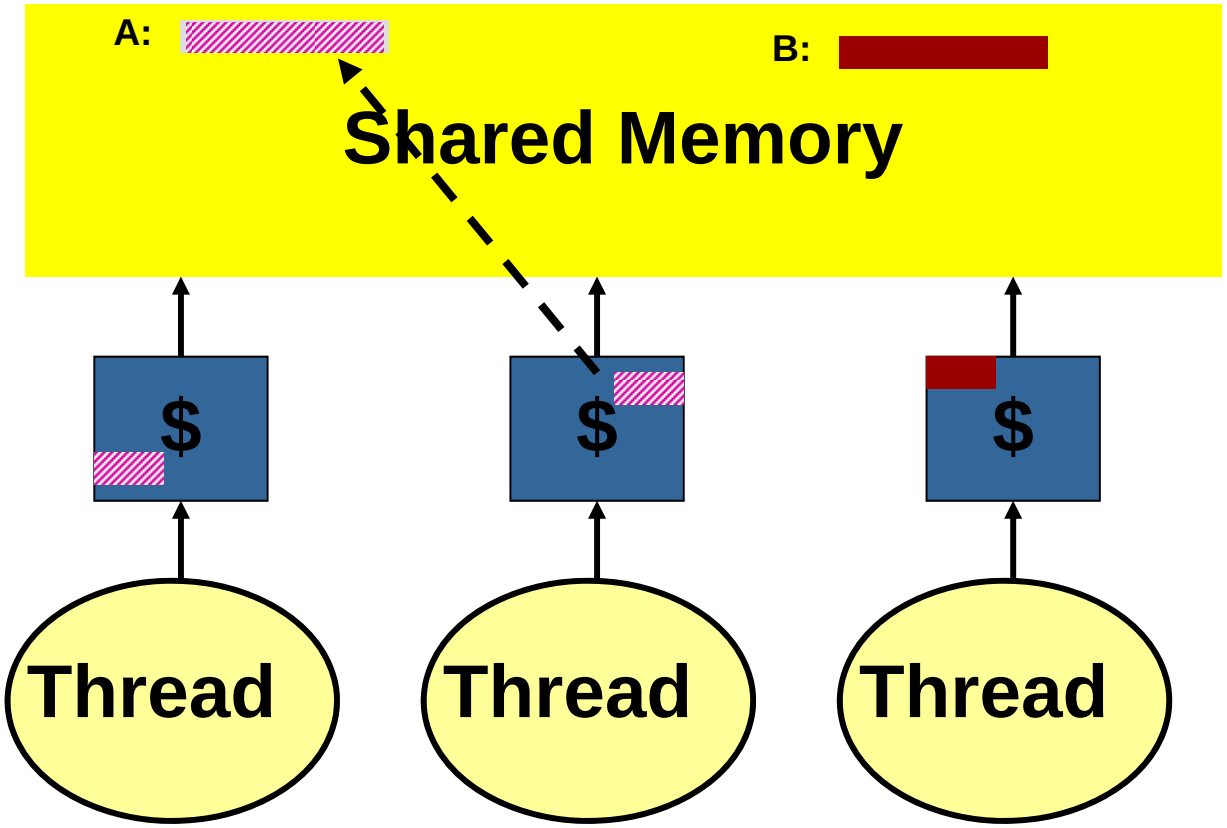
The Cache Coherent Memory System



The Cache Coherent 2



Writeback



Read A	...	Read B
Read A	Read A	...
...	...	Read A
...	Write A ...	
Read A	A gets replaced	

UPPSALA UNIVERSITET

Summing up Coherence

Sloppy: there can be many copies of a datum, but only one valu

Too strong definition!

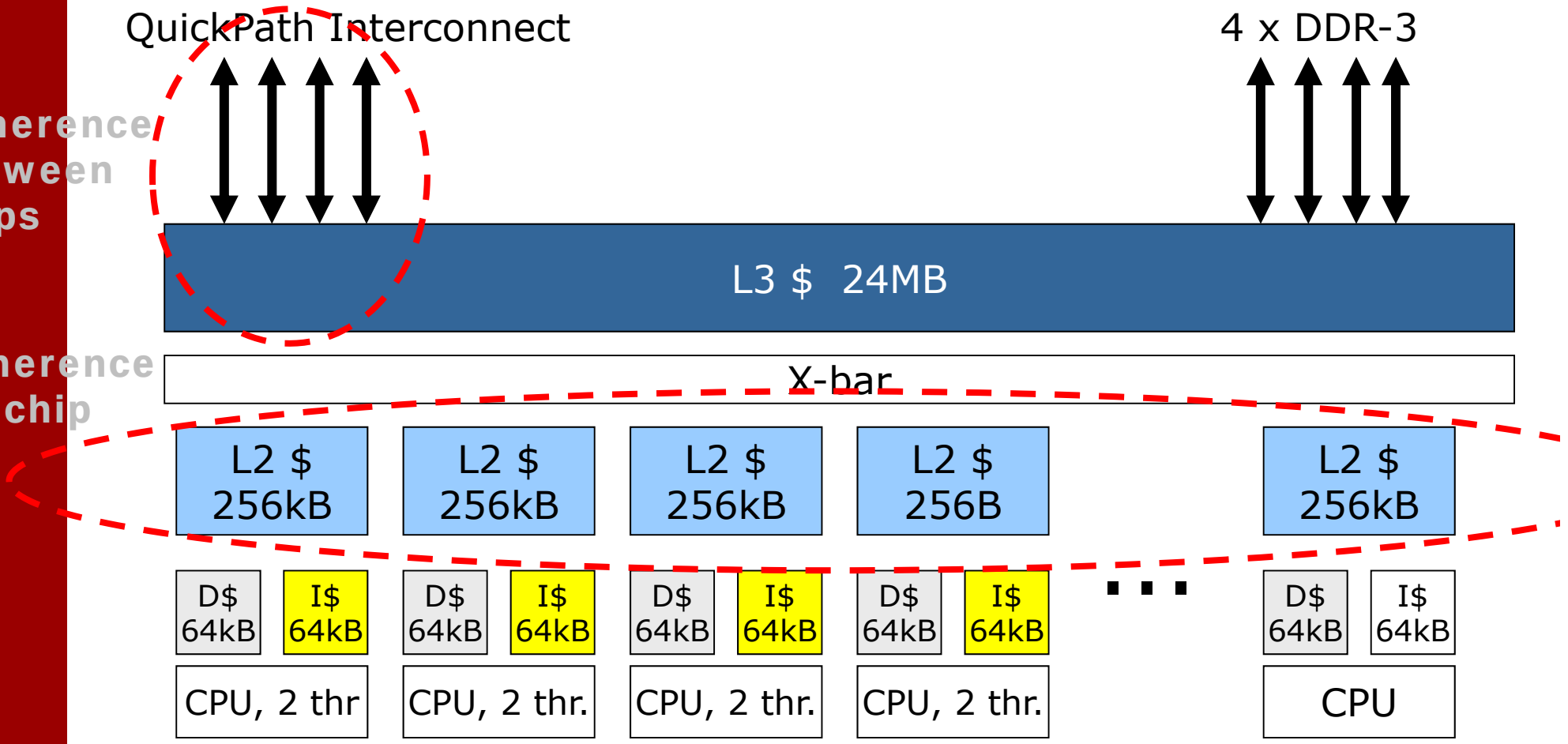
Coherence: *There is a single global order of value changes to each datum*

Memory order/model: *Defines the order between accesses to many data*

Where does coherence matter?

