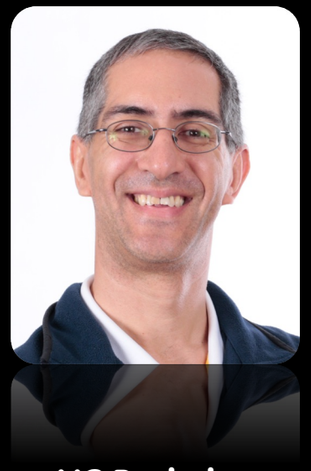


UC Berkeley
Teaching Professor
Dan Garcia

CS61C

Great Ideas in Computer Architecture

(a.k.a. Machine Structures)

Caches I

Binary Prefix

- Binary Prefix
- Library Analogy
- Memory Hierarchy
- Locality, Design, Management

Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta

- Common use prefixes (all SI, except K [= k in SI])
- Confusing! Common usage of "kilobyte" means 1024 bytes, but the "correct" SI value is 1000 bytes

- Hard Disk manufacturers & Telecommunications are the only computing groups that use SI factors
  - What is advertised as a 1 TB drive actually holds about 90% of what you expect
  - A 1 Mbit/s connection transfers 106 bps.

Name	Abbr	Factor	SI size
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

kibi, mebi, gibi, tebi, pebi, exbi, zebi, yobi

- IEC Standard Prefixes [only to exbi officially]

Name	Abbr	Factor
kibi	Ki	2^{10} = 1,024
mebi	Mi	2^{20} = 1,048,576
gibi	Gi	2^{30} = 1,073,741,824
tebi	Ti	2^{40} = 1,099,511,627,776
pebi	Pi	2^{50} = 1,125,899,906,842,624
exbi	Ei	2^{60} = 1,152,921,504,606,846,976
zebi	Zi	2^{70} = 1,180,591,620,717,411,303,424
yobi	Yi	2^{80} = 1,208,925,819,614,629,174,706,176

- International Electrotechnical Commission (IEC) in 1999 introduced these to specify binary quantities.
- Names come from shortened versions of the original SI prefixes (same pronunciation) and bi is short for "binary", but pronounced "bee" :-(
- Now SI prefixes only have their base-10 meaning and never have a base-2 meaning.

Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta

1. Kid meets giant Texas people exercising zen-like yoga. – Rolf O
2. Kind men give ten percent extra, zestfully, youthfully. – Hava E
3. Kissing Mentors Gives Testy Persistent Extremists Zealous Youthfulness. – Gary M
4. Kindness means giving, teaching, permeating excess zeal yourself. – Hava E
5. Killing messengers gives terrible people exactly zero, yo
6. Kindergarten means giving teachers perfect examples (of) zeal (&) youth
7. Kissing mediocre giraffes teaches people (to) expect zero (from) you
8. Kinky Mean Girls/Guys Teach People Exciting Zen Yoga
9. Kissing Mel Gibson, Teddy Pendergrass exclaimed: "Zesty, yo!" – Dan G
10. Kissing me gives ten percent extra zeal & youth! – Dan G (borrowing parts)

Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta

Let's ask ChatGPT…

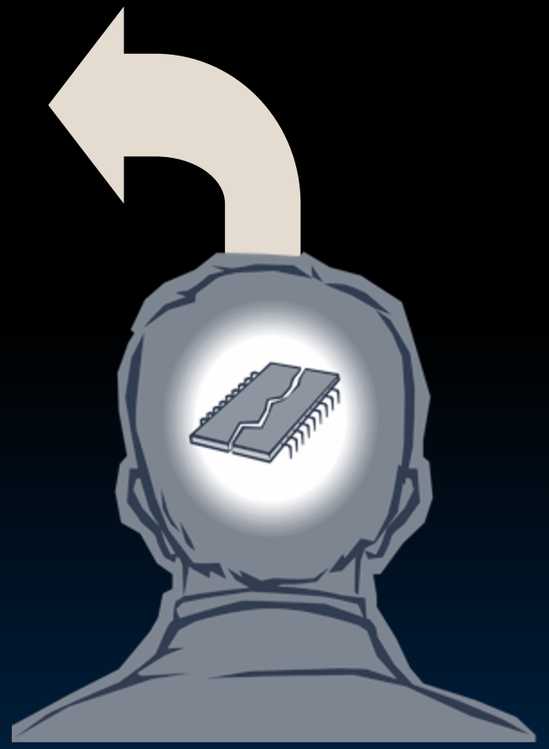
1. Kids May Give Tiny People Excellent Zeppelins, Yonder
2. Kind Mothers Give Tender Patience, Especially Zebras Yawning
3. Kittens Might Grow Taller Playing Every Zoo Yodel
4. Keep My Garden Tidy, Please; Every Zebra Yawns
5. Koalas Make Great Teachers, Proudly Educating Zealous Youths
6. Kites Must Glide To Pass Every Zipping Year
7. King Midas Grabs Ten Pearls, Each Zealously Yearned (for)
8. Keep My Goat Tranquil; Penguins Eat Zebra Yogurt
9. Kangaroos Might Grow Tall, Preferring Extra Zebra Yogurt
10. Kids Make Games To Play Every Zesty Year
11. Kids Make Games To Play. EZY! (Easy) – Dan G (borrowing parts)

