

Estructura de Computadors

Grau d'Enginyeria Informàtica
ETSINF

Tema 6: Jerarquia de memòria



Curs 2013-2014



Objectius

- Comprendre el concepte de jerarquia de memòria i les raons que la justifiquen
- Conéixer les característiques operacionals i paràmetres de disseny de la memoria cache y comprender cómo estos influyen en las prestaciones del sistema de memoria
- Conocer el concepto de memoria virtual y su soporte por parte de la arquitectura del procesador

Contenido

1. Concepto de jerarquía de memoria
2. La memoria cache
 - ✓ Conceptos básicos
 - ✓ Correspondencias y políticas
 - ✓ Aspectos de rendimiento
3. La memoria virtual
 - ✓ Concepto y utilidad
 - ✓ Direccionamiento virtual
 - ✓ Traducción de direcciones
4. Ejemplo conjunto TLB y cache
 - ✓ El MIPS R2000 en la DECstation 3100

Bibliografia

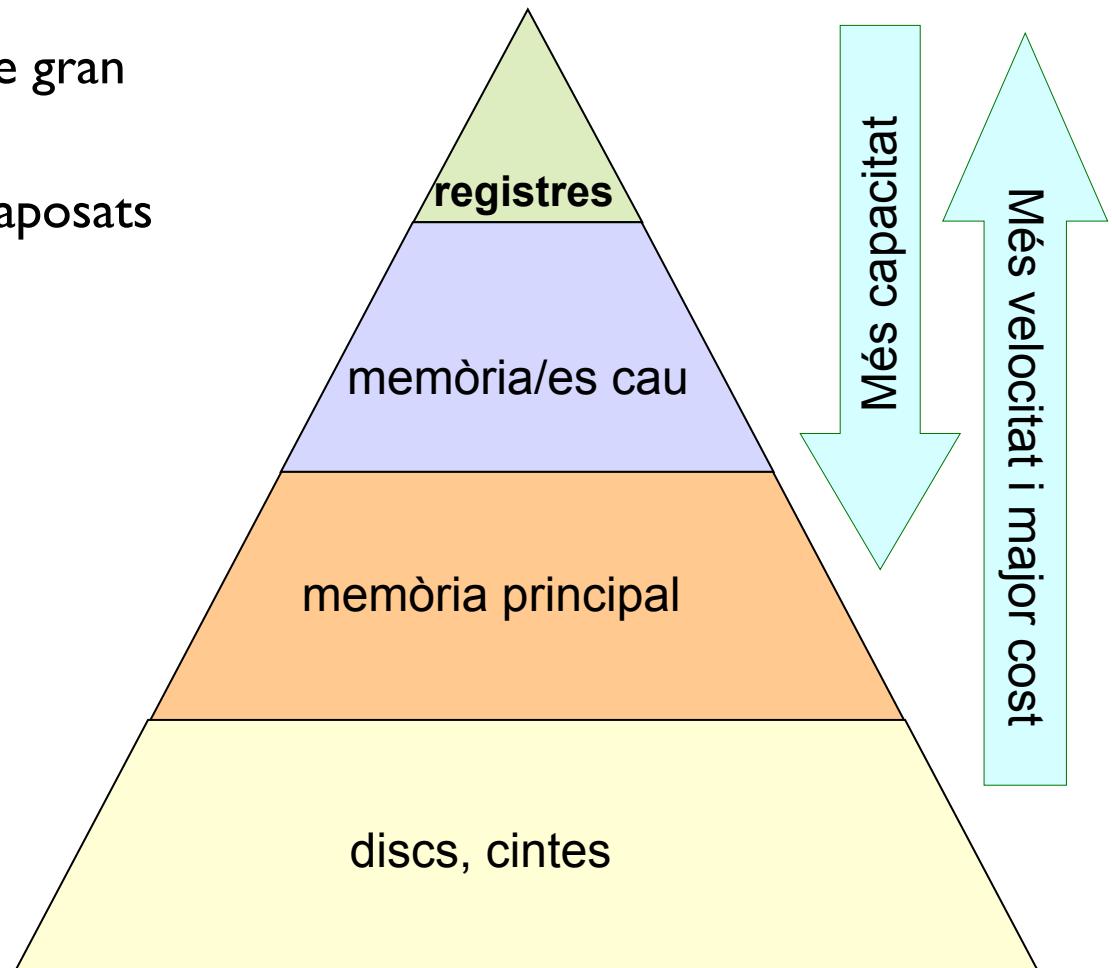
- Patterson, D.A., Hennessy, J.L.
 - ✓ Estructura y diseño de computadores. Interfaz circuitería/programación. Reverté, 2000
 - ✓ Cap.5 (5.1, 5.2, 5.3, 5.4, 5.5)
- Stallings, W.
 - ✓ Organización y arquitectura de computadores. 7^a edición. Prentice Hall, 2006
 - ✓ Cap.4 (4.2)
- Hamacher, V.C., Vranesic, Z.G., Zaky, S.G.
 - ✓ Organización de computadores, 5^a edición. McGraw Hill, 2003
 - ✓ Cap.5 (5.5, 5.6, 5.7)

1. Concepte de jerarquia de memòria

- Què entenem per jerarquia de memòria?
- Paràmetres de la jerarquia
- El principi de localitat

Jerarquía de memoria: ¿qué es?

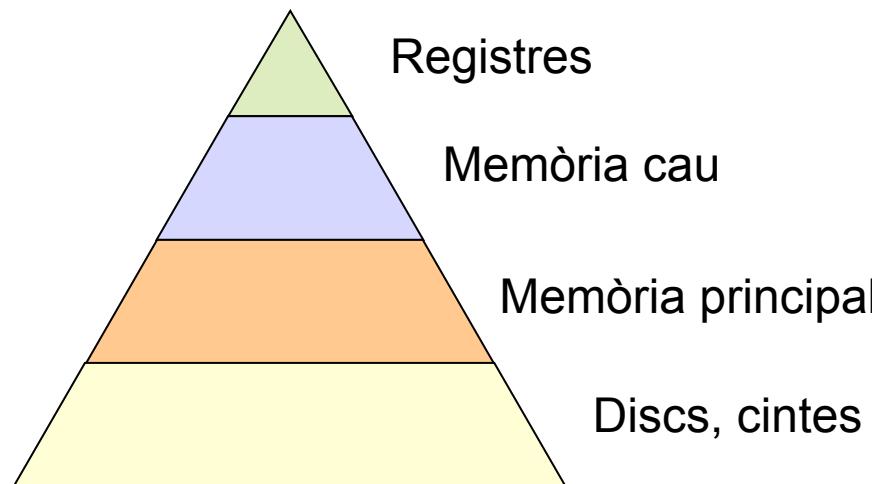
- Situació ideal: disposar de memòria ràpida i barata de gran capacitat
- Criteris tecnològics contraposats
 - ✓ Velocitat d'accés
 - ✓ Capacitat d'emmagatzematge
 - ✓ Cost per bit emmagatzemat
 - ✓ Consum d'energia
 - ✓ Fiabilitat
- Solució
 - ✓ Organització jeràrquica



Paràmetres de la jerarquia de memòria

- Mode d'accés, temps d'accés i capacitat

✓ Registres	Aleatori,	< 0.1 ns,	pocs bytes
✓ Memòria cau	Aleatori,	0.1 – 10 ns,	8 KB – 32 MB
✓ Memòria principal	Aleatori,	10 – 100 ns,	256 MB – 64 GB
✓ Discs	Directe,	10 ms,	100 – 1000 GB
✓ Cintes	Seqüencial,	minuts,	GB – TB



Per què és eficient la jerarquia?

- Principi de localitat de la informació (dades i instruccions)
 - ✓ **Localitat temporal**
 - Tendència a tornar a referenciar la mateixa dada o instrucció
 - ✓ **Localitat espacial**
 - Tendència a referenciar paraules en adreces properes
 - Aquest principi justifica l'organització en blocs o pàgines, que actuen com a unitat de transferència
- És un principi **empíric**, derivat del disseny i comportament dels programes
 - ✓ També es diu “regla 20-80”: el 20% del codi és responsable del 80% del temps d'execució

2. La memòria cau

- Conceptes bàsics
- Correspondències i polítiques
- Conceptes avançats (rendiment)

Rendiment UCP-memòria (performance gap)

- Augment anual del rendiment

- ✓ Processador: 60% per any
- ✓ Memòria (DRAM): 7% per any

- Per tant, la diferència és d'un 50% cad

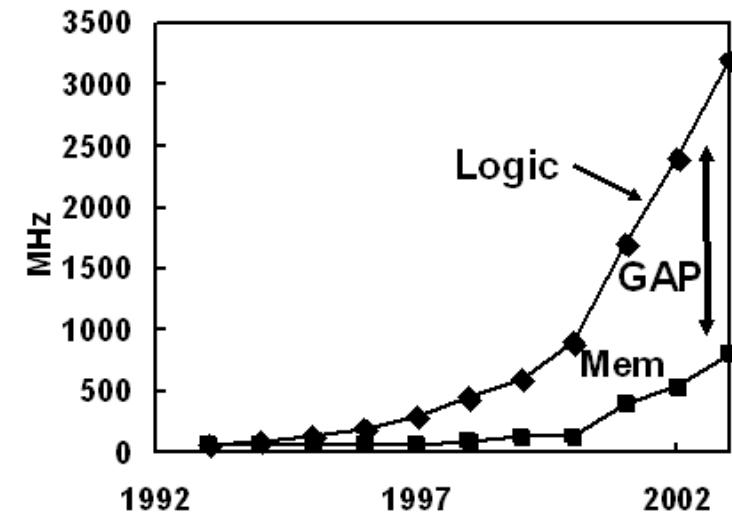
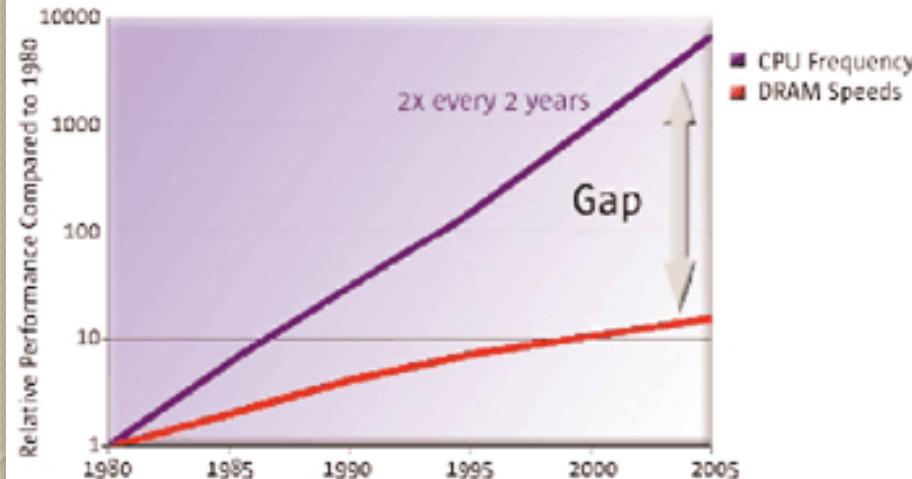
Intel Xeon (fins a 3,8 GHz)

$$T_{clock} = 0,263 \text{ ns}$$

$\approx \times 144$

DDR3-1600

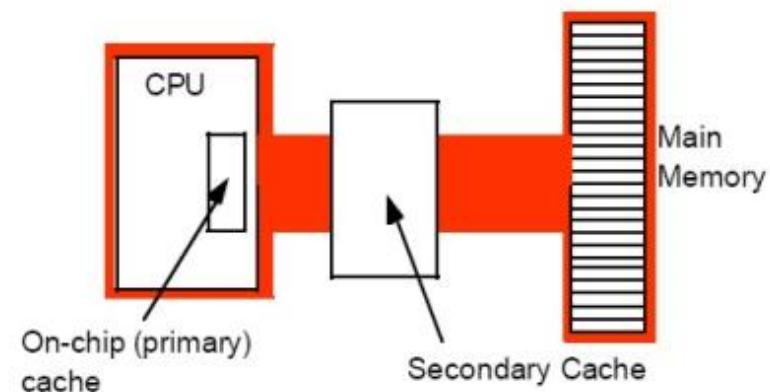
$$\text{Latència } (t_{RCD} + t_{CL}) \approx 37,5 \text{ ns}$$



Com ocultar aquesta diferència?
Establir un o més nivells de memòria cau (SRAM)

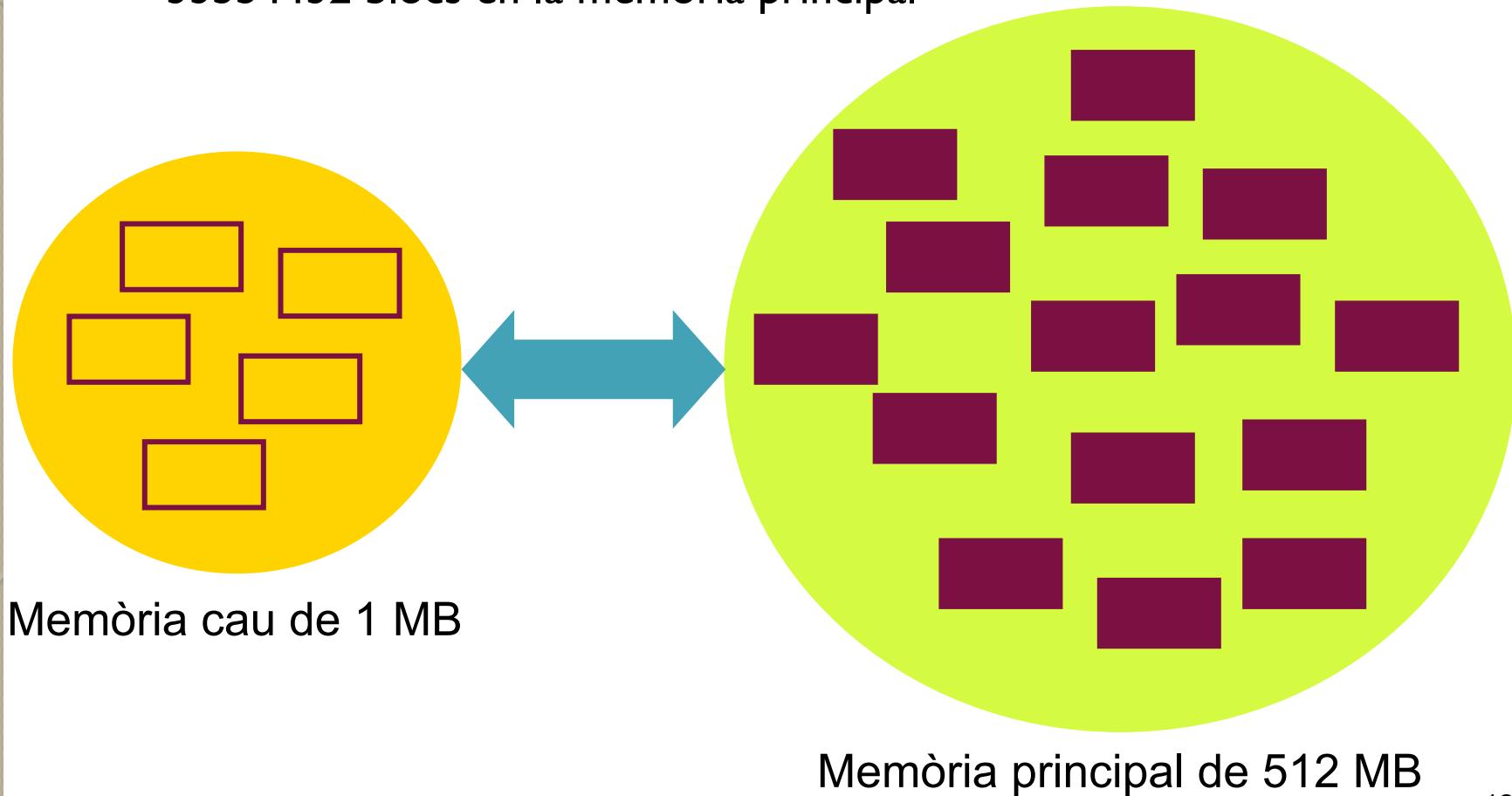
Que és la memòria cau?

- Memòria ràpida i de “poca capacitat” situada entre la memòria principal i el processador
- Objectiu: reduïr el temps d'accés a la informació
 - ✓ Accés al codi (cerca d'instruccions)
 - ✓ Accés a les dades: **lw, lh, lb, sw, sh, sb**
- Funcionament
 - ✓ Primer es busca la informació en la memòria cau (tecnologia SRAM). Si no es troba ací s'accedeix a la memòria principal (tecnologia DRAM)
 - ✓ La unitat de transferència entre la memòria cau i la memòria principal és el **BLOC** (conjunt de 4 o 8 paraules)



Relació entre la memòria principal i la cau

- Blocs de 4 paraules de 32 bits
 - ✓ 65536 línies en la memòria cau
 - ✓ 33554432 blocs en la memòria principal

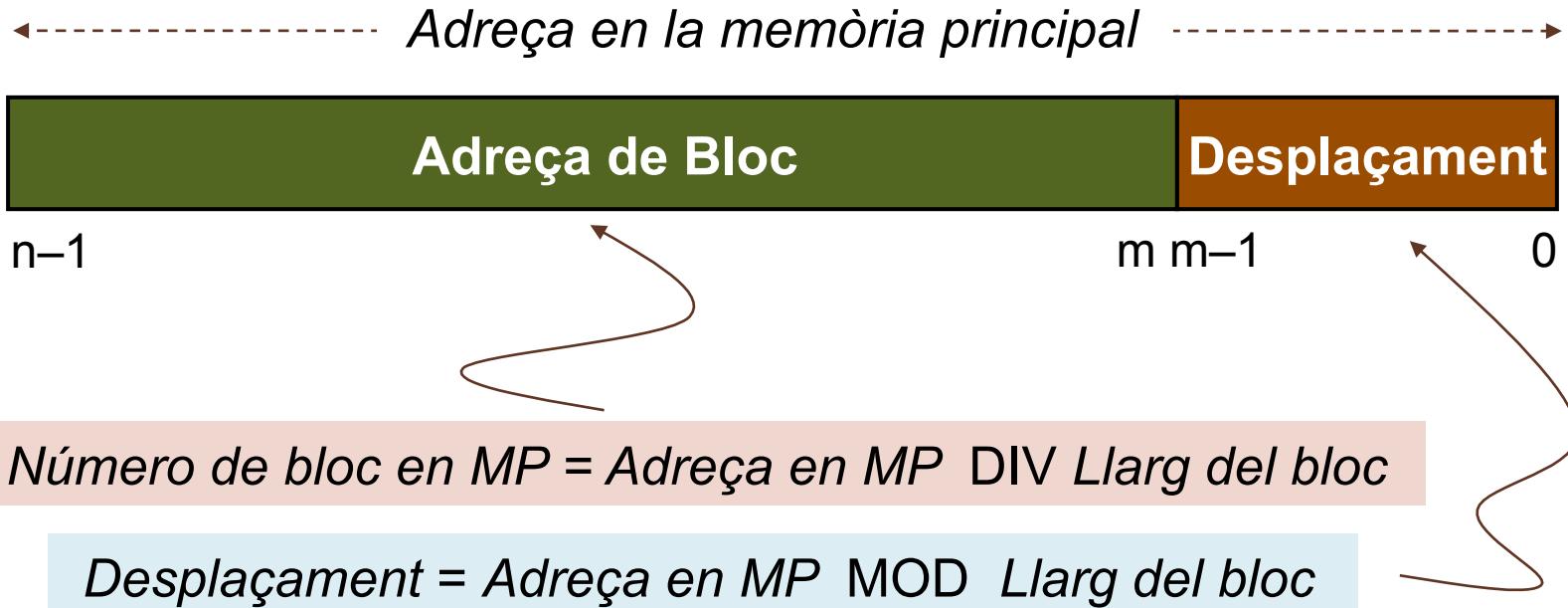


Càcul del bloc

Espai d'adreçament: 2^n bytes

Llarg del bloc: 2^m bytes

$$\text{Nombre total de blocs en MP} = \frac{\text{Espai d'adreçament}}{\text{Llarg de bloc}}$$



