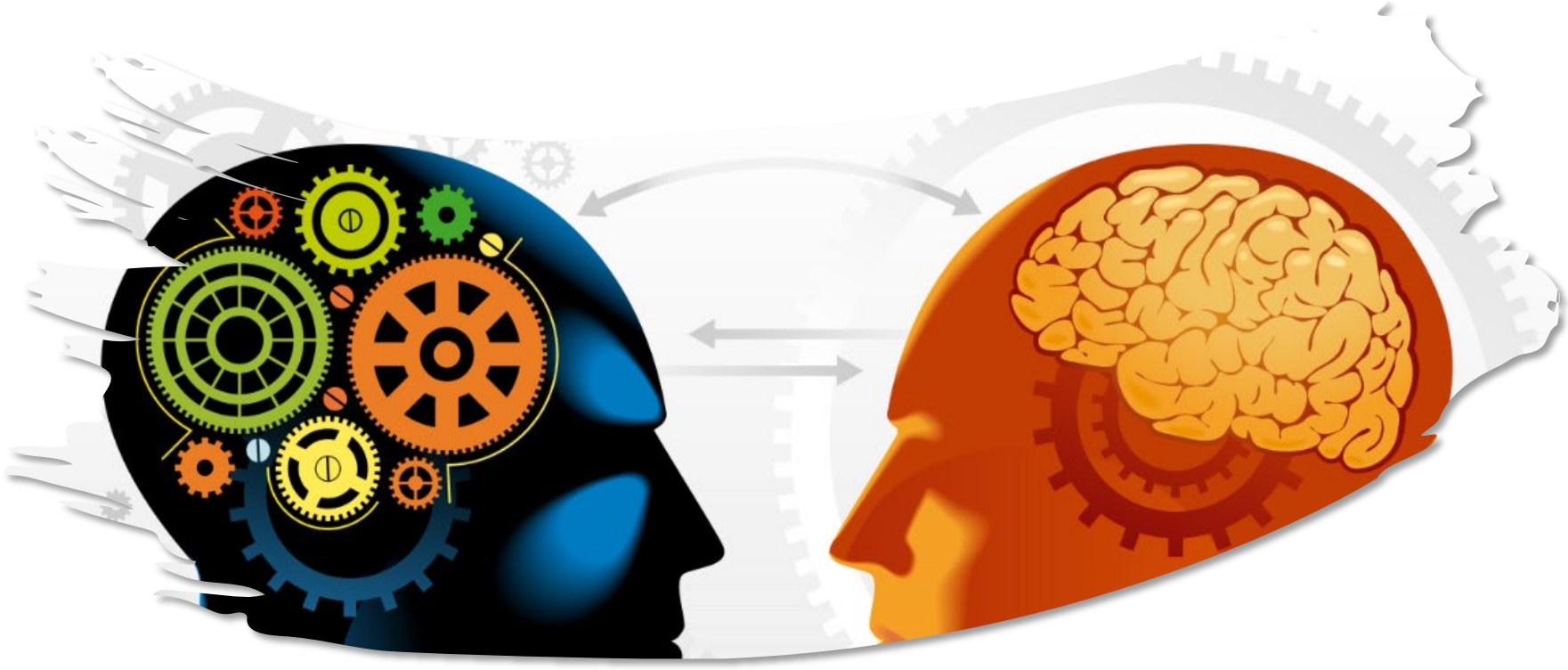


Building the Infinite Brain

COMP 590/790

Raghavendra Pradyumna Pothukuchi

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

✉ raghav@cs.unc.edu

Takeaways

What are the ways to improve throughput beyond single instruction stream?

Multithreading, chip multiprocessing, and combinations

What is the difference between multithreading and chip multiprocessing?

Multithreading: multiple instructions on the same core/processor/CPU
Chip multiprocessing: using multiple cores on the same chip

What are the aspects of multicore organization?

