

Jerarquías de memorias

Arquitectura de Computadoras I

Fac. Cs. Exactas

UNCPBA

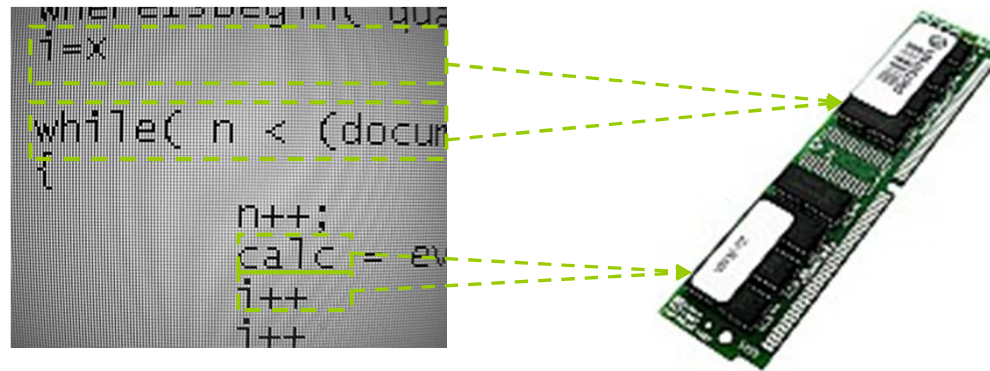
Mg. Marcelo Tosini

2020



Introducción

Los programas comparten en la memoria tanto su **código** como sus **datos**.



Estrategia de optimización de rendimiento

- posibilitar a la CPU el acceso **ilimitado y rápido** tanto al código como a los datos.

Inconveniente

- **tecnológicamente**, cuanto más rápidas son, más costosas resultan.

Tecnologías de memorias

- **RAM estática (SRAM)**
 - 0,5 a 2,5 ns
 - 100 U\$/GB
- **RAM Dinámica (DRAM)**
 - 50 a 70 ns
 - 20 U\$/GB
- **Flash**
 - 5 a 50 μ s
 - 1 U\$/GB
- **Disco estado sólido (SSD)**
 - 0,2 ms
 - 0,2 U\$/GB
- **Disco rígido (HDD)**
 - 5 a 20 ms
 - 0,1 U\$/GB



Introducción

Ley de localidad

“Todo programa favorece una parte de su espacio de direcciones en cualquier instante de tiempo.”

2 dimensiones:

Localidad temporal (tiempo). Si se referencia un elemento tenderá a ser referenciado pronto.

Localidad espacial (espacio). Si se referencia un elemento, los elementos cercanos a él tenderán a ser referenciados pronto.

Introducción

Jerarquía de memoria

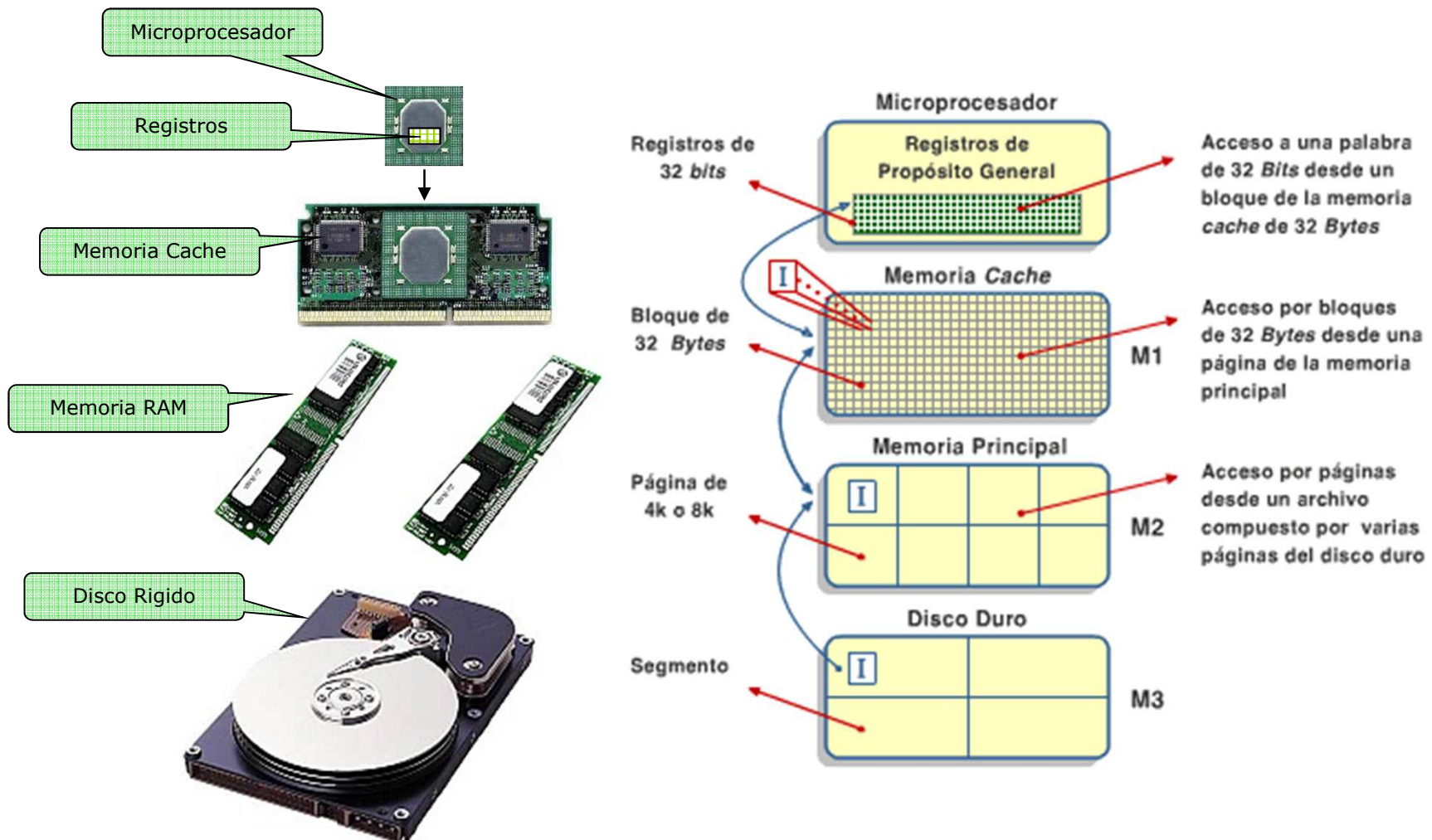
Es la reacción natural a la localidad y tecnología

El principio de localidad y la directriz que el hardware más rápido es más caro, mantienen el concepto de una jerarquía basada en diferentes localidades y tamaños.

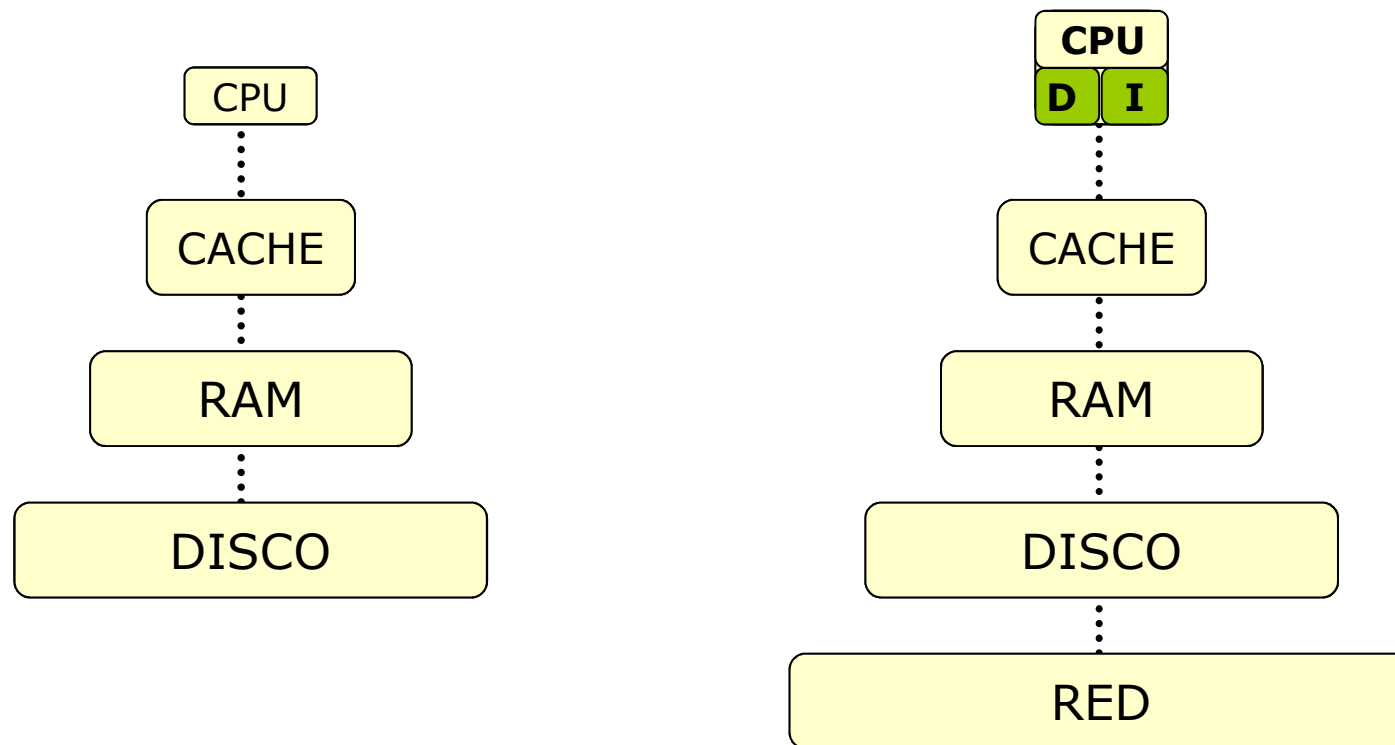
Organizada en varios niveles -cada uno más pequeño, más caro y más rápido que el anterior.

Todos los datos de un nivel se encuentran también en el nivel siguiente, y todos los datos de ese nivel inferior se encuentran también en el siguiente a él, hasta el extremo inferior de la jerarquía.

