Tema 6. Memòria Cache

Estructura de Computadors (EC)

Rubèn Tous

rtous@ac.upc.edu
Computer Architecture Department
Universitat Politecnica de Catalunya

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

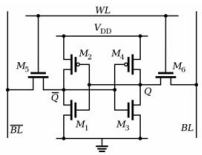
Índex

6.1.1 Memòria d'un computador

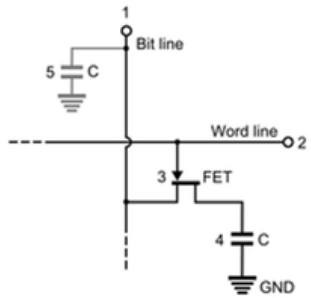
- Memòria persistent (emmagatzemament secundari):
  - Discos durs (magnètics) o xips de memòria SSD (d'estat sòlid).
  - Cost per bit molt baix i gran capacitat però temps d'accés extremadament lent (5 ms en el disc dur, i més de 0,1 ms en els SSD).
- Memòria principal d'accés aleatori o RAM (random access memory)
  - Volàtil i d'accés molt més ràpid que el disc (50 ns).
  - Instruccions i les dades d'un programa mentre s'executa.

6.1.1 Memòria d'un computador

- La RAM es pot implementar com memòria estàtica (SRAM) o dinàmica (DRAM).
- SRAM més costosa (cada cel·la 6 transistors per bit) que DRAM (1 transistor i un element capacitiu per bit).



(a) 1 bit en SRAM (6 T)



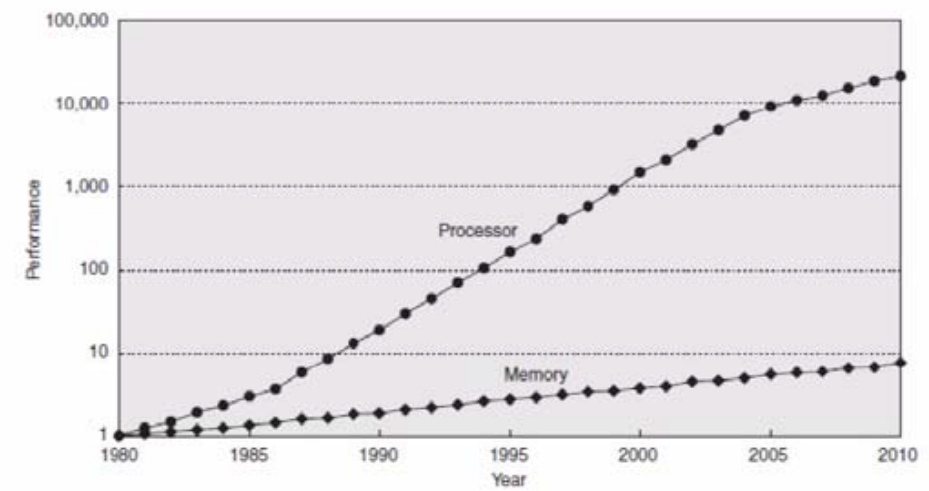
(b) 1 bit en DRAM (1 T + 1C)

- Solem usar DRAM per a la memòria principal (més capacitat per a una mateixa àrea de xip i menor cost per bit).

Índex

6.1.2 El problema del gap entre memòria i processador

- Augment exponencial del rendiment del processador (reducció dels temps de commutació i creixent processament paral·lel d'instruccions).
- Reducció temps d'accés a la memòria principal (DRAM) més lent.



6.1.2 El problema del gap entre memòria i processador

Aclariment sobre la terminologia

- L'execució d'una instrucció implica com a mínim un accés a la memòria (fetch o cerca de la instrucció).
- A més a més, les de load o store realitzen un segon accés per llegir o escriure una dada.
- "dada" = instrucció o dada.
- "referència" o "accés" a memòria = lectura o una escriptura d'una instrucció o dada.

6.1.2 El problema del gap entre memòria i processador

- El temps d'un accés a memòria (t_m) pot arribar a ser de 100 o 200 cicles de CPU.
- 100 cops major que el temps (t_{proc}) emprat en la resta d'operacions del processament d'una instrucció (incloses descodificació, lectura de registres, ALU, etc.).
- Diferència que anomenem **gap entre processador i memòria**.

Índex

6.1.3 Localitat dels programes

- Part de la solució a aquest problema sorgeix de l'anàlisi del comportament dels programes.
- Hi ha dades que són accedides amb major freqüència que altres: principis de localitat:
  - **Localitat temporal**: Si accedim a una adreça de memòria, és probable que hi tornem a accedir en un futur proper. Motiu: bucles en els programes.
  - **Localitat espacial**: Si accedim a una adreça de memòria, és probable que s'accedeixi a adreces "properes" en un futur proper. Motius: execució seqüencial i recorreguts de vectors/matrius.

6.1.3 Localitat dels programes

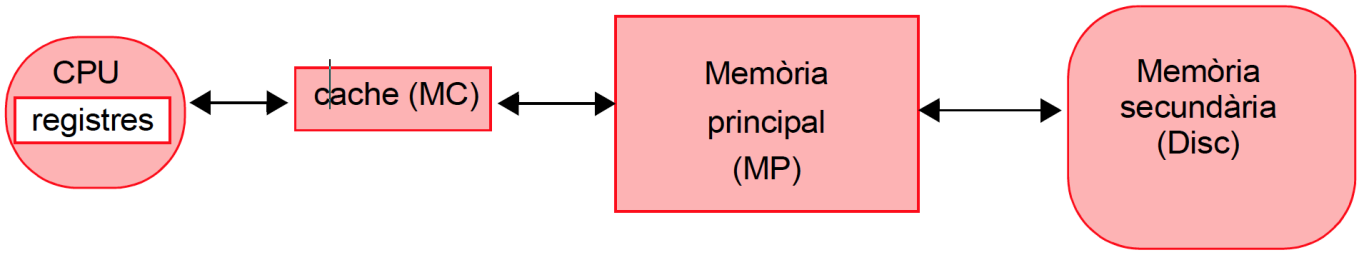
Analogia de la biblioteca:

- Principi de localitat en l'accés als llibres d'una biblioteca.
- Si agafem un llibre, la probabilitat de que tornem a agafar el mateix llibre en un futur proper és molt més alta que la probabilitat de que agafem un llibre qualsevol (localitat temporal).
- La probabilitat de que agafem un llibre del mateix prestatge també serà més alta (principi de localitat espacial).