Compute (symmetric or asymmetric) and memory (centralized or distributed)

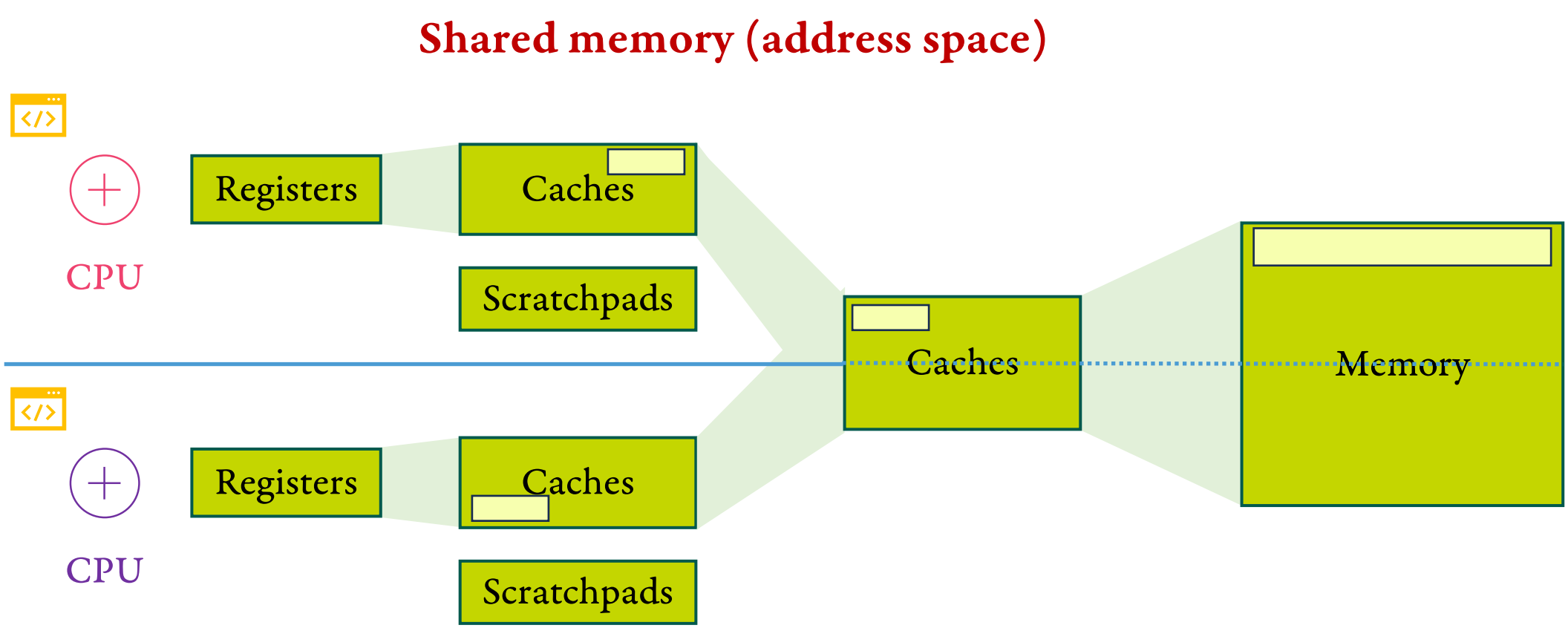
How are hardware and software parallelism related?

Abstractions are different—software (threads, processes), hardware (threads, contexts)
Hardware: implicit (single instruction stream) or explicit (multiple instruction streams)
Software: sequential (via automatic parallelism) or parallel (shared memory or message passing)

For Today

- Quick review
- **Memories in multicore and distributed processors**

Utilizing Multicore Parallelism



How to coordinate memory accesses?

Coordinating Memory Accesses

Define correct behavior!

Correct or expected?

Shared data block

Cache coherence

Also called cache consistency

Can this be "fixed" in software?

