

The memory hierarchy

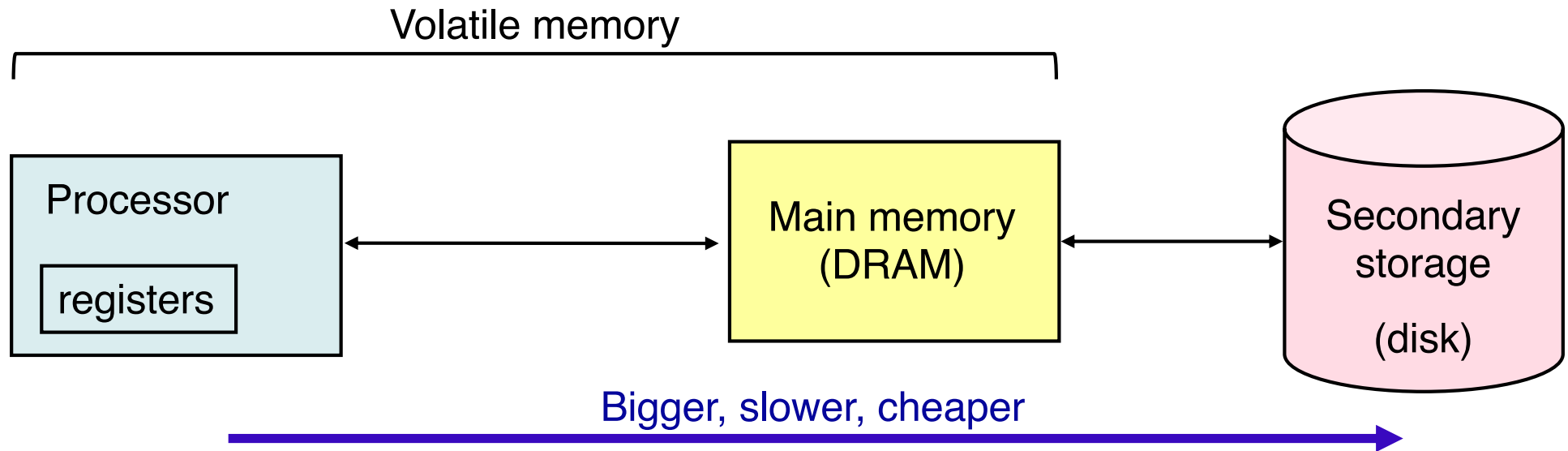
M1 MOSIG – Operating System Design

Renaud Lachaize

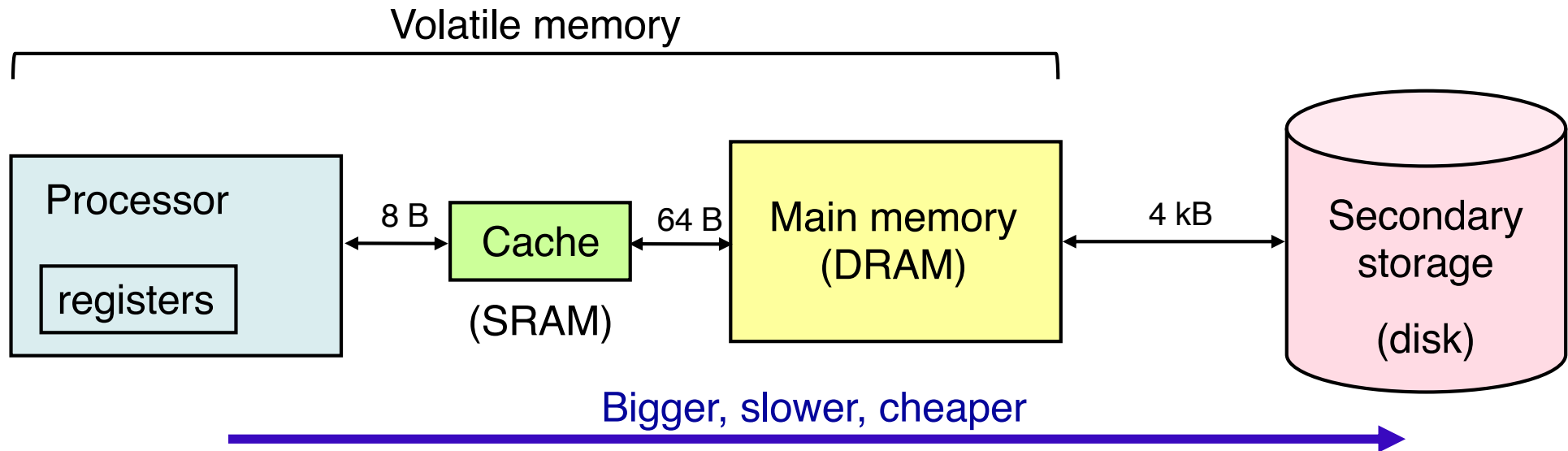
Acknowledgments

- 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)

