

Estructura de Computadores

Tema 5. Memoria

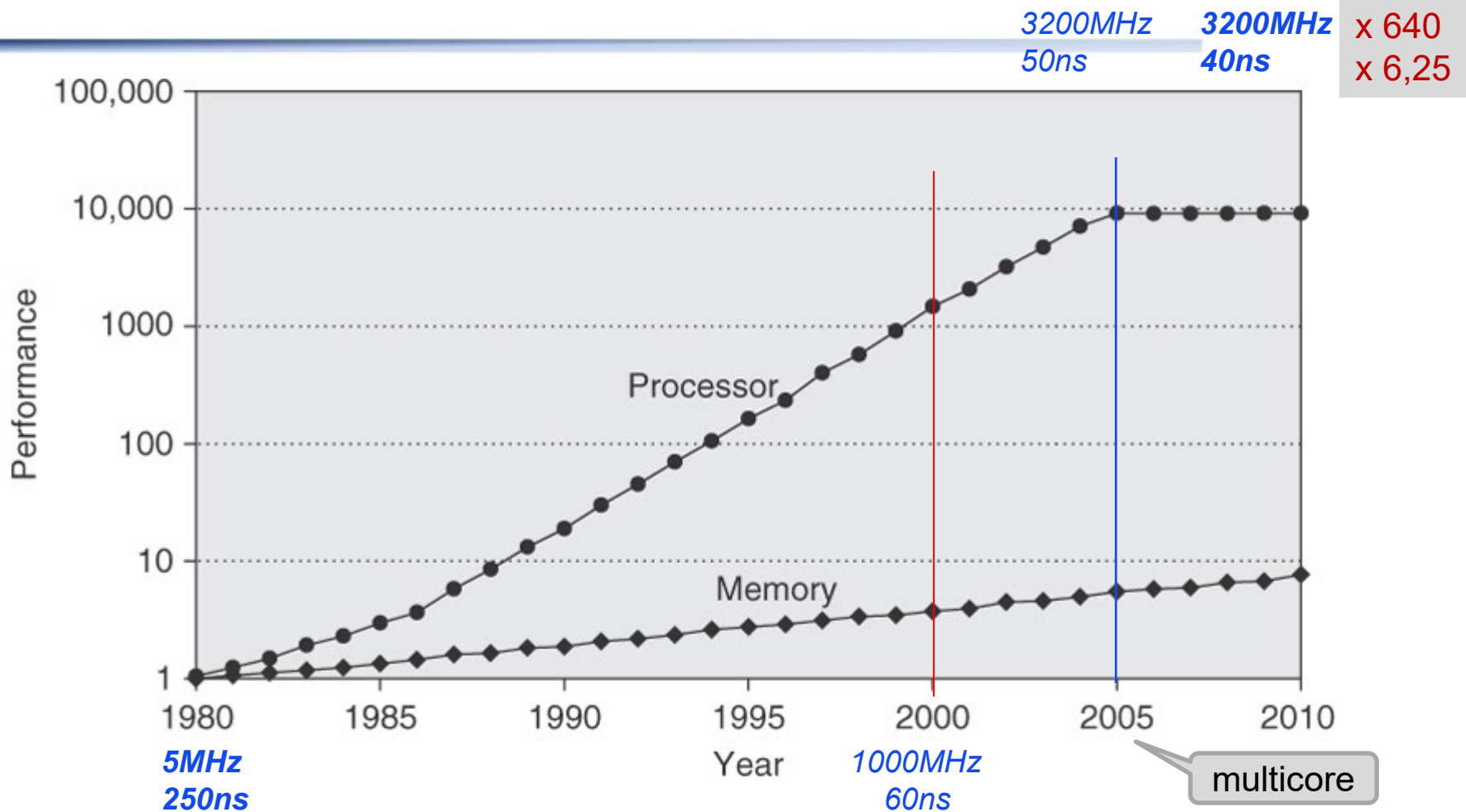



Figure 2.2 Starting with 1980 performance as a baseline, the gap in performance, measured as the difference in the time between processor memory requests (for a single processor or core) and the latency of a DRAM access, is plotted over time.

(Hennessy & Patterson 5th ed)

- **Jerarquía de memoria**
 - Organización y funcionamiento
 - Concepto de traza de ejecución. Ejemplos
 - Proximidad de referencias
 - Parámetros característicos
- **Memorias caché**
 - Políticas de ubicación y escritura
- Memoria virtual
 - Traducción de direcciones
 - Gestión de fallos
 - Paginación

- García Clemente, M.I. Sistema de Memoria. Facultad de Informática, 2003 y 2016.
- Hennessy, J.L.; Patterson, D. A. Computer Architecture: A quantitative Approach. Morgan-Kaufmann. 2007. 4ª edición.
- Patterson, D. A. Hennessy, J.L Estructura y Diseño de Computadores . 2011. 4ª edición.
- Bryant, R.E.; O'Hallaron, D.R. Computer Systems: A programmer's perspective. Pearson. 2011. 2ª edición.
- García Clemente y otros. Estructura de computadores. Problemas resueltos. RAMA, 2006. 1ª edición.

- Objetivos del diseño eficiente del Sistema de memoria:
 - **Velocidad** de respuesta \approx Necesidades de la CPU
Almacenamiento de instrucciones y datos
 factor determinante en la velocidad de ejecución:
p.ej., 3,3 GHz \rightarrow 0,3 ns/ciclo
 - **Capacidad** de almacenamiento “ilimitada” y **coste** razonable

\uparrow velocidad + \uparrow capacidad + \downarrow coste

- **Objetivos del diseño eficiente del Sistema de memoria:**

- **Velocidad** de respuesta \approx Necesidades de la CPU

Almacenamiento de instrucciones y datos



factor determinante en la velocidad de ejecución:

p.ej., 3,3 GHz \rightarrow 0,3 ns/ciclo

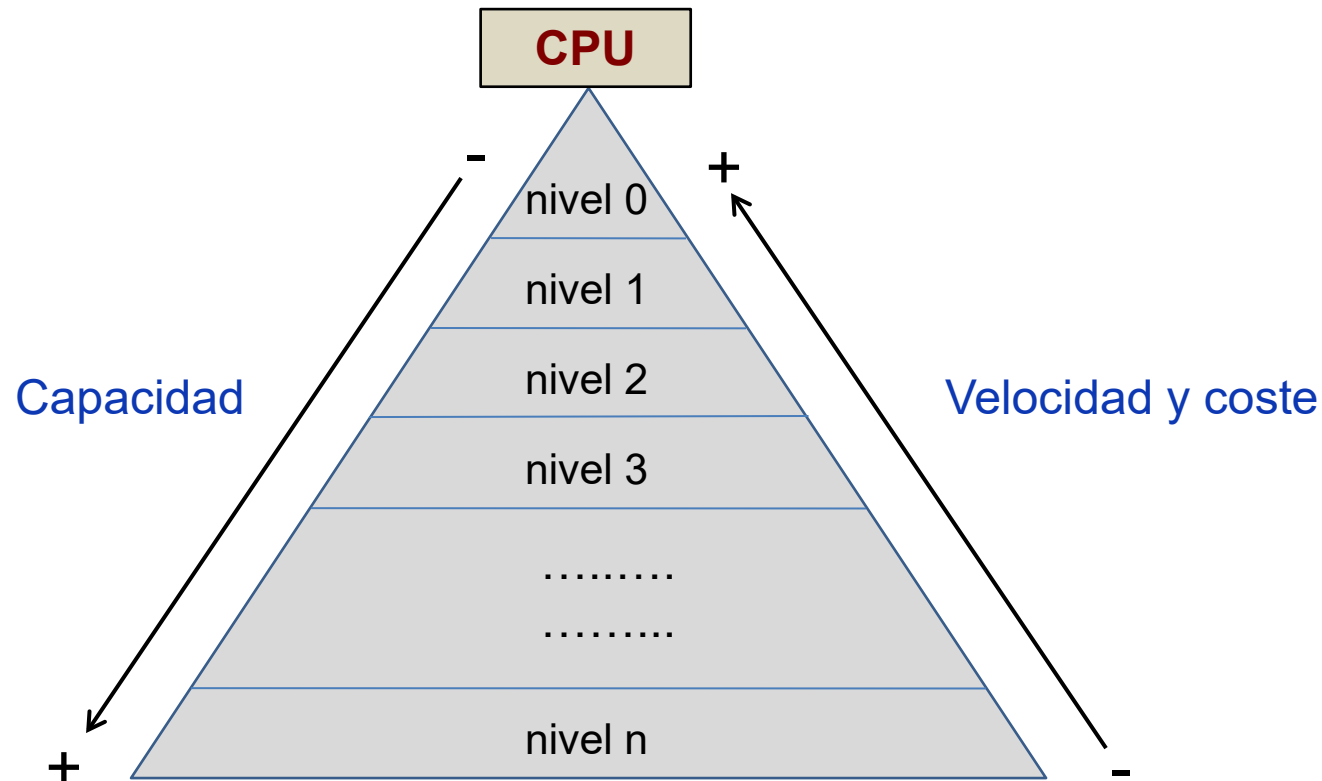
- **Capacidad** de almacenamiento “ilimitada” y **coste** razonable

\uparrow *velocidad* + \uparrow *capacidad* + \downarrow *coste*

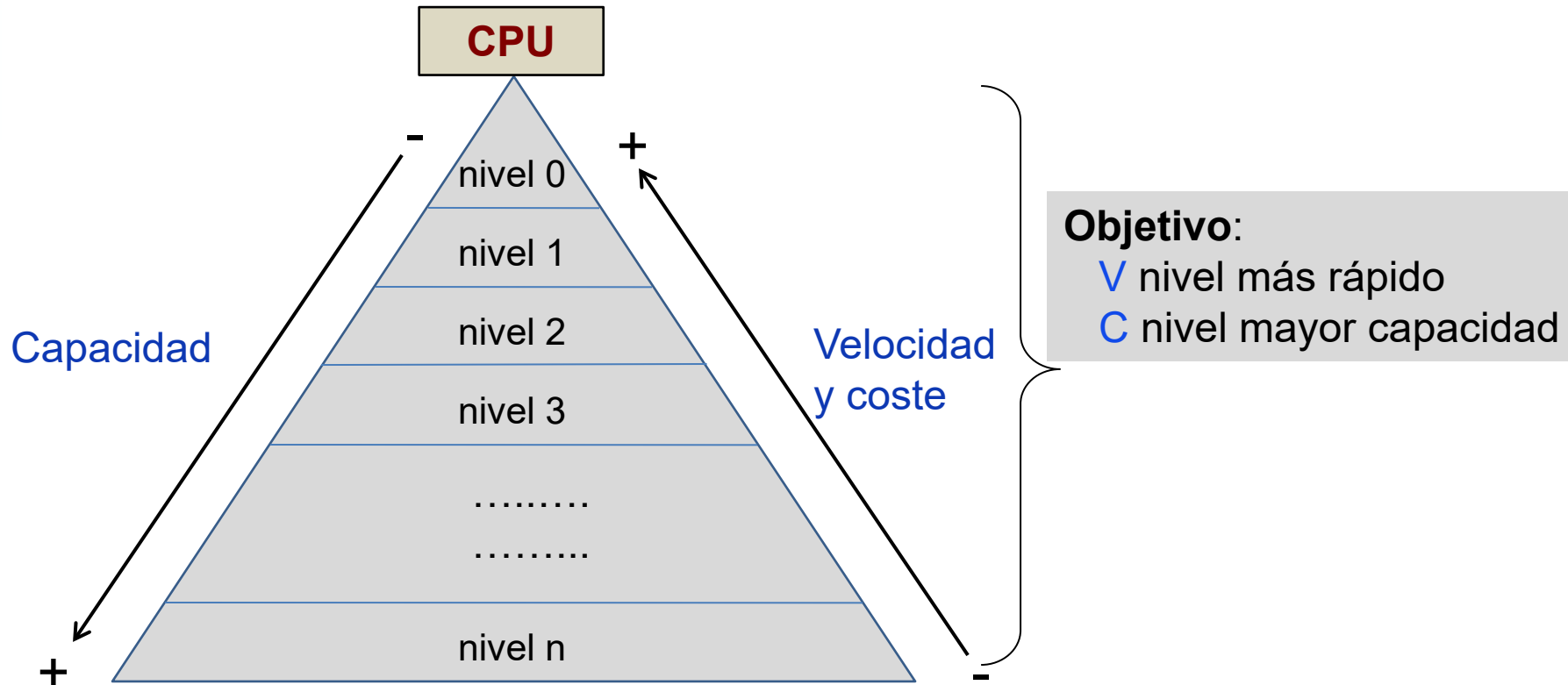
- **Jerarquía de Memorias:**

Múltiples dispositivos de almacenamiento con diferente:
velocidad, capacidad y coste
organizados de forma jerárquica.