"Write the spec"

All accesses

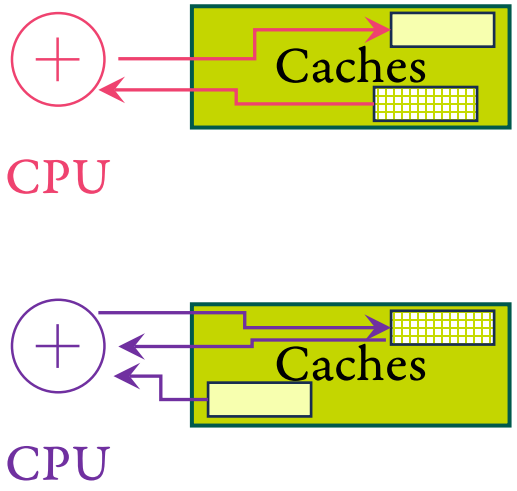
Memory consistency

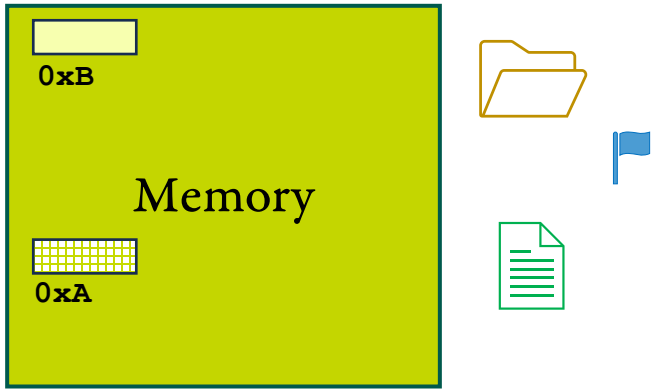
Why not all operations?

Applies to sequential programs too!

```
1
Read 0xA
Write 1 to 0xB
```

```
1
Read 0xB
Write 1 to 0xA
Read 0xA
0
```





Cache Coherence

Reads and writes to a single shared block

Various definitions

Each write is *eventually visible* and writes to the same location are *serialized*

Reads after a write from a core return the written value, reads after a write from another core read the written value after some time, and writes to the same location from multiple cores appear in the same order for all cores

"Enforcing the spec"

Cache coherence protocols

Cache Coherence Protocols

Mode of communication

Snoop: All cores listen to all accesses
Directory: Arbiter handles communication

> Snooping appears decentralized, but needs a broadcast channel
> Directory appears centralized, but can be distributed

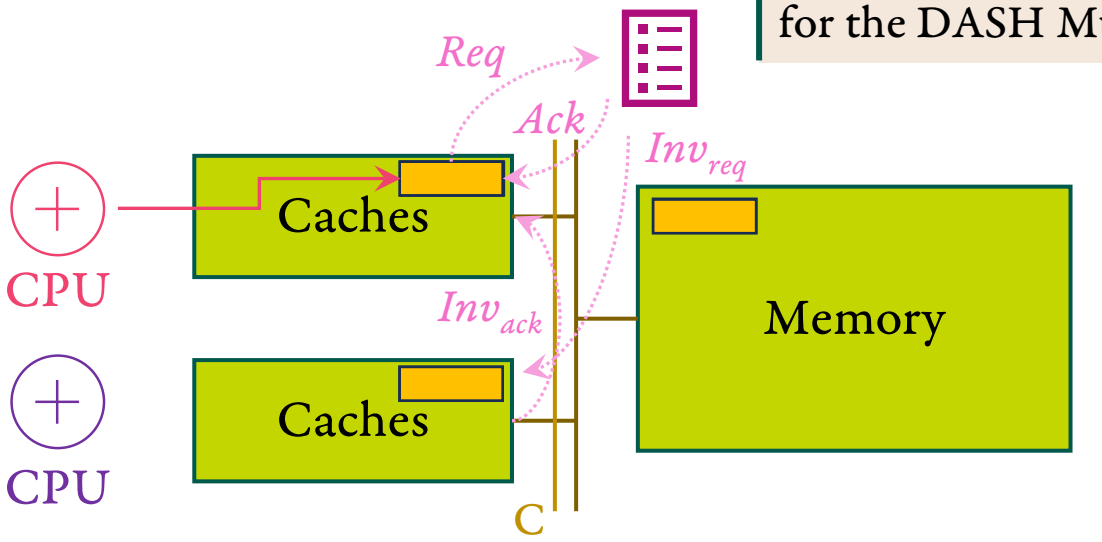
Communicating writes

Update: All cores receive new value
Invalidate: All other cores must re-read data

