

Memory Systems

Memory hierarchy, cache memory, virtual memory

Gloria Beraldo (gloria.beraldo@dei.unipd.it)

Department of Information Engineering, University of Padova

Topics:

- Memory hierarchy and access time
- Locality of reference
- Types of cache memories and their usage
- Virtual memory

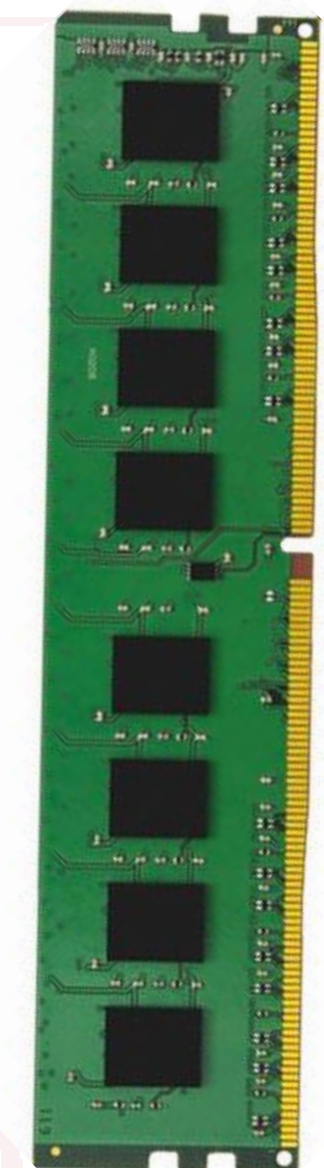
Book reference:

- Chapter 12



Memory Hierarchy (1)

- The main features of a memory are:
 - Size
 - Access time
 - Cost per bit
- There are some trades off:
 - Less access time, higher cost per bit (SRAM)
 - Higher size, less cost per bit (DRAM, SDRAM)
 - Higher size, higher access time (DRAM, SDRAM)



Memory Hierarchy (2)

- RAM memories can have a big size and low cost per bit, but they are very slow!
- The data transfer rate from the memory to the processor is much lower than the data processing speed of the CPU.
- Many computations are slowed down by the memory access and not by the CPU
 - The CPU often has to wait for the incoming data from the memory!
- The gap is unlikely to be closed technologically

