Αρχιτεκτονική Υπολογιστών

Εισαγωγή στην Αρχιτεκτονική Παράλληλων Υπολογιστών

Διονύσης Πνευματικάτος

pnevmati@cslab.ece.ntua.gr

5ο εξάμηνο ΣΗΜΜΥ – Ακαδημαϊκό Έτος: 2019-20

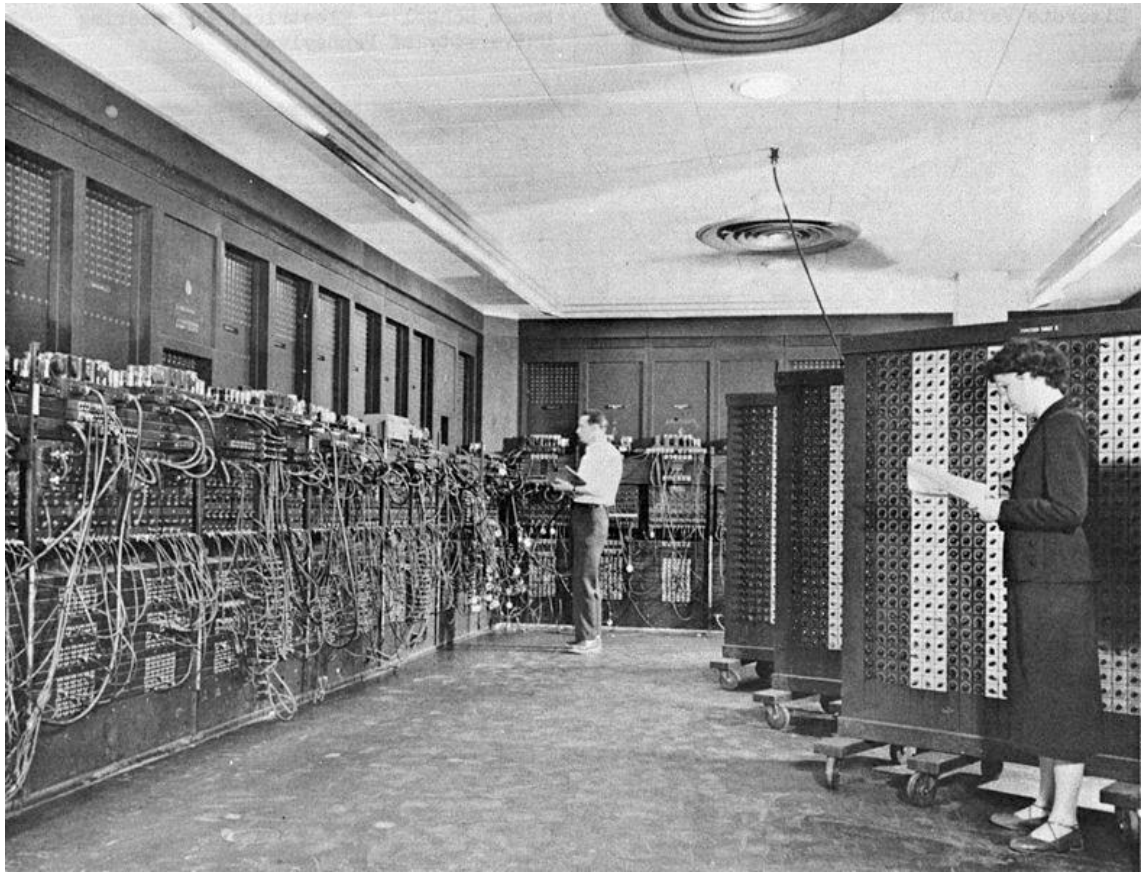
Τμήμα 2 & 3

http://www.cslab.ece.ntua.gr/courses/comparch/

Where did it all start? ENIAC

- At Penn
- Lt Gillon, Eckert and Mauchley
- Cost $486,804.22, in 1946 (~$6.3M today)
- 5000 ops/second
- 19K vacuum tubes
- Power = 200K Watts

[picture from Wikipedia]



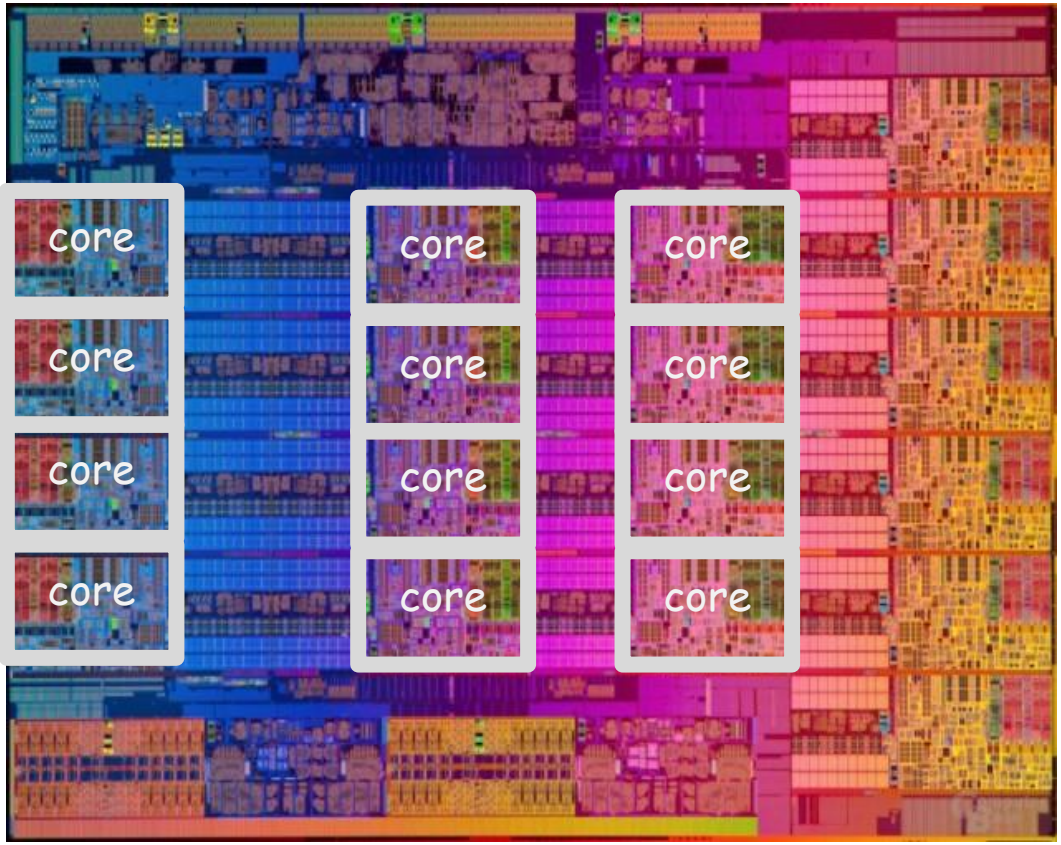
67 m^3, 27 tons(!)

Where are we ~today?
Intel's Haswell-EX

- 5.6 billion transistors
- 12 cores
- 45 MB L3 cache
- 2.5 GHz
- Roughly 165W

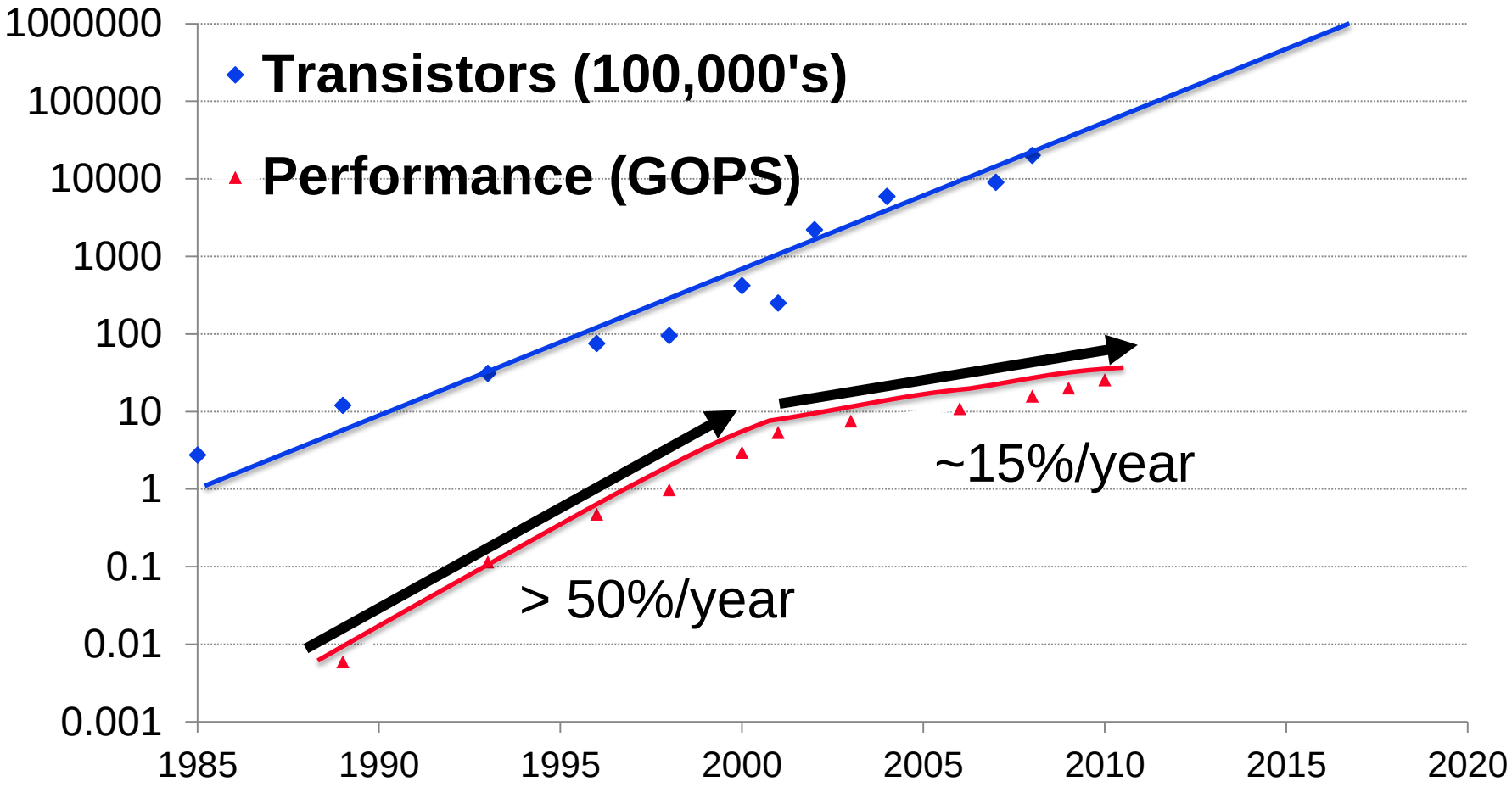
Newer i9:

