Multicore Computing

Geilo Winter School in eScience, January 2008

Sverker Holmgren

Henrik Löf

Uppsala University

Sweden

UPPSALA UNIVERSITET

Outline of January 23

- Session 1, 9.00 – 10.30
  - What are multicore processors and why should CSE researchers bother? (S.H)
- Session 2, 15.00 – 16.30
  - Programming multicore processors (H.L)
- Session 3, 17.00 – 18.30
  - Multicore applications (S.H., H.L.)

UPPSALA UNIVERSITET

Who are Sverker and Henrik ?

Sverker:

- PhD in Numerical Analysis 1993 (Uppsala)
- Head of the research program in Computational Science at Uppsala University
- Director for SNIC (c.f. Notur)

Henrik:

- PhD in Scientific Computing 2006 (Uppsala)
- Joint PhD project Num./Comp.Arch.
- PostDoc at Stanford (Dept. of Energy Resources Engineering)
- Now moving to Swedish industry...

UPPSALA UNIVERSITET

Outline of Session 1

- Conclusions
- Why? (Including a Crash-Course in Computer Architecture)
- Multicore Systems, Now and in the Future
- Impact on Performance
- Impact on CSE Software in General
- Impact on Algorithms
- Conclusions

Many of the slides originate from Erik Hagersten!

UPPSALA UNIVERSITET

Conclusions

- The Multicore Era is here
- There will be multicore processors with different architectures
- The cost of parallelism is dropping dramatically, but
  - Cache/thread is dropping
  - Memory bandwidth/thread is dropping
  - => The performance is determined by the flow of data
- The introduction of multicore processors will have profound impact on CSE software

  "For high-performance computing (HPC) applications, multicore processors introduces an additional layer of complexity, which will force users to go through a phase change in how they address parallelism or risk being left behind."

UPPSALA
UNIVERSITET

Why?

UPPSALA UNIVERSITET

Why multiple cores/threads (even in your laptop)?

- Instruction level parallelism is running out
- Wire delay is hurting
- Power dissipation is hurting
- The memory latency/bandwidth bottleneck is becoming even worse

UPPSALA UNIVERSITET

Current Doubling/Halving Times

Dynamic RAM Memory (bits per dollar)	1.5 years
Average Transistor Price	1.6 years
Microprocessor Cost per Transistor Cycle	1.1 years
Total Bits Shipped	1.1 years
Processor Performance in MIPS	1.8 years
Transistors in Intel Microprocessors	2.0 years
Microprocessor Clock Speed	2.7 years

UPPSALA
UNIVERSITET

A Crash-Course in Computer Architecture (with a focus on performance)

UPPSALA UNIVERSITET

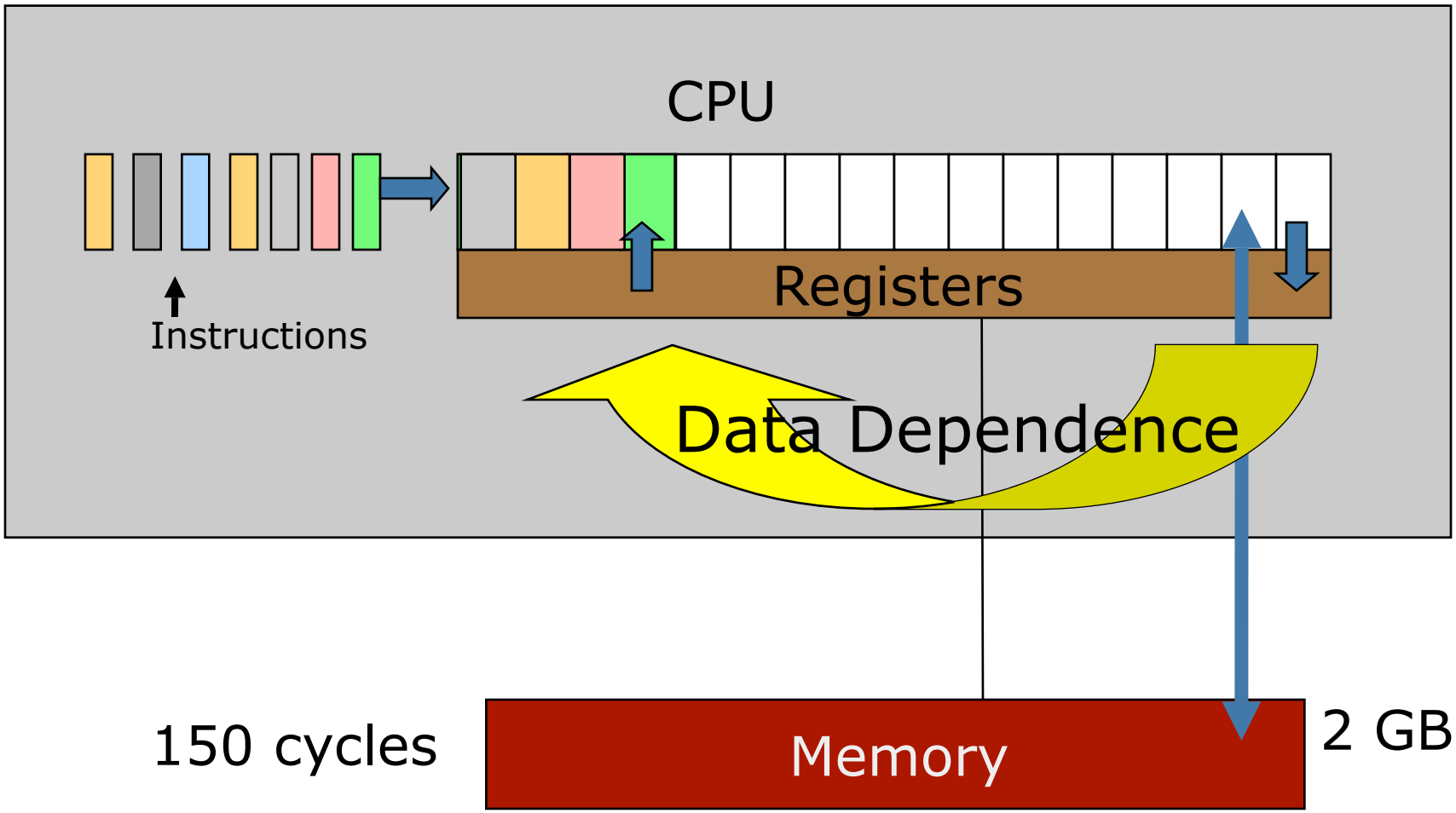
Performance so far?

Create and explore:

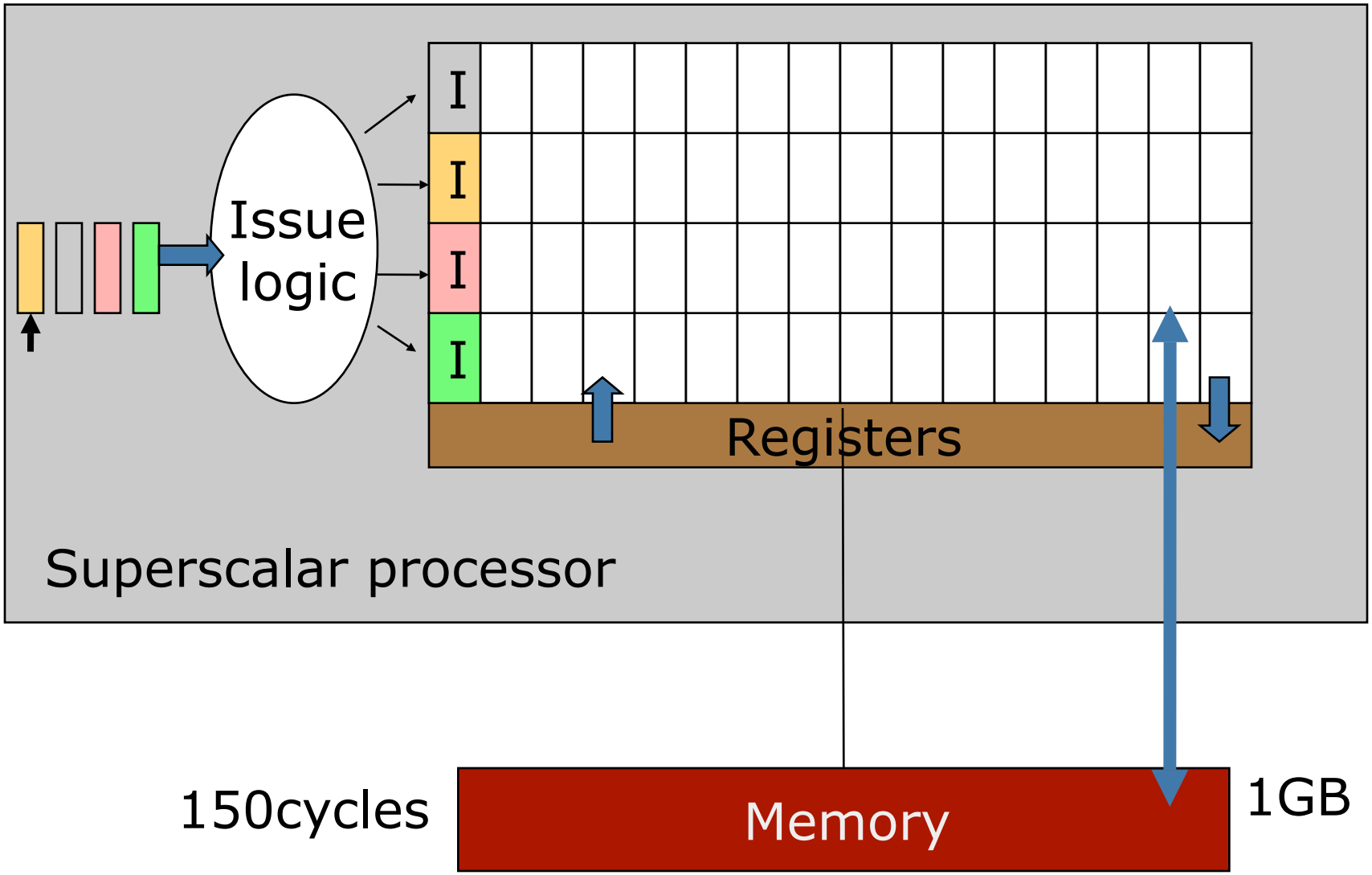
1. Parallelism
   - Instruction level parallelism (ILP)
   - Possibly parallelism at the program/ algorithm level (“Parallel computing”)

2. Locality of data
   - Caches
   - Temporal locality
   - Cache blocks/lines: Spatial locality

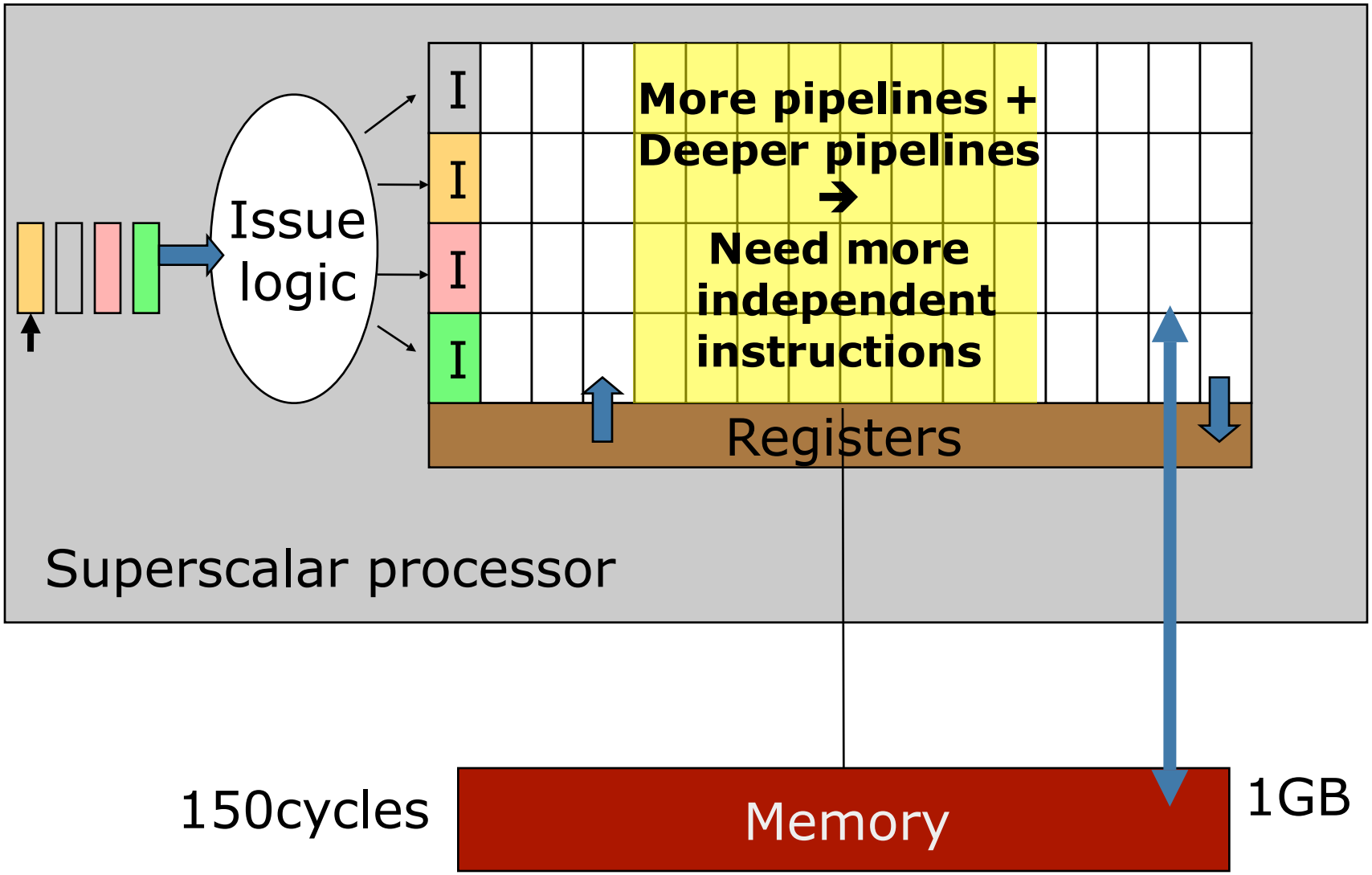
Old Trend 1: Deeper Pipelines (= Exploring ILP)



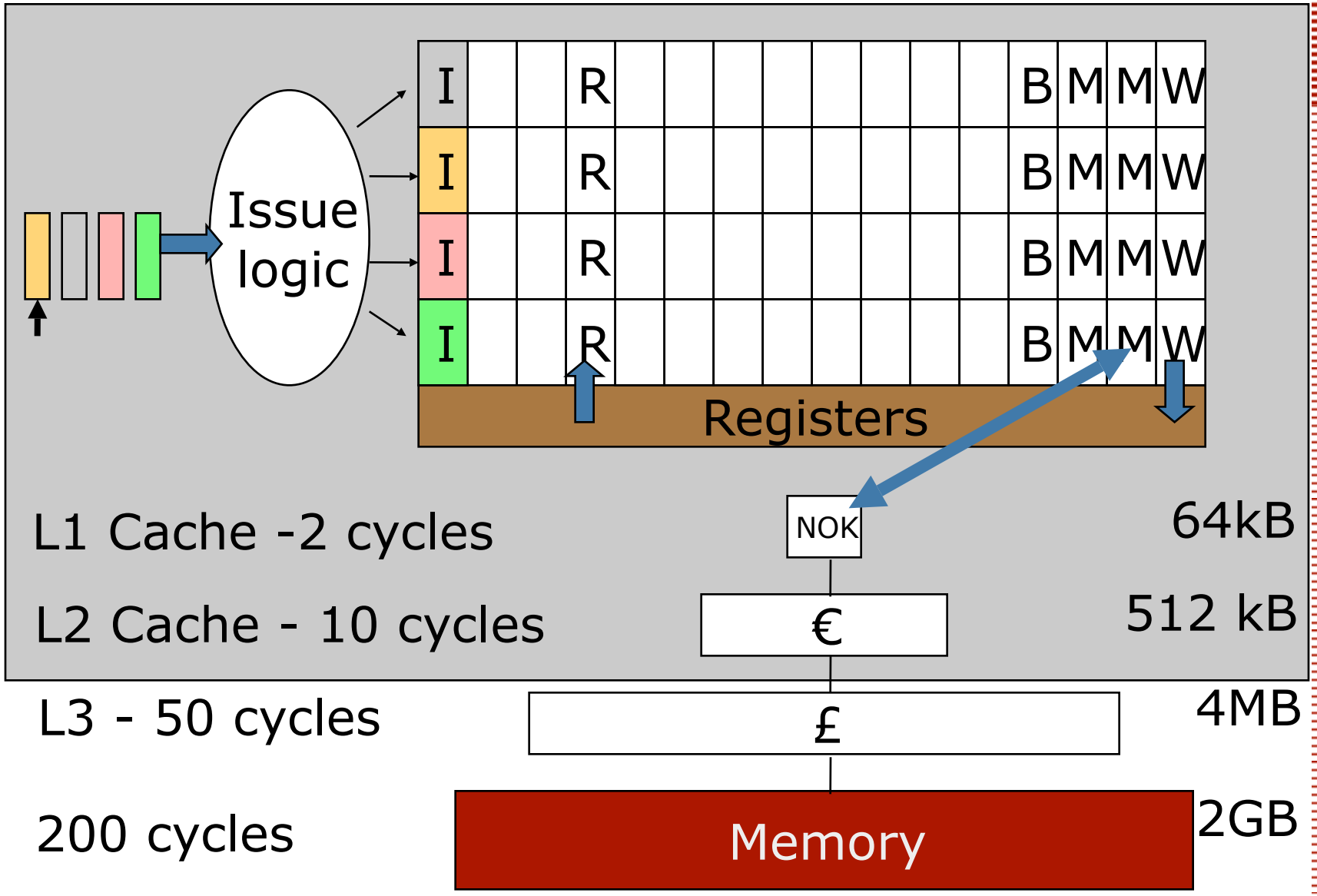
Old Trend 2: Wider Pipelines (= Exploring More ILP)



Old Trend 2: Wider Pipelines (= Exploring More ILP)

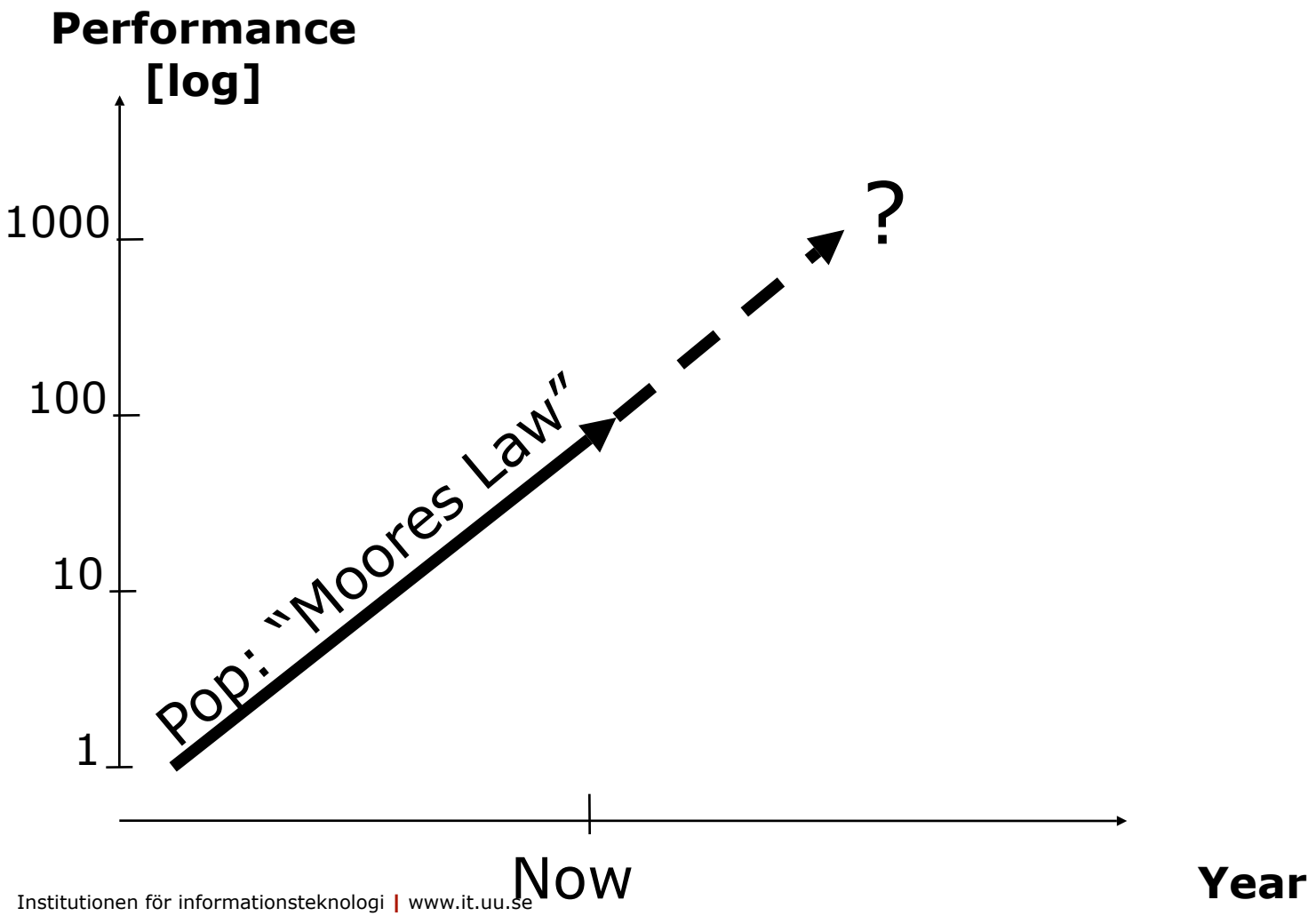


Old Trend3: Deeper Memory Hierarchy (= Exploring Locality of Data)