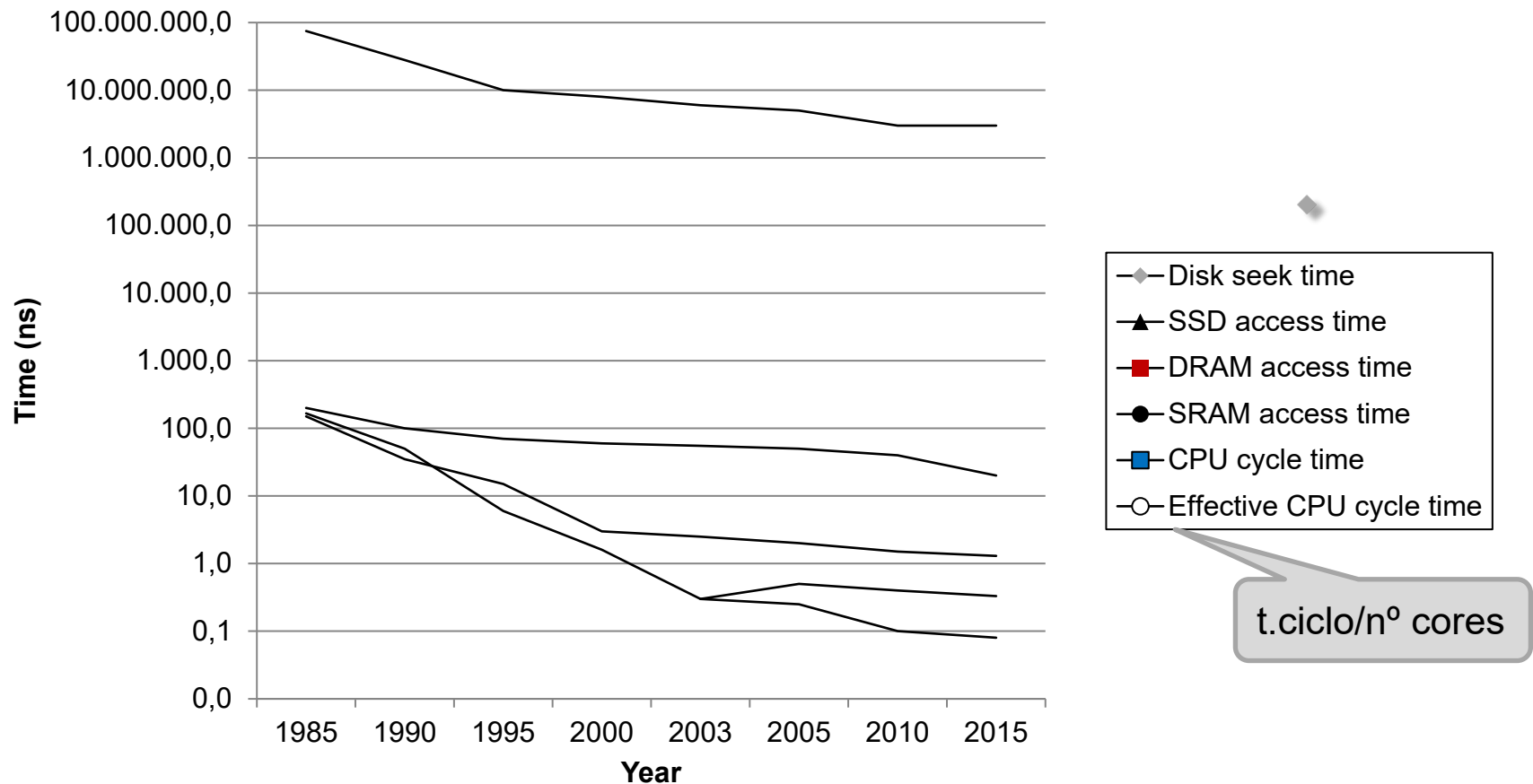
Jerarquía de Memorias: Esquema genérico



Jerarquía de Memorias: Esquema genérico



Jerarquía de memorias: Tecnologías



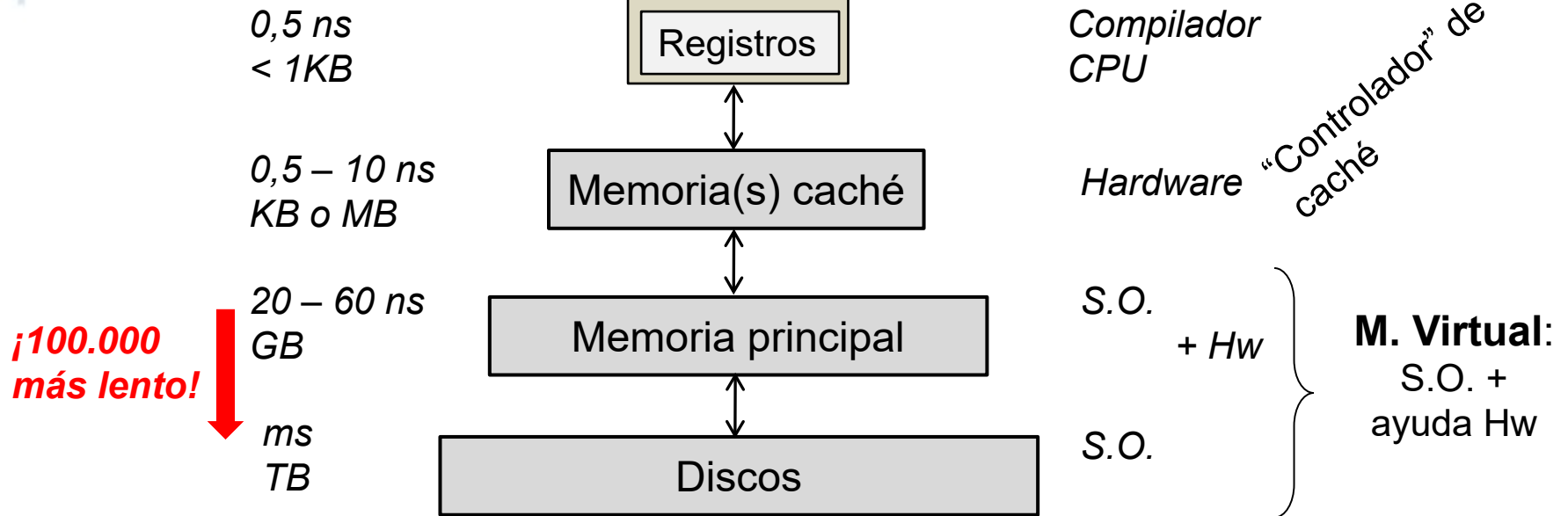
Bryant, R.E. & O'Hallaron, 2016

Jerarquía de memorias típica



Capacidad y
Tiempo de acceso

Gestión

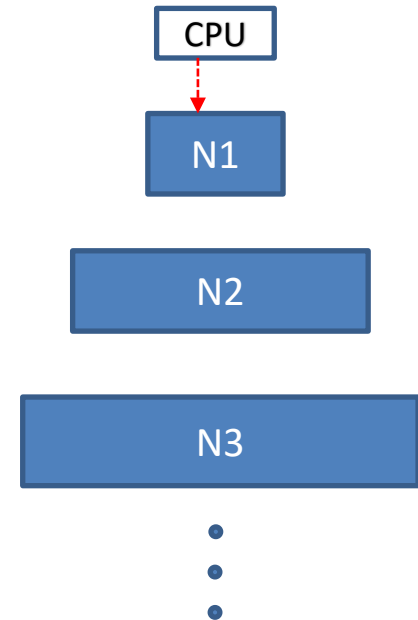


Propiedad de Inclusión

Funcionamiento de la jerarquía



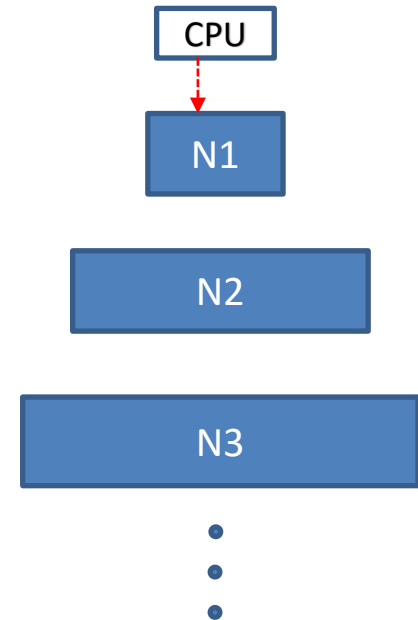
- La CPU solicita acceso a una posición de memoria (*fetch*, *load* o *store*)
- **Se comprueba si la información está en el nivel 1**



Funcionamiento de la jerarquía



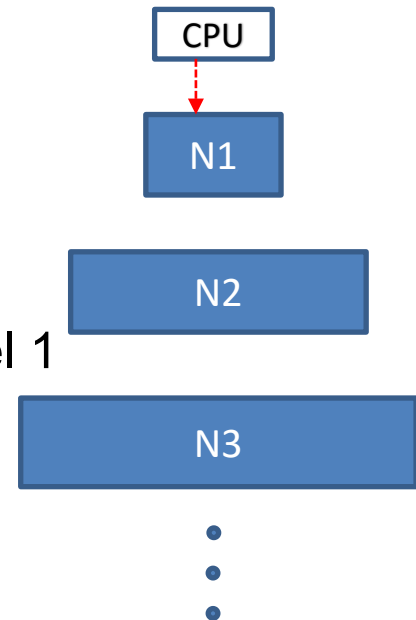
- La CPU solicita acceso a una posición de memoria (*fetch* o *load/store*)
- Se comprueba si la información está en el nivel 1
- **Si está:**
 - Se sirve el acceso desde el nivel 1 (**el más rápido**)



Funcionamiento de la jerarquía



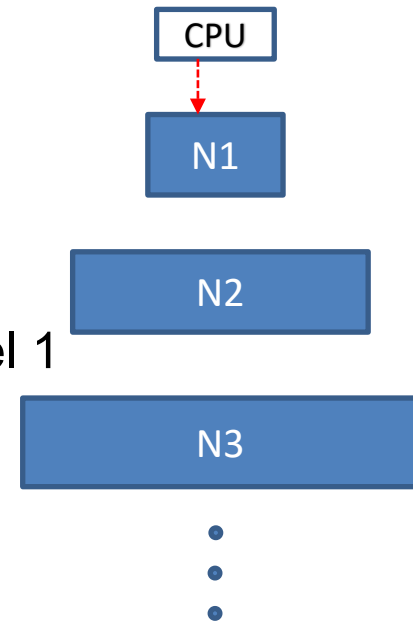
- La CPU solicita acceso a una posición de memoria (*fetch* o *load/store*)
- Se comprueba si la información está en el nivel 1
- Si está:
 - Se sirve el acceso desde el nivel 1 (**el más rápido**)
- **Si no está:**
 - Se comprueba si está la información en el nivel 2
 - **Si está:**
 - Se transfiere la información desde el nivel 2 al nivel 1
 - El nivel 1 transfiere la información a la CPU



Funcionamiento de la jerarquía

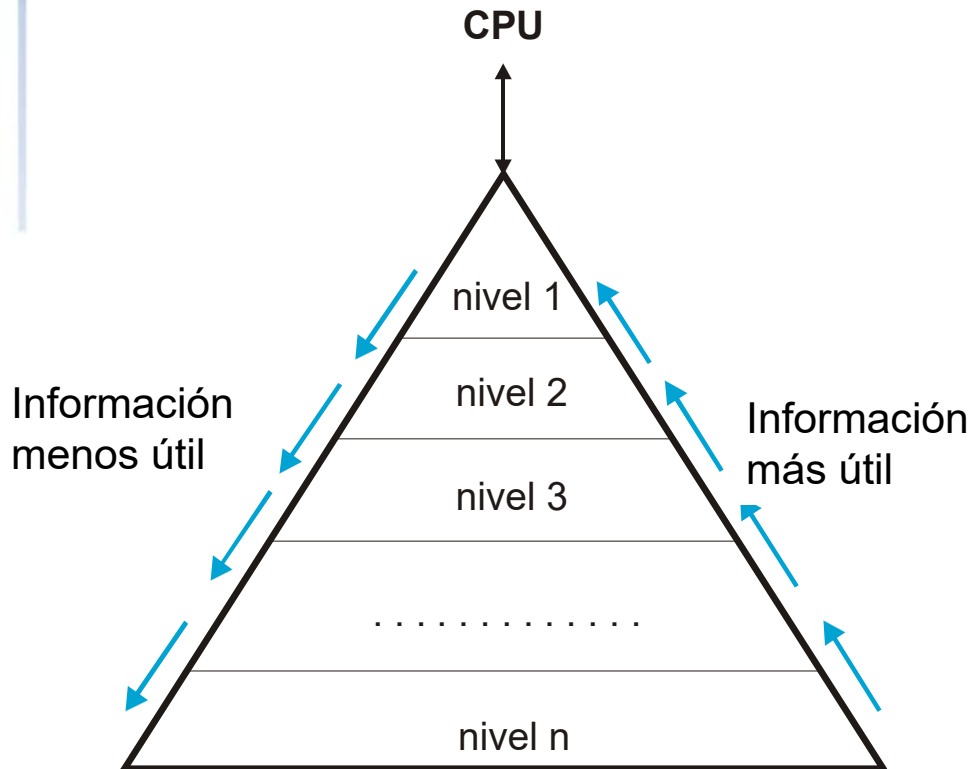


- La CPU solicita acceso a una posición de memoria (*fetch* o *load/store*)
- Se comprueba si la información está en el nivel 1
- Si está
 - Se sirve el acceso desde el nivel 1 (**el más rápido**)
- **Si no está:**
 - Se comprueba si está la información en el nivel 2
 - Si está:
 - Se transfiere la información desde el nivel 2 al nivel 1
 - El nivel 1 transfiere la información a la CPU
 - **Si no está:**
 - Se comprueba si está la información en el nivel 3



Nota: se copia siempre un conjunto de direcciones consecutivas: bloques de caché (unos cuantos bytes) o páginas (de orden de KB)

Funcionamiento de la jerarquía



- **Decisiones de diseño:**
 - Políticas de **extracción**
 - Políticas de **ubicación**
 - Políticas de **reemplazo**
 - Políticas de **escritura**
 - **Traducción de direcciones**

Jerarquía de memorias: Motivación



Proximidad de Referencias (“locality”)

Los programas tienden a hacer referencia a datos e instrucciones “próximos” a los recientemente utilizados.

