



# Exercise 1

Given a computer with virtual memory and cache with the following characteristics:

- 20-bit Virtual Addresses generated by the processor.
- 128KB Physical Main Memory with 32KB pages. LRU Replacement.
- 1KB Physical Cache Memory. Direct-mapped and with a block size of 256B.

On this system we execute an application that randomly selects a song from a data base of songs. Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

- 1) Obtain the virtual and physical address format (from the point of view of the virtual memory and the cache memory).
- 2) Calculate in which virtual page (or pages) each song is included.
- 3) Given the following contents for the TLB and the cache memory. Which have been the latest played songs? Which is their physical address range?

TLB	Virtual page	Physical page	CACHE	Tag	Block
	0x0A	1		0x70	0
	0x0C	2		0x27	1
	0x02	0		0x27	2
	0x1E	3		0x27	3

- 4) The following sequence of virtual references is generated: 0xF40F5, 0x66000, 0x40040 y 0x51D04. Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

# Exercise 1



Given a computer with virtual memory and cache with the following characteristics:

- 20-bit Virtual Addresses generated by the processor.
- 128KB Physical Main Memory with 32KB pages. LRU Replacement.
- 1KB Physical Cache Memory. Direct-mapped and with a block size of 256B.

1) Obtain the virtual and physical address formats (both for the memory and for the cache).

1) Obtain the virtual and physical address formats (both for the memory and for the cache).

- 20-bit Virtual Addresses generated by the processor.
- 128KB Physical Main Memory with 32KB pages. LRU Replacement.
- 1KB Physical Cache Memory. Direct-mapped and with a block size of 256B.

**Virtual address: 20 bits**

**Page size: 32 KB =  $2^{15}$  bytes**

#### Fields of the virtual address

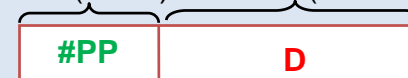
Virtual page number (5 bits)      byte/page (15 bits)



**Memory: 128 KB =  $2^{17}$  bytes**

#### Fields of the physical address

Physical page number (2 bits)      byte/page (15 bits)



7      2      8



$$\frac{1 \text{ KB}}{256 \text{ B/block}} = 4 \text{ blocks}$$

# Exercise 1



On this system we execute an application that randomly selects a song from a data base of songs. Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

2) Calculate in which virtual page (or pages) each song is included.



## 2) Calculate in which virtual page (or pages) each song is included.

Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

**First: compute the virtual address included in each song**

**Song: 8 KB =  $2^{13}$  bytes = 0b10 0000 0000 0000 = 0x2000**

Virtual address range

0x50000

0x51FFF

0x52000

Song 1

Virtual address range

0101 0000 0000 0000 0000

0101 0001 1111 1111 1111

**Fields of the virtual address**

Virtual page number (5 bits)    byte/page (15 bits)

#VP

D



## 2) Calculate in which virtual page (or pages) each song is included.

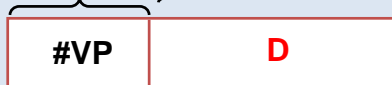
Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

**Second: compute the virtual pages for that range**

### Fields of the virtual address

Virtual page number (5 bits)    byte/page (15 bits)



Song	Virtual address range	Virtual pages
1	0101 0000 0000 0000 0000 - 0101 0001 1111 1111 1111	0x0A

000 0101 0  
0    A



## 2) Calculate in which virtual page (or pages) each song is included.

Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

### Fields of the virtual address

Virtual page number (5 bits)      byte/page (15 bits)



**Song: 8 KB =  $2^{13}$  bytes =**

**0b10 0000 0000 0000 = 0x2000**

Song	Virtual address range	Virtual pages
1	0101 0000 0000 0000 0000 - 0101 0001 1111 1111 1111	0x0A
2	1111 0100 0000 0000 0000 - 1111 0101 1111 1111 1111	0x1E
3	0110 0110 0000 0000 0000 - 0110 0111 1111 1111 1111	0x0C
4	0001 0010 0000 0000 0000 - 0001 0011 1111 1111 1111	0x02
5	0100 0000 0000 0000 0000 - 0100 0001 1111 1111 1111	0x08
6	0011 0110 0000 0000 0000 - 0011 0111 1111 1111 1111	0x06

# Exercise 1



On this system we execute an application that randomly selects a song from a data base of songs. Songs occupy 8KB and their starting addresses are:

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

- 3) Given the following contents for the TLB and the cache memory. Which have been the latest played songs? Which is their physical address range?

