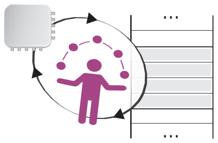
cs704fig_mem.cdr
Thursday, April 18, 2024 12:59:48 PM
Color profile: Disabled

Composite Default screen

Memory

Recall ...



The processor can only keep a part of a computation; portions must move back and forth as needed.

Same memory holds both instructions and data in a modern stored-program computer.

➪ Essential role: hold computation

🗝 ✎ Close interaction with processor

✎ Limit computation size

✎ Limit performance

therefore,
3 issues matter

➪ Design goal (ideally)

Maximum amount of fastest memory

Memory Solutions

➪ Primary storage

Memory raised substantial challenges due to a growing performance gap between the two parts, essential to performing computations, that must interact closely.

👁

Primary storage is an older term for main memory where actively running code is stored in RAM or ROM.

➪ CPU-memory technology gap

Not all storage systems are usable as memory (i.e., to hold jobs marked or selected for execution)

➪ Revised Design Goal

Maximum amount of fastest memory economically

cs704fig_mem.cdr
Thursday, April 18, 2024 12:59:49 PM
Color profile: Disabled

Composite Default screen

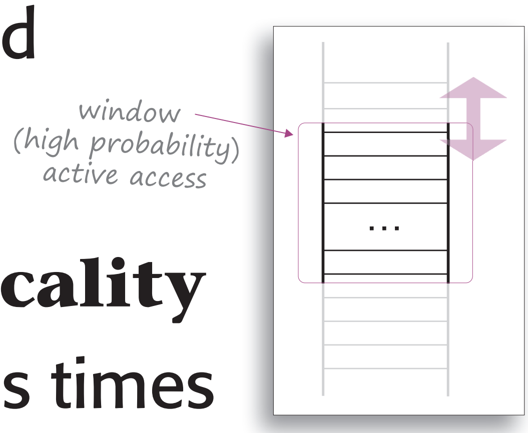
Principle of Locality

➪ [Memory] References

Memory generally stores program instructions and data, but how are they actually used?

➪ Memory access patterns

- ✎ Most recently used
- ✎ Close locations



Where do these patterns come from? (Next.) How probable? (Later.)

➪ Mem *reference* locality

- ✎ Temporal ~ access times
- ✎ Spatial ~ access locations

Quiz
What's the difference? [memory access context] *Address*, *label*, *pointer*, and *reference*.

Locality of Reference Examples

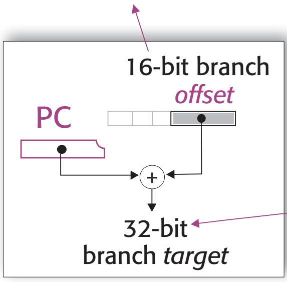
Give examples from high-level programs for temporal and spatial locality in data (min 2–3 each). Repeat for instructions.

Exercise
Lookup how *objects* are typically stored in memory. Suggest locality opportunities.

Think about your programs

- ✎ Data locality examples ← vars, arrays, data structures...
- ✎ Instruction locality examples ← loops, subroutines conditionals...

±32K ?



Quiz
Which type of locality is exploited?

Note MIPS PC-relative addr

Store an offset (a difference)

```
beq $4,$5,Exit
```

In 2021–22, as much as 4–5 orders of magnitude (approx. 10K to 100K); see next.

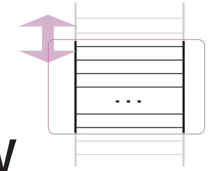
Only a relatively small amount at a time needs to be referenced very quickly.

Watch keywords: amount (**capacity**), economical (**cost**), and access speed (**performance**).