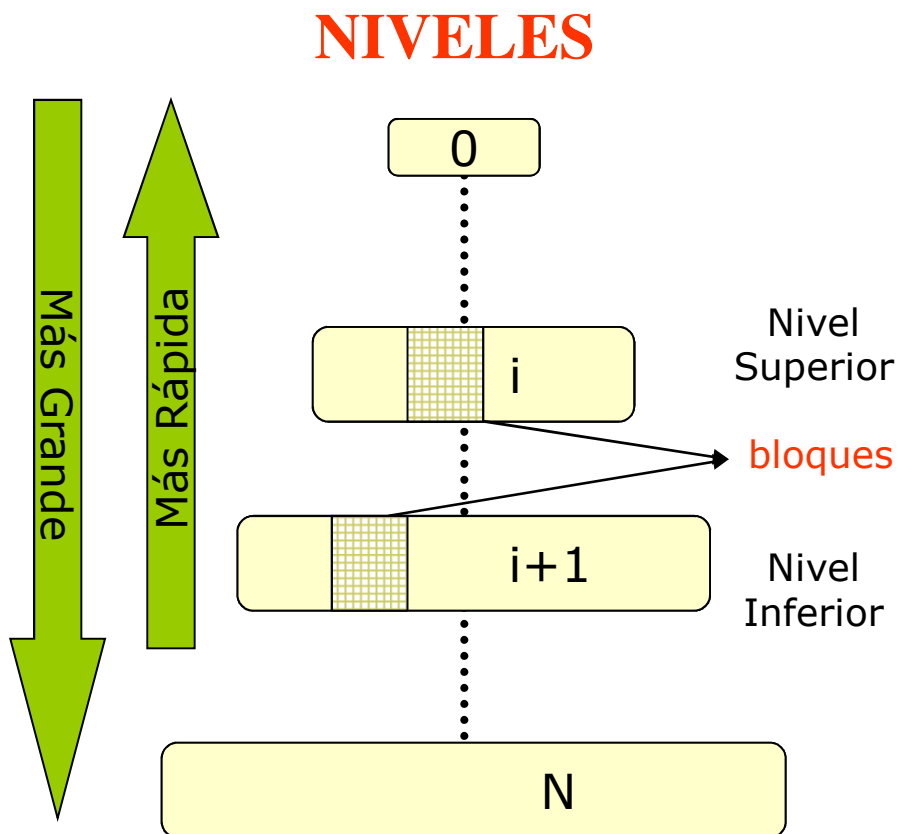
Jerarquía en un sistema actual



Ejemplos de jerarquías de memoria



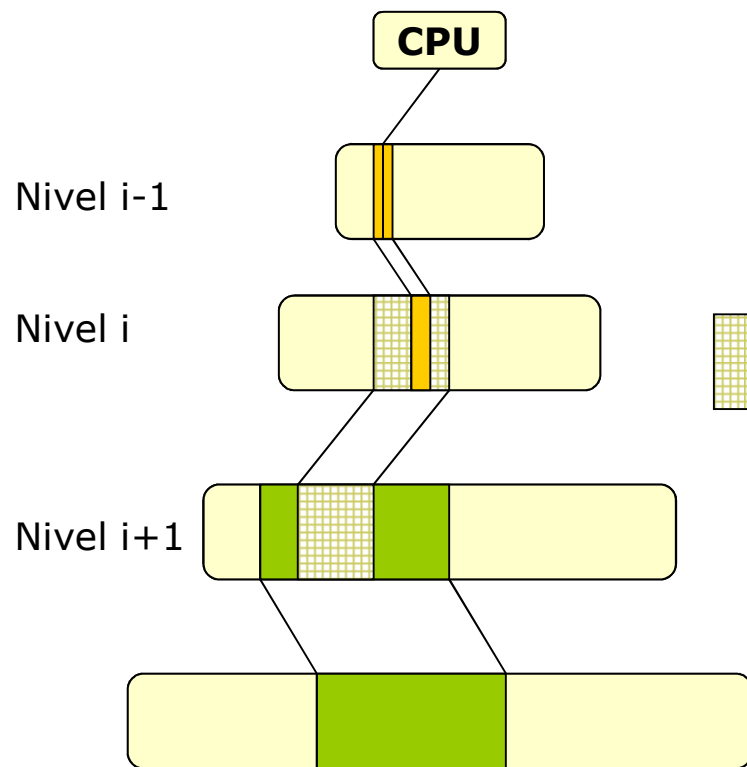
Terminología básica



BLOQUE

- Mínima unidad de información en una jerarquía de dos niveles.
- El nivel superior -el más cercano al procesador es mas rápido y pequeño que el nivel inferior.

Comunicación entre niveles



- Los bloques entre niveles adyacentes deben tener el mismo tamaño.

Bloque de nivel i

- Los tamaños dependen del nivel de la jerarquía, la granularidad de los datos, el diseño, etc.

Terminología básica (1)

- ★ **Acierto** (hit) : un acceso a un bloque de memoria que se encuentra en el nivel superior
- ★ **Fallo** (miss) : el bloque no se encuentra en ese nivel
- ★ **Frecuencia de aciertos** : fracción de accesos a memoria encontrados en el nivel superior
- ★ **Frecuencia de fallos** ($1 - \text{frecuencia de aciertos}$) : fracción de accesos a memoria no encontrados en el nivel superior

Terminología básica (2)

- ★ **Tiempo de acierto** : tiempo necesario para acceder a un dato presente en el nivel superior de la jerarquía

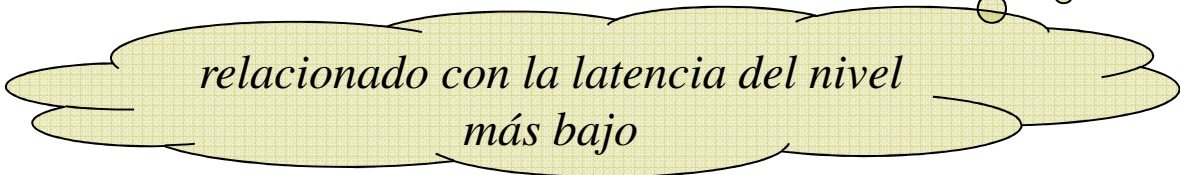
*incluye el tiempo necesario para saber si el acceso es un **acierto** o un **fallo***

- ★ **Tiempo de fallo** : tiempo necesario para sustituir un bloque de nivel superior por el correspondiente bloque de nivel inferior.
- ★ **Penalización de fallo** : tiempo de fallo + tiempo de acierto

Tiempo de fallo...

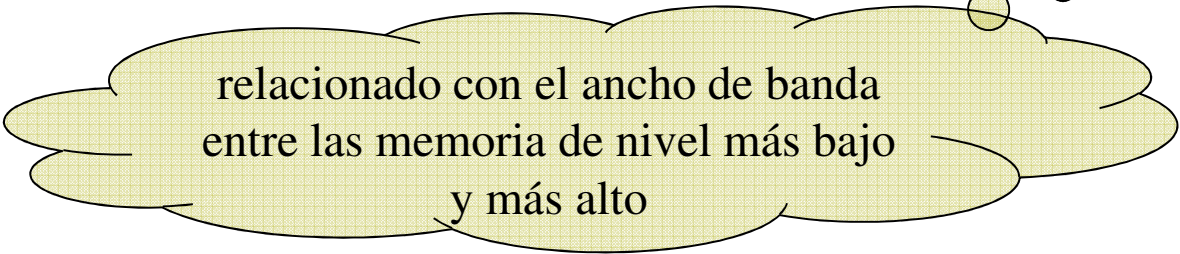
2 componentes:

tiempo de acceso : tiempo necesario para acceder a la primera palabra de un bloque en un fallo



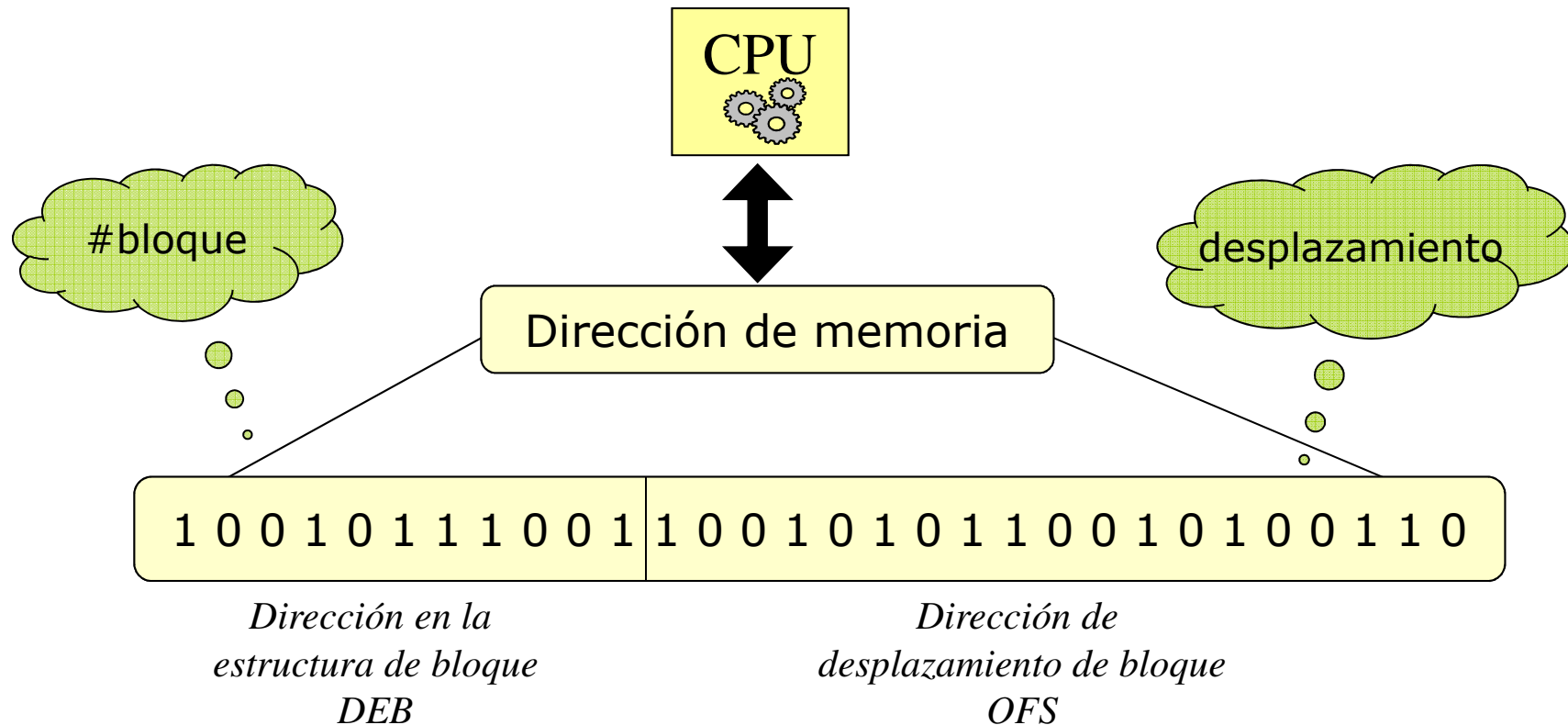
*relacionado con la latencia del nivel
más bajo*

tiempo de transferencia : tiempo para transferir las restantes palabras del bloque