The way to remember #s

- What is 2^{34}? How many bits to address (I.e., what's **ceil $\log_2$ = lg** of) 2.5 TiB?
- Answer!

2^{XY} means…

X=0	---	Y=0	1
X=1	kibi ~10^3	Y=1	2
X=2	mebi ~10^6	Y=2	4
X=3	gibi ~10^9	Y=3	8
X=4	tebi ~10^{12}	Y=4	16
X=5	pebi ~10^{15}	Y=5	32
X=6	exbi ~10^{18}	Y=6	64
X=7	zebi ~10^{21}	Y=7	128
X=8	yobi ~10^{24}	Y=8	256
		Y=9	512



Library Analogy

- Binary Prefix
- Library Analogy
- Memory Hierarchy
- Locality, Design, Management

New-School Machine Structures

Software

Parallel Requests
Assigned to computer
e.g., Search "Cats"

Parallel Threads
Assigned to core e.g., Lookup, Ads

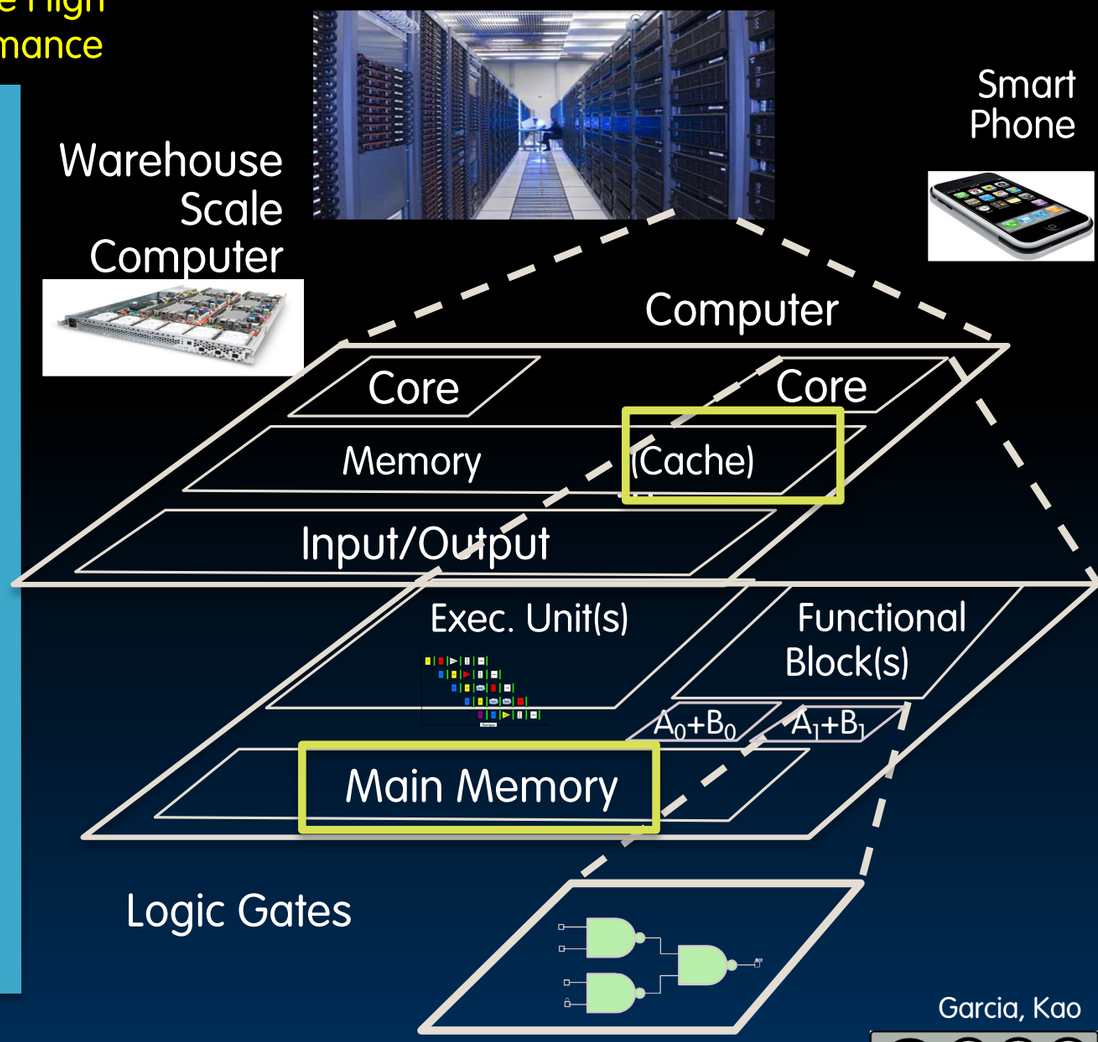
Parallel Instructions
>1 instruction @ one time
e.g., 5 pipelined instructions

