



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament d'Arquitectura de Computadors

Tema 6

Memòria Cache

Estructura de Computadors (EC)

2023 - 2024 Q2

Adrià Armejach (adria.armejach@upc.edu)



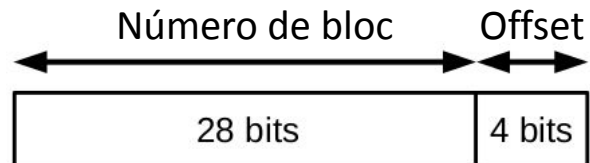


Resum



Resum

- L'espai d'adreçament de memòria es divideix en blocs de N bytes
- Una línia de cache emmagatzema un bloc sencer
- Una adreça de memòria es divideix en dos parts:
 - Número de bloc
 - Offset: byte accedit dintre del bloc



- Polítiques d'ubicació
 - Correspondència directa
 - Els bits de menor pes del número de bloc indiquen la línia
 - Completament associativa
 - Un bloc pot anar a qualsevol línia
 - Associativa per conjunts
 - Els bits de menor pes del número de bloc indiquen el conjunt

Problemes en correspondència directa

- **Conflictes**

- Tamany de línia = 16 bytes
- Número línies MC = 16

Adreça

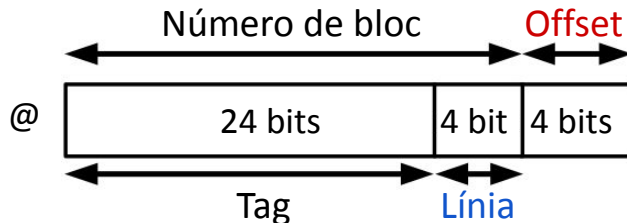
LW 0x10010000 Miss!

LW 0x10010A00 Miss!

LW 0x10010004 Miss!

➔ LW 0x10010A04 Miss!

Correspondència directa



Memòria Cache

num_linia	V	Tag	Dades
0	1	10010A	
1	0		
2	0		
3	0		
4	0		
5	0		
6	0		
7	0		
8	0		
9	0		
A	0		
B	0		
C	0		
D	0		
E	0		
F	0		



Millores: Associativitat i Multinivell

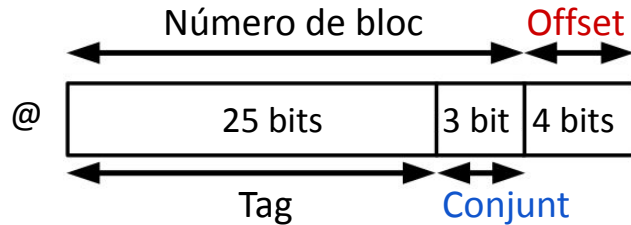


Memòria cache associativa per conjunts

Procesador MIPS de 32 bits

Tamany de línia = 16 bytes

MC: 8 conjunts de 2 línies



Memòria Cache

conjunt	V	Tag	Dades	V	Tag	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		

- Cada bloc de memòria es mapeja a un conjunt fixe
- El bloc pot anar a qualsevol línia dintre del conjunt

Memòria cache associativa per conjunts

Procesador MIPS de 32 bits

Tamany de línia = 16 bytes

MC: 8 conjunts de 2 línies

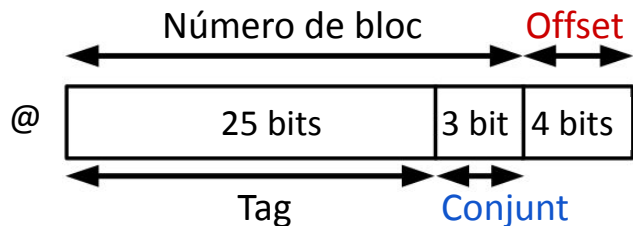
Adreça

LW 0x10010000 Miss!

LW 0x10010A00 Miss!

LW 0x10010004 Hit!

➔ LW 0x10010A04 Hit!



Memòria Cache

conjunt	V	Tag	Dades	V	Tag	Dades
0	1	200200		1	200214	
1	0			0		
2	0			0		
3	0			0		
4	0			0		
5	0			0		
6	0			0		
7	0			0		

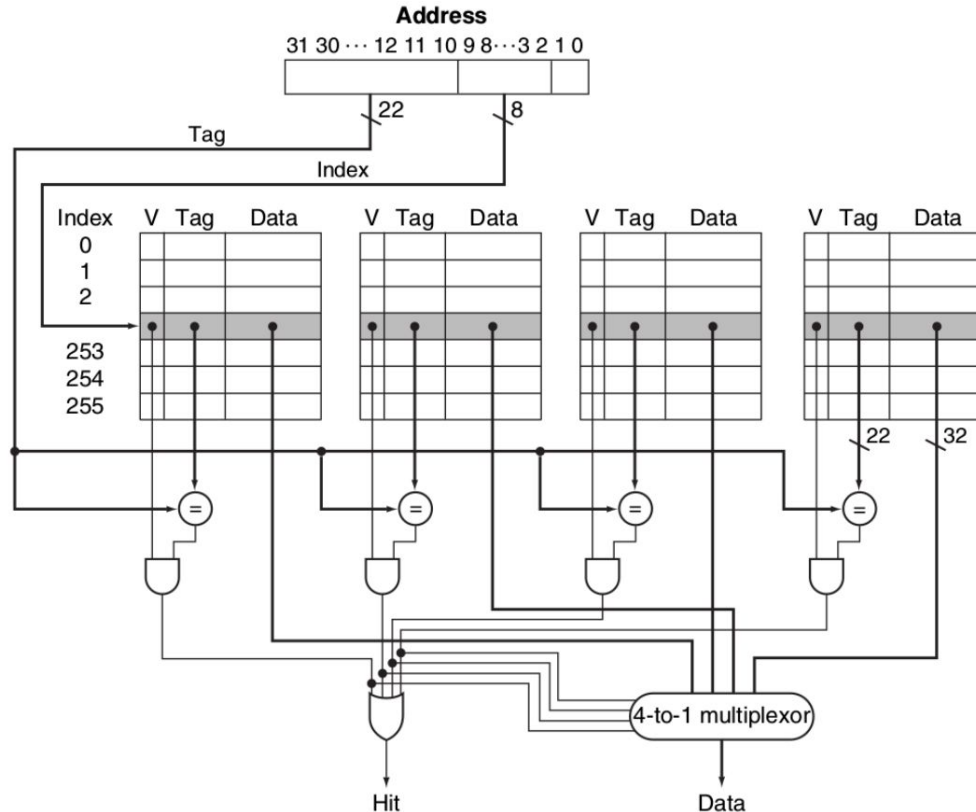
- Cada bloc de memòria es mapeja a un conjunt fixe
- El bloc pot anar a qualsevol línia dintre del conjunt