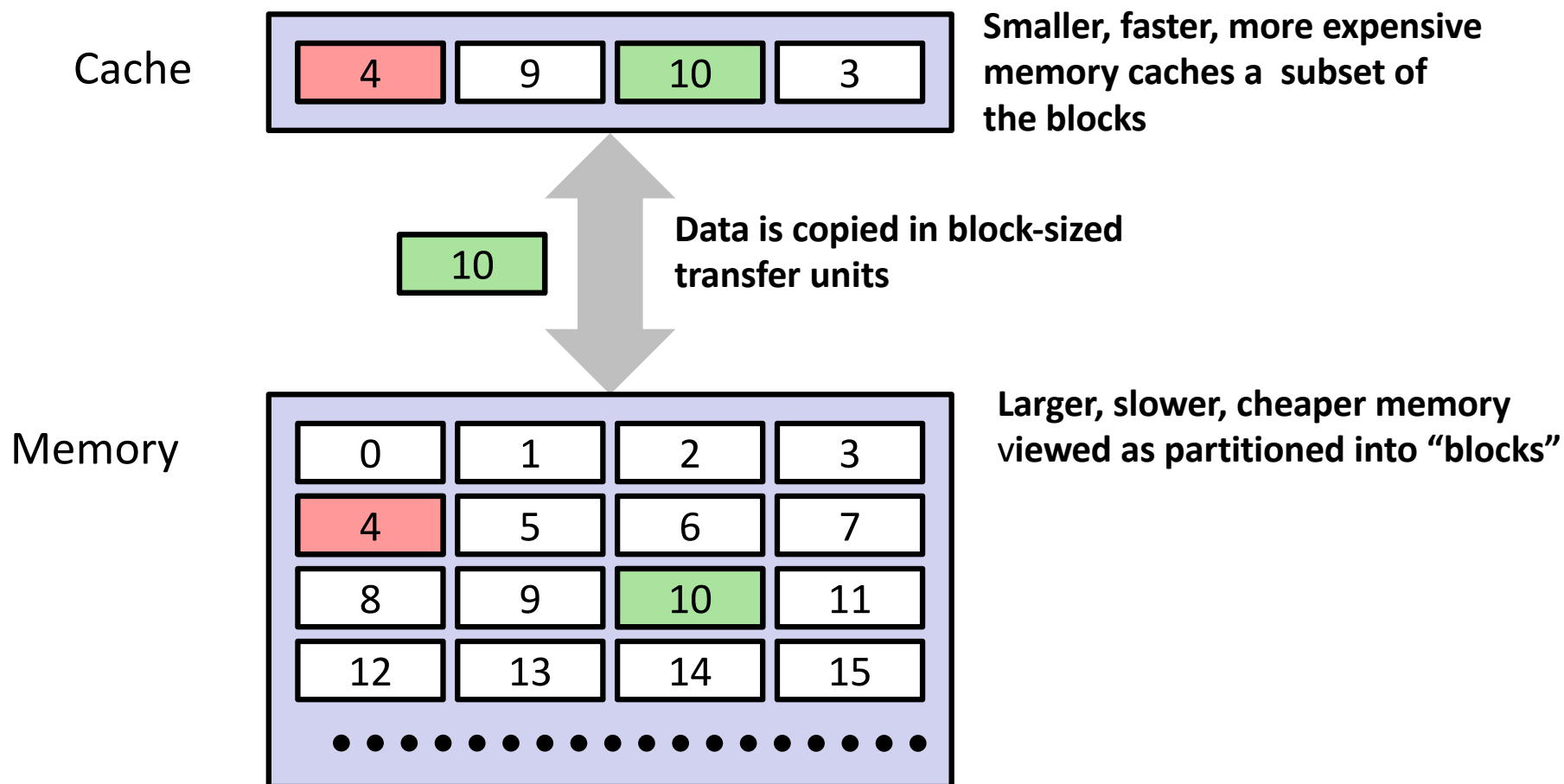


	Registers	Cache	DRAM	Disk
Capacity	~ 100-200 B	~ 32kB-12 MB	~ GBs	~ TBs
Access time	0-1 ns	2-10 ns	40 ns	3 ms
Cost	-	60 \$/MB	0,06 \$/MB	0,0003 \$/MB
Size of transfer unit	4-8 Bytes	32-64 B	4-8 kB	

