

Tema 6. Memòria Cache

Joan Manuel Parcerisa



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona



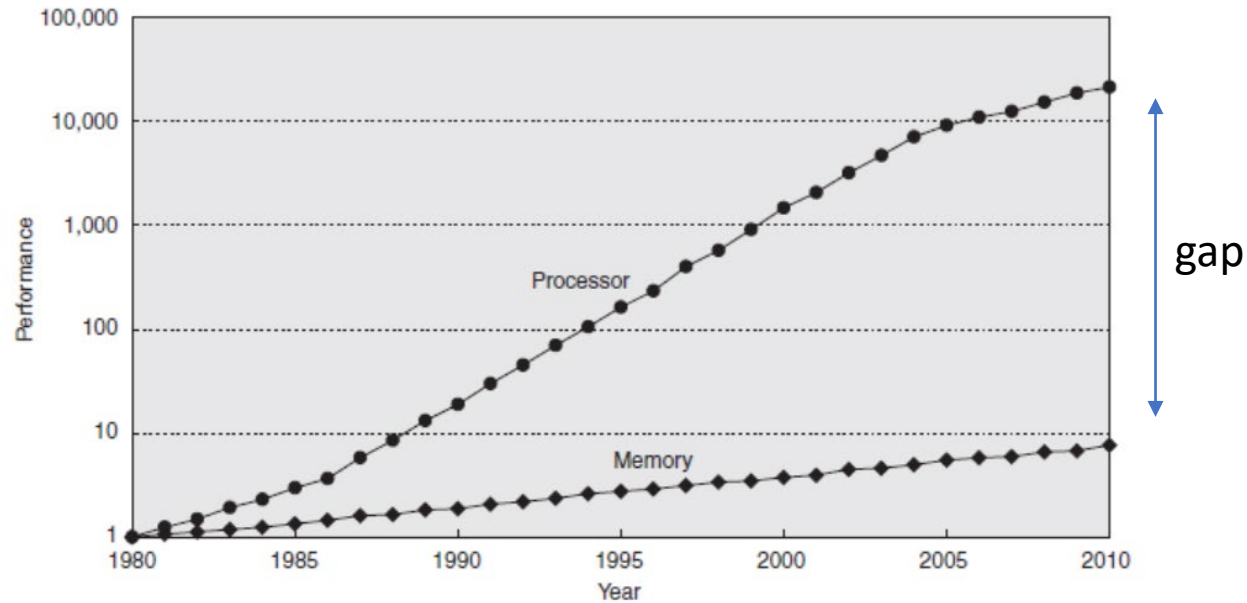
Memòria Cache

- Introducció
- Aspectes de disseny
- Impacte en el rendiment
- Millores de rendiment

Memòria Cache

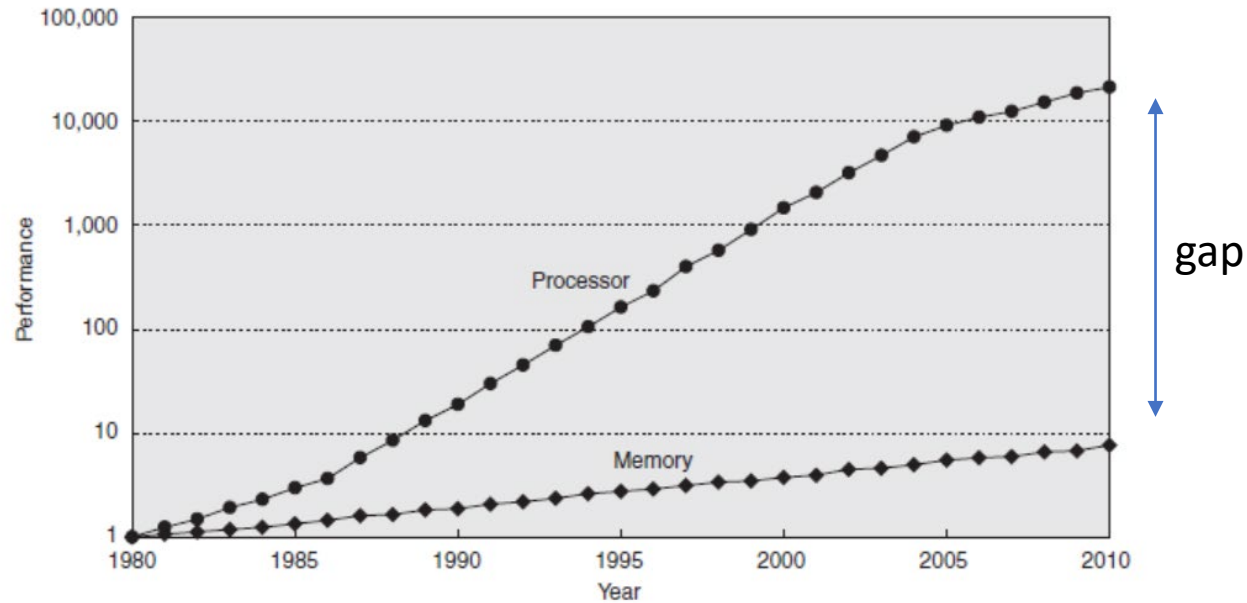
- Introducció
 - "Gap" entre els rendiments de CPU i memòria
 - Principi de localitat
 - Memòria cache. Terminologia
 - La jerarquia de memòria
- Aspectes de disseny
- Impacte en el rendiment
- Millores de rendiment

El “gap” de rendiment entre CPU i memòria



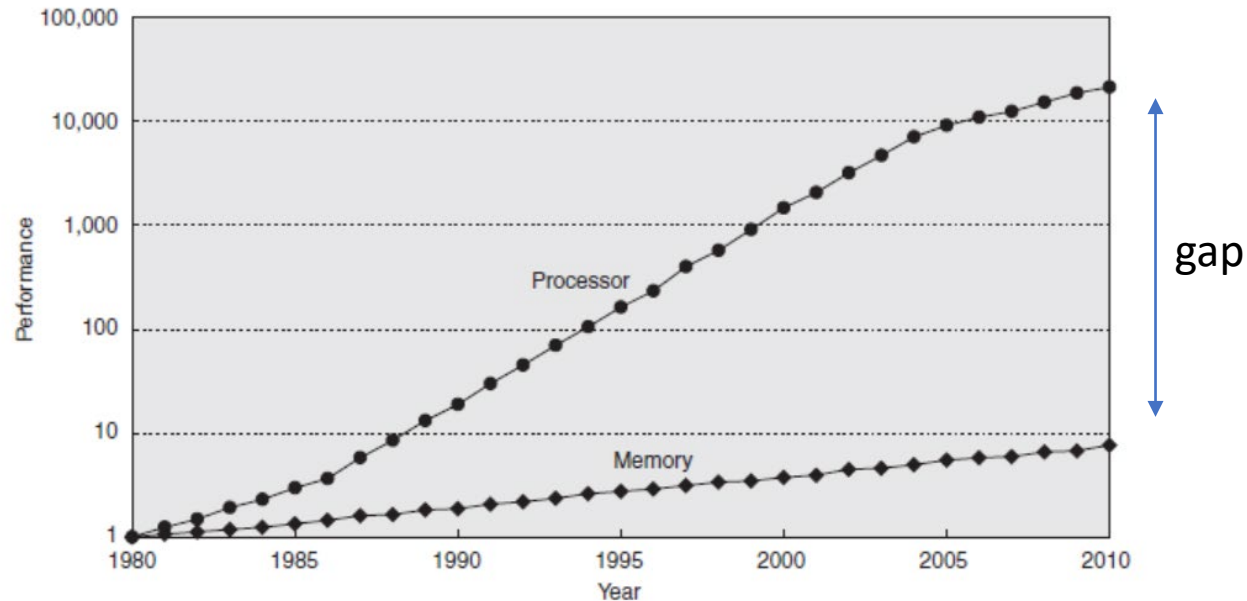
- Actualment $t_{\text{accés_memòria}} > 100 \times t_{\text{exec_aritmètica}}$

El “gap” de rendiment entre CPU i memòria



- Actualment $t_{\text{accés_memòria}} > 100 \times t_{\text{exec_aritmètica}}$
- → 99% del temps d'execució dedicat a accedir a memòria
 - fetch de cada instrucció
 - accés a dades (loads i stores)

El “gap” de rendiment entre CPU i memòria



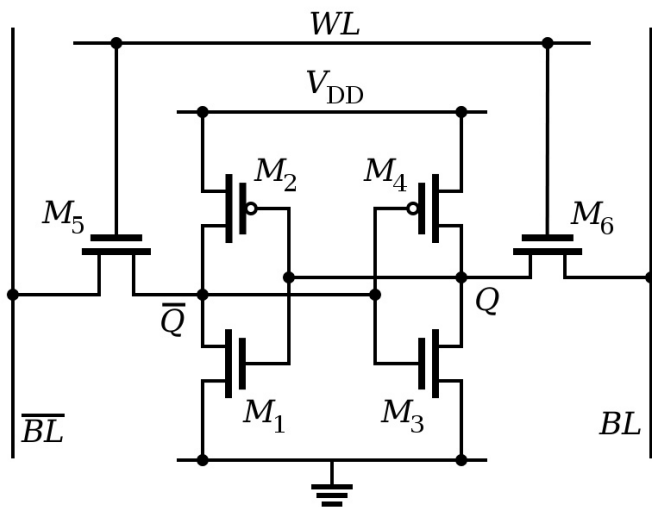
- Actualment $t_{\text{accés_memòria}} > 100 \times t_{\text{exec_aritmètica}}$
- → 99% del temps d'execució dedicat a accedir a memòria
 - fetch de cada instrucció
 - accés a dades (loads i stores)
- → Amdahl: $\text{speedup} \leq 1,01$
- → La millora de $t_{\text{accés_memòria}}$ és **imprescindible!**

El sistema de memòria

- Requisits de la memòria
 - Ràpida: temps d'accés curt
 - Gran capacitat: encabir tots els programes
 - Baix cost: \$ per GB

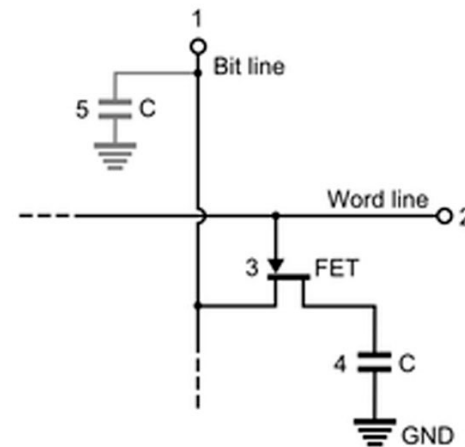
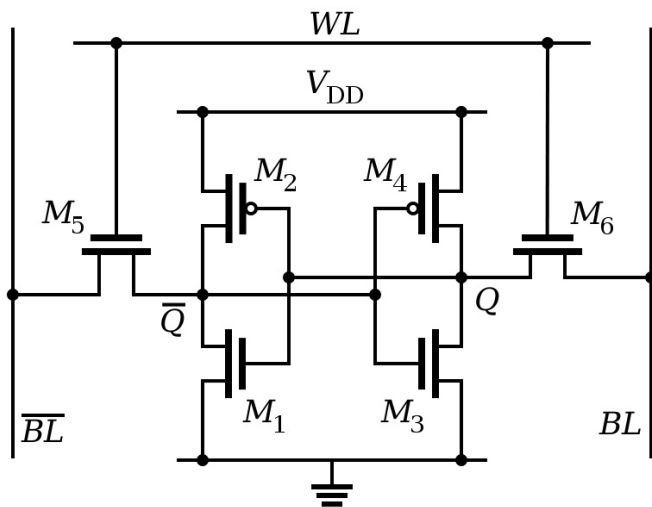
El sistema de memòria

- Requisits de la memòria
 - Ràpida: temps d'accés curt
 - Gran capacitat: encabir tots els programes
 - Baix cost: \$ per GB
- Tecnologia SRAM
 - ràpida
 - costosa (+6T/bit)
 - Baixa capacitat



El sistema de memòria

- Requisits de la memòria
 - Ràpida: temps d'accés curt
 - Gran capacitat: encabir tots els programes
 - Baix cost: \$ per GB
- Tecnologia SRAM
 - ràpida
 - costosa (+6T/bit)
 - Baixa capacitat
- Tecnologia DRAM
 - lenta
 - barata (1T+ 1C/bit)
 - Gran capacitat



El sistema de memòria

- Cap tecnologia reuneix tots els requisits

Tecnologia	Temps d'accés	Cost
SRAM	0,25 – 2,5 ns	\$2000/GB
DRAM	50 -100 ns	\$10/GB
Disc magnètic	5,000.000 – 20,000.000 ns	\$0.02/GB

- Una memòria petita es pot accedir més ràpidament
- Però no podem renunciar a una memòria més gran

La clau: principi de localitat

- Observació

- Els programes accedeixen a una porció relativament petita de l'espai d'adreces en cada instant de temps

⇒ No totes les adreces tenen la mateixa probabilitat de ser accedides

La clau: principi de localitat

- Observació

- Els programes accedeixen a una porció relativament petita de l'espai d'adreces en cada instant de temps

⇒ No totes les adreces tenen la mateixa probabilitat de ser accedides

- **Localitat temporal**

- *Alta probabilitat d'accedir a **les mateixes dades** repetidament*

La clau: principi de localitat

- Observació

- Els programes accedeixen a una porció relativament petita de l'espai d'adreces en cada instant de temps

⇒ No totes les adreces tenen la mateixa probabilitat de ser accedides

- **Localitat temporal**

- *Alta probabilitat d'accedir a **les mateixes dades** repetidament*

- **Localitat espacial**

- *Alta probabilitat d'accedir a **dades pròximes** a les ja accedides*

Localitat: exemple 1

- Un programa realitza la següent seqüència d'accessos:

accés	Adreça de memòria
1	0
2	1024
3	0
4	1024
5	0
6	1024
7	0
8	1024

- Quina mena de localitat té aquest programa?

Localitat: exemple 2

- Un programa realitza la següent seqüència d'accessos:

accés	Adreça de memòria
1	0
2	4
3	8
4	12
5	16
6	20
7	24
8	28

- Quina mena de localitat té aquest programa?

Localitat: exemple 3

- Un programa realitza la següent seqüència d'accessos:

accés	Adreça de memòria
1	0
2	4
3	8
4	12
5	0
6	4
7	8
8	12

- Quina mena de localitat té aquest programa?

Origen de la localitat

- **Localitat temporal**

- Degut als **bucles**

- Probablement accedirem a les mateixes dades i instruccions en cada iteració

Origen de la localitat

- **Localitat temporal**

- Degut als **bucles**

- Probablement accedirem a les mateixes dades i instruccions en cada iteració

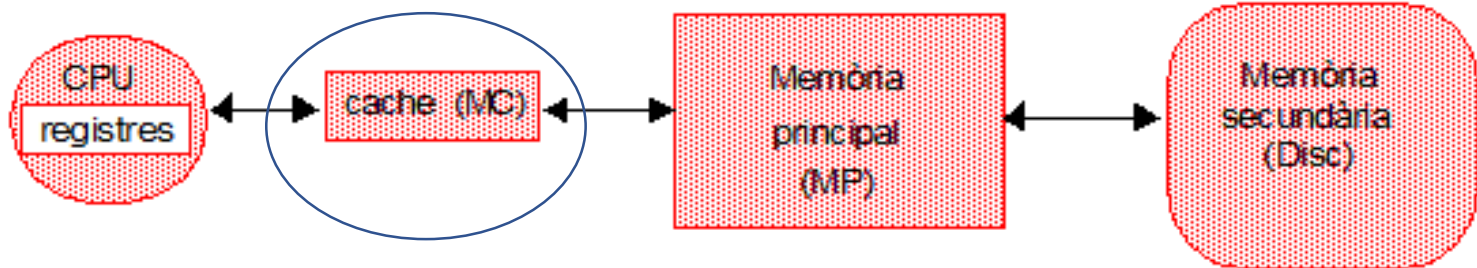
- **Localitat espacial**

- Degut a **vectors** (dades) i al **seqüenciament implícit** (instruccions)

- Probablement accedim a elements pròxims als ja accedits
 - Executarem les instruccions següents (excepte si saltem)

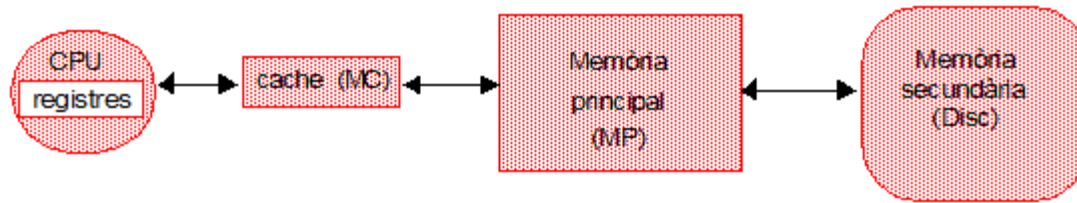
Explotant la localitat: la cache

- Idea: interposar la memòria **cache** (MC)



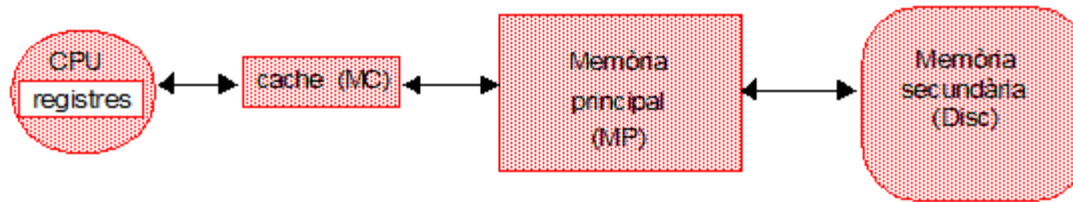
- És una memòria **petita i ràpida** (SRAM)
- Entre la CPU i la memòria principal (MP), gran i lenta (DRAM)

Explotant la localitat: la cache



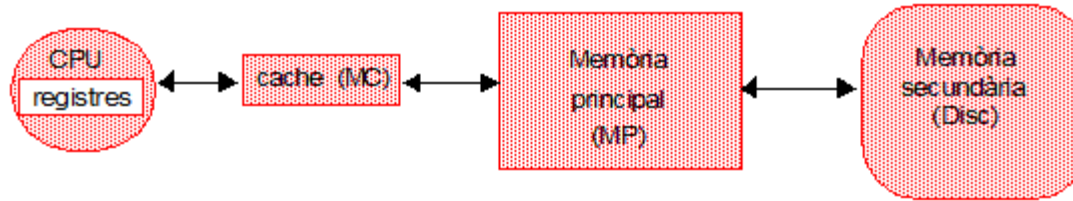
- La MC aprofita la localitat
 - Guarda dades amb més probabilitat de ser accedides en el futur
 - Per accedir-hi amb més rapidesa
 - La majoria dels accessos són servits per la MC

Explotant la localitat: la cache



- La MC aprofita la localitat
 - Guarda dades amb més probabilitat de ser accedides en el futur
 - Per accedir-hi amb més rapidesa
 - La majoria dels accessos són servits per la MC
- Localitat temporal
 - Per cada dada accedida, en guarda una còpia, per si la torna a accedir

Explotant la localitat: la cache



- La MC aprofita la localitat
 - Guarda dades amb més probabilitat de ser accedides en el futur
 - Per accedir-hi amb més rapidesa
 - La majoria dels accessos són servits per la MC
- Localitat temporal
 - Per cada dada accedida, en guarda una còpia, per si la torna a accedir
- Localitat espacial
 - Per cada dada accedida, en guarda el bloc sencer on pertany, per si accedim a una dada pròxima
 - La unitat de transferència entre MP i MC és sempre 1 bloc

Terminologia

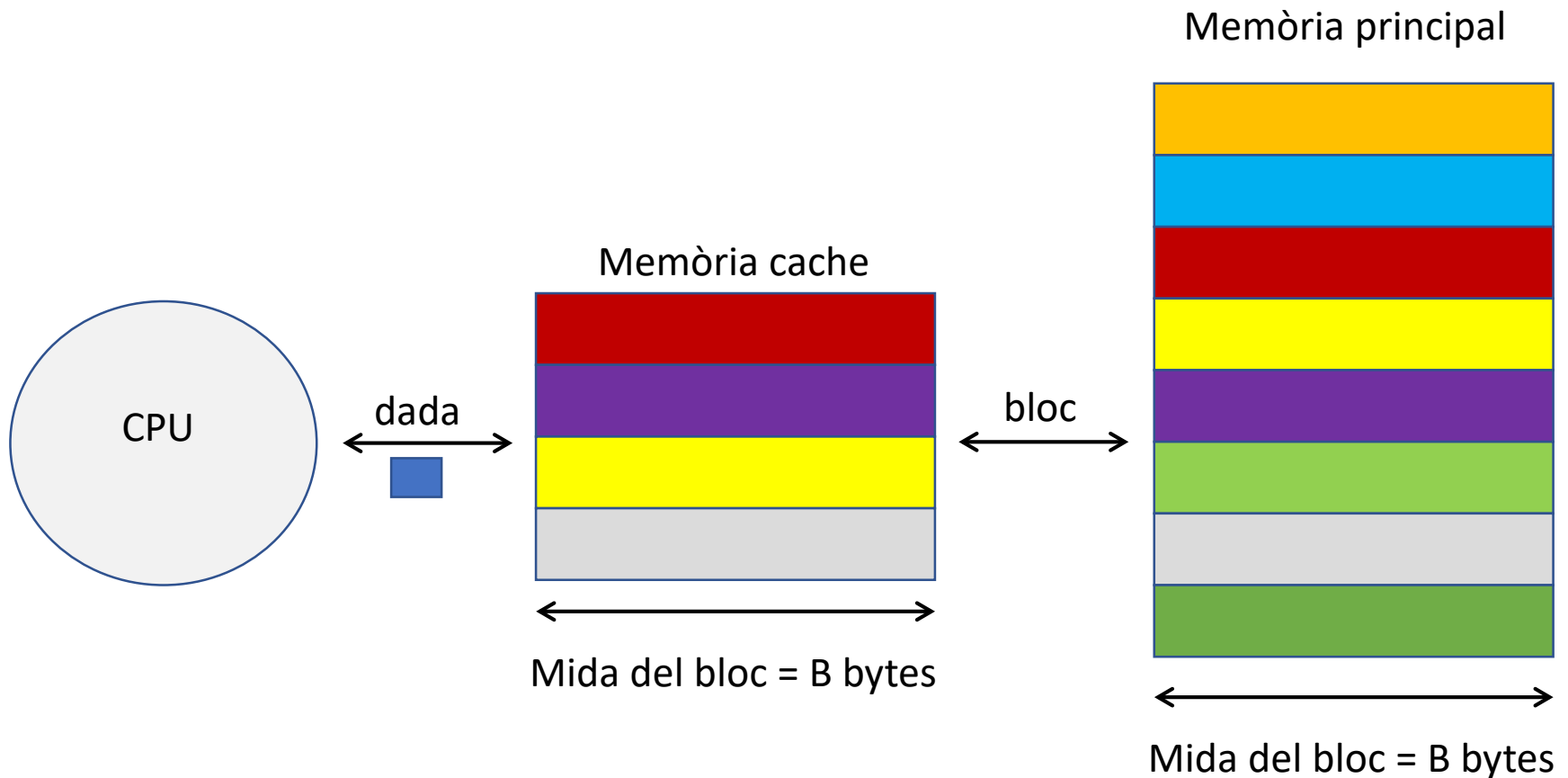
- *Referència a memòria* = *accés a memòria*
- *Encert (hit)*
 - La dada accedida per la CPU està present a la MC
- *Fallada (miss)*
 - La dada accedida per la CPU **no** està present a la MC
 - Cal portar (copiar) un bloc de la MP a la MC
 - Si el bloc no hi cap a la MC, *reemplaçarà* un altre bloc

Terminologia

- *Referència a memòria = accés a memòria*
- *Encert (hit)*
 - La dada accedida per la CPU està present a la MC
- *Fallada (miss)*
 - La dada accedida per la CPU **no** està present a la MC
 - Cal portar (copiar) un bloc de la MP a la MC
 - Si el bloc no hi cap a la MC, *reemplaçarà* un altre bloc
- *Taxa d'encerts (hit ratio)*
 - Percentatge d'accessos a memòria que esdevenen encerts
$$h = \text{núm_encerts} / \text{núm_referències}$$
- *Taxa de fallades (miss ratio)*
 - Percentatge d'accessos a memòria que esdevenen fallades
$$m = \text{núm_fallades} / \text{núm_referències}$$
$$m = 1 - h$$

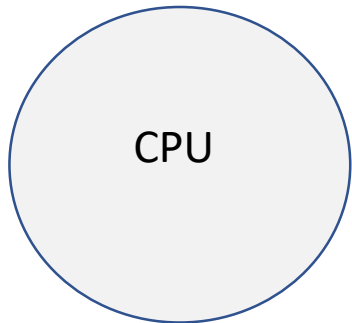
Organització de la memòria en blocs

- L'espai d'adreçament es divideix en blocs
 - Bloc: la mínima unitat d'informació que pot estar o no a la cache

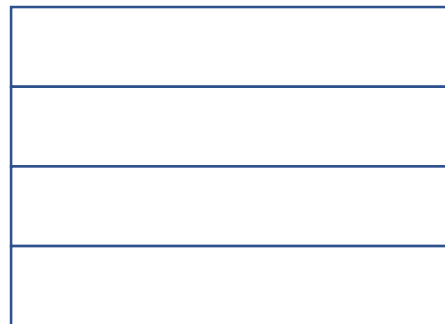


Encerts i fallades

Seqüència d'accessos: 

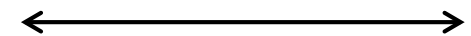


Memòria cache



Mida del bloc = B bytes

Memòria principal



Mida del bloc = B bytes

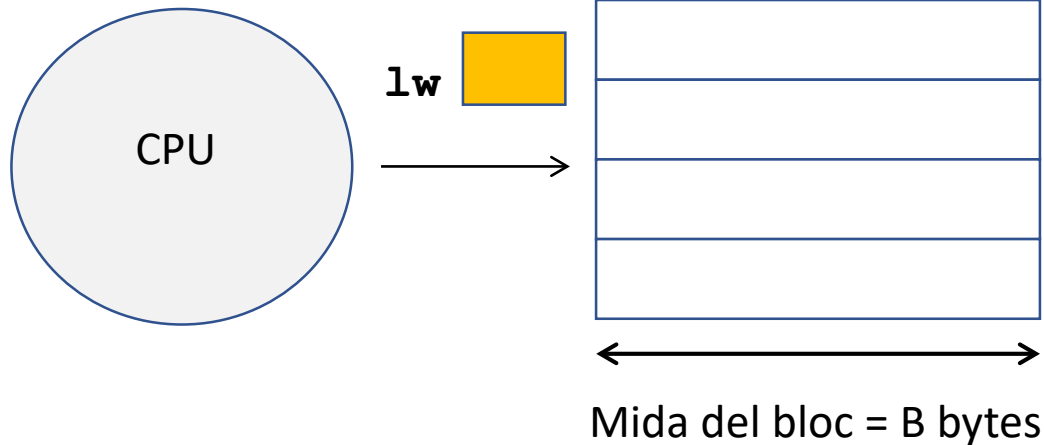
Encerts i fallades

Seqüència d'accessos: 

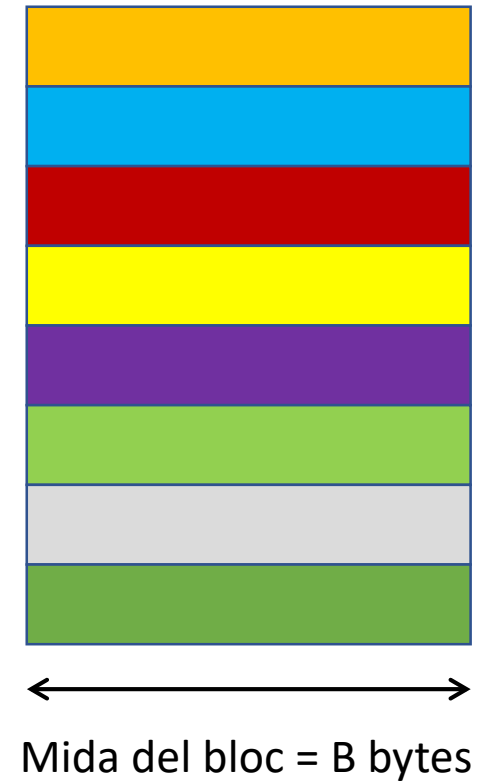
↑

Cache Miss!

Memòria cache



Memòria principal



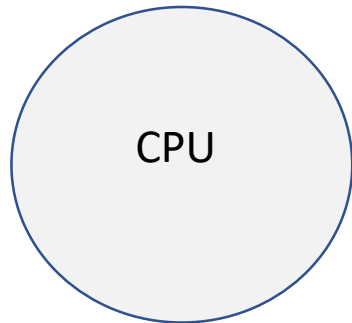
Encerts i fallades

Seqüència d'accessos: 

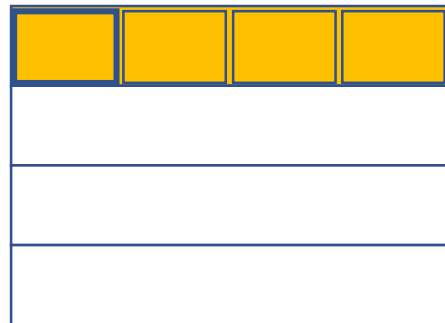
↑

Llegim el bloc de MP

Memòria cache



$1w$



Mida del bloc = B bytes

MP



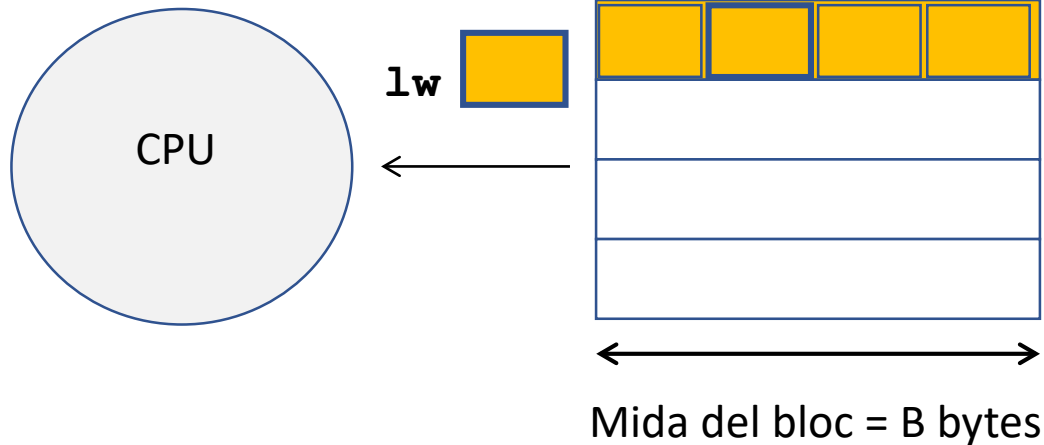
Mida del bloc = B bytes

Encerts i fallades

Seqüència d'accessos: 
M 

Cache hit!

Memòria cache



MP

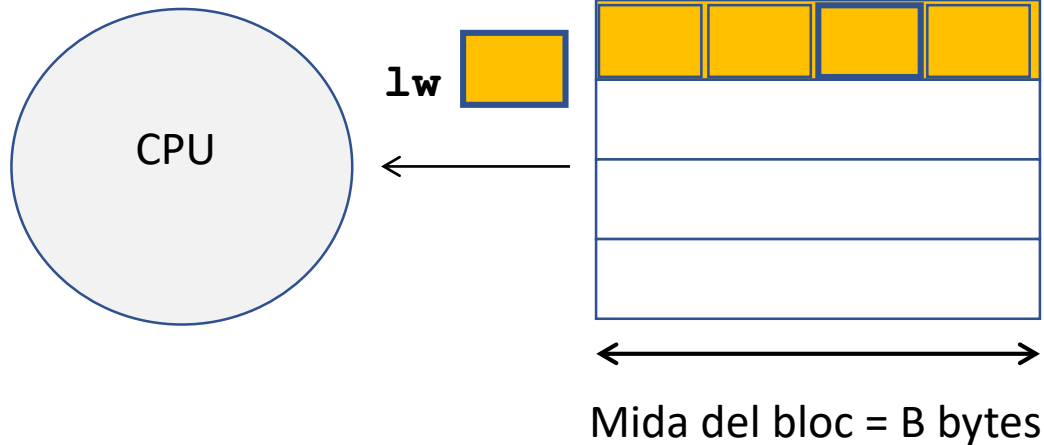


Encerts i fallades

Seqüència d'accessos: 
M H 

Cache Hit!

Memòria cache



MP

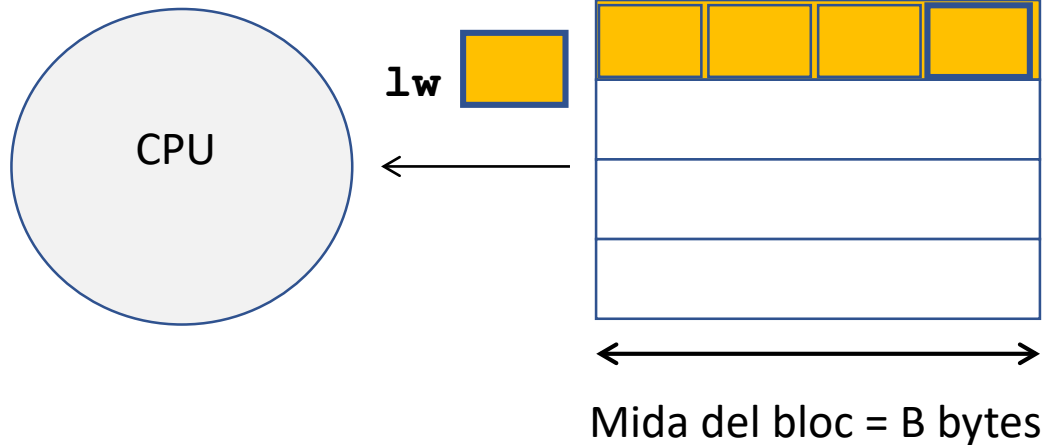


Encerts i fallades

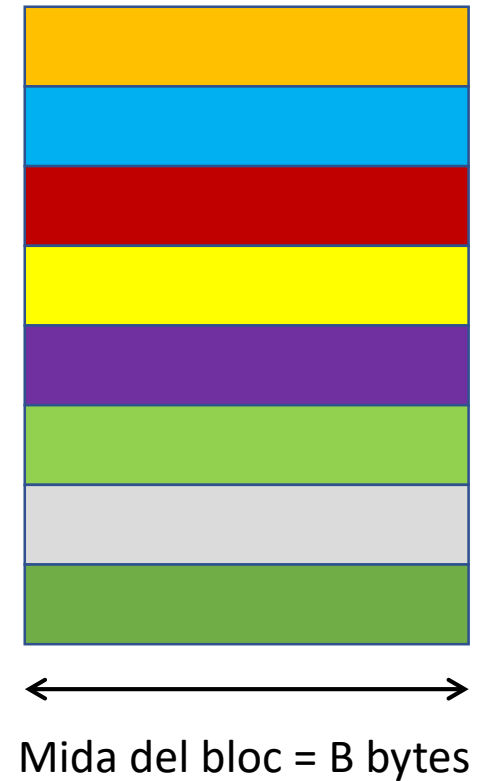
Seqüència d'accessos: 
M H H 

Cache Hit!