Coherence Between chips

Coherence On chip



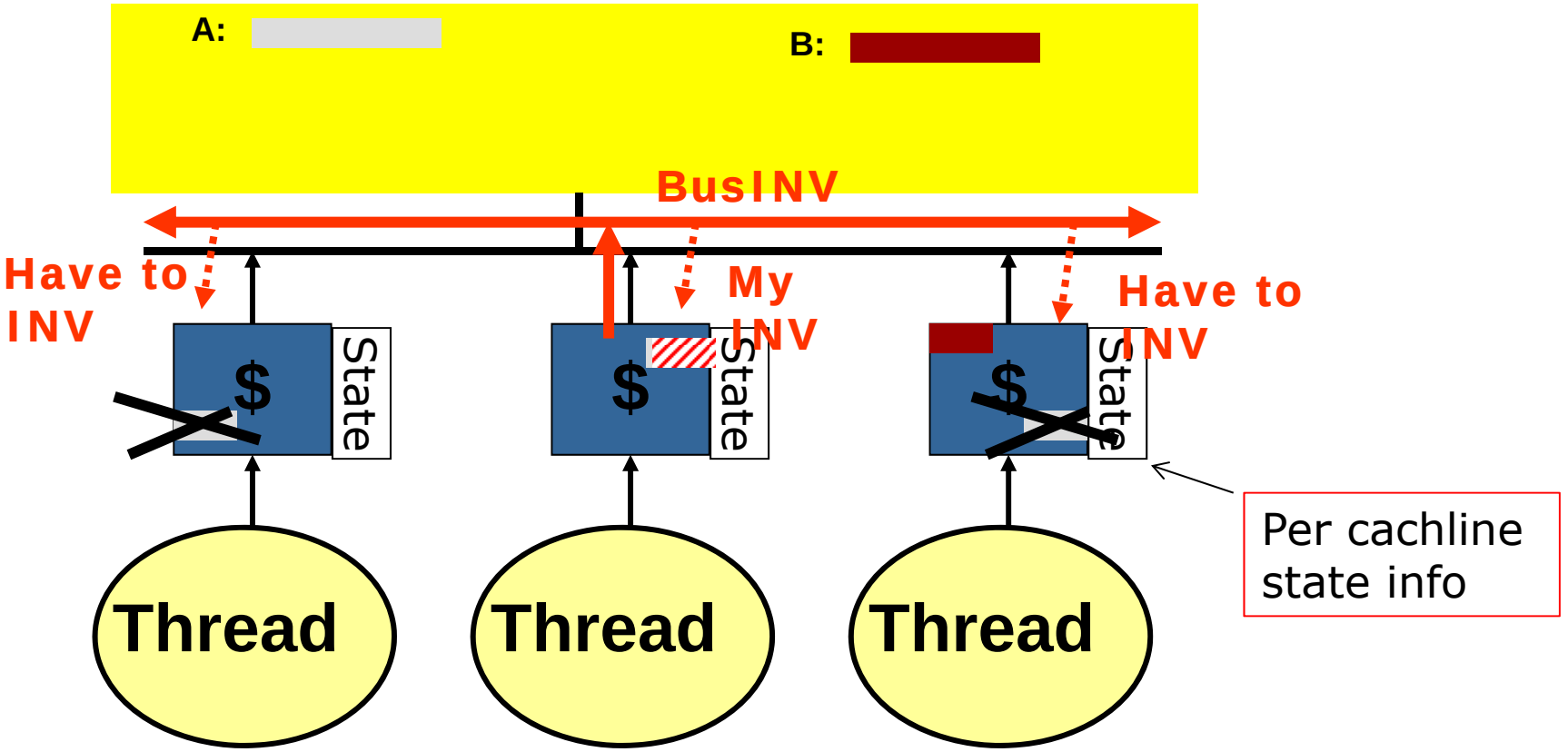
8 cores x 2 threads

UPPSALA
UNIVERSITET

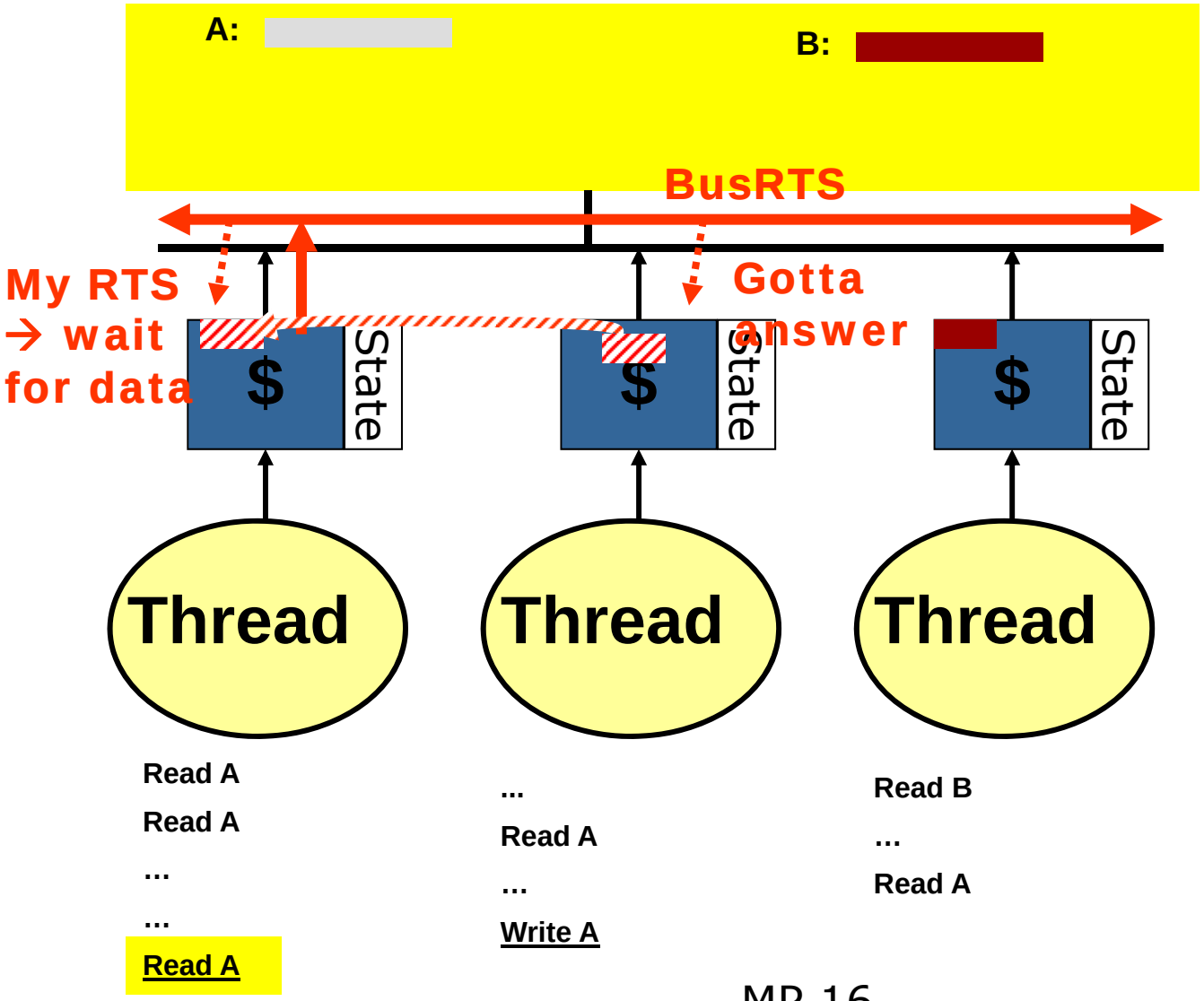
Implementing Coherence

Parallel
Comp
2012

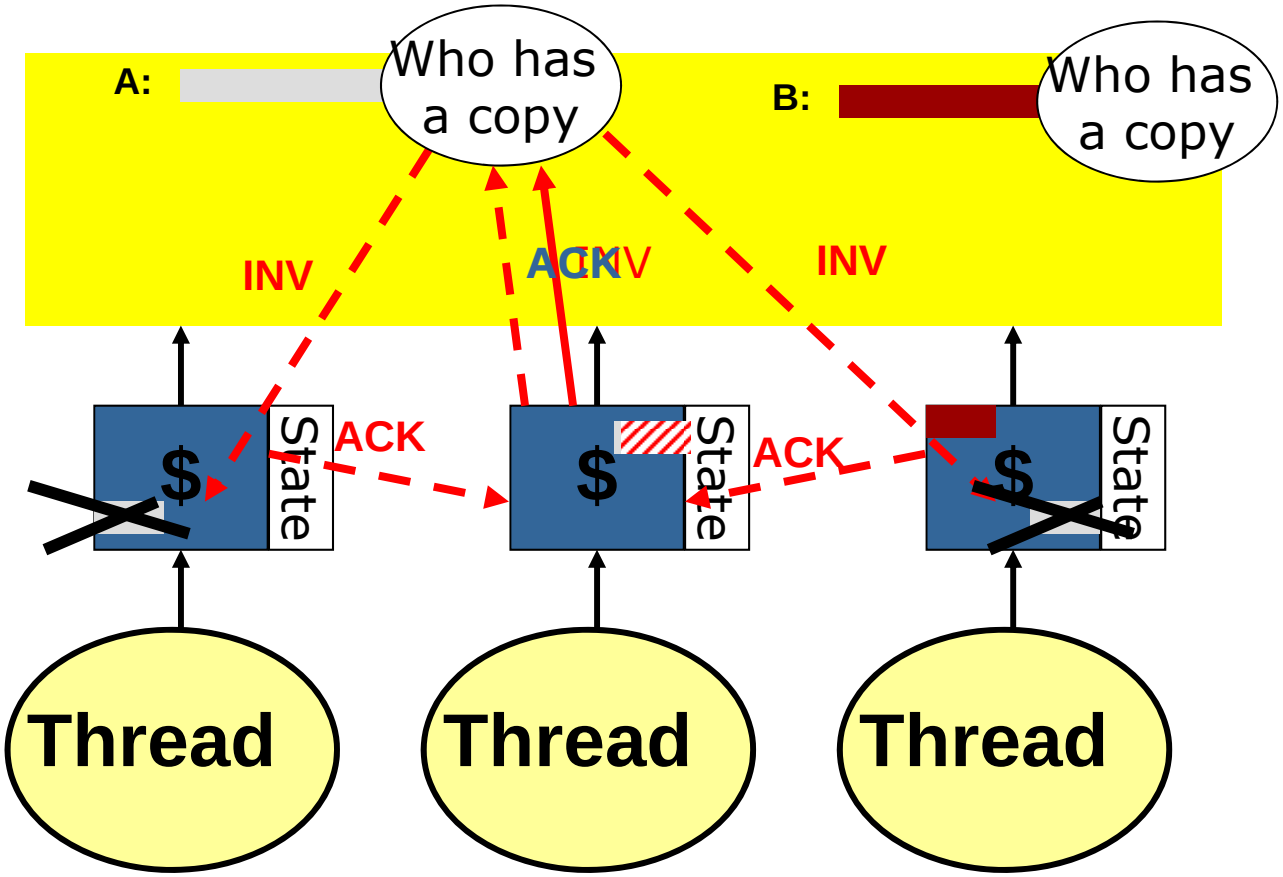
”Upgrade” in snoop-based



Cache-to-cache in snoop-based

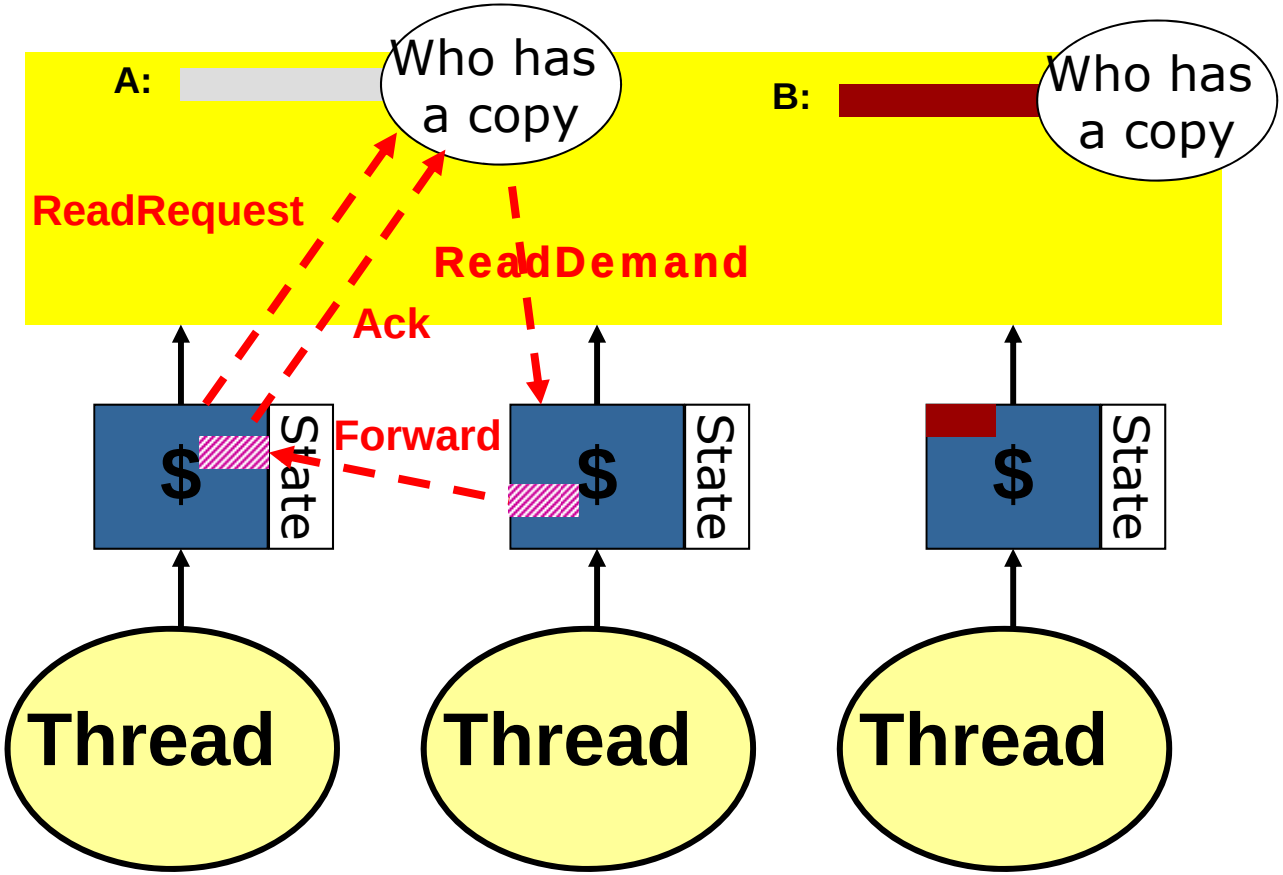


”Upgrade” in dir-based