- **Temporal**: Mismas direcciones a las que se ha accedido en un pasado reciente.
- **Espacial**: Direcciones próximas a las que se ha accedido recientemente → **Secuencial**: direcciones consecutivas

Ejemplo :

```
sum = 0;  
for (i=0; i<1000; i++)  
    sum = sum + v[i];
```


Proximidad de referencias



Traza de un programa: secuencia de direcciones a las que accede durante su ejecución.

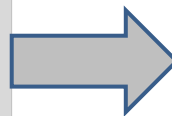
4 bytes/instr., 4 bytes/v[i] V[i] desde dir 10000; código desde dir 0,
regs: i, sum

Ejemplo

```
sum = 0;  
for (i=0; i<1000; i++)  
    sum = sum + v[i];
```

ORG 0

```
        LD  .R3, #0  
        LD  .R1, #1000  
        LD  .R2, #10000  
bucle:  ADD  .R3, [.R2]  
        ADD  .R2, #4  
        DEC  .R1  
        BNZ  $bucle
```



```
and r3, r0, 0  
or r1, r0, 0  
or r2, r0, low(v)  
or.u r2, r2, high(v)  
buc: ld r4, r2, 0  
    add r3, r3, r4  
    add r2, r2, 4  
    add r1, r1, 1  
    cmp r15, r1, 1000  
    bb1 lt, r15, $buc
```

Proximidad de referencias



- La proximidad **temporal** aconseja
Enviar el contenido de la dirección referenciada
al nivel más próximo a la CPU

Proximidad de referencias



- La proximidad **temporal** aconseja
Enviar el contenido de la dirección referenciada
al nivel más próximo a la CPU
- La proximidad **espacial** (secuencial) aconseja
Enviar también el contenido de las direcciones cercanas

Proximidad de referencias



- La proximidad **temporal** aconseja
Enviar el contenido de la dirección referenciada
al nivel más próximo a la CPU
- La proximidad **espacial** (secuencial) aconseja
Enviar también el contenido de las direcciones cercanas



Bloque:

- Agrupación de palabras de memoria consecutivas
- Unidad de información que se transfiere entre dos niveles consecutivos de la jerarquía
- Unidad de información que puede, o no, estar presente en un nivel de la jerarquía (indivisible)

Jerarquía de memorias: Terminología

Para un determinado nivel “j” de la jerarquía:

- **Acierto**: La información a la que hace referencia la CPU se encuentra en dicho nivel
 - **Tasa de aciertos** (Hr_j): Fracción de los accesos a memoria con acierto en dicho nivel ¿De que depende?
 - **Tiempo de acierto** (T_j): Tiempo de acceso a dicho nivel

Jerarquía de memorias: Terminología

Para un determinado nivel “j” de la jerarquía:

- **Acierto**: La información a la que hace referencia la CPU se encuentra en dicho nivel
 - **Tasa de aciertos** (Hr_j): Fracción de los accesos a memoria con acierto en dicho nivel
 - **Tiempo de acierto** (T_j): Tiempo de acceso a dicho nivel
- **Fallo**: La información no se encuentra en dicho nivel, hay que buscarla en el siguiente
 - **Tasa de fallos** (Mr_j) = $1 - Hr_j$
 - **Tiempo de fallo** (T_{fallo}) =
Tiempo de acceso (T_j) + Tiempo de penalización

Rendimiento de la jerarquía



- **Tiempo medio de acceso** o tiempo efectivo

$$= Hr_1 \times T_1 + (1-Hr_1) \times T_{\text{fallo}}$$

Rendimiento de la jerarquía



- Tiempo medio de acceso o tiempo efectivo

$$= Hr_1 \times T_1 + (1 - Hr_1) \times T_{\text{fallo}}$$

Ejemplo:

- 2 niveles: Mca (T = 2ns) y Mp (T=40ns)
- Bloques de 4 palabras (4B/palabra)
- Tiempo de fallo = xx ns

Tiempo medio de acceso = ??

Rendimiento de la jerarquía



- Tiempo medio de acceso o tiempo efectivo

$$= Hr_1 \times T_1 + (1 - Hr_1) \times T_{\text{fallo}}$$

Ejemplo: Solución

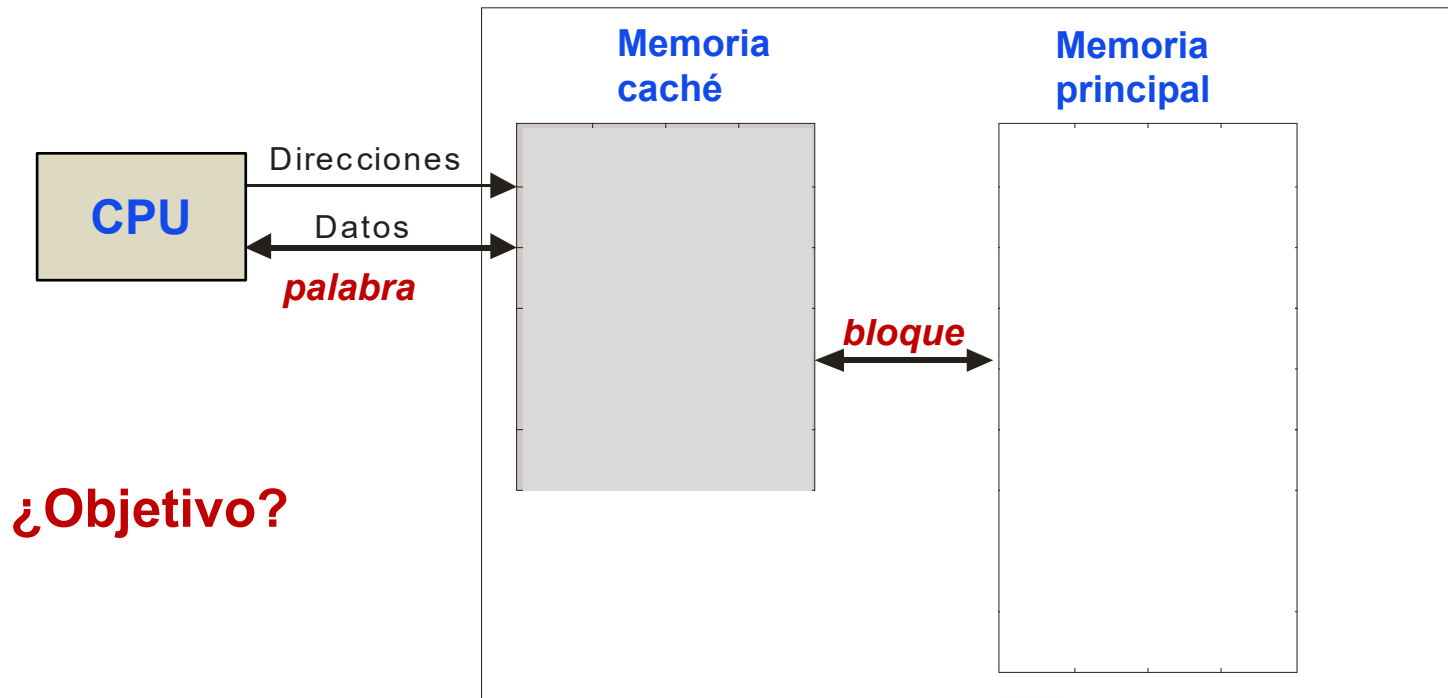
- 2 niveles: Mca (T = 2ns) y Mp (T=40ns)
- Bloques de 4 palabras (4B/palabra)
- Tiempo de fallo = **2 + 4x40 + 2 = 164 ns**
- **Hr = 96,38%**

Tiempo medio de acceso = 7,86ns

¡ 5 veces más rápido que con solo Mp !

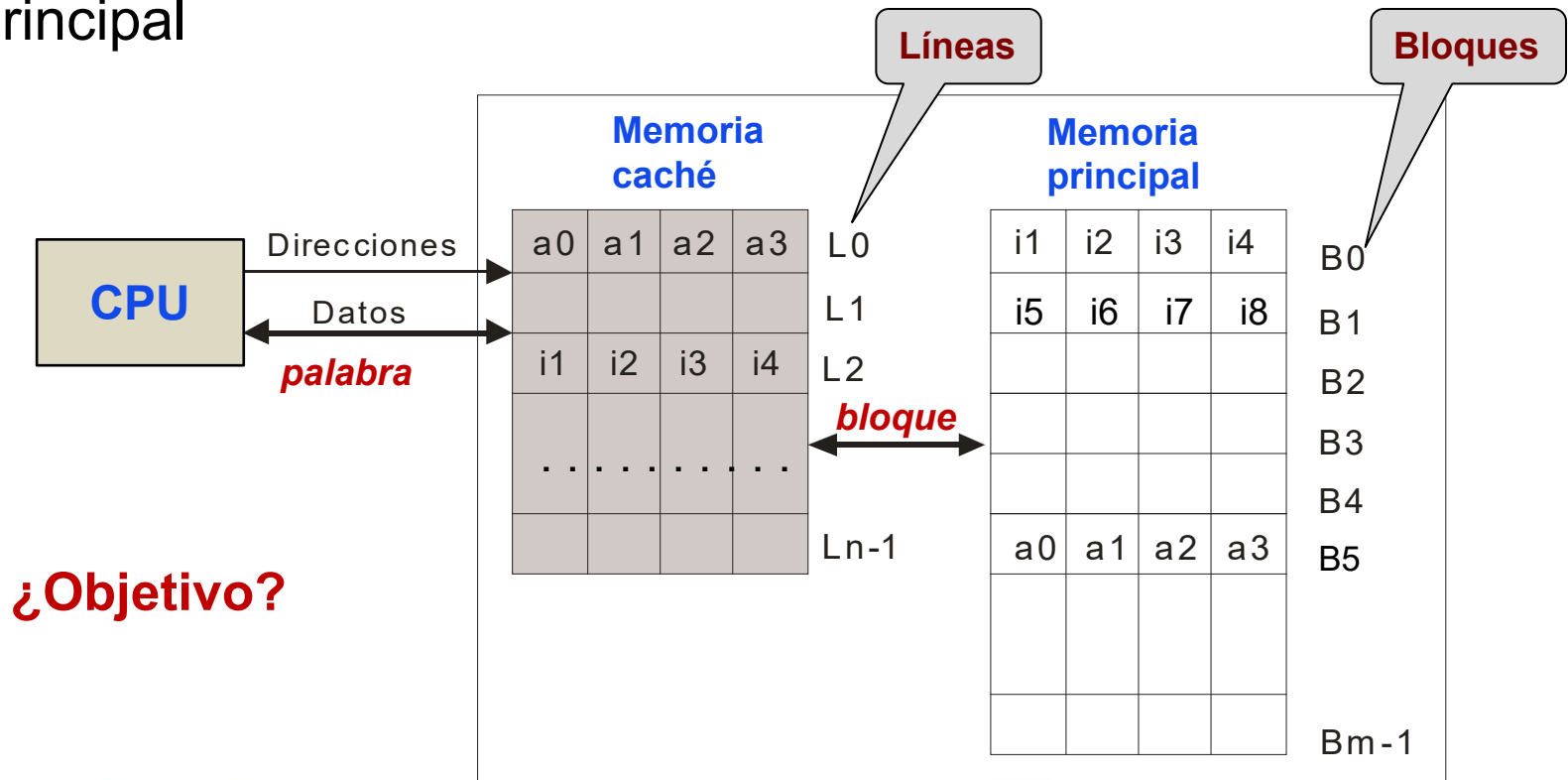
Memoria caché

- Memoria **rápida** y de **pequeña capacidad**, situada entre la CPU y la memoria principal
- Contiene parte de la información residente en la memoria principal



Memoria caché

- Memoria **rápida** y de **pequeña capacidad**, situada entre la CPU y la memoria principal
- Contiene parte de la información residente en la memoria principal



Memoria caché: Componentes



- Se accede utilizando la dirección de la información en Mp
↳ ¿Cómo saber si la información buscada está en la caché?