Memòria cache



MP

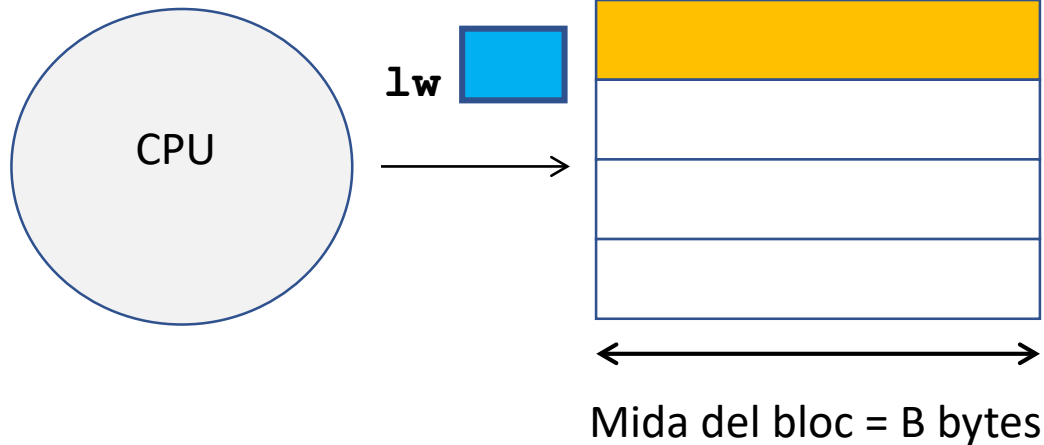


Encerts i fallades

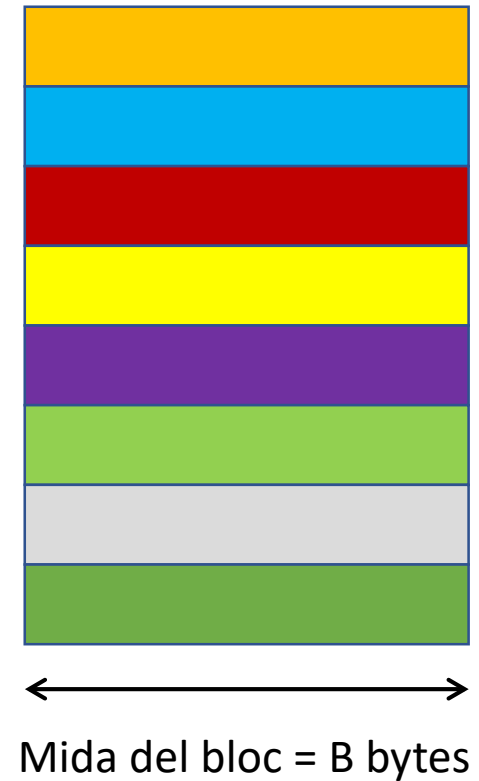
Seqüència d'accessos: 
M H H H 

Cache Miss!

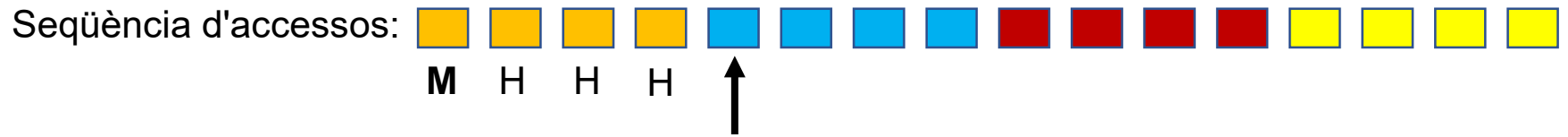
Memòria cache



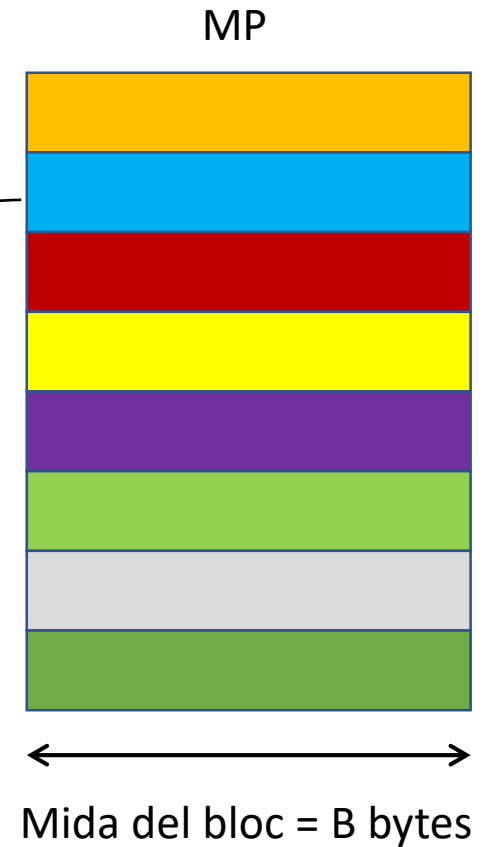
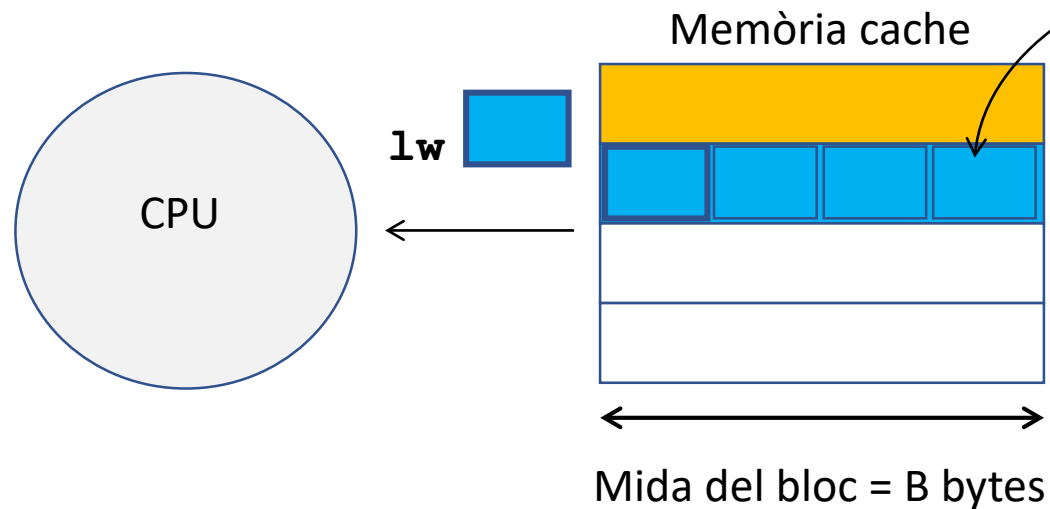
MP





Encerts i fallades



Llegim el bloc de MP

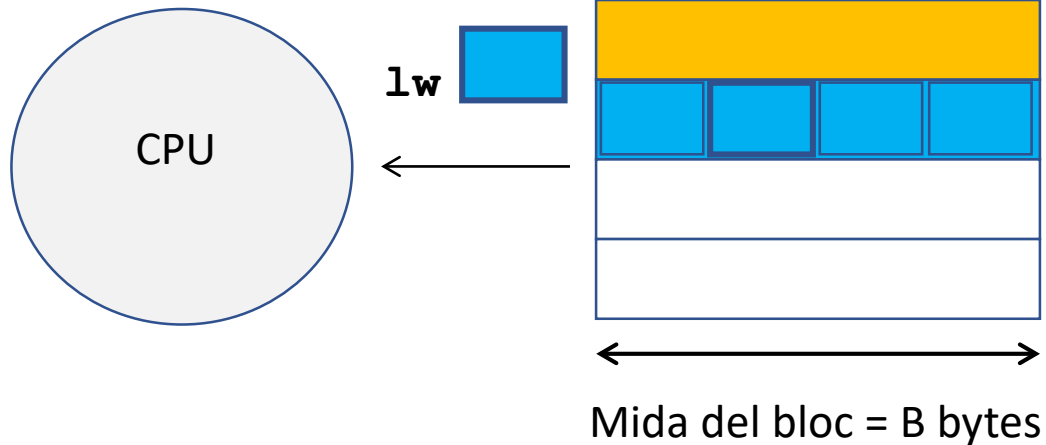


Encerts i fallades

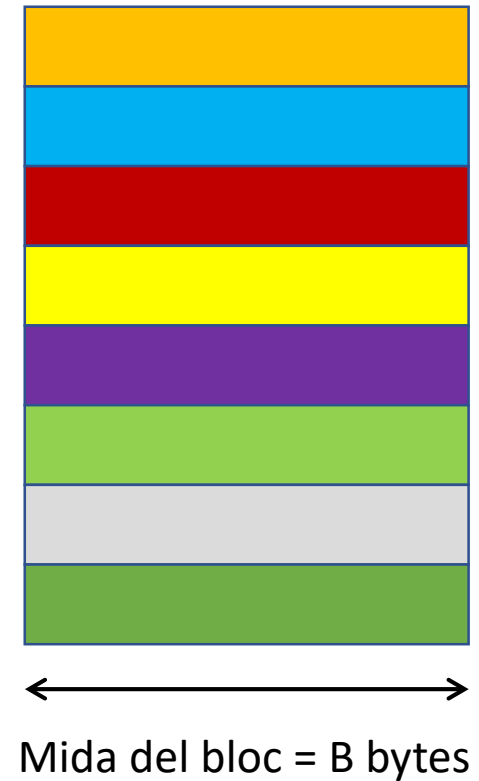
Seqüència d'accessos: 
M H H H M 

Cache Hit!

Memòria cache



MP

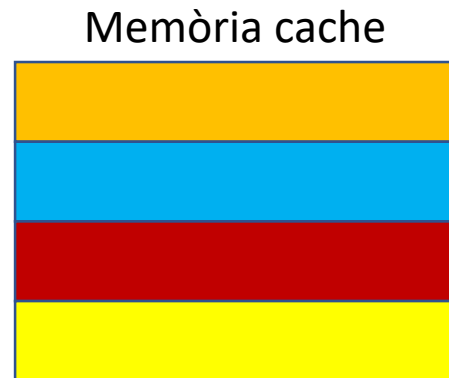
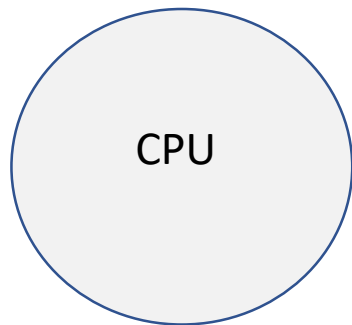


Encerts i fallades

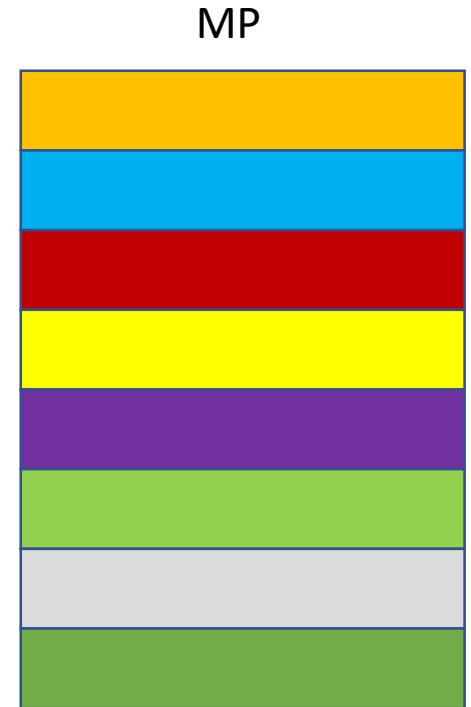
Seqüència d'accessos: 
M H H H M H H H M H H H M H H H

Encerts: $h = 0,75$

Fallades: $m = 0,25$



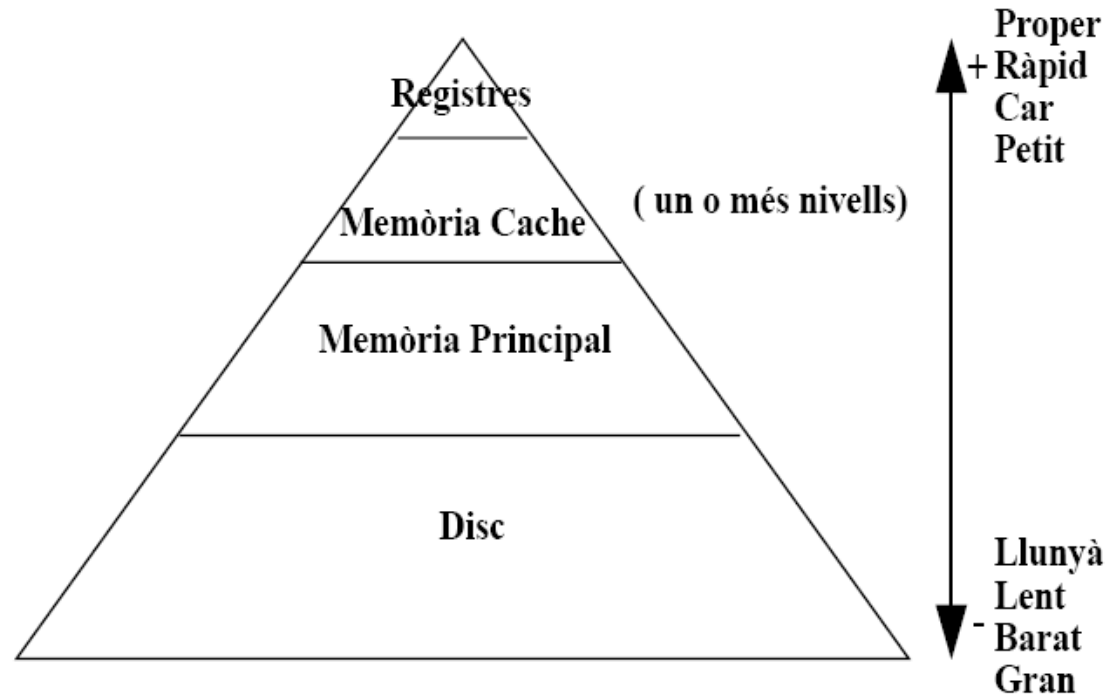
Mida del bloc = B bytes



Mida del bloc = B bytes

Jerarquia de memòria

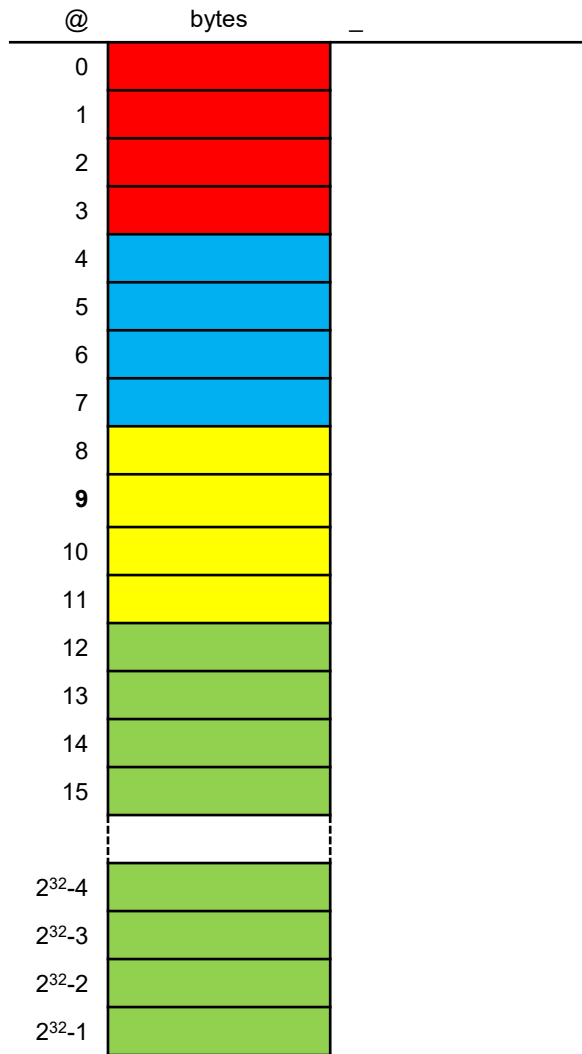
- Generalitzant i fent abstracció de la idea
 - El subsistema de memòria és una jerarquia de diversos nivells
 - Cada nivell guarda un subconjunt de les dades del nivell inferior
 - Les que tindran més probabilitat de ser accedides



Memòria Cache

- Introducció
- Aspectes de disseny
 - Organització de la memòria i la cache en blocs
 - Correspondència directa
 - Mida òptima del bloc
 - Polítiques d'escriptura
- Impacte en el rendiment
- Millores de rendiment

Organització de la memòria en blocs



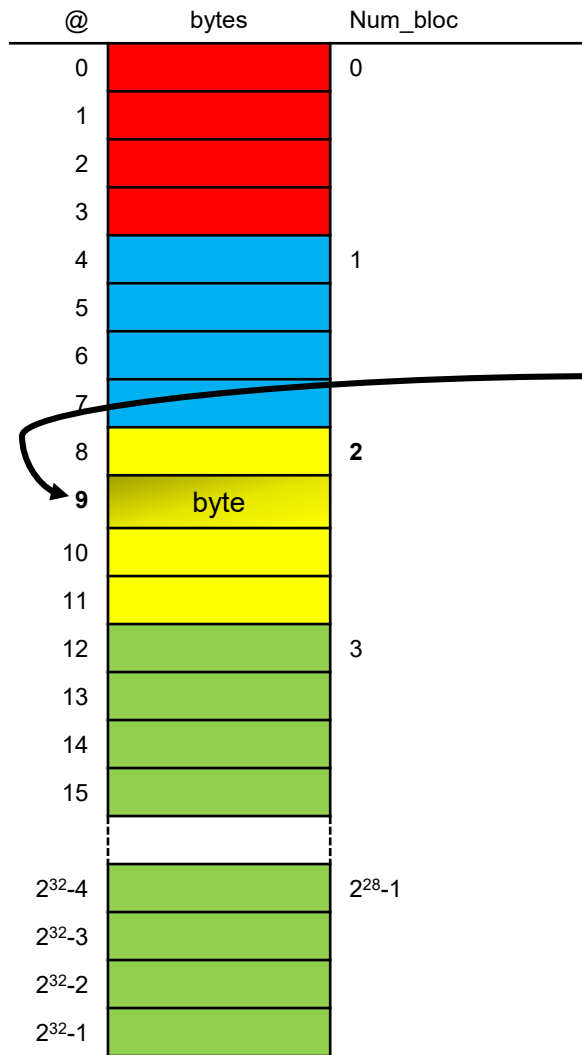
- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes

Organització de la memòria en blocs

@	bytes	Num_bloc
0		0
1		
2		
3		
4		1
5		
6		
7		
8		2
9		
10		
11		
12		3
13		
14		
15		
$2^{32}-4$		$2^{28}-1$
$2^{32}-3$		
$2^{32}-2$		
$2^{32}-1$		

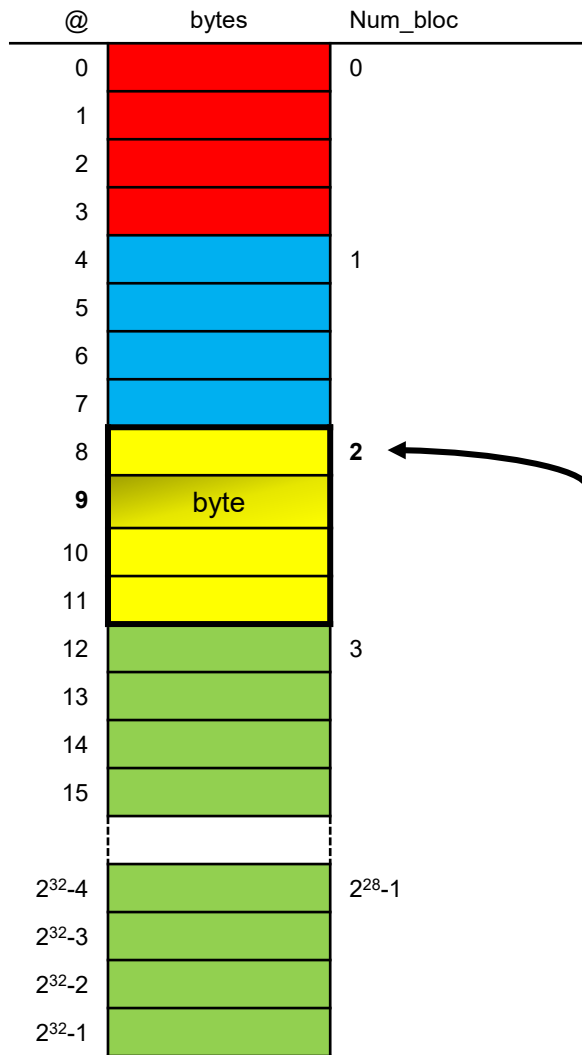
- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes
 - Numerem els blocs des de 0

Organització de la memòria en blocs



- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes
 - Numerem els blocs des de 0
- Suposem que accedim a l'adreça $A=9$
 - A quin bloc pertany?

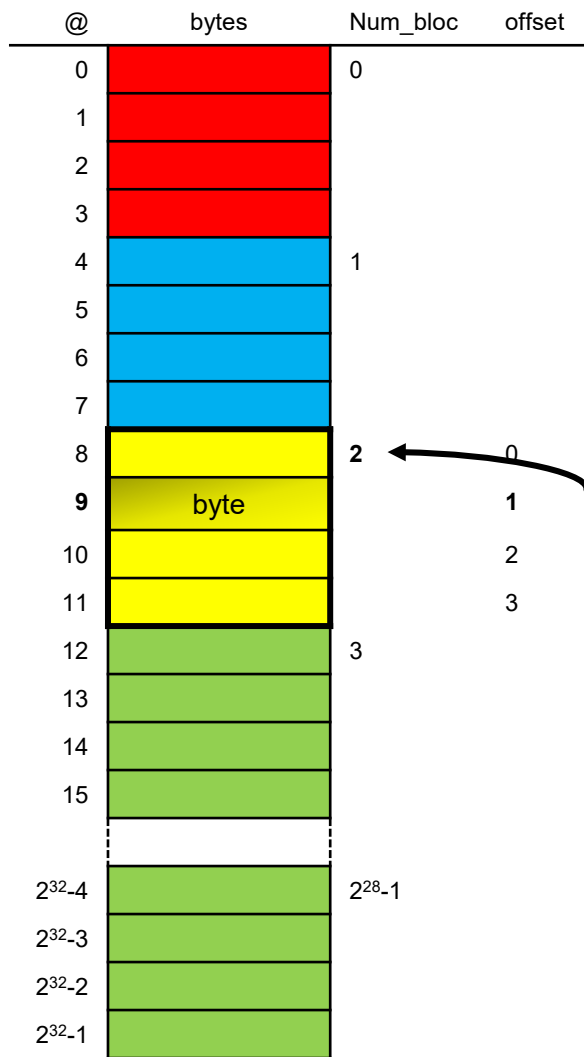
Organització de la memòria en blocs



- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes
 - Numerem els blocs des de 0
- Suposem que accedim a l'adreça $A=9$
 - A quin bloc pertany?

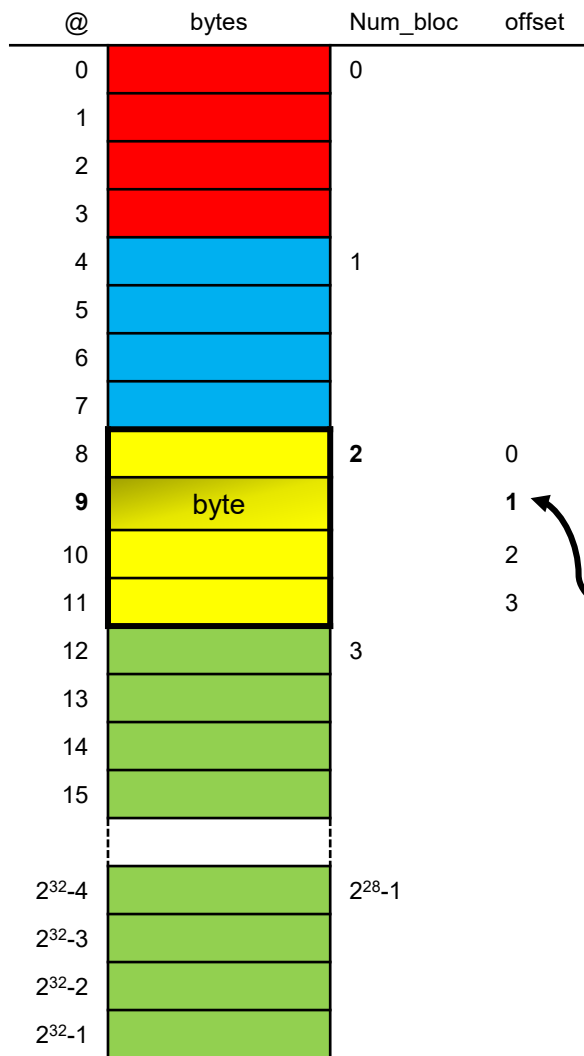
$num_bloc = 9 \div 4 = 2$

Organització de la memòria en blocs

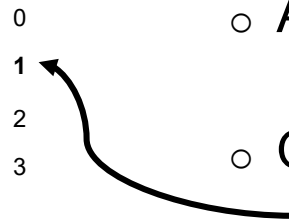


- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes
 - Numerem els blocs des de 0
- Suposem que accedim a l'adreça $A=9$
 - A quin bloc pertany?
 $num_bloc = 9 \div 4 = 2$
 - Quin byte (*offset*) és, de dins el bloc?

Organització de la memòria en blocs



- L'espai d'adreçament es subdivideix en blocs, de mida fixa $B = 2^b$
 - Suposem $B=4$ bytes
 - Numerem els blocs des de 0
- Suposem que accedim a l'adreça $A=9$
 - A quin bloc pertany?
 $num_bloc = 9 \div 4 = 2$
 - Quin byte (*offset*) és, de dins el bloc?
 $offset = 9 \bmod 4 = 1$



num_bloc i *offset* d'una adreça A

- Suposem la mida de bloc $B=2^b$
- Suposem que accedim a l'adreça A

$$num_bloc = A \underline{\text{div}} B$$

$$offset = A \underline{\text{mod}} B$$

num_bloc i *offset* d'una adreça A

- Suposem la mida de bloc $B=2^b$
- Suposem que accedim a l'adreça A
$$num_bloc = A \underline{\text{div}} B$$
$$offset = A \underline{\text{mod}} B$$
- En binari, sols cal fer una selecció de bits
 - Suposem $B=16 (= 2^4)$
 - Executem un *load byte* a $A = 0x100100F8$

num_bloc i *offset* d'una adreça A

- Suposem la mida de bloc $B=2^b$
- Suposem que accedim a l'adreça A

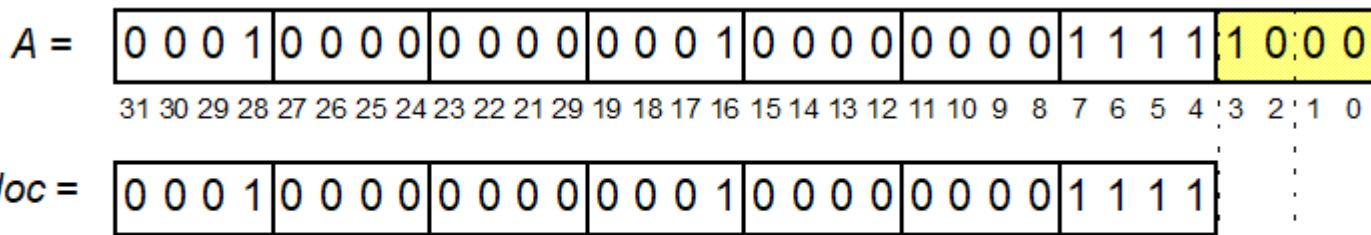
$$num_bloc = A \text{ div } B$$

$$offset = A \text{ mod } B$$

- En binari, sols cal fer una selecció de bits

- Suposem $B=16 (= 2^4)$
- Executem un *load byte* a $A = 0x100100F8$

$$\begin{aligned} num_bloc &= 0x100100F8 \text{ div } 16 \\ &= 0x100100F \text{ (descartem 4 bits de menor pes)} \end{aligned}$$



num_bloc i *offset* d'una adreça A

- Suposem la mida de bloc $B=2^b$
- Suposem que accedim a l'adreça A

$$num_bloc = A \div B$$

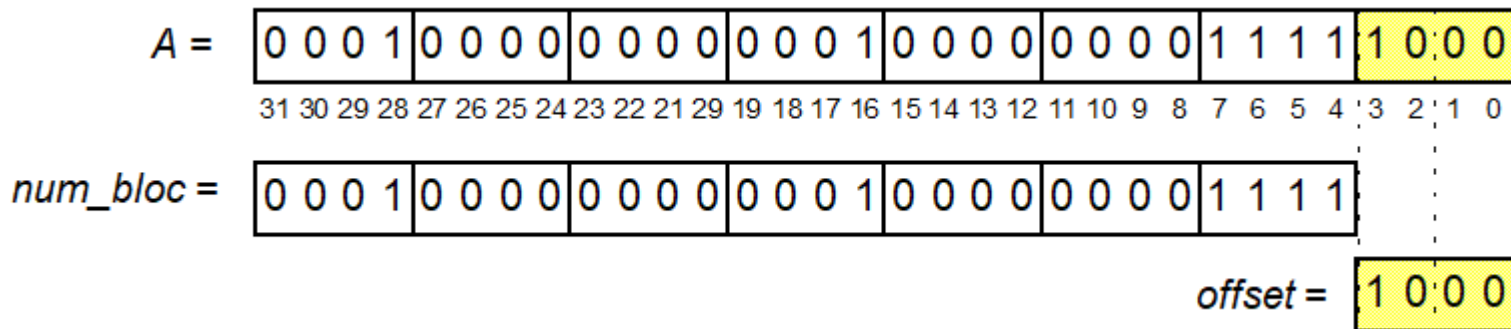
$$offset = A \bmod B$$

- En binari, sols cal fer una selecció de bits

- Suposem $B=16$ ($= 2^4$)
- Executem un *load byte* a $A = 0x100100F8$

```
num_bloc = 0x100100F8 div 16
           = 0x100100F (descartem 4 bits de menor pes)
```

```
offset = 0x100100F8 mod 16
       = 0x8 (4 bits de menor pes)
```



Línies, etiquetes i bits de validesa

- La cache és com una taula
 - Cada entrada (anomenada *línia*) conté 1 bloc (o cap)
- Cada línia conté
 - **bit de validesa V**: indica si la línia conté un bloc o cap
 - V=0 (línia buida)
 - V=1 (línia conté un bloc vàlid)
 - **Etiqueta (o tag)**: identifica el *num_bloc* de cada bloc guardat
 - **Dades**: conté els bytes del bloc

línia	V	Etiqueta	Dades
0			Bloc
1			Bloc
2			Bloc
3			Bloc

Correspondència directa: índex

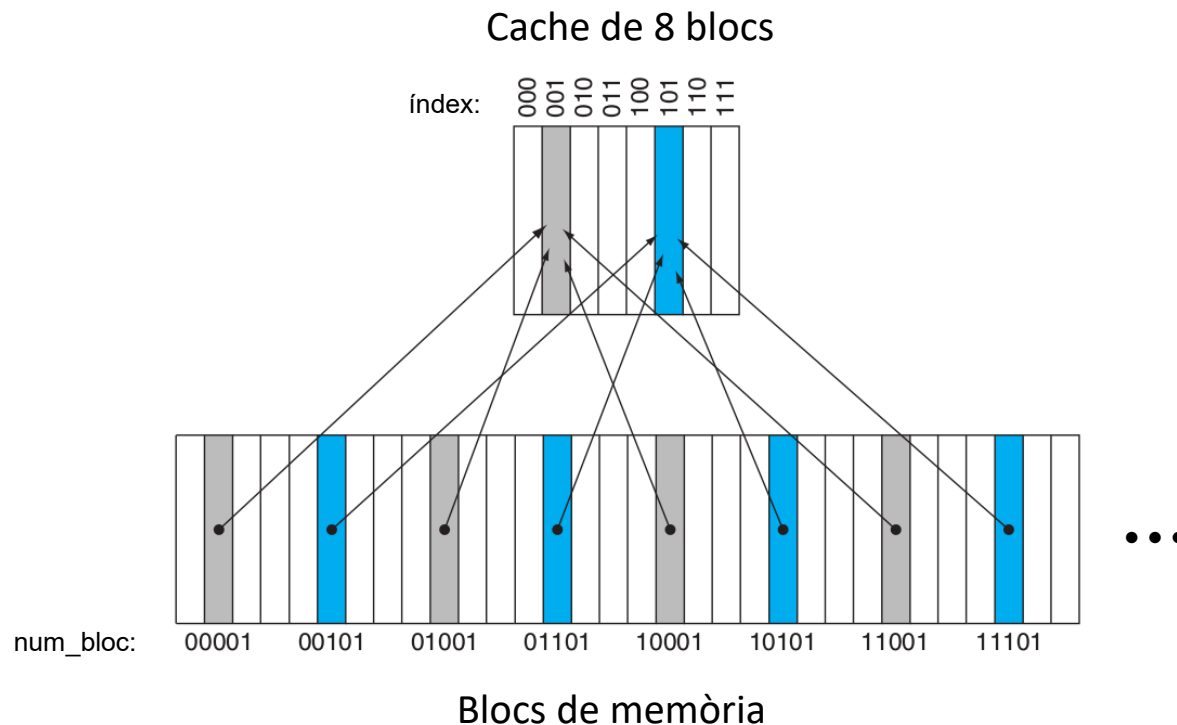
- Donat un *num_bloc*, **a quina línia guardar-lo?**
 - Depèn de *l'algorisme d'emplaçament (placement)*

Correspondència directa: índex

- Donat un *num_bloc*, **a quina línia guardar-lo?**
 - Depèn de *l'algorisme d'emplaçament*
- Cas més simple: cache de *correspondència directa*
 - Cada bloc de memòria es "mapeja" a una línia fixa, i sols es pot guardar en aquesta línia

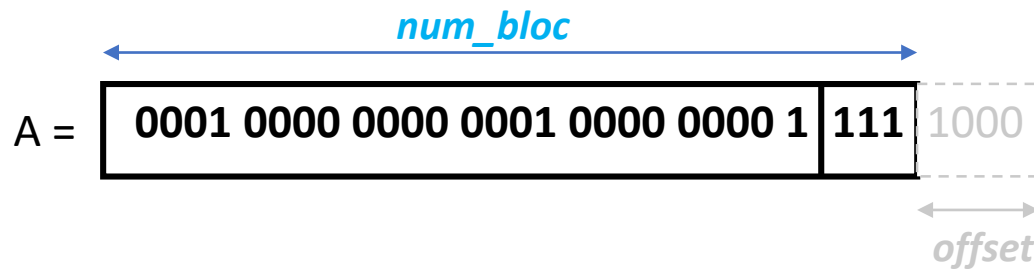
Correspondència directa: índex

- Donat un *num_bloc*, a quina línia guardar-lo?
 - Depèn de l'algorisme d'emplaçament
- Cas més simple: cache de *correspondència directa*
 - Cada bloc de memòria es "mapeja" a una línia fixa, i sols es pot guardar en aquesta línia
 - L'índex de línia és funció de l'adreça, i.e. del número de bloc
$$\text{índex} = \text{num_bloc} \bmod (\text{número de línies de cache})$$



Correspondència directa: índex

- Exemple anterior: accés a $A = 0x100100F8$
 - Blocs de $B=16$ bytes
 $num_bloc = A \div 16 = 0x100100F$



Correspondència directa: índex

- Exemple anterior: accés a $A = 0x100100F8$

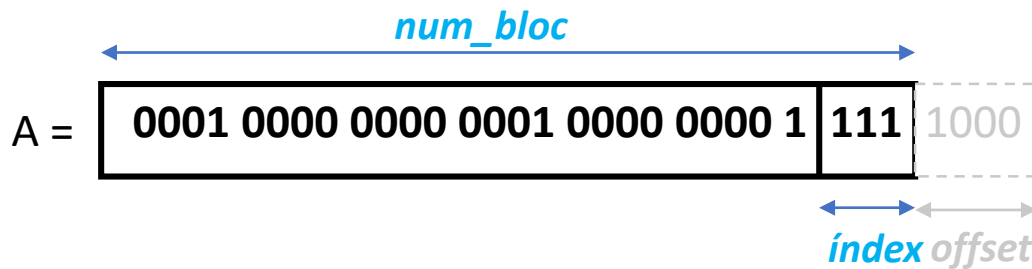
- Blocs de $B=16$ bytes

$$num_bloc = A \div 16 = 0x100100F$$

- Cache de 8 línies

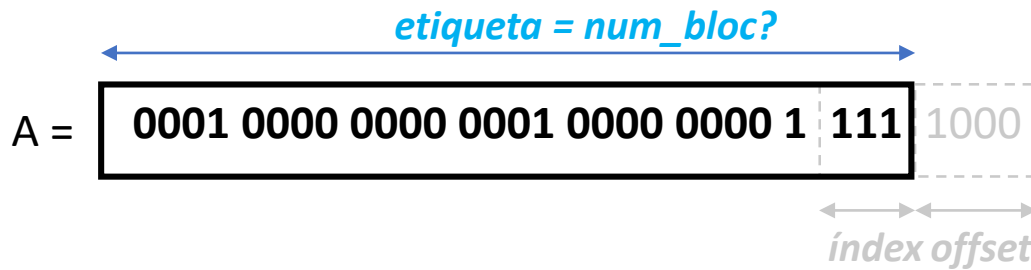
$$índex = num_bloc \bmod 8 = 7$$

→ el buscarem a la línia 7!



