Recontextualizing the Memory System

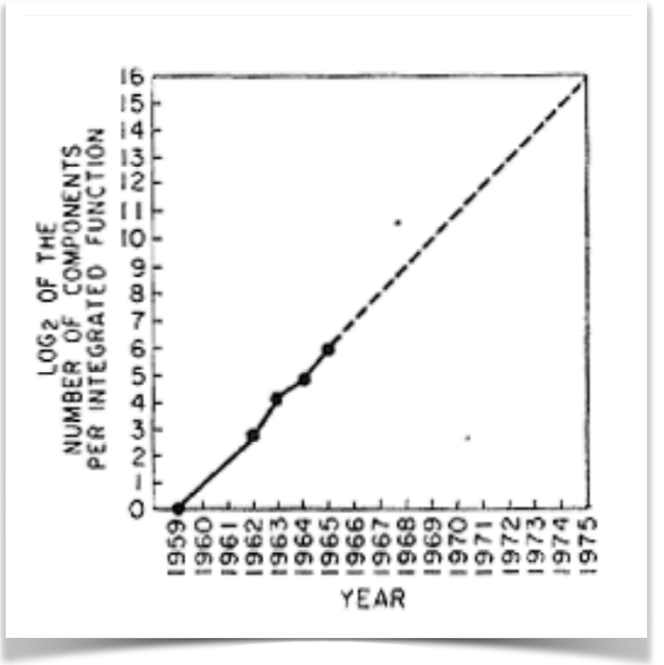
No colloquium today

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

By Gordon E. Moore

Director, Research and Development Laboratories, Fairchild Semiconductor division of Fairchild Camera and Instrument Corp.



Day of reckoning

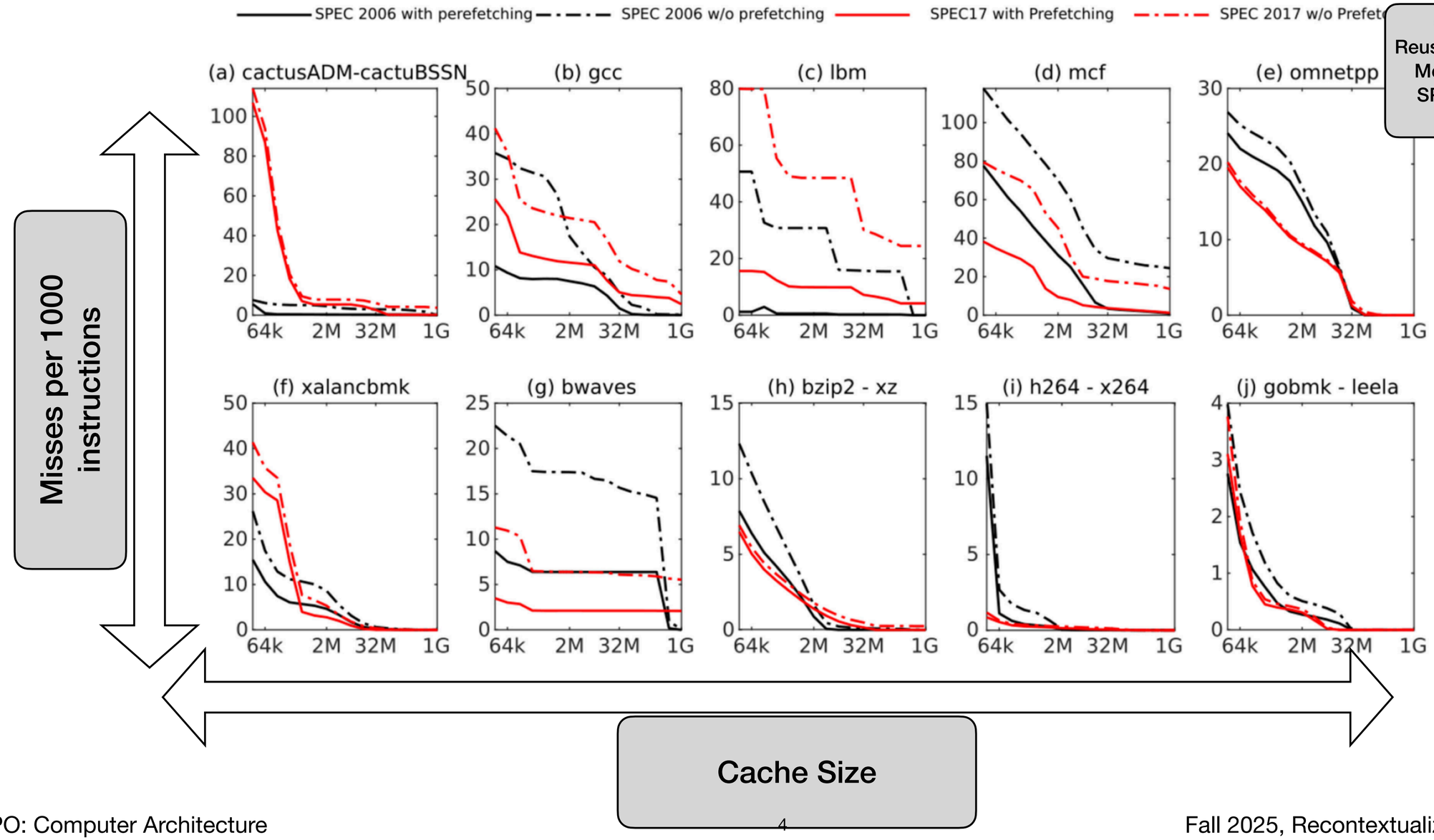
Clearly, we will be able to build such component-crammed equipment. Next, we ask under what circumstances we should do it. The total cost of making a particular system function must be minimized. To do so, we could amortize the engineering over several identical items, or evolve flexible techniques for the engineering of large functions so that no disproportionate expense need be borne by a particular array. Perhaps newly devised design automation procedures could translate from logic diagram to technological realization without any special engineering.