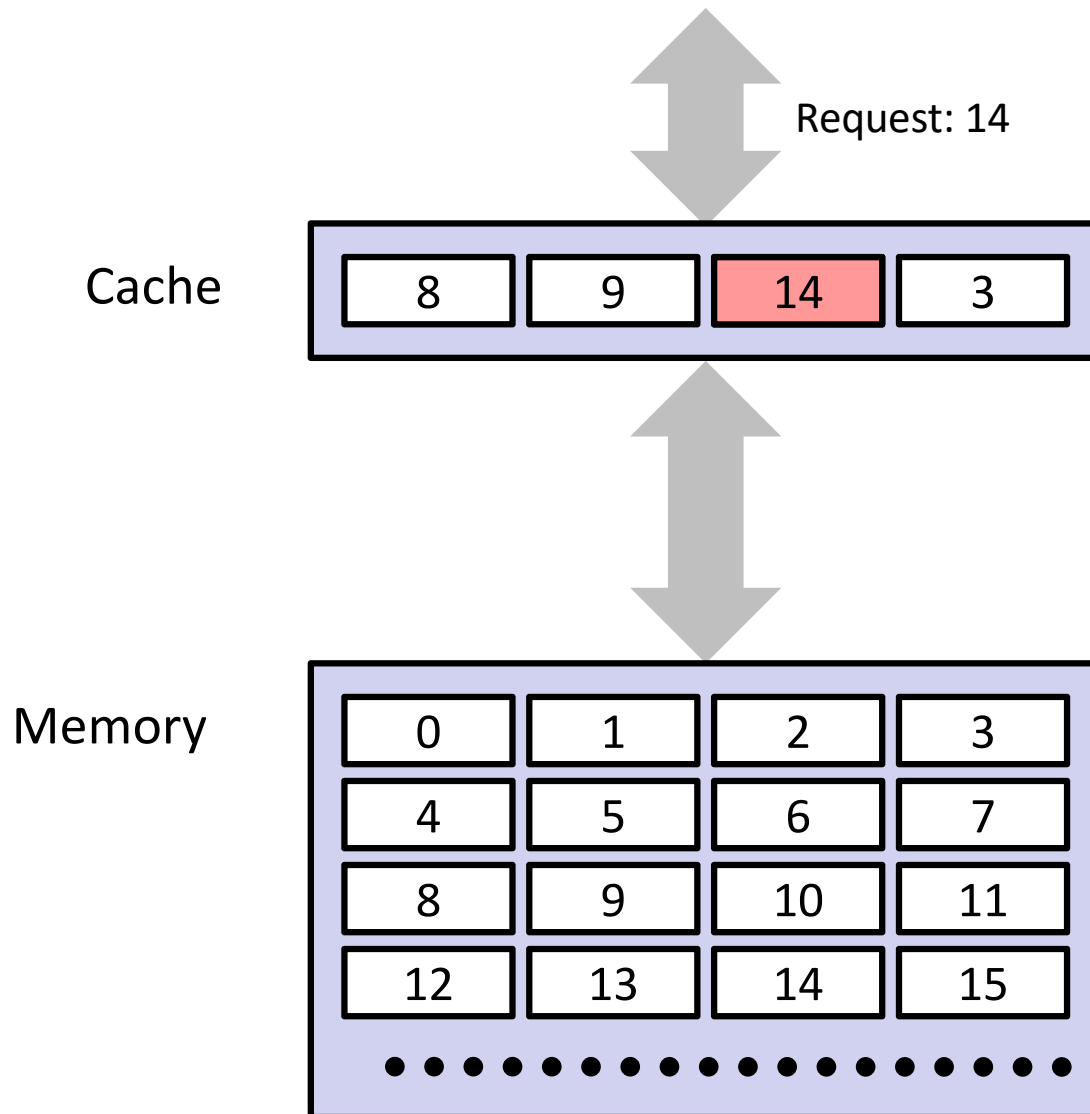
Cache

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

General Cache Mechanics