*relacionado con el ancho de banda
entre las memoria de nivel más bajo
y más alto*

Direccionamiento



Rendimiento de la jerarquía

$$T_{medioAcc} = f_{acierto} * T_{acierto} + f_{fallo} * P_{fallos}$$

Ref. filmina 11

$$= f_{acierto} * T_{acierto} + f_{fallo} * (T_{acierto} + T_{fallo})$$

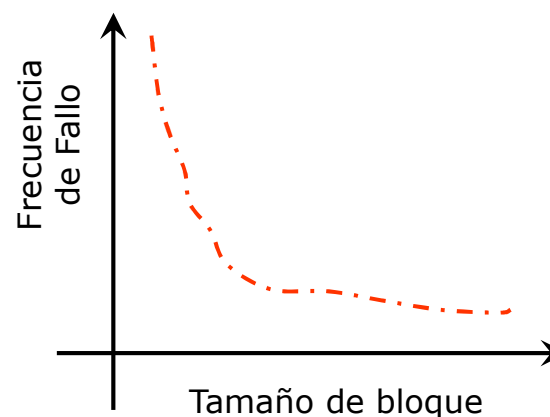
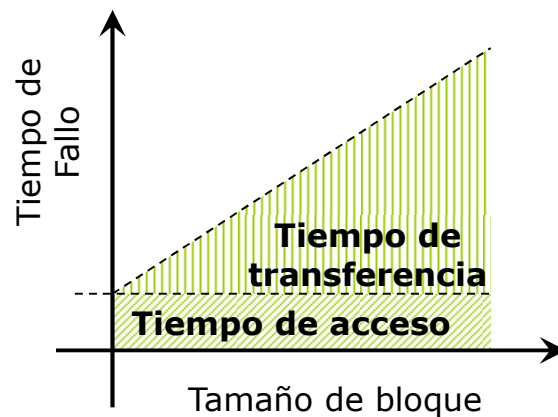
$$= f_{acierto} * T_{acierto} + f_{fallo} * T_{acierto} + f_{fallo} * T_{fallo}$$

$$= f_{acierto} * T_{acierto} + (1 - f_{acierto}) * T_{acierto} + f_{fallo} * T_{fallo}$$

$$= \cancel{f_{acierto} * T_{acierto}} + T_{acierto} - \cancel{f_{acierto} * T_{acierto}} + f_{fallo} * T_{fallo}$$

$$T_{medioAcc} = T_{acierto} + f_{fallo} * T_{fallo}$$

f = Frecuencia
 T = Tiempo
 P = Penalización



Modificaciones de la CPU

Los procesadores diseñados sin jerarquía de memoria son más simples (*los accesos a memoria siempre emplean la misma cantidad de tiempo*)

Los fallos en una jerarquía de memoria implican que la CPU debe manejar tiempos de acceso a memoria variables

- penalización del orden de decenas de ciclos
la CPU espera a que se termine el ciclo de memoria
- penalización del orden de cientos de ciclos
la CPU es interrumpida para atender otras tareas mientras dura el fallo

La CPU debe soportar interrupciones por accesos a memoria

Clasificación de las jerarquías de memoria

- ★ Ubicación del bloque

Dónde puede ubicarse un bloque en el nivel superior

- ★ Identificación del bloque

Cómo se encuentra un bloque en el nivel superior

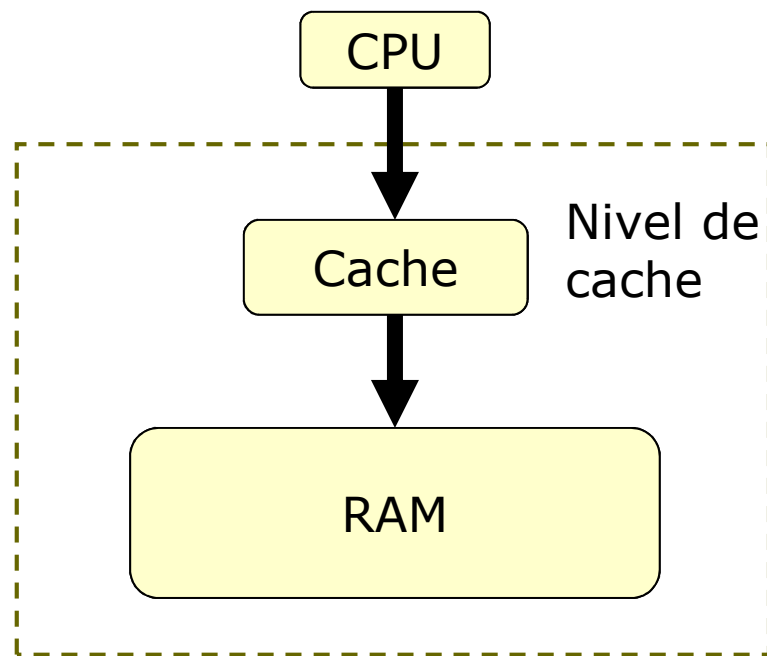
- ★ Sustitución de bloque

Qué bloque debe reemplazarse en caso de fallo

- ★ Estrategia de escritura

Qué ocurre en una escritura

Primer nivel : Memoria CACHE



CACHE

- ★ Memorias muy rápidas
- ★ Poca capacidad
- ★ Se interponen entre el procesador y la memoria principal

Ubicación de un bloque en la cache

correspondencia directa

- cada bloque debe ir solamente en un lugar dentro de la cache

asociativa

- un bloque puede ubicarse en cualquier lugar de la cache

asociativa por conjuntos

- un bloque puede ser colocado en un grupo restringido de lugares de la cache



un conjunto es un grupo de dos o más bloques de la cache

Ecuaciones de ubicación de bloques

correspondencia directa

- $\text{ubicación} = (\# \text{bloque}) \bmod (\# \text{bloques de cache})$

asociativa

- $\text{ubicación} = \text{cualquiera}$

asociativa por conjuntos

- $\text{ubicación} = (\# \text{bloque}) \bmod (\# \text{de conjuntos de cache})$

Notas . . .



Los espacios donde se ubican los bloques en la cache se denominan *líneas* (*line*)



En una cache asociativa por conjuntos, si hay n bloques por conjunto, la cache se llama asociativa por conjuntos de n vías (asociatividad n)



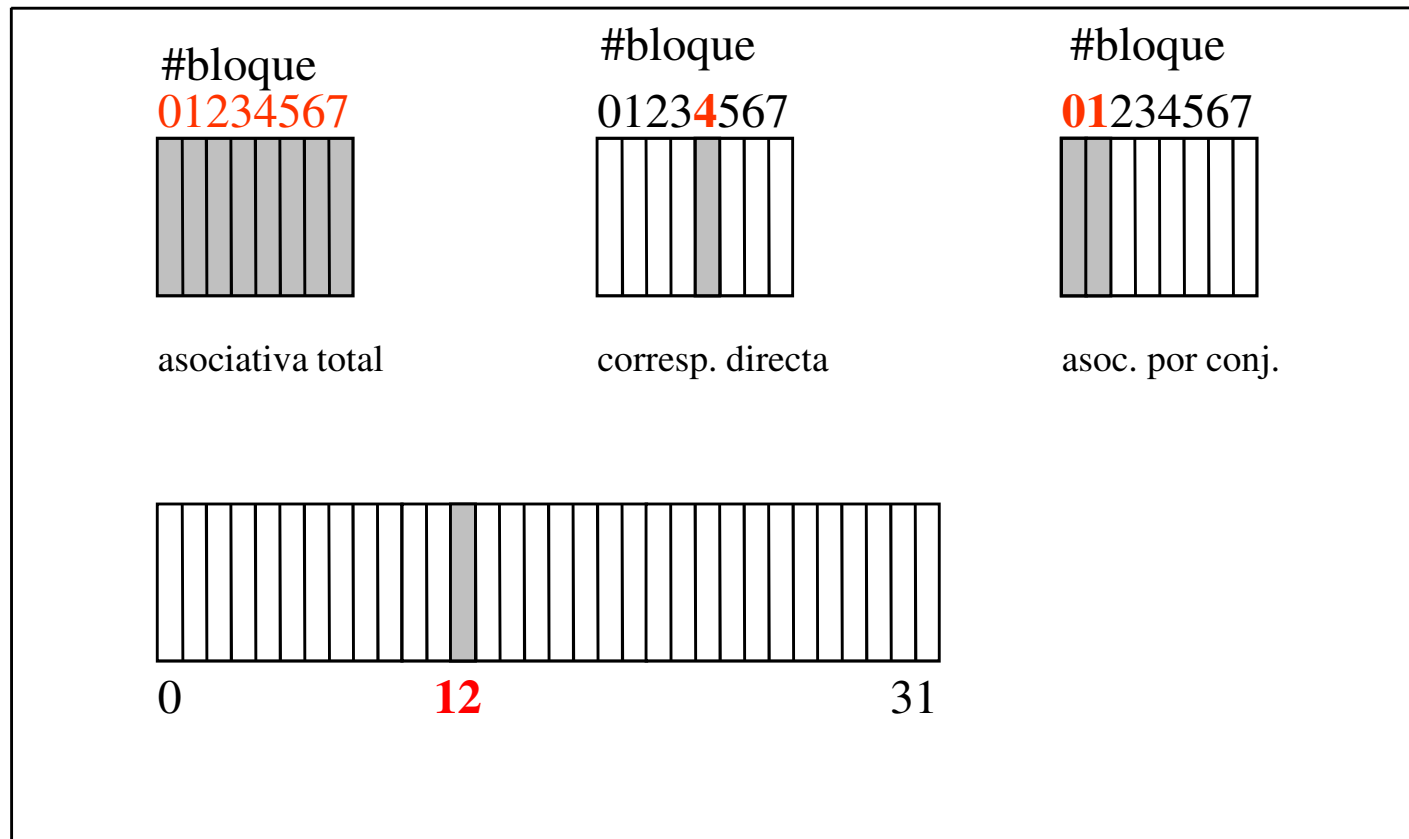
Una cache de correspondencia directa podría decirse que es asociativa por conjuntos de una sola vía



Una cache totalmente asociativa posee un solo conjunto con grado de asociatividad m (si posee m bloques en total)

