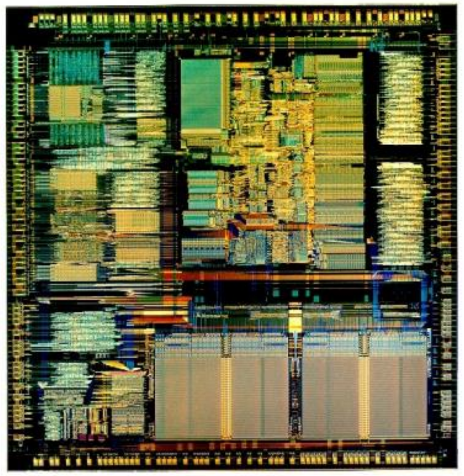
ECE411: Computer Organization and Design

Lecture 16: Multi-core and Multi-threading

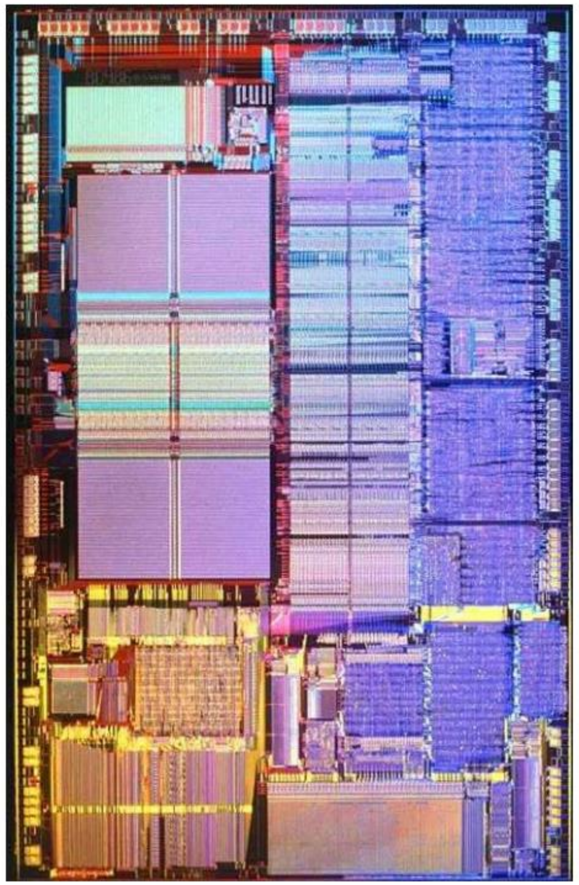
Rakesh Kumar

ILLINOIS

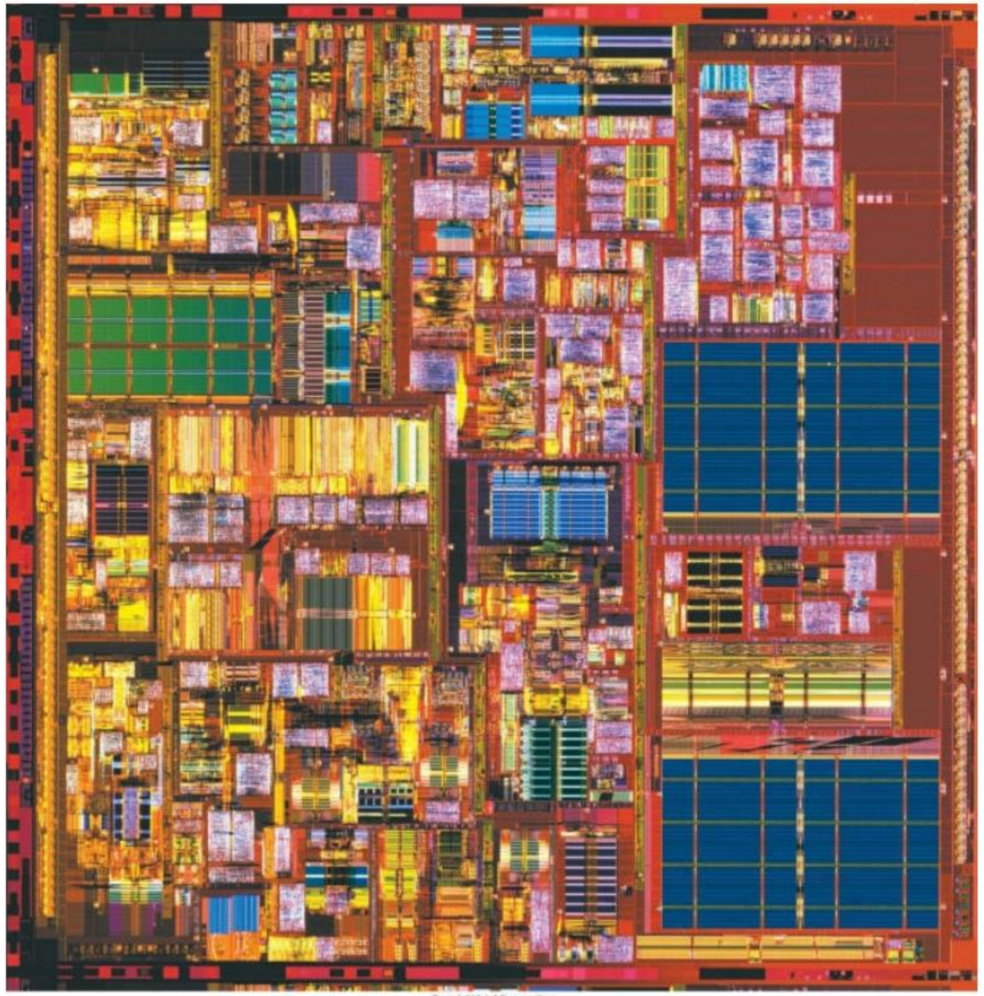
Uniprocessor (single-core) Architecture



Intel 386



Intel 486

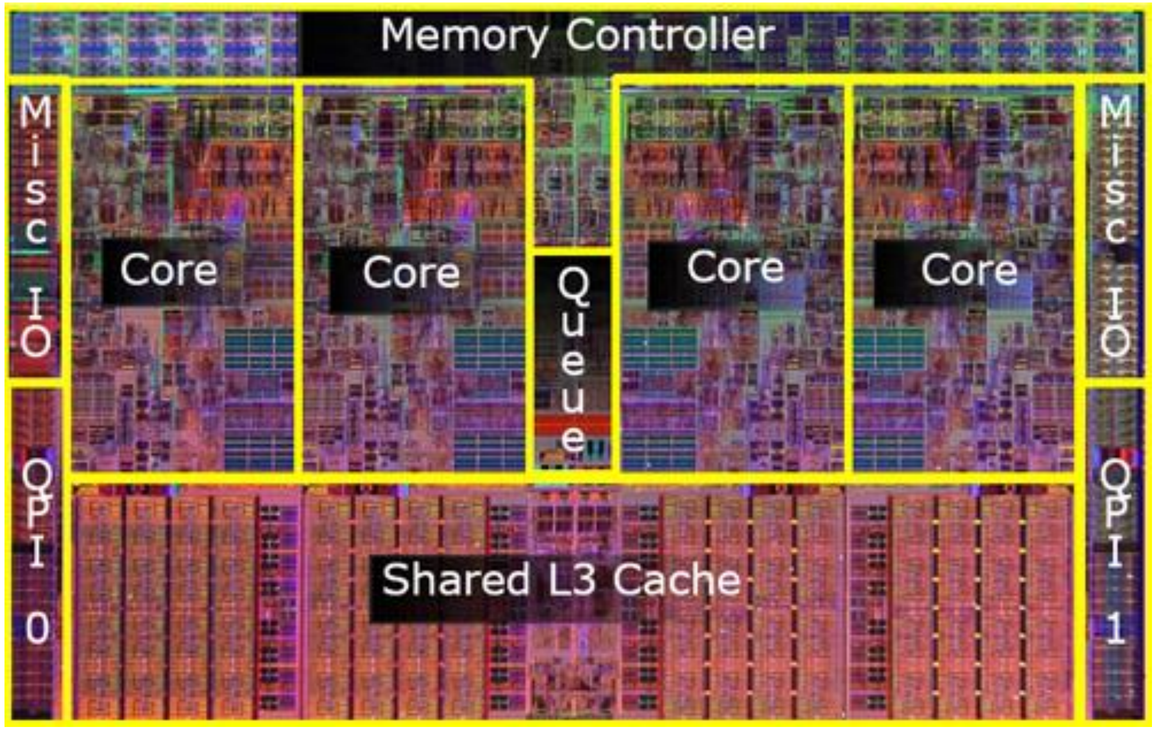


Intel Pentium

Multi-core Architecture

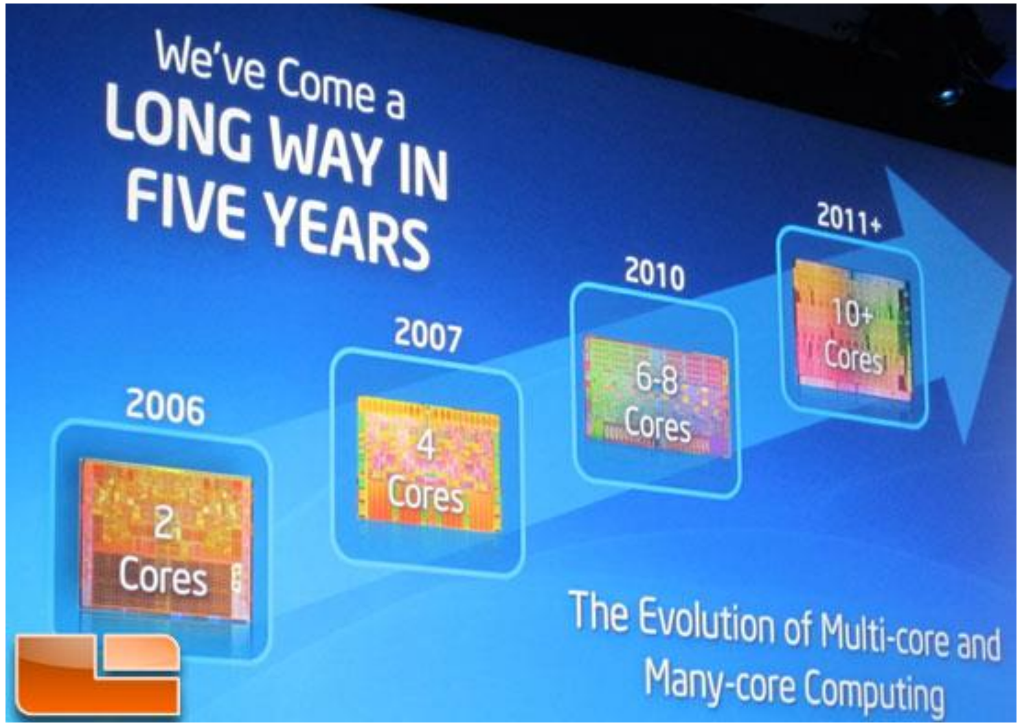
- Performance demand from applications
- Faster computation

Intel Nehalem – First true multi-core processor

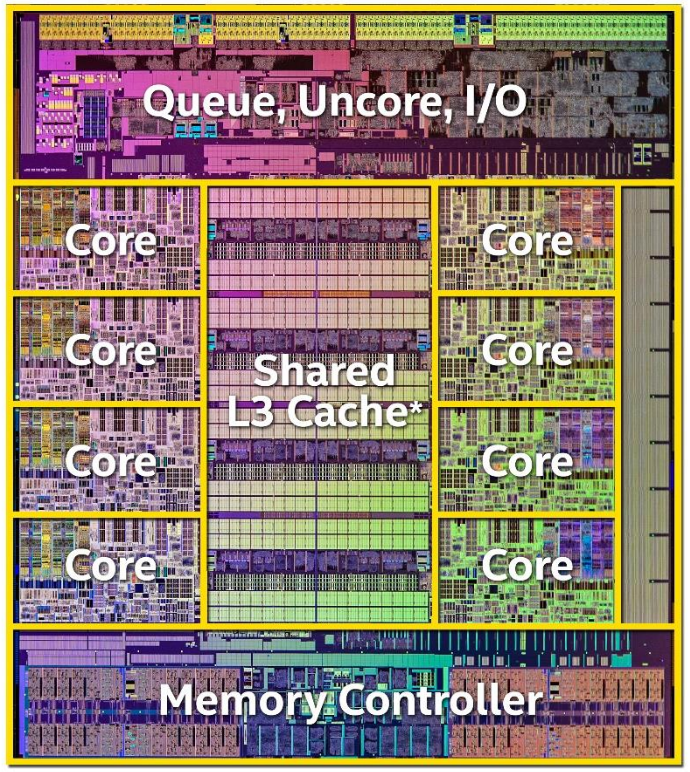


Multi-core Architecture

- Performance demand from applications
- Faster computation


We've Come a LONG WAY IN FIVE YEARS
2006
2 Cores
2007
4 Cores
2010
6-8 Cores
2011+
10+ Cores
The Evolution of Multi-core and Many-core Computing

New 8-Core Intel® Core™ i7 Processor Extreme Edition



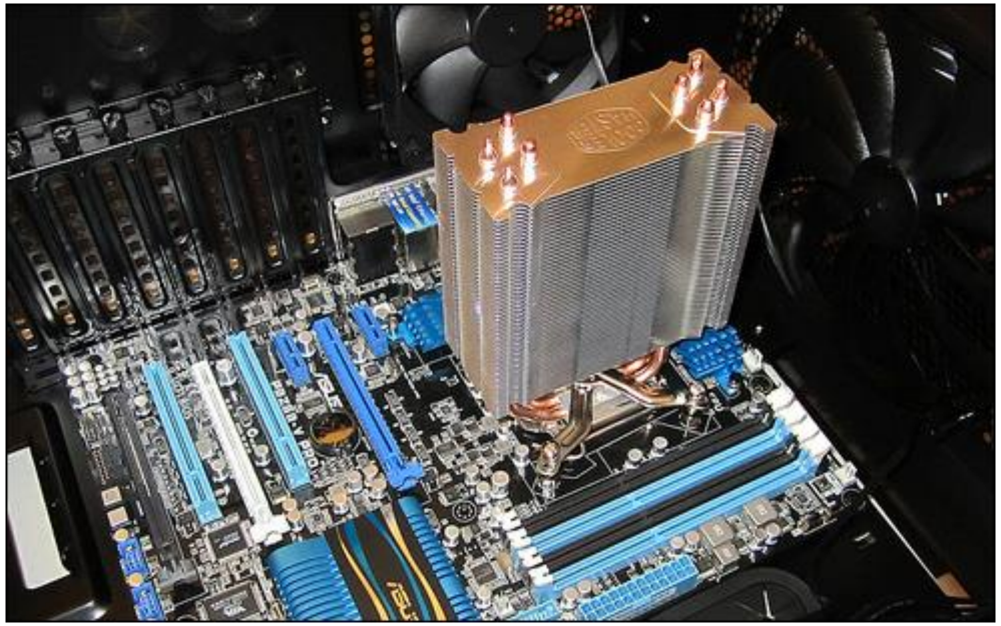
Intel® Core™ i7-5960X Processor Extreme Edition
Transistor count: 2.6 Billion
Die size: 17.6mm x 20.2mm

* 20MB of cache is shared across all 8 cores

Why Multi-core Architecture?