## TLB

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

## CACHE

Tag	Block
0x70	0
0x27	1
0x27	2
0x27	3





3) Given the following contents for the TLB and the cache memory. **Which have been the latest played songs?** Which is their physical address range?

### TLB

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

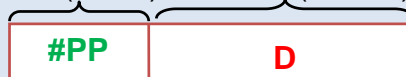
### CACHE

Tag	Block
0x70	0
0x27	1
0x27	2
0x27	3

1. To know the latest played songs we need to look in the cache (it holds latest blocks accessed)

### Fields of the physical address

Physical page number (2 bits)      byte/page (15 bits)



7      2      8



Physical address      P. page

111 0000 <b>00</b> xxxx xxxx	11
010 0111 <b>01</b> xxxx xxxx	01
010 0111 <b>10</b> xxxx xxxx	01
010 0111 <b>11</b> xxxx xxxx	01

Last played songs are stored in physical pages 1 and 3



3) Given the following contents for the TLB and the cache memory. **Which have been the latest played songs?** Which is their physical address range?

2. To know which songs are stored in those physical addresses we need to look in the TLB

### TLB

Virtual page	Physical page
<b>0x0A</b>	<b>1</b>
0x0C	2
0x02	0
<b>0x1E</b>	<b>3</b>

### CACHE

Tag	Block
0x70	0
0x27	1
0x27	2
0x27	3

3. To know which songs are stored in those virtual pages we need to look in the results of section 2

Song	Virtual pages
<b>1</b>	<b>0x0A</b>
<b>2</b>	<b>0x1E</b>
3	0x0C
4	0x02
5	0x08
6	0x06

Last played songs: 1,2



3) Given the following contents for the TLB and the cache memory. Which have been the latest played songs? **Which is their physical address range?**

In section 2 we had the virtual address range, so we just replace virtual page number with physical page number

V. A.: 0101 0000 0000 0000 0000

P. A.: 0 1000 0000 0000 0000

#### Fields of the physical address

Physical page number (2 bits)    byte/page (15 bits)



Song	Virtual page	Physical page	Physical address range
1	0x0A	0x1	0 1000 0000 0000 0000 – 0 1001 1111 1111 1111
2	0x1E	0x3	1 1100 0000 0000 0000 – 1 1101 1111 1111 1111

0x08000  
0x09FFF  
0x1C000  
0x1DFFF

# Exercise 1



- 4) The following sequence of virtual references is generated: 0xF40F5, 0x66000, 0x40040 y 0x51D04. Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

## TLB

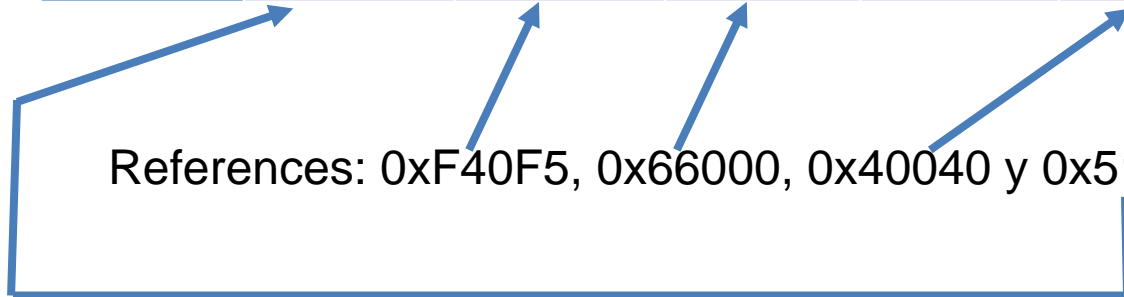
Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

## CACHE

Tag	Block
0x70	0
0x27	1
0x27	2
0x27	3

Song	1	2	3	4	5	6
Address	0x50000	0xF4000	0x66000	0x12000	0x40000	0x36000

References: 0xF40F5, 0x66000, 0x40040 y 0x51D04.



4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

First reference: 0xF40F5

First: look for address translation in TLB

Virtual address: 0xF40F5

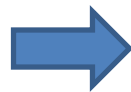
1111 0100 0000 1111 0101

Virtual page number: 0x1E

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3



TLB hit.



