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

6.3.1 Model de temps

- t_{acces} = temps de servei d'una referència a memòria.
- $t_{acces} = t_h + t_p$
- t_h = temps de determinar si és un encert o una fallada i servir la referència en cas d'encert.
- t_p = temps de penalització per resoldre la referència en accedir al següent nivell de la jerarquia de memòria.

6.3.1 Model de temps

buffer d'escriptura

Quan tinguem política d'escriptura immediata es considera l'existència d'un **buffer d'escriptura** amb llargada il·limitada on queden emmagatzemades les escriptures pendents de portar a MP. **No caldrà esperar que la dada s'escrigui a MP**.

6.3.1 Model de temps

t_p	Immediata sense assig.	Retardada amb assignació
Lectura - Encert	0	0
Lectura - Fallada	$t_{block} + t_h$	bloc modif.: $2 * t_{block} + t_h$ bloc no mod.: $t_{block} + t_h$
Escriptura - Encert	0^3	0
Escriptura - Fallada	0^3	bloc modif.: $2 * t_{block} + t_h$ bloc no mod.: $t_{block} + t_h$

[3]buffer d'escriptura.

6.3.1 Model de temps

- En el cas de la l'*escriptura immediata sense assignació* l'única penalització introduïda per les fallades serà la penalització de les fallades de lectura. Si ens diuen que la proporció d'escriptures és p_e:

$$t_p = (1 - p_e) * (t_{block} + t_h)$$

- En el cas de la l'*escriptura retardada amb assignació*, suposant que la proporció de blocs modificats és p_m:

$$t_p = p_m * (2 * t_{block} + t_h) + (1 - p_m) * (t_{block} + t_h)$$

Índex

Càlcul del temps mitjà d'accés a memòria

- t_{ma} = *temps mitjà d'accés a memòria* (Average Memory Access Time o AMAT).
- Temps mitjà d'una referència a memòria:

$$t_{ma} = t_h + m * t_p$$

- En funció de la política d'escriptura, el *model de temps* ens indicarà de quina manera haurem de calcular t_p.

Càlcul del temps mitjà d'accés a memòria

- De vegades, treballarem amb dues cache diferents, una per les instruccions i una per les dades.
- Valors diferents per als diferents paràmetres per a cadascuna de les cache (t_h, t_{block}, etc.).
- *número de referències a memòria per instrucció*:

$$nr_{instr} = \frac{nr}{I}$$

- nr_{instr} serà sempre més gran que 1 (com a mínim el fetch).
- Proporció referències de lectura o escriptura de dades = $nr_{instr} - 1$
- $t_{ma} = \frac{t_{ma:fetch} + (nr_{instr} - 1) * t_{ma:dades}}{nr_{instr}}$

Càlcul del temps d'execució en funció de t_p i CPI_{ideal}

- Com ja sabem, el nombre de cicles d'execució es calcula:

$$n_{cicles} = n_{ins} \cdot CPI$$

- Sovint, ens caldrà calcular n_{cicles} quan el CPI varia segons sigui el nombre de fallades de cache.
- Cada fallada es tarden t_p cicles addicionals.

Càlcul del temps d'execució en funció de t_p i CPI_{ideal}

- n_{cicles_ideal} el que tarda una execució ideal (sense fallades de memòria cache).
- $$CPI_{ideal} = \frac{n_{cicles\_ideal}}{n_{ins}}$$

Càlcul del temps d'execució en funció de t_p i CPI_{ideal}

$$t_{exe} = n_{ins} \cdot CPI \cdot t_c = (n_{ins} \cdot CPI_{ideal} + n_{fallades} \cdot t_p) \cdot t_c$$