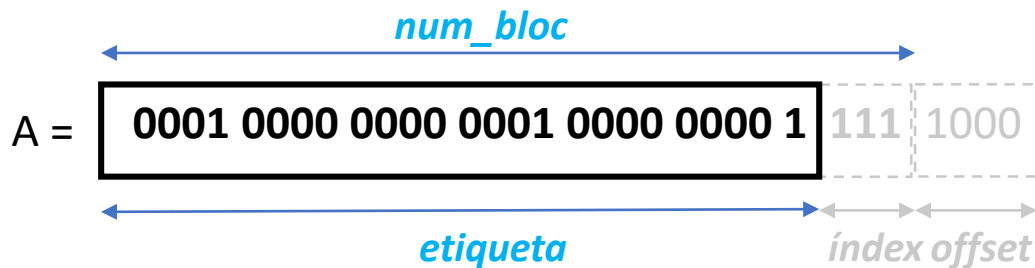
Correspondència directa: etiqueta

- Com sabem quin bloc hi ha guardat en una línia particular?
 - L'etiqueta que l'identifica podria ser el *num_bloc*?



Correspondència directa: etiqueta

- Com sabem quin bloc hi ha guardat en una línia particular?
 - L'etiqueta que l'identifica podria ser el *num_bloc*
 - Però tots els blocs que mapegen la mateixa línia tenen el mateix *índex*!
 - L'etiqueta sols ha d'incloure els bits més alts del *num_bloc*, prescindint dels bits que indiquen l'*índex*



Exercici

- Suposant una mida de bloc $B = 64$ bytes, i una cache amb 32 blocs, indica l'*índex de línia*, l'*etiqueta*, i l'*offset* per a cada una de les adreces de memòria:
 - $A = 0x100101C0$
 - $A = 0x1001060F$

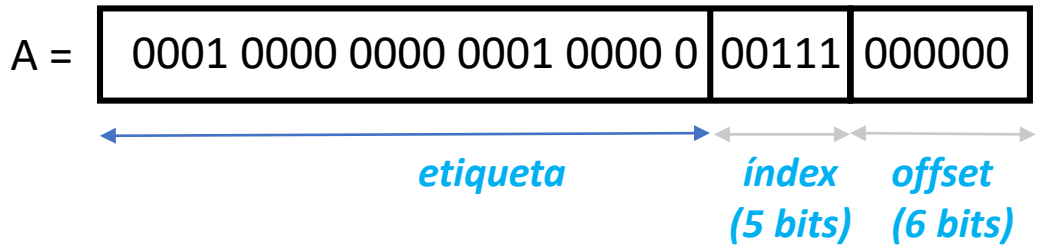
Exercici

- Suposant una mida de bloc B = 64 bytes, i una cache amb 32 blocs, indica l'índex de línia, l'etiqueta, i l'offset per a cada una de les adreces de memòria:
 - A = 0x100101C0 = 0001 0000 0000 0001 0000 0001 1100 0000

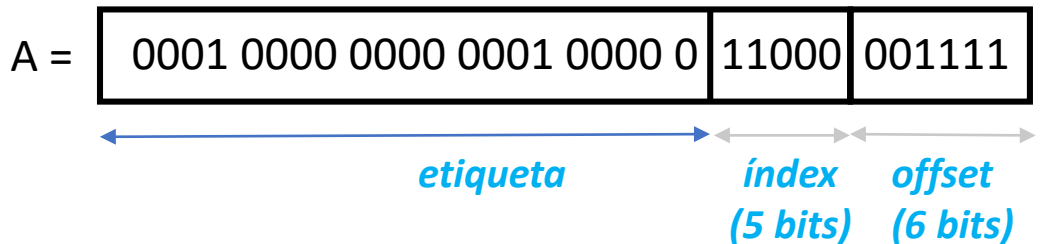
Índex = 0x07

Offset = 0x00

Etiqueta = 0x020020



- $A = 0x1001060F$



Exercici

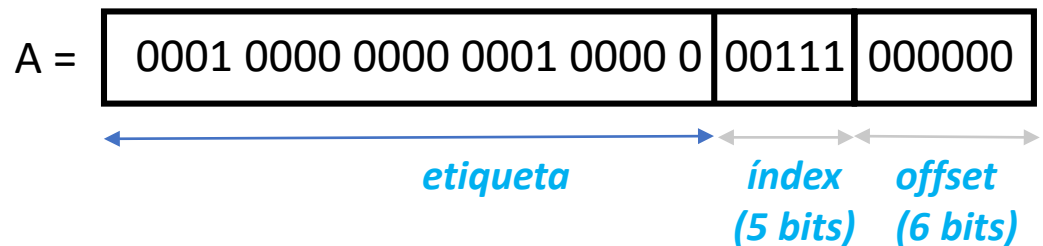
- Suposant una mida de bloc $B = 64$ bytes, i una cache amb 32 blocs, indica l'índex de línia, l'etiqueta, i l'offset per a cada una de les adreces de memòria:

- $A = 0x100101C0$

Índex = 0x07

Offset = 0x00

Etiqueta = 0x020020

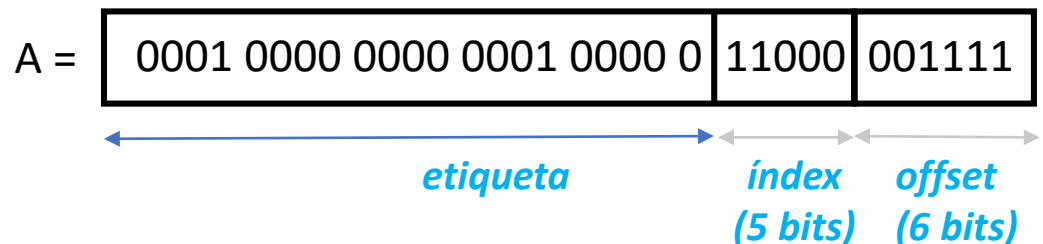


- $A = 0x1001060F = 0001\ 0000\ 0000\ 0001\ 0000\ 0110\ 0000\ 1111$

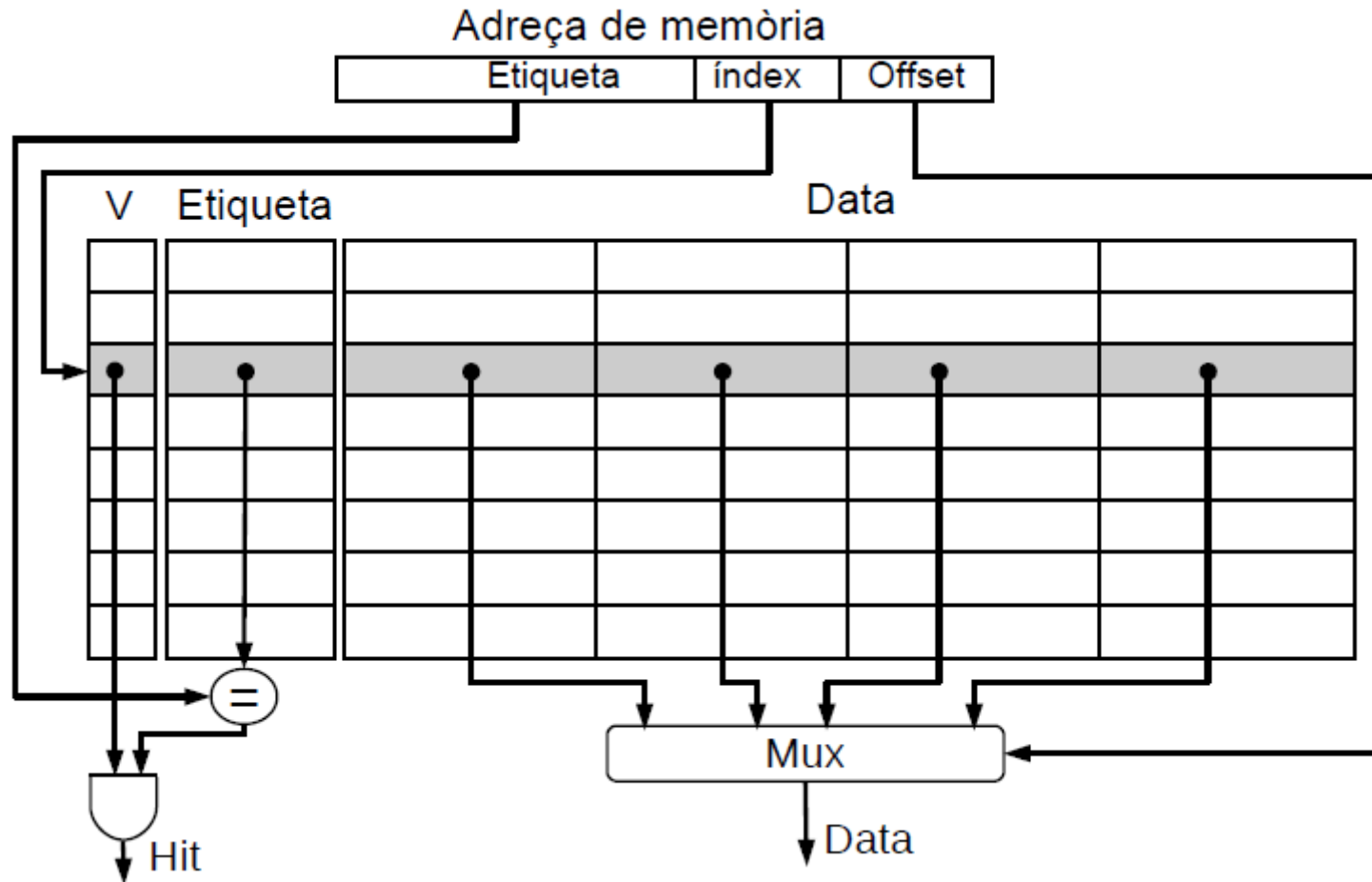
Index = 0x18

Offset = 0x0F

Etiqueta = 0x020020



Esquema d'una lectura



Mida òptima del bloc (a igual capaciat total)

- Què és millor?

Doble núm de línies...



... o doble llarg de blocs?



Mida òptima del bloc (a igual capacitat total)

- Què és millor?

Doble núm de línies...



... o doble llarg de blocs?



- Llargades majors...

... permetem aprofitar millor la localitat espacial → major hit rate 😊

... però hi ha menys línies → més competició → major miss rate ☹️

Mida òptima del bloc (a igual capacitat total)

- Què és millor?

Doble núm de línies...



... o doble llarg de blocs?

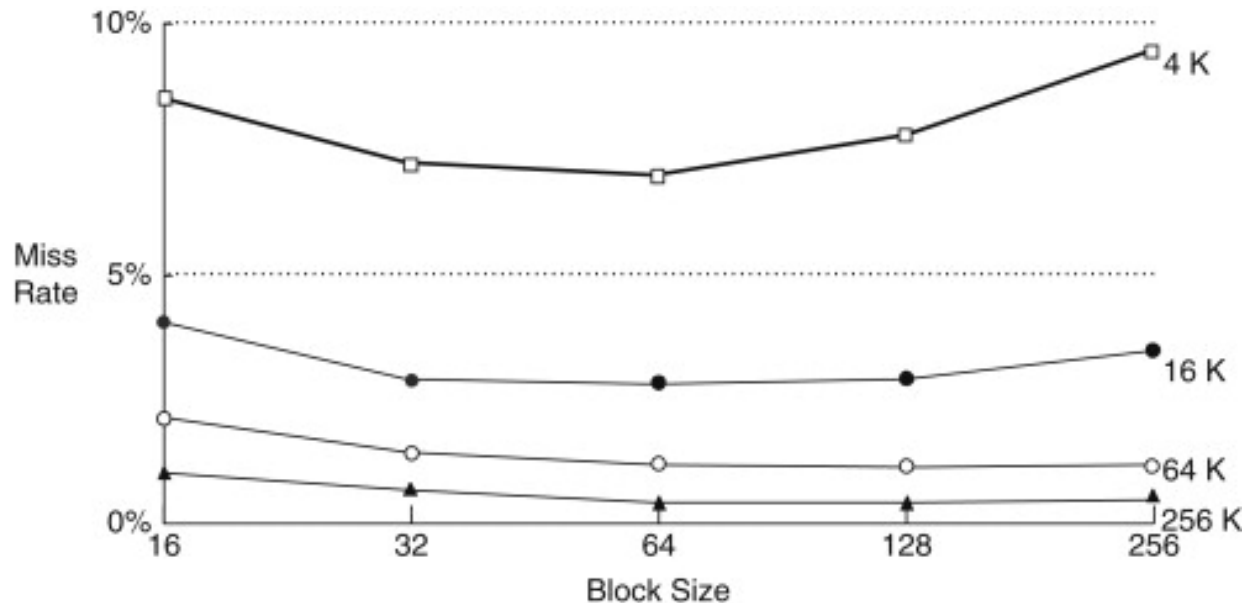


- Llargades majors...

... permetem aprofitar millor la localitat espacial → major hit rate 😊

... però hi ha menys línies → més competició → major miss rate ☹️

- Empíricament...

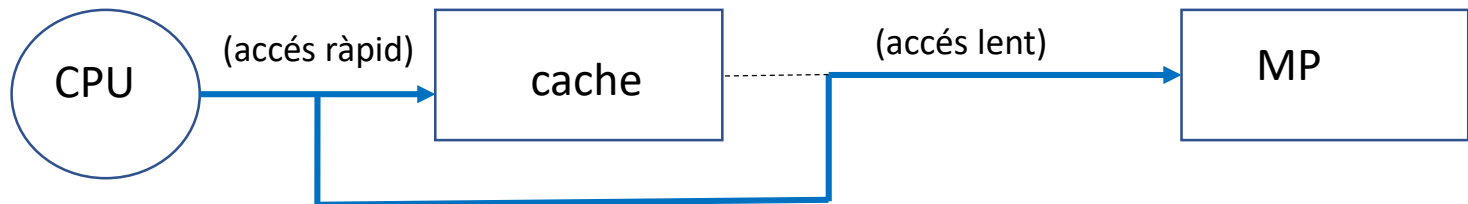


Polítiques d'escriptura en cas d'encert

- Què fer en cas **d'encert** d'escriptura?
 - Podríem actualitzar només el bloc a MC, i no a MP?
 - MC i MP quedarien **inconsistentes**
 - Si un accés posterior reemplaça el bloc inconsistent: **Error!**

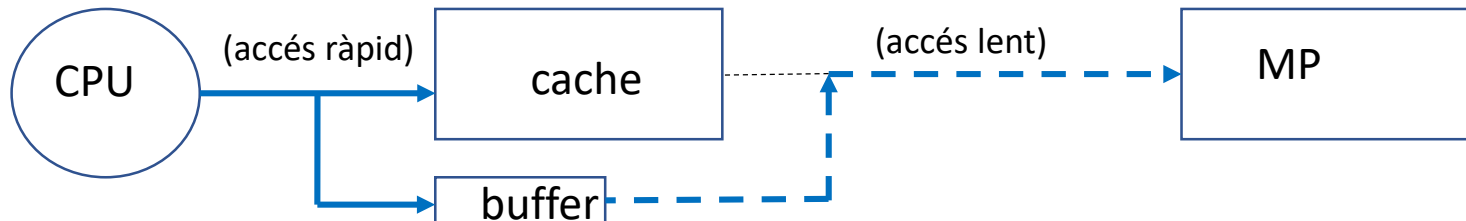
Polítiques d'escriptura en cas d'encert

- Què fer en cas **d'encert** d'escriptura?
 - Podríem actualitzar només el bloc a MC, i no a MP?
 - MC i MP quedarien **inconsistents**
 - Si un accés posterior reemplaça el bloc inconsistent: **Error!**
- **Escriptura immediata (write-through)**
 - Actualitza **MC** i també **MP**
 - Però les escriptures a MP tarden molt més!



Polítiques d'escriptura en cas d'encert

- Què fer en cas **d'encert** d'escriptura?
 - Podríem actualitzar només el bloc a MC, i no a MP?
 - MC i MP quedarien **inconsistents**
 - Si un accés posterior reemplaça el bloc inconsistent: **Error!**
- **Espectura immediata (write-through)**
 - Actualitza **MC** i també **MP**
 - Però les escriptures a MP tarden molt més!
 - Solució: Escrivim en un **buffer d'escriptura**
 - Conté dades pendents d'escriure a memòria
 - I la CPU pot continuar immediatament 😊
 - Només es bloqueja si el buffer està ple



Polítiques d'escriptura en cas d'encert

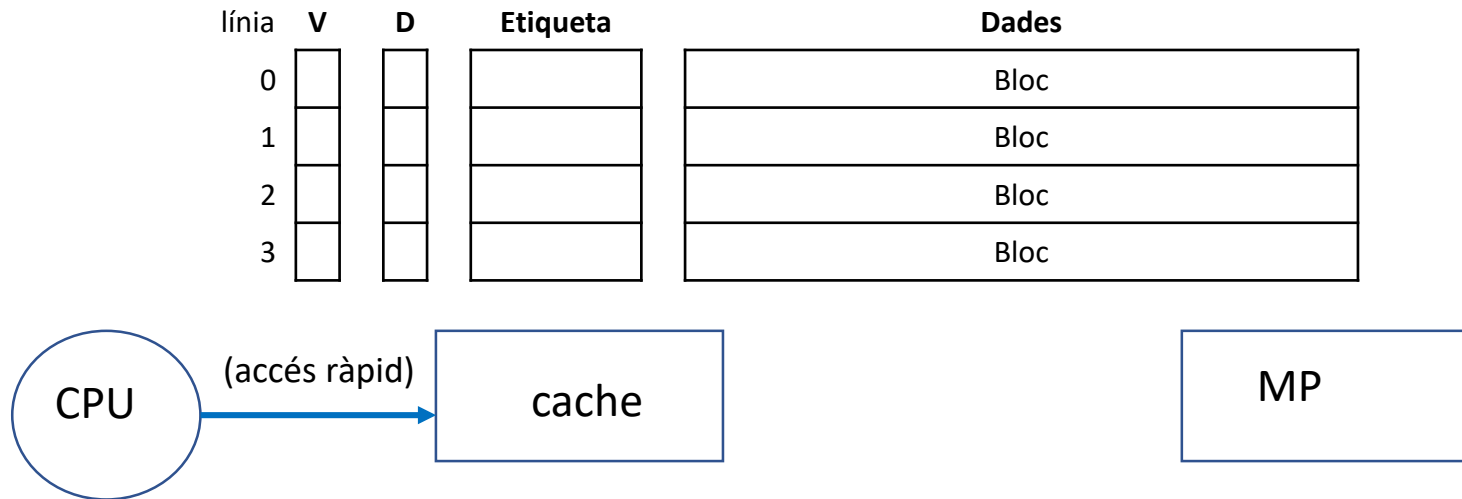
- **Escriptura retardada (write-back)**
 - Actualitza només **MC**
 - Escriptura ràpida 😊
 - Però el bloc en MP queda **inconsistent!**



Polítiques d'escriptura en cas d'encert

- **Escriptura retardada (write-back)**

- Actualitza només **MC**
 - Escriptura ràpida 😊
- Però el bloc en MP queda **inconsistent!**
- Solució: afegim un **Dirty Bit (D)** a cada línia
 - Indica si el bloc és inconsistent (D=1) o no s'ha modificat (D=0)



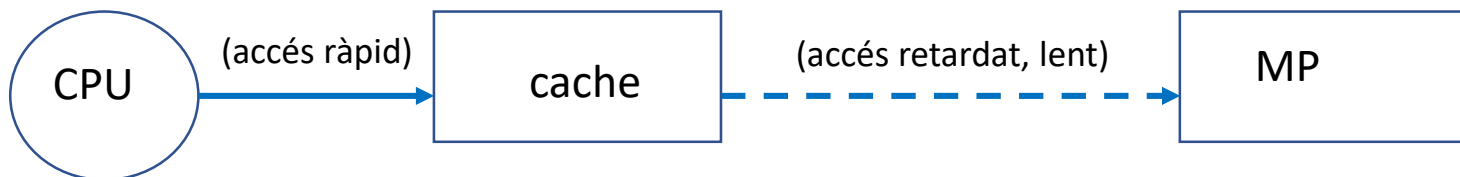
Polítiques d'escriptura en cas d'encert

- **Escriptura retardada (write-back)**

- Actualitza només **MC**
 - Escriptura ràpida 😊
- Però el bloc en MP queda **inconsistent!**
- Solució: afegim un **Dirty Bit (D)** a cada línia
 - Indica si el bloc és inconsistent (D=1) o no s'ha modificat (D=0)

línia	V	D	Etiqueta	Dades
0				Bloc
1				Bloc
2				Bloc
3				Bloc

- Un bloc modificat no s'escriurà a MP **fins que sigui reemplaçat**
 - Pot ser degut a un miss de lectura o d'escriptura
 - Vàries escriptures al mateix bloc requereixen 1 sol accés a MP 😊



Polítiques d'escriptura en cas de fallada

- Què fer en cas **de fallada** d'escriptura?
- Alternatives per a **escriptura immediata**
 - **Escriptura amb assignació (write allocate)**
 - Copiar primer el bloc de MP a MC, i procedir com si fos un encert, és a dir, escrivint als dos llocs
 - (és l'única que implementa Mars)

Polítiques d'escriptura en cas de fallada

- Què fer en cas de **fallada** d'escriptura?
- Alternatives per a **escriptura immediata**
 - **Escriptura amb assignació (write allocate)**
 - Copiar primer el bloc de MP a MC, i procedir com si fos un encert, és a dir, escrivint als dos llocs
 - (és l'única que implementa Mars)
 - **Escriptura sense assignació (write no-allocate)**
 - Sols s'escriu la dada en MP (posteriors accessos seguiran fallant)
 - Alguns programes escriuen molts blocs sencers abans de llegir-los (inicialització)

Polítiques d'escriptura en cas de fallada

- Què fer en cas de **fallada** d'escriptura?
- Alternatives per a escriptura immediata
 - **Escriptura amb assignació (write allocate)**
 - Copiar primer el bloc de MP a MC, i procedir com si fos un encert, és a dir, escrivint als dos llocs
 - (és l'única que implementa Mars)
 - **Escriptura sense assignació (write no-allocate)**
 - Sols s'escriu la dada en MP (posteriors accessos seguiran fallant)
 - Alguns programes escriuen molts blocs sencers abans de llegir-los (inicialització)
- Per a **escriptura retardada**
 - **Escriptura amb assignació (write allocate)**
 - Copiar primer el bloc de MP a MC, i procedir com si fos un encert

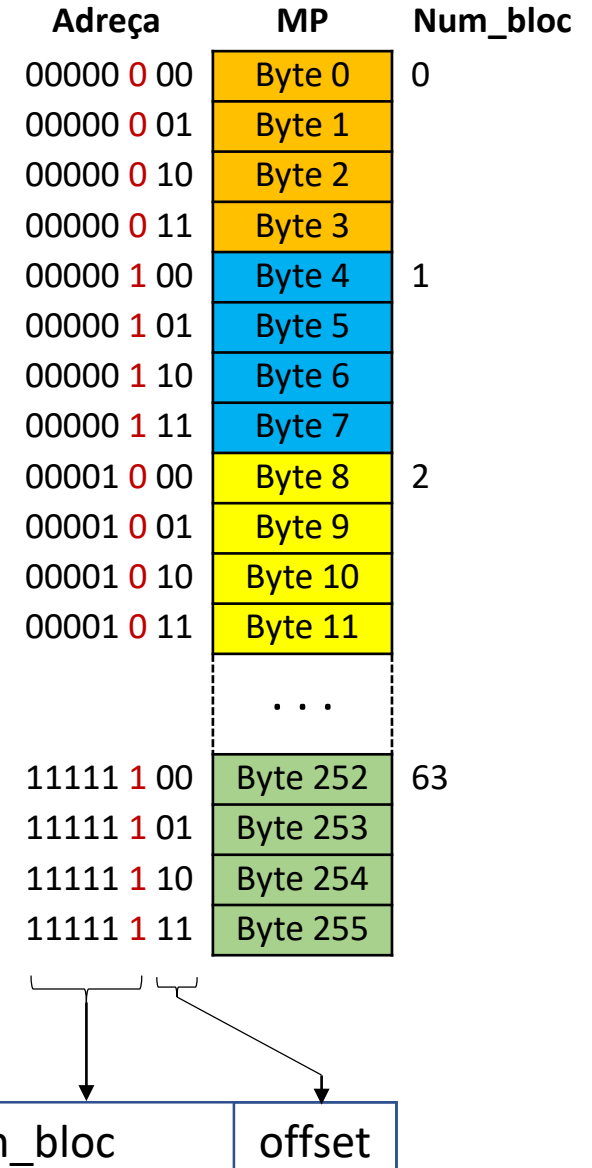
Exemple

Processador de 8 bits

Mida de bloc B = 4 bytes

→ 2 bits d'*offset*

→ 6 bits de *num_bloc*



Exemple

Processador de 8 bits

Mida de bloc B = 4 bytes

→ 2 bits d'*offset*

→ 6 bits de *num_bloc*

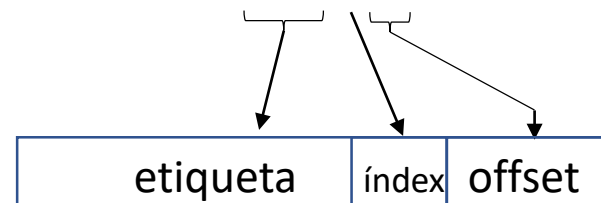
Cache de 2 línies (corresp. directa)

→ 1 bit d'*índex*

Memòria cache

Índex						
línia	V	Etiqueta	Dades			
0	0					
1	0					

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		
11111 1 00	Byte 252	63
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	



