The memory hierarchy

M1 MOSIG – Operating System Design

Renaud Lachaize

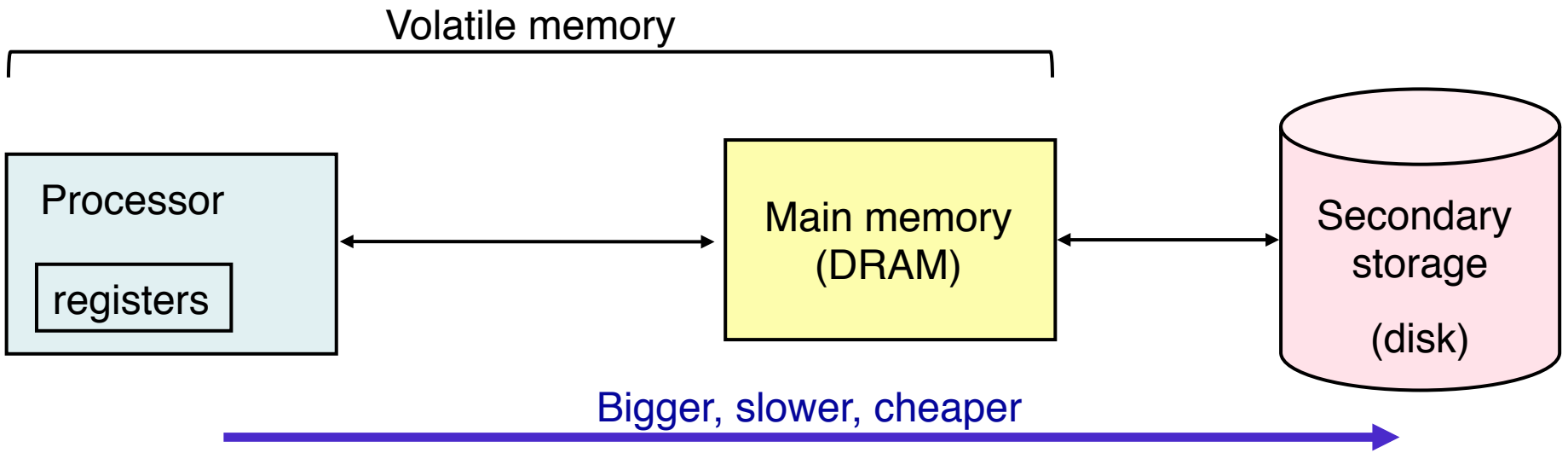
Acknowledgments

- Many ideas and slides in these lectures were inspired by or even borrowed from the work of others:

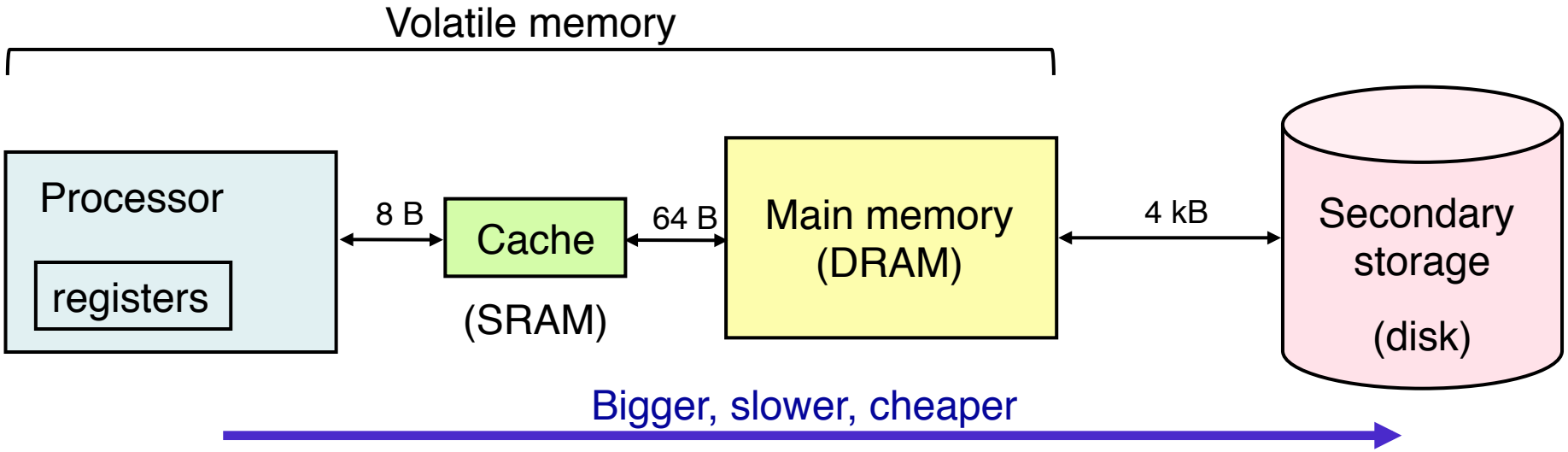
  – Arnaud Legrand, Noël De Palma, Sacha Krakowiak

