Memory hierarchy

Recap thus far

Architecture studies HW/SW interface ("how a computer works")

ISAs: interface between high-level languages and hardware

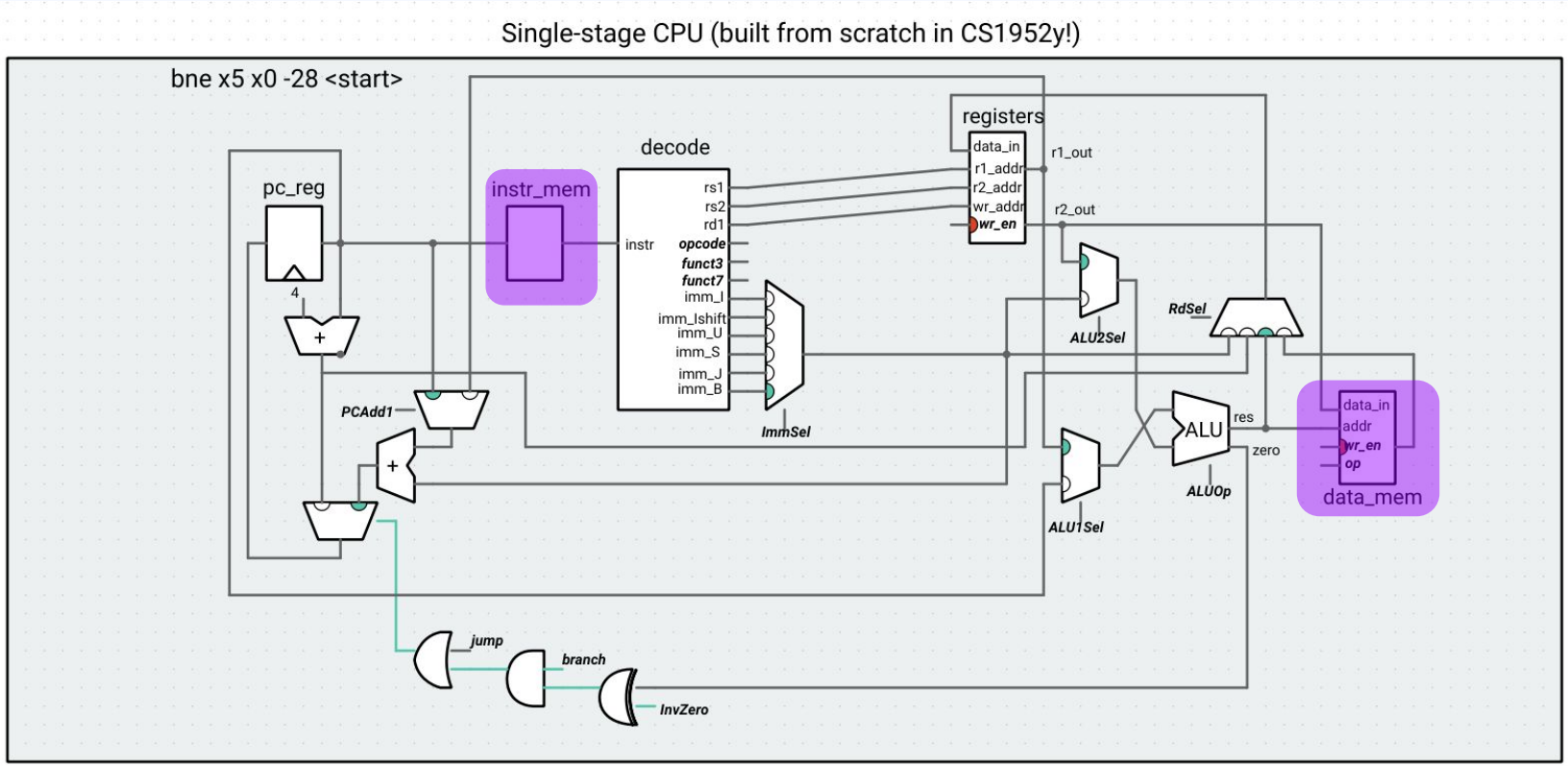
Microarchitectures: implementation of ISAs

Focused mostly on correctness, started talking about speed/throughput and *vaguely* power

We've built two CPUs (and two-ish more in HW2), which can execute programs from memory...