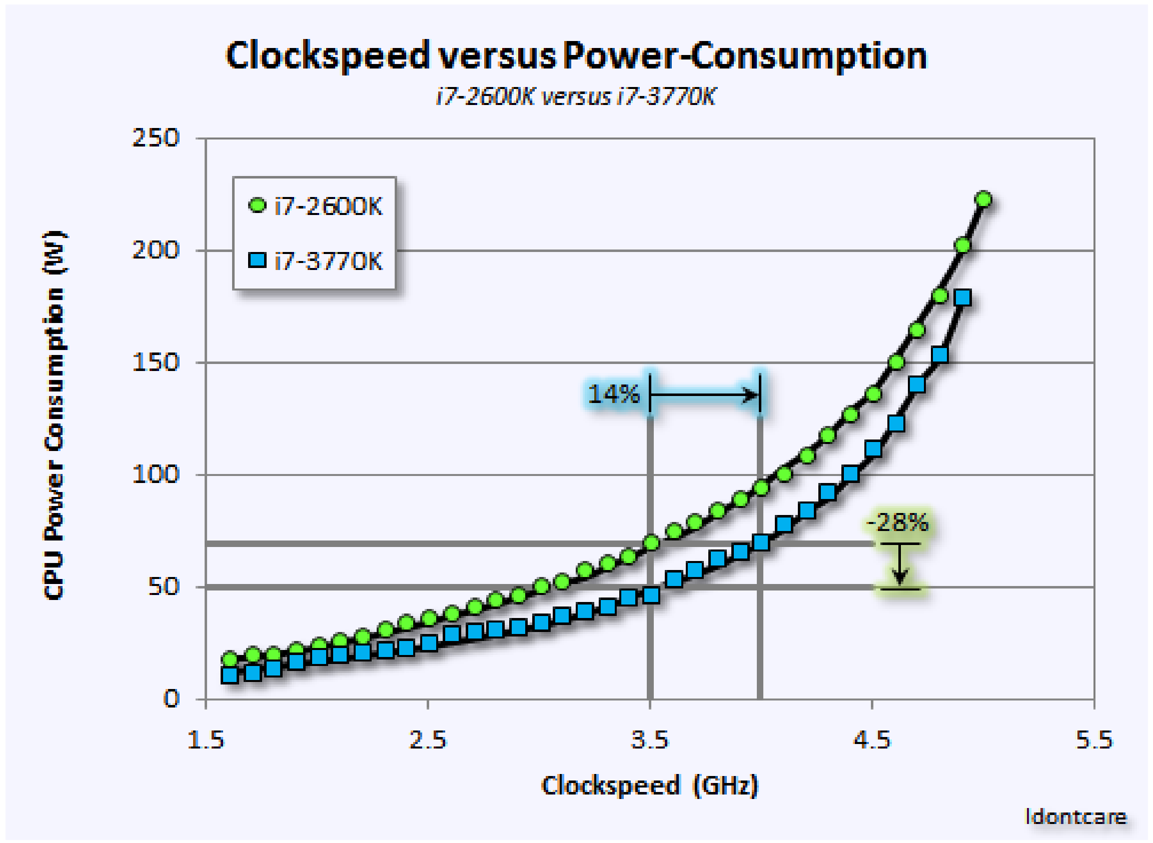
- Difficult to make single-core clock frequencies even higher
  - hard to extract more ILP b/c larger instruction windows are very expensive, slow, power-hungry circuits

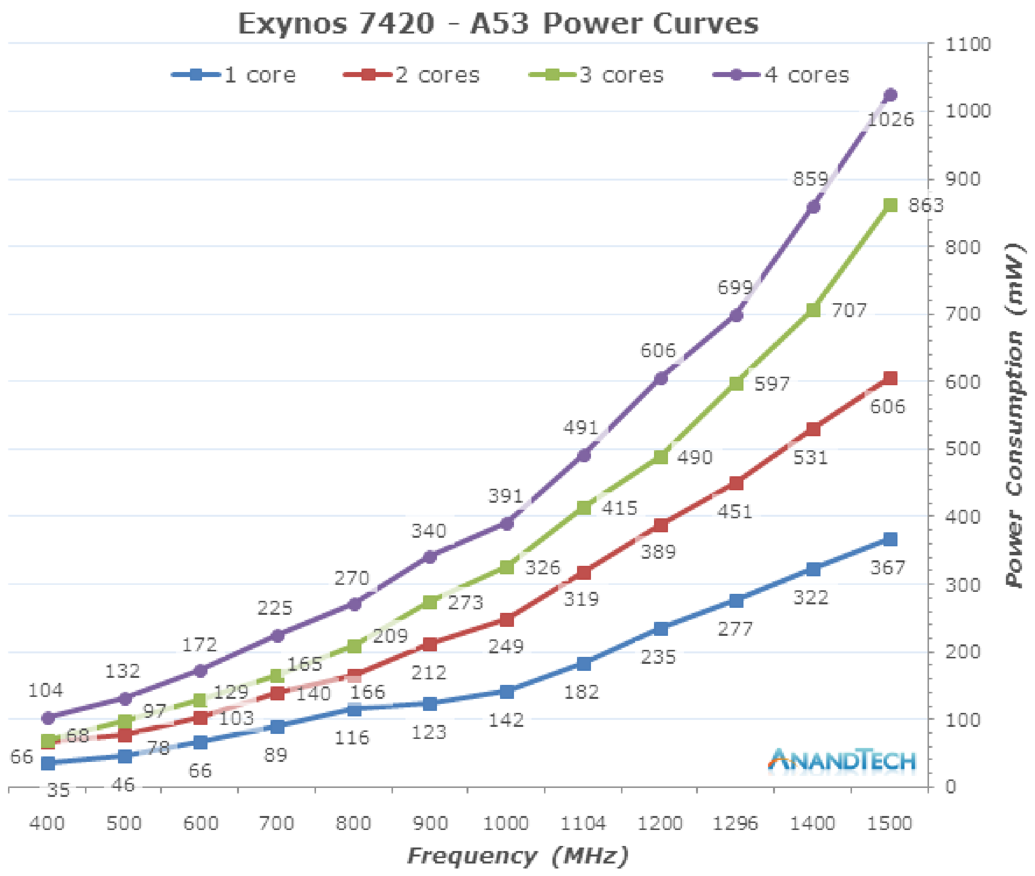
- Deeply pipelined circuits:
  - Heat problems
  - Difficult design and verification
  - Large design teams necessary
  - Server farms need expensive air-conditioning



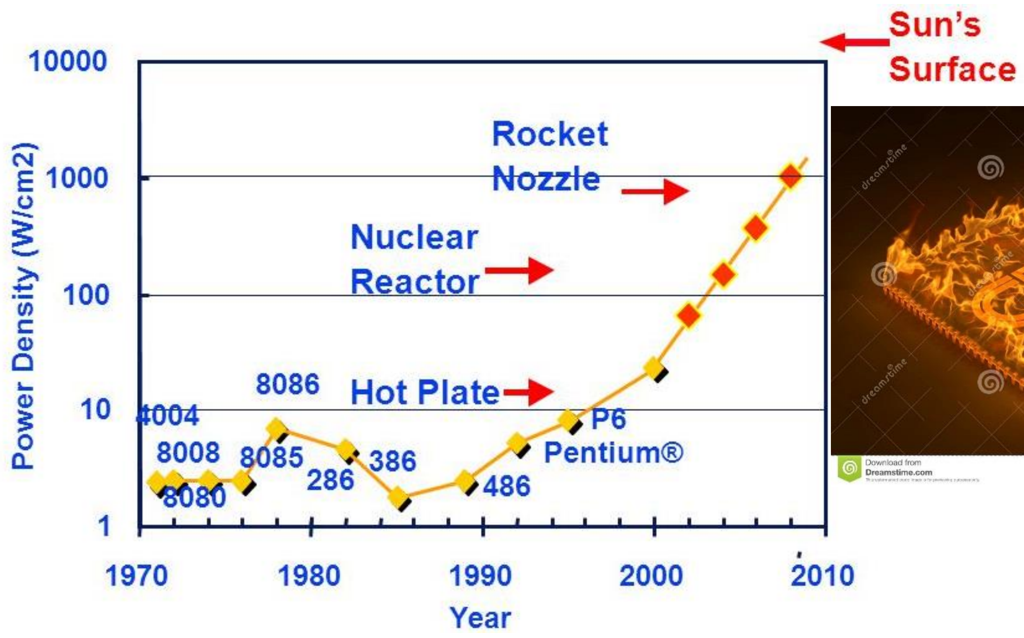
https://dl.acm.org/doi/pdf/10.5555/774861.774897

Power vs. Frequency





Power and Thermal Limit



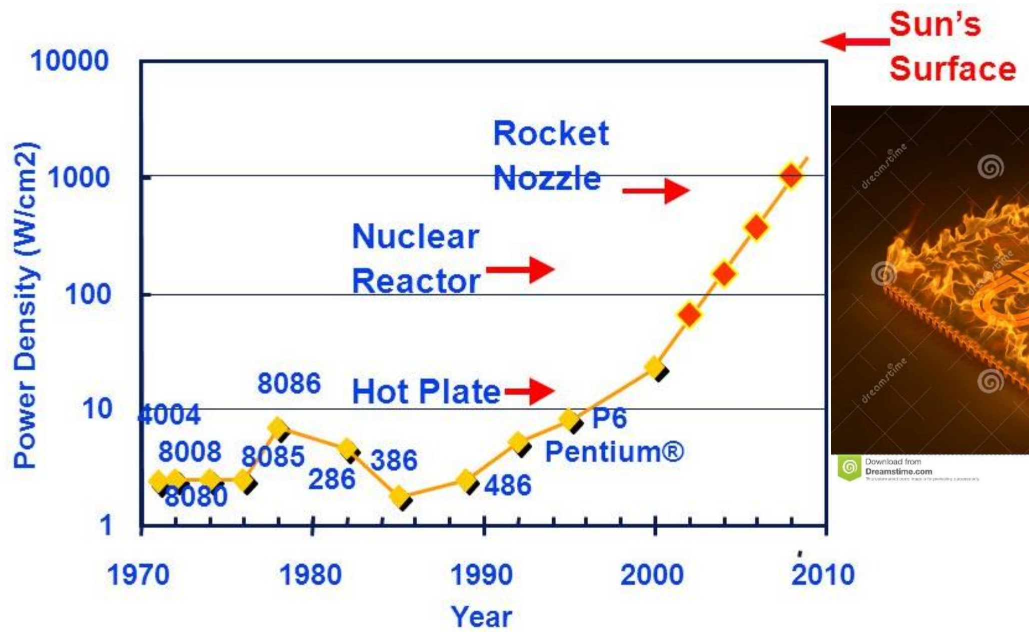
https://doi.ieeecomputersociety.org/cms/Computer.org/dl/mags/mi/2012/03/figures/mmi20120301221.gif
https://hsto.org/storage2/e68/410/9b5/e684109b586e5e9612cff3befdf36fbe.png

How to reduce the power consumption

- Multicore
  - One core with frequency of 2 GHz
  - Two cores with 1GHz frequency (each)
    - Same throughput
    - Two 1GHz cores require less power/energy

- New Challenges – Performance
  - How to utilize the cores
  - It is difficult to find parallelism in programs to keep all these cores busy

Power and Thermal Limit



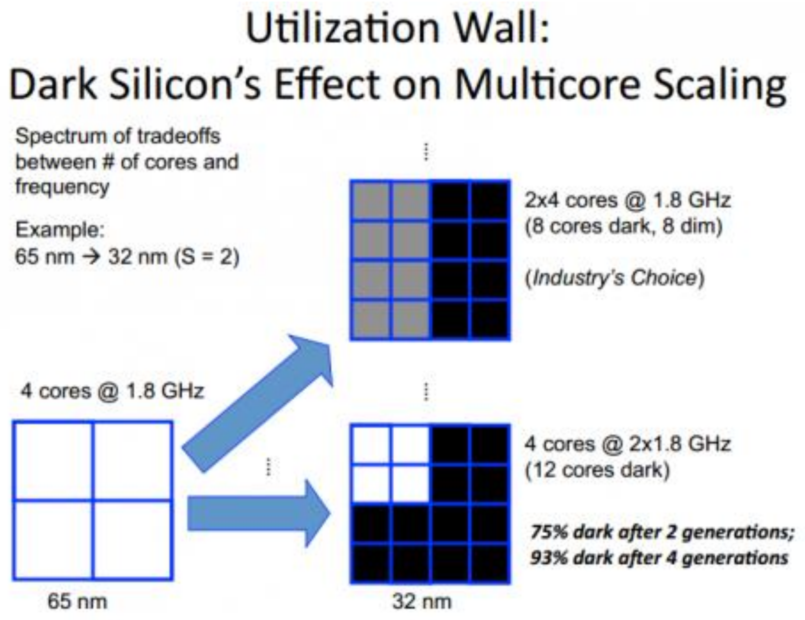


Figure 1: Multicore scaling leads to large amounts of Dark Silicon. (From[7].)

https://doi.ieeecomputersociety.org/cms/Computer.org/dl/mags/mi/2012/03/figures/mmi20120301221.gif
https://hsto.org/storage2/e68/410/9b5/e684109b586e5e9612cff3befdf36fbe.png

Why Multi-core Architecture?