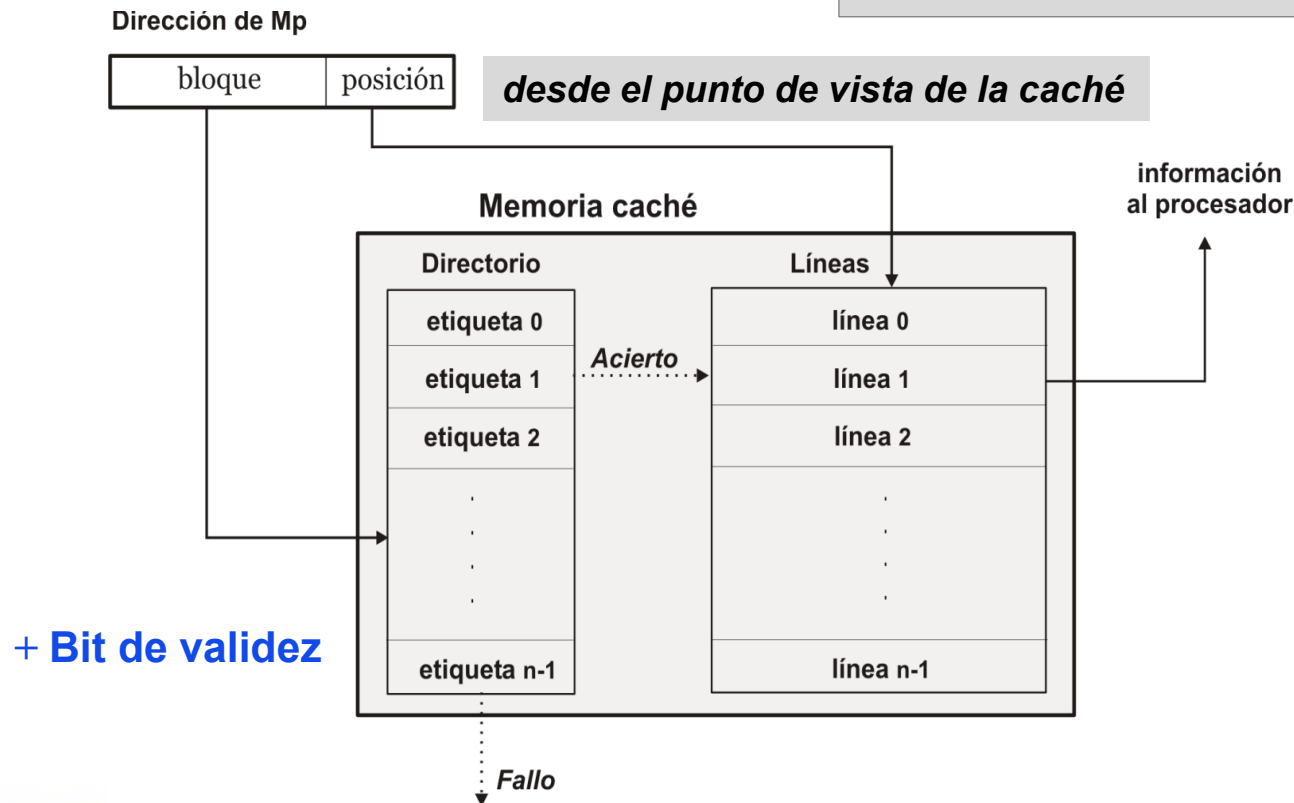
Memoria caché: Componentes

- Se accede utilizando la dirección de la información en Mp

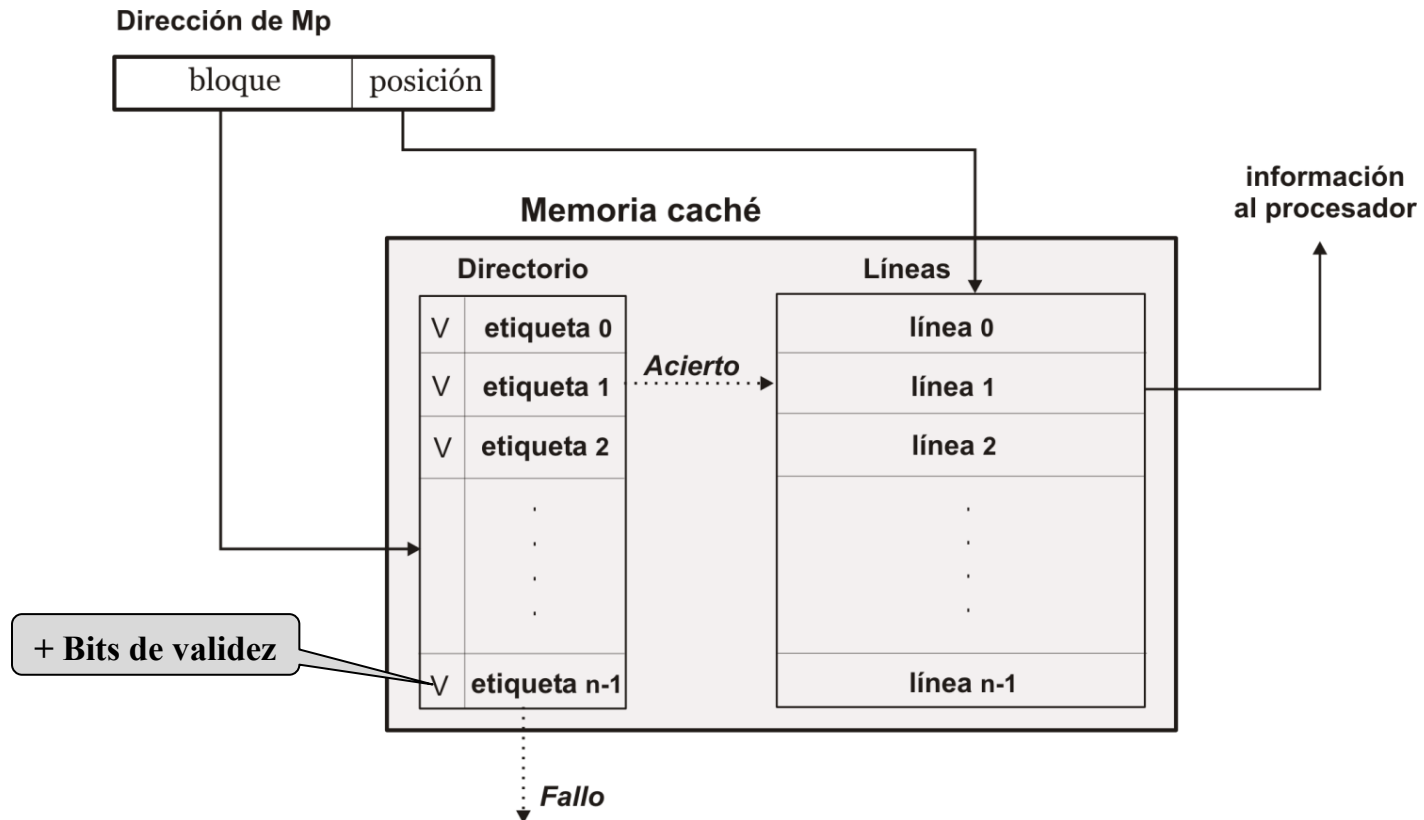


¿Cómo saber si la información buscada está en la caché?

Almacenando también qué bloque de Mp reside en cada línea de la caché: etiqueta



Memoria caché: Componentes



Ejemplo: Mp direccionable: 4 GB
bloques de 4 palabras de 4 bytes

¿Formato de la dirección, tamaño y rango de las etiquetas?

Memoria caché: Componentes



- **Almacenamiento** ('**líneas**'): donde se albergan físicamente los bloques* que se suben del siguiente nivel
- **Directorio**: qué bloque de Mp está almacenado en cada línea (dependerá de la política de ubicación)
- **Controlador**: implementa todas las políticas y realiza el servicio a los fallos

(*) El almacenamiento se estructura en **líneas**, y cada una contiene un conjunto de direcciones consecutivas de Mp (**bloque de caché**) entre 16 y 64 bytes → el espacio de direcciones de Mp se considera dividido en bloques

Memoria caché: Decisiones de diseño



- Políticas de **ubicación**

Dónde se coloca la información procedente de Mp

➡ Cómo encontrar la información en la caché

- Políticas de **extracción**

Cuándo se envía la información a la caché

- Políticas de **reemplazo**

Qué información sale (se sustituye) de la caché

- Políticas de **escritura**

Cómo se realizan los accesos de escritura

- **Tamaño** de la caché y de los bloques

Cuál es su influencia en la tasa de aciertos y en el tiempo medio de acceso.

Memoria caché: Políticas de ubicación



Determina cómo se establece la **correspondencia**:

bloques de $M_p \Rightarrow$ líneas de M_{ca}

- **Directa**

Cada bloque de M_p puede ubicarse solo en una línea predefinida de M_{ca} .

Memoria caché: Políticas de ubicación



Determina cómo se establece la **correspondencia**:

bloques de Mp \Rightarrow líneas de Mca

- **Directa**

Cada bloque de Mp puede ubicarse solo en una línea predefinida de Mca.

- **Asociativa**

Cada bloque de Mp puede ubicarse en cualquier línea de Mca

Memoria caché: Políticas de ubicación

Determina cómo se establece la **correspondencia**:

bloques de Mp \Rightarrow líneas de Mca

- **Directa**

Cada bloque de Mp puede ubicarse solo en una línea predefinida de Mca.

- **Asociativa**

Cada bloque de Mp puede ubicarse en cualquier línea de Mca

- **Asociativa por conjuntos**

Cada bloque de Mp puede ubicarse en cualquier línea dentro de un conjunto predefinido de líneas de Mca

Políticas de ubicación: Caso de estudio



Memoria caché:

Capacidad total: 8 KB (2^{13} bytes)

Tamaño de los bloques: 16 bytes (2^4 bytes)

→ N° de líneas = 512 líneas (2^9 líneas)

Memoria principal:

Espacio direccionable: 4 GB (2^{32} bytes)

→ N° de bloques = 2^{28} bloques

Política de ubicación:

bloque_i de Mp \Rightarrow línea_j de Mca

Memoria caché: Directa

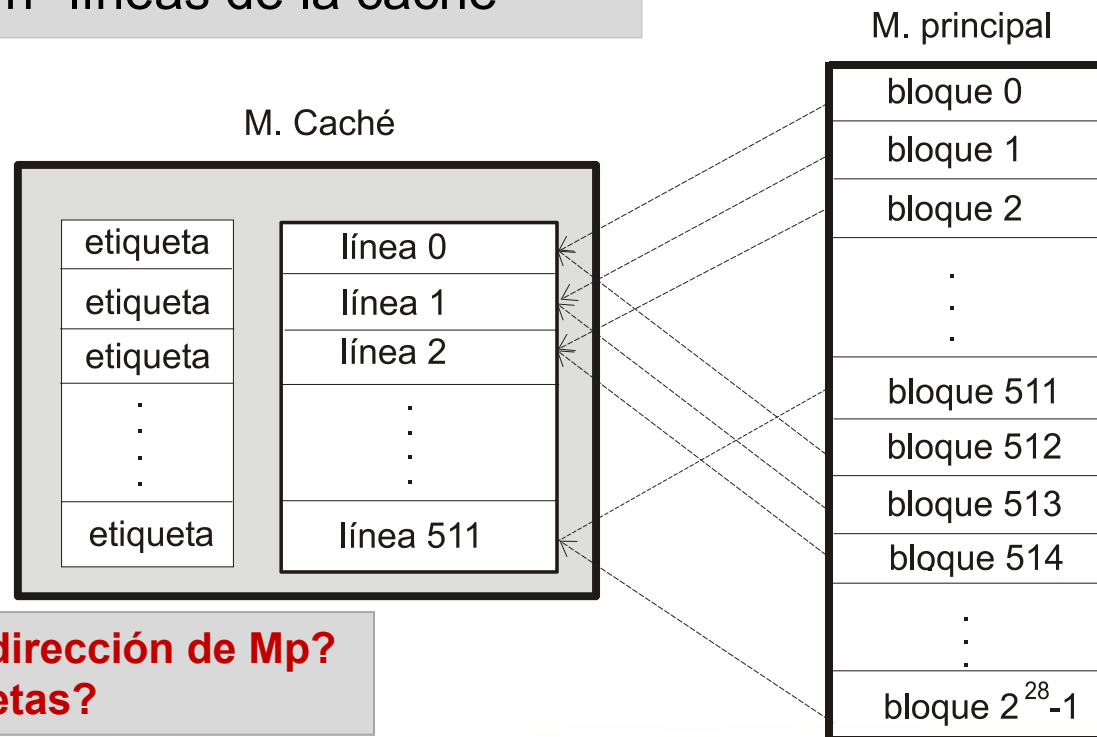


Cada bloque de M_p puede ubicarse en una sola línea predefinida de M_{ca} :

bloque $i \Rightarrow$ línea $(i \bmod L)$

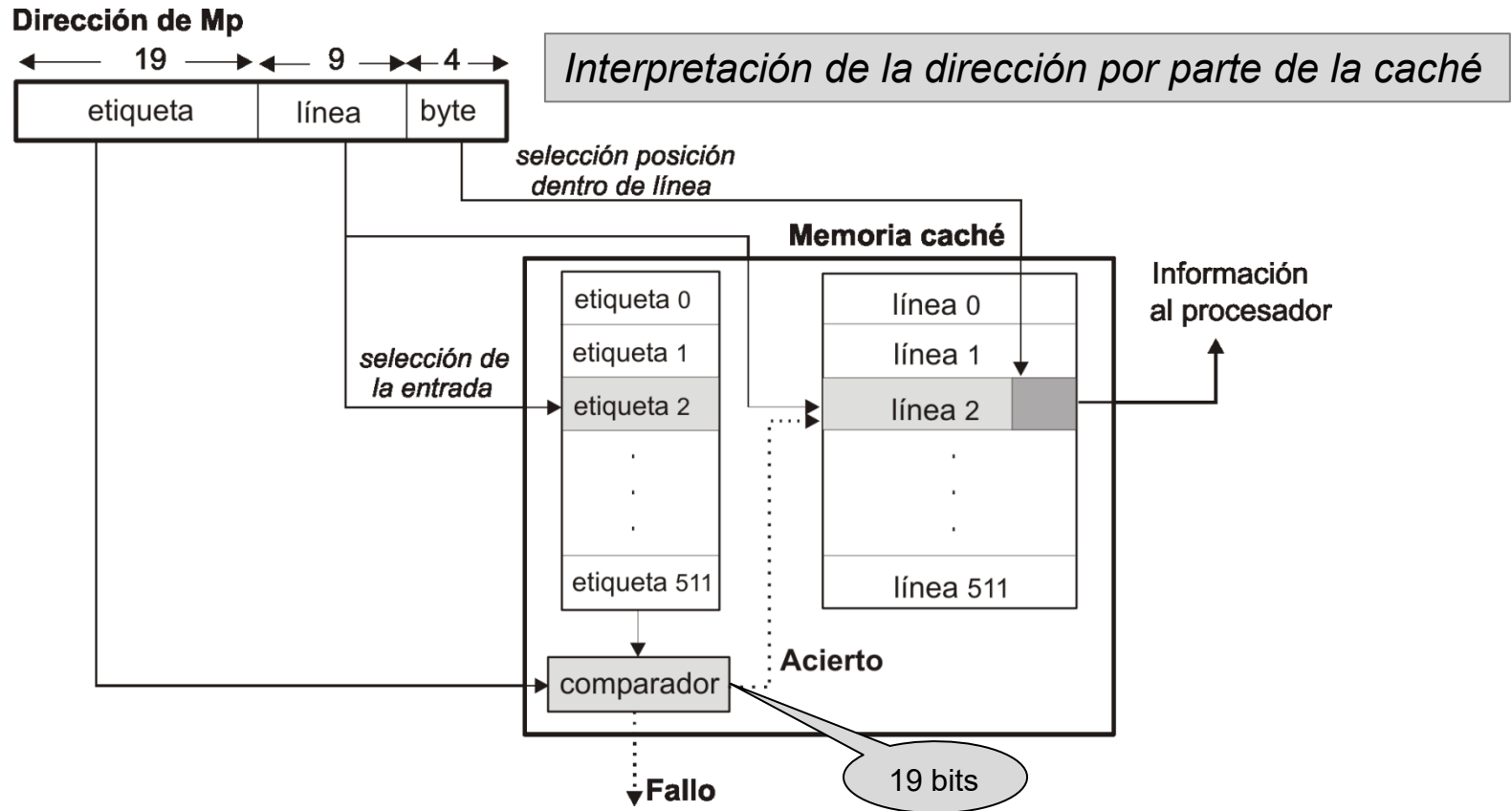
$L = n^\circ$ líneas de la caché

Caso de estudio:



¿Interpretación de la dirección de M_p ?
¿Tamaño de las etiquetas?