  – Randall Bryant, David O'Hallaron, Gregory Kesden, Markus Püschel (Carnegie Mellon University)
    - **Textbook: Randall Bryant, David O'Hallaron. Computer Systems: A Programmer's Perspective, Prentice Hall. See chapter on "memory hierarchy".**
    - CS 15-213/18-243 classes (many slides/figures directly adapted from these classes)

Introduction



Introduction (continued)



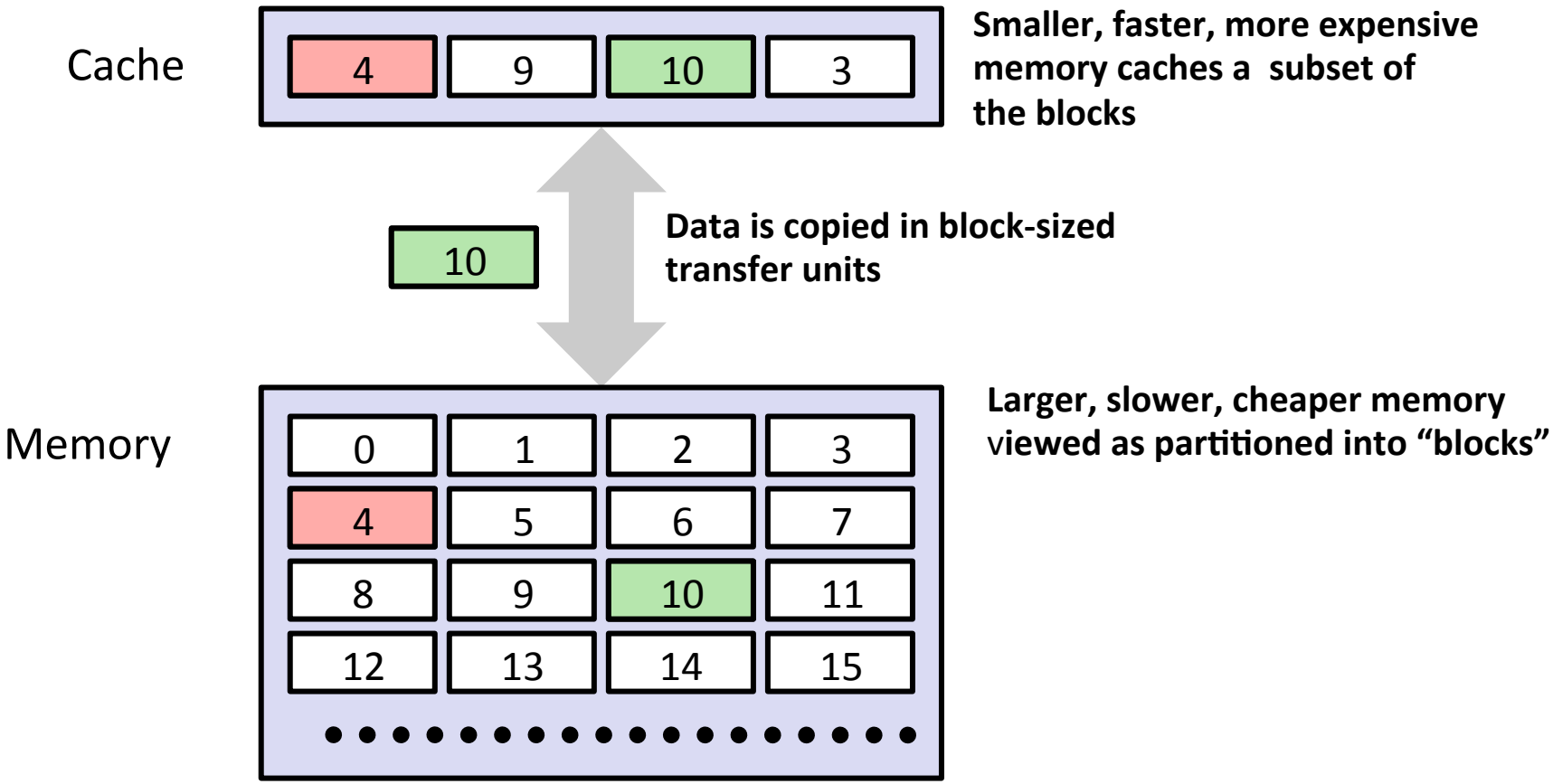
	Registers	**Cache**	**DRAM**	**Disk**
Capacity	~ 100-200 B	~ 32kB-12 MB	~ GBs	~ TBs
Access time	0-1 ns	2-10 ns	40 ns	3 ms
Cost	-	60 $/MB	0,06 $/MB	0,0003 $/MB
Size of transfer unit	4-8 B	32-64 B	4-8 kB	