Parallel Data
>1 data item @ one time
e.g., Add of 4 pairs of words

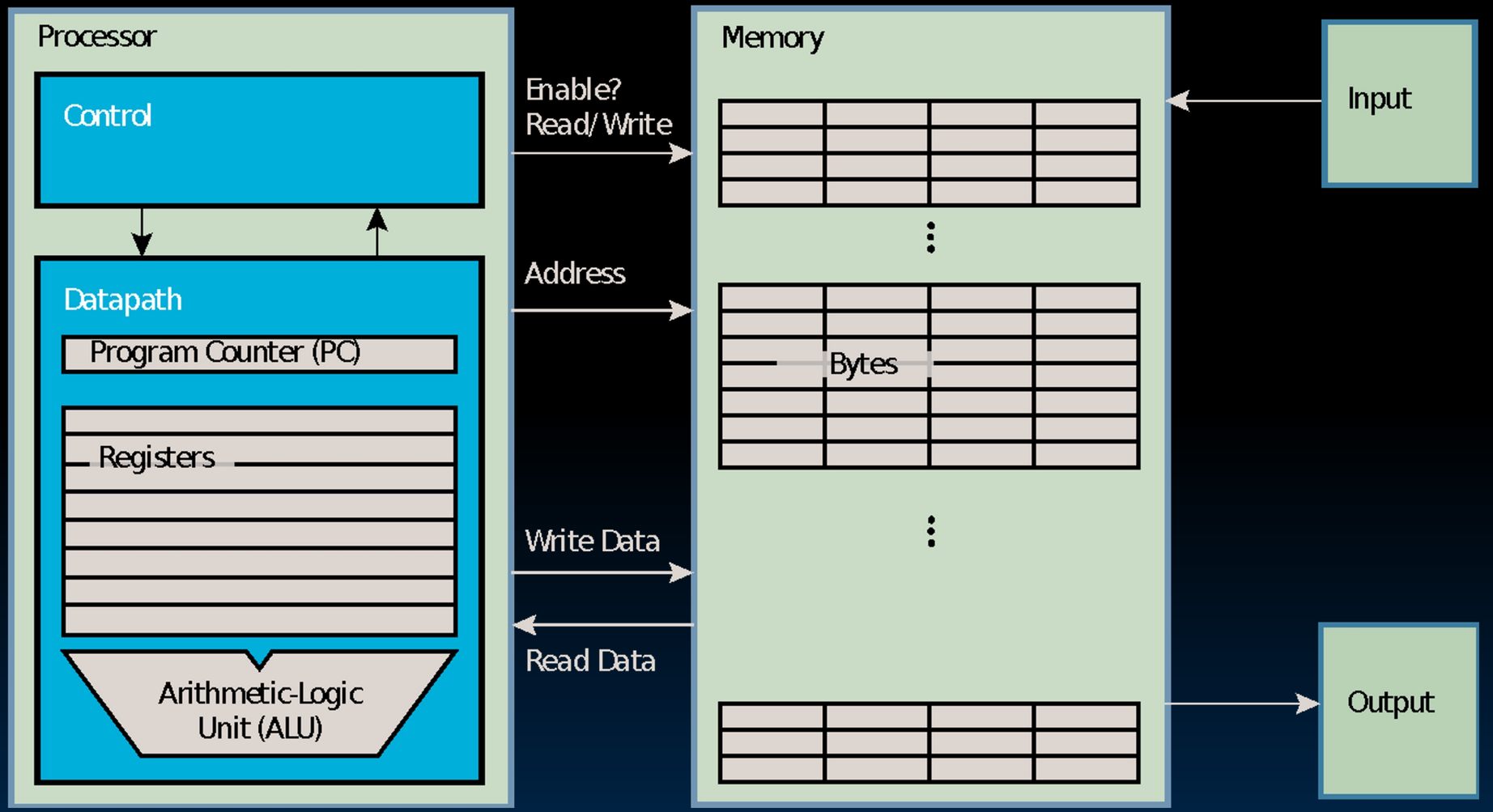
Hardware descriptions
All gates work in parallel at same time