Memoria caché: Directa

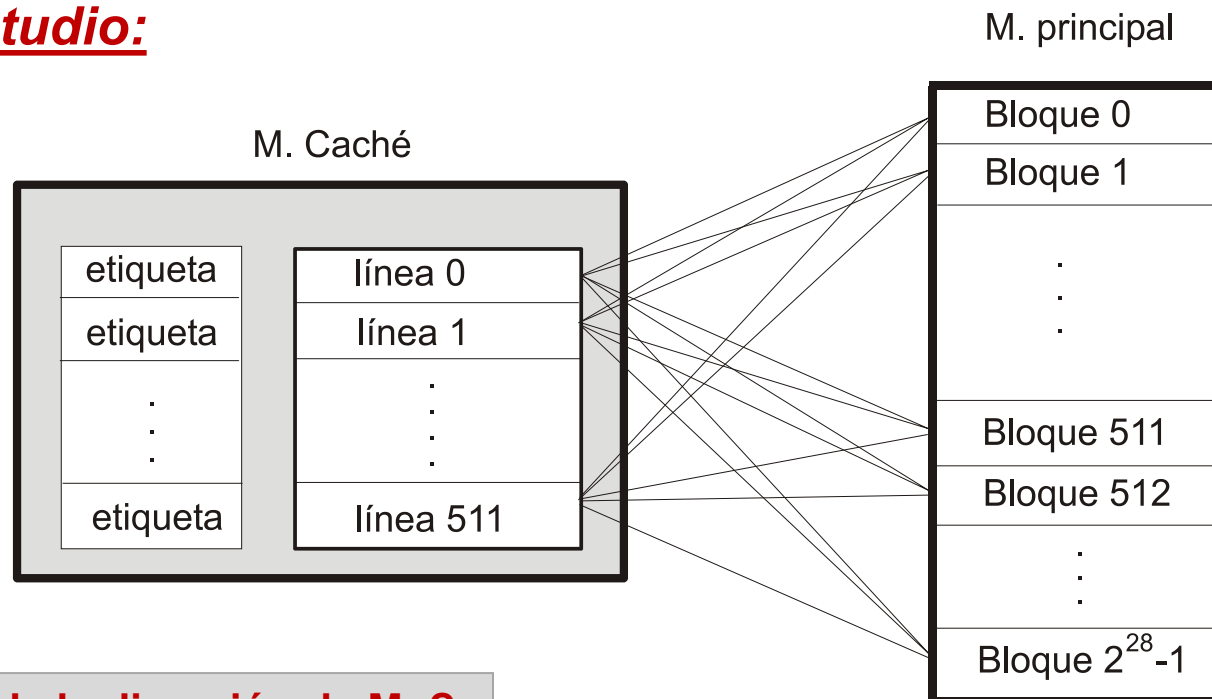


- **Ventajas:** Sencillez, velocidad y bajo coste
- **Inconvenientes:** Dependiendo de la traza del programa, puede producir una tasa de fallos elevada

Memoria caché: Asociativa

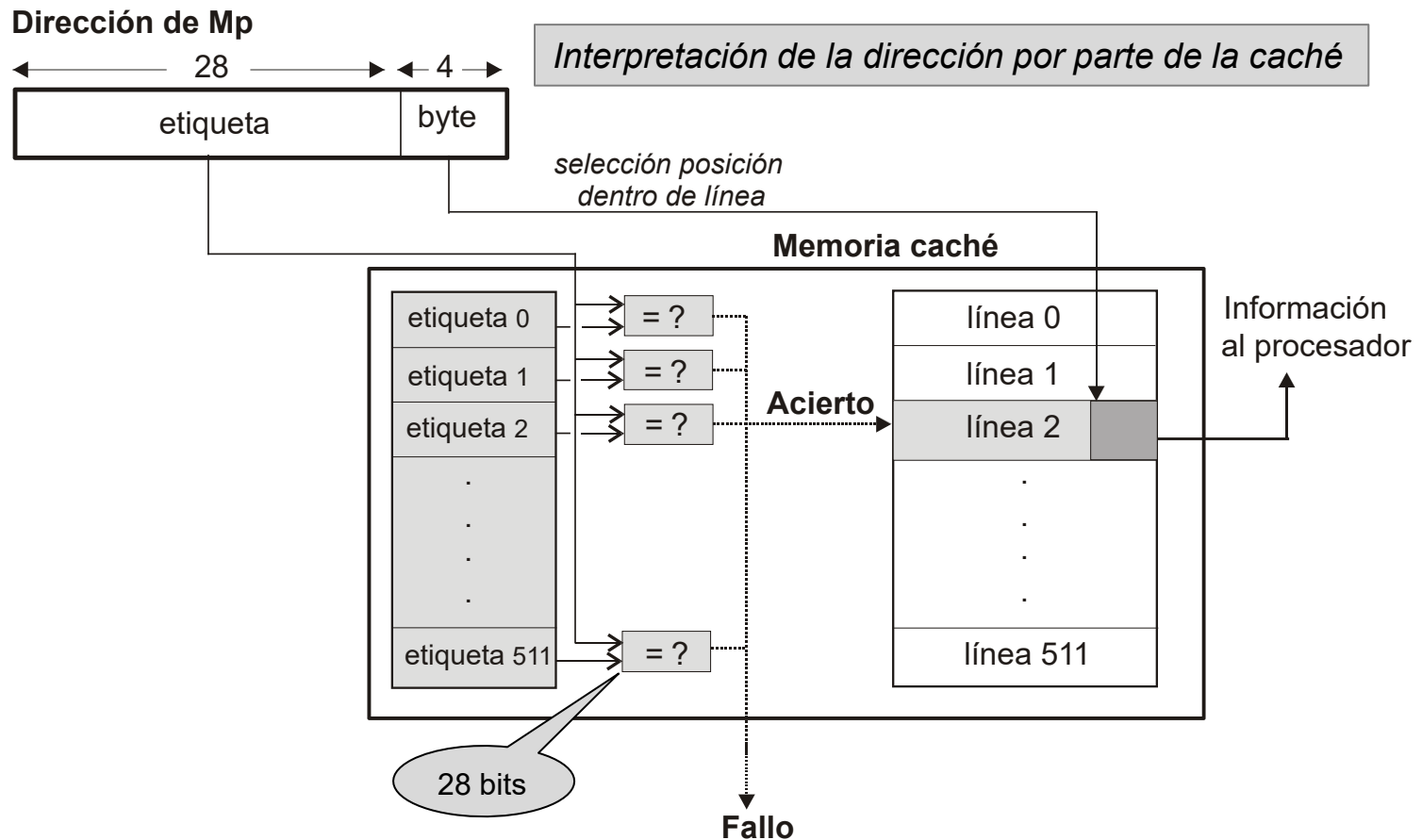
Cada bloque de Mp puede ubicarse en cualquier línea de Mca

Caso de estudio:



¿Interpretación de la dirección de Mp?
¿Tamaño de las etiquetas?

Memoria caché: Asociativa



- **Ventajas:** Independiente de la traza del programa
La mejor tasa de aciertos con política de reemplazo óptima
- **Inconvenientes:** Complejidad y coste elevado

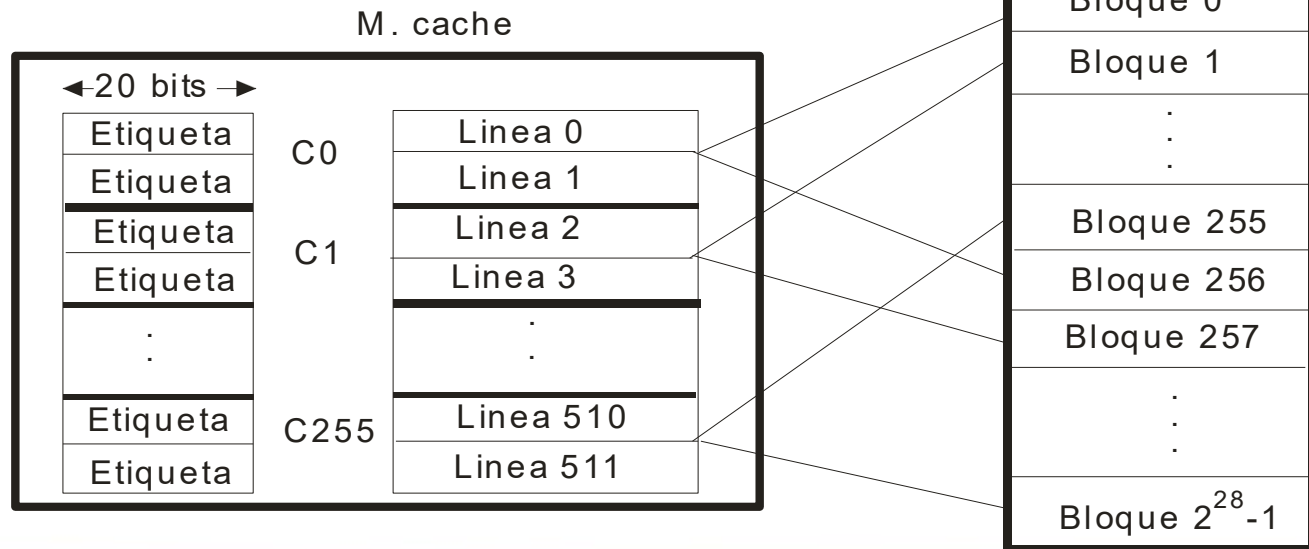
Memoria caché: Asociativa por conjuntos

Cada bloque de M_p puede situarse en cualquier línea de un conjunto predefinido de líneas de M_{ca}

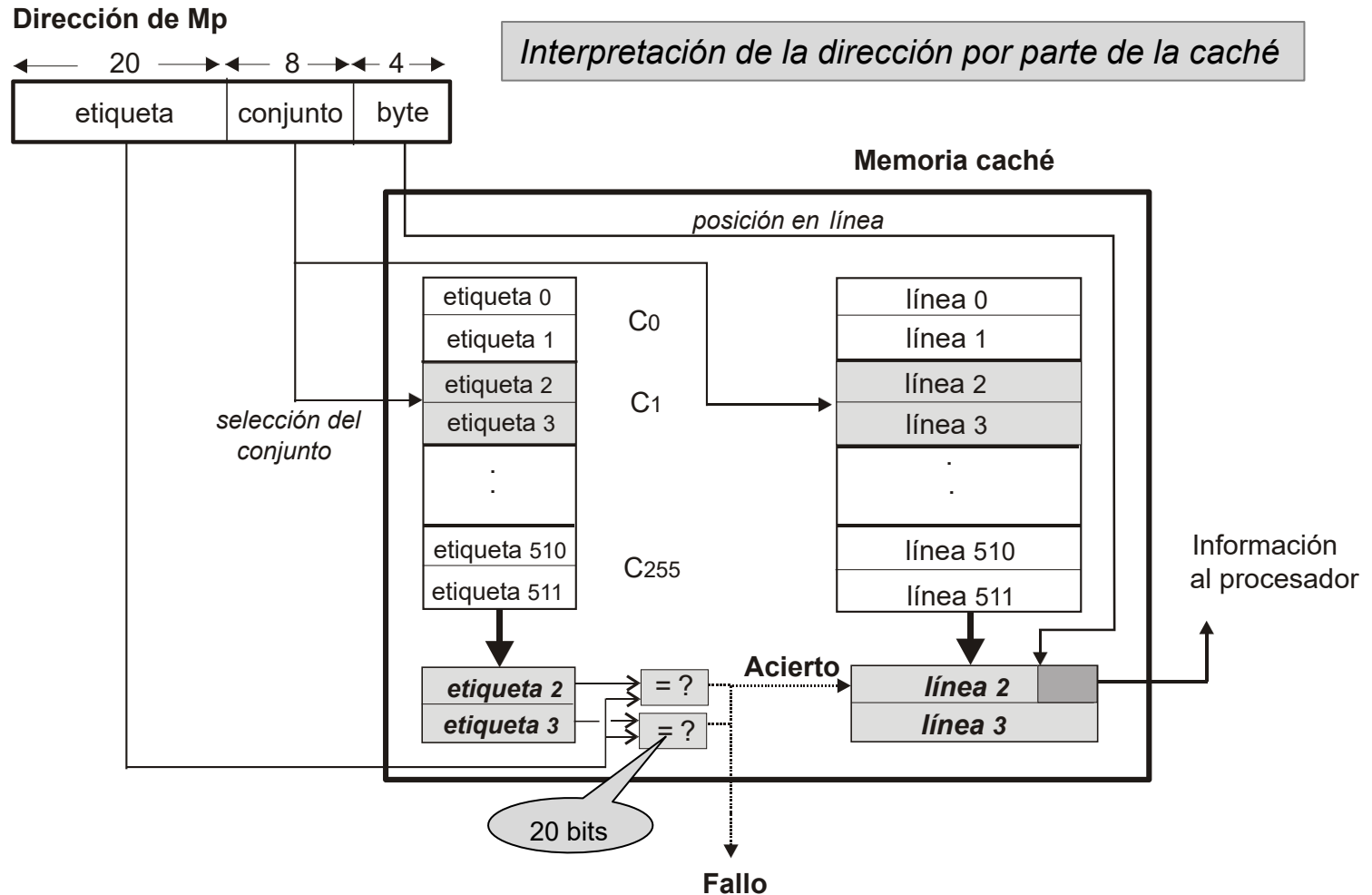
bloque $i \Rightarrow$ conjunto ($i \bmod C$)