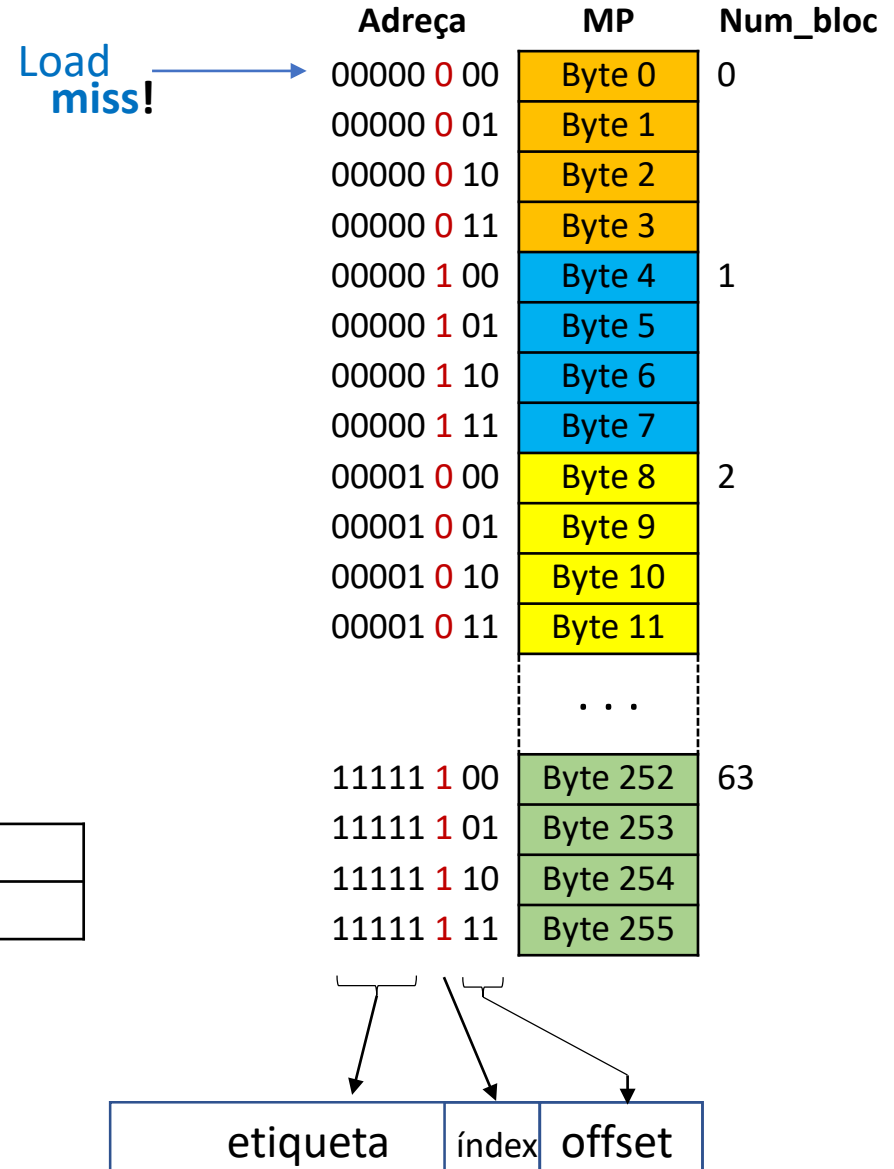
Exemple: lectures

Lectura

Load @ = 00000000

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	0					
1	0					



Exemple: lectures

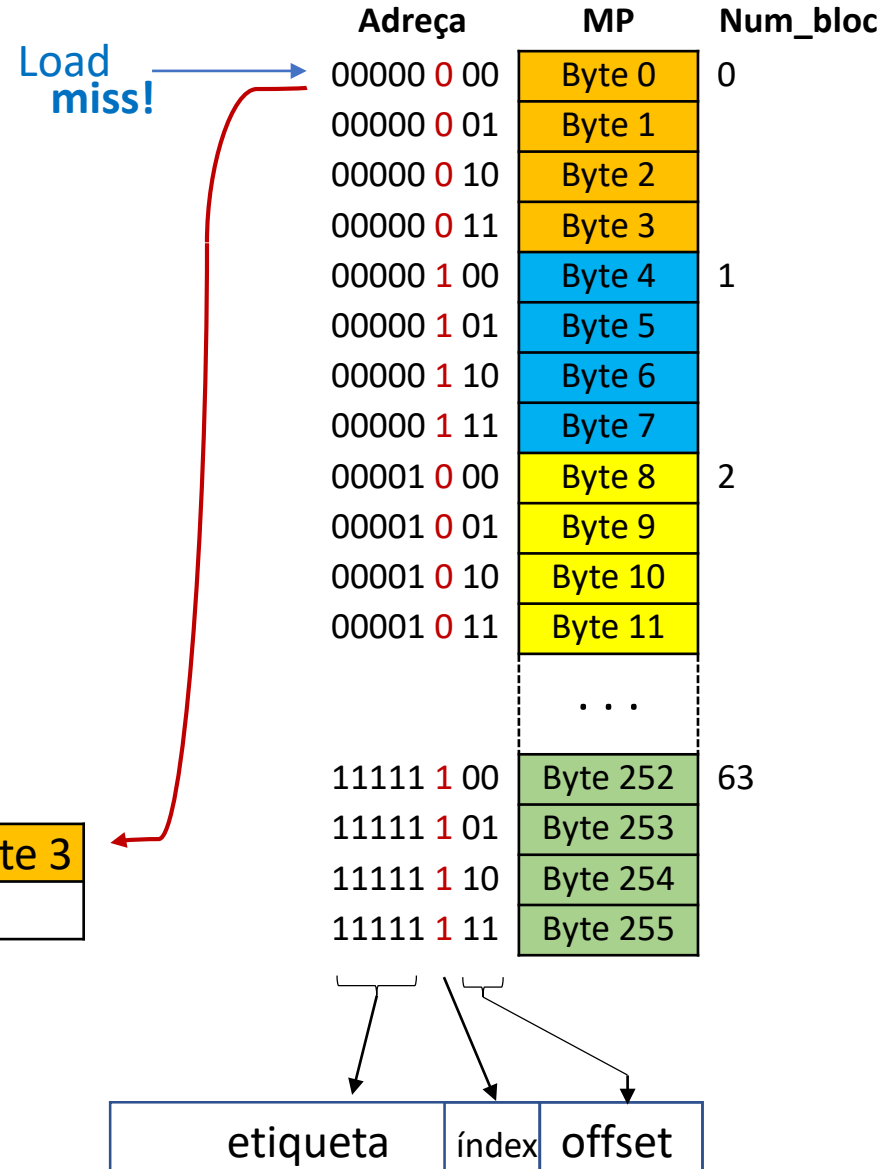
Lectura

Load @ = 00000000

copiem el bloc a la línia 0

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	0					



Exemple: lectures

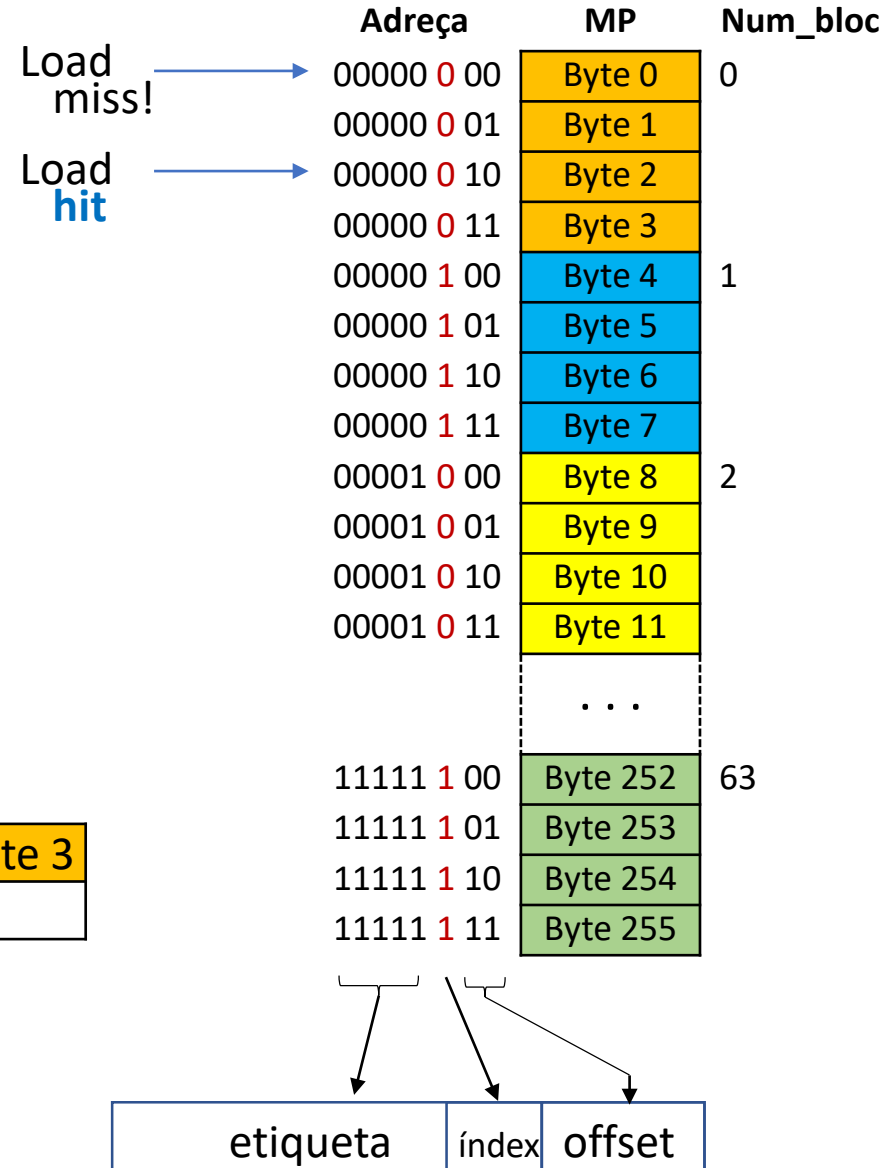
Lectura

Load @ = 00000000

Load @ = 00000010

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	0					



Exemple: lectures

Lectura

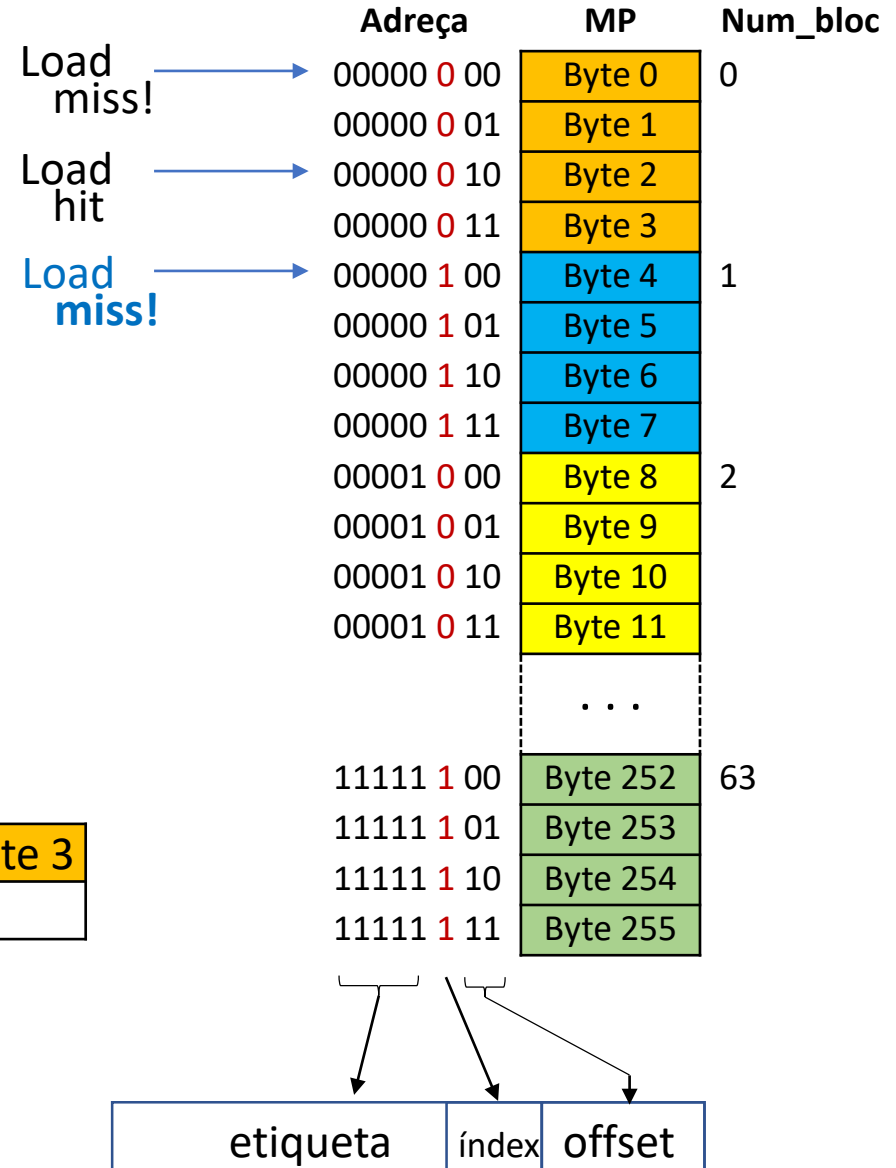
Load @ = 00000000

Load @ = 00000010

Load @ = 00000100

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	0					



Exemple: lectures

Lectura

Load @ = 00000000

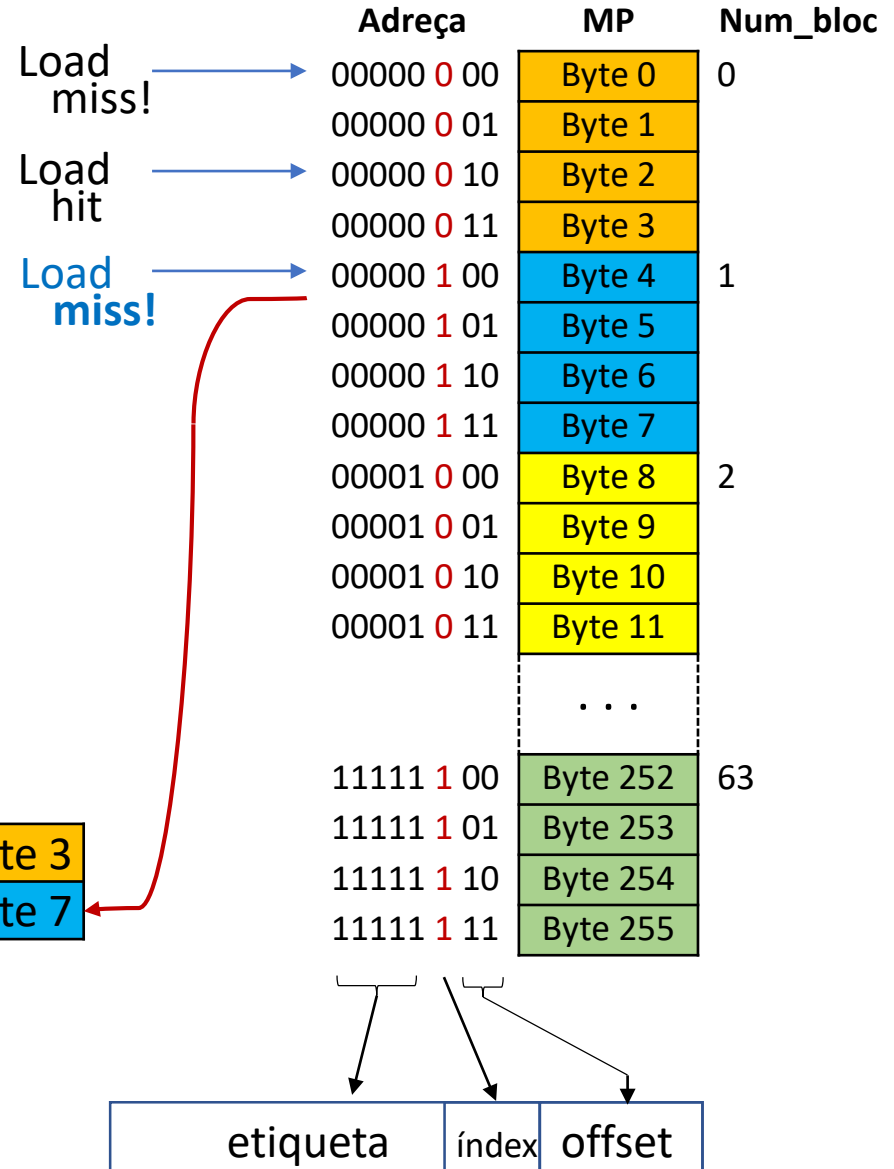
Load @ = 00000010

Load @ = 00000100

copiem el bloc a la línia 1

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: lectures

Lectura

Load @ = 00000000

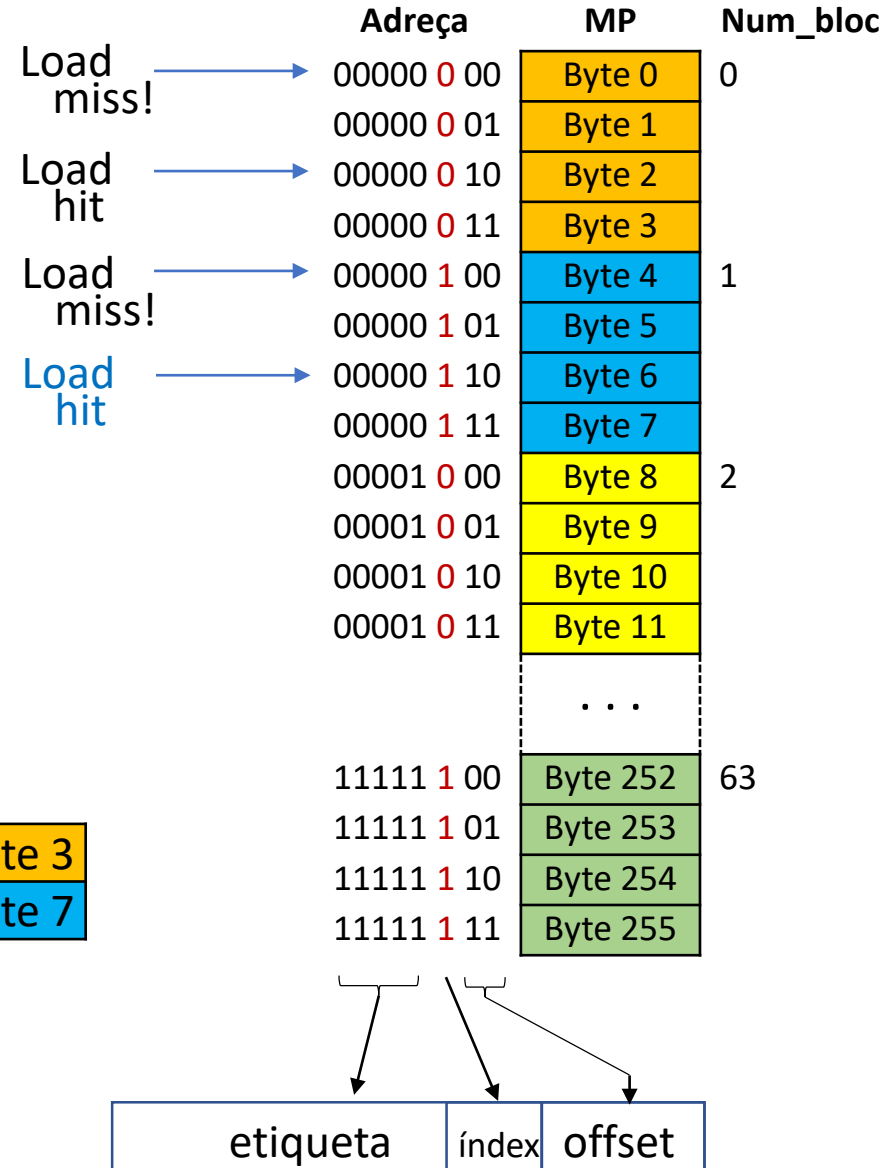
Load @ = 00000010

Load @ = 00000100

Load @ = 00000110

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



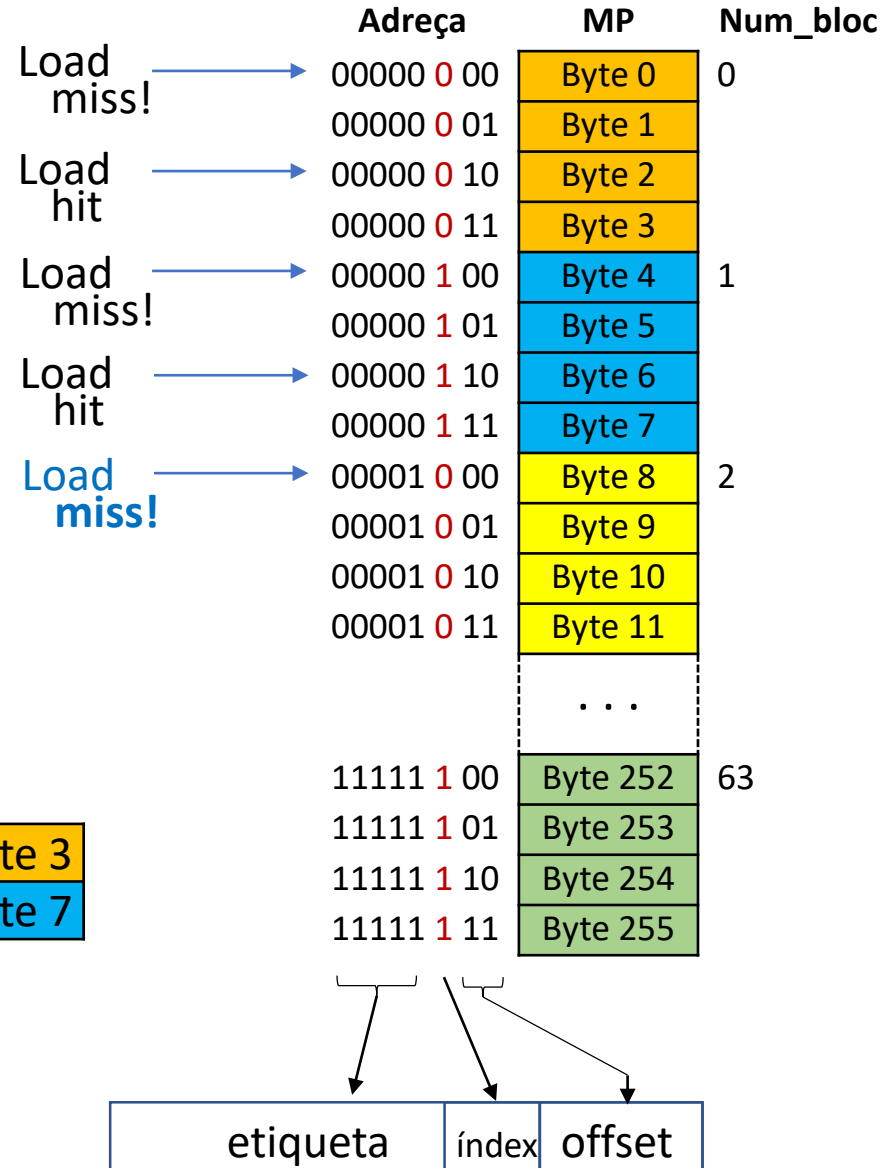
Exemple: lectures

Lectura

Load @ = 00000000
 Load @ = 00000010
 Load @ = 00000100
 Load @ = 00000110
 Load @ = 00001000

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: lectures

Lectura

Load @ = 00000000

Load @ = 00000010

Load @ = 00000100

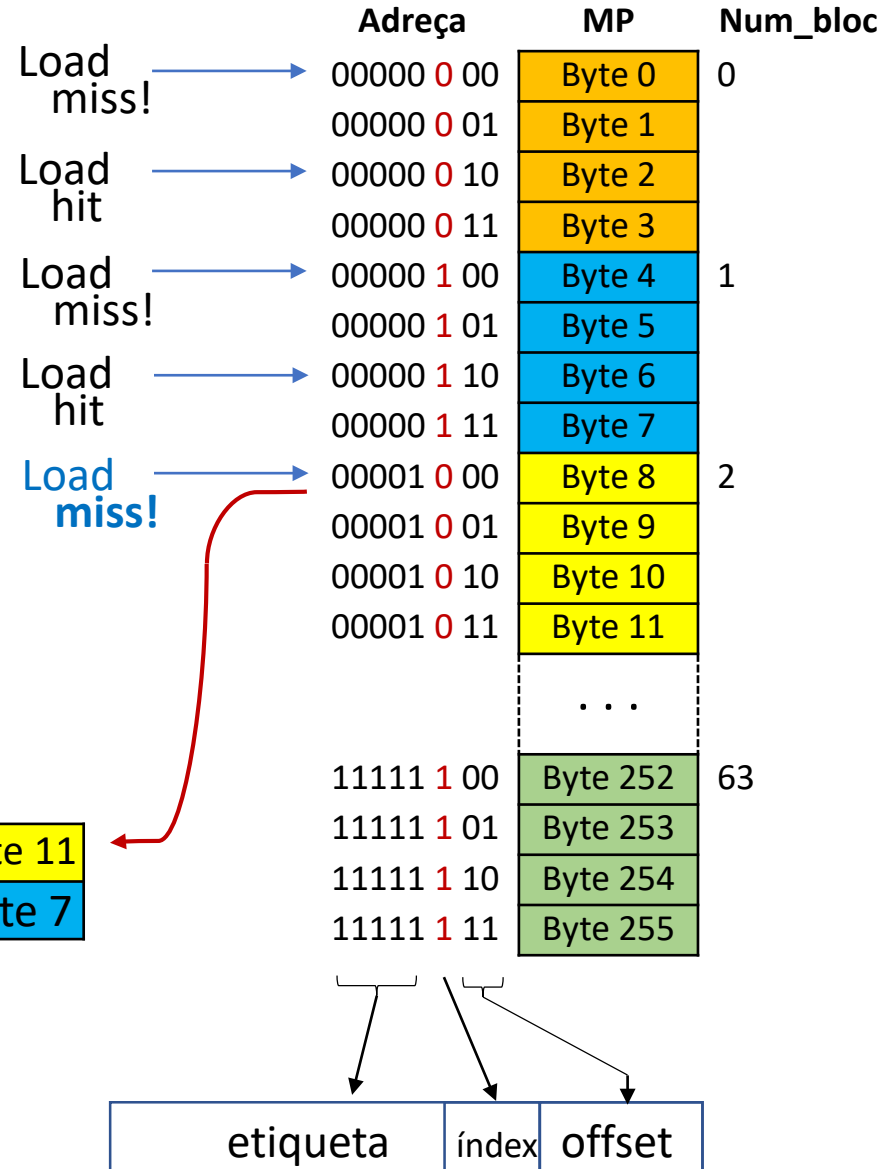
Load @ = 00000110

Load @ = 00001000

copiem el bloc a la línia 0

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Escriptura immediata
amb assignació

Exemple: escriptura

Espectura immediata amb assignació

Store @ = 00001010

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	



Exemple: escriptura

Espectura immediata amb assignació

Store @ = 00001010
escrivim a MC i MP

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	



Exemple: escriptura

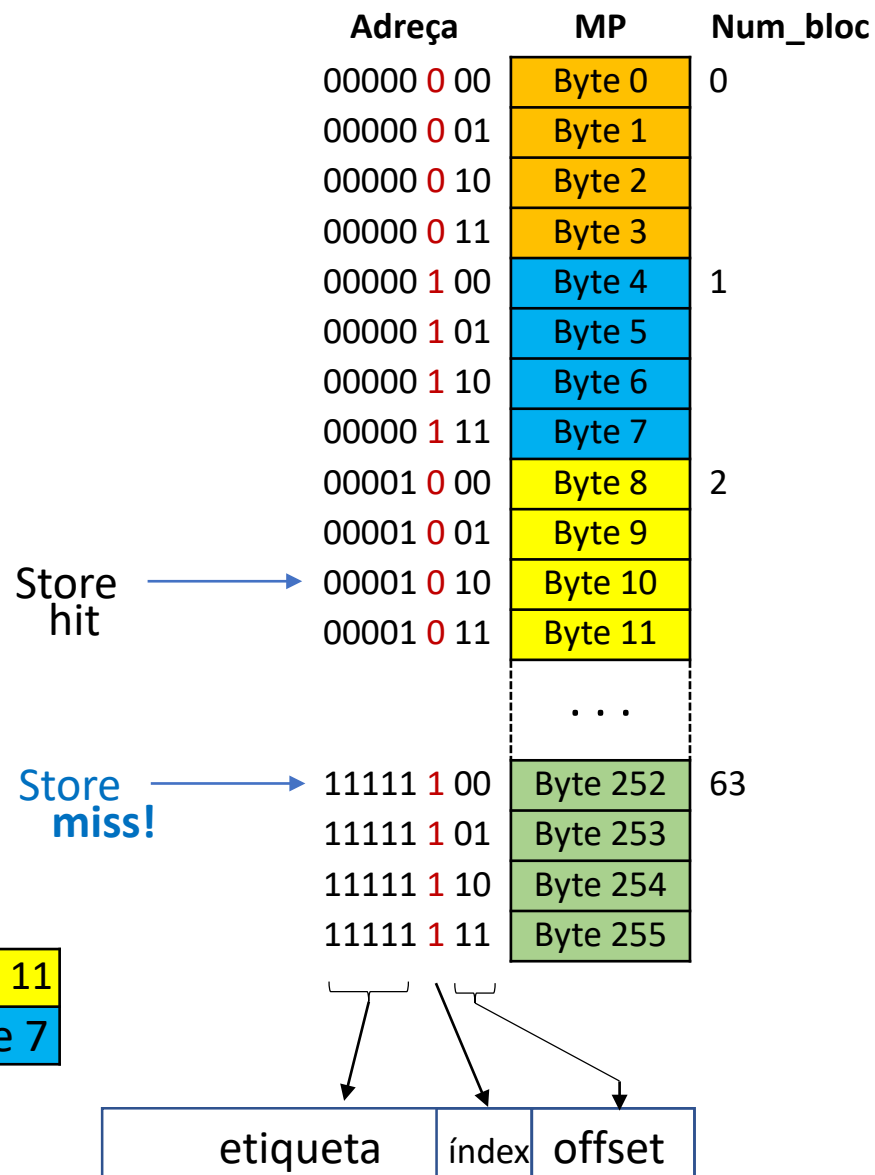
Espectura immediata amb assignació

Store @ = 00001010

Store @ = 11111100

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Espectura immediata amb assignació

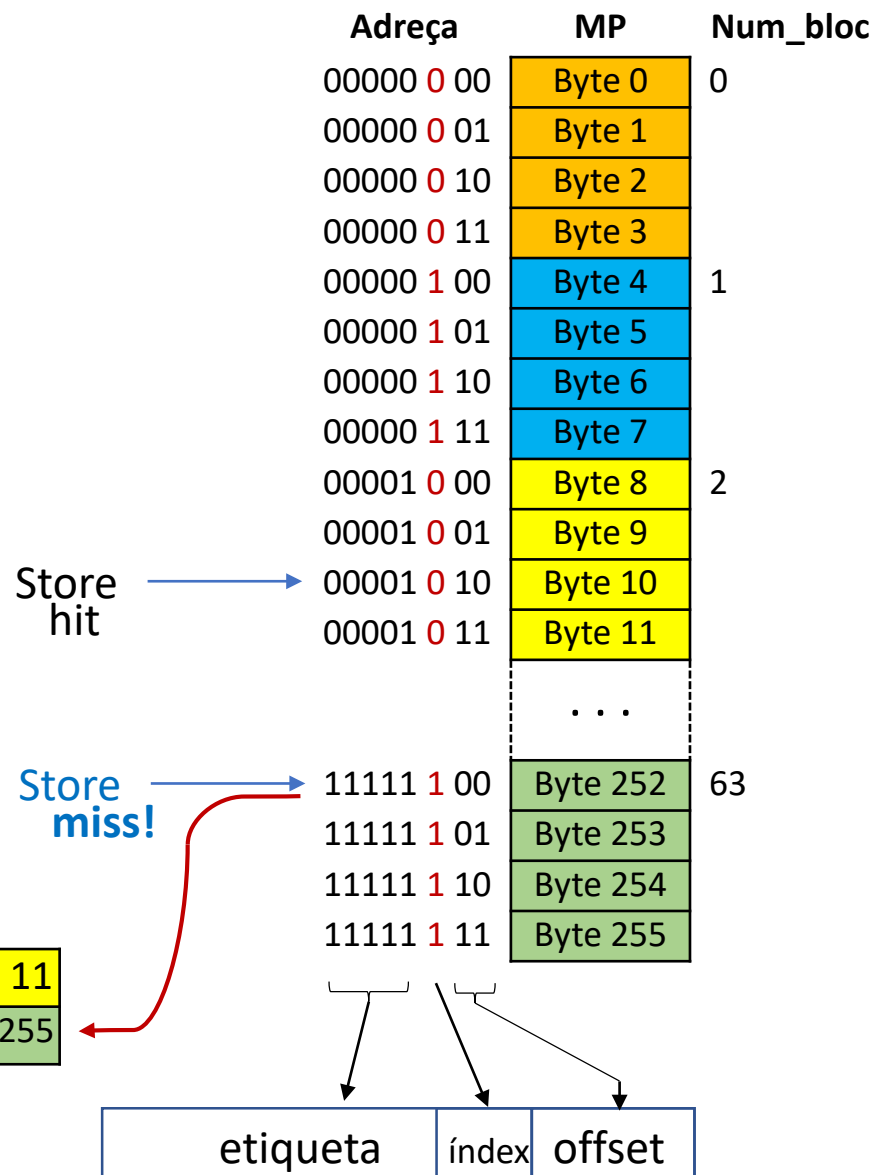
Store @ = 00001010

Store @ = 11111100

copiem el bloc a la línia 1

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	11111	Byte 252	Byte 253	Byte 254	Byte 255



Exemple: escriptura

Escriptura immediata amb assignació

Store @ = 00001010

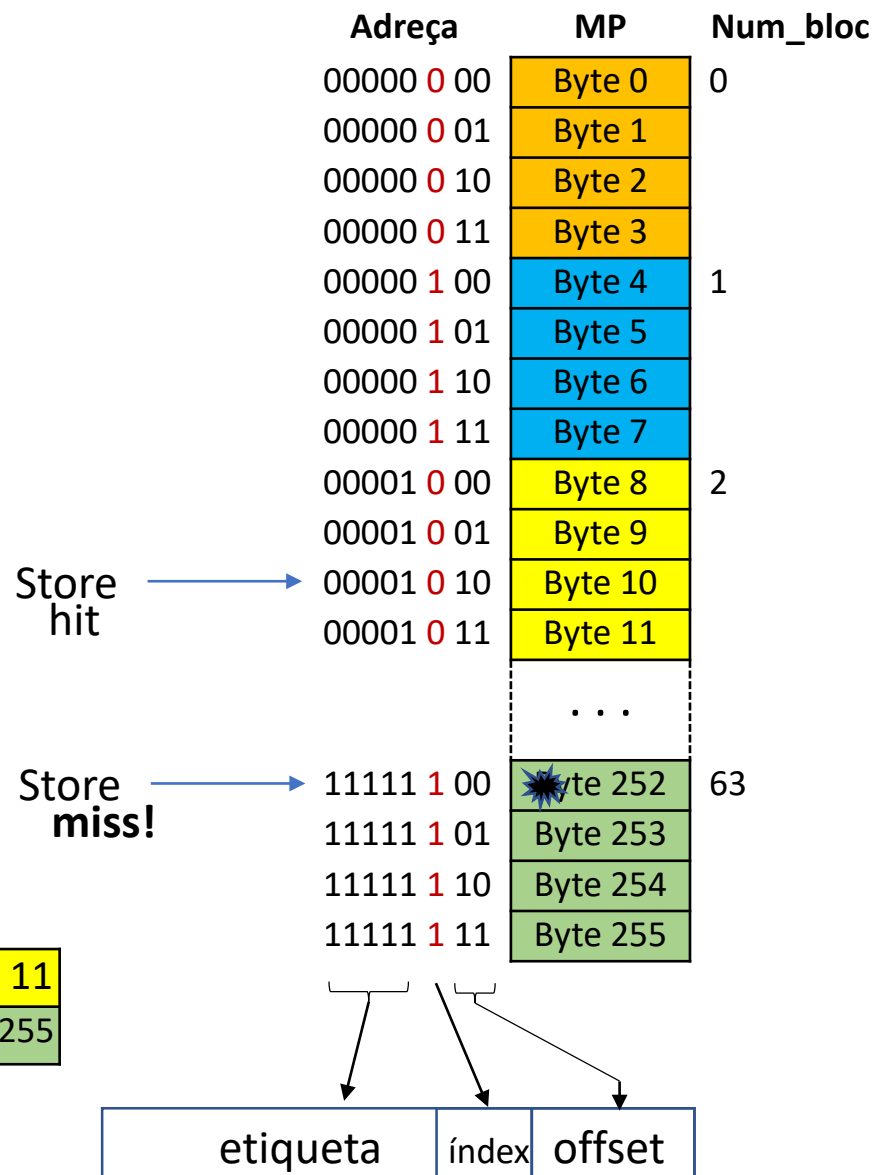
Store @ = 11111100

copiem el bloc a la línia 1

escrivim a MC i MP

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	11111	Byte 252	Byte 253	Byte 254	Byte 255



Exemple: escriptura

Escriptura immediata
sense assignació

Exemple: escriptura

Escriptura immediata sense assignació

Store @ = 00001010

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	



Exemple: escriptura

Espectura immediata sense assignació

Store @ = 00001010

→ Escriu a MC i MP

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	



Exemple: escriptura

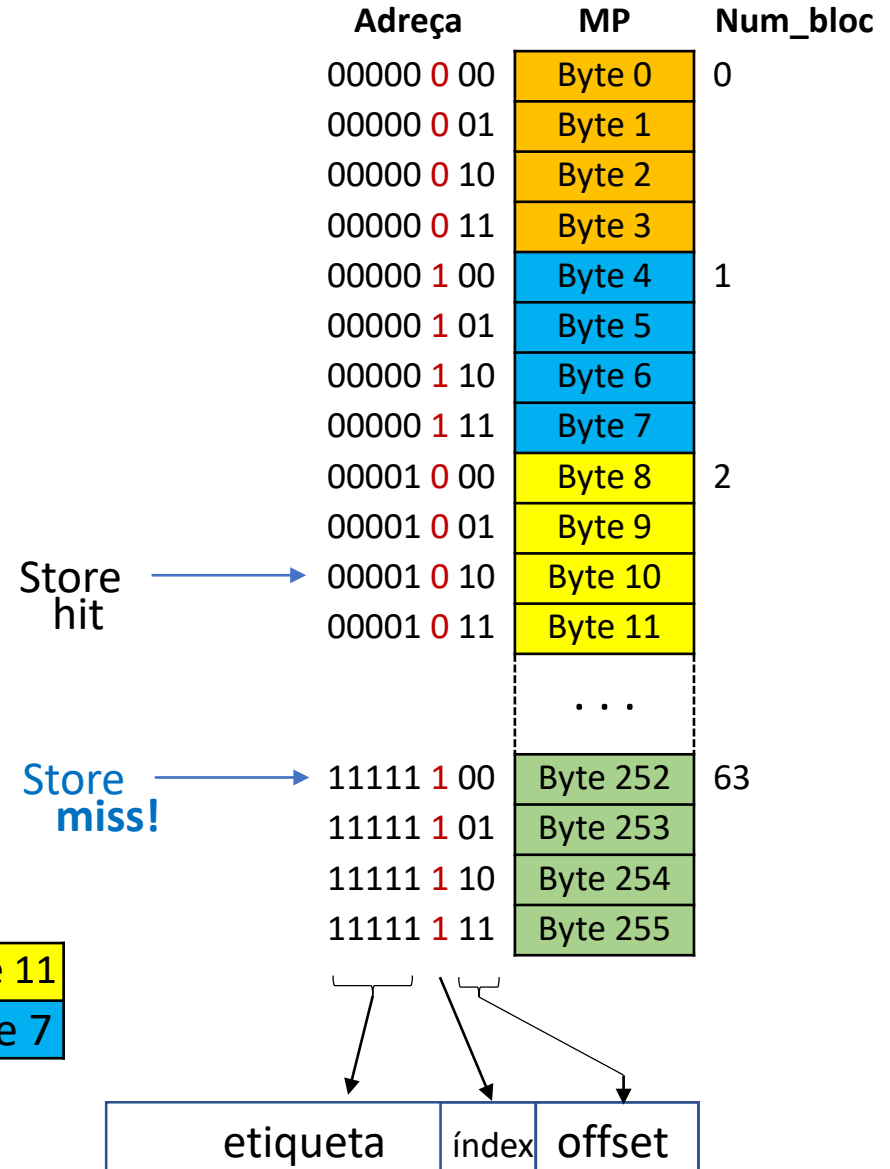
Espectura immediata sense assignació

Store @ = 00001010

Store @ = 11111100

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Espectura immediata sense assignació