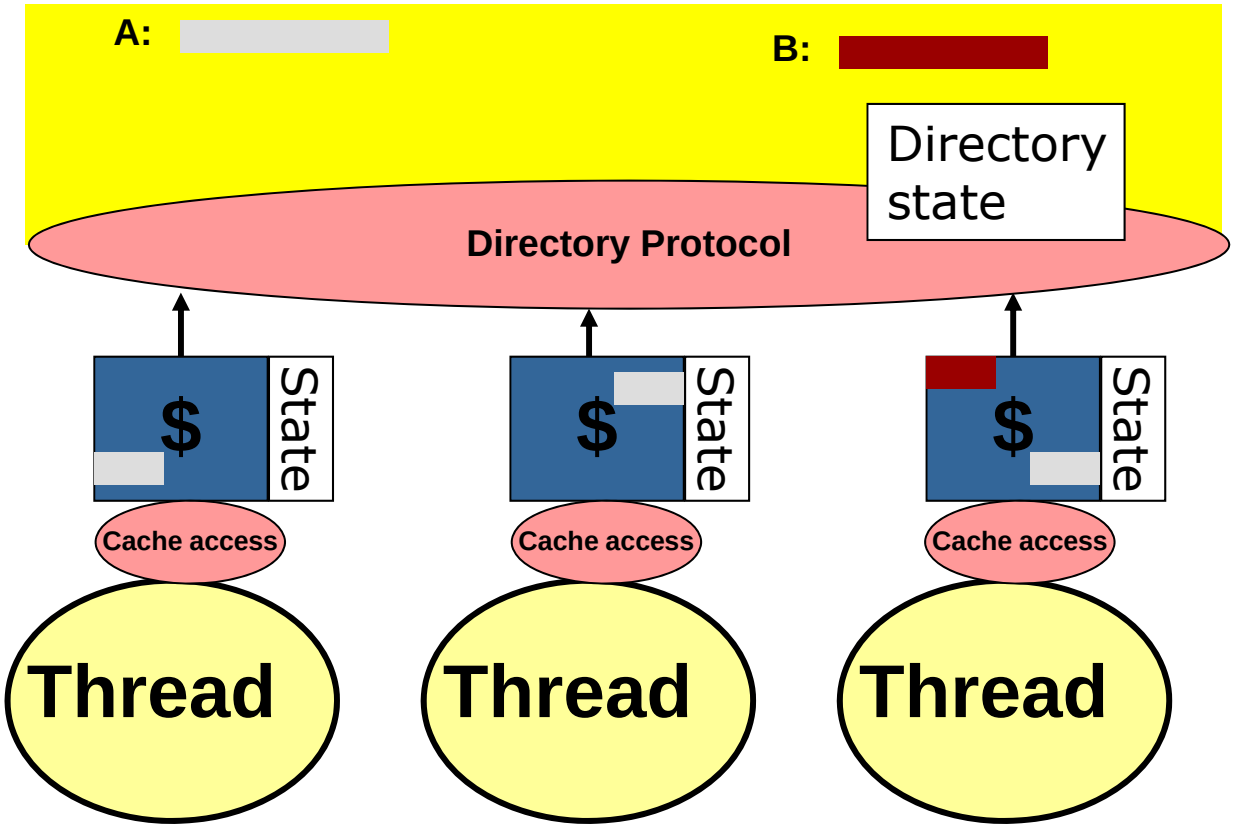


Read A	...	Read B
Read A	Read A	...
...	...	Read A
...	**Write A**	

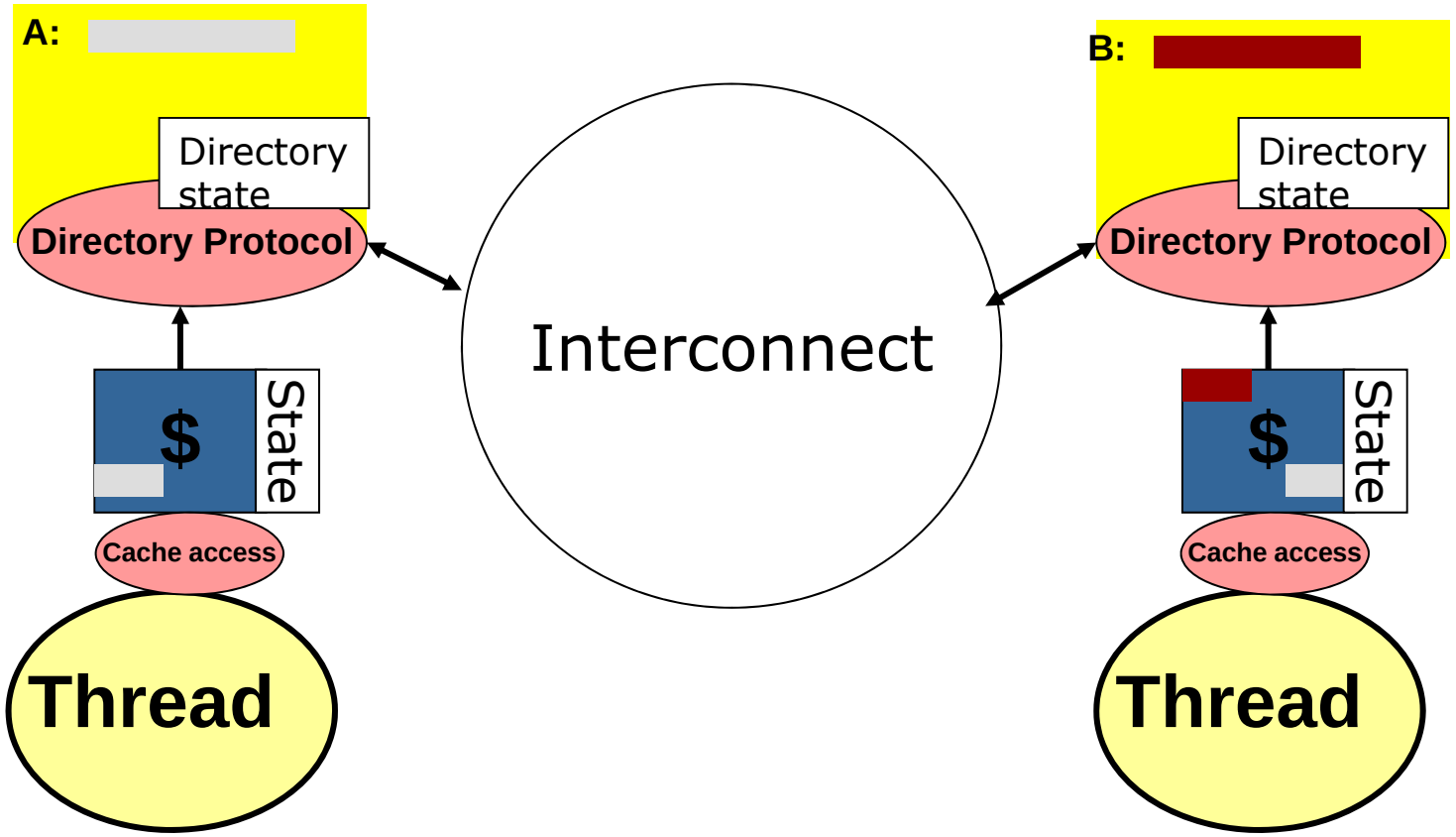
Cache-to-cache in dir-based



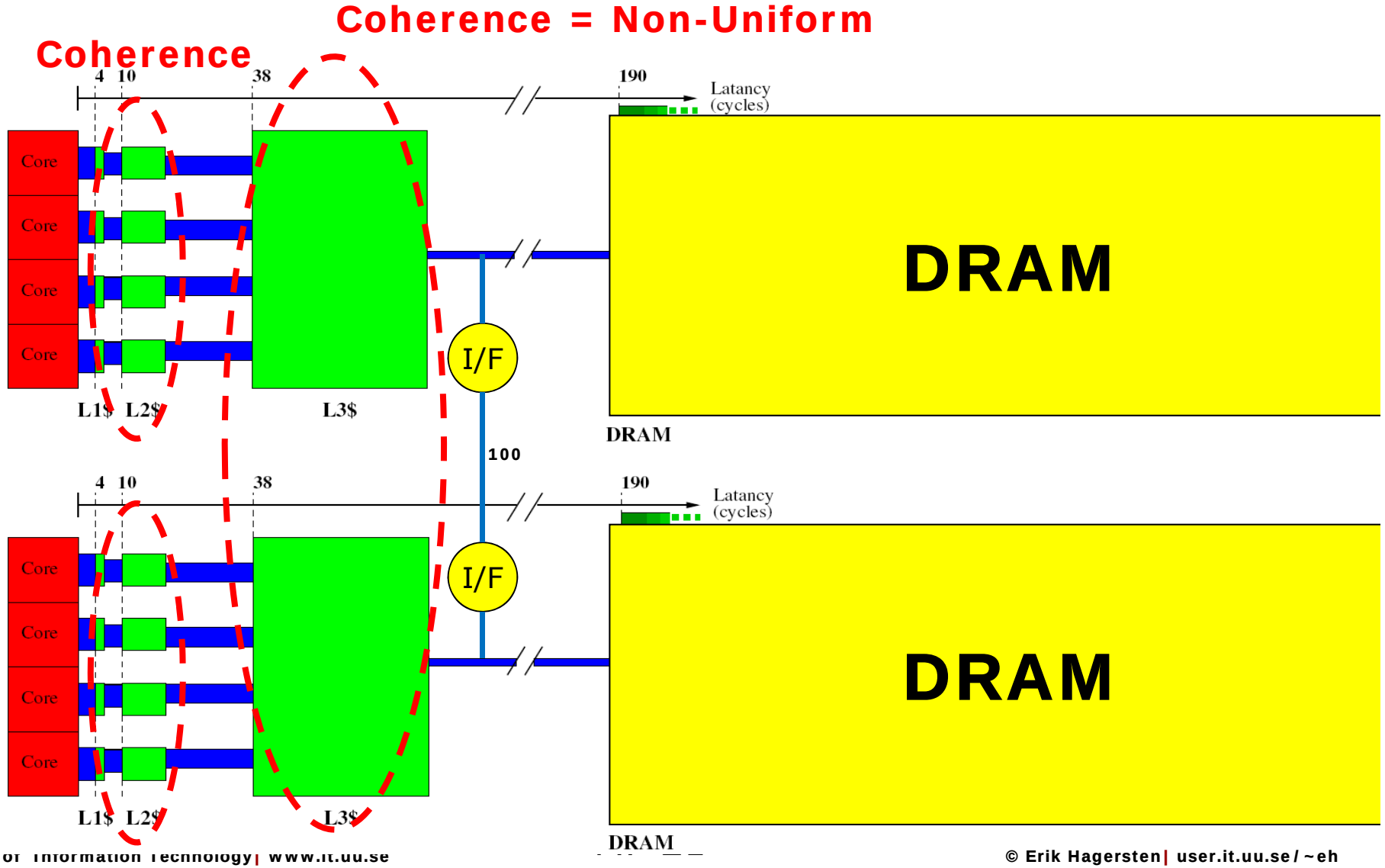
Directory-based coherence: Per-cachline info in the memory



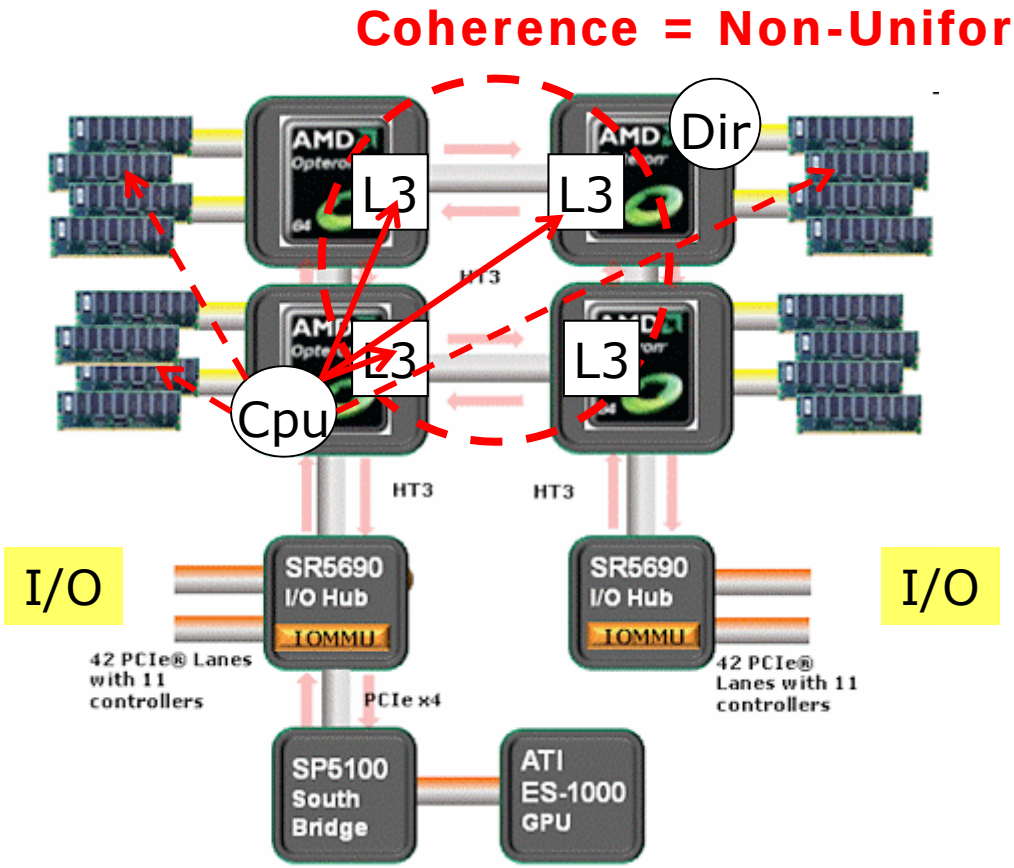
Directory-based snooping: NUMA. Per-cachline info in the home node



