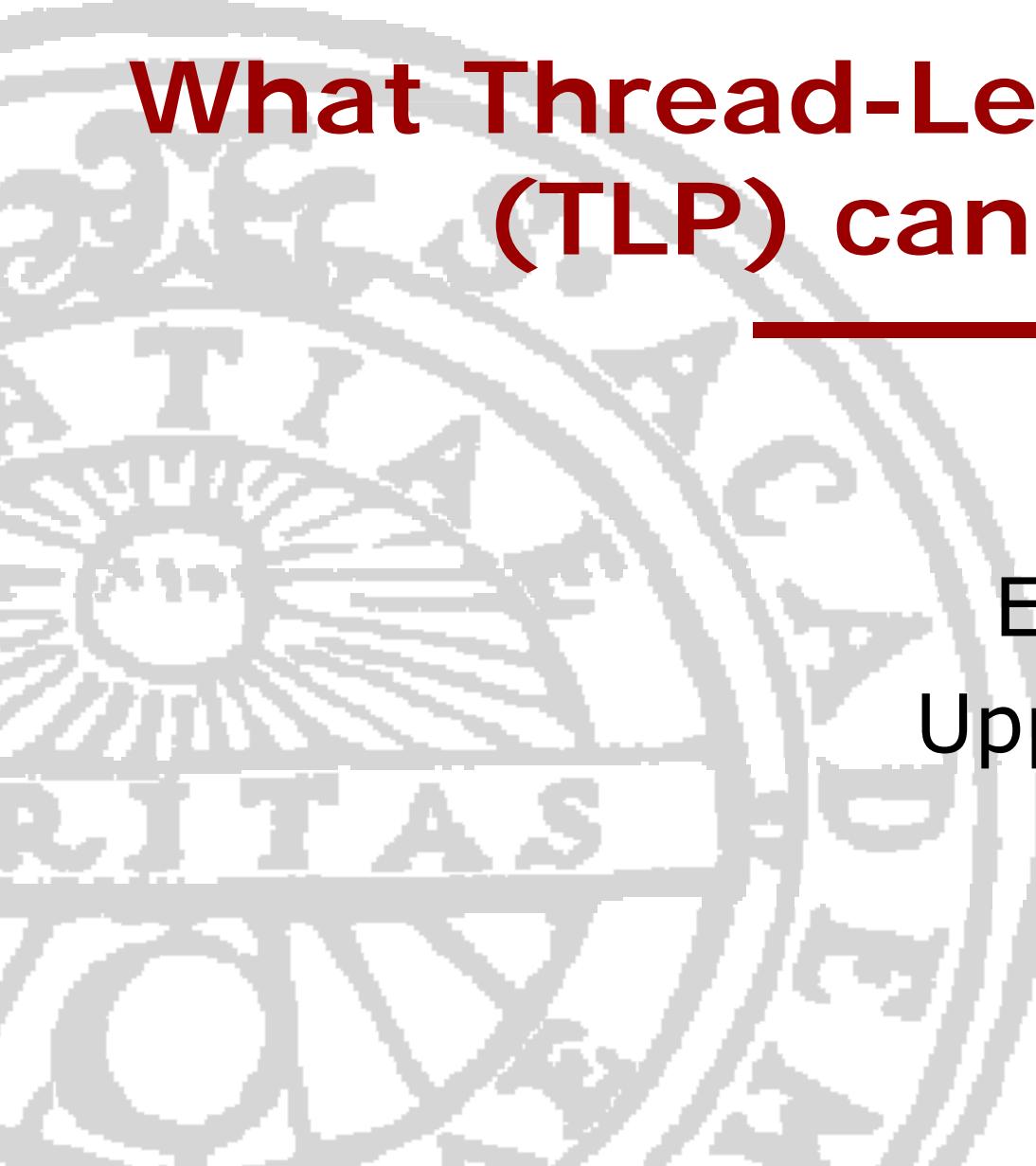
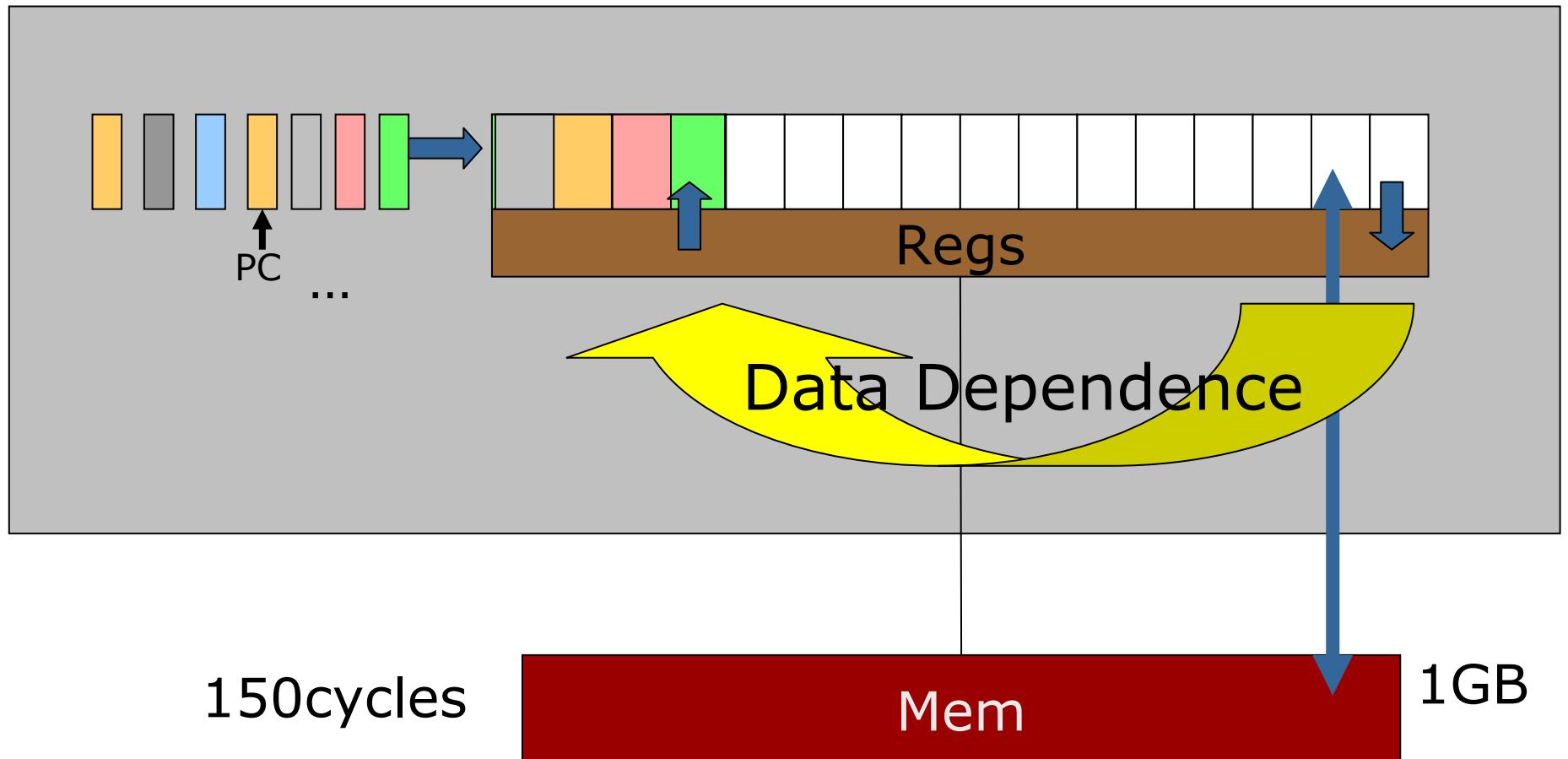


What Thread-Level Parallelism (TLP) can buy you



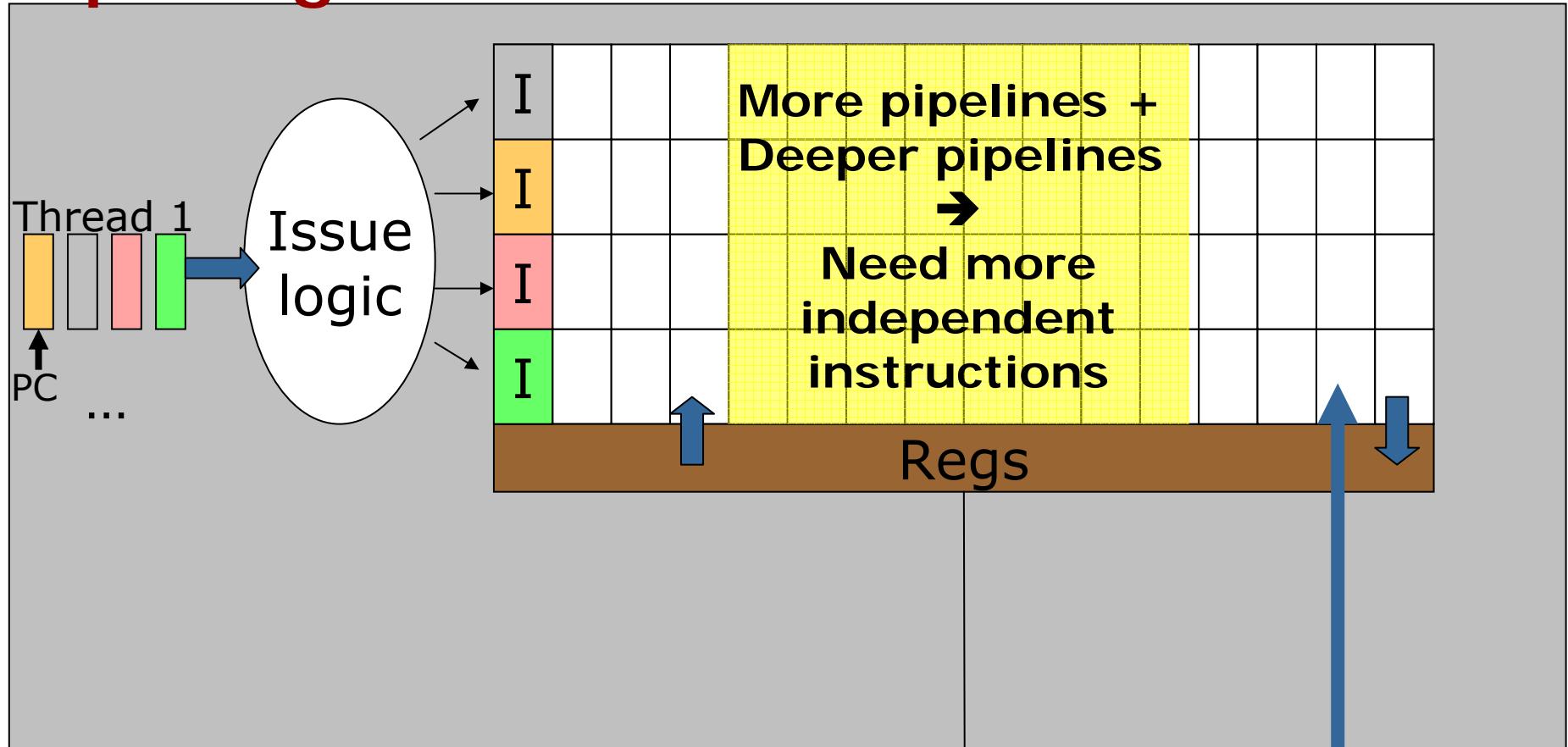
Erik Hagersten
Uppsala University
Sweden

Old Trend1: Deeper pipelines/higher freq. Exploring ILP (instruction-level parallelism)