Cache

- Definition: Computer memory with short access time used for the storage of frequently or recently used instructions or data

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

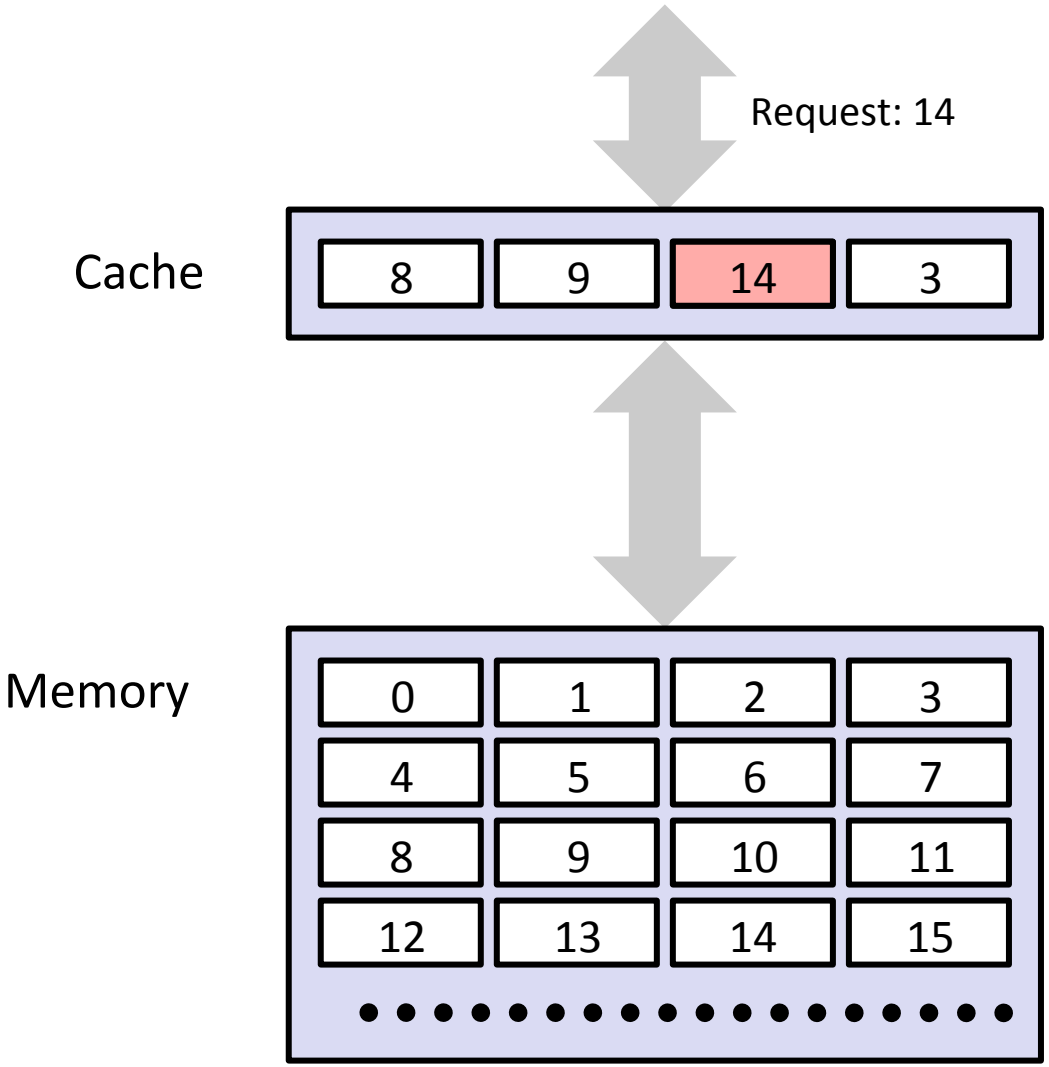
General Cache Mechanics



(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