Índex

Índex

Associativitat total o per conjunts

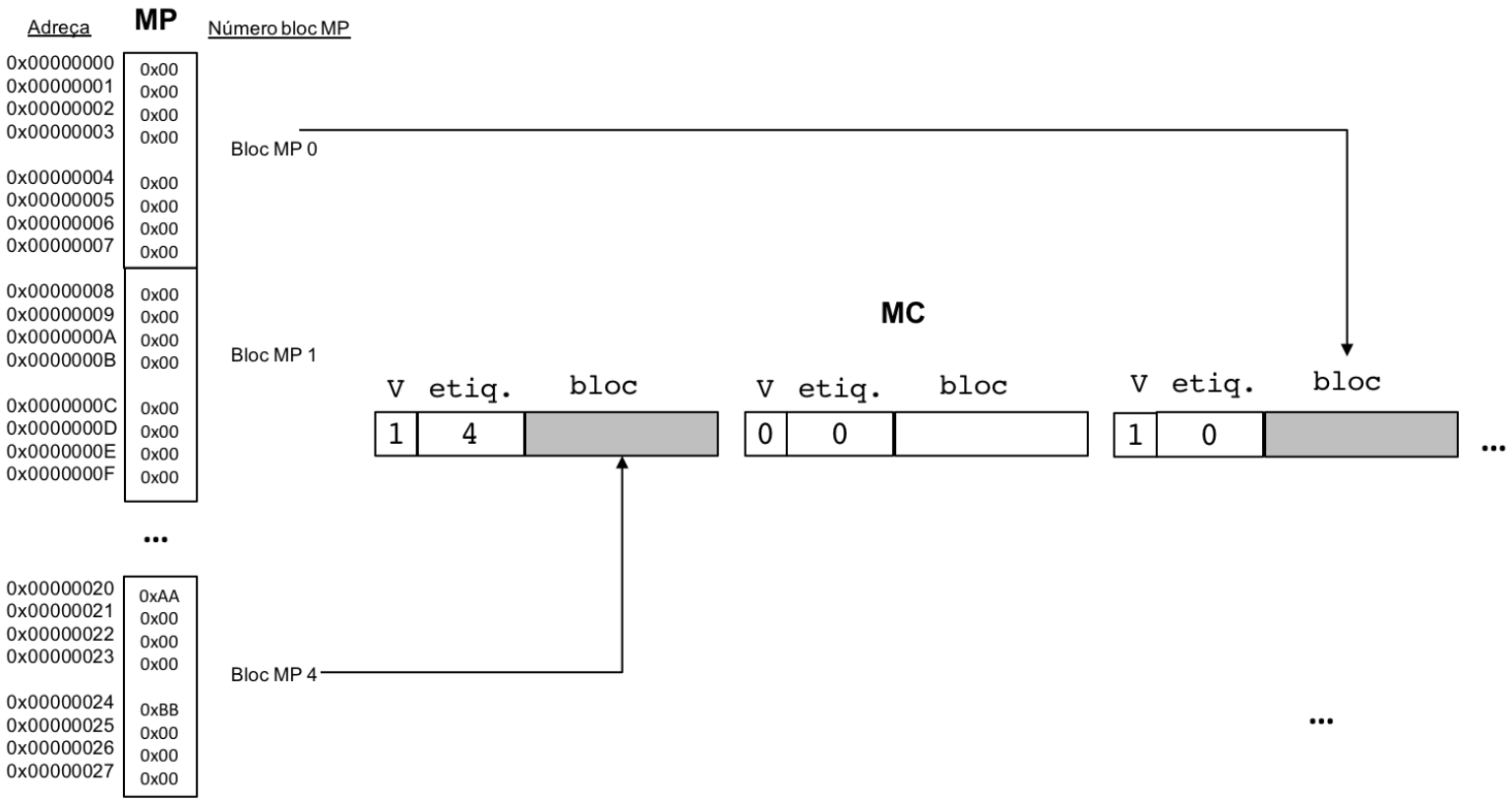
- Fins ara hem ubicat els blocs d'MP dins la memòria cache mitjançant la tècnica de *correspondència directa*.
- Bon rendiment però no té en compte el nivell d'ocupació de la resta de línies de l'MC.
- Podríem tenir tota la cache buida tret d'una línia i veure'ns obligats a reemplaçar-la si els bits de l'adreça així ho indiquen.

Associativitat total o per conjunts

MC completament associativa:

- Ubiquem els blocs d'MP a qualsevol posició de la memòria cache que estigui lliure.
- Això maximitza l'aprofitament de l'espai, reduint així la taxa de fallades.
- Localitzar un bloc serà molt més costós i lent.
- Ens veurem obligats a inspeccionar totes les entrades una per una.