Old Trend2: Wider pipelines Exploring more ILP

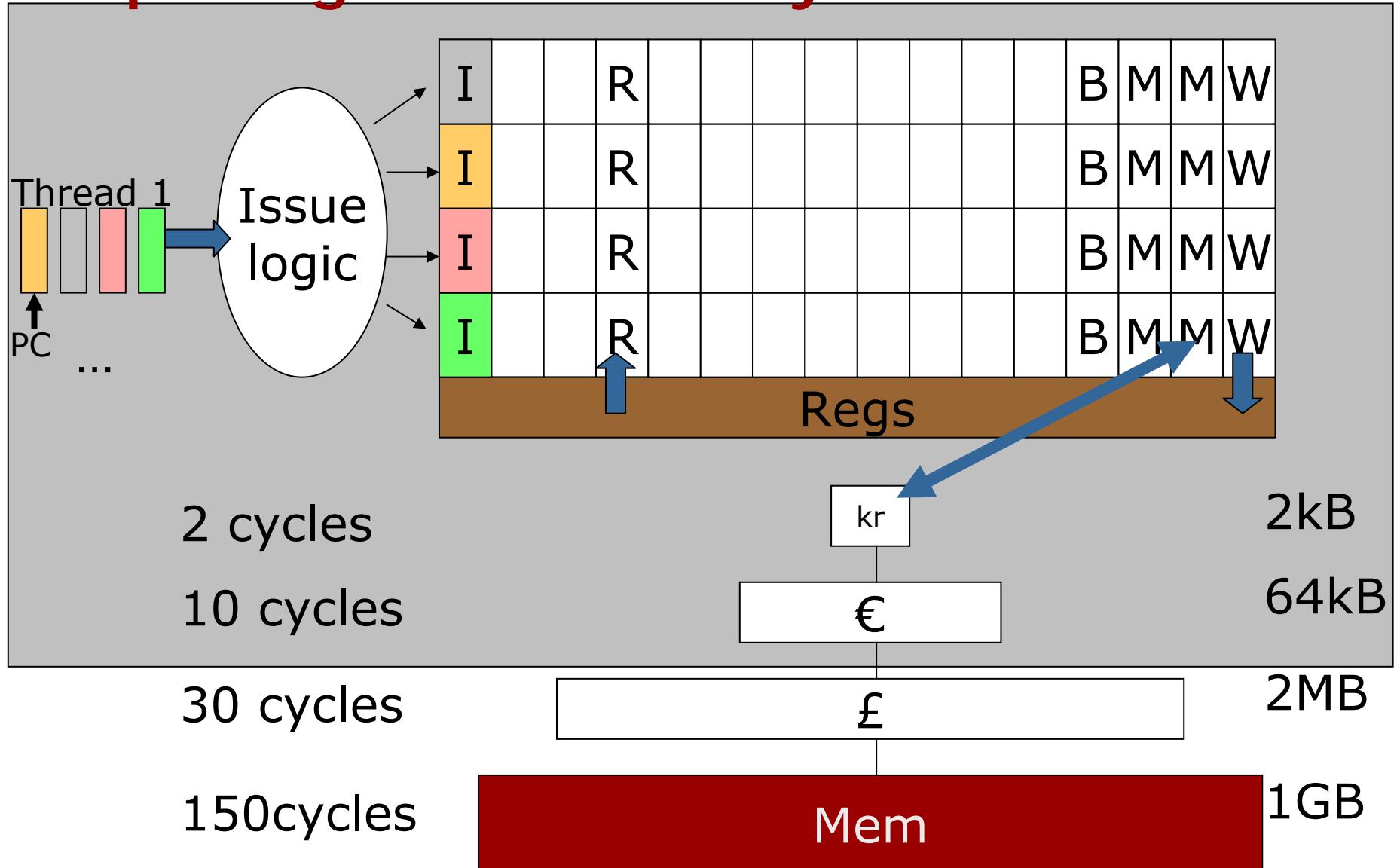


150cycles

Mem

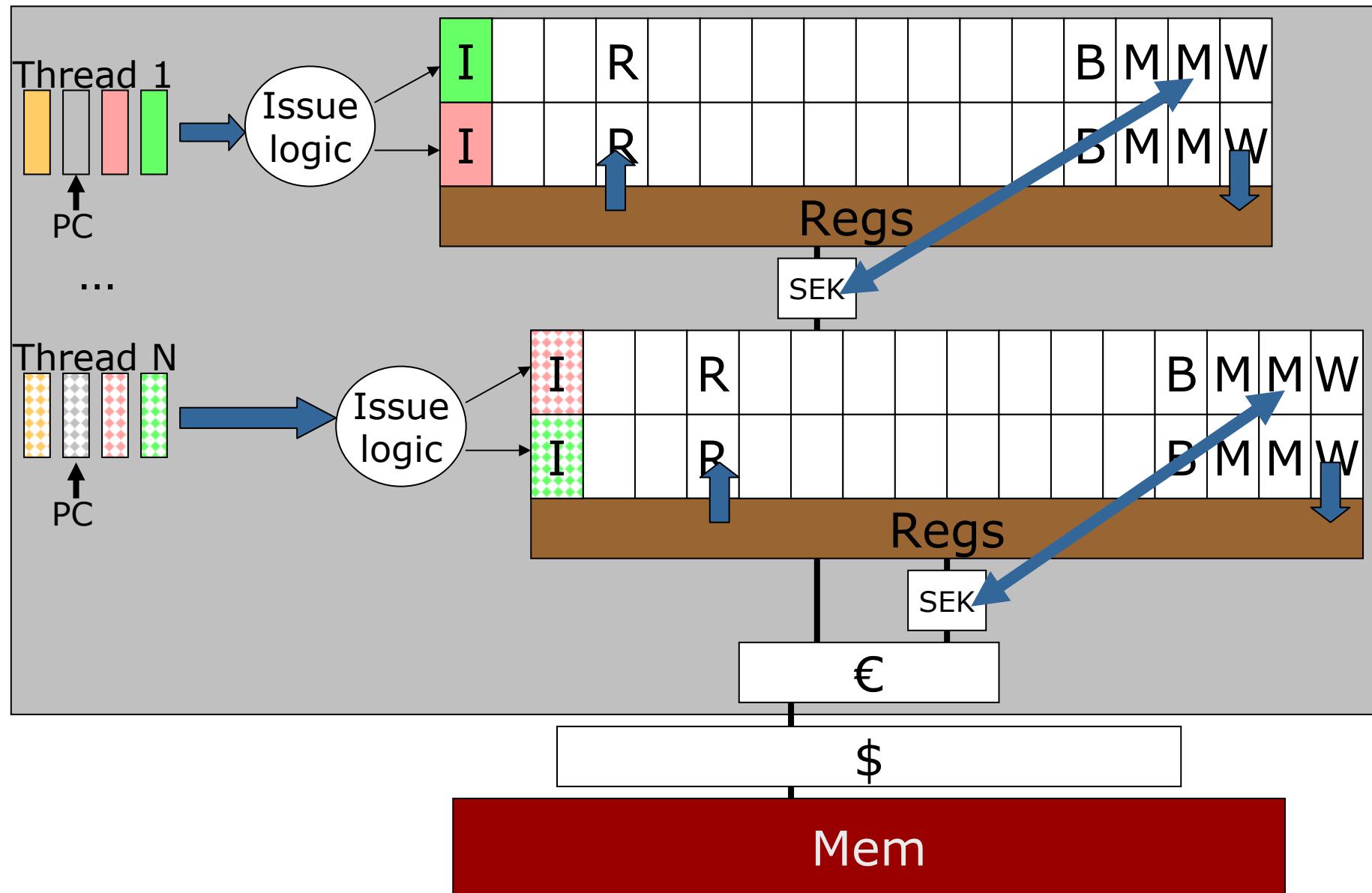
1GB

Old Trend3: Deeper memory hierarchy Exploring access locality



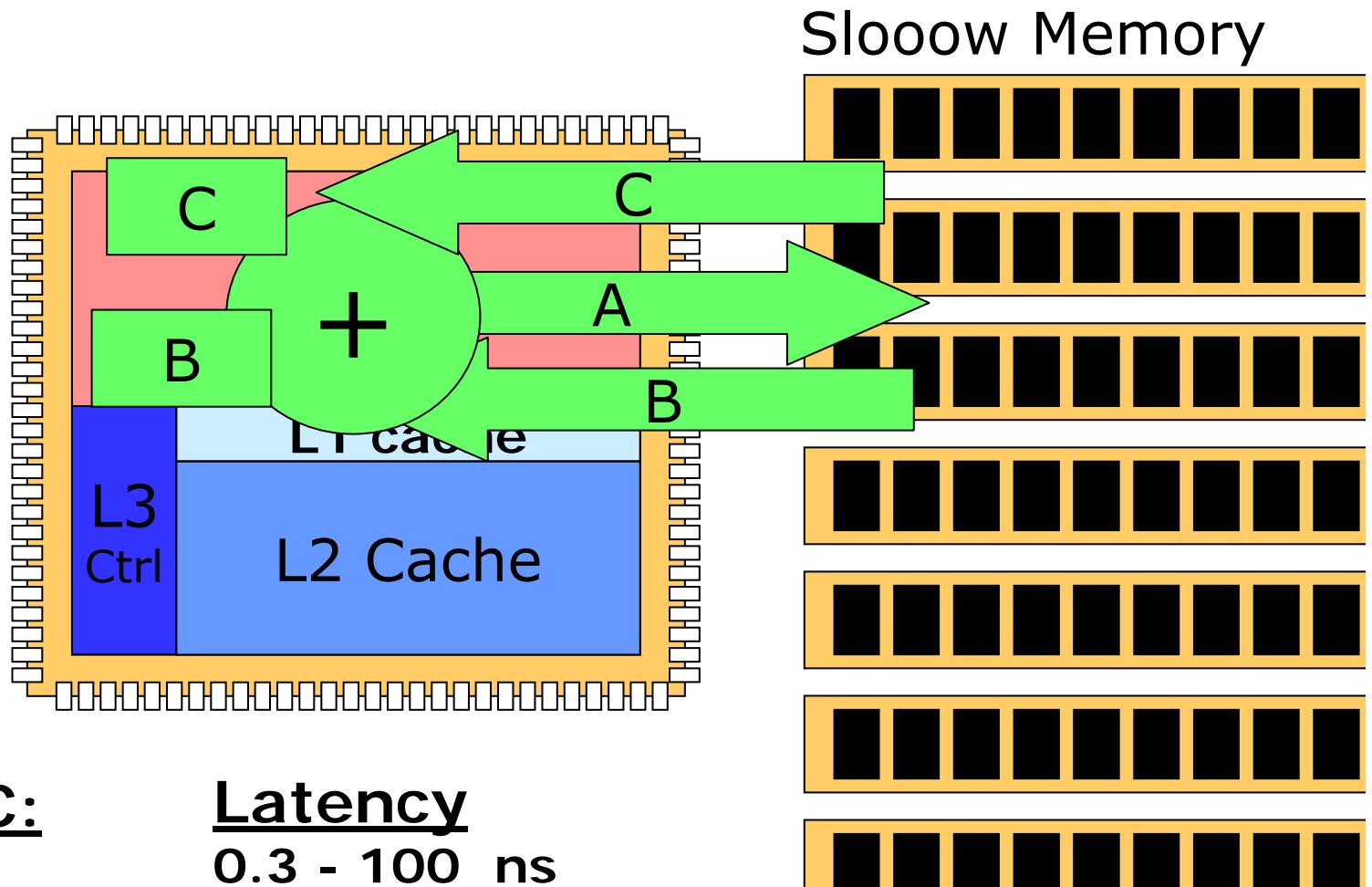


CMP: Chip Multiprocessor (aka Multicores) more TLP & geographical locality





Not enough MLP?