Image credit: http://cva.stanford.edu/classes/cs99s/papers/moore-crammingmorecomponents.pdf

Outline

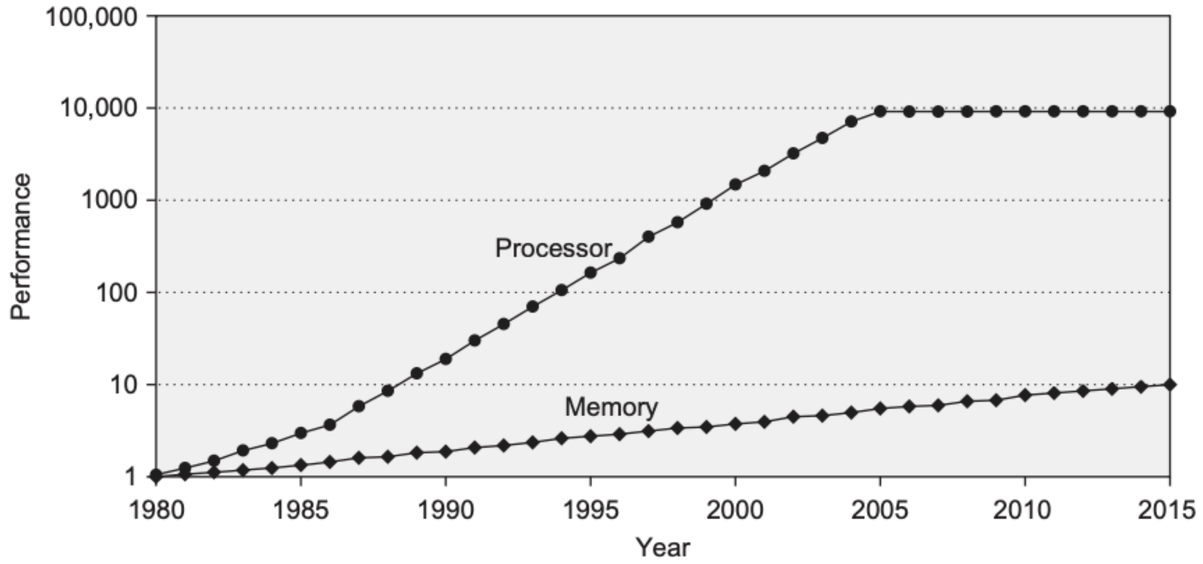
- Recap of the memory system
- Revisiting memory instructions as part of the “processor story”
- What’s next (in this course and in memory design)?

Applications are Increasingly Memory Demanding



Credit: Hassan, et al. "A Reusable Characterization of the Memory System Behavior of SPEC2017 and SPEC2006". ACM TACO 2021