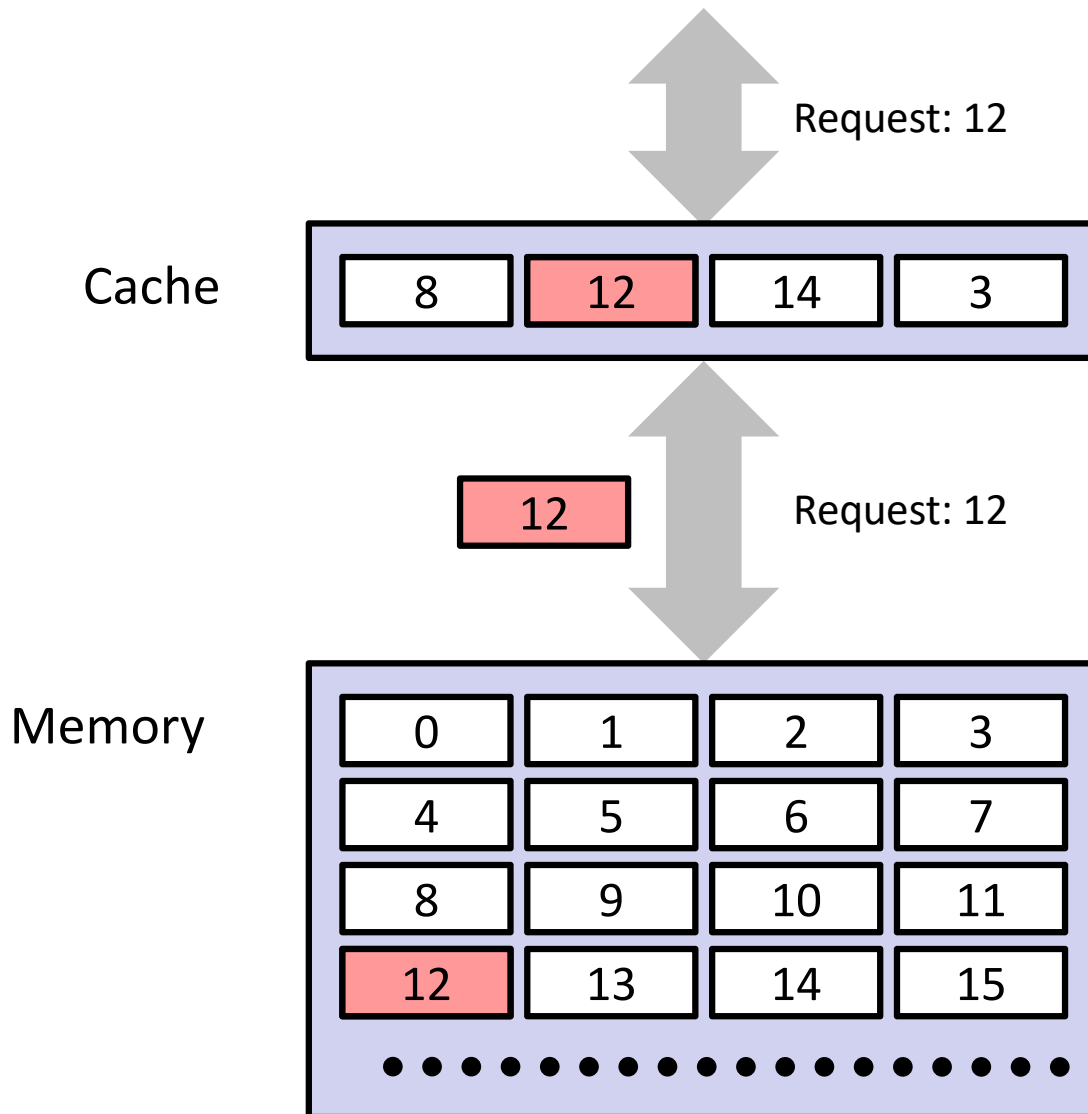
General Cache Concepts: Hit



Data in block b is needed

Block b is in cache:
Hit!