Write sharing

> "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", ISCA 1990

Write through/Write back

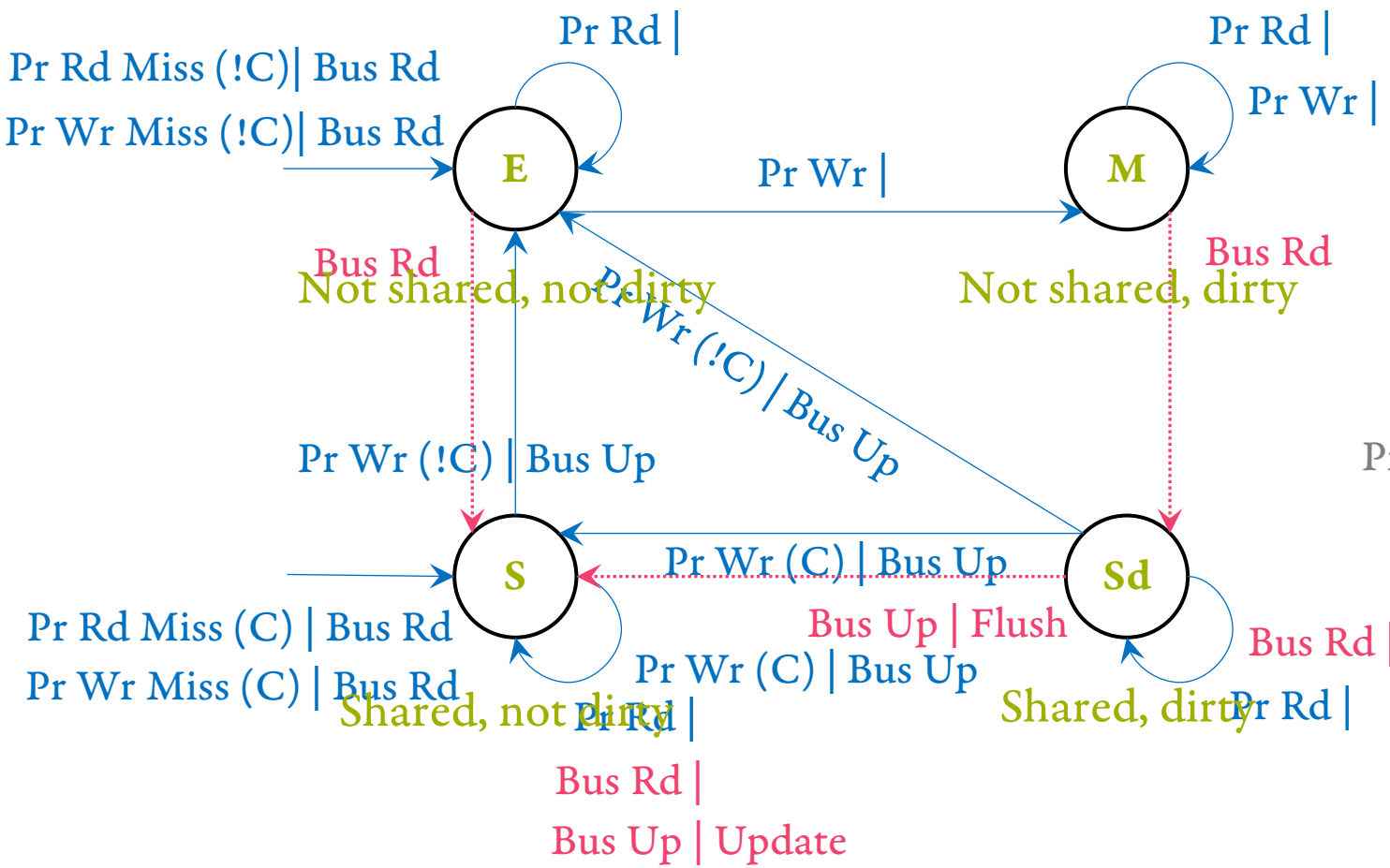


Snoop Update

"Firefly: A Multiprocessor Workstation"
Charles P. Thacker, Lawrence C. Stewart, and Edwin H. Satterthwaite Jr., ASPLOS'87

When is this good?