- 8 cores, 16 threads
- 3.6->5 GHz clock
- TDP 95W
- 16MB Cache

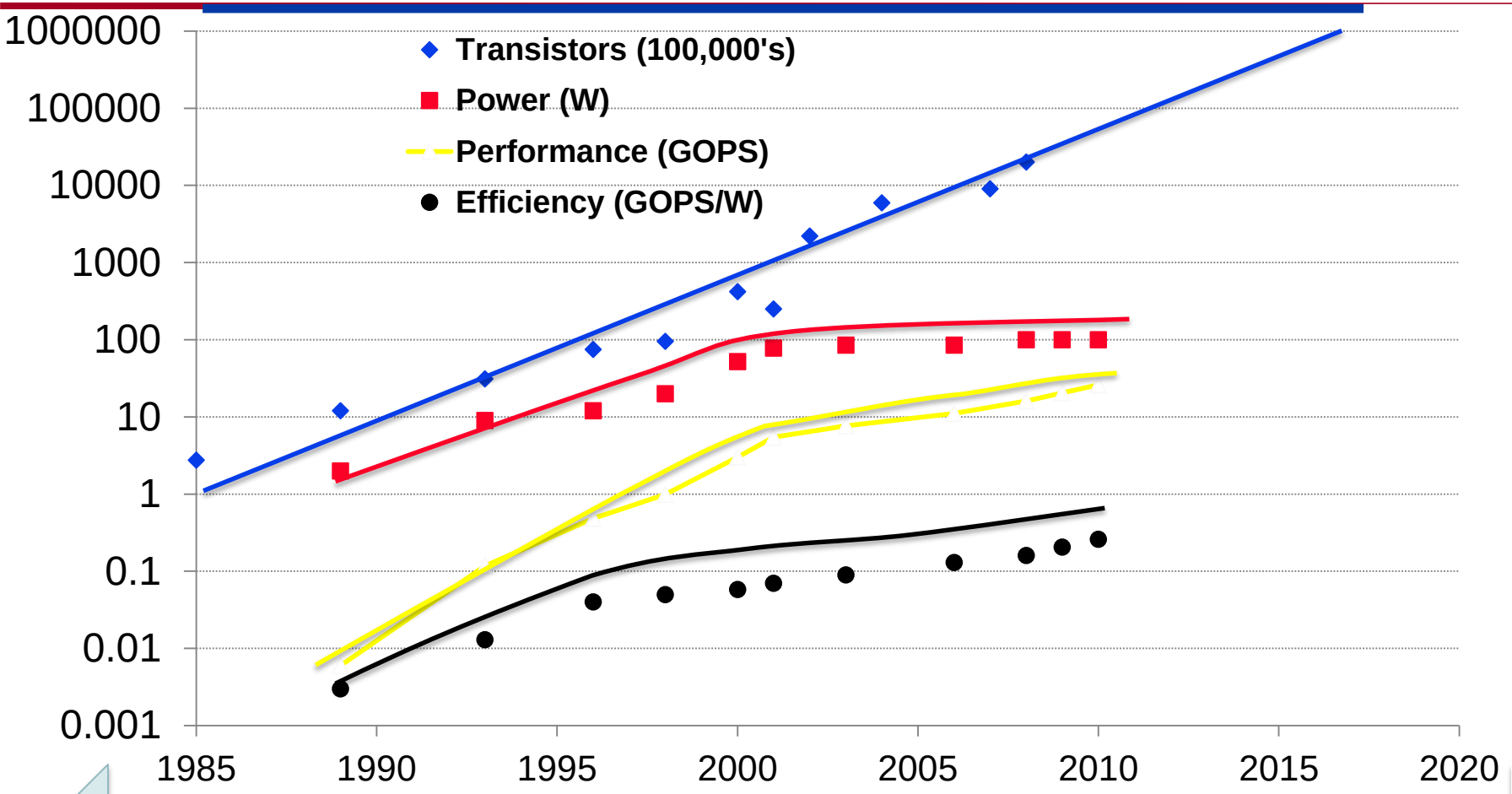


662 mm^2

The chips got bigger but not faster!!!!



Why? Must keep power at ~100W!



Era of Uniprocesors

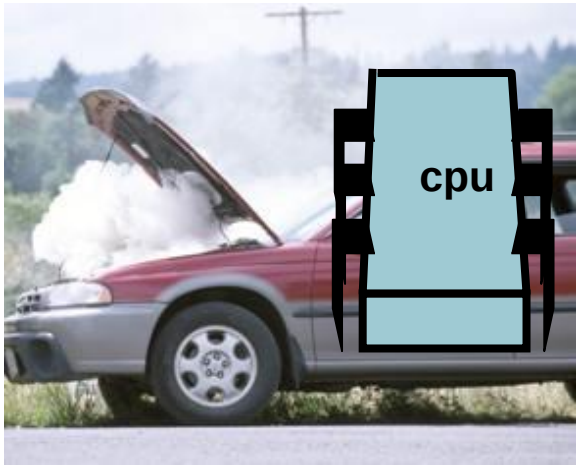
c. 2005

Era of Multiprocessors

What is Power?

For fixed Capacitance:

- Power $\propto V^2F$
- V = Operating voltage
- F = Clock frequency

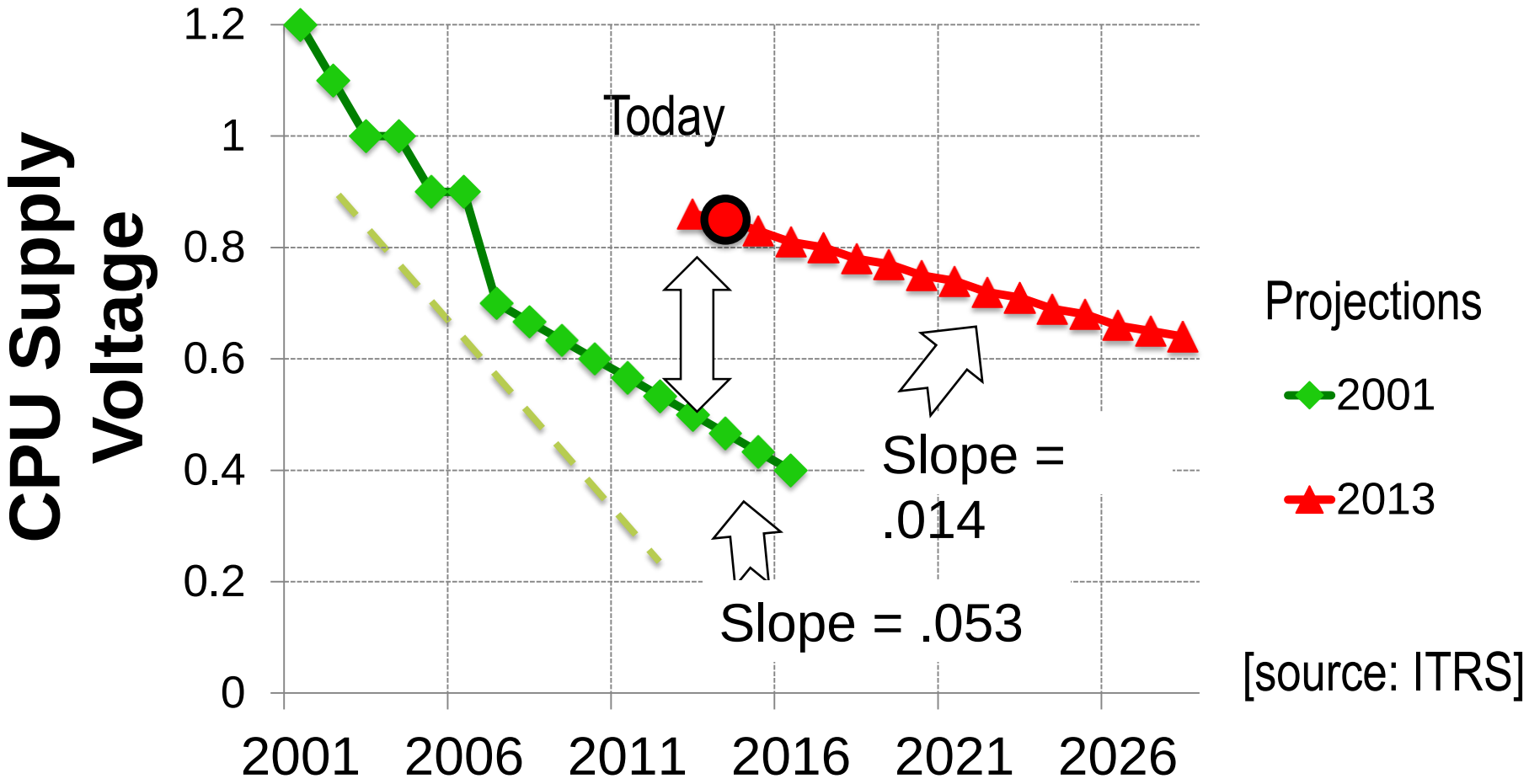


CPU type	Power	Constraint
Mobile	< a few W	Battery usage
Laptop	< 10s of W	Battery + heat
Desktop/Server	< 100 W	Cooling
Supercomputer	< 10s of W	Cooling + electricity