Harness Parallelism & Achieve High Performance

Hardware



Components of a Computer

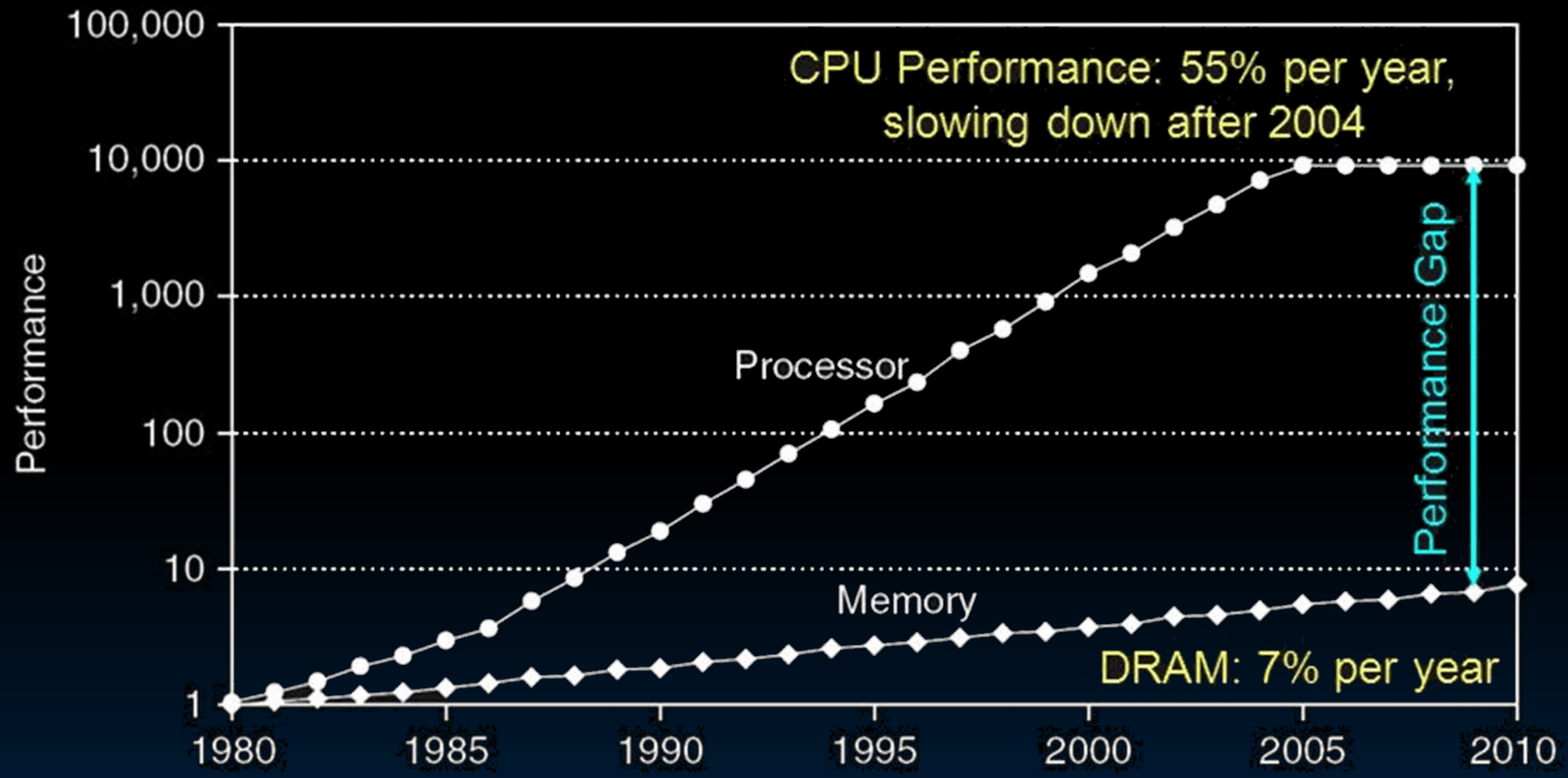


Why are Large Memories Slow? Library Analogy

- Time to find a book in a large library
  - Search a large card catalog – (mapping title/author to index number)
  - Round-trip time to walk to the stacks and retrieve the desired book
- Larger libraries worsen both delays
- Electronic memories have same issue, plus the tech used to store a bit slow down as density increases (e.g., SRAM vs. DRAM vs. Disk)

What we want is a large, yet fast memory!

Processor-DRAM Gap (Latency)



1980 microprocessor executes ~one instruction in same time as DRAM access
2020 microprocessor executes ~1000 instructions in same time as DRAM access

Slow DRAM access has disastrous impact on CPU performance!

