

# Multi-Prozessor Systeme

---

## Multi-Core Prozessoren

Seminar „Parallele Architekturen und Algorithmen“

# Kapitel:

A

Single-Processor Parallelität

B

Cache Kohärenz

C

Multi-Processor Systeme

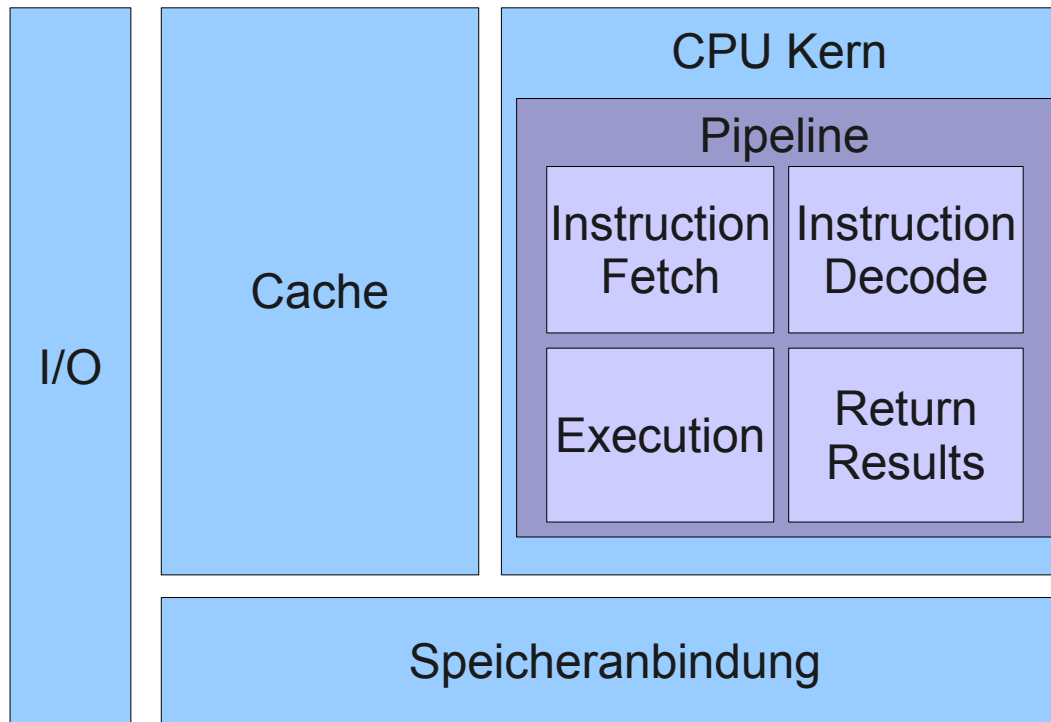
D

Ausblick: Transactional Memory

**A**

Single-Processor Parallelität

- Grundlegende Komponenten:



- Superskalarität?

parallele Ausführung von Befehlen  
(in-order/out-of-order)

- MMX, 3DNow! & AltiVec?

SIMD = Single Instruction, Multiple Data

→ Parallelität bereits in Single-Core, Single-Prozessor  
Maschinen

- Vektorprozessoren:

ein Befehl auf mehrere Daten pro Schritt

→ SIMD

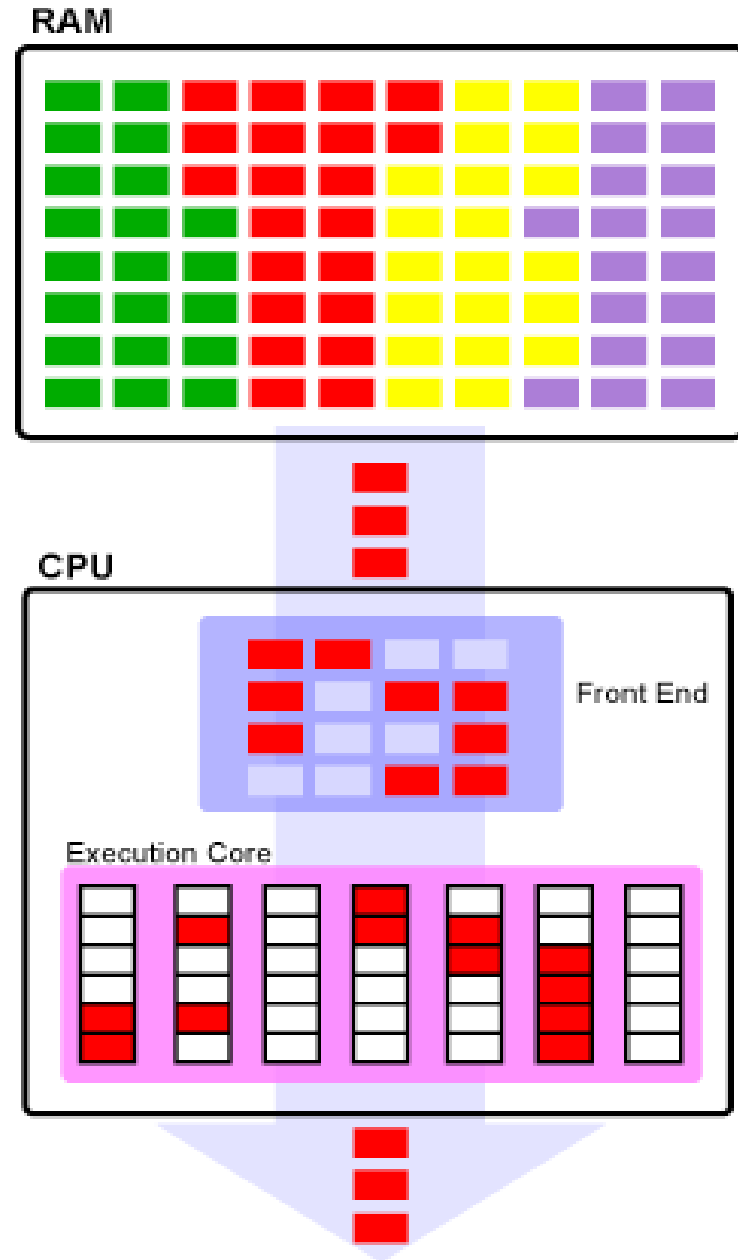
Beispiel: GPUs, DSPs

- VLIW: Very Long Instruction Word

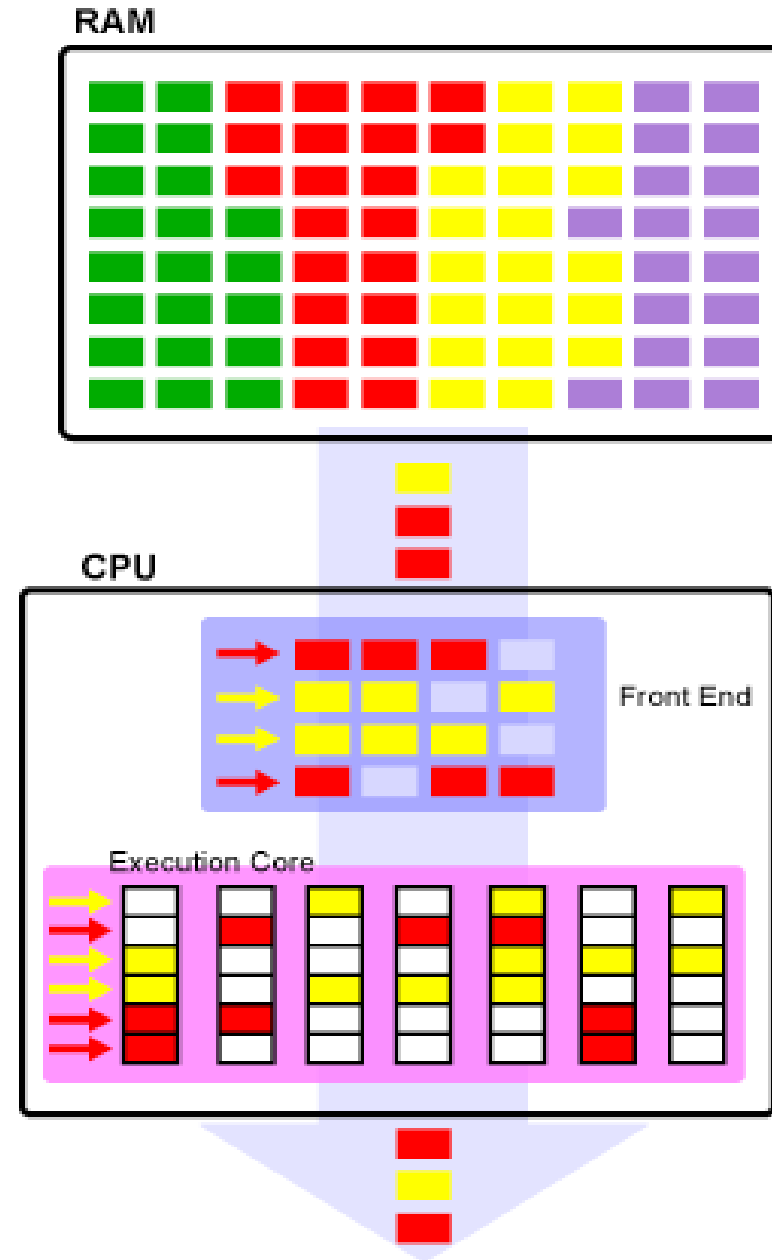
→ Parallelität auf Compiler-Ebene

- Hyper-Threading oder Simultaneous Multi-Threading
  - effizientere Ausnutzung der CPU Ressourcen
  - exklusiv: Instruction Pointer, bestimmte Register
  - sonst alles geteilt/partitioniert
- Mehrere Threads werden gleichzeitig ausgeführt, verschiedene Threads pro Pipelinestufe möglich

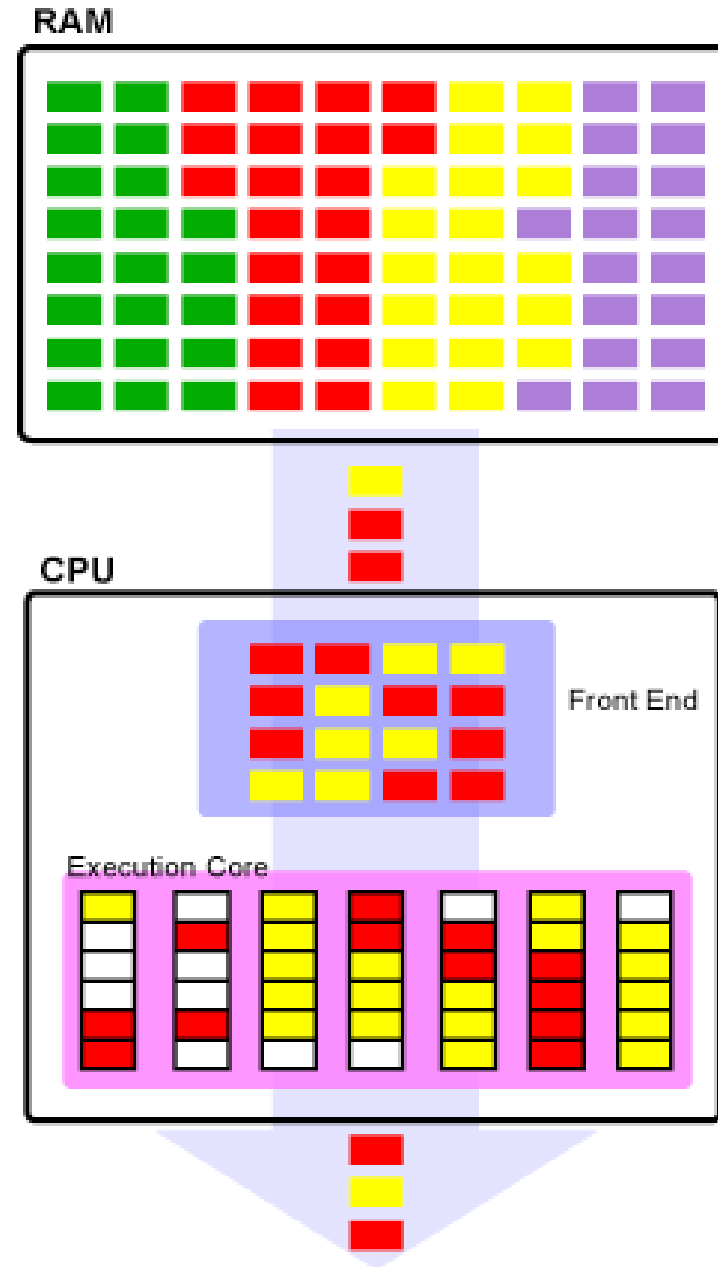
- Single-Threading:



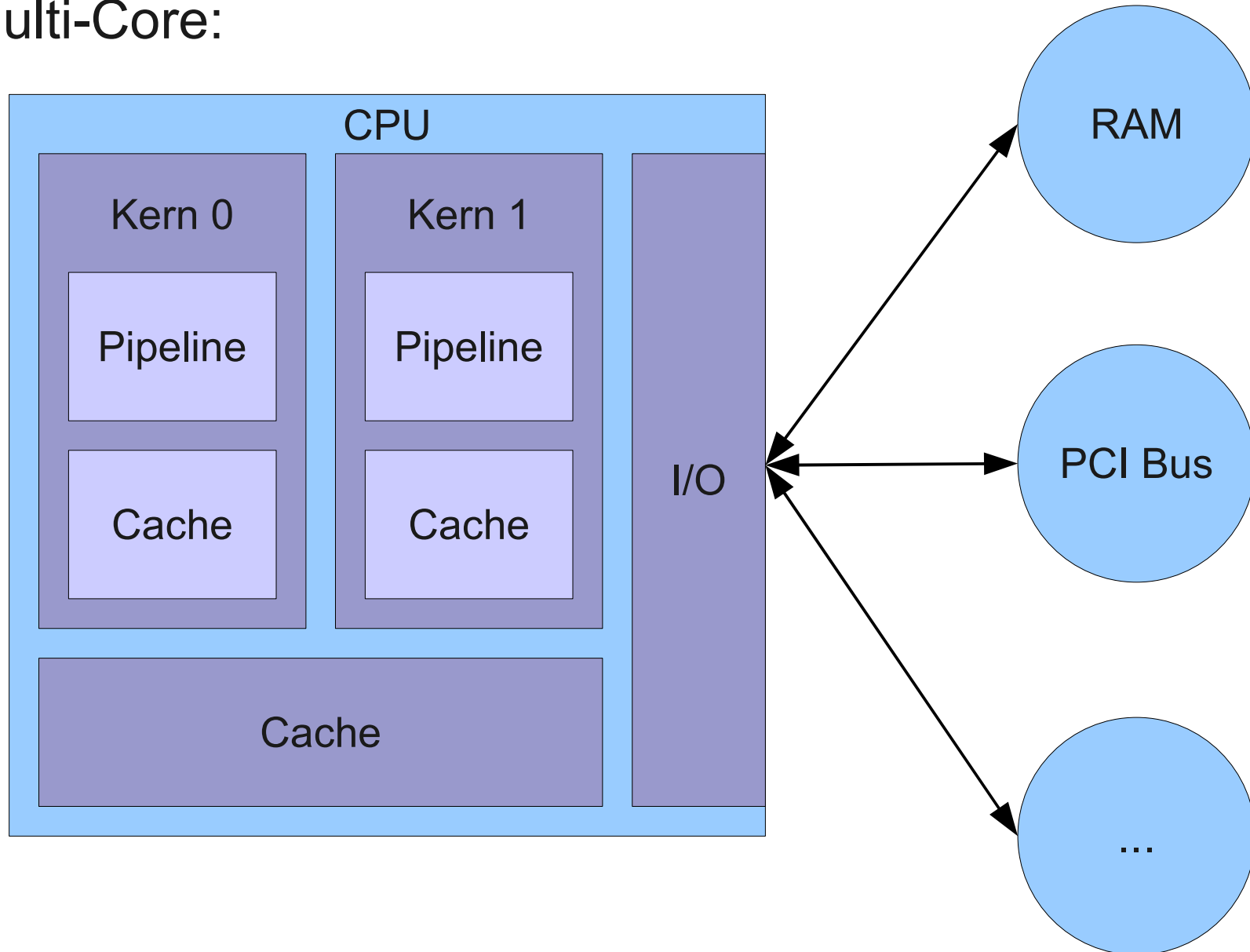
- Super-Threading:  
Mehrere Threads,  
1 pro Pipeline-Stufe



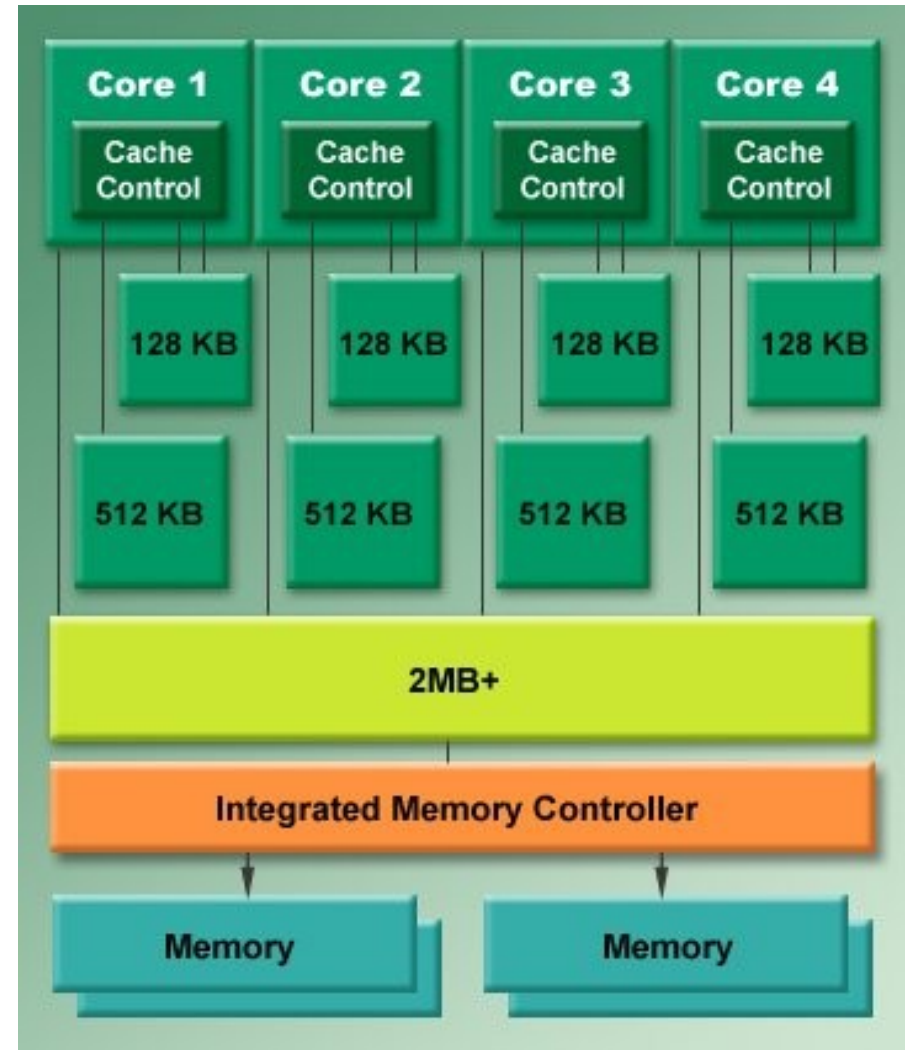
- Hyper-Threading:  
Mehrere Threads pro  
Pipeline-Stufe