Índex

6.1.4 La memòria cache

- Memòria cache (MC) = memòria auxiliar petita i ràpida.
- S'interposa entre el processador i la memòria principal (MP), més gran i més lenta.
- A diferència de l'MP, generalment implementada en DRAM, la cache s'implementa en SRAM i sovint s'integra dins el mateix xip del processador.



	registres	SRAM	DRAM	Hard disk
capacitat:	0,5 KB	64KB	10GB	1 TB
t_accés:	0,25 ns	0,5-2,5 ns	50-100 ns	5-20 ms
cost per GB:		2000-5000 $	10 $	0.020 $

Capacitat, temps d'accés i cost per GB típics

6.1.4 La memòria cache

- L'MC emmagatzema un subconjunt del contingut de l'MP.
- Objectiu: retenir aquelles dades que tinguin més probabilitat de ser accedides en el futur.
- Cada cop que la CPU accedeix a una nova dada de l'MP, la copiem a l'MC juntament amb les dades "properes" (en realitat copiem un *bloc* de dades).

6.1.4 La memòria cache

- Localitat temporal: Si en endavant la mateixa dada es torna a referenciar podrem accedir-la directament a l'MC.
- Localitat espacial: Si en endavant accedim a una altra dada que està pròxima a aquesta (en el seu mateix *bloc*), s'hi accedirà a l'MC en un temps molt més curt.

6.1.4 La memòria cache

Analogia de la biblioteca (cache = taula):

- Quan agafem un llibre del prestatge aprofitem per agafar altres llibres del mateix prestatge i els deixem tots a la taula on estem treballant.
- Si tornem a necessitar el llibre (cosa probable degut al principi de localitat temporal) ens estalviarem d'haver-lo d'anar a buscar.
- Si necessitem un llibre del mateix prestatge (cosa probable degut al principi de localitat espacial) també ens estalviarem d'haver-lo d'anar a buscar.

Índex

6.1.5 Terminologia

- **Fallada**
  - Primer la CPU intenta trobar la dada a la cache. Si no hi és direm que es produeix una *fallada* (*miss*).
  - Es copiarà de l'MP a l'MC el bloc al que la dada pertany.
- **Reemplaçament**: Si el lloc de l'MC al que correspon el bloc ja està ocupat per un altre bloc caldrà *reemplaçar-lo*.
- **Encert**: Si la dada és a l'MC es produeix un *encert* (*hit*).

6.1.5 Terminologia

- **num_referencies** = Número de referències a memòria (fetch/lectures i escriptures).
- **num_encerts** = Referències que s'han resolt amb encert a la cache.
- **num_fallades** = Referències que han provocat fallada a la cache.
- **Taxa d'encert**

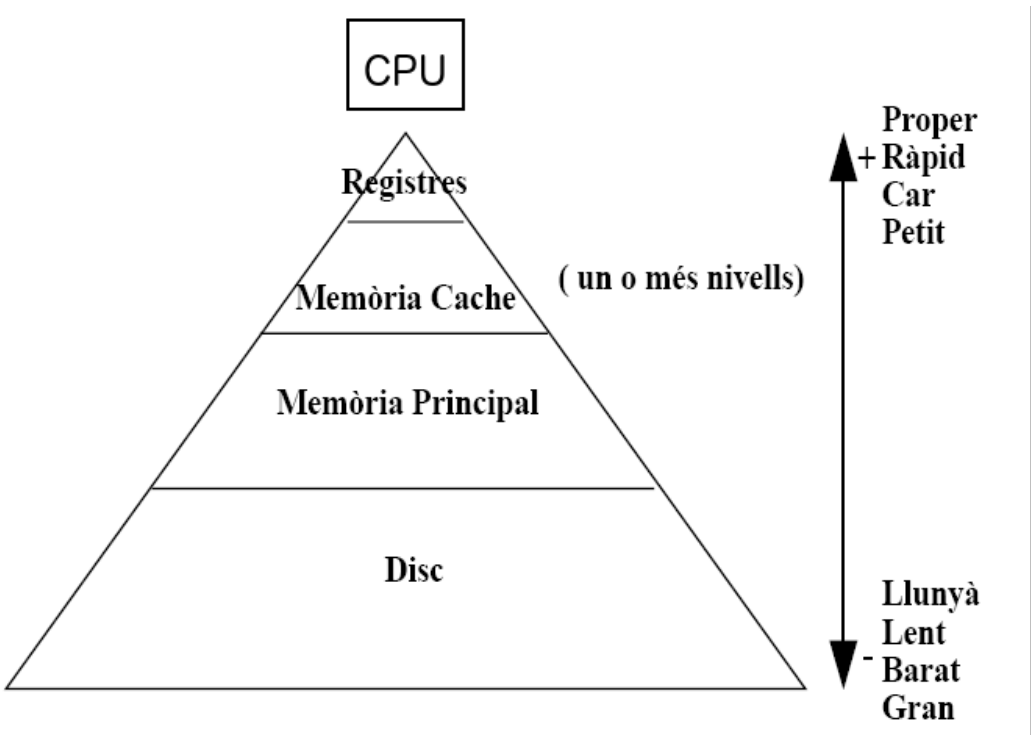
$$h = \frac{num\_encerts}{num\_referencies}$$

- **Taxa de fallada**

$$m = \frac{num\_fallades}{num\_referencies} = 1 - h$$

Índex

6.1.4 La memòria cache



Jerarquia de memòria

Índex

Índex

6.2.1 Organització de la memòria en blocs

El bloc

La cache no guarda dades soltes sinó *blocs* sencers. Això permet aprofitar el principi de localitat espacial (el bloc conté les dades "properes"). Un bloc té mida fixa *TAMBLOC* bytes, essent *TAMBLOC* una potència de 2 (mides típiques són entre 16 i 128 bytes).