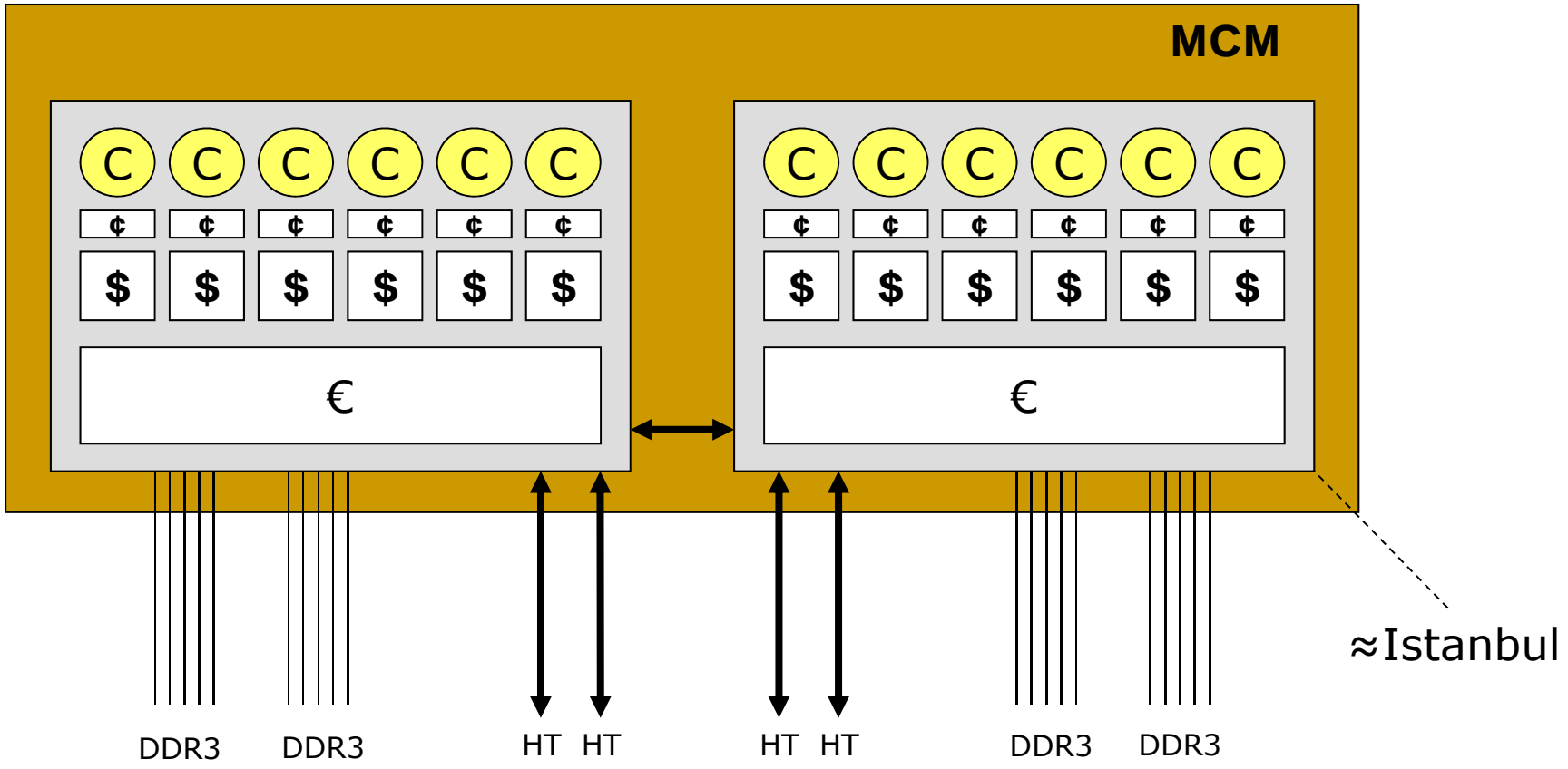
Multisocket



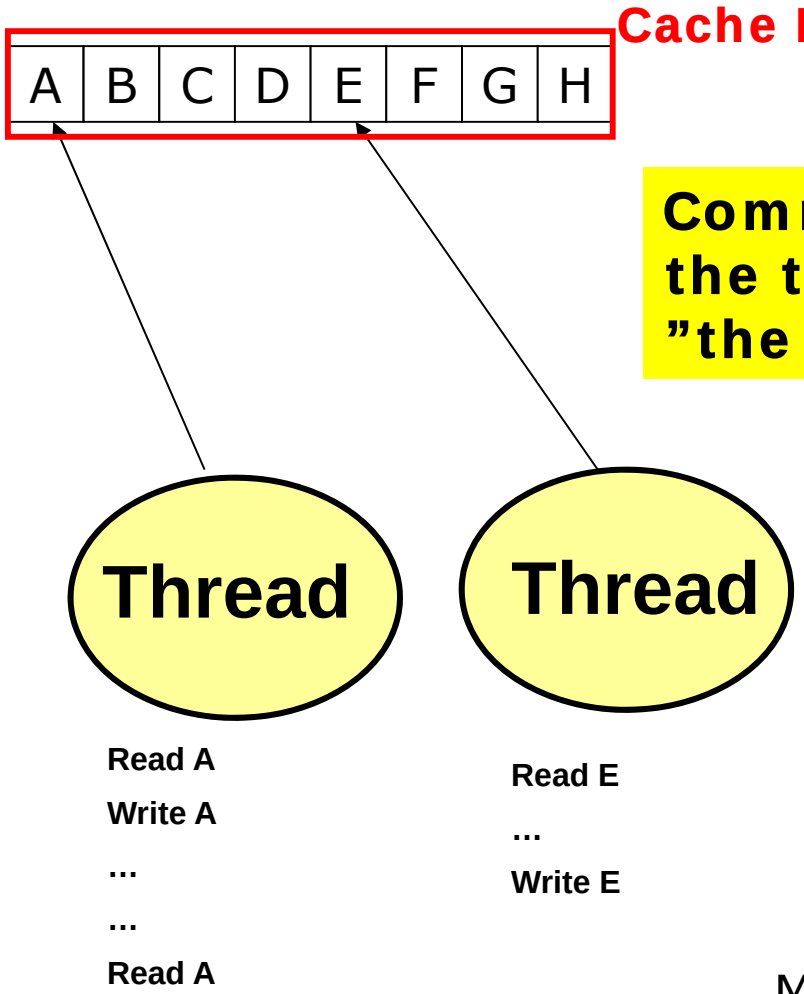
AMD Multi-socket Architecture (same applies to Intel multi-sockets)



AMD Magny-Cours
NUMA & NUCA on a socket
Non-Uniform Memory Architecture
Non-Uniform Communication Architecture



False sharing:
Coherence is maintained with a cache-line granularity



Communication misses even though the threads do not share data
”the cache line is too large”

UPPSALA UNIVERSITET

More Cache Lingo

- **Capacity miss** – too small cache
- **Conflict miss** – limited associativity
- **Compulsory miss** – accessing data the first time
- **Coherence miss** – I would have had the data unless it had been invalidated by someone else
- **Upgrade miss** (only for writes) – I would have had a writable copy, but gave away readable data and downgraded myself to read-only
- **False sharing:** Coherence/downgrade is caused by a shared cacheline, to by shared data:

False sharing example:

Read A	**…**
…	**Read D**
Write A	**…**
…	**Write D**
Read A	

cacheline:

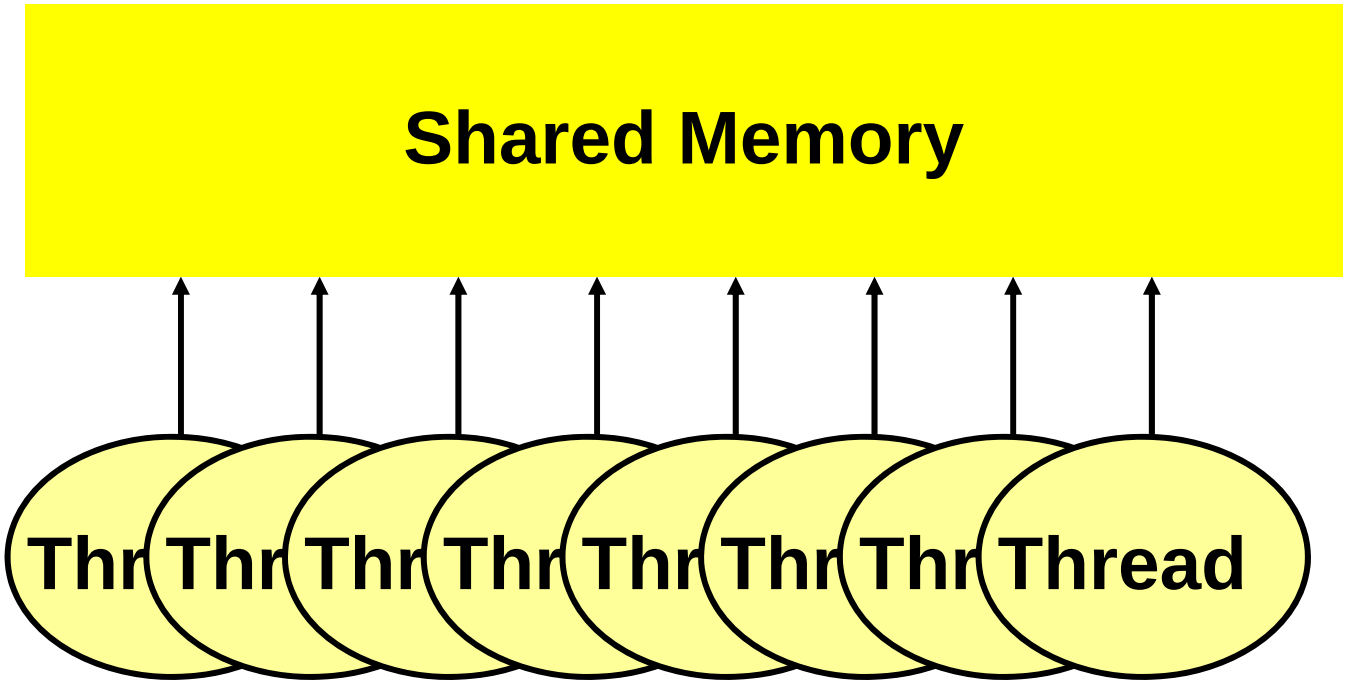
A, B, C, D

Parallel Comp 2012

Memory Ordering (aka Memory Consistency) -- tricky but important stuff

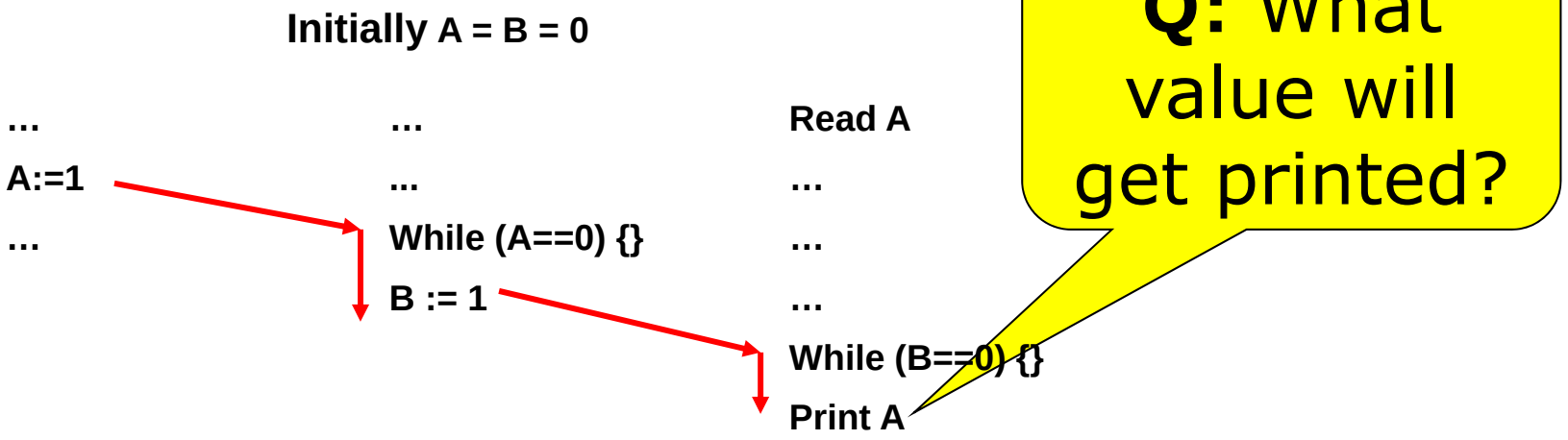
Erik Hagersten
Uppsala University
Sweden