To sum up

✎ Storage vary wildly in cost

✎ Small active reference window



Hierarchical storage: a solution

Largest amount of economical storage accessed, most of the time, at the speed of the fastest storage

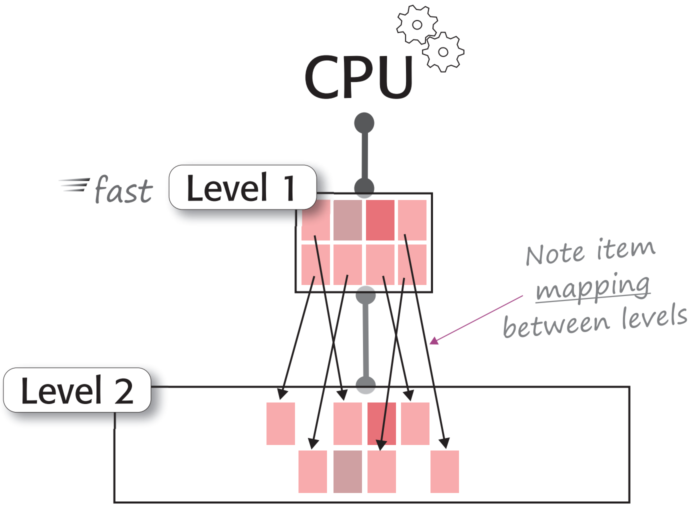
Hierarchical Storage

➪ Inclusion

The fastest memory, typically expensive and therefore small (as much as design economics afford), is directly connected to the CPU.

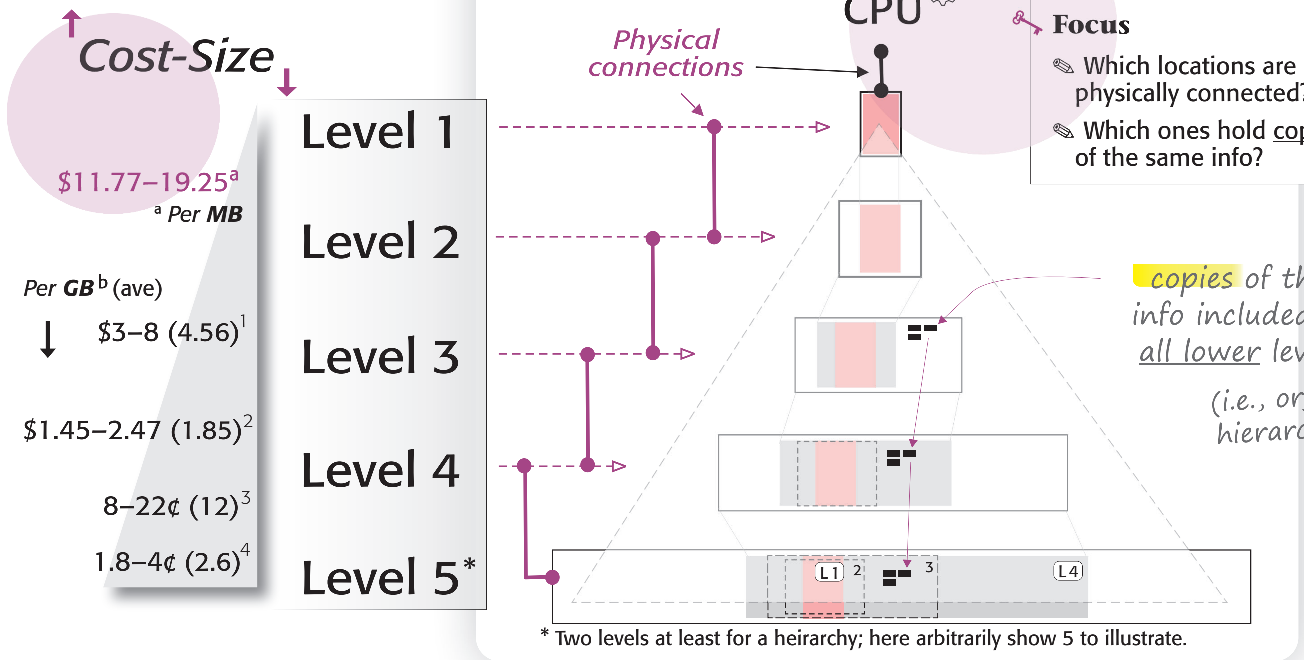
Typically insufficient, so additional, more economical, but slower memory (too slow to hook to the CPU) may be connected as a secondary back store.