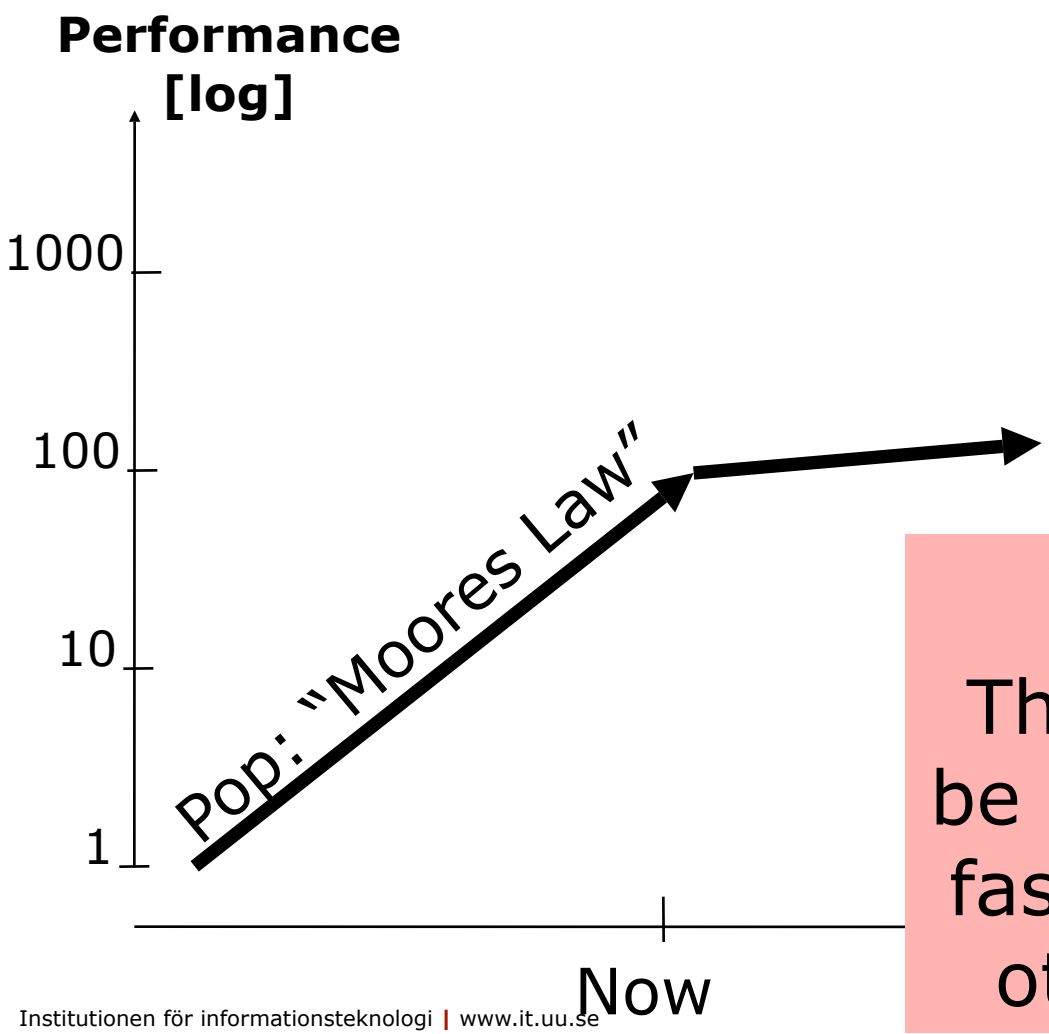


UPPSALA UNIVERSITET

Are we hitting the wall now?



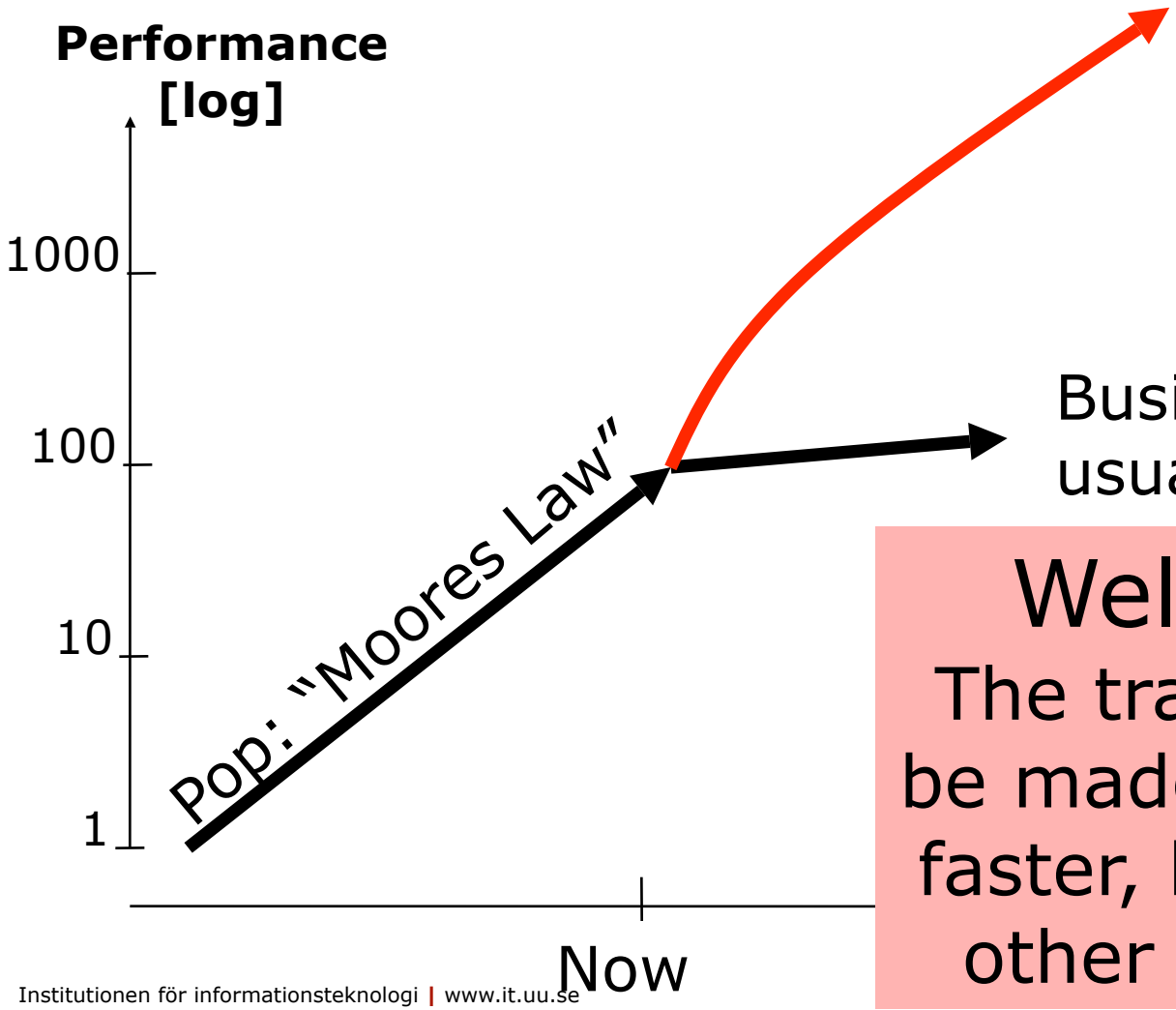
Are we hitting the wall now?



Well uhm …

The transistors can be made smaller and faster, but there are other problems ☹

Are we hitting the wall now?



Well uhm …

The transistors can be made smaller and faster, but there are other problems ☹

UPPSALA UNIVERSITET

Classical microprocessors: Whatever it takes to run one program fast.

Exploring ILP (instruction-level parallelism):

- Faster clocks ➔ Deep pipelines
- Superscalar Pipelines
- Branch Prediction
- Out-of-Order Execution
- Trace Cache
- Speculation
- Predicate Execution
- Advanced Load Address Table
- Return Address Stack
- ….

UPPSALA UNIVERSITET

Classical microprocessors: W…er it takes to run one progra…

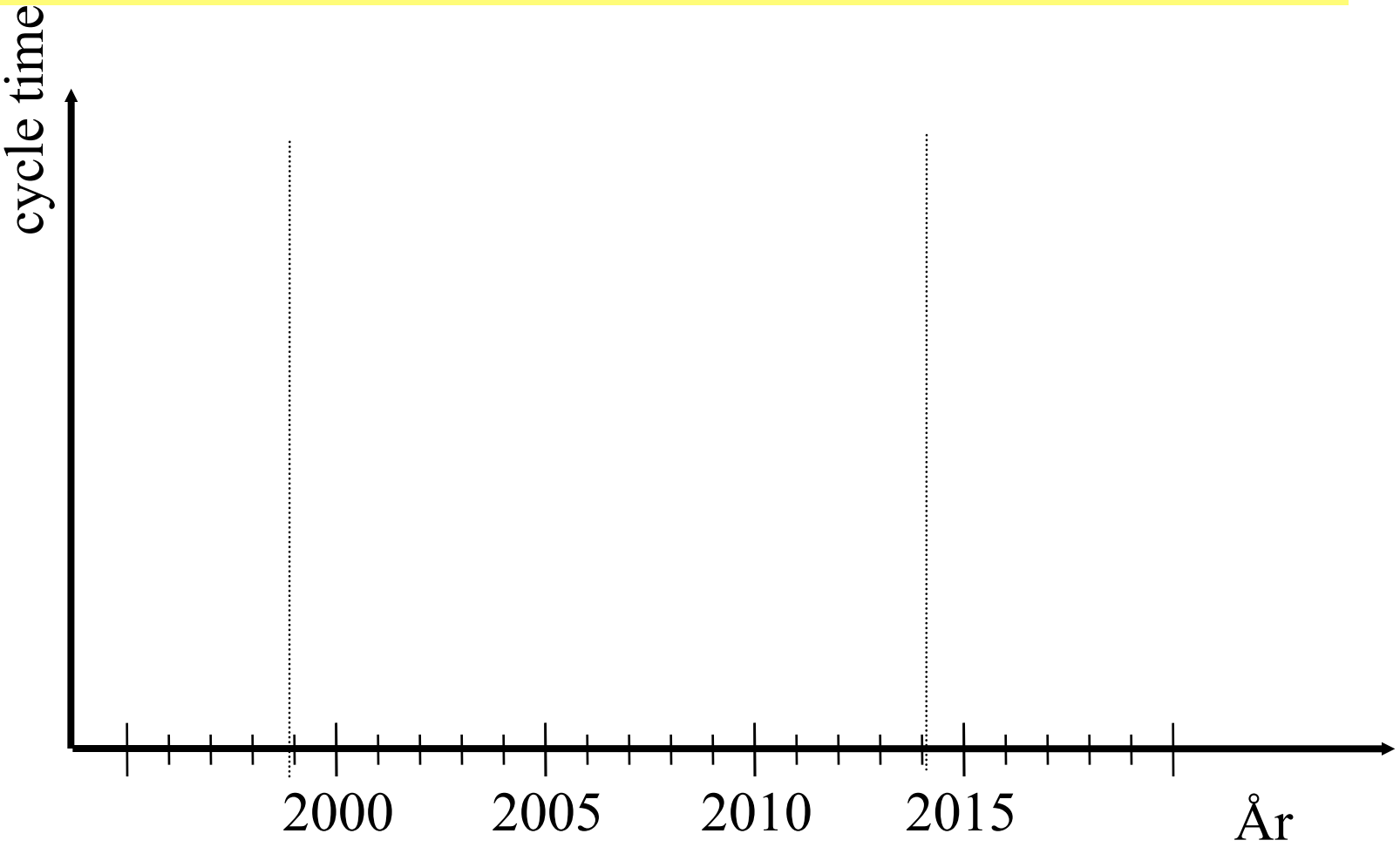
Exploring ILP (instruction-level…

- Faster clocks ➔ Deep pi…
- Superscalar Pipelines
- Branch Predictio…
- Out-of-Order …
- Trace Cach…
- Specul…
- Pre…
- …ddress Table
- …ss Stack
- …

Bad News #1:
We have already explored most ILP
(instruction-level parallelism)

UPPSALA UNIVERSITET

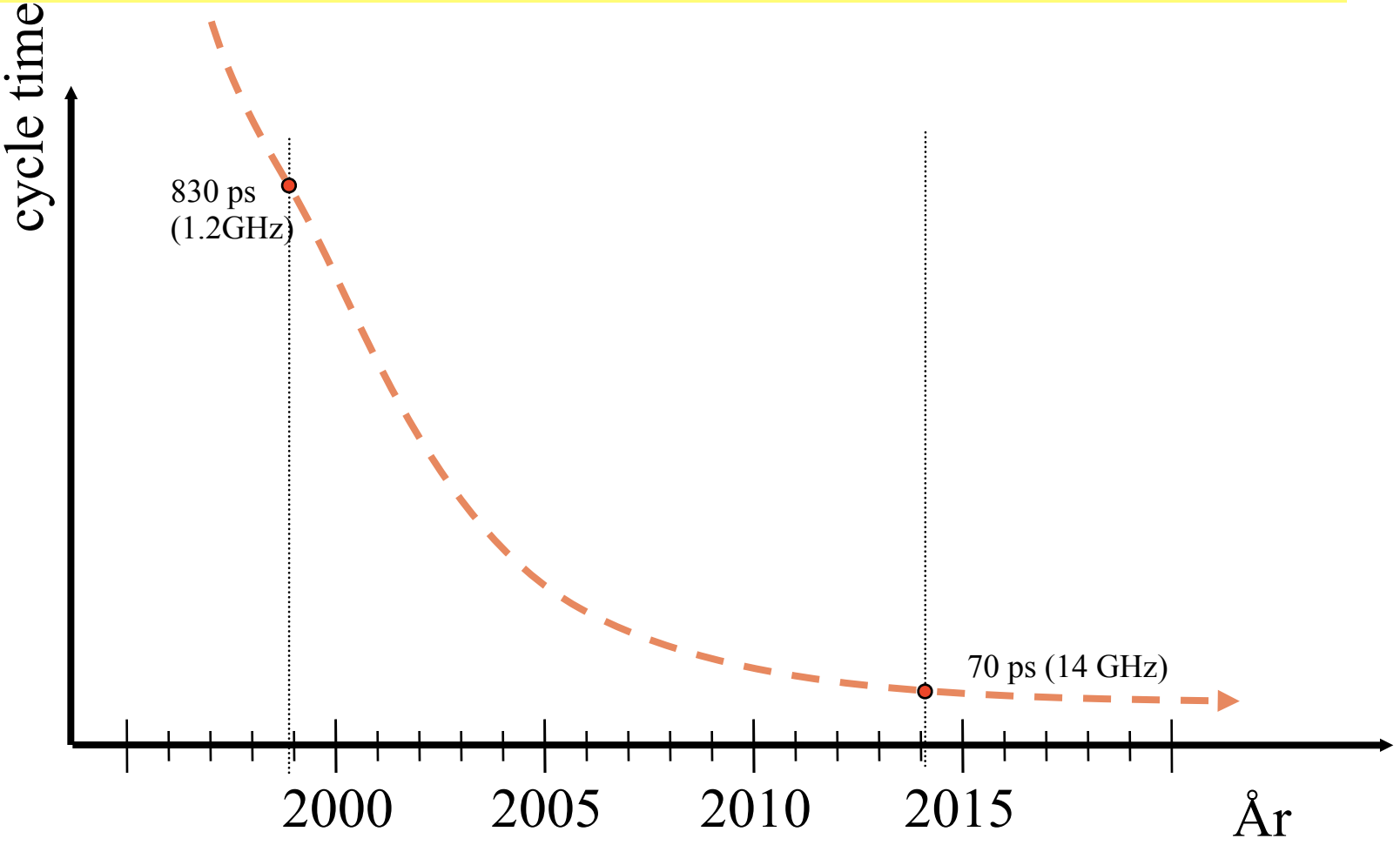
Bad News #2
Looong wire delay →slow CPUs



Quantitative data and trends according to V. Agarwal et al., ISCA 2000
***Based on SIA* (Semiconductor Industry Association) prediction, 1999**

UPPSALA UNIVERSITET

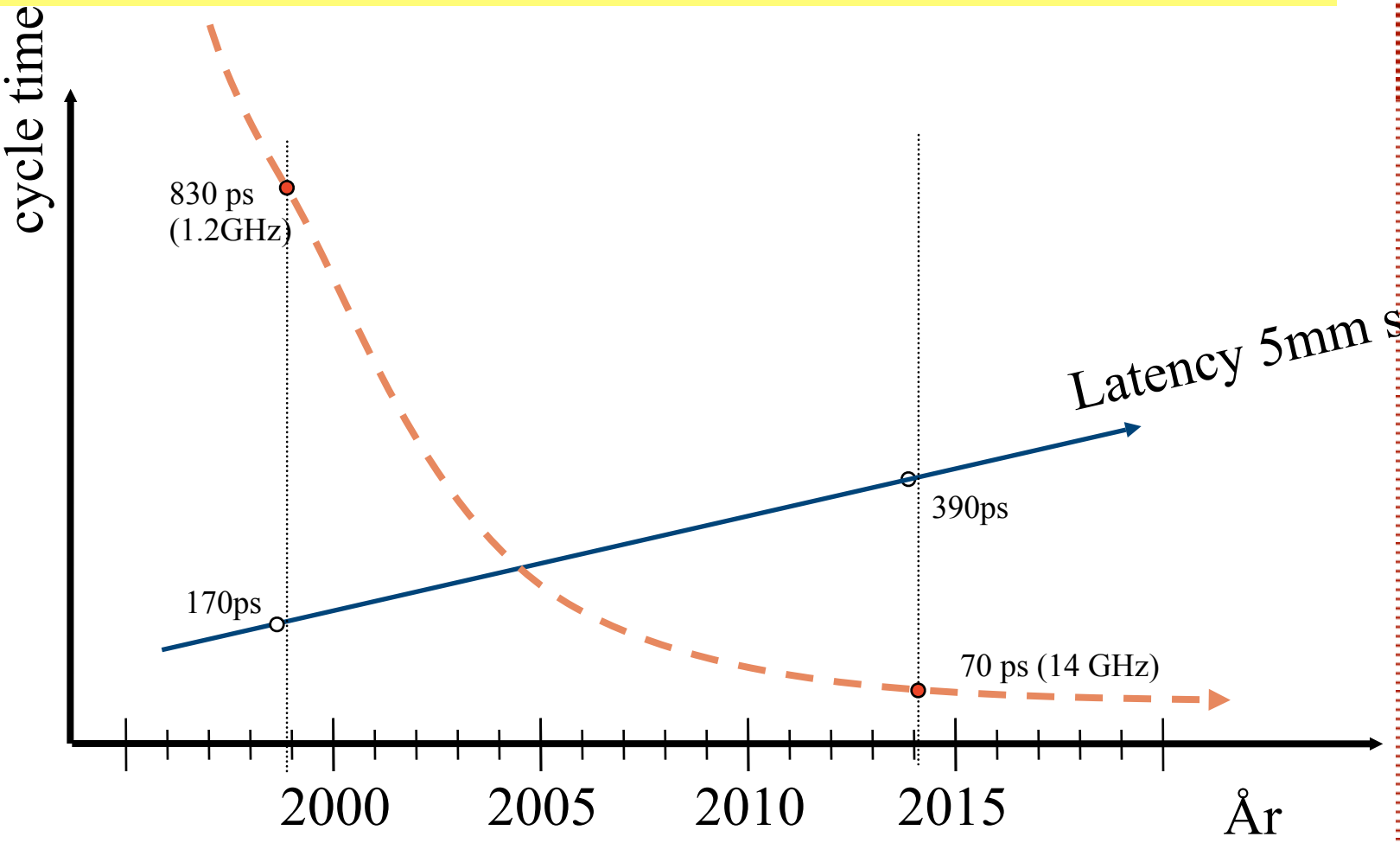
Bad News #2
Looong wire delay →slow CPUs



Quantitative data and trends according to V. Agarwal et al., ISCA 2000
***Based on SIA* (Semiconductor Industry Association) prediction, 1999**