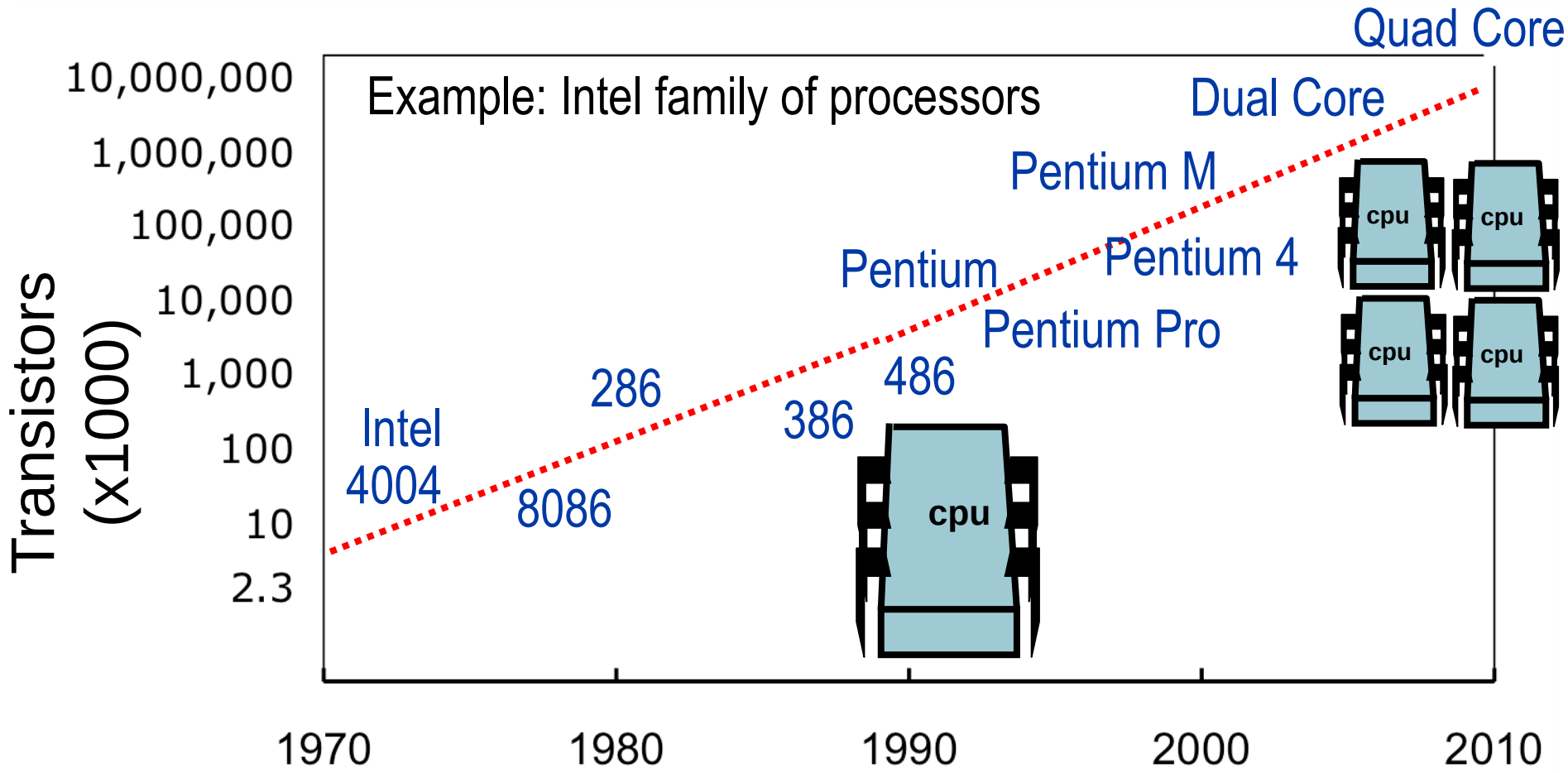
What happened to Power?

- Voltages used to go down:
  - From 5v (1970's) to 1v (2000's)
  - Power $\propto V^2F$
  - But, voltage is squared! Great!
  - Power went down: 25x reduction!
  - Gave us enough room to increase clock frequency

Voltages are not going down!



New Moore's Law: more cores

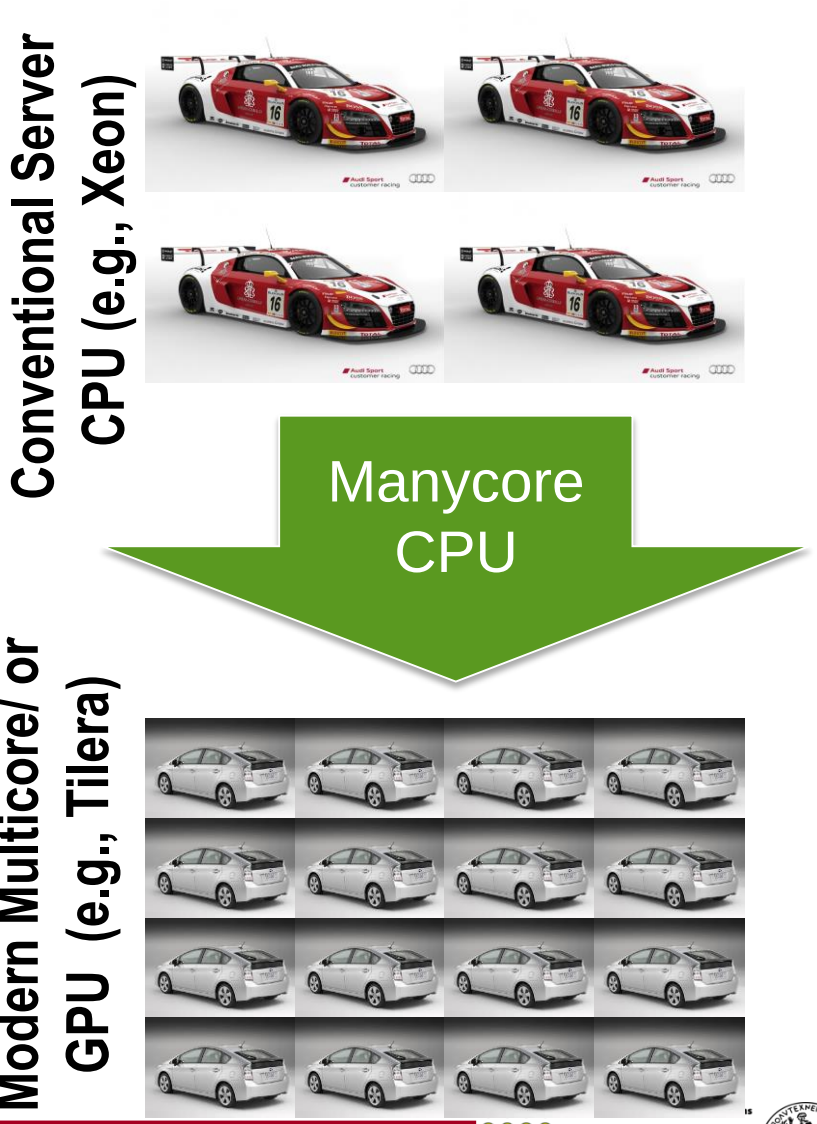


2x transistors ➔ 2x cores

And with more cores, come simpler cores…

With fixed voltages & clocks:

- Parallelism/concurrency is the only solution to more transistors
- Processor @ 100W
- Lots of tiny "cores" (i.e., CPUs)
  - Prius instead of Audi
- Each core ➔ fewer joules/op
- Need parallel software!



Terms of interest

- Multicores & Multiprocessors
- Cache Coherence
- Memory Ordering
- Synchronization
- Interconnects

Hardware Jargon

- Processor & core used interchangeably
- Cores run threads
- Each chip has multiple cores
- Cores may be heterogeneous
  - CPU cores vs. GPU cores vs. accelerators
- Each board can have multiple chips/sockets
- Each platform can have multiple boards
  - Mobile platforms usually a single chip/single board
  - Datacenters 10’s to 100’s of thousands of boards

Issues: Where we are going?

- How do we connect the cores together?
- How do we make sure they see one copy of memory (or something we can understand & program)?
- How do we implement synchronization?
- How do we build cores that run multiple operations with a single instruction?
- How do we build cores that run multiple threads?
- What is inside a GPU?