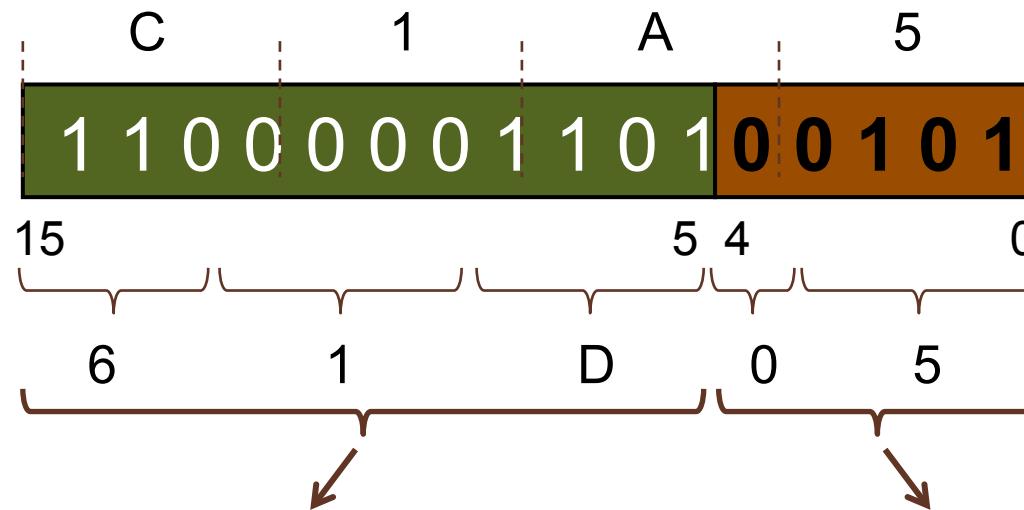
Càcul del número de bloc

- Exemple

Espai d'adreçament de MP: 64 KB

Llarg del bloc: 32 bytes

Adreça en MP: 0xC1A5



Número de bloc en MP = 0x61D

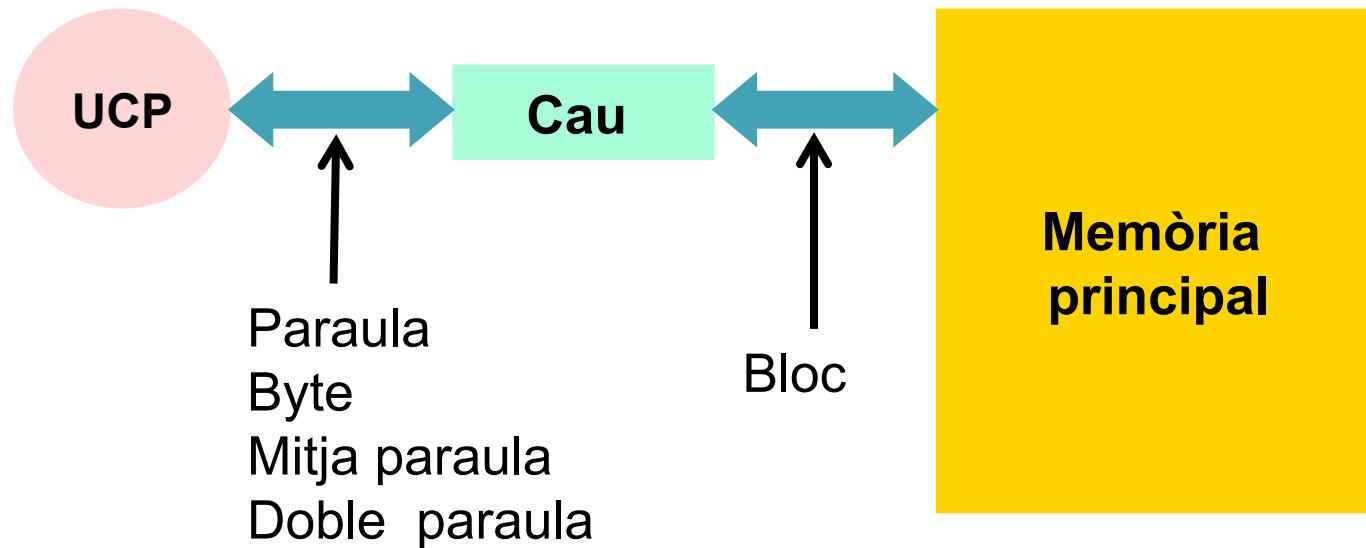
Desplaçament = 0x05

Classificació dels accessos

- En fer un accés, la CPU pot trobar o no la informació a la memòria cau
- **Encert (hit):** la informació es troba en la memòria cau
 - ✓ Temps d'encert: temps d'accés a la memòria cau
- **Fallada (miss):** la informació no es troba en la cau
 - ✓ Cal accedir als nivells inferiors de la jerarquia de memòria
 - ✓ Temps a resoldre la fallada: latència de penalització

Com explota la cau la localitat espacial?

- Quan es produeix una fallada, es porta un bloc (4 o 8 paraules) des de la memòria principal
- Línia: espai de la cau on es copia el bloc procedent de la memòria
 - ✓ La longitud de la línia coincideix amb la del bloc, i s'utilitzen indistintament els termes línia i bloc

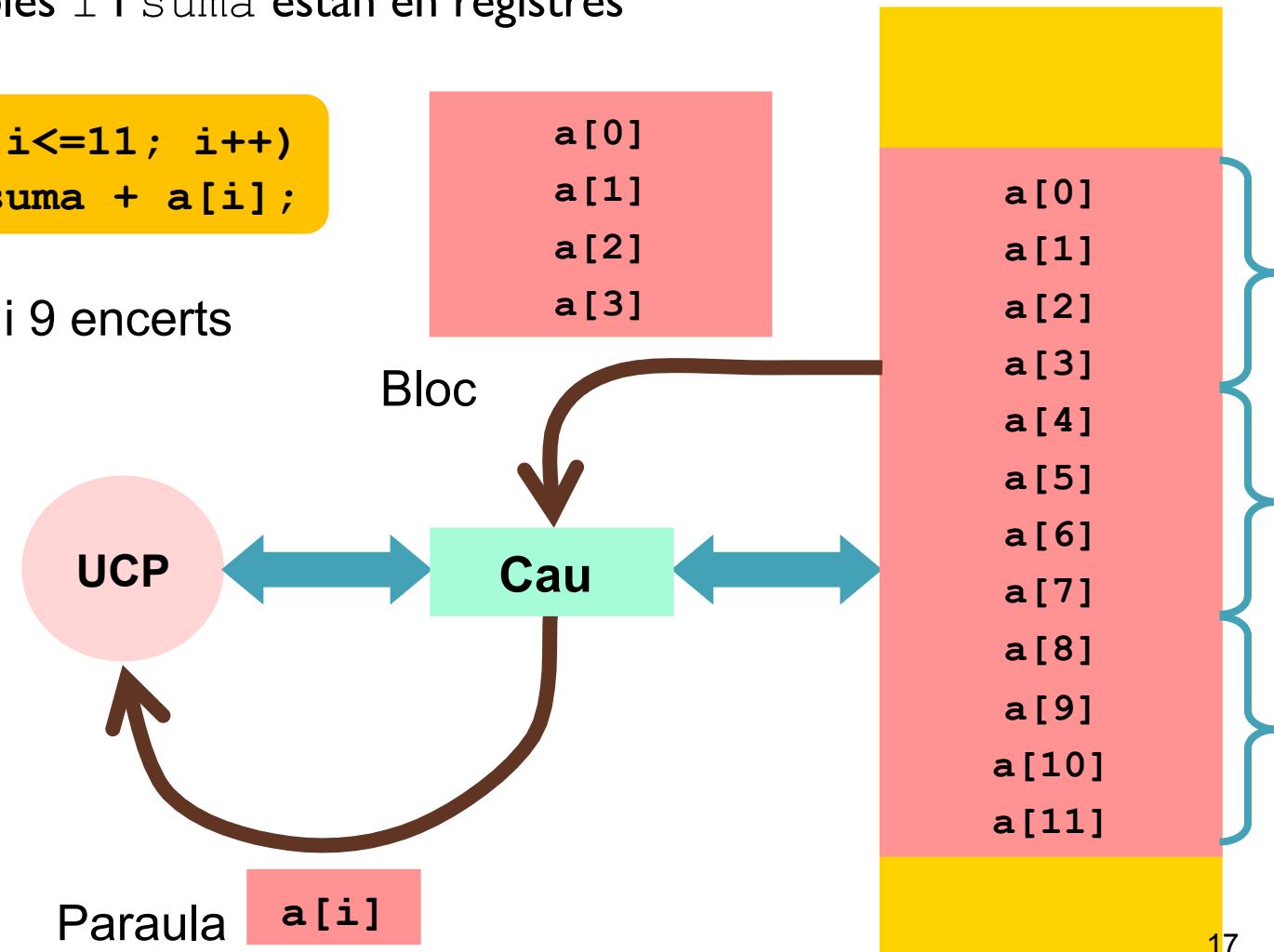


Localitat espacial en accés a dades

- Lectura del vector $a[]$ de 12 enters de 32 bits
 - ✓ Les variables i i $suma$ estan en registres

```
for (i=0; i<=11; i++)  
    suma = suma + a[i];
```

Hi ha 3 fallades i 9 encerts



Taxa d'encerts i temps d'accés a les dades

- Taxa d'encerts (H): proporció d'encerts

$$\begin{aligned} H &= \frac{\text{Nombre d'encerts}}{\text{Nombre d'accessos}} \\ &= \frac{\text{Nombre d'accessos} - \text{Nombre de fallades}}{\text{Nombre d'accessos}} \end{aligned}$$

- Taxa de fallades: $1-H$ (proporció de fallades)
- Temps mitjà d'accés a les dades

$$T_m = H \times T_{encert} + (1-H) \times T_{fallada}$$

Exemple de càlcul

- Processador a 1 GHz
 - ✓ Temps d'accés a la cau: temps de cicle del processador
 - ✓ H (en promig): 0,95 (95%)
 - ✓ Resolució de la fallada per la memòria principal: 40 ns ($t_{RCD} + t_{CL}$)
- Càlcul del temps mitjà d'accés a les dades

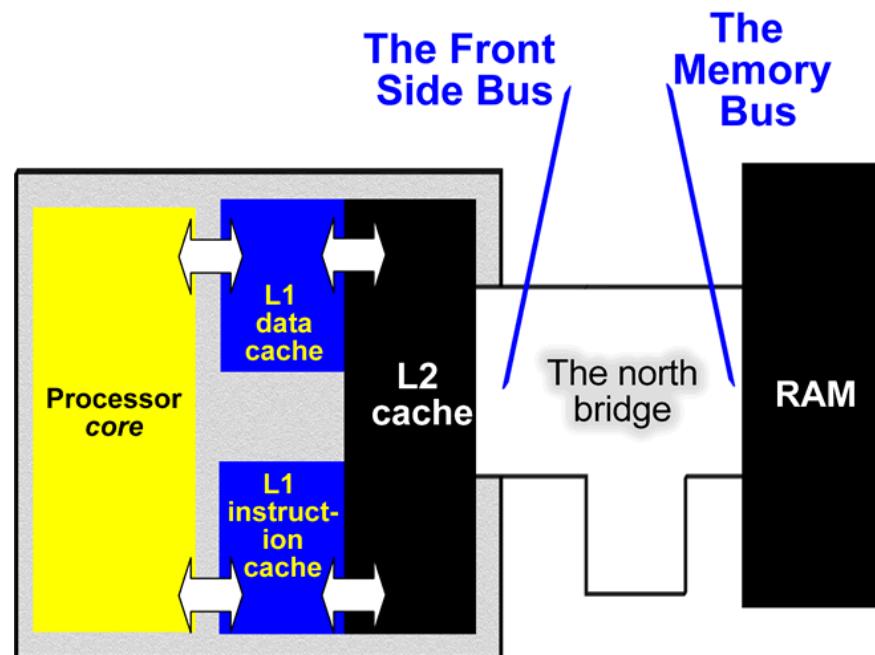
$$T_{cicle} = \frac{1}{1\text{GHz}} = \frac{1}{10^9 \text{ s}^{-1}} = 1\text{ns}$$

$$\begin{aligned} T_m &= H \times T_{encert} + (1 - H) \times T_{fallada} \\ &= 0.95 \times 1 + 0.05 \times 40 = 2,95\text{ ns} \end{aligned}$$

L'objectiu final és que T_m siga el més paregut possible al temps d'accés a la memòria cau

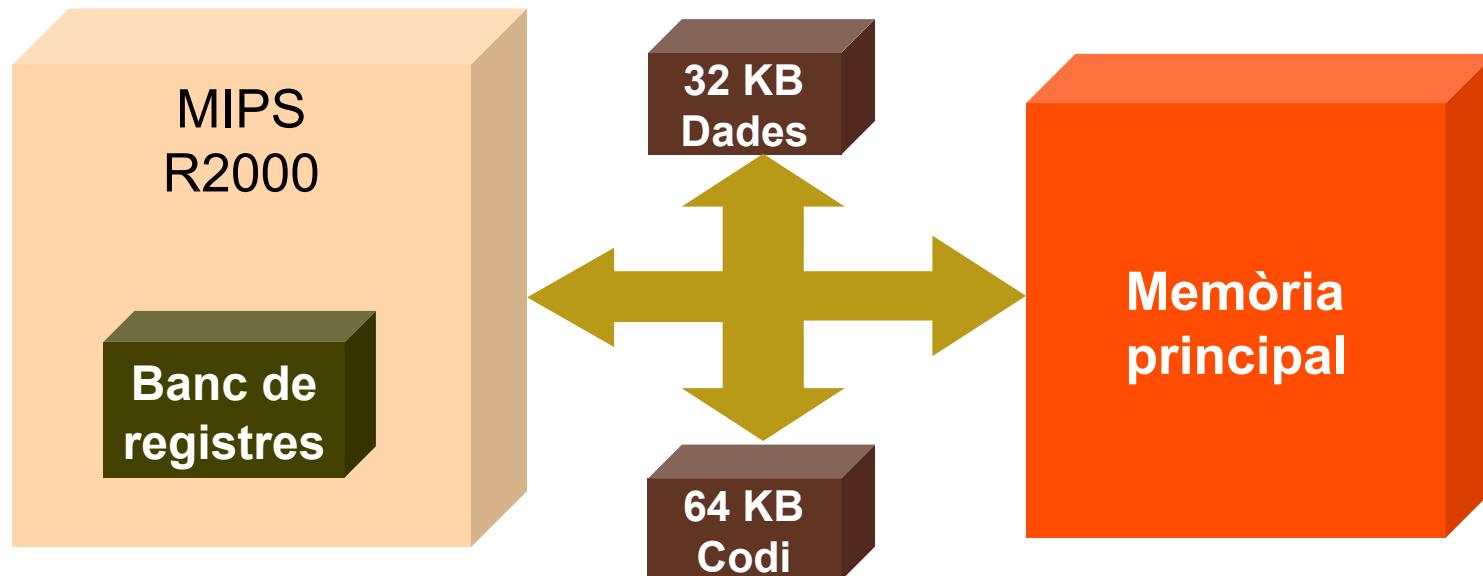
Caus unificades i duals

- Unificades: contenen dades i instruccions
- Duals (*split caches*) o arquitectura Harvard
 - ✓ Hi ha una memòria cau per a emmagatzemar només dades i una altra per a instruccions en el mateix nivell
 - ✓ Es redueix la complexitat hardware i permet l'accés en paral·lel a les dues caus
- En la pràctica
 - ✓ L1 sol ser dual
 - ✓ L2 sol ser unificada



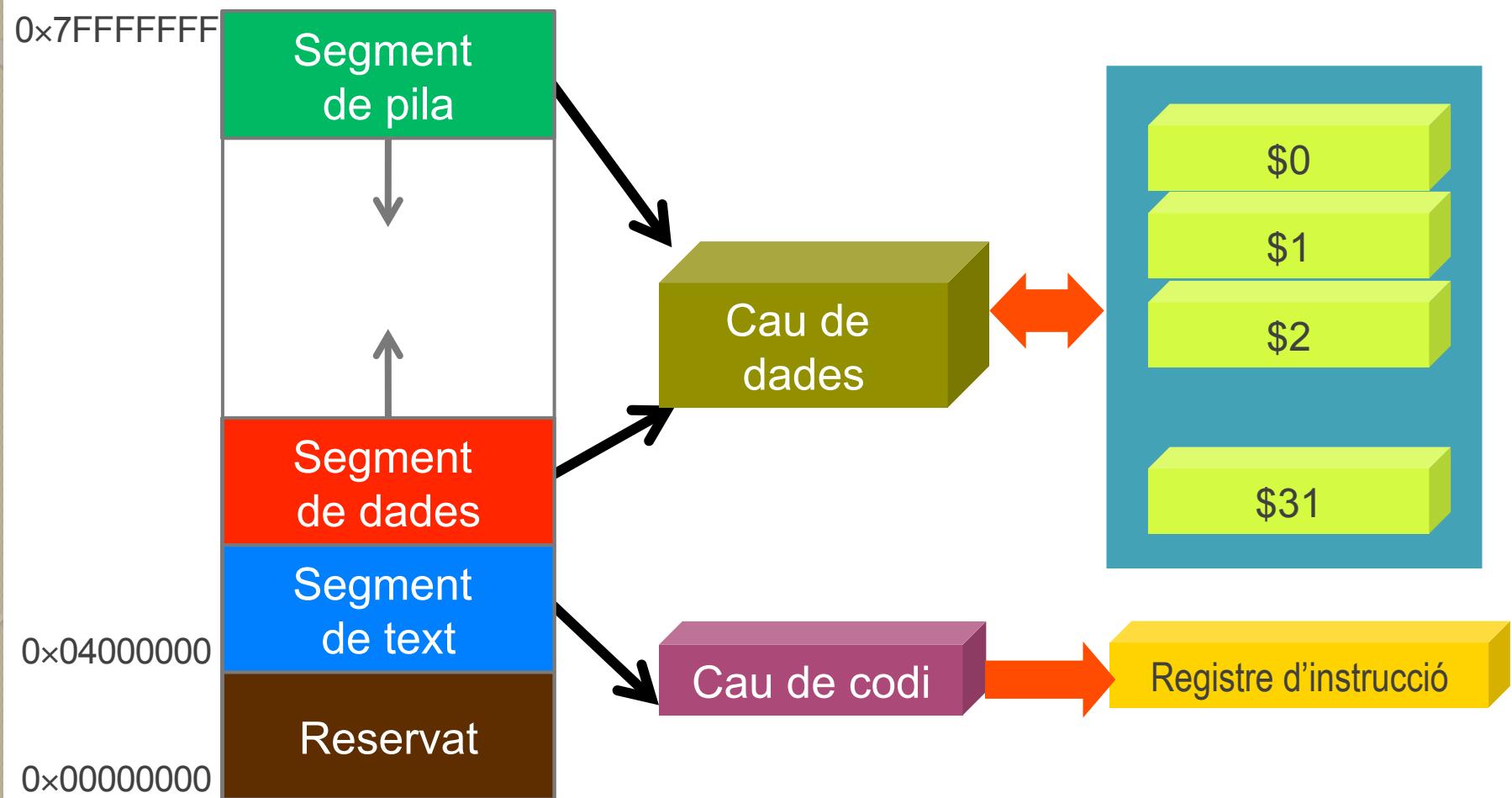
La memòria cau en el MIPS R2000 (1985)

- Hi ha només un nivell de cau (L1)
- Externa al processador (*off chip*)
 - ✓ Cau de dades des de 4 a 64 KB (*D-cache*)
 - ✓ Cau de codi des de 4 a 64 KB (*I-cache*)



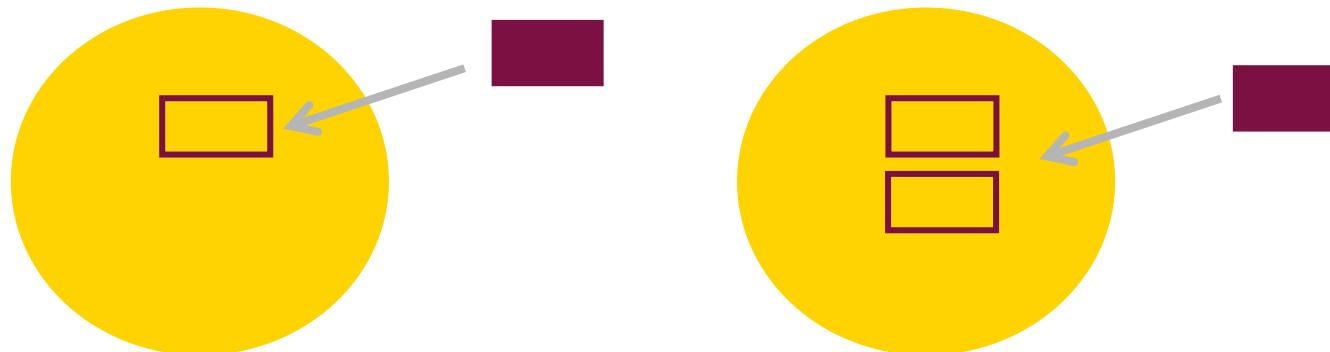
Flux d'informació entre la cau i els registres