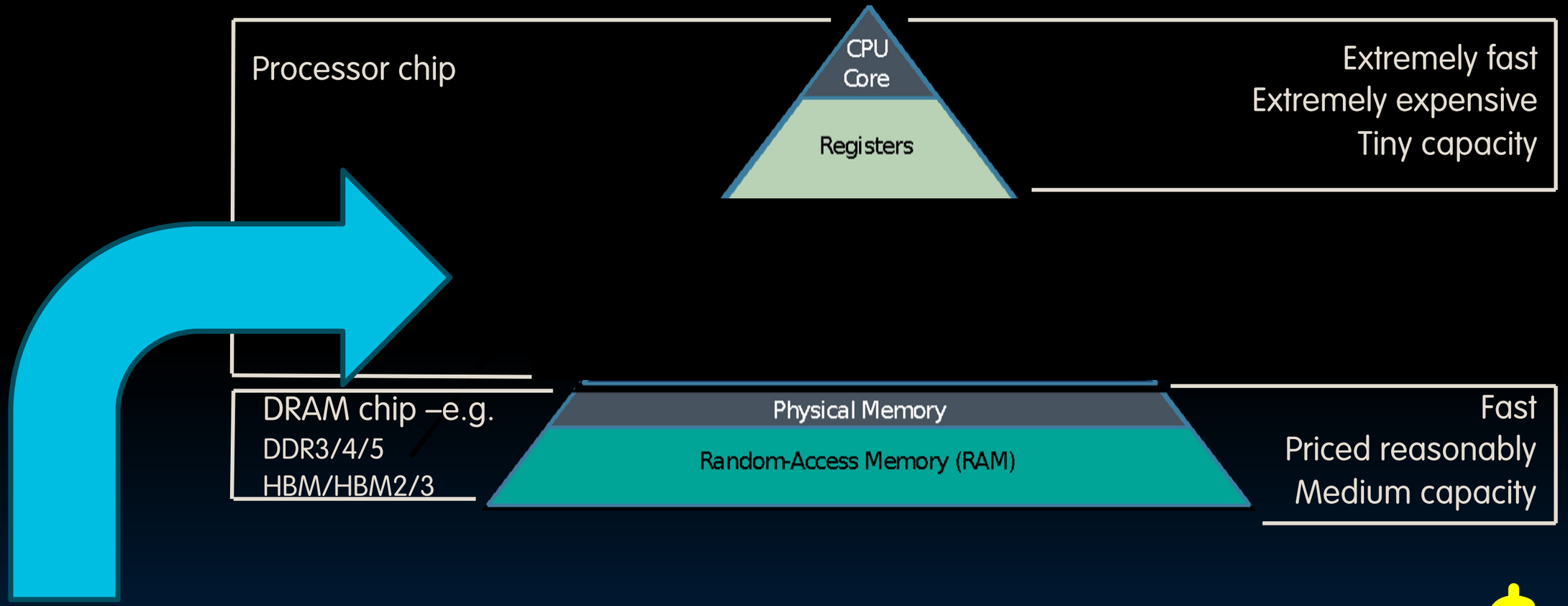
Memory Hierarchy

- Binary Prefix
- Library Analogy
- Memory Hierarchy
- Locality, Design, Management

What to do: Library Analogy

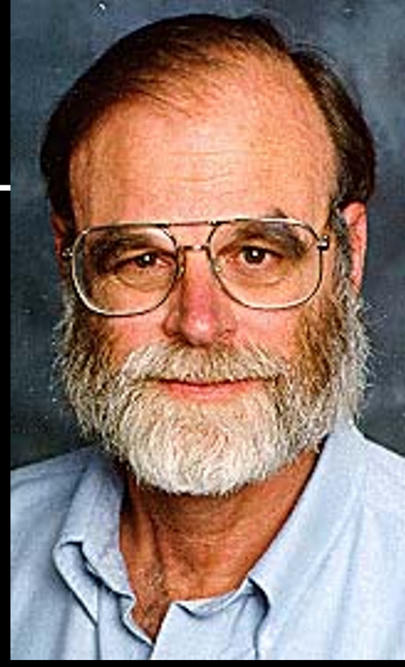
- Write a report using library books
  - E.g., works of J.D. Salinger
- Go to library, look up books, fetch from stacks, and place on desk in library
- If need more, check out, keep on desk
  - But don't return earlier books; might need them!
- You hope this collection of ~10 books on desk enough to write report, despite 10 being only 0.00001% of books in UC Berkeley libraries