Interleaved reads and writes from multiple cores



Track Shared and Dirty
Dirty: Memory stale

Not shared: Write-back
Shared: write-through

Write miss: Read miss, Write hit

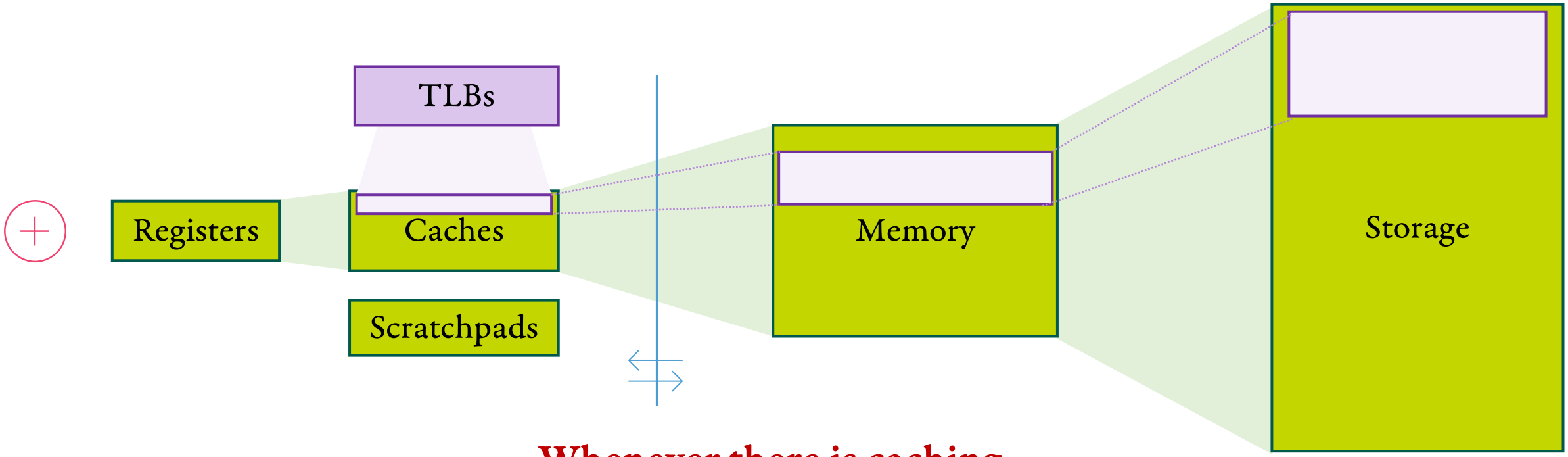
C: Copies
!C: No copies

Processor transactions (Pr) : Rd, Wr
Bus transactions: Rd, Up

Event | Output

Where Can Coherence Become Desirable?

> "TLB consistency on highly-parallel shared-memory multiprocessors"
> Patricia J Teller, Richard Kenner, Marc Snir , 1988



Whenever there is caching

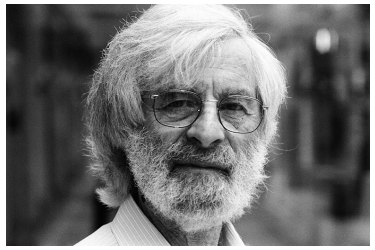
Is coherence required?

"Abstraction" of a single memory without caches

"Yummy" for systems (OS and compiler) developers

Memory Consistency

Ordering of reads and writes



> "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Program", 1979

Leslie Lamport, *Pioneer in formal design of distributed systems*

Sequential consistency

"the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program."

Other notions of accepted behavior

Relax ordering: Write→Read, Write→Write, Read→Write, Read→Write, Read own write early, Read other writes early

Notion of a data race: two simultaneous accesses to the same block, and at least one is a write

Synchronization to enforce ordering

Why? "Efficiency": overlapping memory latency

Specifying Consistency

Software visible?

```
a = 0                    a = 0
for i in …
    a = 1
    print(a)
```

Need a memory consistency model anywhere there is a reordering of memory operations

The model might simply say that any reordering is possible unless there is synchronization!