Associativitat total o per conjunts



Exemple de memòria cache completament associativa

Associativitat total o per conjunts

MC associativa per conjunts:

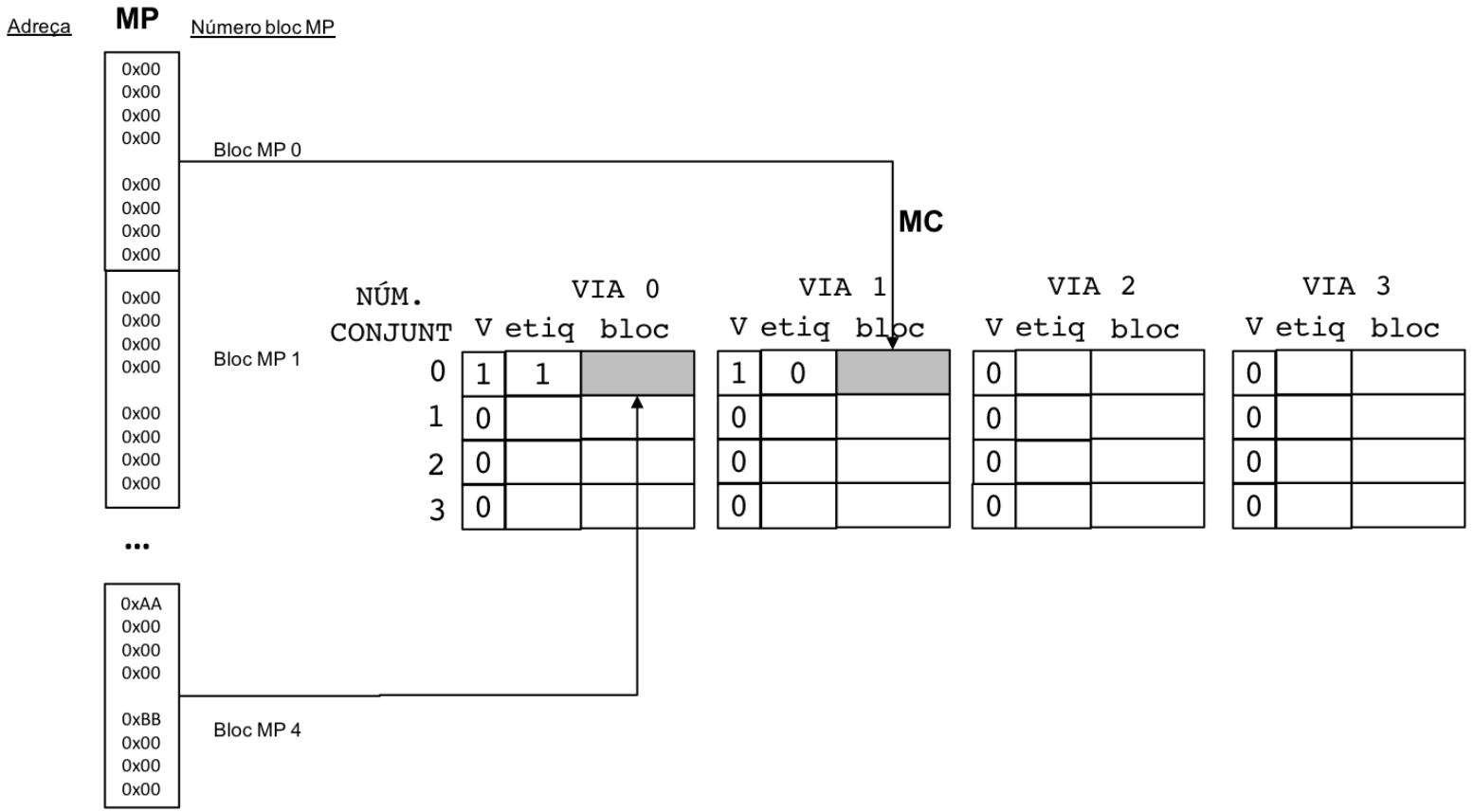
- Solució intermèdia.
- Número determinat d'entrades, però no s'anomenen "línies" sinó *conjunts*.
- El número de bloc MP (que trobem a l'adreça) determinarà a quin *conjunt* ha d'anar el bloc (els c bits del número de bloc MP si hi ha 2^c conjunts).
- No obstant, dins de cada conjunt no només hi haurà espai per a un bloc, sinó que hi podrem encabir n blocs.