CPU vs. DRAM memory

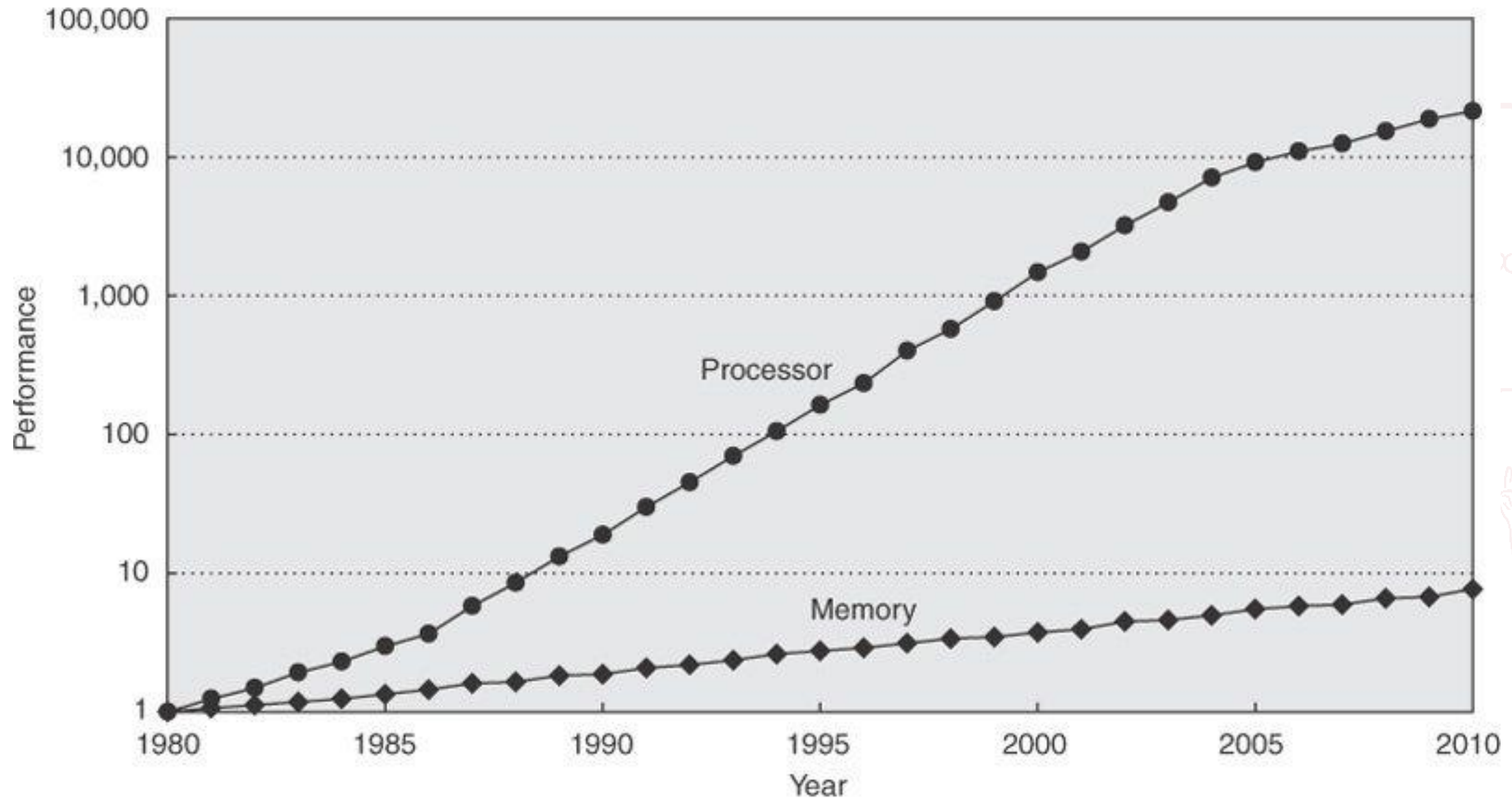


Image from Hennessy and Patterson, Computer Architecture, 2012

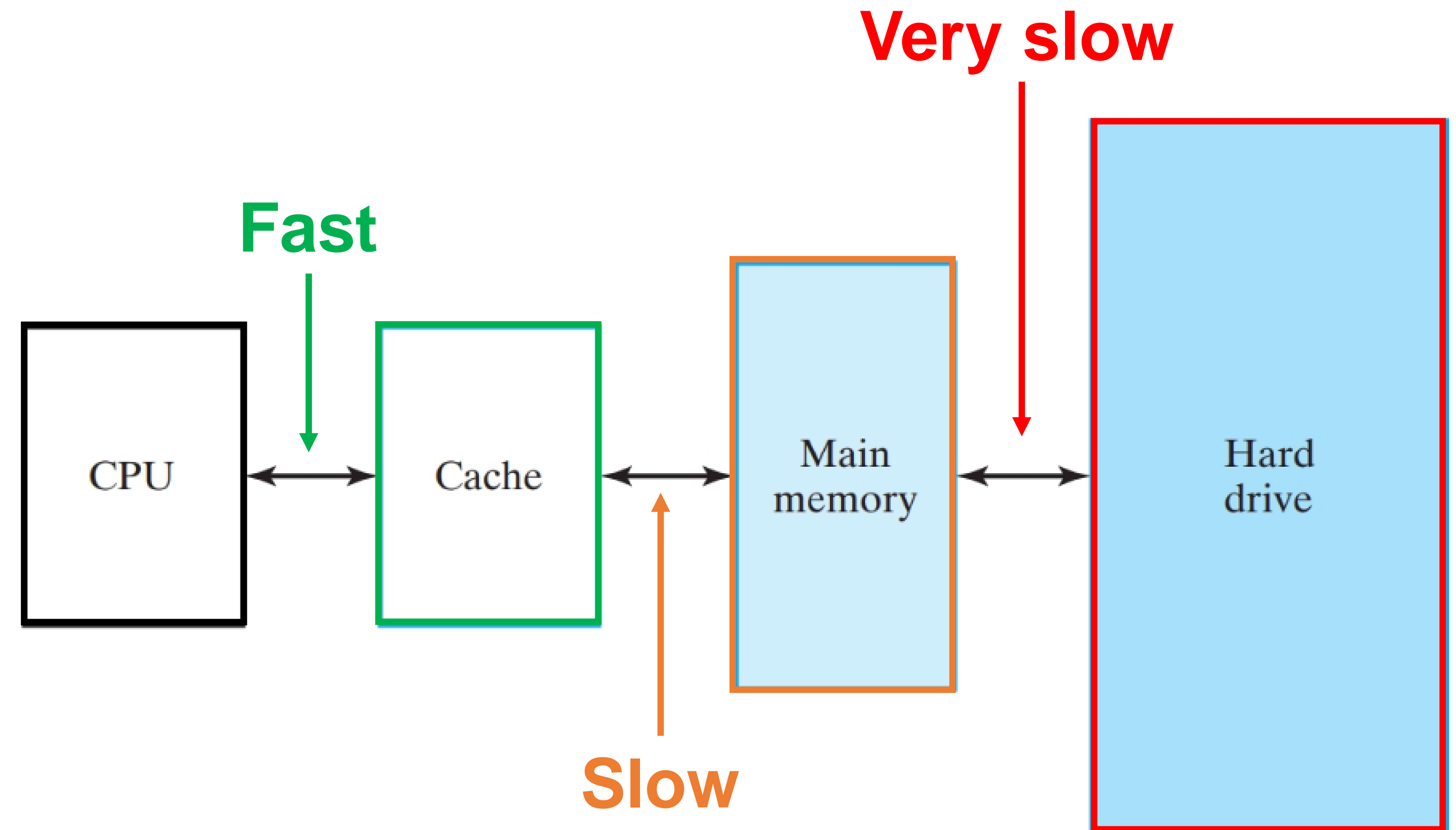
Memory Hierarchy (3)

How can this gap be reduced?

- There are faster memories than RAM (e.g. 7 ns rather than 70 ns) but they are expensive and with a small size (e.g. 512 Kbyte).
- It is not possible to create a large, fast and cheap memory, but it is possible to design a **memory hierarchy** that satisfies all the requirements.

Memory Hierarchy (4)

- **CPU**: high processing speed
- **Cache**: between CPU and RAM
 - It is very fast compared to RAM memory
 - It has limited extension and is expensive (in € and in area)
- **Main memory**: RAM, high capacity, low speed
- **Hard drive**: I/O module

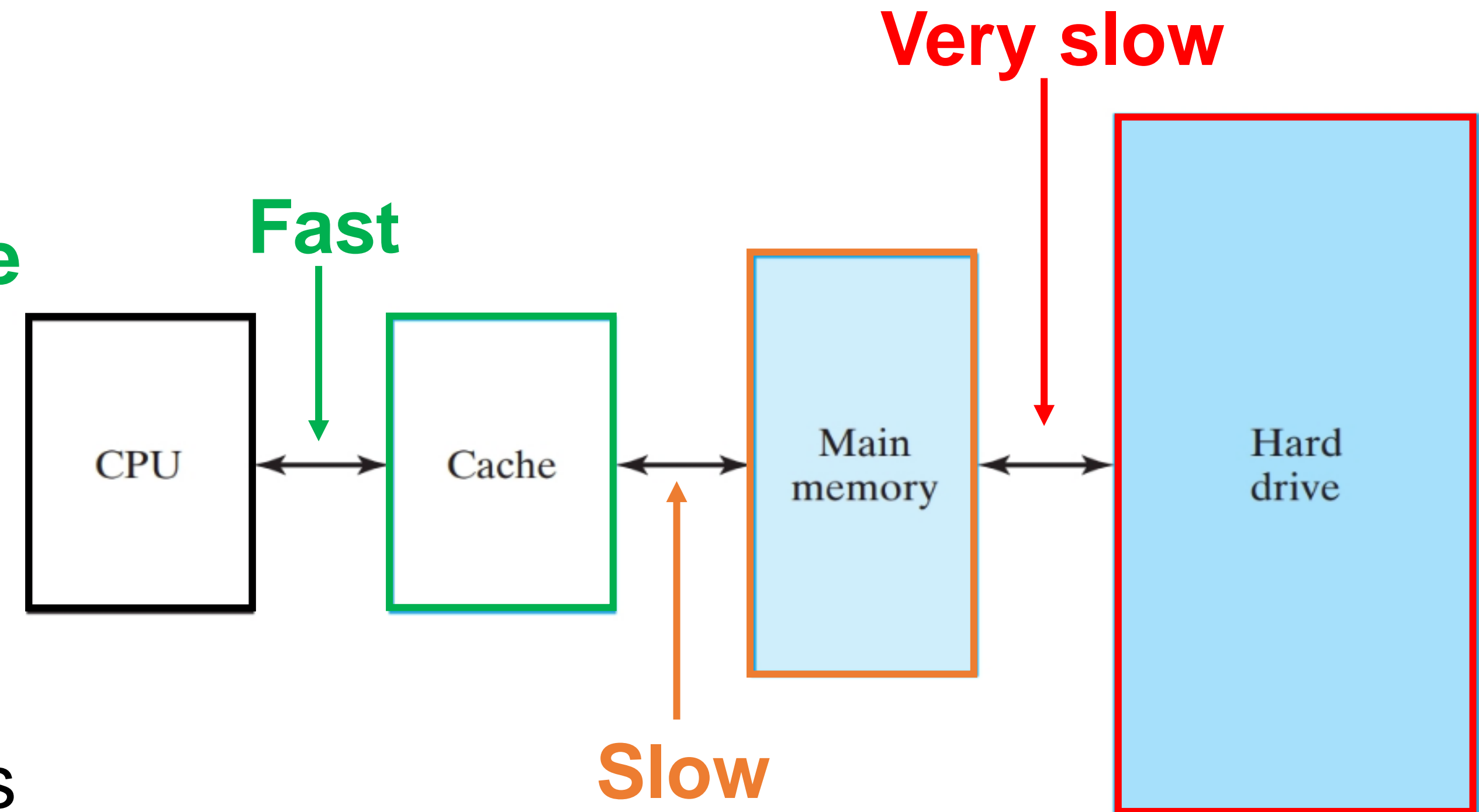


□ **FIGURE 12-1**
Memory Hierarchy

Memory Hierarchy (5)

How the hierarchy works:

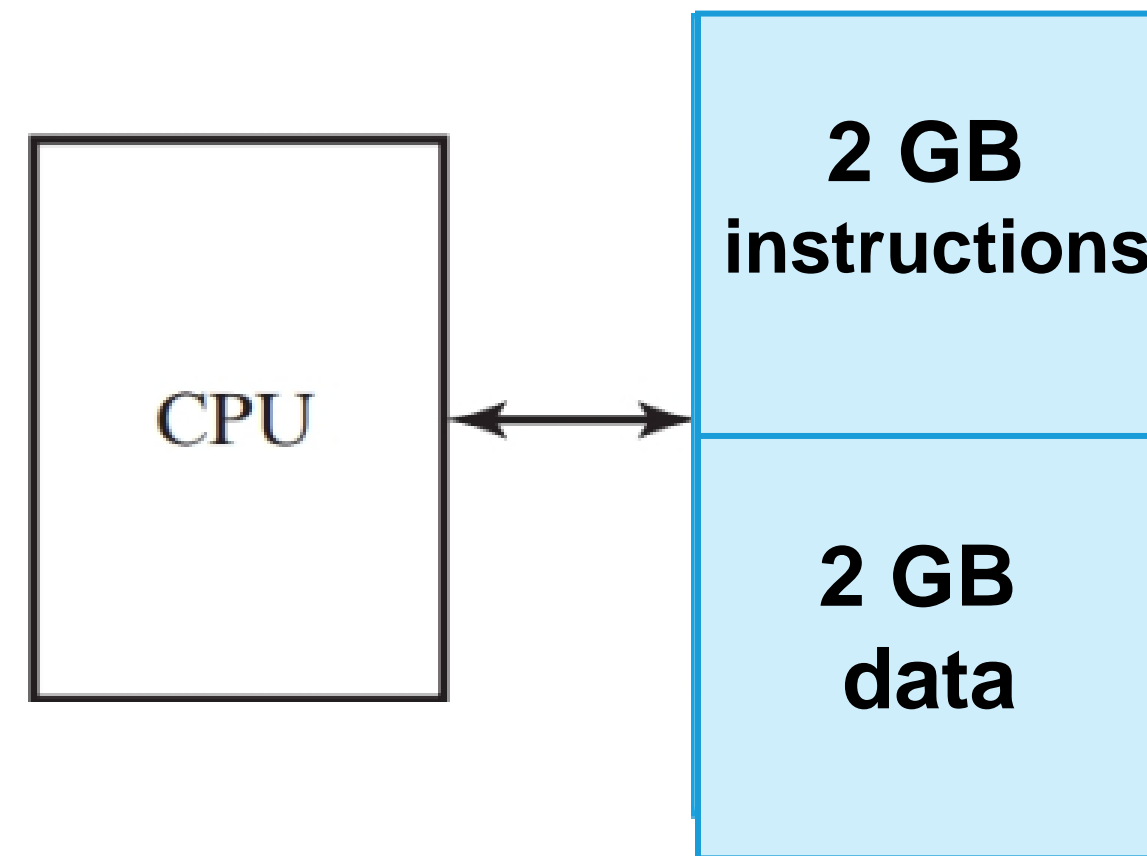
- Most instructions and operands for the **CPU** are expected to be from the **Cache**
- The **Main Memory** serves directly the instructions which are not satisfied by the **Cache**
- The **Hard Drive** is accessed only in the very infrequent cases where instructions are not found in **Main Memory**
- The CPU fetches most instructions from the cache (i.e., fast memory)
- Average data access time \approx cache access time



□ **FIGURE 12-1**
Memory Hierarchy

Example related to cache memory access

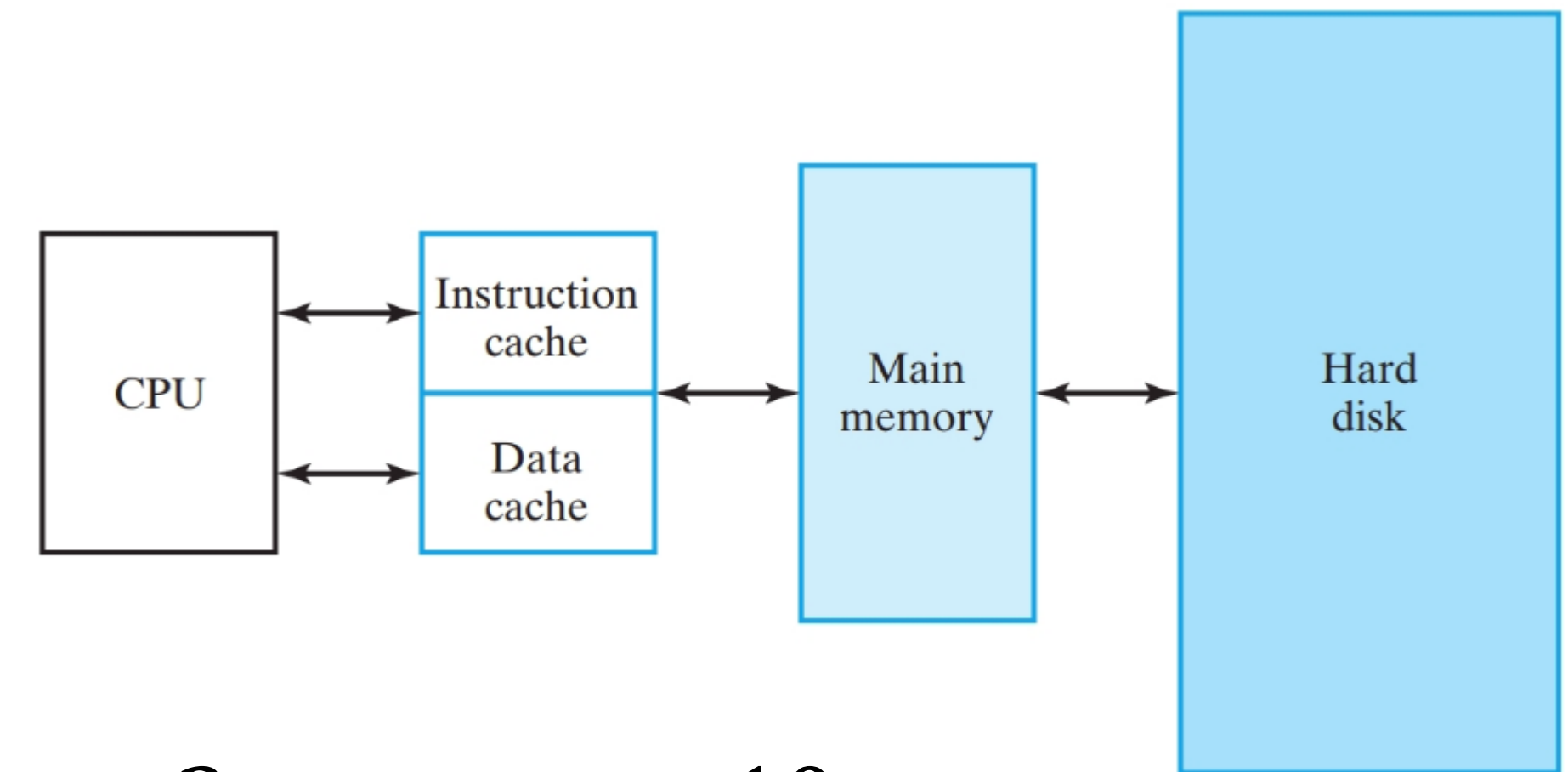
Direct access to RAM



- CPU \leftrightarrow 4GB x 32 RAM
- $CLK_{CPU} = 1GHz \rightarrow T_{CPU} = 1 ns$
- 2 2-GB modules: for instructions and data
- Access time : $t_{RAM} = 10 ns$
- $t_{RAM} \gg T_{CPU}$