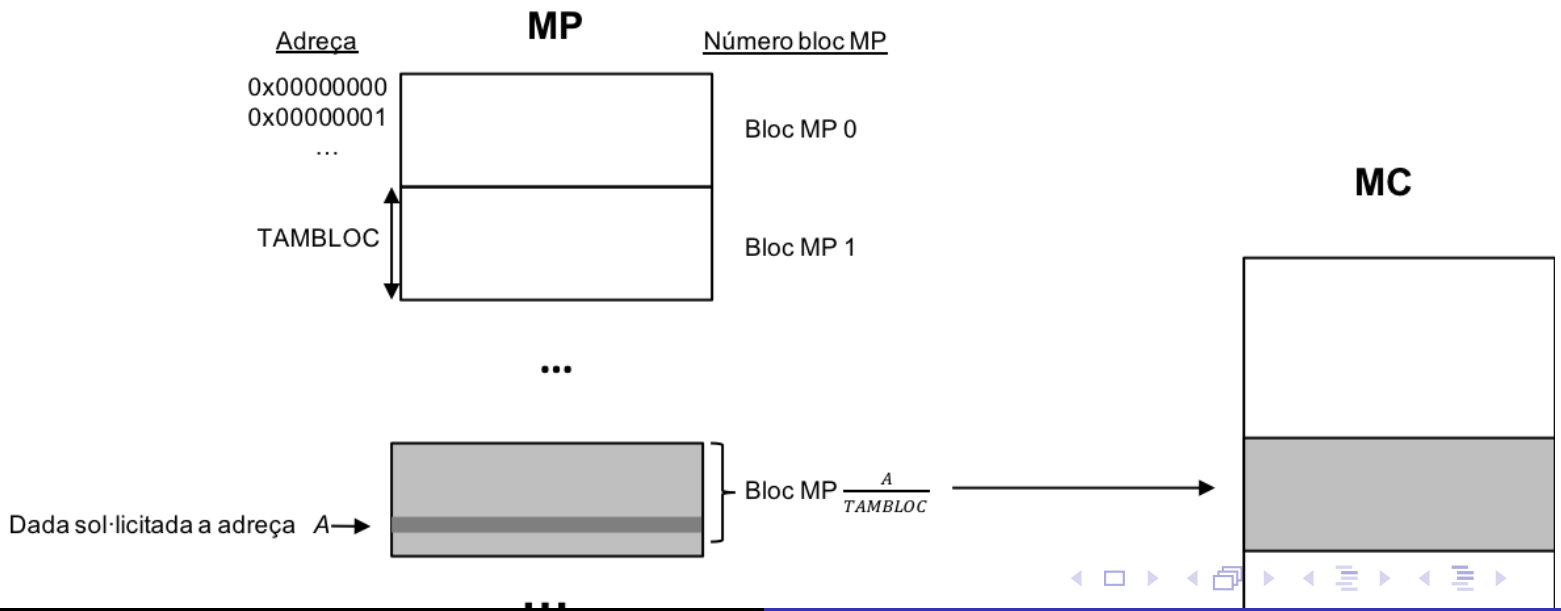
Analogia de la biblioteca (bloc = prestatge)

A l'analogia de la biblioteca el *bloc* podria correspondre, per exemple, a tots els llibres d'un prestatge. Cada cop que necessitéssim un llibre portaríem a la taula tots els llibres del prestatge.

6.2.1 Organització de la memòria en blocs

Si numerem els blocs en que podem dividir la memòria principal de 0 a N-1, podem esbrinar el número de bloc al que pertany una adreça *A* (el *número de bloc MP*) fent:

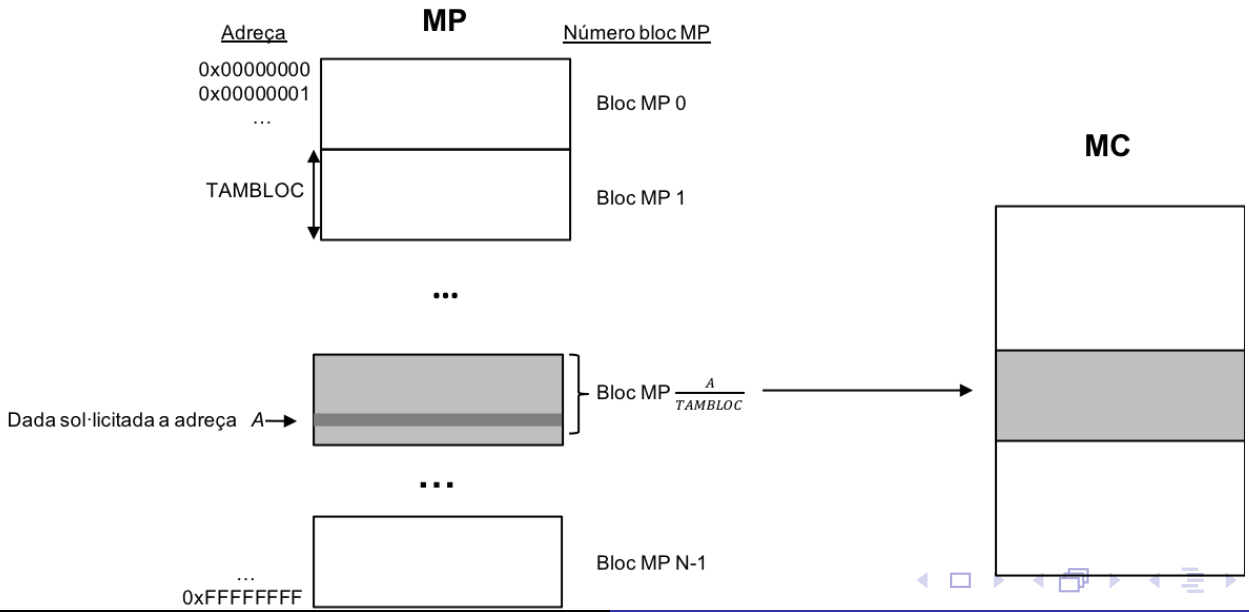
$$\textit{número de bloc MP} = \frac{A}{TAMBLOC}$$