- Parallel processors now play a major role
  - Logical way to improve performance
    - Connect multiple microprocessors
  - Not much left with ILP exploitation
  - Data center servers have parallelism

- Multi-core processor architectures will become increasingly attractive
  - Due to slowdown in advances of uniprocessors

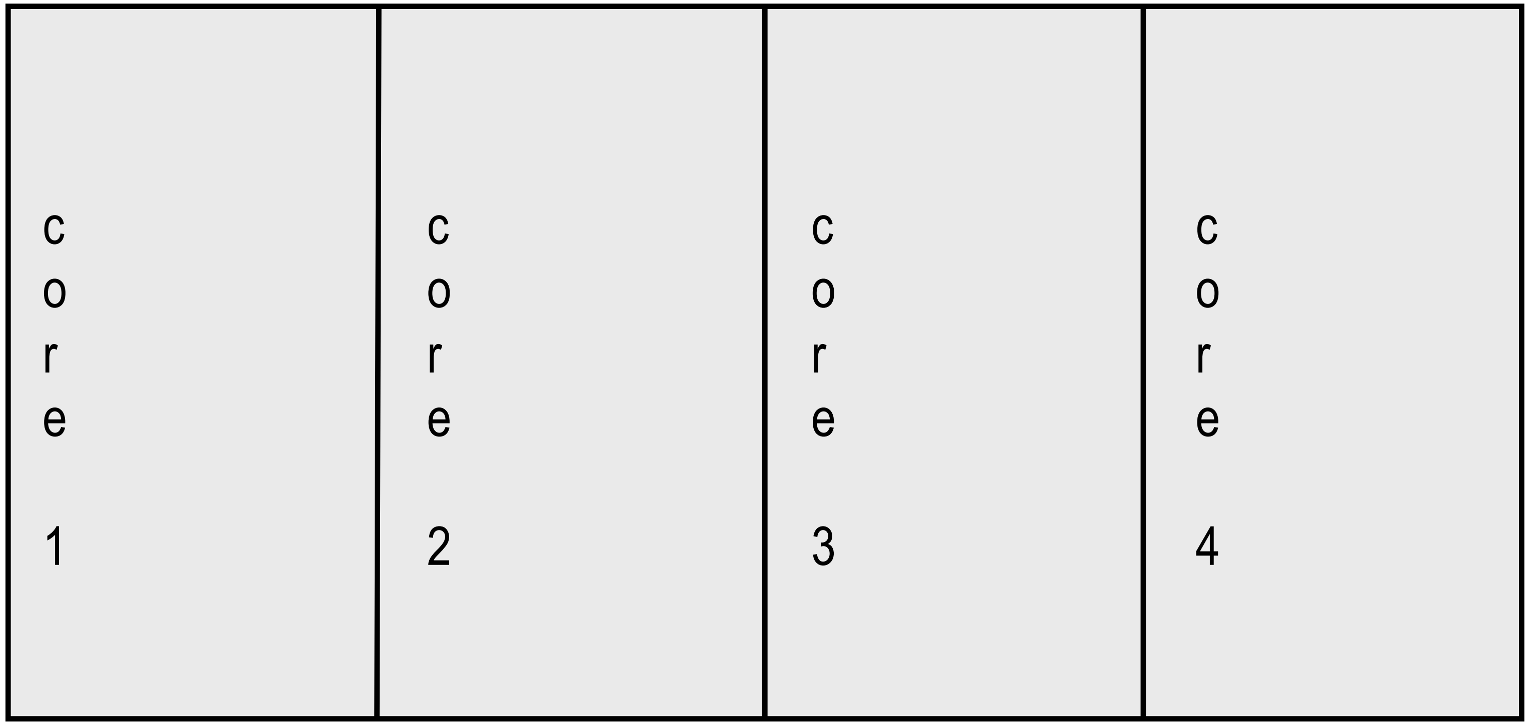
Optional Reading: https://arxiv.org/pdf/1911.11313v1.pdf
https://education.dellemc.com/content/dam/dell-emc/documents/en-us/2019KS_Yellin-Saving_The_Future_of_Moores_Law.pdf

ILP versus TLP

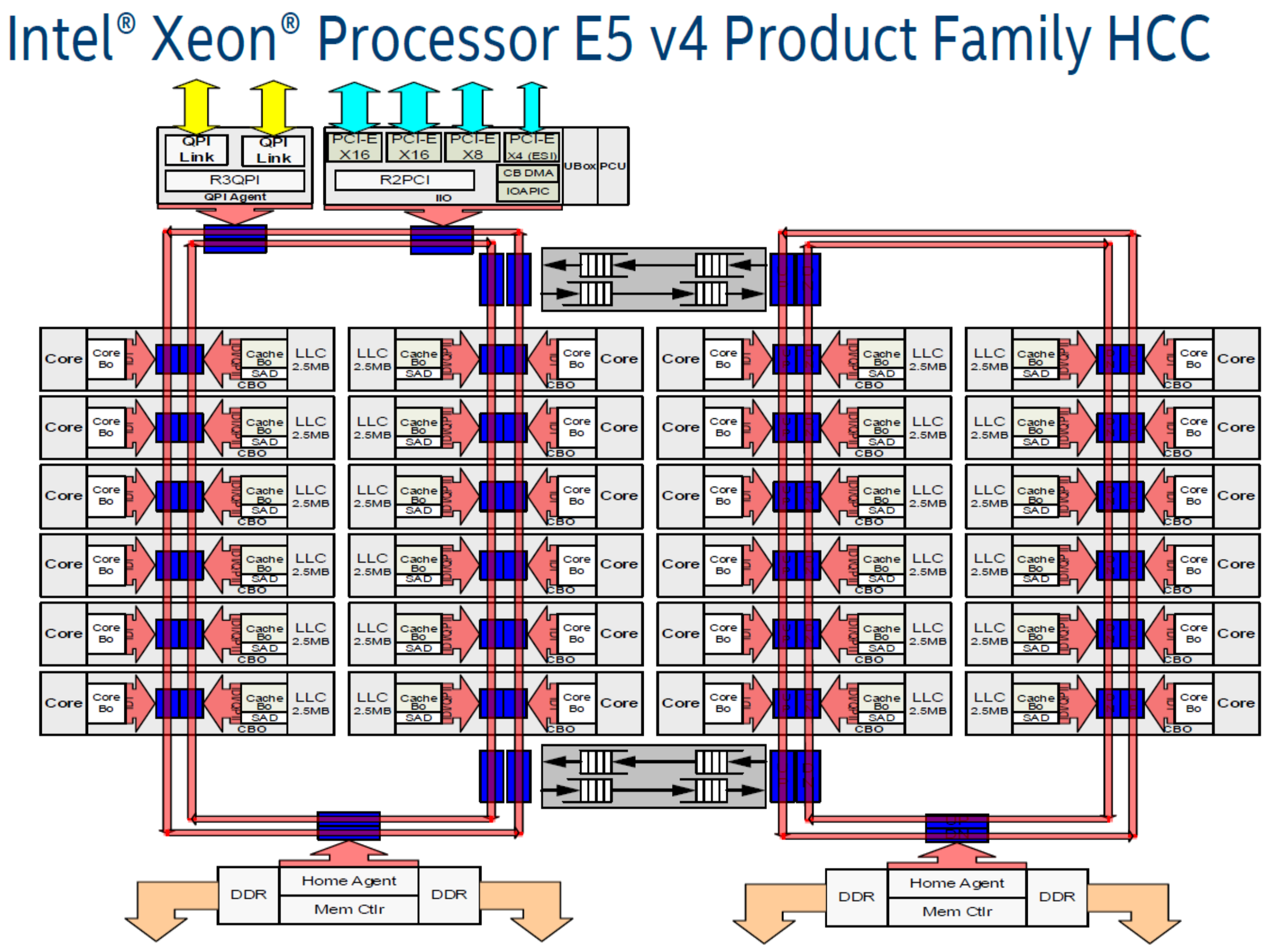
- instruction-level parallelism (ILP)
  - parallelism at the machine-instruction level
  - the processor can re-order, pipeline instructions, split them into microinstructions, do aggressive branch prediction, etc.
  - instruction-level parallelism enabled rapid increases in processor speeds over the last 15 years
- thread-level parallelism (TLP)
  - parallelism on a more coarser scale
  - server can serve each client in a separate thread (web server, database server)
  - a computer game can do AI, graphics, and physics in three separate threads
  - single-core superscalar processors cannot fully exploit TLP
  - multi-core architectures are the next step in processor evolution: explicitly exploiting TLP

Multi-core CPU Chip

- Multiple cores fit on a single processor socket
  - also called CMP (Chip Multi-Processor)



