Chapter 3

Memory Hierarchy

- Technology Trend and Memory Hierarchy
- Four Questions for Cache Designers
- Memory System Performance
  - Improve Cache Performance
- Storage Systems

Memory

- Register
- Cache
- Memory
- Storage

- Mechanical memory
  - Acoustic wave/torque wave delay line memory
  - Magnetic Drum Memory
  - Magnetic core memory
- Electronic memory
  - SRAM
  - DRAM
    - SDRAM
  - Flash
  - ROM
    - PROM
    - EPROM
- Optical memory

Memory

- Register
- Cache
- Memory
- Storage

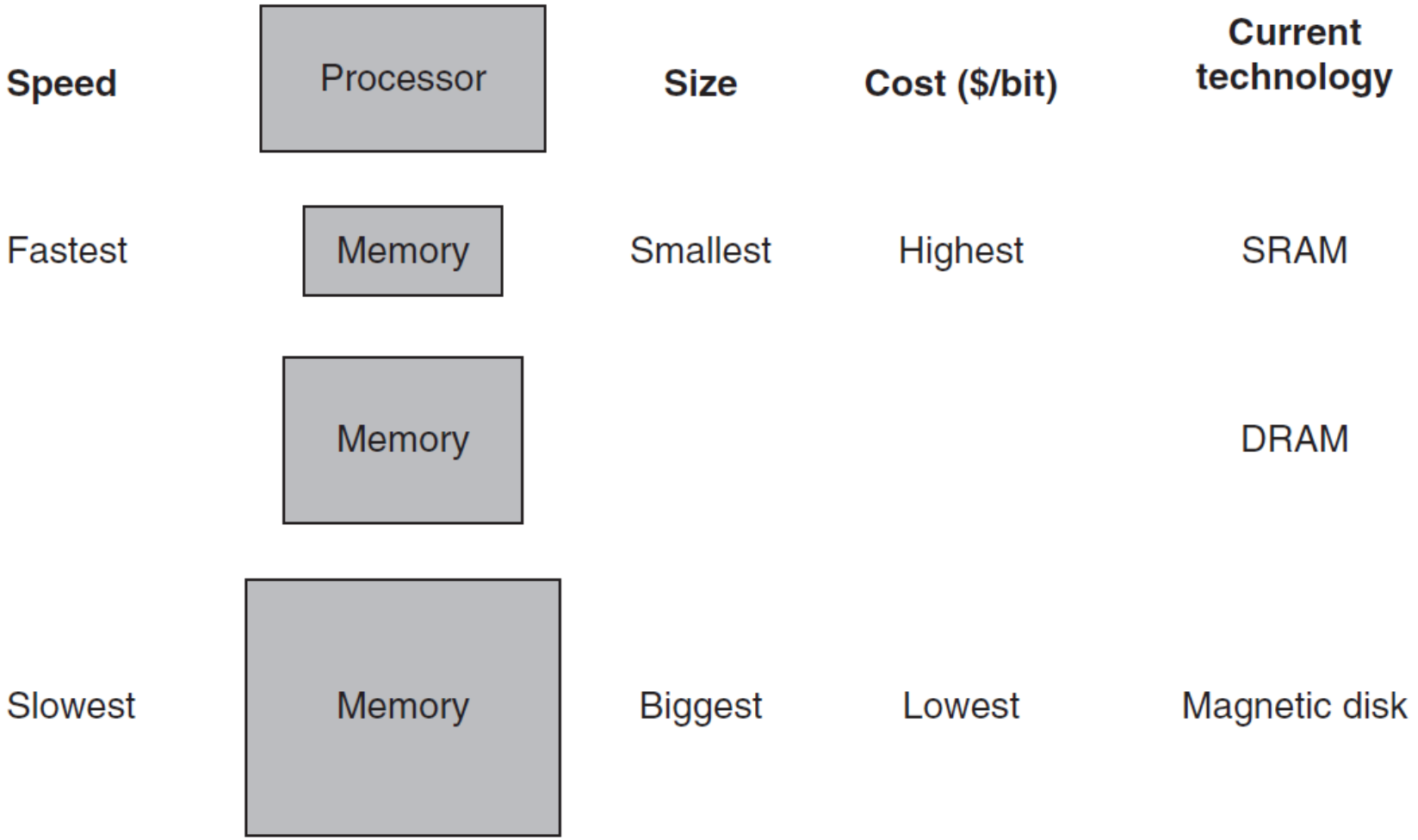
- Mechanical memory
  - Acoustic wave/torque wave delay line memory
  - Magnetic Drum Memory
  - Magnetic core memory
- Electronic memory
  - SRAM
  - DRAM
    - SDRAM
  - Flash
  - ROM
    - PROM
    - EPROM
- Optical memory

Temporal locality (locality in time): if an item is referenced, it will tend to be referenced again soon. If you recently brought a book to your desk to look at, you will probably need to look at it again soon.

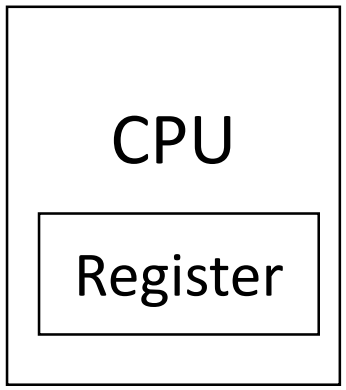
Spatial locality (locality in space): if an item is referenced, items whose addresses are close by will tend to be referenced soon.

Memory?

- Load R2, 0(R1)
- Store R2, 0(R1)

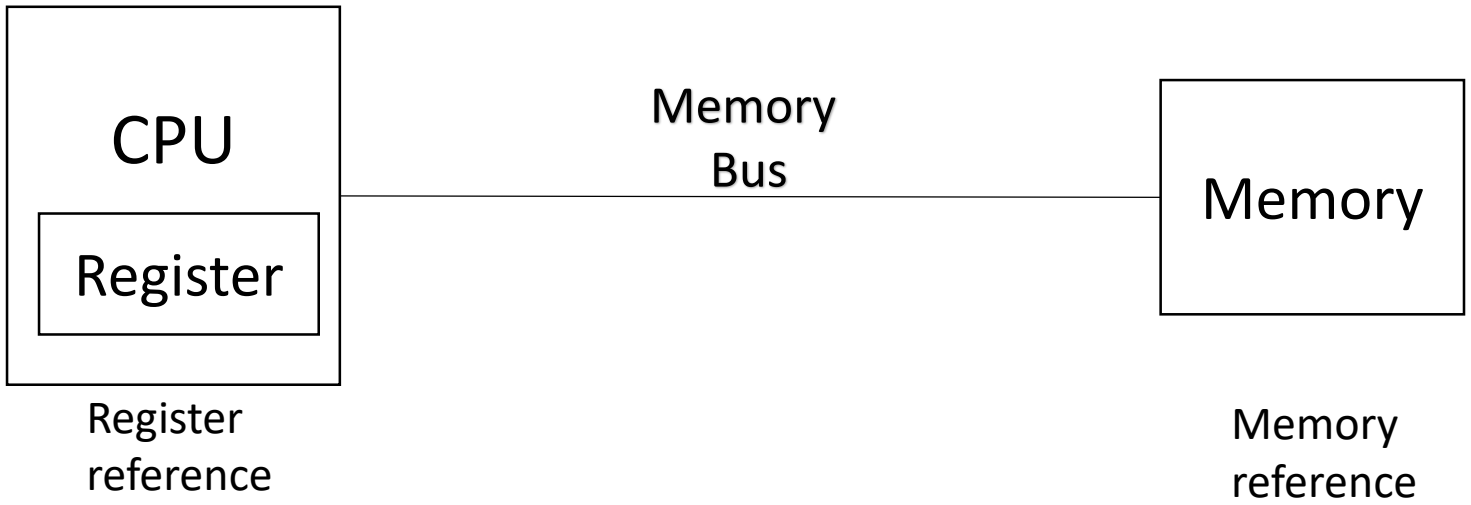

Speed
Processor
Size
Cost ($/bit)
Current technology
Fastest
Memory
Smallest
Highest
SRAM
Memory
DRAM
Slowest
Memory
Biggest
Lowest
Magnetic disk