Memory Advances Slower than Processors

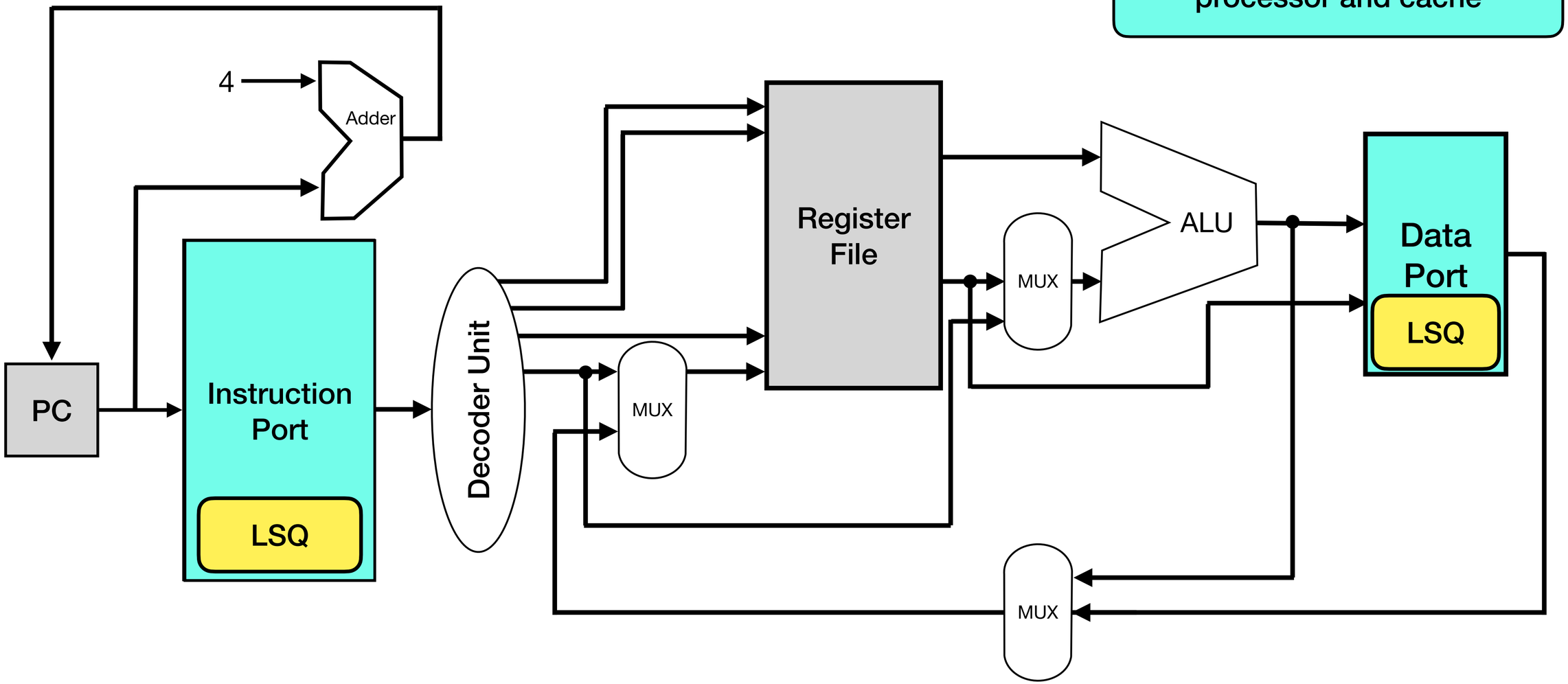


Credit: *Computer Architecture: A Quantitative Approach* (p 80)

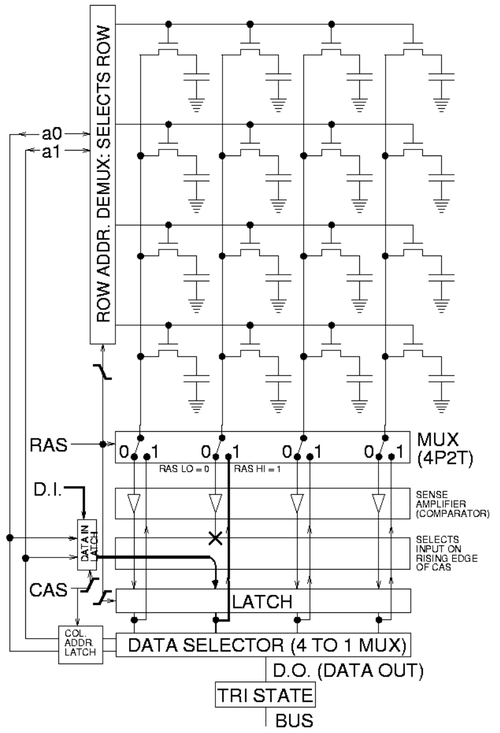
Advances in processor designs are directly related to the increased demand for memory!