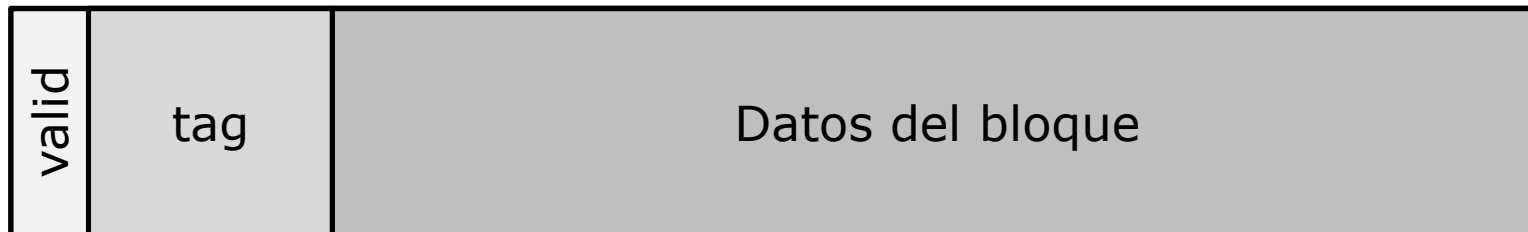
Ejemplo

- cache de 8 líneas
- memoria de 32 bloques.
- cache asociativa por conjuntos tiene 4 conjuntos de 2 bloques c/u



Identificación de un bloque de cache

- Dado que en una línea puede haber distintos bloques es necesario saber cual #bloque se almacenó
 - Para esto, se guarda junto al bloque parte de la dirección en memoria RAM (campo de etiqueta ó *tag*)
 - Para saber si la línea contiene bloques válidos se usa un bit de validez (valid bit)
 - Valid-bit = 0 línea vacía (sin bloques)
 - Valid-bit = 1 bloque presente



Ejemplo

Un sistema de memoria tiene una cache de acceso directo de 16 líneas y bloques de 8 palabras c/u. La memoria RAM es de 1K palabras.

El procesador del sistema realiza 2 accesos al sistema de memoria para buscar las palabras de dirección 296 y 942.

Hallar, para cada palabra:

- A qué bloque (de RAM) pertenece
- Su posición dentro del bloque
- En que línea de cache se ubicará
- Su etiqueta

Ejemplo

$$\text{Dir1} = 296_{(10)} = \mathbf{0100101000}_{(2)}$$

- Posición dentro del bloque
Offset = $296 \bmod 8$
= $0_{(10)} = \mathbf{000}_{(2)}$
- #bloque = $296 \div 8$
= $37_{(10)} = \mathbf{0100101}_{(2)}$
- Línea = $37 \bmod 16$
= $5_{(10)} = \mathbf{0101}_{(2)}$
- Etiqueta = $37 \div 16$
= $2_{(10)} = \mathbf{010}_{(2)}$

0 1 0 **0 1 0 1** 0 0 0

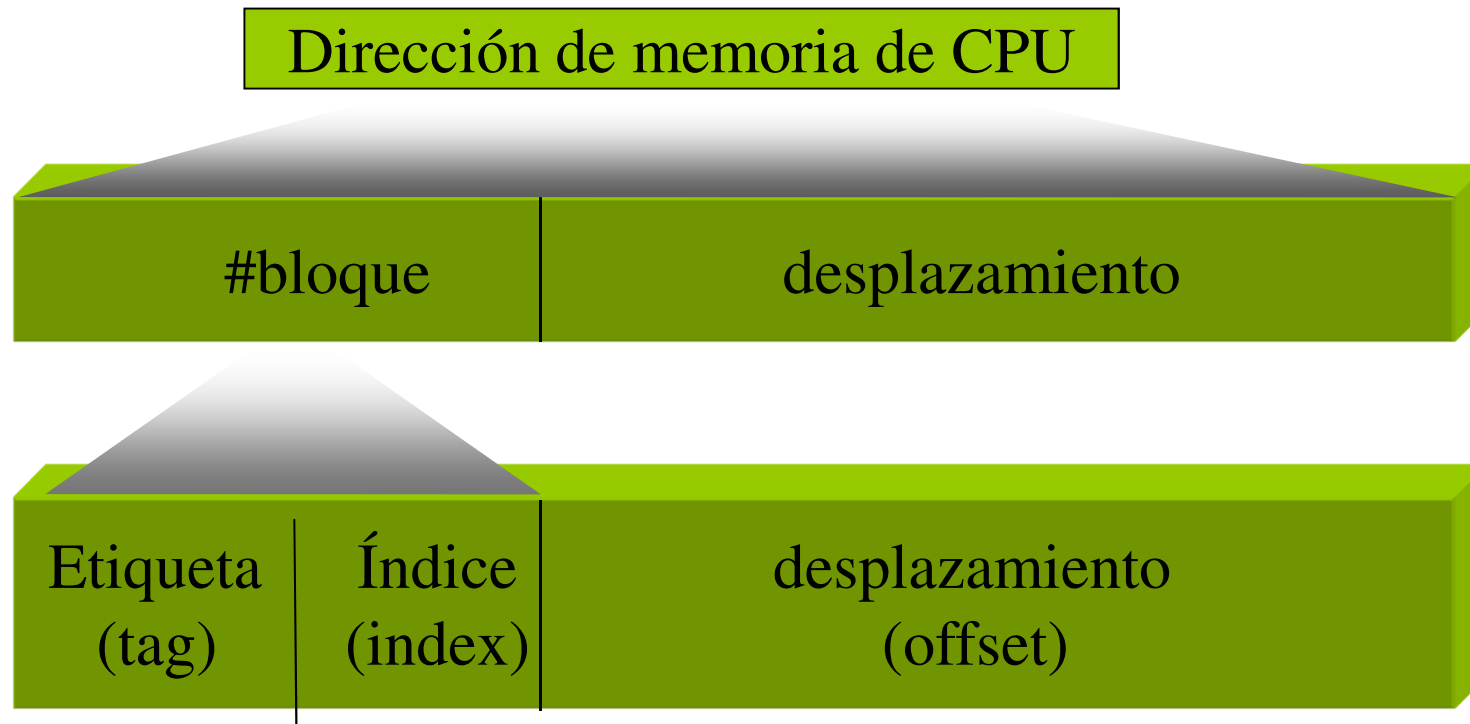
$$\text{Dir1} = 942_{(10)} = \mathbf{1110101110}_{(2)}$$

- Posición dentro del bloque
Offset = $942 \bmod 8$
= $6_{(10)} = \mathbf{110}_{(2)}$
- #bloque = $942 \div 8$
= $117_{(10)} = \mathbf{1110101}_{(2)}$
- Línea = $117 \bmod 16$
= $5_{(10)} = \mathbf{0101}_{(2)}$
- Etiqueta = $117 \div 16$
= $7_{(10)} = \mathbf{111}_{(2)}$

1 1 1 **0 1 0 1** 1 1 0 24

Misma línea

Esquema de direccionamiento



Etiqueta (Tag): Descriptor de identificación del bloque en cache

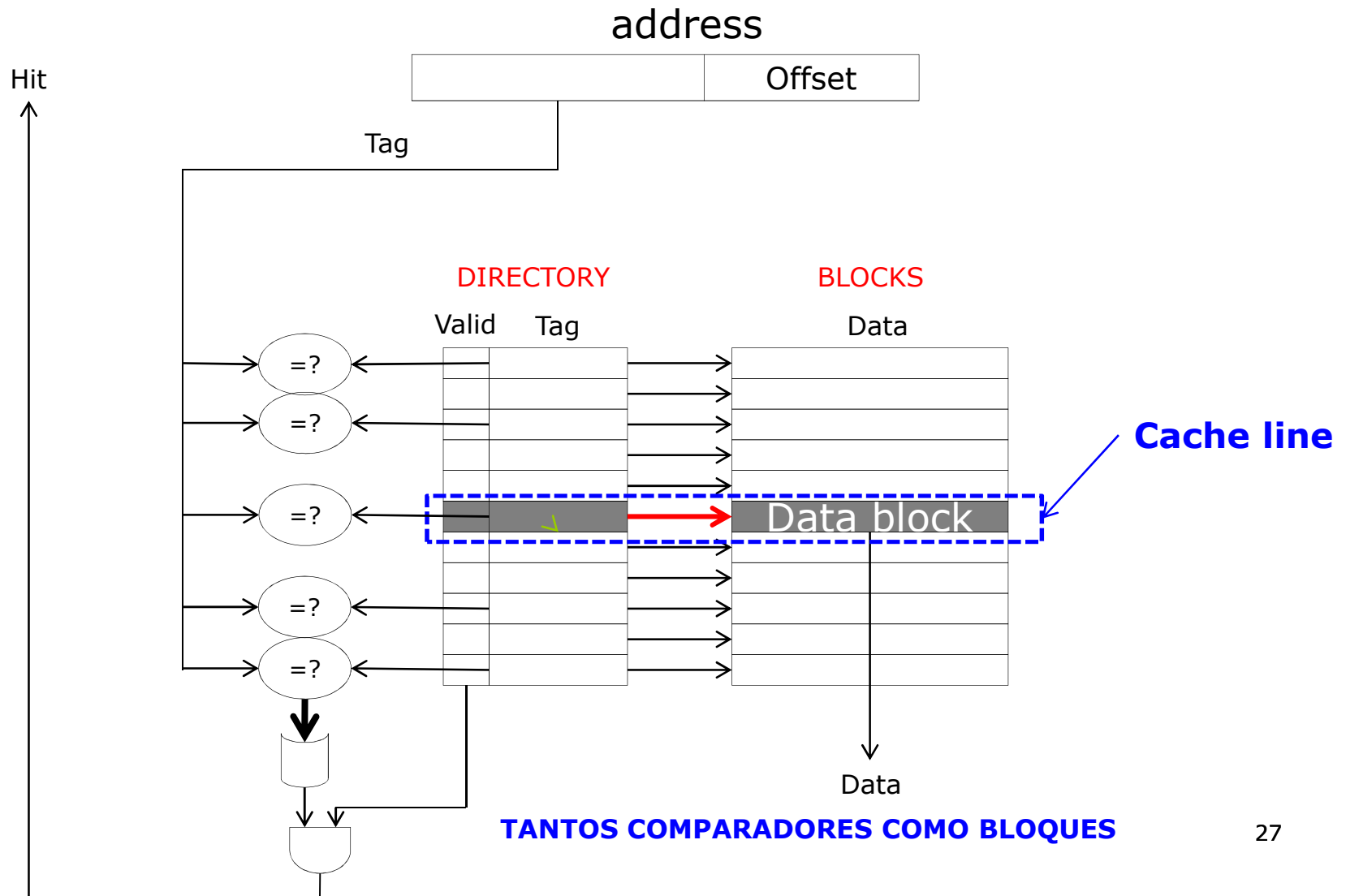
Índice (index): Indicador de línea de cache del bloque