- La unitat de control se n'encarrega de la gestió



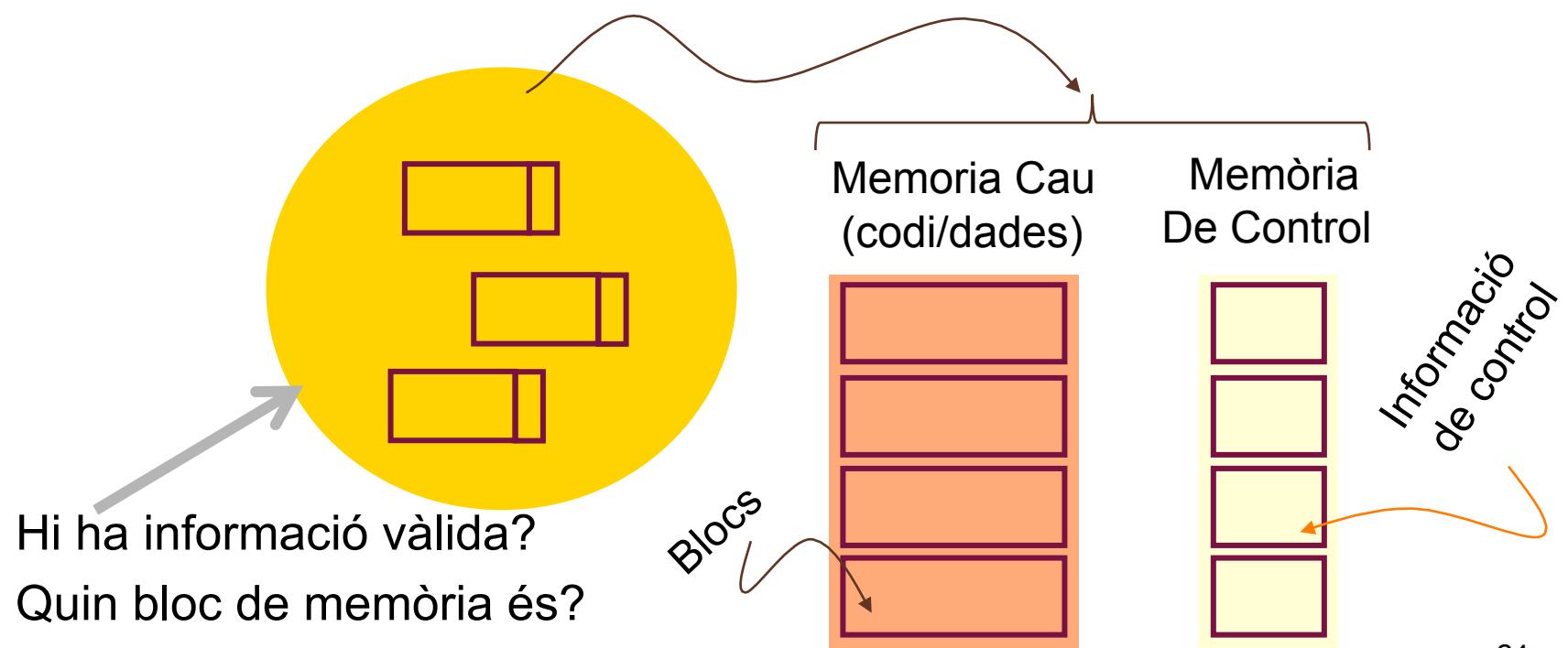
Estratègies d'ubicació d'informació

- Qüestions que ha de resoldre el sistema de memòria cau
 - ✓ On s'emmagatzema un bloc quan es du a la cau
 - ✓ Com es localitza una dada emmagatzemada en la cau
- Per a dur a terme ambdues funcions s'aplica una funció de correspondència o mapeig
- Estratègies
 - ✓ Un bloc sempre s'ubicarà en la mateixa línia
 - ✓ El bloc es pot ubicar en una d'entre un conjunt de línies (hi ha possibilitat d'elecció)



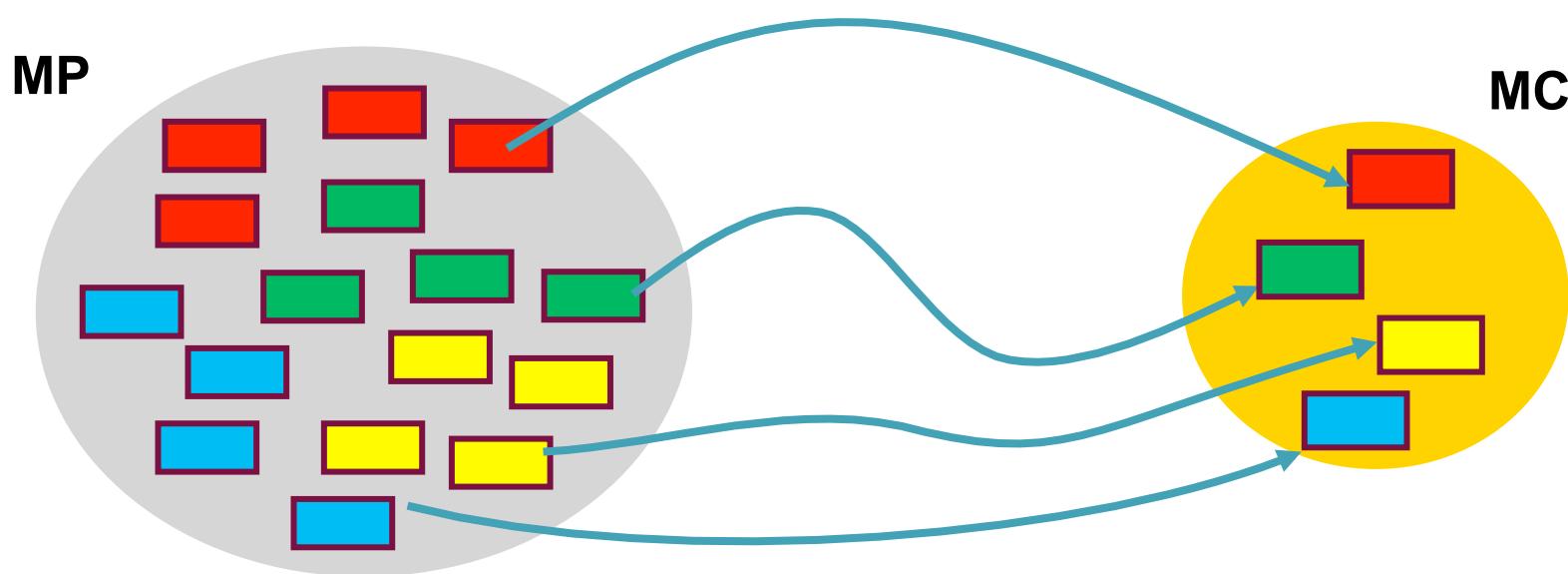
Informació de control de la memòria cau

- Informació emmagatzemada en la cau
 - ✓ Blocs de la memòria principal (codi i dades)
 - ✓ Bits de control per a la gestió de cada línia
 - Bit de vàlid: indica si la línia conté un bloc o no
 - Etiqueta: identifica el bloc de memòria contingut en ella



Funcions de correspondència (*mapping*)

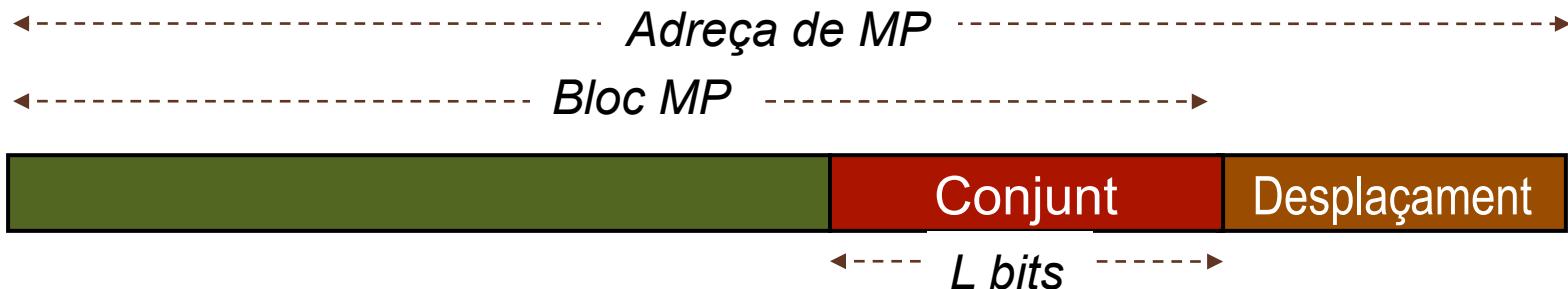
- **Objectiu:** decidir la línia de cau on s'emmagatzema un bloc de MP
- S'utilitza un procés de **classificació**:
 - Definició de classes o recipients en la MC
 - Agrupació dels blocs de MP per classes
 - Els blocs de MP de la classe (i) corresponen amb el recipient (i) de la cau



Funcions de correspondència (*mapping*)

- Les classes o recipients de la cau es denominen CONJUNTS
- El nombre de conjunts de la cau és sempre potència de 2
- Un conjunt està constituït per una o mes línies (potències de 2)
- Els blocs de MP es classifiquen en tantes classes com nombre de conjunts de la memòria cau

Bloc en MP → Conjunt en la cau



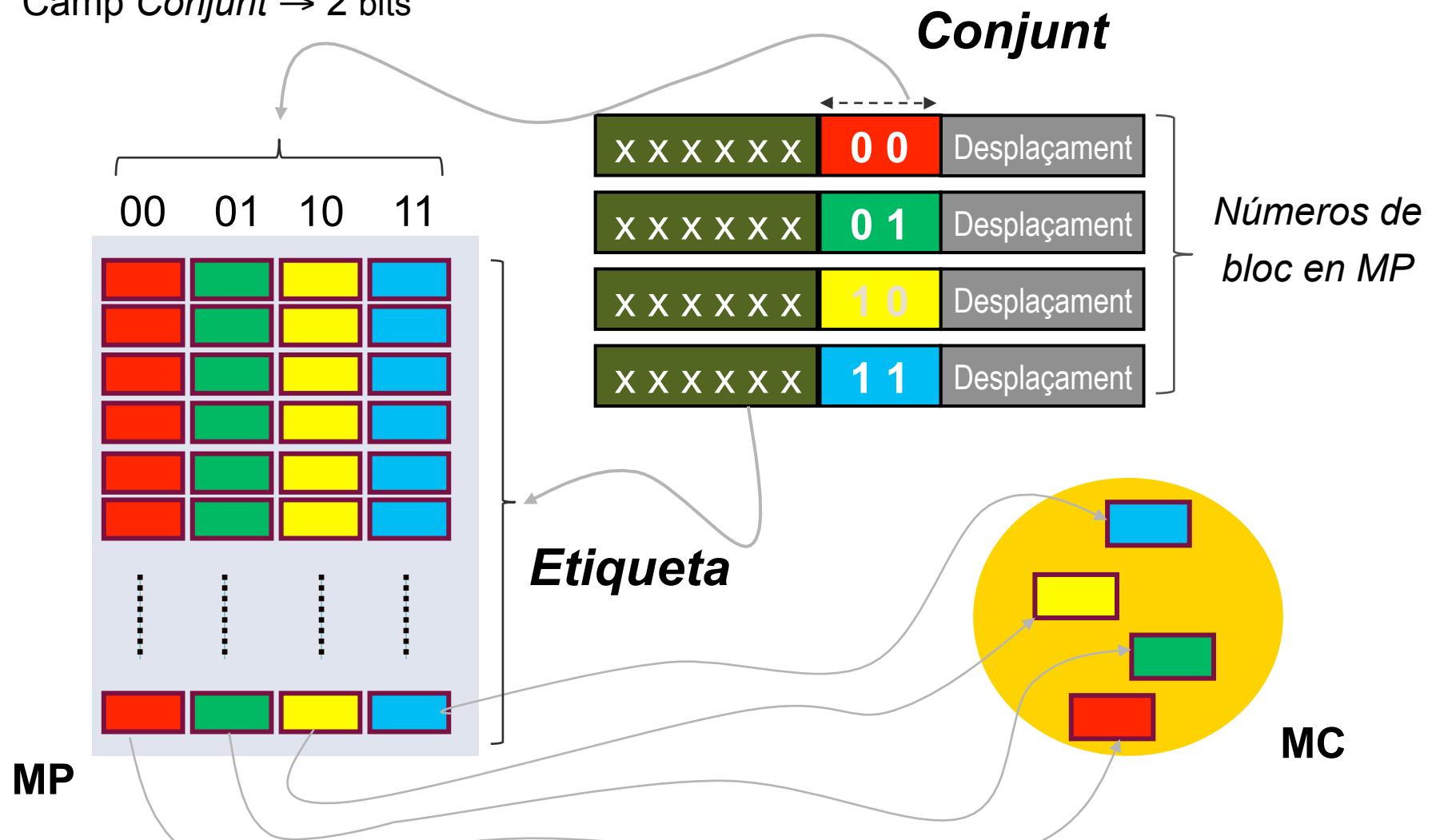
Conjunt = Número de bloc en MP MOD Nombre de conjunts en MC

NOMBRE de conjunts en MC = 2^L

Funcions de correspondència (mapping)

Nombre de conjunts en MC = 4

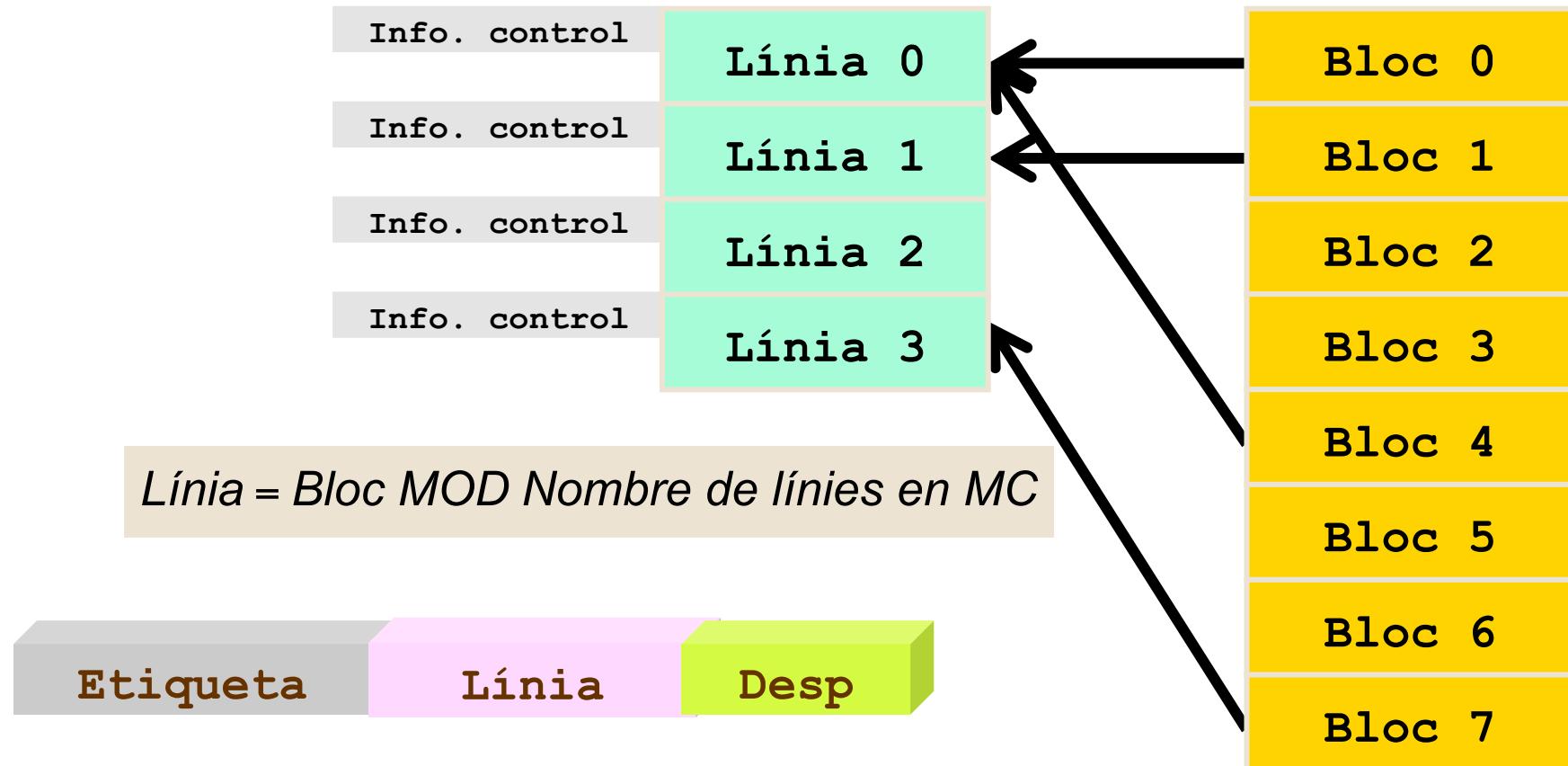
Camp *Conjunt* → 2 bits



Funcions de correspondència (*mapping*)

- **Directa**
 - ✓ Un bloc s'emmagatzema sempre en la mateixa línia
- **Associativa per conjunts de n vies**
 - ✓ Un bloc s'emmagatzema sempre en el mateix conjunt de línies
 - ✓ Dins del conjunt es pot elegir qualsevol de les n línies
- **Completament associativa**
 - ✓ Un bloc es pot emmagatzemar en qualsevol línia de la memòria cau (només hi ha un conjunt)
- En la pràctica, la funció equival a seleccionar aquells bits de l'adreça que identifiquen el conjunt
- L'adreça és interpretada per la cau segons la seu organització interna

Aproximació: directa



Un bloc s'ubica en una única línia
La línia 0 pot albergar els blocs 0 i 4
(encara que no de forma simultània)

Correspondència associativa

- Inconvenients de la correspondència directa: baixa taxa d'encerts i etiquetes massa llargues
- En la pràctica s'inclou un cert grau d'associativitat
- Caus de n vies
 - ✓ Les línies s'agrupen en conjunts de n línies
 - ✓ Un bloc té n ubicacions diferents dins del seu conjunt

$$\text{Nombre de conjunts} = \frac{\text{Nombre de línies en la MC}}{\text{Nombre de vies}}$$

Aproximació: associativa de 2 vies

Conjunt 0

Conjunt 1

Info. control	Línia 0
Info. control	Línia 1
Info. control	Línia 2
Info. control	Línia 3

Bloc 0
Bloc 1
Bloc 2
Bloc 3
Bloc 4
Bloc 5
Bloc 6
Bloc 7

Conjunt = Número de bloc MOD Nombre de conjunts

Etiqueta

Conjunt

Desp

Un bloc s'ubica en un únic conjunt
El conjunt 0 pot albergar els blocs 0, 2, 4 i 6 (encara que en un màxim de dos)

Aproximació: totalment associativa

Conjunt

Info. control	Línia 0
Info. control	Línia 1
Info. control	Línia 2
Info. control	Línia 3