Accessing Memory via Ports

"Ports" are physical interfaces that describe connections between components ➡️ processor and cache



The “Large/Fast” Memory

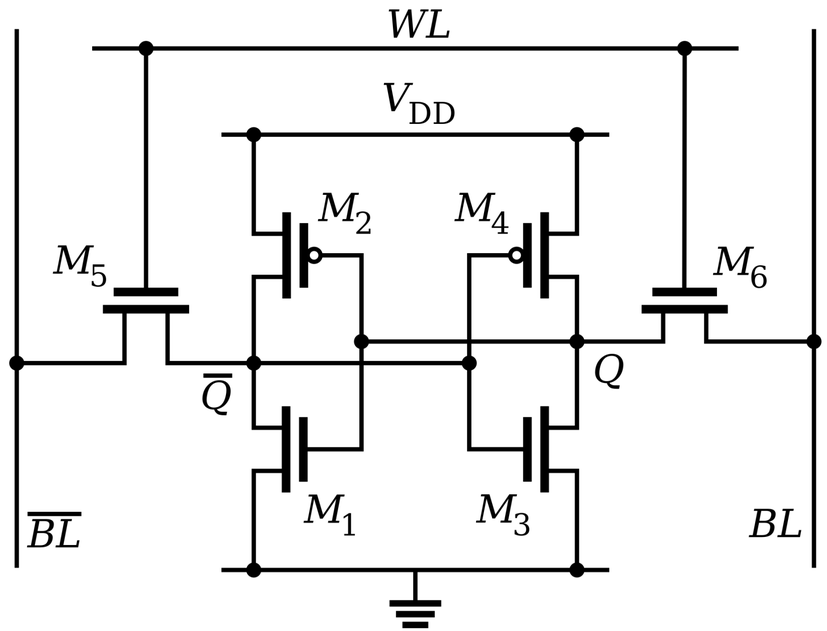


DRAM Diagram

Image credit: https://en.wikipedia.org/wiki/Dynamic_random-access_memory

Do we want our memory system to have a large capacity or be fast to access?

We want large *and* fast!

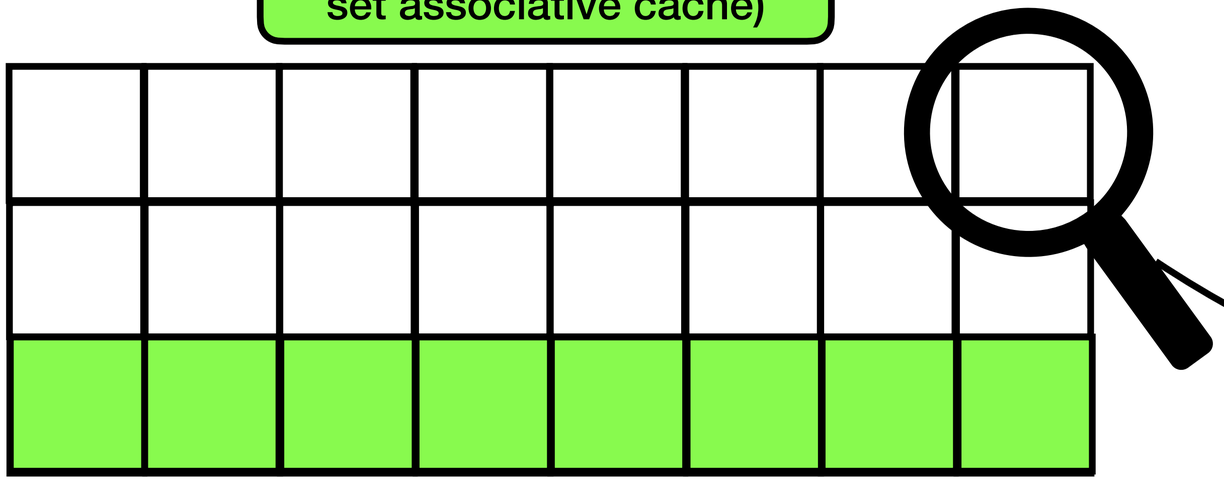


SRAM Transistor Diagram

Image credit: https://en.wikipedia.org/wiki/Static_random-access_memory

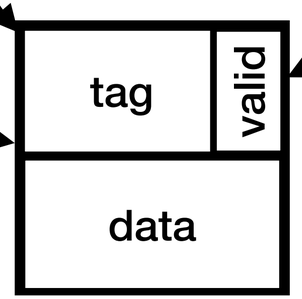
Caches to Build Large/Fast Memory

Associativity: describes size of cache sets in a cache (e.g., this is an 8-way set associative cache)