...assuming the program exists in memory and memory works!

What do we know (so far) about memory?



Persistent storage

Slower
Bigger
Cheaper

By Evan-Amos - Own work, CC BY-SA 3.0 (link)

HDD (Hard Disk Drive)

Magnet-based
10s of TBs of capacity
<$15/TB
100s of MB/s read speed (ms access time)
100s yrs lifetime

SSD (Solid State Disk)

Flash-based
TBs of capacity
<$100/TB
10s of GB/s read speed (10s of us access time)
~10 yrs lifetime

Faster
Smaller
Pricier

By Jacek Halicki - Own work, CC BY-SA 4.0 (link)

this is *secondary storage* (CPU needs I/O bus to access); NOT memory

Main memory

Primary storage – holds instructions + data while program is running

Volatile (does not persist when power is turned off)

DRAM (dynamic random access memory) technology

10s of GBs of capacity

$5-$10/GB

60 ns access time

This is still *really* slow compared with modern processor clock speeds

Processor-memory performance gap

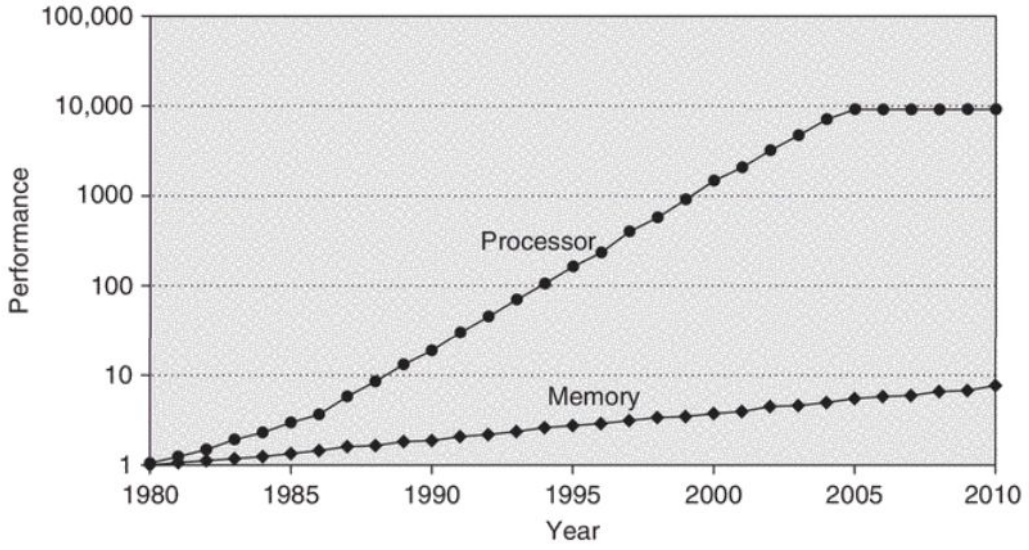


Figure 2.2 Starting with 1980 performance as a baseline, the gap in performance, measured as the difference in the time between processor memory requests (for a single processor or core) and the latency of a DRAM access, is plotted over time.

Computer Architecture: A Quantitative Approach, John L Hennessy and David A Patterson

SRAM

“Static Random Access Memory”