Etiqueta

Desp

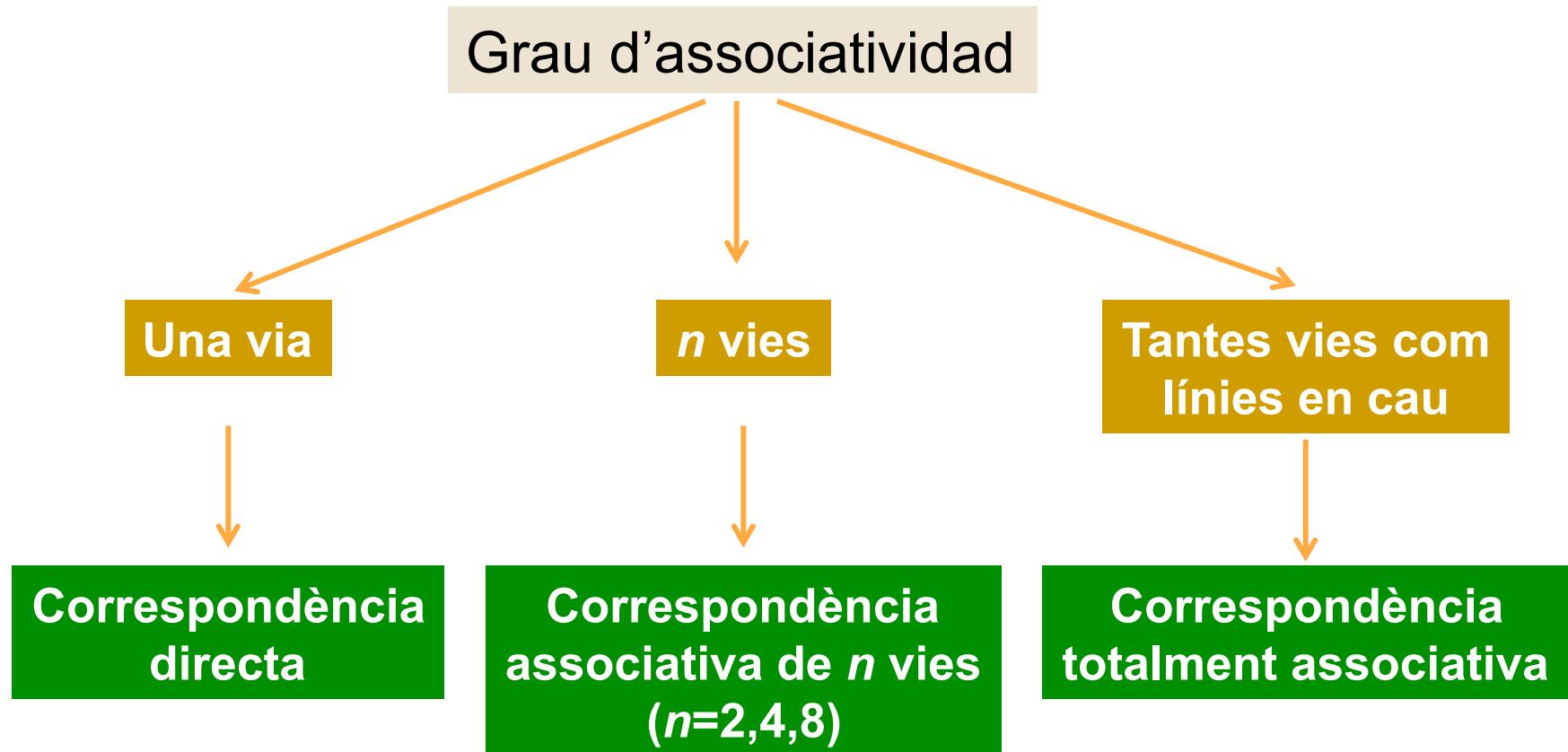
Bloc 0
Bloc 1
Bloc 2
Bloc 3
Bloc 4
Bloc 5
Bloc 6
Bloc 7

Una línia pot albergar qualsevol bloc de memòria

La memòria cau triarà la ubicació de cada bloc segons la disponibilitat de línies en tota la cau

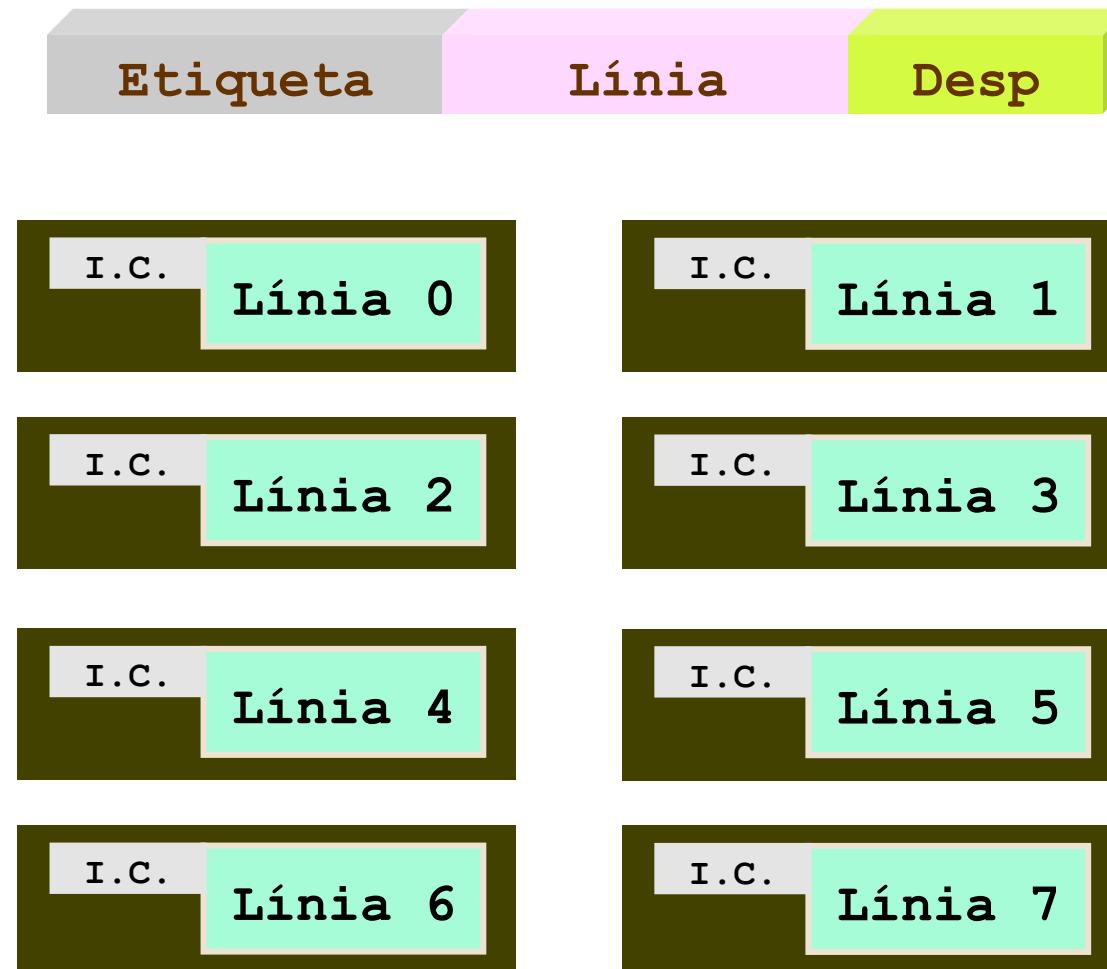
Es tracta d'una correspondència associativa de 4 vies

Resum de correspondències



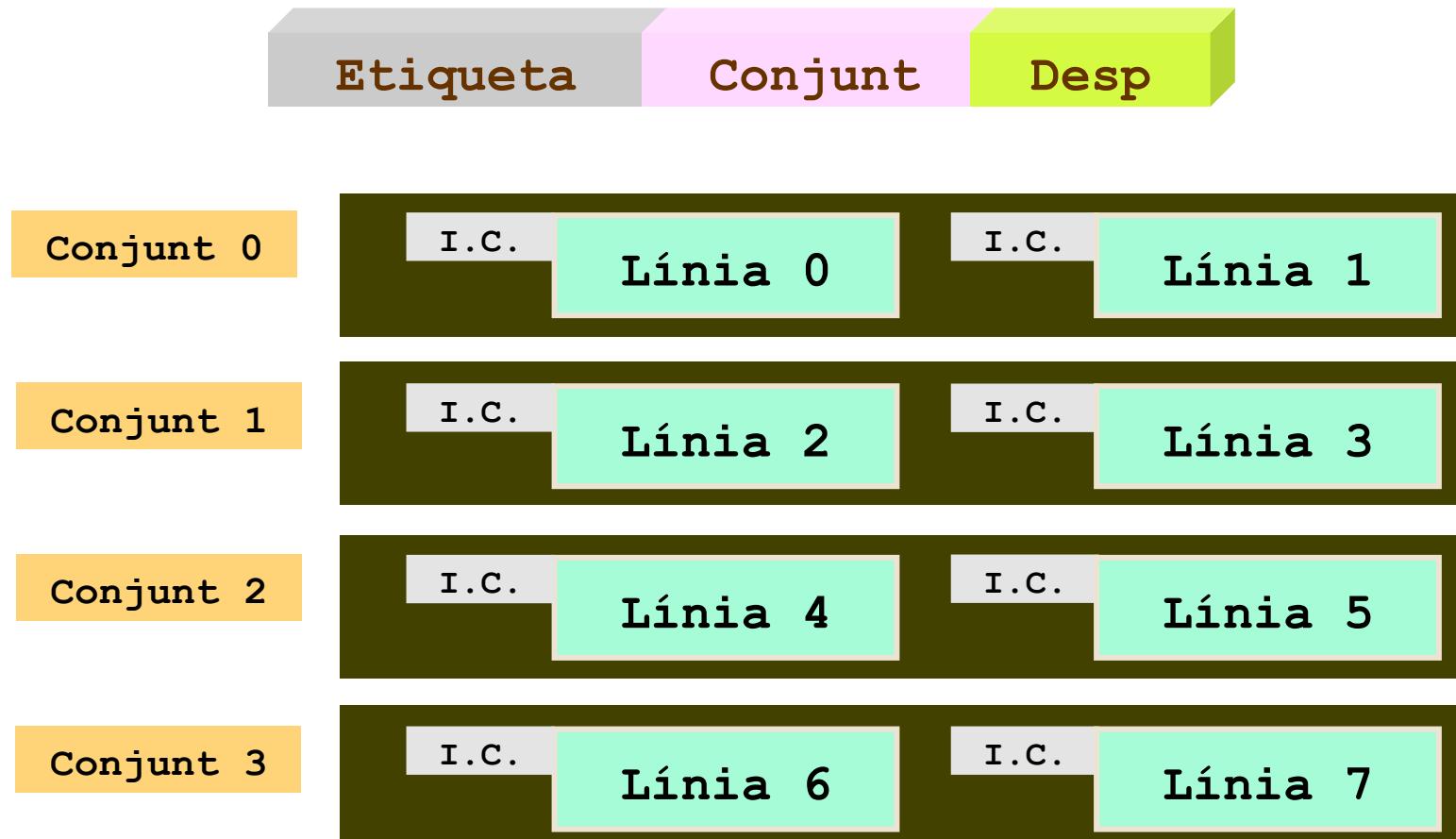
Correspondència associativa d'una via

- Es tracta d'una correspondència directa



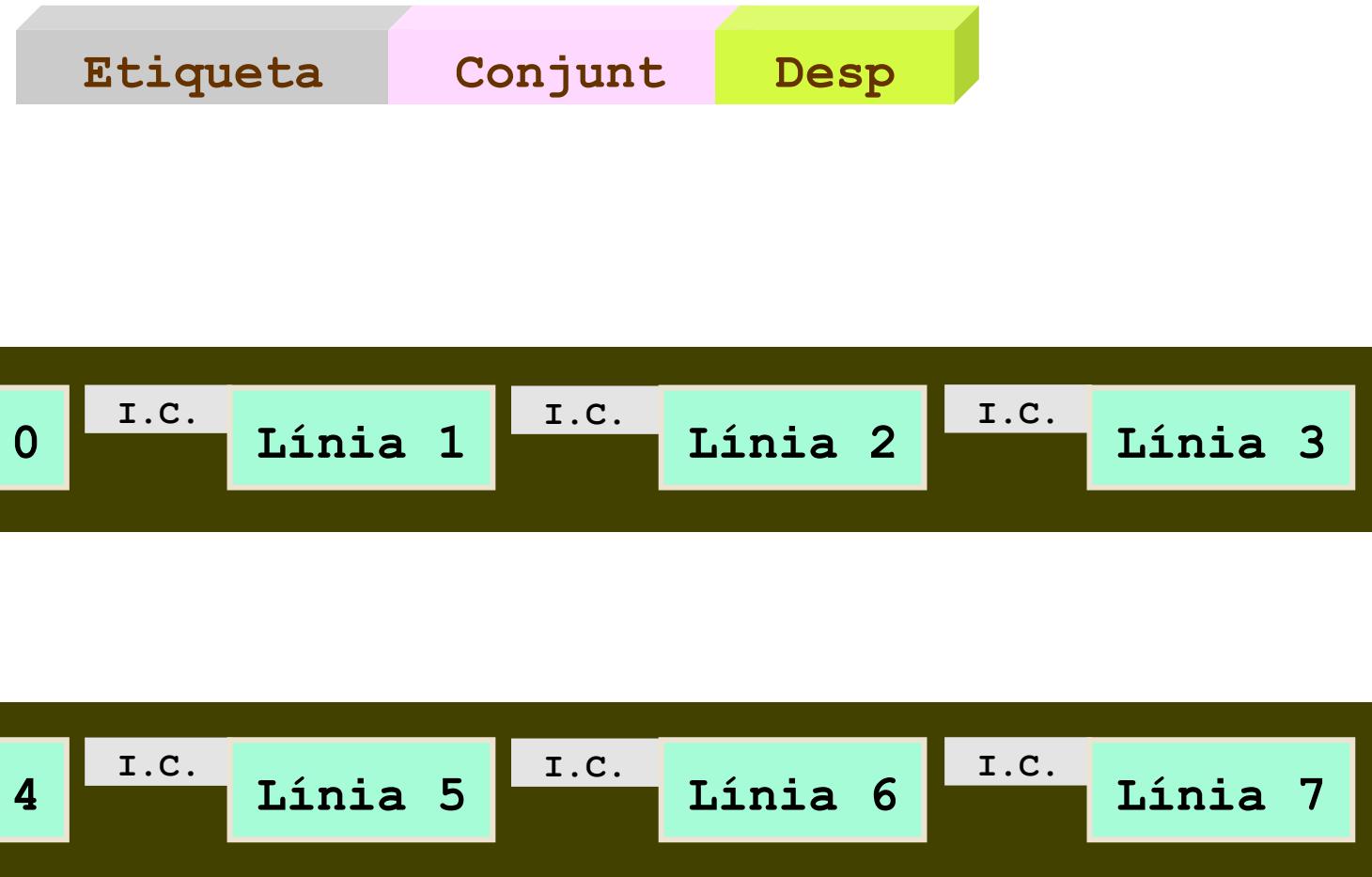
Correspondència associativa de 2 vies

- Hi ha 8 línies agrupades en 4 conjunts de 2 línies
- Cada bloc té dues possibles ubicacions



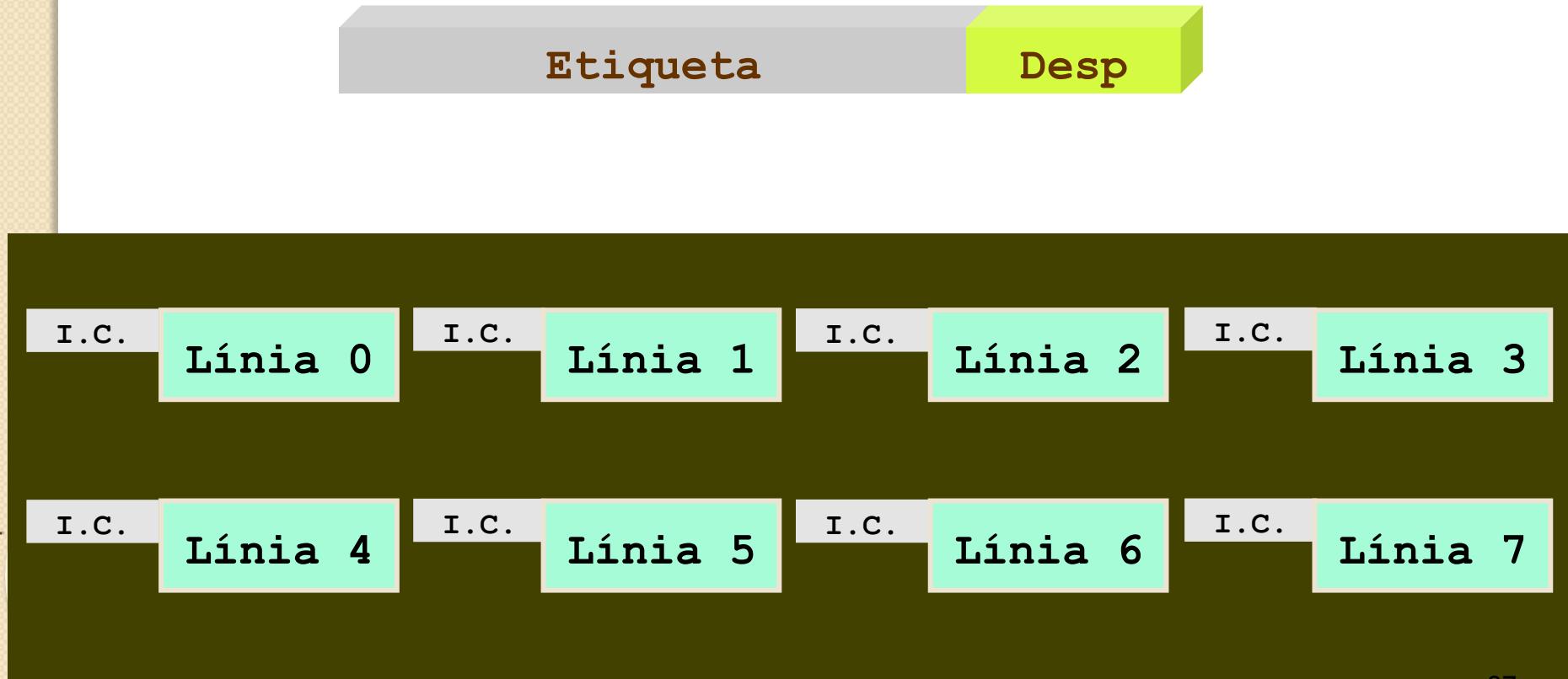
Correspondència associativa de 4 vies

- Hi ha 8 línies agrupades en 2 conjunts de 4 línies



Correspondència associativa de 8 vies

- Només hi ha “un conjunt” amb totes les línies
- Màxim grau d’associativitat



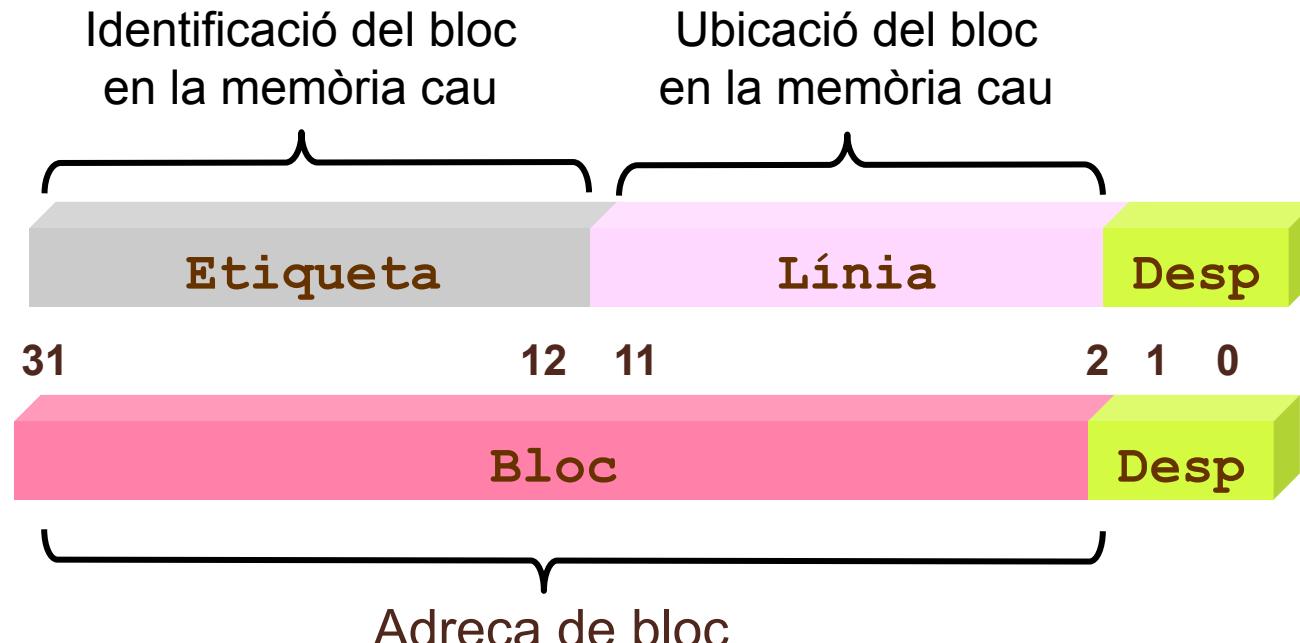
Exercici: ompliu les taules següents

UCP		Memòria Cau				
Adr.	W	Grandària	vies	Grandària Línia	# Línies	#cjts
32	32	32KB	2	32B		
32	32	8KB			256	64
24	32	32KB	1			
24	32		128	32B		
36	64	32KB	2	64B		
36	64	8KB				