6.2.1 Organització de la memòria en blocs

Si numerem els blocs en que podem dividir la memòria principal de 0 a N-1, podem esbrinar el número de bloc al que pertany una adreça *A* (el *número de bloc MP*) fent:

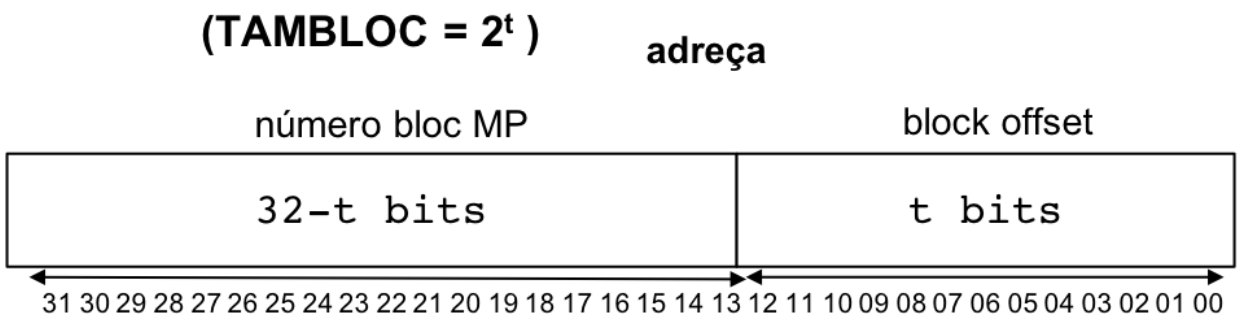
$$\textit{número de bloc MP} = \frac{A}{TAMBLOC}$$



6.2.1 Organització de la memòria en blocs

Com $TAMBLOC = 2^t$, això equival a descartar els t bits de menys pes de l'adreça (el *block offset*):

$$block\ offset = A \mod TAMBLOC$$

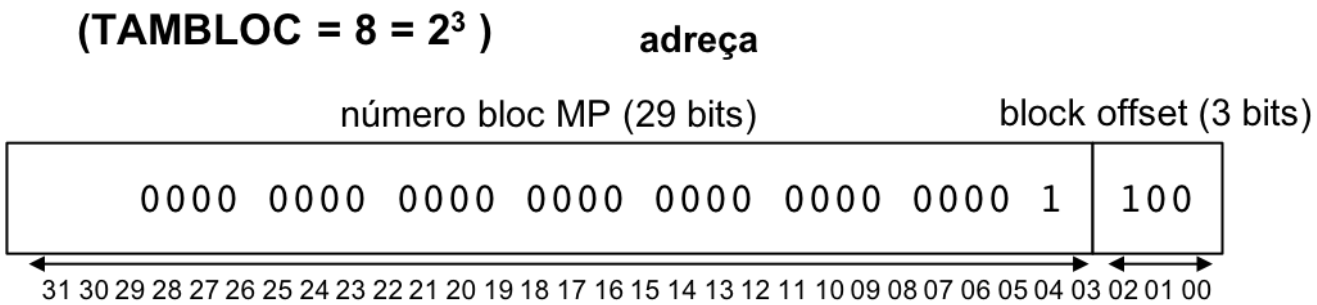


Número de bloc MP i block offset.

6.2.1 Organització de la memòria en blocs

Exemple:

- Blocs de dos paraules cadascun (*TAMBLOC* = 8 bytes i $t = 3$)
- Es produeix una lectura a l'adreça 0x0000000C.



Número de bloc MP i block offset (*A*=0x0000000C).

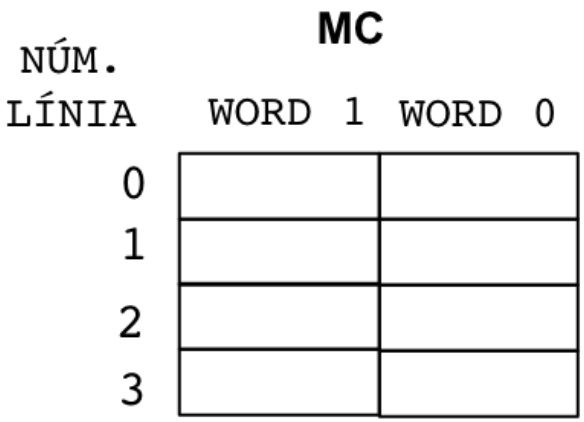
Índex

6.2.2 Cache de correspondència directa

- Ón guarda els blocs la cache?
- Política de *ubicació* (*placement*).
- La política més senzilla és la de *correspondència directa* (*direct-mapped*).

6.2.2 Cache de correspondència directa

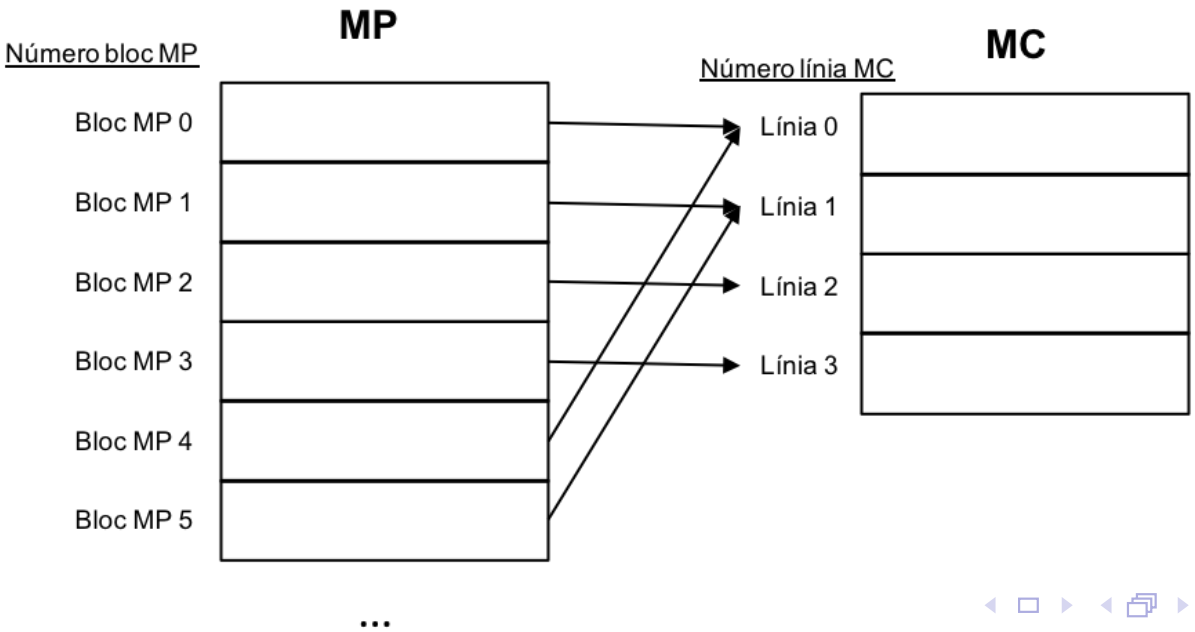
- Una cache de correspondència directa consta d'un número de *línies* determinat (NUM_LíNIES).
- A cada *línia* hi pot anar un bloc de memòria principal.
- Per exemple, podem imaginar-nos una MC de 4 línies i blocs de 2 paraules cadascun.