UPPSALA UNIVERSITET

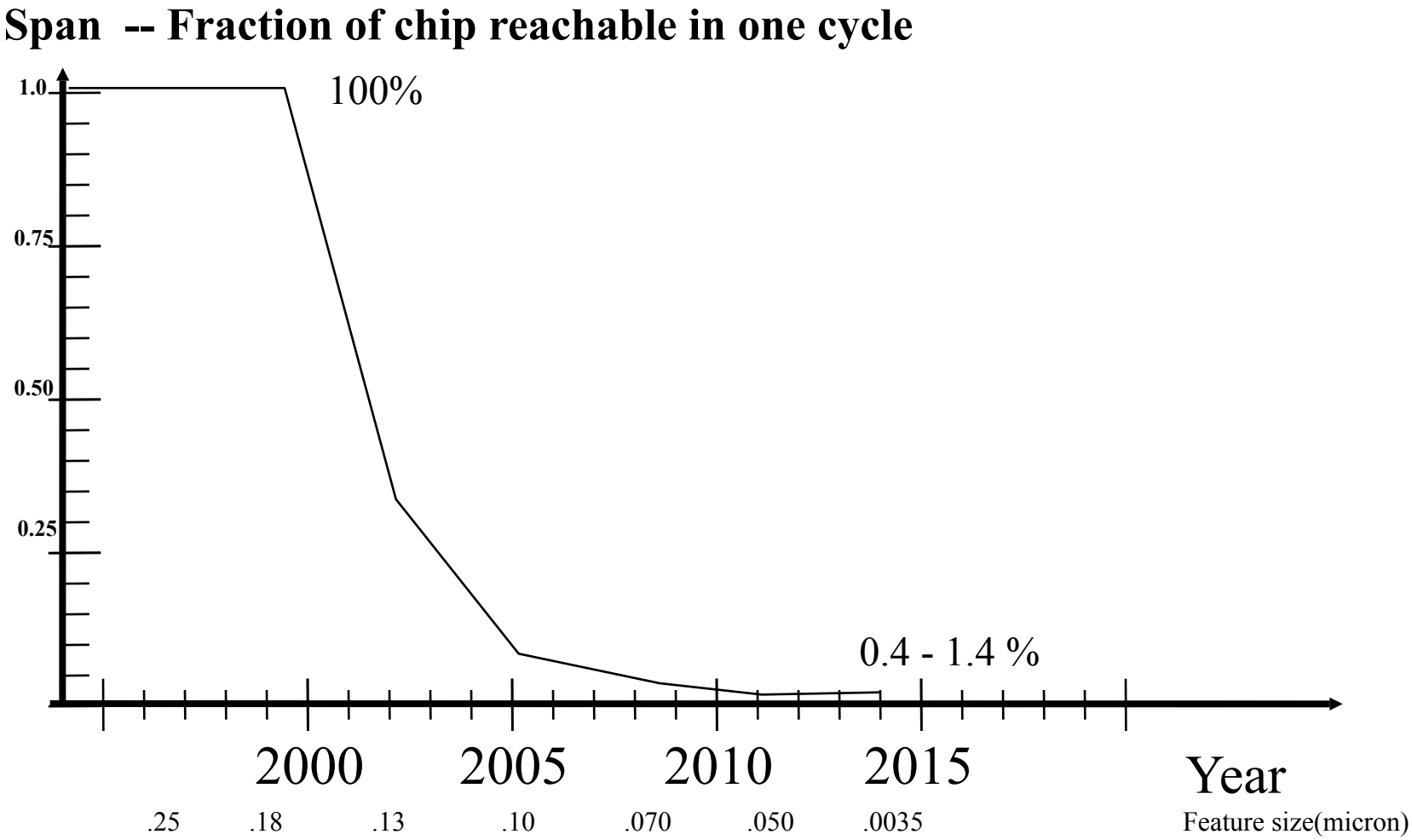
Bad News #2
Looong wire delay →slow CPUs



Quantitative data and trends according to V. Agarwal et al., ISCA 2000
***Based on SIA* (Semiconductor Industry Association) prediction, 1999**

UPPSALA UNIVERSITET

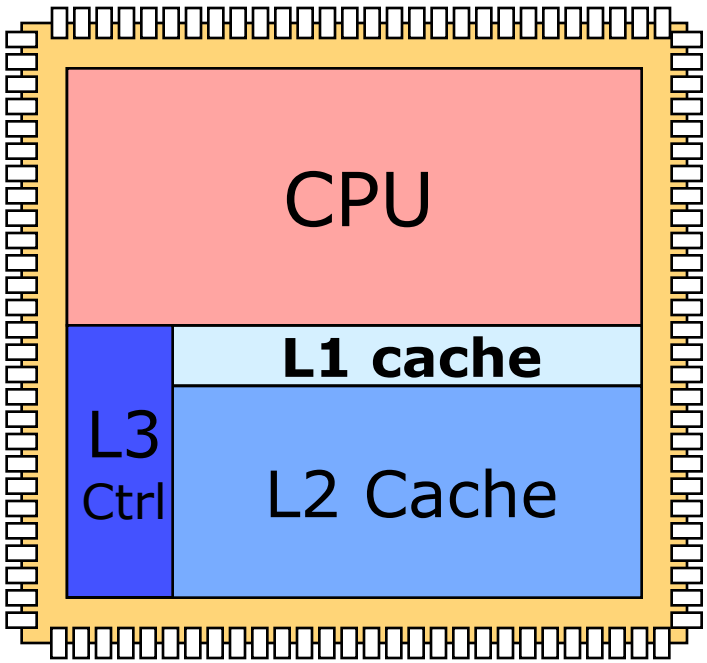
Bad News #2
Looong wire delay →slow CPUs

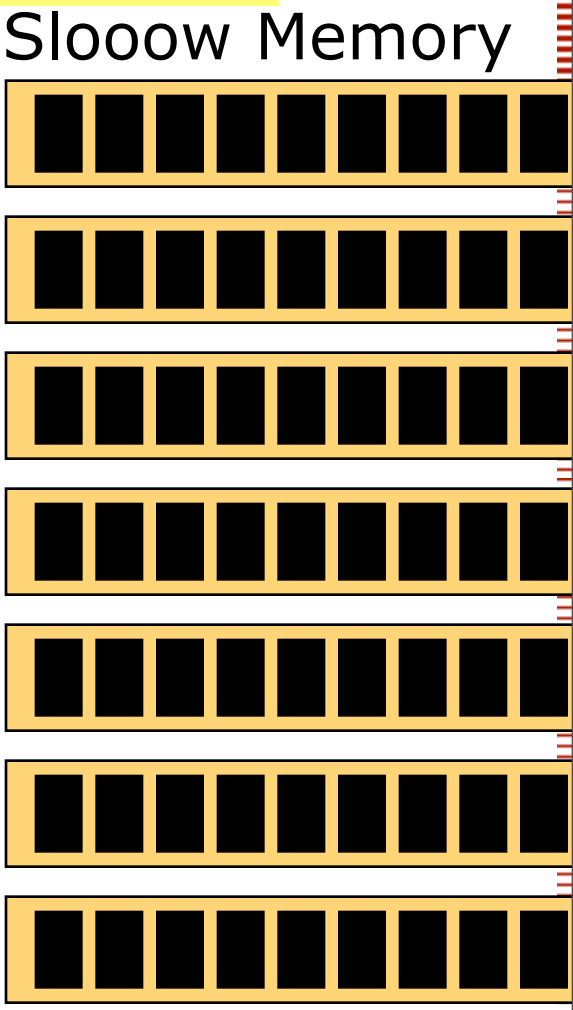


UPPSALA UNIVERSITET

Bad News #3:

Memory latency/bandwidth is the bottleneck...





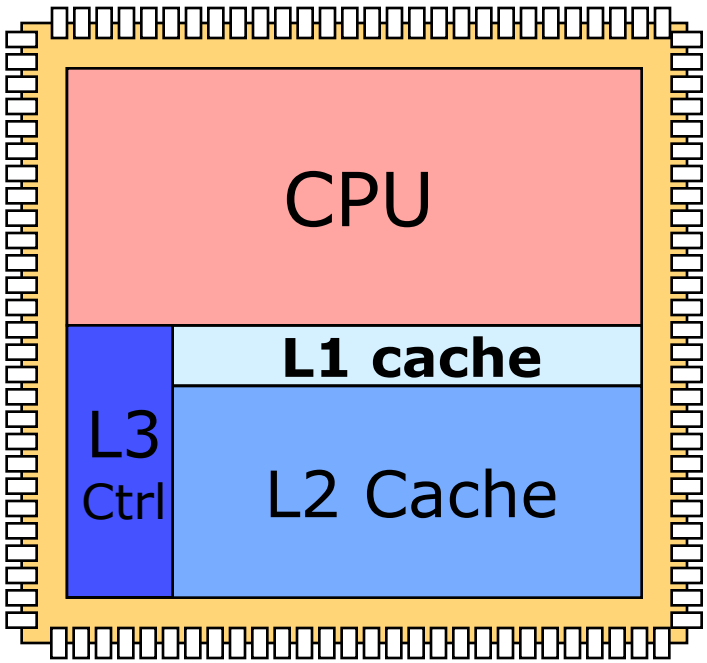
A = B + C:

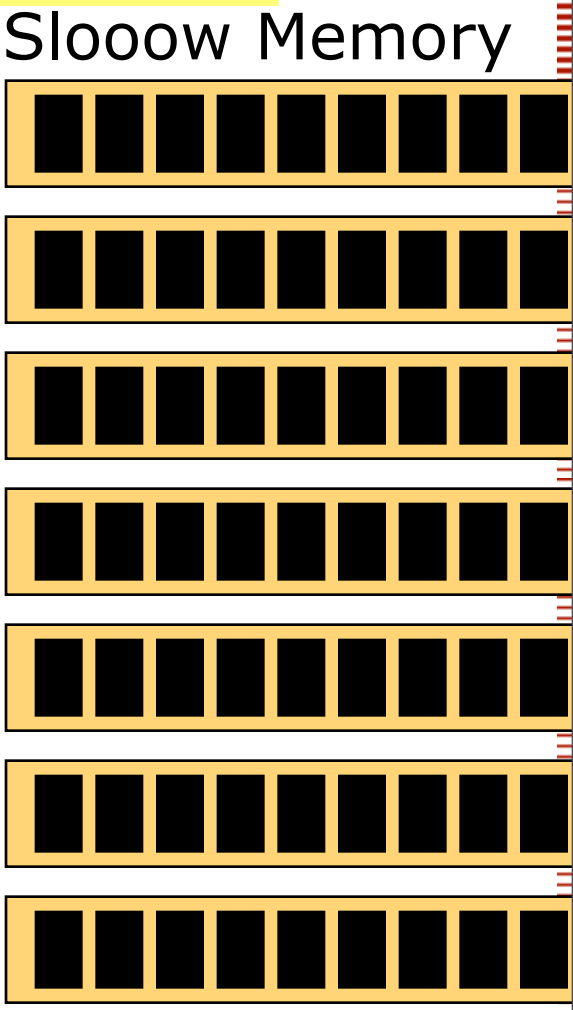
Read B
Read C
Add B & C
WriteA

UPPSALA UNIVERSITET

Bad News #3:

Memory latency/bandwidth is the bottleneck...





A = B + C:	**Latency**
Read B	**0.3 - 100 ns**
Read C	**0.3 - 100 ns**
Add B & C	**0.3 ns**
WriteA	**0.3 - 100 ns**

UPPSALA UNIVERSITET

Bad News #4:
Power is the limit

- Power consumption is the bottleneck
  - Cooling servers is hard
  - Battery lifetime for mobile computers
  - Energy is money

- Dynamic effect is proportional to
  - ~ Frequency
  - ~ Voltage^2

UPPSALA UNIVERSITET

Solving all the problems (?): Exploring thread parallelism

#1: Running out of ILP

➔ feed one CPU with instr. from many threads

#2: Wire delay is starting to hurt

➔ Multiple small CPUs with private caches

#3: Memory is the bottleneck

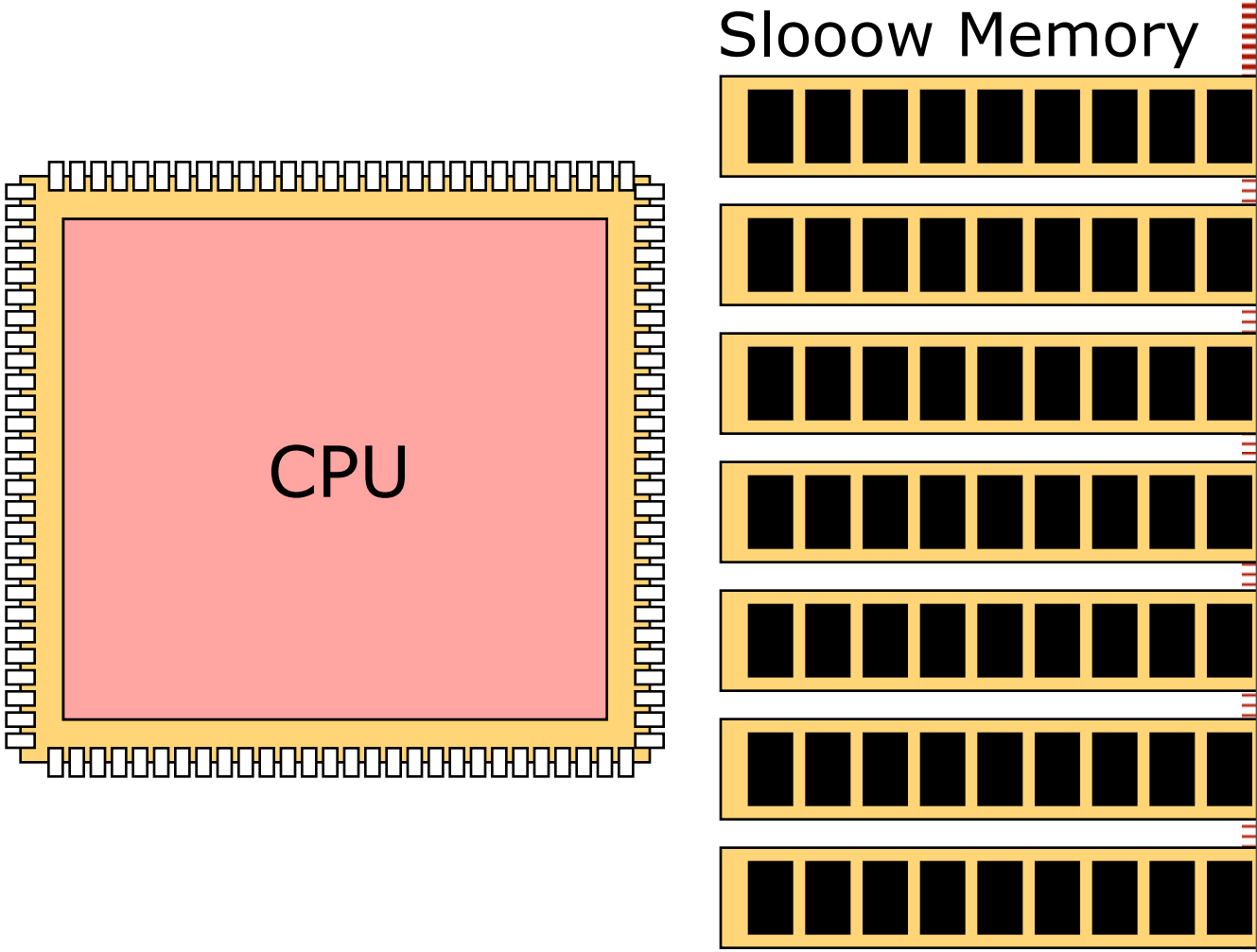
➔ memory accesses from many threads

#4: Power is the limit

➔ Lower the frequency ➔ lower voltage

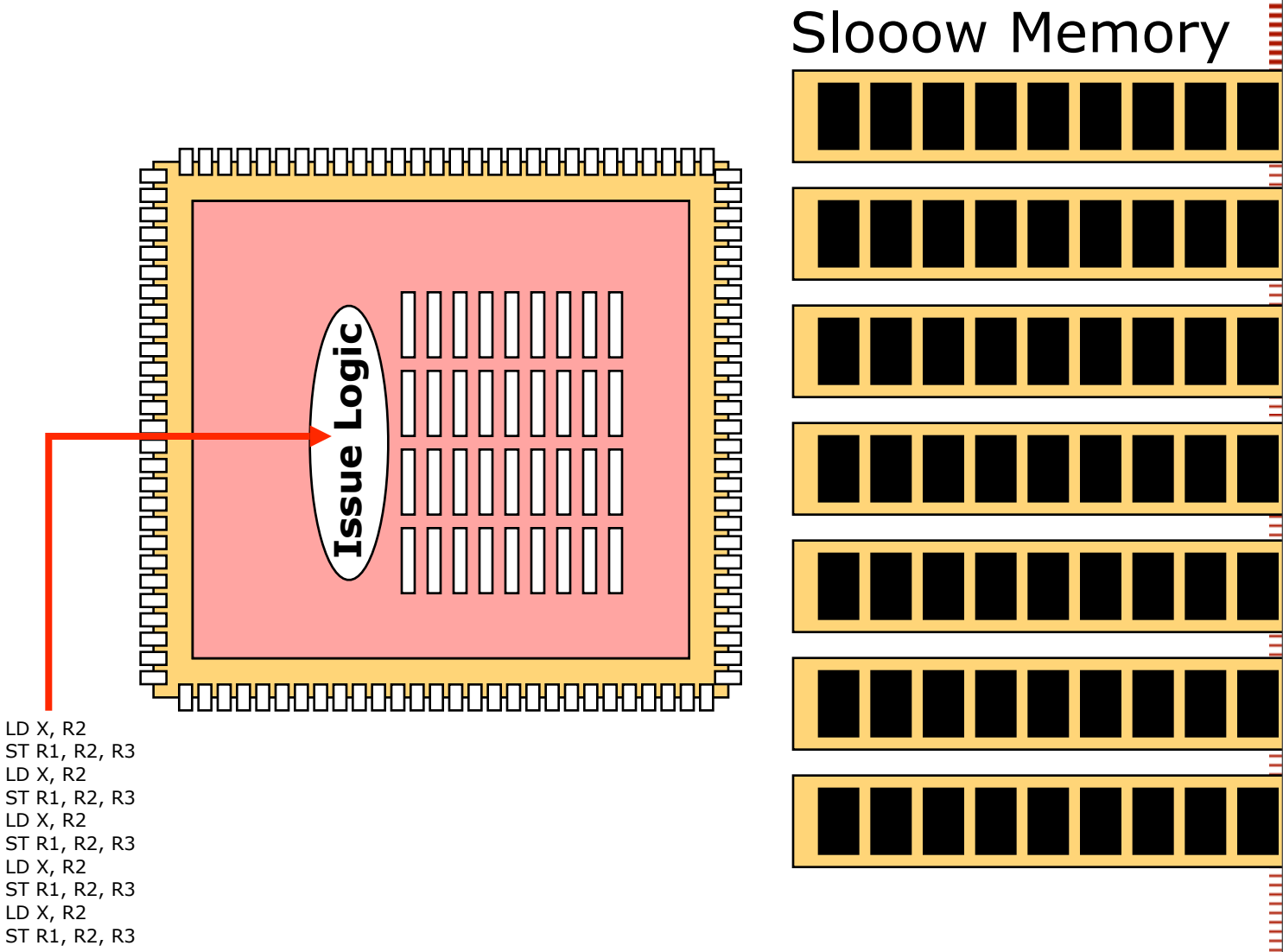
Bad News #1: Not enough ILP 1(2)

➔ **feed one CPU with instr. from many threads**



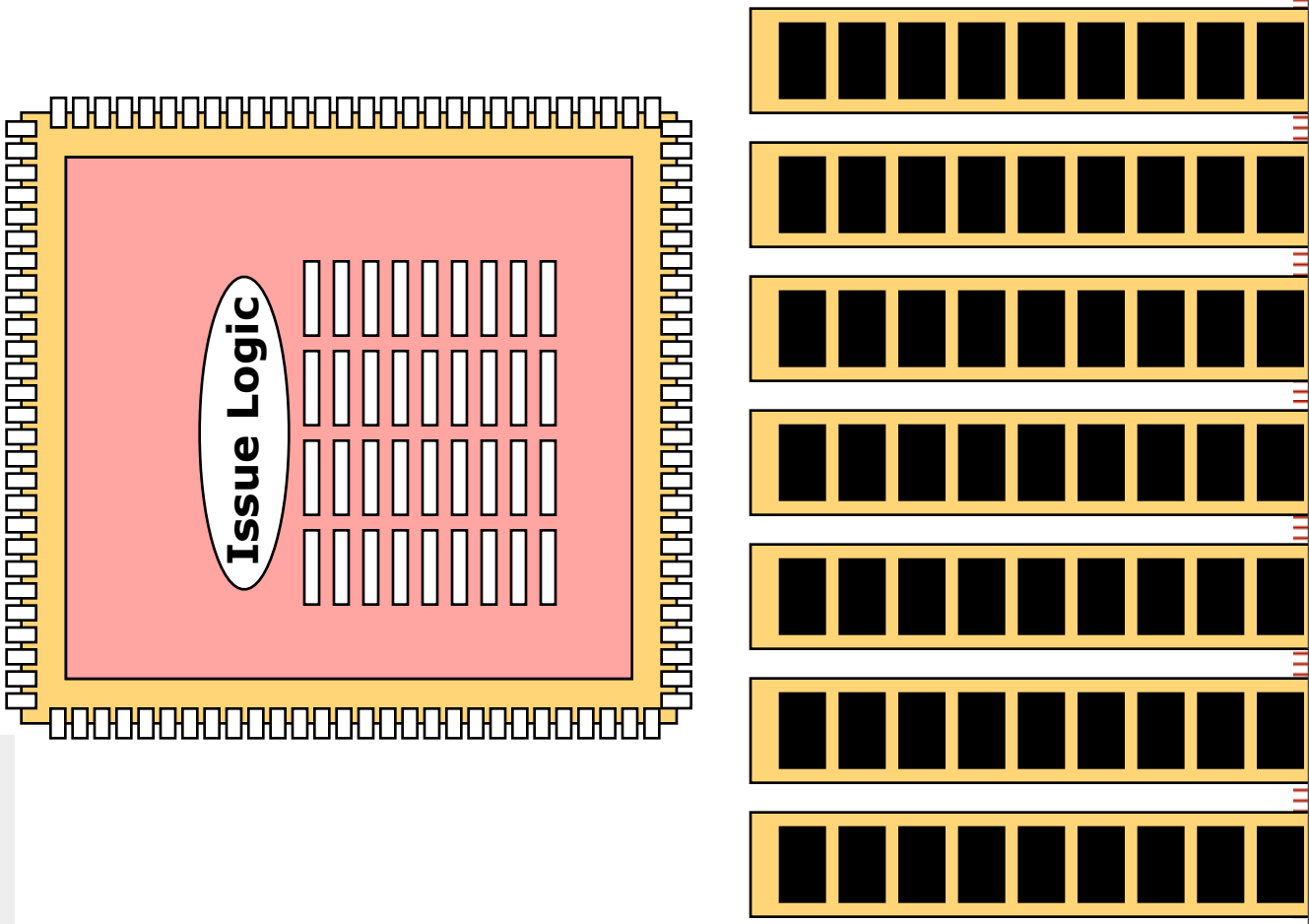
Bad News #1: Not enough ILP 1(2)