Associativitat total o per conjunts

MC associativa per conjunts:

- Cadascun dels blocs que admetrà un conjunt l'anomenarem *via*.
- Si tenim *n* vies parlarem d'una cache *associativa d'n vies* (*n-way associative*).
- La ubicació d'un bloc a una via no dependrà de l'adreça, sinó de la disponibilitat (després veurem que es fa si no n'hi ha cap).
- És a dir, el conjunt el determinarà l'adreça, com en correspondència directa, però la via dins el conjunt es decidirà en funció de la disponibilitat, com en el mètode completament associatiu.

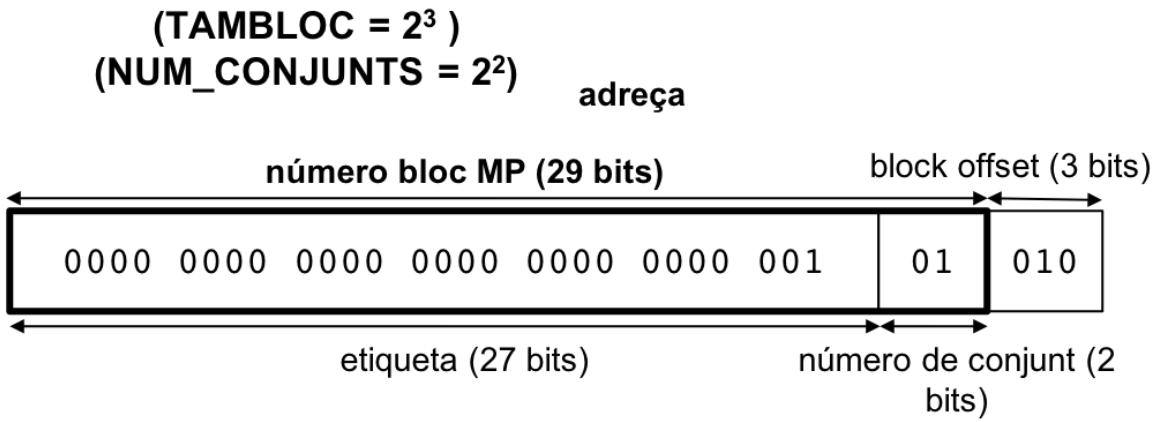
Associativitat total o per conjunts



Exemple de memòria cache associativa de 4 vies i 4 conjunts (blocs de dos paraules).

Associativitat total o per conjunts

L'anàlisi de l'adreça igual que en correspondència directa però tenint en compte només una via:



Anàlisi de l'adreça en una memòria cache associativa de 4 vies i 4 conjunts (blocs de dos paraules).