The Shared Memory Programming Model (Pthreads/OpenMP, …)

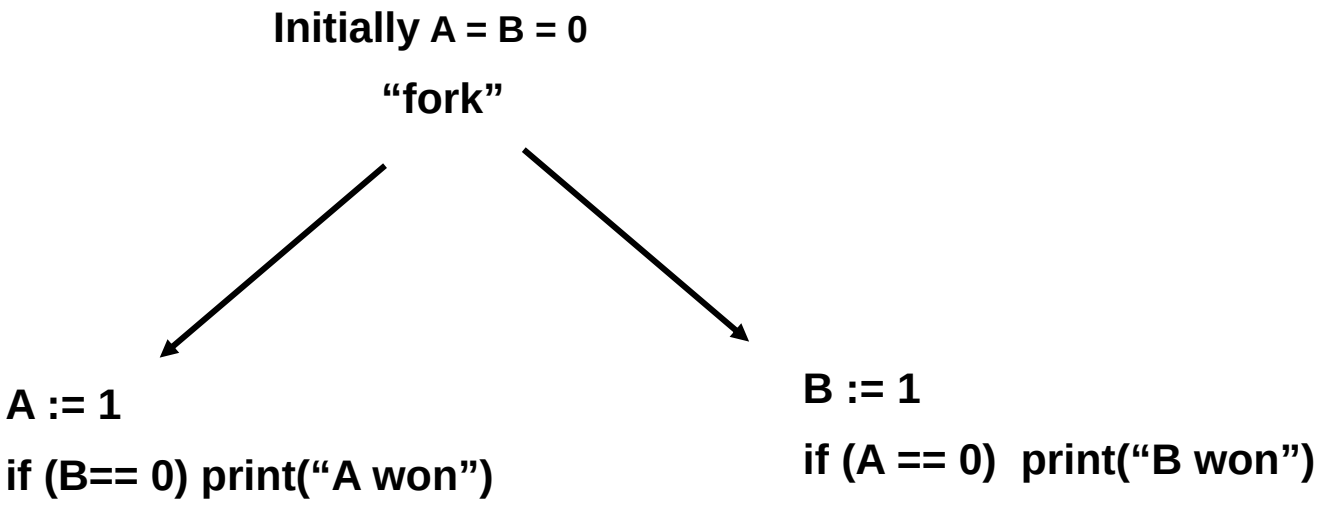


Memory Ordering

- Coherence defines a per-datum valuechange order
- Memory model defines the valuechange order for all the data.



Dekker's Algorithm



Q: Is it possible that both A and B win?

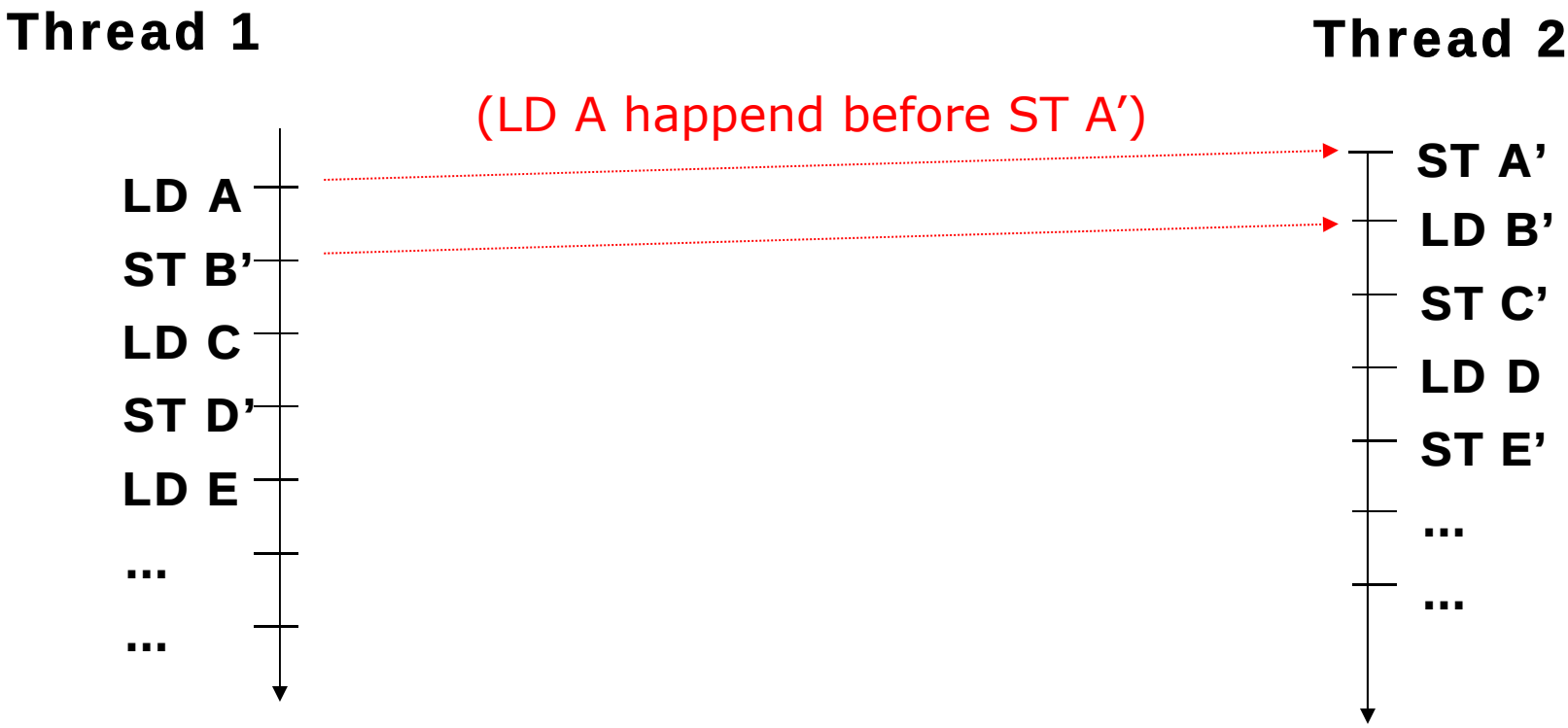
UPPSALA UNIVERSITET

Memory Ordering

- Defines the guaranteed memory ordering: *If a thread has seen that A happens before B, what order can the other threads observe?*
- Is a ”contract” between the HW and SW guys
- Without it, you can not say much about the result of a parallel execution

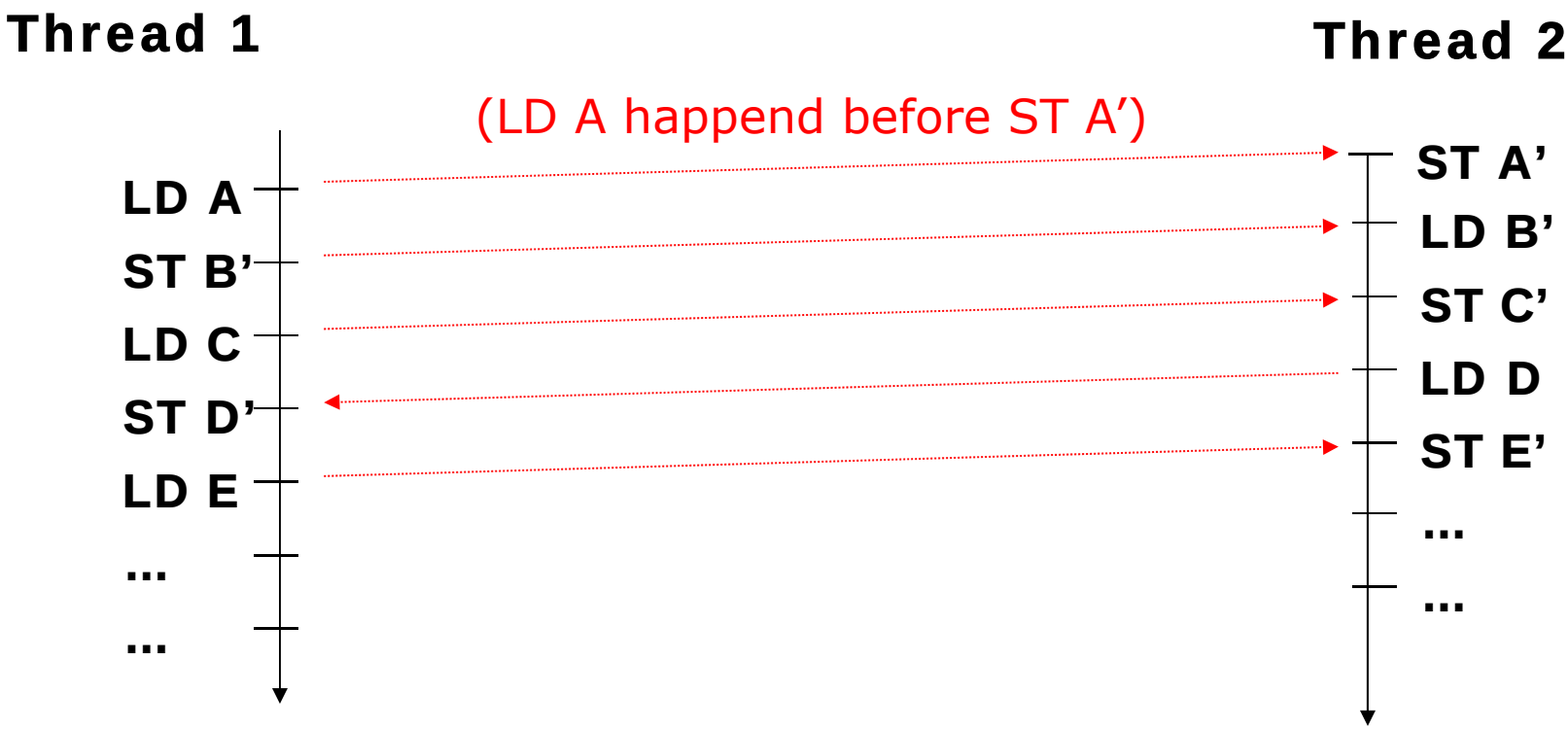
Human intuition: There is one global order!

(A' denotes a modified value to the data at addr A)



Human intuition: There is one global order!

(A' denotes a modified value to the data at addr A)



One possible observed order

Thread 1 Thread 2

LD A
ST B'
LD C
ST D'
LD E
...
...

ST A'
LD B'
ST C'
LD D
ST E'
...
...

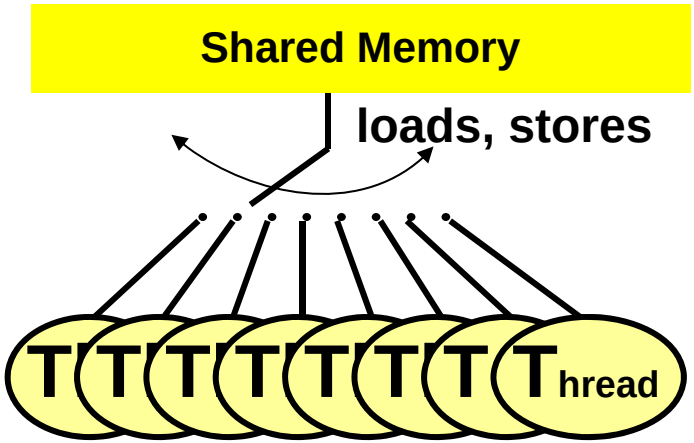
Another possible observed order

Thread 1 Thread 2

LD A
ST B'
LD C
ST D'
LD E
...
...