➔ **feed one CPU with instr. from many threads**



Bad News #1: Not enough ILP 2(2)

➔ feed one CPU with instr. from many threads

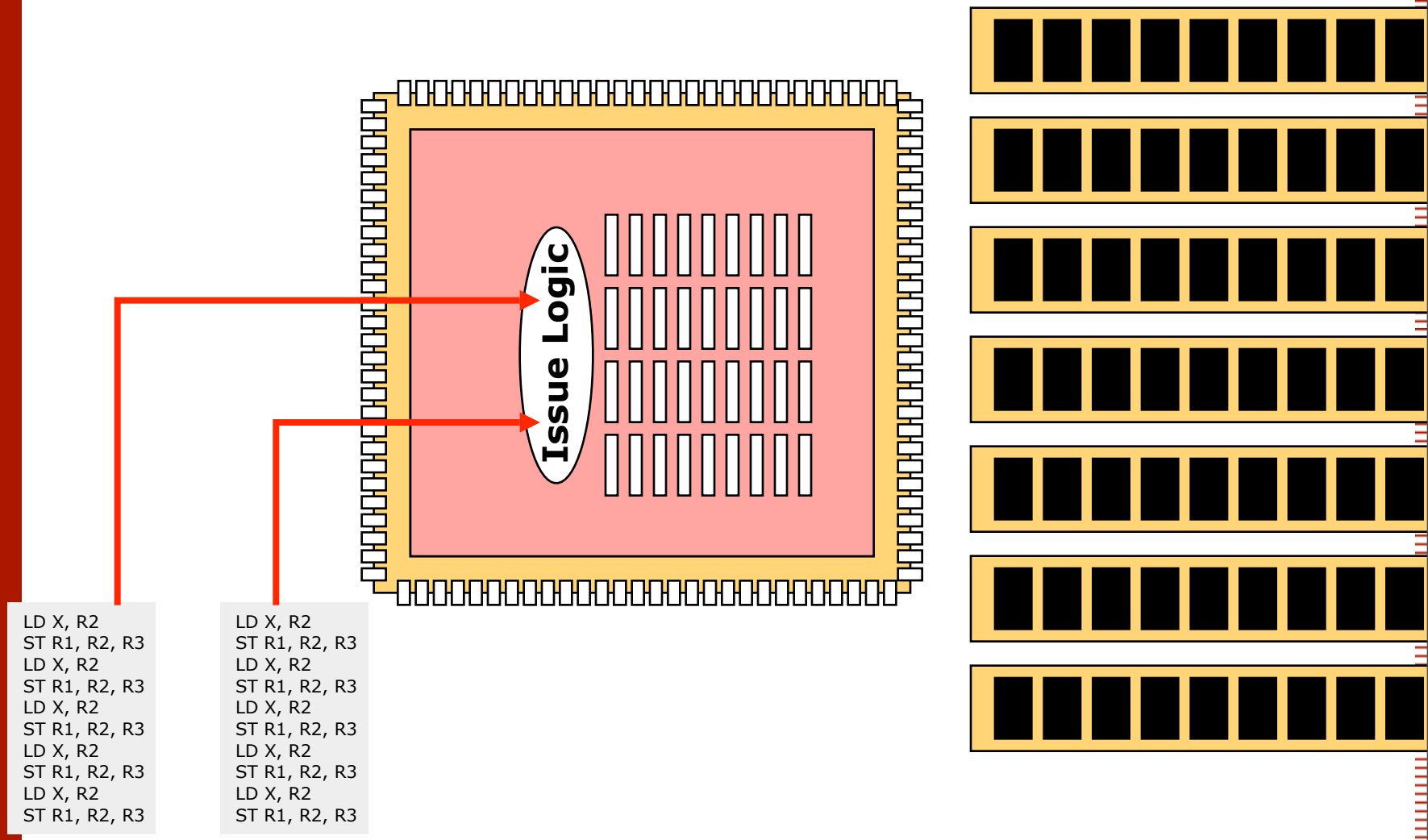



```
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
```

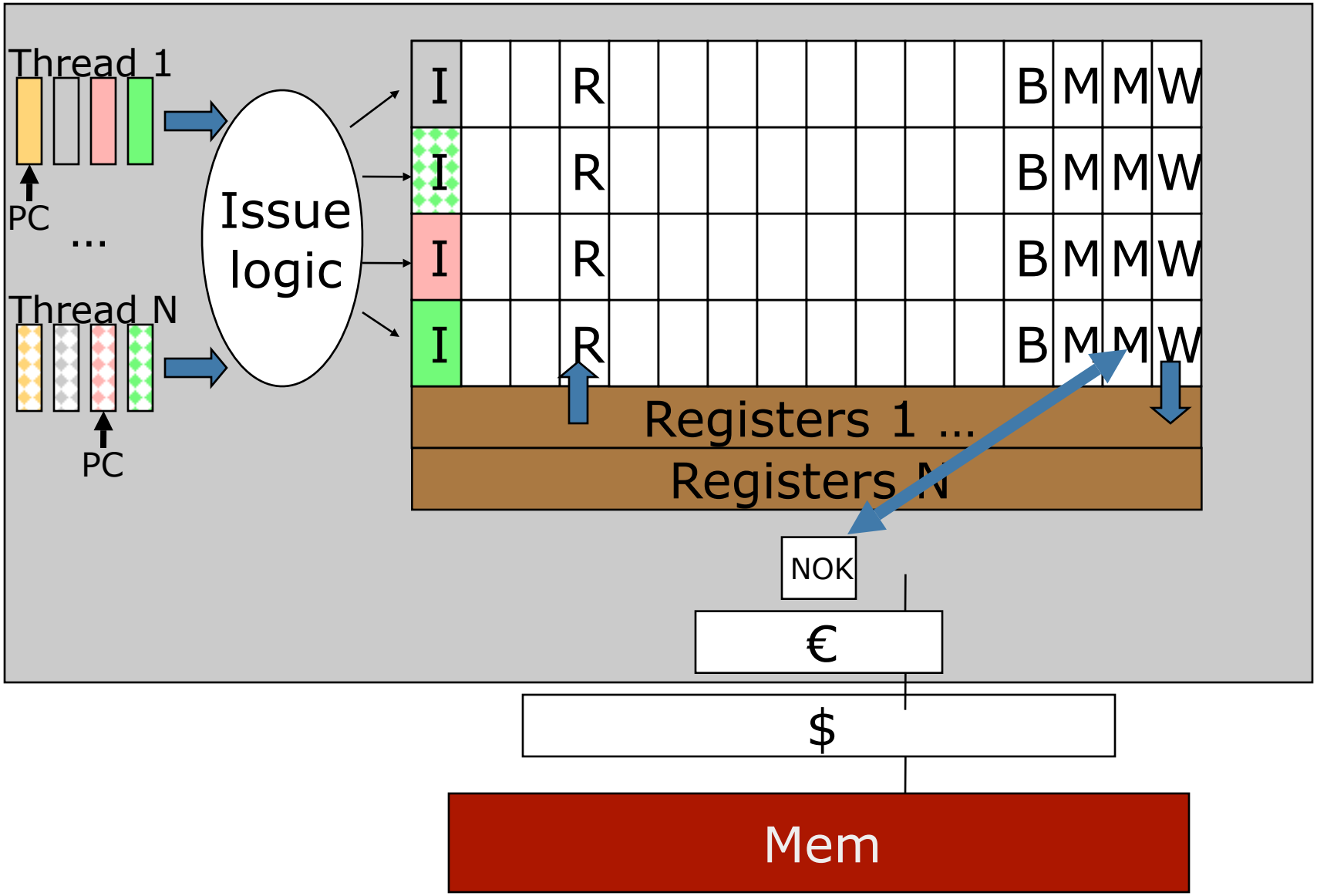
```
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
LD X, R2
ST R1, R2, R3
```

Bad News #1: Not enough ILP 2(2)

➔ feed one CPU with instr. from many threads



SMT: Simultaneous Multithreading



UPPSALA
UNIVERSITET

SMT is Not New ...

- Historical examples (1984 …)

Denelcor, HEP, Tera Computers [B. Smith]