Items in the upper level numbered 1, are <u>included</u> in (a <u>proper</u> subset of) the lower back store level, numbered 2.



A Memory Hierarchy

➪ Inclusion



Focus

- ✎ Which locations are physically connected?
- ✎ Which ones hold copies of the same info?

👁 copies of this info included in all lower levels

(i.e., organized hierarchically)

* Two levels at least for a heirarchy; here arbitrarily show 5 to illustrate.

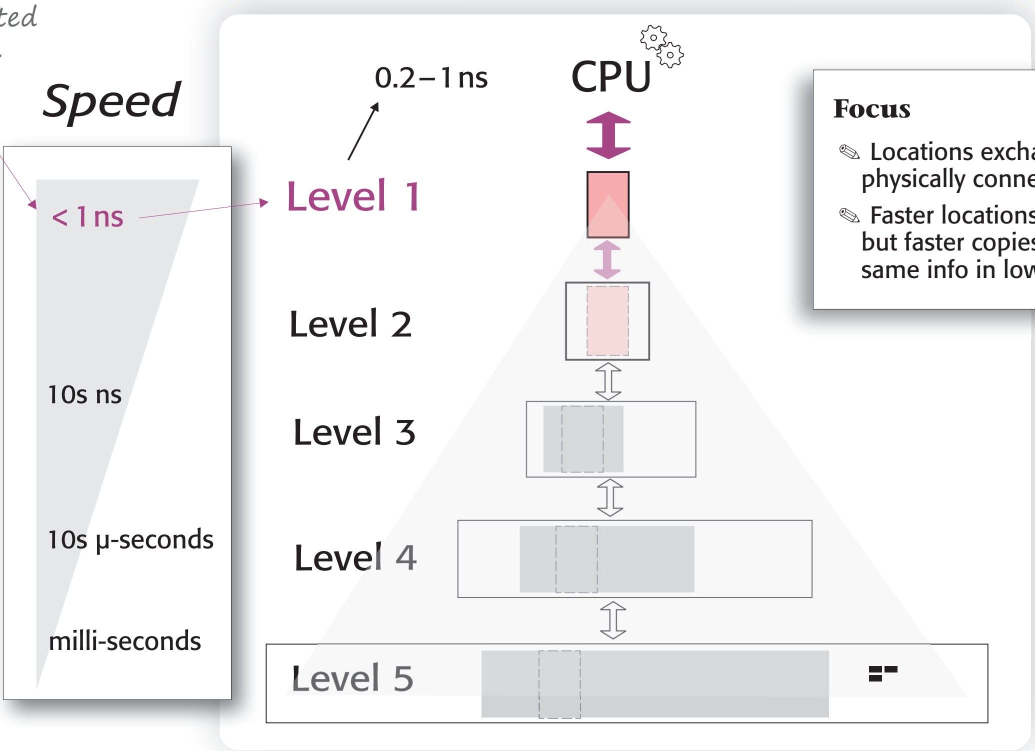
[a] SRAM/Chips (Cypress/Infineon) 0.45/3/45 ns arrow.com@2021/12/13 ▪ Integrated on-die <$10/MB likely (?) in 2021

[b] Consumer/retail newegg.com@2021/12/14 (best selling) ▪ [1]DRAM DDR3-1333 4/8G DIMM ▪ [2]SSD/3D XPoint (Intel Optane) ▪ [3]SSD/NAND 256-1TB ▪ [4]HDD/internal 10-20 TB

Memory Systems
Hierarchy Operation

➪ Miss Time *penalty*

*Best case scenario:
CPU finds requested
info in Level 1.*



Focus

- ✎ Locations exchange info if physically connected
- ✎ Faster locations hold fewer but faster copies of the same info in lower levels

👁

Level access times add up as info is fetched from lower levels.

Least used parts of <u>active</u> programs; note indicated items will be very costly to fetch upon request.

cs704fig_mem.cdr
Thursday, April 18, 2024 1:00:06 PM
Color profile: Disabled

Composite Default screen

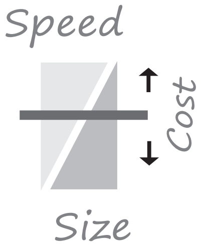
Summary

Quiz
Even if memory were super fast and cheap, is it a good idea to connect the CPU to all the memory?