Interconnect: Bus, network on chip (NoC), interconnect?

"Spec" "Efficient and Correct Execution of Parallel Programs ", Dennis Shaha and Marc Snir, 1988

Enforcing the "spec"

Formal analysis of "happens before" and dependency relations

Where else are consistency models needed?

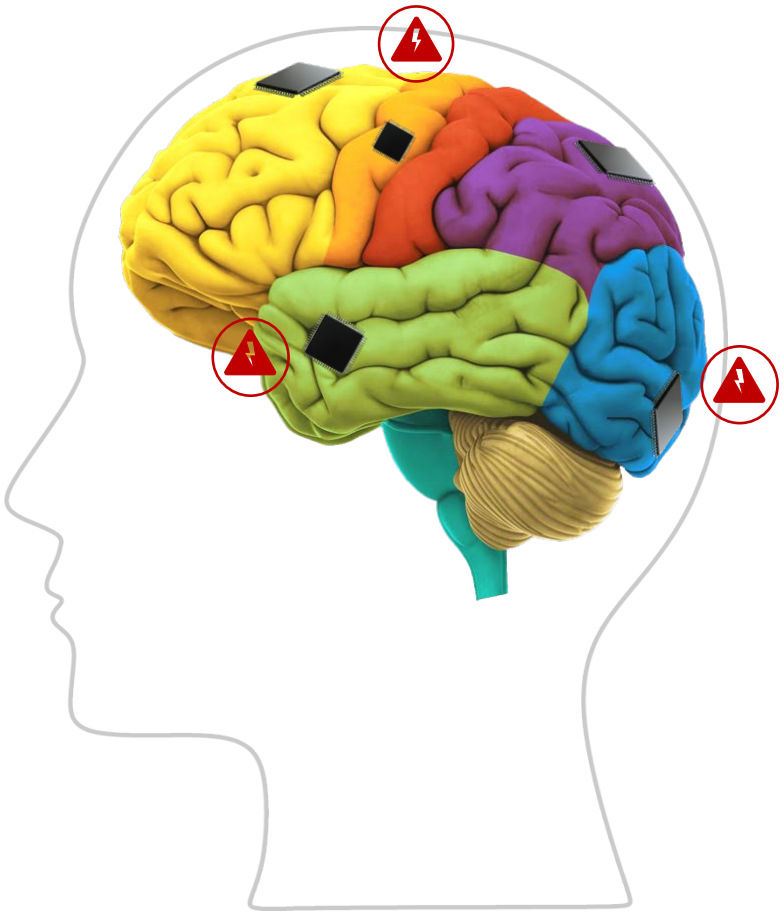
Databases, storage,…wherever there is a data race and reasoning about ordering is needed

Example from a BCI Processor (SCALO)

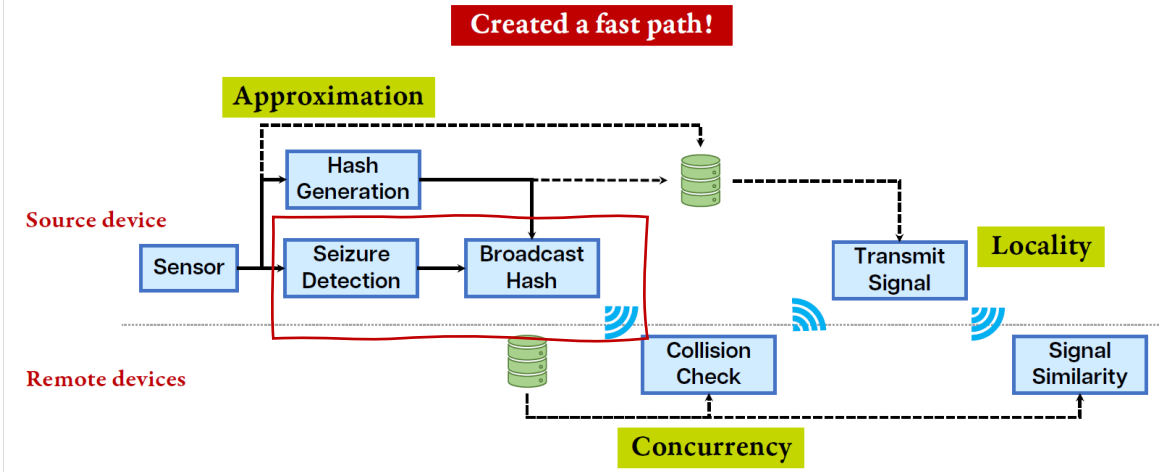
No multitasking, caching, load/store mechanisms or distributed shared memory