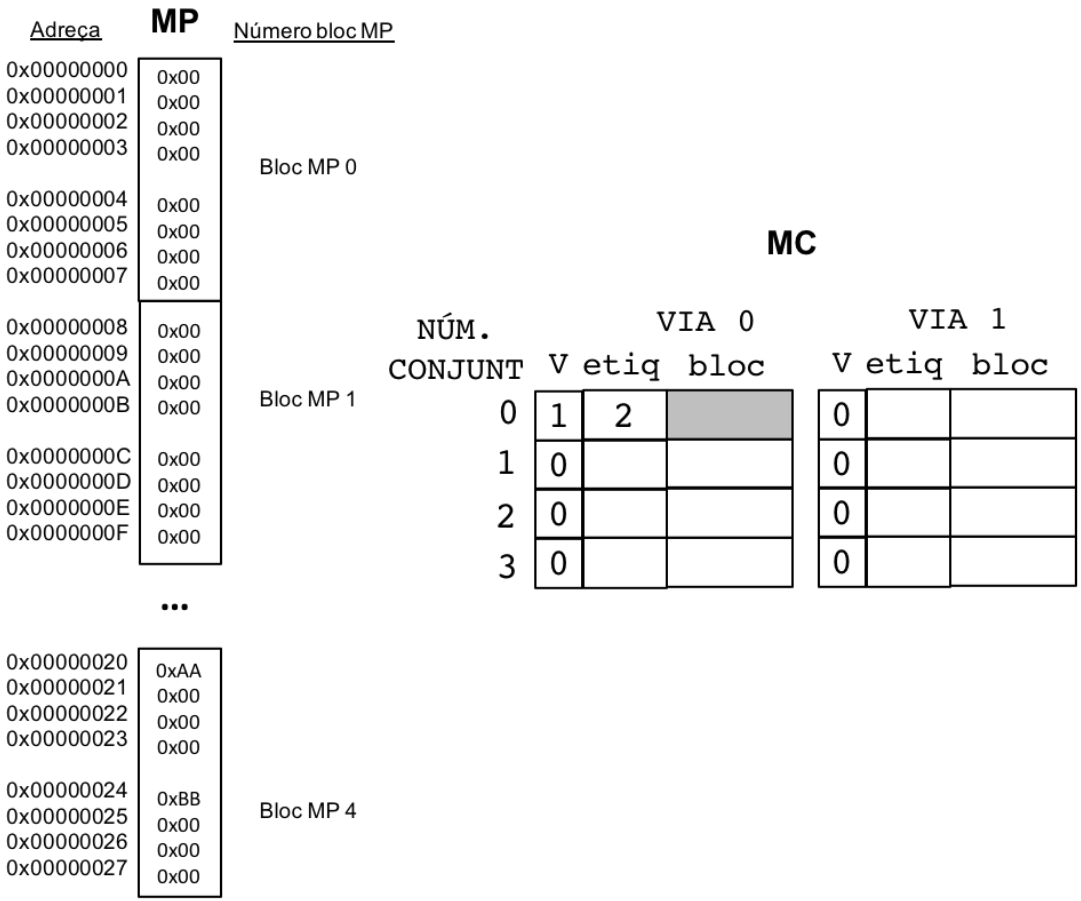
Associativitat total o per conjunts

Reemplaçament LRU (Least Recently Used)

Què farem quan no quedi espai lliure dins el conjunt corresponent? L'algorisme més utilitzat és l'anomenat *Least Recently Used* (LRU), consistent en reemplaçar el bloc que faci més temps que no s'utilitza.