Data processing & temporary storage

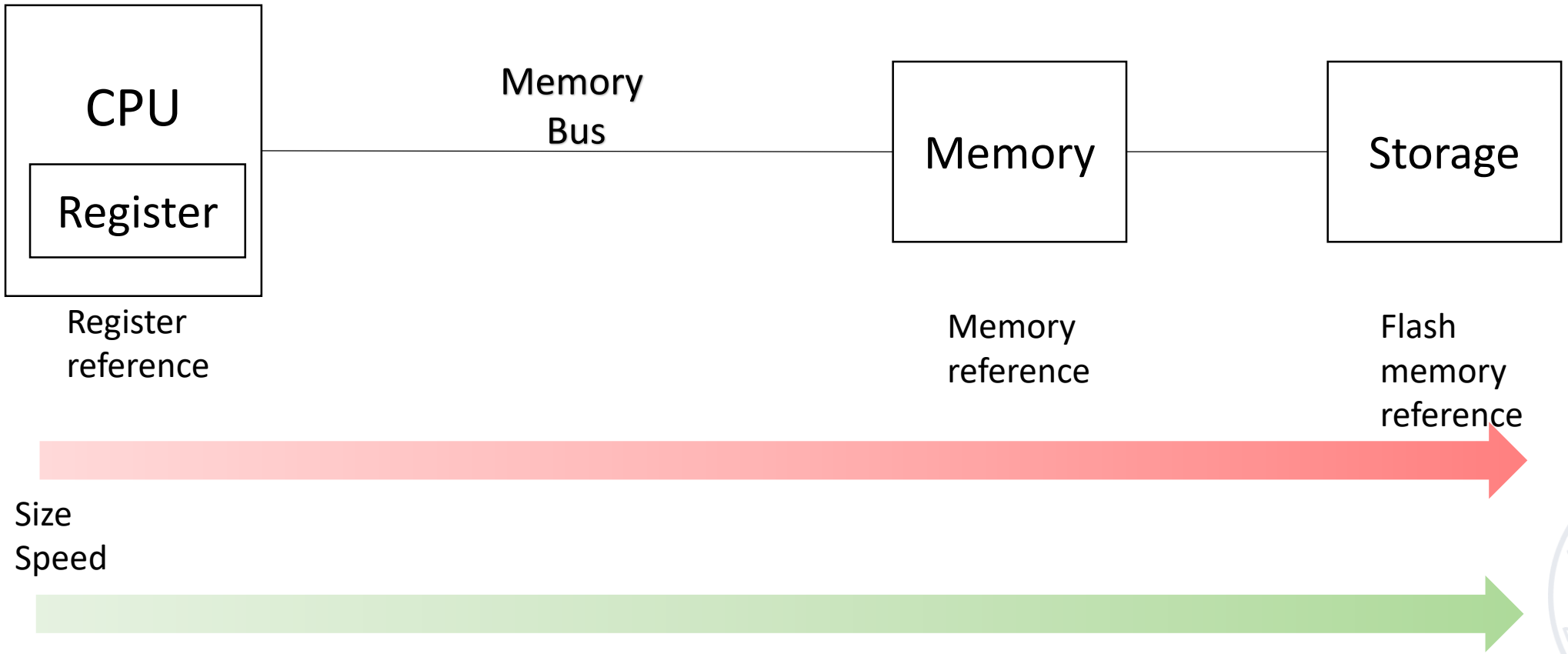


Register reference

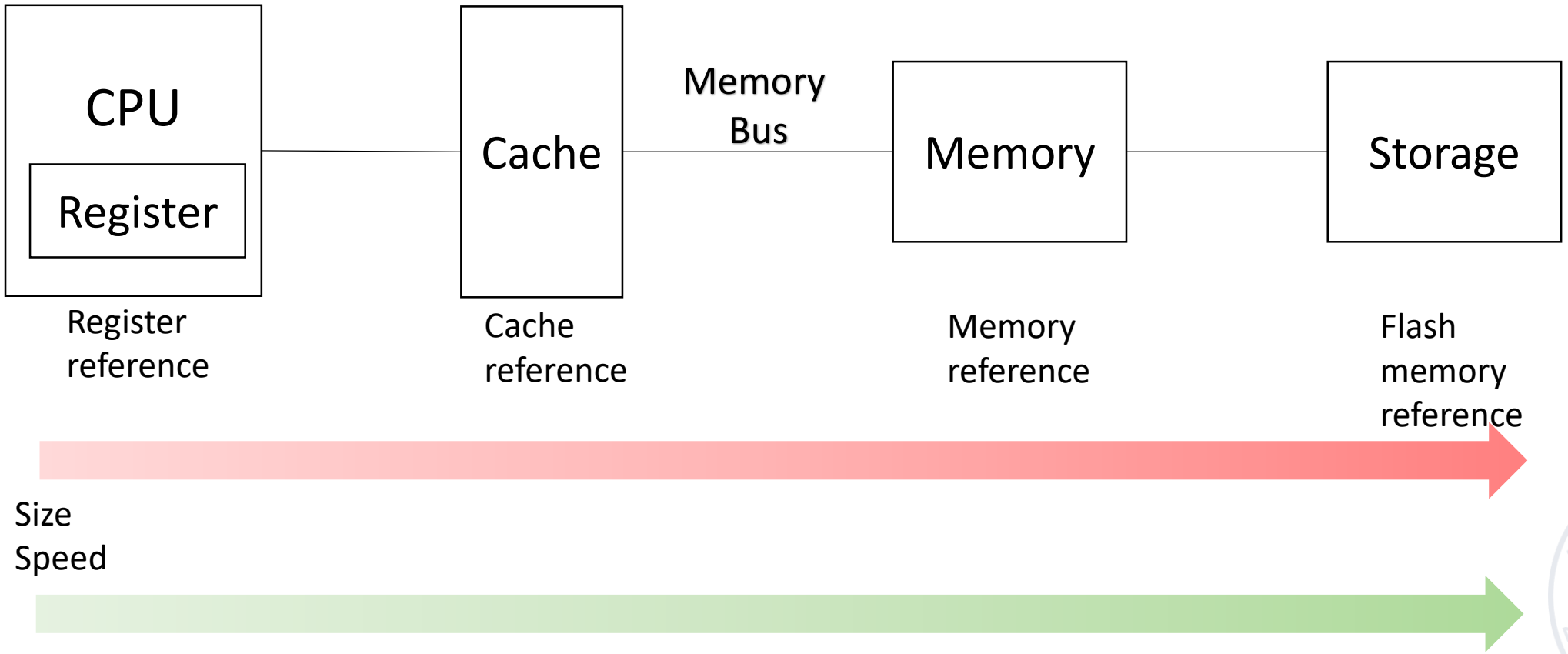
Temporary storage



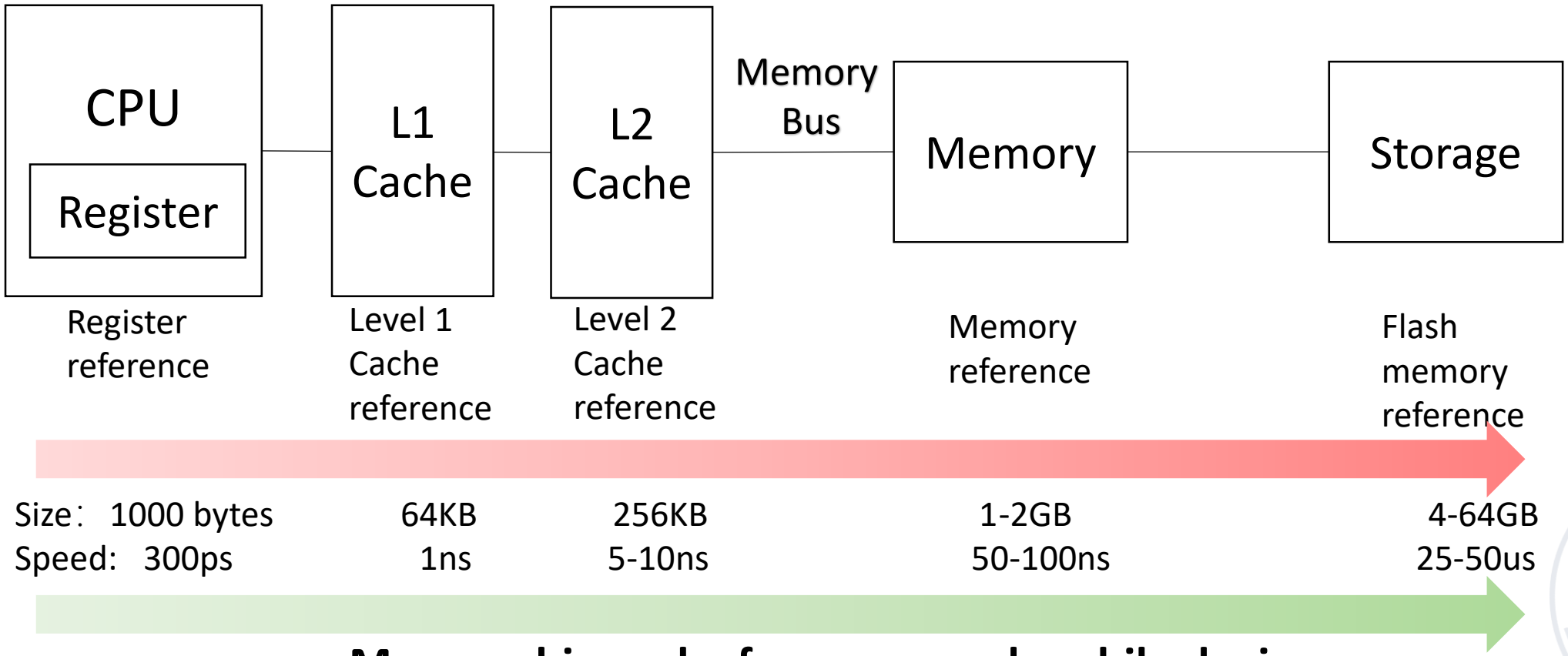
Faster temporary storage



Faster temporary storage



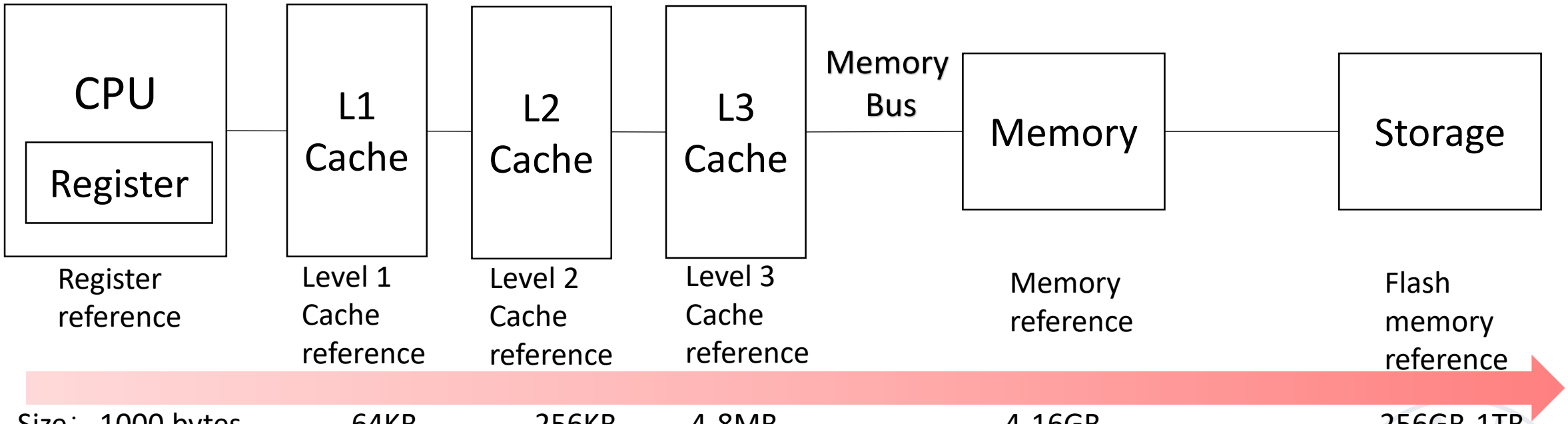
Memory Hierarchy



Memory hierarchy for a personal mobile device

*1000 picoseconds = 1 nanosecond = 10^{-6} millisecond

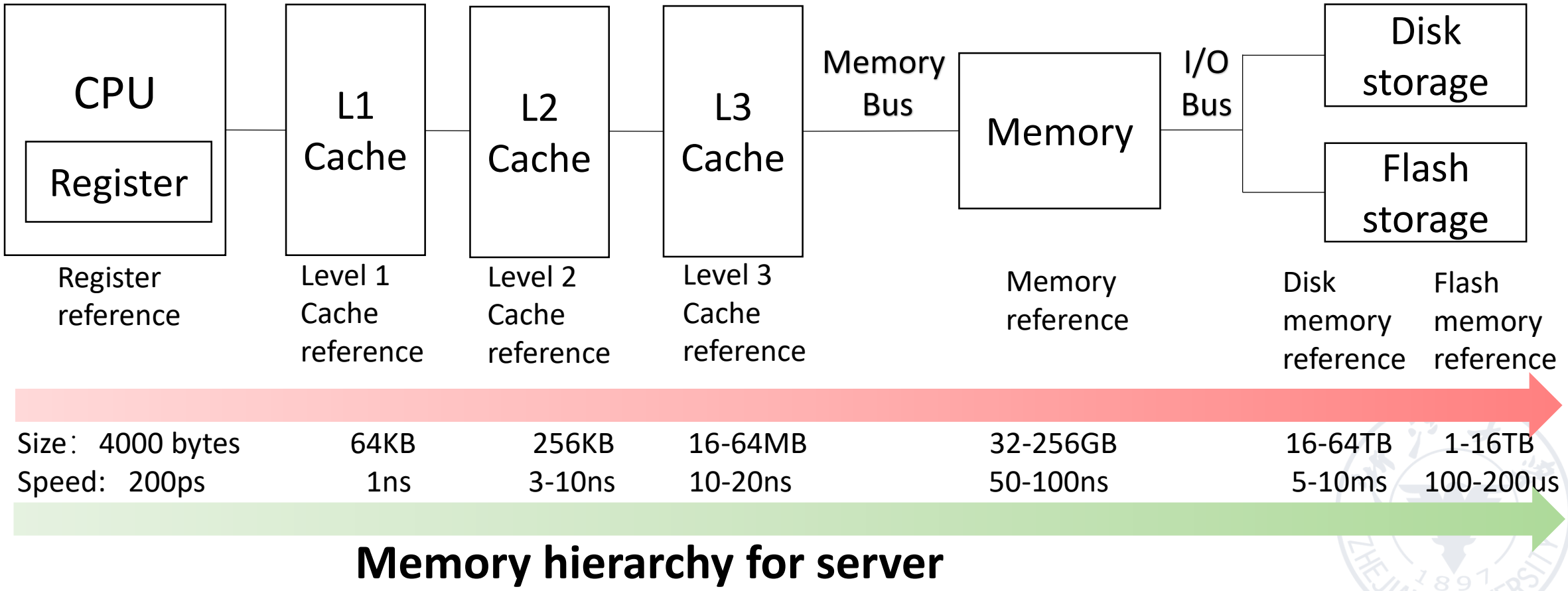
Memory Hierarchy



	Register reference	Level 1 Cache reference	Level 2 Cache reference	Level 3 Cache reference	Memory reference	Flash memory reference
Laptop	Size: 1000 bytes	64KB	256KB	4-8MB	4-16GB	256GB-1TB
	Speed: 300ps	1ns	5-10ns	10-20ns	50-100ns	50-100us
Desktop	Size: 2000 bytes	64KB	256KB	8-32MB	8-64GB	256GB-2TB
	Speed: 300ps	1ns	5-10ns	10-20ns	50-100ns	50-100us

Memory hierarchy for a laptop or a desktop

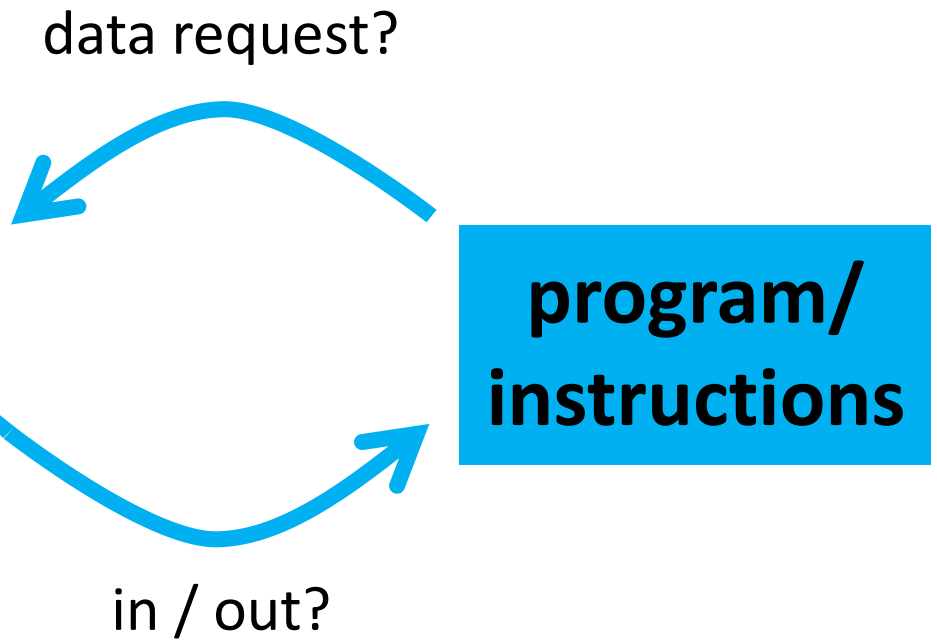
Memory Hierarchy



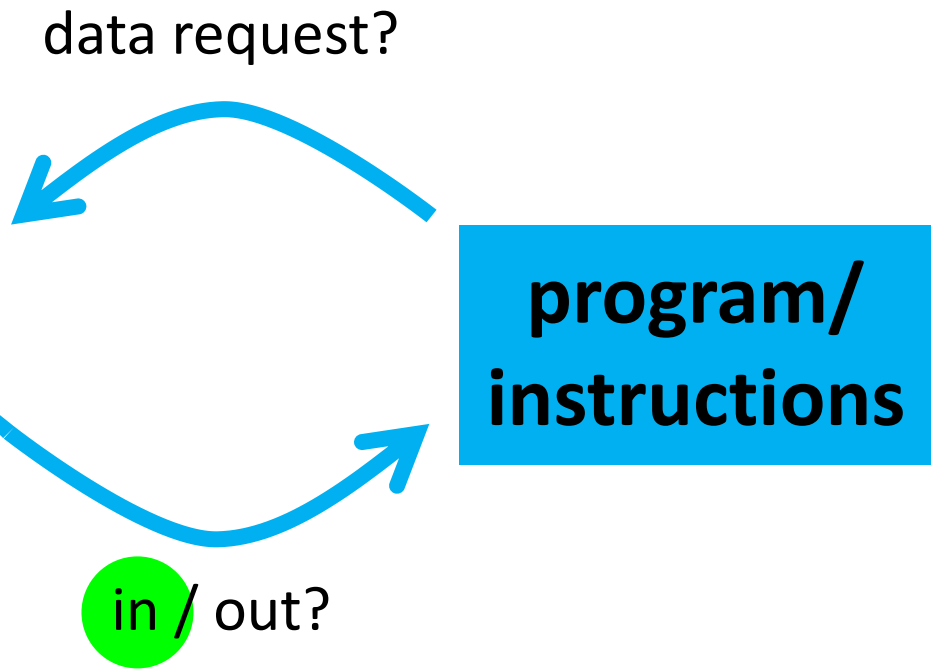
Memory hierarchy for server

Wait, but what's cache?

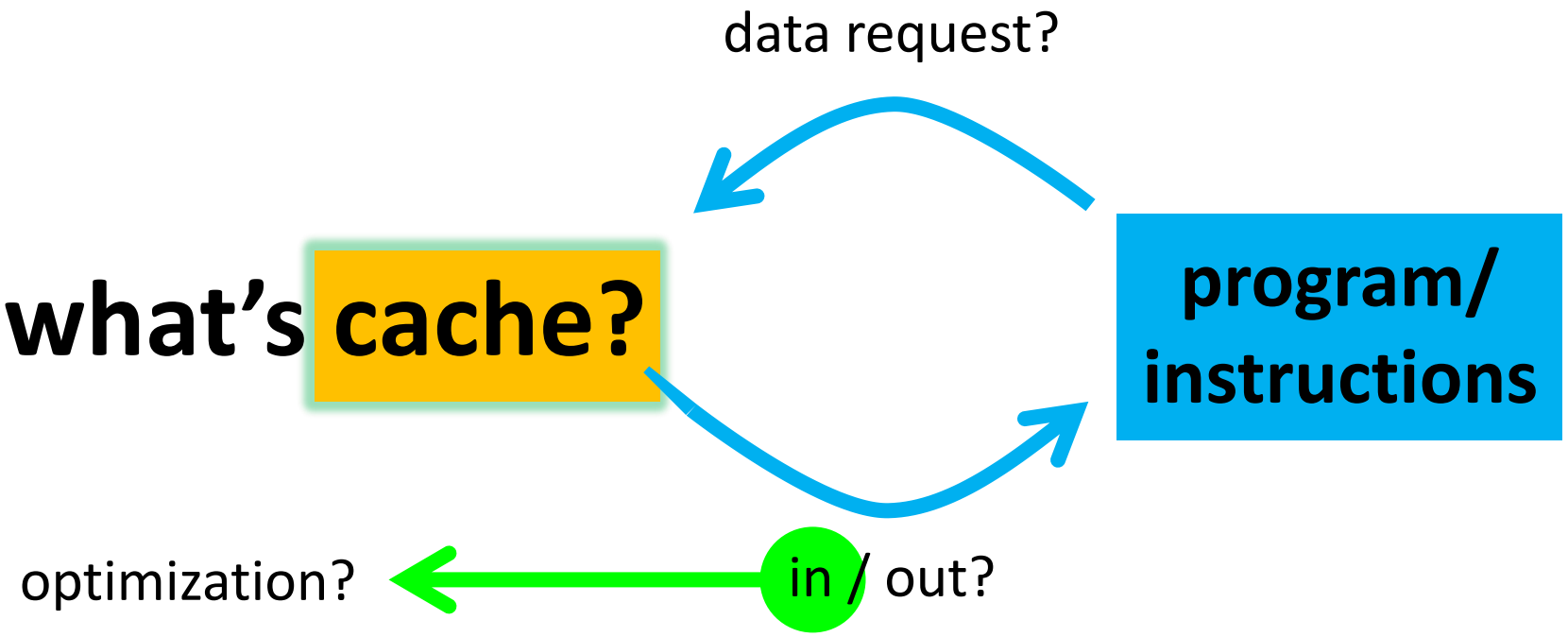
Wait, but what's cache?