Desplazamiento (offset): Posición de la palabra buscada en el bloque²⁵

Age Group	Percentage
18-24	15%
25-34	25%
35-44	30%
45-54	20%
55-64	10%
65-74	5%
75-84	10%
85+	5%

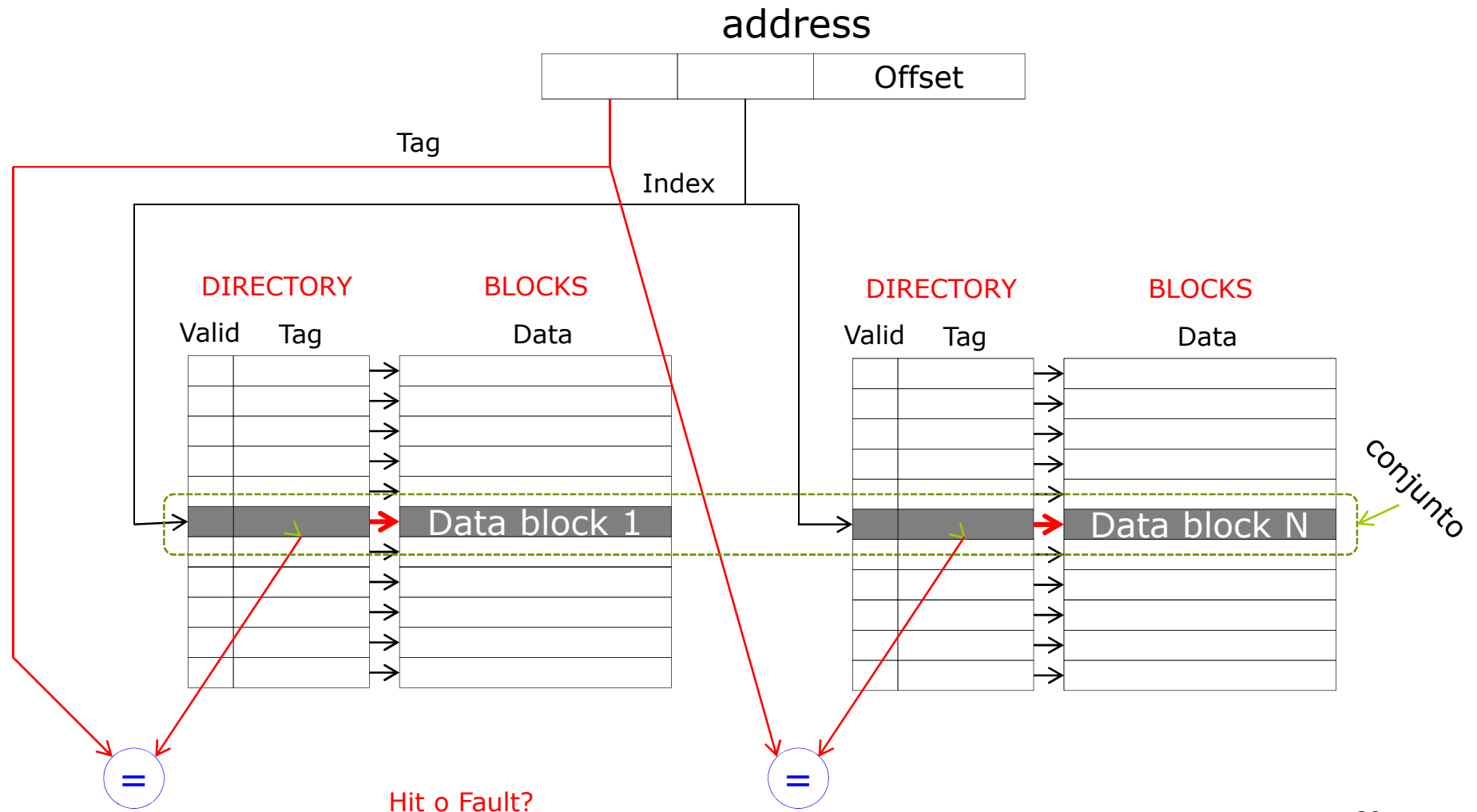


Esquema de la cache (Totalmente Asociativa - FA)



Esquema de la cache

(Asociativa por conjuntos de N vías – N-A)



TANTOS COMPARADORES COMO VIAS

Mas notas...



El campo índice se usa para seleccionar la línea, y el de etiqueta para la comparación



Si se incrementa la asociatividad (para la misma capacidad):

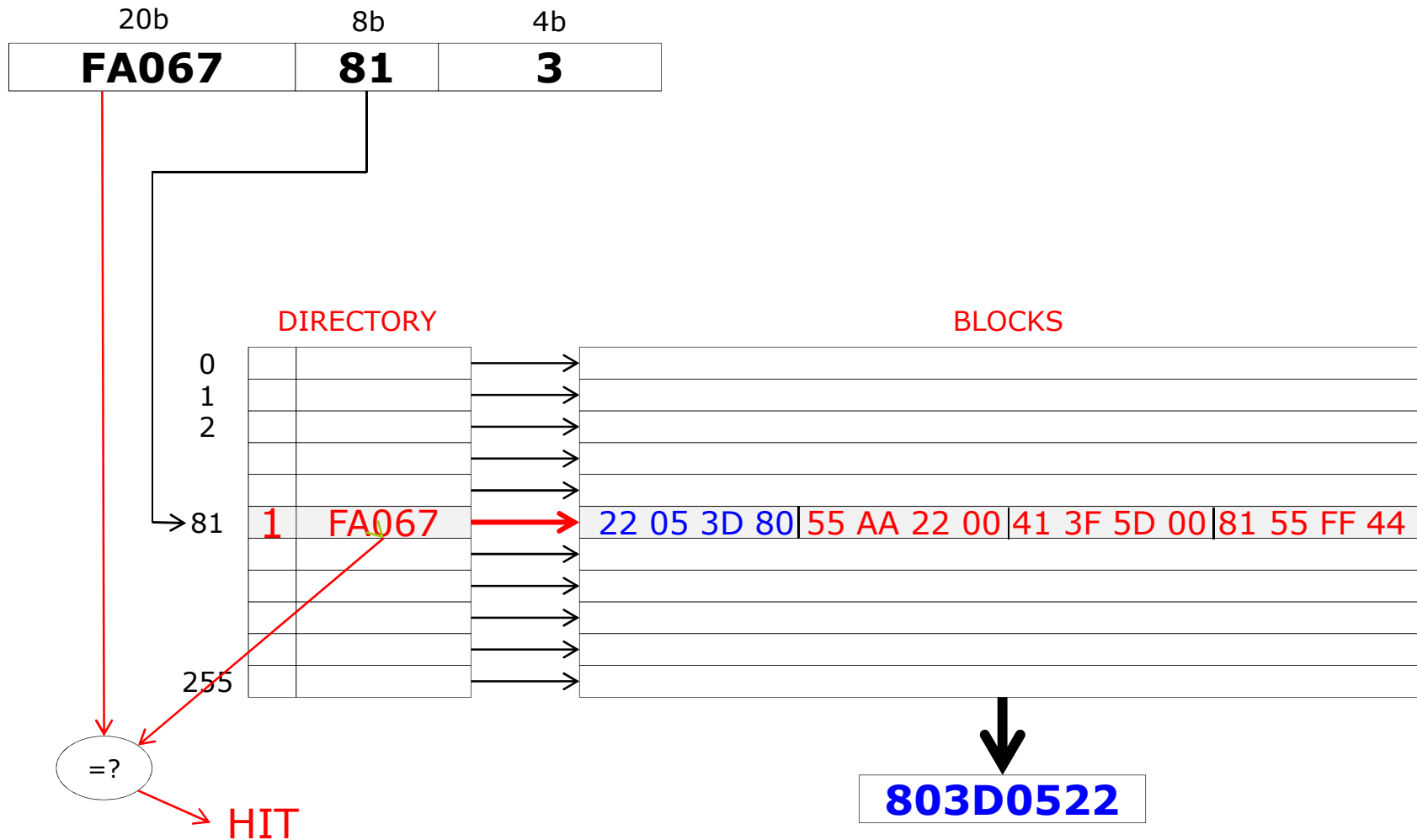
- aumenta el número de bloques por conjunto
- disminuye el tamaño del índice
- aumenta el tamaño de la etiqueta



Una cache totalmente asociativa no tiene índice y la parte de etiqueta posee la dirección de bloque total.