Exemple d'MC de correspondència directa amb 4 línies de 2 paraules

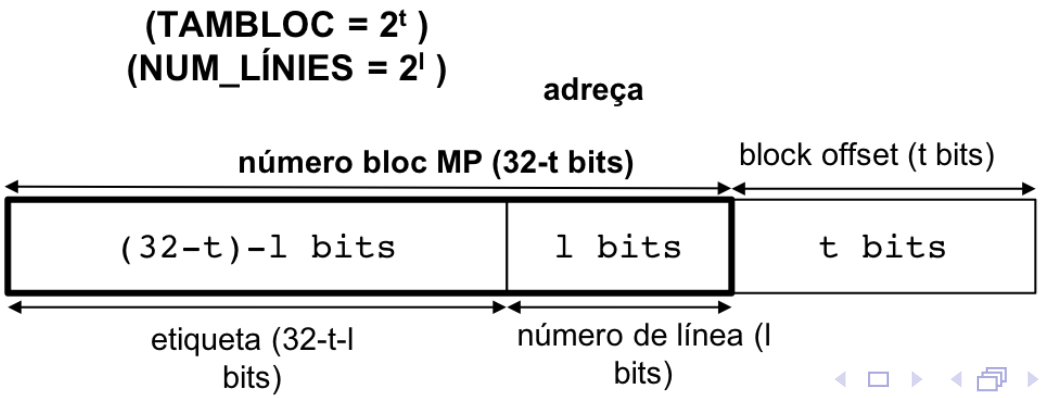
6.2.2 Cache de correspondència directa

- Es determina a quina línia d'MC s'ubiquen els blocs de memòria principal en funció del número de bloc MP.
  número de línia MC = número de bloc MP mod NUM_LíNIES.



6.2.2 Cache de correspondència directa

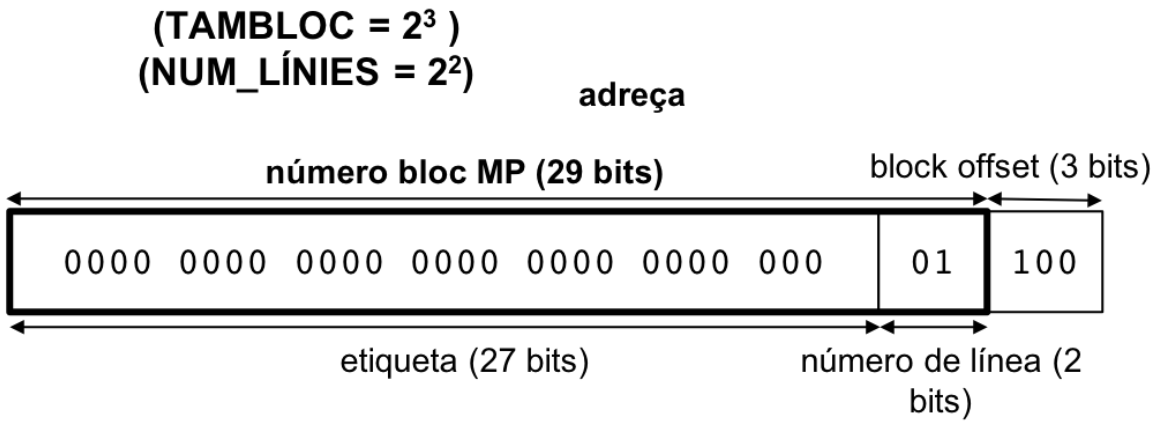
- Podrem saber a quina línia de cache correspón el bloc consultant els *l* bits de menys pes dels bits de *número de bloc MP*.
- La resta de bits dels bits del *número de bloc MP* de l'adreça s'anomenen *etiqueta* (*tag*).
- L'etiqueta ens permetrà distingir entre blocs d'MP diferents, encara que corresponguin a la mateixa línia d'MC.