Exemple associativitat per conjunts

Cache associativa per conjuts amb 4 conjunts, 2 vies, i blocs de 2 paraules.



Exemple associativitat per conjunts

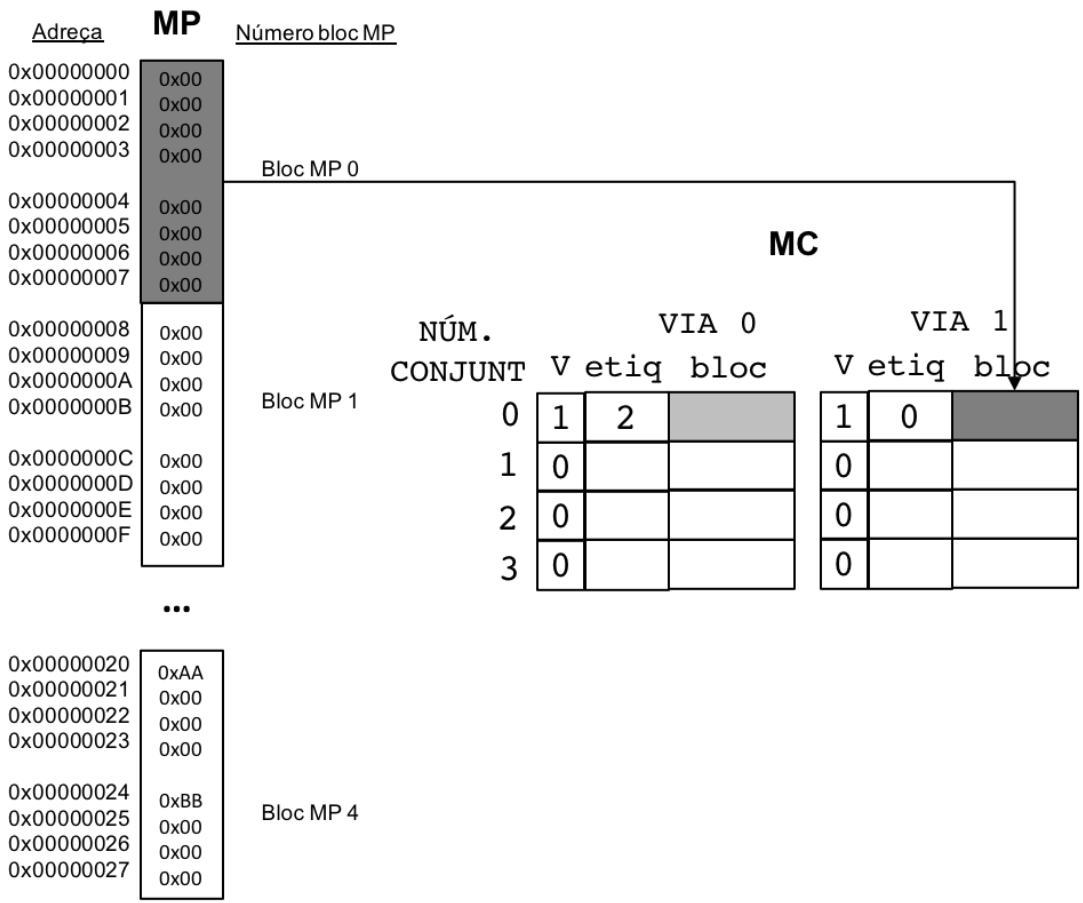
A continuació s'executa el següent codi:

```
li $t0, 0
lw $t1, 0($t0)
```