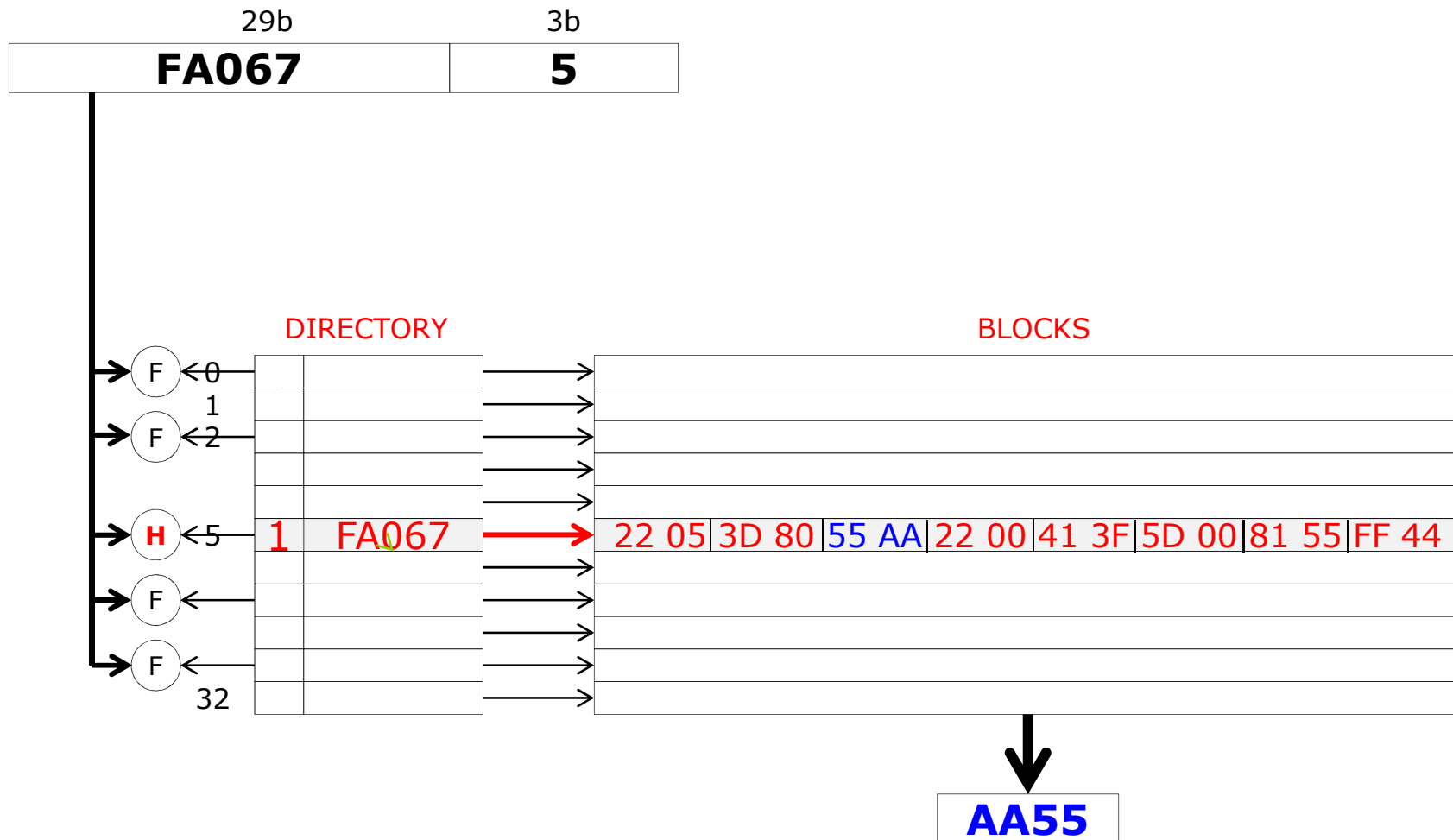
Ejemplo

Cache 4KB, 4 palabras(32b)/bloque

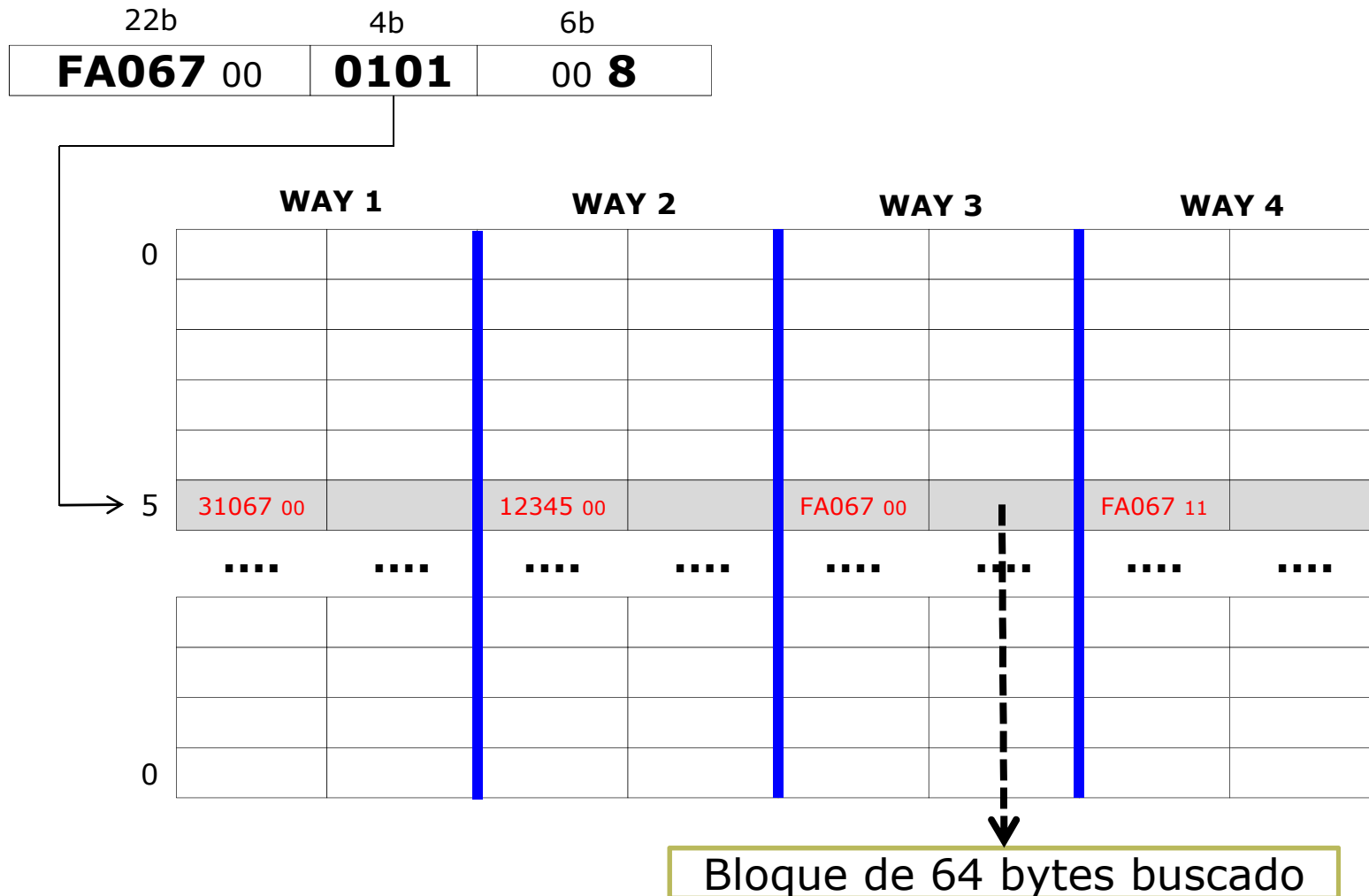


Ejemplo

Cache FA 512B, 8 palabras(16b)/bloque



Cache 4-A 4KB, 64 palabras(8b)/bloque



Mas notas...

- Aumentar el tamaño de bloques reduce la frecuencia de fallos (localidad espacial)
- Aumentar el tamaño de bloques disminuye la cantidad de bloques aumentando la competencia dentro de la cache e incrementando la frecuencia de fallos
- Aumentar el tamaño de bloques aumenta la penalización de fallos
- Estrategias de optimización:
 - Early restart on miss
 - Critical-word-first on miss
 - Read priority over write
 - Merging write buffer

Sustitución de bloques

Ante un fallo de cache es necesario traer un bloque nuevo y ubicarlo en algún lugar del nivel superior

Si existe algún bloque de cache con datos no válidos, el reemplazo se hace en ese lugar.

Debido a la alta frecuencia de aciertos de la cache es necesario tomar estrategias de reemplazo.

Sustitución de bloques

- ➡ Si la cache es de mapeo directo no hay problema, ya que el nuevo bloque puede ir en un solo lugar
- ➡ En caso de caches asociativas el bloque puede ubicarse en diferentes lugares.

Estrategias:

- reemplazo aleatorio
- reemplazo pseudo aleatorio
- LRU (menos recientemente usado)

Sustitución de bloques

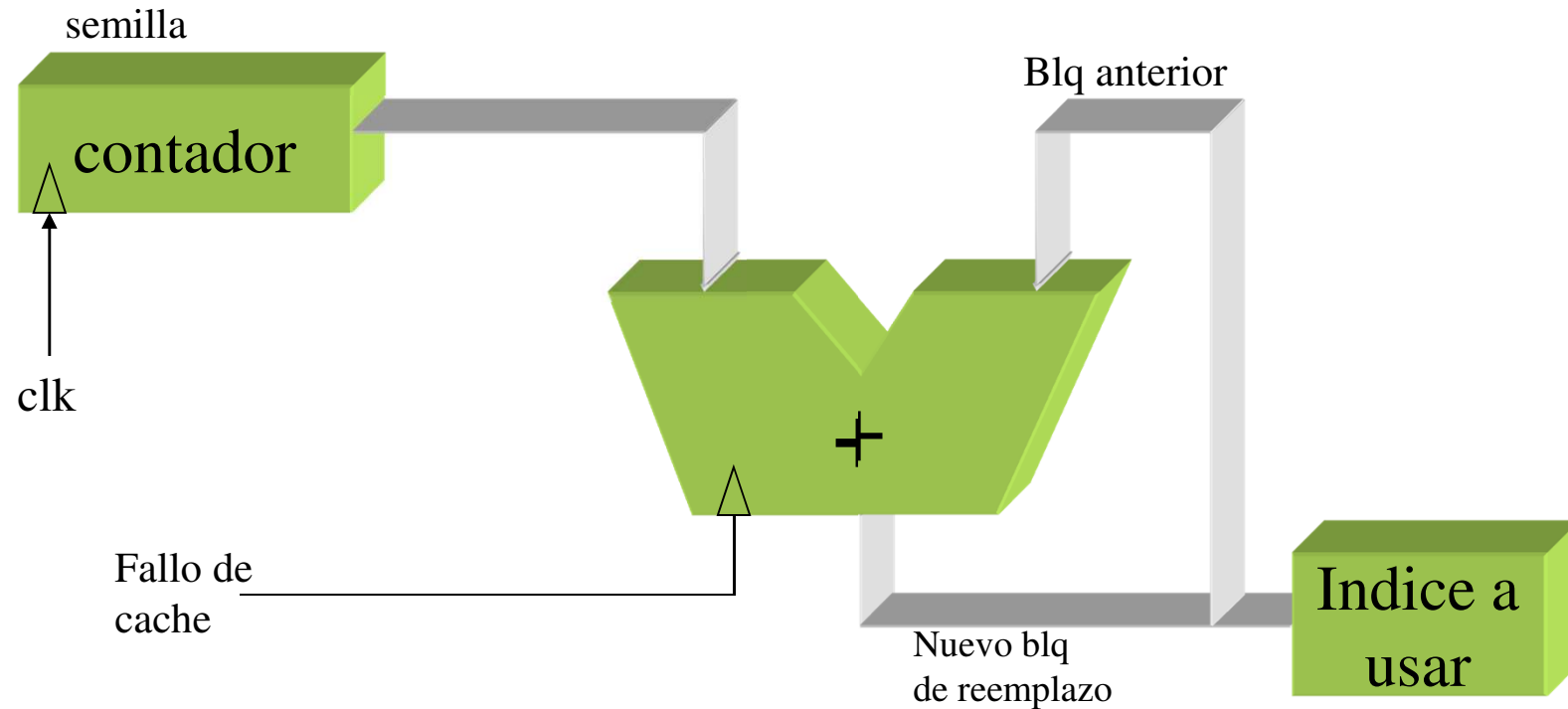
Aleatorio : Mediante algún mecanismo de hardware se seleccionan uniformemente los bloques candidatos.

Pseudo aleatorio : para obtener una distribución de bloques con un comportamiento reproducible útil durante la depuración del hardware.