Too slow and too costly

Access via the memory hierarchy

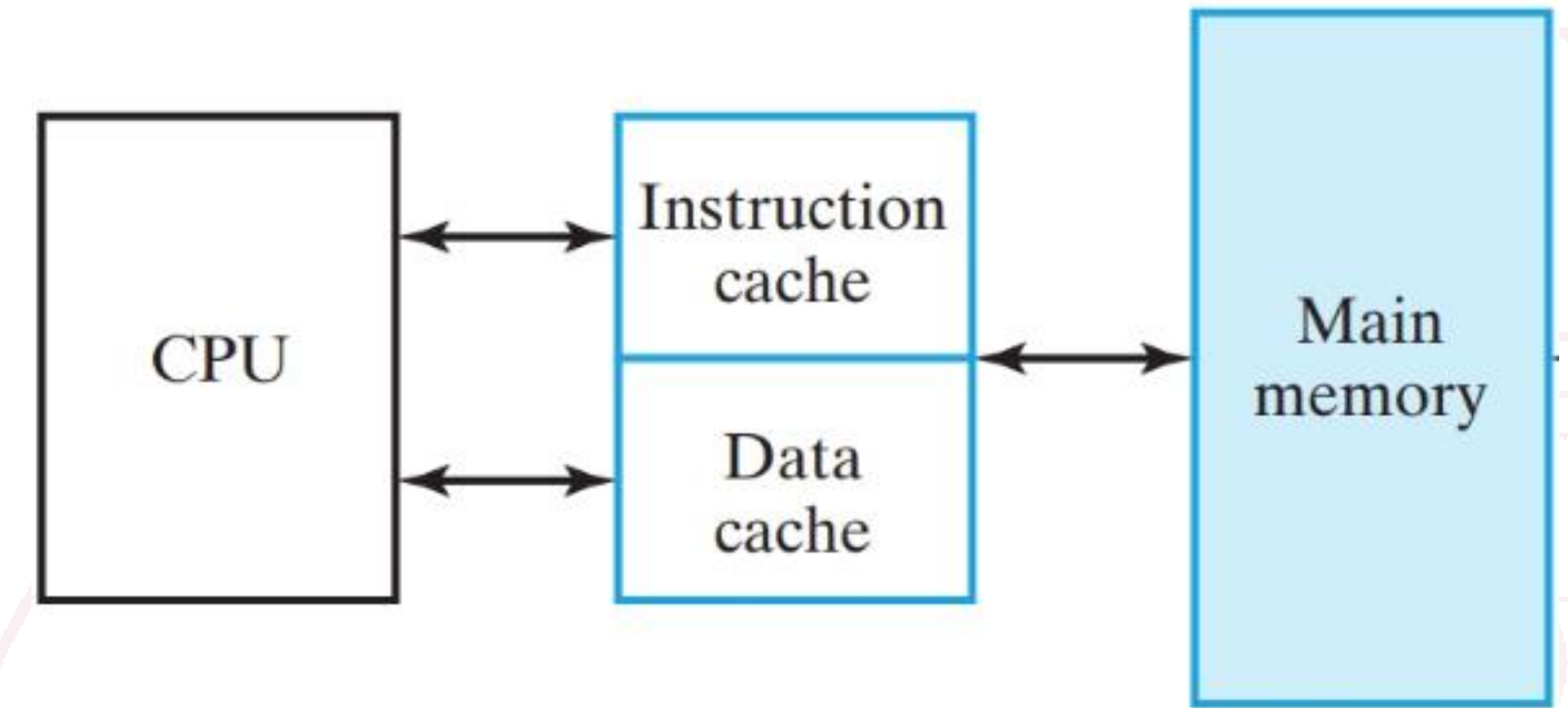


- $t_{Cache} = 2 ns, t_{RAM} = 10 ns$
- 95% of the memory accesses in cache, 5% in RAM
- $0.95 \cdot 2 ns + 0.05 \cdot 10 ns = 2.4 ns \approx T_{CPU}$
- $t_{HD} = 13 ms = 1.3 \cdot 10^7 ns$
- 95% in cache, 4.999995% RAM, 0.000005 HD
- $0.95 \cdot 2 ns + 0.04999995 \cdot 10 ns + 5 \cdot 10^{-8} \cdot 1.3 \cdot 10^7 ns = 3.05 ns \approx T_{CPU}$

Memory Average Access Time (1)

- T_1 : access time in cache
- T_2 : access time for main memory
- $T_1 \ll T_2$
- p = percentage of accesses in cache (**hit rate**)
- Average access time T :

$$T = T_1 p + T_2 (1-p)$$



Memory Average Access Time (2)

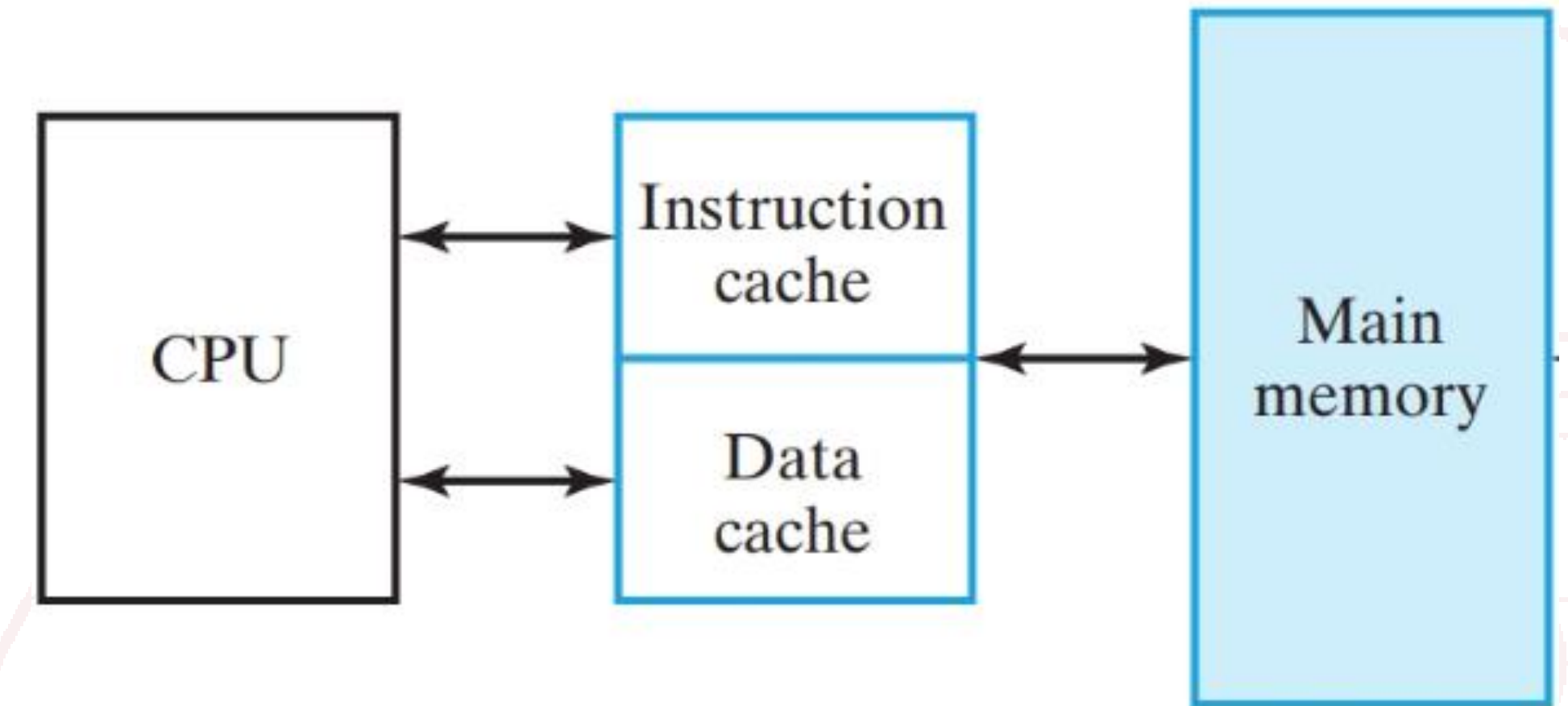
Exercise:

- Cache hit: $p = 95\%$
- Access time for cache $T_1 = 0.01 \mu s$
- Access time for main memory $T_2 = 0.1 \mu s$

- Average access time

$$T = T_1 p + T_2 (1-p) =$$

$$= 0.95 * 0.01 \mu s + 0.05 * 0.1 \mu s = 0.0145 \mu s$$



When does the memory hierarchy work?