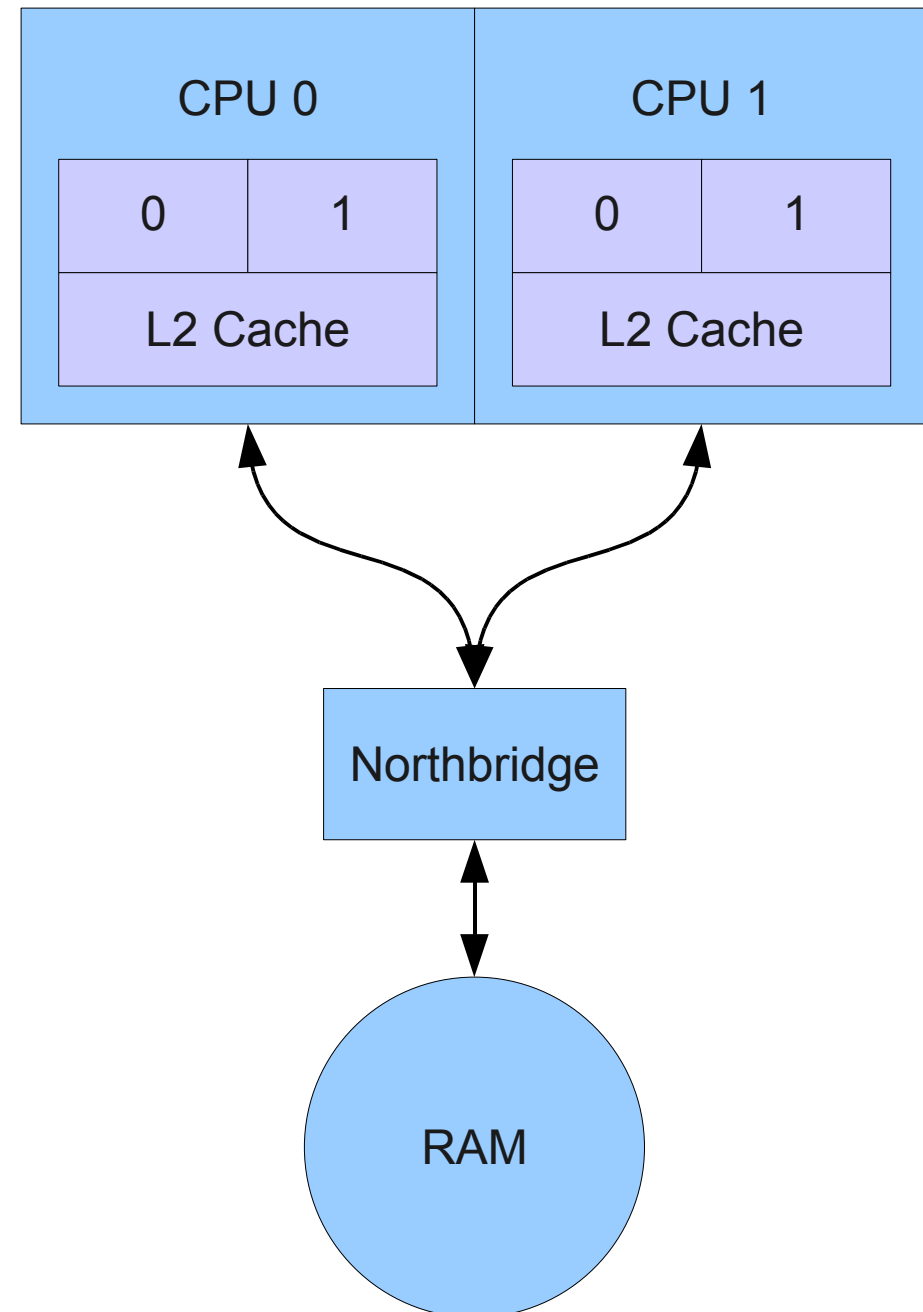
- Multi-Core:



- Beispiel: AMD Phenom
  - 2x assoz. L1 Cache
  - 16x assoz. L2 Cache
  - 32x assoz. L3 Cache
  - Speicher-Controller intern

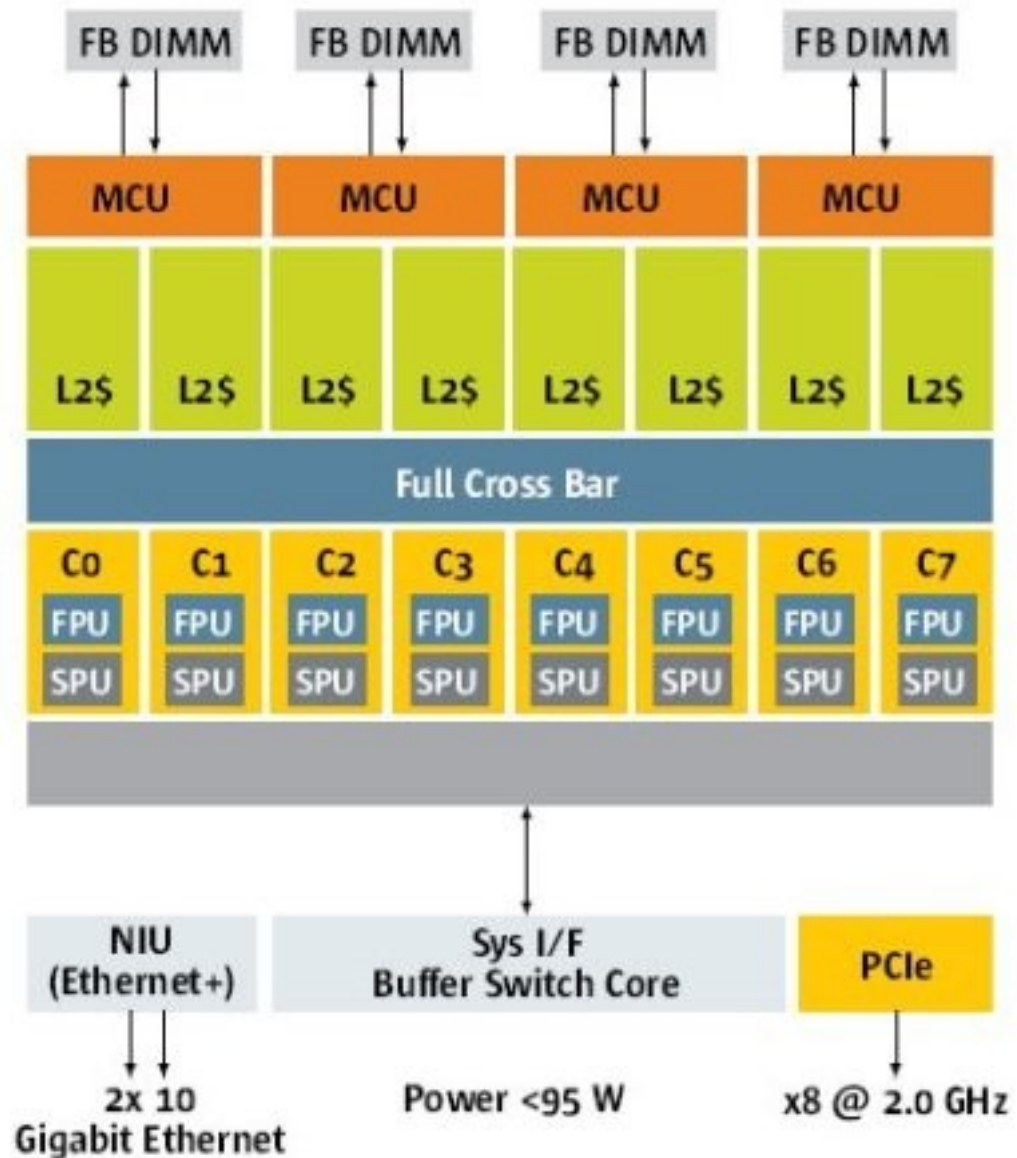


- Beispiel: Intel Core 2 Quad
- 2 Dual-Core Dice
- pro Dualcore:
- 24x assoz. L2 Cache
- L2 Cache wird geteilt
- Speicher-Controller extern
- „Dual Independent Bus“



- Beispiel: Sun UltraSPARC T2

- 8 Kerne
- 8 Threads pro Kern
- 8x assoz. L1 I-Cache
- 4x assoz. L1 D-Cache
- 16x assoz. L2 Cache



Quelle: Sun.com

**B**

Cache Kohärenz

- Cache Kohärenz:

ein Lesezugriff liefert immer den Wert  
des letzten Schreibzugriffs

- Write-Invalidate:

Änderung in einem Cache →  
als ungültig markieren in allen anderen

- Write-Update:

Änderung in einem Cache →  
aktualisieren in allen anderen

- Write-Back:

Änderung werden erst in den Hauptspeicher übertragen, wenn die Cache-Line ersetzt wird

- Write-Through:

Änderung werden sofort in den Hauptspeicher übertragen

- „Bus Snooping“ Protokolle:
  - jede CPU „hört mit“
  - Einsatz bei Bus-basierten Systemen
  - simpel zu implementieren
  - skaliert nicht gut

- **MESI: Modified Exclusive Shared Invalid**
  - weit verbreitete Art eines Write-Invalidate Protokolls
  - Kombinationen aus lokaler und „mitgehörter“ Änderung ergeben Status

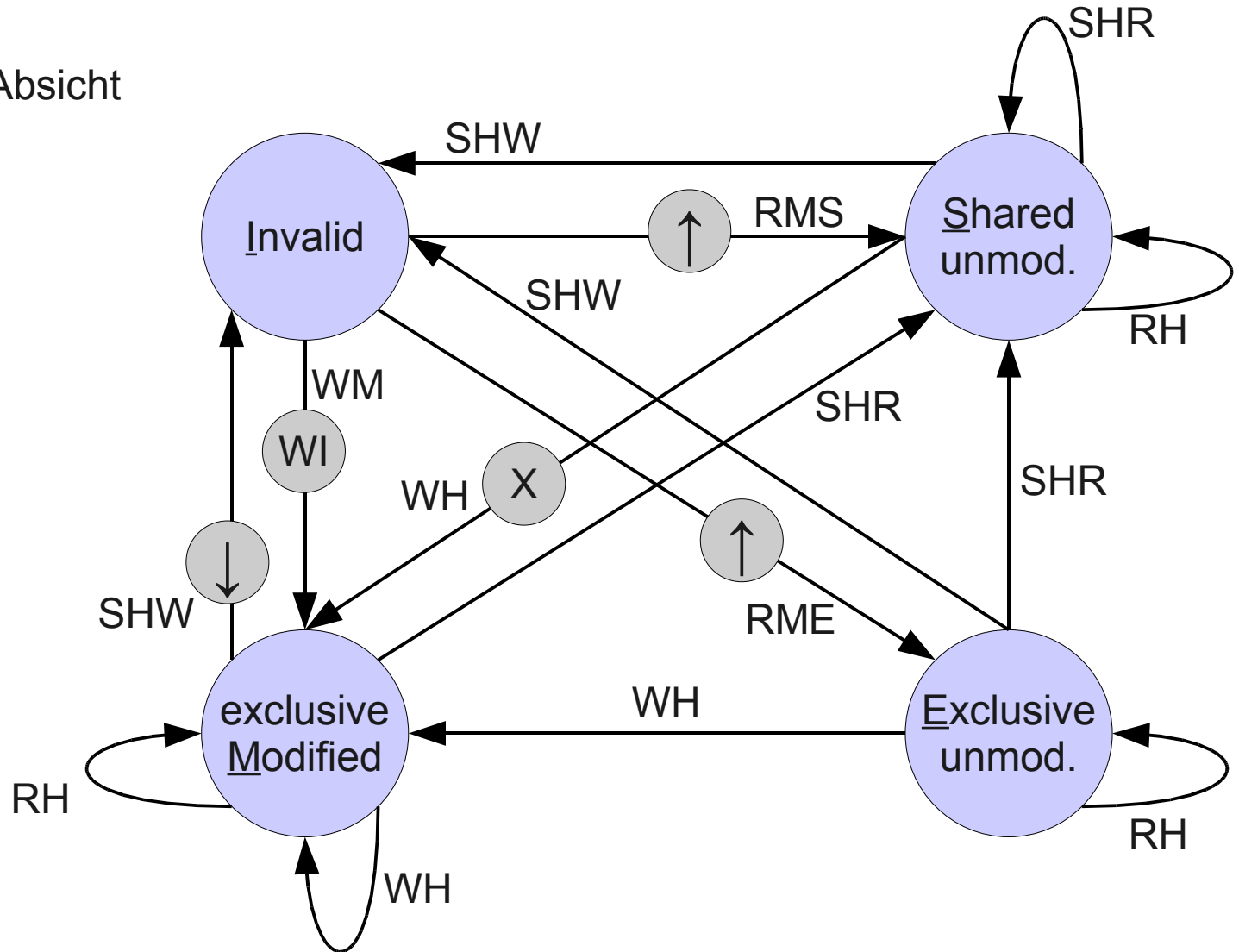
RH	Read Hit	WH	Write Hit	SHR	Snoop Hit on Read
RMS	Read Miss, Shared	WM	Write Miss	SHW	Snoop Hit on Write oder Read mit Write-Absicht
RME	Read Miss, Exclusive				

WI Read mit Write-Absicht

X Invalidate

↑ Copy Fill

↓ Copy Back



- Problemfälle:
  - „False Sharing“
  - „Dead Sharing“
  
- Verbesserungsmöglichkeiten:
  - Bus Read-Sharing
  - „Snarfing“

- Directory-basierte Protokolle:
  - relevante Daten in einer zentralen Struktur
  - höhere Latenz als bei Bus Snooping
  - aufwendig in der Implementierung
  - skaliert gut

Status	0	1	2	3	4	5
Shared	X				X	
Exclusive				X		
Shared			X			X
Shared		X	X		X	
Invalid						
Exclusive			X			
Shared		X		X		X
Exclusive				X		
Shared	X				X	X