$A = B + C$:

Read B	0.3 - 100 ns
Read C	0.3 - 100 ns
Add B & C	0.3 ns
Write A	0.3 - 100 ns

Latency

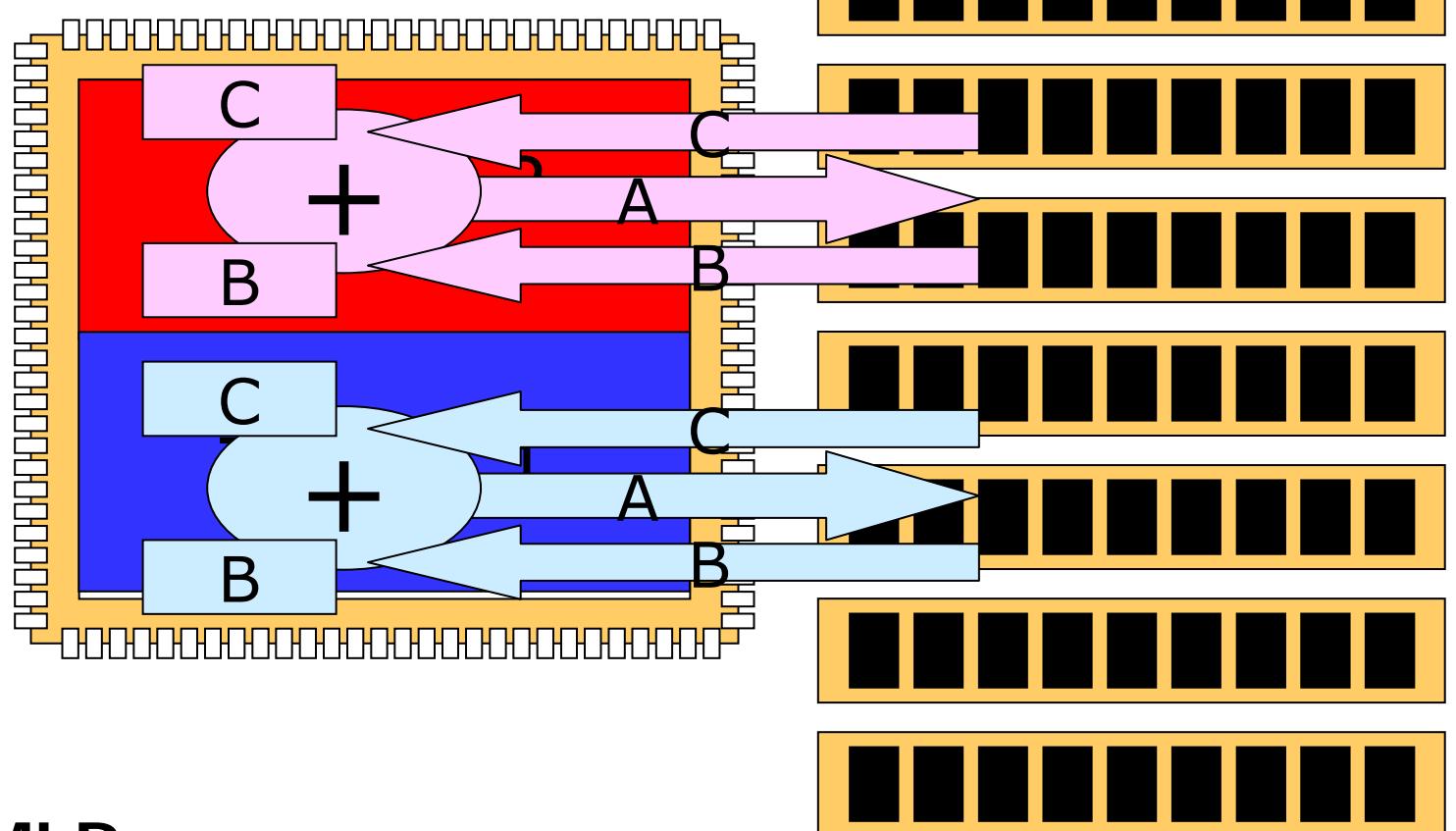
0.3 - 100 ns
0.3 - 100 ns
0.3 ns
0.3 - 100 ns



TLP → MLP

→ memory accesses from many threads (MLP)

Sloooow memory

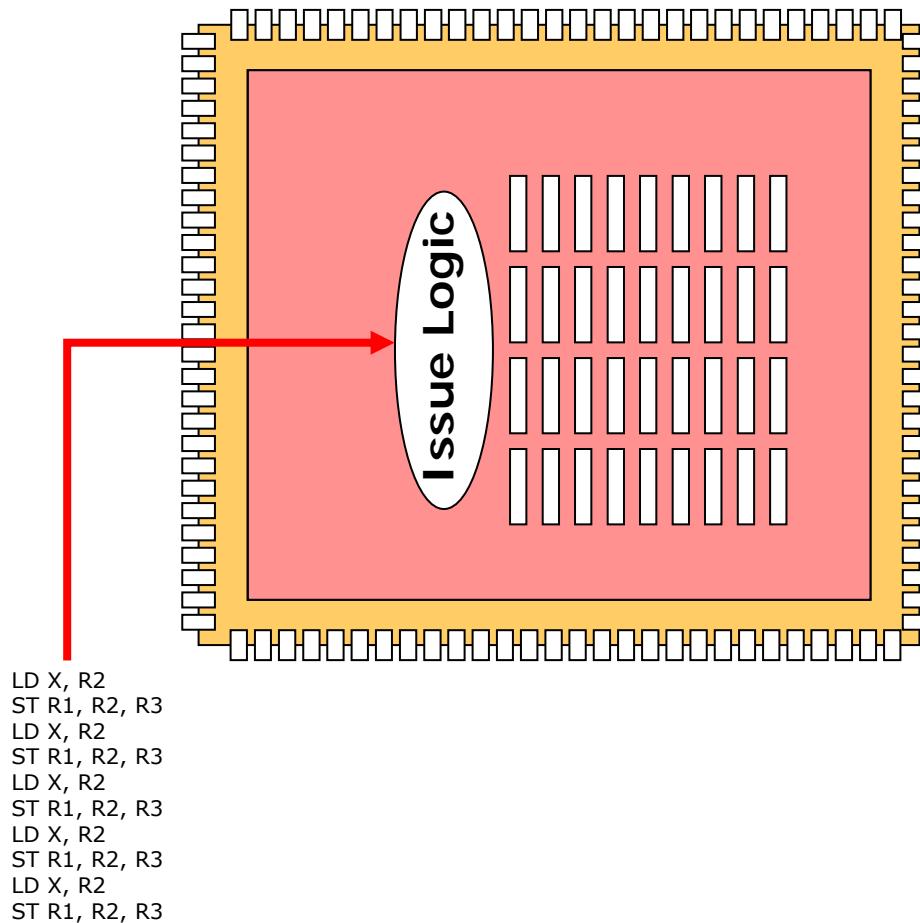


TLP → MLP

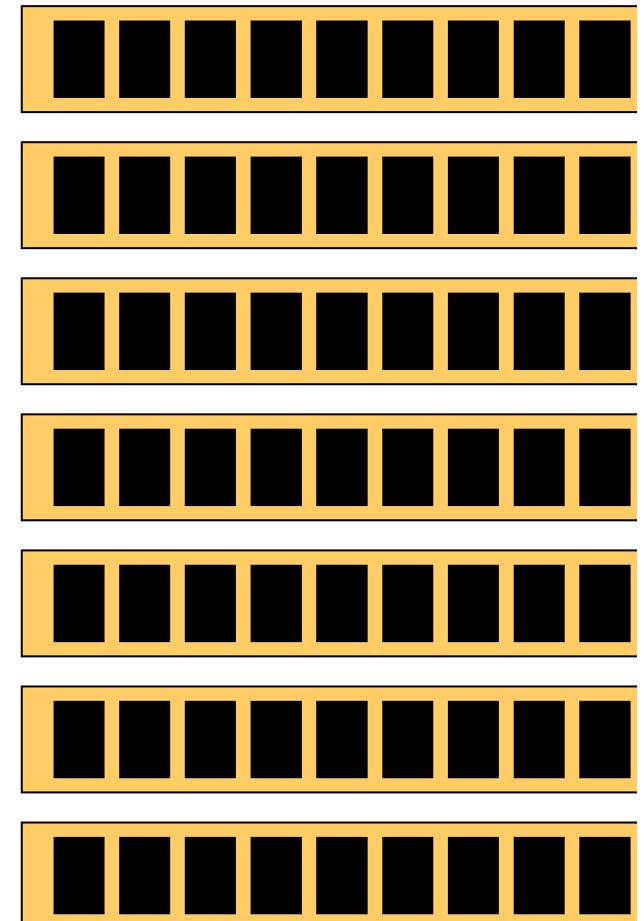
Thread-Level Parallelism → Memory-Level Parallelism (MLP)