6.2.2 Cache de correspondència directa

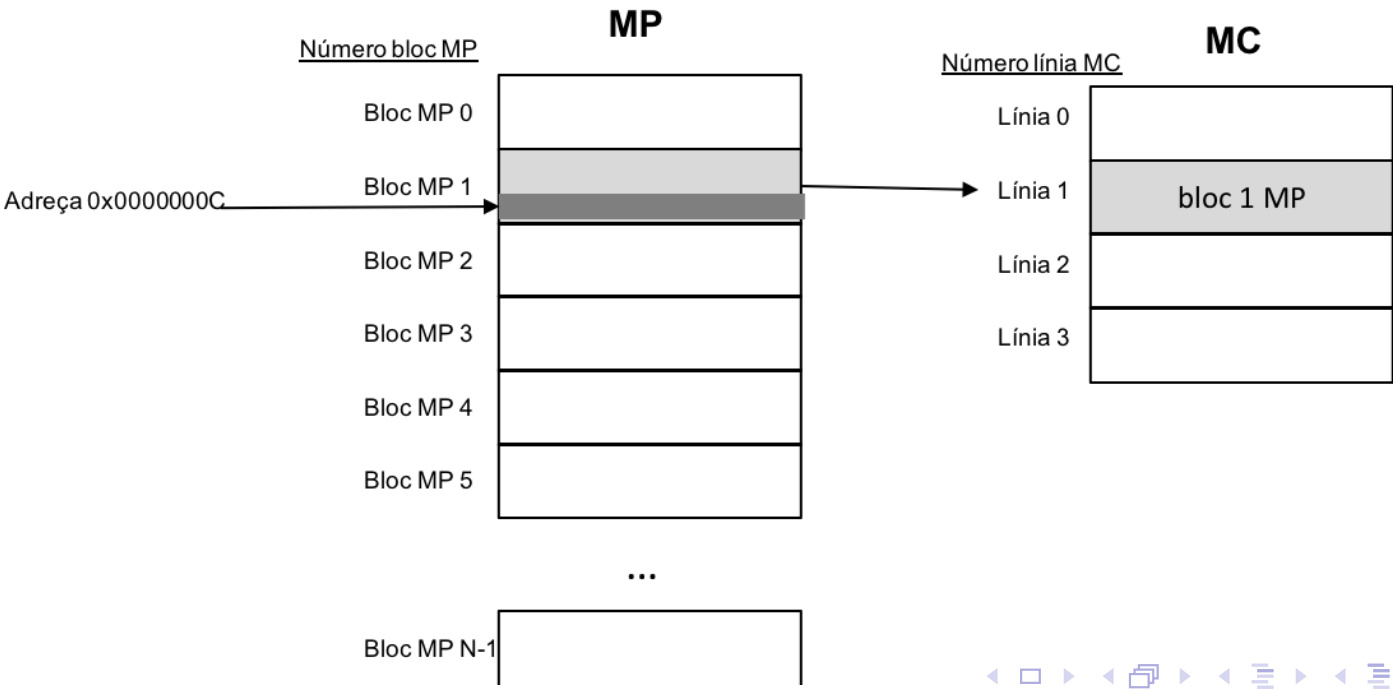
Exemple:

- Cache de 4 línies i blocs de 2 paraules cadascun.
- Lectura a l'adreça 0x0000000C.



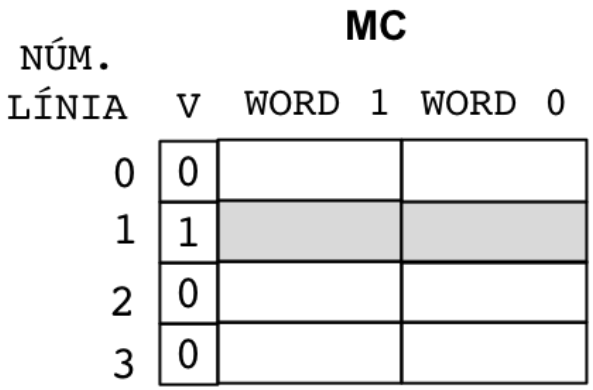
6.2.2 Cache de correspondència directa

- Podem determinar si el bloc està a la cache simplement mirant l'adreça.
- En cas que el bloc no hi sigui, serà copiat a aquesta línia.



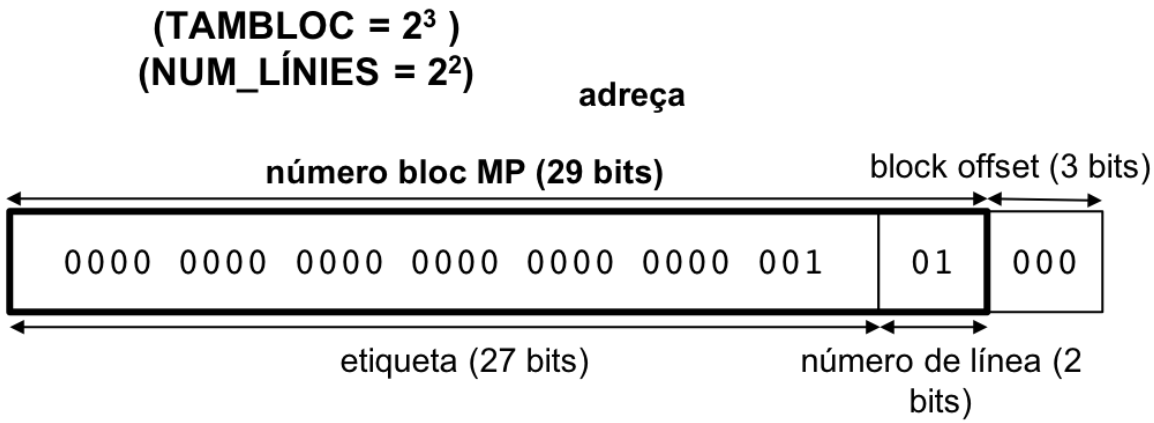
6.2.2 Cache de correspondència directa

Però com sabrem si el bloc ja és a la línia o no? *bit de validesa* (V):