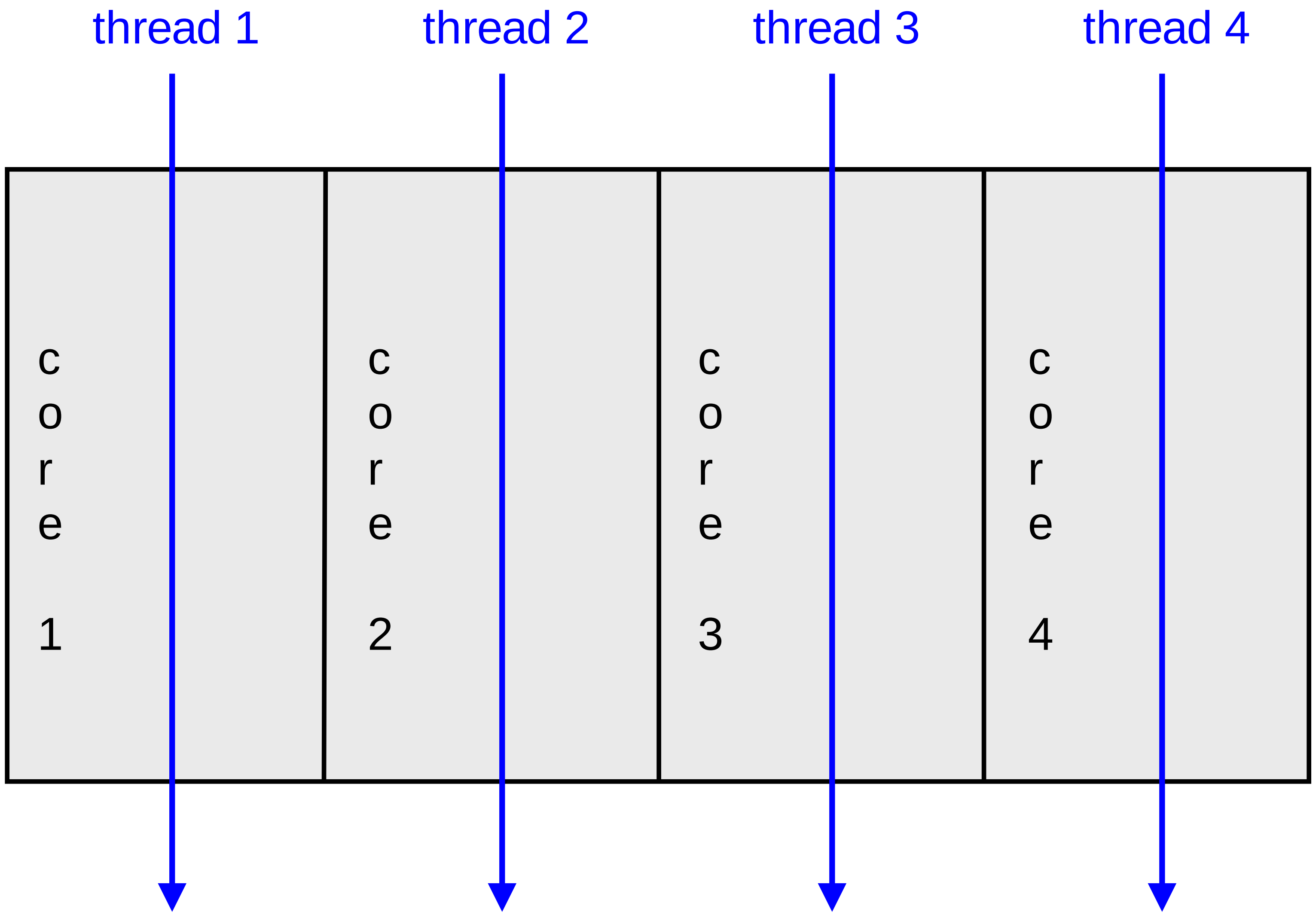
Multi-core Architectures



multi-core CPU chip

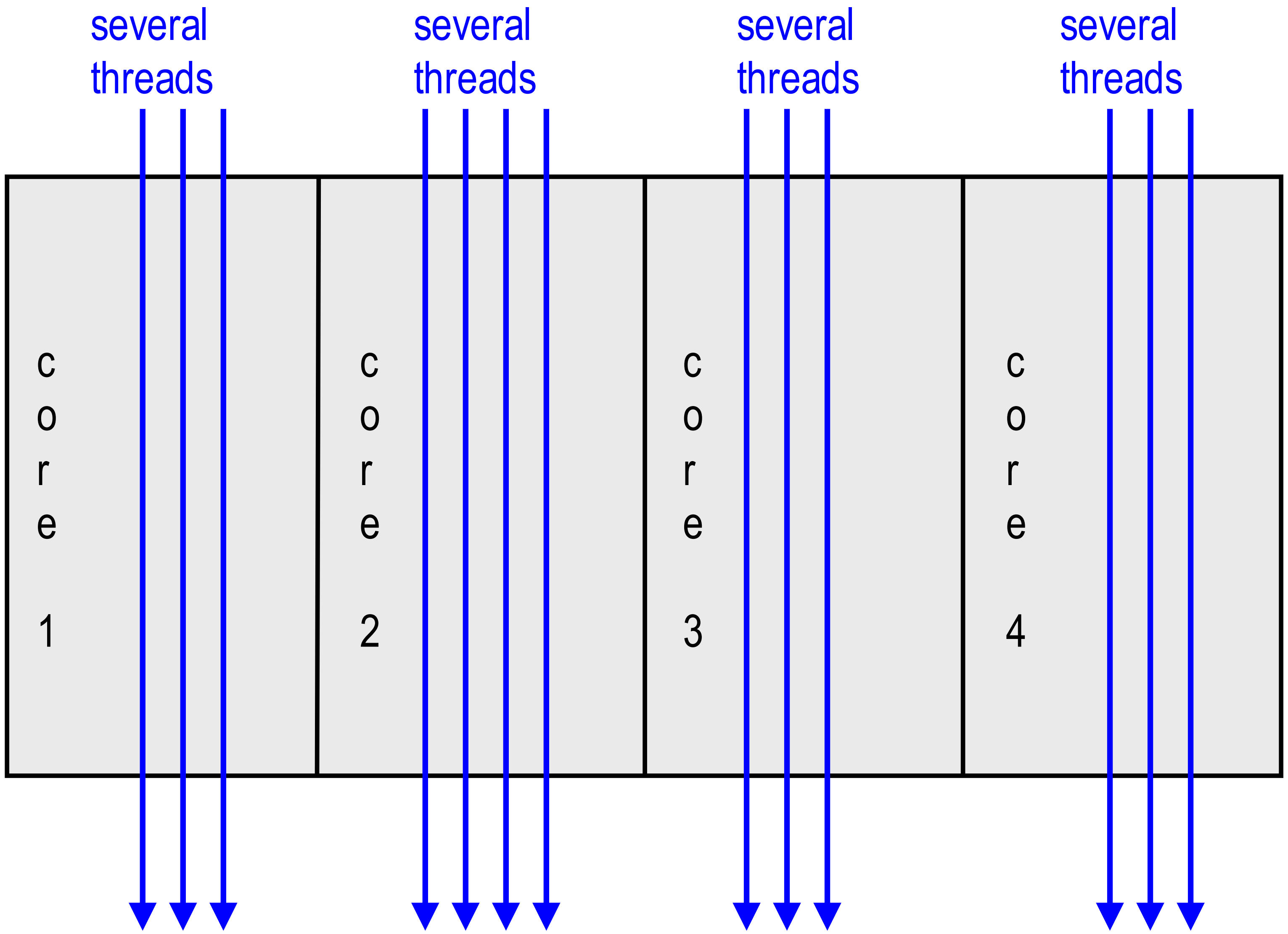
Cores Run in Parallel

- Exploiting thread-level parallelism



Cores Run in Parallel

- Threads can be time-sliced (just like on a uniprocessor) in each core



Interaction with OS

- OS perceives each core as a separate processor
- OS scheduler maps threads/processes to different cores
- Most major OS support multi-core today
  - Windows, Linux, Mac OS X, …

General Context: Multiprocessors

- Multiprocessor is any computer with several processors

- SIMD
  - single instruction, multiple data
    - modern graphics cards

- MIMD
  - multiple instructions, multiple data

Lemieux cluster, Pittsburgh supercomputing center

Multiprocessor

- 2—4 multi-core processors on a motherboard
  - connected by special short distance interconnect
    - Intel QuickPath

Multiprocessor Memory Types

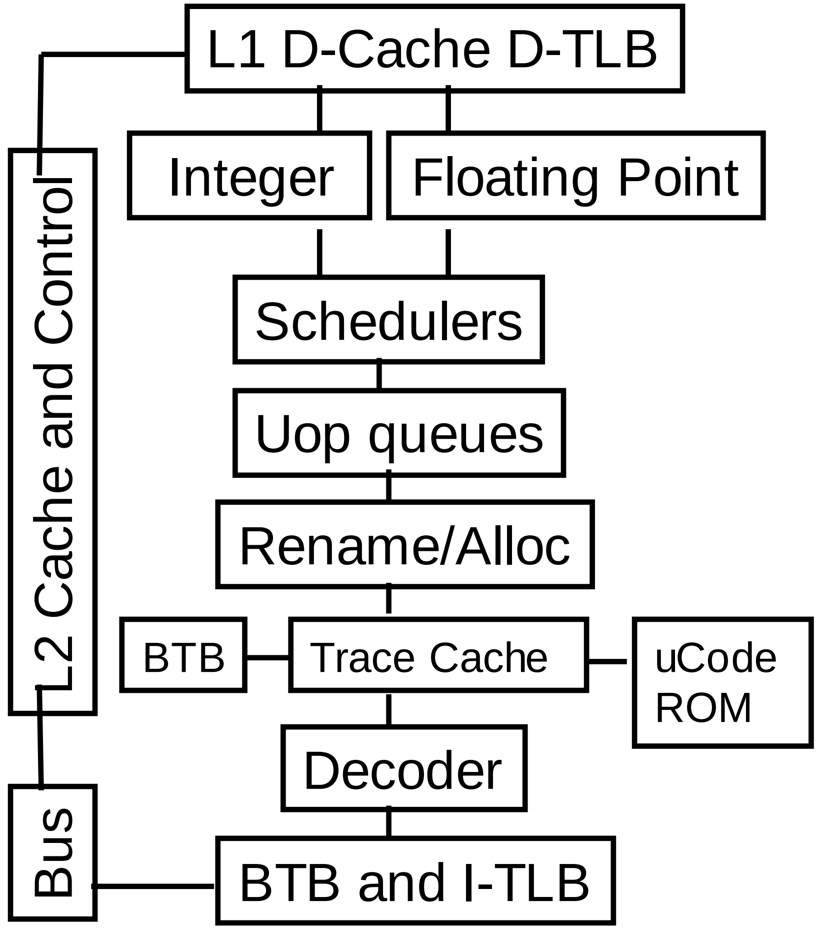
- Shared memory
  - one (large) common shared memory for all processors
- Distributed memory
  - each processor has its own (small) local memory, and its content is not replicated anywhere else

Multiprocessor Vs. Multi-core Processor

- Multi-core processor
  - a special kind of a multiprocessor with all processors on the same chip
- Multi-core processors are MIMD
  - Different cores execute different threads (multiple instructions), operating on different parts of memory (multiple data)
- Multi-core is a shared memory multiprocessor
  - All cores share the same memory

Simultaneous Multi-Threading (SMT)

- Problem addressed
  - processor pipeline can get stalled:
    - Waiting for the result of a long floating-point (or integer) operation
    - Waiting for data to arrive from memory
- Other execution units wait unused



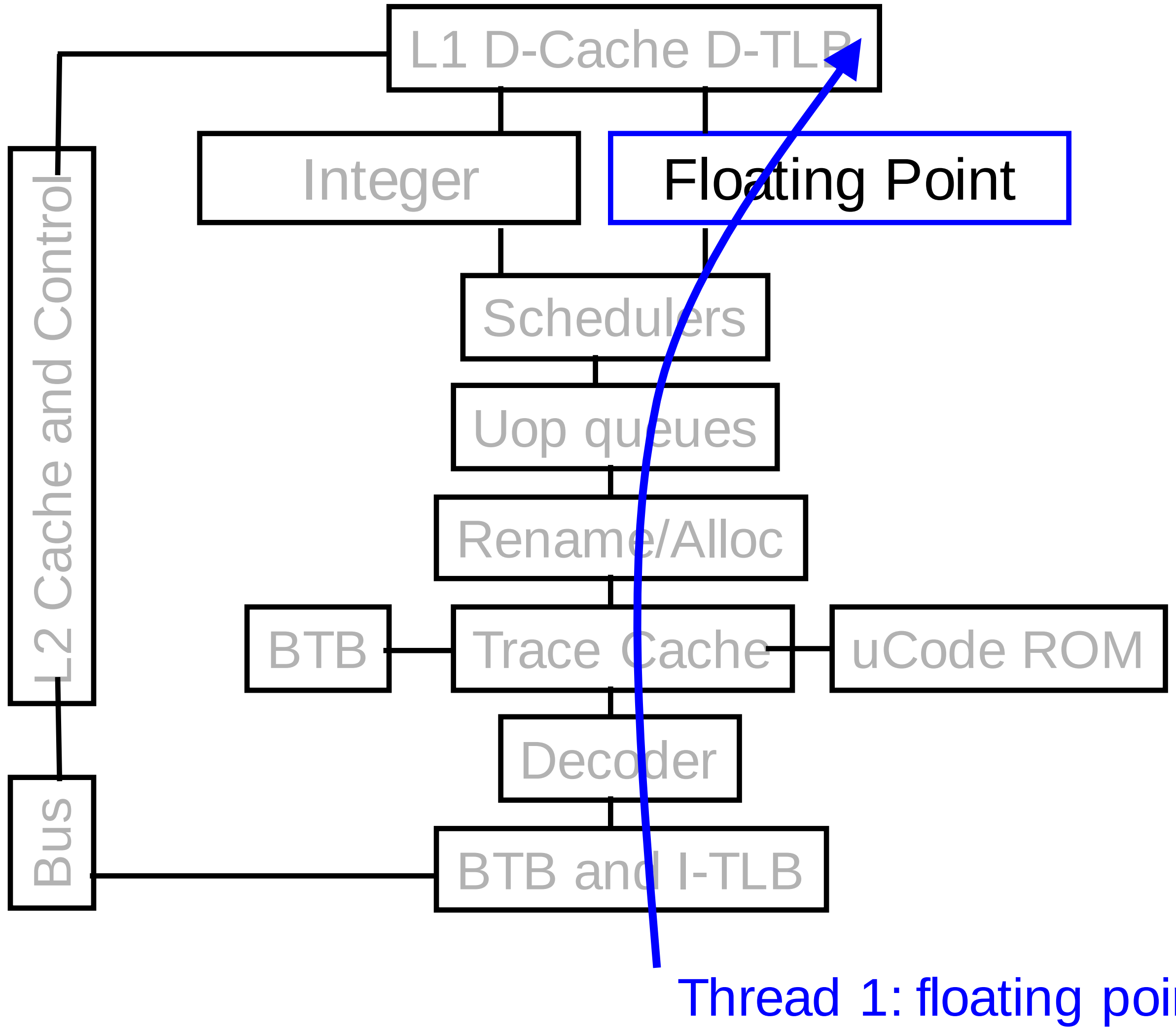
Source: Intel

Simultaneous Multi-Threading (SMT)

- Permits multiple independent threads to execute SIMULTANEOUSLY on the SAME core
- Weaving together multiple “threads” on the same core
- Example:
  - if one thread is waiting for a floating point operation to complete, another thread can use the integer units
- Another name: **hyper-threading**

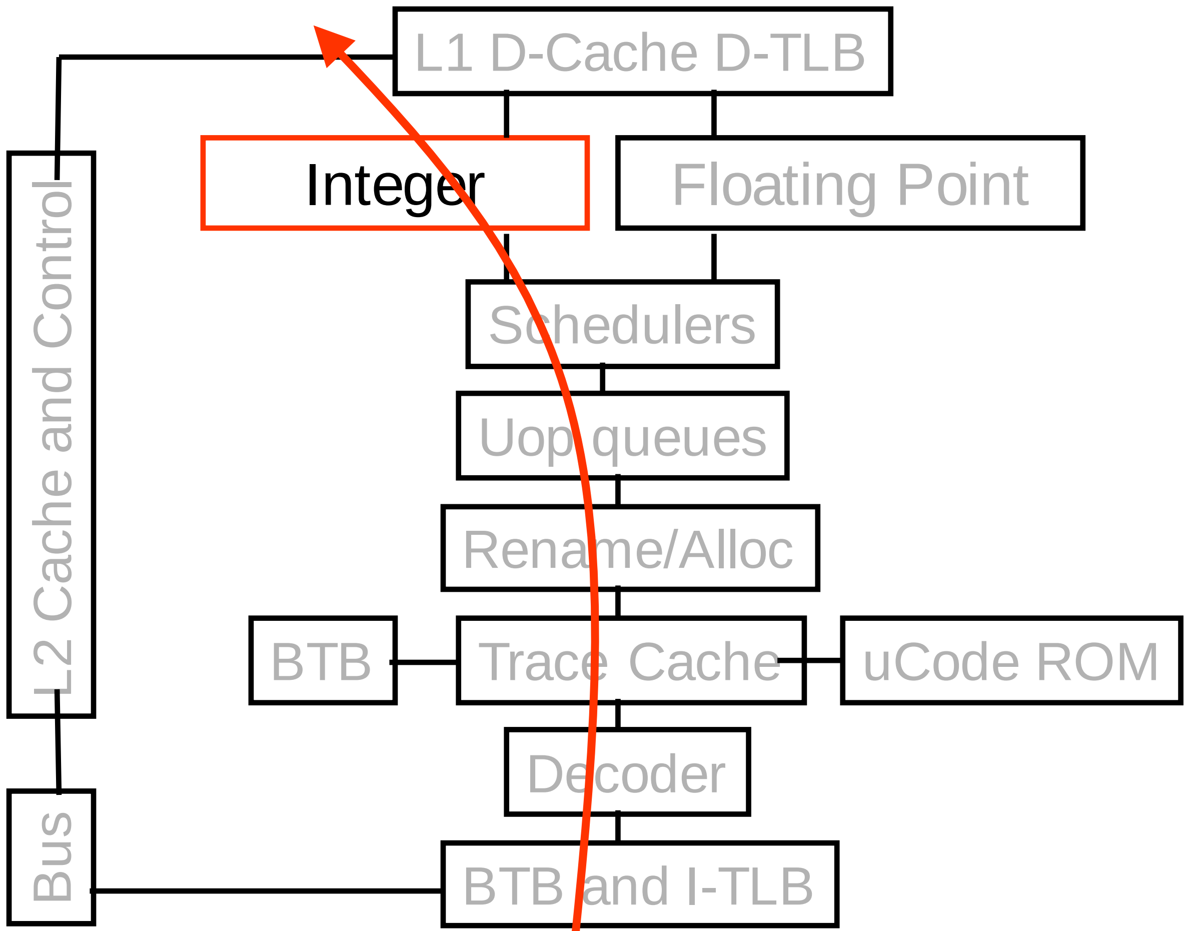
Simultaneous Multi-Threading (SMT)