Not enough ILP

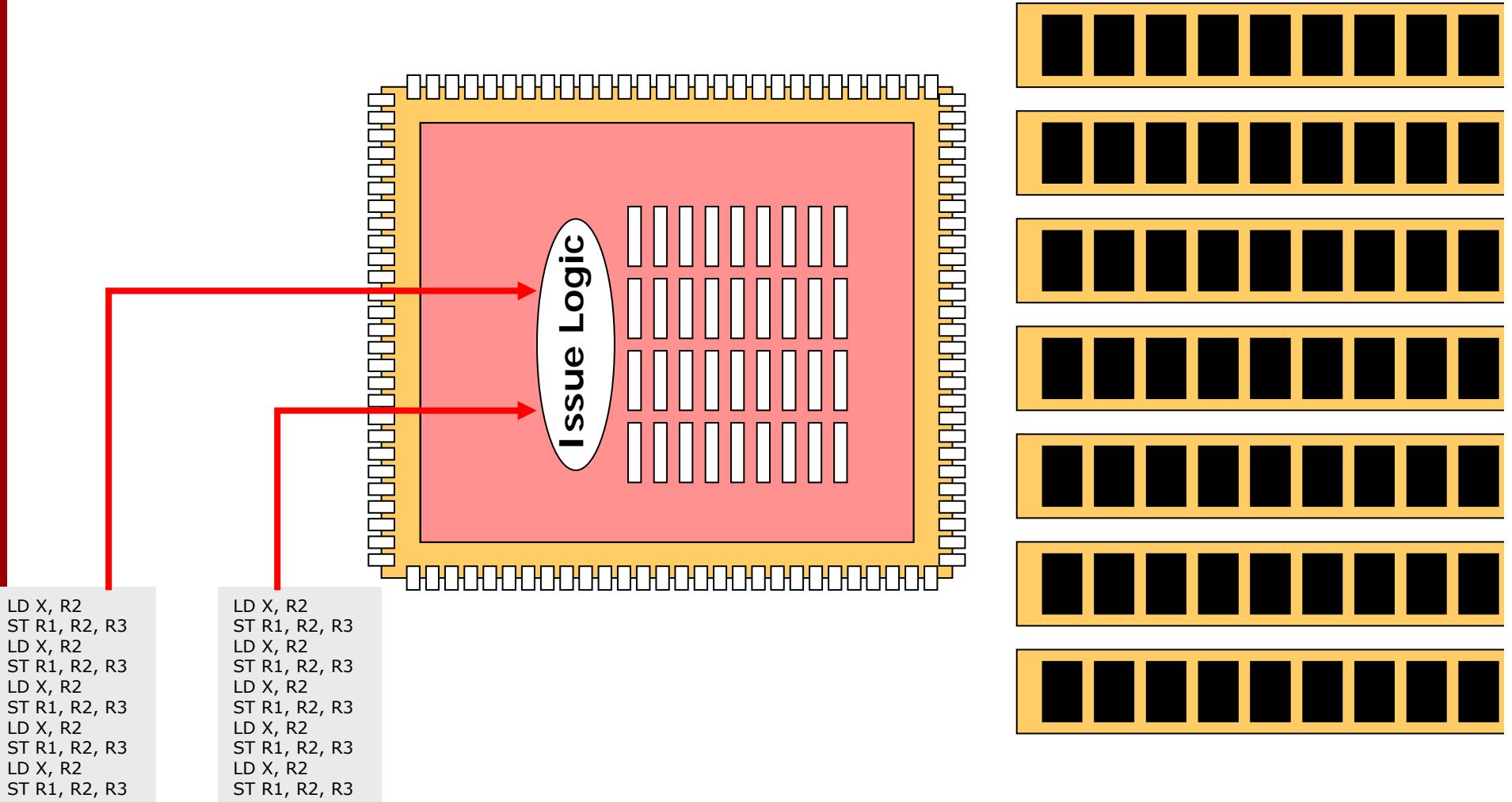


Slooow Memory



TLP → MLP

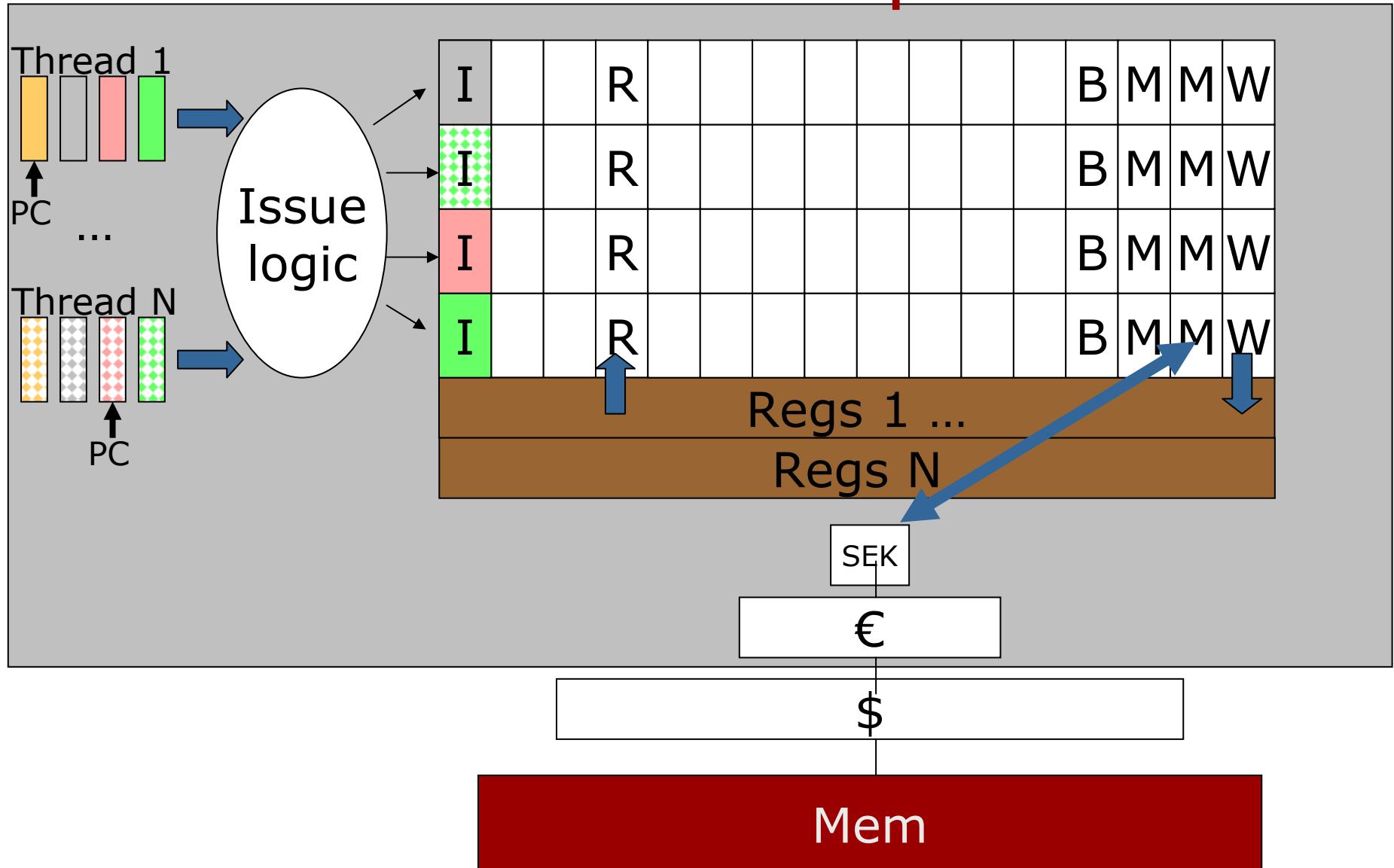
→ feed one superscalar with instr. from many threads
(also used in GPUs)





SMT: Simultaneous Multithreading

"Combine TLP&ILP to find independent instr."



Thread-interleaved

Historical Examples:

- Denelcor, HEP, Tera Computers [B. Smith] 1984

Each thread executes every n:th cycle in a round-robin fashion

- Poor single-thread performance
- Expensive (due to early adoption)

- Intel's "Hyperthreading" (2002)

- Poor implementation

Design Issues for Multicores

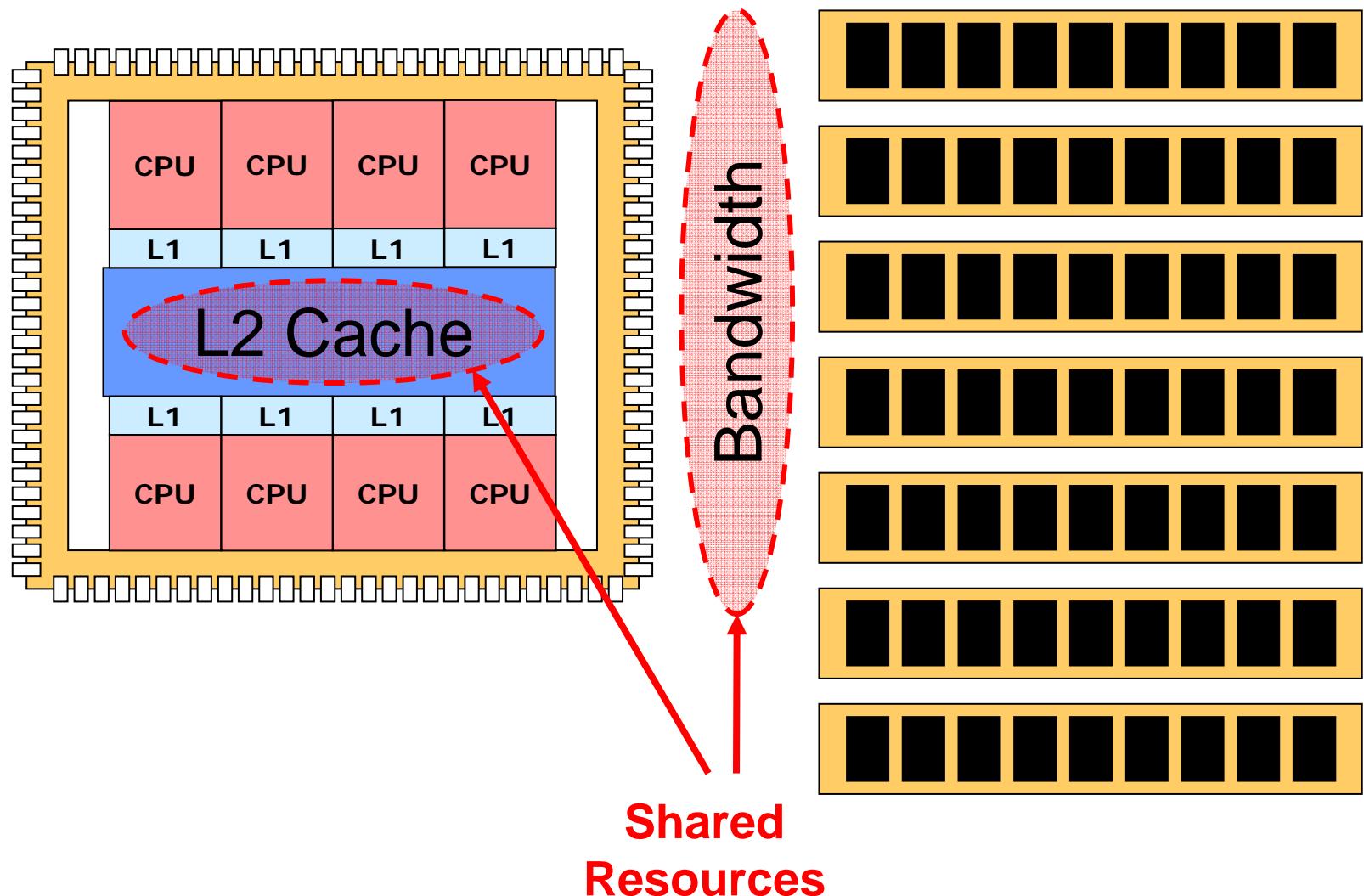
Erik Hagersten
Uppsala University
Sweden

CMP bottlenecks/points of optimization

- Performance per Watt?
- Performance per memory byte?
- Performance per bandwidth?
- Performance per \$?
- ...

- How large fraction of a CMP system cost is in the CPU chip?
- Should the execution (MIPS/FLOPS) be viewed as a scarce resource?

Shared Bottlenecks



Capacity or Capability Computing?

Capacity? (\approx several sequential jobs)

or

Capability? (\approx one parallel job)

Issues:

- ✿ Memory requirement?
- ✿ Sharing in cache?
- ✿ Memory bandwidth requirement?

Memory: the major cost of a CMP system!

How do we utilize it the best?

- ✿ Once the workingset is in memory, work like crazy!

➔ Capability computing suits CMPs the best
(in general)

A Few Fat or Many Narrow Cores?

- Fat:
 - Fewer cores but...
 - wide issue?
 - O-O-O?
- Narrow: More cores but...
 - narrow issue?
 - in-order?
 - have you ever heard of Amdahl?
 - SMT, run-ahead, execute-ahead ... to cure shortcomings?

Read:

Maximizing CMP Throughput with Mediocre Cores
Davis, Laudon and Olukotun, PACT 2006

Amdahl's Law in the Multicore Era

Mark Hill, IEEE Computer, July 2008

http://www.cs.wisc.edu/multifacet/papers/ieeecomputer08_amdahl_multicore.pdf

Cores vs. caches

- Depends on your target applications...
- Niagara's answer: go for cores
 - ✿ In-order 5-stage pipeline
 - ✿ 8 cores a' 4 SMT threads each → 32 threads,
 - ✿ 3MB shared L2 cache (96 kB/thread)
 - ✿ SMT to hide memory latency
 - ✿ Memory bandwidth: 25 GB/s
 - ✿ Will this approach scale with technology?
- Others: go for cache
 - ✿ 2-4 cores for now

How to Hide Memory Latency (and create MLP)

Options:

- O-O-O
- HW prefetching
- SMT
- Run-ahead/Execute-ahead

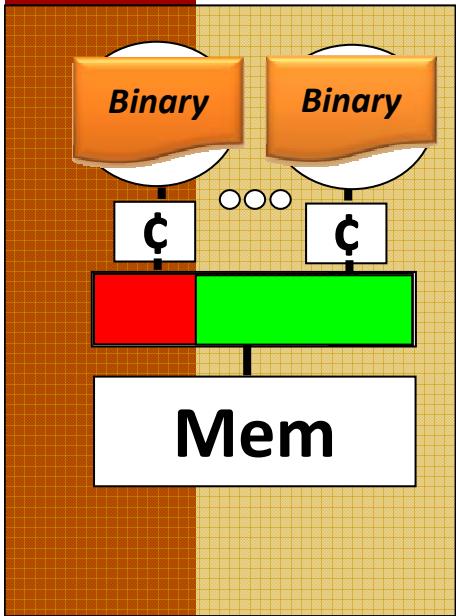


Handling shared resources

Erik Hagersten
Uppsala University
Sweden



Fighting for shared resources



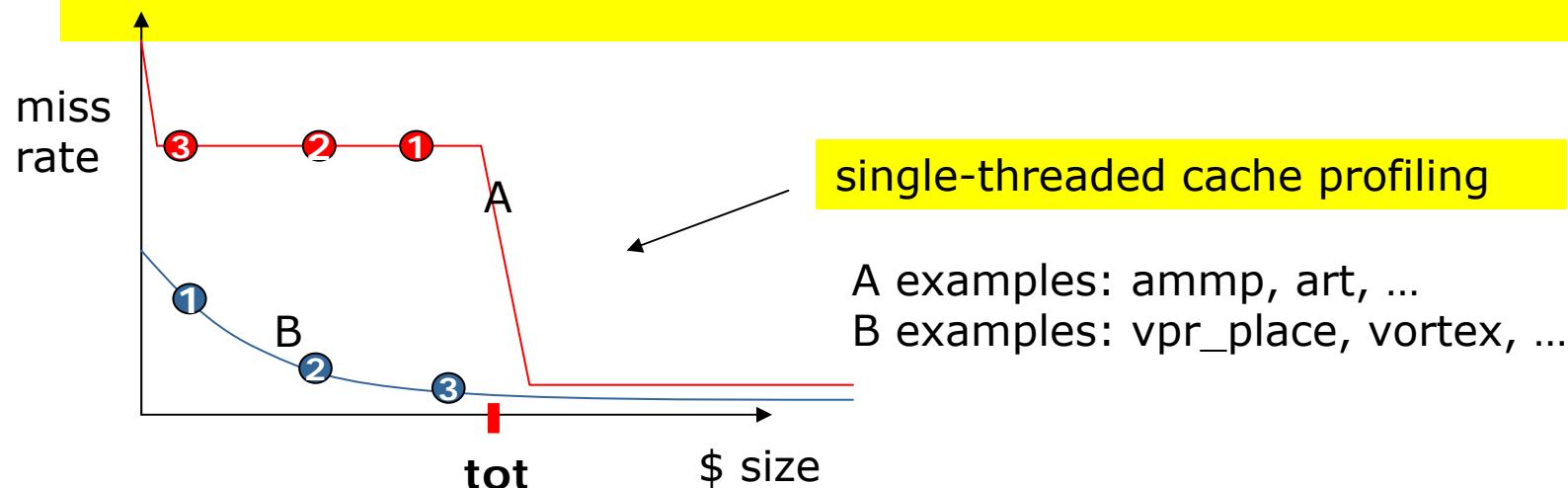
1st Order MC Performance Problems

- Additional multicore issues:
 - Even less cache resources per application
 - Sharing of cache resources
 - Wasted cache usage

Cache Interference in Shared Cache

■ Cache sharing strategies:

1. Fight it out!
2. Fair share: 50% of the cache each
3. Maximize throughput: who will benefit the most?



Read:

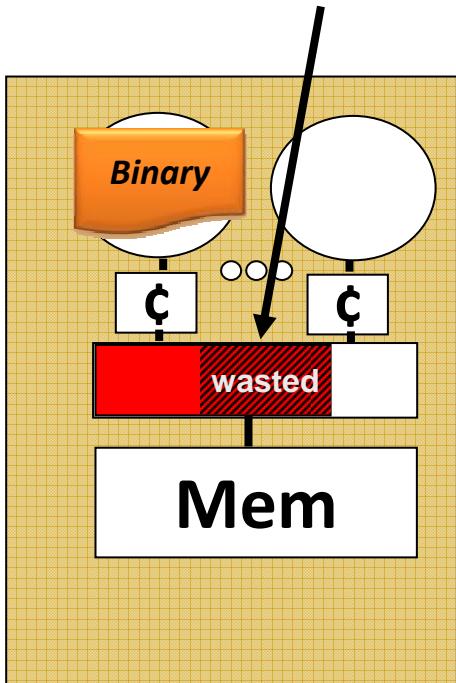
STATSHARE: A Statistical Model for Managing Cache Share via Decay
Pavlos Petoumenos et al in MOBS workshop ISCA 2006

Predicting the inter-thread cache contention on a CMP
Chandra et al in HPCA 2005



Andreas' research: Using Prefetch-NT

Avoid cache pollution
using existing x86
ISA:
NT = Non-temporal



AMD, Prefetch NT:

- Install in L1 cache with NT bit set
- Non-inclusive caching → Not in L2, L3
- Upon eviction from L1, do not install in L2, L3
(if NT is not set, cacheline will get installed)

Intel Core2, Prefetch NT:

- Install in L1&L2 caches (inclusive caching)
- Put in MRU place in L2 → replaced more easily
- Upon eviction from L1, keep in L2

Intel i7, Prefetch NT:

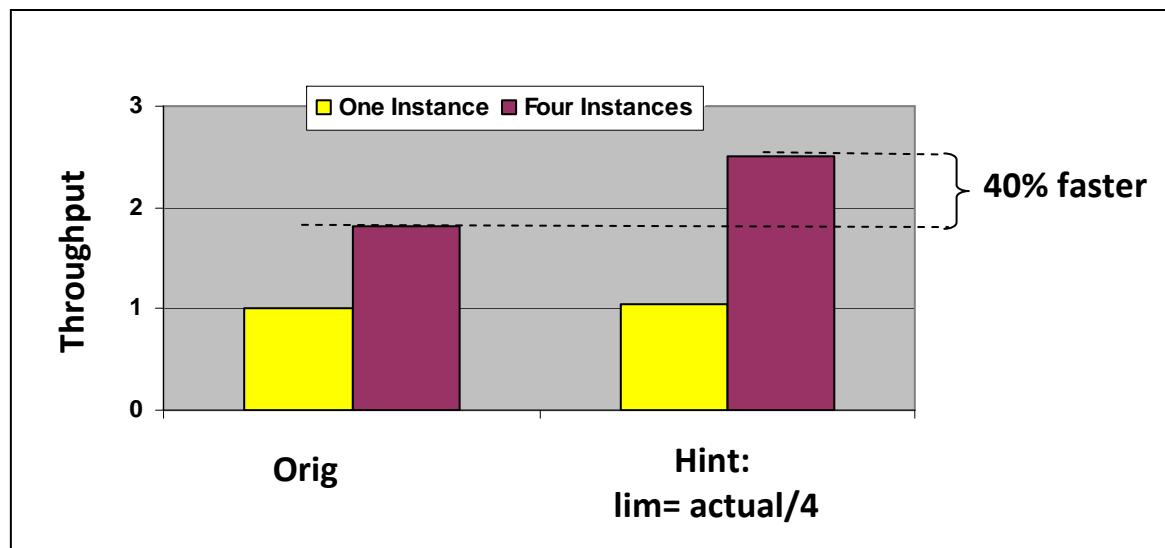
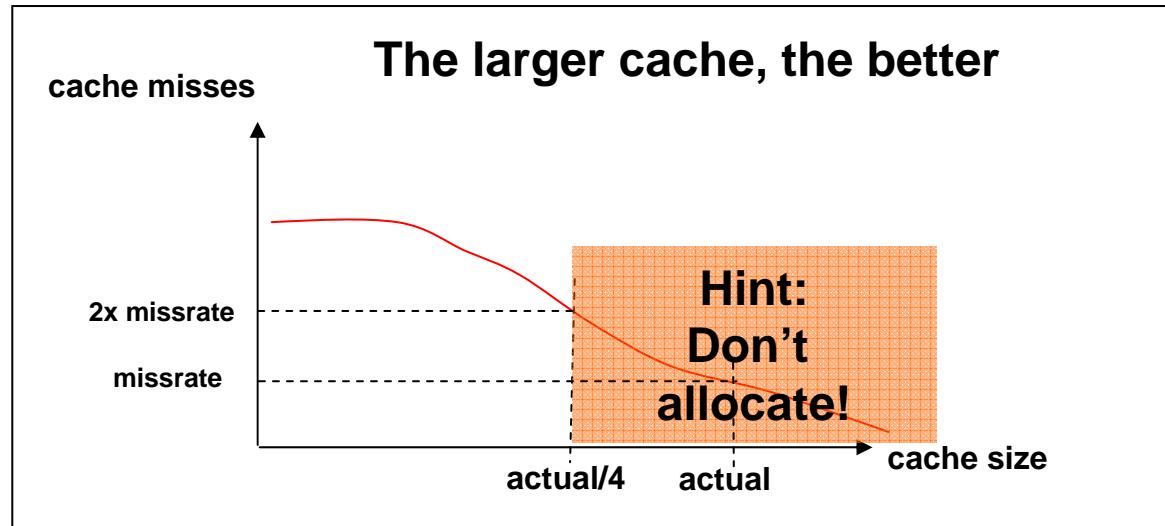
- Install in L1&L2 caches (inclusive caching)
- Put in MRU place in L2 → replaced more easily
- Upon eviction from L1, keep in L2

All, Store-NT:

- Keep cacheline in a special write-buffer
- When all bytes of the cache line has been updated write to memory while bypassing caches
- Huge penalty in not all bytes are updated

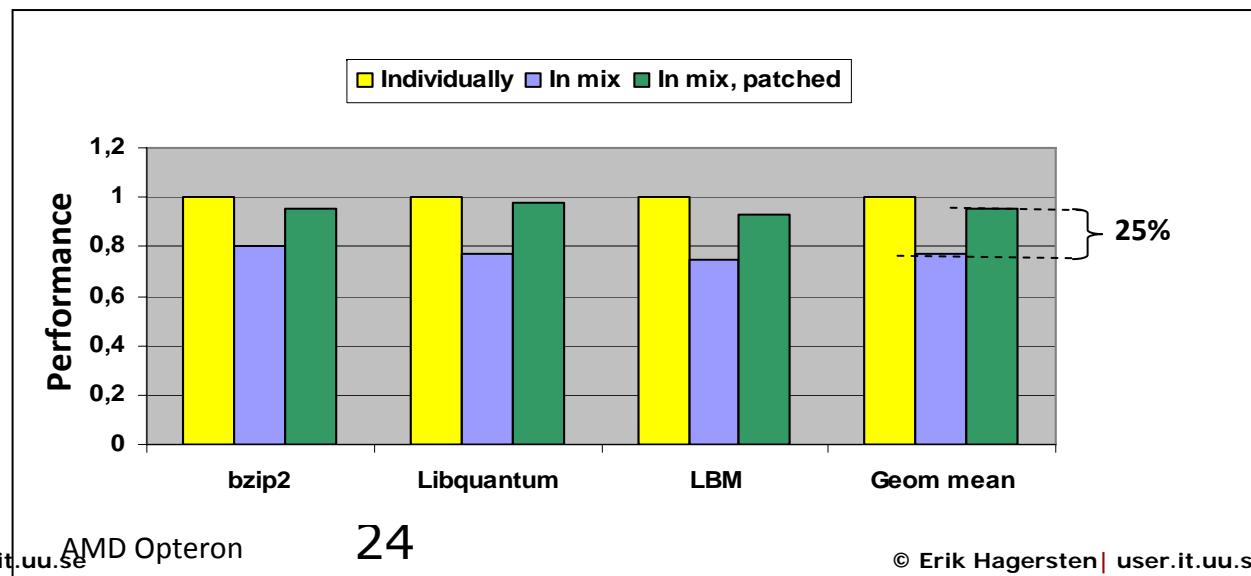
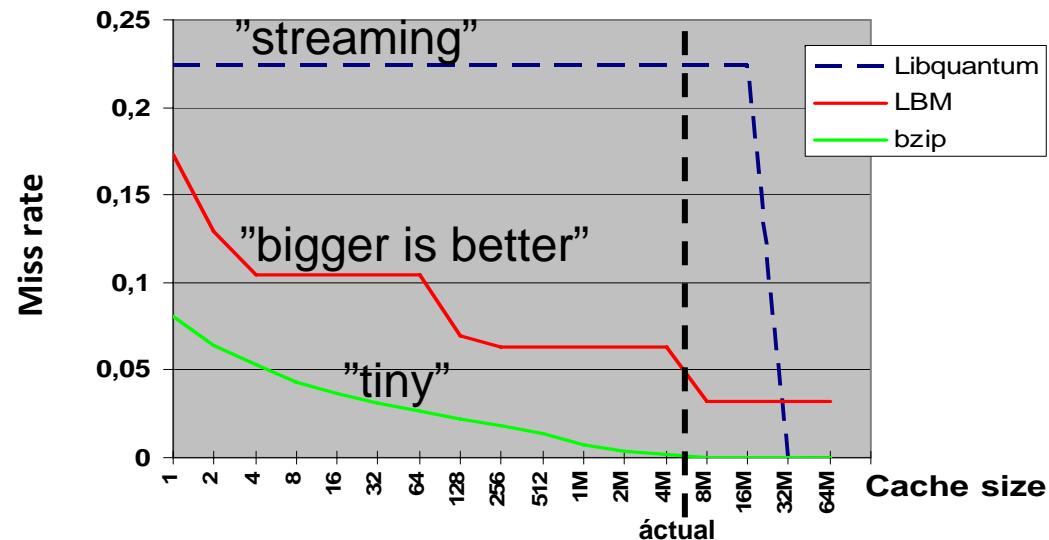


Example: Hints to avoid cache pollution (non-temporal prefetchedes)





Example: Hints for mixed workloads (non-temporal prefetches)





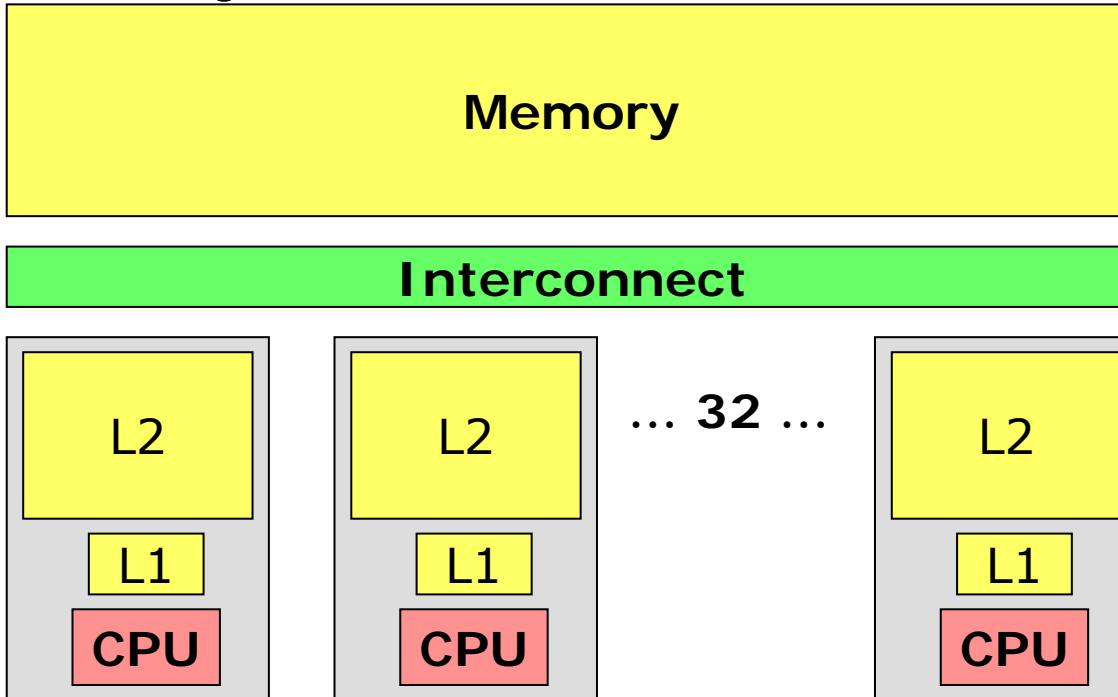
Wrapping up about multicores

Erik Hagersten
Uppsala Universitet

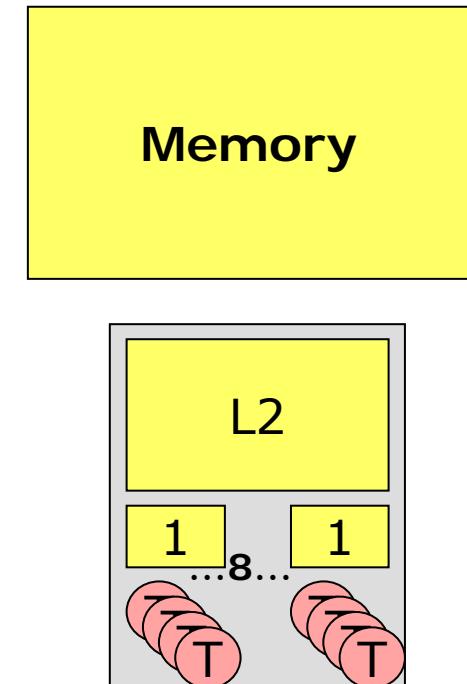


Looks and Smells Like an SMP (aka UMA)?