Algorismes de reemplaçament

- Indiquen la línia que s'ha de reemplaçar de la MC quan no queda espai per colocar un bloc nou
 - Necessari per memòries cache completament associatives i associatives per conjunts
- Per correspondència directa el bloc a reemplaçar és únic, no hi ha alternativa
- **LRU** (Least Recently Used)
 - Es reemplaça la línia que fa més temps que no s'utilitza

Diagrama de blocs: Associativa per conjunts

- Tamany de bloc: 4 bytes
- 256 conjunts
- Associativitat 4 línies





Classificació fallades: Les tres “C”



Classificació de les fallades de memòria cache

- **Cold/Compulsory Misses** (Arrancada en fred)
 - Fallada degut a que és la primera vegada que s'accedeix al bloc de memòria
- **Conflict Misses** (Conflictes)
 - Fallades que es deuen a la falta d'associativitat
 - Apareixen al accedir contínuament a blocs que es mapejen a la mateixa línia o conjunt de la cache
 - Es poden evitar augmentant l'associativitat
- **Capacity Misses** (Capacitat)
 - Fallades degudes a falta de capacitat

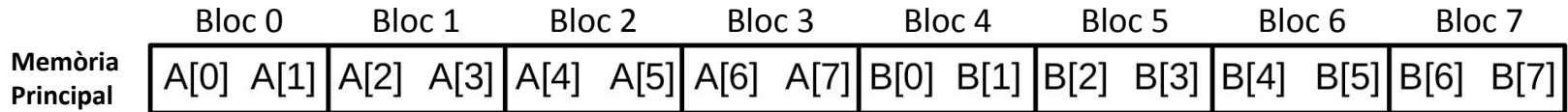
Exemple: Conflict misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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



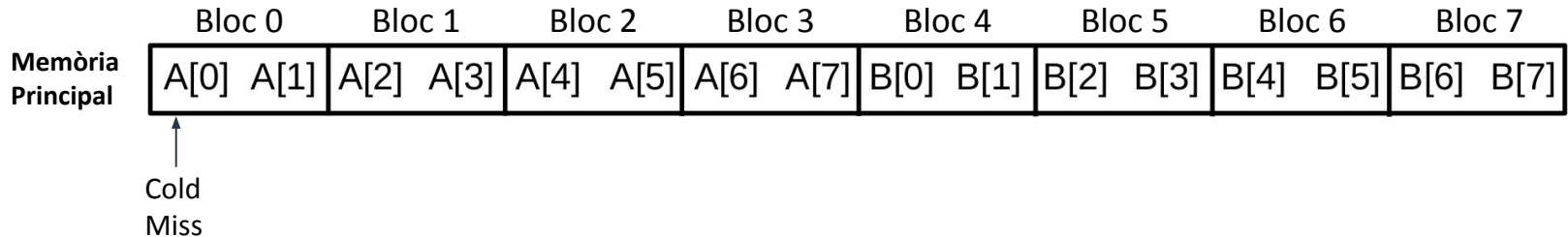
Exemple: Conflict misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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



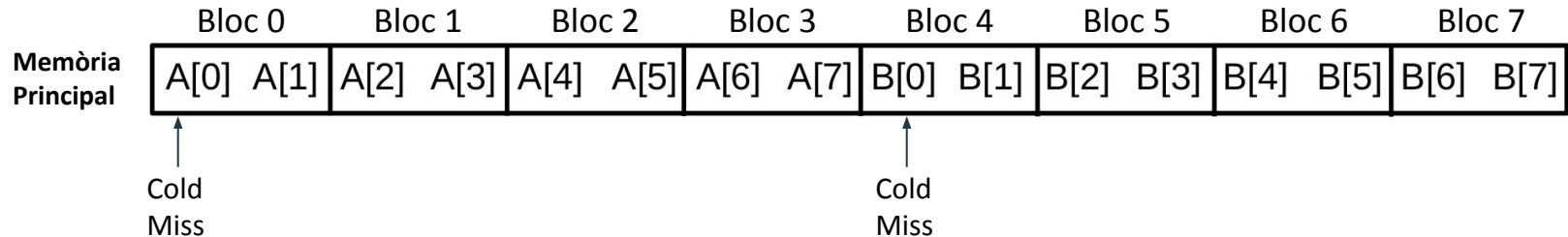
Exemple: Conflict misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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