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)
= $1 - \text{hit rate}$
- **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!)

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!)

Lets think about those numbers

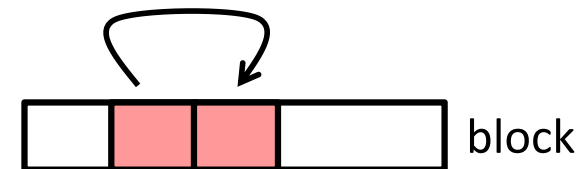
- Huge difference between a hit and a miss
 - Could be 100x, if just L1 and main memory
- Would you believe 99% hits is twice as good as 97%?
 - Consider:
cache hit time of 1 cycle
miss penalty of 100 cycles
 - Average access time:
97% hits: $1 \text{ cycle} + 0.03 * 100 \text{ cycles} = \mathbf{4 \text{ cycles}}$
99% hits: $1 \text{ cycle} + 0.01 * 100 \text{ cycles} = \mathbf{2 \text{ cycles}}$
- 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 $(i \bmod 4)$
 - 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

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



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

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;
}
```

Locality Example #2

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

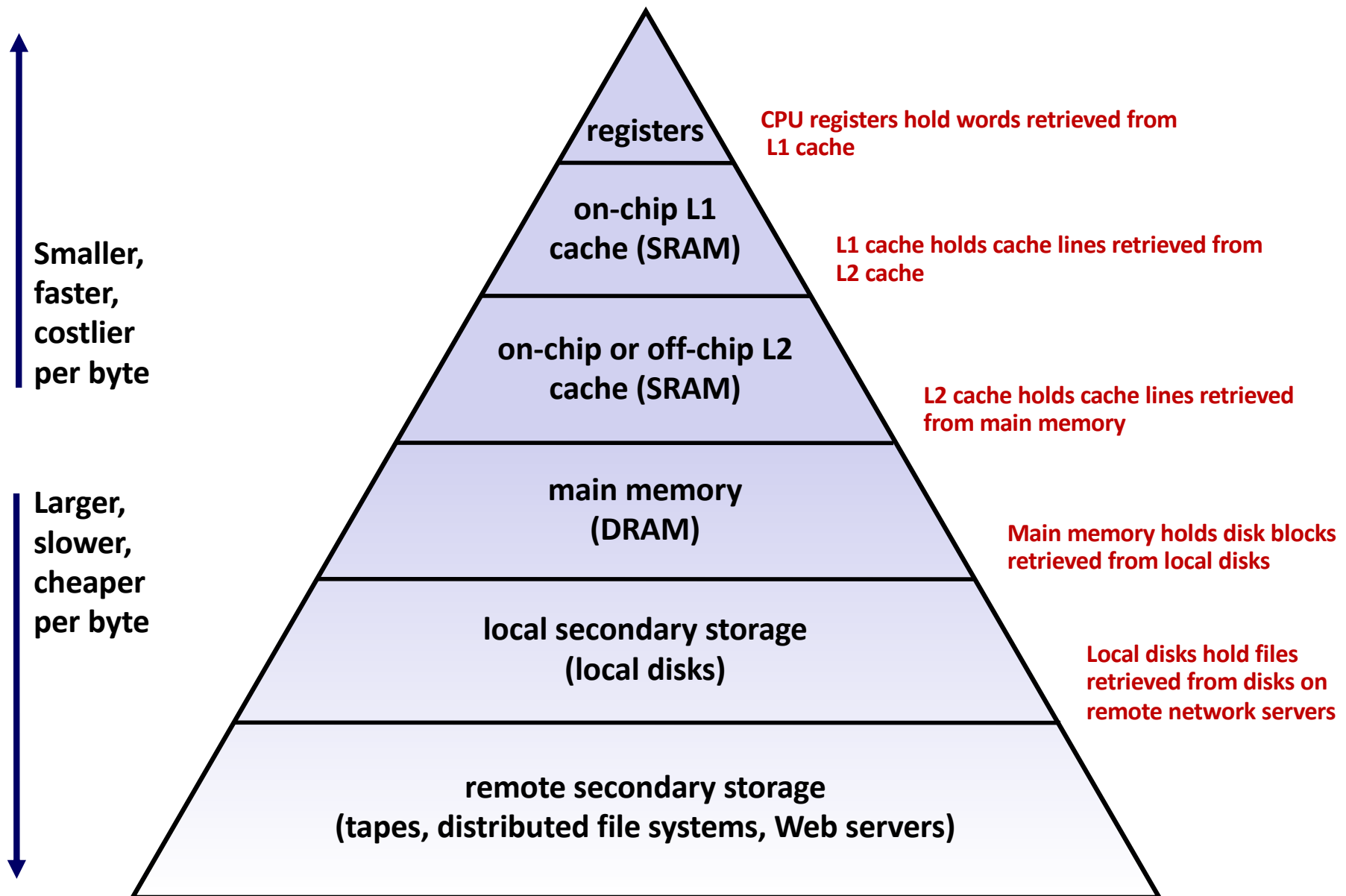
```
int sum_array_v2(int a[M][N])
{
    int i, j, sum = 0;