$C = n^\circ$ conjuntos de la caché

Caso de estudio con conjuntos de 2 líneas:



Memoria caché: Asociativa por conjuntos



- **Ventajas:** Las de la directa y la asociativa

Políticas de ubicación: Comparación



- Complejidad y coste
- Tiempo de acierto
- Flexibilidad de la política de reemplazo
- Hr (dependiendo de la traza)

- Directa, Asociativa por conjuntos, Asociativa → +

Memoria caché: Decisiones de diseño



- ✓ Políticas de **ubicación**

Dónde colocar la información procedente de Mp

→ Cómo encontrar la información en la caché

- Políticas de **extracción**

Cuándo se envía la información a la caché

- Políticas de **reemplazo**

Qué información sale (se sustituye) de la caché

- Políticas de **escritura**

Cómo se realizan los accesos de escritura

- **Tamaño** de la caché y de los bloques

Cuál es su influencia en la tasa de aciertos y en el tiempo medio de acceso.

Memoria caché: Políticas de extracción



Deciden **cuándo y qué bloque** se lleva a la caché

- **Bajo demanda:**

Se lleva a la caché el bloque que produce el fallo (“por defecto”)

Memoria caché: Políticas de extracción



Deciden **cuándo y qué bloque** se lleva a la caché

- Bajo demanda:

Se lleva a la caché el bloque que produce el fallo (“por defecto”)

- Con anticipación (*prefetch*):

Se llevan a la caché bloques que previsiblemente se van a necesitar en un futuro próximo → se fundamenta en el fenómeno de proximidad espacial

Objetivo: $\uparrow H_{r_{Mca}}$

Memoria caché: Políticas de reemplazo



Deciden **qué bloque se “desaloja”** de la caché para albergar al bloque que llega de Mp

- Solo necesarias en memorias caché No-Directas
- Tipos:
 - Aleatoria
 - FIFO (*First In, First Out*)
Se reemplaza el bloque más antiguo de la Mca/conjunto
 - LRU (*Least Recently Used*)
Se reemplaza el bloque al que no se ha accedido durante un mayor periodo de tiempo

FIFO y LRU requieren almacenar **información adicional en Mca y actualizarla**

Memoria caché: Políticas de escritura



Deciden **cuándo se actualiza la información** en el siguiente nivel

1) Si hay **acierto** en el acceso a la caché

- **Escritura inmediata** (*Write-through*)

Se escribe en la caché y en el siguiente nivel de la jerarquía

➔ Se asegura en todo momento la coherencia entre niveles adyacentes.

Memoria caché: Políticas de escritura



Deciden **cuándo se actualiza la información** en el siguiente nivel

1) Si hay **acierto** en el acceso a la caché

- **Escritura inmediata** (*Write-through*)

Se escribe en la caché y en el siguiente nivel de la jerarquía

➡ Se asegura en todo momento la coherencia entre niveles adyacentes.

- **Escritura aplazada** (*Copy-back*)

– Se escribe solo en la caché

– El bloque de caché modificado se escribe en el siguiente nivel solo cuando es reemplazado

➡ Necesario **bit de modificación/línea**
Posible problema de **coherencia**

Memoria caché: Políticas de escritura



2) Si hay **fallo** en el acceso a la caché

- Con ubicación del bloque en la caché (*with allocation*)
El bloque **se envía** a la caché (igual que en los fallos de lectura)
- Sin ubicación del bloque en la caché (*with no allocation*)
El bloque **no se envía** a la caché

Memoria caché: Políticas de escritura



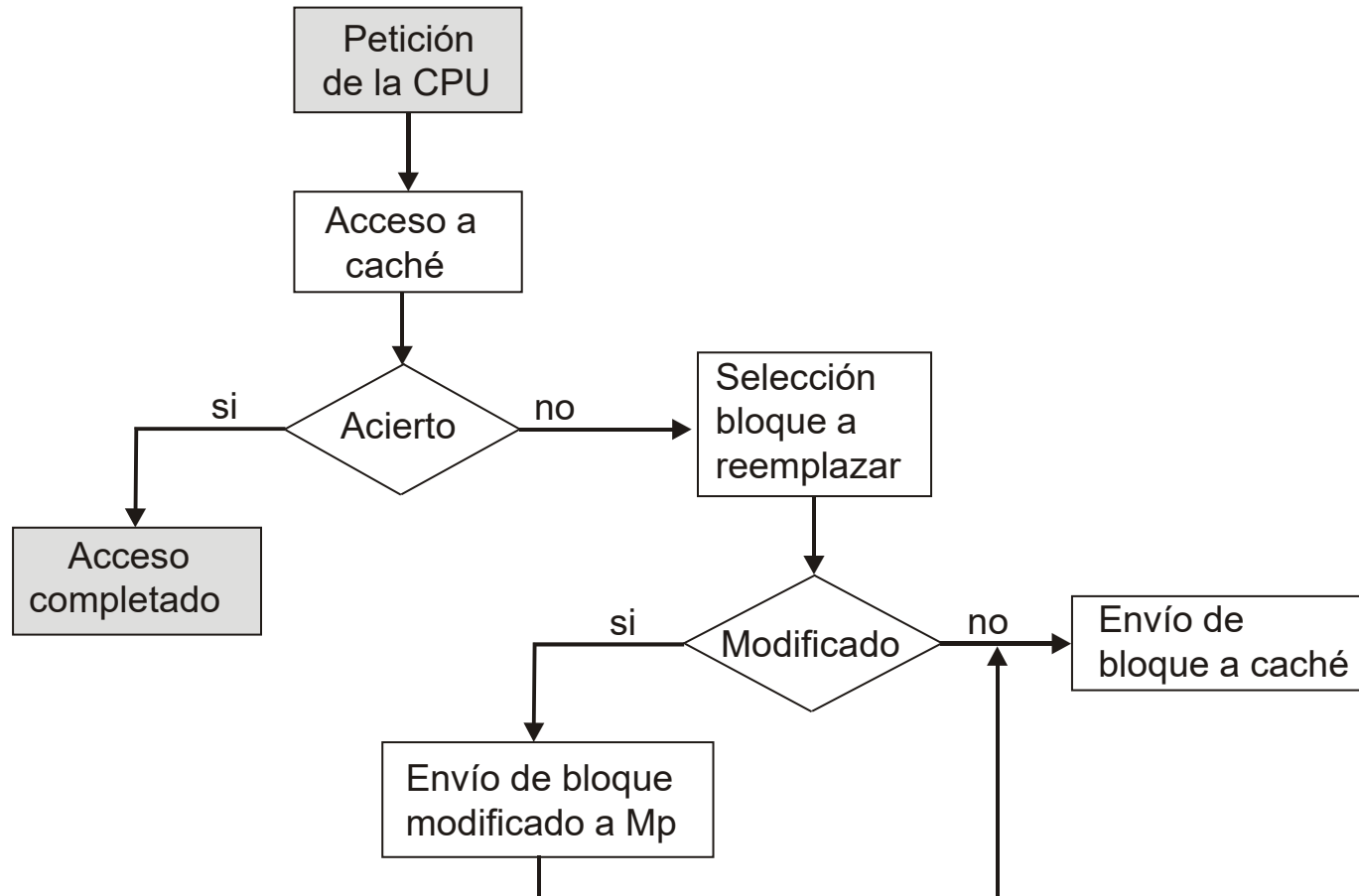
2) Si hay **fallo** en el acceso a la caché

- Con ubicación del bloque en la caché (*with allocation*)
El bloque **se envía** a la caché (igual que en los fallos de lectura)
- Sin ubicación del bloque en la caché (*with no allocation*)
El bloque **no se envía** a la caché

Habitualmente se utiliza:

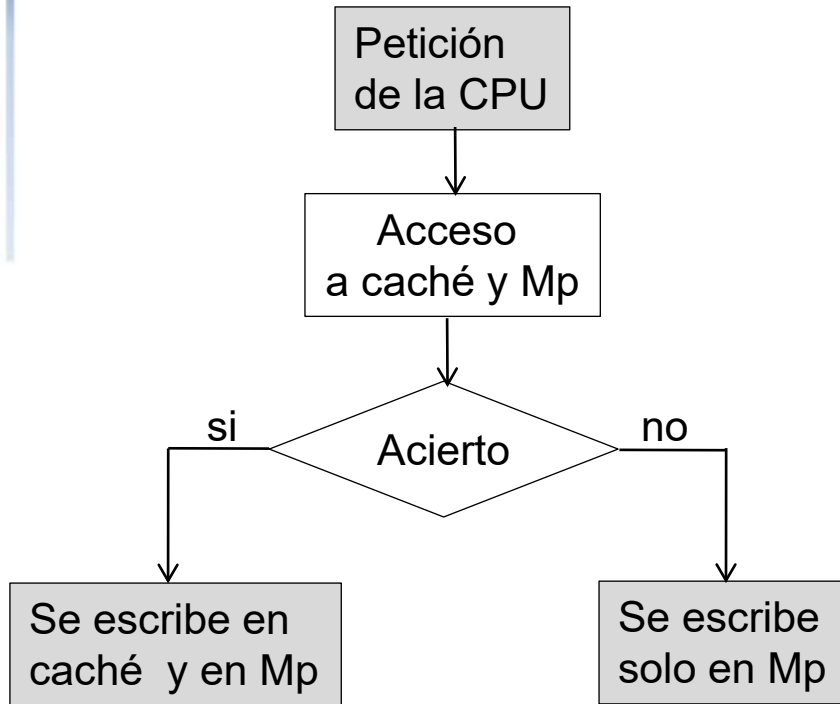
- Escritura inmediata sin ubicación
(*WTWNA: Write Through With No Allocation*)
- Escritura aplazada con ubicación
(*CBWA: Copy Back With Allocation*)

CBWA: Diagrama de flujo de un acceso

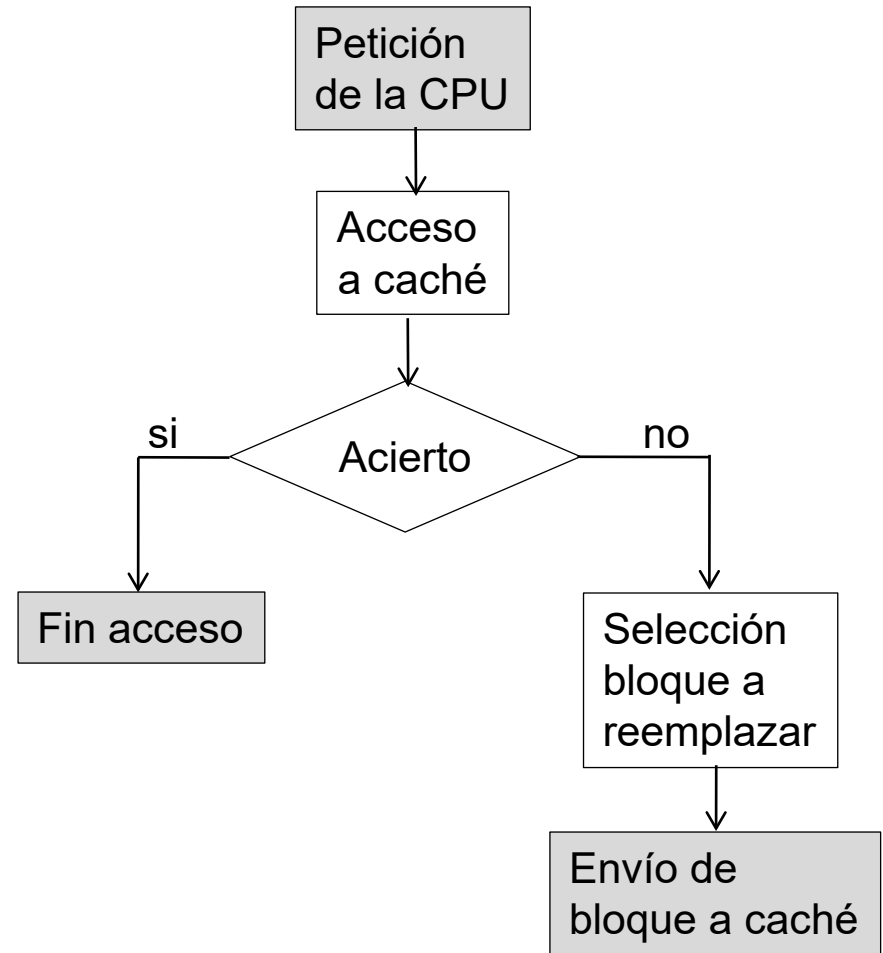


Igual para lecturas y escrituras

WTWNA: Diagrama de flujo de un acceso



Escritura



Lectura

Memoria caché: Medidas de rendimiento



- Tasa de aciertos (Hit ratio)