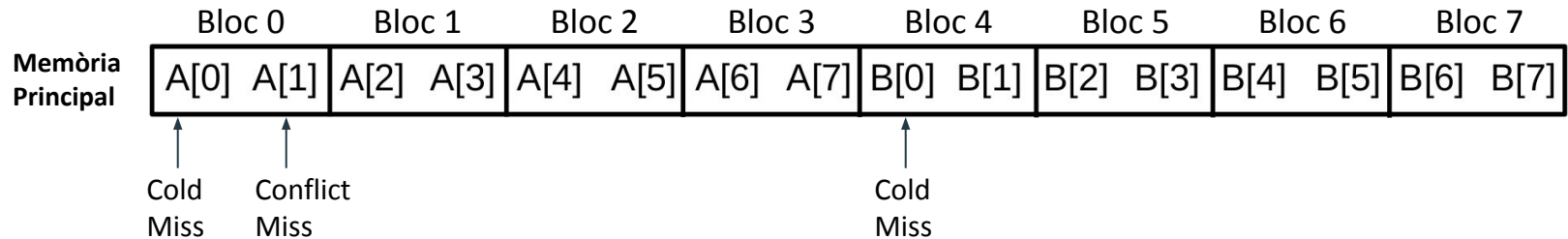
Exemple: Conflict misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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



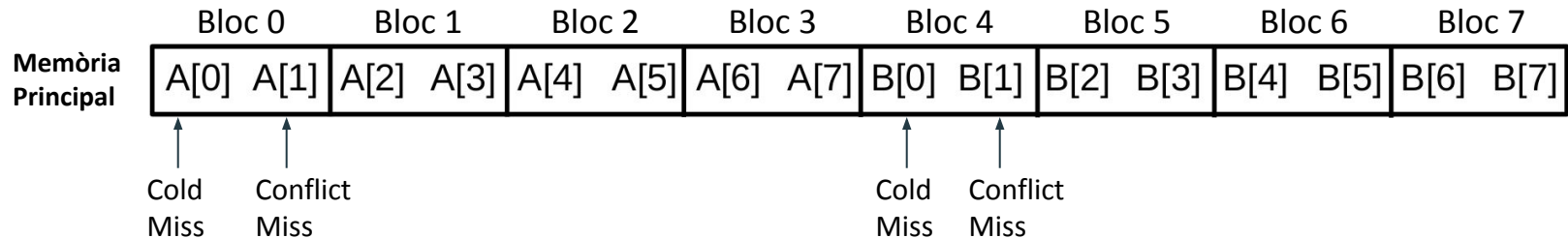
Exemple: Conflict misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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



Exemple: Capacity misses

Processador de 32 bits

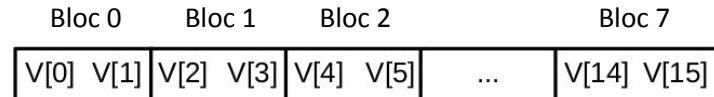
Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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

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

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



Exemple: Capacity misses

Processador de 32 bits

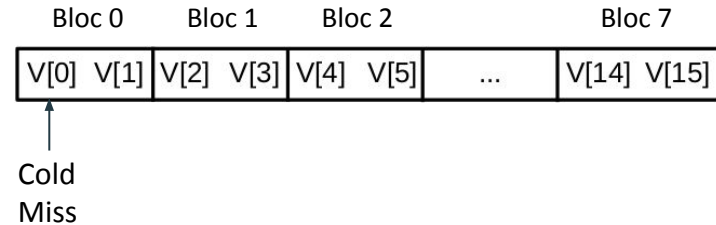
Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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

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

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



Exemple: Capacity misses

Processador de 32 bits

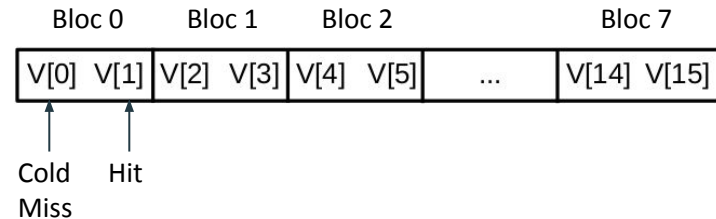
Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

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

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

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



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

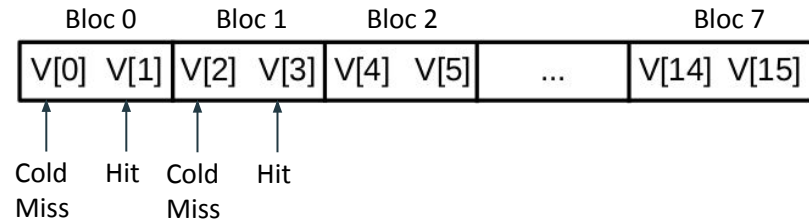
```
int i, s = 0;
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i];
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i] * 3;
```



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

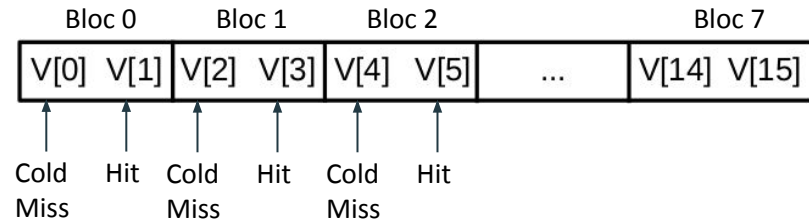
```
int i, s = 0;
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i];
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i] * 3;
```



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

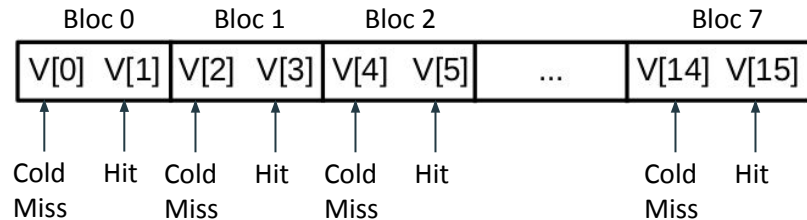
MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

```
int i, s = 0;
```