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


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 \leftrightarrow DRAM, DRAM \leftrightarrow 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

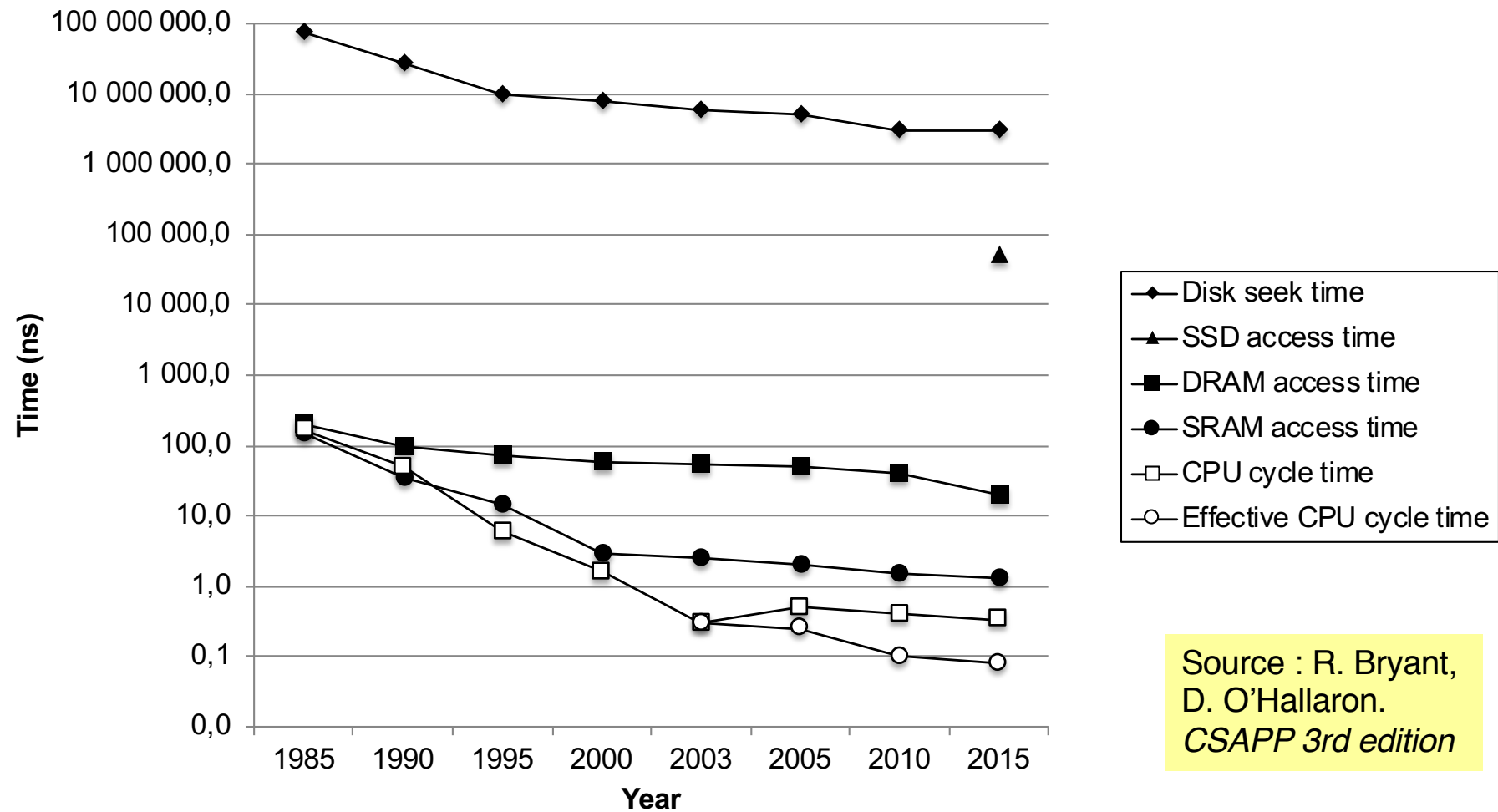


Examples of caches in the hierarchy

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



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 μ s	2-6 days	<div> The distance/analogy depends on the vehicle that you consider ... </div>
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	

The memory hierarchy – yet another summary (1/2)

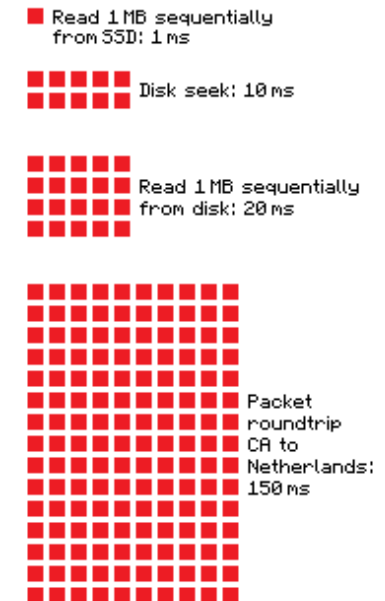
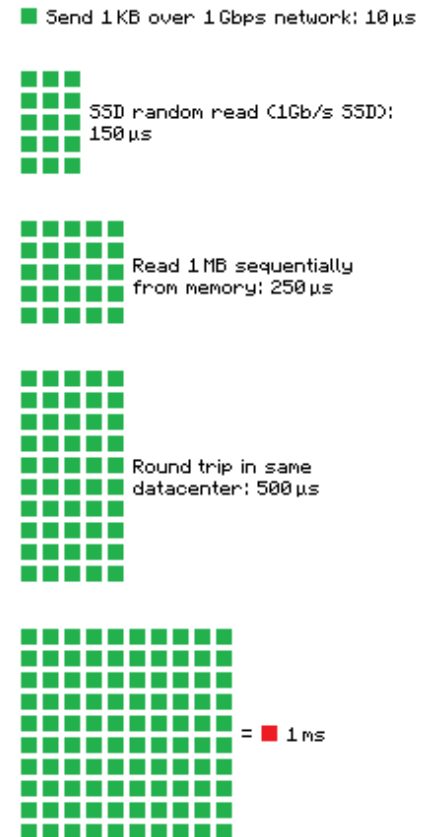
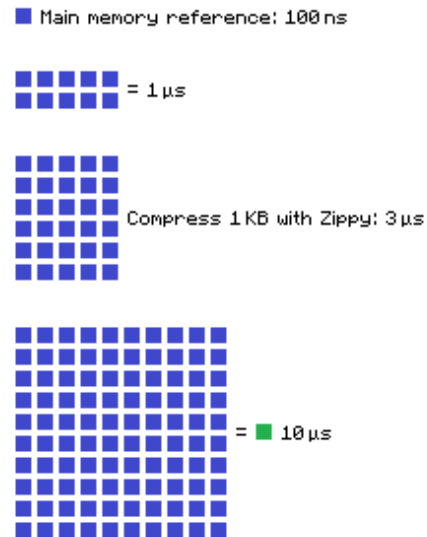
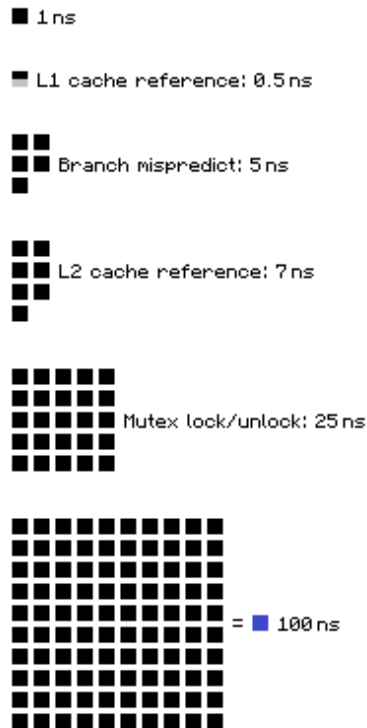
L1 cache reference	0.5	ns				
Branch mispredict	5	ns				
L2 cache reference	7	ns				14x L1 cache
Mutex lock/unlock	25	ns				
Main memory reference	100	ns				20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	3,000	ns	3	us		
Send 1K bytes over 1 Gbps network	10,000	ns	10	us		
Read 4K randomly from SSD*	150,000	ns	150	us		~1GB/sec SSD
Read 1 MB sequentially from memory	250,000	ns	250	us		
Round trip within same datacenter	500,000	ns	500	us		
Read 1 MB sequentially from SSD*	1,000,000	ns	1,000	us	1 ms	~1GB/sec SSD, 4X memory
Disk seek	10,000,000	ns	10,000	us	10 ms	20x datacenter roundtrip
Read 1 MB sequentially from disk	20,000,000	ns	20,000	us	20 ms	80x memory, 20X SSD
Send packet CA->Netherlands->CA	150,000,000	ns	150,000	us	150 ms	