General Cache Concepts: Hit

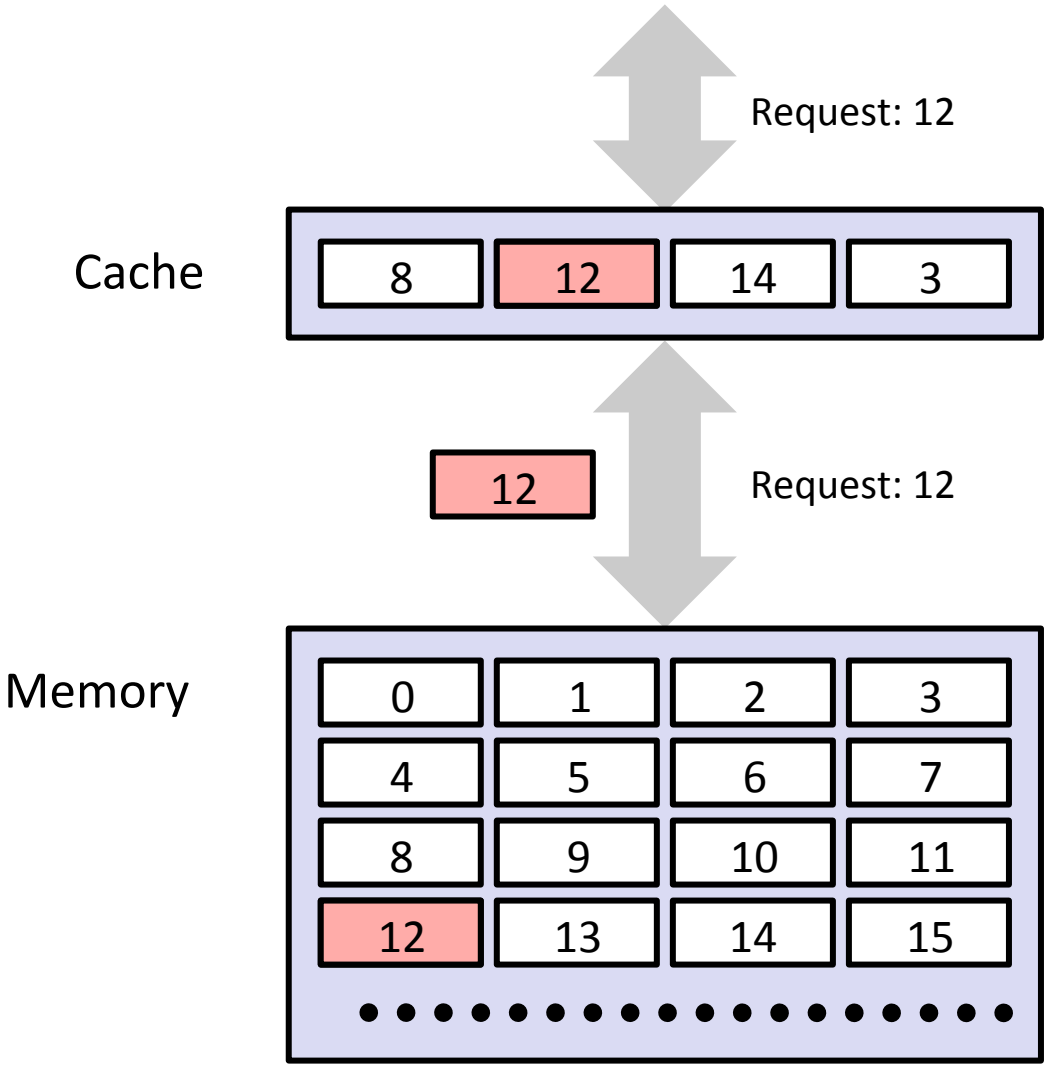


Data in block b is needed

Block b is in cache:
Hit!

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

General Cache Concepts: Miss



Data in block b is needed

Block b is not in cache:
Miss!