Wait, but what's cache?



Wait, but what's cache?



So, what’s cache?

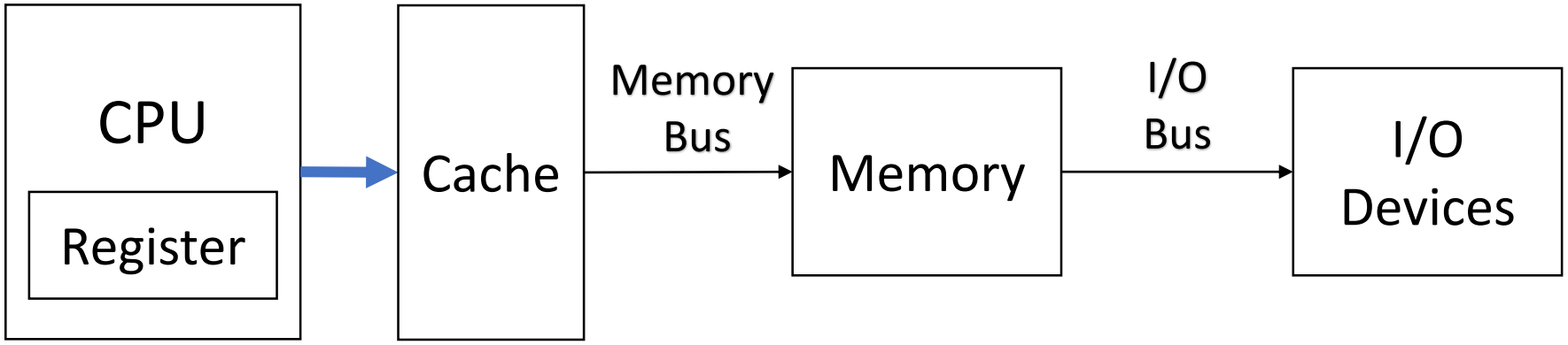
So, what's cache?

- **Cache: a safe place for hiding or storing things.**

Webster's New World Dictionary of the American Language

Second College Edition (1976)

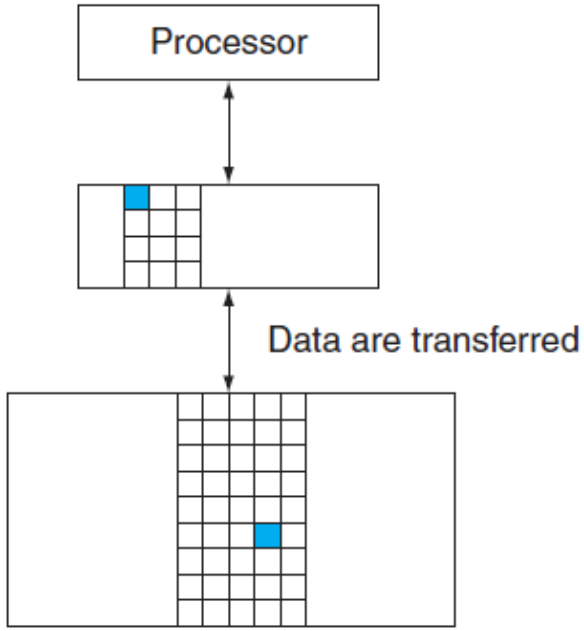
Cache



- The highest or first level of the memory hierarchy encountered once the *addr* leaves the processor
- Employ buffering to reuse commonly occurring items

Cache Hit/Miss

- When the processor can/cannot find a requested data item in the cache

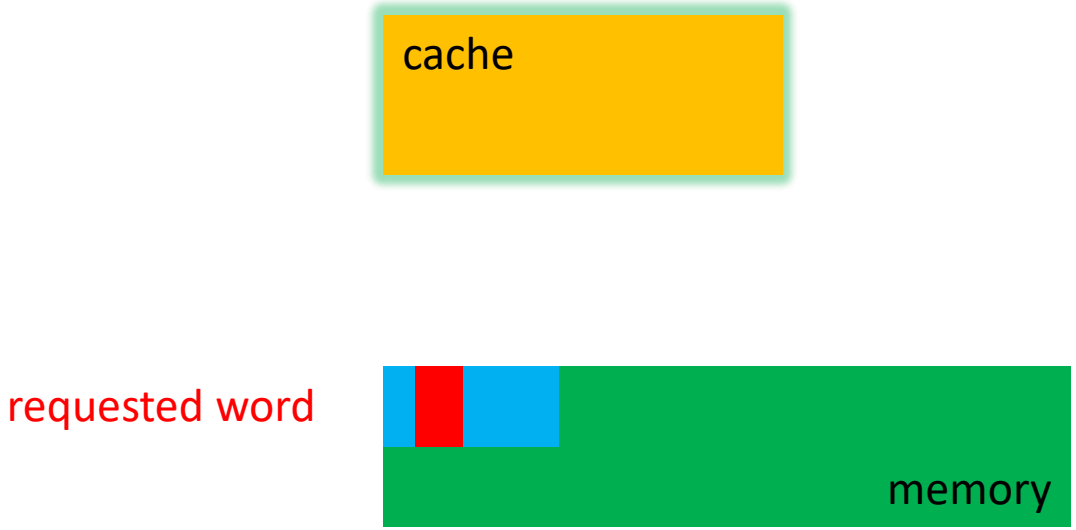


hit rate **miss rate**

hit time **miss penalty**

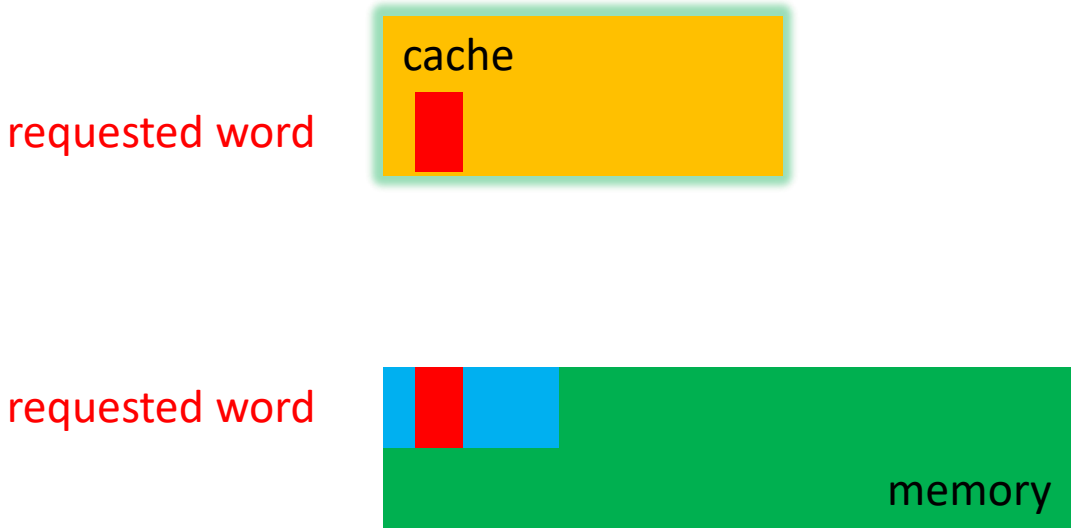
Block/Line Run

- A fixed-size collection of data containing the requested word, retrieved from the main memory and placed into the cache



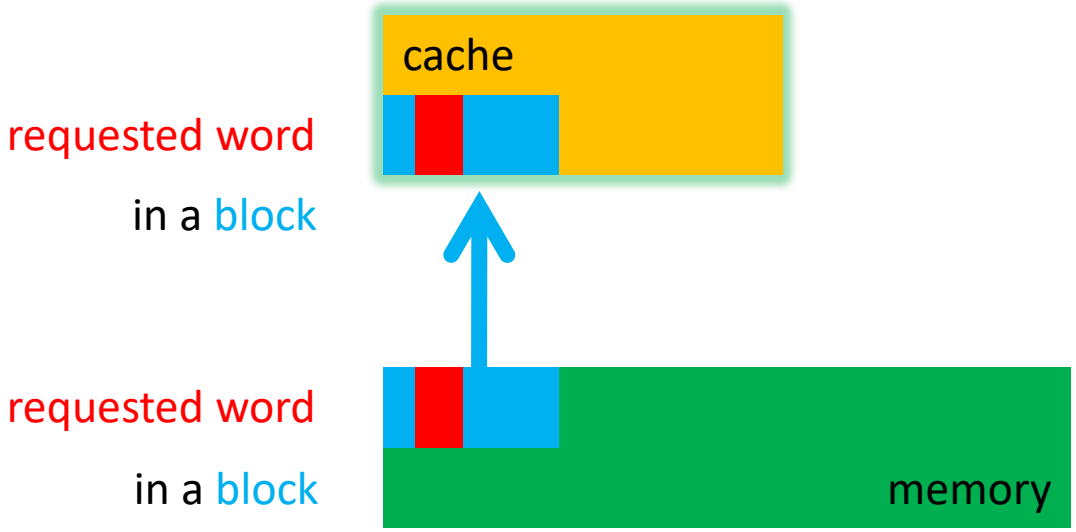
Block/Line Run

- A fixed-size collection of data containing the requested word, retrieved from the main memory and placed into the cache



Block/Line Run

- A fixed-size collection of data containing the requested word, retrieved from the main memory and placed into the cache



Cache Locality

- **Temporal locality**

  need the requested word again soon

- **Spatial locality**

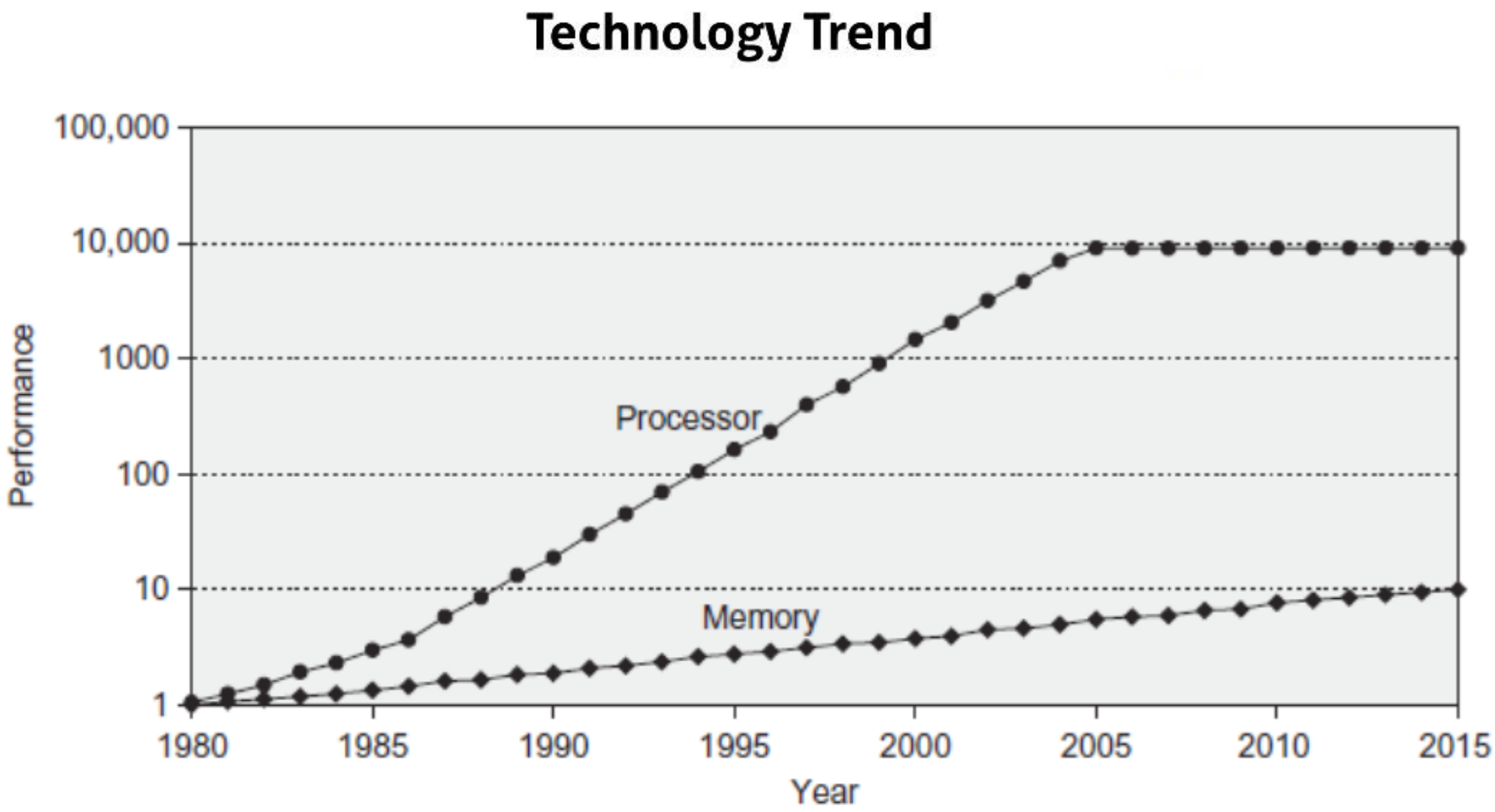
  likely need other data in the block soon

Cache Miss

- Time required for cache miss depends on:

  Latency: the time to retrieve the first word of the block

  Bandwidth: the time to retrieve the rest of this block


Technology Trend
Performance
100,000
10,000
1000
100
10
1
Processor
Memory
1980
1985
1990
1995
2000
2005
2010
2015
Year

§ 3.2 Technology Trend and Memory Hierarchy

Memory technology	Typical access time	$ per GiB in 2012
SRAM semiconductor memory	0.5–2.5 ns	$500–$1000
DRAM semiconductor memory	50–70 ns	$10–$20
Flash semiconductor memory	5,000–50,000 ns	$0.75–$1.00
Magnetic disk	5,000,000–20,000,000 ns	$0.05–$0.10

SRAM (*static random access memory*)

DRAM (*dynamic random access memory*)

SDRAM (Synchronous DRAM)