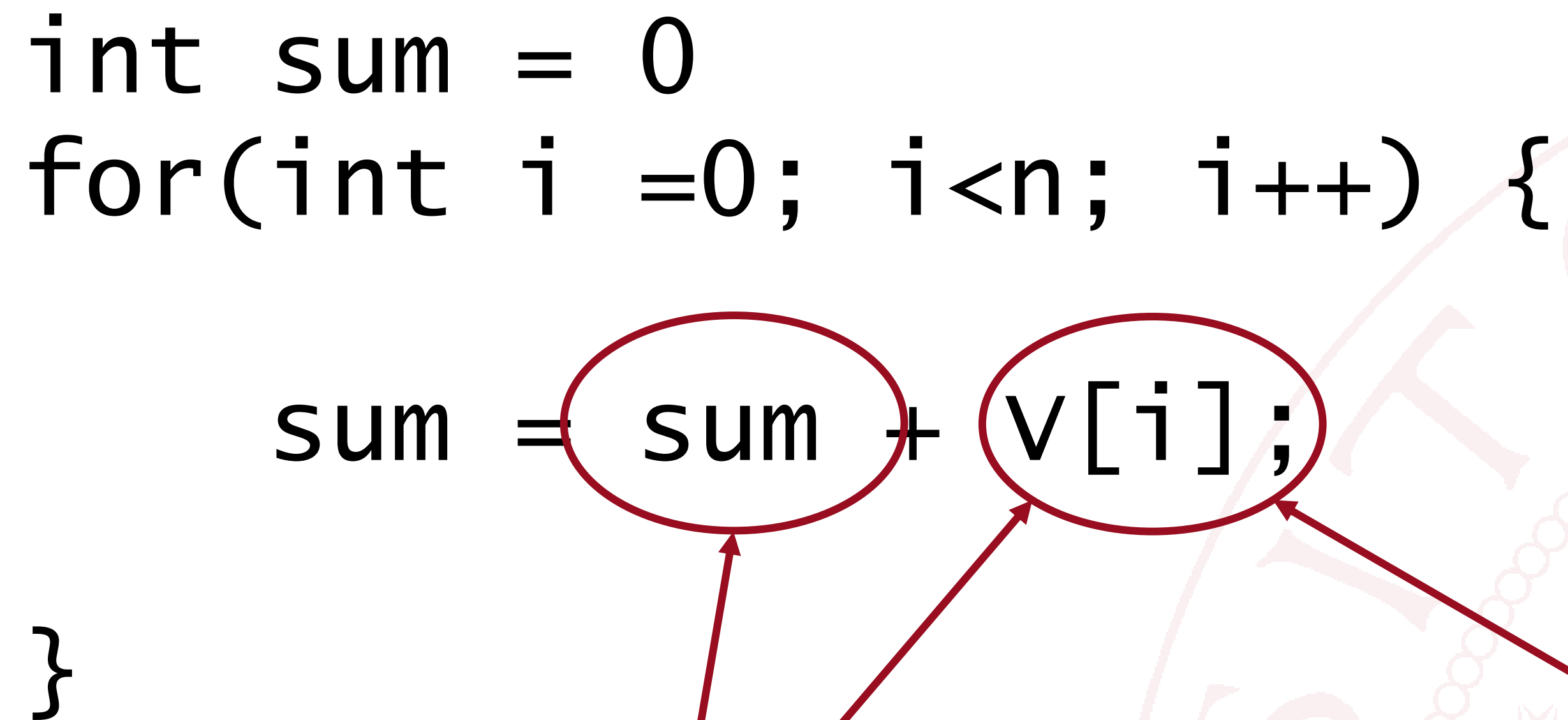
The success of the memory hierarchy is based on assumption that the necessary information should be available in cache:

- If the word W is there (**cache hit**), the word is provided to the processor in a few clock cycles
- If the word is not there (**cache miss**), a consecutive word block containing W is loaded into the cache and the word is provided to the processor
- When the assumption is true:
 - The processor frequently reuses the same data
→ **Temporal locality**
 - The processor requests «close» data in memory in a short period of time → **Spatial locality**

Examples related to locality (1)

Sum of the n elements contained in a vector V :

```
int sum = 0
for(int i = 0; i < n; i++) {
    sum = sum + v[i];
}
```



Their addresses are reused in a short period of time

Temporal locality: the variables *sum* and *i* are used at each iteration

Spatial locality: the vector V is read position by position

We assume that the addresses of the instructions will be in sequential order

Temporal locality

- The instructions can be fetched from the caches, which provides a faster access.
- The least used data are stored in the main memory: access to the them takes more time, but they are rarely requested
- Problems:
 - The cache size is limited
 - It is not known which data will be used most frequently in the future

Example for loop:

- At the first access, the variables **sum** and **i** are loaded in cache from the RAM
- The next access will exploit the cache to retrieve data

Spatial locality

- «Nearby» data in memory means data with nearby addresses
- When accessing to the memory for fetching data at **the address W , the nearby words are loaded too** (e.g., the ones at the addresses $W, W+1, W+2, \dots, W+K$)
- **Block transfer:** a block of K words is loaded

Example for loop:

- At the first access, the first element in the vector V and the «nearby» elements are loaded into the cache
- The next access will exploit the cache to retrieve data

Examples related to locality (2)

Sum of the n elements contained in a vector V :

```
int sum = 0
for(int i =0; i<n; i++) {
    sum = sum + v[i];
}
```

The algorithm takes (approximately):

- **n accesses to the cache** to retrieve the elements in V
- **n/B accesses** to the memory to retrieve the n/B blocks containing V
- Without cache, the algorithm will takes n access in memory

Kinds of Memories cache

- An optimized cache memory aims at:
 - Maximizing the **cache hit**
 - Minimizing the **cache miss**
- This is possible by exploiting the **locality**
- In other words, the cache aims at storing the instructions/data that are estimated to take more time in the next operations
- To optimize this, there are different kinds of memories cache:
 - **Direct-mapped** cache
 - **Fully associative** cache
 - **K-way set-associative** cache

Step back: memory cells, word and bytes (1)

Before introducing the different kinds of memories cache, we recall the basic concepts

Address	Data
000000000	AABBCCDD
0000000100	
0000001000	
0000001100	
0000010000	
0000010100	
0000011000	
0000011100	
	⋮
1111100000	
1111100100	
1111101000	
1111101100	
1111110000	
1111110100	
1111111000	
1111111100	

- A memory is divided in **cells**
- Each cell contains **L** bits
 - Typically $L = 8 \text{ bits} = 1 \text{ byte}$, but in some cases $L = 32 \text{ bits} = 4 \text{ bytes}$
- When $L=32$ bits, **the word in memory is composed of 4 bytes**

Example:

- 1KB memory with 32 bits per cell
- 1 word = 4 bytes = 32 bits
- The memory will contain: $\frac{1024 \text{ bytes}}{4 \text{ bytes}} = 256 \text{ words}$

Step back: memory cells, word and bytes (2)

Before introducing the different kinds of memories cache, we recall the basic concepts

Address	Data
0000000000	
0000000100	
0000001000	
0000001100	
0000010000	
0000010100	
0000011000	
0000011100	
⋮	
1111100000	
1111100100	
1111101000	
1111101100	
1111110000	
1111110100	
1111111000	
1111111100	

- Each memory cell is identified by an **address**
- Given an address of **n bit**, a total of **2^n cells** is possible to address (note: $2^{10} = 1024 = 1\text{K}$, $2^{20} = 1\text{M}$, $2^{30} = 1\text{G}$)

Example:

- 1KB memory with 32 bits per cell
- 1 word = 4 bytes = 32 bits
- The memory will contain: $\frac{1024 \text{ bytes}}{4 \text{ bytes}} = 256 \text{ words} = 2^8$
- To address all the bytes in memory, we need an address with length of $\log_2 1024 = 10 \text{ bits}$
- However, the last 2 bits of the address will identify the 4 bytes of the word
- Since the word is the unit, the last 2 bits are irrelevant (and we always consider them equal to 00)

Direct mapped cache

Given a 32 bits **cache memory** with 8 words (4 bytes each) and a 1K **RAM memory** (256 words)

The concept behind the direct mapped cache is to directly map **groups** of cells of **RAM** with consecutive addresses inside the **cache** by using the same **index**

- In our case, the cache can contain 8 words of the RAM memory
- The first 5 bits of the address identify the corresponding **group (TAG)**
- The next 3 bits of the address identify the cell (**INDEX**) inside the cache
- There is a **direct link** among the bits 4, 3, 2 of the address and the index in the cache
- Example: the data at the address **0000001100** will be at the cell with **INDEX 011** in the cache

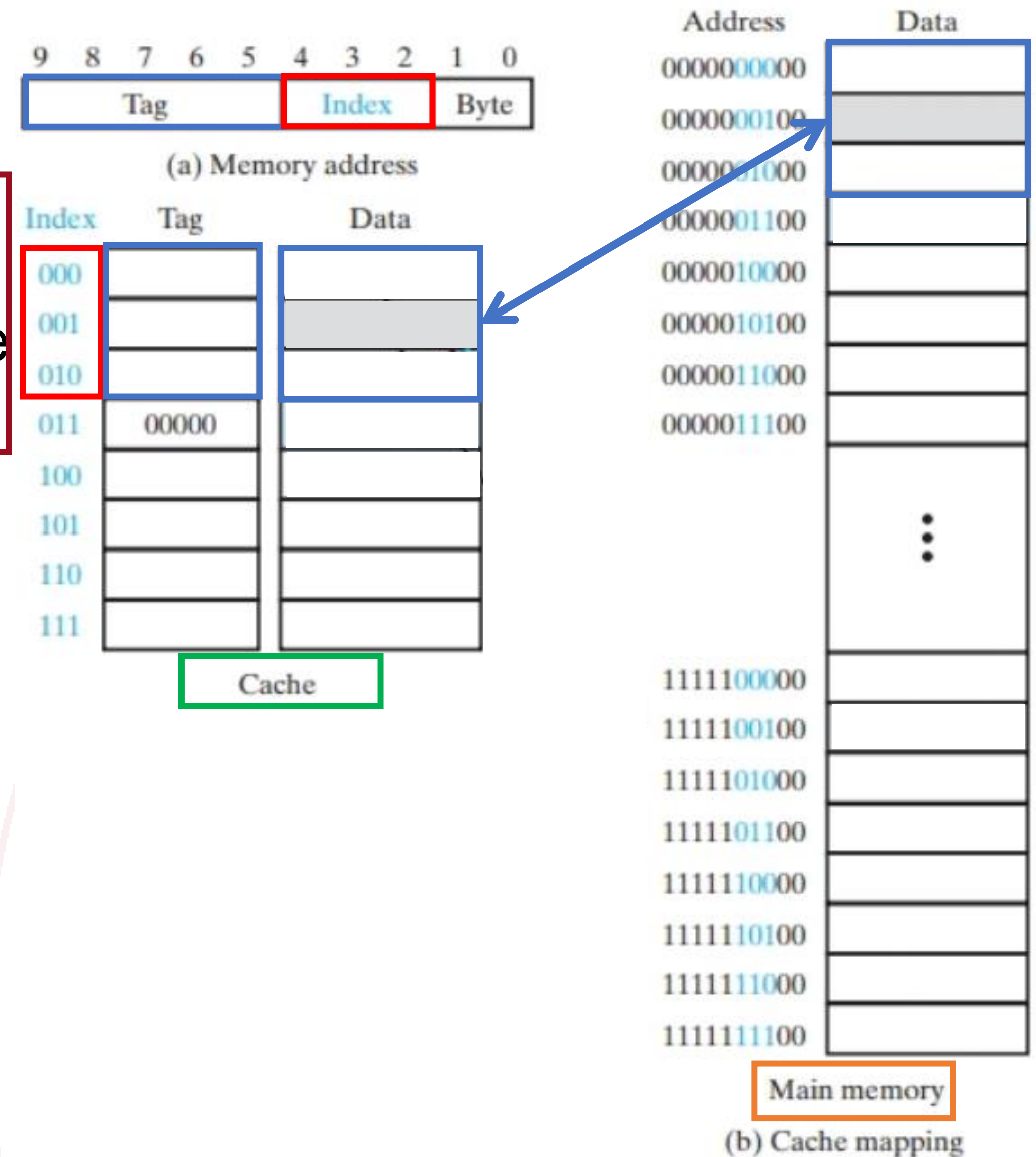
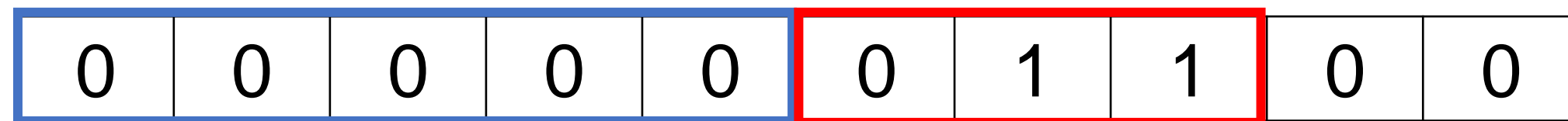