LRU (Least Recently Used):

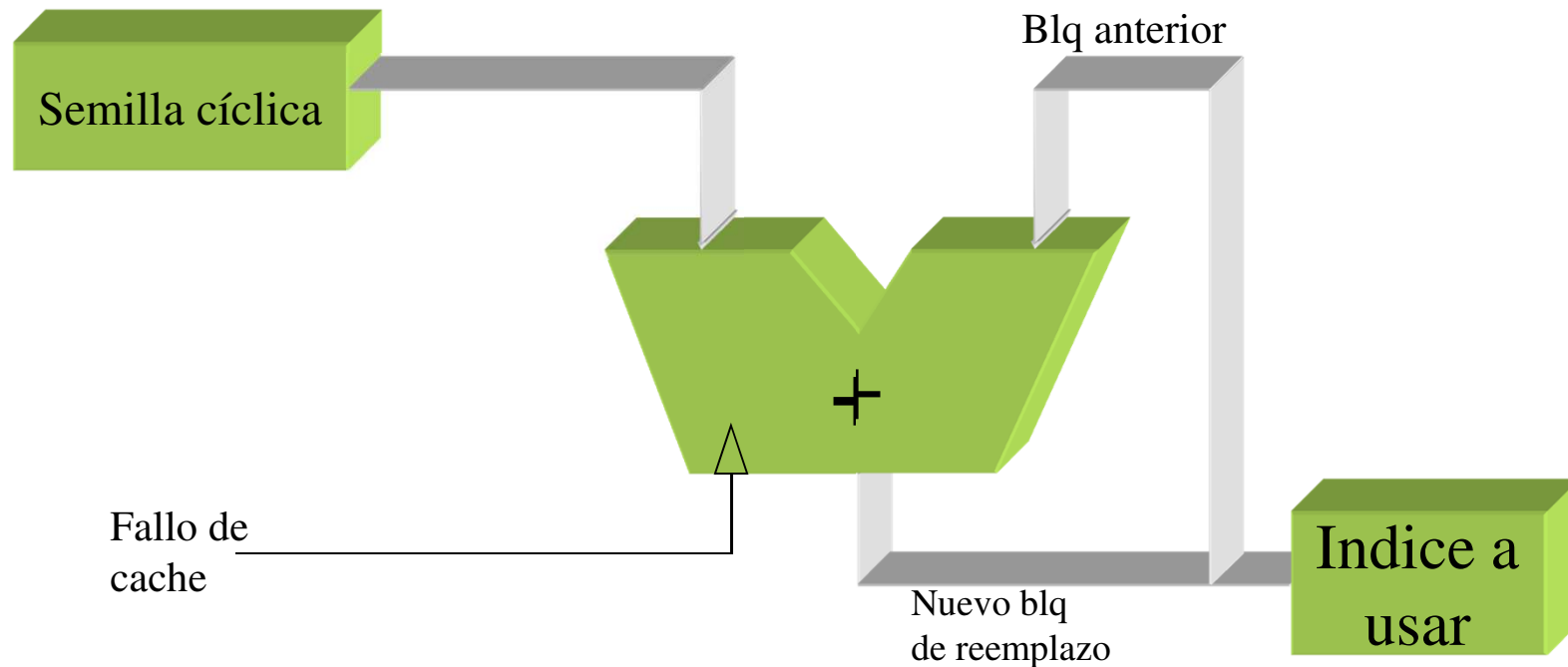
- disminuye la posibilidad de desechar información que se necesitará pronto
- se registran los accesos a los bloques de manera que el bloque sustituido es aquel que hace más tiempo que no se usa.

Mecanismo aleatorio



$$X(i+1) = [m.X(i) + k] \bmod n$$

Mecanismo pseudo aleatorio



En lugar de una ecuación basada en el reloj para calcular el próximo valor se usa una semilla cíclica fija

Para ahorrar la demora del sumador se calcula el bloque para el siguiente fallo y se usa el anterior para atender el fallo actual

Mecanismo LRU

Agrega bits adicionales a cada línea asociativa para registrar la longevidad de la línea

Cantidad de bits LRU = \log_2 #líneas en la mem

LRU

0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
valid	valid	valid	valid	valid	valid	valid	valid
Tag	Tag	Tag	Tag	Tag	Tag	Tag	Tag
Block	Block	Block	Block	Block	Block	Block	Block

Asociativa

Asoc. Por conjuntos

LRU 0		LRU 1		LRU 2		LRU 3	
0	0	0	0	0	0	0	0
valid	valid	valid	valid	valid	valid	valid	valid
Tag	Tag	Tag	Tag	Tag	Tag	Tag	Tag
Block	Block	Block	Block	Block	Block	Block	Block
0		1		2		3	

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	0	0	0	0
LRU	00	00	00	00

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	0	0	0
LRU	00	01	01	01
	0			

- Elige una línea libre (valid = 0)
- Si hay, pone el bloque en esa línea
- Resetea su contador LRU
- Incrementa el resto de los contadores LRU

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	1	0	0
LRU	01	00	10	10
	0	2		

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	1	1	0
LRU	10	01	00	11
	0	2	4	

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	1	1	1
LRU	11	10	01	00
	0	2	4	1

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	1	1	1
LRU	00	11	10	01
	5	2	4	1

- Si no hay línea disponible (todos los valid = 1)
Se elige el LRU mas grande (mas longevo)
Se reemplaza esa línea

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7

valid	1	1	1	1
LRU	01	00	11	10
	5	10	4	1

- Si es un acierto solo se actualizan los contadores LRU
- Los contadores LRU saturan en el máximo

Ejemplo LRU

Suponga una secuencia de accesos a memoria que deriva en la siguiente secuencia de accesos a bloque en una jerarquía con una cache asociativa de 4 líneas y mecanismo de reemplazo LRU:

0, 2, 4, 1, 5, 10, 5, 4, 3, 15, 8, 7, 7, 8, 8, 8, 8, 7




valid	1	1	1	1
LRU	00	01	11	11
acierto →	5	10	4	1

- Si es un acierto solo se actualizan los contadores LRU
- Los contadores LRU saturan en el máximo

Problema

- Continúe la resolución del ejemplo anterior hasta el final de la lista de bloques
- Repita el ejercicio pero suponiendo una memoria cache de 4 conjuntos asociativos de 2 líneas cada uno

Más notas...

-  La estrategia aleatoria es sencilla de construir en hardware
-  Cuando el número de bloques a gestionar aumenta, la estrategia LRU es muy cara, frecuentemente se utiliza una aproximación de ésta.
-  La política de reemplazo juega un papel más importante en las caches más pequeñas que en las más grandes donde hay más opciones para sustituir.

Asociatividad tamaño	2 vías		4 vías		8 vías	
	LRU	aleatorio	LRU	aleatorio	LRU	aleatorio
16 Kb	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
64 Kb	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
256 Kb	1.15%	1.17%	1.17%	1.13%	1.12%	1.12%

Estrategias de escritura

Lecturas $\approx 91\%$

El bloque se lee simultáneamente a la lectura de la etiqueta para que la lectura del bloque comience tan rápido como esté disponible la DEB

- Si la lectura es un acierto el bloque pasa inmediatamente a la CPU
- Si es un fracaso no hay beneficio, pero tampoco perjuicio

Escrituras $\approx 9\%$

El procesador especifica el tamaño de la escritura -normalmente entre 1 y 8 bytes- y sólo esa porción del bloque debe ser cambiada

Estrategias de escritura

Políticas de escritura:

Escritura directa (Write through): la información se escribe en el bloque de cache y también en el bloque de memoria de nivel inferior.

Postescritura (Write back): la información se escribe sólo en el bloque de la cache. El bloque modificado de la cache se escribe en el nivel inferior sólo cuando es reemplazado.

Ventajas de cada política

Postescritura

- las escrituras se realizan a velocidad de la cache
- múltiples escrituras de un bloque requieren una única escritura en la memoria de nivel inferior

Escritura directa

- los fallos de lectura no ocasionan escrituras en el nivel inferior
- más fácil de implementar que el anterior
- mantiene siempre coherente la memoria cache y la memoria inferior (útil en sistemas multiproceso en los que varias CPU's acceden simultáneamente a los datos)

Fallos de escritura

El fallo se produce cuando se intenta escribir una palabra de un bloque que no está en la cache

Políticas

Ubicar en escritura (Write allocate on miss): el bloque se carga en la cache, seguido de las acciones anteriores de acierto de escritura. Esto es similar a un fallo de lectura.

No ubicar en escritura (Write around): el bloque se modifica en el nivel inferior y no se carga en la cache.

Rendimiento de la cache

En una Jerarquía de memoria con caché y RAM el Tiempo de CPU Se compone de las contribuciones siguientes:

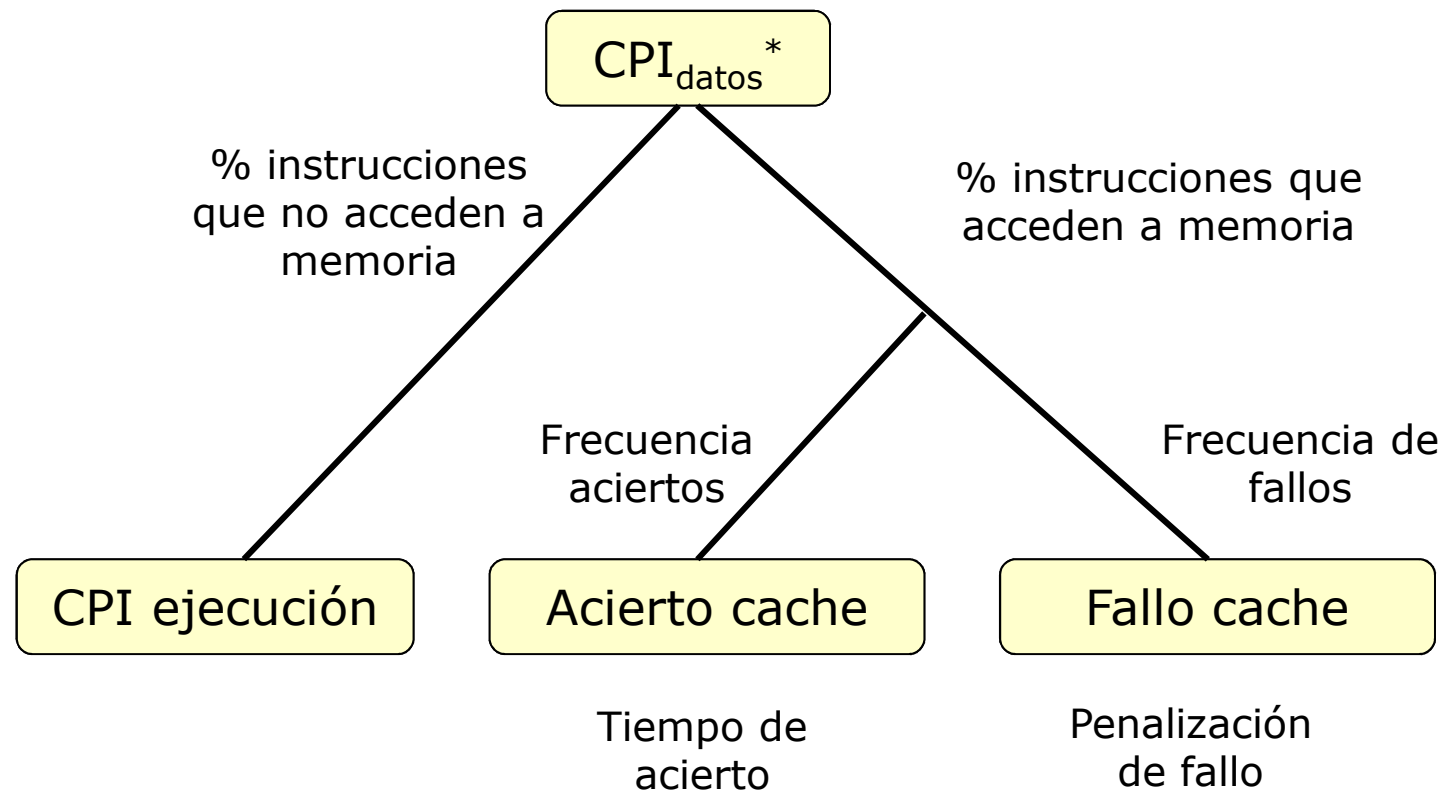
ciclos de reloj de ejecución del programa