Double Data Rate (DDR) SDRAM

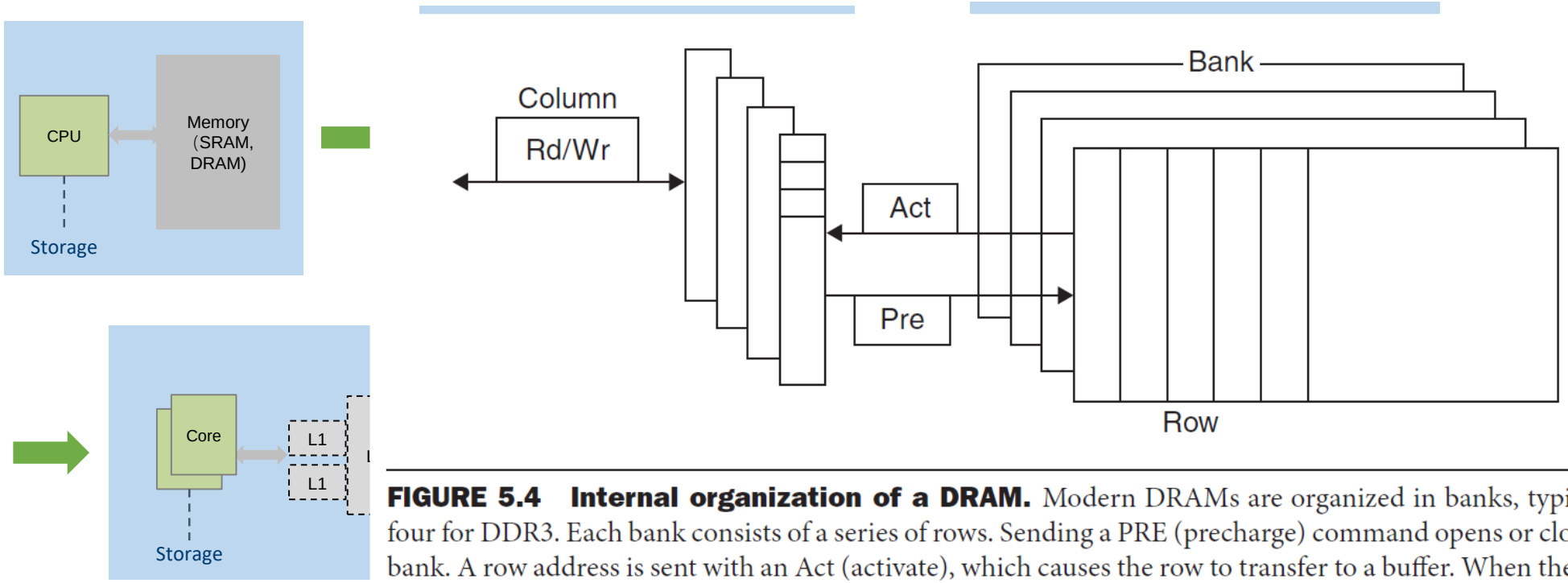
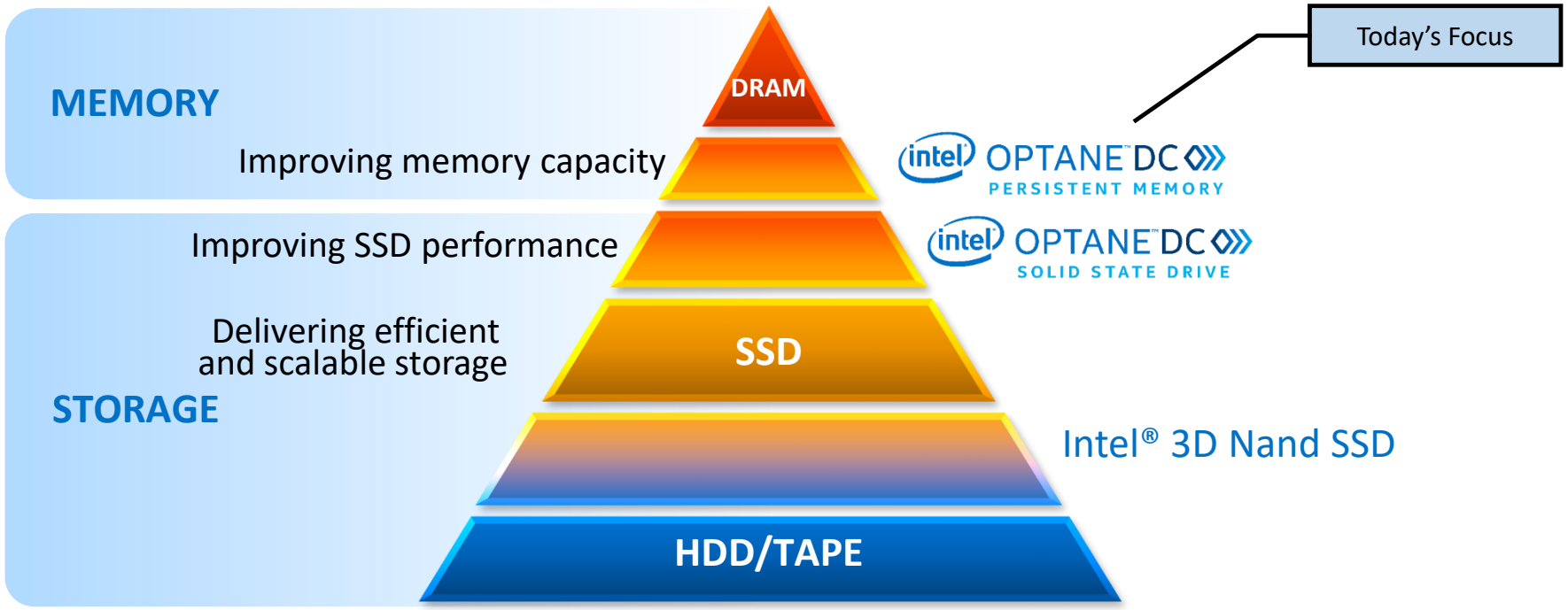


FIGURE 5.4 Internal organization of a DRAM. Modern DRAMs are organized in banks, typically four for DDR3. Each bank consists of a series of rows. Sending a PRE (precharge) command opens or closes a bank. A row address is sent with an Act (activate), which causes the row to transfer to a buffer. When the row is in the buffer, it can be transferred by successive column addresses at whatever the width of the DRAM is (typically 4, 8, or 16 bits in DDR3) or by specifying a block transfer and the starting address. Each command, as well as block transfers, is synchronized with a clock.

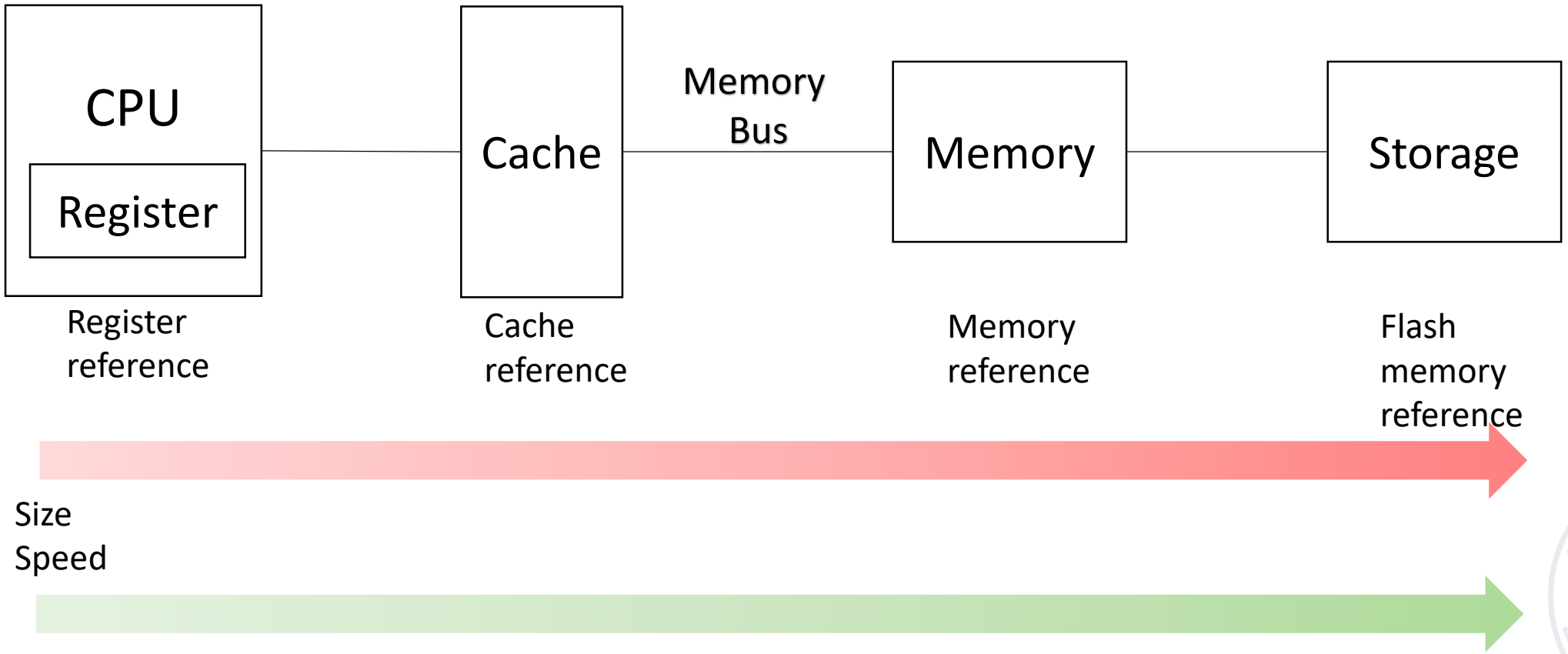
HBM (*High Bandwidth Memory)*

Memory/Storage

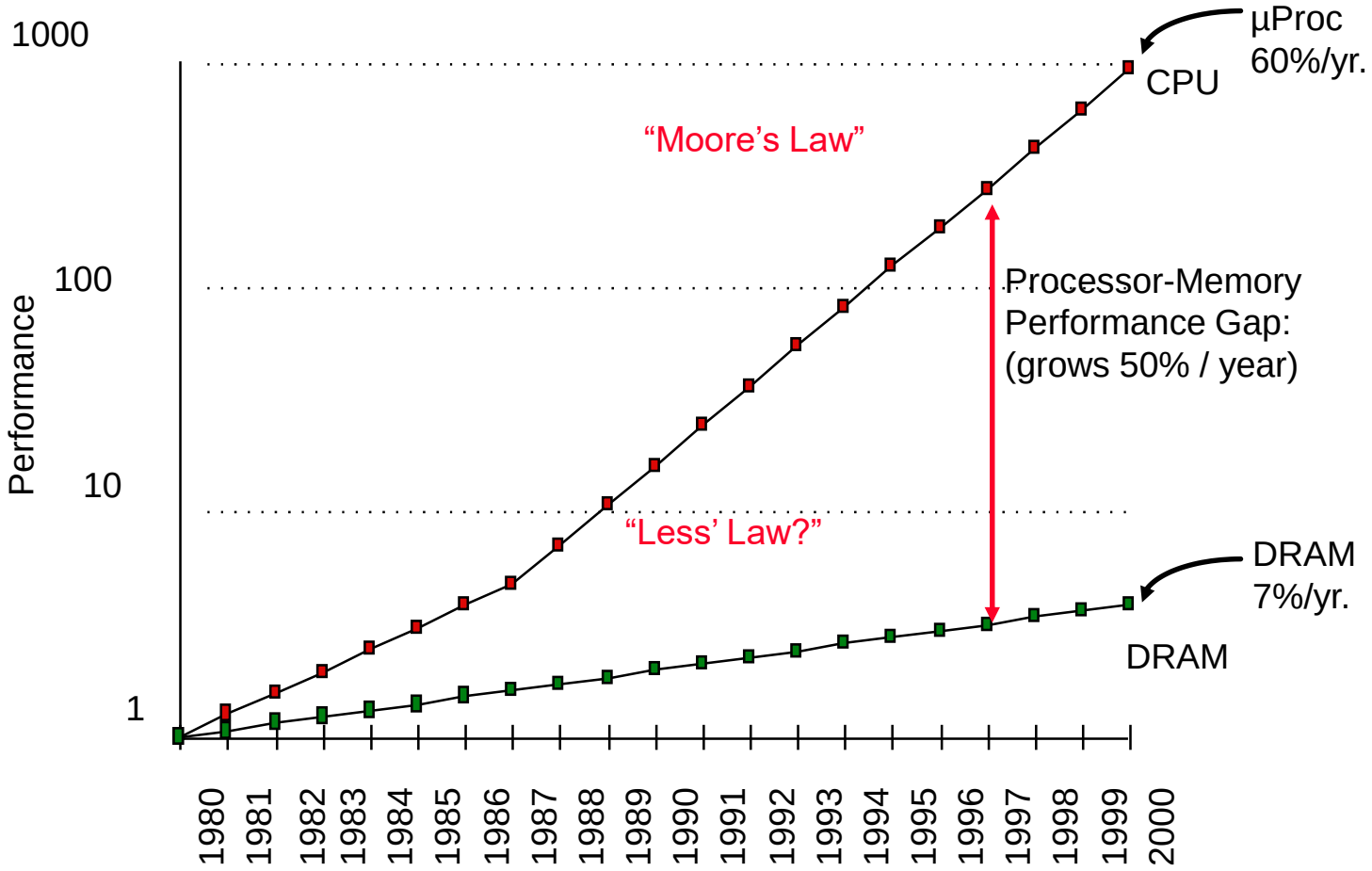


Flash Memory: electrically erasable programmable read-only memory (EEPROM).

Processor-Memory Performance Gap



Who Cares About the Memory Hierarchy?



1980: no cache in µproc;

1995 2-level cache on chip (1989 first Intel µproc with a cache on chip)

Three classes of computers with different concerns in memory hierarchy

- **Desktop computers:**
  - Are primarily running one application for single user
  - Are concerned more with average latency from the memory hierarchy.
- **Server computers:**
  - May typically have hundreds of users running potentially dozens of applications simultaneously.
  - Are concerned about memory bandwidth.

Three classes of computers with different concerns in memory hierarchy.

- **Embedded computers:**
  - Real-time applications.
    - Worst-case performance vs Best case performance
  - Are concerned more about power and battery life.
    - Hardware vs software
  - Running single app & use simple OS
    - The protection role of the memory hierarchy is often diminished.
  - Main memory is very small
    - often no disk storage

Enhance speed of memory

- Component character of hardware:
  - Smaller hardware is faster and more expensive
  - Bigger memories are lower and cheaper

The goal:

- There are speed of smallest memory and capacity of biggest memory
- To provide cost almost as low as the cheapest level of memory and speed almost as fast as the fastest level.

The method enhance speed of memory

By taking advantage of the principle of locality:

- **most programs do not access all code or data uniformly**
- Temporal Locality (Locality in Time):
- If an item is referenced, the same item will tend to be referenced again soon
  - Keep most recently accessed data items closer to the processor
- Spatial Locality (Locality in Space):
- If an item is referenced, nearby items will tend to be referenced soon
  - Move recently accessed groups of contiguous words(block) closer to processor.

The method enhance speed of memory

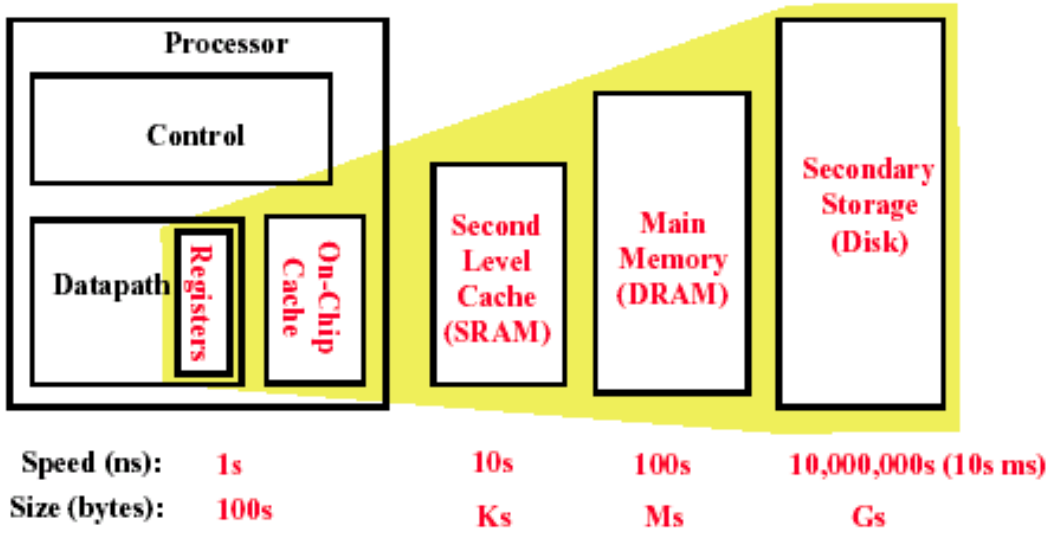
The method

- Hierarchies bases on memories of different speeds and size
- The more closely CPU the level is, the faster the one is.
- The more closely CPU the level is, the smaller the one is.
- The more closely CPU the level is, the more expensive one is.