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

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



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

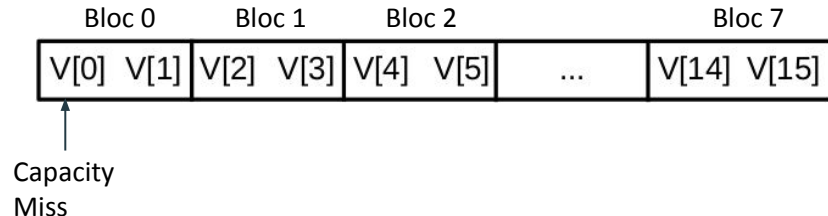
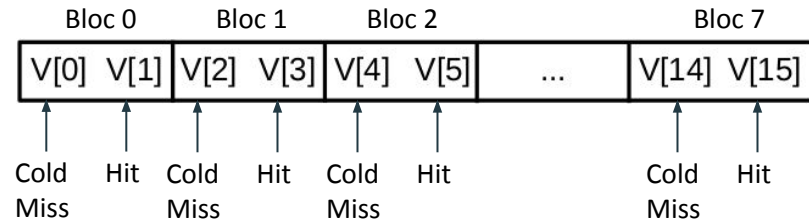
```
int i, s = 0;
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i];
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i] * 3;
```



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

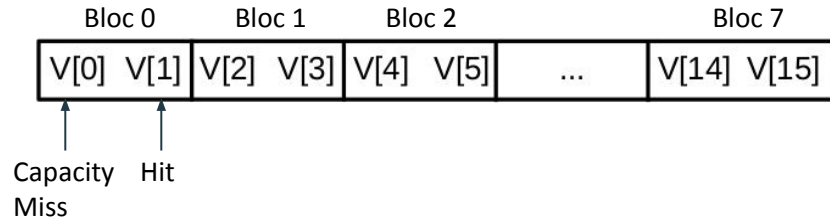
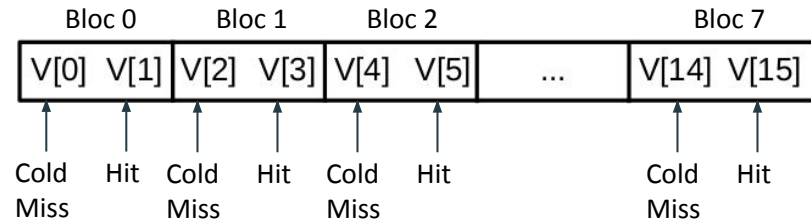
MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

```
int i, s = 0;
```

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

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



Exemple: Capacity misses

Processador de 32 bits

Tamany de bloc = 8 bytes

MC de 4 línies amb correspondència directa (32 bytes de capacitat)

```
int V[16];
```

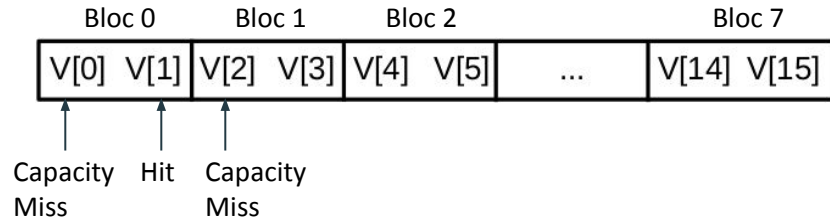
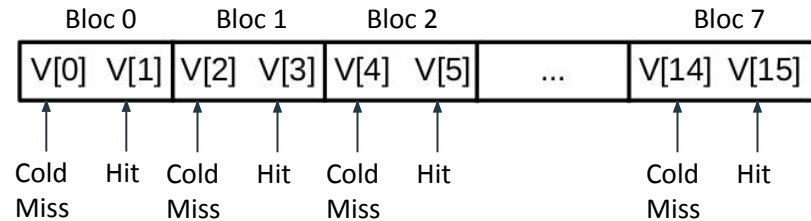
```
int i, s = 0;
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i];
```

```
for (i = 0; i < 16; i++)
```

```
    s += V[i] * 3;
```





Mesures de rendement



Model de temps

- El temps de servir un accés (referència) a memòria inclou:
 - t_h : Comprovar etiquetes (hit/miss) i servir la referència
 - Temps d'encert
 - t_p : Penalització deguda al accés a memòria principal en cas de fallada
 - Temps de penalització

$$t_{acces} = t_h + t_p$$

- En cas d'escriptura immediata assumim que:
 - Tenim un **buffer d'escriptura** de tamany il·limitat amb totes les escriptures pendents de portar a MP
 - El processador pot continuar amb l'execució mentre les escriptures a MP es realitzen

Temps mitjà d'accés a memòria

- Taxa d'encerts

$$h = \frac{\text{num_encerts}}{\text{num_referencies}}$$

- Taxa de fallades

$$m = \frac{\text{num_fallades}}{\text{num_referencies}} = 1 - h$$

- El temps mitjà d'accés a memòria és una manera de mesurar el rendiment d'una MC
 - Average memory access time (AMAT) - t_{ma}

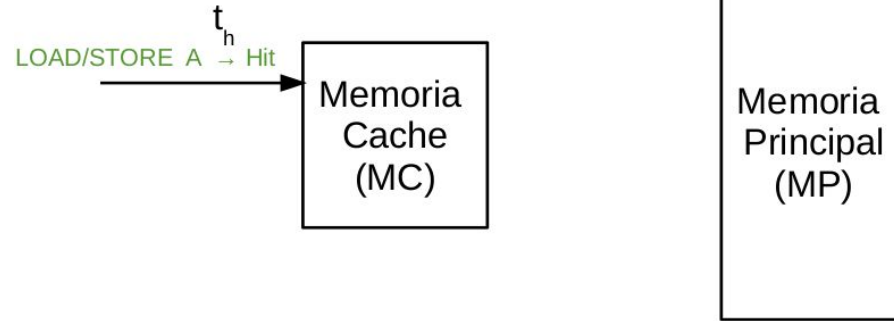
$$t_{ma} = t_h + m * t_p$$
 - on m és la taxa de fallades