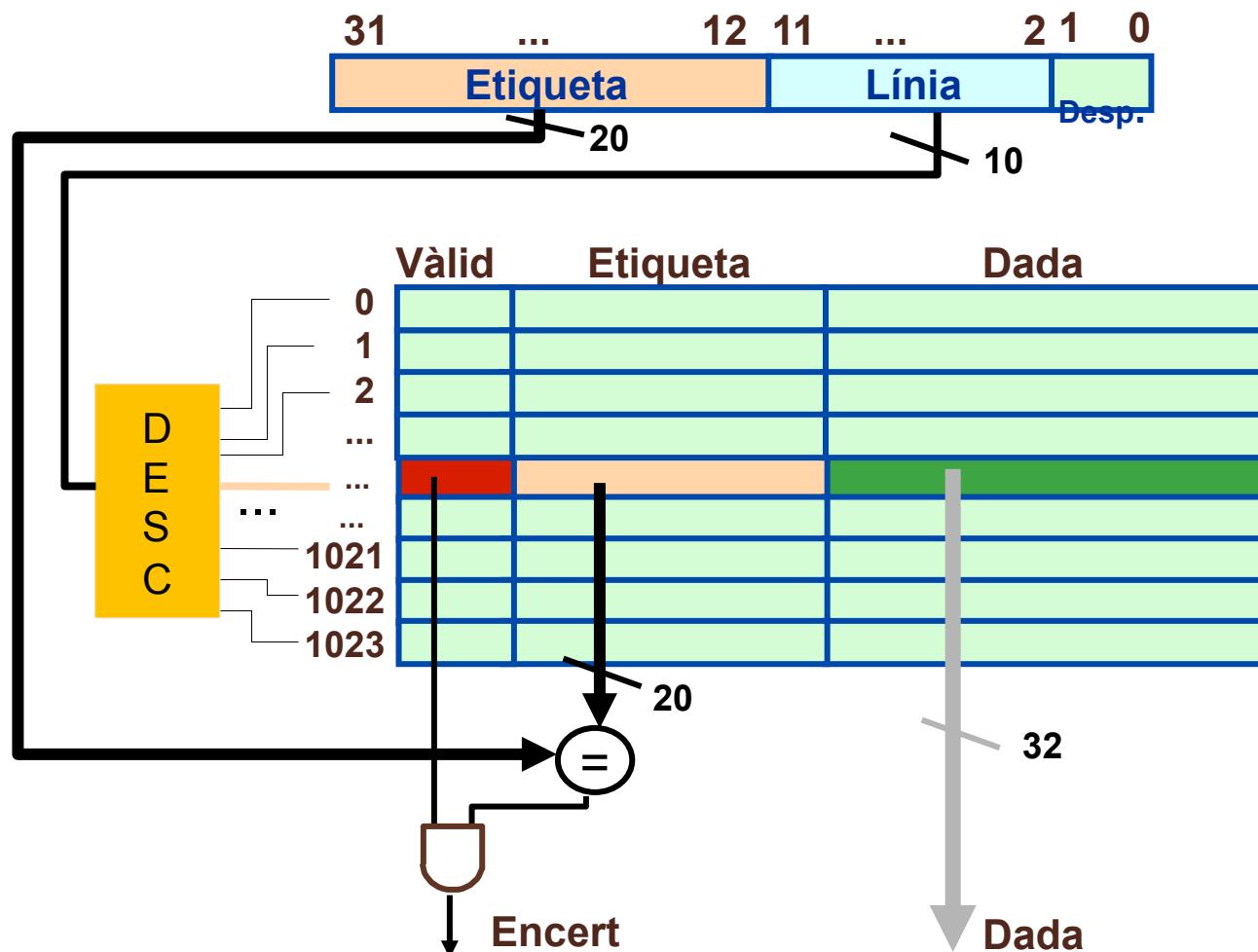
Bits d'adreça		
etiqueta	conjunt	desp
9	10	5
19	0	5
25	5	6

Exemple per al MIPS R2000 (directa)

- Bus d'adreces de 32 bits
- Longitud de bloc: 1 paraula (4 bytes)
- Capacitat de cau: 4 KB
 - ✓ Línies en cau: 1024
- Interpretació dels camps de l'adreça



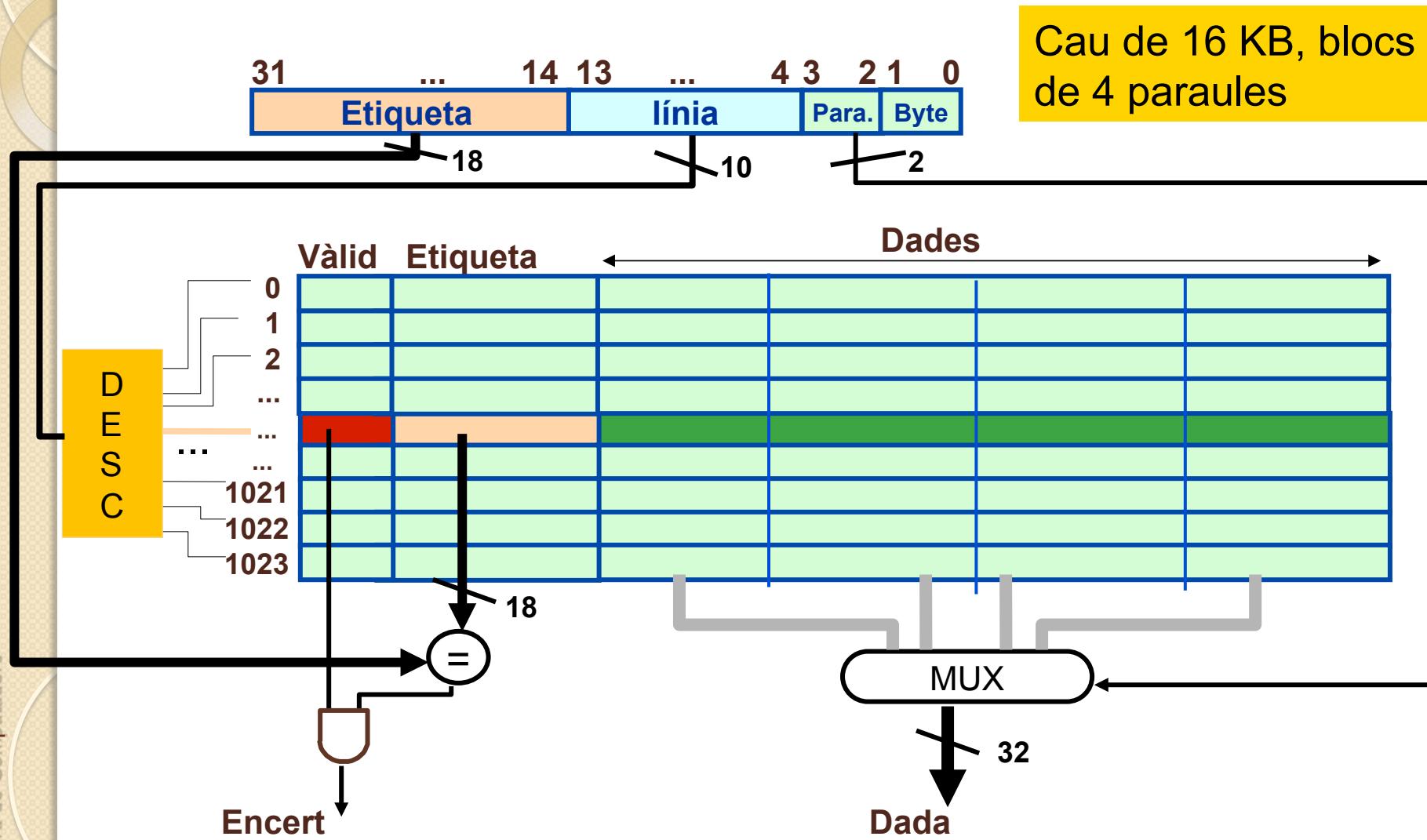
Correspondència directa: organització interna



Cau de 4 KB,
blocs d'una
paraula

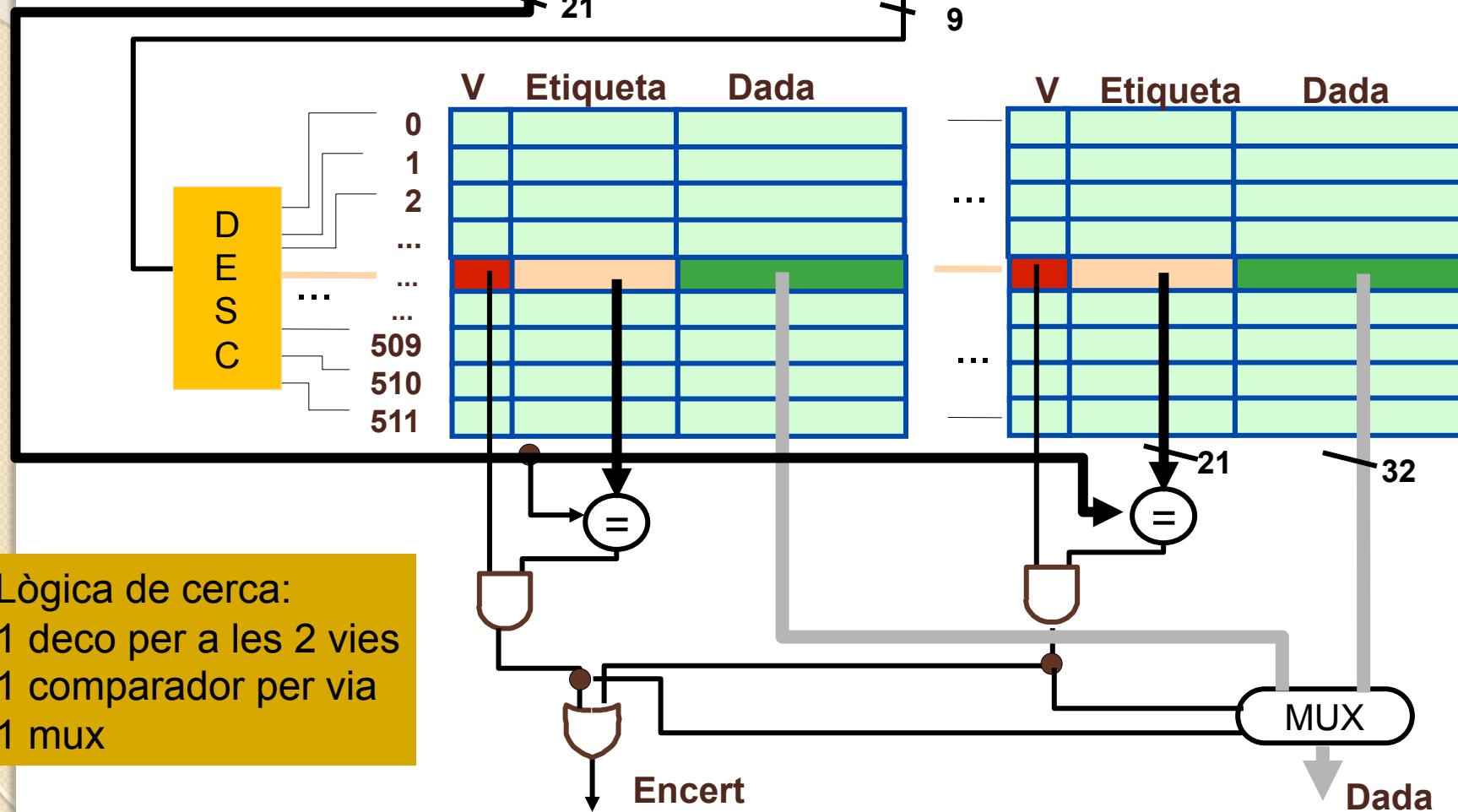
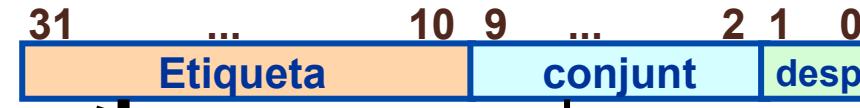
Lògica de cerca: 1 descodificador, 1 comparador, 1 porta AND

Correspondència directa: organització interna



Implementació de una cau associativa

Cau de 4 KB,
2 vies, 1 paraula



Exemple d'accés a un vector

```
.data 0x10000000
.word 2,6,5,7,8,3,4,1,9,0
.globl __start

.text 0x00400000

_start:
    lui $t0, 0x1000
    addi $t1, $zero, 10
    addi $a0, $zero, $zero
bucle:   lw $t2, 0($t0)
          add $a0, $a0, $t2
          addi $t2, $t2, 1
          addi $t0, $t0, 4
          addi $t1, $t1, -1
          bnez $t1, bucle
```

Segment de dades:
10 paraules de 32 bits

Segment de codi:
9 instruccions de 32 bits

Adreces emeses pel processador

Segment de dades

0x10000000
0x10000004
0x10000008
0x1000000C
0x10000010
0x10000014
0x10000018
0x1000001C
0x10000020
0x10000024

10 accessos de lectura
(efecte de la
instrucció lw)

Segment de codi

0x00400000
0x00400004
0x00400008
0x0040000C
0x00400010
0x00400014
0x00400018
0x0040001C
0x00400020
0x0040000C
0x00400010
0x00400014
0x00400018
0x0040001C
0x00400020
...

Adreces de les
instruccions
del bucle

$3 + 10 \times 6 = 63$ accessos de
lectura (lectura de les
instruccions per la unitat
de control del processador)

Taxa d'encerts en l'accés a les dades

- Suposem cau directa de 32 KB amb blocs de 16 bytes
 - ✓ Les dades s'ubiquen en 3 blocs
 - ✓ 17 bits d'etiqueta, 11 de línia, 4 de desplaçament

0001000000000000 000000000000 0000

0001000000000000 000000000000 0100

0001000000000000 000000000000 1000

0001000000000000 000000000000 1100

0001000000000000 000000000001 0000

0001000000000000 000000000001 0100

0001000000000000 000000000001 1000

0001000000000000 000000000001 1100

0001000000000000 000000000010 0000

0001000000000000 000000000010 0100

$$H = \frac{10 - 3}{10} = 0,7$$

Taxa d'encerts al codi

- Suposem cau directa de 64 KB amb blocs de 16 bytes
 - ✓ 18 bits d'etiqueta, 12 de línia, 4 de desplaçament

0000000001000000 0000000000000 0000

0000000001000000 0000000000000 0100

0000000001000000 0000000000000 1000

0000000001000000 0000000000000 1100

0000000001000000 0000000000001 0000

0000000001000000 0000000000001 0100

0000000001000000 0000000000001 1000

0000000001000000 0000000000001 1100

0000000001000000 0000000000010 0000

0000000001000000 0000000000000 1100

0000000001000000 0000000000001 0000

0000000001000000 0000000000001 0100

...

46

Adreces de les instruccions del bucle

$$H = \frac{63 - 3}{63} = 0,95$$

Tipus de fallades (són exclusius)

- **Fallades d'arrancada**

- ✓ Es produeixen la primera vegada que es referència un bloc (no ha estat encara en la cau)

- **Fallades de capacitat**

- ✓ Es produeixen degut a la limitació de la grandària de la cache

- **Fallades de conflicte (o col·lisió)**

- ✓ Es produeixen quan el conjunt al què mapeja es troba ple però la cau no
 - ✓ Només es produeixen en memòries cau de correspondència directa i associativa per conjunts, però no en les completament associatives

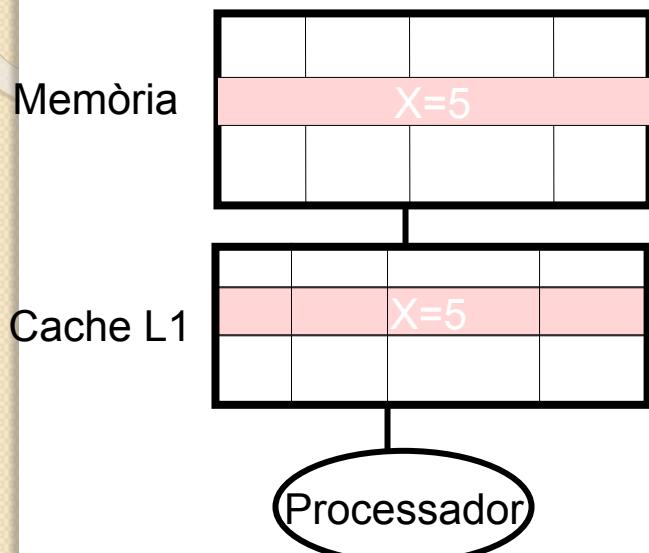
- Una cau més gran reduiria les fallades de capacitat i un major nombre de vies les de conflicte; però podria incrementar el temps d'accés

Polítiques de lectura

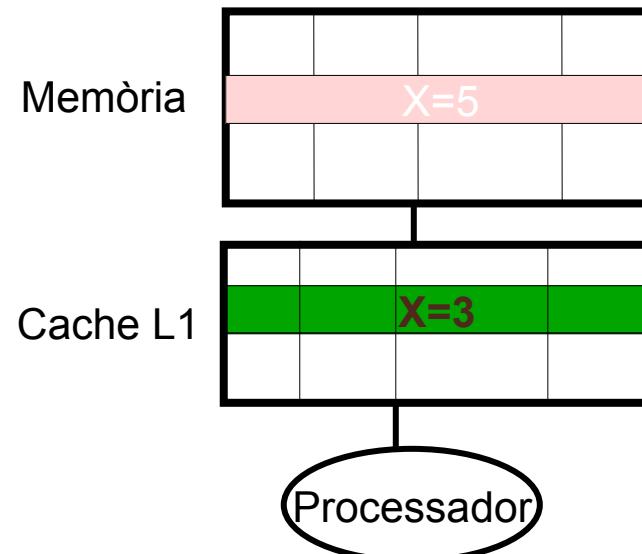
- Encert de lectura
 - ✓ Situació desitjada!
- Fallada de lectura
 - ✓ Cal dur el bloc des de la memòria principal (o del nivell inferior de la jerarquia) a la memòria cau
 - El temps d'accés s'incrementa
 - ✓ La memòria principal, en llegir el bloc, ofereix primer la paraula que va causar la fallada (*critical word first*)
 - La resta del bloc s'acaba de portar en ordre circular
 - ✓ Per tant, el temps a resoldre la fallada és la latència d'accés a les dades $t_{RCD} + t_{CL}$

El problema de la coherència

- Les escriptures poden provocar incoherència



Situació 1: fallada de lectura



Situació 2: encert en escriptura
Si el processador escriu només en L1 →
No hi ha coherència de L1 amb MP

Solució al problema: polítiques d'escriptura

Polítiques d'encert en escriptura

- Encert d'escriptura

- ✓ Escriptura directa (*write through*)

- S'escriu la dada en la cau i en la memòria principal (nivell inferior)

- ✓ Escriptura posterior (*write back*)

- S'escriu en la cau però no en la memòria principal (nivell inferior)

- Quan es reemplaça el bloc s'escriu en el nivell inferior

- Hi cal un bit addicional: bit MODIFICAT (*dirty bit*)

➤ Buffers d'escriptura

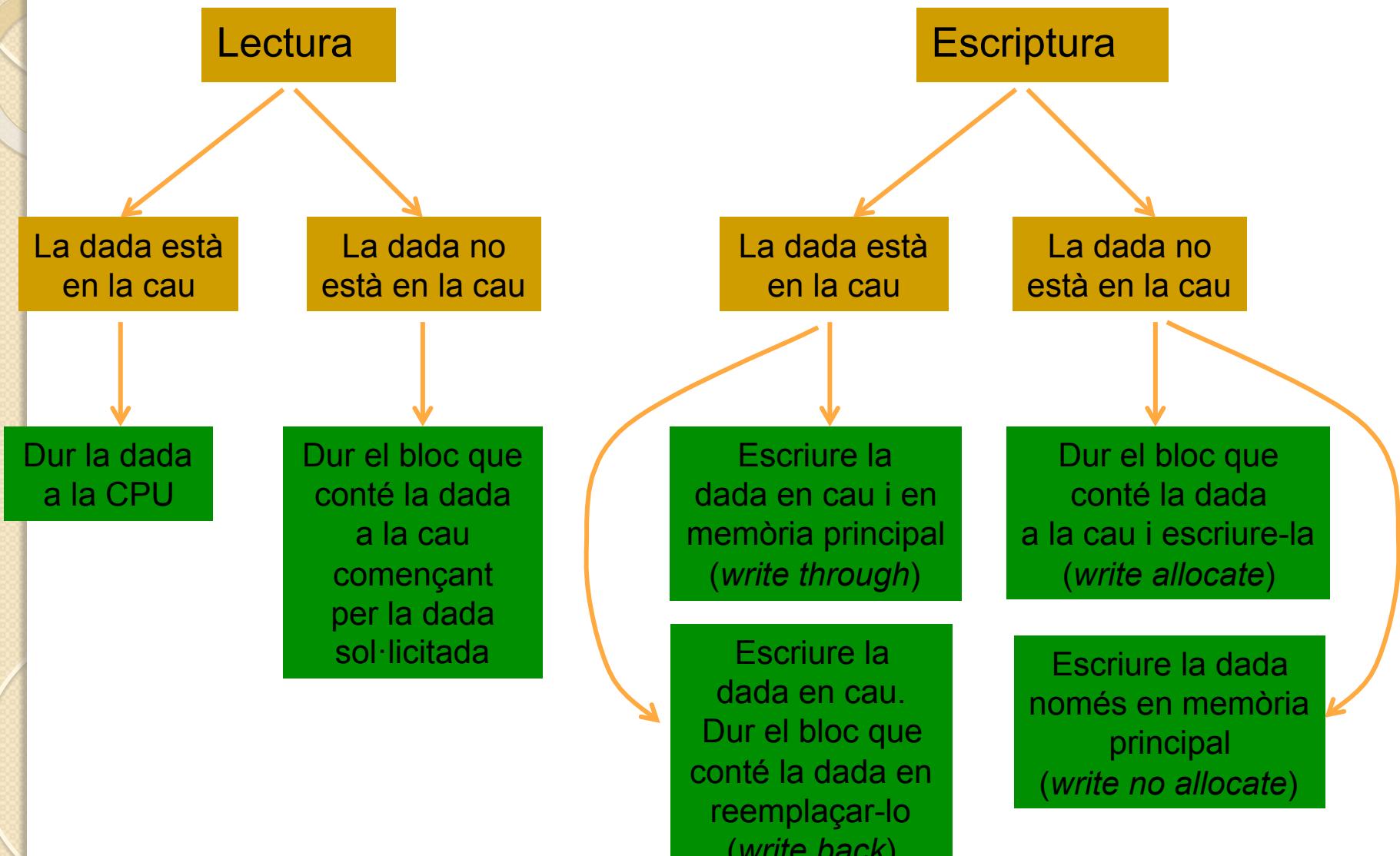
- ✓ Aprofiten per a evitar que la ruta de dades espere a que una escriptura en la memòria principal estiga feta