Memory Hierarchy of a Modern Computer System

By taking advantage of the principle of locality:

- Present the user with as much memory as is available in the cheapest technology.
- Provide access at the speed offered by the fastest technology.

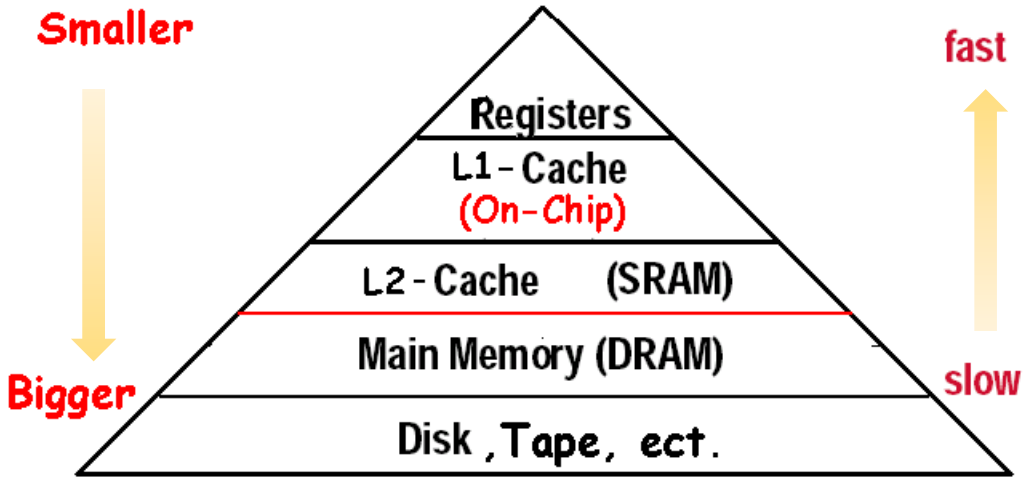


36 terms of Cache

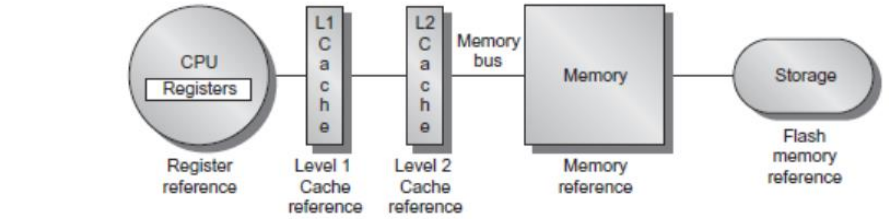
Cache	full associative	write allocate
Virtual memory	dirty bit	unified cache
Memory stall cycles	block	block offset
misses per instruction	direct mapped	write back
Valid bit	data cache	locality
Block address	hit time	address trace
Write through	cache miss	set
Instruction cache	page fault	miss rate
random replacement	index field	cache hit
Average memory access time	page	tag field
n-way set associative	no-write allocate	miss penalty
Least-recently used	write buffer	write stall

What is a cache?

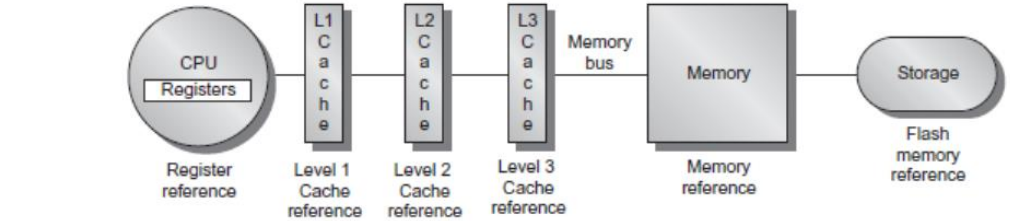
- Small, fast storage used to improve average access time to slow memory.
- In computer architecture, almost everything is a cache!
  - Registers “a cache” on variables – software managed
  - First-level cache a cache on second-level cache
  - Second-level cache a cache on memory
  - Memory a cache on disk (virtual memory)
  - TLB a cache on page table
  - Branch-prediction a cache on prediction information?



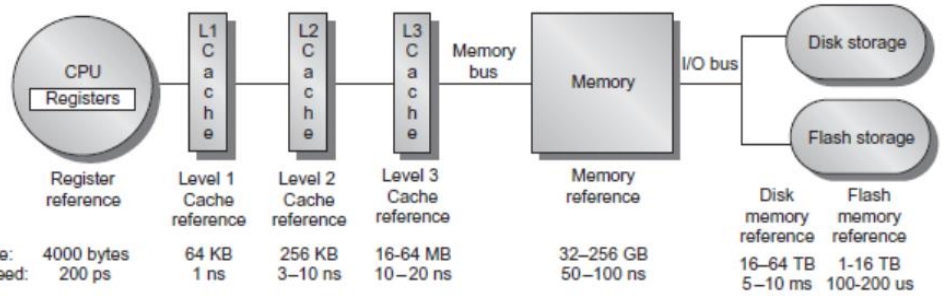
§ 3.2 Technology Trend and Memory Hierarchy



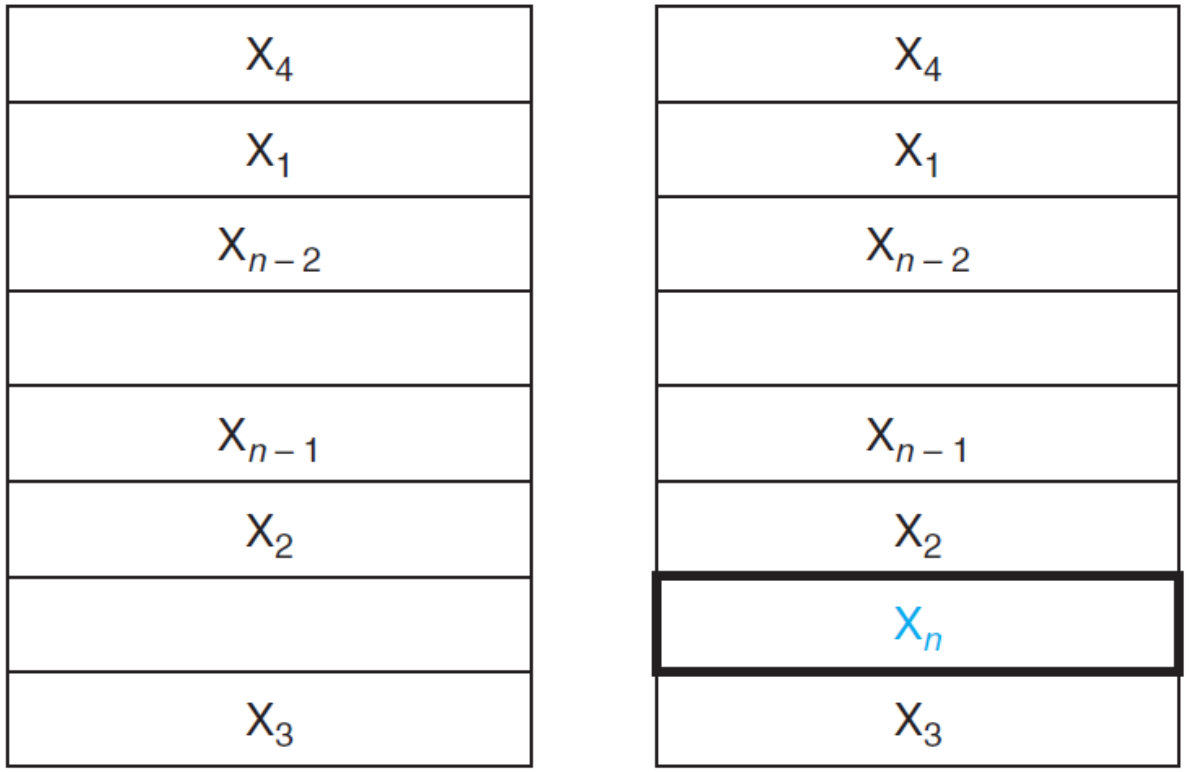
(A) Memory hierarchy for a personal mobile device



(B) Memory hierarchy for a laptop or a desktop



(C) Memory hierarchy for server



a. Before the reference to X_n b. After the reference to X_n

How do we know if a data item is in the cache?
Moreover, if it is, how do we find it?

Four Questions for Cache Designers

Caching is a general concept used in processors, operating systems, file systems, and applications.

There are **Four Questions** for Cache/Memory Hierarchy Designers

- Q1: Where can a block be placed in the upper level/main memory?

  (Block placement)

  - Fully Associative, Set Associative, Direct Mapped
- Q2: How is a block found if it is in the upper level/main memory?

  (Block identification)

  - Tag/Block
- Q3: Which block should be replaced on a Cache/main memory miss?

  (Block replacement)

  - Random, LRU,FIFO
- Q4: What happens on a write?

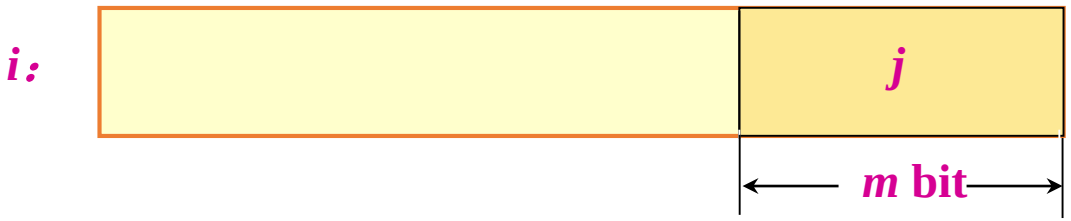
  (Write strategy)

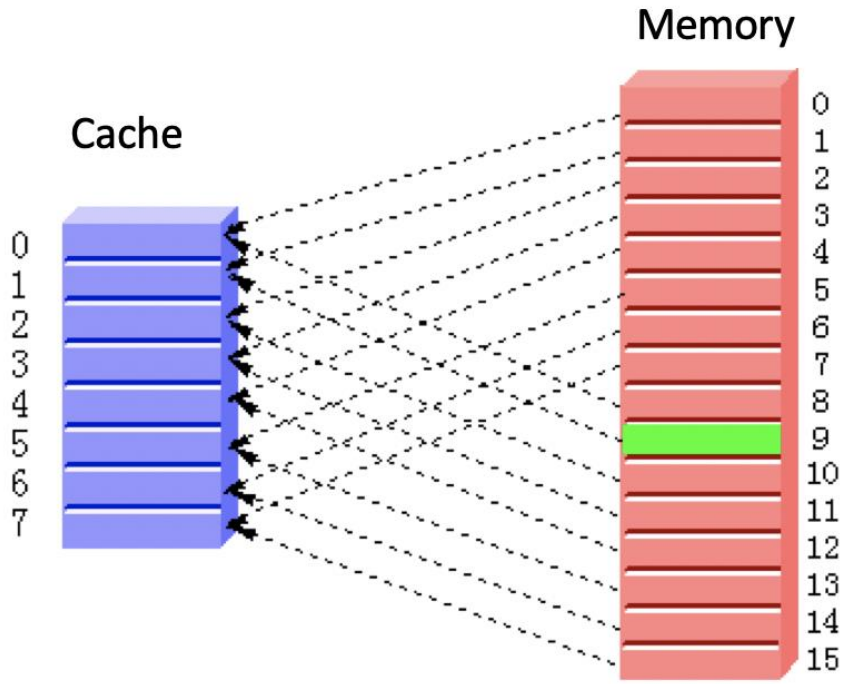
  - Write Back or Write Through (with Write Buffer)

Q1: Block Placement

Direct mapped

(Block address) modulo (Number of blocks in the cache)