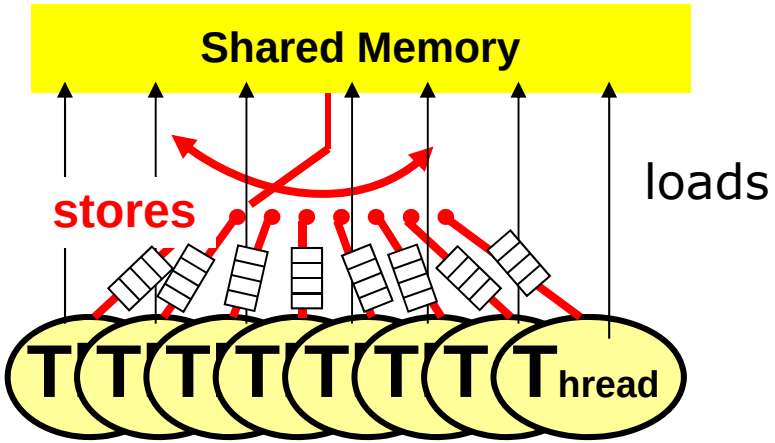
ST A'
LD B'
ST C'
LD D
ST E'
...
...

"The intuitive memory order" Sequential Consistency (Lamport)



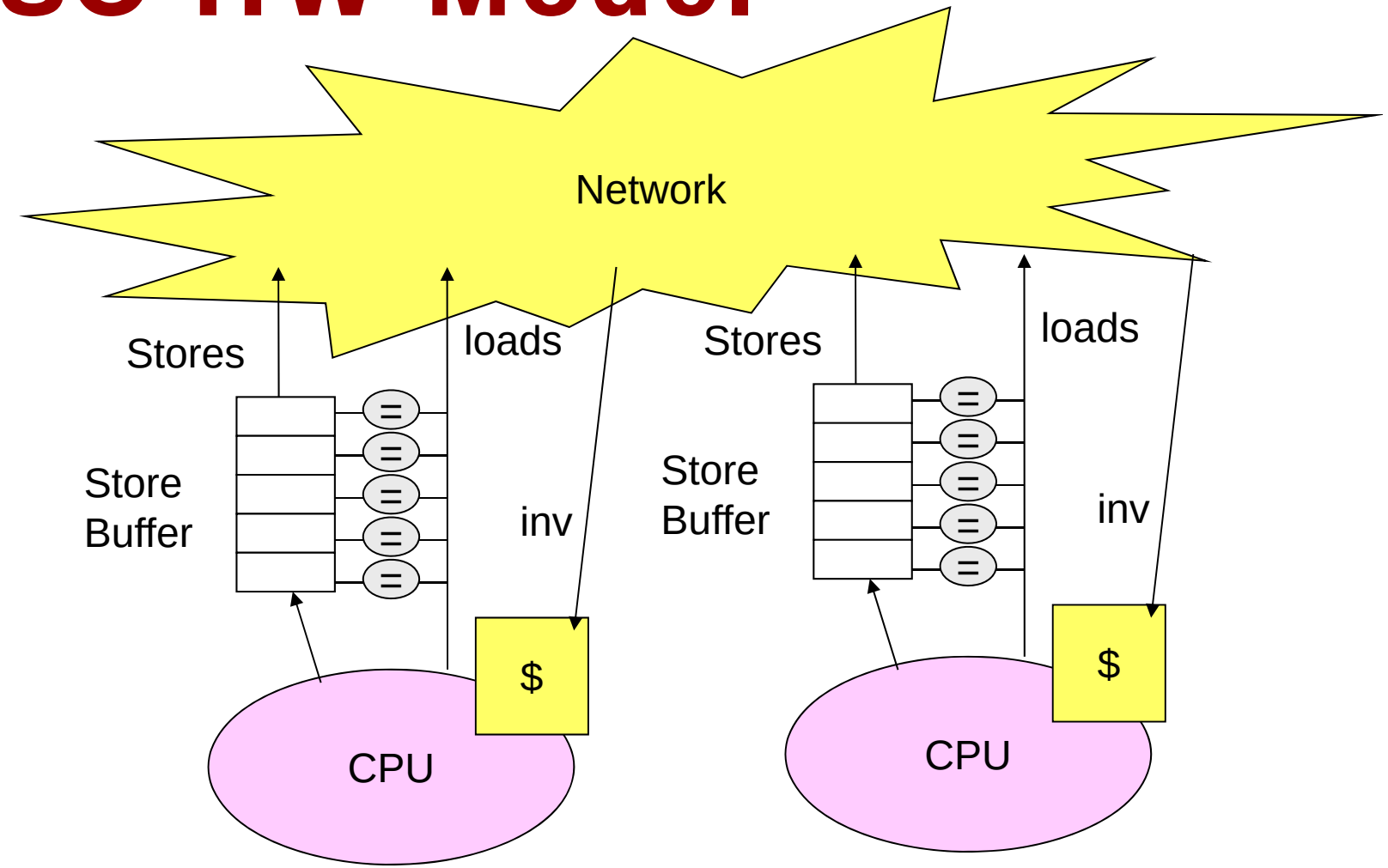
- Global order achieved by *interleaving* all memory accesses from different threads
- "Programmer's intuition is maintained"
  - Store causality? Yes
  - Does Dekker work? Yes
- Unnecessarily restrictive ==> performance penalty

"Almost intuitive memory model" Total Store Ordering [TSO] (P. Sindhu)



- Global *interleaving* [order] for <u>all</u> stores from different threads (own stores excepted)
- Is programmer's intuition is maintained?
  - Store causality? Yes
  - Does Dekker work? No
- Unnecessarily restrictive ==> performance penalty

TSO HW Model



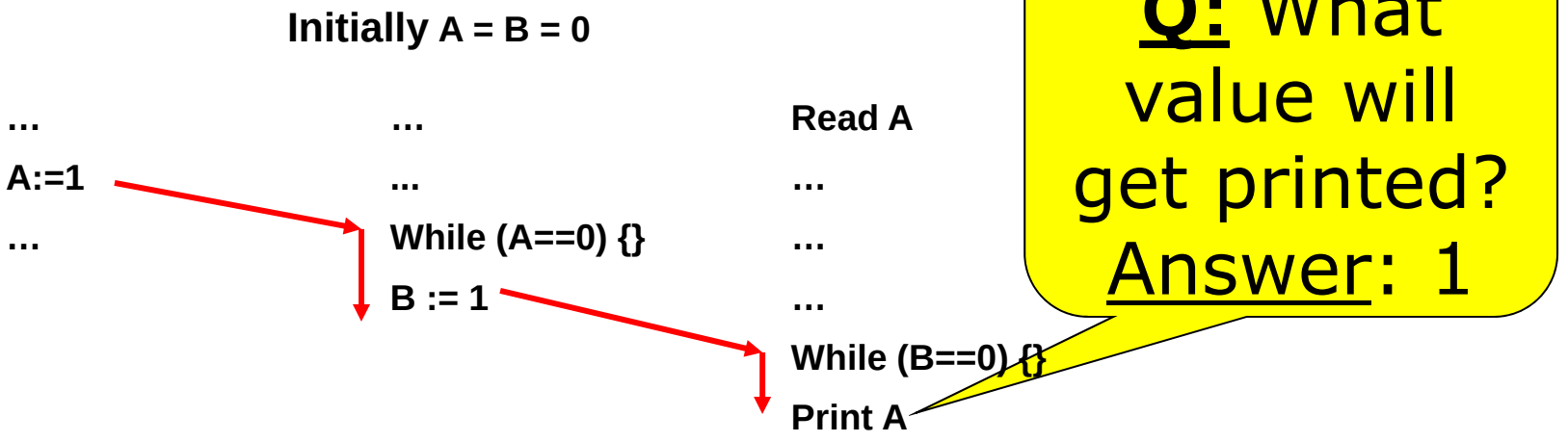
➔Stores are moved off the critical path

TSO

- Flag synchronization works

```
A := data                    while (flag != 1) {};
flag := 1                    X := A
```

- Provides causal correctness



Dekker's Algorithm, TSO

Initially A = B = 0

"fork"

Does the write become globally visible before the read is performed?

A := 1

if (B== 0) print("A won")

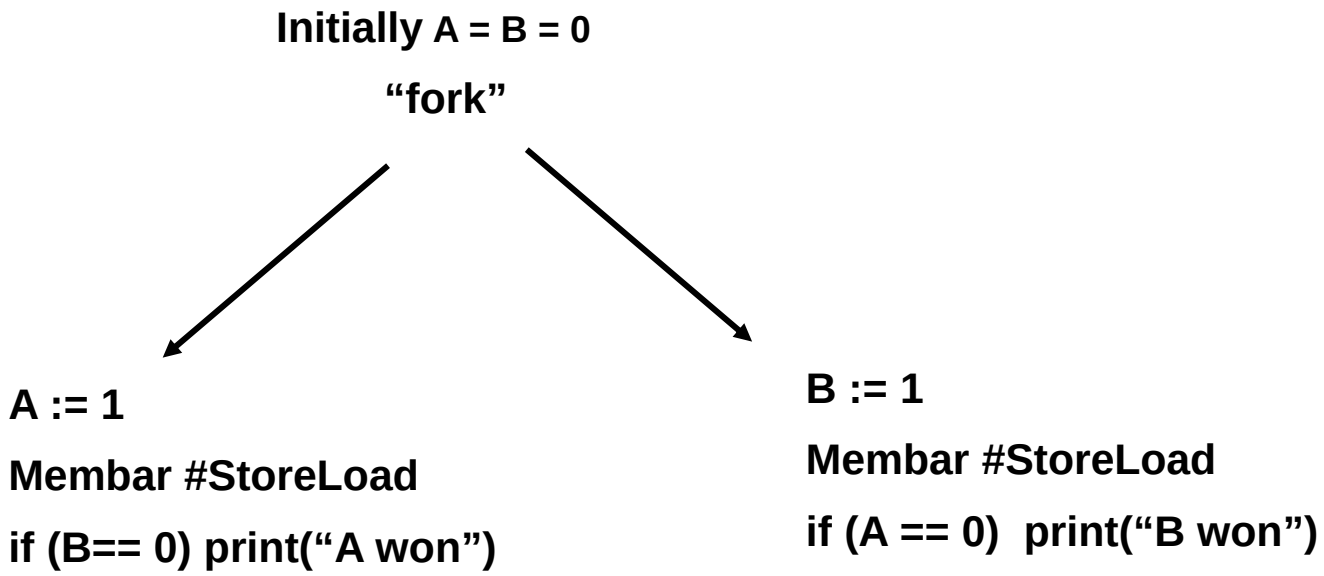
B := 1

if (A == 0) print("B won")

Q: Is it possible that both A and B wins?

Left: The read (i.e., test if B==0) can bypass the store (A:=1)
Right: The read (i.e., test if A==0) can bypass the store (B:=1)
➔both loads can be performed before any of the stores
➔yes, it is possible that both wins
➔➔ Dekker's algorithm breaks

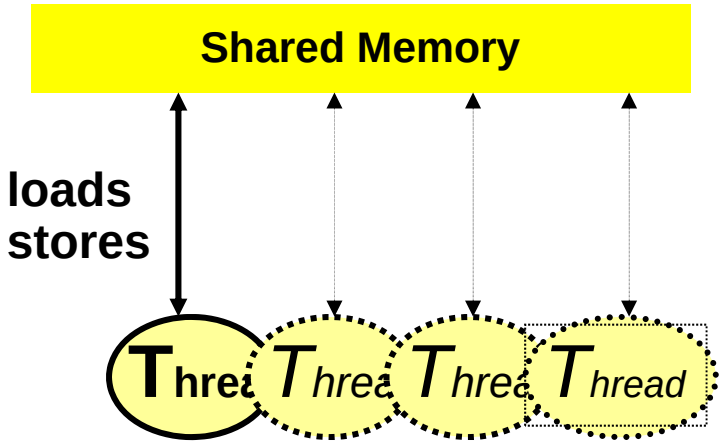
Dekker's Algorithm for TSO



Q: Is it possible that both A and B win?

Membar: The read is stared after all previous stores have been "globaly ordered"
➔ behaves like SC
➔ Dekker's algorithm works!