Block b is fetched from *memory*

Block b is stored in cache

- **Placement policy:** determines where b goes
- **Replacement policy:** determines which block gets evicted (victim)

Cache Performance Metrics

- Miss Rate
  - Fraction of memory references not found in cache (misses / accesses) = 1 – hit rate
- Hit Time
  - Time to deliver a line in the cache to the processor
    - includes time to determine whether the line is in the cache
- Miss Penalty
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory (Trend: increasing!)

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Cache Performance Metrics (continued)

- Typical numbers for a CPU cache
  - Miss Rate
    - 3-10% for L1
    - can be quite small (e.g., < 1%) for L2, depending on size, etc.
  - Hit Time
    - 1-2 clock cycle for L1
    - 5-20 clock cycles for L2
  - Miss Penalty
    - typically 50-200 cycles for main memory (Trend: increasing!)

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Lets think about those numbers

- Huge difference between a hit and a miss
  - Could be 100x, if just L1 and main memory

- Would you believe 99% hits is twice as good as 97%?
  - Consider:
    cache hit time of 1 cycle
    miss penalty of 100 cycles

  - Average access time:
    97% hits: 1 cycle + 0.03 * 100 cycles = **4 cycles**
    99% hits: 1 cycle + 0.01 * 100 cycles = **2 cycles**

- This is why “miss rate” is used instead of “hit rate”

Types of Cache Misses

- Cold (compulsory) miss
  - Occurs on first access to a block

- Conflict miss
  - Most hardware caches limit blocks to a small subset (sometimes a singleton) of the available cache slots
    - e.g., block i must be placed in slot (i mod 4)
  - Conflict misses occur when the cache is large enough, but multiple data objects all map to the same slot
    - e.g., referencing blocks 0, 8, 0, 8, ... would miss every time

- Capacity miss
  - Occurs when the set of active cache blocks (working set) is larger than the cache

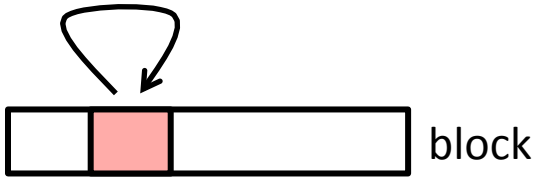
(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

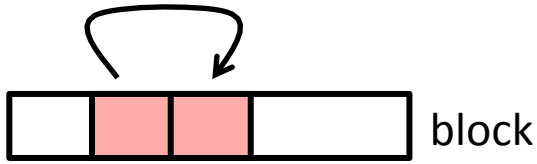
Why Caches Work

- Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently

- Temporal locality:
  - Recently referenced items are likely to be referenced again in the near future



- Spatial locality:
  - Items with nearby addresses tend to be referenced close together in time



(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Example: Locality?

```
sum = 0;
for (i = 0; i < n; i++)
        sum += a[i];
return sum;
```

- Data:
  - Temporal: **`sum`** referenced in each iteration
  - Spatial: array **`a[]`** accessed in stride-1 pattern
- Instructions:
  - Temporal: cycle through loop repeatedly
  - Spatial: reference instructions in sequence