- w/o SMT, only a single thread can run at any given time



Simultaneous Multi-Threading (SMT)

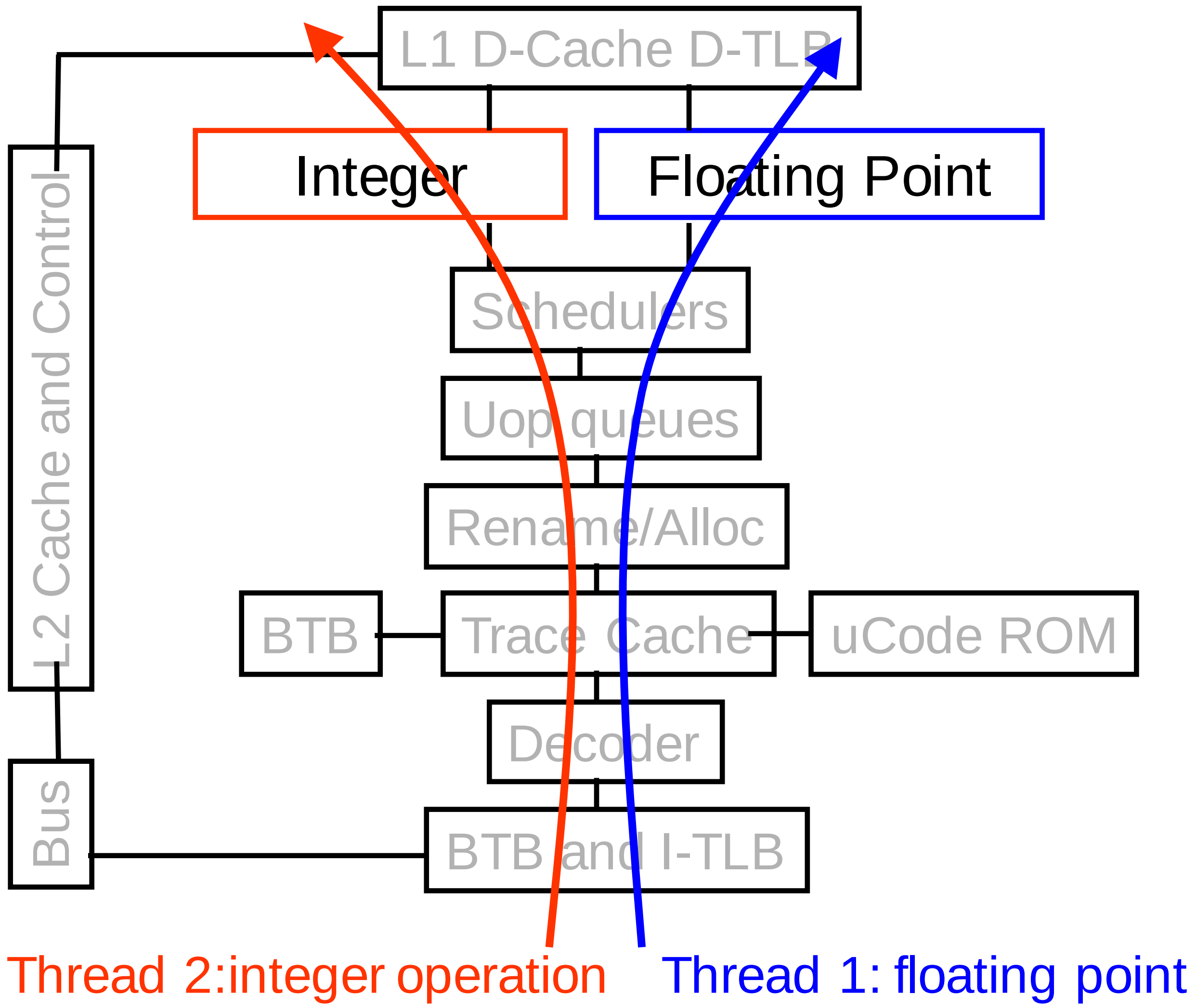
- w/o SMT, only a single thread can run at any given time



Thread 2:integer operation

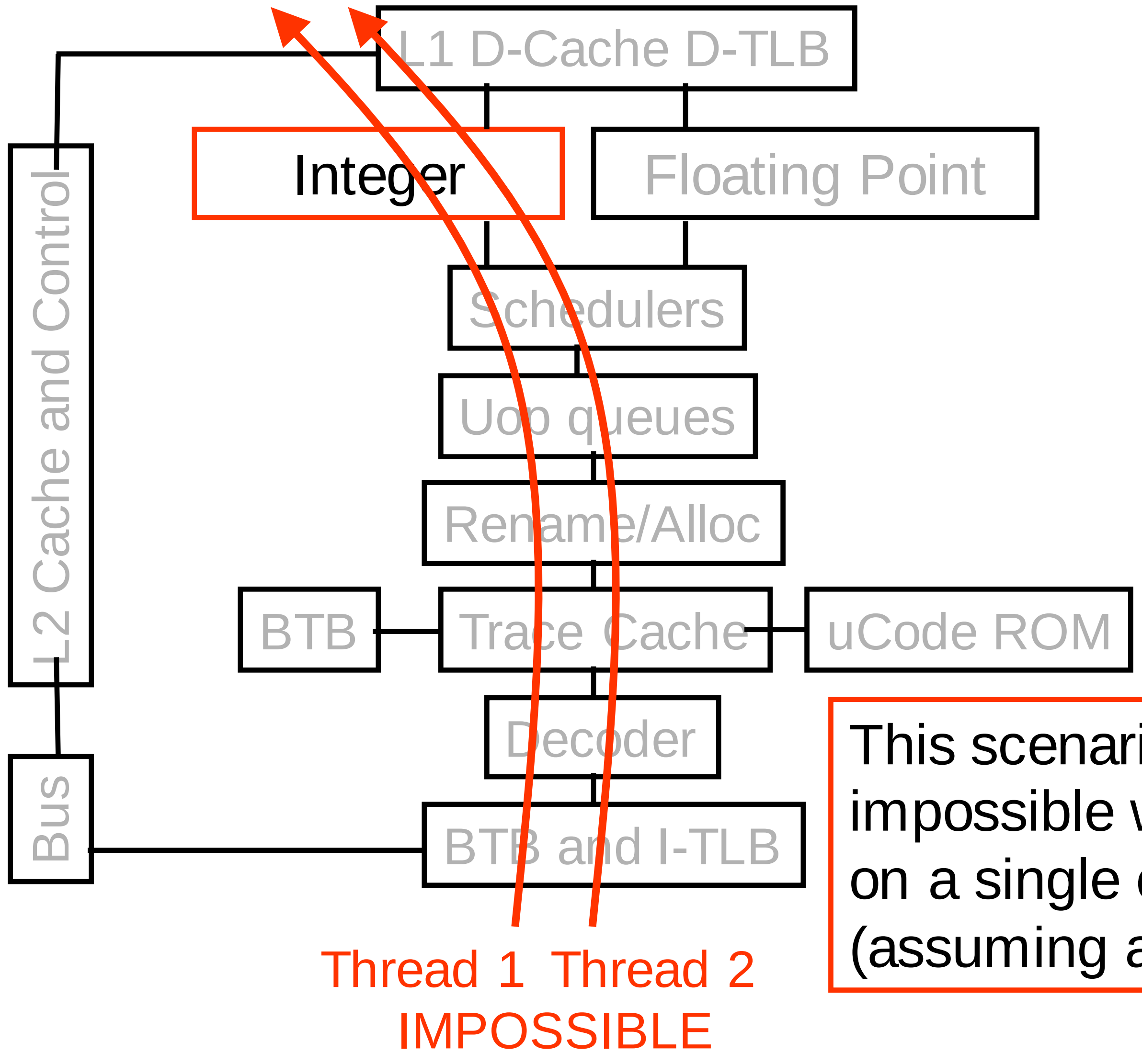
Simultaneous Multi-Threading (SMT)

- SMT processor: both threads can run concurrently



Simultaneous Multi-Threading (SMT)

- But SMT can't simultaneously use the same functional unit



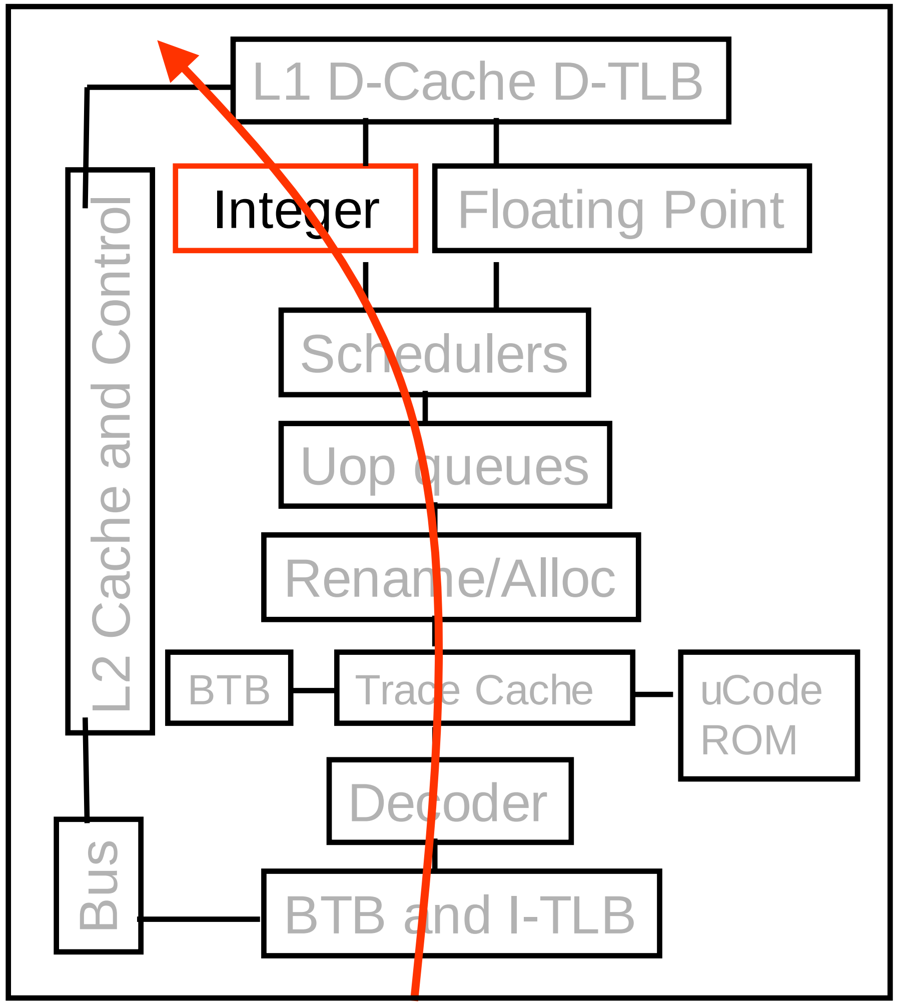
This scenario is impossible with SMT on a single core (assuming a single integer unit)

Simultaneous Multi-Threading (SMT)