1 1100 0000 1111 0101

0x1C0F5 Physical address

We can access the cache.

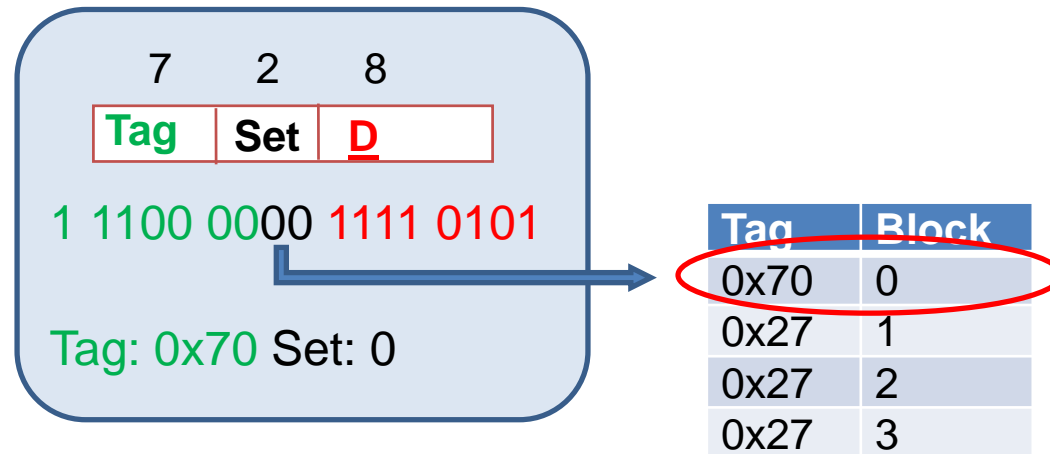


4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

First reference: 0xF40F5

Second: look for address the data in the cache

Physical addr.: 1 1100 0000 1111 0101

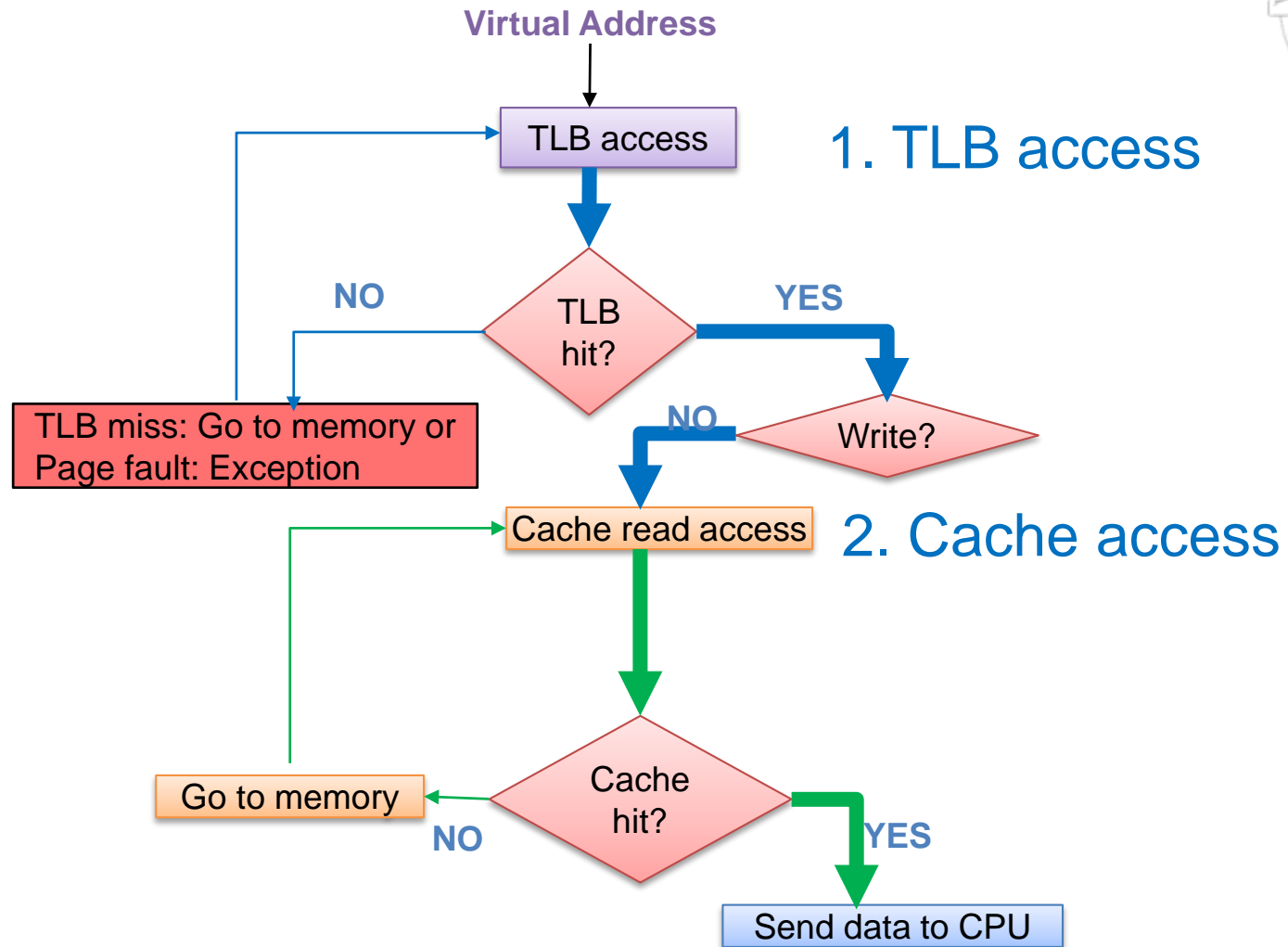


Cache hit: block 0x70 is stored in set 0.

We send the data to the CPU



# Physical cache



4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Second reference: 0x66000

First: look for address translation in TLB

Virtual address: 0x66000

0110 0110 0000 0000 0000

Virtual page number: 0x0C

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3



TLB hit.



1 0110 0000 0000 0000

0x16000 Physical address

We can access the cache.



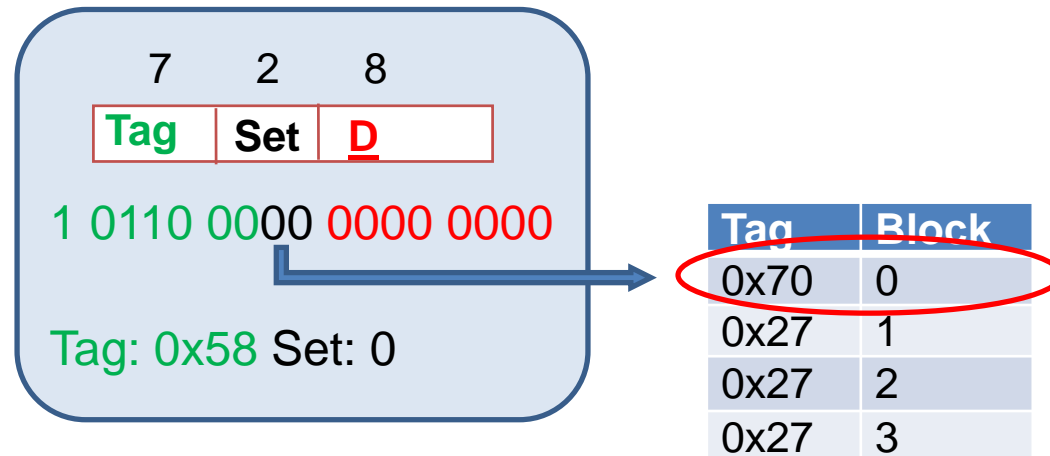


4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Second reference: 0x66000

Second: look for address the data in the cache

Physical addr.: 1 0110 0000 0000 0000



Cache miss: block 0x58 is not stored in set 0.

We need to bring the block from memory

4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Second reference: 0x66000

Third: bring block from memory to the cache

Physical addr.: 0x16000

Tag: 0x58 Set: 0

Size: 256 B

Tag	Block
0x70	0
0x27	1
0x27	2
0x27	3

Tag	Block
0x58	0
0x27	1
0x27	2
0x27	3



Send data  
to CPU

Replacement in direct mapped cache:  
the block in set 0 is replaced

4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

First: look for address translation in TLB

Virtual address: 0x40040

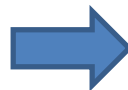
0x40040

0100 0000 0000 0100 0000

Virtual page number: 0x08

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

TLB miss



We need to look for the translation in the Page table.