- Being able to assess the locality of code is a crucial skill for a programmer

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Locality Example #1

Does this function have good locality with respect to array `a`?

```
int sum_array_v1(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Locality Example #2

Does this function have good locality with respect to array `a`?

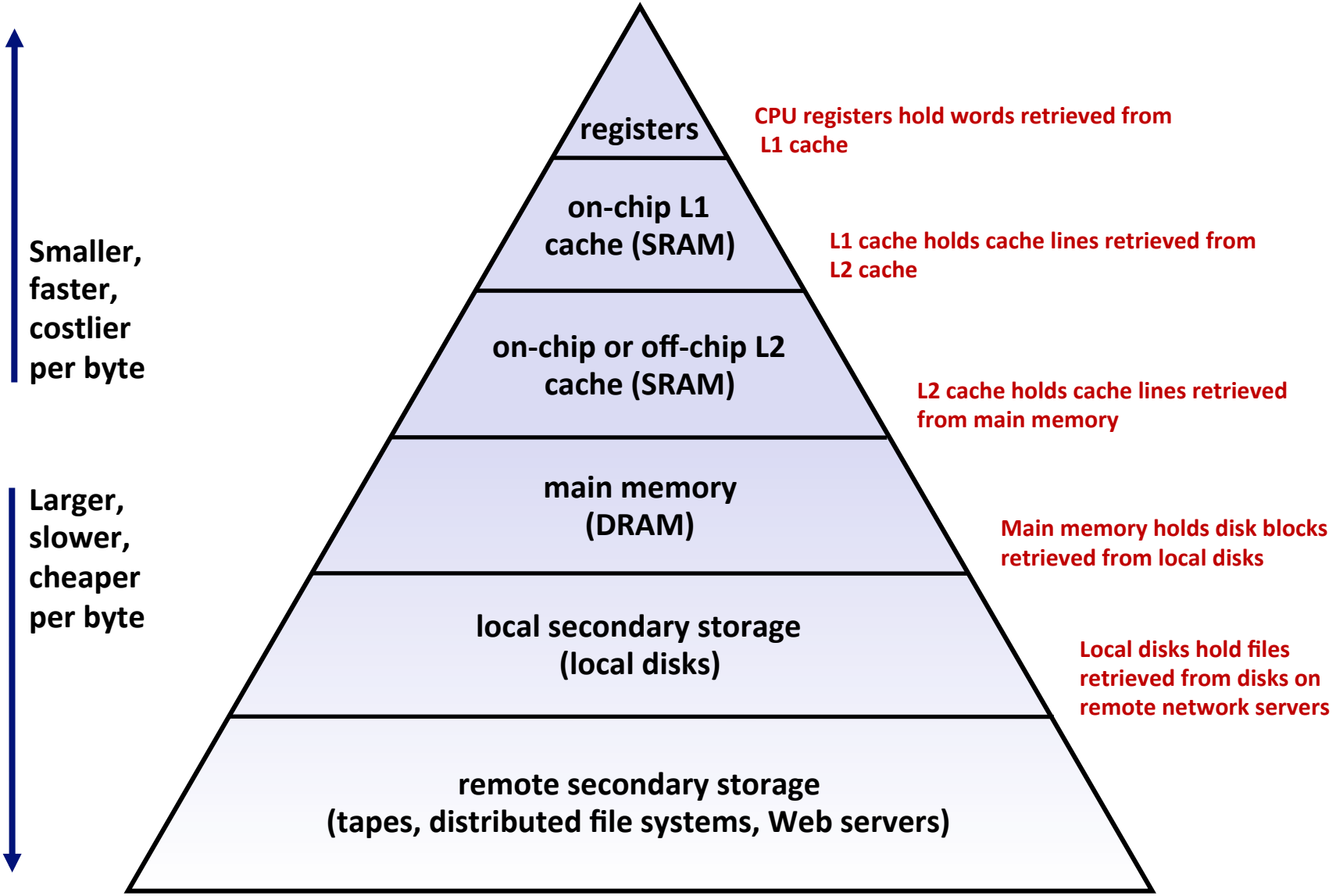
```
int sum_array_v2(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Memory Hierarchies

- Some fundamental and enduring properties of hardware and software systems:
  - Faster storage technologies almost always cost more per byte and have lower capacity
  - The gaps between memory technology speeds are widening
    - True of registers ↔ DRAM, DRAM ↔ disk, etc.
  - Well-written programs tend to exhibit good locality
- These properties complement each other beautifully
- They suggest an approach for organizing memory and storage systems known as a memory hierarchy