SMP system



Multicore system



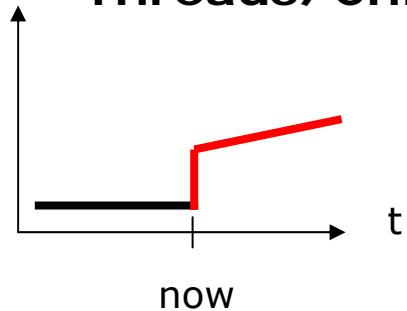
Well, how about:

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

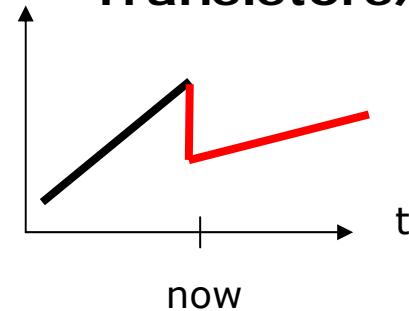


Trends (my guess!)

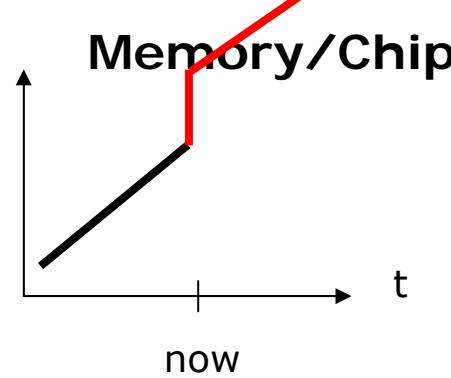
Threads/Chip



Transistors/Thread



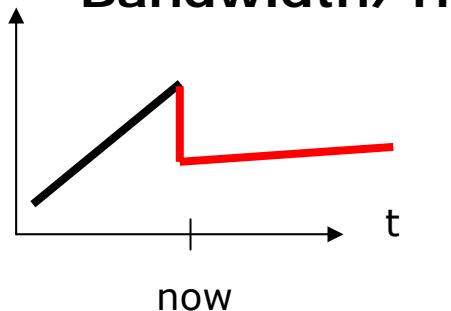
Memory/Chip



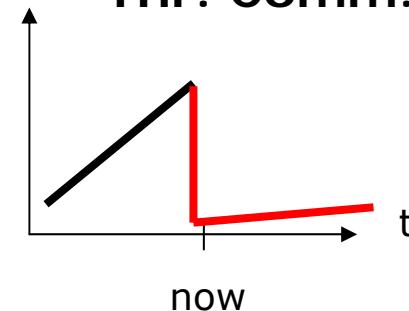
Cache/Thread



Bandwidth/Thread



Thr. Comm. Cost (temporal)

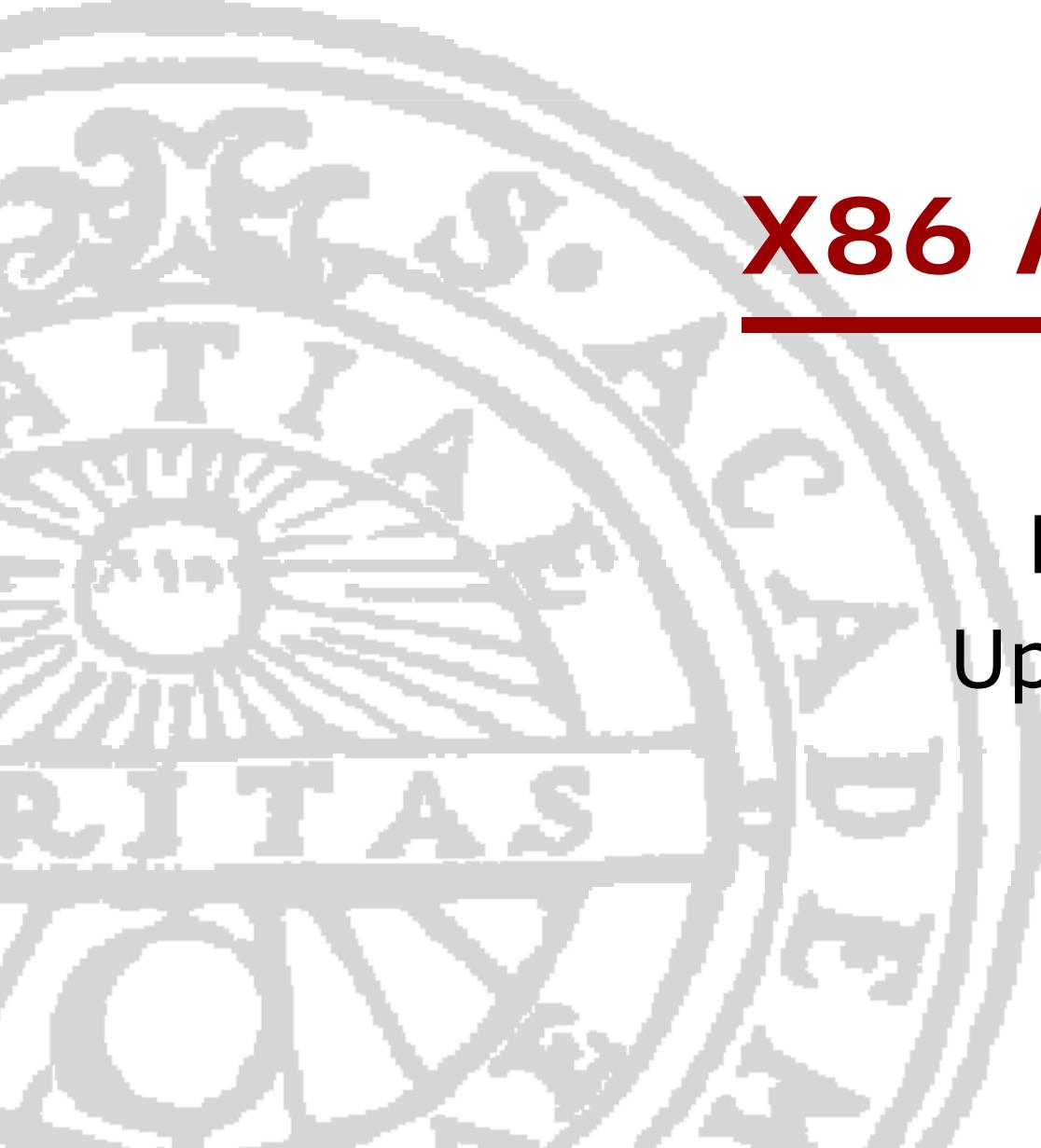


What matters for multicore performance?

- Are we buying...
 - ✿ CPU frequency?
 - ✿ Number of cores?
 - ✿ MIPS and FLOPS?
 - ✿ Memory bandwidth?
 - ✿ Cache capacity?
 - ✿ Memory capacity?
 - ✿ Performance/Watt?

MC Questions for the Future

- How to get parallelism?
- How to get performance/data locality?
- How to debug?
- A case for new funky languages?
- A case for automatic parallelization?
- Are we buying:
 - ✿ compute power,
 - ✿ memory capacity, or
 - ✿ memory bandwidth?
- Will 128 cores be mainstream in 5 years?
- Will the CPU market diverge into desktop/capacity/capability/special-purpose CPUs again?
- **A non-question: will it happen?**



X86 Architecture

Erik Hagersten
Uppsala University
Sweden

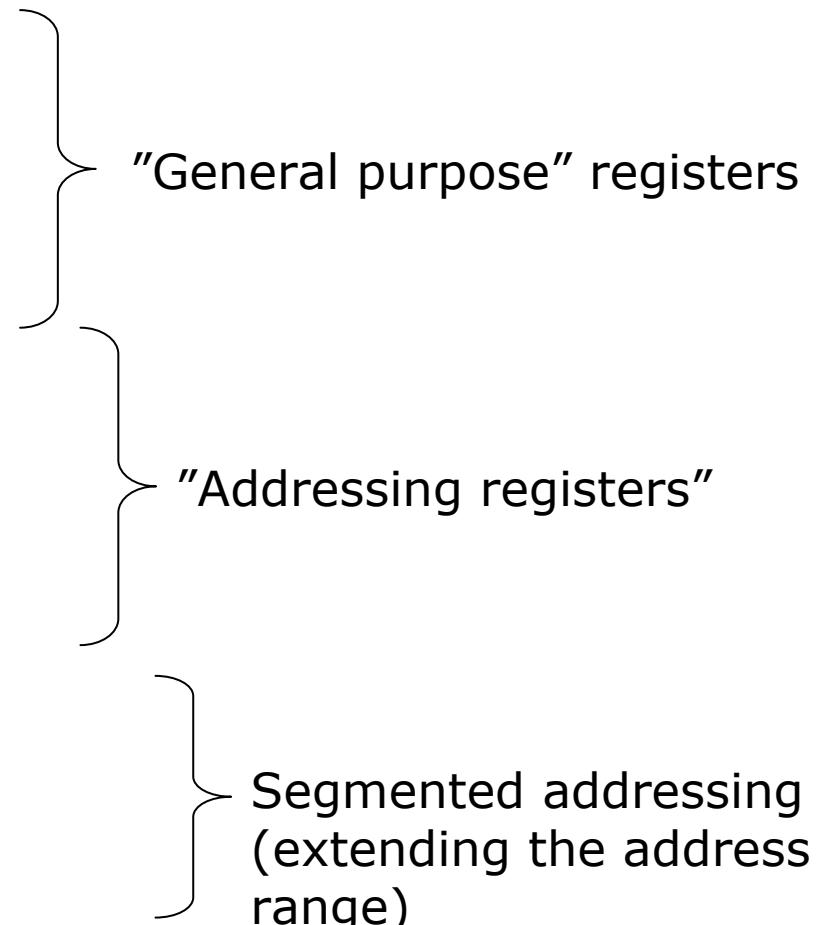
Intel Archeology

- (8080: 1974, 6.0 kTransistors, 2MHz, 8bit)
- 8086: 1978, 29 kT, 5-10MHz, 16bit (PC!)
- (80186:1982 ? kT, 4-40MHz, integration!)
- 80286: 1982, 0.1MT, 6-25MHz, chipset (PC-AT)
- 80386: 1985, 0.3MT, 16-33MHz, 32 bits
- 80486: 1989, 1.2MT, 25-50MHz, I&D\$, FPU
- Pentium: 1993, 3.1 MT, 66 MHz, superscalar
- Pentium Pro: 1997, 5.5 MT, 200 MHz, O-O-O, 3-way superscalar
- Intel Pentium4:2001, 42 MT, 1.5 GHz, Super-pipe, L2\$ on-chip
- ...