4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

Second: look for address translation in the Page Table

Virtual page number: 0x08

**We don't have the Page table!**

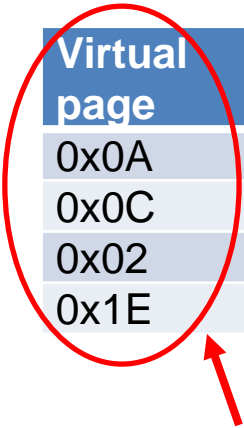


4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

Second: look for address translation in the Page Table

Virtual page number: 0x08



Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

There are only 4 page frames in main memory, so we have all the translations in the TLB.

Only these entries in the page table have  $V = 1$



4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

So, let's fill the page table

Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

Virtual page	Page table	
	V	LRU Physical page
	0	
02	1	0
...	0	
0A	1	1
	0	
0C	1	2
	0	
1E	1	3

Page table miss: virtual page 8 is not in main memory

Exception



Page fault: the OS will bring the page from disk





4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

And what about the LRU bits?

Earlier, in this section we accessed

Second reference

Virtual page number: 0x0C

First reference

Virtual page number: 0x1E

In the previous section we found out that songs 1 and 2 were the most recent ones

Song	Virtual pages	LRU
1	0x0A	1
2	0x1E	

LRU  
3  
2

Virtual page	Page table		
	V	LRU	Physical page
	0		
02	1	0	0
...	0		
0A	1	1	1
	0		
0C	1	3	2
	0		
1E	1	2	3

Virtual page 0x02 will be replaced by virtual page 0x08 in the physical page frame 0.

4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

Third: once the page is in memory we have to update the page table and then we can look for address translation in the Page Table



Virtual page	Page table		
	V	LRU	Physical page
	0		
02	0	0	0
08	1	3	0
0A	1	0	1
	0		
0C	1	2	2
	0		
1E	1	1	3