Review: Sources of Cache Misses

- Computer Organization
  - 3 C's: Compulsory, conflict, capacity misses
- 4th type of miss:
  - **Coherence**: reads/writes from multiple processors

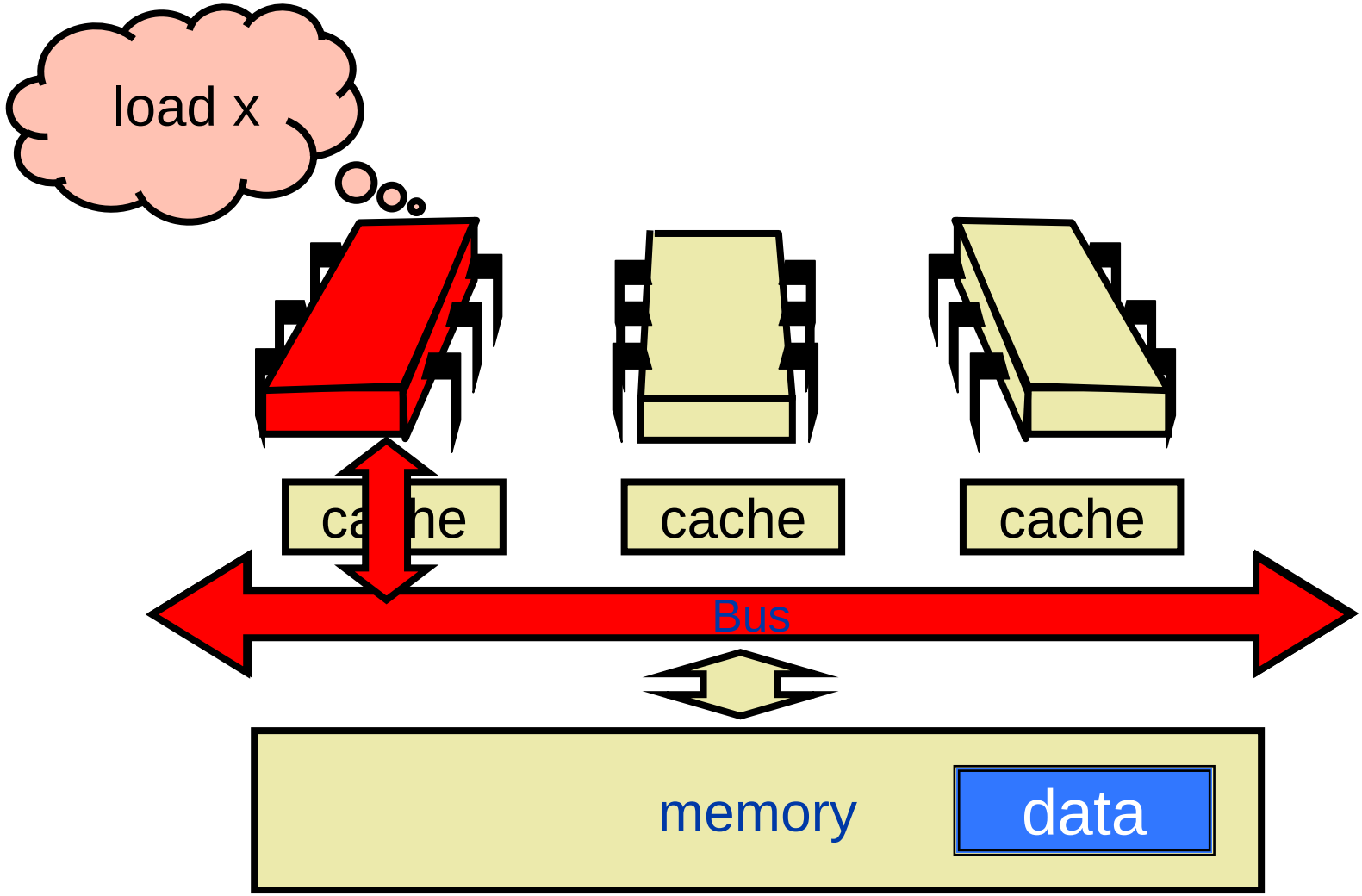
=> 4 C's: Compulsory, conflict, capacity, coherence misses

Why Cache Coherence?

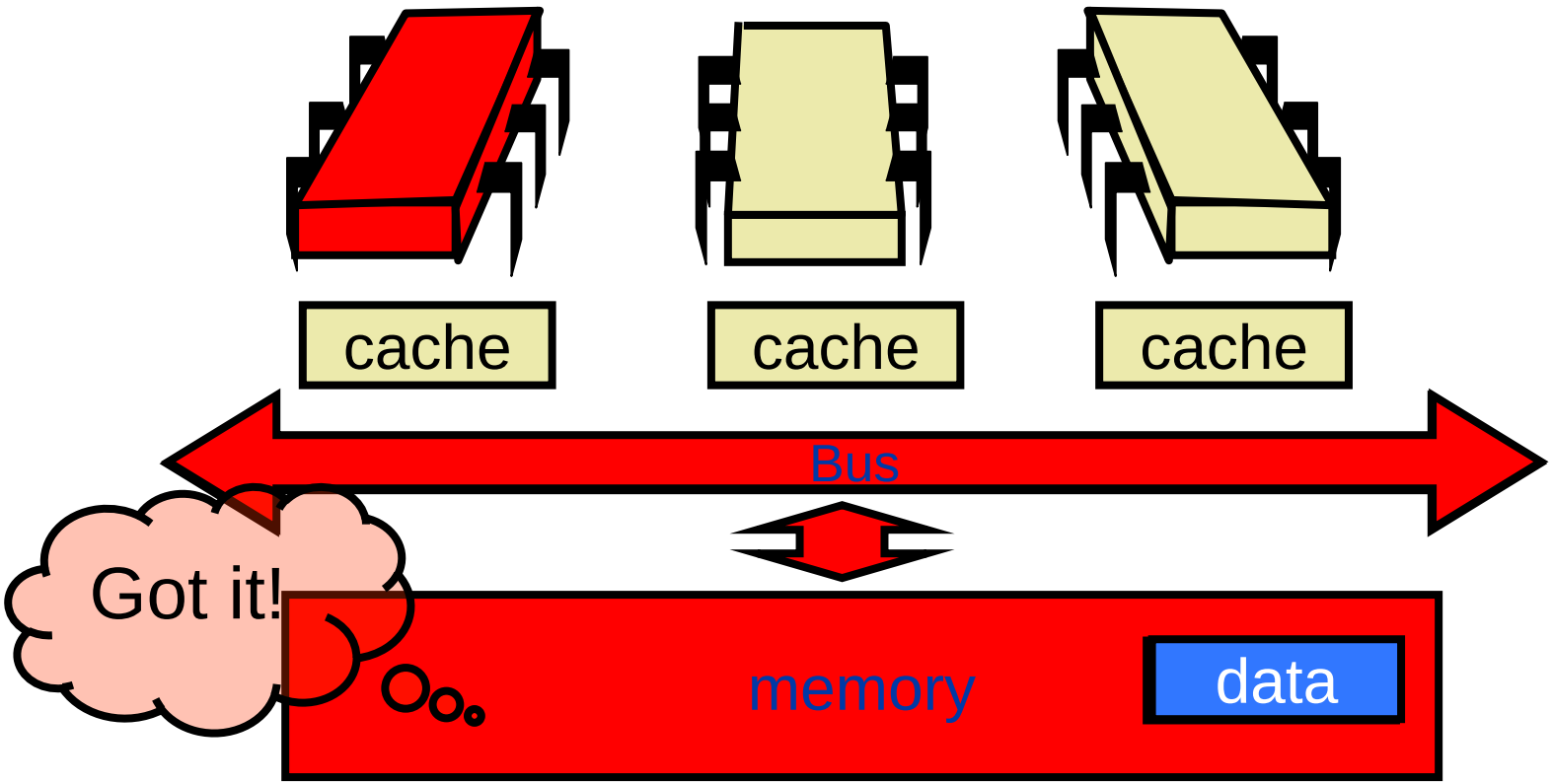
- Initially $x = 0$
- A and B both read and cache address x (with value = 0)
- A writes to *x*, new value is 42
  - Updates its cache, following reads by A return 42
- B re-reads *x*, should read 0 or 42?

- How does B find out there was a change?