8086 registers

- AX (Accumulator)
- BX (Base)
- CX (Count)
- DX (Data)
- SP (Stack ptr)
- BP (Base ptr)
- SI (Source index)
- DI (Destination index)
- CS (Code segment)
- DS (Data segment)
- SS (Stack segment)



Complex instructions of x86

- RISC (Reduced Instruction Set Computer)
 - ✿ LD/ST with a limited set of address modes
 - ✿ ALU instructions (a minimum)
 - ✿ Many general purpose registers
 - ✿ Simplifications (e.g., read R0 returns the value 0)
 - ✿ Simpler ISA → more efficient implementations
- x86 CISC (Complex Instruction Set Computer)
 - ✿ ALU/Memory in the same instruction
 - ✿ Complicated instructions
 - ✿ Few specialized registers (actually accumulator architecture)
 - ✿ Variable instruction length
 - ✿ x86 was lagging in performance to RISC in the 90s

Micro-ops

- Newer pipelines implements RISC-ish μ-ops.
- Some complex x86 instructions expanded to several micro-ops at runtime.
- The translated μ-ops may be cached in a trace-cache [in their predicted order] (first: Pentium4)
- Expanded to “loop cache” in Core-2



x86-64

- ISA extension to x86 (by AMD 2001)
- 64-bit virtual address space
- 64-bit GP registers x16
 - x86's regs extended: rax, rbx, rcx, rdx, rbp, rsp, rsi, rdi
 - x86-64 also has: r8, r9, ... r15 (i.e., a total of 16 regs)
 - NOTE: dynamic register renaming makes the effective number of regs higher
- SSE n : 16 128-bit SSE "vector" registers
- Backwards compatible with x86

Intel adoptions: IA-32e, EM64T, Intel64

NOTE: IA-64 is Itanium



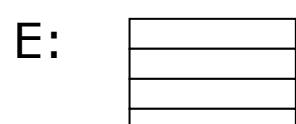
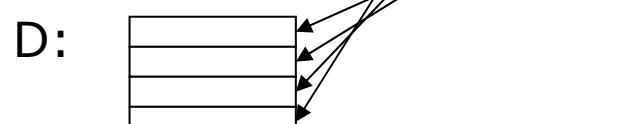
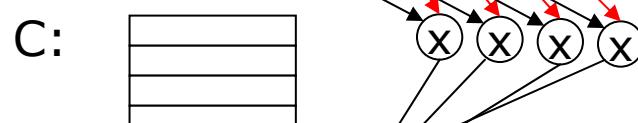
x86 Vector instructions

- 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)
- MIC: "16-way vectors". Is this 16×32 bits??



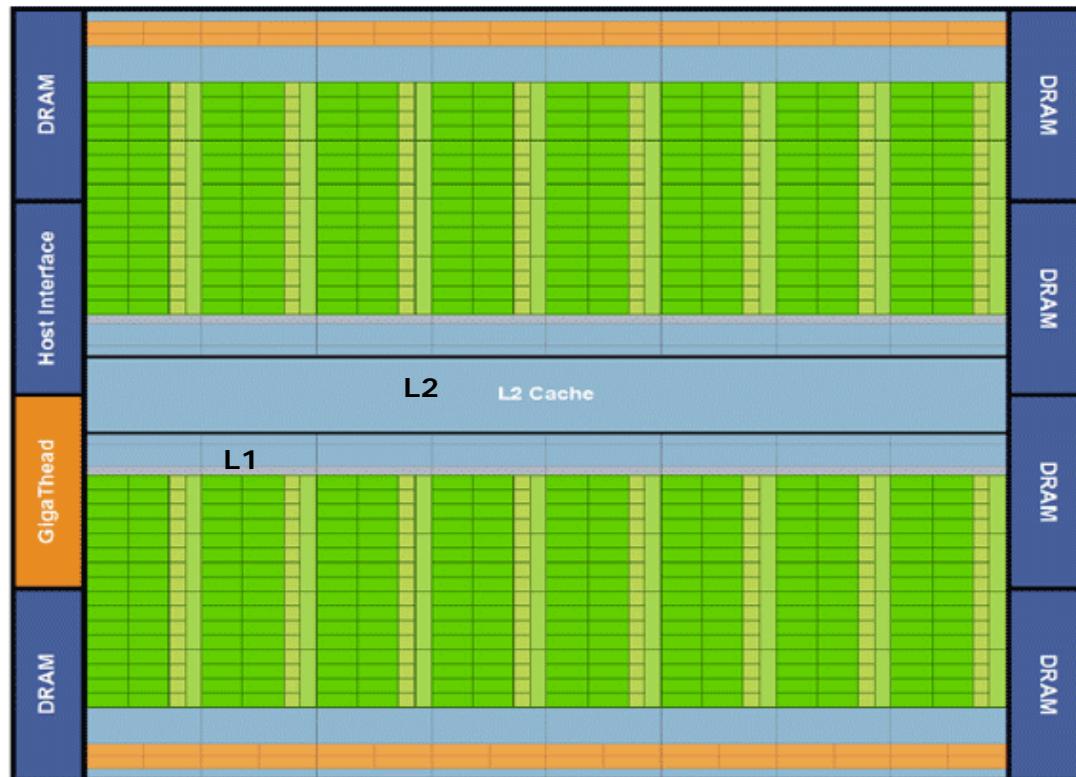
Examples of vector instructions

Vector Regs



...

How to explore SIMD: nVIDIA

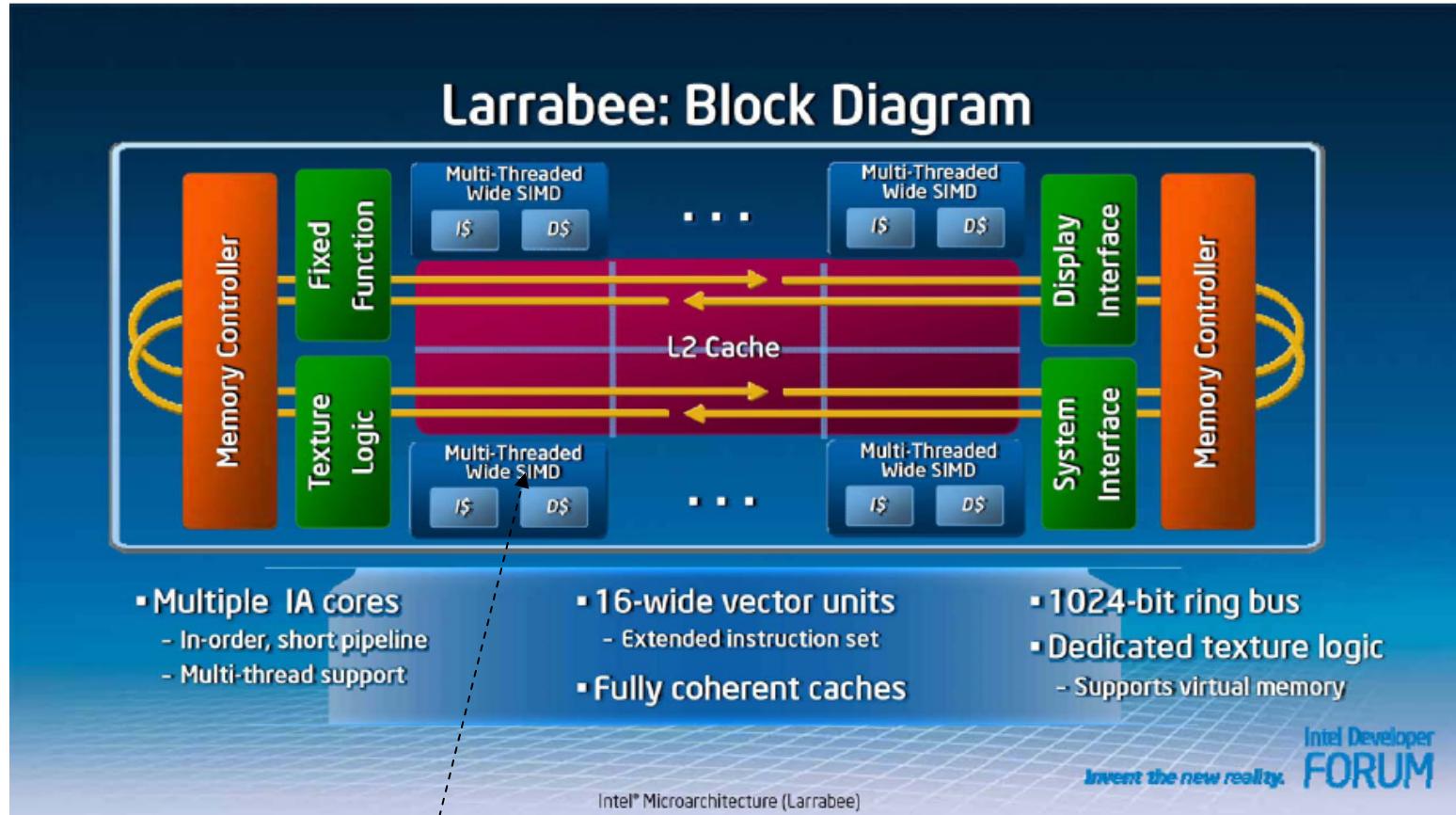


Fermi's 16 SM are positioned around a common L2 cache. Each SM is a vertical rectangular strip that contains an orange portion (scheduler and dispatch), a green portion (execution units), and light blue portions (register file and L1 cache).

- 512 "processors" (P)
- 16 P /StreamProcessor (SP)
- SP is SIMD-ish (sort off)
- Full DP-FP IEEE support
- 64kB L1 cache / SP
- 768kB global shared cache (less than the sum of L1:s)
- Atomic instructions
- ECC correction
- Debugging support
- Giant chip/high power
- ...

How to explore SIMD: Intel MIC

"more than 50 cores"



SIMD instructions: 16-way, how wide?