- Sources :

- <https://gist.github.com/jboner/2841832>
- <http://i.imgur.com/k0t1e.png>

The memory hierarchy – yet another summary (2/2)

Latency Numbers Every Programmer Should Know



Source: <https://gist.github.com/2841832>

- Sources :
 - <https://gist.github.com/jboner/2841832>
 - <http://i.imgur.com/k0t1e.png>

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)

The future memory hierarchy? (1/3)

Emerging technology: Non Volatile Memory (NVM)

- Also known as “Storage Class Memory” (SCM)
- Like traditional RAM:
 - Fast
 - Directly accessible by the CPUs, at byte-level granularity
- Like disks:
 - Cheap cost per byte, high storage density
 - No energy consumption when idle
 - **Persistent**

The future memory hierarchy? (2/3)

NVM: Various physical technologies

Technology	Read latency	Write latency	Density	Cost
DRAM (baseline)	15 ns	15 ns	Low	\$\$\$\$
PCM	50 ns	500 ns	Medium	\$\$
ReRAM	10 ns	50 ns	High	\$\$\$\$
STT-MRAM	10 ns	50 ns	Low	\$\$\$
CNT	< 50 ns	< 50 ns	High	\$\$\$

(source: M. Seltzer et al. An NVM Carol. ICDE 2018.)

The future memory hierarchy? (3/3)

- NVM technology expected to become mainstream soon ...
 - (But we have been hearing this since ~2010 ...)
- What will be the impact of NVM on:
 - The hardware memory hierarchy?
 - The software stack?

More generally:

System events and their latencies

Event	Latency Range
Nanosecond events	
Register access [Lev09]	0.4ns
L1 cache hit [Lev09]	1ns
Branch mispredict [Lev09]	3ns
L2 cache hit [Lev09]	4ns
L3 cache hit [Lev09]	12ns-40ns
DRAM access [Lev09]	100ns
Switch Layer 1 [Exa18a]	2.4ns-4.6ns
Switch Layer 2 (cut-through) [Pao10; Neta]	330ns-500ns
PCIe Interconnect [NAZ ⁺ 18]	400ns-900ns
1m vacuum	3.3ns
1m copper	4.3ns
1m fibre	4.9ns
Microsecond events	
NIC [Exa18b]	880ns-1.2 μ s
Switch Layer 2 (store-and-forward) [Netb]	<4 μ s
Data centre network propagation delay [MLD ⁺ 15]	1 μ s-10 μ s
Intel Optane memory access [Int18e]	<10 μ s
NVMe SSD I/O [Int18d]	18 μ s-77 μ s
SATA SSD I/O [Int18c]	36 μ s-37 μ s
Millisecond events	
HDD I/O [AA15]	6ms-13.2ms
London-San Francisco RTT	152ms

(Source: D. A. Popescu. Latency-driven performance in data centres. 2019.)