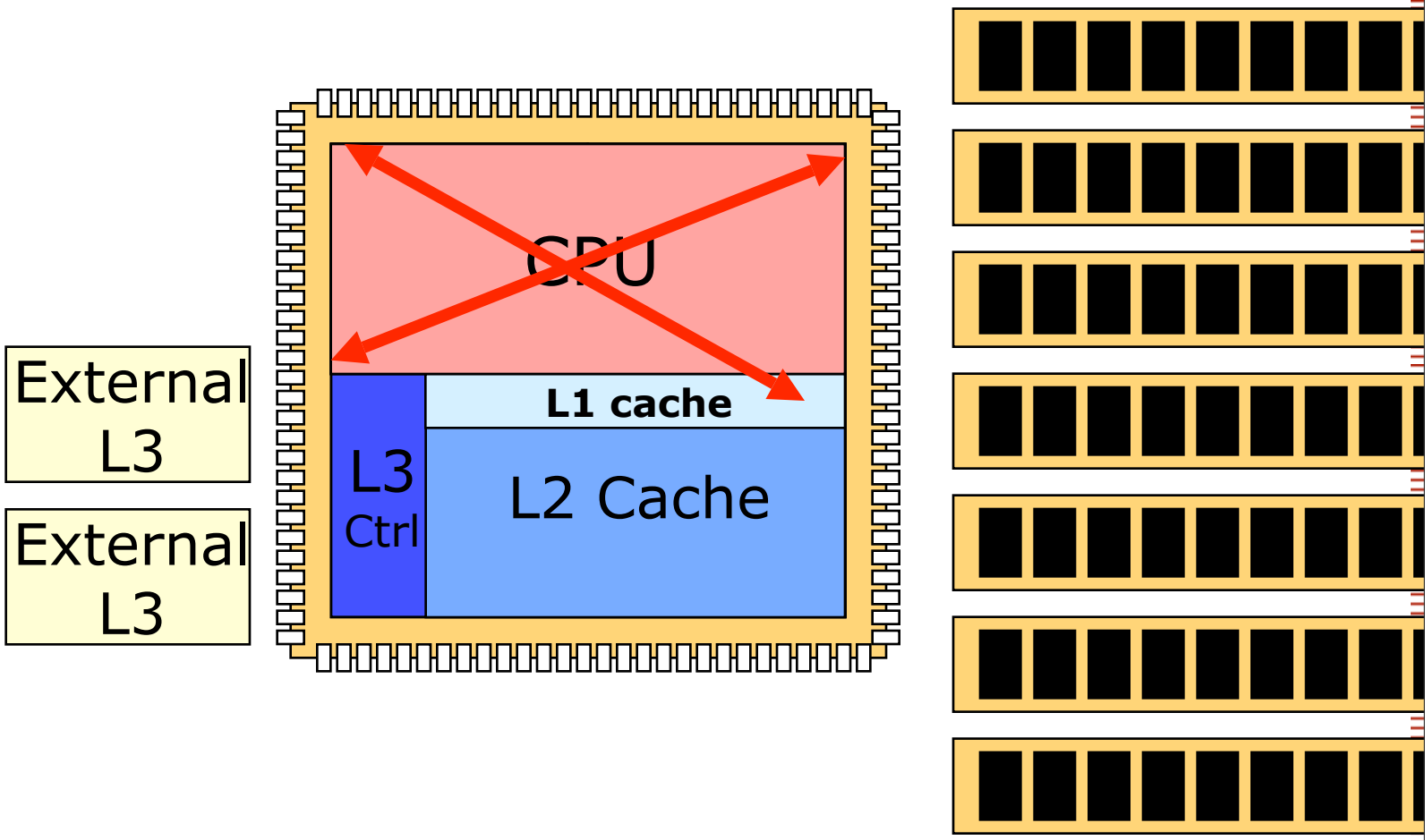
-- Poor single-thread performance

-- Expensive

UPPSALA UNIVERSITET

Bad News #2: wire delay

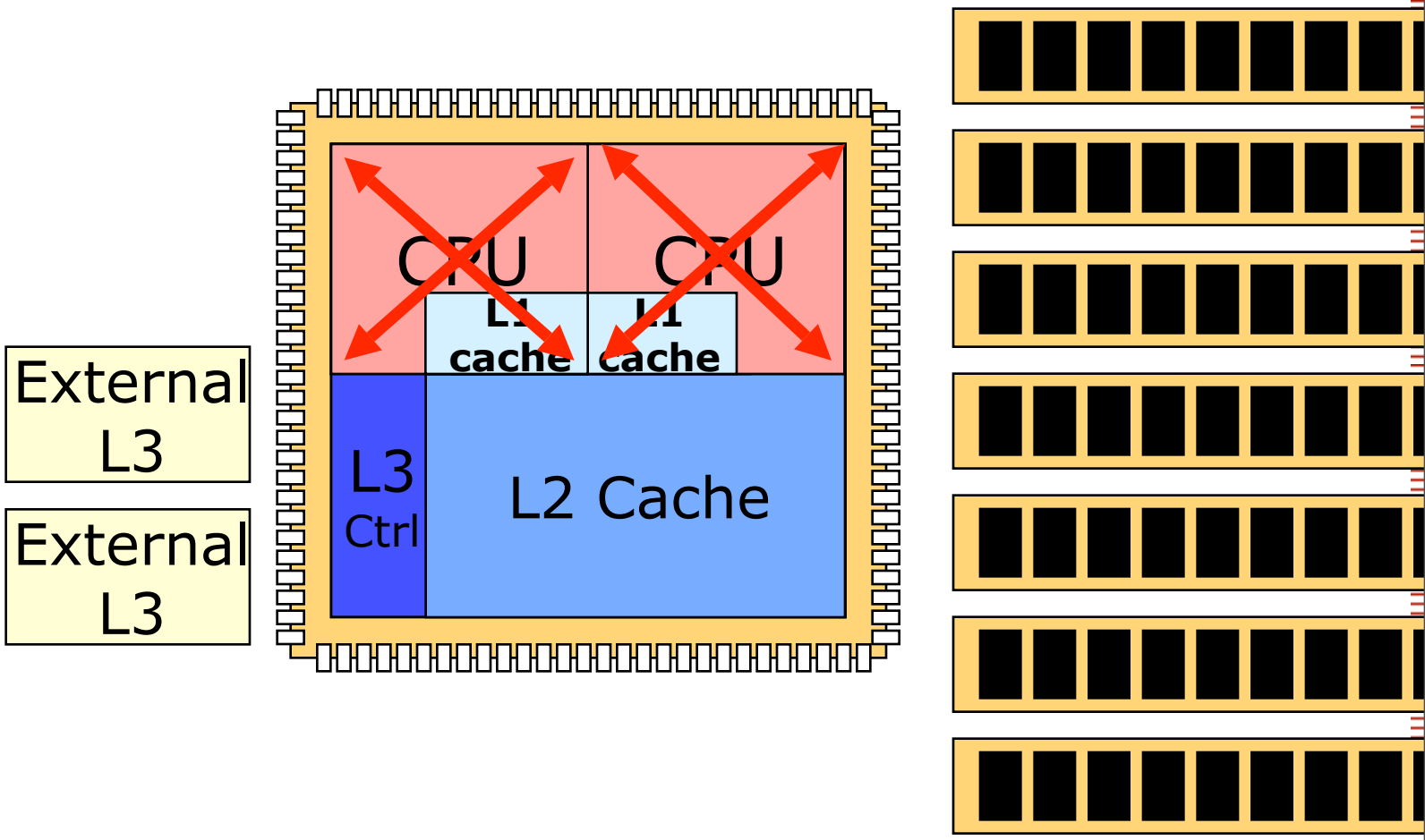
➔ Multiple small CPUs with private L1$



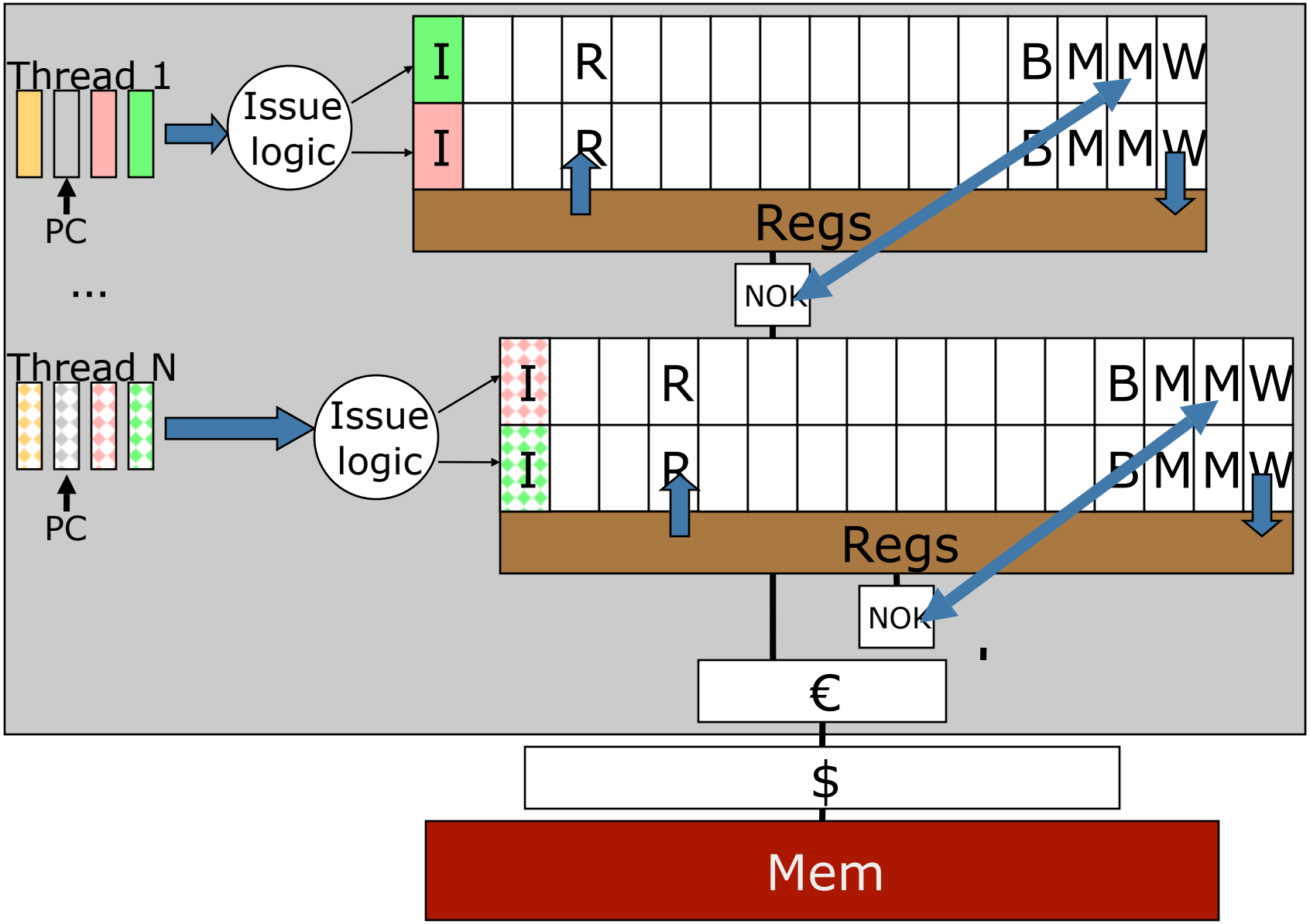
UPPSALA UNIVERSITET

Bad News #2: wire delay

➔ Multiple small CPUs with private L1$



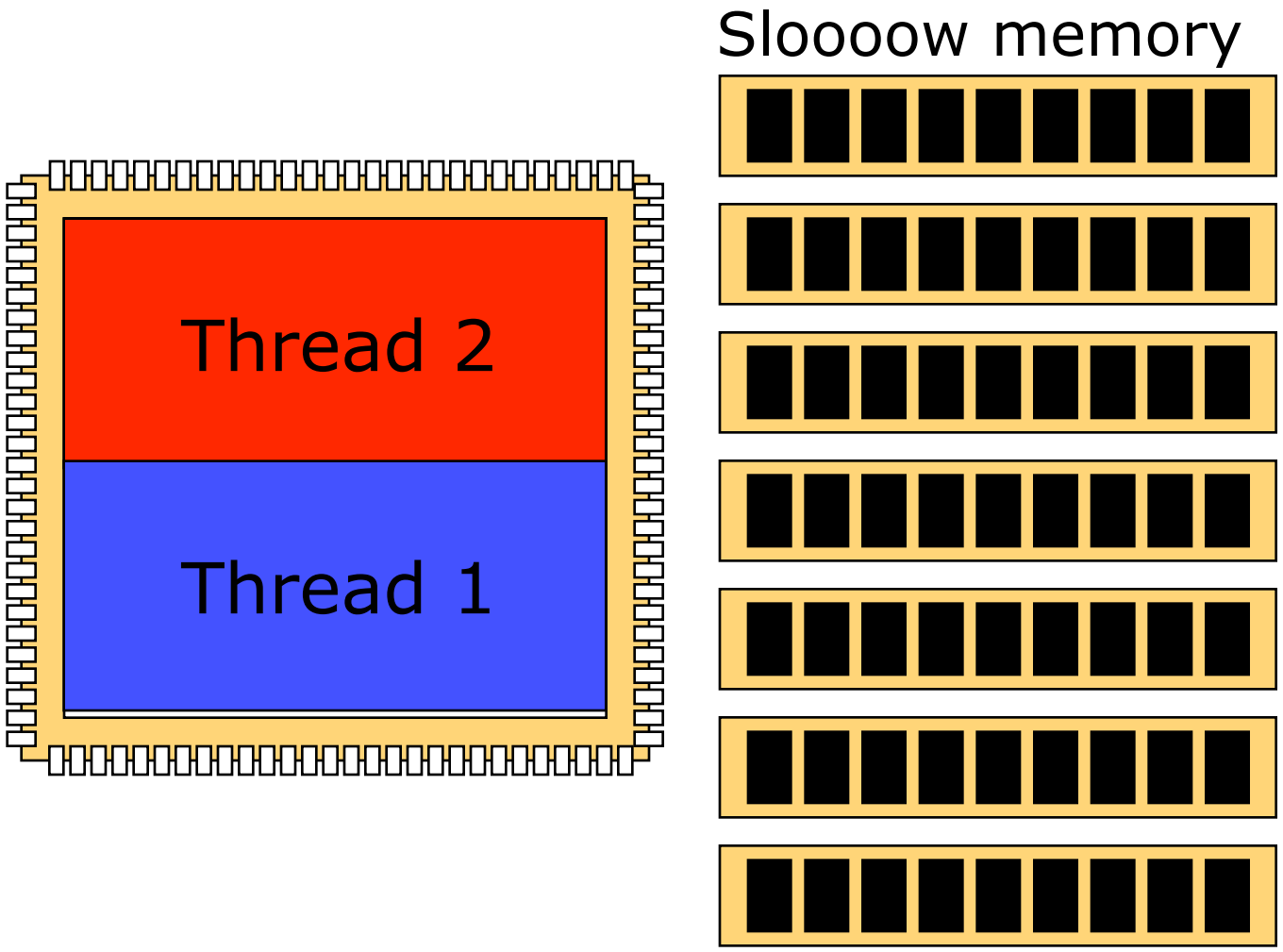
CMP: Chip Multiprocessor



UPPSALA UNIVERSITET

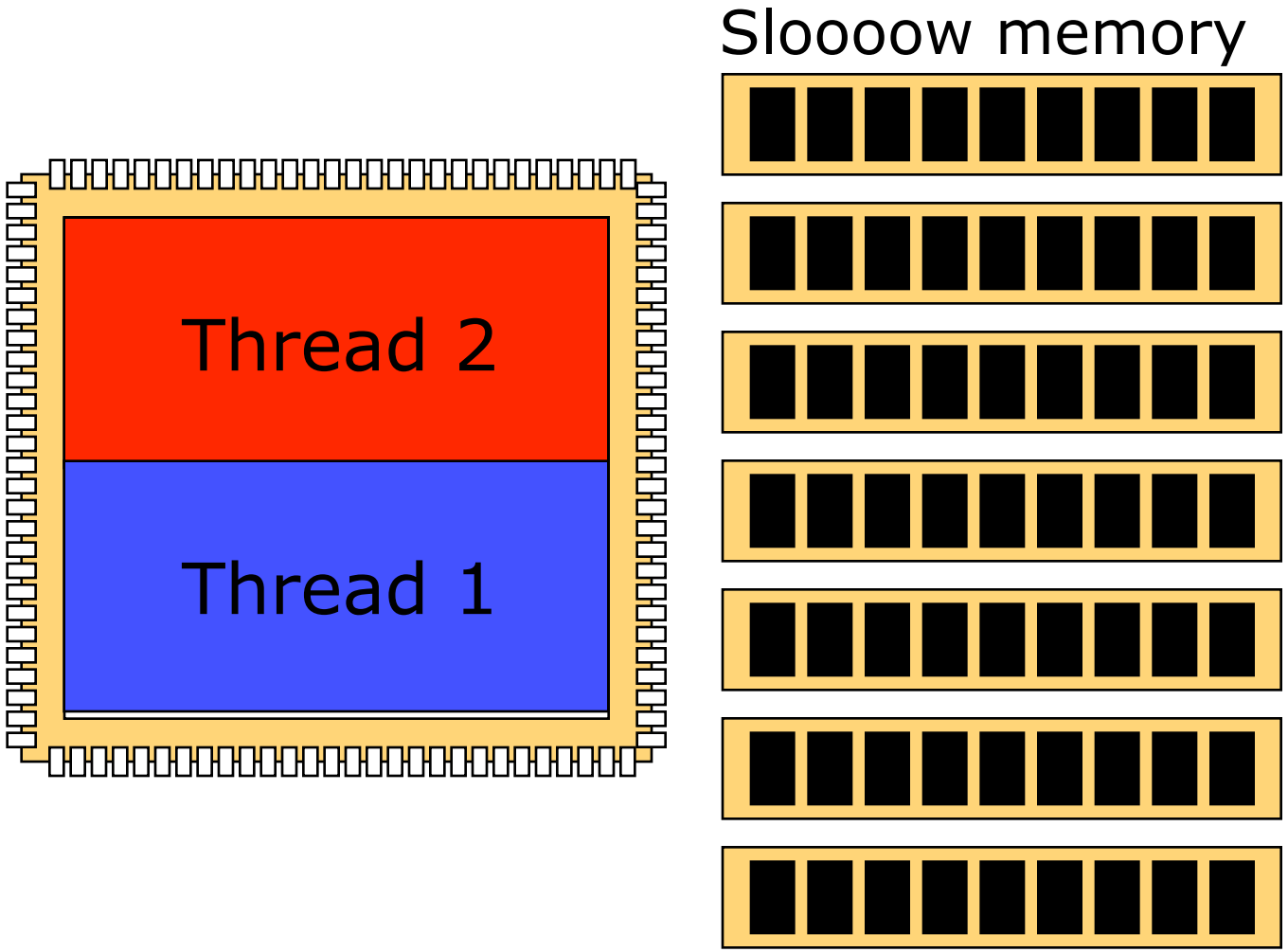
Bad News #3: memory latency/bandwidth

➔ **memory accesses from many threads**



Bad News #3: memory latency/bandwidth

➔ memory accesses from many threads



Thread-Level Parallellism ➔ Memory-Level Parallelism (MLP)

UPPSALA UNIVERSITET

Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

CPU
freq=f

$P_{dyn}(C, f, V) = CfV^2$

UPPSALA UNIVERSITET

Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

$P_{dyn}(C, f, V) = CfV^2$

Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

$P_{dyn}(C, f, V) = CfV^2$

$P_{dyn}(2C, f/2, <V) < CfV^2$

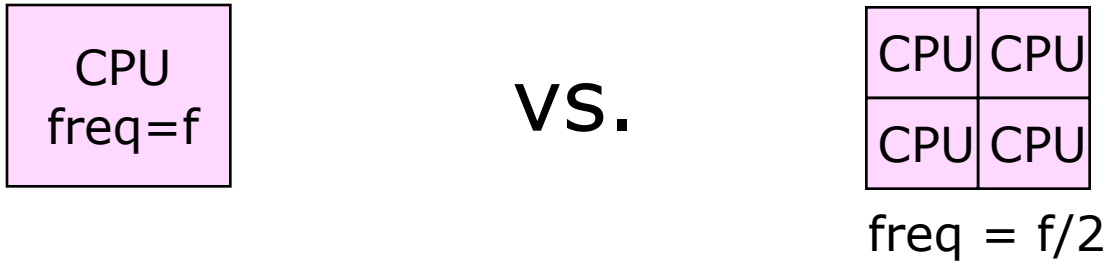
Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

$P_{dyn}(C, f, V) = CfV^2$

$P_{dyn}(2C, f/2, <V) < CfV^2$



Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

$P_{dyn}(C, f, V) = CfV^2$

$P_{dyn}(2C, f/2, <V) < CfV^2$

$P_{dyn}(C, f, V) = CfV^2$

$P_{dyn}(C, f/2, <V) < ½\ CfV^2$

Bad News #4: Power consumption

➔ Lower the frequency ➔ lower voltage

$$P_{dyn} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

$P_{dyn}(C, f, V) = CfV^2$

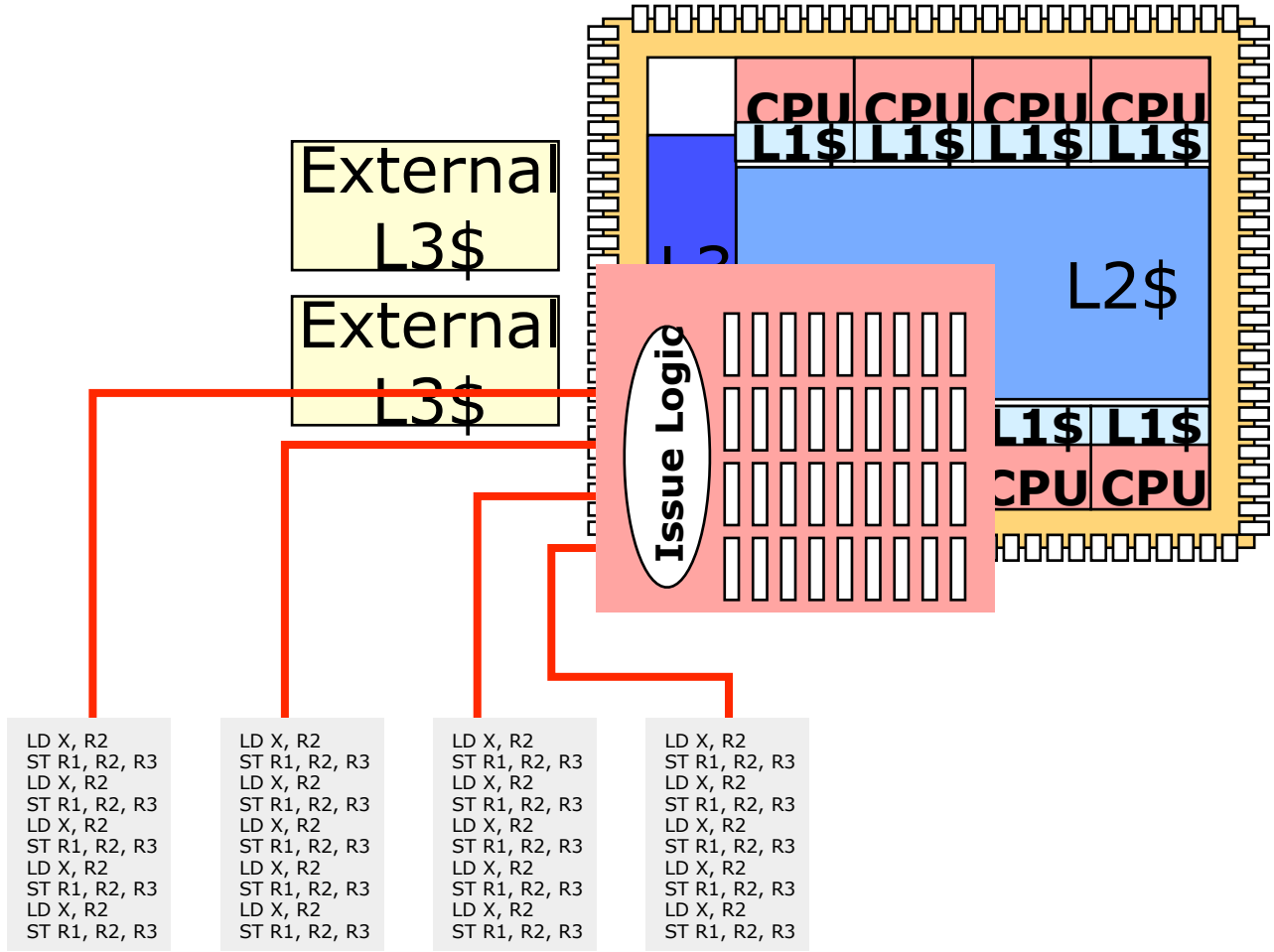
$P_{dyn}(2C, f/2, <V) < CfV^2$

$P_{dyn}(C, f, V) = CfV^2$

$P_{dyn}(C, f/2, <V) < ½\ CfV^2$

In all computers very soon…

Chip multiprocessors, probably also with simultaneous multithreading



UPPSALA UNIVERSITET

Why multiple cores/threads?

- Instruction level parallelism is running out
  - Feed the CPU with instructions from many threads (Simultaneous MultiThreading, SMT)
- Wire delay is hurting now
  - Use the chip for multiple smaller CPU cores (Chip MultiProcessors, CMP)
- Power is a limit now
  - Lower the frequency (which enables lower voltage)
- Memory latency/bandwidth bottleneck
  - Access the memory from many threads

UPPSALA UNIVERSITET

Shared memory systems

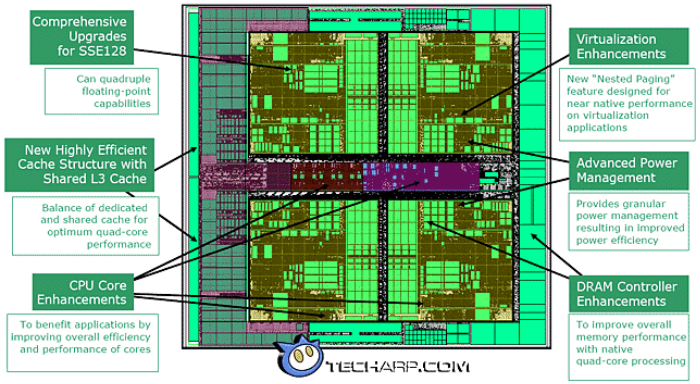
Several CPUs are now sharing the same memory!

- Several CPUs may try to write to a memory location (”shared variable”) at the same time
  - We need synchronization/communication primitives!
- Local caches (including registers) will result in multiple copies of data
  - We need a coherency protocol!

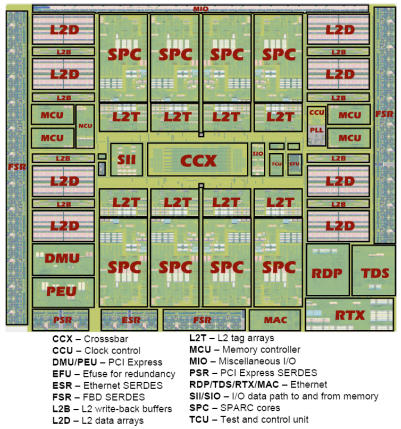
UPPSALA
UNIVERSITET

Multicore Systems, Now and in the Future

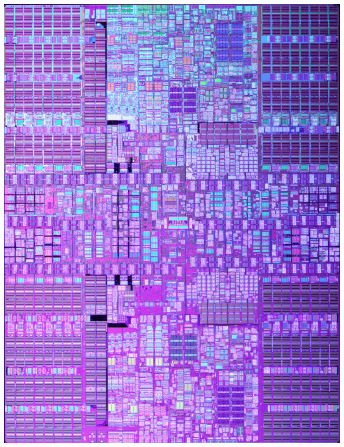
Some Current Multicore Proc.



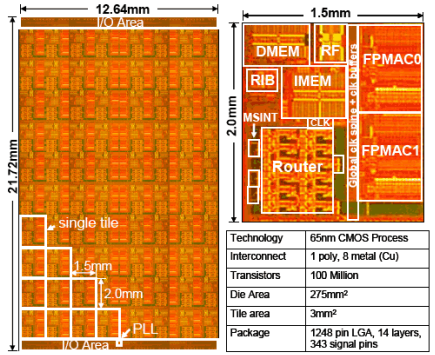
Quad-core AMD Opteron (Barcelona)



Eight-core, 64 thread Sparc T2 (Niagara 2)



Dual-core, four thread IBM Power6



80-core "Intel Teraflop Research Chip"

UPPSALA UNIVERSITET

CMP Design Space

- Number of cores/chip
- Type of core (in-order, scalar, superscalar)
- Core clock frequency
- Number of cache levels and cache structure (L1,L2,L3,...; shared or local; associativity, ...)
- Threads per core (1 or SMT)
- Number and type of channels to memory, on-chip memory controller or not, ...

UPPSALA UNIVERSITET

CMP Design Space

”Fat cores” ⇔ ”Thin cores” ?

- In PC processors today: A few fat, supercalar cores with rather high clock frequency
- In some server processors tomorrow: Many thin cores with mediocre clock frequency

- In PC processors tomorrow: Inhomogenous architectures with a few fat cores, many thin cores, and possibly a graphics unit/GPU ???

UPPSALA UNIVERSITET

Some Current Multcore Proc.

- **Intel Quad-core ("Harpertown")**: Two dual-core on a chip, 3.2 GHz, 2*6 MB L2, superscalar (November 2007)
- **AMD Quad-core ("Barcelona")**: Monolitic, 2.3 GHz, Local L1, L2, shared L3 and memory controller on chip, superscalar (December 2007)
- **IBM** Power6: Dual core, 2-way SMT. 4.7 GHz, 8MB L2, in-order (May 21 2007)
- Sun SPARC T2 ("Niagara 2"): Eight-core, 8-way SMT, 1.4 GHz, 4MB L2, in-order (August 7 2007)

UPPSALA UNIVERSITET

Some Future Multcore Proc.

- Intel ”Nehalem” (Late 2008 ?):
  - 8 cores
  - 16 threads
  - 24 MB L3 cache
  - New interconnect (not ”Front Side Bus”)
  - GPU in the same socket ??
  - Followed by ”Sandy Bridge”, up to 32 cores ??
- Sun ”Rock” (Early 2009 ?):
  - 16 cores
  - 32 threads
  - New architectural optimization features
  - Support for transactional memory

UPPSALA UNIVERSITET

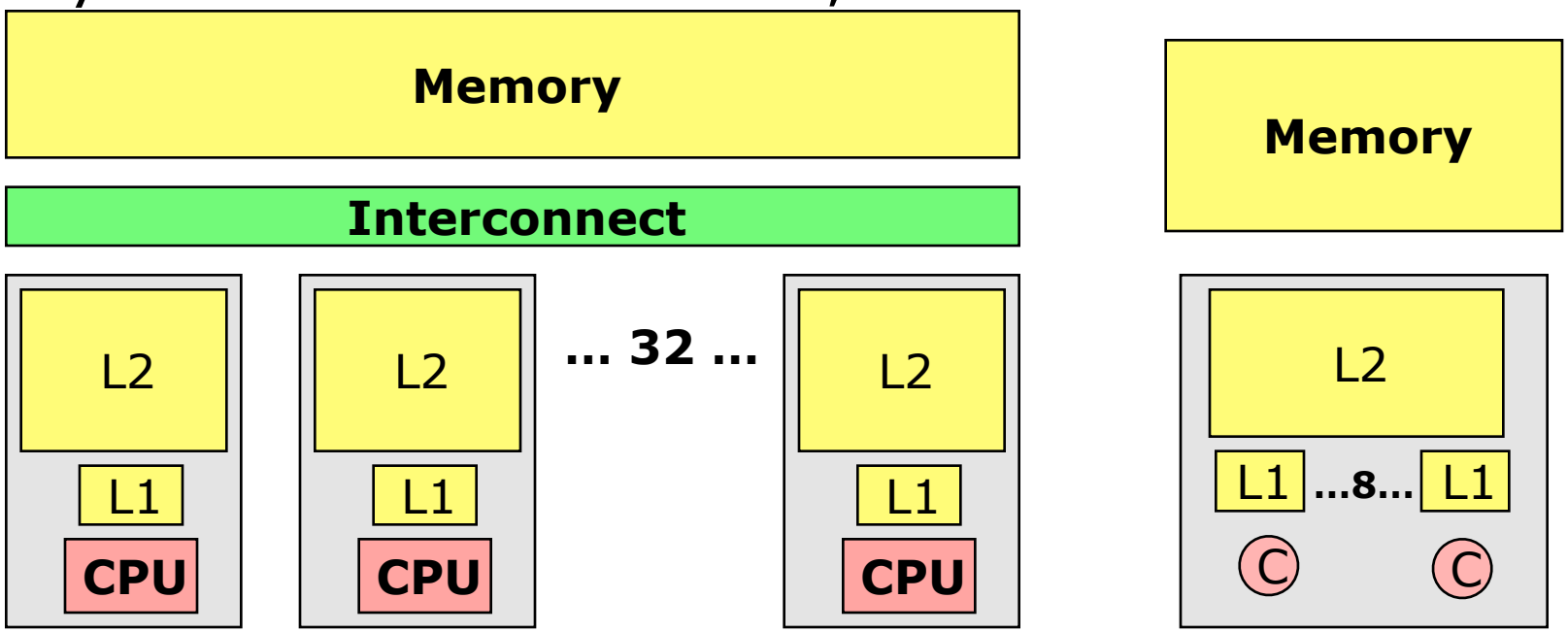
The Multicore Era has been here for a while!