Tag: unique identifier for the data in a cache block

Valid bit: indicates if block currently holds legitimate data



Set: the possible blocks to which some data may be stored within the cache

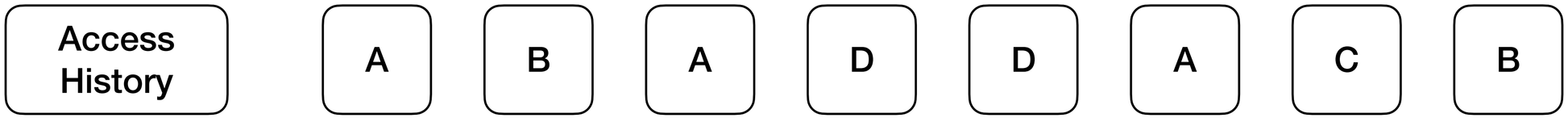
Cache: the collection of *cache blocks* that buffers/stores commonly reused items

Cache Block: the minimum unit of information that can be present/not present in a cache (typically data is the size of a "word" ➡️ 64 bytes in modern processors)

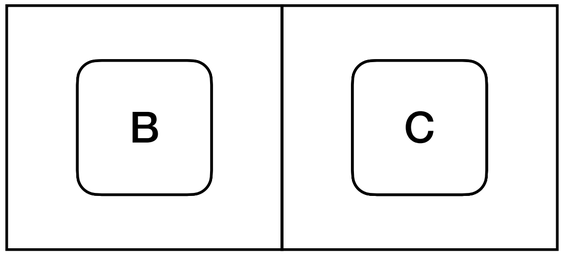
Principles of Locality to Mitigate Misses

- Applications tend to exhibit *temporal* and *spatial* locality when they are deployed!
- *Temporal locality* describes the likelihood of an application to reuse data within similar periods of time
- *Spatial locality* describes the likelihood of an application to reuse data at similar addresses to one another

Cache Replacement Policies*



Least Recently Used

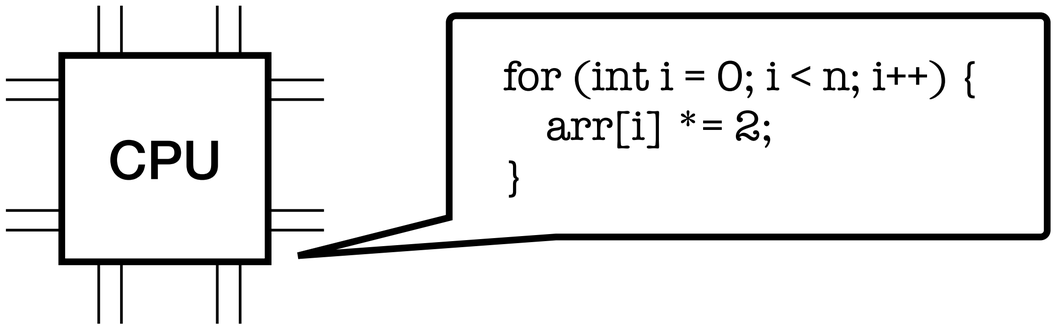


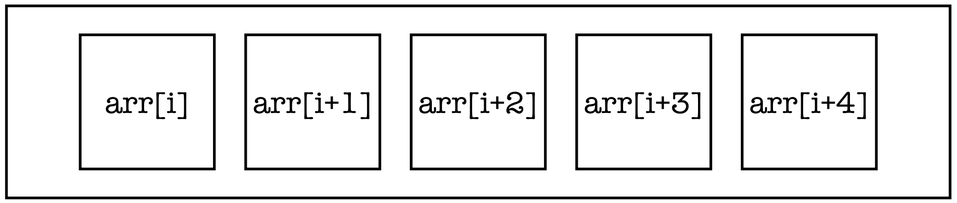
Chat with your neighbor(s)!

If we wanted to implement least-recently used in the construction of the cache logic, what additional state would need to be maintained?

Takeaway: if we want to implement a replacement policy based on application behavior, then we need to track *metadata* (e.g., time of last access) in the cache to make replacement decisions