Great Idea #3: Principle of Locality / Memory Hierarchy



A UC Berkeley EECS invention: "Cache" sounds like "Cash", so use shorthand $

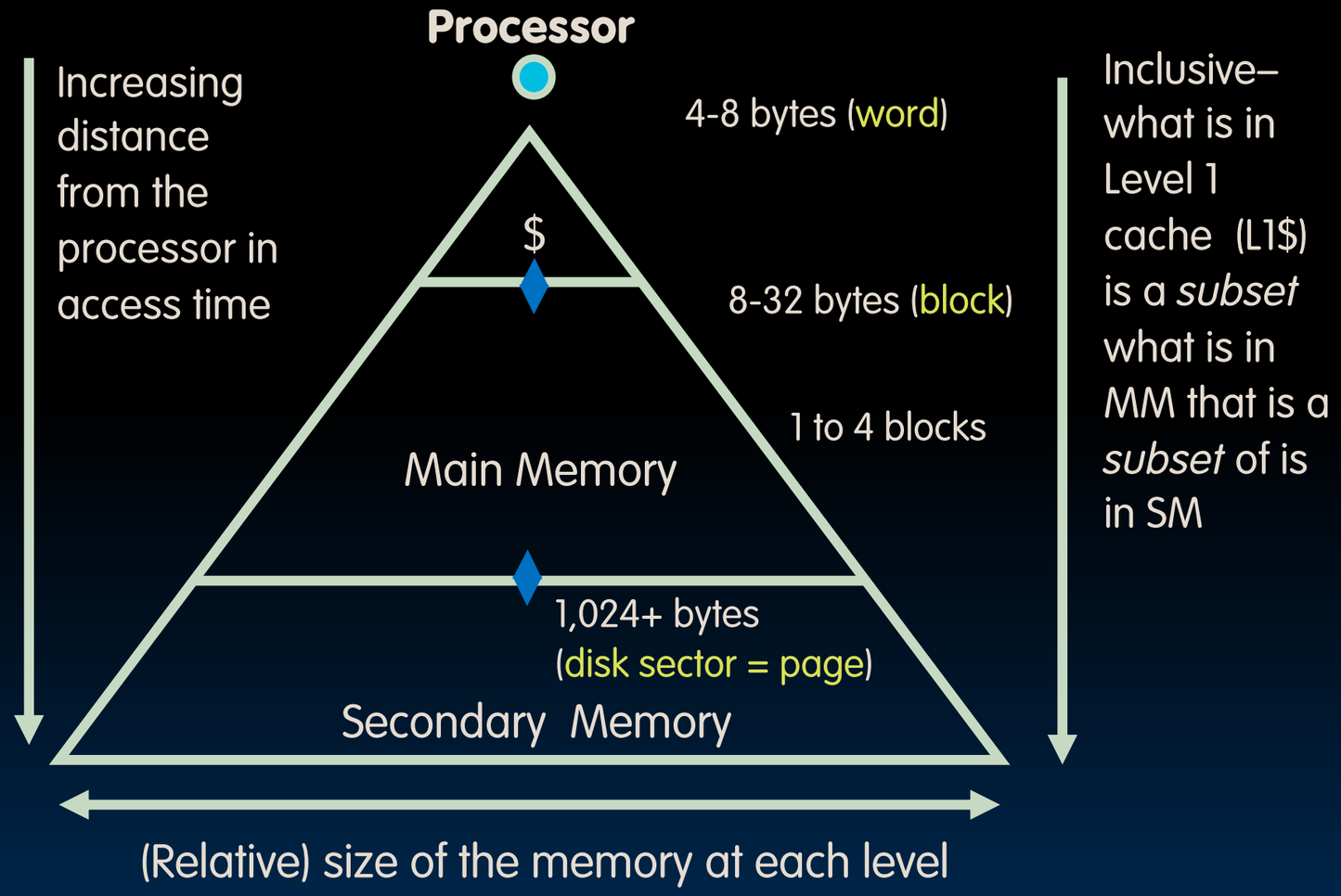
Jim Gray's Storage Latency Analogy: How Far Away is the Data?



Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969

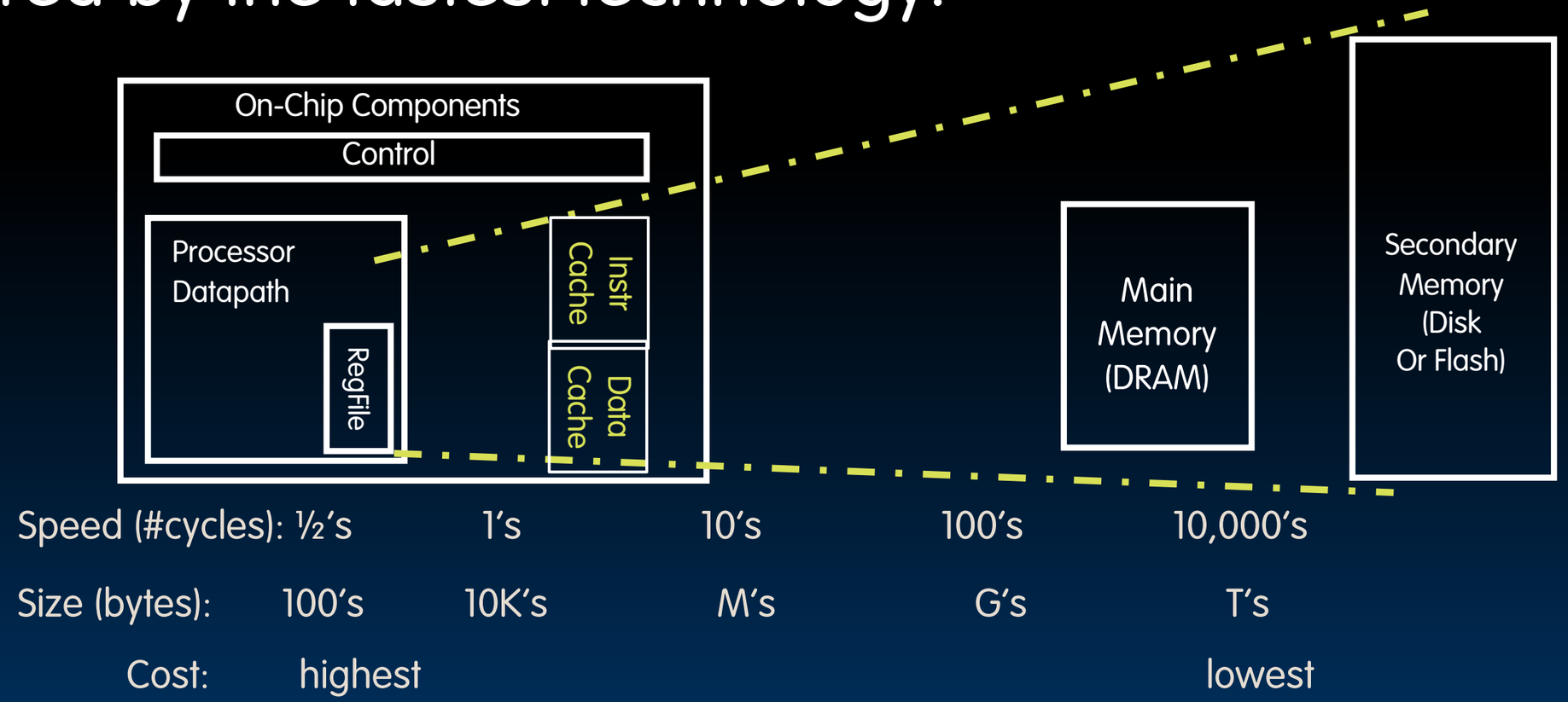
100	Memory	Sacramento	1.5 hr
2	On-chip cache	This Room	2 min
1	Registers	My Head	1 min
[ns]			

Characteristics of the Memory Hierarchy



Typical Memory Hierarchy

- The Abstraction: present processor with as much memory as is available in the cheapest technology at the speed offered by the fastest technology.



Memory Hierarchy

- If level closer to Processor, it is:
  - Smaller
  - Faster
  - More expensive
  - Subset of lower levels (contains most recently used data)
- Lowest Level (usually disk=HDD/SSD) contains all available data (does it go beyond the disk?)
- Memory Hierarchy presents the processor with the illusion of a very large & fast memory

Locality, Design, Management

- Binary Prefix
- Library Analogy
- Memory Hierarchy
- Locality, Design, Management

Memory Hierarchy Basis

- Cache contains copies of data in memory that are being used.
- Memory contains copies of data on disk that are being used.
- Caches work on the principles of temporal and spatial locality.
  - Temporal locality (locality in time): If we use it now, chances are we'll want to use it again soon.
  - Spatial locality (locality in space): If we use a piece of memory, chances are we'll use the neighboring pieces soon.

What to Do About Locality

- Temporal Locality

  If a memory location is referenced then it will tend to be referenced again soon

  ⇒ Keep most recently accessed data items closer to the processor

- Spatial Locality

  If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon

  ⇒ Move blocks consisting of contiguous words closer to the processor

Cache Design

- How do we organize cache?
- Where does each memory address map to?

  (Remember that cache is subset of memory, so multiple memory addresses map to the same cache location.)
- How do we know which elements are in cache?
- How do we quickly locate them?

How is the Hierarchy Managed?

- registers ↔ memory
  - By compiler (or assembly level programmer)
- cache ↔ main memory
  - By the cache controller hardware
- main memory ↔ disks (secondary storage)
  - By the operating system (virtual memory)
  - Virtual to physical address mapping assisted by the hardware ('translation lookaside buffer' or TLB)
  - By the programmer (files)

Also a type of cache

"And in Conclusion…"

- Caches provide an illusion to the processor that the memory is infinitely large and infinitely fast

  …the power of Abstraction!