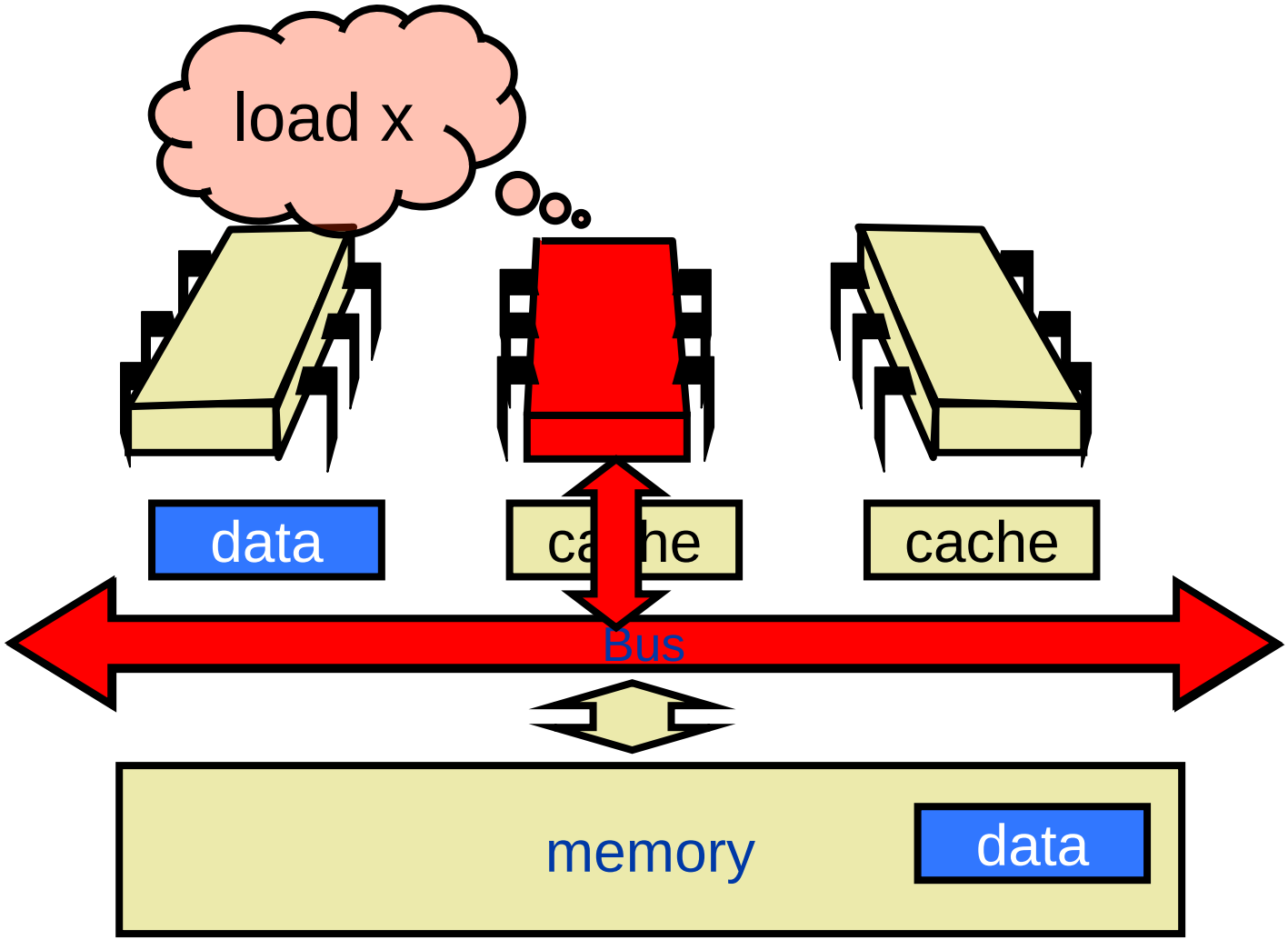
Processor Issues Load Request



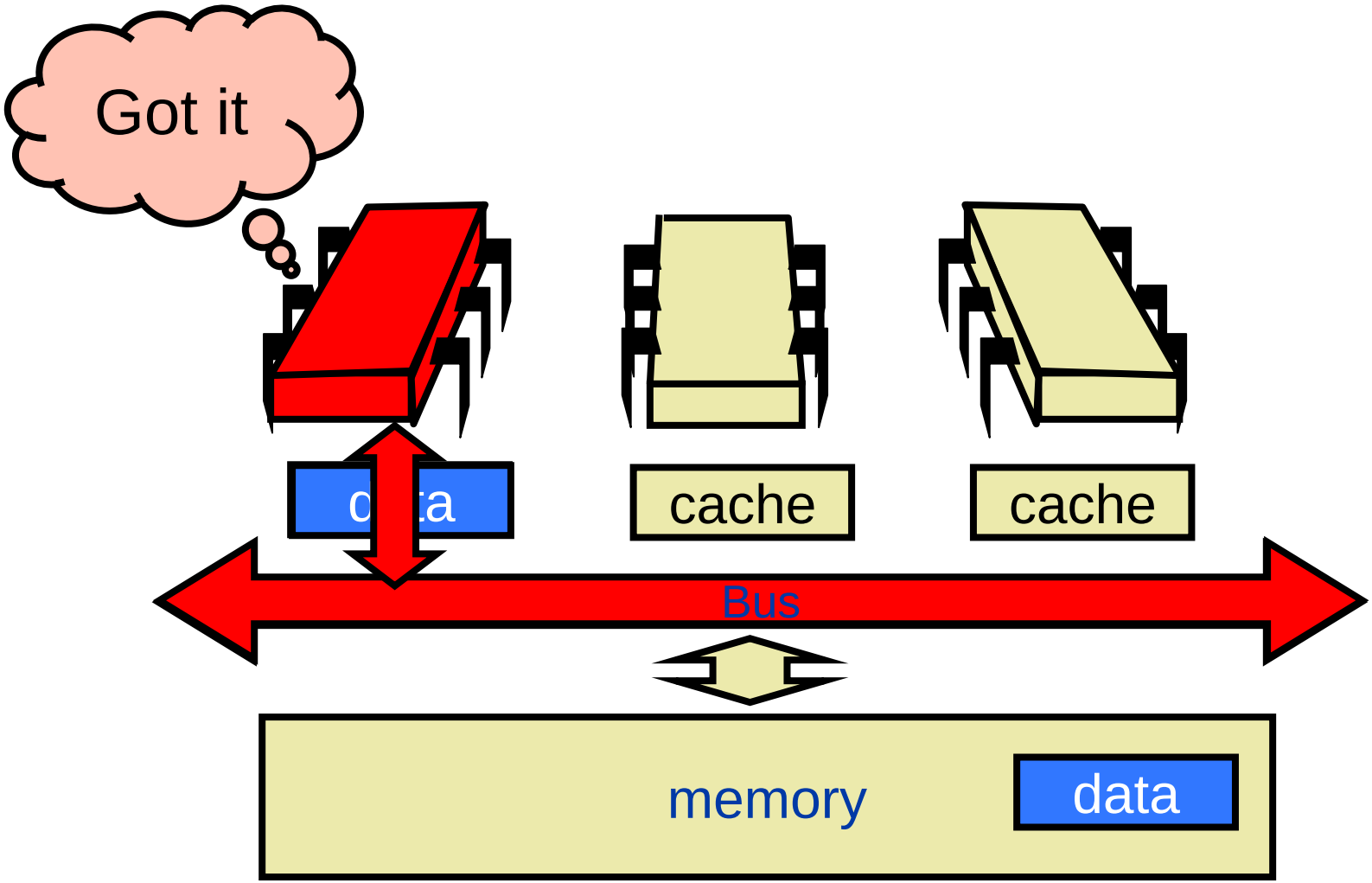
Memory Responds



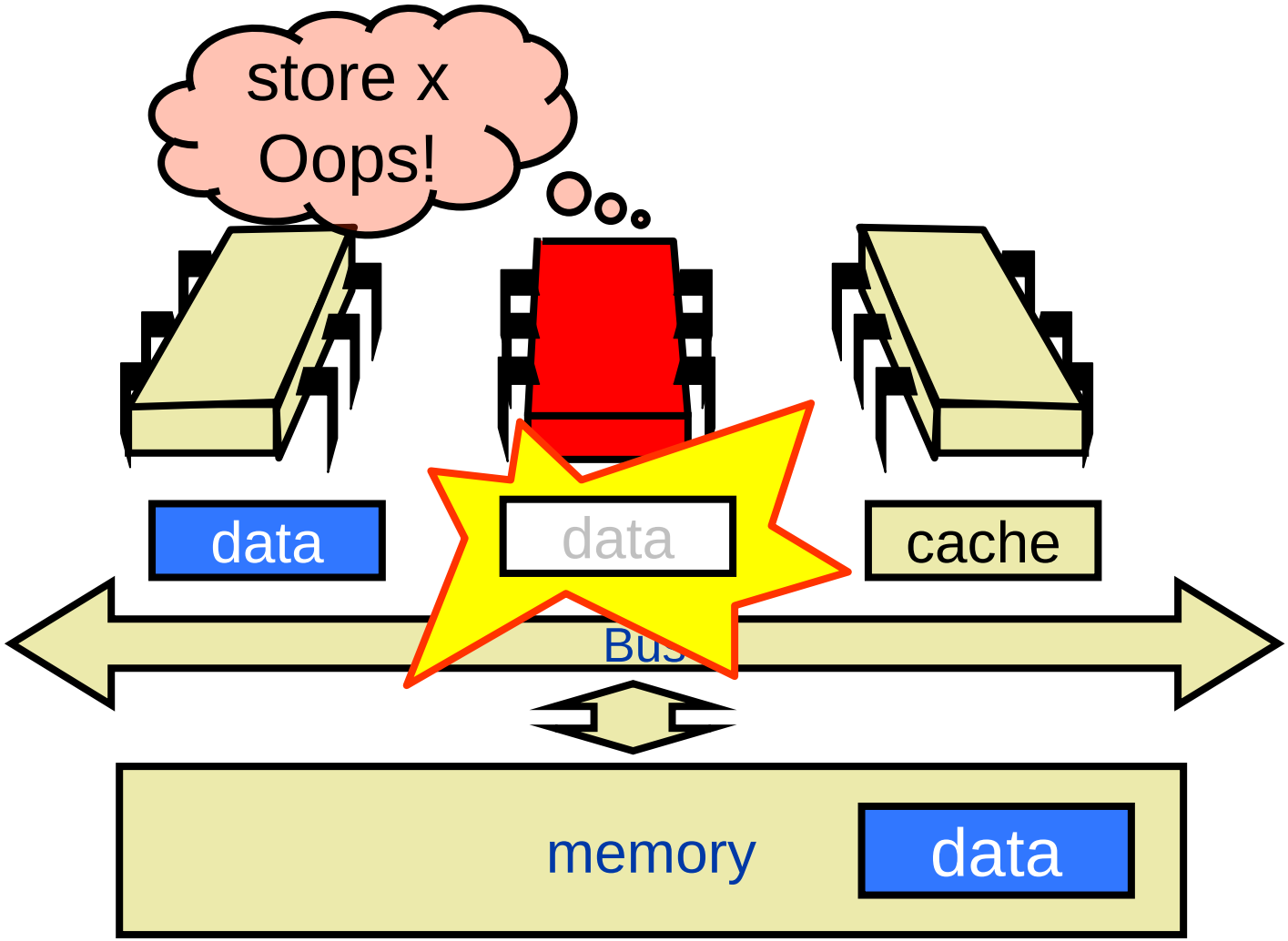
Processor Issues Load Request



Other Processor Responds

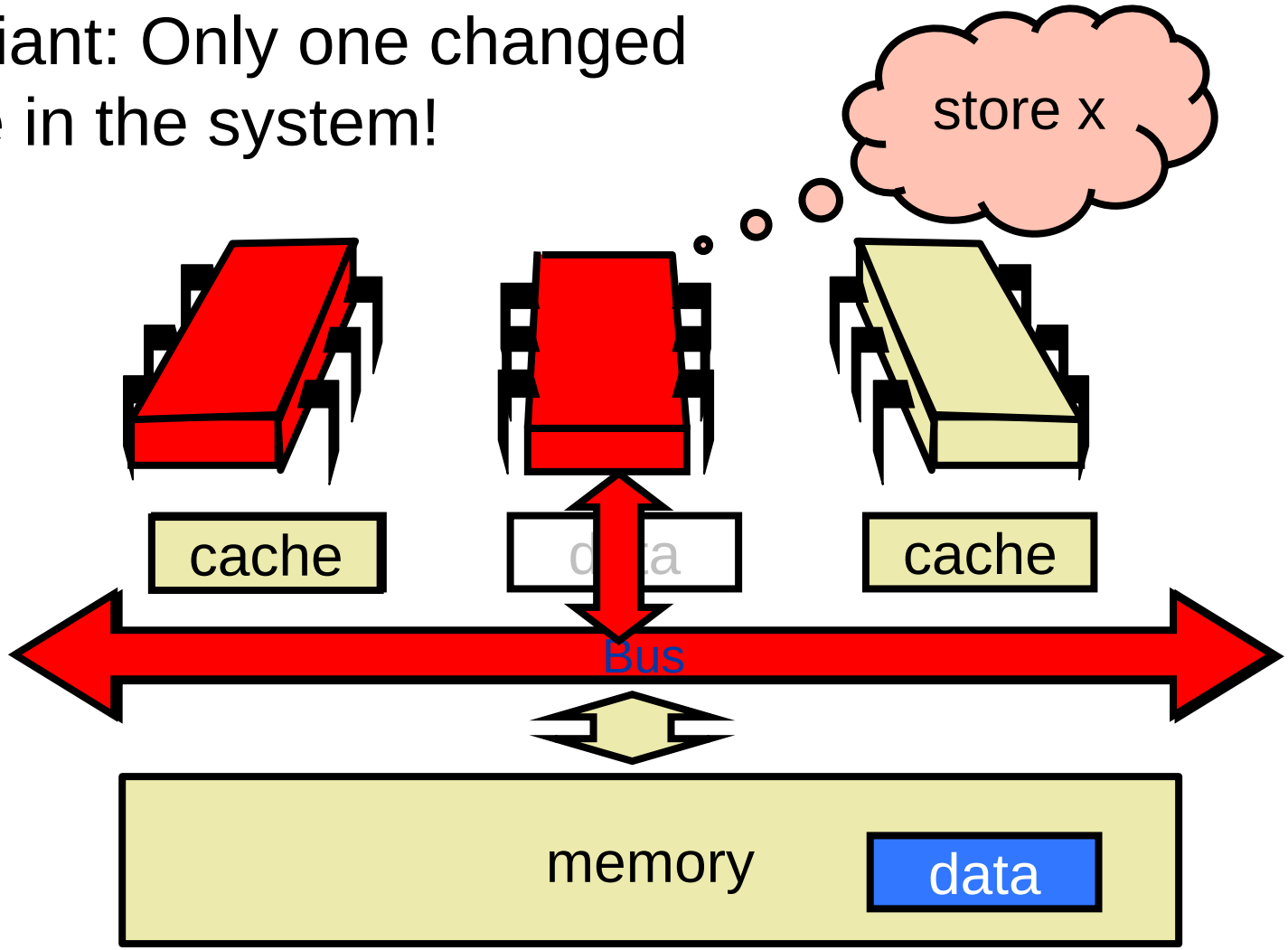