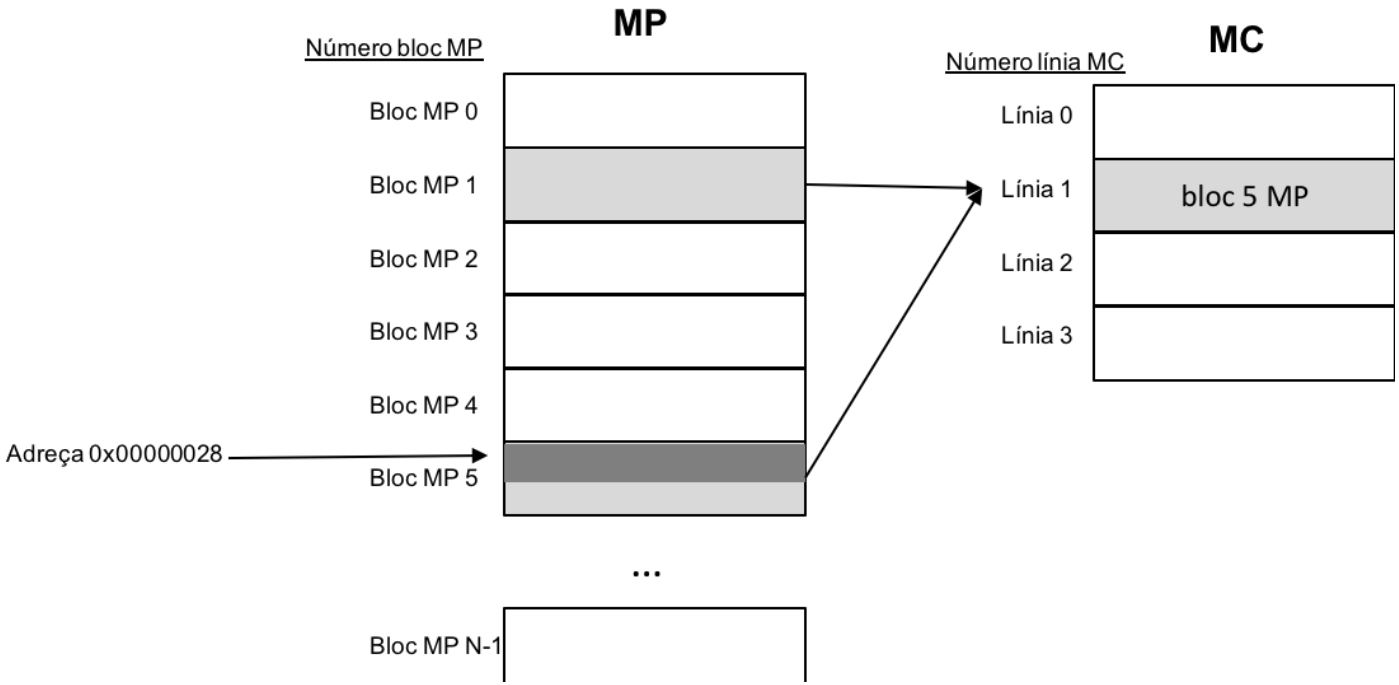
6.2.2 Cache de correspondència directa

- Què passaria si a continuació es produeix una lectura a l'adreça 0x00000028?
- L'adreça pertany al bloc MP 5, que també correspon a la línia 1.



6.2.2 Cache de correspondència directa

Ens caldrà afegir a la memòria cache alguna cosa que ens permeti distingir uns blocs del altres.



6.2.2 Cache de correspondència directa

Guardarem també l'etiqueta de cada bloc.

NÚM. LÍNIA	V	etiqueta	MC WORD 1	MC WORD 0
0	0	0x0		
1	1	0x1		
2	0	0x0		
3	0	0x0		

6.2.2 Cache de correspondència directa

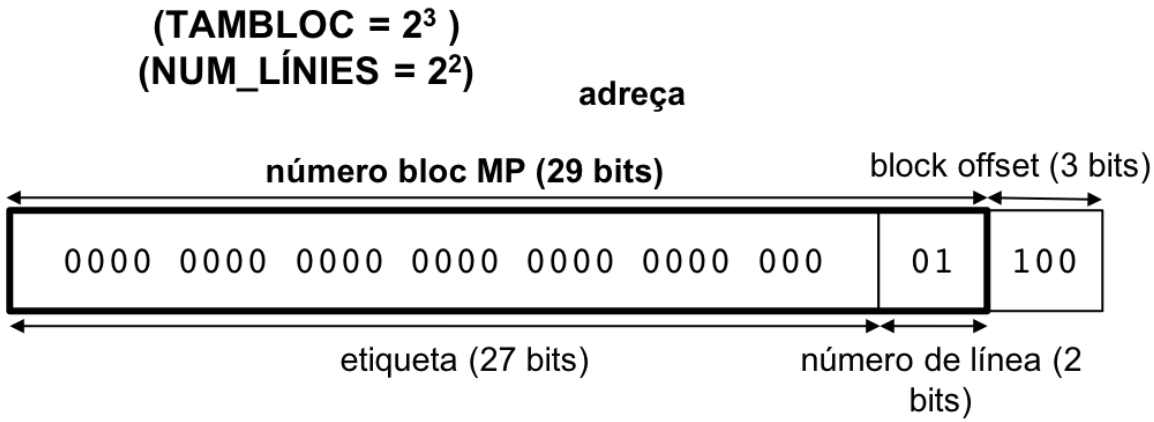
Reemplaçament

Quan s'ha de copiar un bloc a una línia de la cache però la línia ja conté un altre bloc (amb una etiqueta diferent) es reemplaçarà el bloc antic pel bloc nou.

- A l'exemple anterior, el bloc 5 d'MP reemplaça el bloc 1 a la línia 1.
- Quan parlem d'escriptures, veurem que els reemplaçaments poden implicar més accions.