Weak/release Consistency (M. Dubois, K. Gharachorloo)



- **Most accesses are unordered**
- **Is rogrammer's intuition is not maintained?**
  - Store causality? No
  - Does Dekker work? No
- **Global order <u>only</u> established when the programmer explicitly inserts memory barrier instructions**

++ Better performance!!

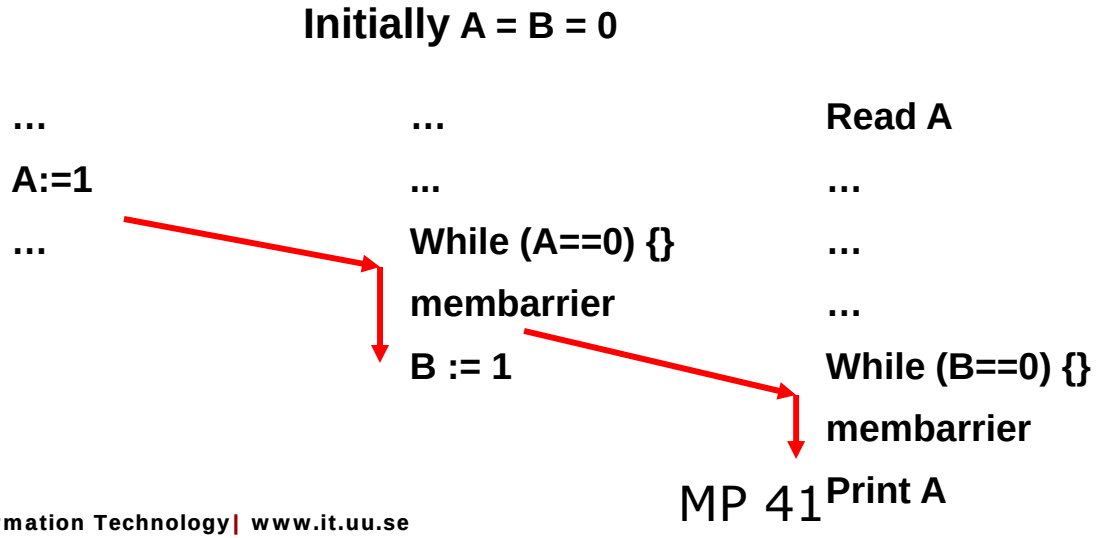
--- Interesting bugs!!

Weak/Release consistency

- New flag synchronization needed

```
A := data;                    while (flag != 1)  {};
membarrier;                   membarrier;
flag := 1;                    X := A;
```

- Dekker's: same as TSO
- Causal correctness provided for this code



Q: What value will get printed?
Answer: 1

UPPSALA UNIVERSITET

Learning more about memory models

Shared Memory Consistency Models: A Tutorial
by Sarita Adve, Kouroush Gharachorloo
in IEEE Computer 1996

RFM: Read the F****ng Manual of the system you are working on!
(Different microprocessors and systems supports different memory models.)

Issue to think about:
What code reordering may compilers really do?
Have to use ”volatile” declarations in C.

UPPSALA UNIVERSITET

X86's current memory model
Common view in academia: TSO

If you ask Intel:

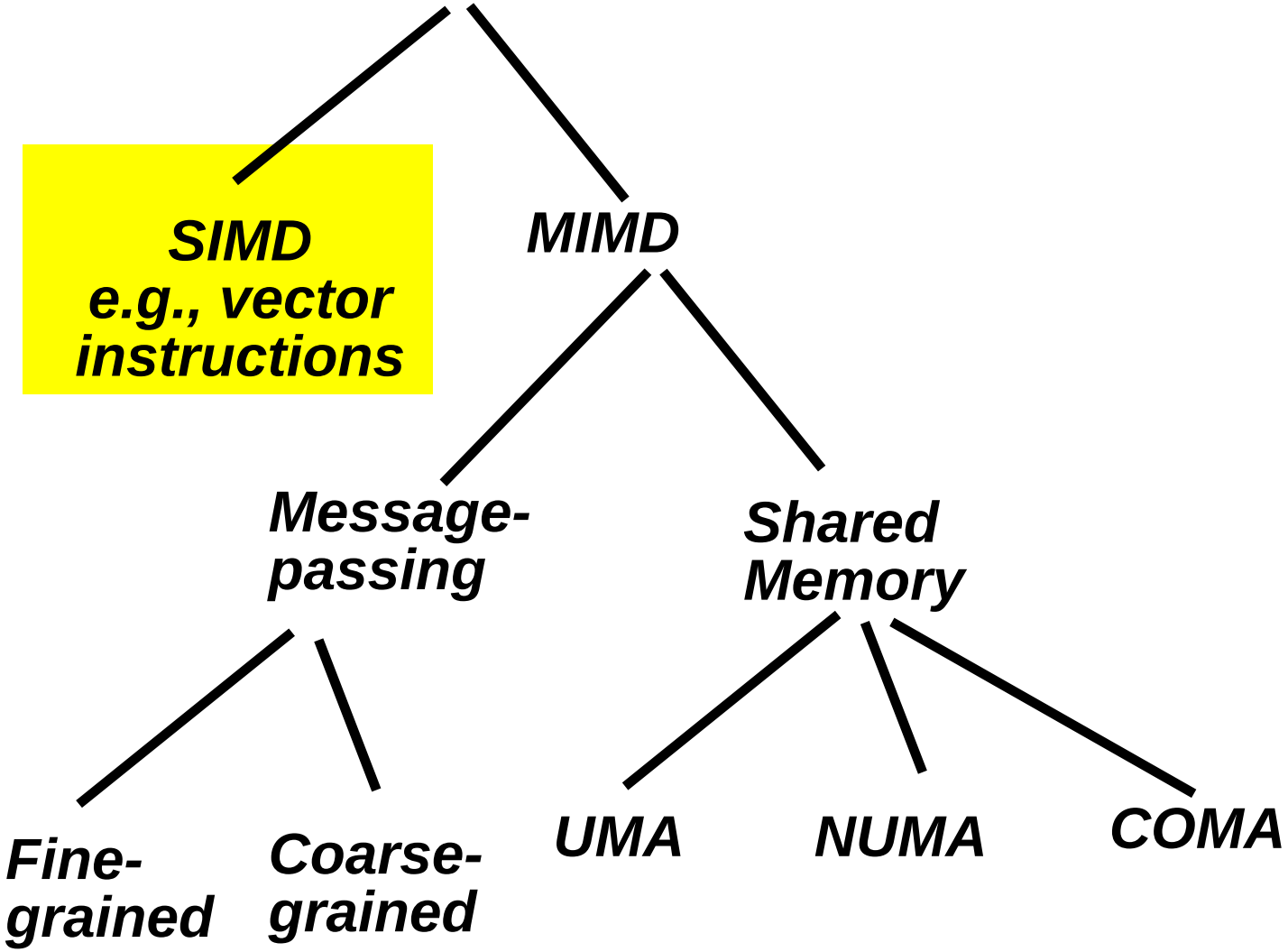
- Processor consistency with causual correctness for non-atomic memory ops
- TSO for atomic memory ops

- Video presentation:
  http://www.youtube.com/watch?v=WUfvvFD5tAA&hl=sv

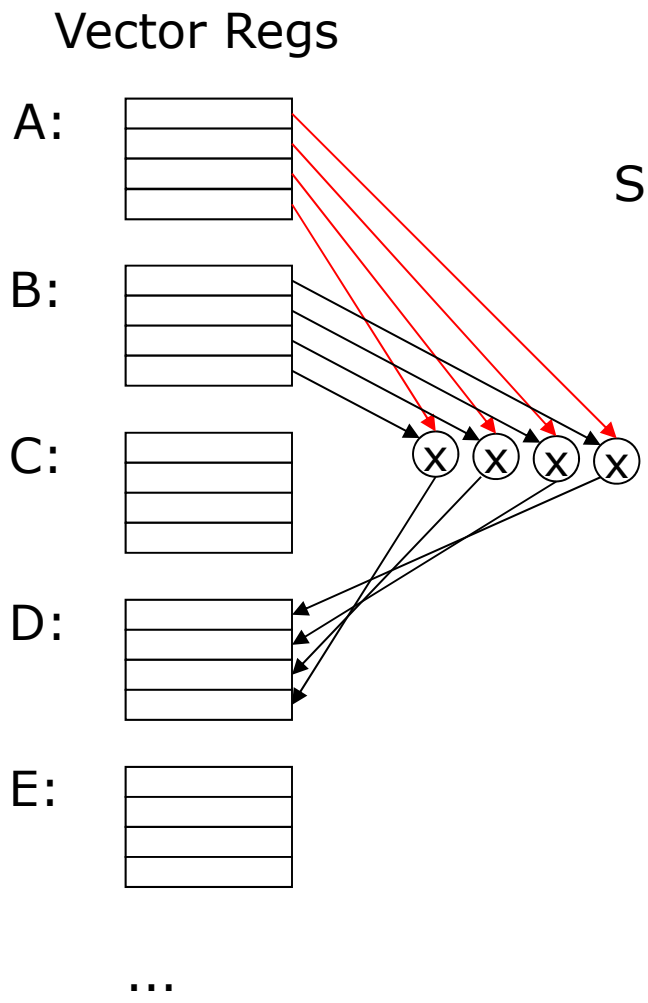
- See section 8.2 in this manual:
  http://developer.intel.com/Assets/PDF/manual/253668.pdf