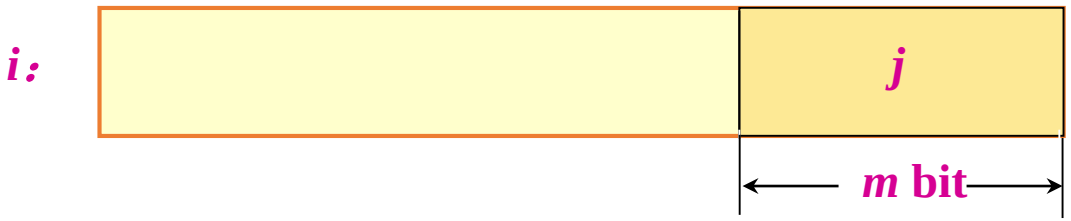


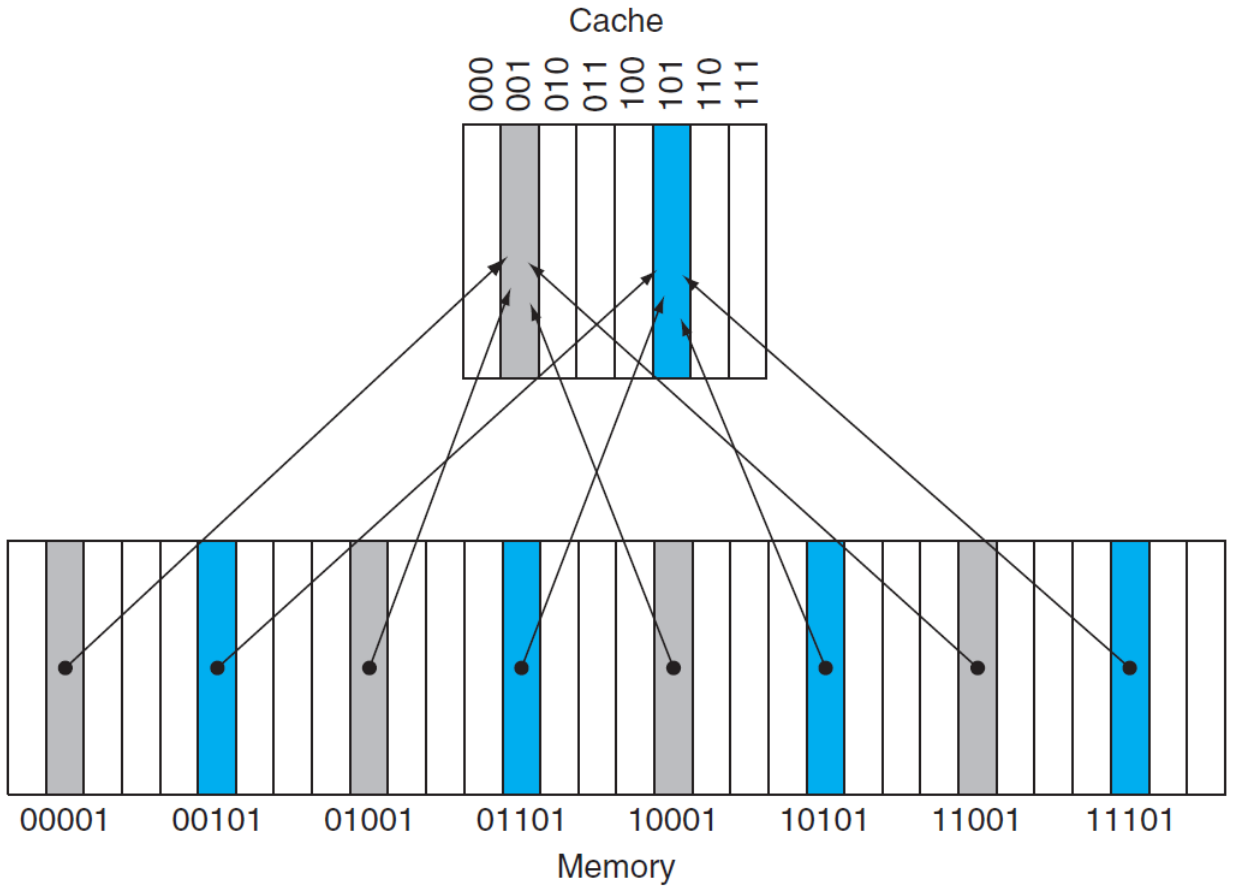


Q1: Block Placement

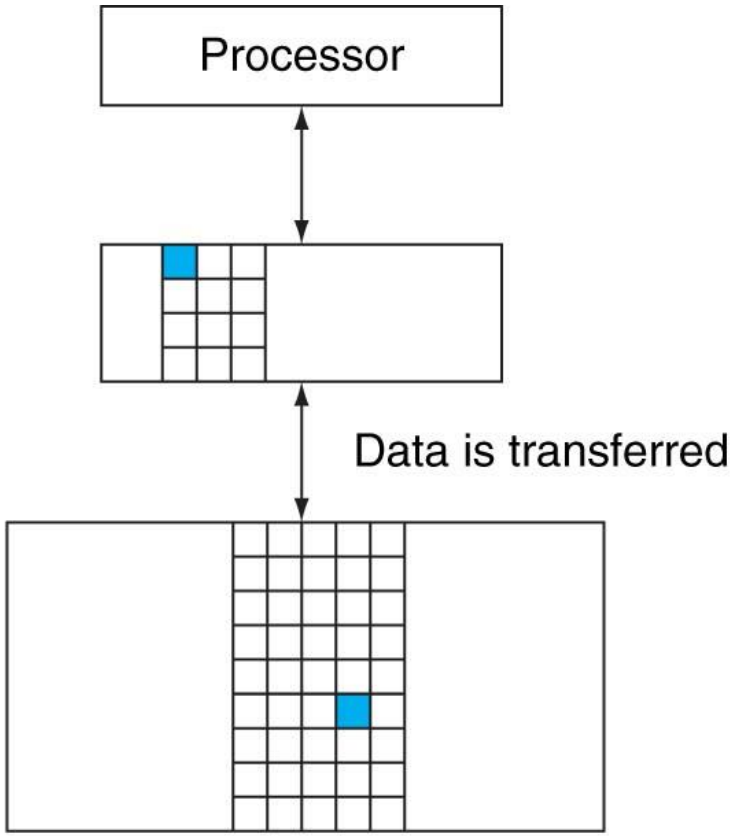
Direct mapped

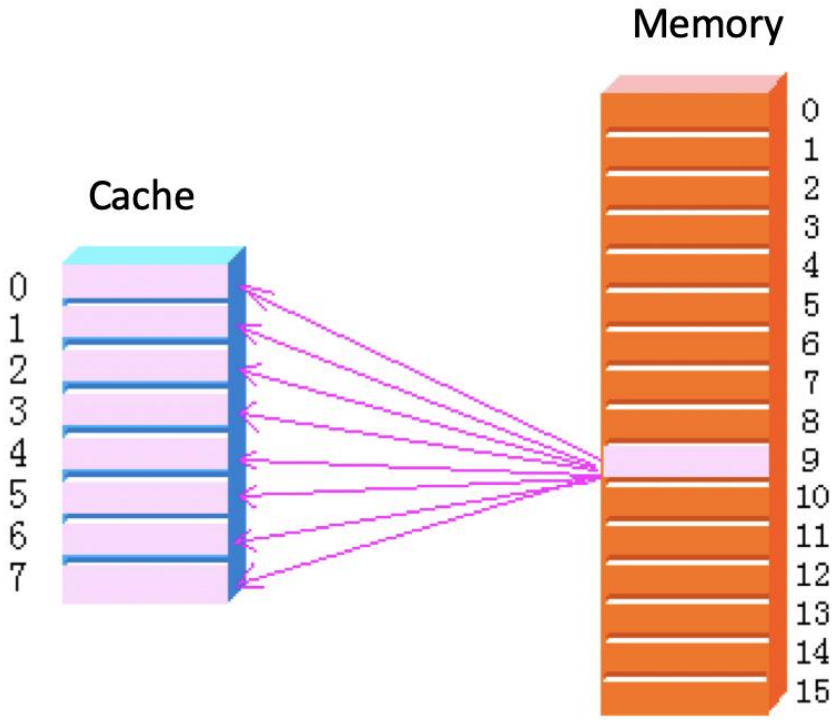
(Block address) modulo (Number of blocks in the cache)



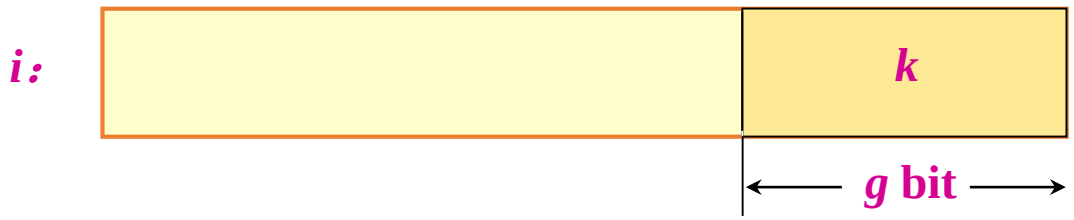


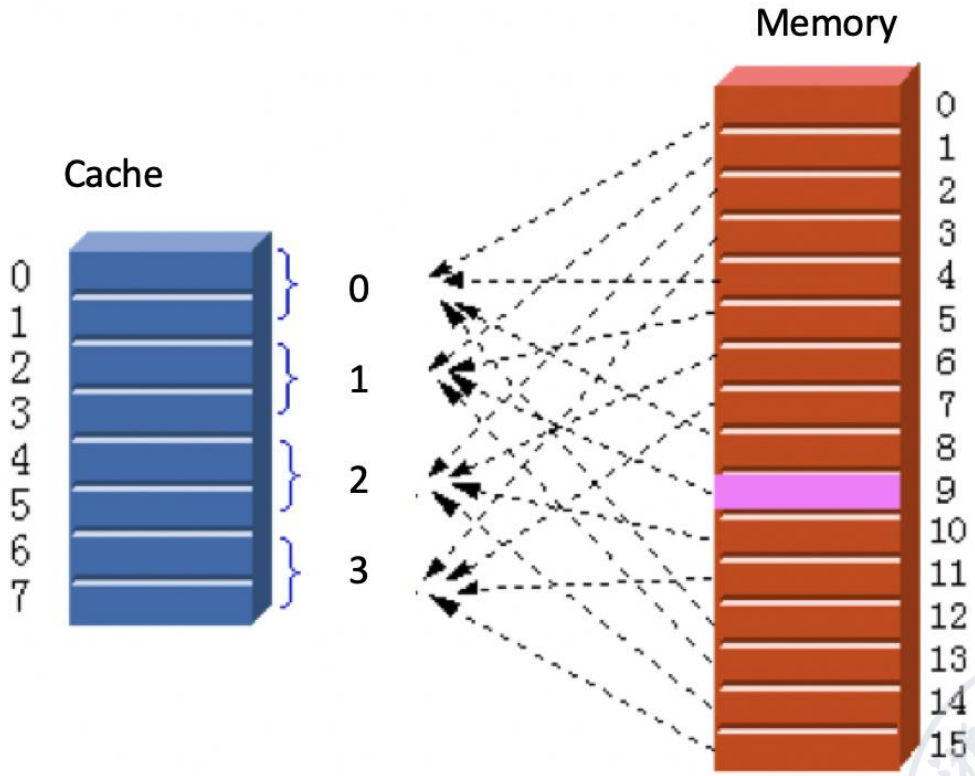
Fully-associative





2-way Set-associative





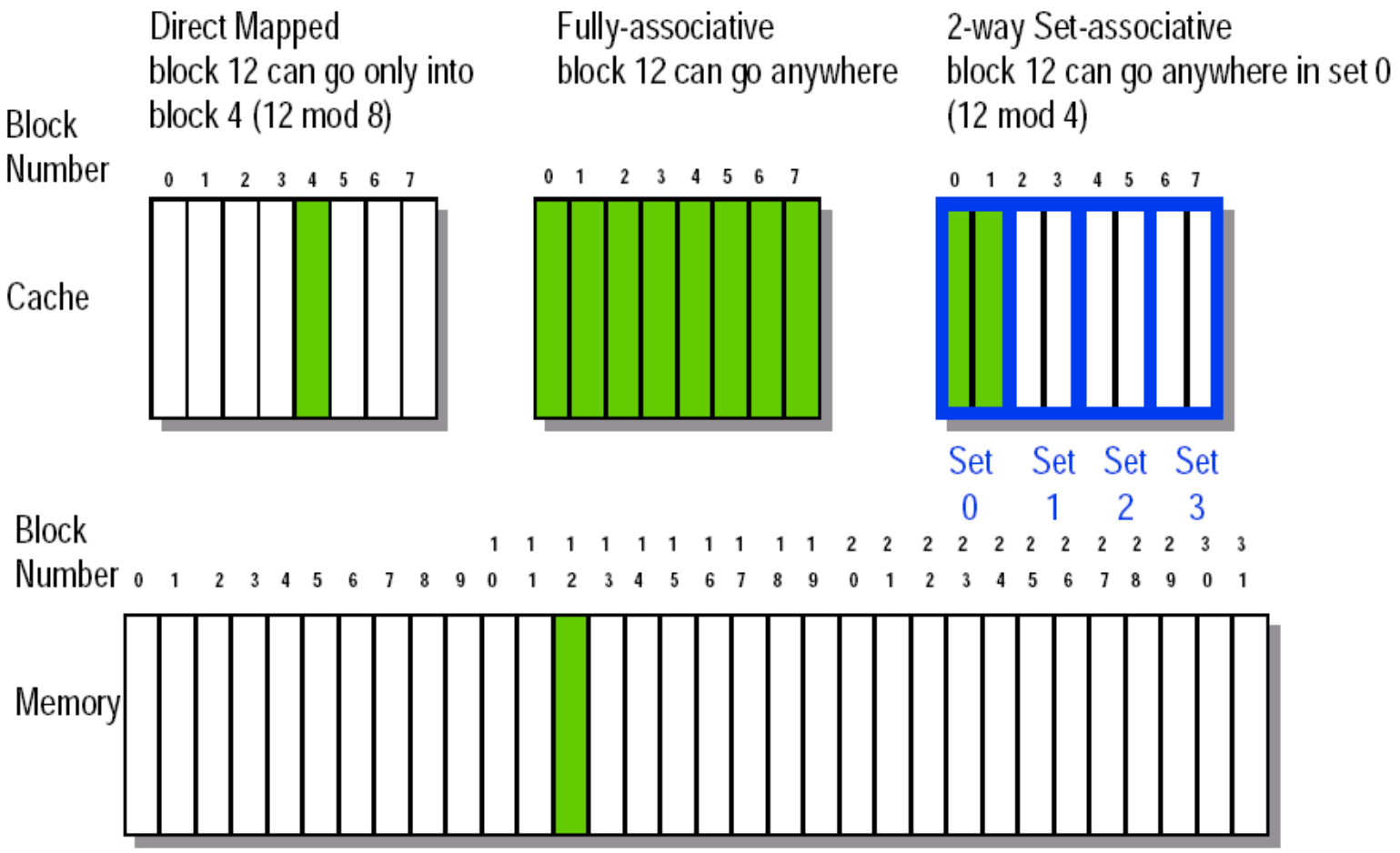
Q1: Block Placement

- Direct mapped
  - Block can only go in one place in the cache.
- Fully
- Set as
  - Block can go in one of a set of places in the cache.
  - A set is a group of blocks in the cache.

    Block address MOD Number of *sets* in the cache
  - If sets have n blocks, the cache is said to be n-way set associative.

•Note that direct mapped is the same as 1-way set associative, and fully associative is m-way set-associative (for a cache with m blocks).

8-32 Block Placement



N-way Set-associative

- The higher the degree of association, the higher the utilization of cache space, the lower the probability of block collision and the lower the failure rate.

	n	G
Full-associative	***M***	***1***
Direct mapped	***1***	***M***
Set-associative	***1 < n < M***	***1 < G < M***

- Most Cache: $n \leq 4$
- Question： Is the greater the number n, the better?

Q2: Block Identification

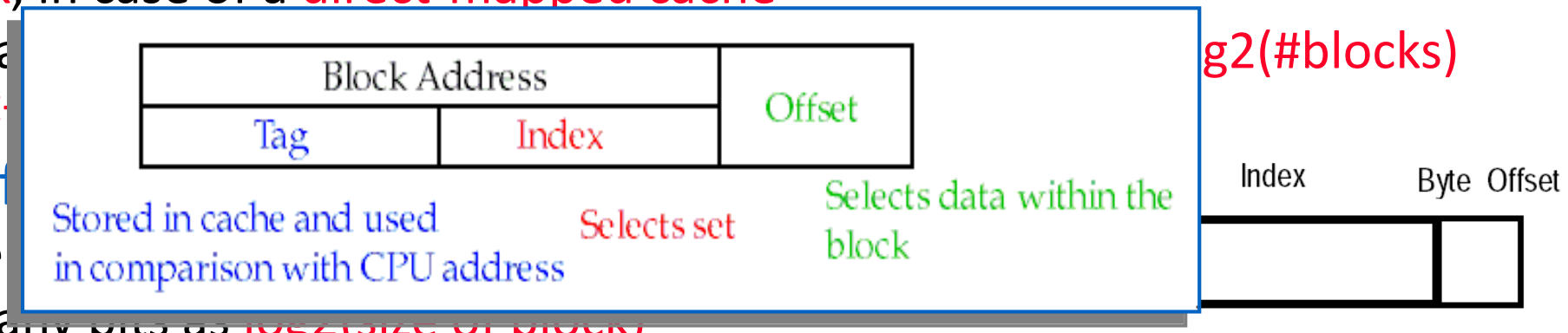
- Every block has an **address tag** that stores the main memory address of the data stored in the block.