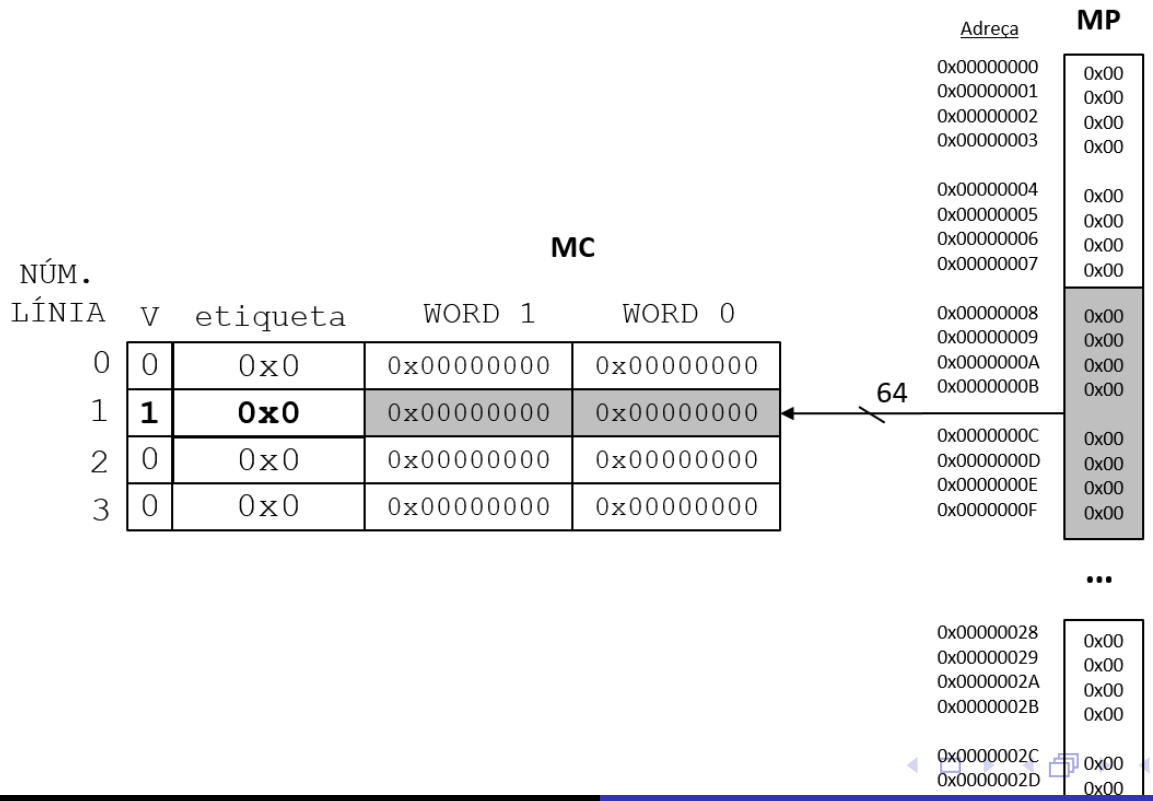
Exemple complert

- Cache de 4 línies i blocs de 2 paraules cadascun.
- Lectura de l'adreça 0x0000000C (suposem un *lw*).



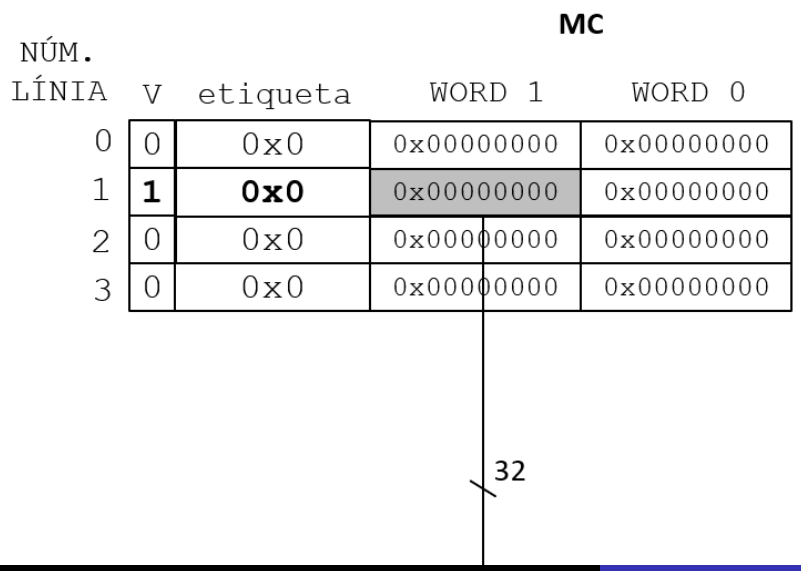
Exemple complert

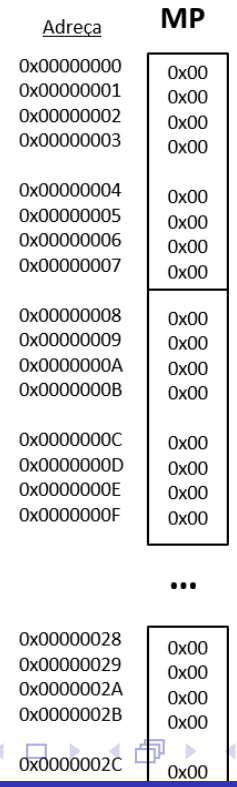
- 0x0000000C ⇒ bloc 1 d'MP ⇒ línia 1.
- V = 0 ⇒ fallada ⇒ Bloc 1 a MC.



Exemple complert

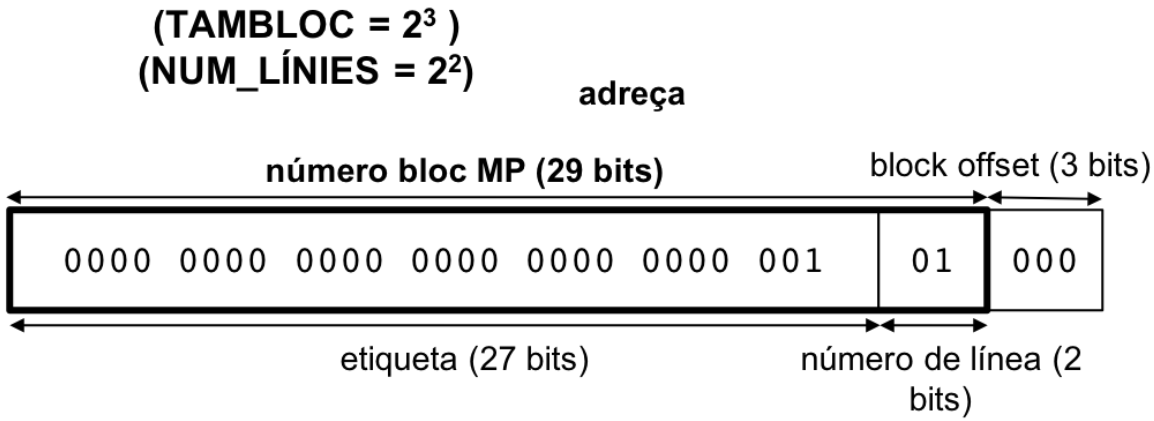
- Un cop s'ha copiat el bloc, el processador accedeix a la dada (una paraula) a l'MC.





Exemple complert

- A continuació lectura de l'adreça 0x00000028.
- Bloc 5 d'MP (també correpon a la línia 1).



Exemple complert

- Bit V a 1, etiqueta diferent (la 1) $\Rightarrow$ fallada.
- Reemplaçament: el bloc 1 d'MP és reemplaçat pel bloc 5.