We need to:  
4. update TLB

Virtual page	Physical page
0x0A	1
0x0C	2
0x08	0
0x1E	3

and  
5. write the block in the cache



4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Third reference: 0x40040

Fifth: bring the block from memory to the cache

Virtual address: 0x40040

0100 0000 0000 0100 0000

Physical address: 0 0000 0000 0100 0000



Tag	Block
0x58	0
0x27	1
0x27	2
0x27	3

Replacement in direct mapped cache:  
the block in set 0 is replaced

Tag	Block
0x00	0
0x27	1
0x27	2
0x27	3

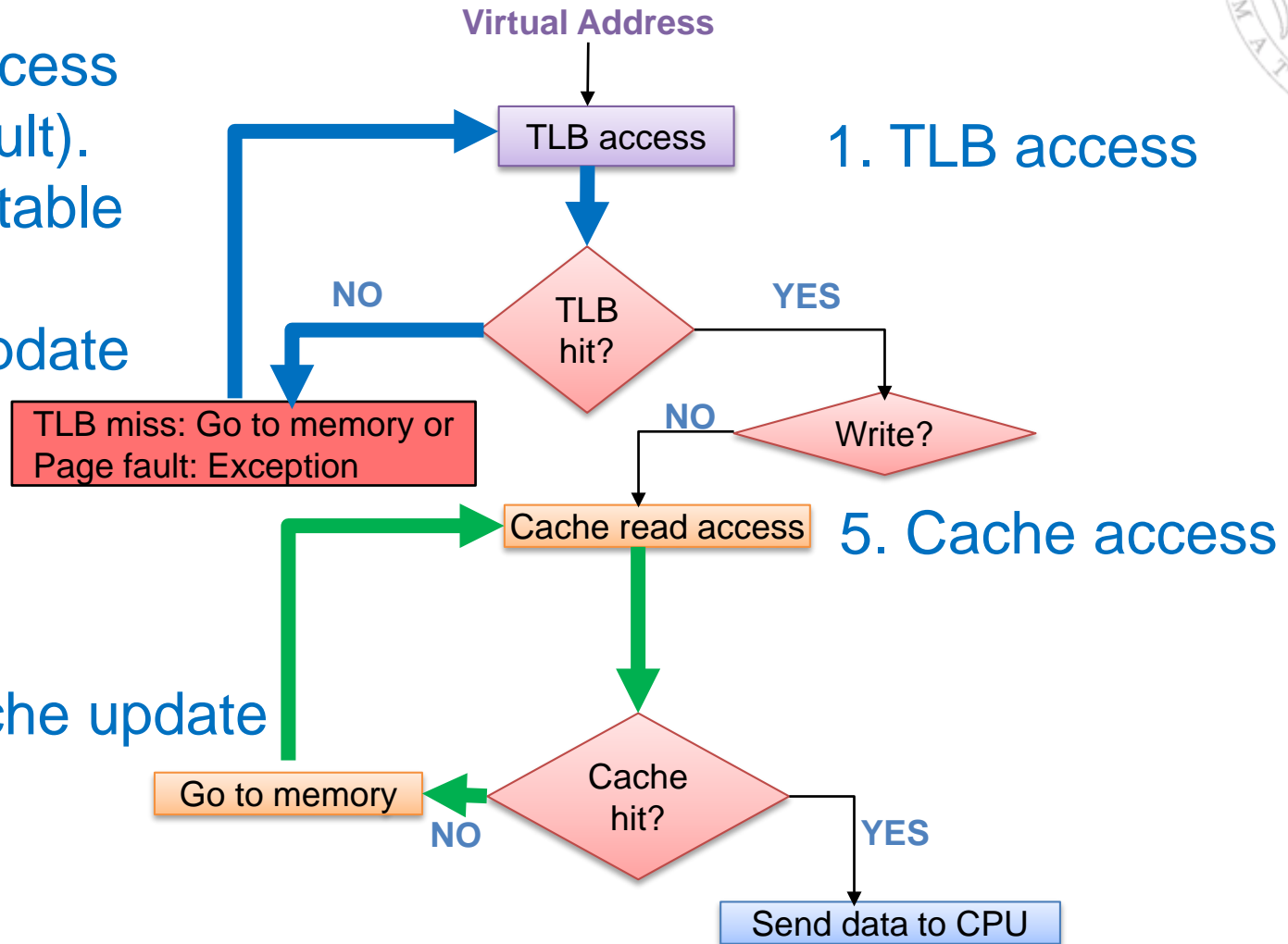
Send data  
to CPU

# Physical cache



- 2. PT Access  
(page fault).
- 3. Page table  
update
- 4. TLB update

1. TLB access



5. Cache access

6. Cache update

4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.


Fourth reference: 0x51D04

First: look for address translation in TLB

Virtual address: 0x51D04

0101 0001 1101 0000 0100

Virtual page number: 0x0A

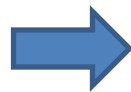


Virtual page	Physical page
0x0A	1
0x0C	2
0x02	0
0x1E	3

0 1001 1101 0000 0100

0x09D04 Physical address

TLB hit.



We can access the cache.

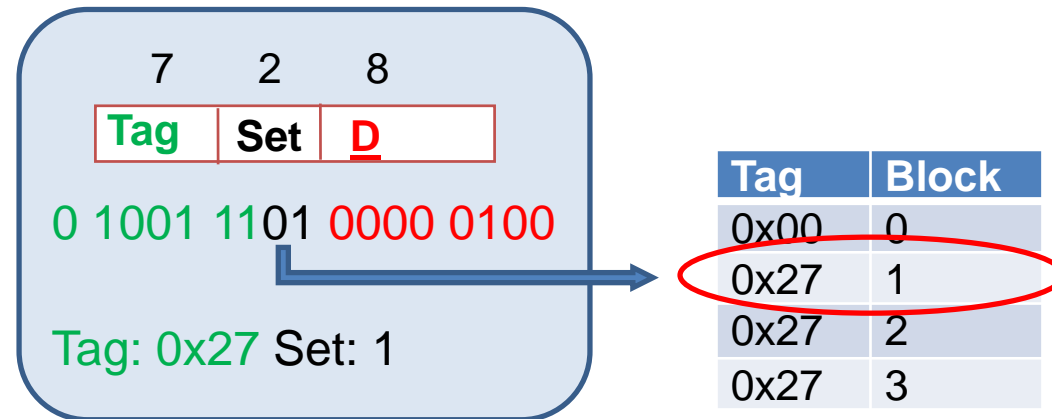


4) Say which accesses generate a miss and which generate a hit, showing the evolution of main memory and cache.

Fourth reference: 0x51D04

Second: look for address the data in the cache

Physical addr.: 0 1001 1101 0000 0100



Cache hit: block 0x27 is stored in set 1.

We send the data to the CPU