Temps d'accés a memòria

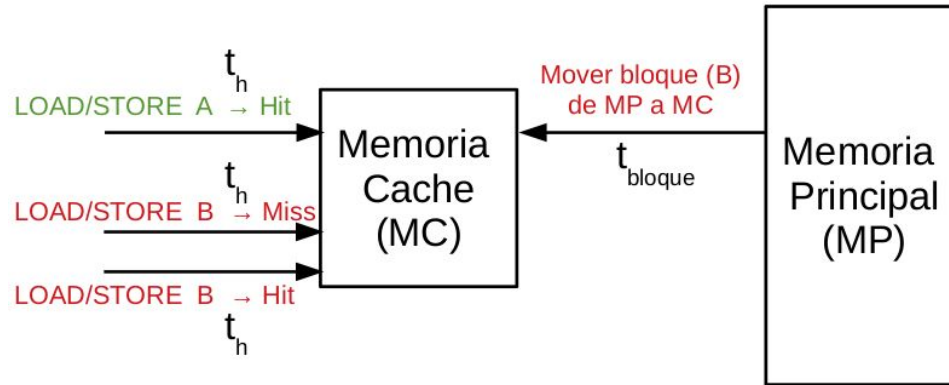
- $t_{\text{acces}} = t_h + t_p$
- t_p depèn de la política d'escriptura
- t_{block} : temps per moure un bloc entre MP i MC

t_p	Immediata amb assignació (laboratori)	Immediata sense assignació	Retardada amb assignació
Lectura - Encert	0	0	0
Lectura - Fallada	$t_{\text{block}} + t_h$	$t_{\text{block}} + t_h$	bloc modif.: $2 * t_{\text{block}} + t_h$ bloc no mod.: $t_{\text{block}} + t_h$
Espectura - Encert	0	0	0
Espectura - Fallada	$t_{\text{block}} + t_h$	0	bloc modif.: $2 * t_{\text{block}} + t_h$ bloc no mod.: $t_{\text{block}} + t_h$

t_{ma} - Escriptura immediata amb assignació

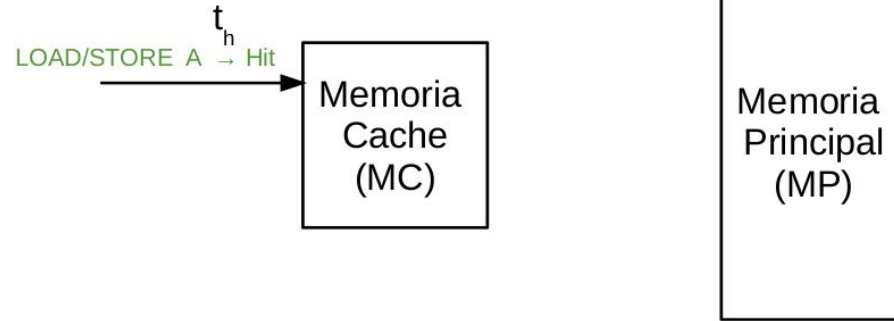


t_{ma} - Escriptura immediata amb assignació

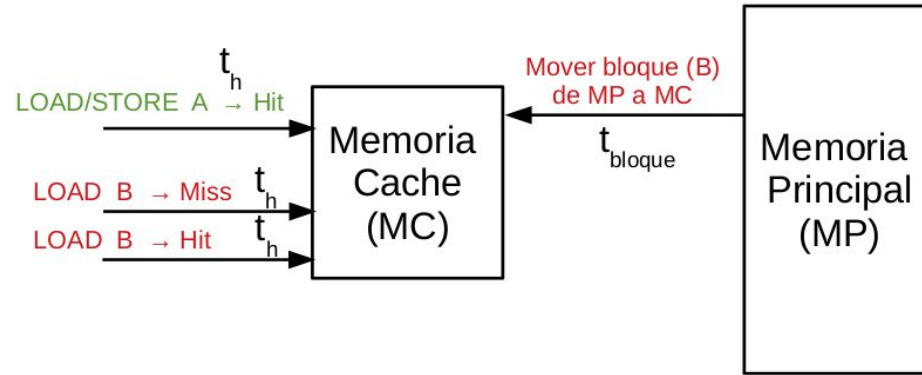


- m : taxa de fallades (miss ratio)
- $t_{ma} = t_h + m \times t_p = t_h + m \times (t_{block} + t_h)$

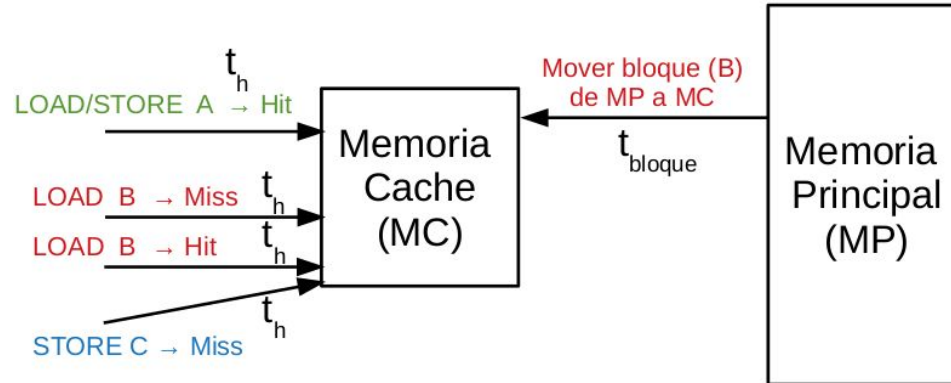
t_{ma} - Escriptura immediata sense assignació