Less dense than DRAM (more $$$/area) but faster

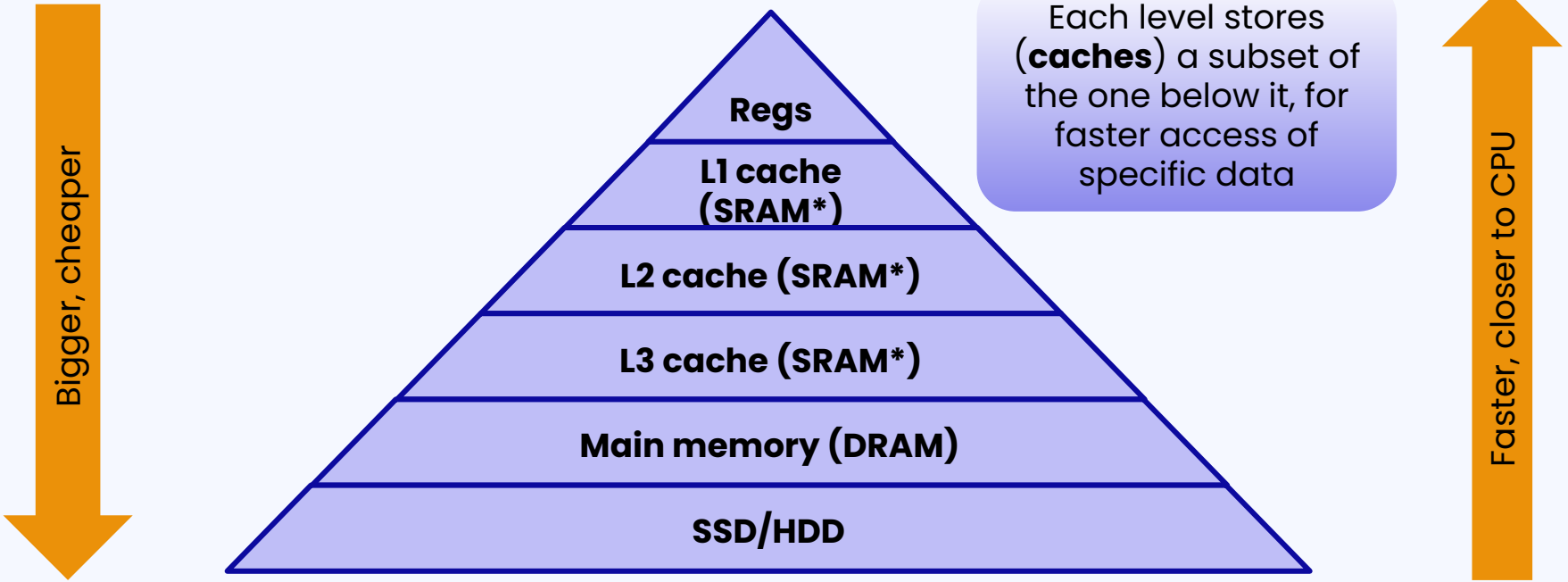
1-100s of MB of capacity

0.5-10s of ns access time

not practical for main memory, but want to take advantage of speed

What do do?

Memory hierarchy



Each level stores (**caches**) a subset of the one below it, for faster access of specific data

L1-L3 caches use the same technology (SRAM).
Why do they have different access speeds?