Cache Prefetching



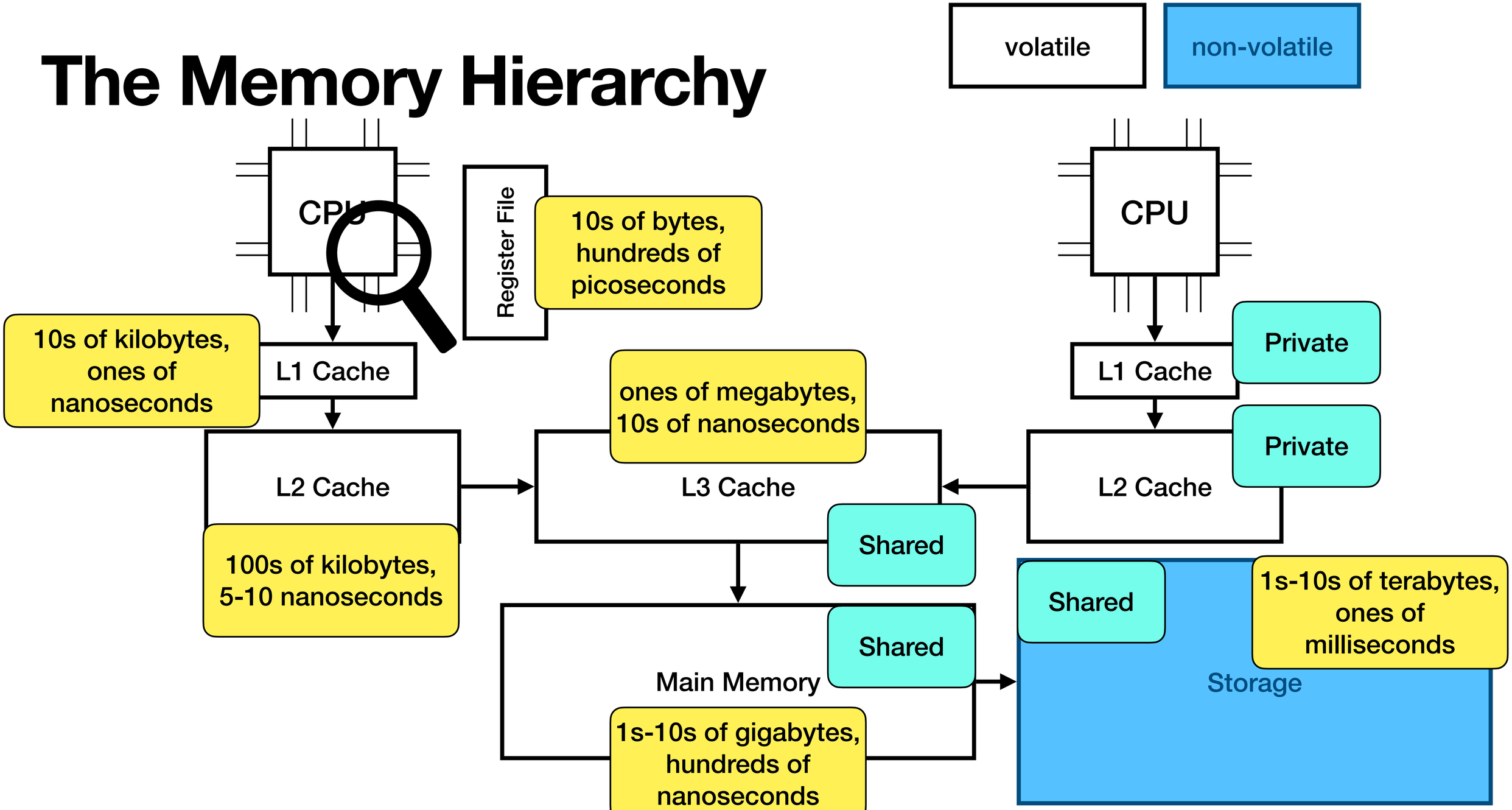


Regular “stride length”