- *Intel have 10 projects in the works that contain four or more computing cores per chip* [Paul Otellini, Intel Chief Excecutive at IDF fall 2005]
- Starting Q3 2006, Intel sells more multicore processors than single cores
- All major microprocessor vendors now focus on CMPs (with fat and/or with thin cores)
- Rumours of a 2048-thread single-box system in 2008 (Sun ”Maramba”)

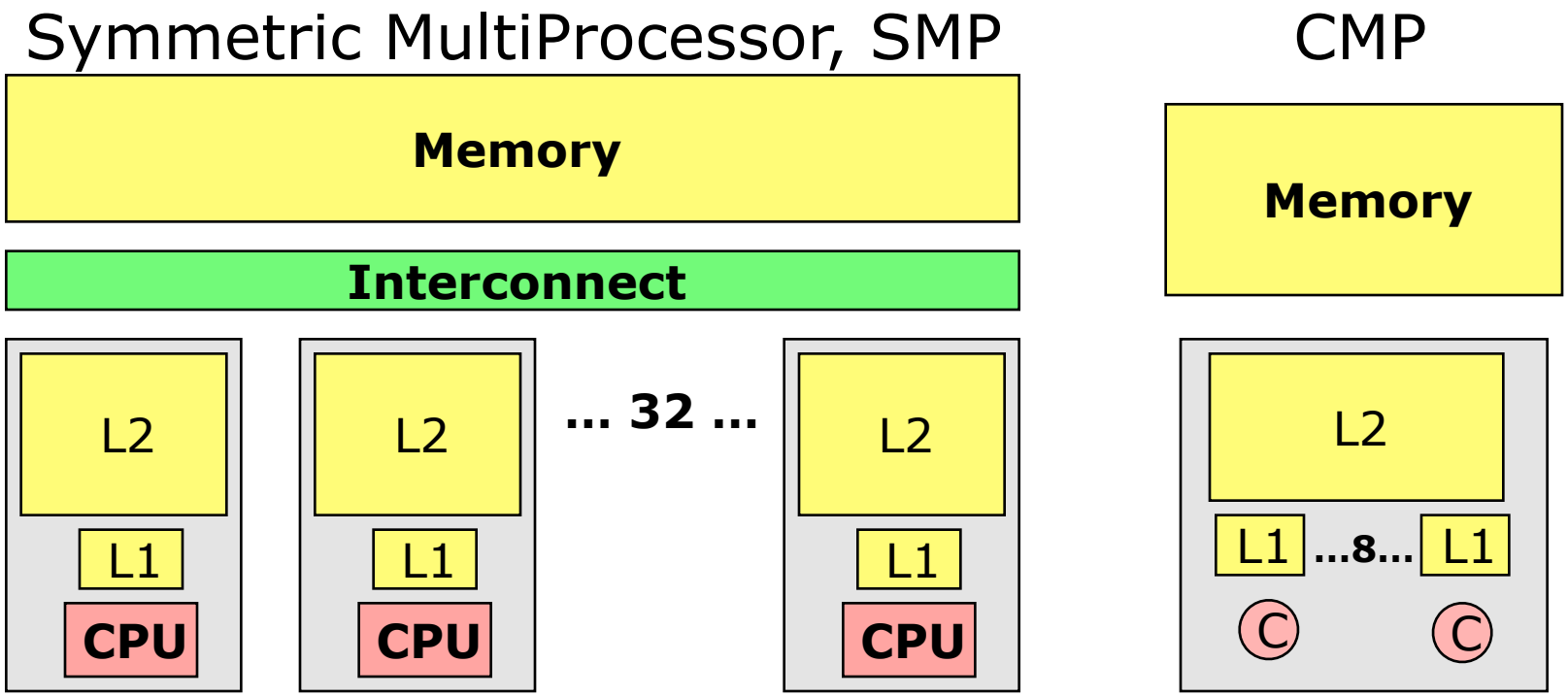
UPPSALA
UNIVERSITET

Impact on Performance

Isn't a CMP just an SMP on a chip?



Isn't a CMP just an SMP on a chip?



Well, how about:

- Cost of parallelism?
- Cache capacity per thread?
- Memory bandwidth per thread?
- Cost of thread communication?

UPPSALA UNIVERSITET

Cost of parallelism

- *Individual processor [core] costs will drop so quickly that in the near future the world can view processors as a nearly free commodity* [IDC(*) presentation on HPC at International Supercomputing Conference, ICS07, Dresden, June 26-29 2007]

(*) IDC (International Data Corporation) is a major market research and analysis firm specializing in information technology, telecommunications and consumer technology.

UPPSALA UNIVERSITET

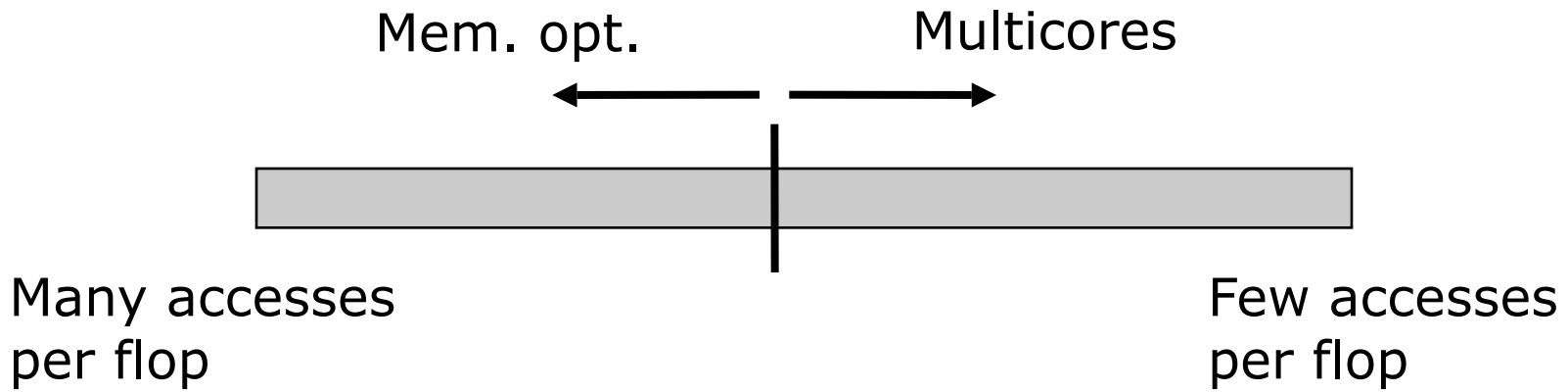
Cache per Thread

- **Intel Quad-core**: 3 MB/thread L2
- **IBM** Power6: 2 MB/thread L2
- **AMD Quad-core**: 512 kB local L2, 512 kB/thread L3
- Sun SPARC T2: 64 kB/thread L2

UPPSALA UNIVERSITET

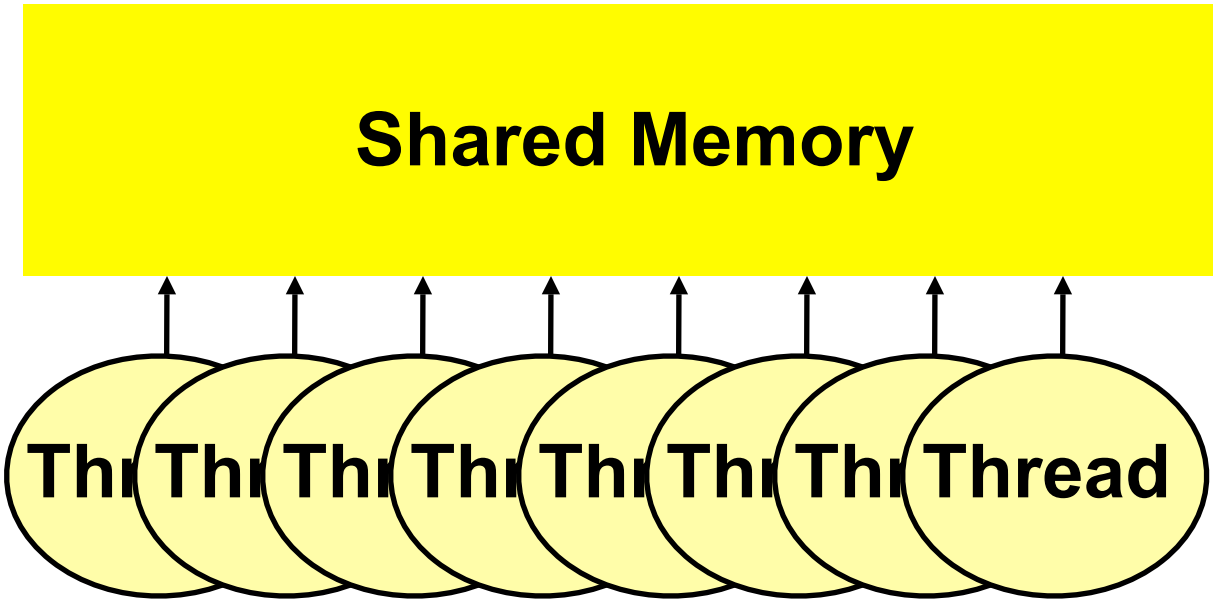
Memory Bandwidth

- *Memory bandwidth to a single processor core will decline at a fast rate* [IDC presentation on HPC]
- *The flow of data will become the major road block and hence the area of opportunity for performance speedups* [IDC...]

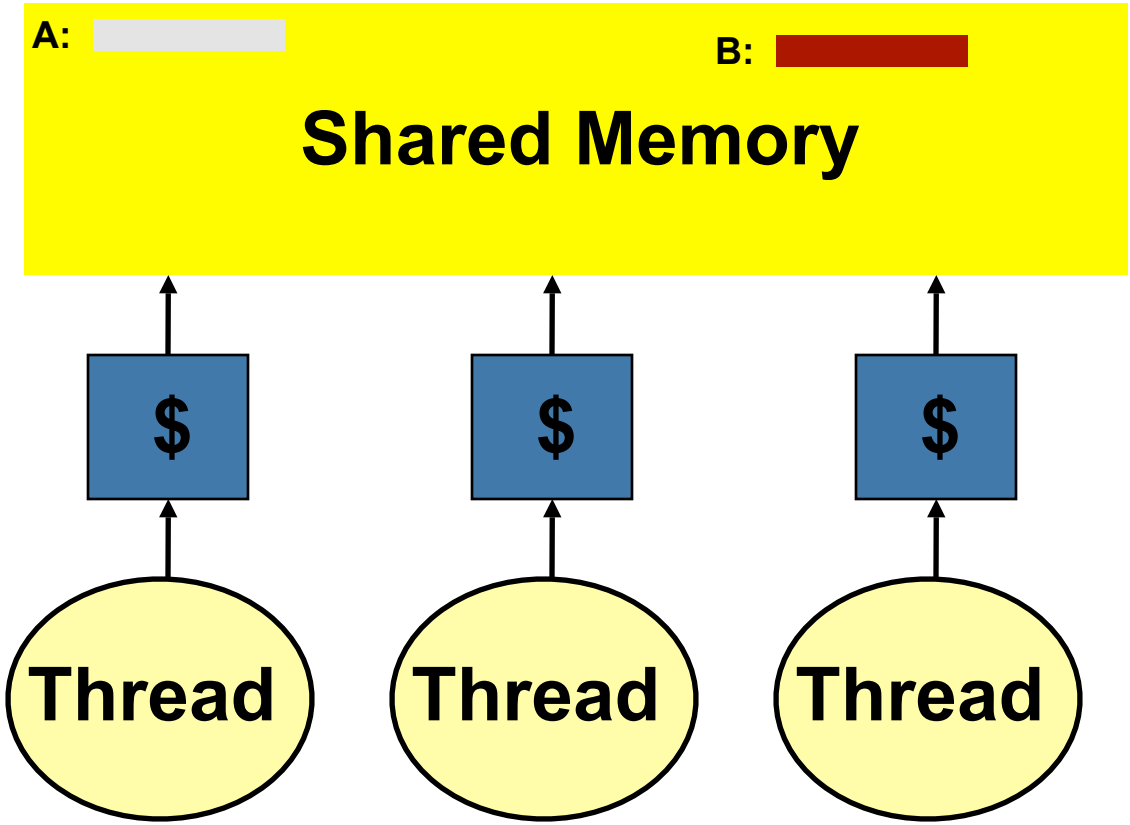


The Coherent Memory System

Multithread programming model (e.g. OpenMP):