UPPSALA UNIVERSITET

A few words about SIMD



Examples of vector instructions



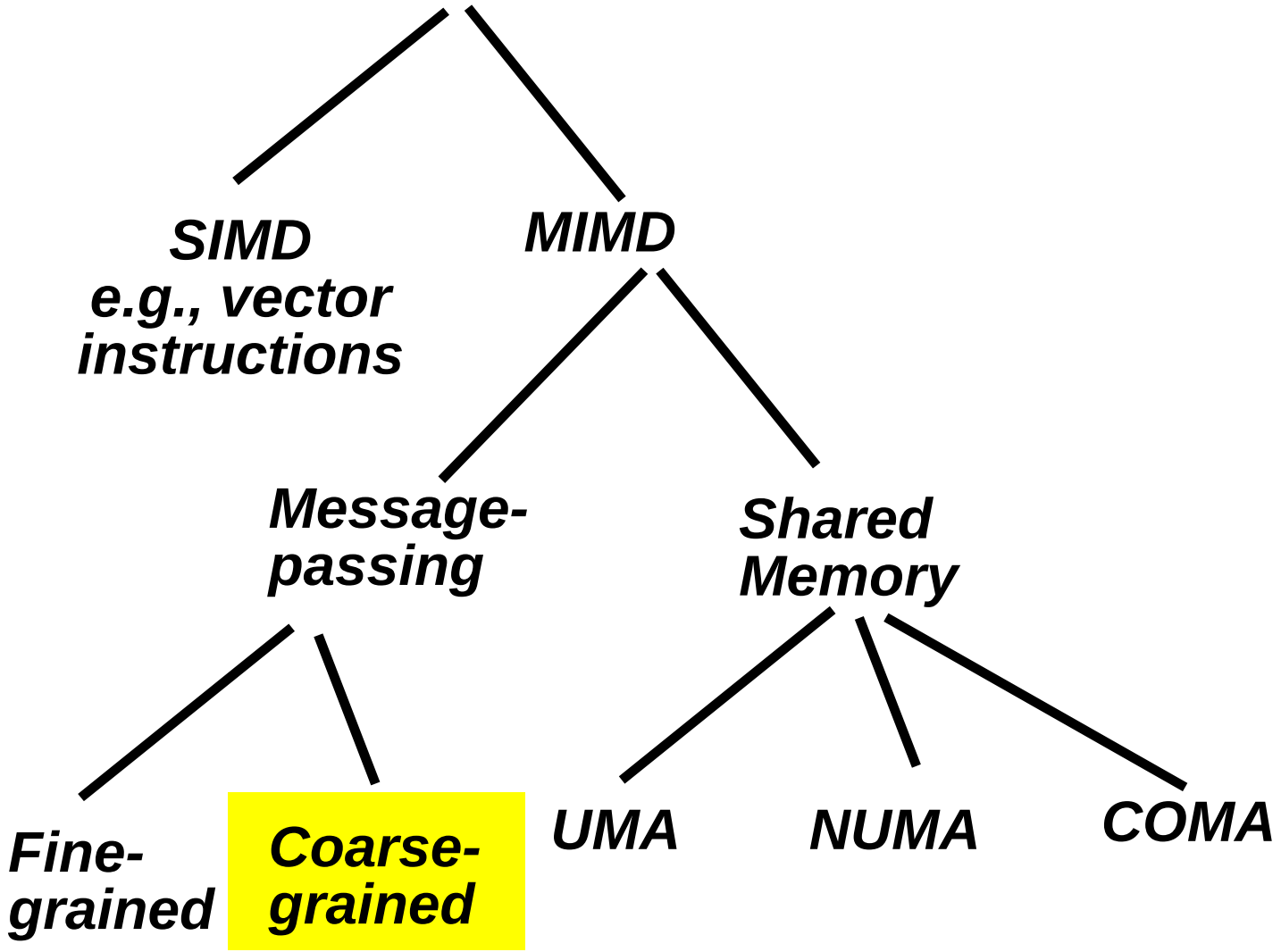
SSE_MUL D, B, A

UPPSALA UNIVERSITET

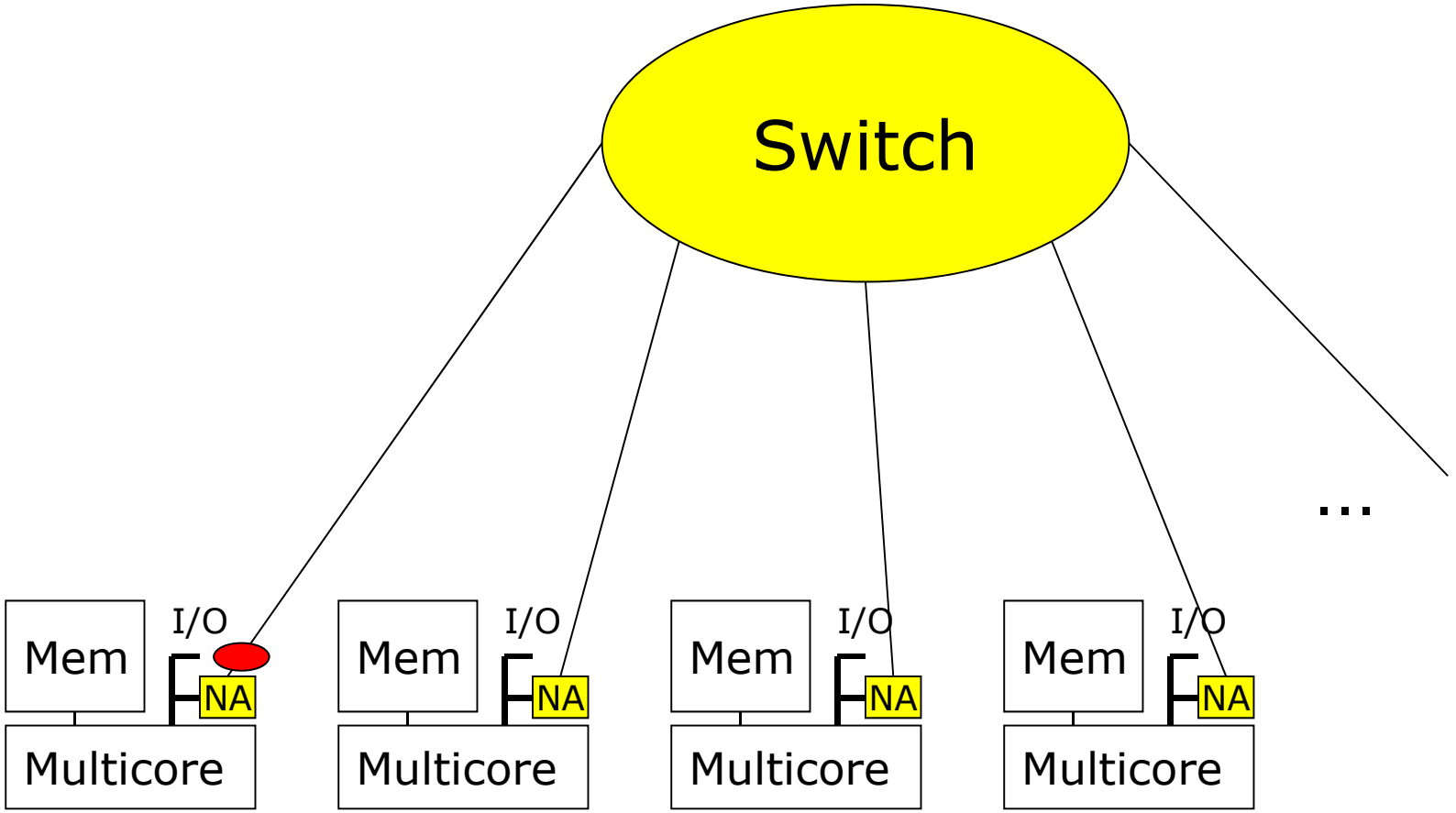
x86 Vector instructions

- MMX: 64 bit vectors (e.g., two 32bit ops)
- SSE*n*: 128 bit vectors(e.g., four 32 bit ops)
- AVX: 256 bit vectors(e.g., eight 32 bit ops) (in Sandy Bridge, Q1 2011)
- Intel MIC: ”16-way vectors” late 2012??

A few words on Message-passing



A modern "supercomputer"




```
X = vec[i];
MPI_send(X, to_dest);
...
```

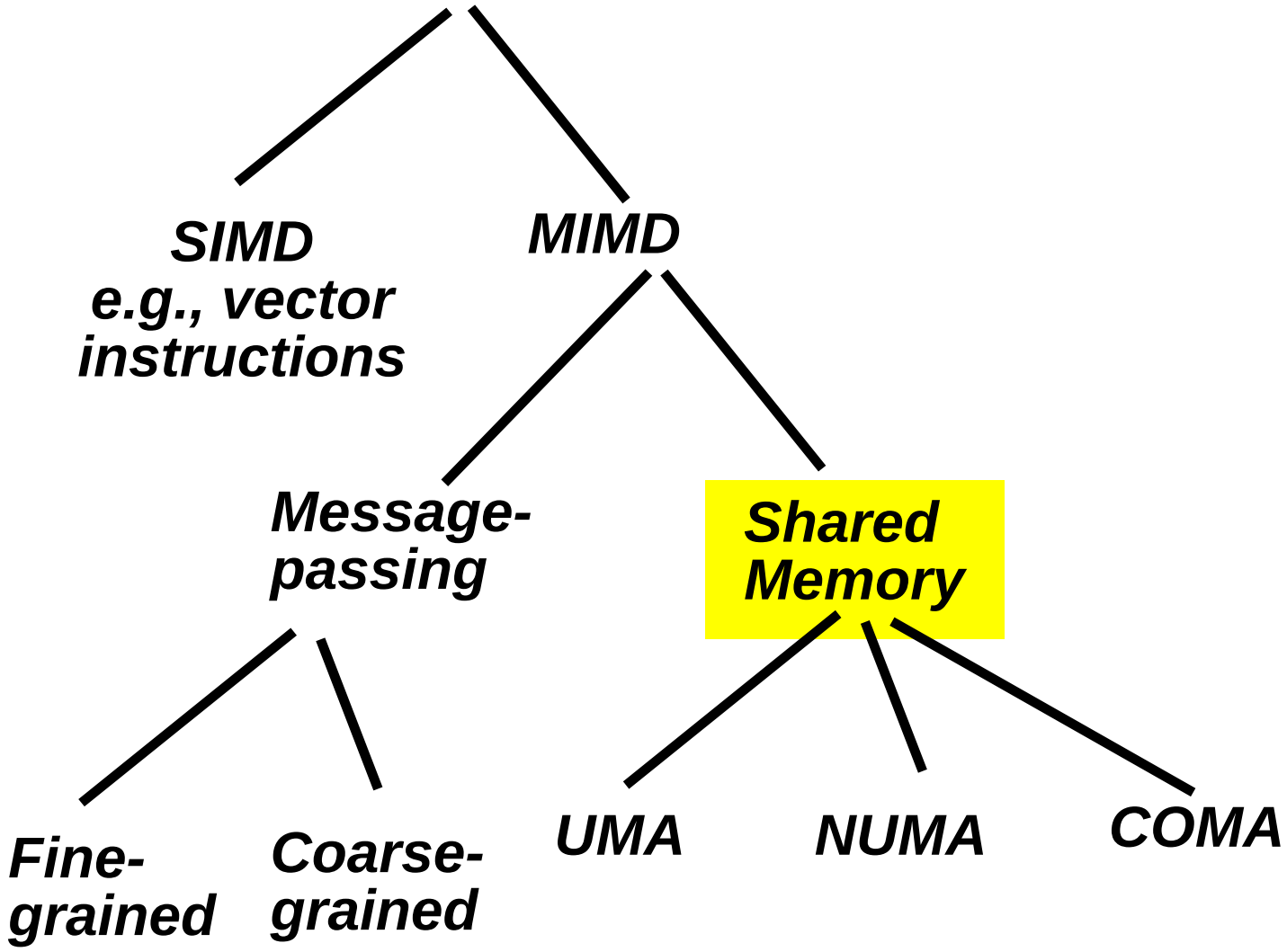
```
...
MPI_receive(Y, from_source;
print (Y);
```

UPPSALA UNIVERSITET

MPI inside a multicore?

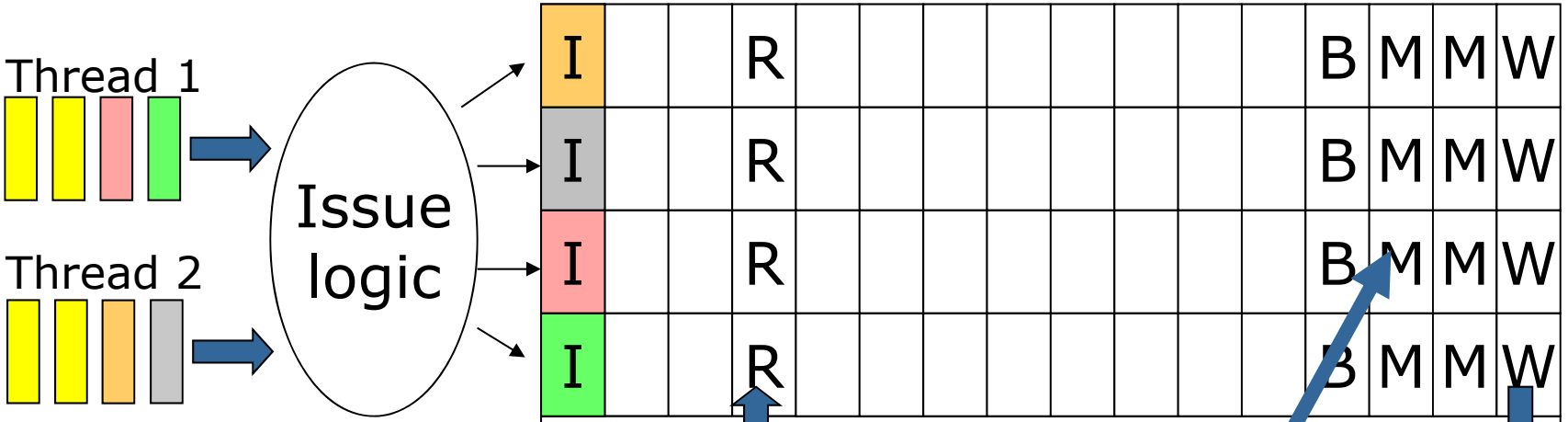
- MPI can be implemented on top of coherent shared memory
- Coherent shard memory cannot easily be implemented on top of MPI
- Many options for parallelism within a ”node”:
  - OpenMP
  - MPI
  - Posix threads
  - ...

A few words about simultaneously multithreading (SMT) or "Hyper-threading"



Several threads sharing a pipeline

TLP helps finding more independent instructions



Many different policies for how to interleave instructions from different threads

Each thread has its own register file

1-10 cycles

$

The cache hierarchy is typically shared between the threads

150cycles

Mem

1GB