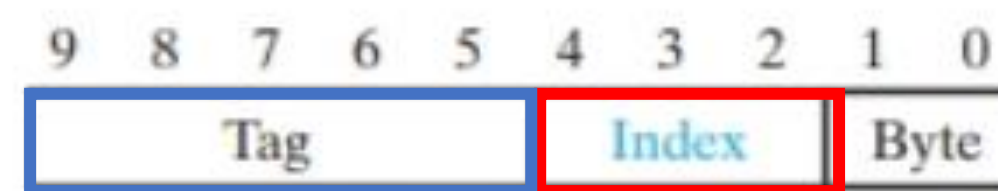
FIGURE 12-3
Direct Mapped Cache

Direct mapped cache: functioning

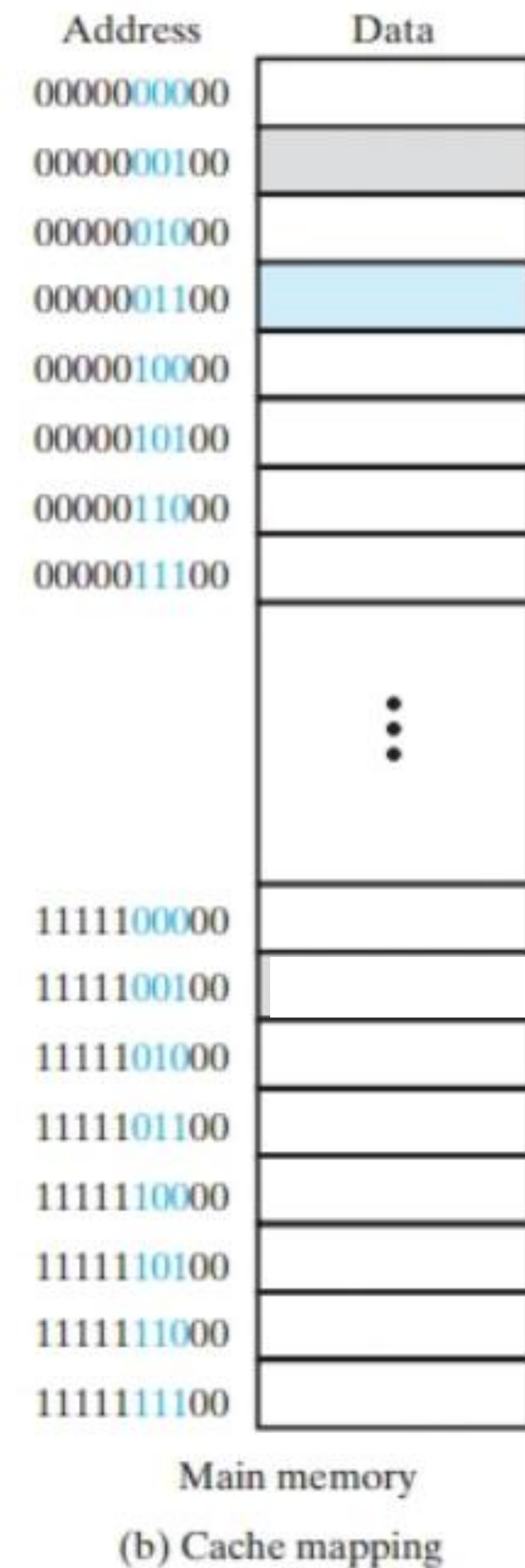
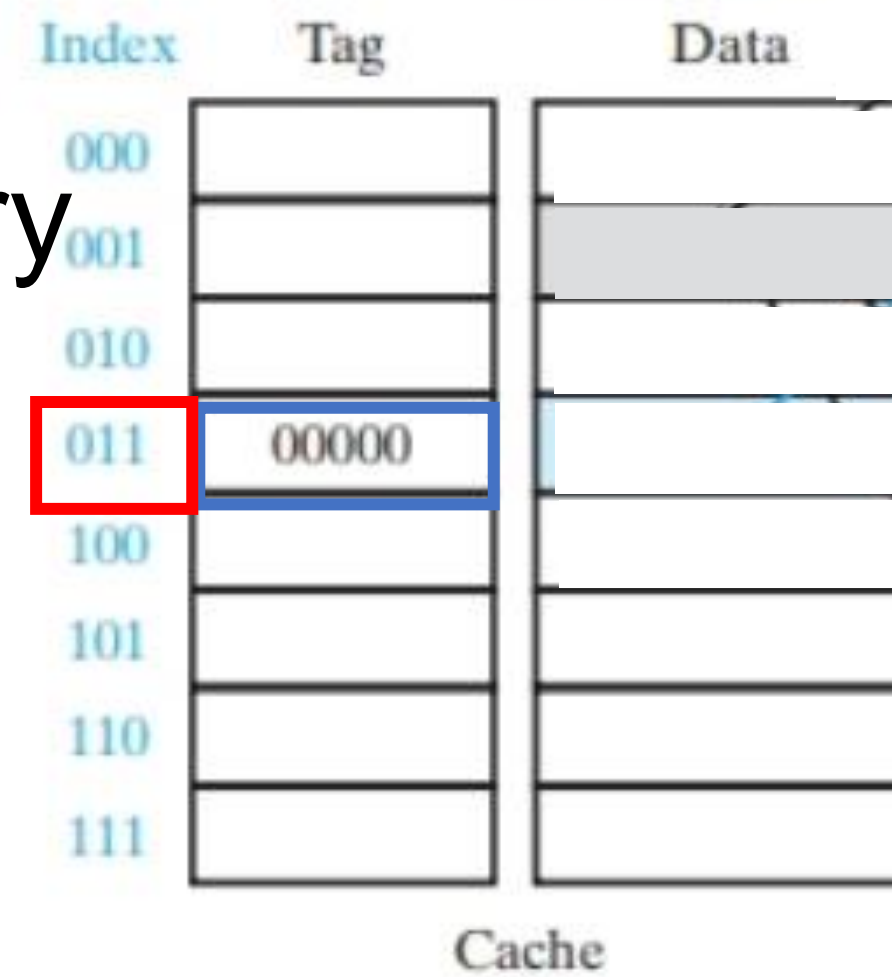
- The CPU will fetch the instruction with the address:



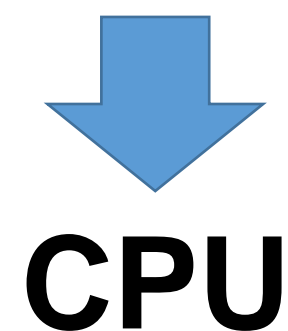
- The instruction can be stored in cache or in memory
- The cache divide the **TAG** and the **INDEX** (00000 e 011)
- Retrieve the line in the cache related to the **INDEX**
- Compare the **TAG** achieved in the previous step with the one in the instruction



(a) Memory address



It is the same
The retrieve data is
the expected one
Cache Hit



CPU

It is different
The data is not
available in cache
Cache Miss



Request to memory

A new block of data
is loaded into the
cache memory

FIGURE 12-3
Direct Mapped Cache

Direct mapped cache: pro and cons

- **Pro:**
 - Simple organization
- **Cons:**
 - The line to be overwritten is predetermined: a block cannot be inserted in a line chosen from those that are probably no longer needed
 - Low Hit rate (less efficiency).

Example: The CPU requires the access to the following addresses, at the beginning, the cache is empty

Indirizzi	
1	00000001000
2	00100001000
3	0100101000
4	1001001000

INDEX	TAG	DATA
000		
001		
010	00000	
011		
100		
101		
110		
111		

1. Load the data at the index **010**
2. **TAG** different, delete the data at the **index 010**, and load the new data with **TAG 00100** (also with the free lines)
3. **TAG** different, delete the data at the **index 010**, and load the new data with **TAG 01001** (also with the free lines)
4. **TAG** different, delete the data with **index 010**, and load the data with **TAG 10010** (also with the free lines)

Cache entry: number of words

- It has been assumed that each cache entry includes a **TAG** and a **WORD**
- In real scenarios, the spatial locality is exploited and more than one **WORD** is included in a cache entry
- So when a cache miss happens, a block of L **WORDS**, called **LINE**, is loaded into the cache

Example:

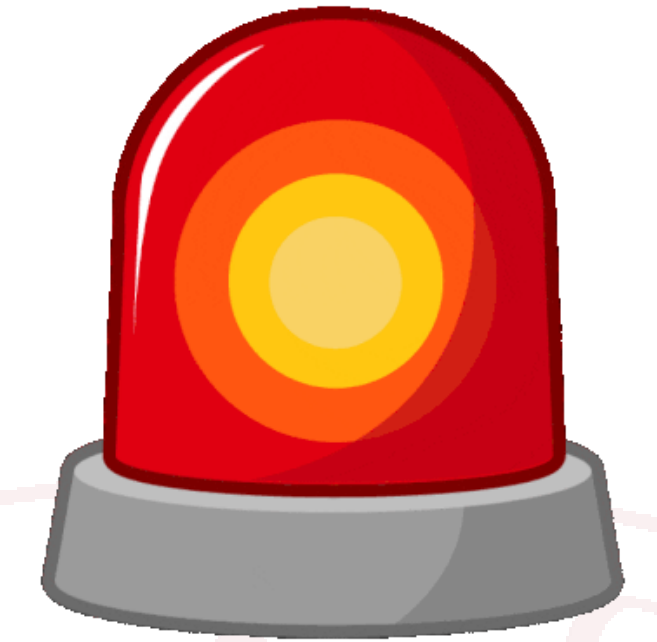
- A **LINE** is composed of 4 **WORDS**
- Cache miss for the address **0000 01** 0000
- All the **WORDS** from «000001 0000» to «000001 1100» are simultaneously loaded in the cache block identified from the **INDEX 01** and from the **TAG 0000**

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via 16-bit words;
- There is a direct mapped cache with 2 lines connected only to the data memory. Each line contains 8 bytes.