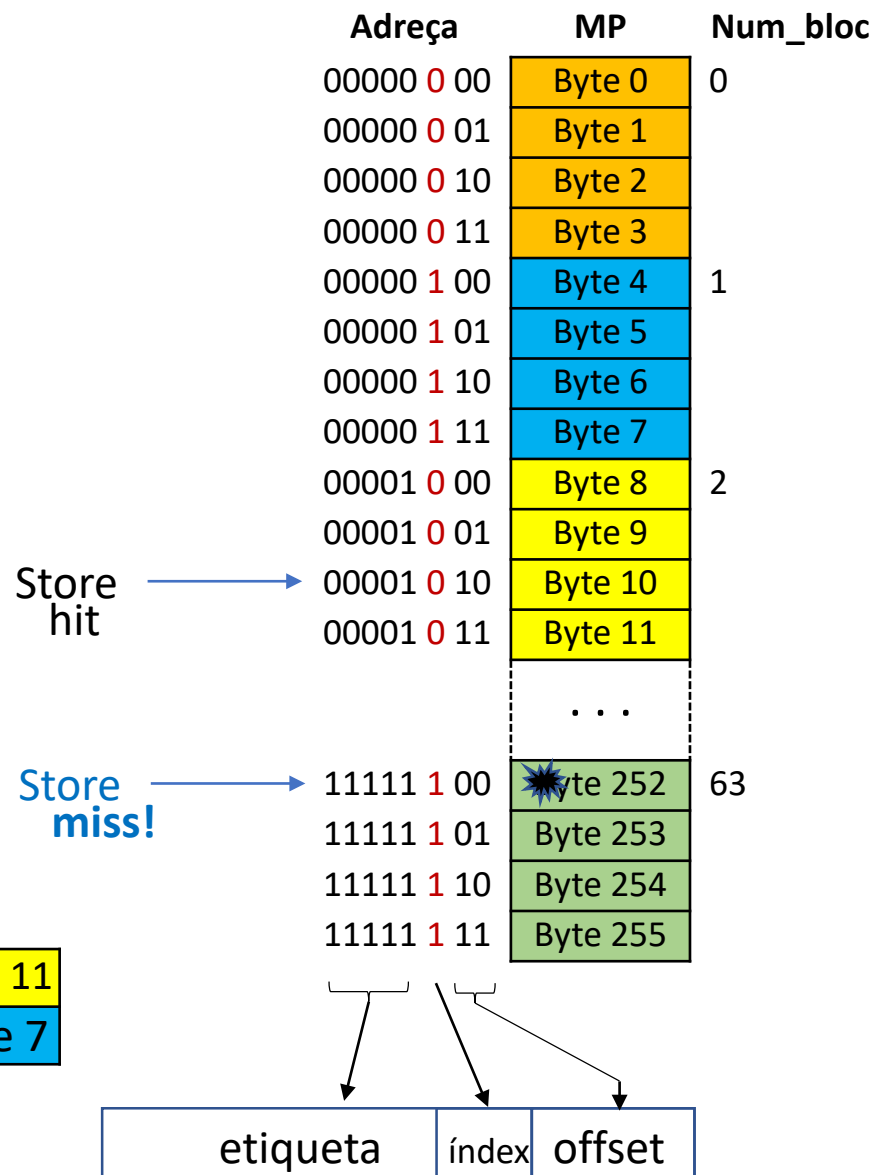
Store @ = 00001010

Store @ = 11111100

NO copia el bloc a MC
sols escriu a MP

Memòria cache

Índex línia	V	Etiqueta	Dades			
0	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

**Espectura retardada
amb assignació**

Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	0	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	0	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	

etiqueta

índex

offset

Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Només escrivim a MC
posem D=1

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	0	00000	Byte 4	Byte 5	Byte 6	Byte 7

Store hit

Adreça	MP	Num_bloc
00000 0 00	Byte 0	0
00000 0 01	Byte 1	
00000 0 10	Byte 2	
00000 0 11	Byte 3	
00000 1 00	Byte 4	1
00000 1 01	Byte 5	
00000 1 10	Byte 6	
00000 1 11	Byte 7	
00001 0 00	Byte 8	2
00001 0 01	Byte 9	
00001 0 10	Byte 10	
00001 0 11	Byte 11	
...		63
11111 1 00	Byte 252	
11111 1 01	Byte 253	
11111 1 10	Byte 254	
11111 1 11	Byte 255	

etiqueta

índex

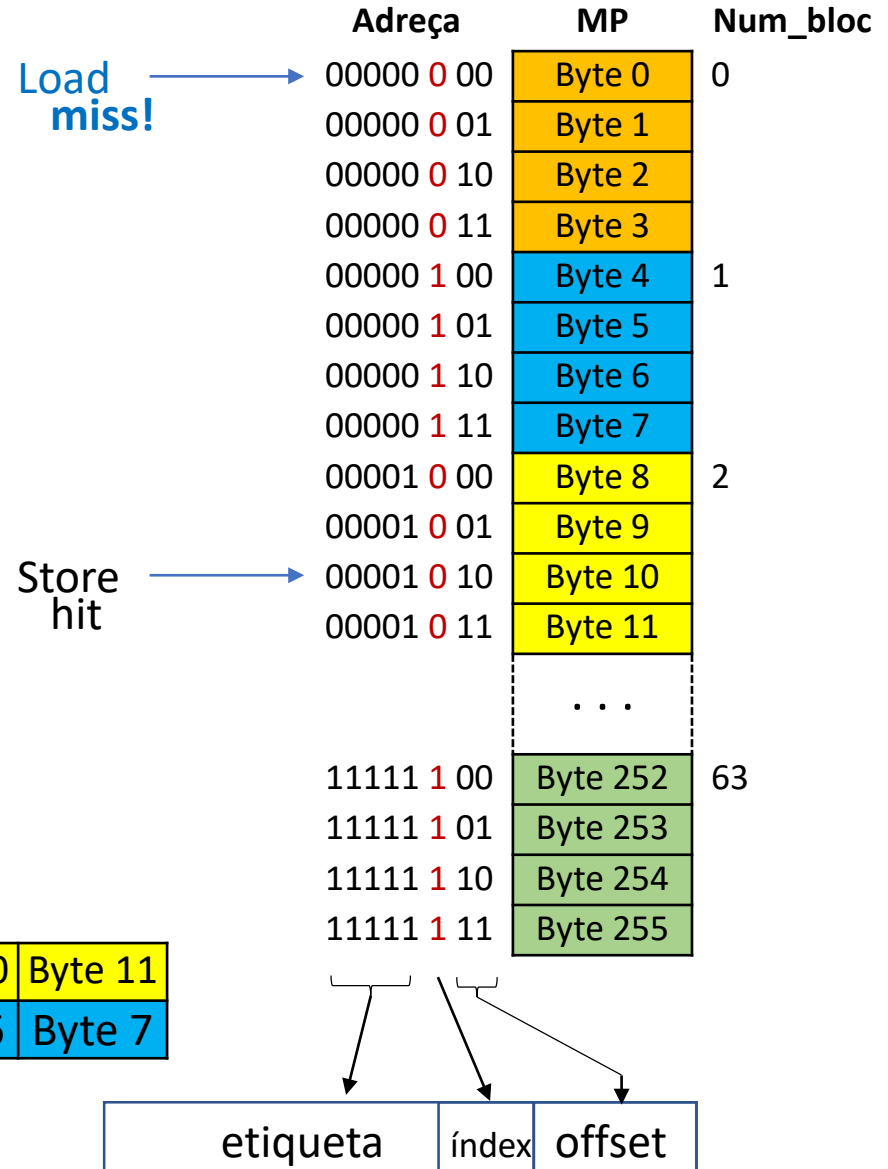
offset

Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Load @ = 00000000



Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

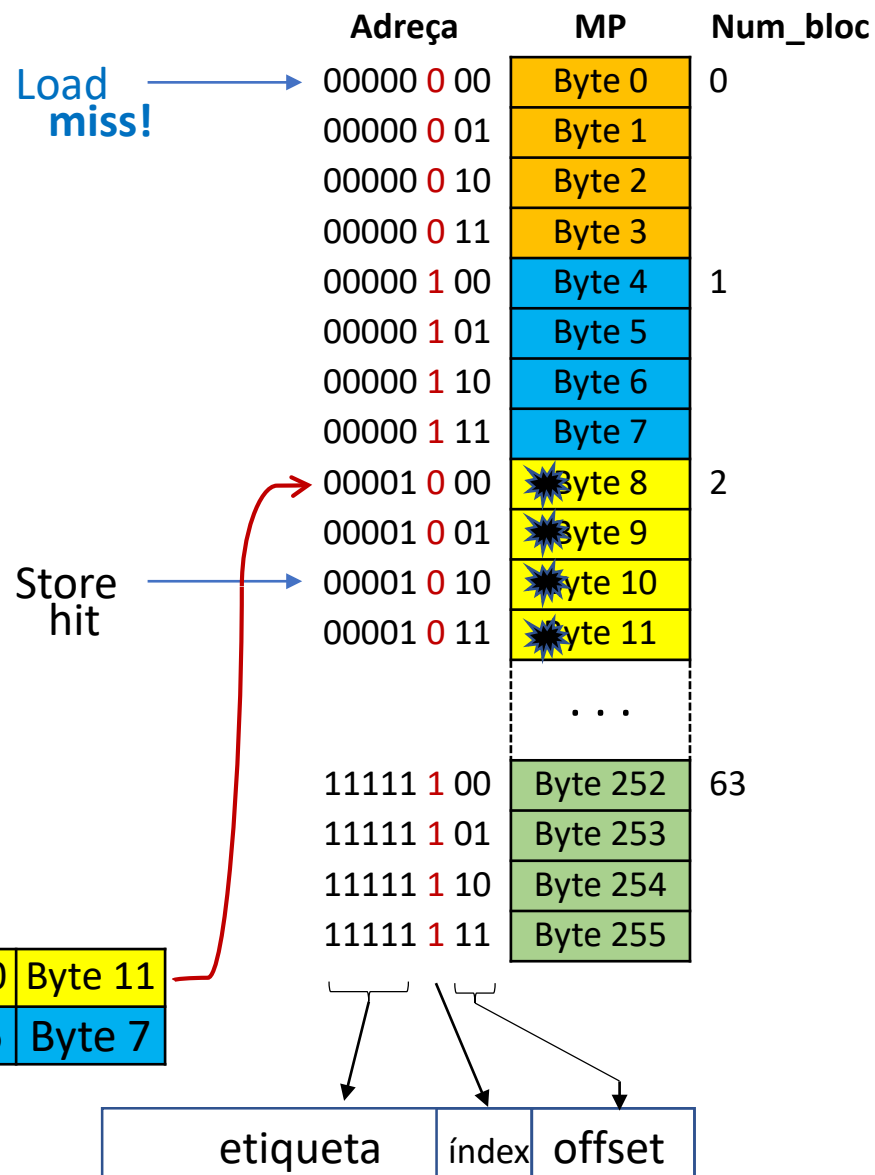
Load @ = 00000000

la línia 0 té D=1:

1.- escrivim línia 0 a MP

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	1	00001	Byte 8	Byte 9	Byte 10	Byte 11
1	1	0	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Load @ = 00000000

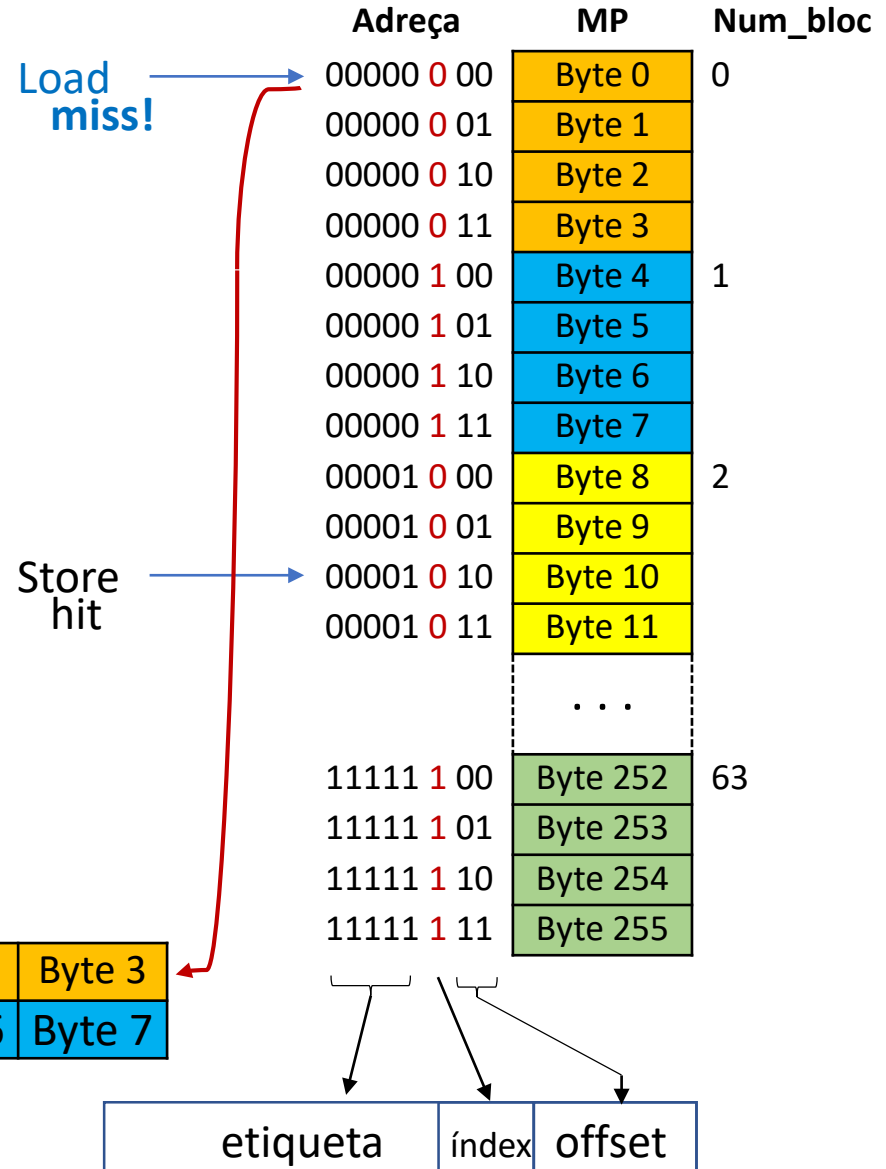
falla a la línia 0

1.- escrivim línia 0 a MP

2.- reemplaçem línia 0
i posem D=0

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	0	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	0	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Espectura retardada amb assignació

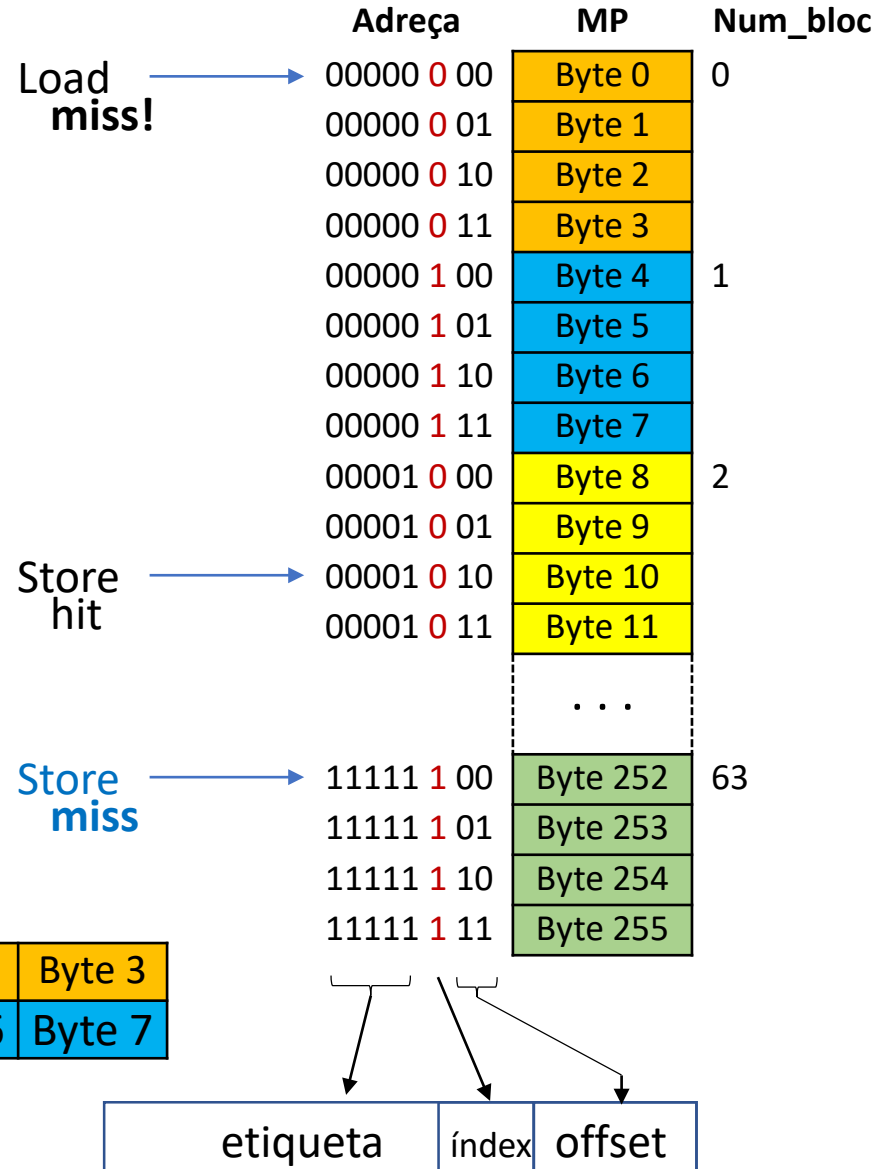
Store @ = 00001010

Load @ = 00000000

Store @ = 11111100

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	0	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	0	00000	Byte 4	Byte 5	Byte 6	Byte 7



Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Load @ = 00000000

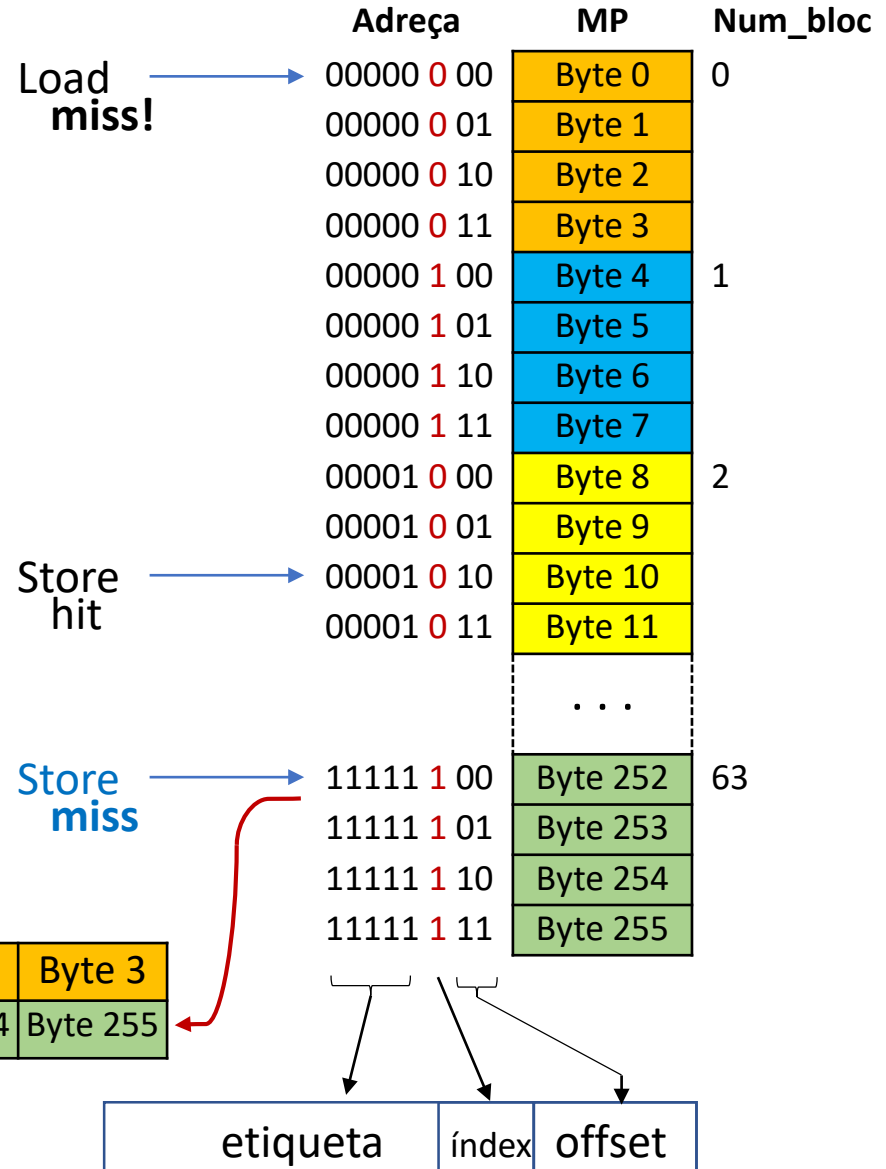
Store @ = 11111100

la línia 1 té D=0

copiem el bloc a la línia 1

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	0	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	0	11111	Byte 252	Byte 253	Byte 254	Byte 255



Exemple: escriptura

Espectura retardada amb assignació

Store @ = 00001010

Load @ = 00000000

Store @ = 11111100

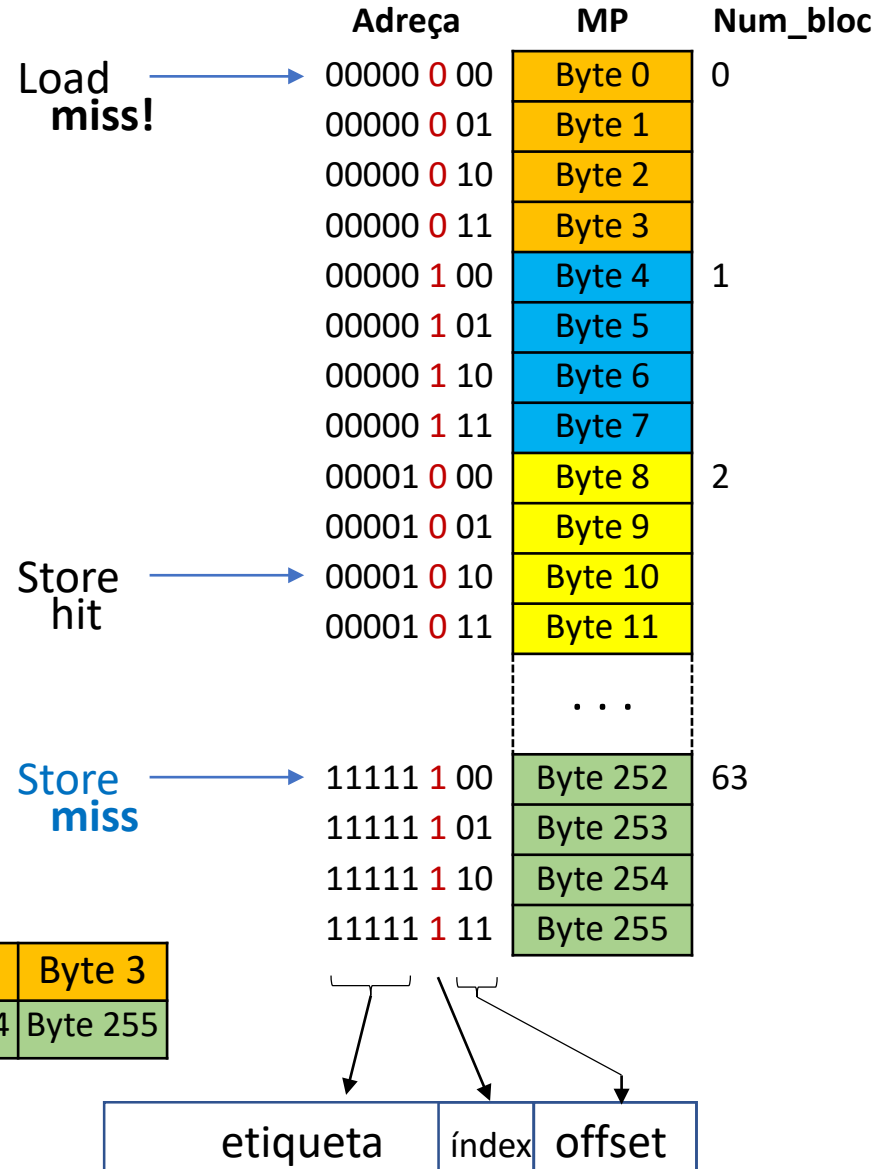
la línia 1 té D=0

copiem el bloc a la línia 1

escriuim solament a a MC
i posem D=1

Memòria cache

Índex línia	V	D	Etiqueta	Dades			
0	1	0	00000	Byte 0	Byte 1	Byte 2	Byte 3
1	1	1	00000	Byte 252	Byte 253	Byte 254	Byte 255



Memòria Cache

- Introducció
- Aspectes de disseny
- **Impacte en el rendiment**
 - Temps d'accés a memòria
 - Model de temps simplificat
 - Impacte de la cache en el rendiment
 - Temps d'accés mitjà
- Millores de rendiment

Temps d'accès a memòria

Definim el temps d'accés a memòria

- En cas de hit
 - $t_{\text{accés}} = t_h$
 - t_h : temps d'encert, que inclou:
 - comprovar l'etiqueta (hit/miss)

Temps d'accés a memòria

Definim el temps d'accés a memòria

- En cas de hit
 - $t_{\text{accés}} = t_h$
 - t_h : temps d'encert, que inclou:
 - comprovar l'etiqueta (hit/miss)
 - llegir o escriure la dada a la MC (al mateix temps)

Temps d'accés a memòria

Definim el temps d'accés a memòria

- En cas de hit
 - $t_{\text{accés}} = t_h$
 - t_h : temps d'encert, que inclou:
 - comprovar l'etiqueta (hit/miss)
 - llegir o escriure la dada a la MC (al mateix temps)
- En cas de miss
 - $t_{\text{accés}} = t_h + t_p$
 - t_h : comprovació de l'etiqueta

Temps d'accés a memòria

Definim el temps d'accés a memòria

- En cas de hit
 - $t_{\text{accés}} = t_h$
 - t_h : **temps d'encert**, que inclou:
 - comprovar l'etiqueta (hit/miss)
 - llegir o escriure la dada a la MC (al mateix temps)
- En cas de miss
 - $t_{\text{accés}} = t_h + t_p$
 - t_h : comprovació de l'etiqueta
 - t_p : **temps de penalització**, que inclou:
 - copiar blocs de MP a MC o viceversa

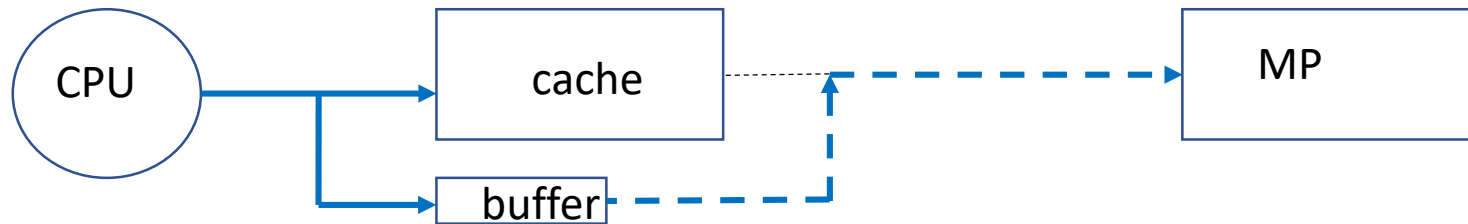
Temps d'accés a memòria

Definim el temps d'accés a memòria

- En cas de hit
 - $t_{\text{accés}} = t_h$
 - t_h : **temps d'encert**, que inclou:
 - comprovar l'etiqueta (hit/miss)
 - llegir o escriure la dada a la MC (al mateix temps)
- En cas de miss
 - $t_{\text{accés}} = t_h + t_p$
 - t_h : comprovació de l'etiqueta
 - t_p : **temps de penalització**, que inclou:
 - copiar blocs de MP a MC o viceversa
 - llegir o escriure la dada a la MC

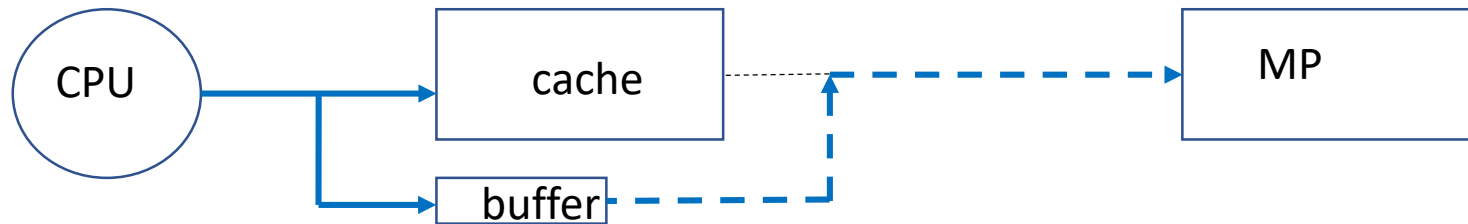
Model de temps simplificat

- Escriptura immediata: assumirem que
 - Hi ha un buffer d'escriptura de mida il·limitada amb totes les escriptures pendents d'anar a MP



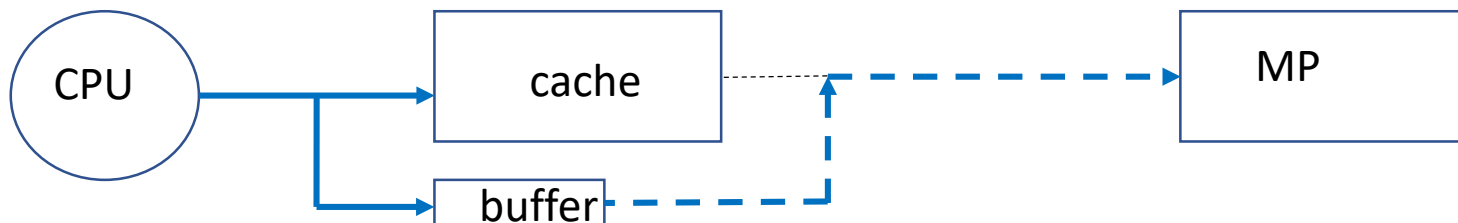
Model de temps simplificat

- Escriptura immediata: assumirem que
 - Hi ha un buffer d'escriptura de mida il·limitada amb totes les escriptures pendents d'anar a MP
 - Un cop ha escrit al buffer, la CPU prossegueix, en paral·lel a les escriptures a MP



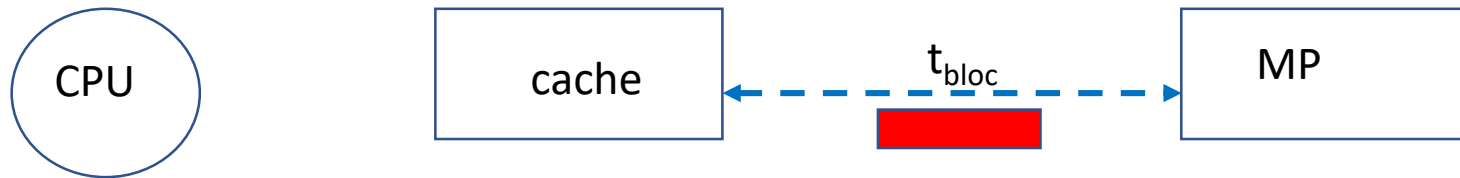
Model de temps simplificat

- Escriptura immediata: assumirem que
 - Hi ha un buffer d'escriptura de mida il·limitada amb totes les escriptures pendents d'anar a MP
 - Un cop ha escrit al buffer, la CPU prossegueix, en paral·lel a les escriptures a MP
 - Cap accés posterior entra en conflicte amb escriptures pendents



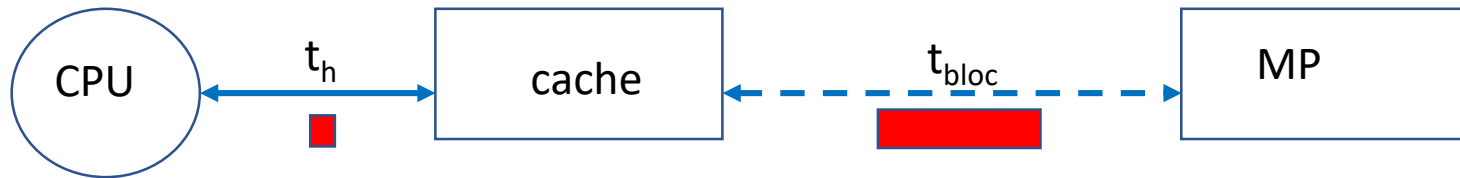
Temps de penalització t_p

- Per tal de modelar t_p , definim
 t_{bloc} : Copiar un bloc de MP a MC o viceversa



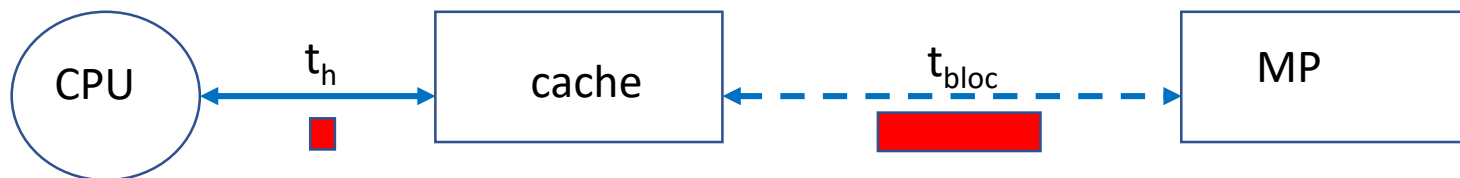
Temps de penalització t_p

- Per tal de modelar t_p , definim
 - t_{bloc} : Copiar un bloc de MP a MC o viceversa
 - t_h : Transferir la dada de MC a CPU o viceversa



Temps de penalització t_p

- Per tal de modelar t_p , definim
 - t_{bloc} : Copiar un bloc de MP a MC o viceversa
 - t_h : Transferir la dada de MC a CPU o viceversa