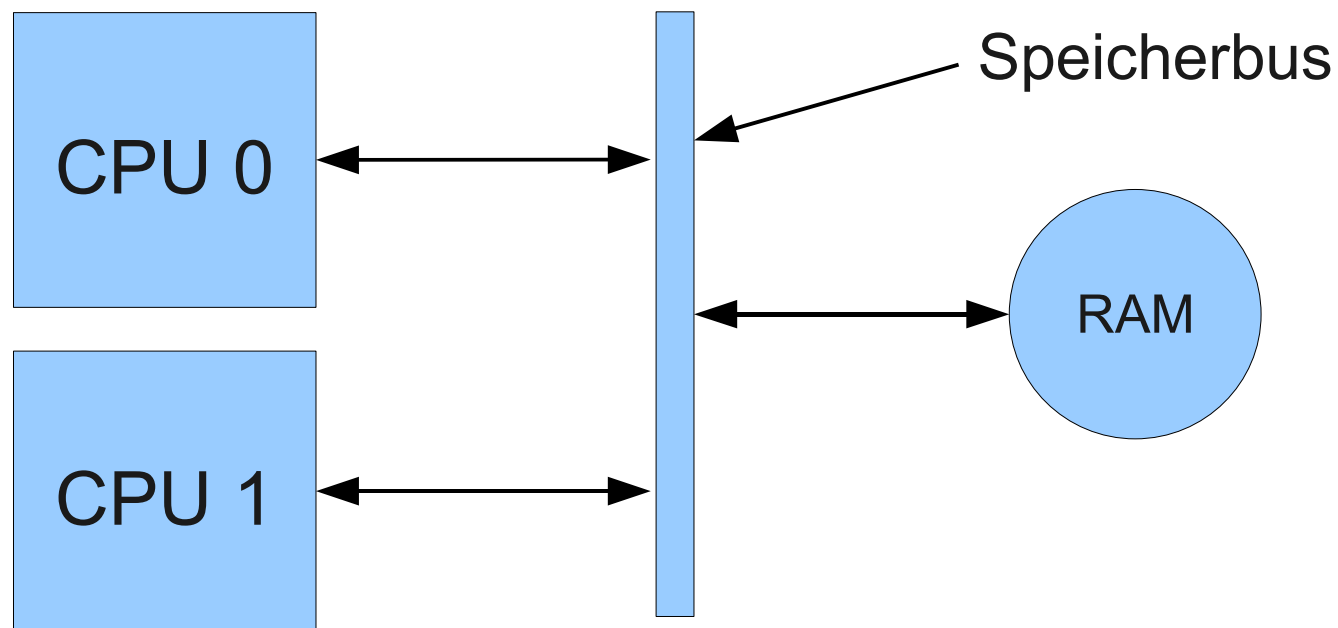
Status:  
M / E / S / I

Status:  
S / I

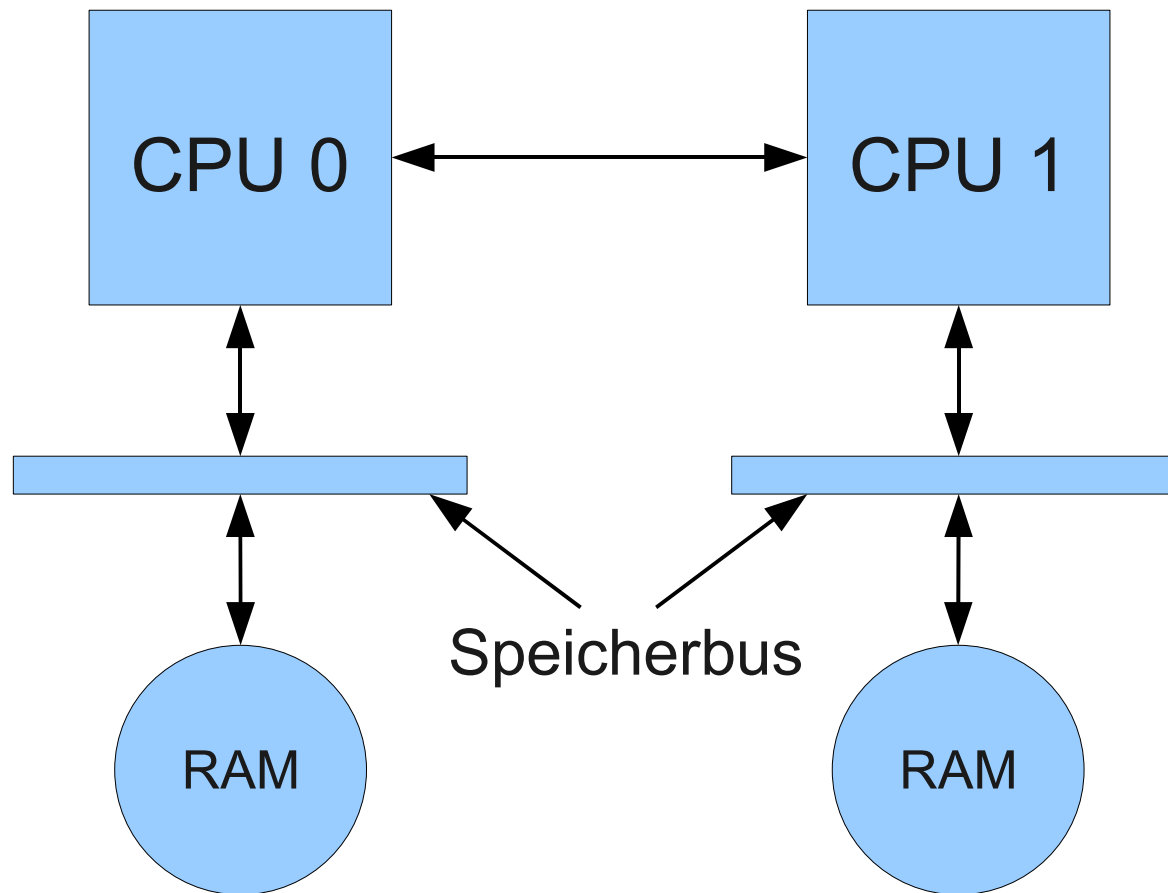
**C**

# Multi-Processor Systeme

- SMP: Mehrere Prozessoren (Kerne) teilen sich einen Adressraum.
- Simple Form von SMP als UMA realisiert



- bessere Skalierung für SMP: NUMA



- ccNUMA: Cache Coherent NUMA

Speicherstelle A in Cache von CPU 0 und CPU 1:

„Änderung“ von A in Cache von CPU 0

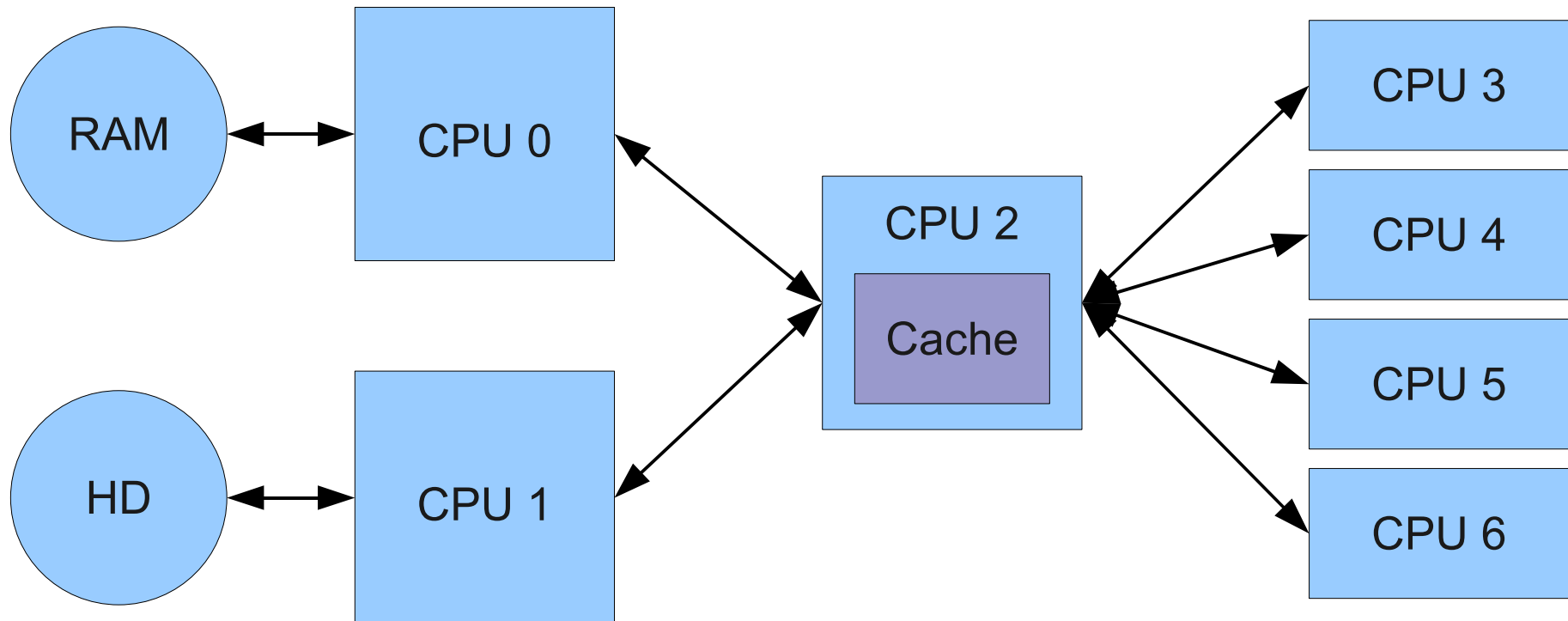
↔

„Änderung“ von A in Cache von CPU 1

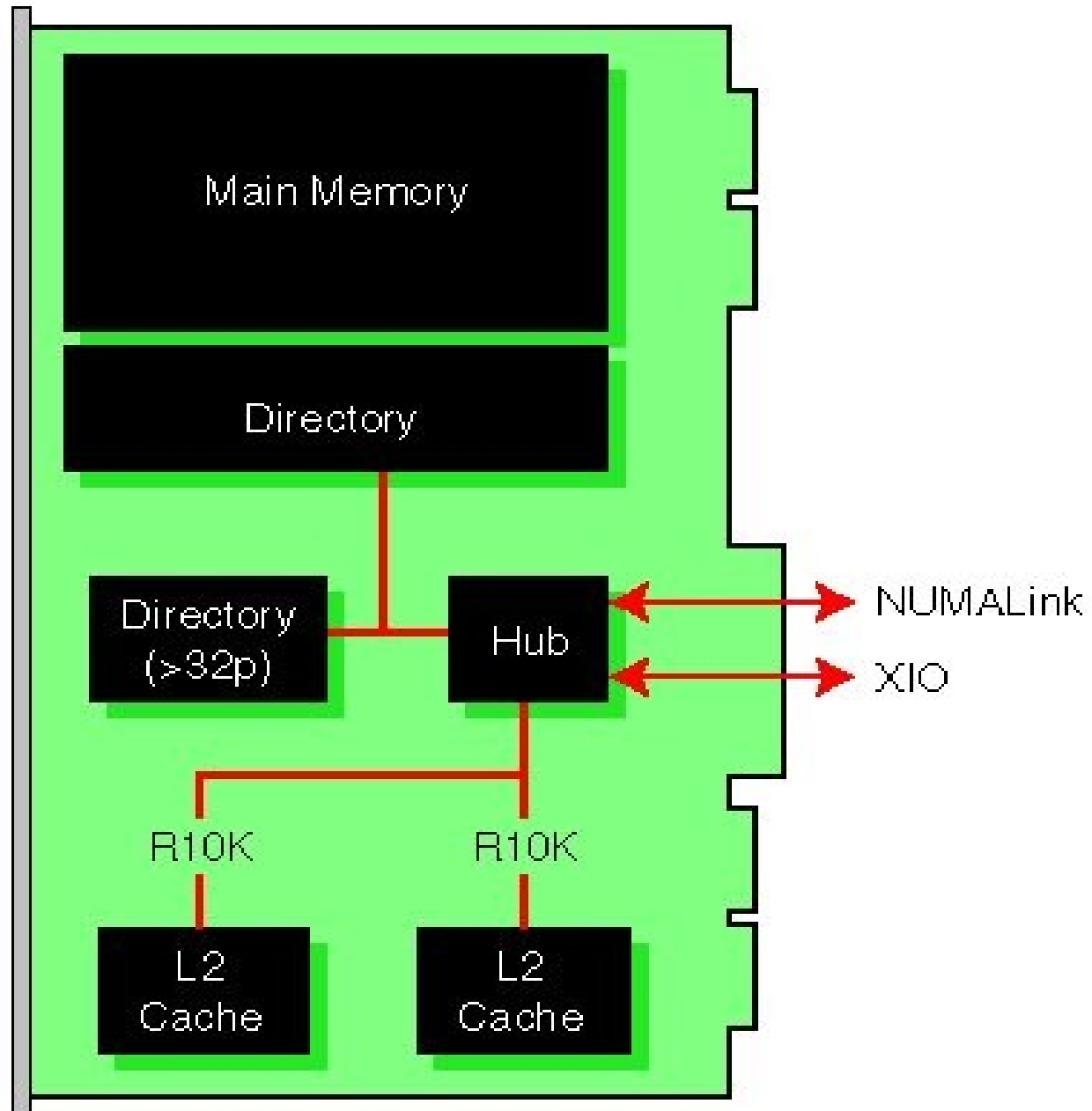
- Praktisch relevant sind nur ccNUMA Systeme

- ASMP:

Mehrere Prozessoren (Kerne) für spezielle, unterschiedliche Aufgaben



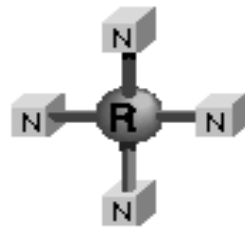
- Beispiel: SGI SN0 Architektur
  - NUMA
  - Directory Cache Kohärenz
  - Origin System ausgelegt auf bis zu 2048 CPUs
  - organisiert in Hypercubes