YOU ARE ASKED TO DESIGN THE CACHE DESCRIBED ABOVE

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via **16-bit words**;
- There is a direct mapped cache with **2 lines** connected only to the data memory. Each line contains **8 bytes**.

- $\frac{8*8 \text{ bit}}{16 \text{ bit}} = 4 \text{ words per line}$
- *2 lines*



Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via **16-bit words**;
- There is a direct mapped cache with **2 lines** connected only to the data memory.

Each line contains **8 bytes**.

THE DATA IN THE MAIN MEMORY ARE PROVIDED

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via 16-bit words;
- There is a direct mapped cache with 2 lines connected only to the data memory.

Each line contains 8 bytes.

The cache described above is composed of **2 lines**. Each line contains **4 words**, we can divide the memory into the following **blocks**:



C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
 - 16 bits address lines;
 - A 16 KB SDRAM for instructions;
 - A 32 KB SDRAM for data;
 - Memories are addressed via 16-bit words;
 - There is a direct mapped cache with 2 lines connected only to the data memory.
- Each line contains 8 bytes.

THE DATA IN **M[3]** HAS TO BE READ

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via 16-bit words;
- There is a direct mapped cache with 2 lines connected only to the data memory.
Each line contains 8 bytes.

The cache is empty, we have a **MISS**, the corresponding **block is copied in the cache**

	4	3	8	2

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
 - 16 bits address lines;
 - A 16 KB SDRAM for instructions;
 - A 32 KB SDRAM for data;
 - Memories are addressed via 16-bit words;
 - There is a direct mapped cache with 2 lines connected only to the data memory.
- Each line contains 8 bytes.

THE DATA IN **M[1]** HAS TO BE READ

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via 16-bit words;
- There is a direct mapped cache with 2 lines connected only to the data memory. Each line contains 8 bytes.

The cache contains the data we need, we have a **HIT**

	4	3	8	2

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
 - 16 bits address lines;
 - A 16 KB SDRAM for instructions;
 - A 32 KB SDRAM for data;
 - Memories are addressed via 16-bit words;
 - There is a direct mapped cache with 2 lines connected only to the data memory.
- Each line contains 8 bytes.

THE DATA IN **M[4]** HAS TO BE READ

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
- 16 bits address lines;
- A 16 KB SDRAM for instructions;
- A 32 KB SDRAM for data;
- Memories are addressed via 16-bit words;
- There is a direct mapped cache with 2 lines connected only to the data memory. Each line contains 8 bytes.

The data is not in the cache, we have a **MISS**, the corresponding block is copied in the cache

	4	3	8	2
	1	3	2	8

C) RAM 16 bits data	
0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
 - 16 bits address lines;
 - A 16 KB SDRAM for instructions;
 - A 32 KB SDRAM for data;
 - Memories are addressed via 16-bit words;
 - There is a direct mapped cache with 2 lines connected only to the data memory.
- Each line contains 8 bytes.

THE DATA IN **M[9]** HAS TO BE READ

C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Example from previous exam

Description:

A single-cycle computer has the following specifications:

- Clock equal to 2 MHz;
 - 16 bits address lines;
 - A 16 KB SDRAM for instructions;
 - A 32 KB SDRAM for data;
 - Memories are addressed via 16-bit words;
 - There is a direct mapped cache with 2 lines connected only to the data memory.
- Each line contains 8 bytes.

The data is not in the cache, we have a **REPLACEMENT**,
the corresponding block is copied in the cache

	10	2	2	8
	1	3	2	8

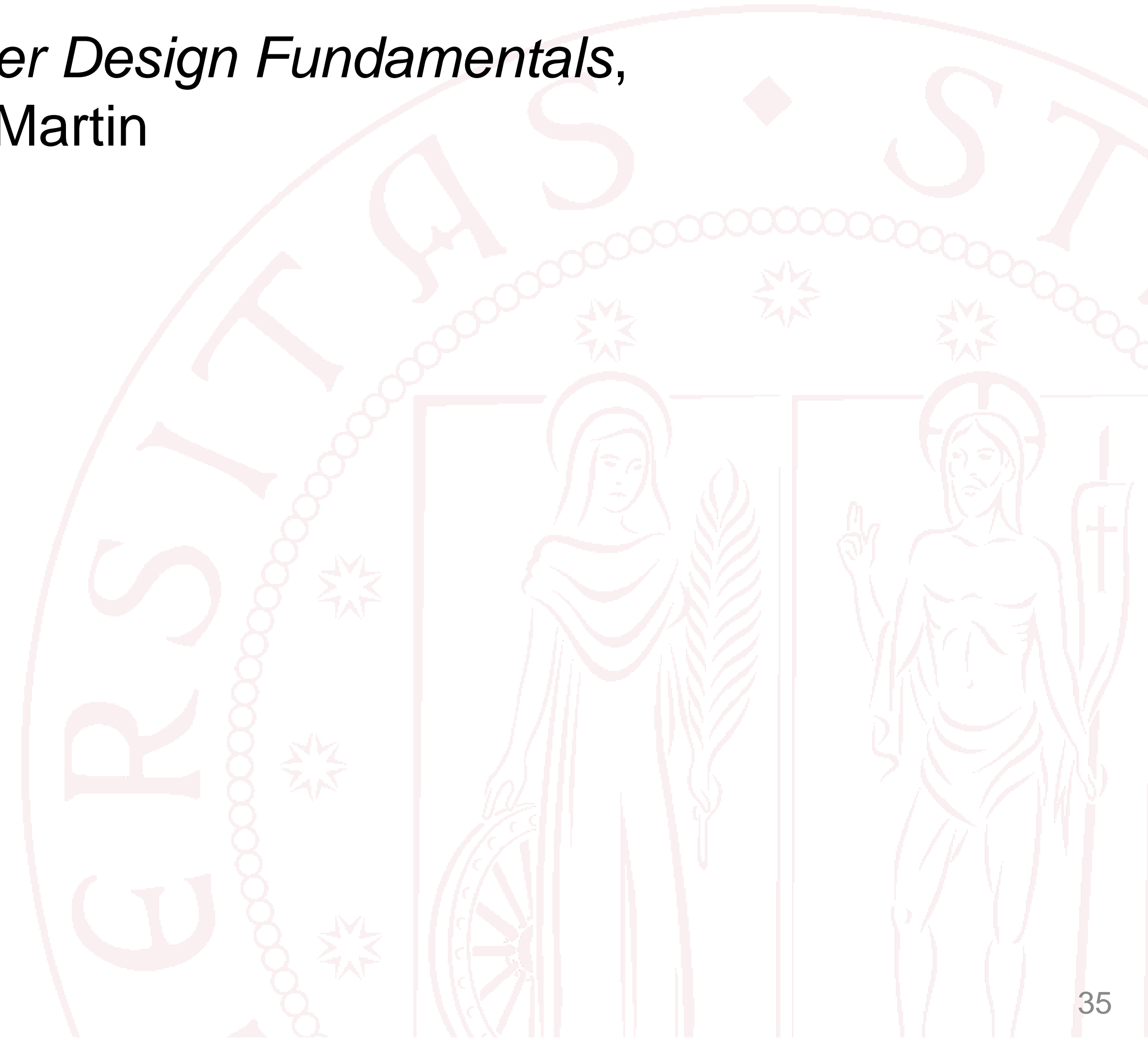
C) RAM 16 bits data

0x0000	4
0x0001	3
0x0002	8
0x0003	2
0x0004	1
0x0005	3
0x0006	2
0x0007	8
0x0008	10
0x0009	2
0x000A	2
0x000B	8
0x000C	11
0x000D	18

Disclaimer

Figures from *Logic and Computer Design Fundamentals*,
Fifth Edition, GE Mano | Kime | Martin

© 2016 Pearson Education, Ltd



Questions