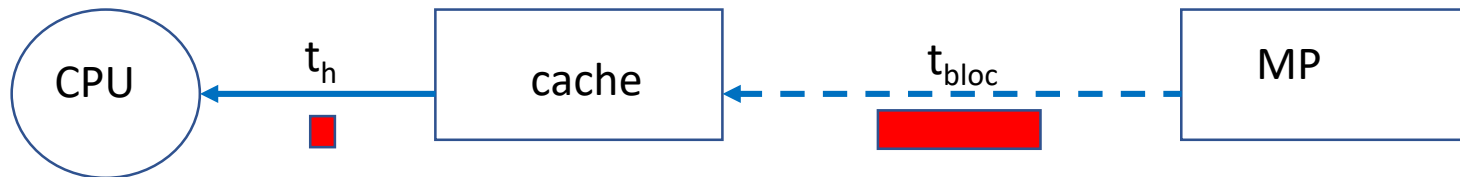
- t_p depèn de la política d'escriptura

Temps de penalització t_p

- Escriptura **immediata sense assignació**

- Fallada de lectura

$t_p = t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU



Temps de penalització t_p

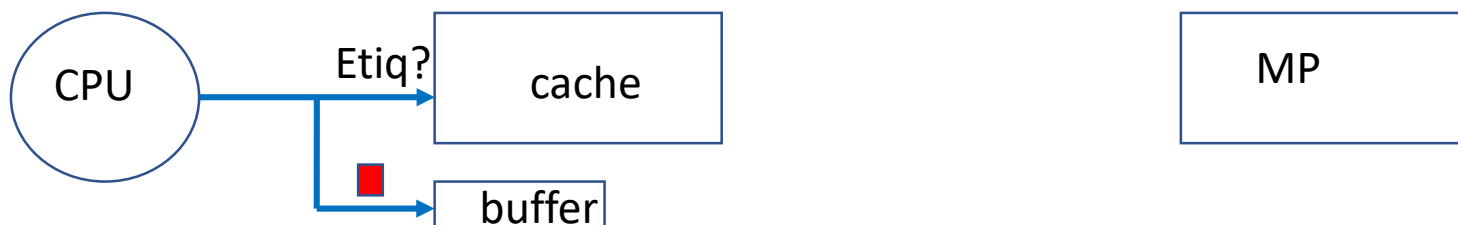
- Escriptura **immediata sense assignació**

- Fallada de lectura

$t_p = t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU

- Fallada d'escriptura

$t_p = 0$: S'escriu al buffer en paral·lel a comprovar etiqueta

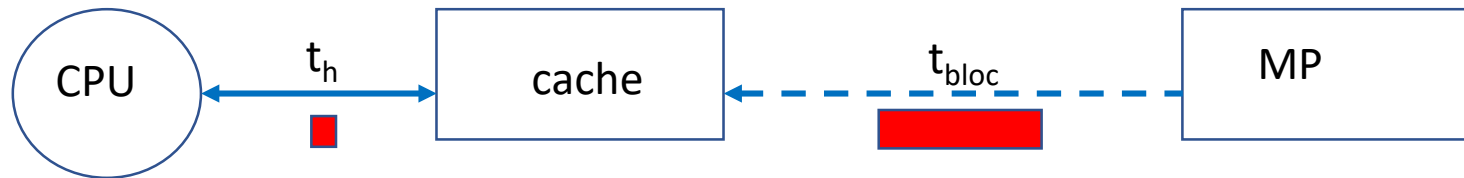


Temps de penalització t_p

- Escriptura **immediata amb assignació**

- Fallada de lectura o escriptura

$t_p = t_{\text{bloc}} + t_h$: Copiar bloc de MP a MC i transferir la dada a la CPU

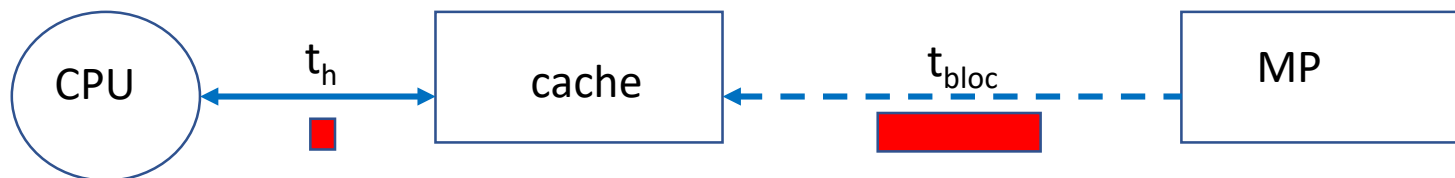


Temps de penalització t_p

- Escriptura **retardada** (**amb assignació**)

- Igual per a fallades de lectura o d'escriptura
- Si el bloc reemplaçat està **no-modificat** ($D=0$)

$t_p = t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU

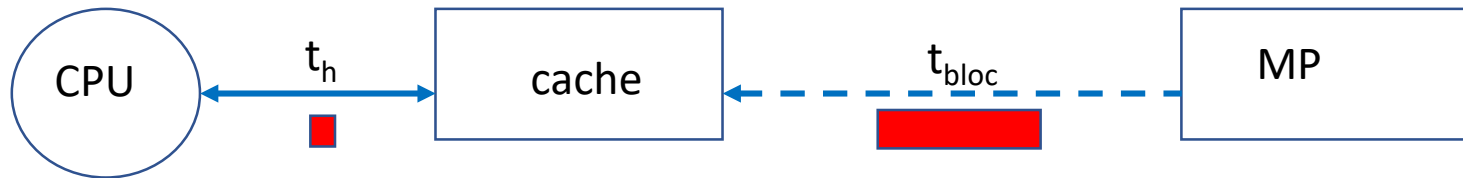


Temps de penalització t_p

- Escriptura **retardada** (amb assignació)

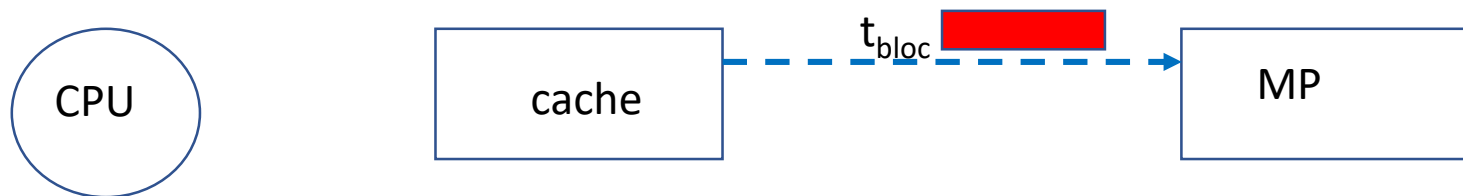
- Igual per a fallades de lectura o d'escriptura
- Si el bloc reemplaçat està **no-modificat** ($D=0$)

$t_p = t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU



- Si el bloc reemplaçat està **modificat** ($D=1$)

t_{bloc} : Escriure el bloc modificat en MP

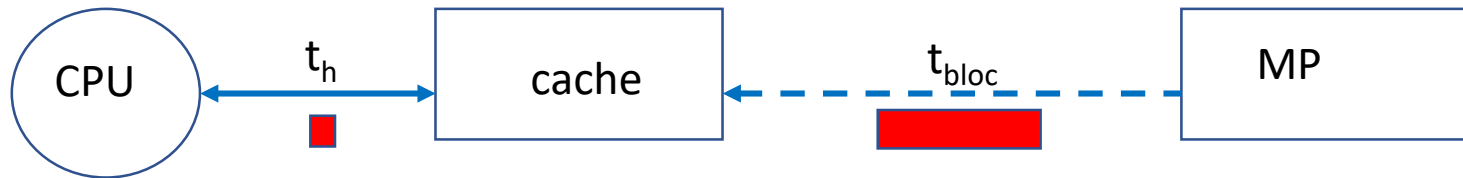


Temps de penalització t_p

- Escriptura **retardada** (amb assignació)

- Igual per a fallades de lectura o d'escriptura
- Si el bloc reemplaçat està **no-modificat** ($D=0$)

$t_p = t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU

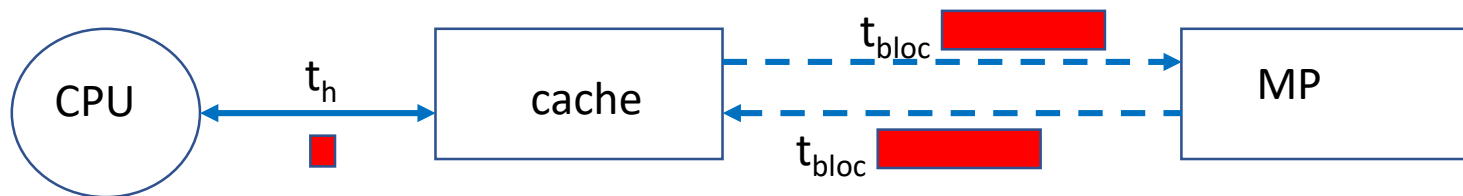


- Si el bloc reemplaçat està **modificat** ($D=1$)

t_{bloc} : Escriure el bloc modificat en MP

$t_{\text{bloc}} + t_h$: Copiar el bloc de MP a MC i servir la dada a la CPU

$$t_p = 2 \times t_{\text{bloc}} + t_h$$



Model de temps simplificat: t_p

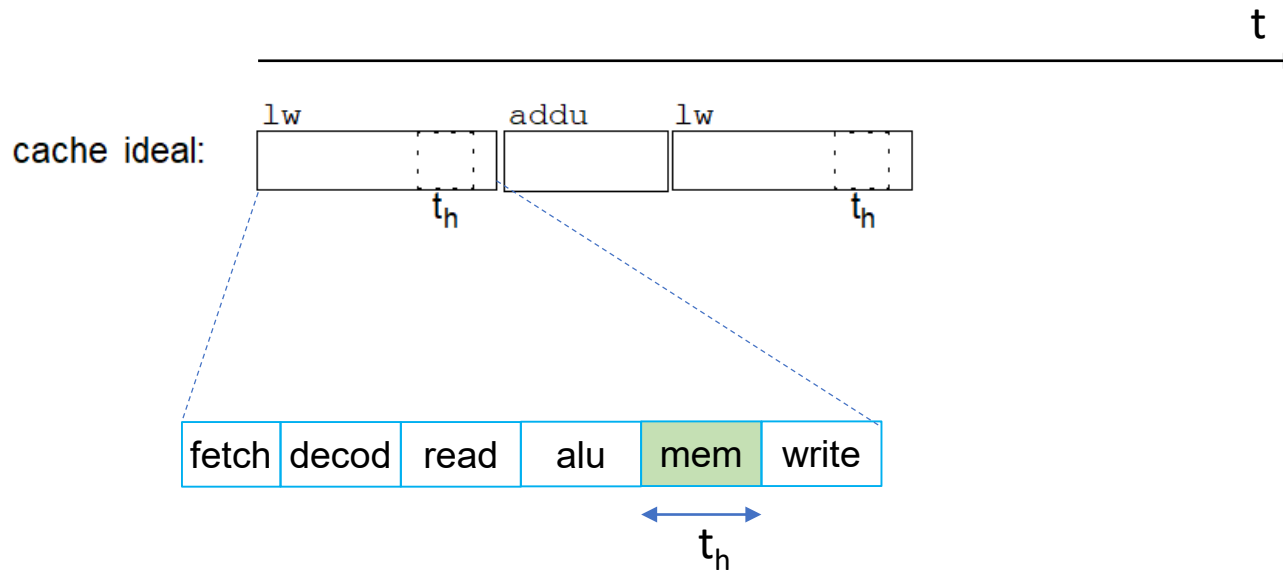
En resum:

t_p	Immediata sense assignació	Immediata amb assignació
Lectura	$t_{\text{bloc}} + t_h$	$t_{\text{bloc}} + t_h$
Escriptura	0	$t_{\text{bloc}} + t_h$

t_p	Retardada amb assignació
Bloc NO modificat	$t_{\text{bloc}} + t_h$
Bloc modificat	$2 \times t_{\text{bloc}} + t_h$

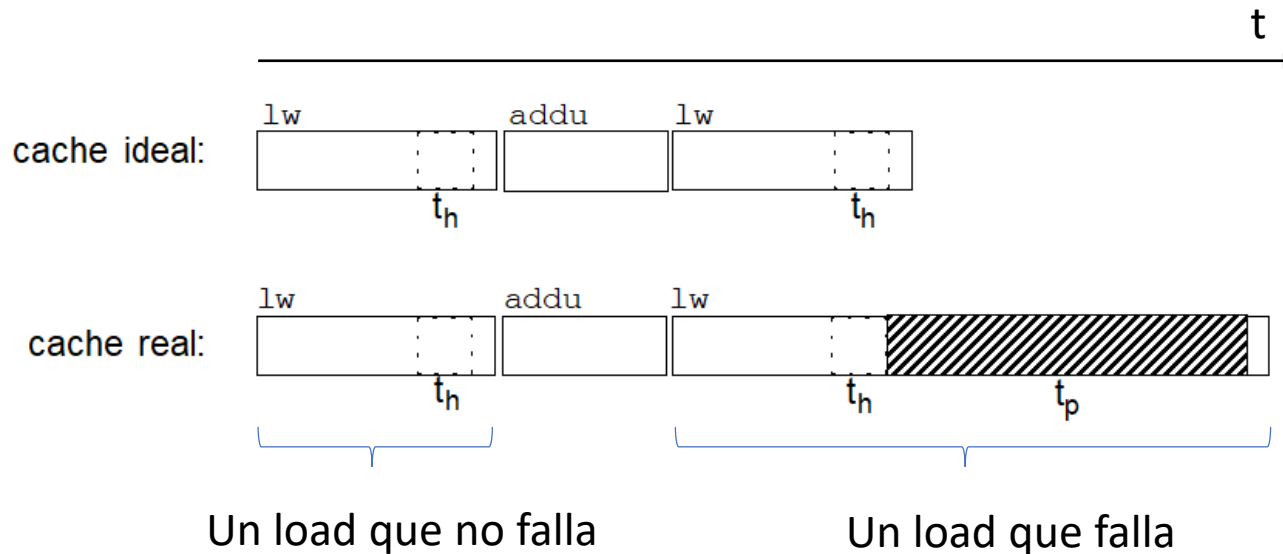
Impacte de les fallades en el rendiment

- Exemple: lw - addu - lw
(ignorem el fetch d'instruccions)
 - Cas 1: cache ideal, sense fallades



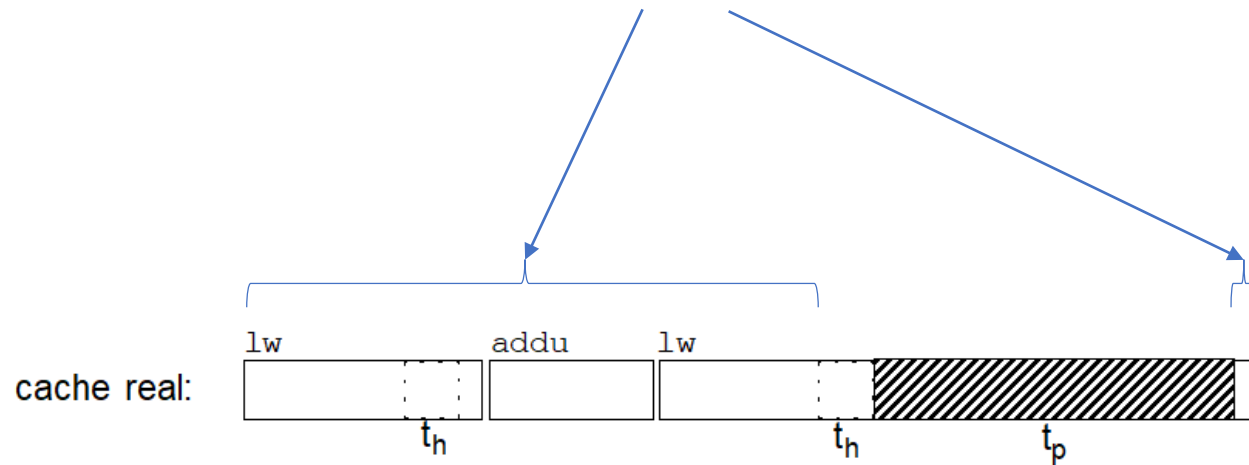
Impacte de les fallades en el rendiment

- Exemple: lw - addu - lw
(ignorem el fetch d'instruccions)
 - Cas 1: cache ideal, sense fallades
 - Cas 2: cache real



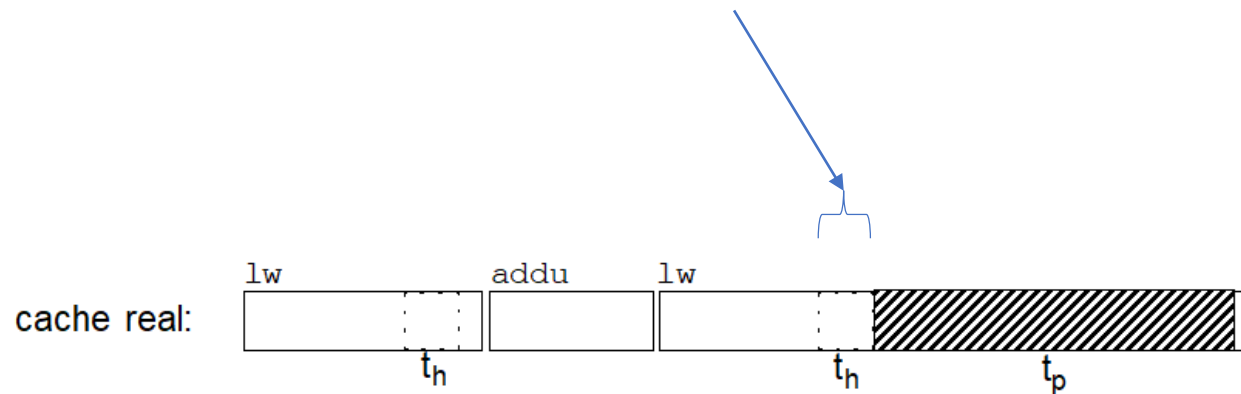
Impacte de les fallades en el rendiment

- Components del temps d'execució (de CPU)
 - Cicles executant-se el programa (**part blanca**)
 - Inclou tot el processament: decodificació, registres, ALU, etc.



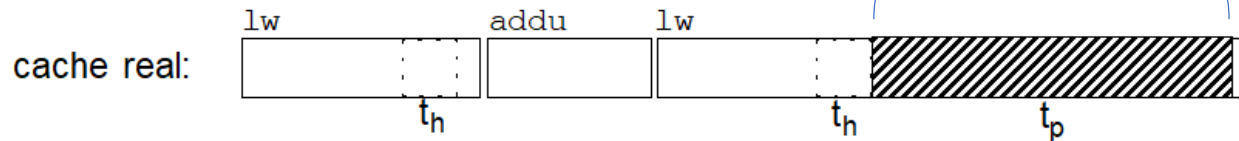
Impacte de les fallades en el rendiment

- Components del temps d'execució (de CPU)
 - Cicles executant-se el programa (**part blanca**)
 - Inclou tot el processament: decodificació, registres, ALU, etc.
 - Inclou el temps d'encert t_h



Impacte de les fallades en el rendiment

- Components del temps d'execució (de CPU)
 - Cicles executant-se el programa (part blanca)
 - Inclou tot el processament: decodificació, registres, ALU, etc.
 - Inclou el temps d'encert t_h
 - Cicles de penalització esperant la memòria (part ratllada)
 - Consisteix només en les penalitzacions t_p



Impacte de les fallades en el rendiment

- Components del temps d'execució (de CPU)
 - Cicles executant-se el programa (part blanca)
 - Inclou tot el processament: decodificació, registres, ALU, etc.
 - Inclou el temps d'encert t_h
 - Cicles de penalització esperant la memòria (part ratllada)
 - Consisteix només en les penalitzacions t_p
- Definim alguns termes:
 - n_{cicles} = els que tarda l'execució en el cas real
 - $n_{\text{cicles_ideal}}$ = els que tarda l'execució del cas ideal
 - $n_{\text{cicles_penal}}$ = cicles de penalització per fallades
 - n_{ins} = instruccions executades
 - nr = referències a memòria (per instrucció)
= $\text{num_referencies} / n_{\text{ins}}$

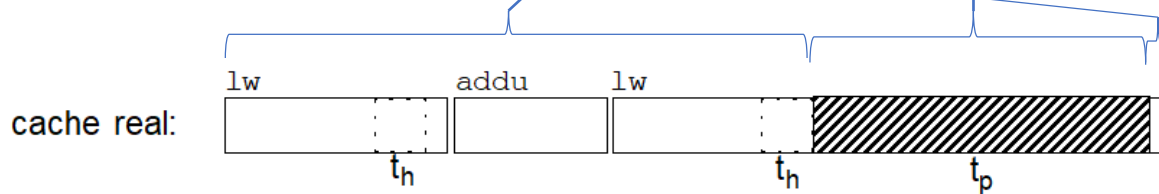
Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$

- Cicles d'execució

- $n_{\text{cicles}} = n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}$



Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$

Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

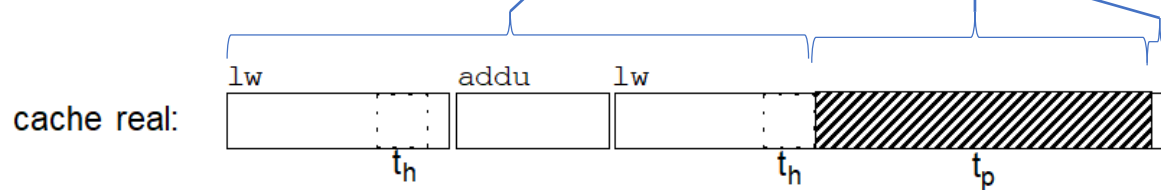
Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$



Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

- Cicles de penalització per instrucció

- $\text{CPI}_{\text{penal}} = n_{\text{cicles_penal}} / n_{\text{ins}}$

Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$


- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

- Cicles de penalització per instrucció

- $\text{CPI}_{\text{penal}} = n_{\text{cicles_penal}} / n_{\text{ins}}$
 $= t_p \times \text{num_fallades} / n_{\text{ins}}$

t_p expressat
en cicles



Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

- Cicles de penalització per instrucció

- $\text{CPI}_{\text{penal}} = n_{\text{cicles_penal}} / n_{\text{ins}}$
 $= t_p \times \text{num_fallades} / n_{\text{ins}}$
 $= t_p \times m \times \text{num_referencies} / n_{\text{ins}}$

t_p expressat
en cicles

Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

- Cicles de penalització per instrucció

- $\text{CPI}_{\text{penal}} = n_{\text{cicles_penal}} / n_{\text{ins}}$
 $= t_p \times \text{num_fallades} / n_{\text{ins}}$
 $= t_p \times m \times \text{num_referencies} / n_{\text{ins}}$
 $= t_p \times m \times \text{nr}$

t_p expressat
en cicles

Impacte de les fallades en el rendiment

- Temps d'execució (recordem el tema 1)

- $t_{\text{exe}} = n_{\text{cicles}} \times t_c$
 $= \text{CPI} \times n_{\text{ins}} \times t_c$

- Cicles d'execució per instrucció

- $\text{CPI} = (n_{\text{cicles_ideal}} + n_{\text{cicles_penal}}) / n_{\text{ins}}$
 $= \text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}$

- Cicles de penalització per instrucció

- $\text{CPI}_{\text{penal}} = n_{\text{cicles_penal}} / n_{\text{ins}}$
 $= t_p \times \text{num_fallades} / n_{\text{ins}}$
 $= t_p \times m \times \text{num_referencies} / n_{\text{ins}}$
 $= t_p \times m \times nr$

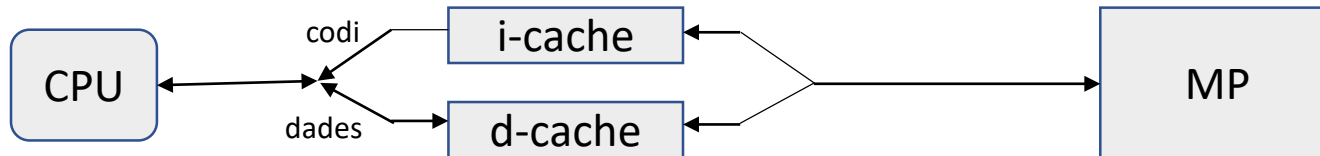
t_p expressat
en cicles

- Resumint, t_{exe} depèn de t_p i m

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

Exemple

- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$



Exemple

- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$
 - $t_p = 100$ cicles
 - $CPI_{ideal} = 2,5$
 - 36% de les instruccions accedeixen d-cache (loads/stores)
 - 100% de les instruccions fan "fetch" a la i-cache

Exemple

- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$
 - $t_p = 100$ cicles
 - $CPI_{ideal} = 2,5$
 - 36% de les instruccions accedeixen d-cache (loads/stores)
 - 100% de les instruccions fan "fetch" a la i-cache
- Calcular els cicles de penalització (per instrucció)
 - CPI_{penal} (i-cache) =
 - CPI_{penal} (d-cache) =

Exemple

- Donat el sistema

- Cache d'instruccions (i-cache): $m_i = 2\%$
- Cache de dades (d-cache): $m_d = 4\%$
- $t_p = 100$ cicles
- $CPI_{ideal} = 2,5$
- 36% de les instruccions accedeixen d-cache (loads/stores)
- 100% de les instruccions fan "fetch" a la i-cache

- Calcular els cicles de penalització (per instrucció)

- $CPI_{penal} \text{ (i-cache)} = (100 \times 0,02 \times 1,00) = 2$
- $CPI_{penal} \text{ (d-cache)} =$

$$CPI_{penal} = t_p \times m \times nr$$

Exemple

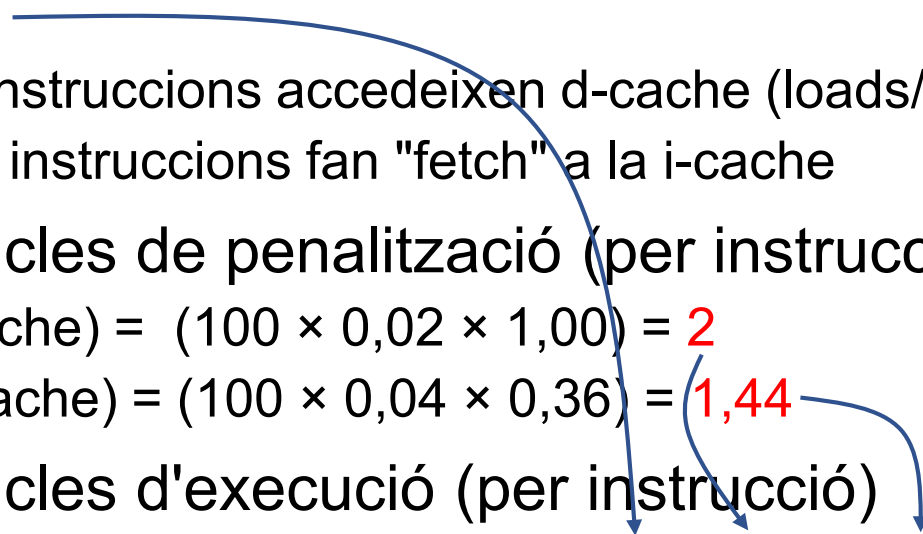
- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$
 - $t_p = 100$ cicles
 - $CPI_{ideal} = 2,5$
 - 36% de les instruccions accedeixen d-cache (loads/stores)
 - 100% de les instruccions fan "fetch" a la i-cache
 - Calcular els cicles de penalització (per instrucció)
 - $CPI_{penal} (i-cache) = (100 \times 0,02 \times 1,00) = 2$
 - $CPI_{penal} (d-cache) = (100 \times 0,04 \times 0,36) = 1,44$
-

$$CPI_{penal} = t_p \times m \times nr$$

Exemple

- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$
 - $t_p = 100$ cicles
 - $CPI_{ideal} = 2,5$
 - 36% de les instruccions accedeixen d-cache (loads/stores)
 - 100% de les instruccions fan "fetch" a la i-cache
- Calcular els cicles de penalització (per instrucció)
 - $CPI_{penal} \text{ (i-cache)} = (100 \times 0,02 \times 1,00) = 2$
 - $CPI_{penal} \text{ (d-cache)} = (100 \times 0,04 \times 0,36) = 1,44$
- Calcular els cicles d'execució (per instrucció)
 - $CPI =$

Exemple

- Donat el sistema
 - Cache d'instruccions (i-cache): $m_i = 2\%$
 - Cache de dades (d-cache): $m_d = 4\%$
 - $t_p = 100$ cicles
 - $CPI_{ideal} = 2,5$
 - 36% de les instruccions accedeixen d-cache (loads/stores)
 - 100% de les instruccions fan "fetch" a la i-cache
 - Calcular els cicles de penalització (per instrucció)
 - $CPI_{penal} (i-cache) = (100 \times 0,02 \times 1,00) = 2$
 - $CPI_{penal} (d-cache) = (100 \times 0,04 \times 0,36) = 1,44$
 - Calcular els cicles d'execució (per instrucció)
 - $CPI = CPI_{ideal} + CPI_{penal_i} + CPI_{penal_d} = 2,5 + 2 + 1,44 = 5,94$
- 

Temps d'accés mitjà

- Hem vist que t_{exe} depèn de m i t_p
 - Ens interessa reduir la taxa de fallades m augmentant la capacitat?

Temps d'accés mitjà

- Hem vist que t_{exe} depèn de m i t_p
 - Ens interessa reduir la taxa de fallades m augmentant la capacitat?
 - **Sí i no:** Reduiríem m però augmentaríem el temps d'encert t_h !
- Però com influeix t_h en t_{exe} ?

Temps d'accés mitjà

- Hem vist que t_{exe} depèn de m i t_p
 - Ens interessa reduir la taxa de fallades m augmentant la capacitat?
 - **Sí i no:** Reduiríem m però augmentaríem el temps d'encert t_h !
- Però com influeix t_h en t_{exe} ?
 - Està inclòs dins n_{cicles_ideal} → no sabem com influeix
 - Ens cal una mètrica que tingui en compte m , t_p i també t_h

Temps d'accés mitjà

- Hem vist que t_{exe} depèn de m i t_p
 - Ens interessa reduir la taxa de fallades m augmentant la capacitat?
 - **Sí i no:** Reduiríem m però augmentaríem el temps d'encert t_h !
- Però com influeix t_h en t_{exe} ?
 - Està inclòs dins $n_{\text{cicles_ideal}}$ → no sabem com influeix
 - Ens cal una mètrica que tingui en compte m , t_p i també t_h
- Temps d'accés mitjà a memòria t_{am} (=AMAT en anglès)
$$t_{\text{am}} = h \times t_{\text{encert}} + m \times t_{\text{fallada}}$$

Temps d'accés mitjà

- Hem vist que t_{exe} depèn de m i t_p
 - Ens interessa reduir la taxa de fallades m augmentant la capacitat?
 - **Sí i no:** Reduiríem m però augmentaríem el temps d'encert t_h !
- Però com influeix t_h en t_{exe} ?
 - Està inclòs dins $n_{\text{cicles_ideal}}$ → no sabem com influeix
 - Ens cal una mètrica que tingui en compte m , t_p i també t_h
- Temps d'accés mitjà a memòria t_{am} (=AMAT en anglès)

$$t_{\text{am}} = h \times t_{\text{encert}} + m \times t_{\text{fallada}}$$

$$t_{\text{am}} = (1 - m) \times t_h + m \times (t_h + t_p)$$

$$t_{\text{am}} = t_h + m \times t_p$$

Exemple

- Suposem un sistema amb
 - $t_c = 0,75 \text{ ns}$
 - $t_h = 1 \text{ cicle}$
 - $t_p = 20 \text{ cicles}$
 - $m = 5 \%$
- Calcular el temps d'accés mitjà: en cicles, i en ns

Exemple

- Suposem un sistema amb
 - $t_c = 0,75$ ns
 - $t_h = 1$ cicle
 - $t_p = 20$ cicles
 - $m = 5$ %
- Calcular el temps d'accés mitjà: en cicles, i en ns

$$t_{am} = t_h + m \times t_p$$

$$t_{am} = 1 + 0,05 \times 20 = 2 \text{ cicles}$$

Exemple

- Suposem un sistema amb
 - $t_c = 0,75$ ns
 - $t_h = 1$ cicle
 - $t_p = 20$ cicles
 - $m = 5\%$
- Calcular el temps d'accés mitjà: en cicles, i en ns

$$t_{am} = t_h + m \times t_p$$

$$t_{am} = 1 + 0,05 \times 20 = 2 \text{ cicles}$$

$$t_{am} = 2 \times 0,75 = 1,5 \text{ ns}$$

Observacions sobre el rendiment

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}) \times n_{\text{ins}} \times t_c$$

- Observacions

- Les millores en microarquitectura han reduït $\text{CPI}_{\text{ideal}}$ 😊

Observacions sobre el rendiment

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}) \times n_{\text{ins}} \times t_c$$

- Observacions

- Les millores en microarquitectura han reduït $\text{CPI}_{\text{ideal}}$ 😊
- ⇒ Ha augmentat la importància relativa de $\text{CPI}_{\text{penal}}$ 😞
- ⇒ És imprescindible millorar el comportament de la cache

Observacions sobre el rendiment

$$\begin{aligned} t_{\text{exe}} &= (\text{CPI}_{\text{ideal}} + \text{CPI}_{\text{penal}}) \times n_{\text{ins}} \times t_c \\ &= (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c \end{aligned}$$

- Observacions

- Les millores en microarquitectura han reduït $\text{CPI}_{\text{ideal}}$ 😊
- ⇒ Ha augmentat la importància relativa de $\text{CPI}_{\text{penal}}$ 😞
- ⇒ És imprescindible millorar el comportament de la cache
 - Com millorar m ? → associativitat
 - Com millorar t_p ? → caches multinivell

Memòria Cache

- Introducció
- Aspectes de disseny
- Impacte en el rendiment
- **Millores de rendiment**
 - Associativitat
 - Caches multinivell
 - Tipologia de les fallades

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

Memòria cache

Índex de línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

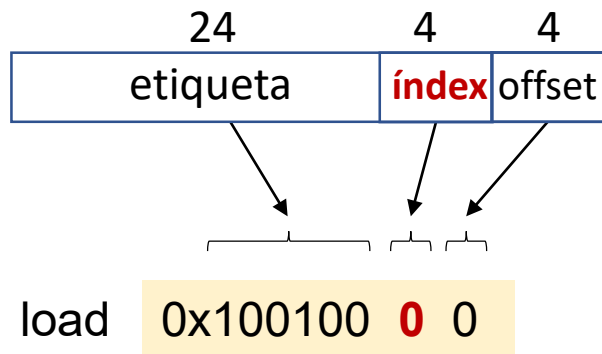


Memòria cache

Índex de línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

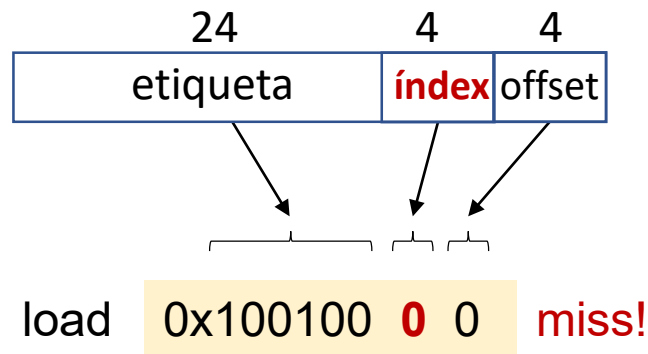


Memòria cache

Índex de línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

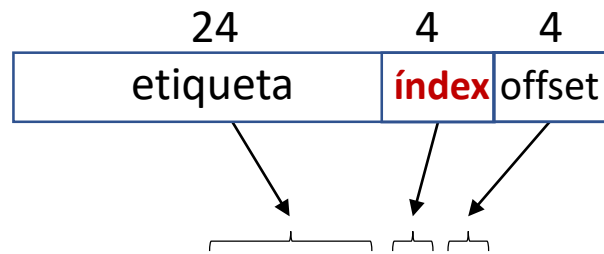


V=0!