The memory hierarchy



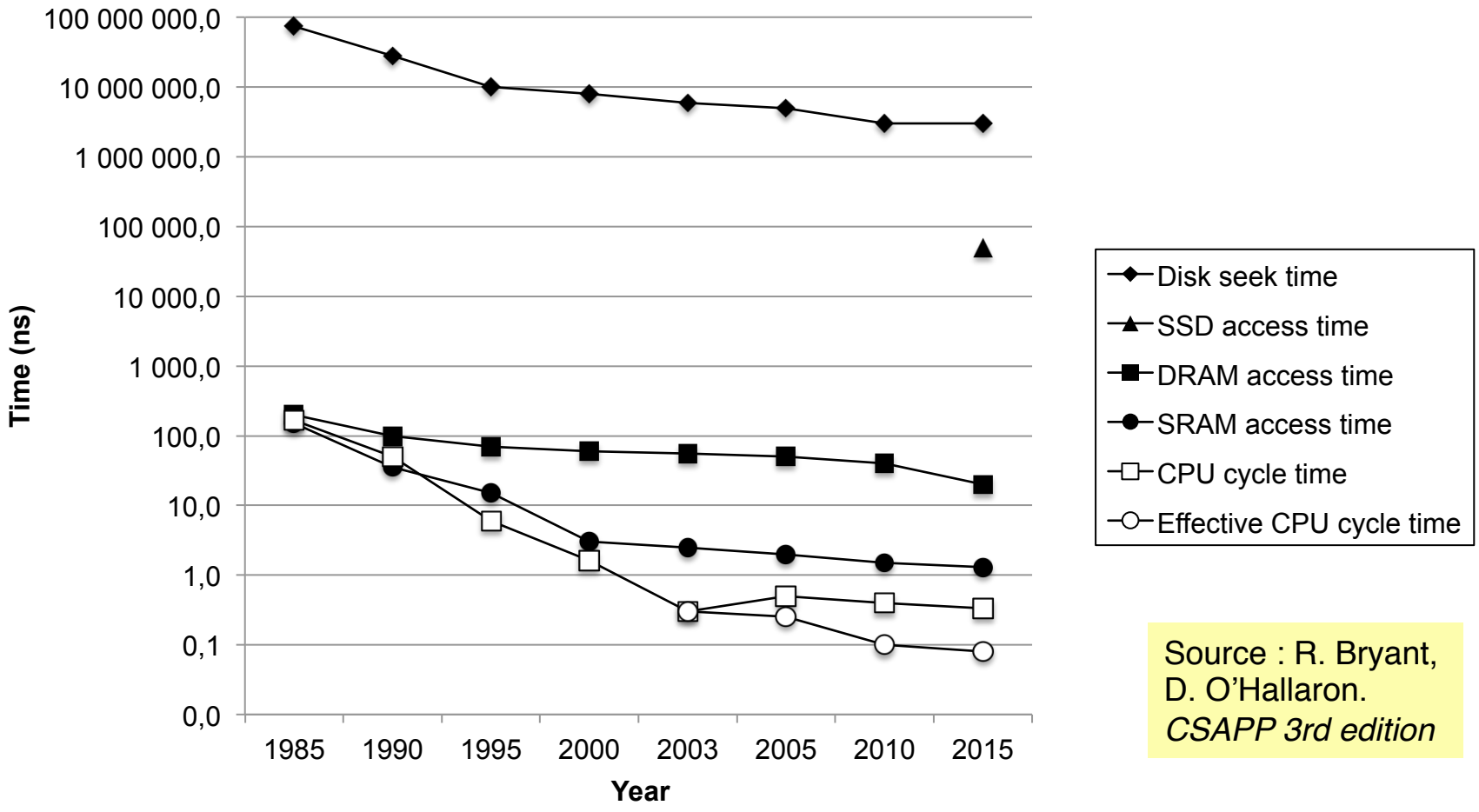
(adapted from the following source: Carnegie Mellon University – 15-213/18-243 class)

Examples of caches in the hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	8-byte words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main memory	100	Hardware+OS
Buffer cache	Parts of files	Main memory	100	OS
Network buffer cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Source : R. Bryant, D. O'Hallaron. *CSAPP 2nd edition*

The memory hierarchy - Trends



Source : R. Bryant, D. O'Hallaron. *CSAPP 3rd edition*

The memory hierarchy – An analogy

Memory layer	Access latency	Analogy 1	Analogy 2
CPU register	1 cycle ~0.3 ns	1 s	Your brain
L1 cache	0.9 ns	3 s	This room
L2 cache	2.8 ns	9 s	This floor
L3 cache	12.9 ns	43 s	This building
Main memory	120 ns	6 minutes	This campus
Solid state disk (SSD)	50-150 µs	2-6 days	
Hard disk drive (HDD)	1-10 ms	1-12 months	
Main memory of a remote server (over the Internet)	~100 ms	1 century	
Optical storage (DVDs) and tapes	seconds	Several millennia	

(Inspired by presentations by Jim Gray, Brendan Gregg and Jeff Antwood.
http://blog.codinghorror.com/the-infinite-space-between-words/)

Summary

- Computers are built with a **memory hierarchy**
  - Registers, multiple levels of cache, main memory
  - Data is brought in bulk (cache line) from a lower level (slower, cheaper, bigger) to a higher level
  - When the cache is full, we need a policy to decide what should stay in cache and what should be replaced
  - Hopefully the data brought in a cache line is reused soon
    - **Temporal locality**
    - **Spatial locality**
  - Programs must be aware of the memory hierarchy (at least to some extent)