t_{ma} - Escripura immediata sense assignació

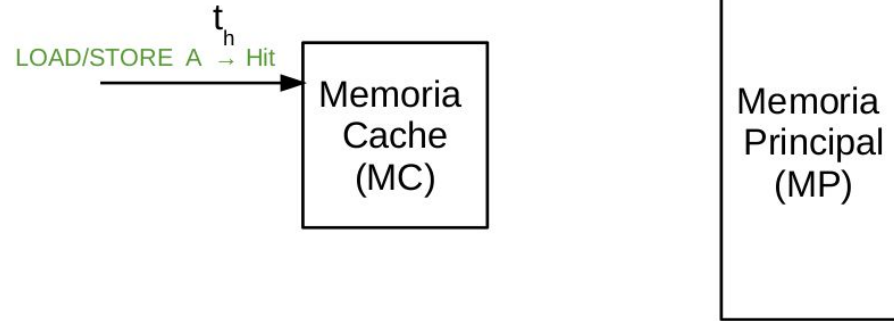


t_{ma} - Escriptura immediata sense assignació

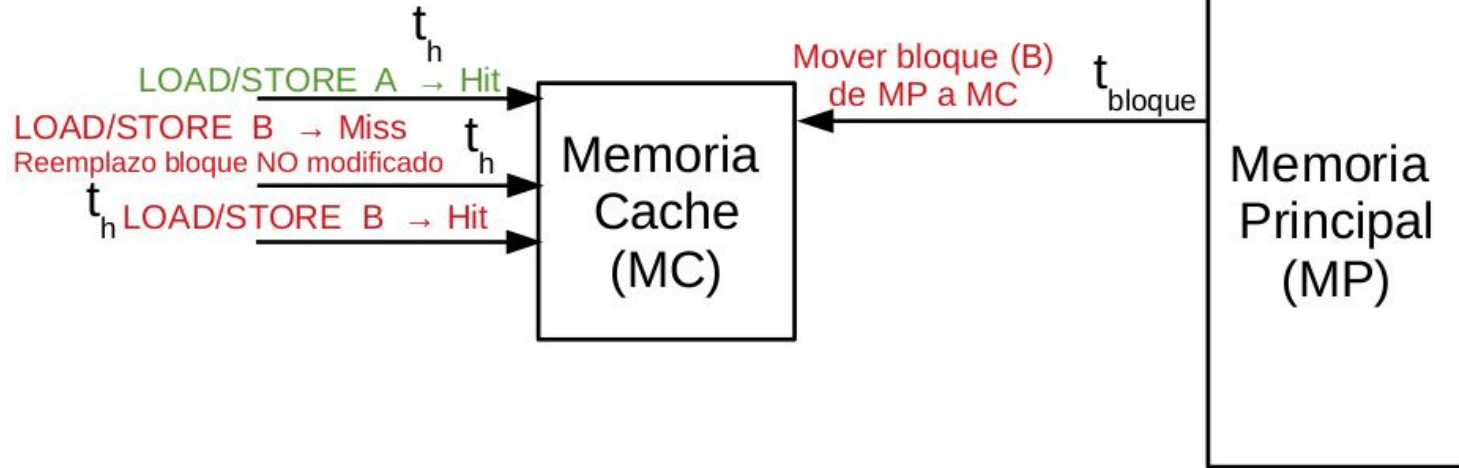


- m : taxa de fallades (miss ratio)
- p_1 : proporció de lectures
- $t_{ma} = t_h + m \times t_p = t_h + m \times p_1 \times (t_{block} + t_h)$

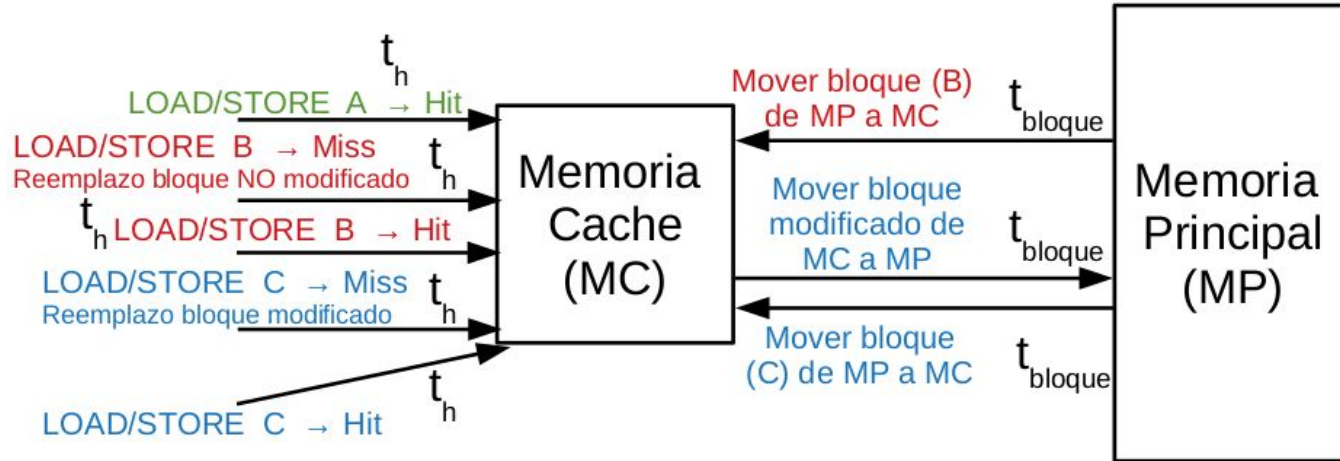
t_{ma} - Escriptura retardada amb assignació



t_{ma} - Escripura retardada amb assignació



t_{ma} - Escripura retardada amb assignació



- m : taxa de fallades (miss ratio)
- p_m : proporció de bloc modificats

$$t_{ma} = t_h + m \times (p_m \times (2 \times t_{block} + t_h) + (1-p_m) \times (t_{block} + t_h))$$

Temps d'accés a memòria

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

Exercici

Es vol definir la política d'escriptura de la memòria cache d'un determinat processador. Es consideren les alternatives: (1) escriptura immediata sense assignació i (2) escriptura retardada amb assignació.