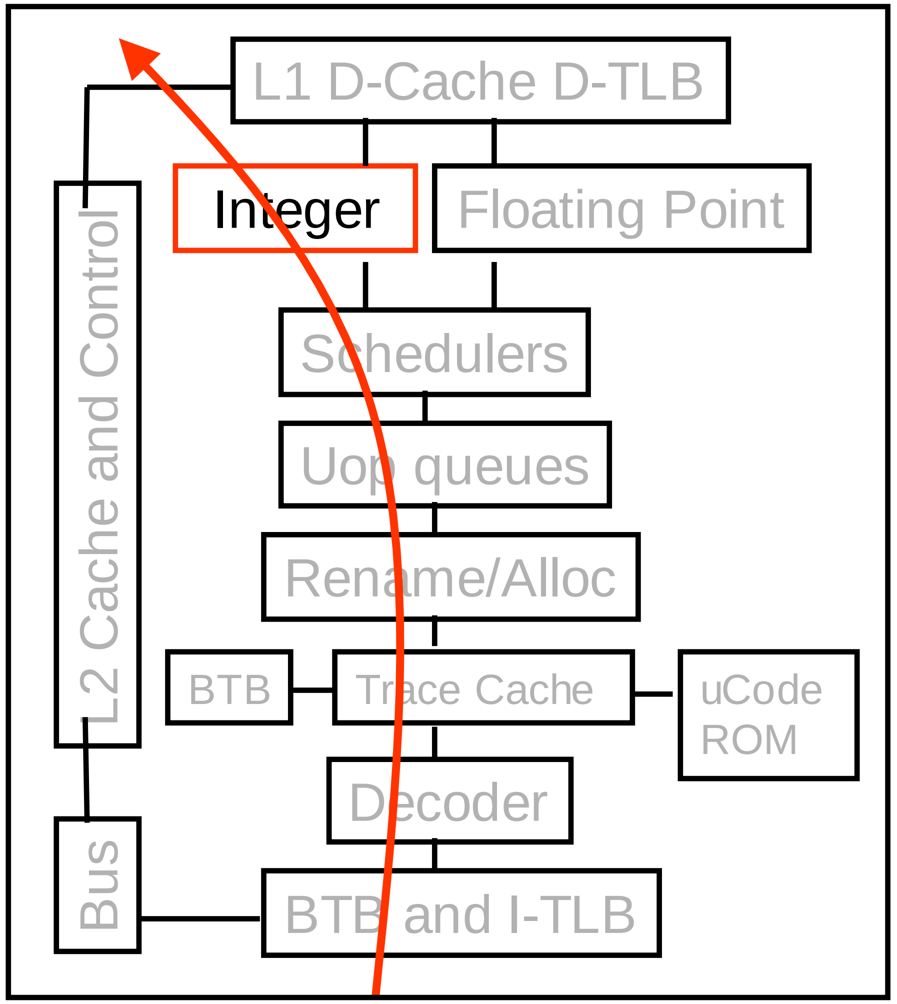
- SMT not a “true” parallel processor
  - Enables better threading (e.g., up to 30%)
  - OS and applications perceive each simultaneous thread as a separate “virtual processor”
  - The chip has only a single copy of each resource
  - Compared to multi-core, each core has its own copy of resources

Multi-core

- Threads can run on separate cores



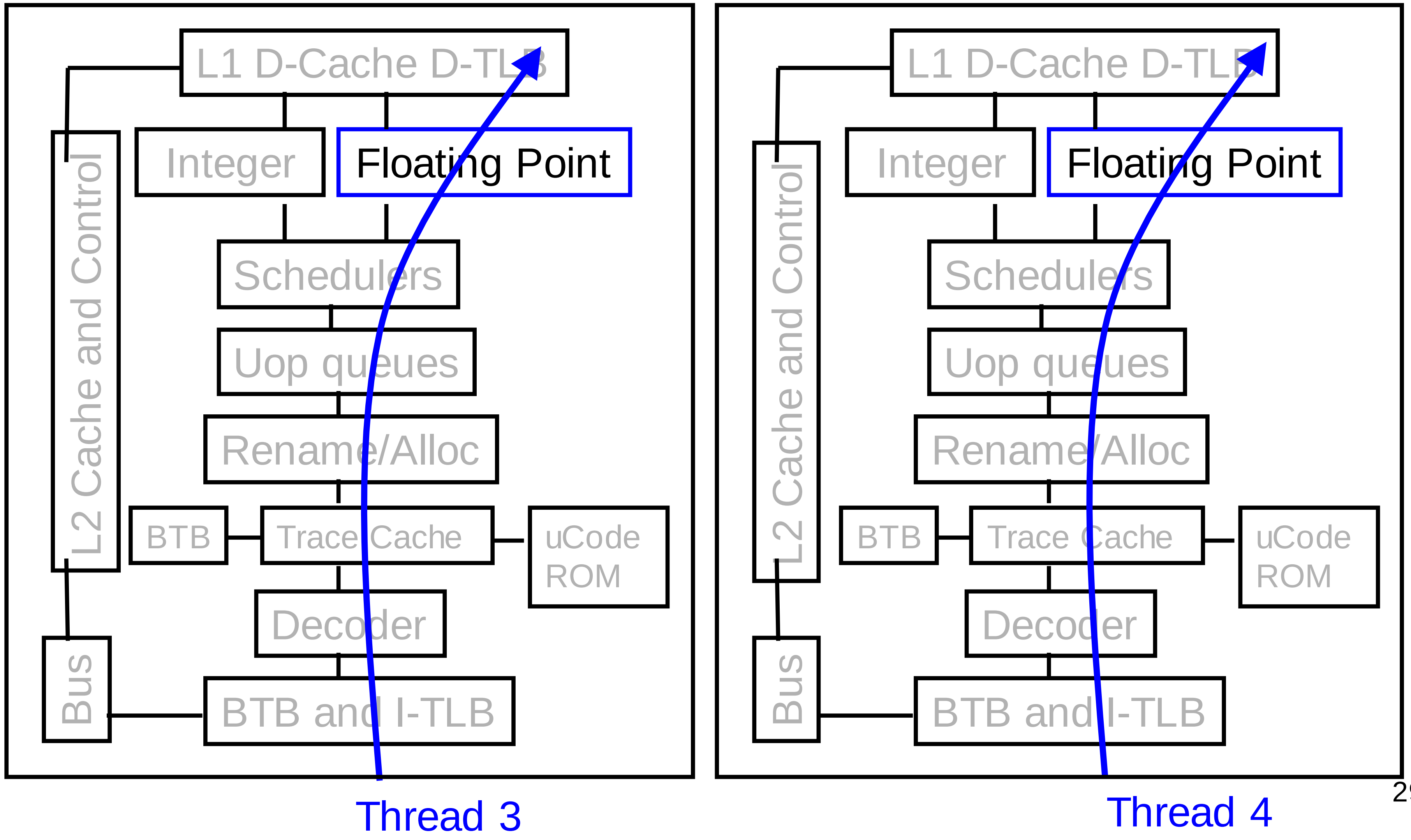
Thread 1



Thread 2

Multi-core

- Threads can run on separate cores

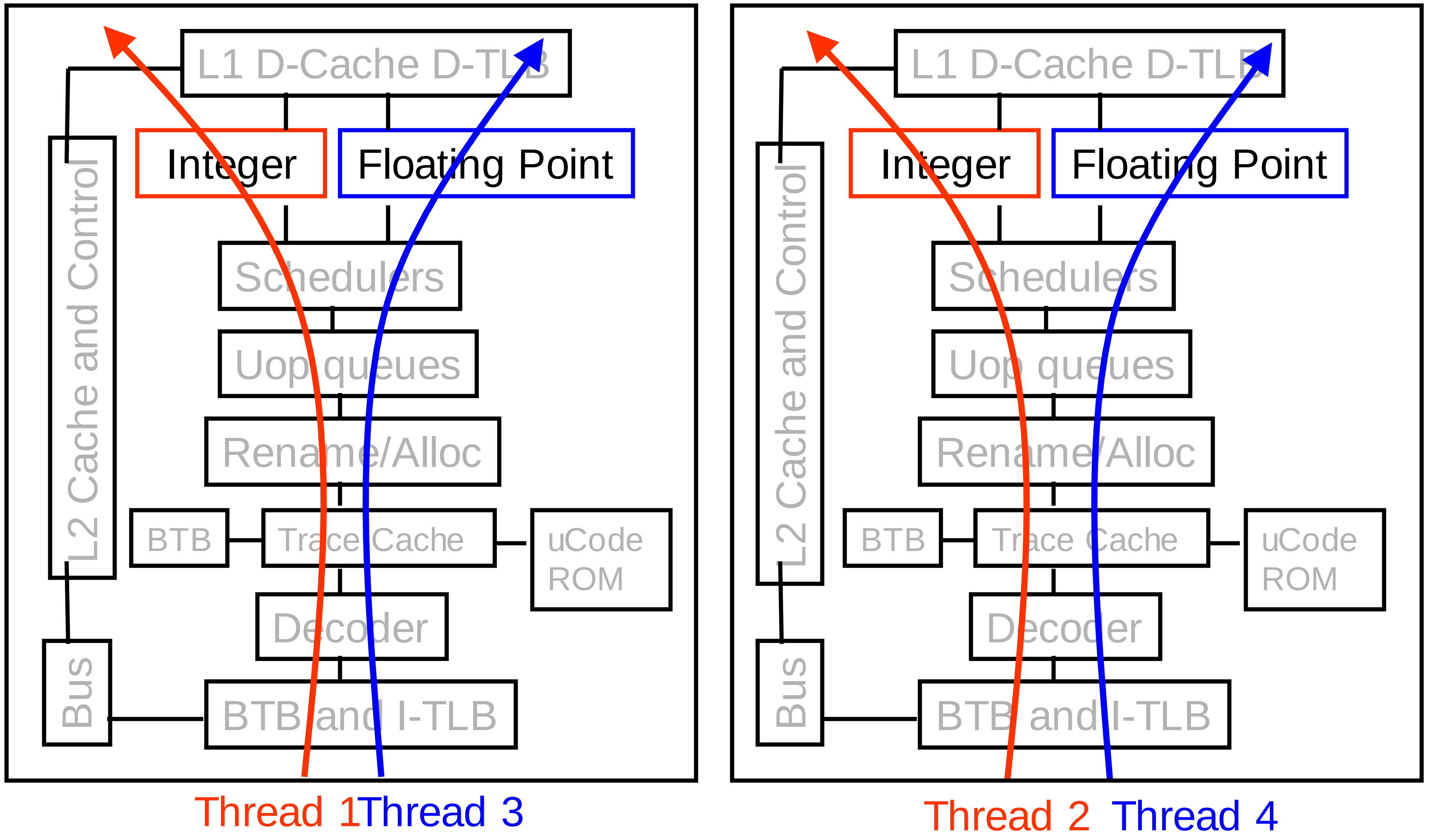


Combining Multi-core and SMT

- Cores can be SMT-enabled (or not)
- different combinations:
  - single-core, non-SMT: standard uniprocessor
  - single-core, w/ SMT
  - multi-core, non-SMT
  - multi-core, w/ SMT
- Number of SMT threads
  - 2, 4, or sometimes 8 simultaneous threads
- Intel calls them “hyper-threads”

SMT Dual-core

- All four threads can run concurrently



Comparison: Multi-core vs. SMT

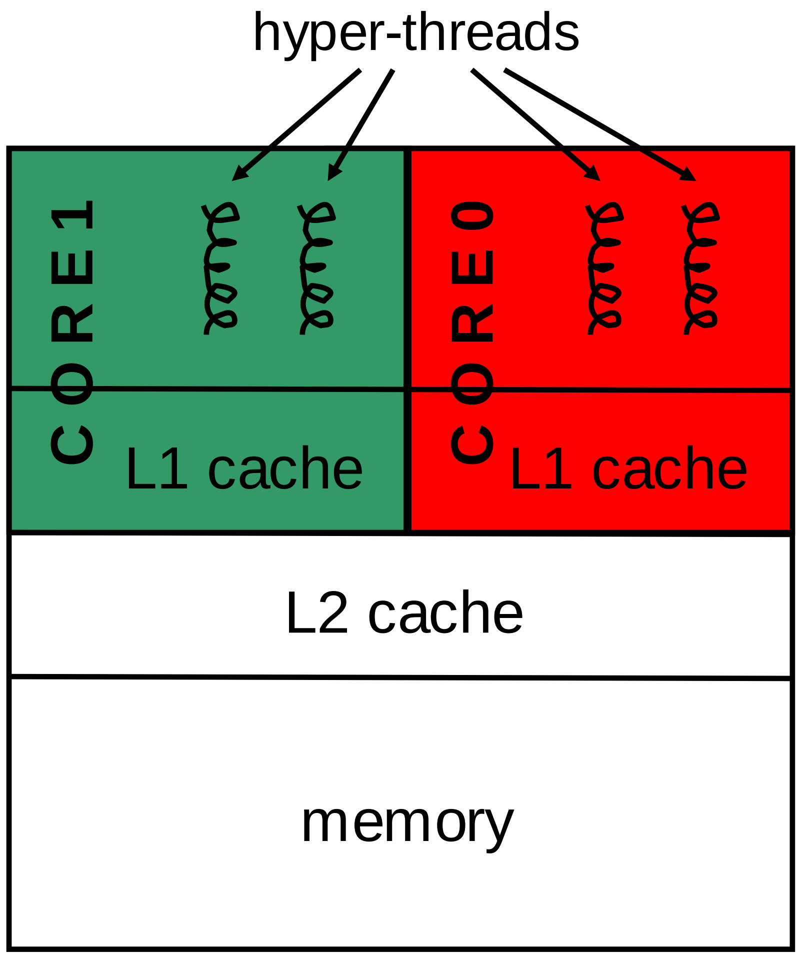
- Multi-core:
  - Since there are several cores, each is smaller and not as powerful (but also easier to design and manufacture)
  - However, great with thread-level parallelism
- SMT
  - Can have one large and fast superscalar core
  - Great performance on a single thread
  - Mostly still only exploits instruction-level parallelism