- ✓ Mentre el buffer està escrivint, la ruta de dades continua amb el cicle d'instrucció

Polítiques de fallada en escriptura

- Fallada d'escriptura
 - ✓ Ubicació en escriptura (*write-allocate*)
 - Gestió de la fallada semblant a una fallada de lectura
 - Es porta el bloc i s'escriu la dada (escriptura directa o posterior)
 - ✓ No ubicació en escriptura (*no-write-allocate*)
 - S'escriu la dada en el nivell inferior, no en la memòria cau
 - Només s'utilitza amb la política d'escriptura directa

Resum de polítiques de lectura i escriptura



Algorismes de reemplaçament

- S'apliquen quan un bloc ha d'emmagatzemar-se en un conjunt que es troba ple
- Es troben implementats en el hardware
- Exemples
 - ✓ Menys recentment usat (LRU)
 - Implementació
 - Cal un comptador per bloc per a mantindre l'orde de referència
 - Nombre de bits del comptador:
$$n = \log_2(\text{Nombre de vies})$$
 - ✓ Per ordre d'arribada (FIFO)
 - Bits addicionals (el mateix nombre que LRU)
 - ✓ ALEATORI (*random*)
 - No cal afegir bits

Detalls de l'algorisme LRU

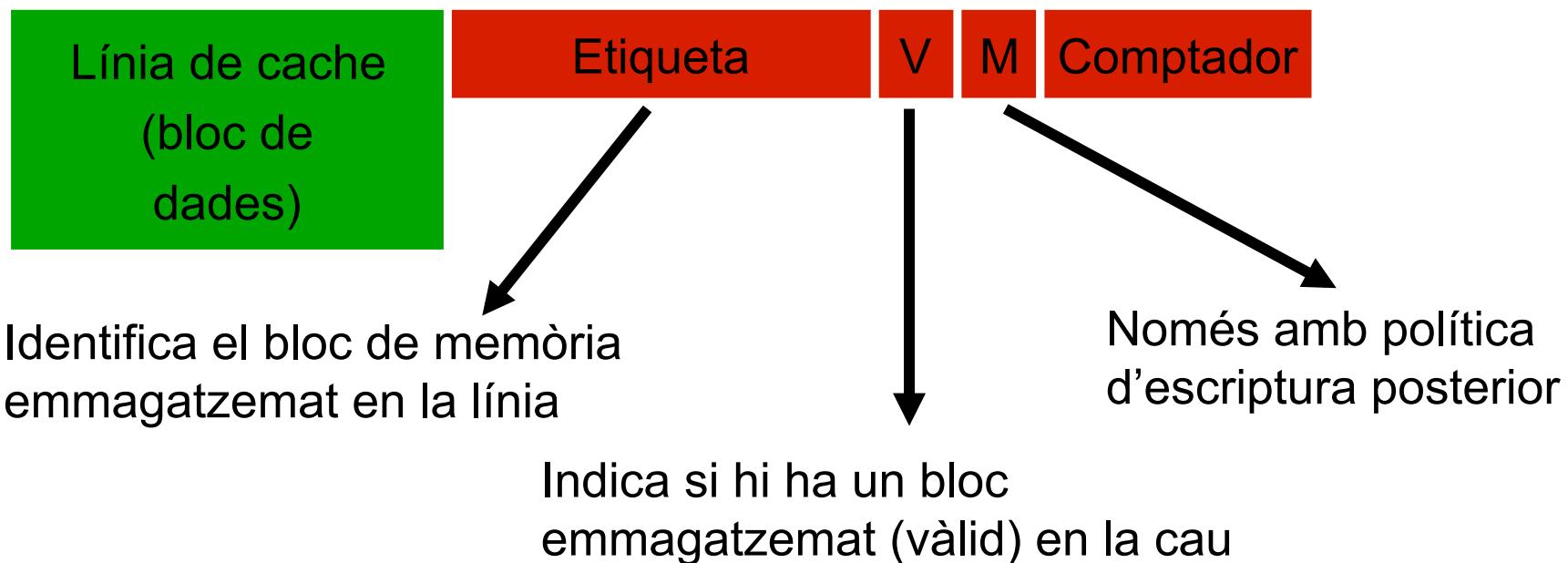
- Fallada en la cau
 - ✓ Si conjunt ple (tots els bits de línia vàlida actius)
 - Eliminar línia amb comptador màxim ($\text{ctr} = 2^n - 1$)
- Encert
 - ✓ $\forall \text{ ctr} < \text{ctr}_{\text{línia_referenciada}}; \text{ctr}++$
- En qualsevol cas, $\text{Ctr}_{\text{línia_referenciada}} = 0$

- Exemple:
 - ✓ Si els comptadors d'un conjunt de 4 vies són:

- $\text{Ctr_via_0} = 1$
 - $\text{Ctr_via_1} = 3$
 - $\text{Ctr_via_2} = 2$
 - $\text{Ctr_via_3} = 0$
-
- The diagram shows four circles arranged vertically. The top circle contains the number '1' and is connected by an arrow to a yellow box containing the text 'línia LRU (least recently used)'. The bottom circle contains the number '0' and is connected by an arrow to a yellow box containing the text 'línia MRU (most recently used)'. The middle two circles contain the numbers '3' and '2' respectively.

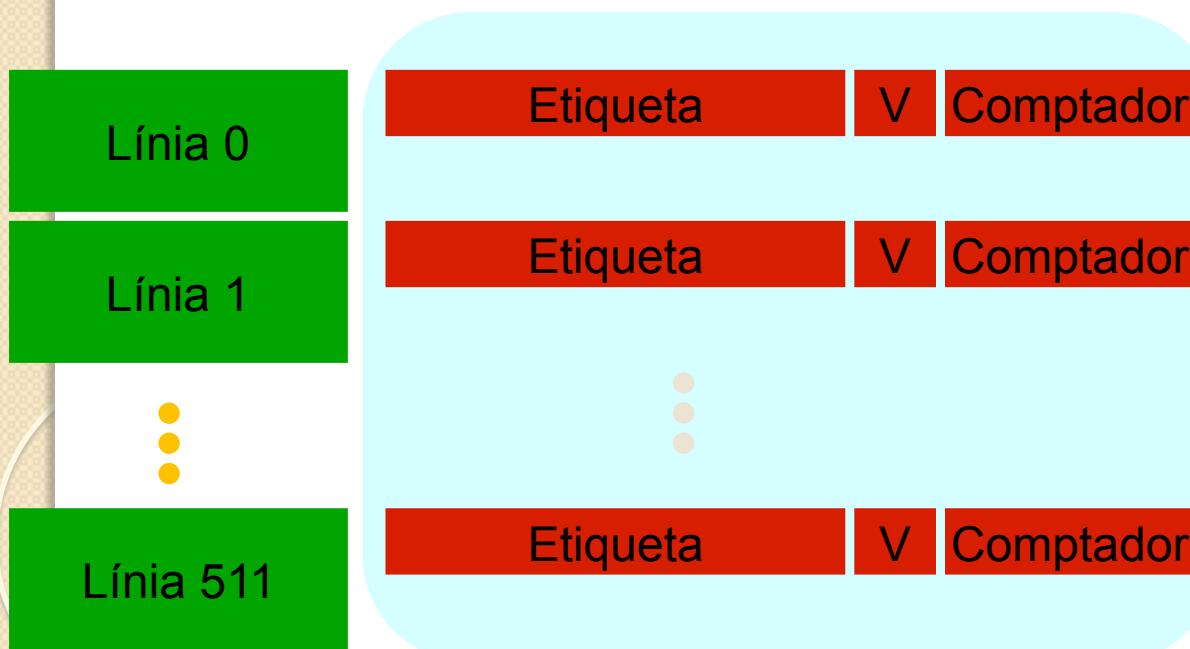
Informació de control: directori

- Informació addicional al bloc emmagatzemat en una línia
- El seu volum depén de la configuració de la memòria cau
 - ✓ Bits de l'etiqueta
 - ✓ Bit de vàlid, bit de modificat (sols política *writeback*)
 - ✓ Bits del comptador de l'algorisme de reemplaçament



Càcul del volum del directori

- Processador amb 32 bits d'adreça
- Cau de 512 línies de 16 bytes (8 KB)
 - ✓ Associativa per conjunts de 4 vías (128 conjunts)
 - ✓ Política d'escriptura: actualització directa (*write through*)
 - ✓ Algorisme de reemplaçament FIFO (tria d'un bloc entre quatre)



Etiqueta: 21 bits
Bit de vàlid: 1 bit
Comptador: 2 bits

$$\begin{aligned} V &= 512 \times (21 + 1 + 2) \\ &= 12288 \text{ bits} \\ &= 1.5 \text{ KB} \end{aligned}$$

SiSoftware Sandra: memòria cau

Processors - SiSoftware Sandra

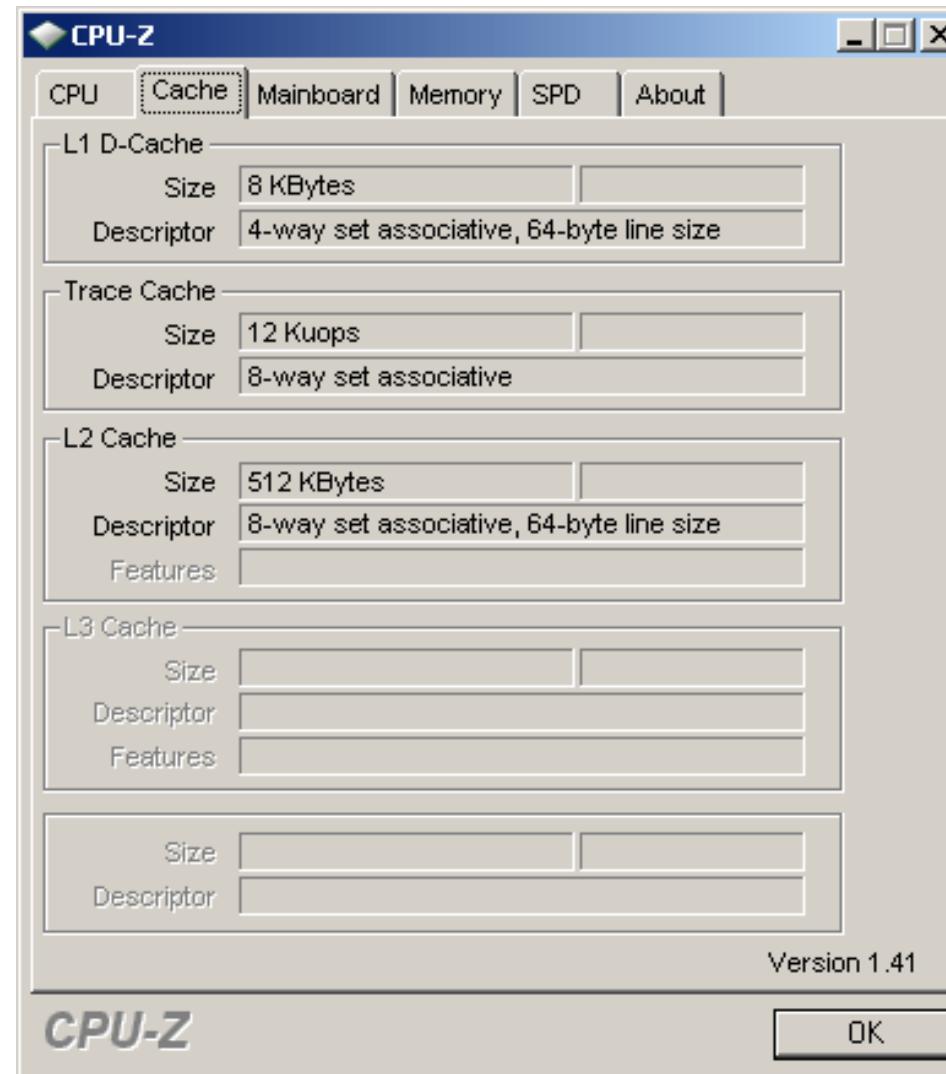
Information about your computer's CPU(s), FPU(s), cache(s) and other related devices.

Processor: CPU 1 [Processor 0, Core 0, Thread 0]

Item	Value
Co-Processor (FPU)	
Type	Built-in
Revision/Stepping	2 / 7 (9)
Processor Cache(s)	
Internal Data Cache	8kB Synchronous, Write-Thru, 4-way set, 64 byte line size, ...
Internal Trace Cache	12kB Synchronous, Write-Thru, 8-way set, 64 byte line size
L2 On-board Cache	512kB ECC Synchronous, ATC, 8-way set, 64 byte line size, ...
L2 Cache Multiplier	1/1x (2424MHz)
Upgradeability	
Socket/Slot	PGA 478
Upgrade Interface	ZIF Socket
Supported Speed(s)	3.20GHz+

The screenshot shows the SiSoftware Sandra application interface. The title bar reads "Processors - SiSoftware Sandra". Below it is a status message: "Information about your computer's CPU(s), FPU(s), cache(s) and other related devices.". A toolbar at the bottom contains various icons for file operations and navigation. The main content area displays processor information for "CPU 1 [Processor 0, Core 0, Thread 0]". A table lists various processor components and their values. A green dashed oval highlights the "Processor Cache(s)" section, which includes details for Internal Data Cache, Internal Trace Cache, L2 On-board Cache, and L2 Cache Multiplier. Another green dashed oval highlights the "Upgradeability" section, which includes details for Socket/Slot, Upgrade Interface, and Supported Speed(s). The "Processor Cache(s)" section is expanded, showing its sub-components and their specific settings.

CPU-Z: memòria cau



Com millorar el rendiment?

- Temps mitjà d'accés a les dades

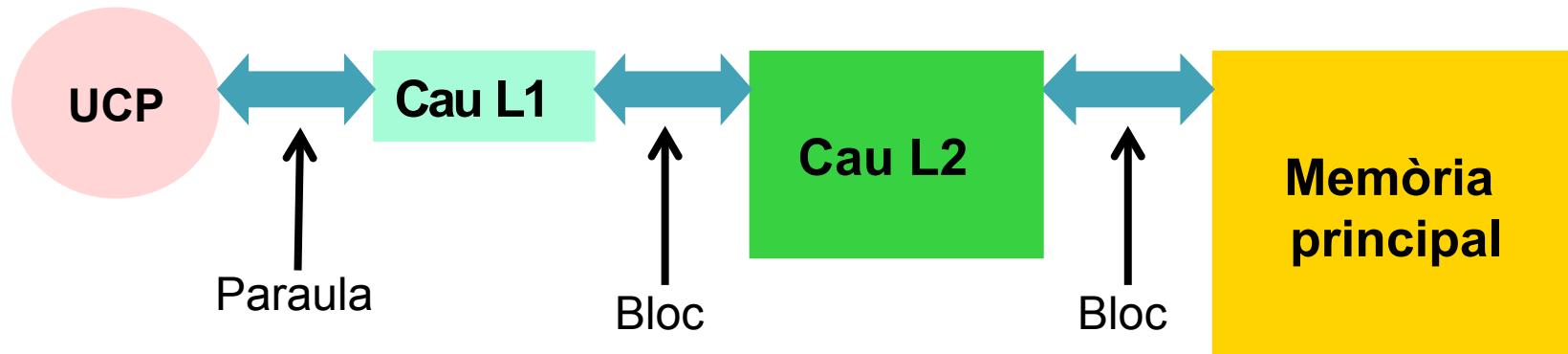
$$T_m = H \times T_{encert} + (1-H) \times T_{fallada}$$

- Estratègies de millora

- ✓ Reduir el temps d'encert: qüestions tecnològiques
- ✓ Reduir la taxa de fallades: organitzacions més eficients (afegir més vies)
- ✓ Reduir el temps per a resoldre la fallada
 - Caus multinivell
 - 2 i fins 3 nivells de cau
 - Noms usuals: L1, L2, L3
 - Busos més amples entre nivells de memòria cau

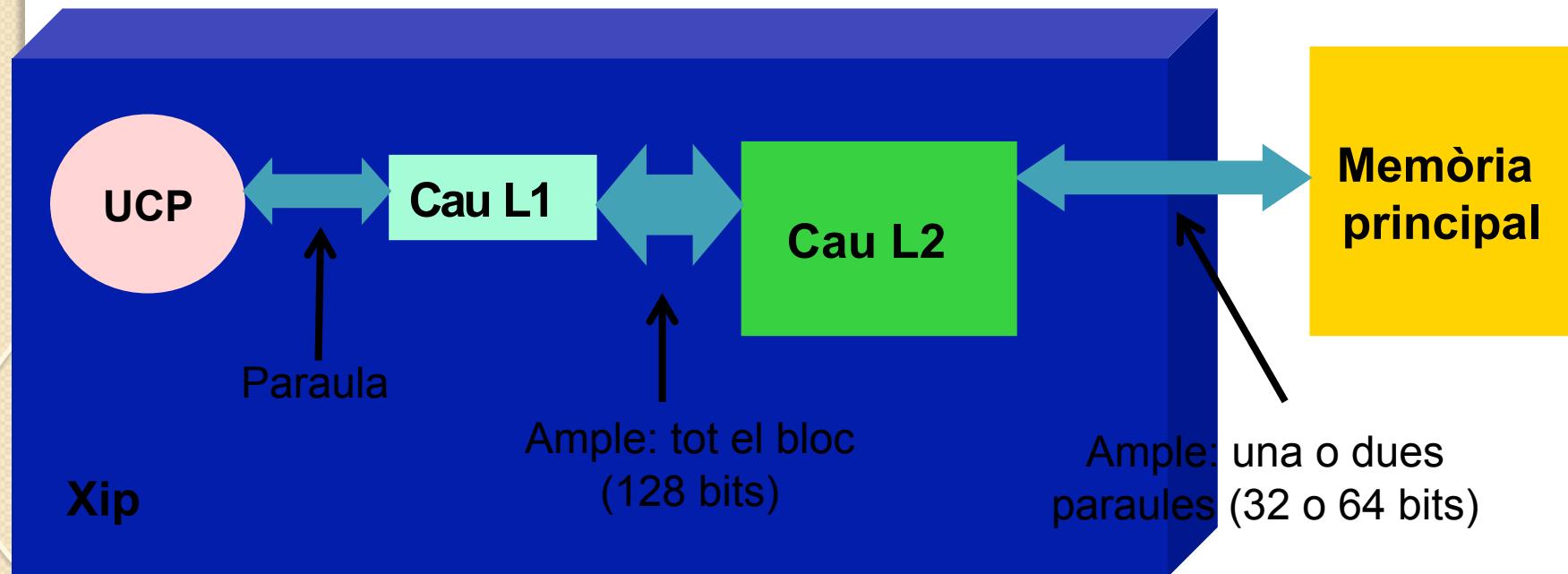
Memòries cau multinivell

- Cau L2: és molt més gran i bastant més lenta que L1
- Organitzacions independents
 - ✓ Les dues caus tenen organitzacions i polítiques independents
 - ✓ Exemple: Pentium 4
 - Cache L1: escriptura directa, no ubicació
 - Cache L2: escriptura posterior, ubicació