Caches: Automatic Replication of Data



Read A

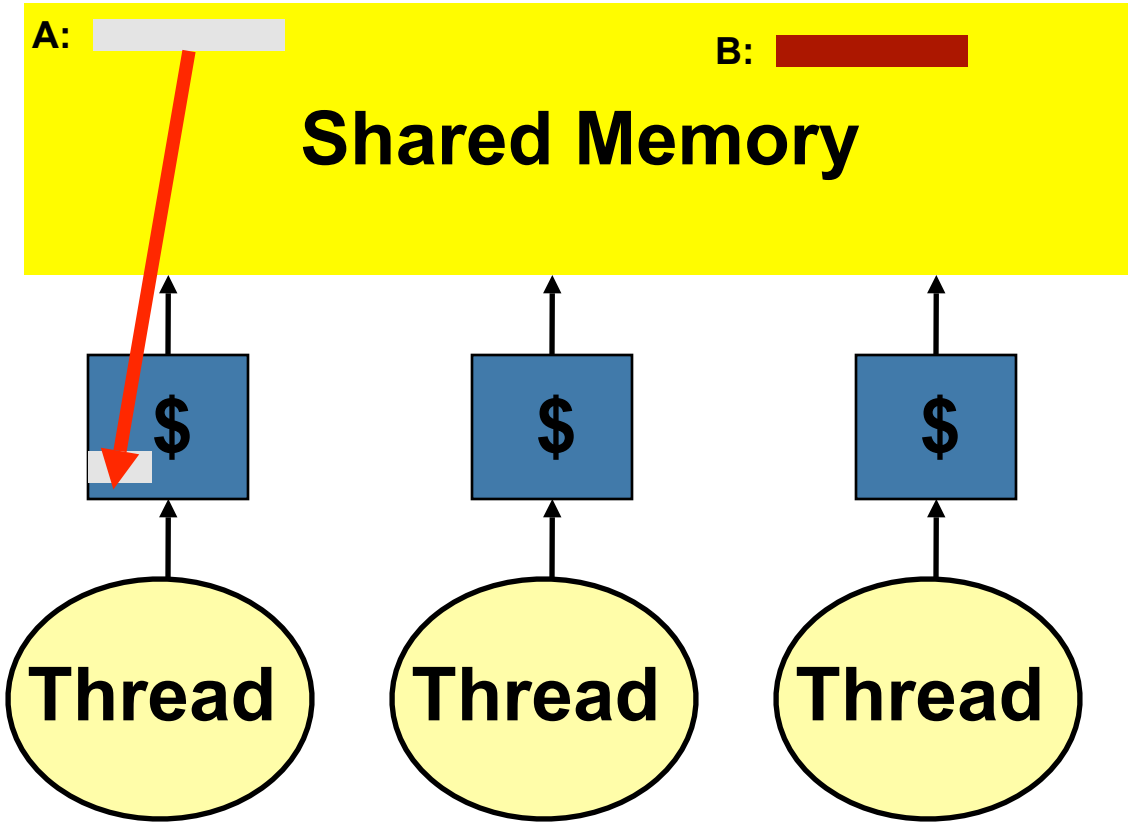
Read A

…

…

Read A

Caches: Automatic Replication of Data



Read A

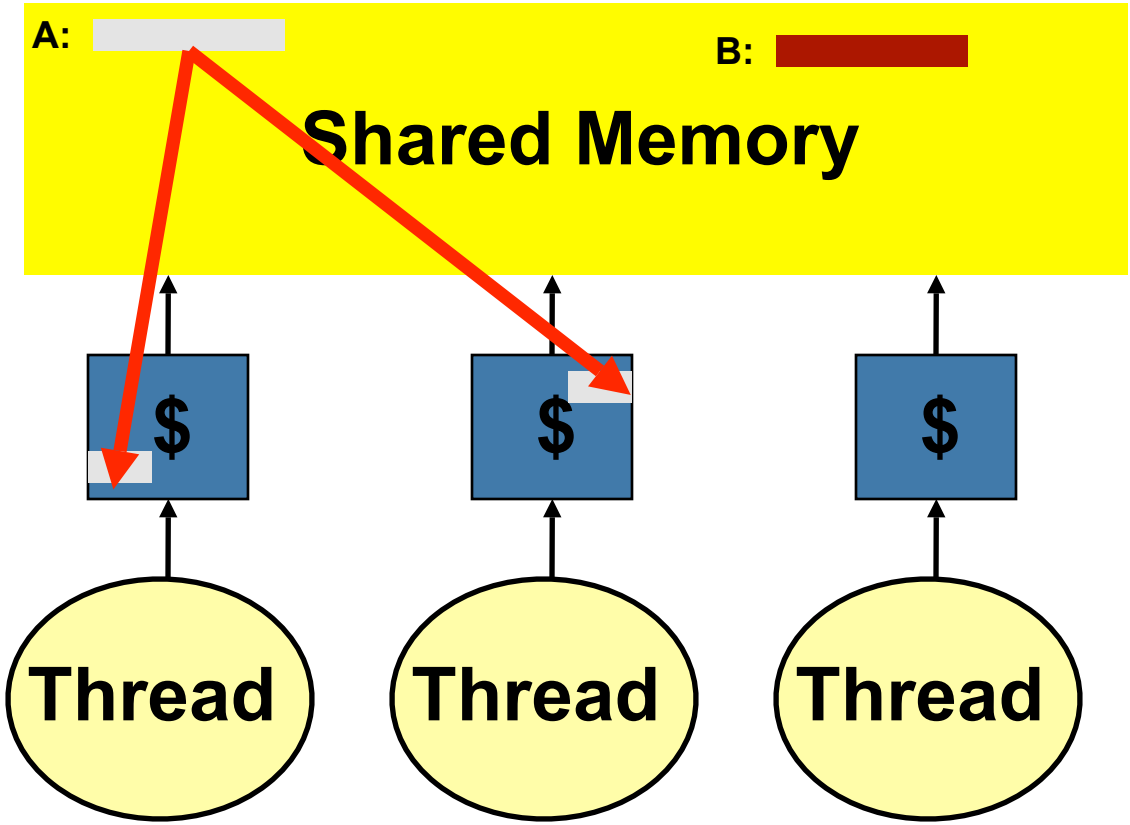
Read A

…

…

Read A

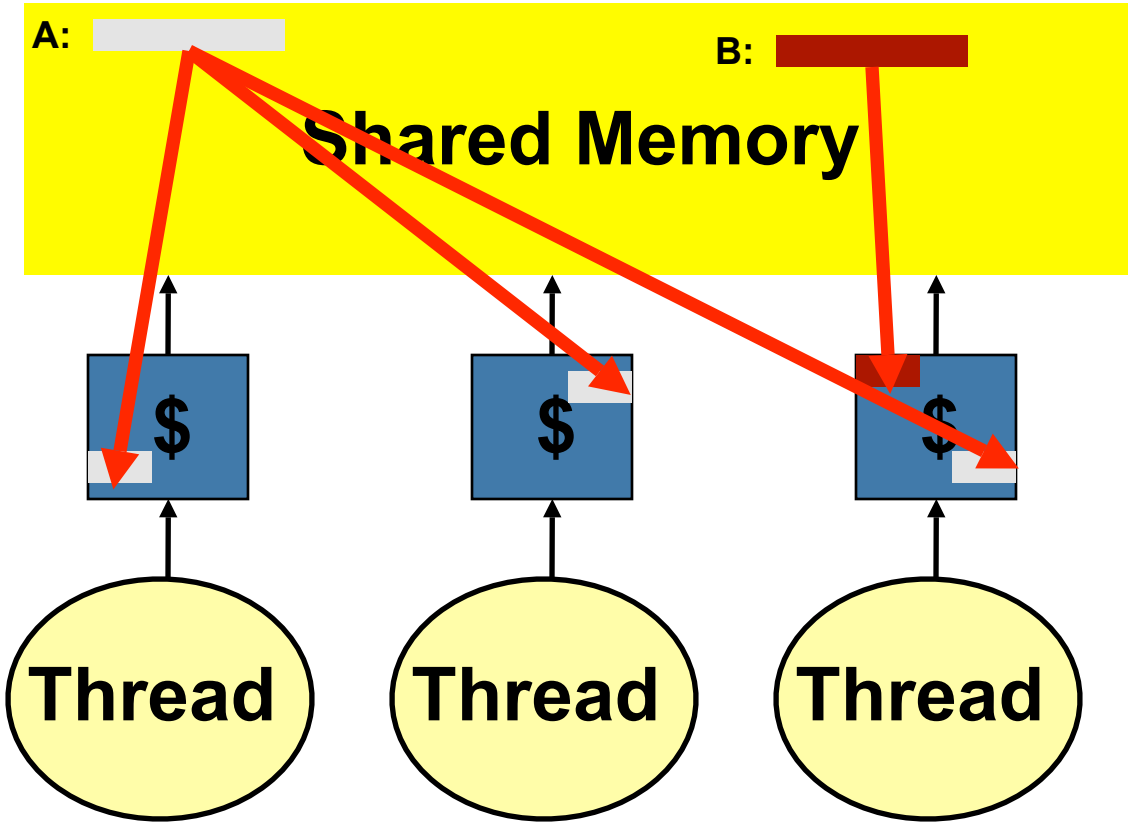
Caches: Automatic Replication of Data



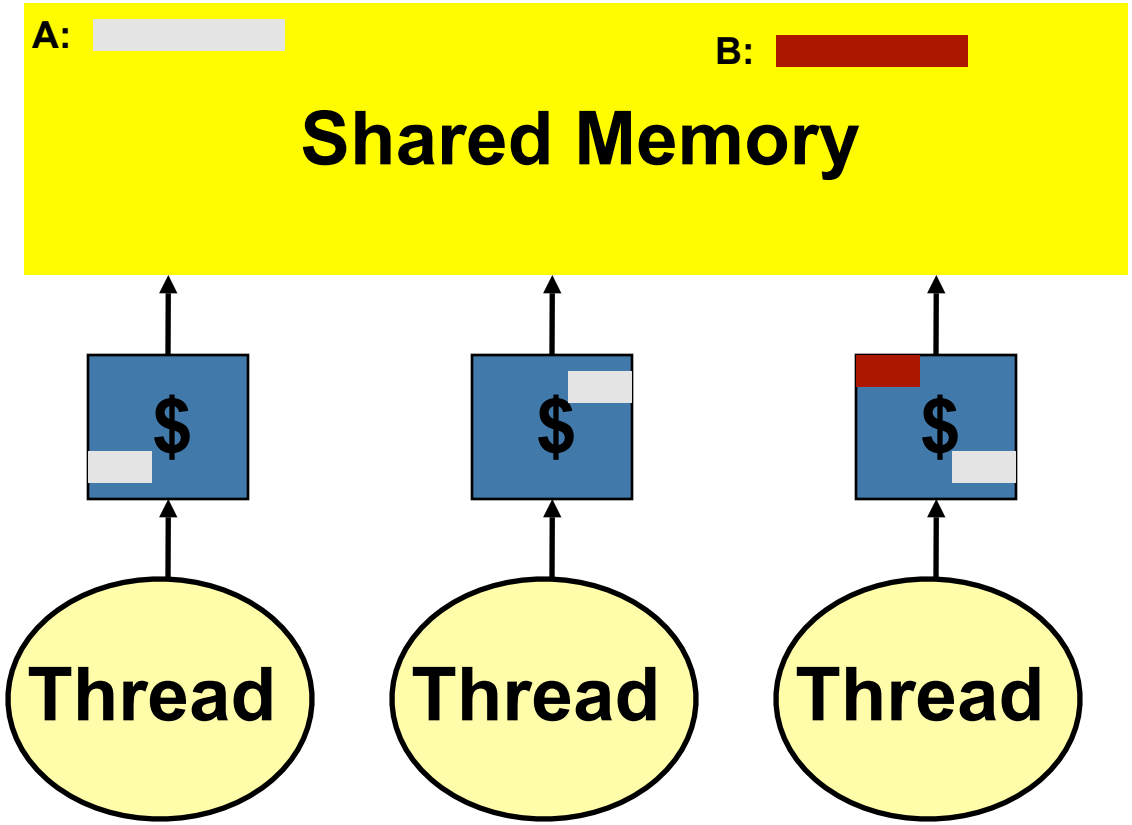
Read A
Read A
…
…
Read A

…
Read A
…

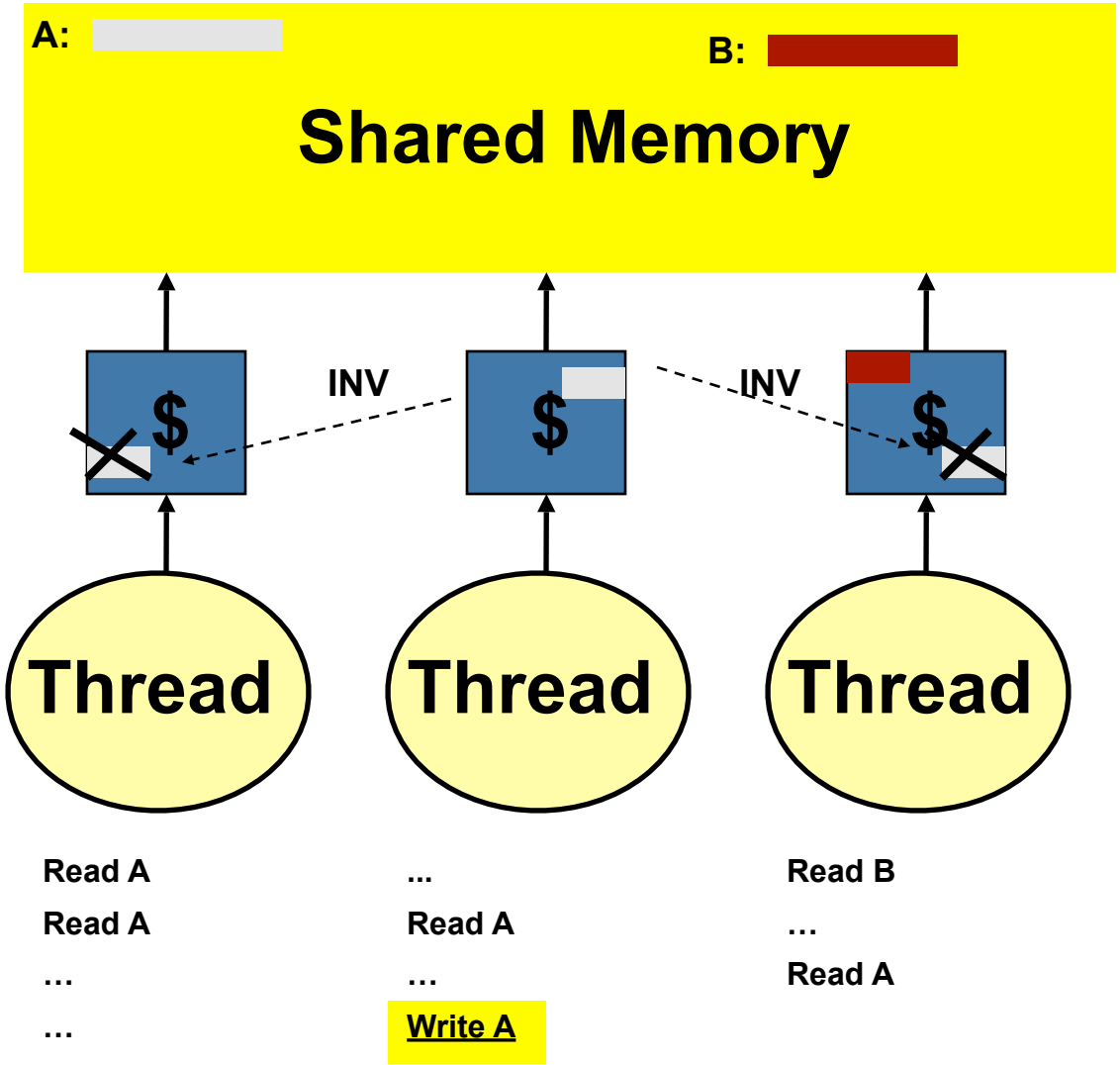
Caches: Automatic Replication of Data



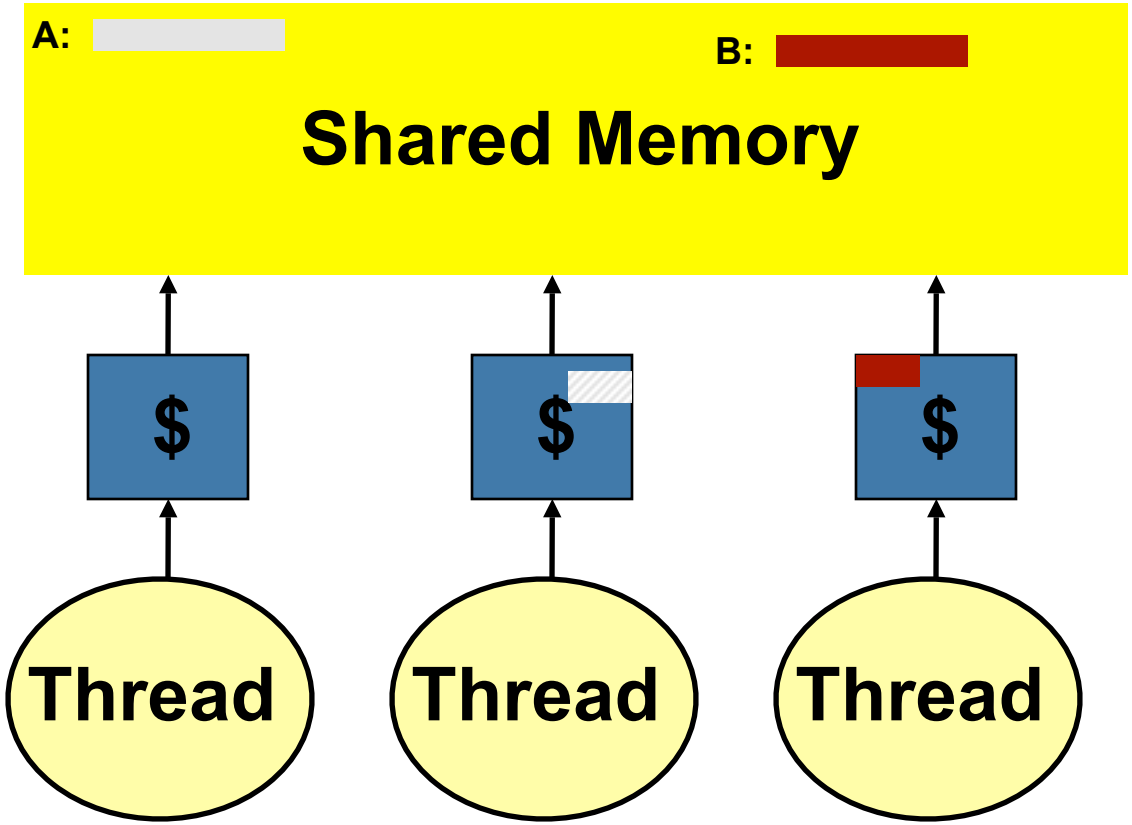
The Cache Coherent Memory System



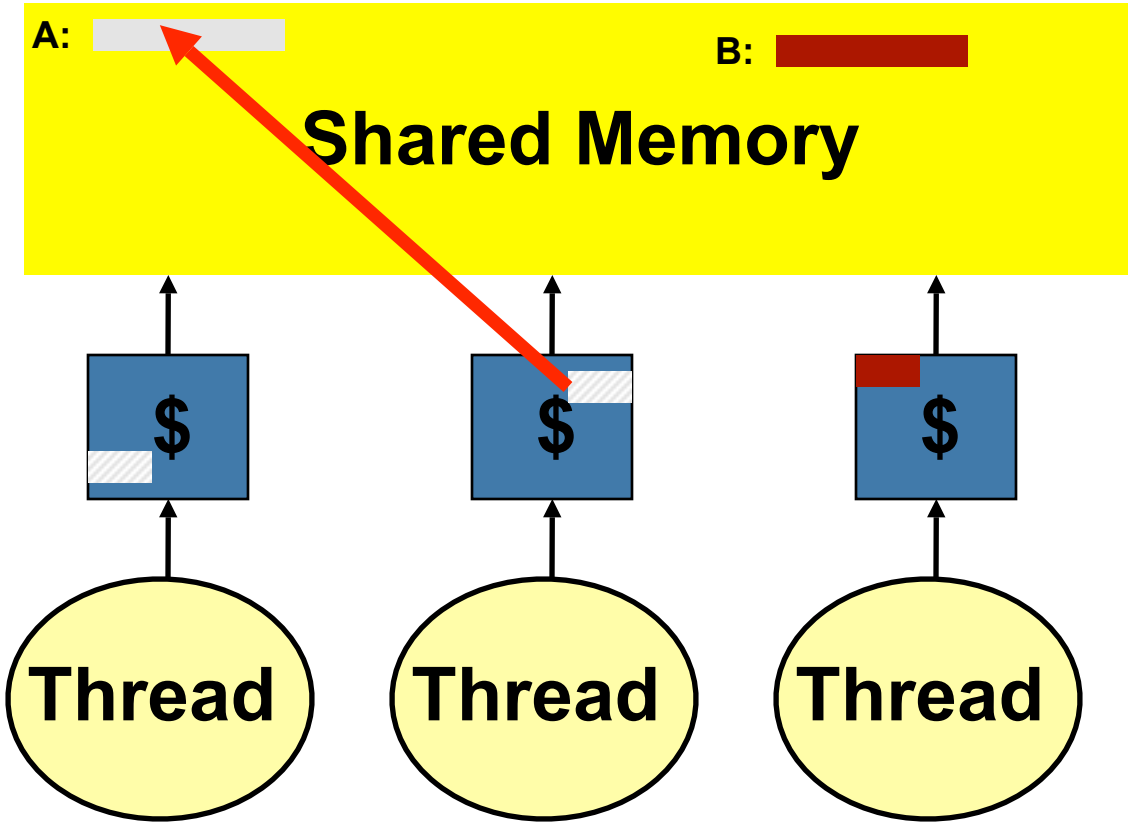
The Cache Coherent Memory System



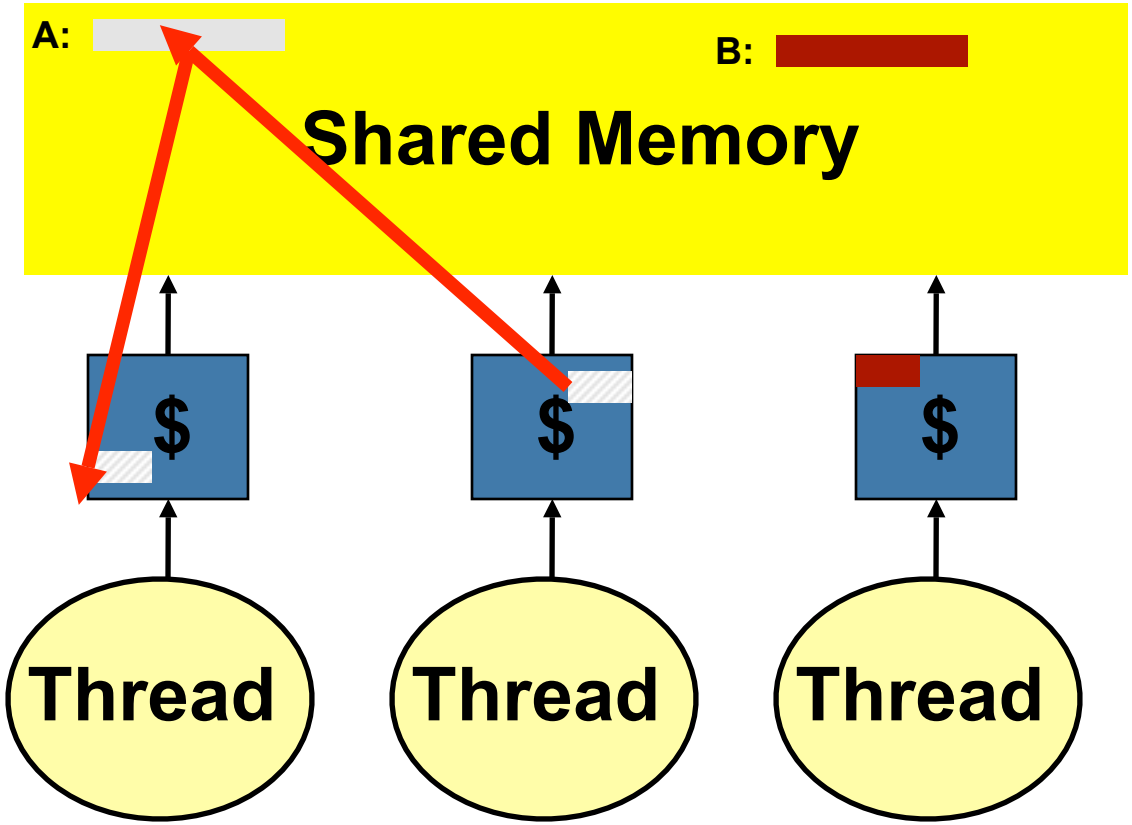
The Cache Coherent Cache-to-cache



The Cache Coherent Cache-to-cache

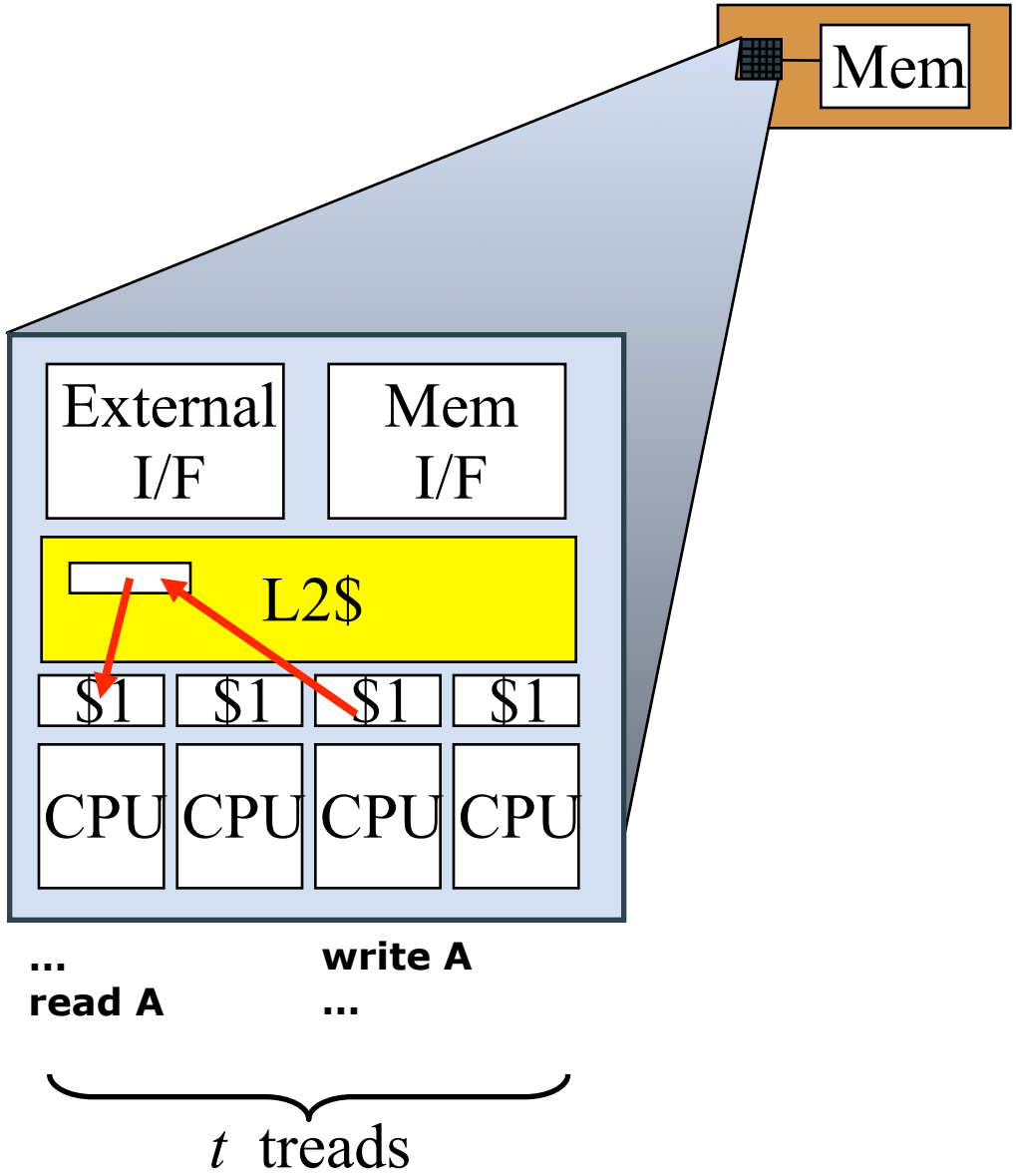


The Cache Coherent Cache-to-cache



UPPSALA UNIVERSITET

Communication in a CMP



UPPSALA
UNIVERSITET

Do we need to bother?

IDC presentation again ...

Users who find new methodologies for dealing with the added level of parallelism could see rapid evolution in capability as processor cores become dramatically less expensive

UPPSALA UNIVERSITET

Similar Thoughts are Shared by Others

The Matlab News&Notes Newsletter,
Summer 2007:

Cleve's Corner: Parallel MATLAB

The proliferation of multicore systems and clusters sets the stage for parallel computing with MATLAB.

Programming Patterns: Maximizing Code Performance by Optimizing Memory Access

Improve memory usage and increase code speed by understanding how MATLAB stores and accesses data.

...

UPPSALA
UNIVERSITET

Impact on CSE Software in General ?

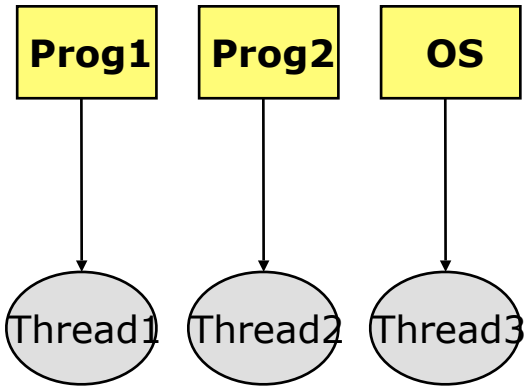
UPPSALA UNIVERSITET

Some Thoughts ...

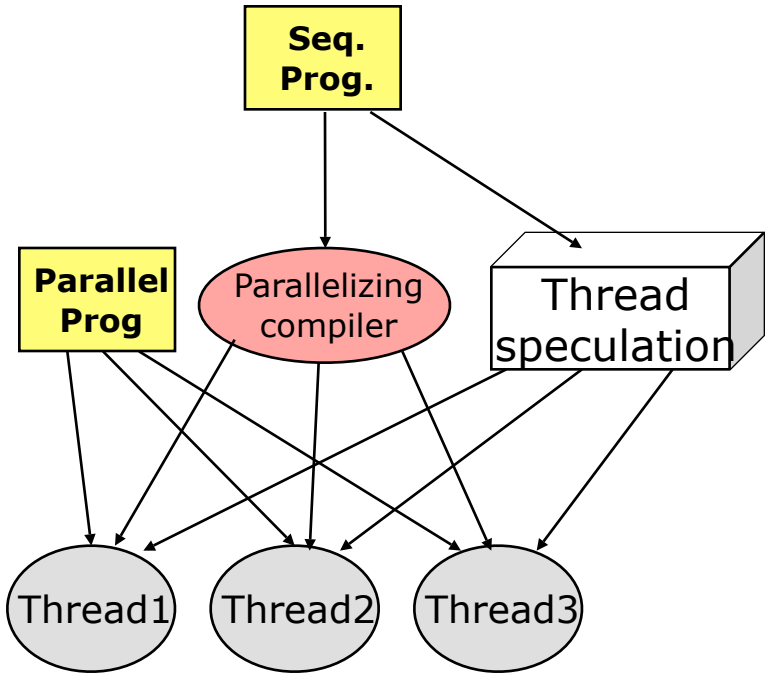
- Parallel programming will be used for all applications where performance is important
  - Parallel programming will become mainstream
  - Large efforts will (have to) be put into tools, programming environments etc

UPPSALA UNIVERSITET

How do we use the parallelism?