Exercise
Why store hierarchically (i.e., include addresses in lower levels)? **Hint**: look for answers in reading material; note lower levels are higher-numbered.

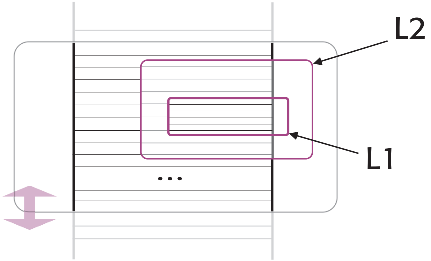
⇨ What is a memory hierarchy?

- ✎ A speed-size tradeoff
- ✎ Cost-driven
- ✎ Access locality-enabled



As much of the most actively used memory is kept in the fastest locations (upper part of the hierarchy) close to the CPU as long as possible (not as long as needed, unfortunately, occasionally stuff is evicted prematurely).

⇨ Why it works?



cs704fig_mem.cdr
Thursday, April 18, 2024 1:00:07 PM
Color profile: Disabled

Composite Default screen

Memory Systems
Classic Hierarchy

➪ Coherence

Some logical memory addresses will have many copies of their contents in the physical memory system.

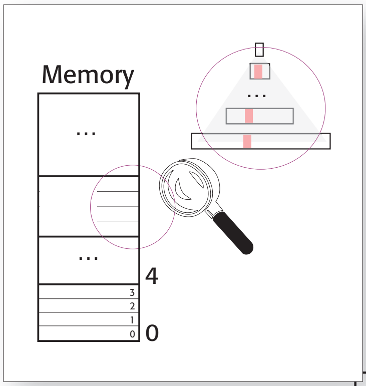
Quiz What if the CPU changes its copy? What about a multiple CPU scenario? Should a cache be read-only?

Static RAM (SRAM) Faster, expensive volatile bits made from transistors.

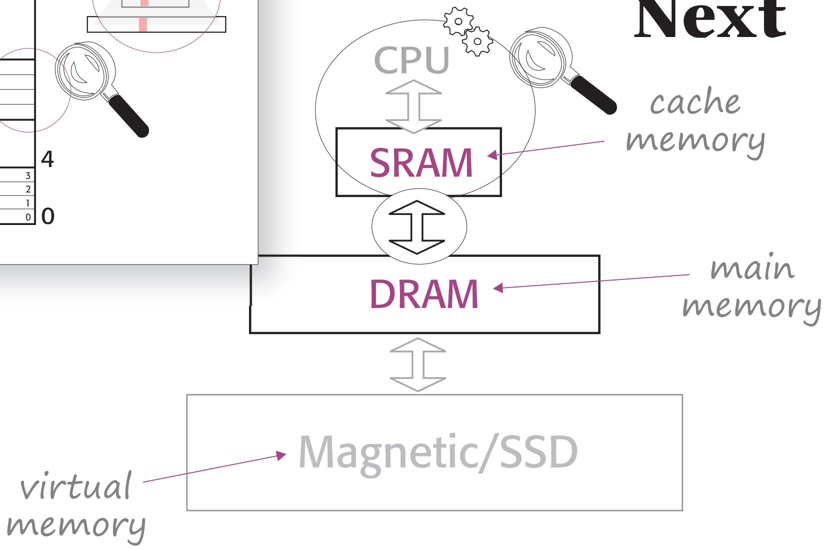
Dynamic RAM (DRAM) Cheaper, denser volatile bits made from capacitors.

Magnetic/SSD Very cheap, persistent bits, electro mechanical magnetic storage or transistor-based solid-state flash memory.

Recently, phase-change solid-state (based on changing resistivity due to crystalline phase change) such as Intel/Micron *3D XPoint* (cross-point).



Next


CPU
SRAM
DRAM
Magnetic/SSD
cache memory
main memory
virtual memory