Mitjançant simulació s'han obtingut les següents mesures:

- percentatge d'escriptures (pe): 20%
- percentatge de blocs modificats sobre el total de blocs reemplaçats (pm): 33.33%
- taxa d'encerts cas (1): 0.9
- taxa d'encerts cas (2): 0.85

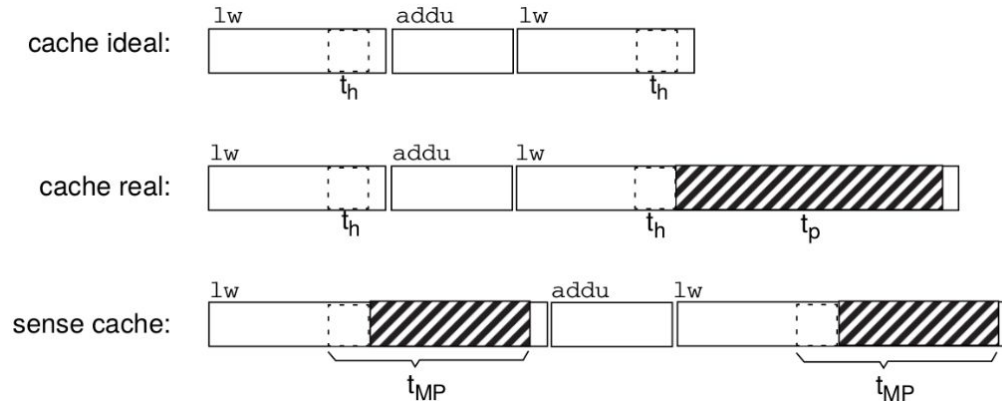
El temps d'accés a memòria cache en cas d'encert (t_h) és de 10 ns. La lectura o escriptura d'un bloc de memòria principal (t_{block}) requereix 100 ns.

Es demana:

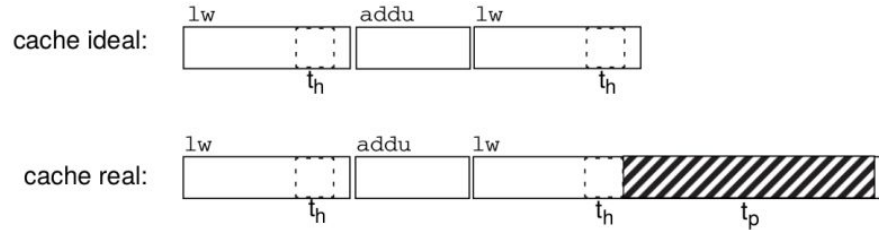
- a)** Calculeu el temps mitjà d'accés a memòria (t_{am}) en ambdues alternatives.

Càlcul del temps d'execució

- Temps d'execució en funció de t_p i CPI_{ideal}
- $n_{cicles} = n_{ins} \times CPI$
 - n_{ins} : número d'instruccions executades
 - CPI: CPI promig
- El CPI varia segons el nombre de fallades de la cache



Càlcul del temps d'execució



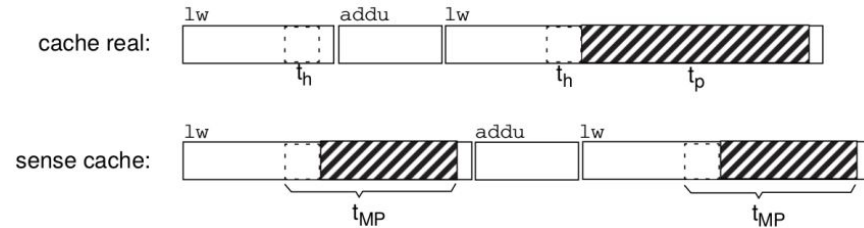
- $n_{\text{cicles_ideal}}$: cicles que triga l'execució ideal

$$CPI_{ideal} = \frac{n_{cicles_ideal}}{n_{ins}}$$

- $CPI_{real} = CPI_{ideal} + (n_{fallades} \cdot t_p) / n_{ins}$

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

Càlcul del temps d'execució

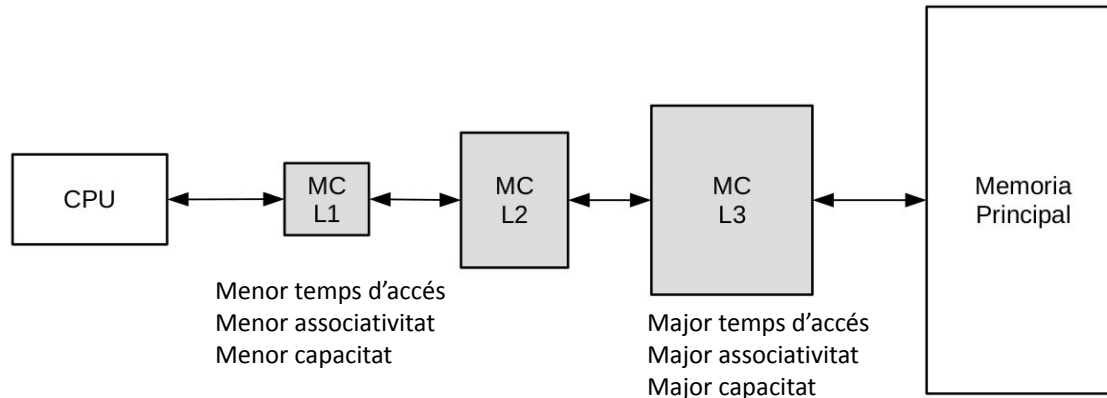


$$t_{exe} = n_{ins} \cdot CPI \cdot t_c = (n_{ins} \cdot CPI_{ideal} + n_{fallades} \cdot t_p) \cdot t_c$$