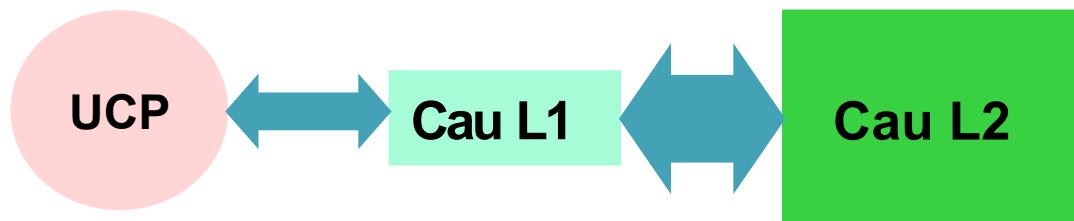
Memòries cau multinivell

- Ample de bus: si L2 es troba integrada amb el processador l'ample del bus L2-L1 pot ser tot el bloc (no hi ha problemes amb el nombre de pins)
 - ✓ L'ample del bus entre la memòria i les caus és d'una o dues paraules: transferència per blocs
 - ✓ Exemple: blocs de 4 paraules de 32 bits



Penalització en l'accés a la memòria cau L1

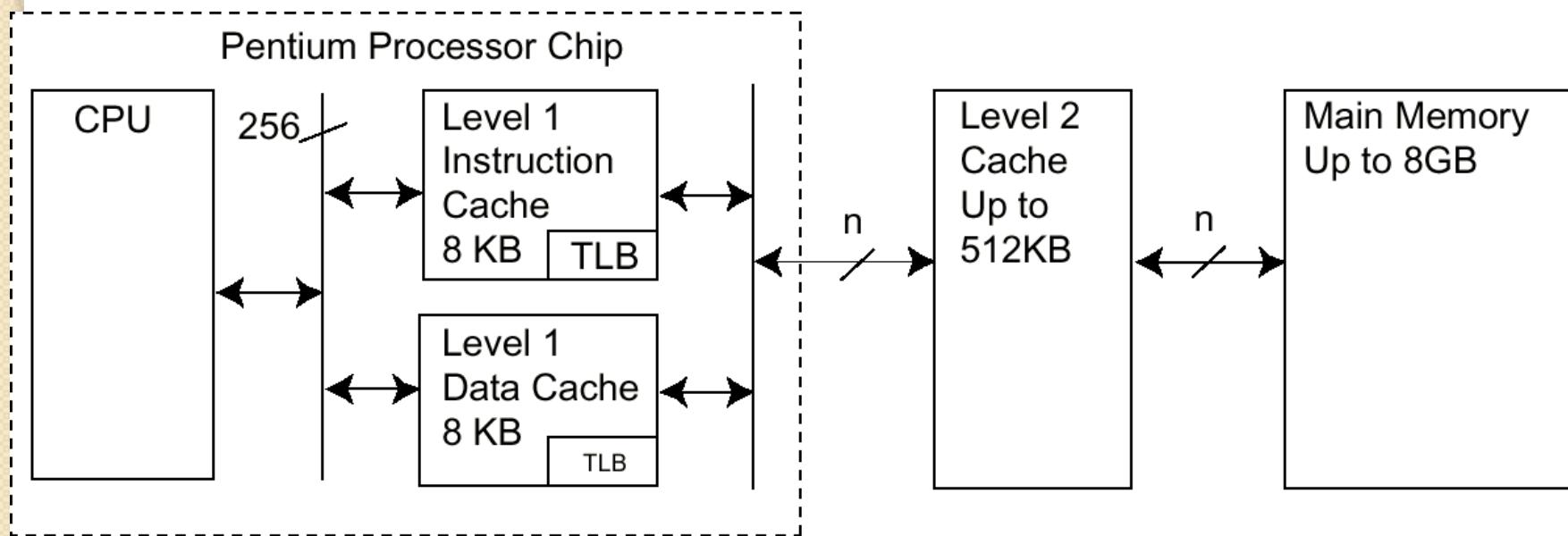
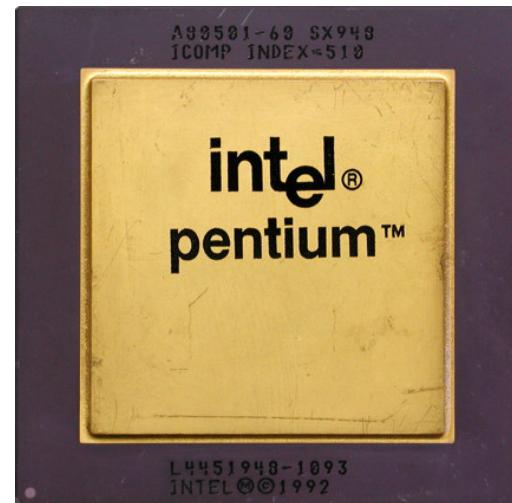
- Hi ha dues formes d'accedir a la cau L2 en relació amb l'accés a L1
 - ✓ Seqüencialment
 - En detectar fallada en L1 s'accedeix a L2
 - ✓ En paral·lel
 - S'accedeix a L1 i a L2 al mateix temps (mateix bus)
 - Si hi ha encert en L1 s'avorta l'operació en L2
 - En cas de fallada, la penalització de L1 és (cas d'encert) el temps d'accés a L2



Rendiment i paràmetres

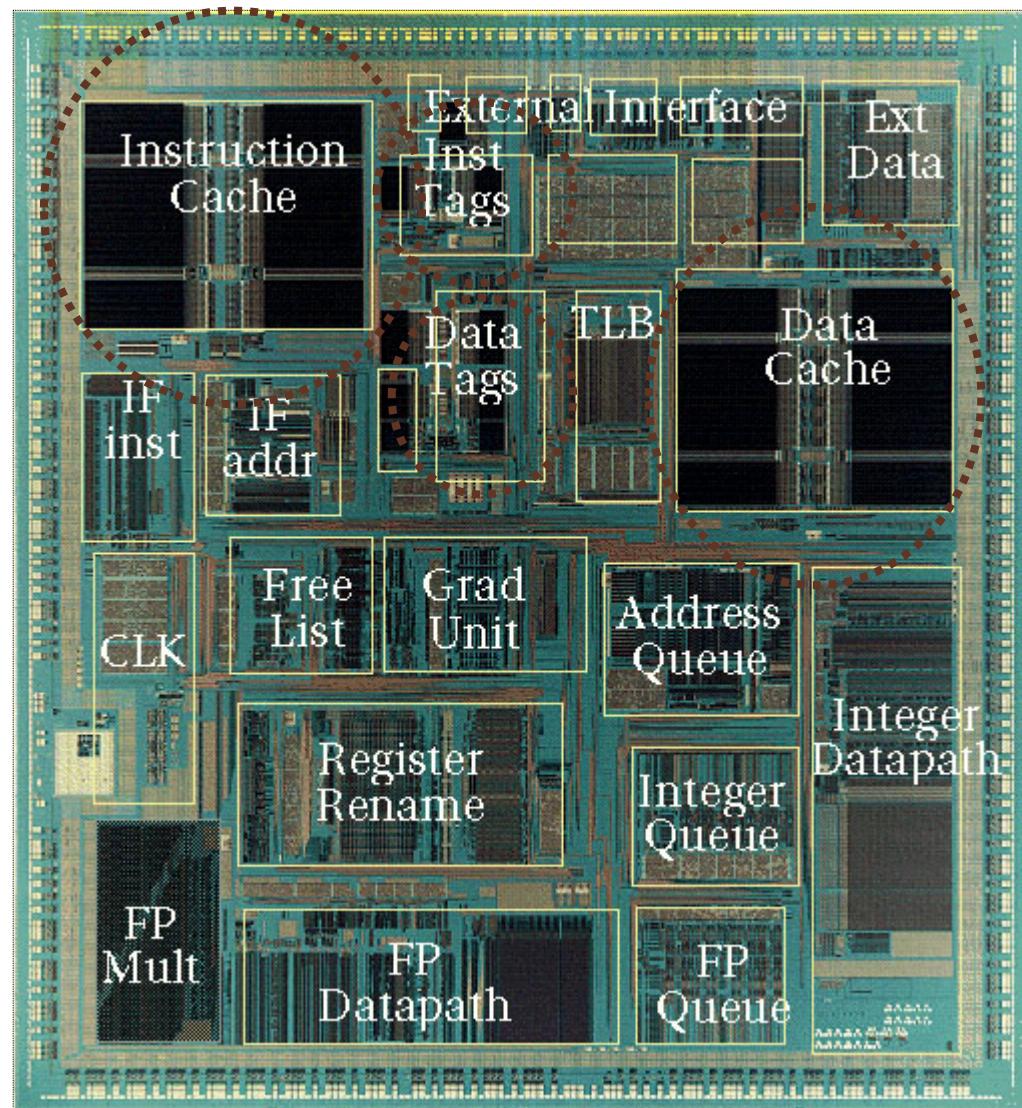
- El disseny d'una memòria cau contempla
 - ✓ el nombre de nivells, la capacitat, l'associativitat, el llarg del bloc, les polítiques d'escriptura i lectura
- El rendiment d'un disseny de memòria cau depén
 - ✓ de les característiques dels programes que executa el processador (tipus de dades, localitat espacial i temporal dels accessos)
 - ✓ de l'amplada de banda demandada pel processador (depén de la freqüència d'accessos a la memòria i de l'ample de paraula)
 - ✓ de l'amplada de banda de la memòria principal

Memòria cau en un Intel Pentium





- L1: codi (32 KB, 2 vies)
- dades (32 KB, 2 vies)
- L2: externa, 2 vies (0,5-16 MB)



Caus en processadors MIPS: evolució

ID-MHz	Nivell L1				Nivell L2		
	Codi	Dades					
R3000-33	32 KB	32 KB	Directa	Off chip			
R4000-100	8 KB	8 KB	Directa	On chip	1 MB	Directa	Off chip
R5000-200	32 KB	32 KB	2 vies	On chip	1 MB	Directa	Off chip
RM7000-250	Nivell L1				Nivell L2		
	16 KB	16 KB	4 vies	On chip	256 KB	4 vies	On chip
				Nivell L3			
RM7000-250				8 MB	Directa	Off chip	

Tendències en el disseny de la cau dels MIPS

- A més freqüència de rellotge del processador més configuracions possibles de cau
- L'existència de L2 permet fer LI més petita
- És habitual accedir a LI en un cicle de rellotge del processador
- L2 sol ser directa per a evitar als circuits comparadors treballar amb moltes etiquetes en paral·lel
- Les caus LI més recents soLEN usar
 - ✓ Escriptura posterior
 - ✓ 2 o 4 vies
 - ✓ La gestió es fa amb la instrucció màquina **cache**

3. La memòria virtual

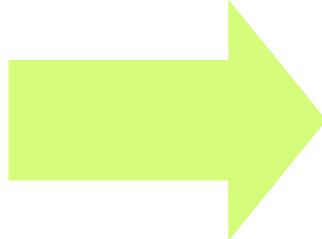
- Concepce i utilitat
- Adreçament virtual
- Traducció de les adreces

La memòria virtual: utilitat

- Si no hi ha memòria virtual el màxim espai que pot adreçar el processador és la memòria principal o memòria física
- L'ús de memòria virtual permet que els programes disposen d'un espai d'adreces major que l'espai de memòria física
 - ✓ En un moment donat, la major part de la MV adreçable es troba a disc
 - ✓ Per a que el processador puga accedir als programes i les dades, aquests han de trobar-se en memòria
 - ✓ El sistema operatiu s'encarrega de realitzar les gestions pertinents (assignatura SOP)

Visió del programador

- El programador no es preocupa de la quantitat de memòria física del computador
- Processador MIPS R2000
 - ✓ 32 bits d'adreça: 4 GB d'espai adreçable dividit en dues parts iguals (usuari i sistema)
 - ✓ La memòria principal sol ser de menor capacitat



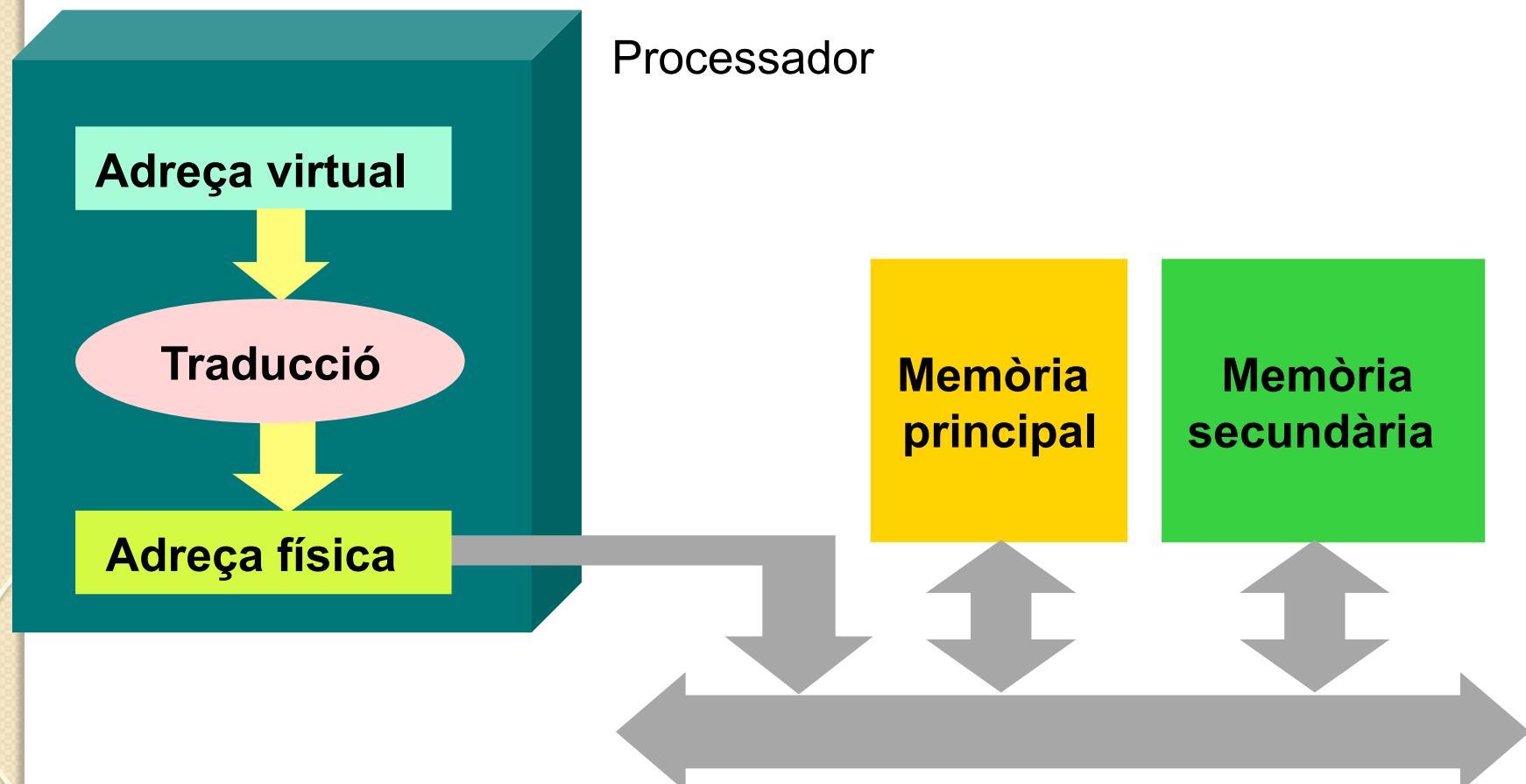
Implementació física de l'espai adreçable gestionada pel SO

Adreces virtuals i físiques

- Quan s'utilitza MV, es pot distingir:
 - ✓ **Adreça virtual**
 - Adreça d'un objecte utilitzada pel processador
 - L'objecte pot estar en MP o en MS
 - ✓ **Adreça física**
 - Adreça d'un objecte en MP
 - Són les adreces emeses pel bus d'adreces
- Cal obtenir l'adreça física a partir de l'adreça virtual per tal d'accendir a l'objecte
 - ✓ Aquesta correspondència es realitza basant-se en **pàgines** de memòria
 - La MV es divideix en pàgines
 - La MP es divideix en marcs de pàgina
 - En cada marc es pot emmagatzemar una pàgina

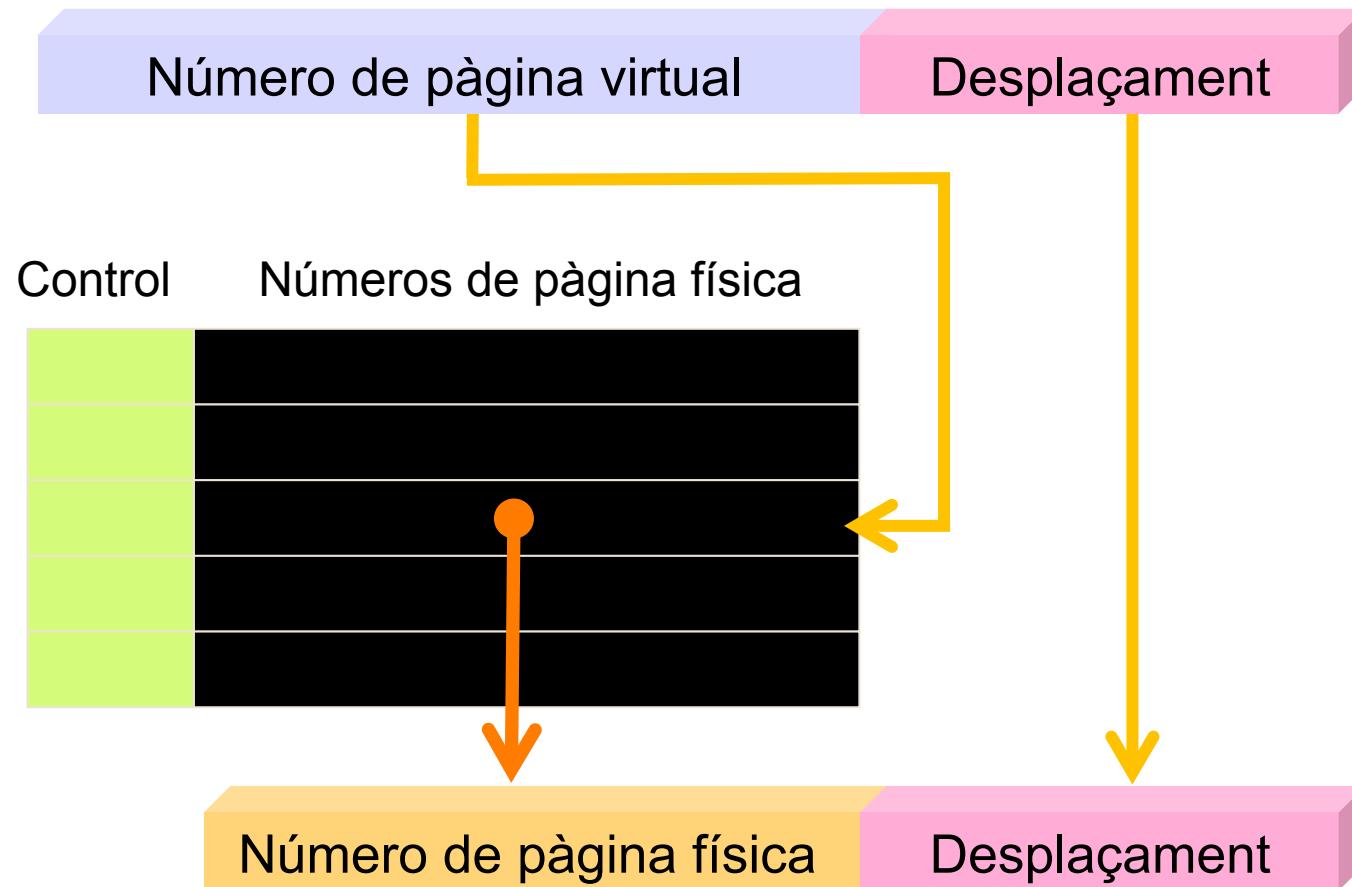
Adreces virtuals i físiques

- Els programes treballen amb adreces virtuals
 - ✓ Abans d'eixir al bus s'han de traduir a adreces físiques