Modify Cached Data



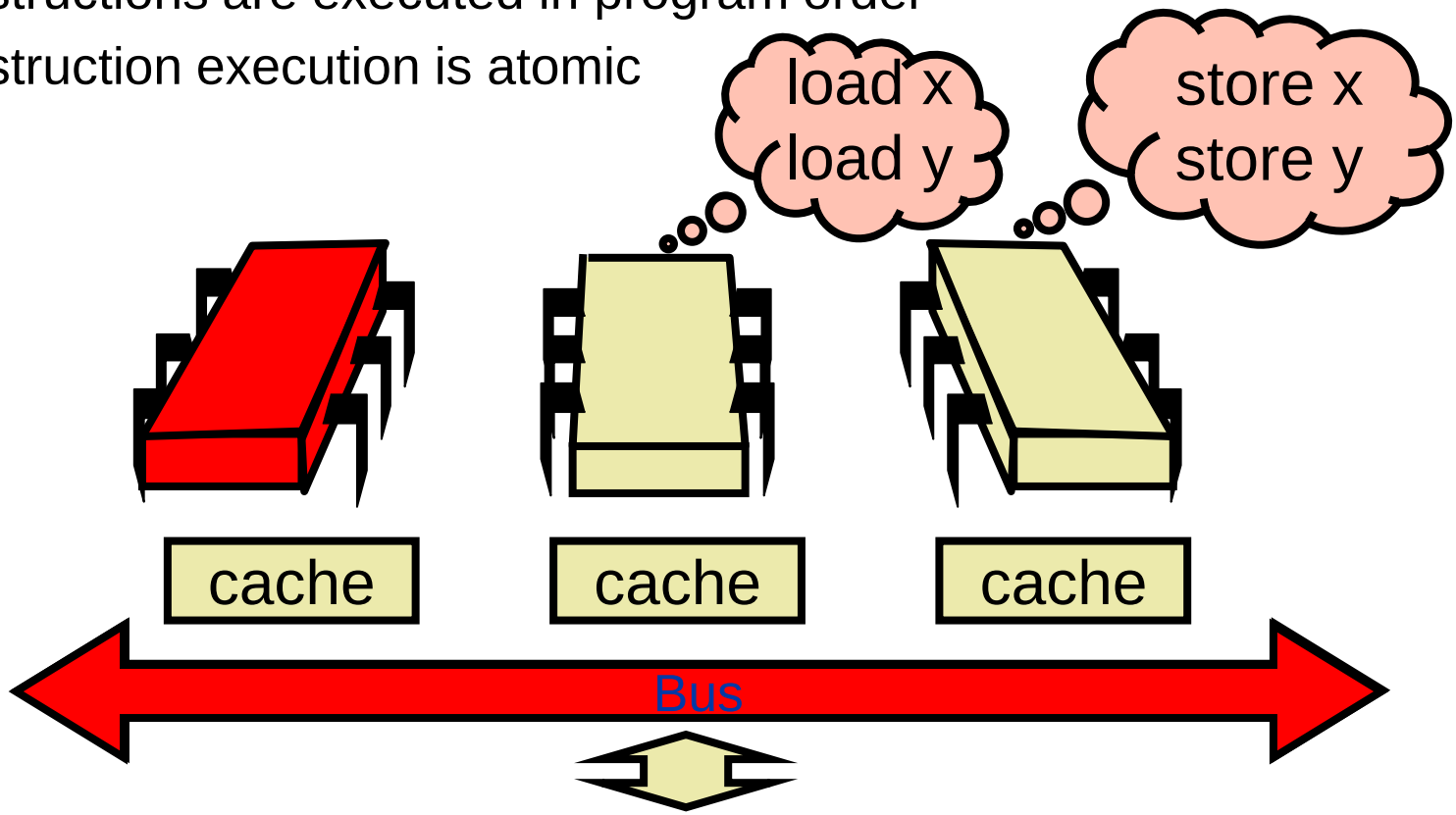
Now Store

Invariant: Only one changed value in the system!



Memory Ordering

- Programs are written *assuming*
  - Instructions are executed in program order
  - Instruction execution is atomic