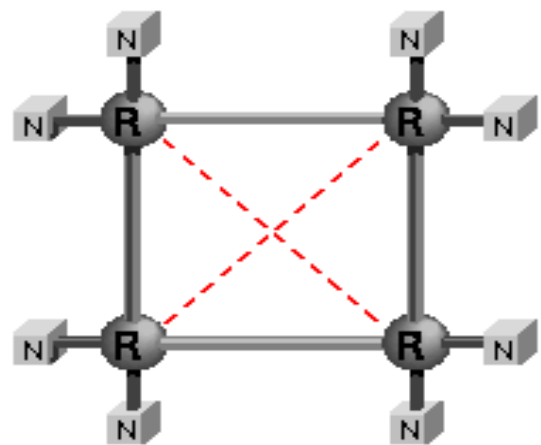
Quelle: SGI.com



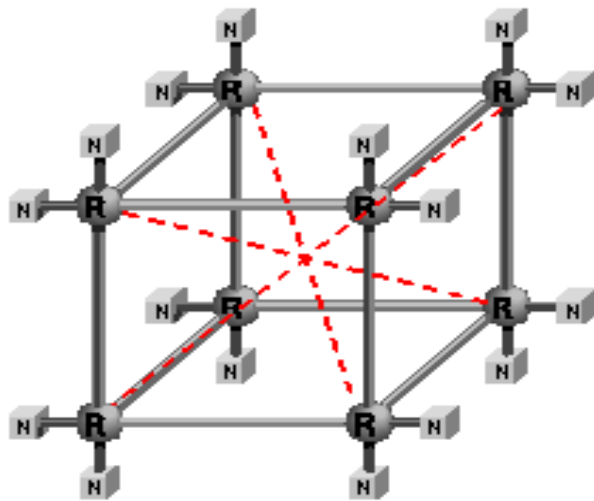
4 processor system



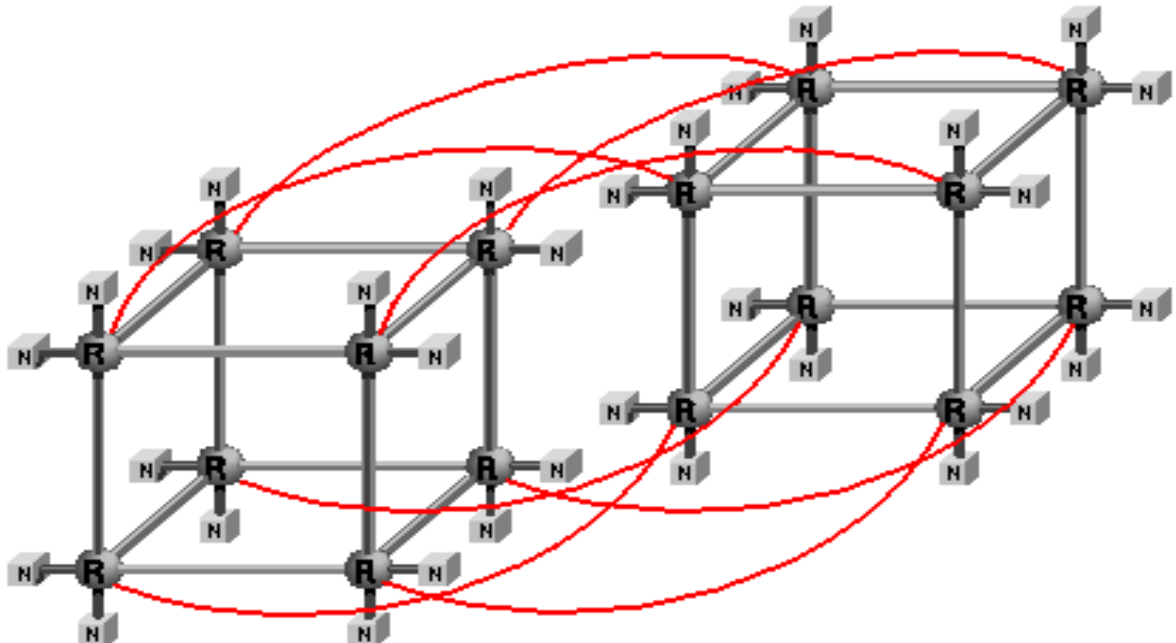
8 processor system



16 processor system



32 processor system



64 processor system

- Vergleich: Bus-System vs. Origin 2000

#CPUs	Class A (s)	X	A	Class B (s)	X	A
1	62.73	1.00	1.00	877.74	1.00	1.00
2	32.25	1.95	1.95	442.66	1.98	1.98
4	16.79	3.74	3.69	225.10	3.90	3.90
8	9.58	6.55	6.68	125.71	6.98	7.54
12	7.96	7.88	9.16	103.44	8.49	10.96
16	7.55	8.31	11.24	98.92	8.87	14.16
1	55.43	1.00		810.41	1.00	
2	29.53	1.88		419.72	1.93	
4	15.56	3.56		215.95	3.75	
8	8.05	6.89		108.42	7.47	
16	4.55	12.18		54.88	14.77	
32	3.16	17.54		28.66	28.28	

**D**

Ausblick: Transactional Memory

- Transactional Memory: ACI Operationen
  - Atomicity: ganz oder gar nicht
  - Consistency: Konsistenz-erhaltend
  - Isolation: unabhängig von anderen Transaktionen
- Vereinfachung paralleler Programmierung bei minimalem Geschwindigkeitsverlust
- Transaktionen explizit im Code
- erste Implementierung: Sun Rock Prozessor (2009)

- Eigenschaften von Transactional Memory:
  - strong isolation vs. weak isolation
  - transaction granularity
  - direct & deferred updates

- Eigenschaften von Transactional Memory:
  - nested transactions

```
int x = 1;

atomic {
    x = 2;
    atomic [flatten/closed]{
        x = 3;
        abort;
    }
}
```

- Eigenschaften von Transactional Memory:
  - nested transactions (weiter):

```
int x = 1;

atomic {
    x = 2;
    atomic open{
        x = 3;
    }
    abort;
}
```

- MPP: Massively Parallel Processing  
→ Super Computer, Cluster



## Quellenangaben:

### Links:

- [http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/srch14@directory20dimm/0650/bks/SGI\\_Developer/books/OrOn2\\_PfTune/sgi\\_html/ch01.html#LE11013-PARENT](http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/srch14@directory20dimm/0650/bks/SGI_Developer/books/OrOn2_PfTune/sgi_html/ch01.html#LE11013-PARENT)
- <http://www.bernd-leitenberger.de/simd-vliw.shtml>
- [http://www.kreissl.info/diggs/ra\\_09.php](http://www.kreissl.info/diggs/ra_09.php)
- <http://www.stanford.edu/class/ee382/vector/index.htm>
- [http://de.wikipedia.org/wiki/Simultaneous\\_Multithreading](http://de.wikipedia.org/wiki/Simultaneous_Multithreading)
- [http://wiki.c0t0d0s0.org/index.php?title=Was\\_ist\\_der\\_UltraSPARC\\_T1-Prozessor\\_%3F](http://wiki.c0t0d0s0.org/index.php?title=Was_ist_der_UltraSPARC_T1-Prozessor_%3F)
- <http://lse.sourceforge.net/numa/faq/>
- <http://lwn.net/Articles/254445/>
- <http://lse.sourceforge.net/numa/>
- <http://www.sharkyforums.com/archive/index.php/t-114071.html>
- [http://www.2cpu.com/articles/42\\_2.html](http://www.2cpu.com/articles/42_2.html)
- <http://arstechnica.com/articles/paedia/cpu/pipelining-1.ars/2>
- <http://arstechnica.com/articles/paedia/cpu/cpu2.ars>
- [http://cs.gmu.edu/cne/modules/dsm/yellow/page\\_dsm.html](http://cs.gmu.edu/cne/modules/dsm/yellow/page_dsm.html)
- <http://cs.gmu.edu/cne/modules/dsm/green/coherence.html>

### Paper/Präsentationen:

- Rober Golla, Sun Microsystems : „Niagara2: A Highly Threaded Server-on-a-Chip“
- Michael Koontz: „Comparison of High Performance Northbridge Architectures in Multiprocessor Servers“
- Jeffrey B. Rothman, Alan J. Smith: „Minerva: An Adaptive Subblock Coherence Protocol for Improved SMP Performance“
- James Laudon, Daniel Lenoski, Silicon Graphics, Inc.: „System Overview of the SGI Origin 200/2000 Product Line“
- Stefano Cozzini , Axel Kohlmeyer, Roger Rousseau: „Benchmark Analysis of 64-bit Servers for Linux Clusters for Application in Molecular Modeling and Atomistic Simulations“
- Mani Azimi, Naveen Cherukuri, et al., Intel Corporation: „Integration Challenges and Tradeoffs for Tera-scale Architectures“

### Bücher:

- James R. Larus, Ravi Rajwar: „Transactional Memory“  
ISBN: 978-1598291247