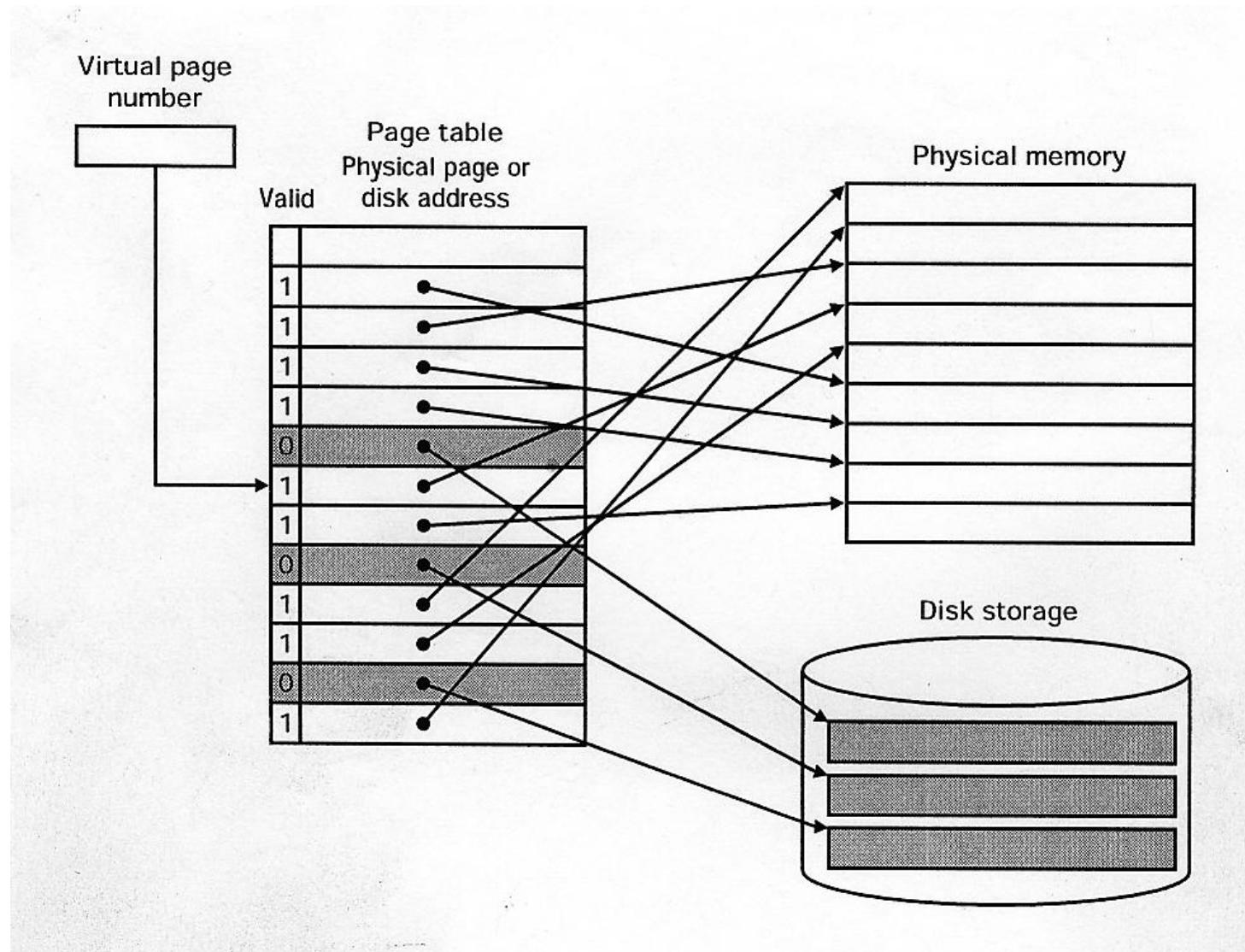


Traducció mitjançant una taula de pàgines

- El número de pàgina virtual serveix d'índex per a accedir a una taula i recuperar el número de pàgina física



Interpretació de la informació en la TP

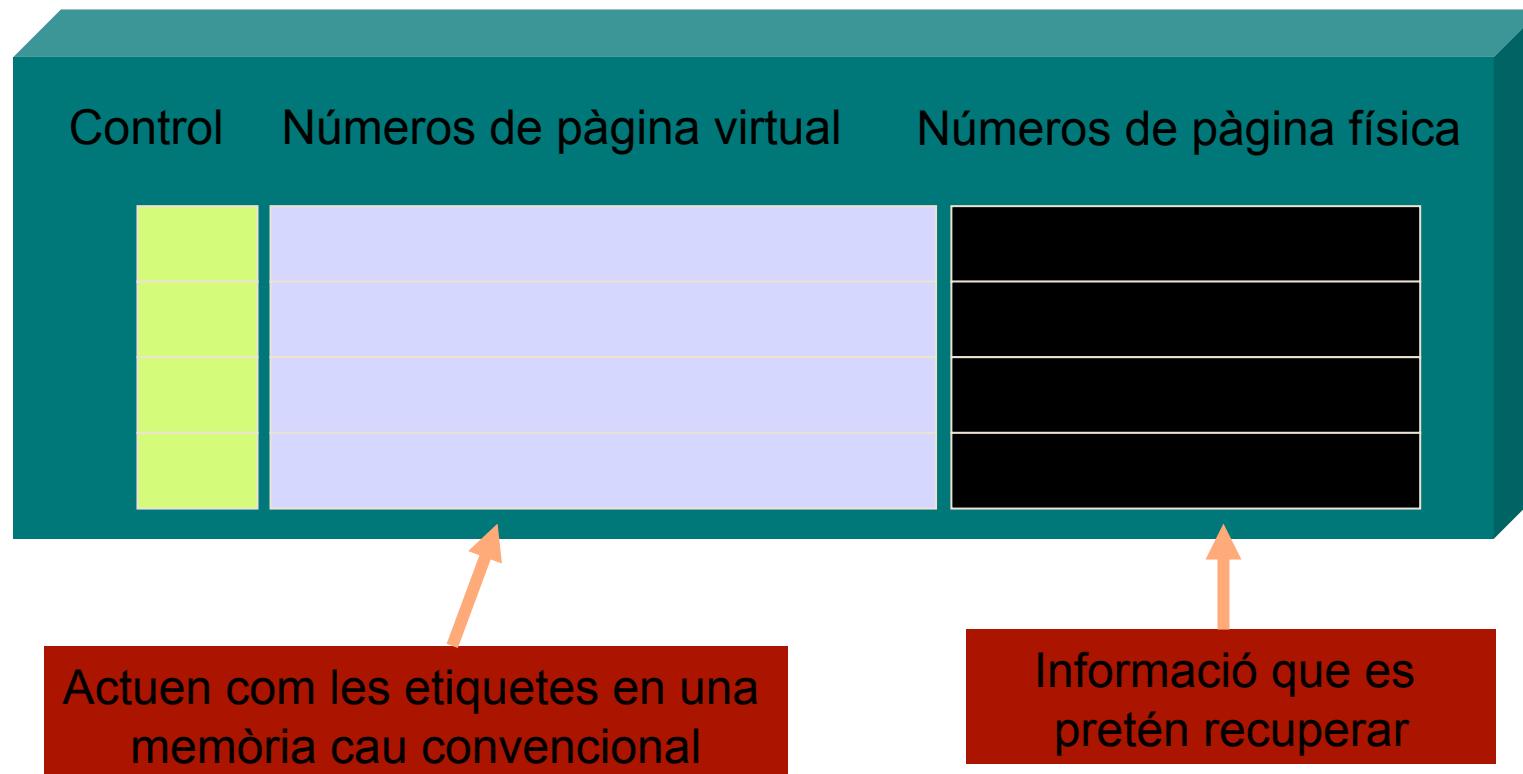


Traducció d'adreces: característiques

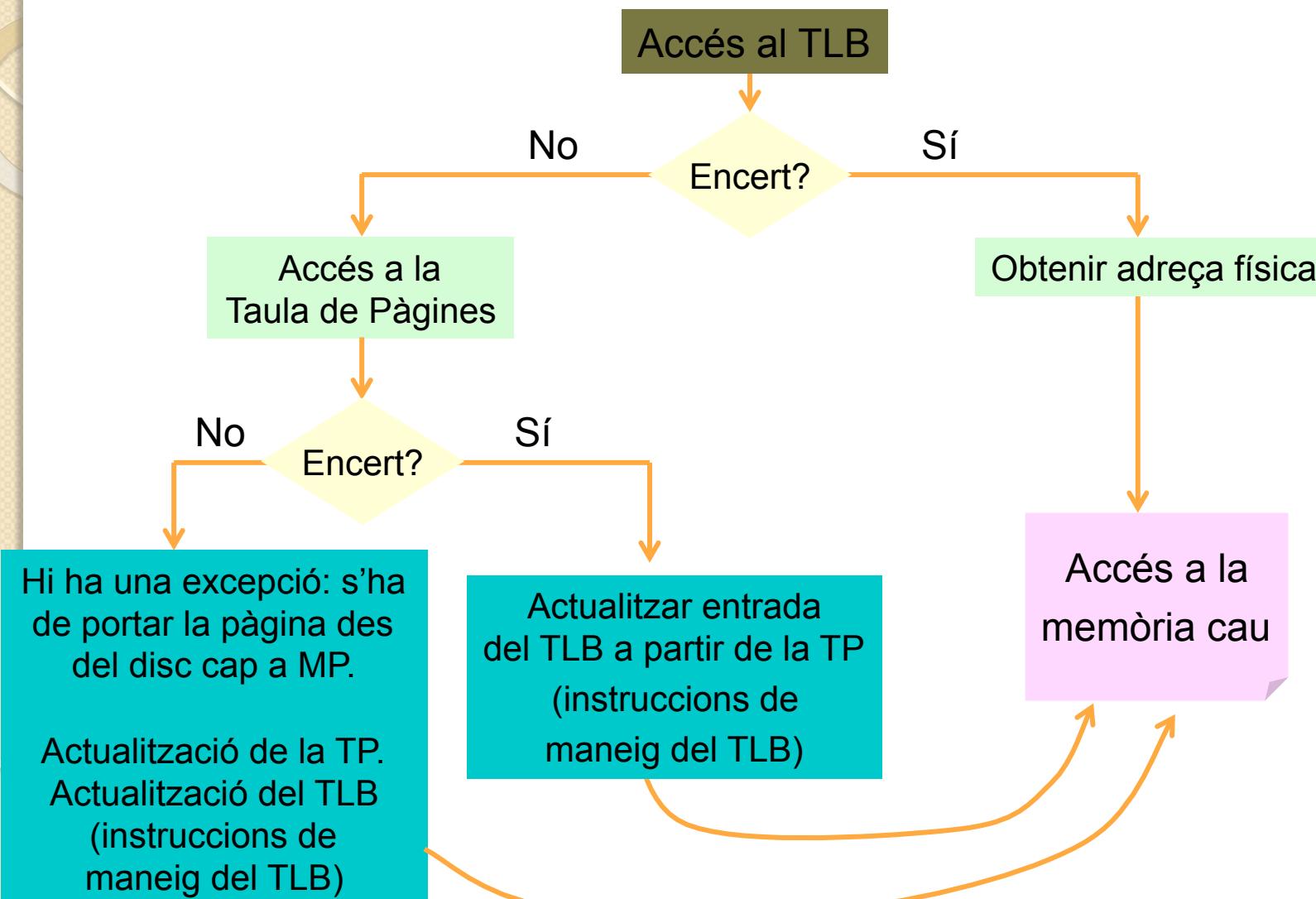
- La taula de pàgines s'emmagatzema en la memòria principal
- Hi ha una entrada per cada pàgina virtual
- Informació emmagatzemada en la taula:
 - ✓ Bit de vàlid: indica si la pàgina està present en MP o en disc
 - ✓ Número d'adreça física (si escau)
 - ✓ Permisos de lectura, escriptura, etc
- Problema: cal molt espai per a emmagatzemar la taula i s'utilitzen molt poques pàgines
 - ✓ Solució: organització mitjançant taules amb estructura jeràrquica que s'actualitzen dinàmicament

Traducció ràpida d'adreces: TLB

- Calen dos accisos a MP: lentitud
- Solució: implementar una “cau” de la taula de pàgines en el processador: TLB (*Translation Lookaside Buffer*)

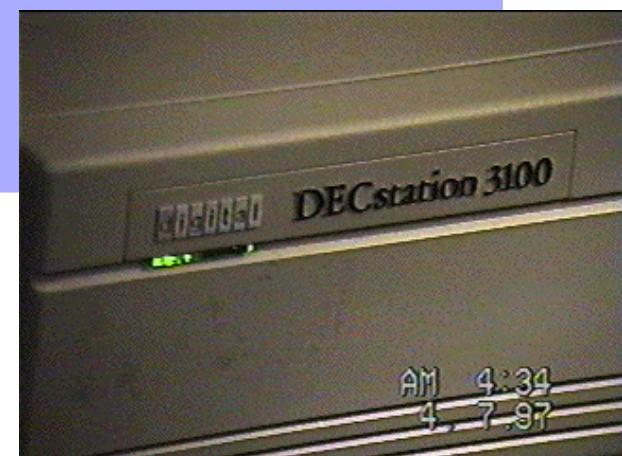


Relació entre Taula de Pàgines (TP) i TLB



4. Exemple conjunt de TLB i cau

- El computador DECstation 3100



La memòria virtual en el MIPS R2000

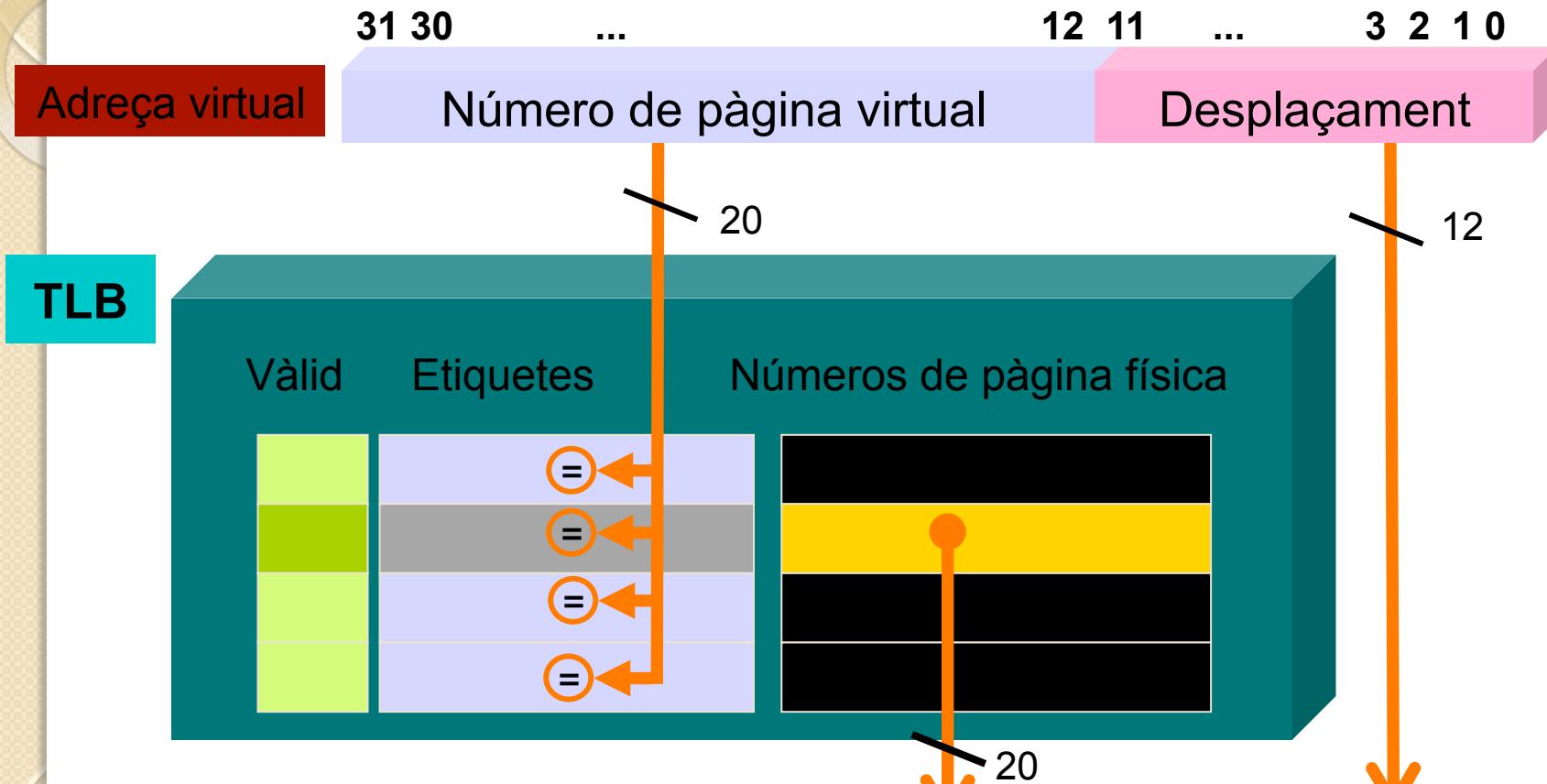
- *Memory Management Unit (MMU)*
- Característiques de l'adreçament de memòria
 - ✓ Adreces físiques: 32 bits
 - ✓ Adreces virtuals: 32 bits
 - ✓ Grandària de les pàgines: 4 KB (2^{12} bytes)
- TLB situat dintre del processador (*on chip*)
 - ✓ Traducció d'adreces de codi i de dades
 - ✓ Correspondència completament associativa
 - ✓ Té 64 entrades: es pot traduir i accedir ràpidament a un total de 64 pàgines de 4 KB de memòria virtual
 - ✓ Cada entrada té 64 bits de llargària
 - Control: bits de vàlid, cacheable, etc.

Control

Pàgina virtual (20)

Marc de pàgina (20)

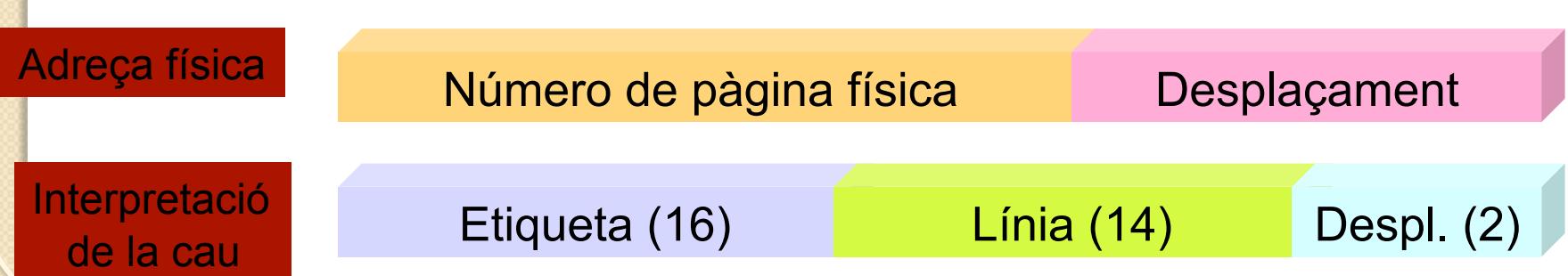
Traducció de l'adreça virtual en el TLB



Aquesta adreça arribarà a la memòria cau L1

La memòria cau en el MIPS R2000

- Configuració en la DECstation 3100
 - ✓ Nivell L1 extern al processador
 - Caus separades de dades i codi
 - ✓ Correspondència directa
 - ✓ Grandària de bloc: 4 bytes (una paraula)
 - ✓ Polítiques d'escriptura
 - Encerts: escriptura directa (*write through*)
 - Fallades: no ubicació (*no write allocate*)
- La cau representada (una de les dues) té 16 KB de capacitat



Interpretació de l'adreça física per la cau

Adreça física

Número de pàgina física

Desplaçament

Etiqueta

Línia

Despl.

16

14

2

Vàlid

Etiquetes

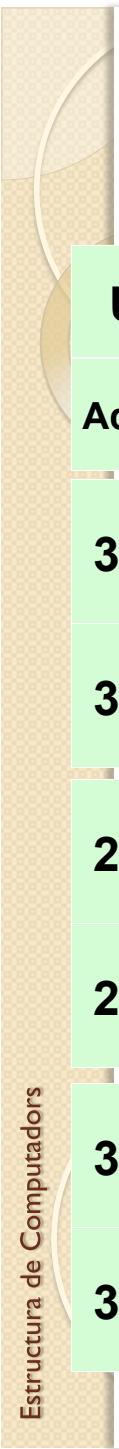
Dades

Cau

Encert en cau

Informació trobada si encert

Exercici: ompliu les taules següents



UCP		Memòria Cau				
Adr.	W	Grandària	vies	Grandària Línia	# Línies	#cjts
32	32	32KB	2	32B	1024	512
32	32	8KB	4	32B	256	64
24	32	32KB	1	32B	1024	1024
24	32	4KB	128	32B	128	1
36	64	32KB	2	64B	512	256
36	64	8KB	4	64B	128	32

Estructura de Computadors

Bits d'adreça		
etiqueta	conjunt	desp
18	9	5
21	6	5
9	10	5
19	0	5
22	8	6
25	5	6