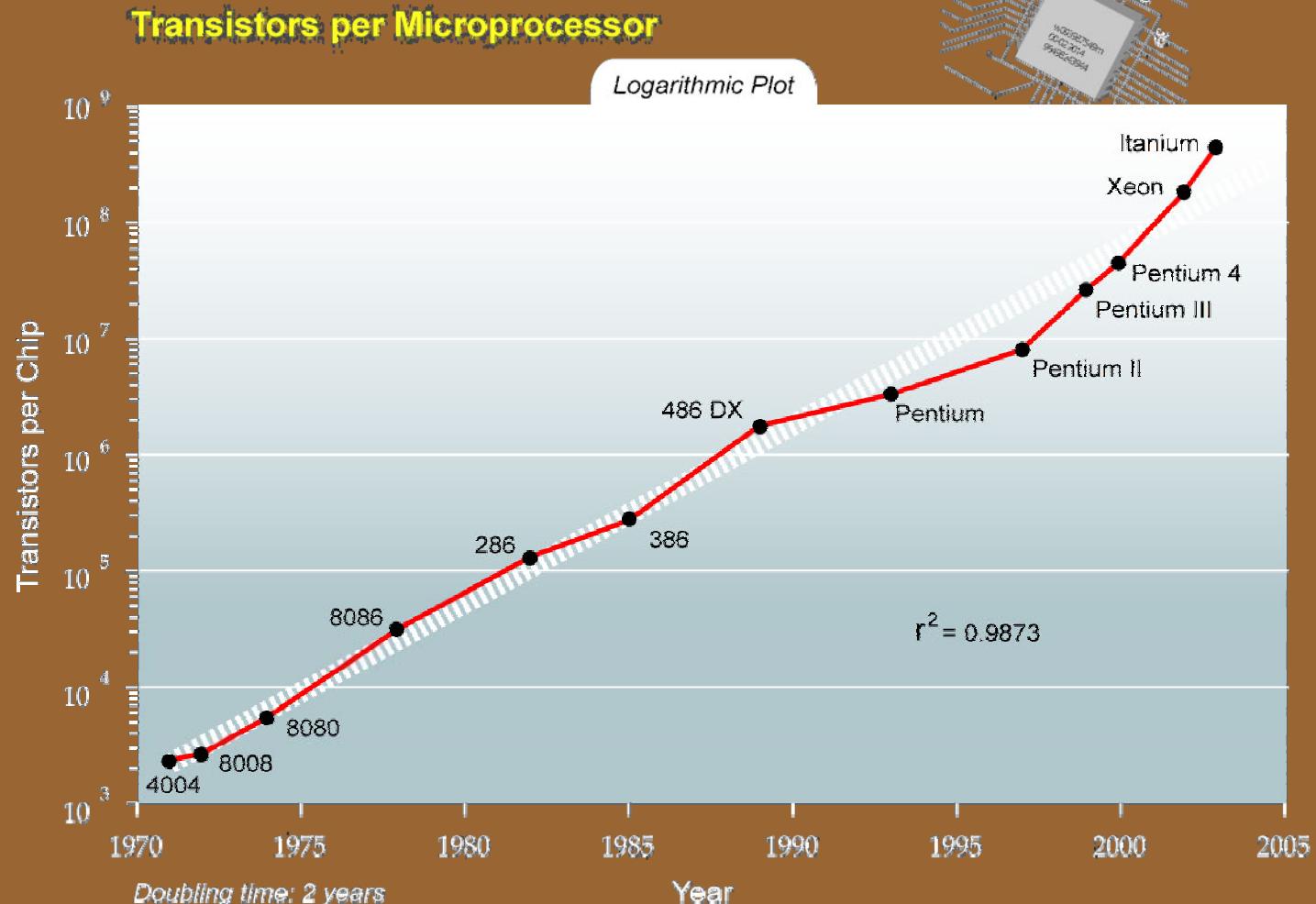
Exponential Growth

Erik Hagersten
Uppsala University
Sweden



UPPSALA
UNIVERSITET

Ray Kurzweil pictures
www.KurzweilAI.net/pps/WorldHealthCongress/

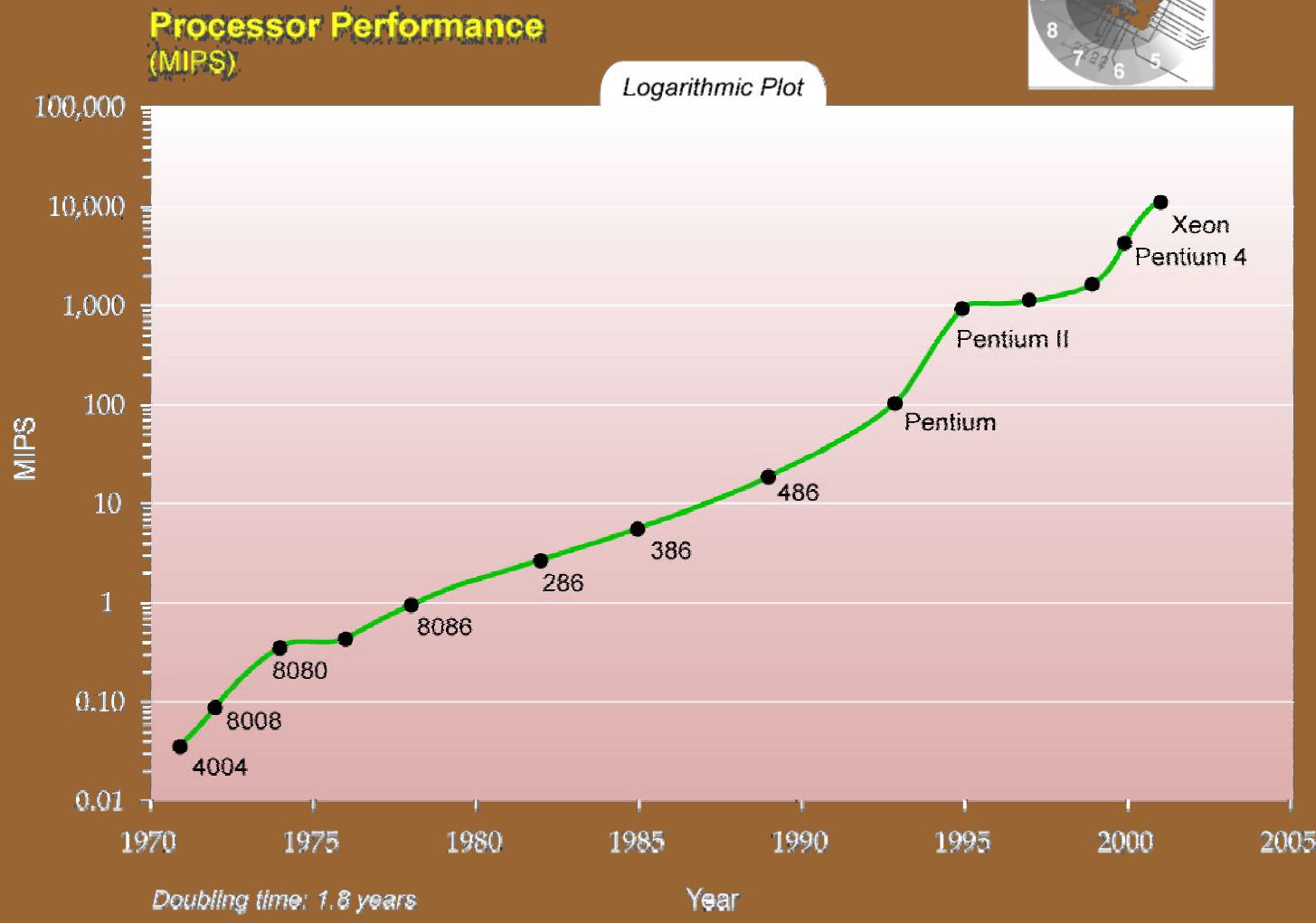
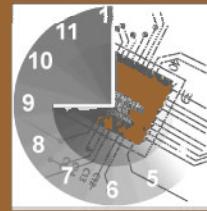


AVDARK
2010



UPPSALA
UNIVERSITET

Ray Kurzweil pictures
www.KurzweilAI.net/pps/WorldHealthCongress/



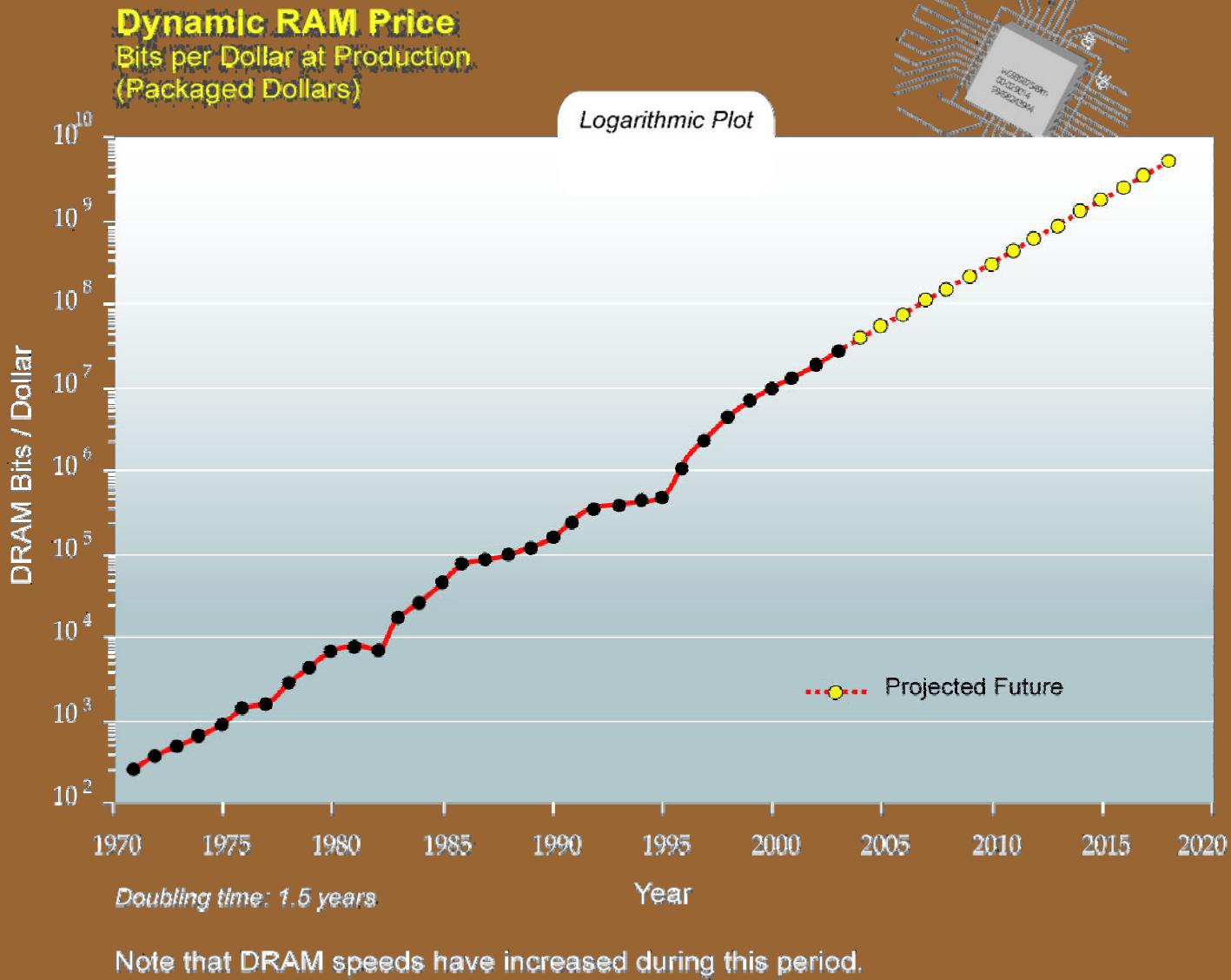
AVDARK
2010



UPPSALA
UNIVERSITET

AVDARK
2010

Ray Kurzweil pictures
www.KurzweilAI.net/pps/WorldHealthCongress/



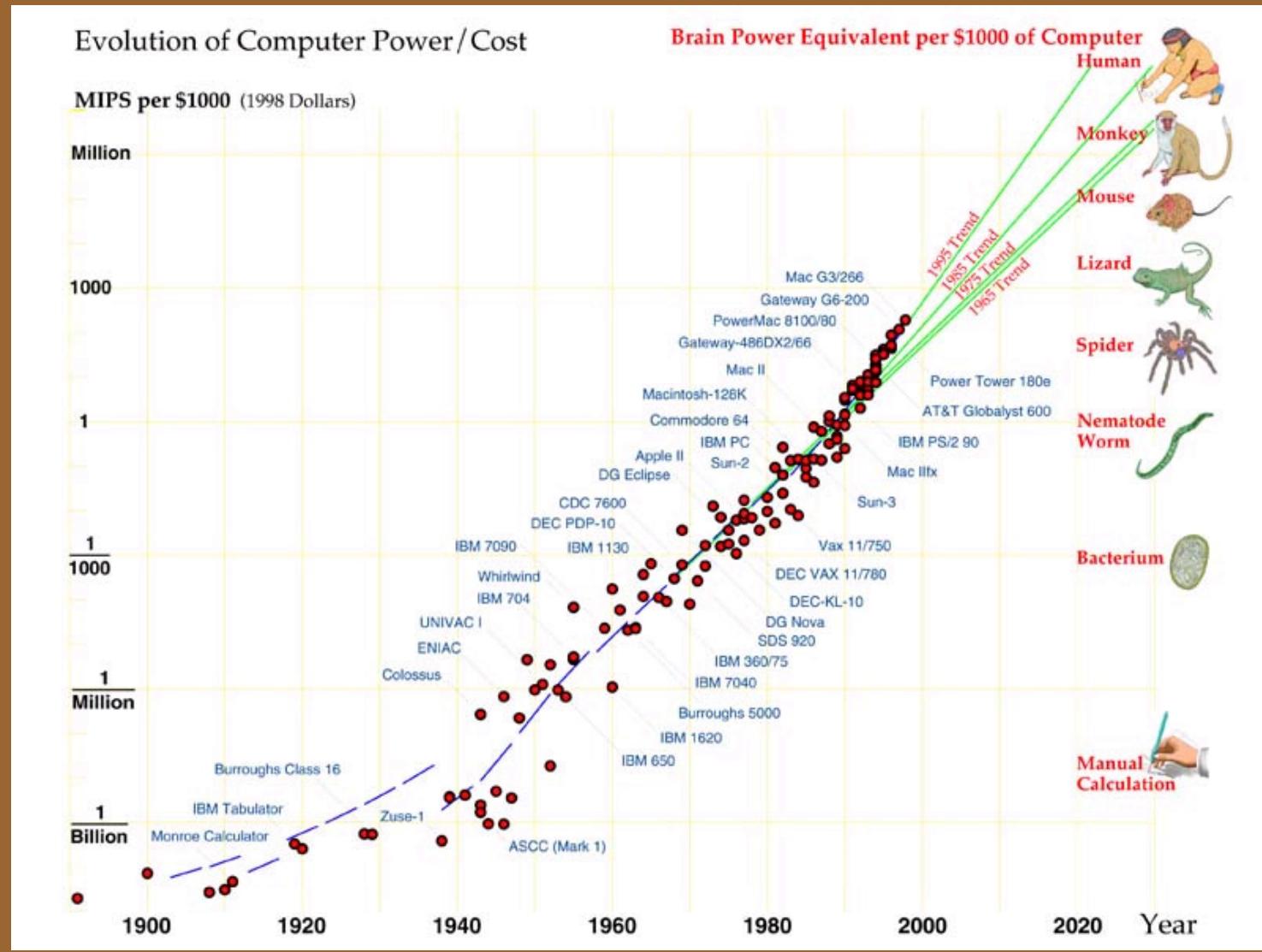


UPPSALA
UNIVERSITET

AVDARK
2010

Ray Kurzweil pictures

www.KurzweilAI.net/pps/WorldHealthCongress/



Doubling (or Halving) times [Kurzweil 2006]

■ 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