Memòria cache			
Índex de línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load 0x100100 **0** 0 **miss**

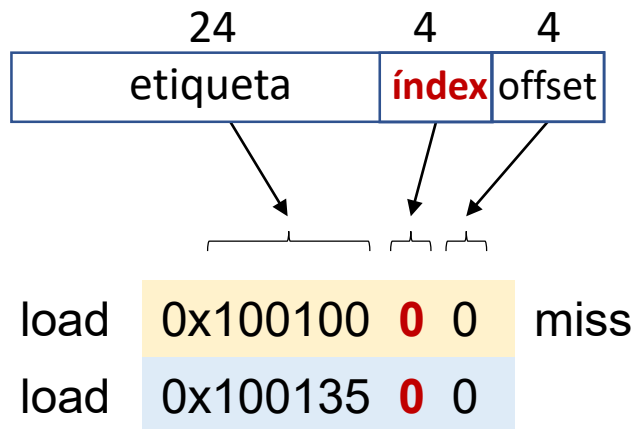
copiar bloc!
V=1

Memòria cache

Índex de línia	V	Etiqueta	Dades
0	1	100100	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

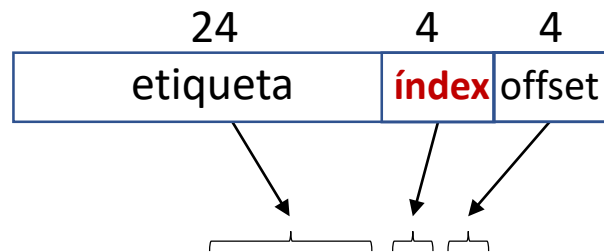


Memòria cache

Índex de línia	V	Etiqueta	Dades
0	1	100100	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load 0x100100 **0** 0 miss

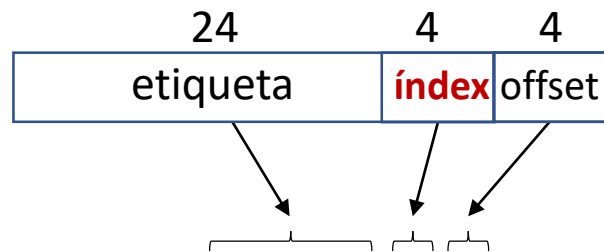
load 0x100135 **0** 0 **miss!**

Memòria cache

Índex de línia	V	Etiqueta	Dades
0	1	100100	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load 0x100100 **0** 0 miss

load 0x100135 **0** 0 **miss**

Reemplaçar
Bloc!

Índex
de línia

0

1

2

3

4

5

...

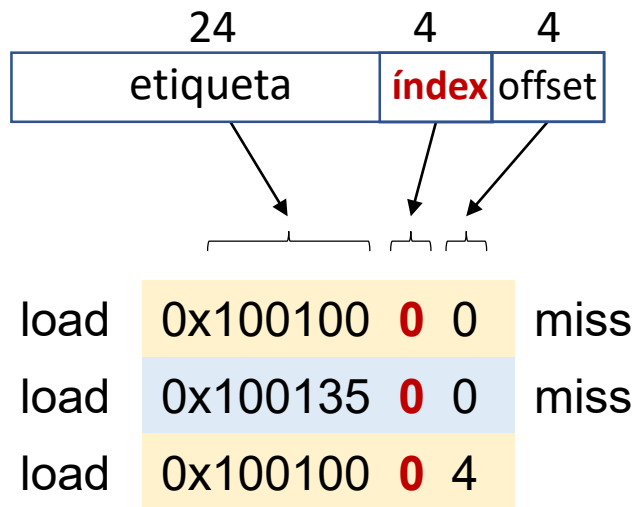
15

Memòria cache

	V	Etiqueta	Dades
0	1	100135	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies

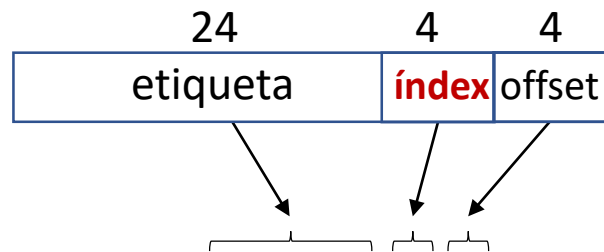


Memòria cache

Índex de línia	V	Etiqueta	Dades
0	1	100135	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load	0x100100	0	0	miss
load	0x100135	0	0	miss
load	0x100100	0	4	miss!

Índex
de línia

0

1

2

3

4

5

...

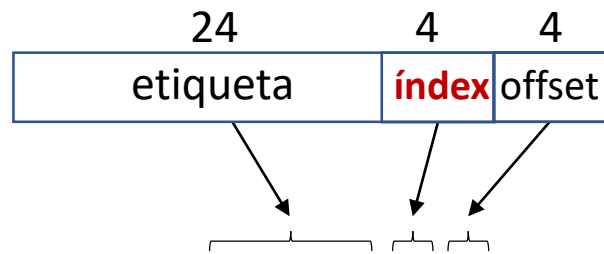
15

Memòria cache

V	Etiqueta	Dades
1	100135	
0		
0		
0		
0		
0		
...
0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load 0x100100 **0** 0 miss

load 0x100135 **0** 0 miss

load 0x100100 **0** 4 **miss!**

Reemplaçar
Bloc!

Índex
de línia

0

1

2

3

4

5

...

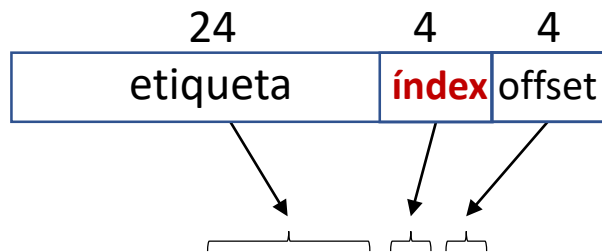
15

Memòria cache

V	Etiqueta	Dades
1	100100	
0		
0		
0		
0		
0		
...
0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load	0x100100	0	0	miss
load	0x100135	0	0	miss
load	0x100100	0	4	miss
load	0x100135	0	4	

Reemplaçar
Bloc!

Índex
de línia

0

1

2

3

4

5

...

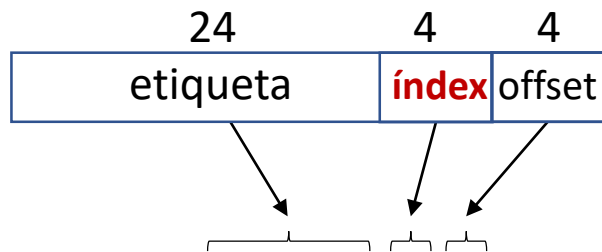
15

Memòria cache

V	Etiqueta	Dades
1	100100	
0		
0		
0		
0		
0		
...
0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



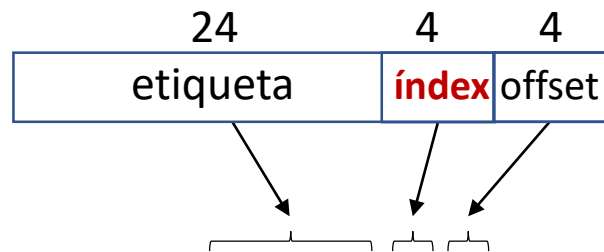
load	0x100100	0	0	miss
load	0x100135	0	0	miss
load	0x100100	0	4	miss
load	0x100135	0	4	miss!

Memòria cache

Índex de línia	V	Etiqueta	Dades
0	1	100100	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

Cache de correspondència directa

- Problema de la correspondència directa: conflictes
- Suposem
 - Adreces de 32 bits
 - Blocs de 16 bytes
 - Cache de 16 línies



load	0x100100	0	0	miss
load	0x100135	0	0	miss
load	0x100100	0	4	miss
load	0x100135	0	4	miss

Reemplaçar
Bloc!

Índex
de línia

0

1

2

3

4

5

...

15

Memòria cache

V	Etiqueta	Dades
1	100135	
0		
0		
0		
0		
0		
...
0		

Cache completament associativa

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

- Com reduir la taxa de fallades (m) per conflictes?

Cache completament associativa

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$


- Com reduir la taxa de fallades (**m**) per conflictes?
- Cache completament associativa
 - **Un bloc es pot guardar en qualsevol línia de MC**

Cache completament associativa

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$


- Com reduir la taxa de fallades (**m**) per conflictes?
- Cache completament associativa
 - **Un bloc es pot guardar en qualsevol línia de MC**
 - Més flexible \Rightarrow menor taxa de fallades

Cache completament associativa

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$


- Com reduir la taxa de fallades (**m**) per conflictes?
- Cache completament associativa
 - **Un bloc es pot guardar en qualsevol línia de MC**
 - Més flexible \Rightarrow menor taxa de fallades
 - Cal comprovar totes les línies alhora, amb un comparador per línia \Rightarrow major cost en hardware

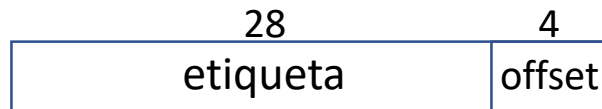
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

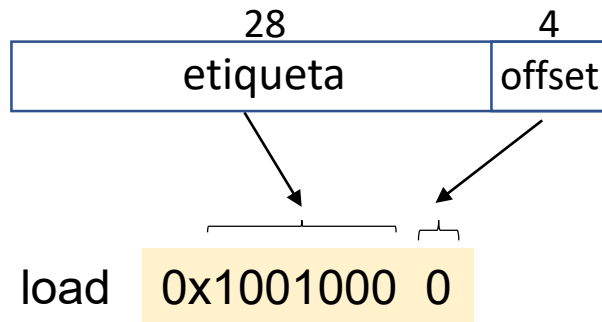
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

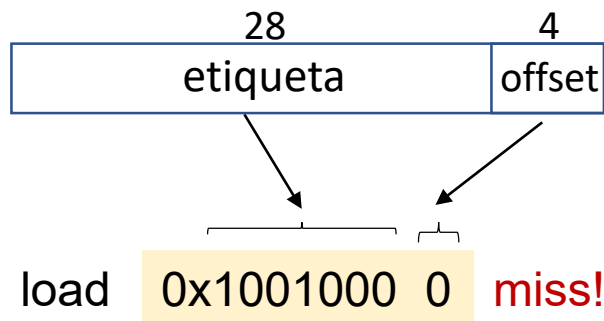
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



cache
Buida!

Memòria cache

línia	V	Etiqueta	Dades
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

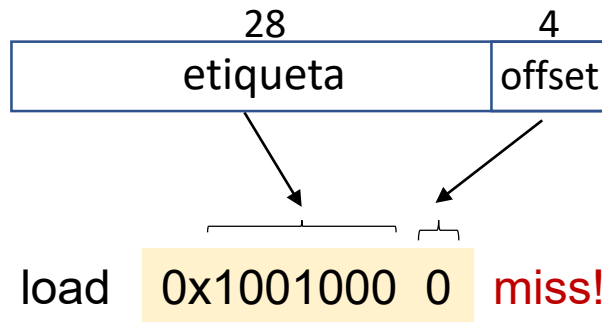
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



copiar
bloc

Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

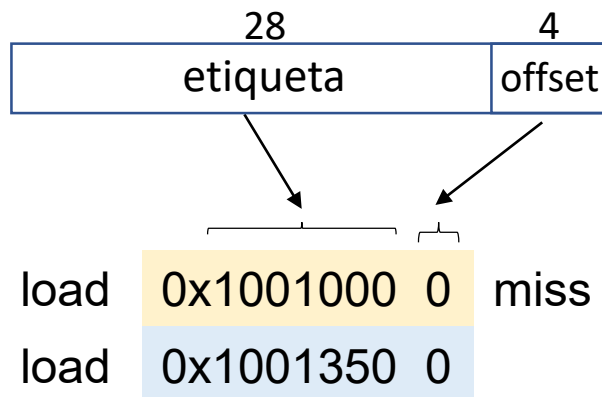
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

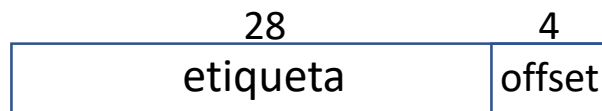
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



load 0x1001000 0 miss
load 0x1001350 0 **miss!**

no hi és!

Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	0		
2	0		
3	0		
4	0		
5	0		
...
15	0		

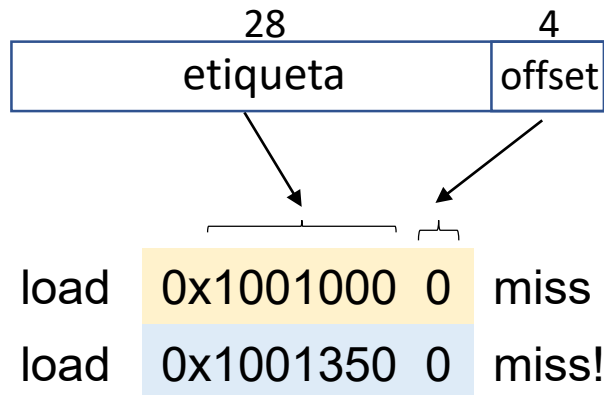
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



copiar bloc

Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

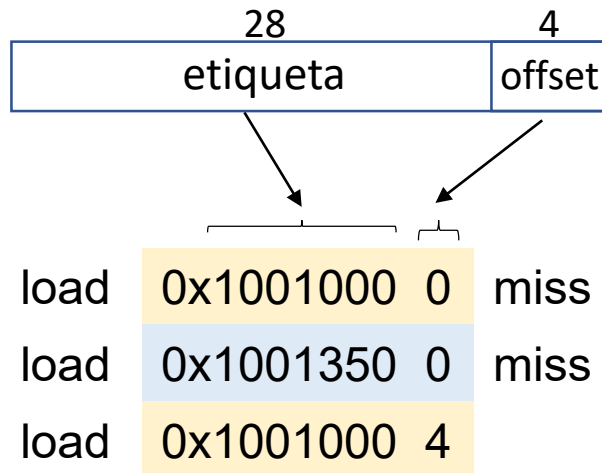
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

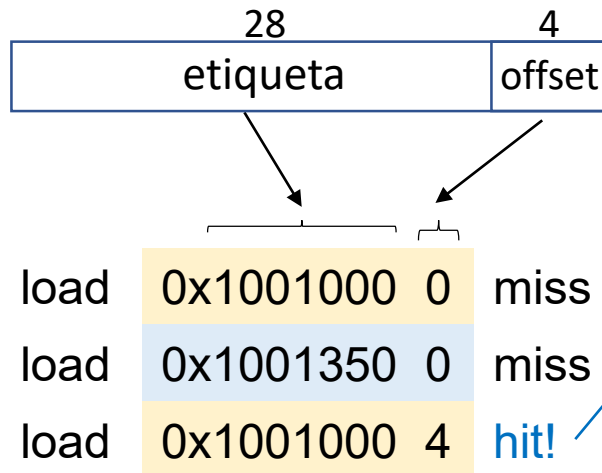
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

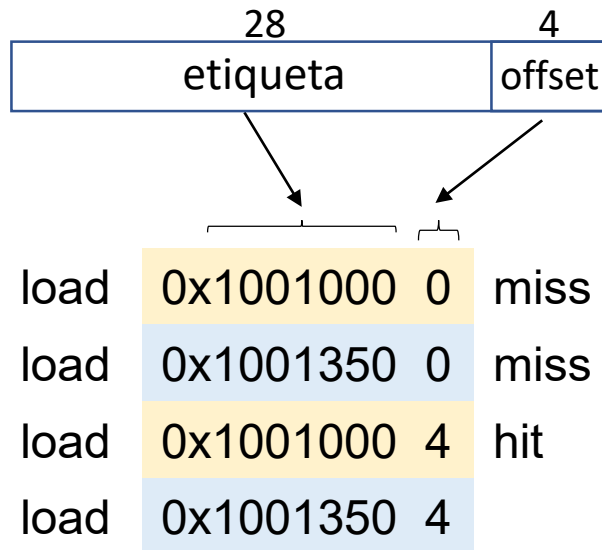
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

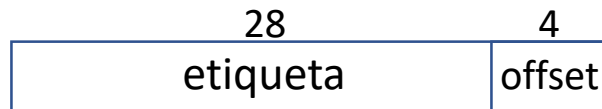
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



load	0x1001000	0	miss
load	0x1001350	0	miss
load	0x1001000	4	hit
load	0x1001350	4	hit!

Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

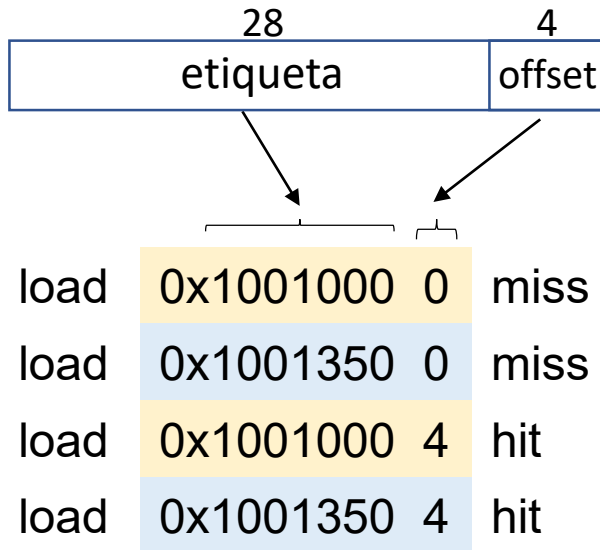
Cache completament associativa

- Memòria cache completament associativa

Un bloc pot anar a qualsevol línia de MC

- Suposem el mateix exemple

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 16 línies



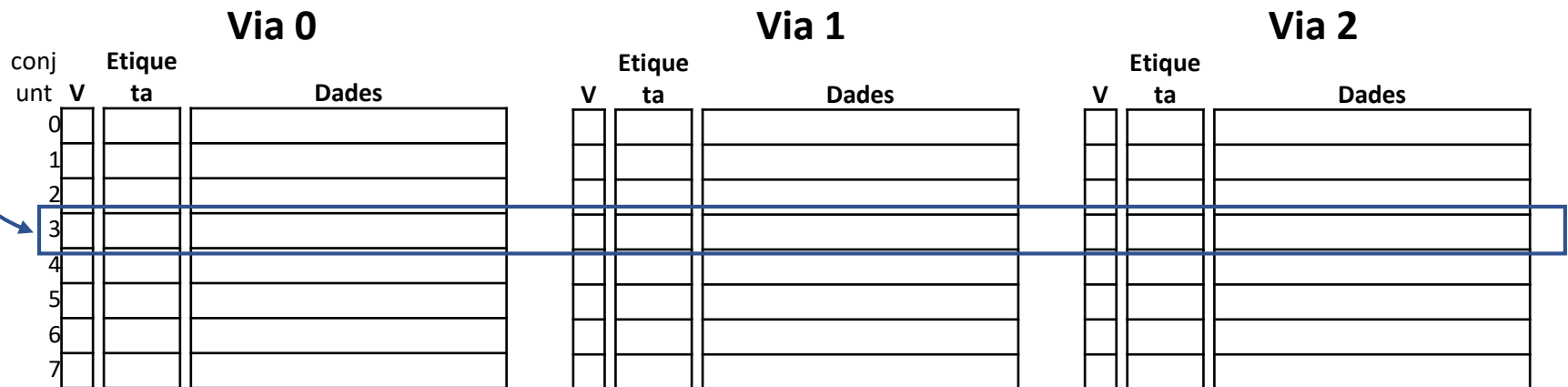
Memòria cache

línia	V	Etiqueta	Dades
0	1	1001000	
1	1	1001350	
2	0		
3	0		
4	0		
5	0		
...
15	0		

- Millor taxa d'encerts
- Però major cost en hardware

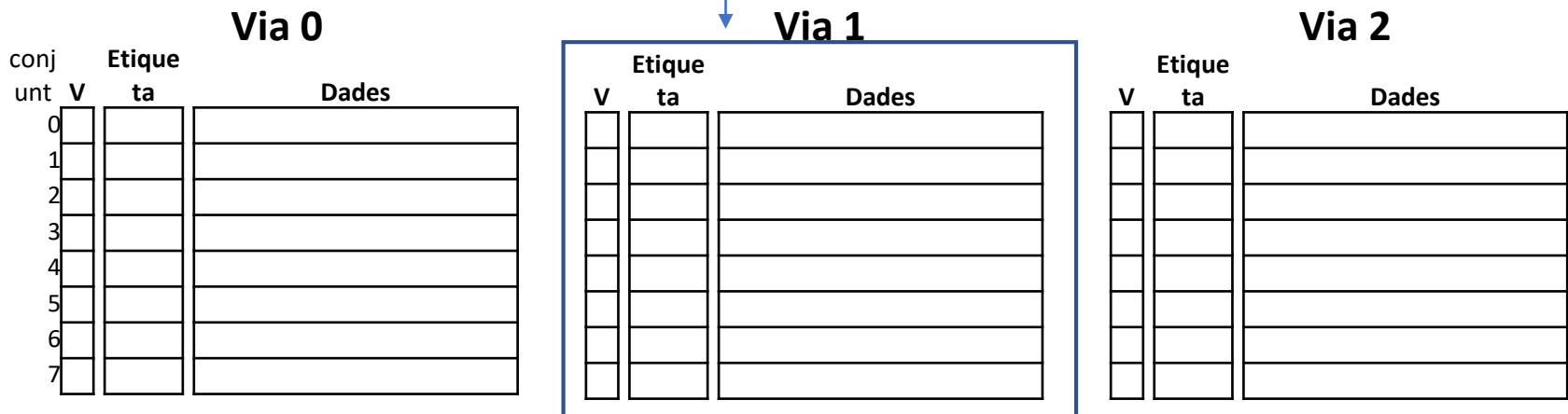
Cache associativa per conjunts

- La cache és com una taula amb n columnes on
 - Cada fila és un **conjunt**



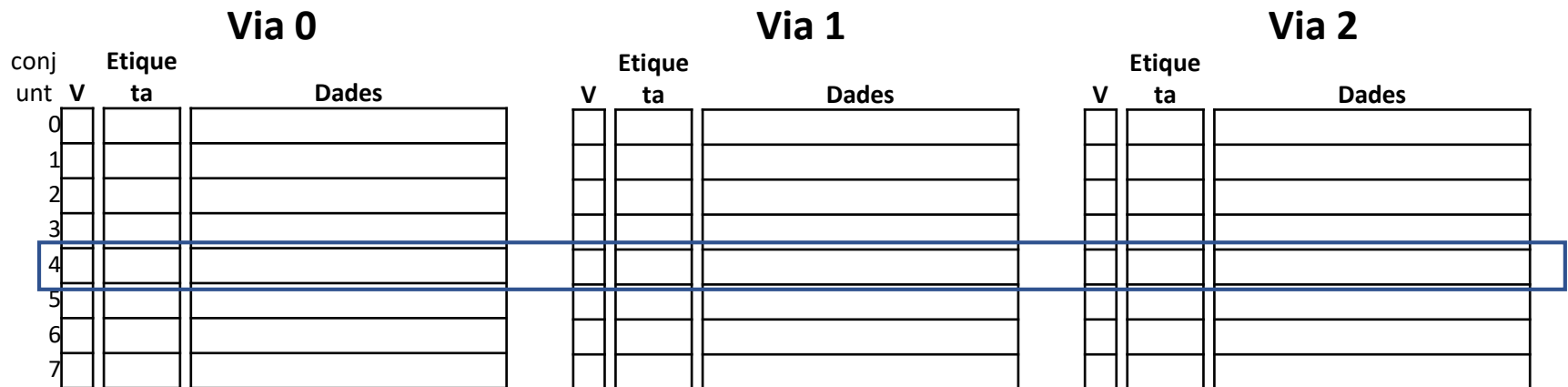
Cache associativa per conjunts

- La cache és com una taula amb n columnes on
 - Cada fila és un *conjunt*
 - Cada columna és una **via**



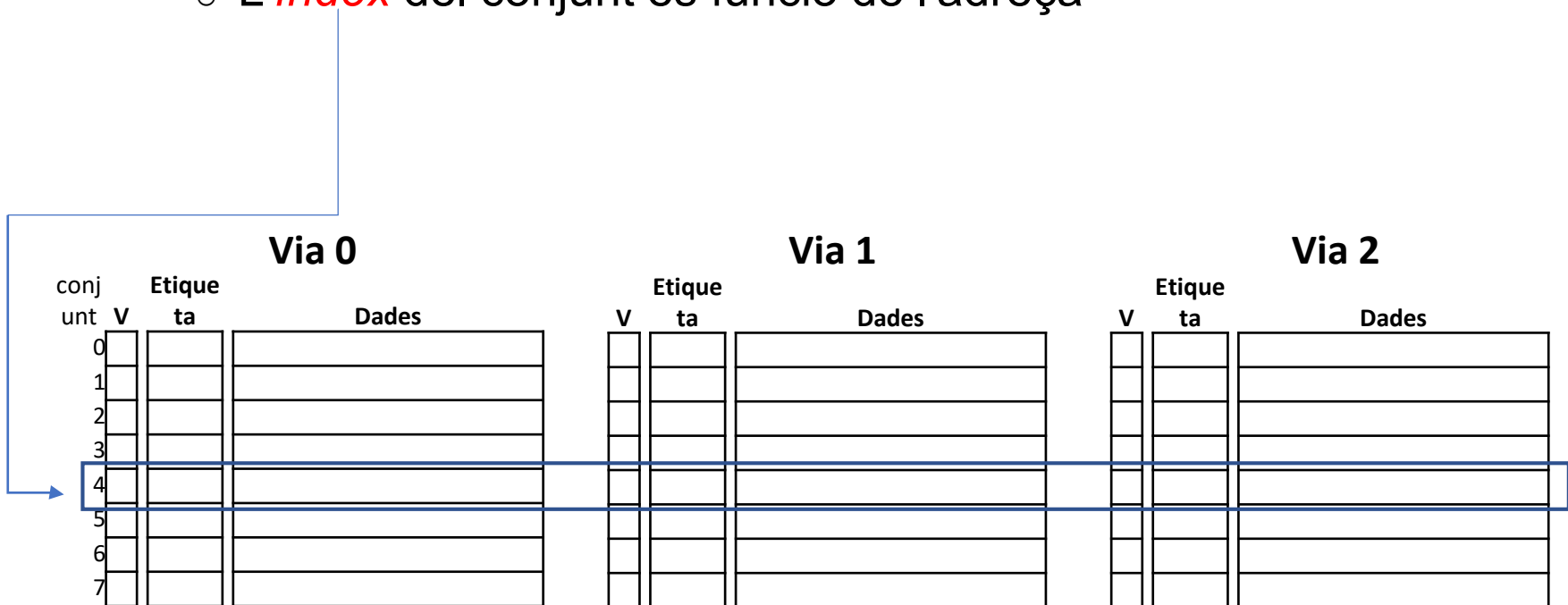
Cache associativa per conjunts

- La cache és com una taula amb n columnes on
 - Cada fila és un *conjunt*
 - Cada columna és una *via*
- Cada *conjunt* pot guardar n blocs (un a cada via)
 - n és fix (i no li cal ser potència de 2)



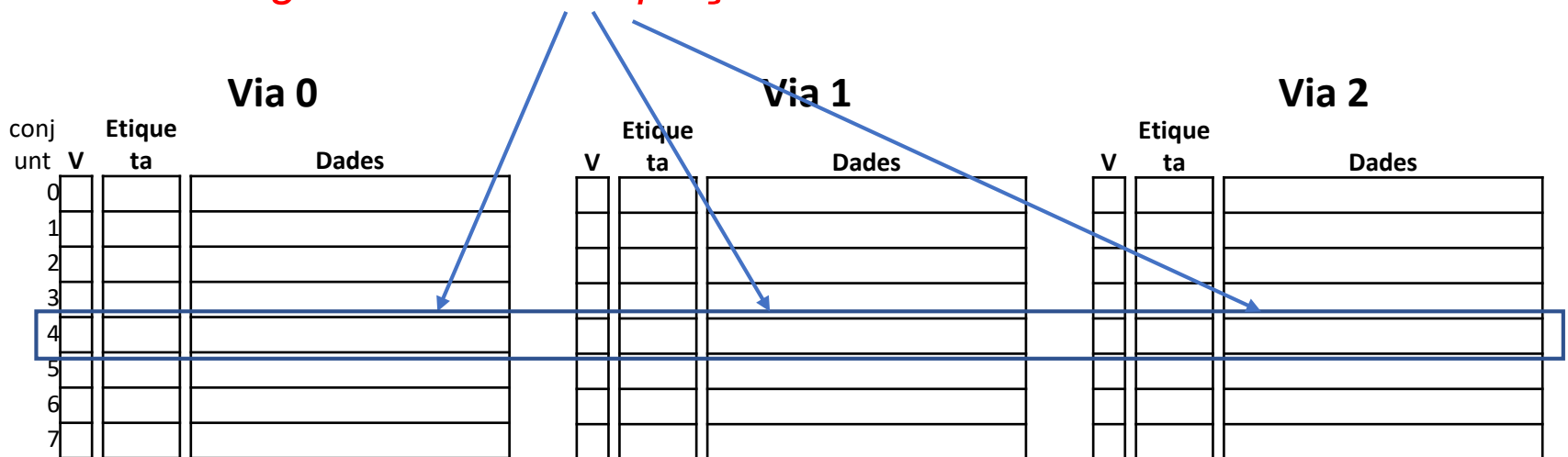
Cache associativa per conjunts

- La cache és com una taula amb n columnes on
 - Cada fila és un *conjunt*
 - Cada columna és una *via*
- Cada *conjunt* pot guardar n blocs (un a cada via)
 - n és fix (i no li cal ser potència de 2)
- Cada bloc de memòria **mapeja a un únic conjunt**
 - L'*índex* del conjunt és funció de l'adreça



Cache associativa per conjunts

- La cache és com una taula amb n columnes on
 - Cada fila és un *conjunt*
 - Cada columna és una *via*
- Cada *conjunt* pot guardar n blocs (un a cada via)
 - n és fix (i no li cal ser potència de 2)
- Cada bloc de memòria mapeja a un únic conjunt
 - L'*índex* del conjunt és funció de l'adreça
- Dins del conjunt podem guardar el bloc **a qualsevol via**
 - L'*algorisme de reemplaçament* selecciona la via



Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load 0x1001000 0

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	0			0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load

0x1001000 0

miss!

Conjunt
buit!

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	0			0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

Memòria Cache

Via 0

Via 1

Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Copia
el bloc

load

0x1001000 0

miss

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss

load 0x1001350 0

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss

load 0x1001350 0 miss!

Memòria Cache

Via 0			Via 1		
Conjunt V	Etiqueta	Dades	V	Etiqueta	Dades
0 1	200200		0		
1 0			0		
2 0			0		
3 0			0		
4 0			0		
5 0			0		
6 0			0		
7 0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss

load 0x1001350 0 **miss!**

Copia
el bloc

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	20026A	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	20026A	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit!

Memòria Cache

Via 0			Via 1		
Conjunt V	Etiqueta	Dades	V	Etiqueta	Dades
0 1	200200		1 1	20026A	
1 0			0 0		
2 0			0 0		
3 0			0 0		
4 0			0 0		
5 0			0 0		
6 0			0 0		
7 0			0 0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x1001350 4	

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	20026A	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x1001350 4	hit!

Memòria Cache

Via 0			Via 1		
Conjunt V	Etiqueta	Dades	V	Etiqueta	Dades
0 1	200200		1 1	20026A	
1 0			0 0		
2 0			0 0		
3 0			0 0		
4 0			0 0		
5 0			0 0		
6 0			0 0		
7 0			0 0		

Cache associativa per conjunts

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt

- **Millor taxa d'encerts**
- **Amb poc hardware adicional**

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x1001350 4	hit

Memòria Cache

Via 0			Via 1		
Conjunt V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200	1	20026A	
1	0		0		
2	0		0		
3	0		0		
4	0		0		
5	0		0		
6	0		0		
7	0		0		

Algorismes de reemplaçament

Quin bloc cal reemplaçar quan un bloc no hi cap?

- Correspondència directa
 - No hi ha alternativa

Algorismes de reemplaçament

Quin bloc cal reemplaçar quan un bloc no hi cap?

- Correspondència directa
 - No hi ha alternativa
- Totalment associativa
 - Cas particular d'associativa per conjunts, però amb un sol conjunt

Algorismes de reemplaçament

Quin bloc cal reemplaçar quan un bloc no hi cap?

- Correspondència directa
 - No hi ha alternativa
- Totalment associativa
 - Cas particular d'associativa per conjunts, però amb un sol conjunt
- Associativa per conjunts: triar dins el conjunt corresponent
 - Preferiblement un bloc invàlid ($V=0$)

Algorismes de reemplaçament

Quin bloc cal reemplaçar quan un bloc no hi cap?

- Correspondència directa
 - No hi ha alternativa
- Totalment associativa
 - Cas particular d'associativa per conjunts, però amb un sol conjunt
- Associativa per conjunts: triar dins el conjunt corresponent
 - Preferiblement un bloc invàlid ($V=0$)
 - Si no n'hi ha cap (el conjunt està ple)
 - **LRU**: (Least Recently Used): triar el bloc que fa més temps que no es referencia

Algorismes de reemplaçament

Quin bloc cal reemplaçar quan un bloc no hi cap?

- Correspondència directa
 - No hi ha alternativa
- Totalment associativa
 - Cas particular d'associativa per conjunts, però amb un sol conjunt
- Associativa per conjunts: triar dins el conjunt corresponent
 - Preferiblement un bloc invàlid ($V=0$)
 - Si no n'hi ha cap (el conjunt està ple)
 - **LRU**: (Least Recently Used): triar el bloc que fa més temps que no es referencia
 - **Random**: triar un bloc a l'atzar (menys efectiu, però molt simple)
 - LRU pot ser complex per a més de 4 vies. Random és simple

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem



load 0x1001000 0 miss!

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
Conjunt no ple!	0	0		0		
	1	0		0		
	2	0		0		
	3	0		0		
	4	0		0		
	5	0		0		
	6	0		0		
	7	0		0		

→
Conjunt
no ple!

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem



load

0x1001000 0

miss

Copia el
bloc

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss

load 0x1001350 0 **miss!**

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		0		
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

Conjunt
no ple

LRU

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss

load 0x1001350 0 miss

Copia el bloc

Establim
LRU:

LRU

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	20026A	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load 0x1001000 0 miss
load 0x1001350 0 miss
load 0x1001000 4 **hit!**

Canvia el
LRU:

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	20026A	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

↑
LRU

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x7FF0010 0	miss!

Memòria Cache

Via 0				Via 1			
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades	
0	1	200200		1	20026A		
1	0			0			
2	0			0			
3	0			0			
4	0			0			
5	0			0			
6	0			0			
7	0			0			

Conjunt ple!