- When checking the cache, the processor will **compare** the requested **memory address to the cache tag** -- if the two are equal, then there is a cache hit and the data is present in the cache

- Often, each cache block also has a **valid bit** that tells if the contents of the cache block are valid

The Format of the Physical Address

- The Index field selects
  - The set, in case of a set-associative cache
  - The block, in case of a direct-mapped cache
  - Has as ma… g2(#blocks)
    for direct-
- The Byte Of…
  - The byte…
  - Has as many bits as log2(size of block)
- The Tag is used to find the matching block within a set or in the cache
  - Has as many bits as Address_size – Index_size – Byte_Offset_Size

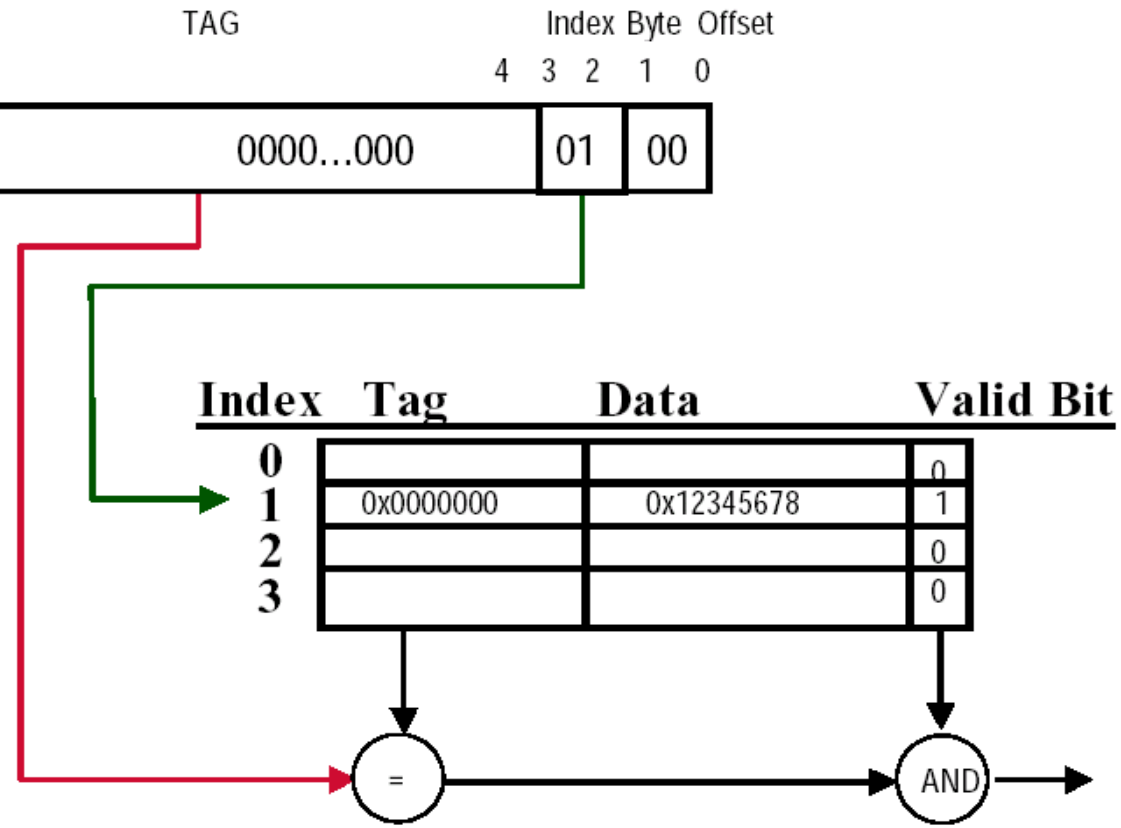


Direct-mapped Cache Example (1-word Blocks)

LOAD R1, 0x04

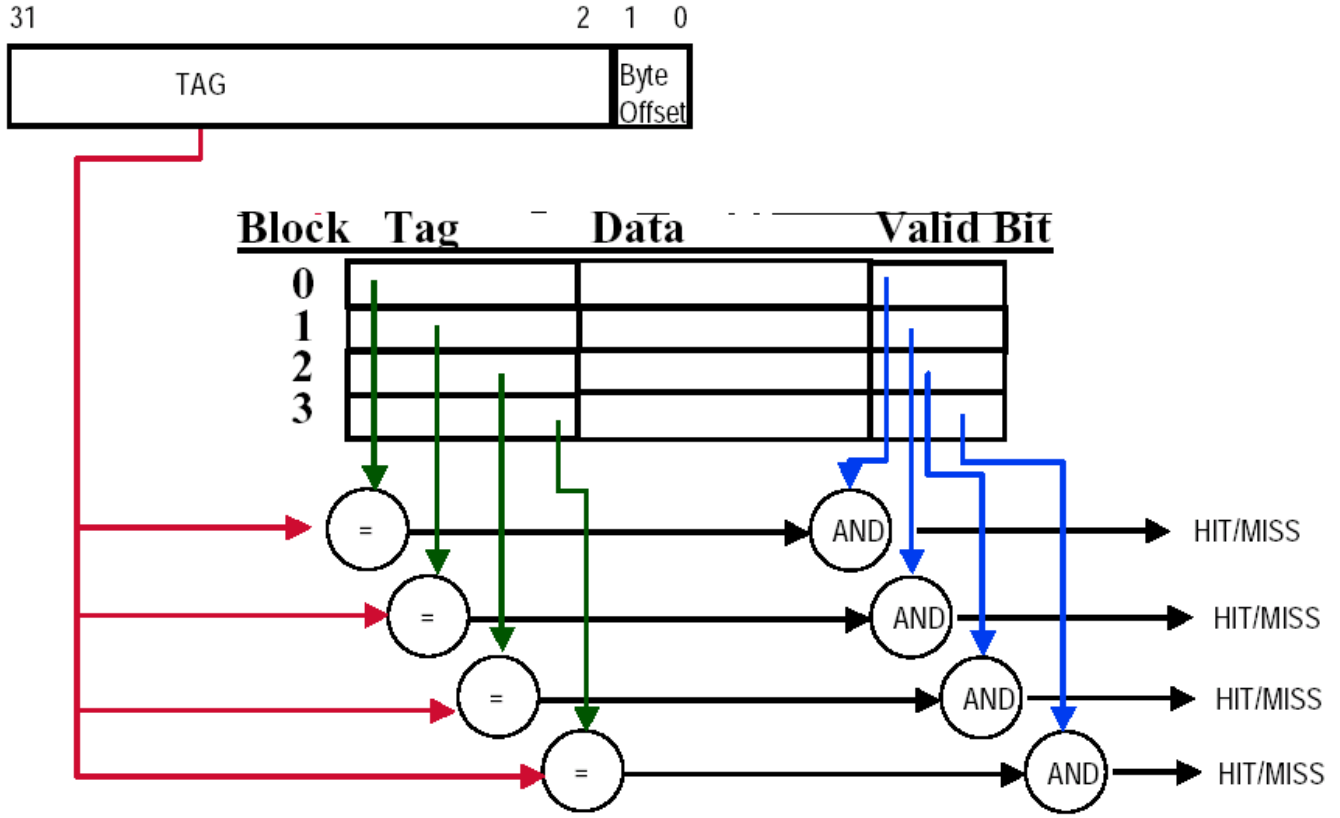
MEMORY

Address	Data
0x00	0x00000000
0x04	0x12345678
0x08	0x87654321
0x0C	0x11111111
0x10	0x22222222
0x14	0x33333333
0x18	0x44444444
0x1C	0x55555555
0x20	0x10101010



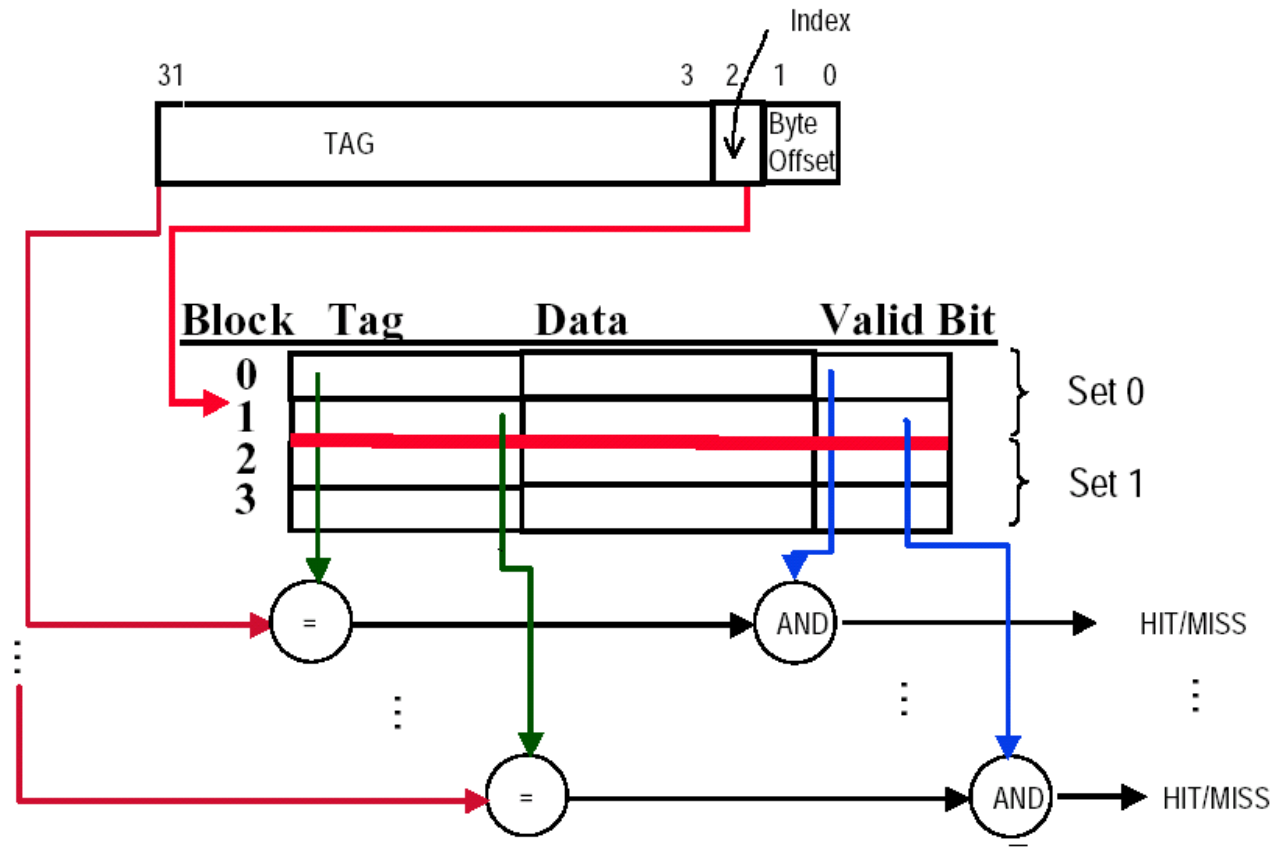
Fully-Associative Cache example (1-word Blocks)

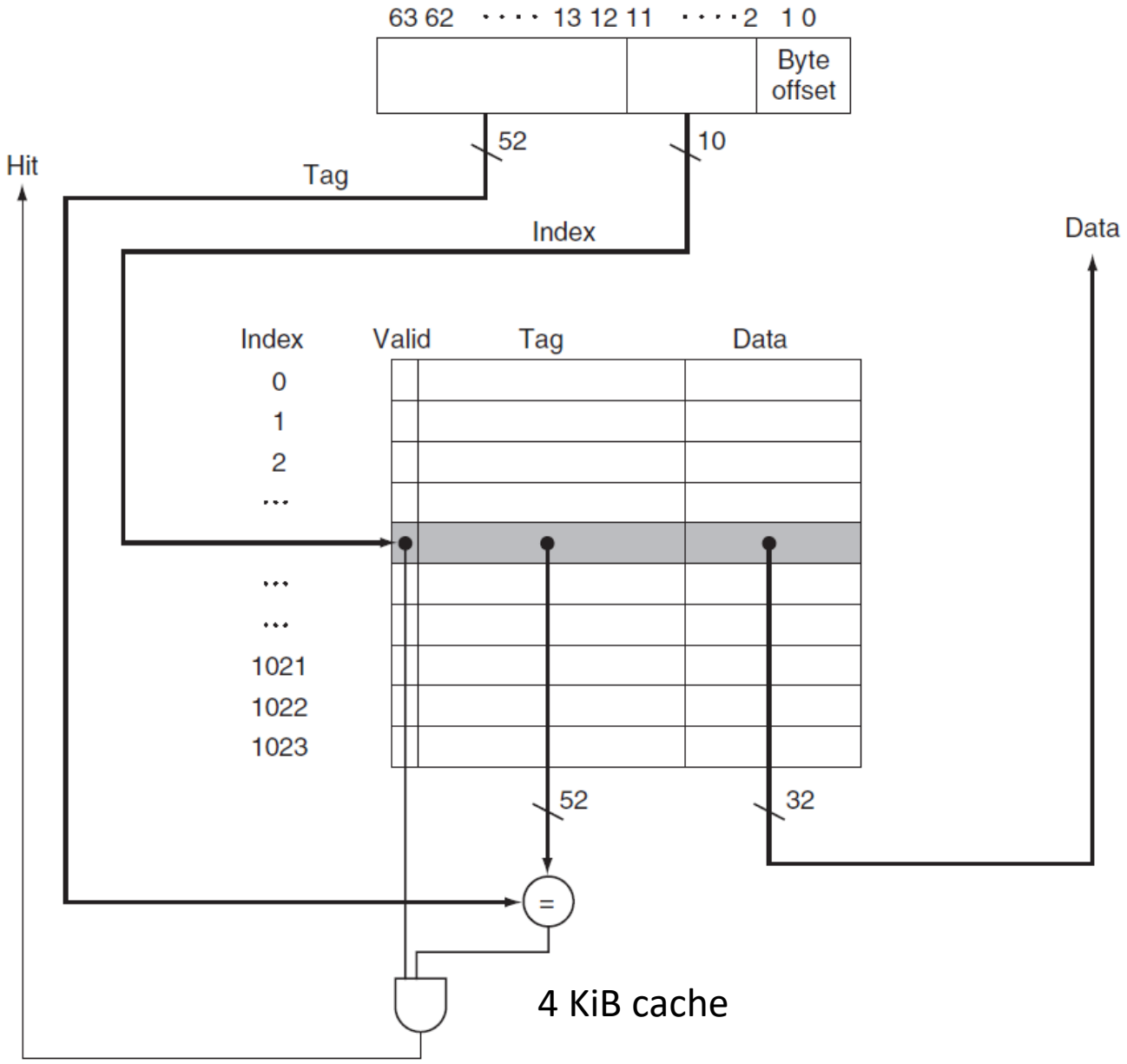
- Assume cache has 4 blocks



2-Way Set-Associative Cache

- Assume cache has 4 blocks and each block is 1 word
- 2 blocks per set, hence 2 sets per cache





4 KiB cache

- 64-bit addresses
- A direct-mapped cache
- The cache size is 2^n blocks, so n bits are used for the index
- The block size is
  - ✓ 2^m words (2^{m+2} bytes = 2^{m+5} bits)
  - ✓ m bits are used for the word within the block, and two bits are used for the byte part of the address

tag $64 - (n + m + 2)$

cache $2^n \times$(block size+ tag size + Valid size).

$= 2^n \times (2^m \times 32 + (64 - n - m - 2) + 1)$

$= 2^n \times (2^m \times 32 + 63 - n - m)$.