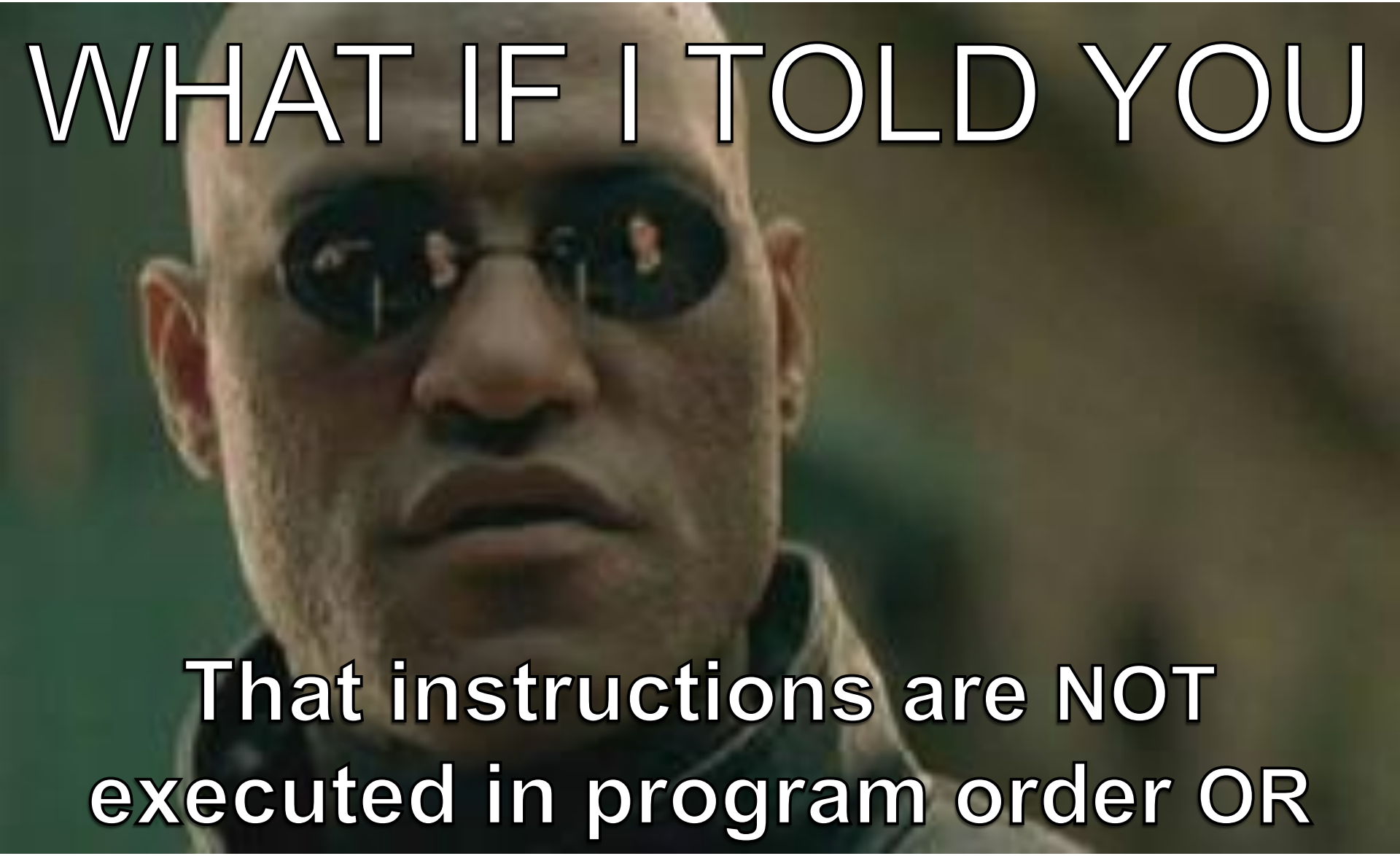


- What are possible interleavings of the loads/stores?

Example: Memory Ordering

In the previous example:

- Assume x and y have the value 0 to begin with
- Both stores write the value 1 into x and y
- What are all possible state values read in various interleavings?


WHAT IF I TOLD YOU
That instructions are NOT executed in program order OR atomically?

Memory Ordering

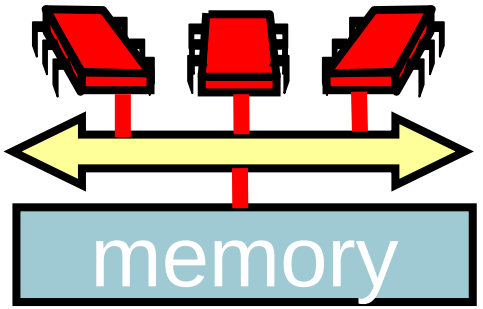
- Loads/stores are not atomic
  - Stores are written in buffers and not visible by all
  - Caches allow multiple outstanding misses
- Instructions are not executed in program order
  - Cores execute out of order
  - Compiler moves instructions around
- Simple model is “Sequential Consistency”
  - we need to define more realistic models
  - …and to implement them

Synchronization

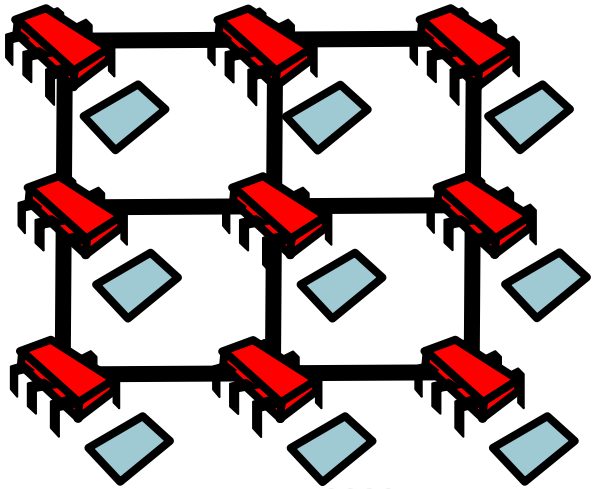
- A spectrum of synchronization primitives is defined
  - Lock, compare and swap, etc
  - Barrier
  - Messages
- Higher level constructs: Transactional Memory (in Haswell, etc)

Interconnects

- Bus
  - Broadcast medium
  - Connects a few cores/chips
  - Few wires => low B/W

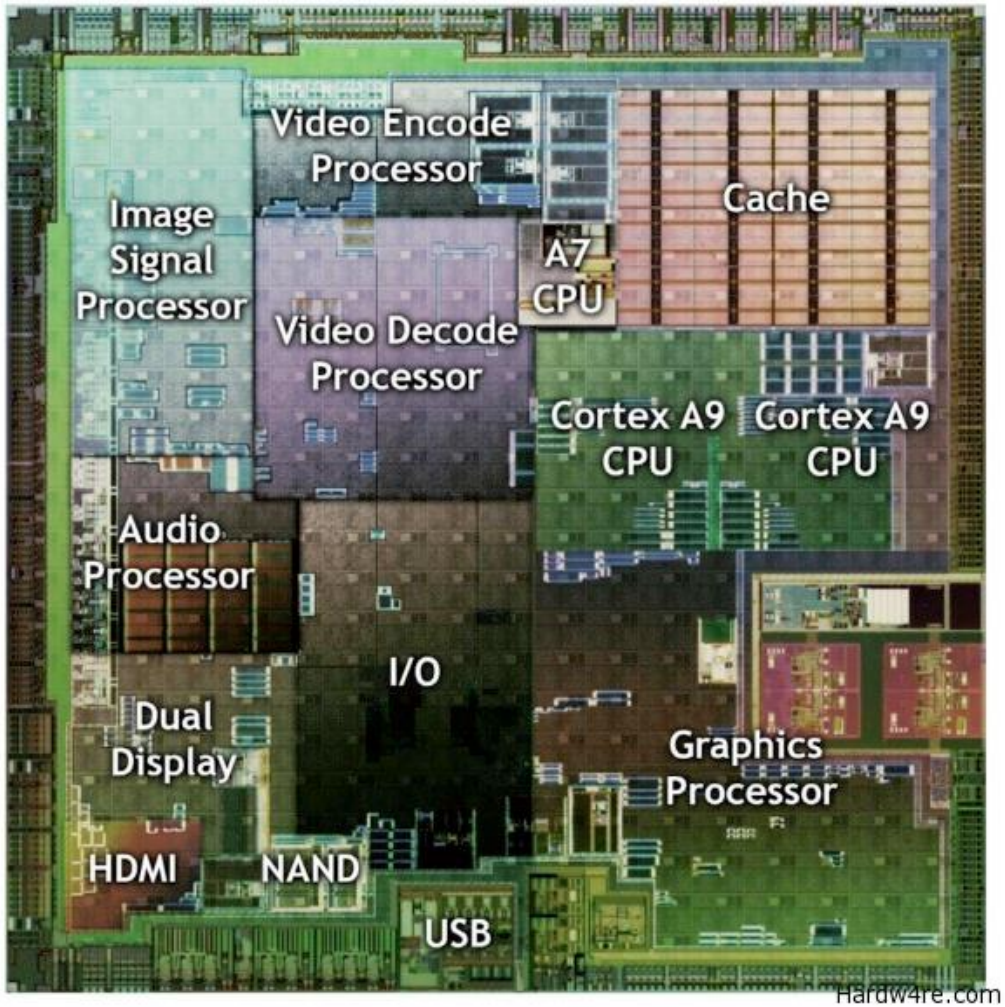


- Network
  - Packet-switched medium
  - Non-atomic transactions
  - Scalable (lots of connectivity)
  - Connecting more cores/chips
  - Many wires => high B/W, but multiple steps => higher latency



Modern mobile chip: Heterogeneous

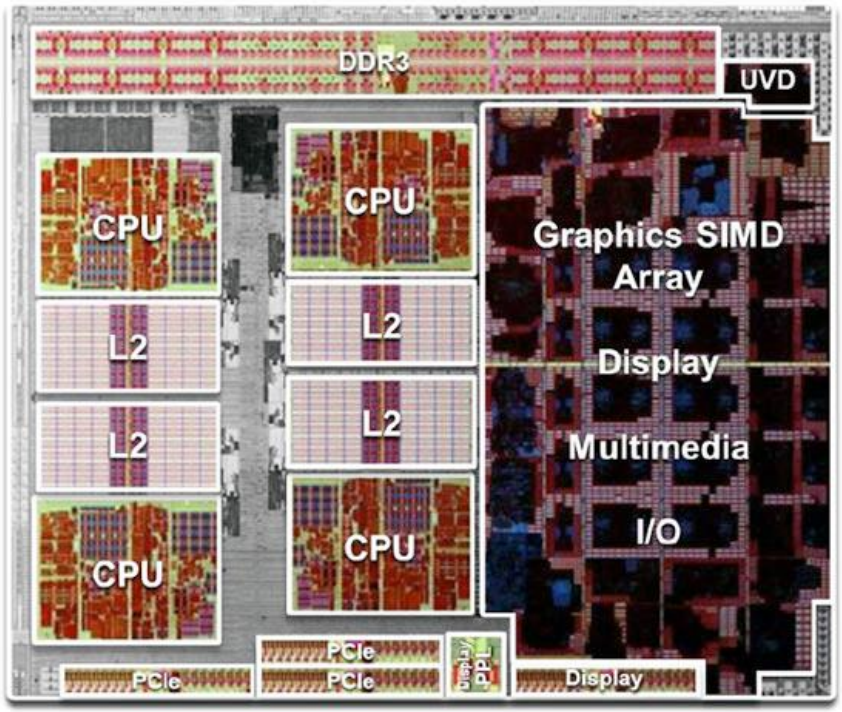
- Specialized processors everywhere
- Maximize ops/sec
- Minimize joules/op
- Exploit parallelism
- Example cores:
  - Superscalar
  - Graphics
  - Media
  - Security