Modern CPU layouts

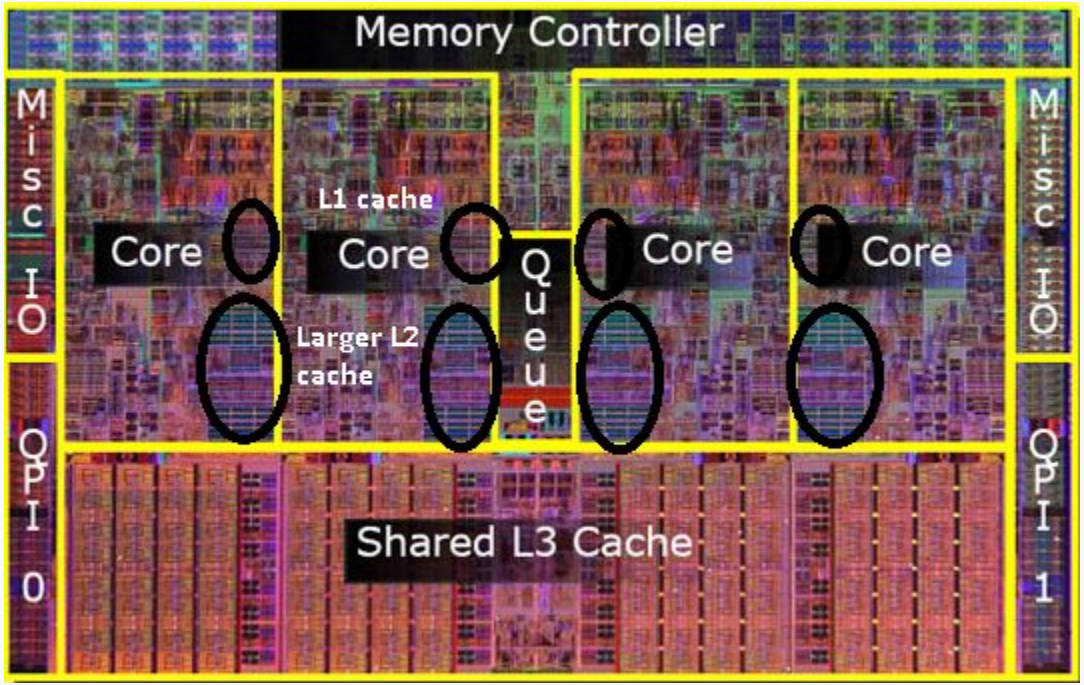


image source

A slightly more detailed picture

Modern CPU specs

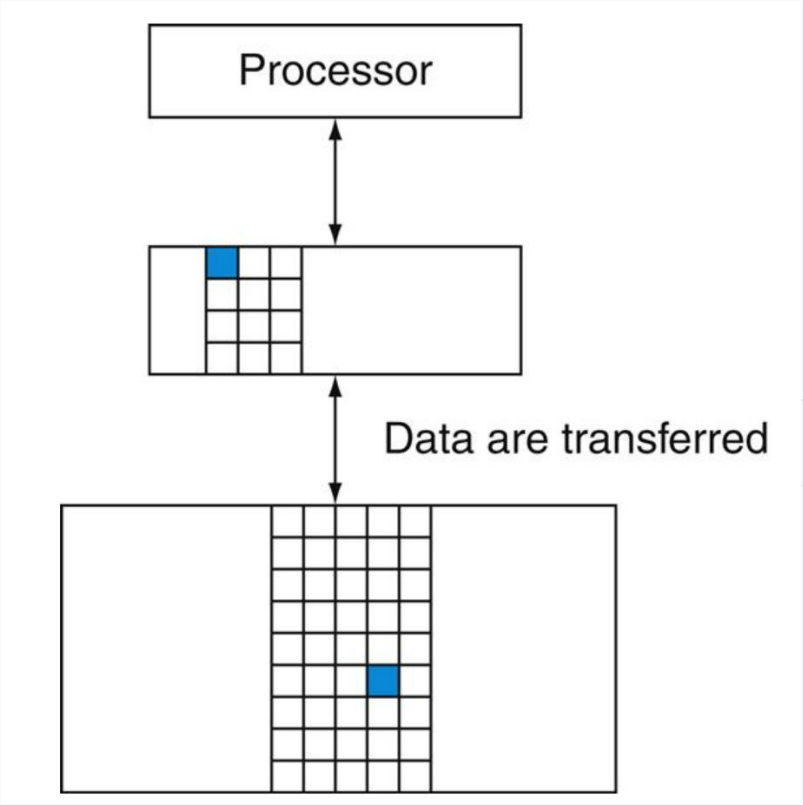
	AMD Ryzen 5 7600	Apple M3	Intel Core Ultra 7 265K	Qualcomm Snapdragon X1E-84-100
L1 cache	64 KB / core	192 KB / core* I-cache, 128 KB / core* D-cache	192 KB / core (market an “L0” 48 KB D- and 64 KB I-cache)	288 KB / core
L2 cache	1 MB / core	16-64 MB / core*	3 MB / core	12 MB / “module” (4 cores)
L3 cache	32 MB	8-96MB* (M2 numbers)	30 MB	6 MB

Blocks

Block (or line) - minimum unit of information that can be present/not present in a cache

Blocks are transferred between levels in the memory hierarchy

Modern CPUs: often 64 bytes (128 for Apple Silicon)



P&H 5.2

Principle of locality

Observations about the ways programs access data

Temporal locality: if data is referenced, it will tend to be referenced again soon

Spatial locality: if data is referenced, data whose addresses are close by will tend to be referenced soon

Is a for-loop that iterates through an array once an example of spatial or temporal locality?

In the context of caching, why is spatial locality useful? Why is temporal locality useful?

What questions do we need to ask when designing a cache?