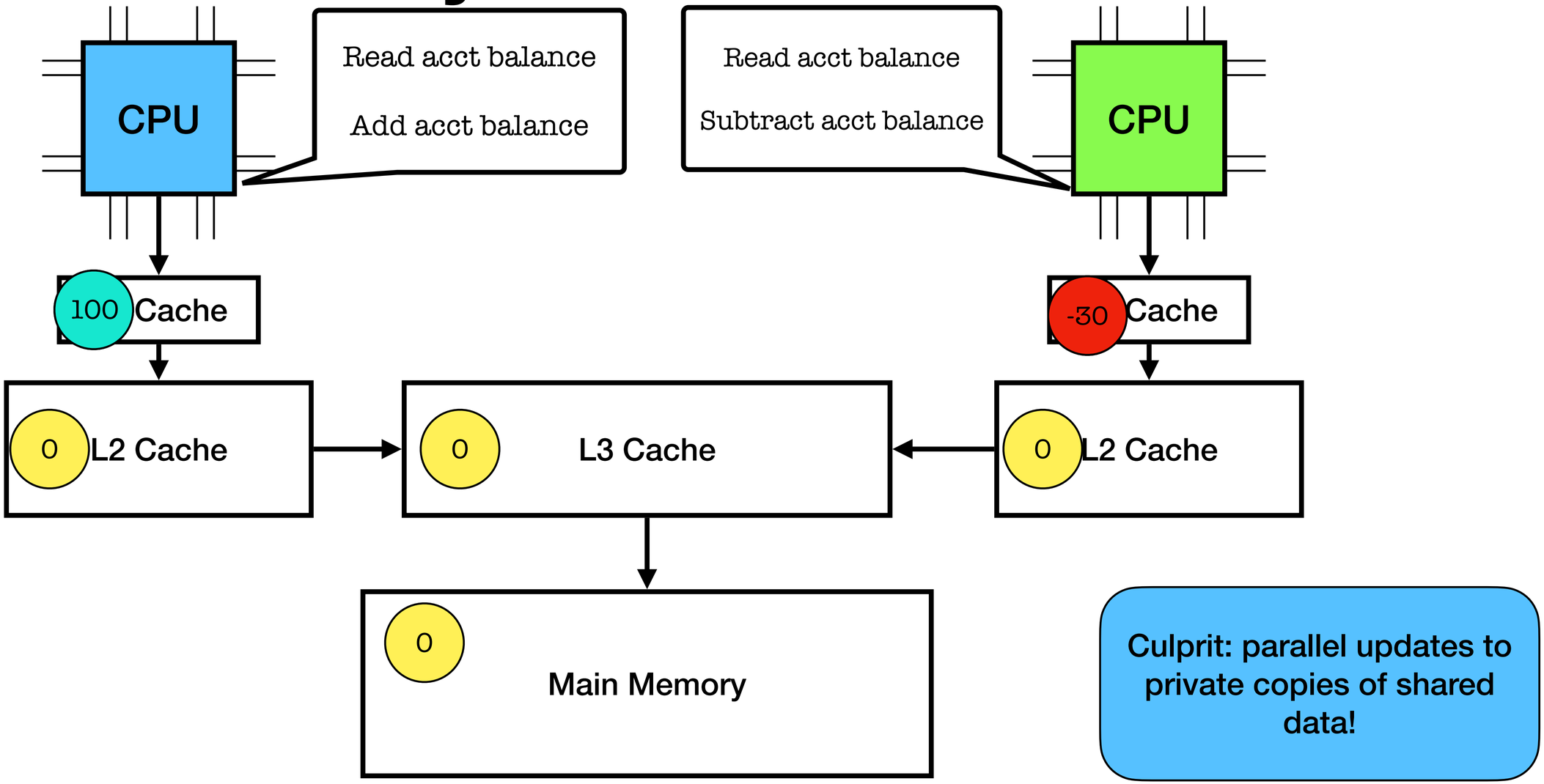
Also true of instructions!

Takeaway: caches can make requests to lower levels of the memory system *on their own* to mitigate compulsory misses by tracking recent access patterns

The Memory Hierarchy



Data Consistency Issues in Shared Caches



Culprit: parallel updates to private copies of shared data!

Side Channel Leakage in Caches

TABLE III: Example dictionary-assisted password guessing attack for password “hello”.

Input	Confidence Vector (Partial)							
	e	h	i	j	l	o	s	y
h	0.0	0.39	0.0	0.23	0.0	0.0	0.0	0.03
e	0.21	0.0	0.0	0.0	0.0	0.0	0.0	0.03
l	0.0	0.0	0.05	0.0	0.37	0.0	0.07	0.0
l	0.0	0.0	0.05	0.0	0.37	0.0	0.07	0.06
o	0.0	0.0	0.0	0.0	0.0	0.15	0.0	0.0

Rank	Dictionary Words	Confidence Value
1	hello	$0.39 + 0.21 + 0.37 + 0.37 + 0.15 = 1.49$
2	jelly	$0.23 + 0.21 + 0.37 + 0.37 + 0.0 = 1.18$
3	hills	$0.39 + 0.0 + 0.37 + 0.37 + 0.0 = 1.13$
4	holly	$0.39 + 0.0 + 0.37 + 0.37 + 0.0 = 1.13$