Memory Hierarchy

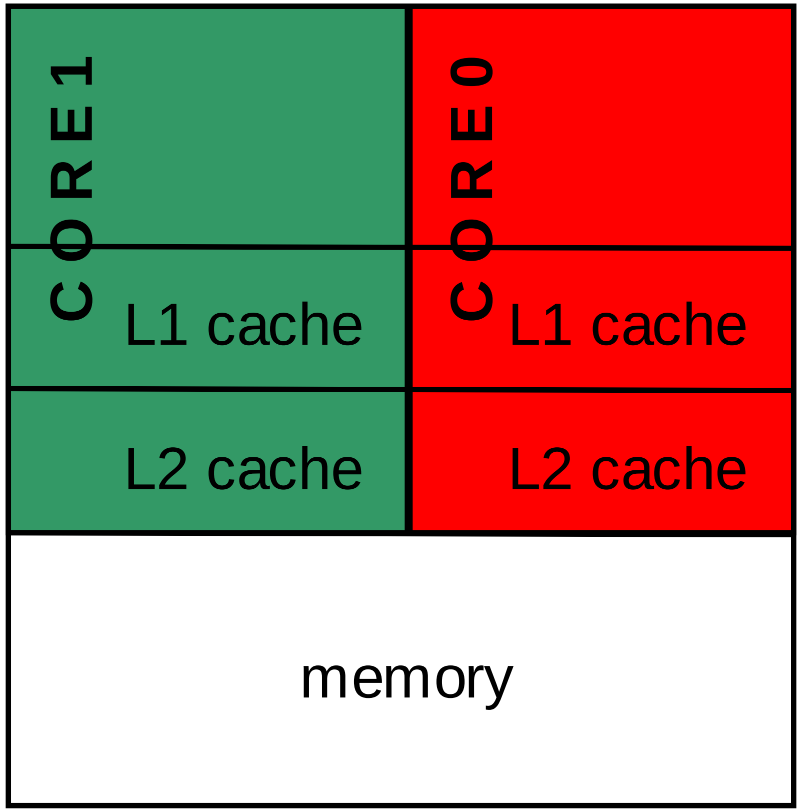
- If simultaneous multithreading only:
  - all caches shared
- Multi-core chips:
  - L1 caches private
  - L2 caches private in some architectures and shared in others
- Memory is always shared

Dual-core Intel Xeon Processors

- Each core
  - hyper-threaded
- Private L1 caches
- Shared L2 caches

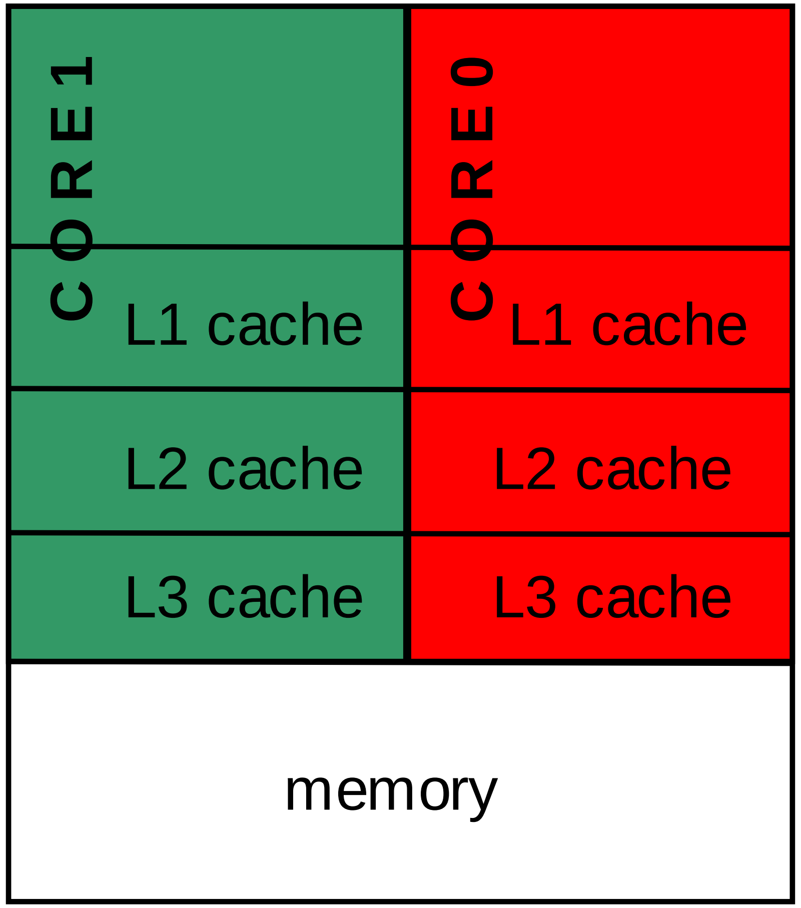


Designs w/ private L2 caches



both l1 and l2 are private

examples: amd opteron,
amd athlon, intel pentium d



a design with l3 caches

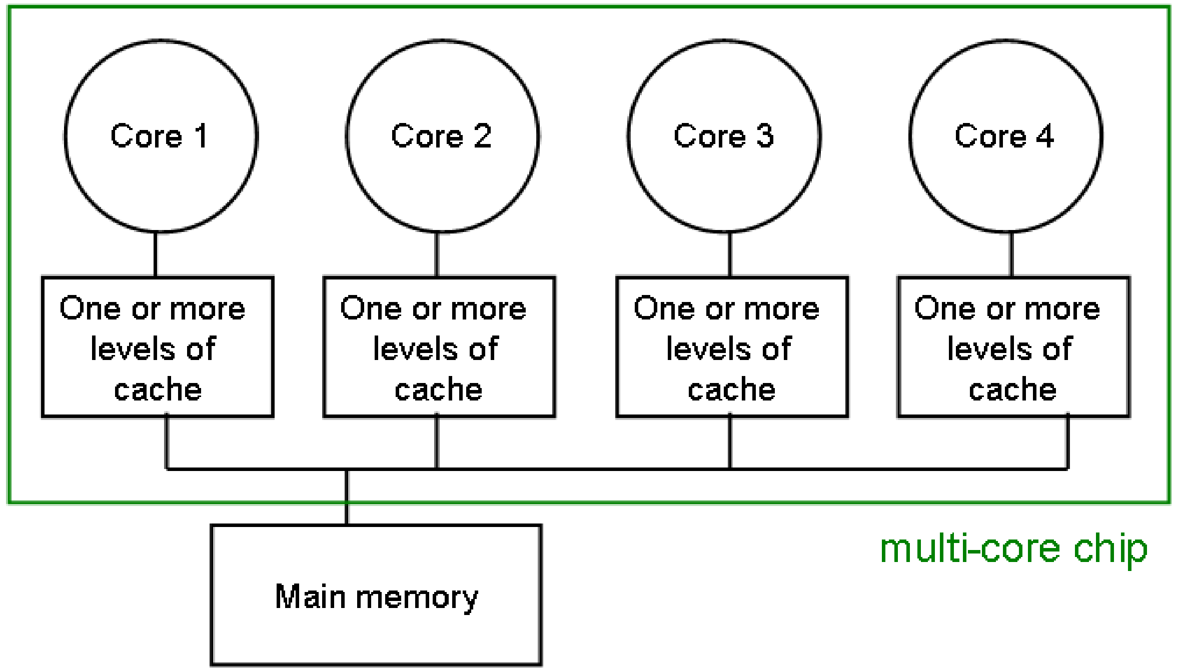
example: intel itanium 2

Private vs shared caches

- Advantages of private:
  - They are closer to core, so faster access
  - Reduces contention
- Advantages of shared:
  - Threads on different cores can share the same cache data
  - More cache space available if a single (or a few) high-performance thread runs on the system

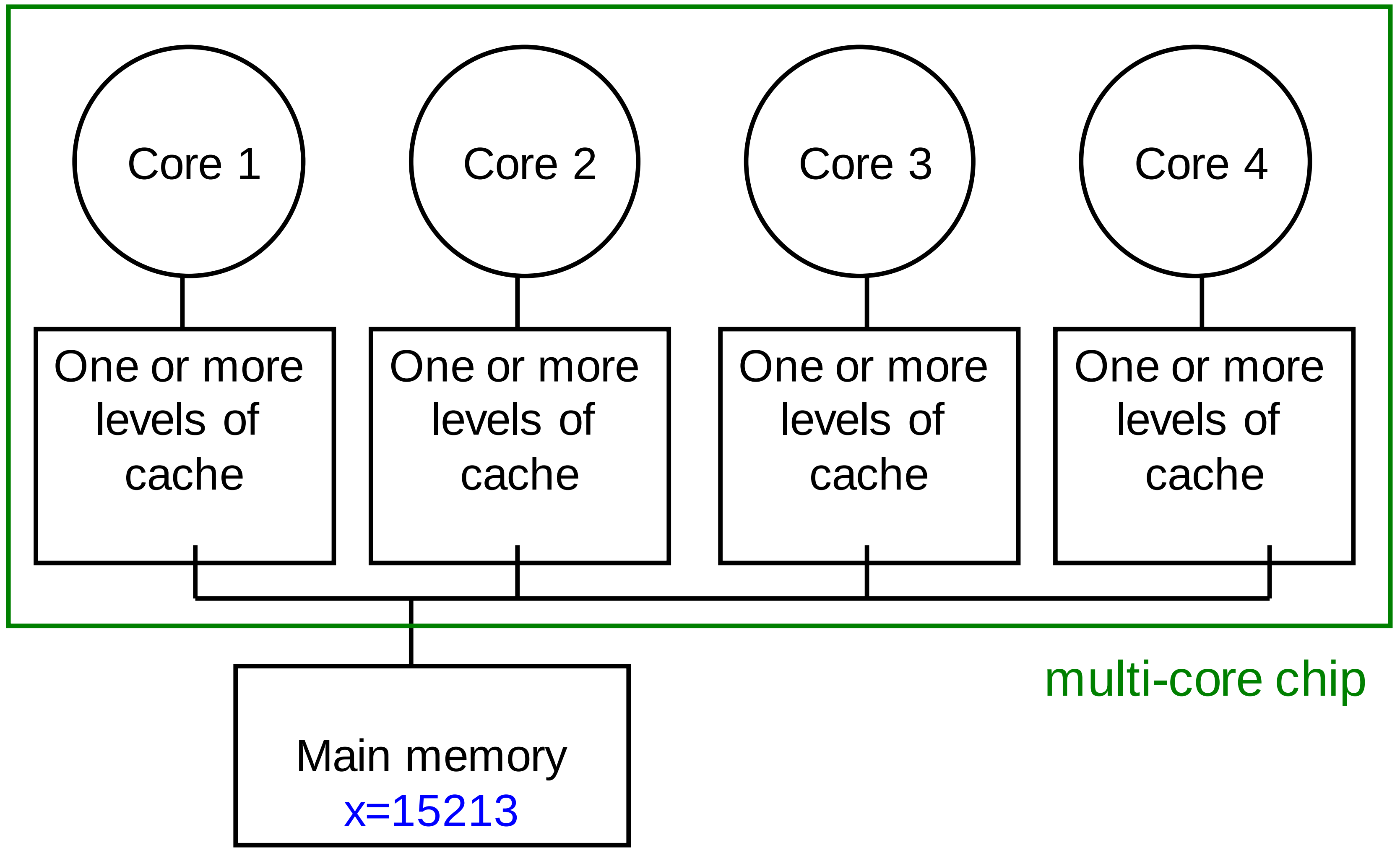
Cache Coherence Problem

- Since we have private caches, how to keep the data consistent across caches?
- Each core should perceive the memory as a monolithic array, shared by all the cores


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
multi-core chip

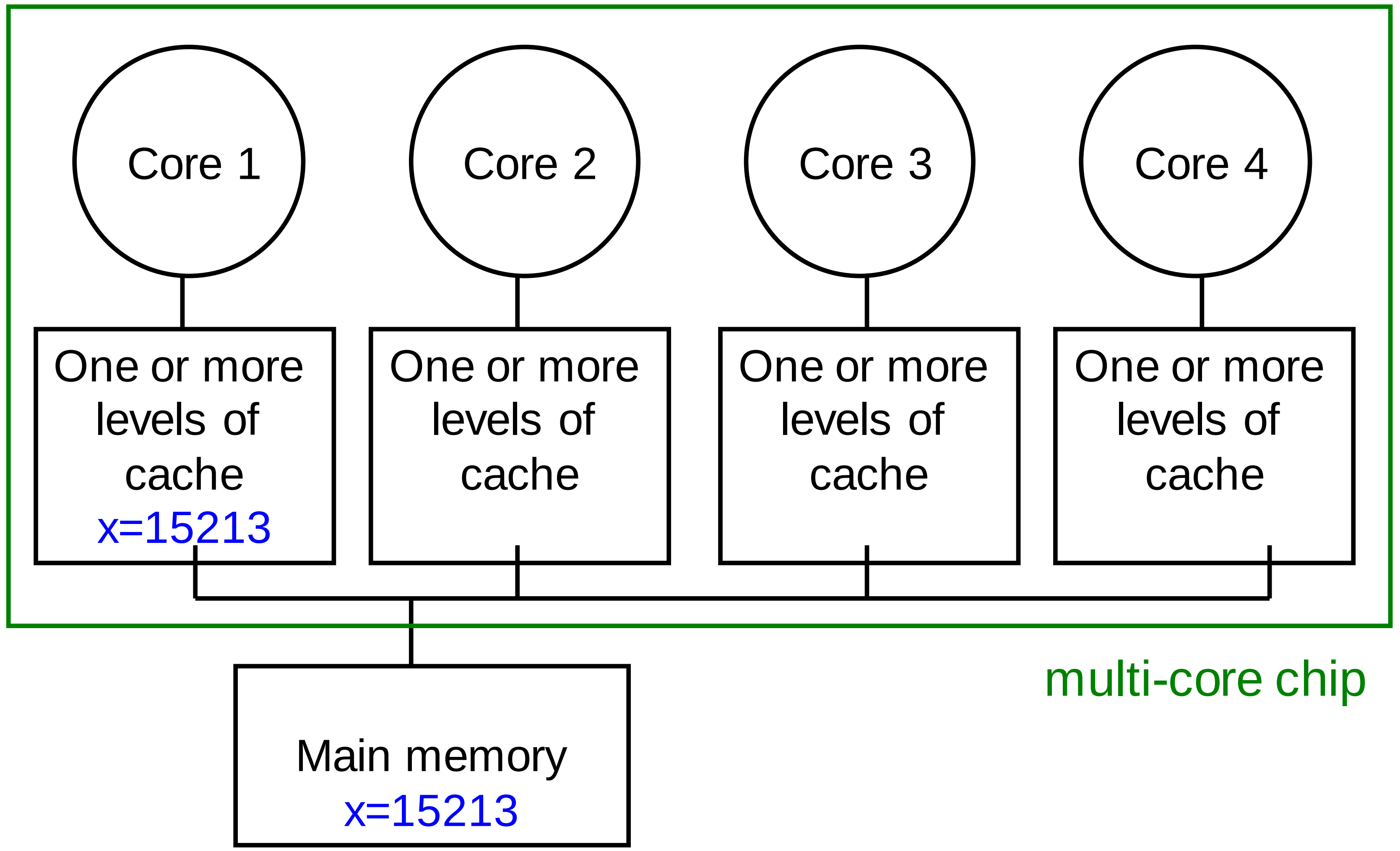
Cache Coherence Problem

- Suppose variable x initially contains 15213


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
multi-core chip
Main memory
x=15213