L'adreça 0 correspon a la primera paraula del bloc 0 d'MP, que alhora correspon al conjunt 0 de l'MC.

Exemple associativitat per conjunts

Donat que el conjut 0 encara té una via lliure (la 1), el bloc serà ubicat en aquesta via.



Exemple associativitat per conjunts

A continuació s'executa el següent codi:

```
li $t0, 32
lw $t1, 0($t0)
```

L'adreça 32 correspon a la primera paraula del bloc 4 d'MP, que també correspon al conjunt 0 de l'MC.

Exemple associativitat per conjunts

No hi ha cap via lliure i hem de decidir quin bloc reemplaçar. L'algorisme LRU ens diu que reemplacem el bloc que hi ha a la via 0.

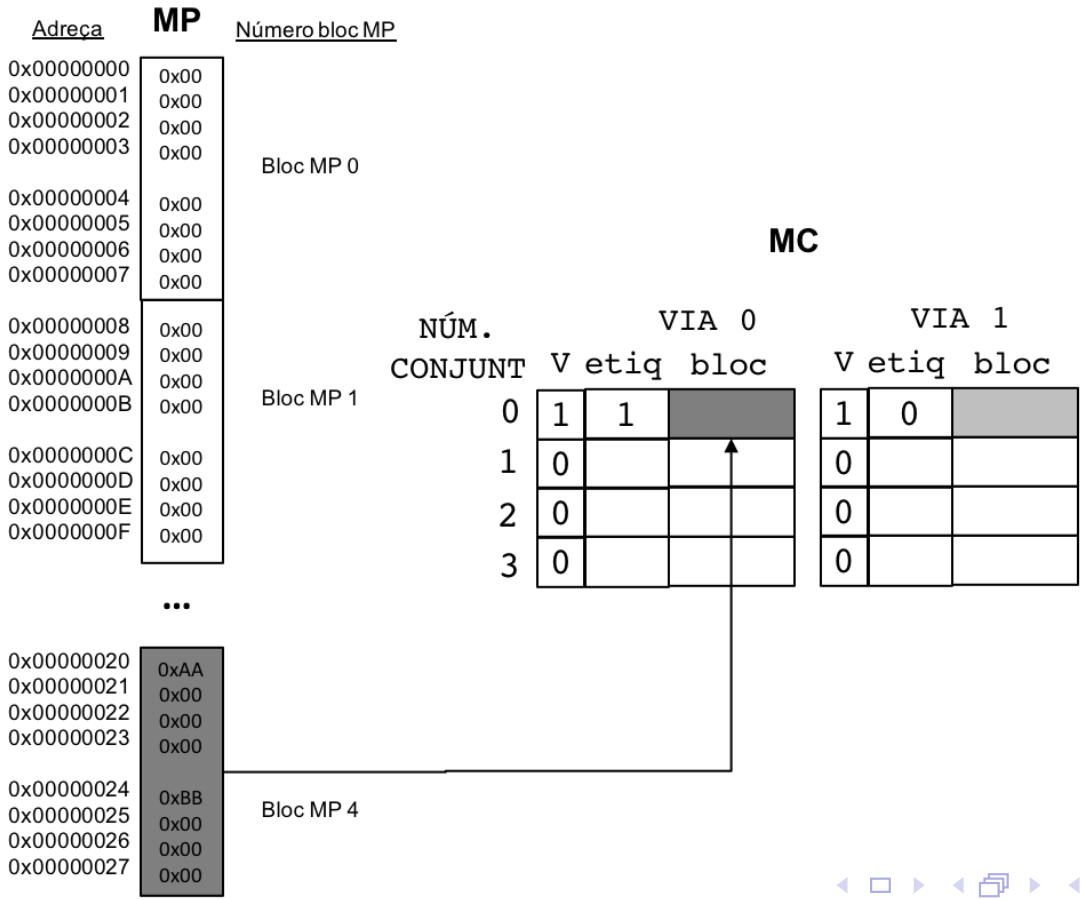


Diagrama de blocs

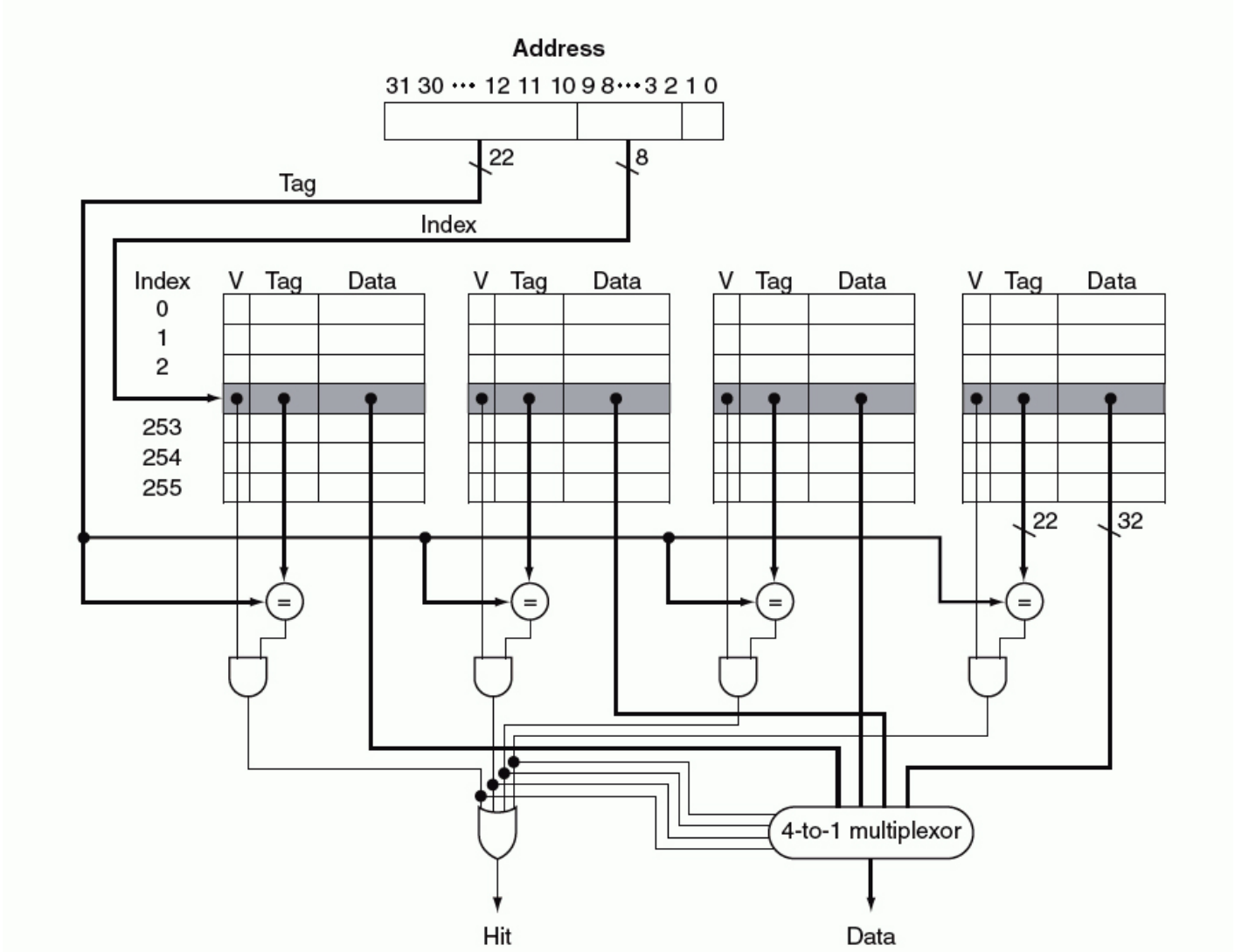
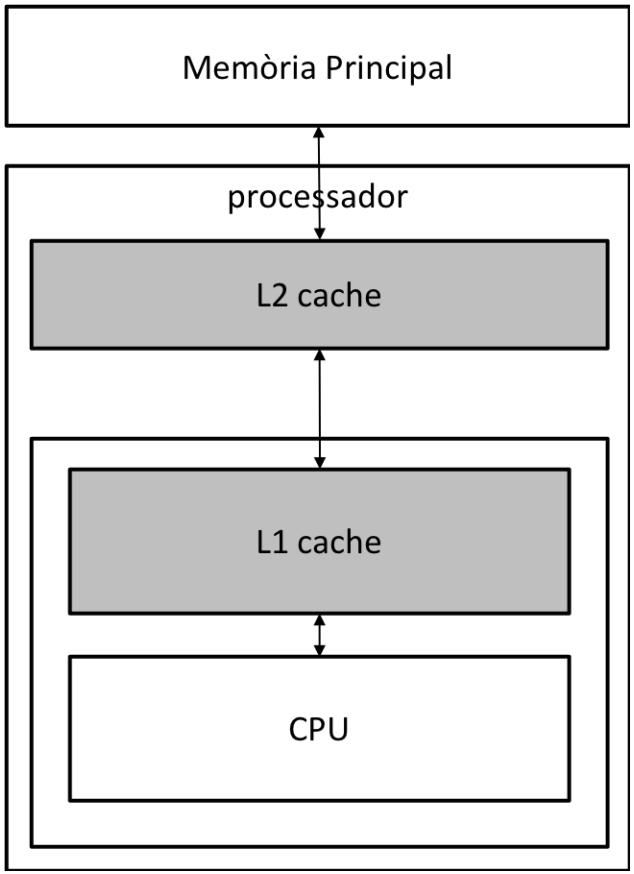


Diagrama de blocs d'una cache associativa per conjunts amb 4 vies

Índex

6.4.2 Cache multinivell



Cache multinivell.

6.4.2 Cache multinivell

- Quin avantatge té això?
- Ens permetrà orientar el disseny de cadascuna de les caches a un objectiu diferent.
- El disseny de la cache de nivell 1 l'orientarem a reduir el temps d'encert :
  - Un nivell baix d'associativitat.
  - Blocs de poques paraules.
  - Poca capacitat.

6.4.2 Cache multinivell

- Aquestes decisions de disseny al nivell 1, per contra, incrementaran molt la taxa de fallades.
- Serien inviables si cada fallada impliqués un accés a la memòria principal.
- Orientarem el disseny del nivell 2 precisament a minimitzar la taxa de fallades:
  - Un nivell alt d'associativitat.
  - Blocs grans.
  - Molta capacitat.

6.4.2 Cache multinivell

- És freqüent tenir una cache de nivell 1 per les instruccions i una per les dades.
- En un processador multi-nucli (multi-core), cada nucli (core) el seu nivell 1. El nivell 2 acostuma a ser compartit.