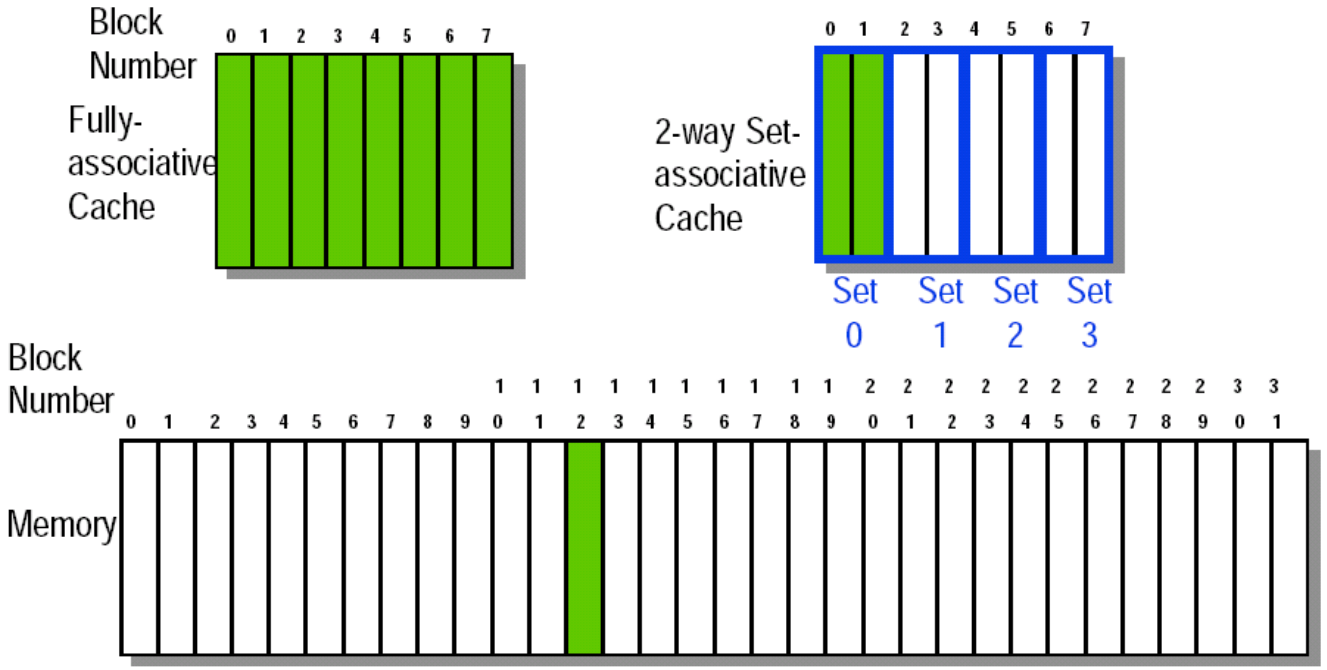
actual size/complete cache size

Binary address of reference	Assigned cache block (where found or placed)
10110_{two}	$(10110_{two} \bmod 8) = 110_{two}$
11010_{two}	$(11010_{two} \bmod 8) = 010_{two}$
10110_{two}	$(10110_{two} \bmod 8) = 110_{two}$
11010_{two}	$(11010_{two} \bmod 8) = 010_{two}$
10000_{two}	$(10000_{two} \bmod 8) = 000_{two}$
00011_{two}	$(00011_{two} \bmod 8) = 011_{two}$
10000_{two}	$(10000_{two} \bmod 8) = 000_{two}$
10010_{two}	$(10010_{two} \bmod 8) = 010_{two}$
10000_{two}	$(10000_{two} \bmod 8) = 000_{two}$

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Q3: Block Replacement

- In a direct-mapped cache, there is only one block that can be replaced
- In set-associative and fully-associative caches, there are N blocks (where N is the degree of associativity)



Strategy of Block Replacement

- Several different replacement policies can be used
  - Random replacement - *randomly pick any block*
    - Easy to implement in hardware, just requires a random number generator
    - Spreads allocation uniformly across cache
    - May evict a block that is about to be accessed
  - Least-Recently Used (LRU) - *pick the block in the set which was least recently accessed*
    - Assumed more recently accessed blocks more likely to be referenced again
    - This requires extra bits in the cache to keep track of accesses.
  - First In, First Out(FIFO)-*Choose a block from the set which was first came into the cache*

Strategy of Block Replacement

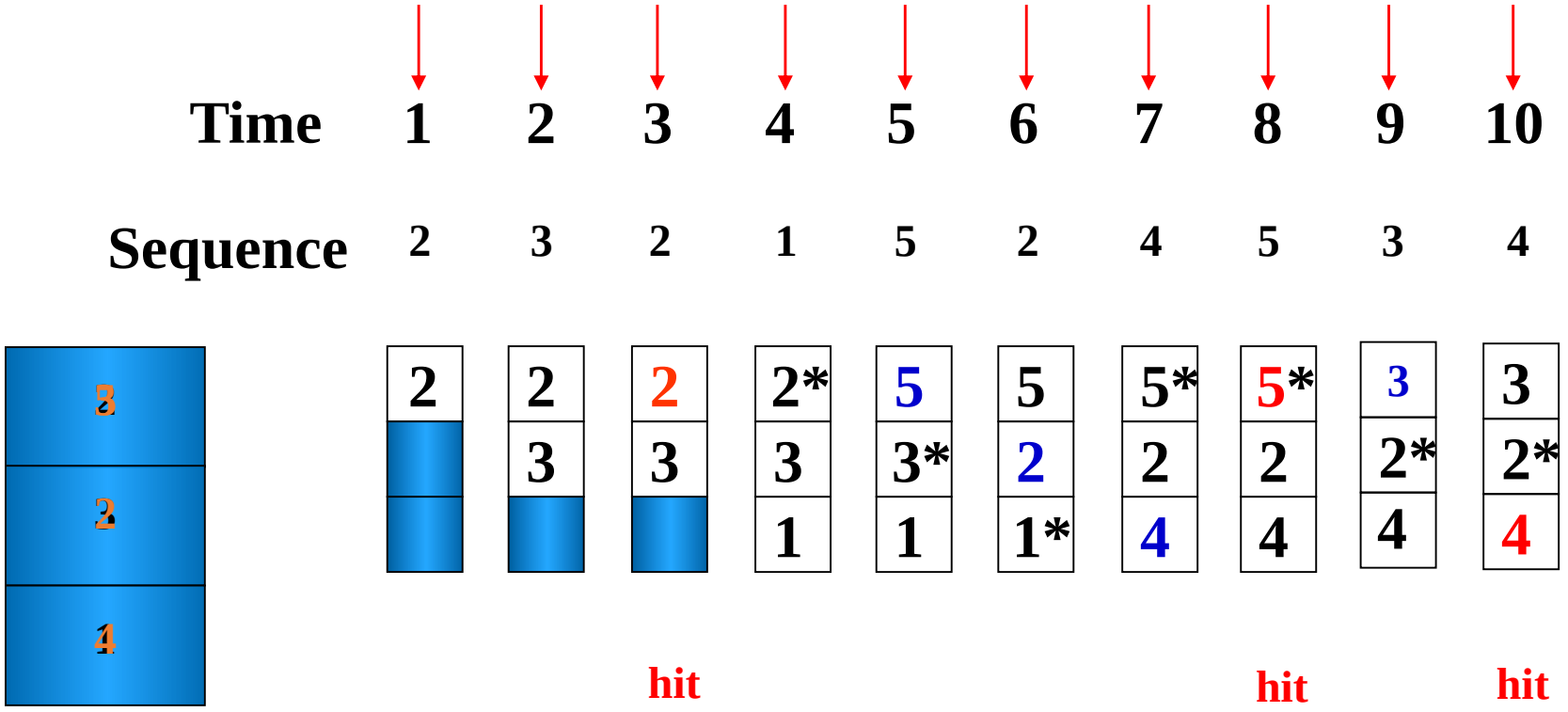
Suppose:

- Cache block size is 3, and access sequence is shown as follows.

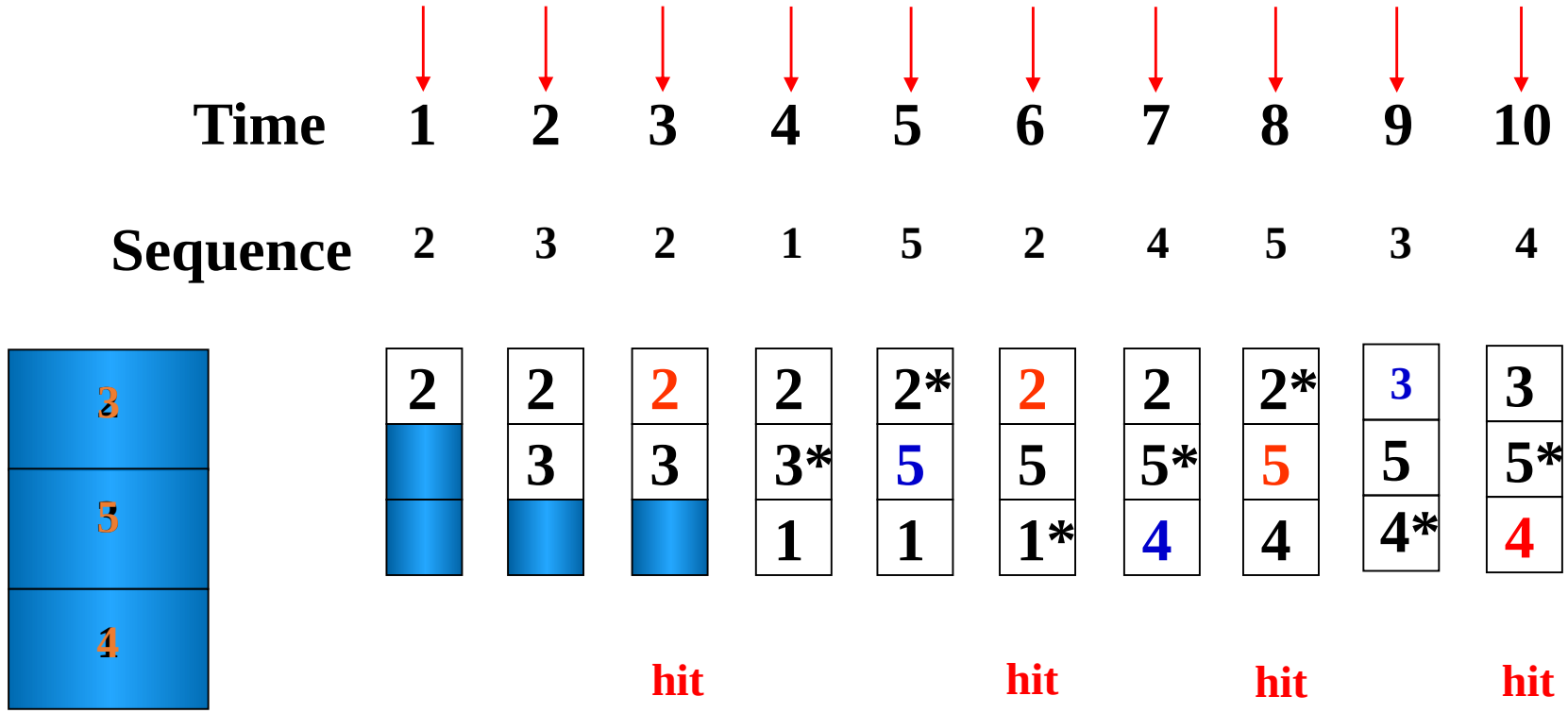
  2, 3, 2, 1, 5, 2, 4, 5, 3, 4

- FIFO, LRU and OPT are used to simulate the use and replacement of cache block.

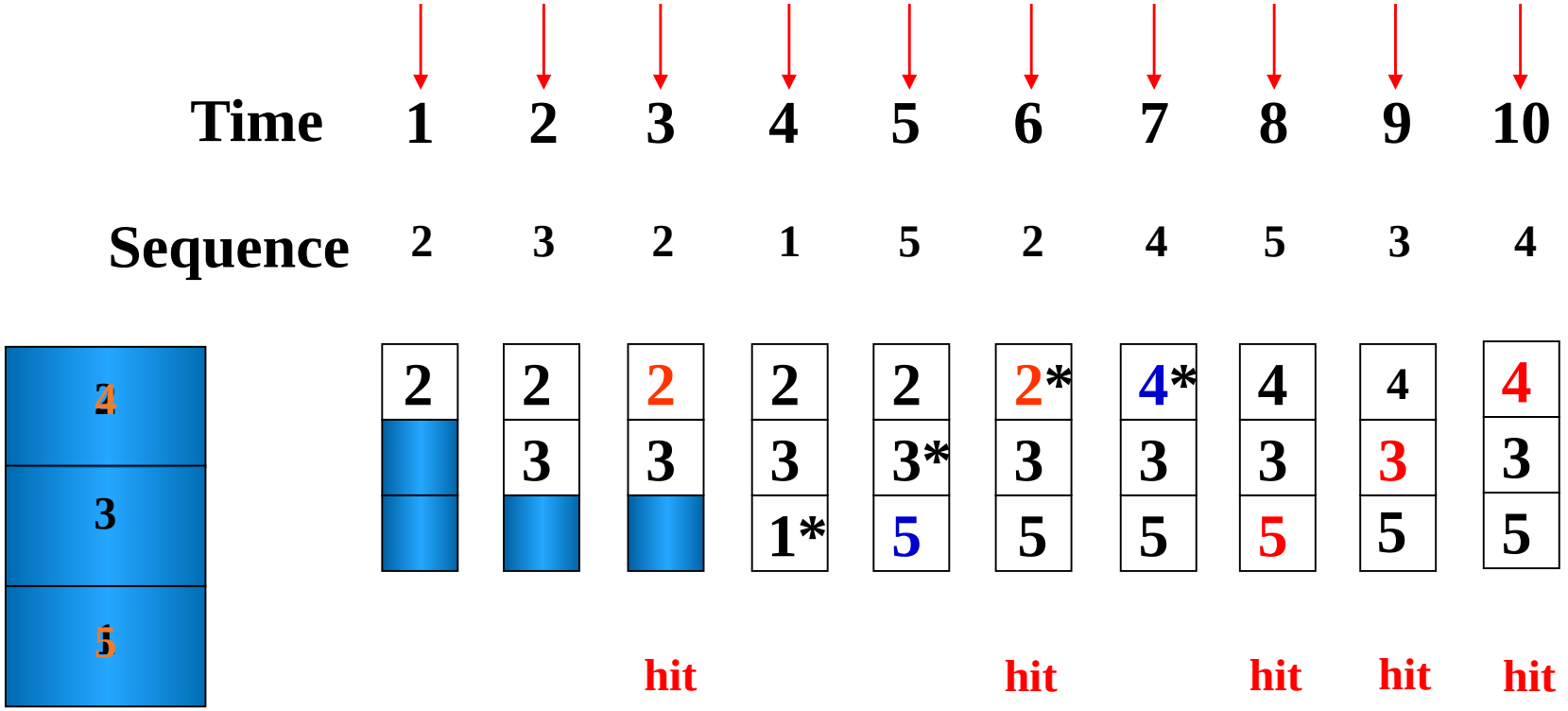
FIFO


Time
1 2 3 4 5 6 7 8 9 10
Sequence
2 3 2 1 5 2 4 5 3 4
3
2
1
2 2 2 2* 5 5 5* 5* 3 3
3 3 3* 2 2 2* 2*
1 1 1* 4 4 4 4
hit
hit
hit

LRU


Time
1 2 3 4 5 6 7 8 9 10
Sequence
2 3 2 1 5 2 4 5 3 4
2 3 4
2 2 2 2 2* 2 2 2* 3 3
3 3 3* 5 5 5* 5 5 5*
1 1 1* 4 4 4* 4
hit
hit
hit
hit

OPT



The hit rate is related to the replacement algorithm.