So, no need for coherence or consistency? What about other forms of concurrency?

Need a concurrency model (spec) anywhere there is a *resource race*



Need concurrency control



Communication is statically time multiplexed

Processor nearest (in time) to the slot will broadcast

Prolongs detect-to-share latency, but acceptable for small numbers

Similar spec even for the single-PE DWT architecture

Takeaways

What is the challenge with (multicore/distributed) system design and programming?

Implementing and reasoning *correctly* about it

What are two specifications that help the above?

Cache coherence: about writes to the same location across caches
Memory consistency: about ordering of memory accesses

What is cache coherence?

Each write is eventually visible and writes to the same location are serialized

What is memory consistency?

Specification of what values a read can return and when

What are the principles at play in defining and implementing coherence and consistency?

Abstraction, Efficiency, “Yummy”, Concurrency, Approximate

Image Credits (Educational, Fair Use)

- Title image: VLADGRIN, https://www.istockphoto.com/vector/human_-machine-gm147409511-16840728 (Educational fair use)
- Infinite brain: Science wonder stories, May 1930, Illustrator: Frank R Paul, Editor: Hugo Gernsback
- Brain color, ICs, cloud server, black rat: No attribution required (Hiclipart)
- Hand with spoon: public domain freepng
- Signals: https://www.nature.com/articles/nrn3724
- Thought clouds: F. Willett et al./*Nature* 2021/Erika Woodrum, https://med.stanford.edu/neurosurgery/news/2022/bci-award. https://www.the-scientist.com/news-opinion/brain-computer-interface-user-types-90-characters-per-minute-with-mind-68762
- Picture of scientists: https://www.cs.auckland.ac.nz/~brian/rutherford8.html (original: Pierre de Latil), Bush (Carnegie Science), Others (Wikipedia, National Academies, IEEE, Heidelberg Laureate Forum, and university profile images)
- Flowchart: Pause08 – flaticon.com; Digital brain: Smashicons – flaticon.com; Quantum processor icons created by Paul J. - Flaticon
- Server rack: upklyak – freepik.com
- Arm, Lotus: Adobe stock
- Quantum processor: Rigetti computing
- Images of implanted users: Top: Case Western Reserve University (https://thedaily.case.edu/man-quadriplegia-employs-injury-bridging-technologies-move-just-thinking/), Bottom: Jan Scheuermann (University of Pittsburgh/UPMC; https://www.upmc.com/media/news/bci-press-release-chocolate)
- Images of wearable BCIs: Cognixion, NextMind
- Types of BCIs: "Brain–computer interfaces for communication and rehabilitation,
- Illustrative BCI: Neuralink
- Electrodes: "Electrochemical and electrophysiological considerations for clinical high channel count neural interfaces", Vatsyayan et al.
- Form factors: Neuropace, Medtronic, Bloomberg, "Fully Implanted Brain–Computer Interface in a Locked-In Patient with ALS" by Vansteensel et al., Blackrock Neurotech
- Jose Delgado's video: Online, various sources (CNN, Youtube)
- Video of Kennedy and Ramsey: Online, various sources (Youtube, Neural signals)
- Code snippet inspiration: ECE 252 slides at Duke (Dan Sorin et al.)
- Apple processor pipeline: https://dougallj.github.io/applecpu/firestorm.html