Objetivo:

$$\frac{N^{\circ} \text{ accesos con acierto en } Mca}{N^{\circ} \text{ total de accesos}}$$

$\approx 100\%$

Memoria caché: Medidas de rendimiento



- Tasa de aciertos (Hit ratio)

Objetivo:

$$\frac{N^{\circ} \text{ accesos con acierto en } M_{ca}}{N^{\circ} \text{ total de accesos}}$$

$\approx 100\%$

- Tiempo medio de acceso o Tiempo efectivo (T_{ef})

Tiempo medio transcurrido desde que la CPU realiza una petición al sistema de memoria hasta que puede continuar

$$T_{Mca} + (1 - Hr_{Mca}) \times T_{\text{penalización}} (*)$$

$\approx TM_{ca}$

(*) Si acceso de lectura o de escritura con CBWA

Memoria caché: Medidas de rendimiento



- Tasa de aciertos (Hit ratio)

Objetivo:

$$\frac{N^{\circ} \text{ accesos con acierto en } M_{ca}}{N^{\circ} \text{ total de accesos}}$$

$\approx 100\%$

- Tiempo medio de acceso o Tiempo efectivo (T_{ef})

Tiempo medio transcurrido desde que la CPU realiza una petición al sistema de memoria hasta que puede continuar

$$T_{Mca} + (1 - Hr_{Mca}) \times T_{penalización}$$

$$T_{Mp} \quad \text{si escritura y WTWNA}$$

$\approx TM_{ca}$

Memoria caché: Medidas de rendimiento



- Tasa de aciertos (Hit ratio)

Objetivo:

$$\frac{N^{\circ} \text{ accesos con acierto en Mca}}{N^{\circ} \text{ total de accesos}}$$

$\approx 100\%$

- Tiempo medio de acceso o Tiempo efectivo (T_{ef})

Tiempo medio transcurrido desde que la CPU realiza una petición al sistema de memoria hasta que puede continuar

$$T_{Mca} + (1 - Hr_{Mca}) \times T_{\text{penalización}}$$

- Tiempo medio de ocupación

$\approx TM_{ca}$

Tiempo medio transcurrido desde que el sistema de memoria sirve una petición hasta que puede servir la siguiente

$$T_{ef} + T_{\text{actualización Mca}}$$

$\approx T$ entre peticiones de la CPU

Memoria caché: Tipos de fallo



- De primera referencia
- De conflicto
- De capacidad
- Otros (p.e. política de reemplazo)

Memoria caché: Reducción de la tasa de fallos



■ Tamaño de la caché

- Típico: 16-64KB (**McaL1**)
- Influye en los fallos por Capacidad
- Influye en el T_{acceso}

Libro:

3.5

■ Tamaño de los bloques

- Típico: 16-64B
- Puede favorecer/perjudicar los distintos tipos de proximidad de referencias
- Influye en los fallos de primera referencia o forzosos
- Influye en el $T_{\text{actualización}}^{\text{Mca}}$ ($T_{\text{penalización}}$ y $T_{\text{ocupación}}$)

■ Extracción con anticipación (*prefetch*)

Cachés unificadas vs. separadas para I/D



- *Arquitectura Harvard:*

Cachés separadas de menor tamaño

Cachés L1

- Optimización de las políticas de gestión para cada caché:

Ubicación, Escritura, Reemplazo, Extracción

- Duplicación de buses

- Experimentalmente:

$HrMcaI > HrMcaD$

Libro:

3.6

- Imprescindible para procesadores con *pipeline* (ILP):

Permite dos accesos simultáneos a memoria

Cachés de primer nivel (L1): Ejemplos



Procesador	Caché de Instrucciones			Caché de Datos			
	Tamaño (KB)	bytes/línea	líneas/cjto.	Tamaño (KB)	bytes/línea	líneas/cjto.	Escritura
ARM C-A8	32	64	4	32	64	4	CBWA
Intel i7	32	64	8	32	64	8	CBWA
Amd Opteron	64	64	2	64	64	2	CBWA
Power 8	32	128	8	64	128	8	WTWNA

La memoria virtual



- Motivaciones
 - Programas no limitados por la capacidad de la Mp disponible
Capacidad aparente >> Capacidad de la Mp

La memoria virtual



- Motivaciones

- Programas no limitados por la capacidad de la Mp disponible
Capacidad aparente >> Capacidad de la Mp
- En entornos multiprogramación
 - **Protección y compartición** de información
 - **Reubicación dinámica**

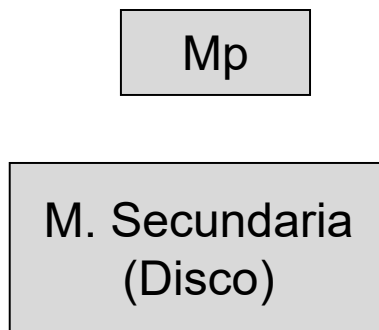
La memoria virtual



■ Motivaciones

- Programas no limitados por la capacidad de la Mp disponible
Capacidad aparente >> Capacidad de la Mp
- En entornos multiprogramación
 - Protección y compartición de información
 - Reubicación dinámica

■ Soporte: Jerarquía de memorias



- Capacidad de almacenamiento aparente ilimitada
- La Mp actúa como una caché de la M. secundaria



En Mp solo la información útil en cada momento

La memoria virtual



- **Espacio de direcciones lógico (EDL)**
Dependiente de la Arquitectura (juego de instrucciones)
- **Espacio de direcciones físico (EDF)**
Dependiente de la Mp

La memoria virtual



- **Espacio de direcciones lógico (EDL)**
Dependiente de la Arquitectura (juego de instrucciones)
- **Espacio de direcciones físico (EDF)**
Dependiente de la Mp

Ejemplo:

Direccionamiento [ri]

Registros de 32 bits

Mp de 1GB

¿EDL?

¿EDF?

La memoria virtual

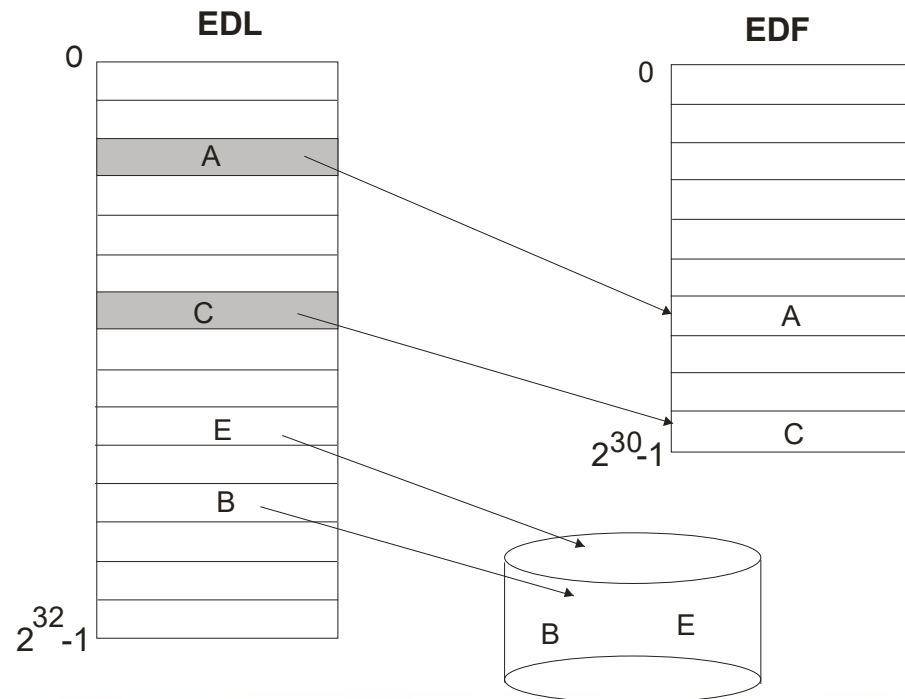
- **Espacio de direcciones lógico (EDL)**

Dependiente de la Arquitectura (juego de instrucciones)

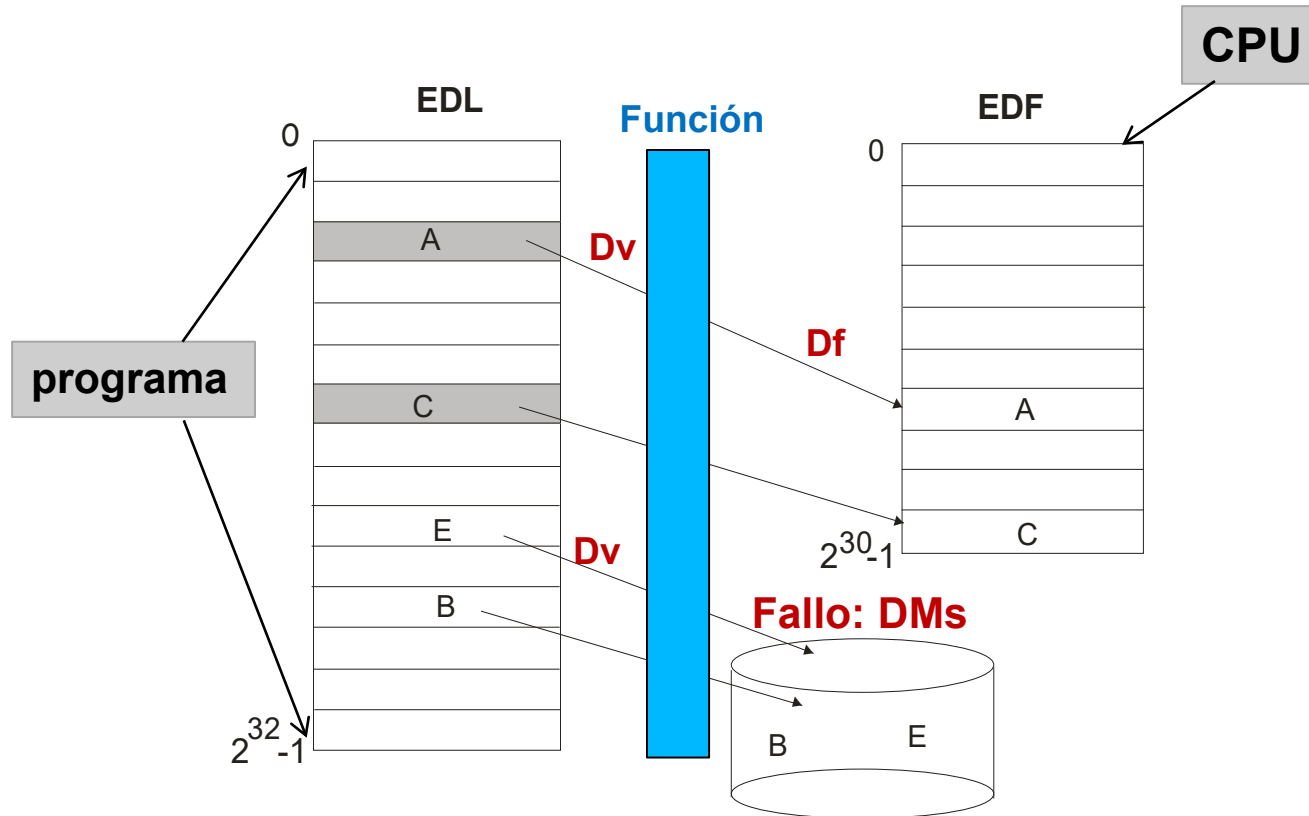
- **Espacio de direcciones físico (EDF)**

Dependiente de la Mp

**Direcciones distintas
para la misma información**



Mv: Traducción de direcciones



**Direcciones distintas
para la misma información**

La memoria virtual



- Problemas a resolver:

- Traducción de direcciones
- Gestión de los fallos



Software (S.O.) + Hardware

- Asignación de Mp a los procesos

S.O.