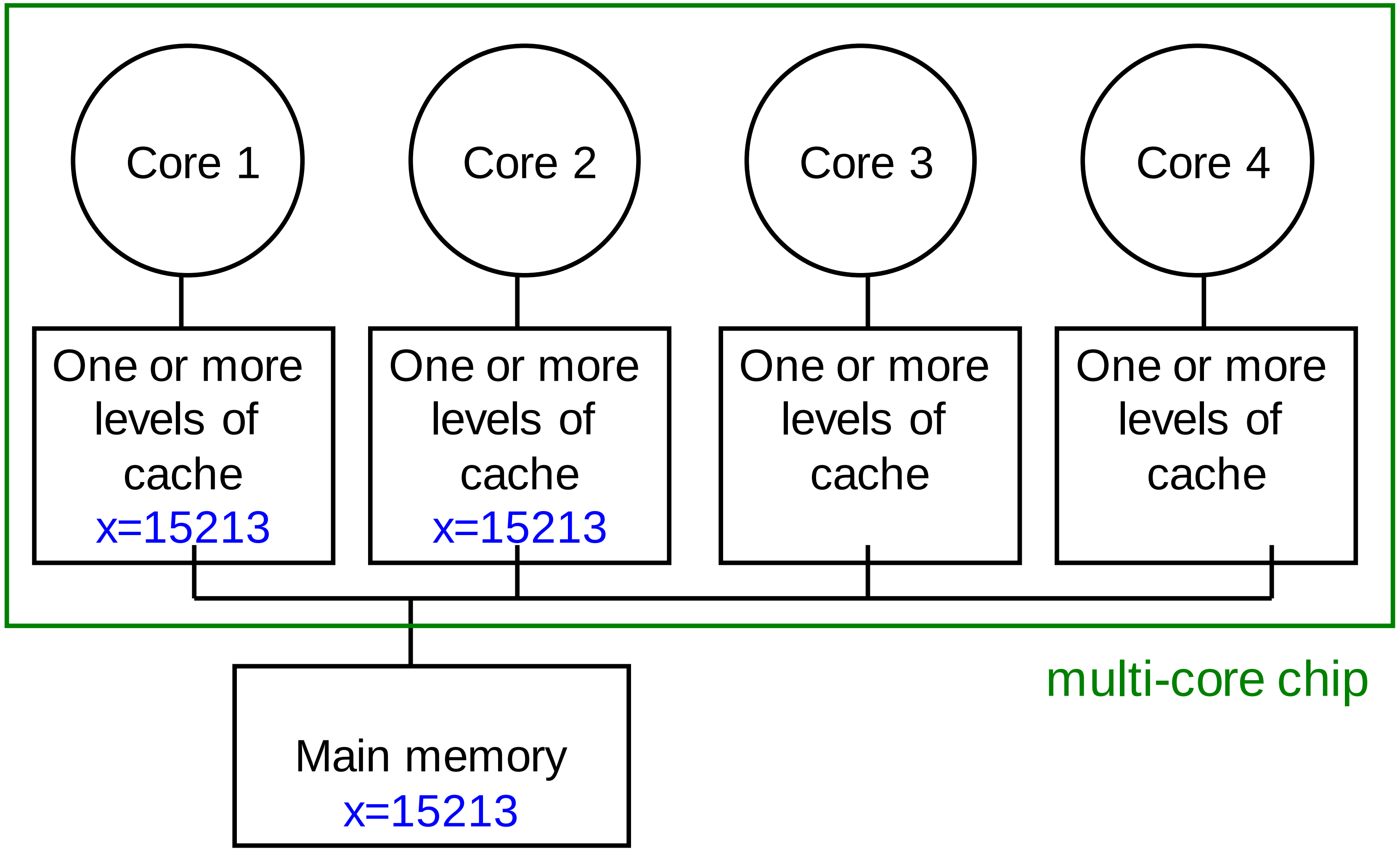
Cache Coherence Problem

- core 1 reads x


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
x=15213
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
x=15213
multi-core chip

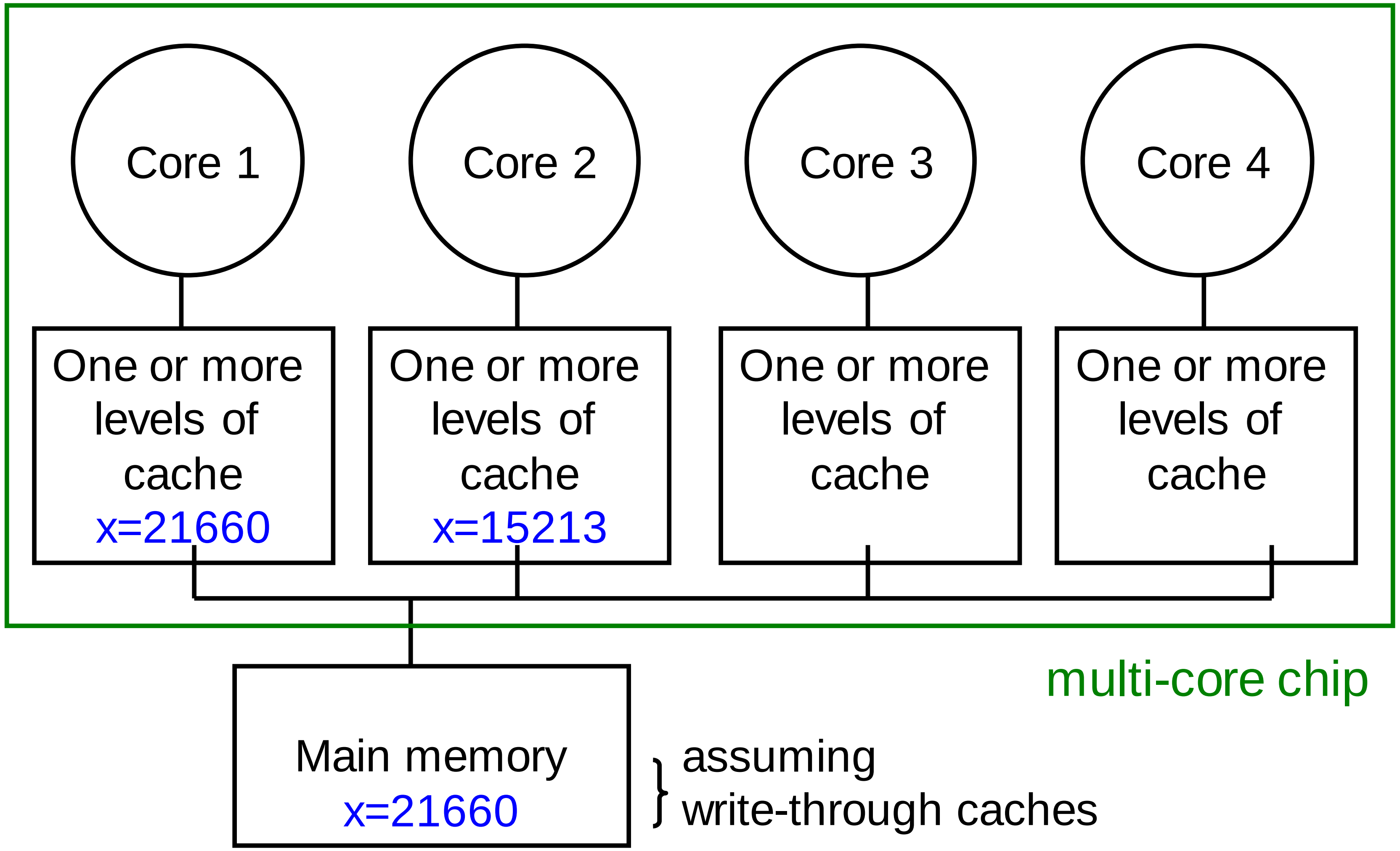
Cache Coherence Problem

- core 2 reads x


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
x=15213
One or more levels of cache
x=15213
One or more levels of cache
One or more levels of cache
Main memory
x=15213
multi-core chip

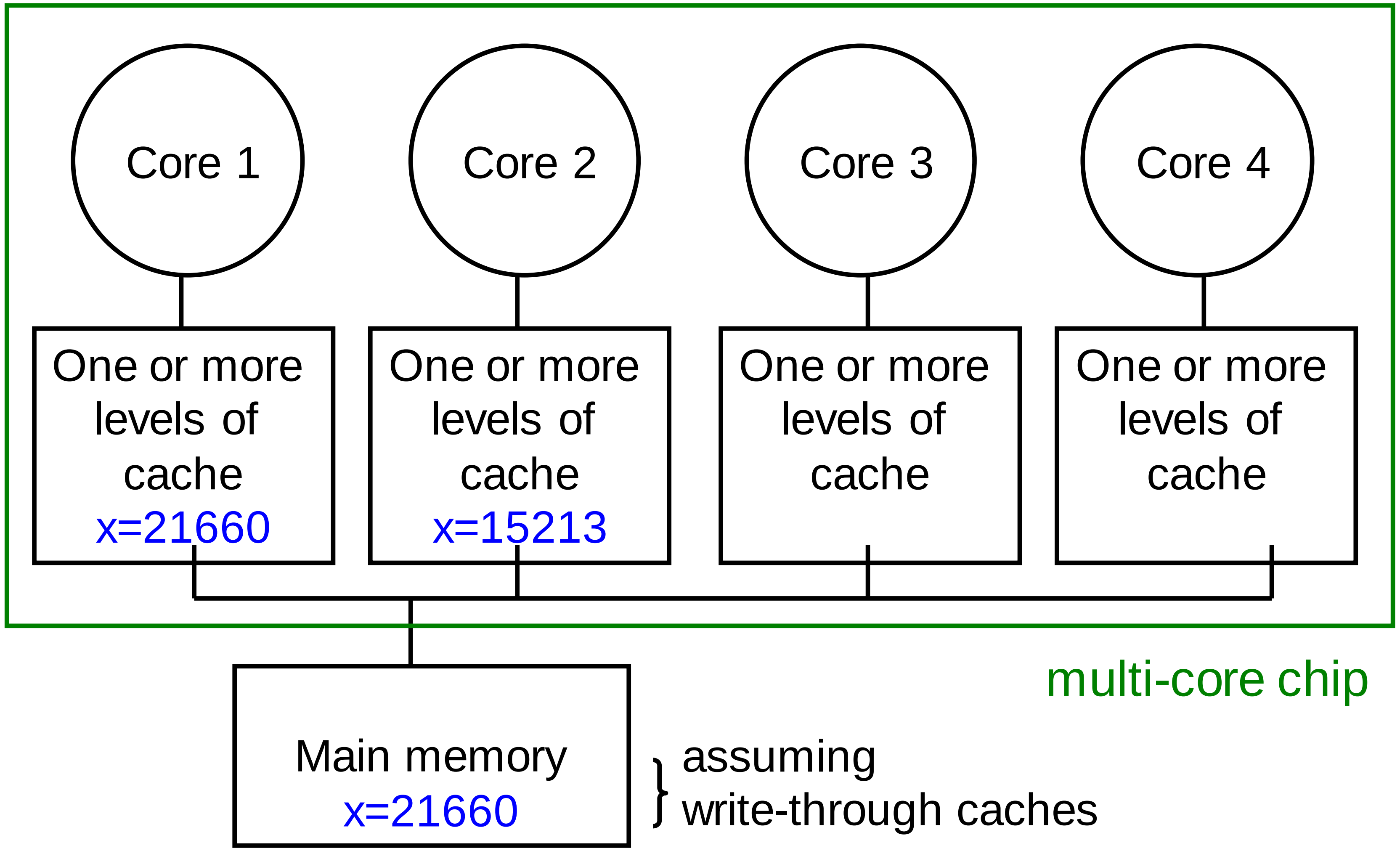
Cache Coherence Problem

- core 1 writes to x, setting it to 21660


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
x=21660
One or more levels of cache
x=15213
One or more levels of cache
One or more levels of cache
multi-core chip
Main memory
x=21660
assuming write-through caches

Cache Coherence Problem

- core 2 attempts to read x… gets a stale copy


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
x=21660
One or more levels of cache
x=15213
One or more levels of cache
One or more levels of cache
multi-core chip
Main memory
x=21660

} assuming write-through caches

Solutions for Cache Coherence Problem

- General problem w/ multiprocessors, not limited just to multi-core
- Many solution algorithms, coherence protocols, etc.
  - A simple solution: invalidation-based protocol with snooping
  - MSI, MESI (Modified, Exclusive, Shared, Invalid)

Announcement

- Next Lecture: cache coherence