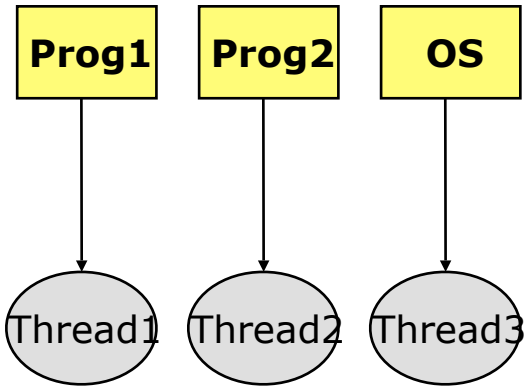


Capacity Computing

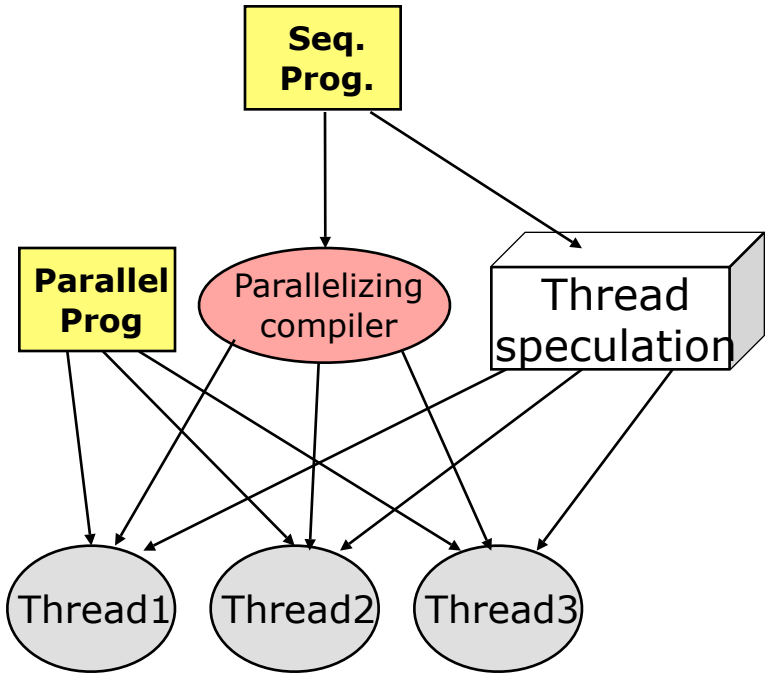


UPPSALA UNIVERSITET

How do we use the parallelism?



Capacity Computing



Capability Computing

UPPSALA UNIVERSITET

Some Thoughts ...

- Can you not just run different jobs/ processes on different threads for increased capacity?

  Can you identify enough independent jobs?
  2048 tasks? What about algorithm scalability?

  Also: Remember
  - Less cache/thread
  - Less memory bandwidth/thread

UPPSALA
UNIVERSITET

Parallel Programming

- Formulating parallelism (algorithms)
  - Humans
- Extracting parallelism
  - Humans and tools
- Expressing parallelism
  - Programming models and languages
- Exploiting parallelism
  - Compilers, runtime systems and hardware

UPPSALA UNIVERSITET

Intel: ”Tera-Scale Research”

Claims that the following research efforts are essential:

- Microprocessor
  - Multicore architectures
  - Specialized cores
  - Scalable interconnects
  - Energy efficient circuits
- Platform
  - Cache systems
  - Compact memory stacking
  - Virtualization/partitioning
  - Scalable operating systems
- Programming
  - Tools
  - Compilers and libraries
  - Data parallel operations
  - Transactional memory
  - Speculative multithreading

UPPSALA UNIVERSITET

More Thoughts …

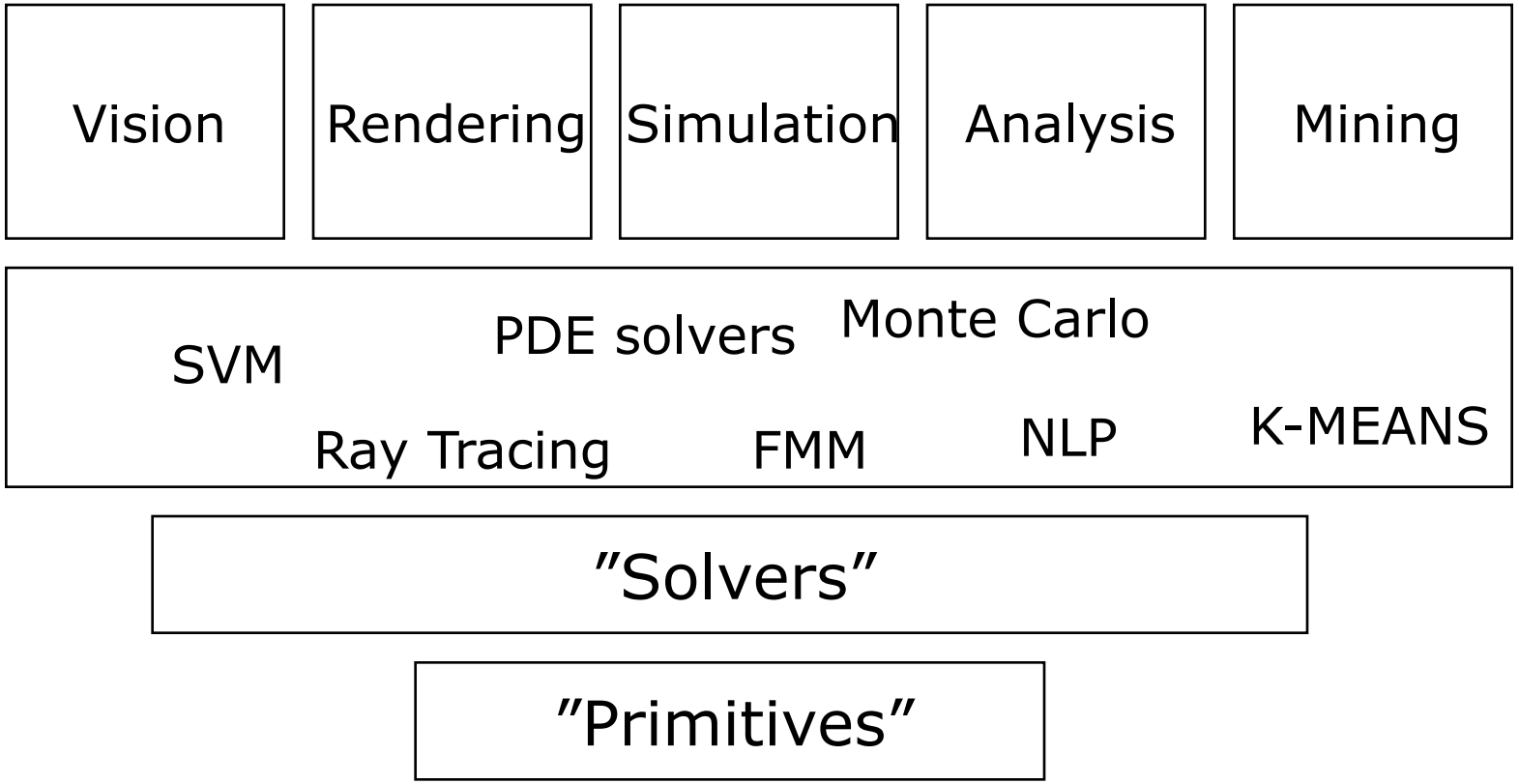
- Better tools for performance analysis and optimization are indeed needed
- Better tools for testing, verification and debugging of parallel codes are also needed

For CSE software:

- Autotuning for critical kernels
- Libraries, re-use of code and implementations becomes even more important
- Another motivation for lifting the level of abstraction

UPPSALA UNIVERSITET

Application Stack



It is easy to oversimplify in this type of graph!

UPPSALA
UNIVERSITET

One valid view (Geoffrey Fox):

We currently assume that the kernels of the scalable parallel algorithms/ applications/libraries will be built by experts with a broader group of programmers (mere mortals) composing library members into complete applications.

UPPSALA UNIVERSITET

Thoughts on Programming CMP systems

- Hybrid programming models will be used (Message passing + multithreading)
- What is really the right programming model?
- OpenMP? What about NUMA and data placement?
- What about languages like X10, PGAS etc?
- Can hardware be of assistance? (e.g. transactional memory)
- Scalability!? (2048 threads in a single-box server in 2008?)
  - Take care: What do you mean by scalability? The aim is to use the resources in the system ”as efficiently as possible”.

UPPSALA
UNIVERSITET

Impact on Algorithms

For performance, we need to understand the interaction between algorithms and architecture.

The rules have changed

We need to question old algorithms and results!

UPPSALA UNIVERSITET

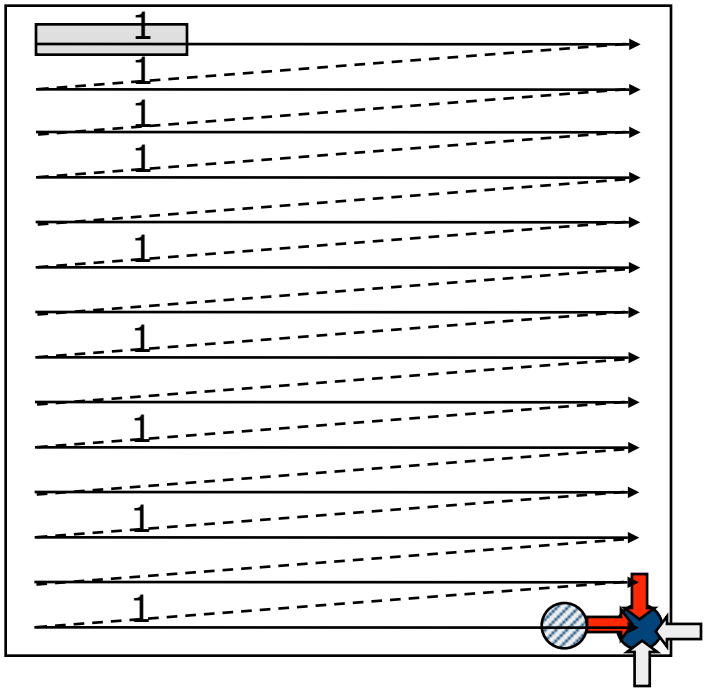
Criteria for algorithm design

- Pre-CMP:
  - Communication is expensive: Minimize communication
  - Data locality is important
  - Maximize scalability for large-scale applications
- Within a CMP chip today:
  - (On-chip) communication is almost to free
  - Data locality is even more important
  - (SMT may help by hiding some poor locality)
  - Scalability to 2-32 threads
- In a multi-CMP system tomorrow:
  - Communication is sometimes almost free (on-chip), sometimes (very) expensive (between chips)
  - Data locality (minimizing of-chip references) is a key to efficiency
  - “Hierarchical scalability”

UPPSALA
UNIVERSITET

Example: Multigrid on Multicore Systems

Natural Order Gauss-Seidel



= sweep path

= previous

= current

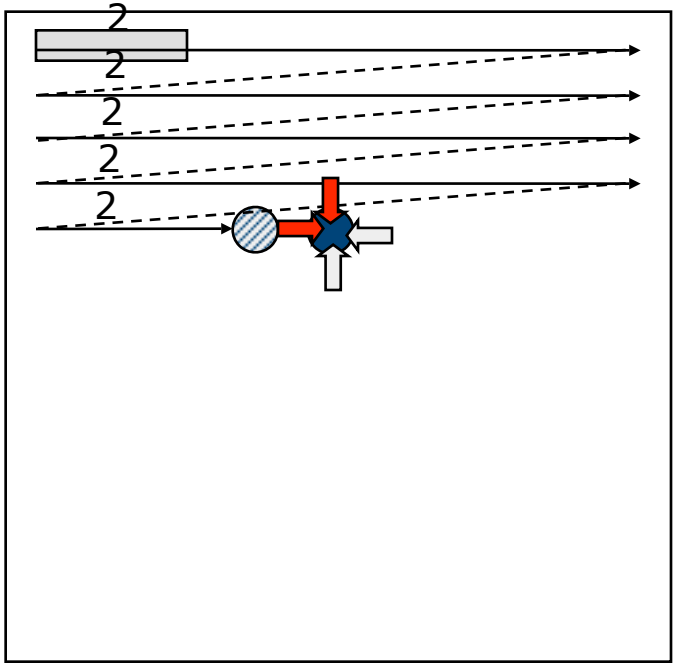
= data dependence

1,2,3,4 = iteration number

= cacheline layout

```
IF (convergence_test)
     <done>
else
     <iterate again>
```

Natural Order Gauss-Seidel



= sweep path

= previous

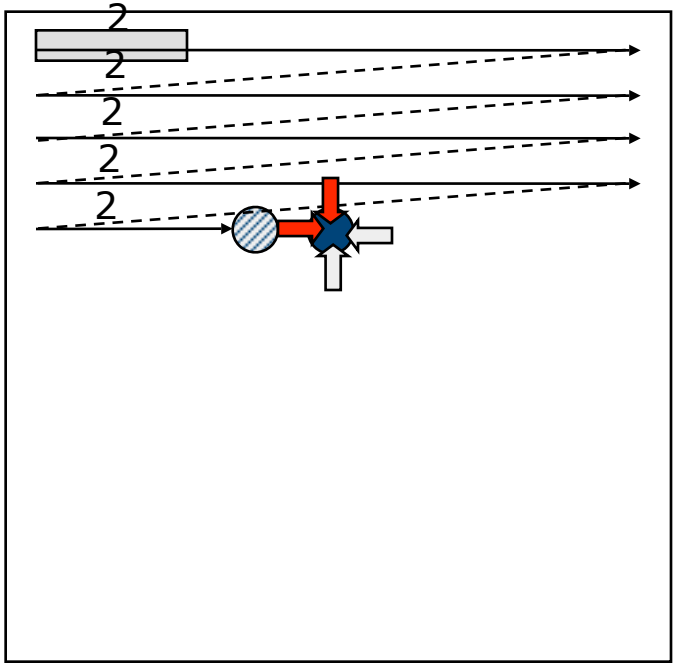
= current

= data dependence

1,2,3,4 = iteration number

= cacheline layout

Natural Order Gauss-Seidel



= sweep path

= previous

= current

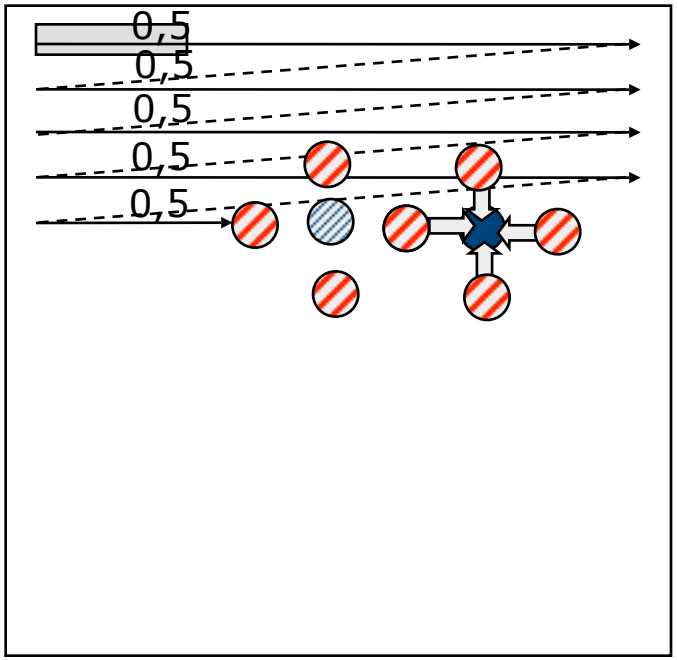
= data dependence

1,2,3,4 = iteration number

= cacheline layout

Data dependence ➔ Poor Parallelism☹

Red-Black Gauss-Seidel step 0,5: update the blacks



= sweep path

= previous

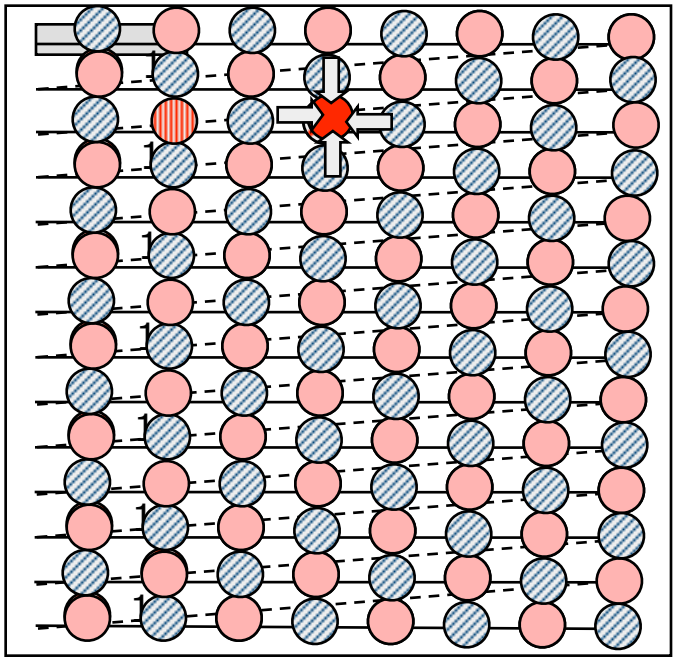
= current

= data dependence

1,2,3,4 = iteration number

= cacheline layout

Red-Black Gauss-Seidel step 1,0 update the reds



= sweep path

= previous

= current

= data dependence

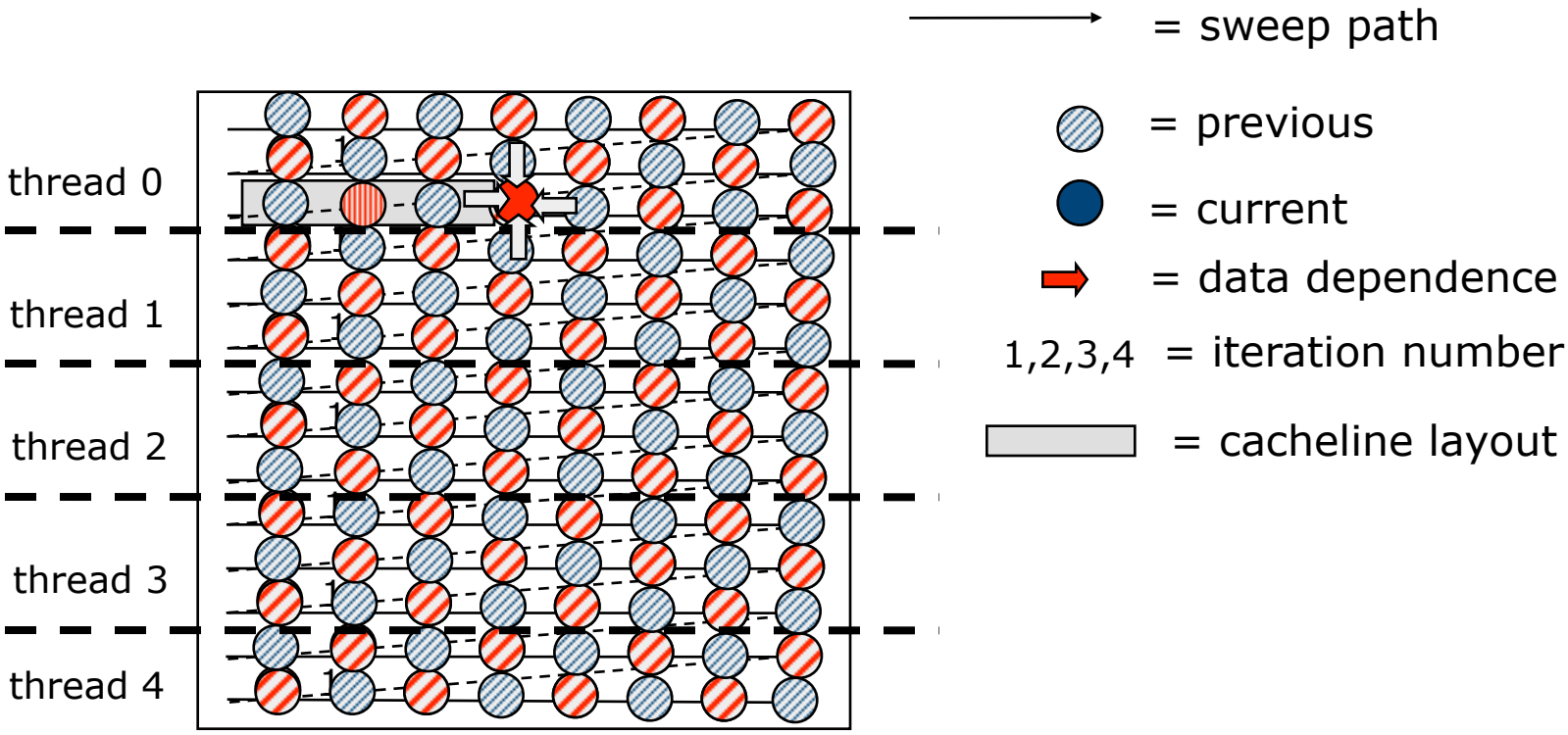
1,2,3,4 = iteration number

= cacheline layout

Update all blacks
<barrier>
Update all reds
<barrier>

➔ great parallelism!!!

Red-Black Gauss-Seidel Parallel version




```
IN PARALLELL {
        Update all blacks
        <barrier>
        Update all reds
        <barrier>
}
```

UPPSALA
UNIVERSITET

Hmm… Is this the way to do it on a multicore processor?

UPPSALA
UNIVERSITET

Hmm… Is this the way to do it on a multicore processor?

Hmm… Is this the way to do it on a multicore processor?

- Each element will be brought into the cache **twice** per iteration ☹

UPPSALA UNIVERSITET

Hmm… Is this the way to do it on a multicore processor?

- Each element will be brought into the cache **twice** per iteration ☹
- Quite poor data locality!

UPPSALA UNIVERSITET

Hmm… Is this the way to do it on a multicore processor?

- Each element will be brought into the cache **twice** per iteration ☹
- Quite poor data locality!
- Can we fix this?

UPPSALA UNIVERSITET

Conclusions

- The Multicore Era is here

- Impact on CSE? Multicore processors adds another level of complexity and change the rules for:
  - Cost of parallelism
  - Cache capacity per thread
  - Memory bandwidth per thread
  - Cost of thread communication