nVidia Tegra 2

Modern tablet chip: CPU + GPU

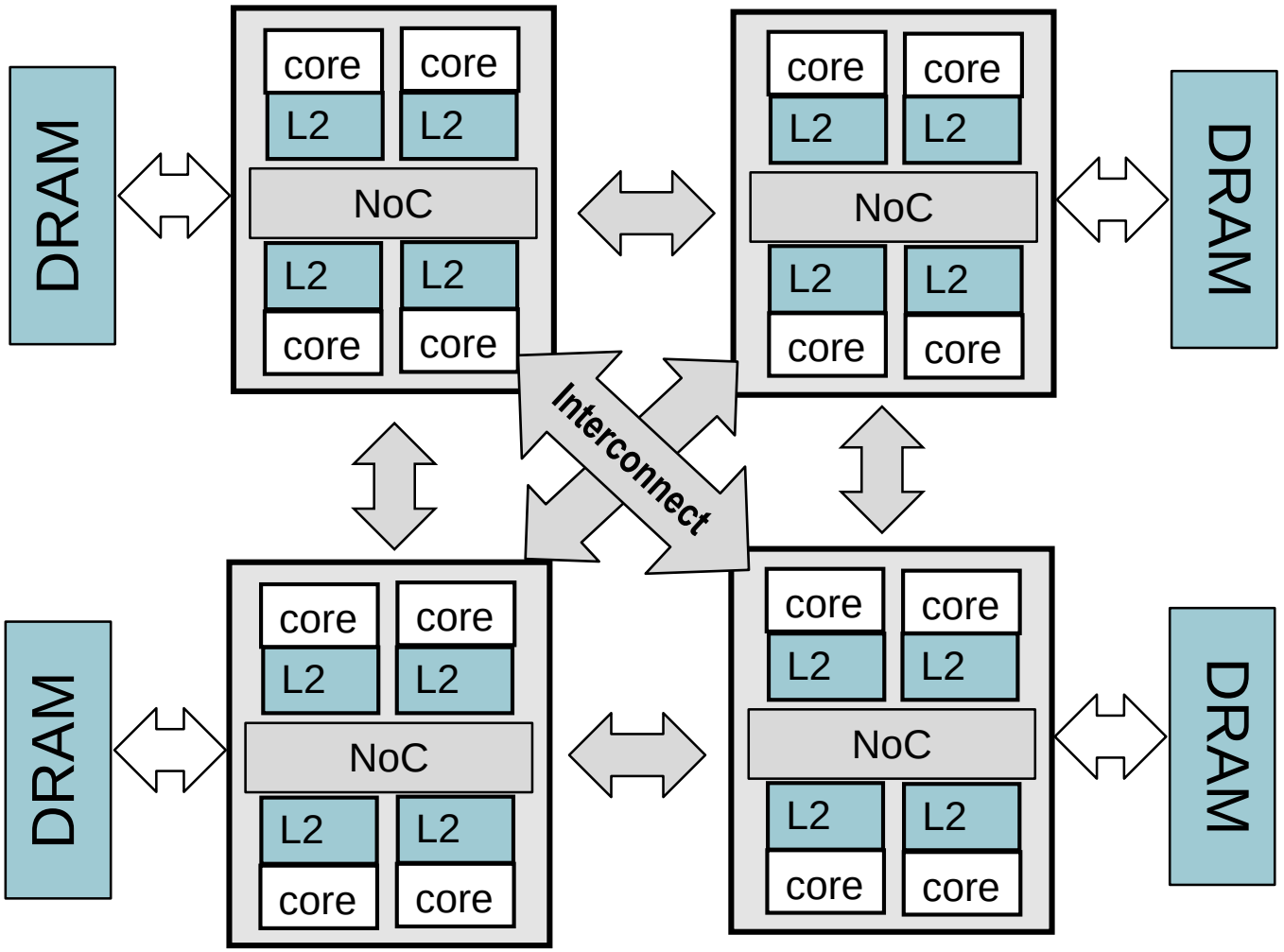
- A few CPUs
  - General-purpose computation
- One big Vector accelerator
  - Multimedia
  - Graphics
  - Games
  - SIMD computation
- Coherent shared memory
  - CPUs + Accelerator can communicate on-chip



AMD Fusion

Modern servers: NUMA (non-uniform access)

- Multiple CPUs/chip
- Multiple levels of memory
- Multiple levels of network



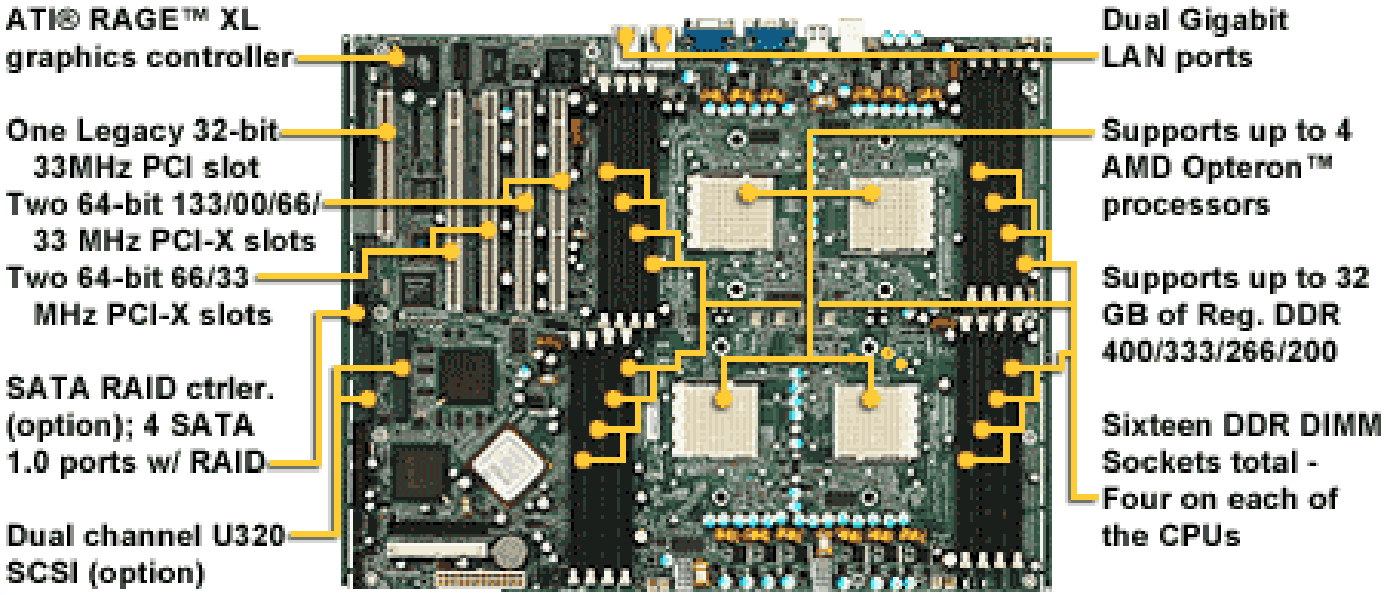
NUMA systems

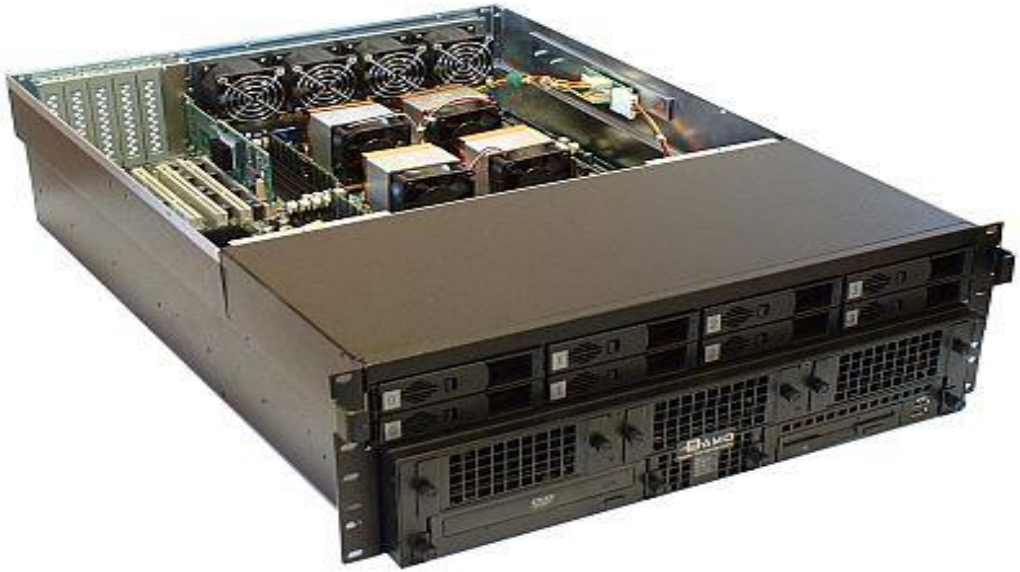
- Non-uniform latency to memory
  - First-level cache ~ 2 cycles (=> 4 !)
  - Second-level cache ~ 10 cycles
  - Third-level cache ~ 50 cycles
  - Off-chip own DRAM ~ 150 cycles
  - Off-chip others' DRAM ~ 300 cycles

- Contention everywhere
  - On-chip network
  - Off-chip network
  - Cache ports, memory channels and cache/memory controllers

Data placement is everything!

Example: AMD Quad Opteron





- All coherence and multiprocessing glue in processor chip & module
- Highly integrated, targeted at high volume
- Low latency, moderate bandwidth

Summary

- Multiprocessor architecture
  - Cache coherence
  - Memory ordering
  - Interconnects
  - Multithreading/Vector/GPU
  - Lots of fundamental tradeoffs to consider in design
  - Lots and lots of critical techniques to handle multiprocessor problems
  - Lots of interesting examples of good/bad design choices