LRU

LRU

- Cache associativa per conjunts de 2 vies

- Suposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x7FF0010 0	miss

Reempla
cem el
bloc LRU

Memòria Cache

Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	FFE002	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

↑
LRU

LRU

- Cache associativa per conjunts de 2 vies

- Supposem

- Adreces de 32 bits
- Blocs de 16 bytes
- Cache de 8 conjunts
- 2 vies per conjunt

- Cada bloc es mapeja en un únic conjunt
- Es pot guardar en qualsevol via del conjunt
- LRU: reemplaçem el que fa més que no usem

25	3	4
etiqueta	índex	offset

load	0x1001000 0	miss
load	0x1001350 0	miss
load	0x1001000 4	hit
load	0x7FF0010 0	miss

Memòria Cache

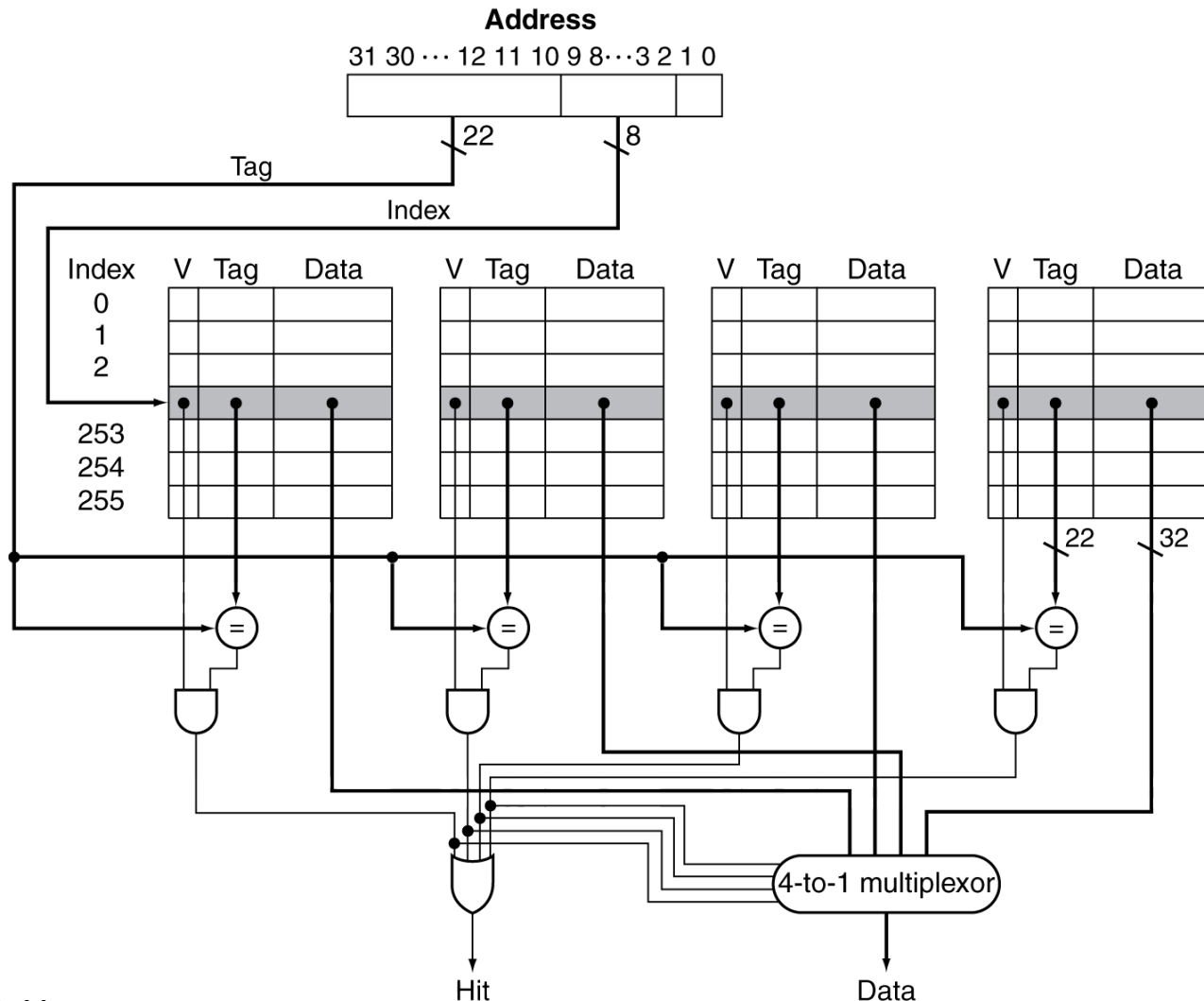
Via 0				Via 1		
Conjunt	V	Etiqueta	Dades	V	Etiqueta	Dades
0	1	200200		1	FFE002	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

I canviem
el LRU:

LRU

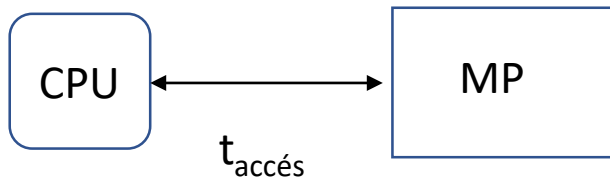
Cache associativa per conjunts

Cache amb 256 conjunts (de 4 bytes), associativa per conjunts de 4 vies
Quants comparadors fan falta?



Caches multinivell

Al principi, $t_{\text{accés}}$ era curt



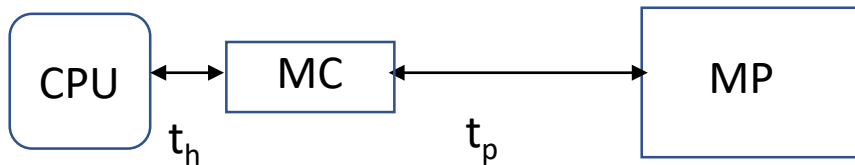
Caches multinivell

Amb el progrés de la CPU, $t_{\text{accés}}$ es fa molt llarg



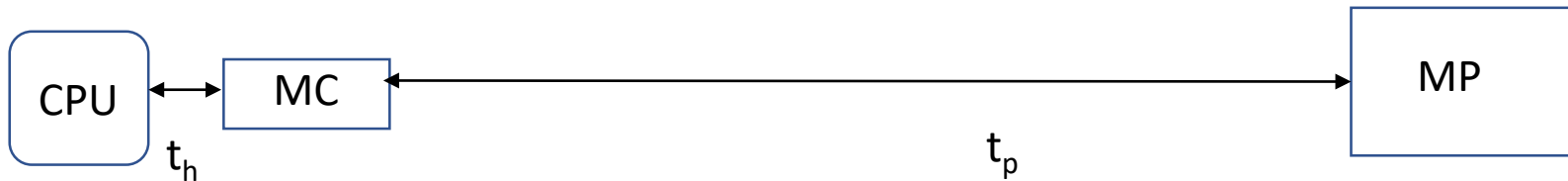
Caches multinivell

Solució amb MC: **Encerts ràpids (t_h)**, les fallades tarden un temps extra (t_p)



Caches multinivell

Amb el progrés de la CPU, t_p es fa molt més llarg en comparació a t_h

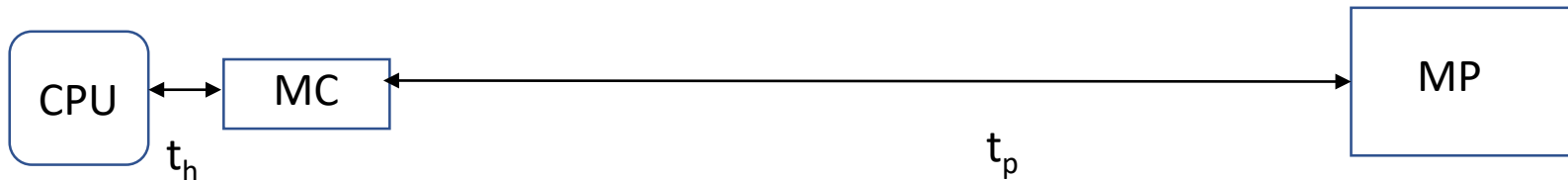


Caches multinivell

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

- A mesura que es redueix $\text{CPI}_{\text{ideal}}$, t_p és més important
 - Com reduir el temps de penalització t_p ?

Amb el progrés de la CPU, t_p es fa molt més llarg en comparació a t_h

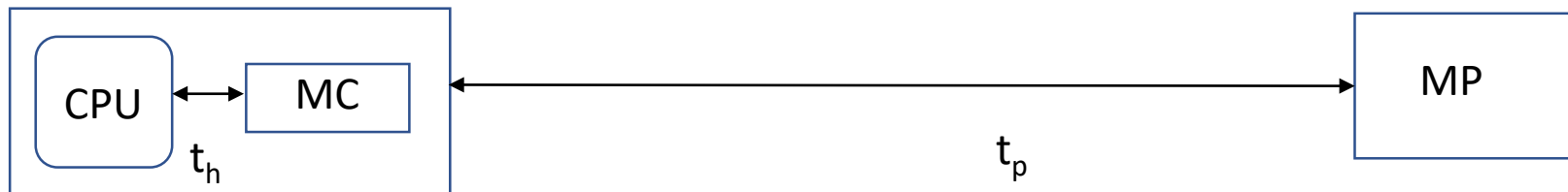


Caches multinivell

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

- A mesura que es redueix $\text{CPI}_{\text{ideal}}$, t_p és més important
 - Com reduir el temps de penalització t_p ?

Veiem que t_p sols afecta a les fallades!

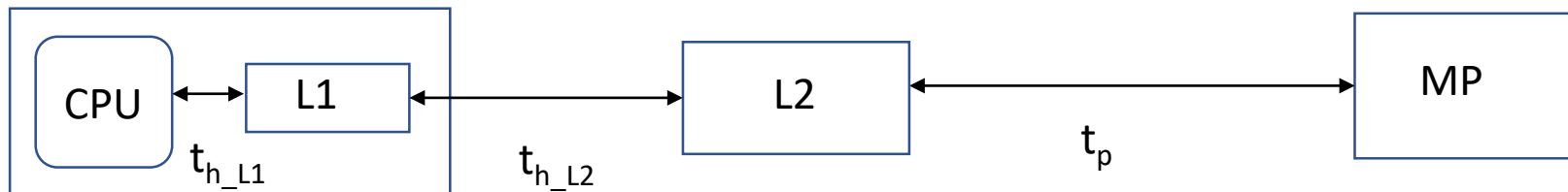


Caches multinivell

$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

- A mesura que es redueix $\text{CPI}_{\text{ideal}}$, t_p és més important
 - Com reduir el temps de penalització t_p ?
- Solució: caches multinivell

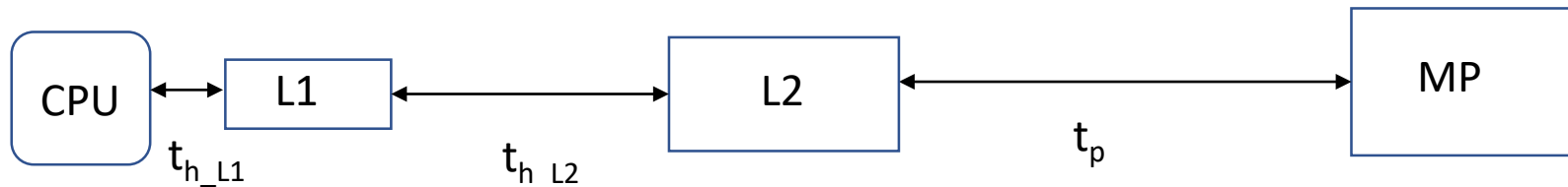
Apliquem la mateixa solució: una segona cache per a les fallades de L1



Caches multinivell

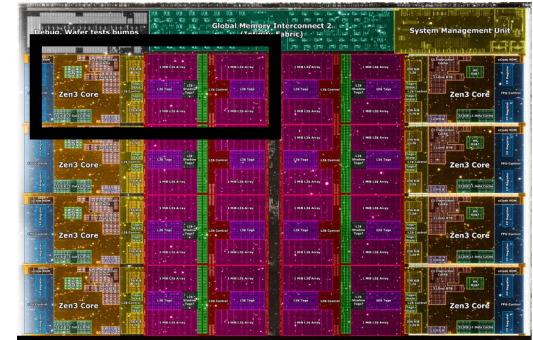
$$t_{\text{exe}} = (\text{CPI}_{\text{ideal}} + t_p \times m \times nr) \times n_{\text{ins}} \times t_c$$

- A mesura que es redueix $\text{CPI}_{\text{ideal}}$, t_p és més important
 - Com reduir el temps de penalització t_p ?
- Solució: caches multinivell
- La cache de segon nivell (L2)
 - És més gran, més lenta que la de primer nivell (L1)
 - Però si falla L1, podem trobar el bloc a L2, sense anar a MP



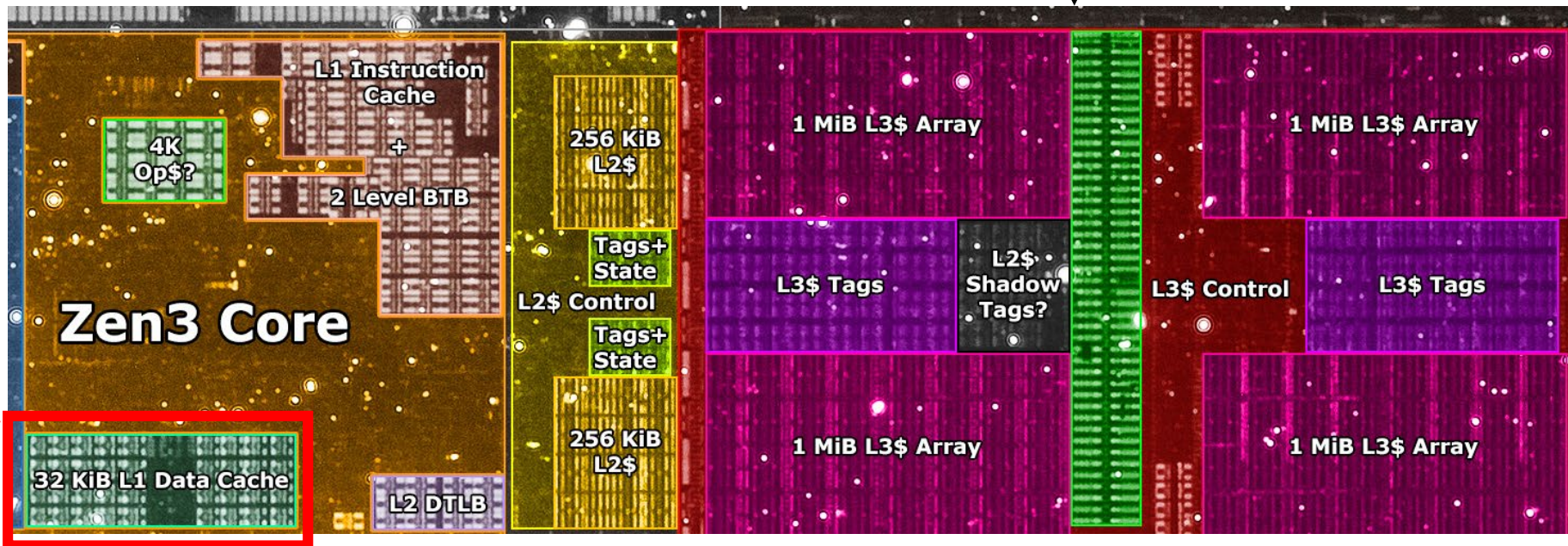
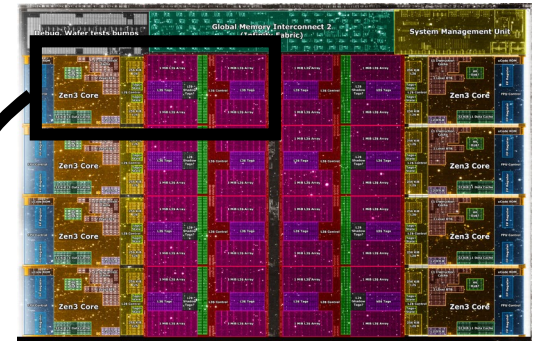
Caches multinivell

- Els computadors poden tenir més nivells: L1, L2, L3 ...
 - Solen integrar-se al mateix xip que la CPU
- Exemple: AMD Zen3 (8 cores)



Caches multinivell

- Els computadors poden tenir més nivells: L1, L2, L3 ...
 - Solen integrar-se al mateix xip que la CPU
- Exemple: AMD Zen3 (8 cores)
 - **L1 de 32 KB (per core), 8 vies**

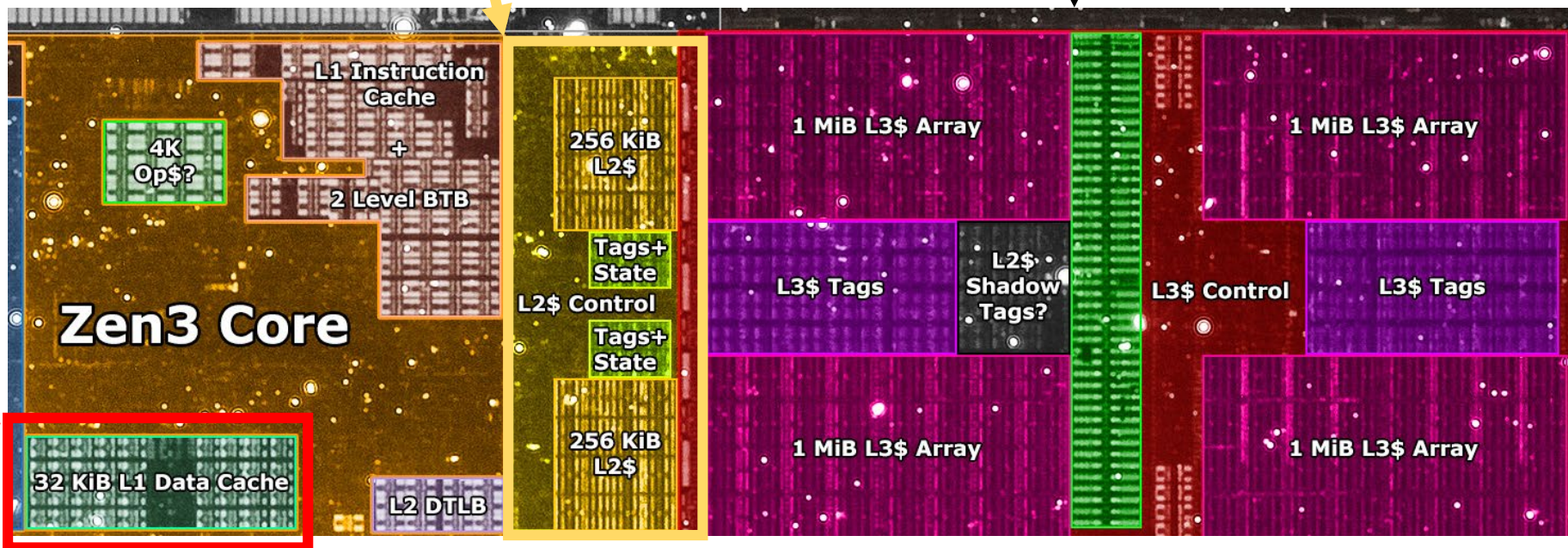
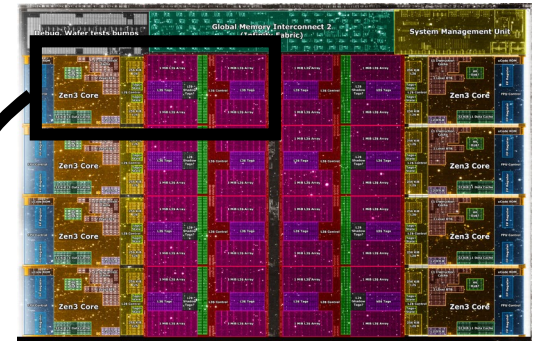


<https://i.redd.it/pa9c5mu9y2y51.jpg>

<https://www.amd.com/es/technologies/zen-core-3>

Caches multinivell

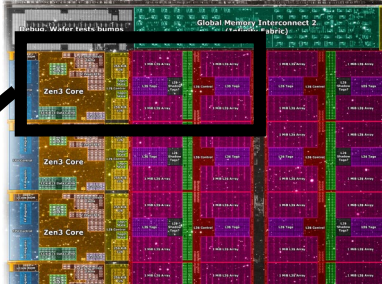
- Els computadors poden tenir més nivells: L1, L2, L3 ...
 - Solen integrar-se al mateix xip que la CPU
- Exemple: AMD Zen3 (8 cores)
 - **L1 de 32 KB (per core), 8 vies**
 - **L2 de 512 KB (per core), 8 vies**

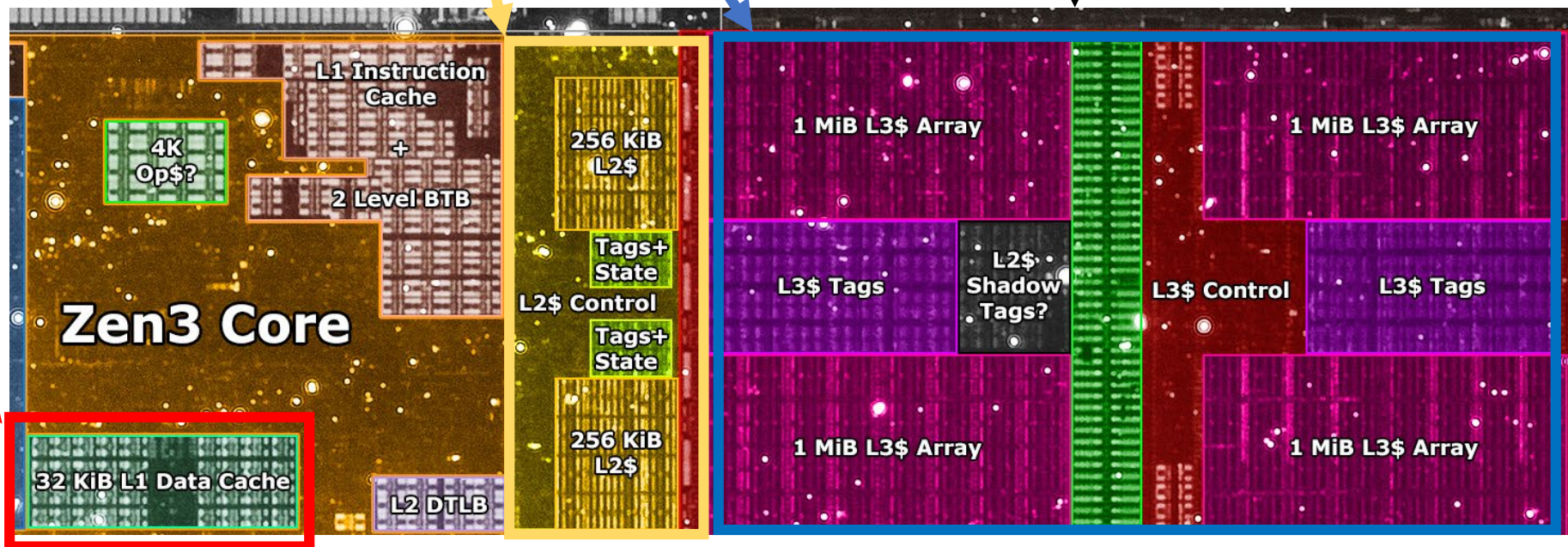
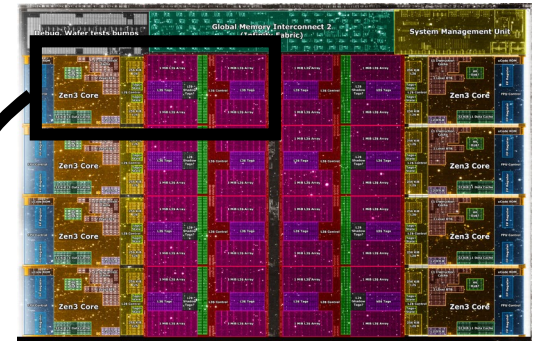


<https://i.redd.it/pa9c5mu9y2y51.jpg>

<https://www.amd.com/es/technologies/zen-core-3>

Caches multinivell

- Els computadors poden tenir més nivells: L1, L2, L3 ...
 - Solen integrar-se al mateix xip que la CPU
 - Exemple: AMD Zen3 (8 cores)
 - **L1** de 32 KB (per core), 8 vies
 - **L2** de 512 KB (per core), 8 vies
 - **L3** de 32 MB (compartida), 16 vies
- 
- A die photograph of an AMD Zen3 processor. The die is rectangular and populated with numerous small, colorful components. A black rectangular box highlights a specific area on the left side of the die, which is labeled 'Zen3 Core'. A thick black line originates from this box and points towards the text 'L3 de 32 MB (compartida), 16 vies' in the list, indicating the shared L3 cache's location and connectivity. Other labels visible on the die include 'Global Memory Interconnect 2' and 'Zen3 Core' repeated in several locations.



Tipologia de les fallades (CCC)

- Arrancada en fred o Obligatoria (**C**old o **C**ompulsory)
 - Primer cop que es referencia el bloc

Tipologia de les fallades (CCC)

- Arrancada en fred o Obligatòria (**C**old o **C**ompulsory)
 - Primer cop que es referencia el bloc
- Per **C**onflicte
 - Blocs ja visitats anteriorment
 - Blocs que es mapegen a la mateixa línia de MC i s'expulsen mútuament
 - Es poden evitar augmentant el *grau d'associativitat* (num vies)

Tipologia de les fallades (CCC)

- Arrancada en fred o Obligatòria (**C**old o **C**ompulsory)
 - Primer cop que es referencia el bloc
- Per **C**onflicte
 - Blocs ja visitats anteriorment
 - Blocs que es mapegen a la mateixa línia de MC i s'expulsen mútuament
 - Es poden evitar augmentant el *grau d'associativitat* (num vies)
- Per **C**apacitat
 - Blocs ja visitats anteriorment
 - No es poden evitar augmentant l'associativitat, només augmentant la *capacitat*

Exemple: Fallades per Conflicte

○ Conflict misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (correspondència directa)

```
int A[8], B[8];
int i, s=0;
for (i=0; i<8 ; i++)
    s = s + A[i] * B[i];
```

Memòria Cache

	V	etiqueta	Dades	
0	0			
1	0			
2	0			
3	0			

	MP	num bloc
	...	
A:	A[0]	0
	A[1]	
	A[2]	1
	A[3]	
	A[4]	2
	A[5]	
	A[6]	3
	A[7]	
B:	B[0]	4
	B[1]	
	B[2]	5
	B[3]	
	B[4]	6
	B[5]	
	B[6]	7
	B[7]	
	...	

Exemple: Fallades per Conflicte

○ Conflict misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (correspondència directa)

```
int A[8], B[8];  
int i, s=0;  
for (i=0; i<8 ; i++)  
    s = s + A[i] * B[i];
```

Memòria Cache

	V	etiqueta	Dades	
0	1		A[0]	A[1]
1	0			
2	0			
3	0			

Cold miss → A:

MP	num bloc
...	
A[0]	0
A[1]	
A[2]	1
A[3]	
A[4]	2
A[5]	
A[6]	3
A[7]	
B[0]	4
B[1]	
B[2]	5
B[3]	
B[4]	6
B[5]	
B[6]	7
B[7]	
...	

Exemple: Fallades per Conflicte

○ Conflict misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (correspondència directa)

```
int A[8], B[8];
int i, s=0;
for (i=0; i<8 ; i++)
    s = s + A[i] * B[i];
```

Memòria Cache

	V	etiqueta	Dades	
0	1		B[0]	B[1]
1	0			
2	0			
3	0			

	MP	num bloc
	...	
Cold miss → A:	A[0]	0
	A[1]	
	A[2]	1
	A[3]	
	A[4]	2
	A[5]	
	A[6]	3
	A[7]	
Cold miss → B:	B[0]	4
	B[1]	
	B[2]	5
	B[3]	
	B[4]	6
	B[5]	
	B[6]	7
	B[7]	
	...	

Exemple: Fallades per Conflicte

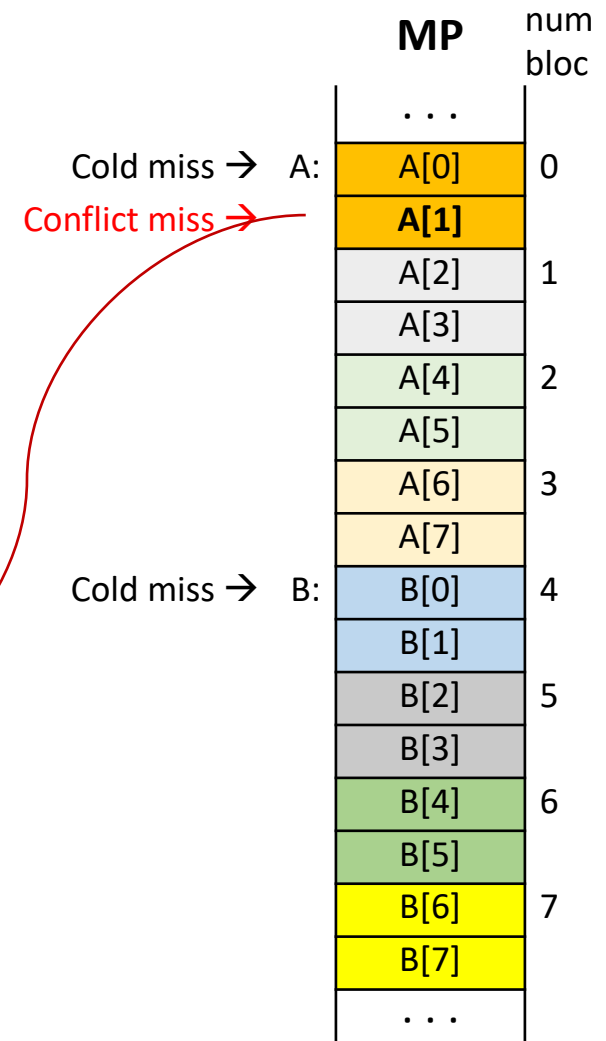
○ Conflict misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (correspondència directa)

```
int A[8], B[8];  
int i, s=0;  
for (i=0; i<8 ; i++)  
    s = s + A[i] * B[i];
```

Memòria Cache

	V	etiqueta	Dades	
0	1		A[0]	A[1]
1	0			
2	0			
3	0			



Exemple: Fallades per Conflicte

○ Conflict misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (correspondència directa)

```
int A[8], B[8];  
int i, s=0;  
for (i=0; i<8 ; i++)  
    s = s + A[i] * B[i];
```

Memòria Cache			
	V	etiqueta	Dades
0	1		B[0] B[1]
1	0		
2	0		
3	0		

MP		num bloc
...		
Cold miss → A:	A[0]	0
Conflict miss →	A[1]	
	A[2]	1
	A[3]	
	A[4]	2
	A[5]	
	A[6]	3
	A[7]	
Cold miss → B:	B[0]	4
Conflict miss →	B[1]	
	B[2]	5
	B[3]	
	B[4]	6
	B[5]	
	B[6]	7
	B[7]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];  
int i, s=0;
```

```
for (i=0; i<16 ; i++)  
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)  
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	0			
1	0			
2	0			
3	0			

	MP	num bloc
V:	...	
	V[0]	0
	V[1]	
	V[2]	1
	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	0			
2	0			
3	0			

MP	num bloc
...	
V[0]	0
V[1]	
V[2]	1
V[3]	
V[4]	2
V[5]	
V[6]	3
V[7]	
V[8]	4
V[9]	
V[10]	5
V[11]	
V[12]	6
V[13]	
V[14]	7
V[15]	
...	

Cold miss → V:

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	0			
2	0			
3	0			

MP	num bloc
...	
Cold miss → V: V[0]	0
Hit → V[1]	
V[2]	1
V[3]	
V[4]	2
V[5]	
V[6]	3
V[7]	
V[8]	4
V[9]	
V[10]	5
V[11]	
V[12]	6
V[13]	
V[14]	7
V[15]	
...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[2]	V[3]
2	0			
3	0			

	MP	num bloc
	...	
Cold miss → V:	V[0]	0
Hit →	V[1]	
Cold miss →	V[2]	1
Hit →	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[2]	V[3]
2	1		V[4]	V[5]
3	0			

	MP	num bloc
	...	
Cold miss → V:	V[0]	0
Hit →	V[1]	
Cold miss →	V[2]	1
Hit →	V[3]	
Cold miss →	V[4]	2
Hit →	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[2]	V[3]
2	1		V[4]	V[5]
3	1		V[6]	V[7]

	MP	num bloc
	...	
Cold miss → V:	V[0]	0
Hit →	V[1]	
Cold miss →	V[2]	1
Hit →	V[3]	
Cold miss →	V[4]	2
Hit →	V[5]	
Cold miss →	V[6]	3
Hit →	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[8]	V[9]
1	1		V[2]	V[3]
2	1		V[4]	V[5]
3	1		V[6]	V[7]

	MP	num bloc
	...	
Cold miss → V:	V[0]	0
Hit →	V[1]	
Cold miss →	V[2]	1
Hit →	V[3]	
Cold miss →	V[4]	2
Hit →	V[5]	
Cold miss →	V[6]	3
Hit →	V[7]	
Cold miss →	V[8]	4
Hit →	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[8]	V[9]
1	1		V[10]	V[11]
2	1		V[12]	V[13]
3	1		V[14]	V[15]

	MP	num bloc
	...	
Cold miss → V:	V[0]	0
Hit →	V[1]	
Cold miss →	V[2]	1
Hit →	V[3]	
Cold miss →	V[4]	2
Hit →	V[5]	
Cold miss →	V[6]	3
Hit →	V[7]	
Cold miss →	V[8]	4
Hit →	V[9]	
Cold miss →	V[10]	5
Hit →	V[11]	
Cold miss →	V[12]	6
Hit →	V[13]	
Cold miss →	V[14]	7
Hit →	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[8]	V[9]
1	1		V[10]	V[11]
2	1		V[12]	V[13]
3	1		V[14]	V[15]

	MP	num bloc
	...	
V:	V[0]	0
	V[1]	
	V[2]	1
	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[10]	V[11]
2	1		V[12]	V[13]
3	1		V[14]	V[15]

	MP	num bloc
	...	
V:	V[0]	0
	V[1]	
	V[2]	1
	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Capacity miss → V:

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[10]	V[11]
2	1		V[12]	V[13]
3	1		V[14]	V[15]

	MP	num bloc
	...	
Capacity miss → V:	V[0]	0
Hit →	V[1]	
	V[2]	1
	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	

Exemple: Fallades per Capacitat

○ Capacity misses

- Processador de 32 bits
- Blocs de 8 bytes (=2 words)
- MC de 4 línies (**totalment associativa**)

```
int V[16];
int i, s=0;
```

```
for (i=0; i<16 ; i++)
    s = s + V[i];
```

```
for (i=0; i<16 ; i++)
    s = s + V[i]*3;
```

Memòria Cache

	V	etiqueta	Dades	
0	1		V[0]	V[1]
1	1		V[2]	V[3]
2	1		V[12]	V[13]
3	1		V[14]	V[15]

	MP	num bloc
	...	
Capacity miss → V:	V[0]	0
Hit →	V[1]	
Capacity miss →	V[2]	1
	V[3]	
	V[4]	2
	V[5]	
	V[6]	3
	V[7]	
	V[8]	4
	V[9]	
	V[10]	5
	V[11]	
	V[12]	6
	V[13]	
	V[14]	7
	V[15]	
	...	