- La mateixa equació es pot aplicar a un sistema sense memòria cache
 - $n_{fallades}$ és igual al número total de loads
 - El temps de penalització és $t_p = t_{MP} - t_h$
 - t_{MP} : temps d'accés a memòria principal
 - Tots els loads trigen t_{MP} , però el CPI ideal ja inclou t_h

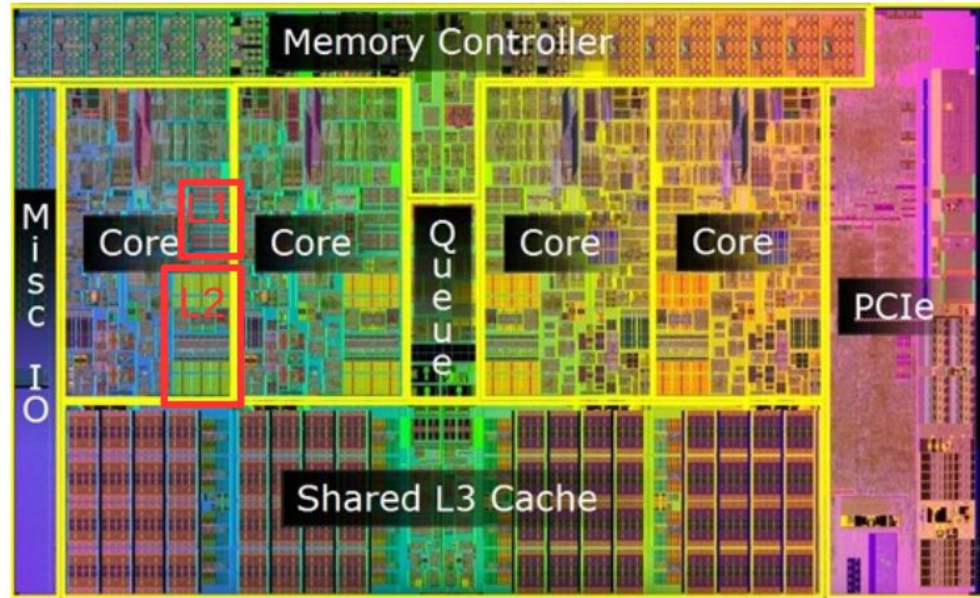
Memòria cache multi-nivell

- Els processadors comercials inclouen una jerarquia de memòria amb múltiples nivells de memòria cache
- Ajuda a reduir el temps d'accés a memòria
 - En cas de fallada a la L1, podem trobar el bloc a la L2, evitant anar a memòria principal



Processador comercial: Intel i5-750 (Nehalem)

- L1 de 32KB (per core)
- L2 de 256KB (per core)
- L3 de 8MB (compartida)



Exercici

Un sistema disposa d'un processador de 20 bits d'adreces, i una memòria cache (MC) de 8 Kbytes amb la següent organització:

- Correspondència associativa per conjunts, de grau 2 (2 blocs per conjunt)
- Blocs de 256 bytes
- Reemplaçament LRU
- Escriptura retardada amb assignació.

Estant la cache inicialment buida, un programa produeix una seqüència de referències a memòria segons s'indica a la següent taula, on apareixen les adreces en hexadecimal i si són lectures o escriptures (L/E). Completa les columnes que falten indicant, per a cada referència: el número de conjunt de MC; si és encert (e) o fallada (f); i el nombre de bytes de Memòria Principal (MP) llegits i/o escrits.

L/E	adreça (hex)	núm. de conjunt	encert (e)/ fallada (f)	bytes de MP	
				llegits	escrits
L	01200				
E	082A0				
L	083F0				
E	01204				
L	04204				
E	083F4				
L	01208				
E	082A0				

Exercici

Suposem que tenim un processador de 32 bits amb una memòria cache de dades de 512 bytes, on cada bloc té 16 bytes. Suposem que executem els següents programes.

```
//programa A

int M[4][128];

void main() {
int i, j; //en registres
  for (i=0;i<4;i++)
    for (j=0;j<128;j++)
      M[i][j]= 0;
}
```

```
//programa B

int M[4][128];

void main() {
int i, j; //en registres
  for (j=0;j<128;j++)
    for (i=0;i<4;i++)
      M[i][j]= M[i][j]+1;
}
```

Calcula el nombre de fallades de la cache suposant que la memòria cache és inicialment buida. L'adreça base de la matriu M és 0.

- a) Suposant que la cache és de correspondència directa i té la política d'escriptura retardada amb assignació.

Fallades A =

Fallades B =

- b) Suposant que la cache és associativa per conjunts de 4 vies (algorisme de reemplaçament LRU), i que té la política d'escriptura immediata sense assignació.

Fallades A =

Fallades B =

Cert o Fals

1. En una memòria cache amb política d'escriptura immediata sense assignació, un accés a la memòria cache pot implicar dos accesos a memòria principal
2. Si en una cache canviem la política d'escriptura immediata amb assignació a retardada amb assignació, sense cap més canvi, el nombre total de fallades no canvia
3. En un processador amb adreces de 32 bits, una cache associativa de 4 vies, de 32KB i blocs de 32 bytes, s'han de dedicar 19 bits a etiqueta (TAG), 8 al número d'entrada (conjunt) i 5 al desplaçament (offset)

Cert o Fals

1. Un sistema que tinguiés una memòria cache més gran que la memòria principal mai tindria fallades
2. En les memòries cache que utilitzen escriptura immediata s'ha de posar el dirty bit a 1 només quan hi ha un encert d'escriptura
3. El temps de penalització en cas de fallada d'escriptura d'una memòria cache d'escriptura immediata sense assignació és zero segons el model estudiat
4. Si s'executa un mateix programa en dues memòries cache diferents, però les dues de correspondència directa, sempre tindrà una major taxa d'encerts la que tingui els blocs de mida més gran.