La memoria virtual



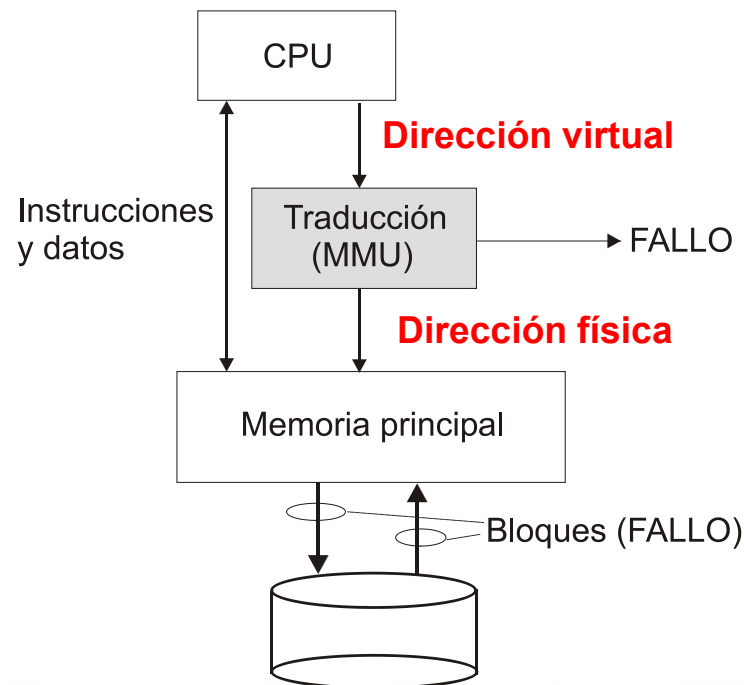
- Problemas a resolver:

- Traducción de direcciones
- Gestión de los fallos

Software (S.O.) + Hardware

- Asignación de Mp a los procesos

S.O.



Fallo: Excepción gestionada por el S.O.

■ Acciones a realizar:

- Suspender el proceso P_i en el que se produjo
- Ordenar la transferencia del bloque en el que se encuentra la información, desde disco a Mp (op. de E/S por DMA)
- Actualizar la función de traducción
- Dar control a otro proceso P_j (**cambio de contexto**)

Fallo: Excepción gestionada por el S.O.

- **Acciones a realizar:**
 - Suspende el proceso P_i en el que se produjo
 - Ordenar la transferencia del “bloque” en el que se encuentra la información, desde disco a Mp (op. de E/S por DMA)
 - Actualizar la función de traducción
 - Dar control a otro proceso P_j (**cambio de contexto**)
- **Finalizada la transferencia**, y cuando así lo decida el S.O.:
 - Retomar la ejecución de P_i :
 - **Reiniciar** la instrucción interrumpida o
 - Completar la instrucción interrumpida

Mv: Implementación



- Paginación

bloques de tamaño fijo: **páginas**

- Segmentación

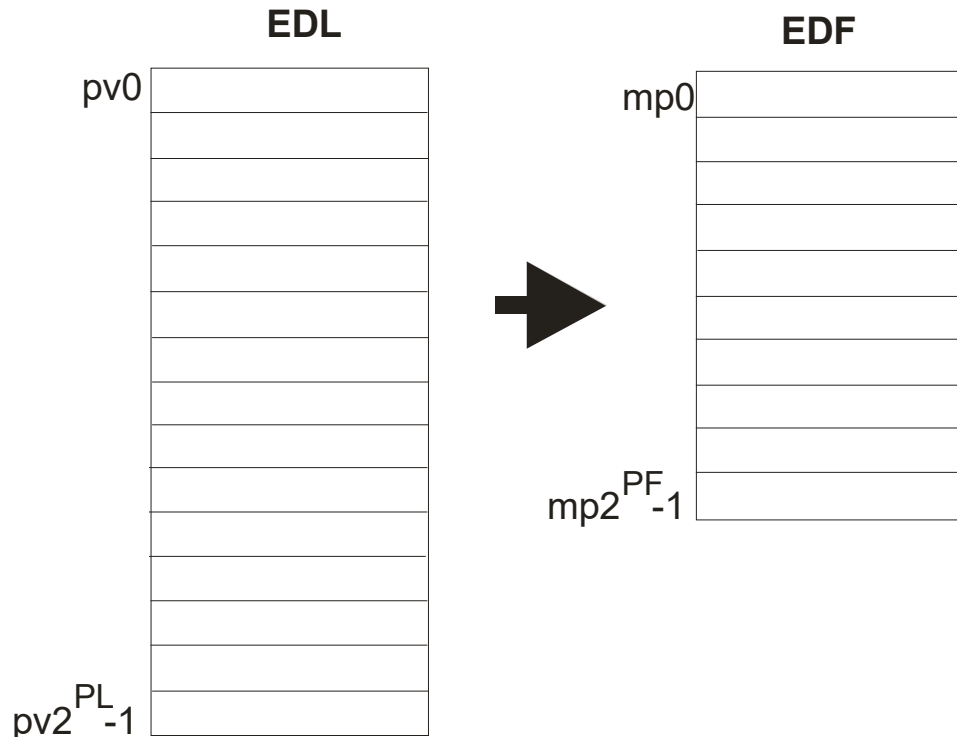
bloques de tamaño variable: **segmentos**

- Segmentación paginada

segmentos, compuestos a su vez por páginas

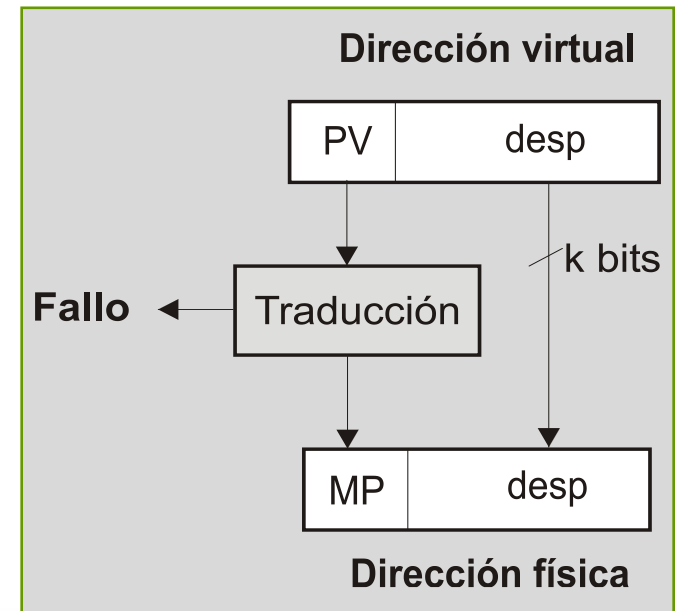
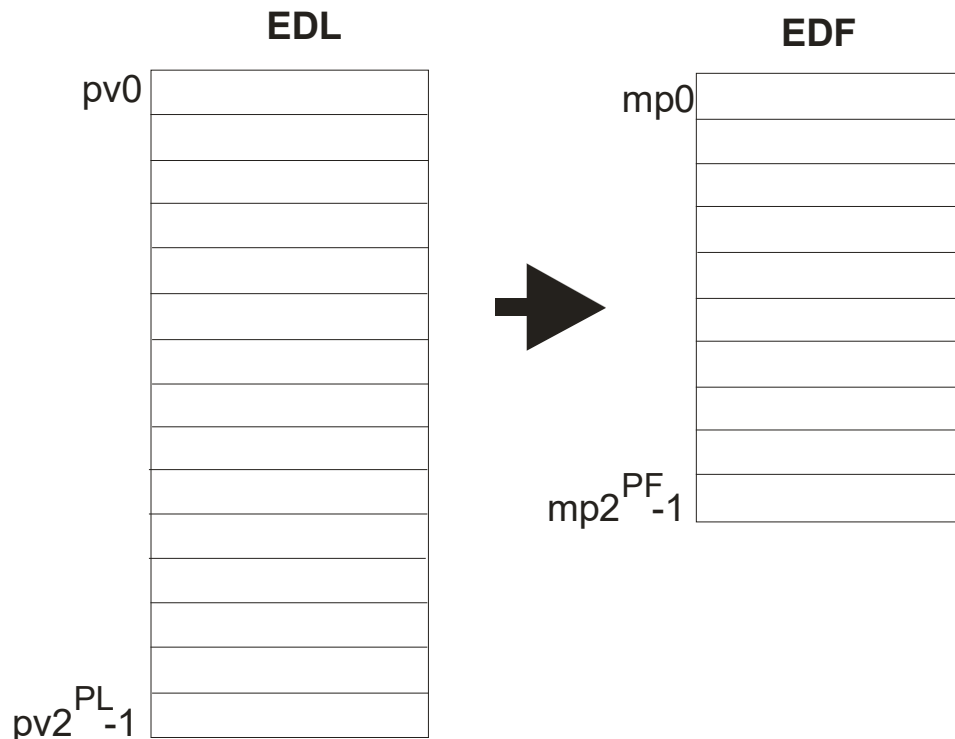
Paginación

- Correspondencia entre **páginas** virtuales y **marcos de página**
- Tamaño de páginas **2^k bytes**



Paginación

- Correspondencia entre **páginas** virtuales y **marcos de página**
- Tamaño de páginas **2^k bytes**



Ejercicio:

EDL = 4GB

EDF = 1GB

Páginas de 4KB

Direccionamiento a nivel de byte y palabras de 4B

¿Nº de páginas?

¿Nº de marcos de página?

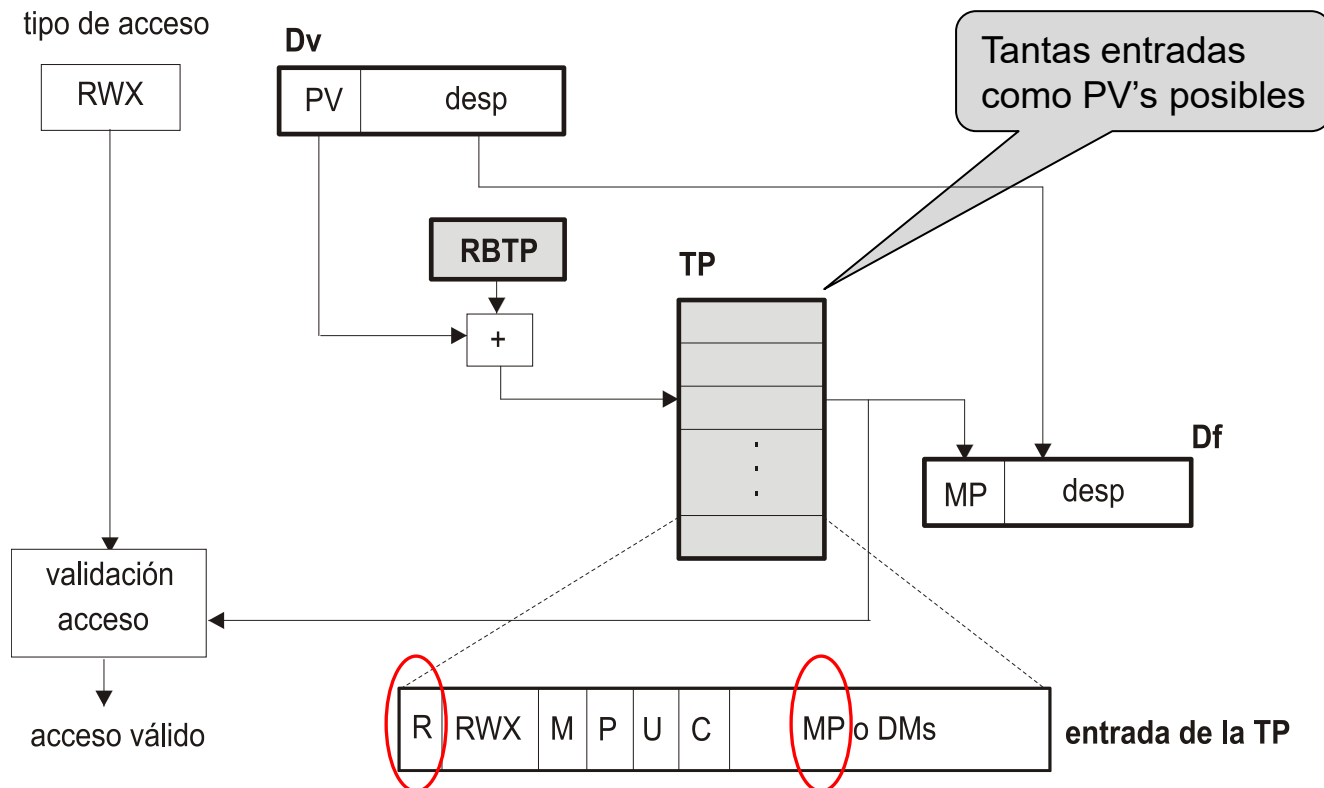
¿Formato de la Dv?

¿Interpretación de la Dv (p.e. *fetch*): H'00010008?

¿Formato de la Df?

Paginación: Traducción

- Implementación de la traducción: **Tabla de páginas**
- Una tabla por proceso + **RBTP**



MV: Necesidad de trad. mediante Hw

- Inconveniente de la MV: Tiempo empleado en la **traducción** de direcciones.
- Todas las instrucciones requieren al menos 1 acceso a memoria (*fetch*) y algunas uno o más accesos adicionales.

Ejemplo:

ld r1, 0(r2)

2 niveles de TP, Tacc(Mp) = 60 ns

¿Tiempo empleado en accesos a memoria **sin** Mv?

$$60+60 = \mathbf{120\ ns}$$

¿Tiempo empleado en accesos a memoria **con** Mv?

$$\mathbf{2 \times 60 + 60 + 2 \times 60 + 60 = 360\ ns}$$

MV: Aceleración de la traducción

- Mecanismo basado en la propiedad de proximidad de referencias.
- **TLB** (*translation lookaside buffer* **tlb fault trap**)
 - Actúa como una caché de las tablas de traducción:

si **acierto** en TLB
 traducción finalizada
si no
 acceso a las tablas de traducción en Mp
 y actualización de la TLB

Ejemplo:

$$T(\text{TLB}) = 2 \text{ ns}$$

¿Tiempo mínimo empleado en accesos a memoria **con** Mv?

$$2+60 + 2+60 = \mathbf{124 \text{ ns}} \text{ frente a } \mathbf{360 \text{ ns}}$$

Combinación Mcaché-Mvirtual



Esquema de componentes