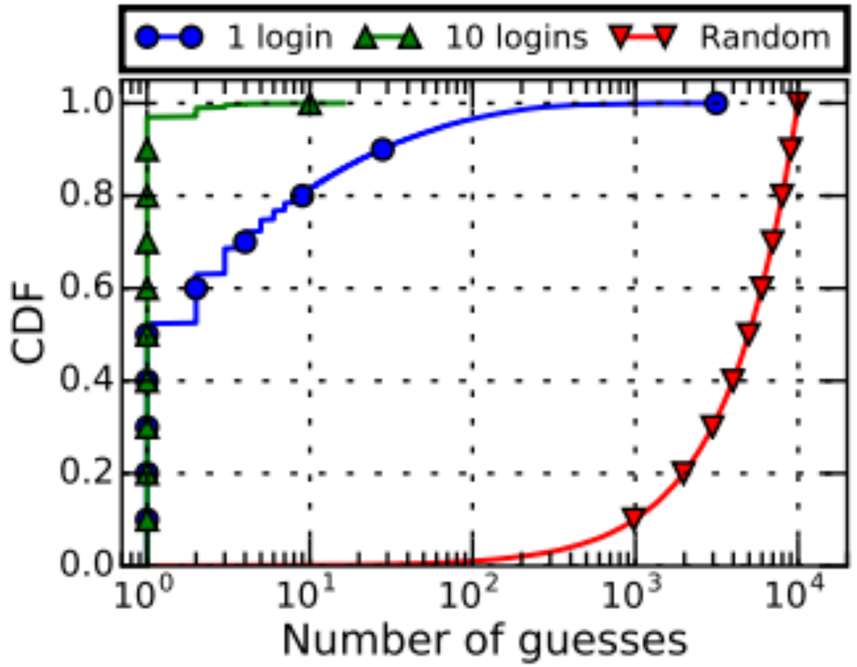


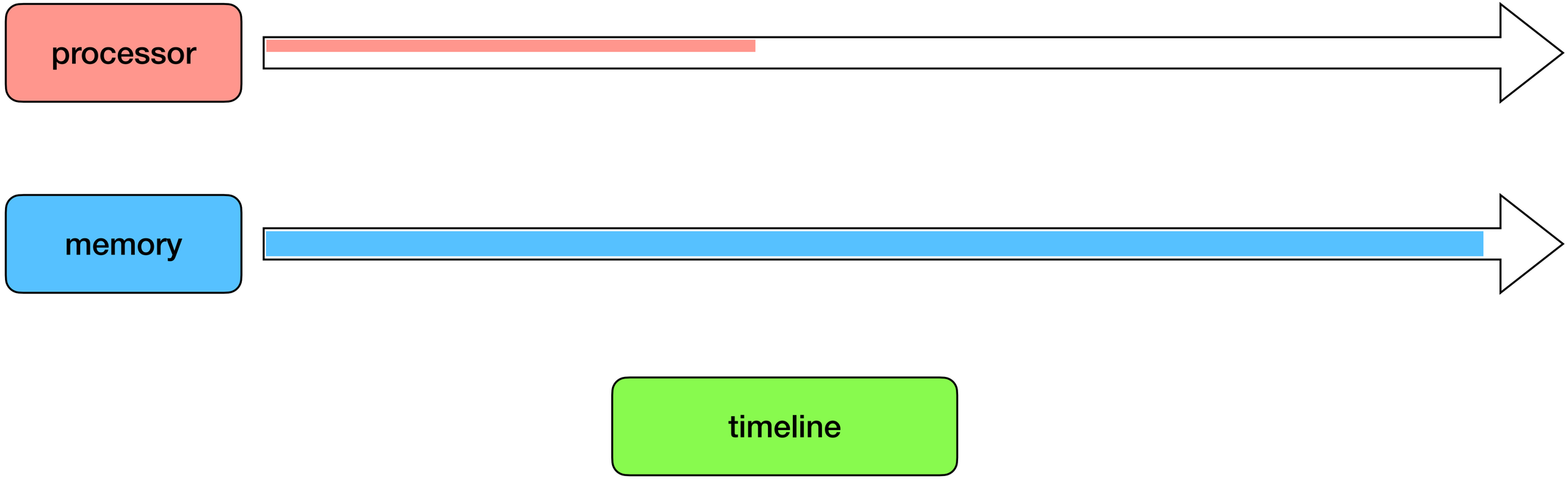
Image credit: https://www.ndss-symposium.org/ndss-paper/unveiling-your-keystrokes-a-cache-based-side-channel-attack-on-graphics-libraries/

Summary

- “Instruction-” and “data-memory” are not physical components that fit on the processor, instead they refer to a *memory system* of components designed to give the illusion of an individual device
- We can leverage common principles of application behavior to, with high probability, and mitigate cache misses by predicting which memory addresses will be accessed next
- “Optimizing the common case” can leads towards pitfalls with respect to the consistency of data throughout the cache hierarchy and leakage via side channels

We've covered most concepts relating to memory construction and memory system design up to and including literature being described today!

We've covered some of the important foundations in processor design, but have omitted details, and our processor is representative of a device from 1990…



What's Next…