- ejecución de instrucciones que no requieren datos de memoria

ciclos de reloj de espera al sistema de memoria

- Aciertos de caché ó
- Penalizaciones de fallo de caché

Rendimiento de la cache



* Se puede hacer similar proceso deductivo para calcular el CPI de acceso a instrucciones

Rendimiento de la cache

$$T_{CPU} = \left(Ciclos_{ejecuciónCPU} + Ciclos_{acierto} + Ciclos_{fallo} \right) * T_{ciclo - reloj}$$

Factorizando el recuento de instrucciones:

$$T_{CPU} = RI * \left(CPI_{ejecución} + f_{acierto} * \frac{aciertos}{instrucción} + f_{fallo} * P_{fallo} \right) * T_{ciclo}$$

$f = Frecuencia$ $T = Tiempo$ $P = Penalización$ $RI = RecuentoInstrucciones$
--





Tipos de fallos de cache

Forzosos : el primer acceso a un bloque no está en la cache; así que el bloque debe ser traído a la misma. Estos también se denominan fallos de arranque en frío o de primera referencia.

Capacidad : si la cache no puede contener todos los bloques necesarios durante la ejecución de un programa, se presentarán fallos de capacidad debido a los bloques que se descartan y luego se recuperan.

Conflicto : si la estrategia de ubicación es asociativa por conjuntos o de mapeo directo, estos fallos ocurrirán, ya que se puede descartar un bloque y posteriormente recuperarlo si a un conjunto le corresponden demasiados bloques.

Mas notas...

-  Los fallos forzosos son independientes de la cache
-  Los de capacidad disminuyen cuando la capacidad aumenta
-  Los fallos de conflicto dependen de la asociatividad de la memoria: si es totalmente asociativa no existen conflictos, en la de mapeo directo los conflictos aumentan hasta su máximo posible
-  Incrementar la capacidad de la cache reduce los fallos de conflicto así como los de capacidad, ya que una cache mayor dispersa las referencias.

Caches de Data/Inst o unificadas

Cache de datos e instrucciones separadas:

- La CPU sabe que tipo de lectura realiza (D/I)
- Aumenta el ancho de banda (política Harvard)
- Permite optimizar los parámetros independientemente
- Disminuye los fallos de conflicto en la cache de inst.

Cache unificada:

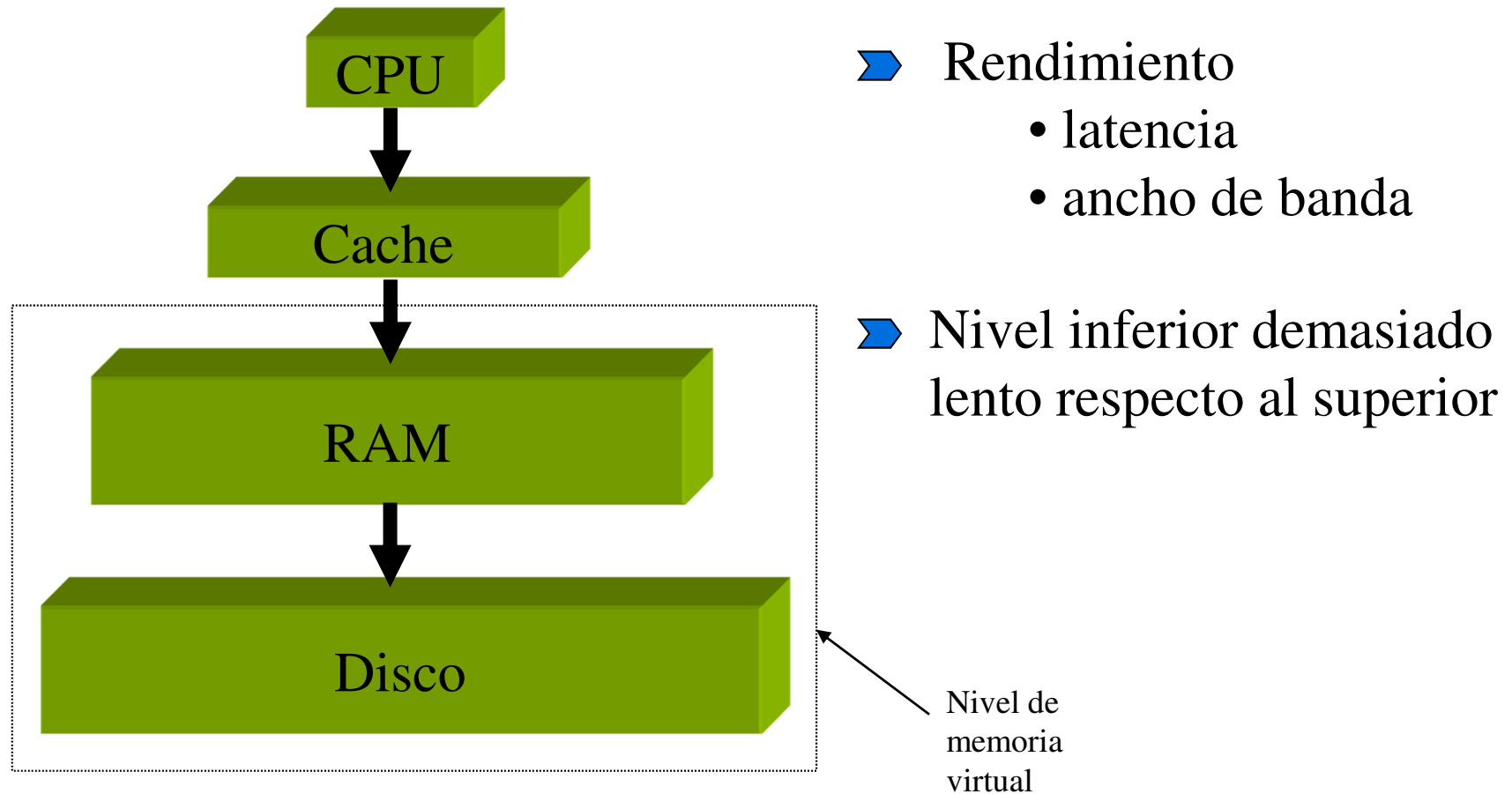
- Más simple de implementar
- No se necesitan puertos duales en la CPU para D/I
- No limita el espacio máximo para datos e inst. (fallos de capacidad)
- (política Von Newmann)



Segundo nivel de la jerarquía

Memoria virtual

Segundo nivel: memoria virtual



Segundo nivel: memoria virtual

Características

- La memoria RAM actúa como cache del disco
- Administrada por HW y sistema operativo
- Los programas comparten la memoria principal entre ellos
 - Separación del espacio de direcciones
 - Protección entre programas

Segundo nivel: memoria virtual

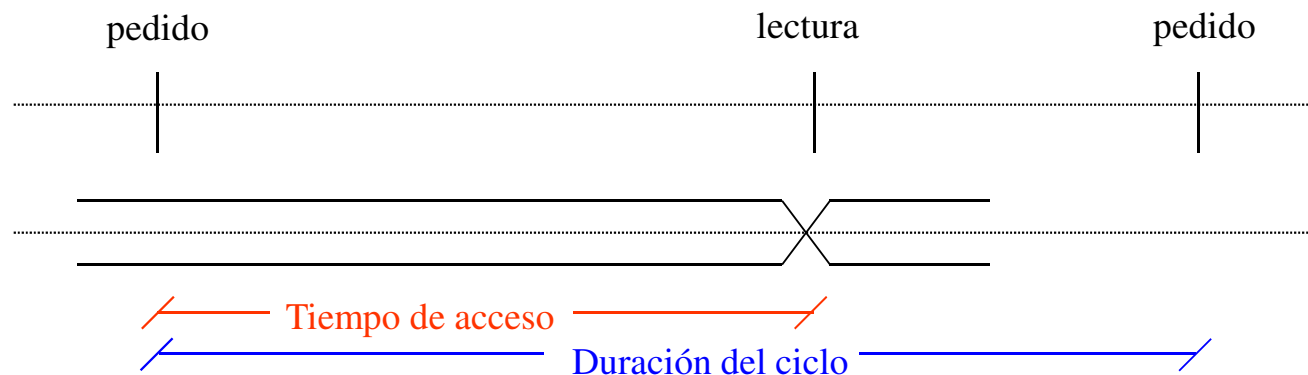
Diferencias con nivel de cache

	Nivel cache	Mem. virtual
Tamaño de bloque	16 a 128 bytes	4096 a 65536 bytes
Tiempo de acierto	1 a 2 ciclos	40 a 100 ciclos
Penaliz. De fallo	8 a 100 ciclos	700K a 6000K ciclos
Tiempo de acceso	6 a 60 ciclos	500K a 4000K ciclos
Tiempo de transf.	2 a 40 ciclos	200K a 2000K ciclos
Frecuencia de fallo	1 a 10%	< 0,001%
Tamaño de memoria	16Kb a 1Mb	16 a 8192Mb

Latencia de la memoria RAM

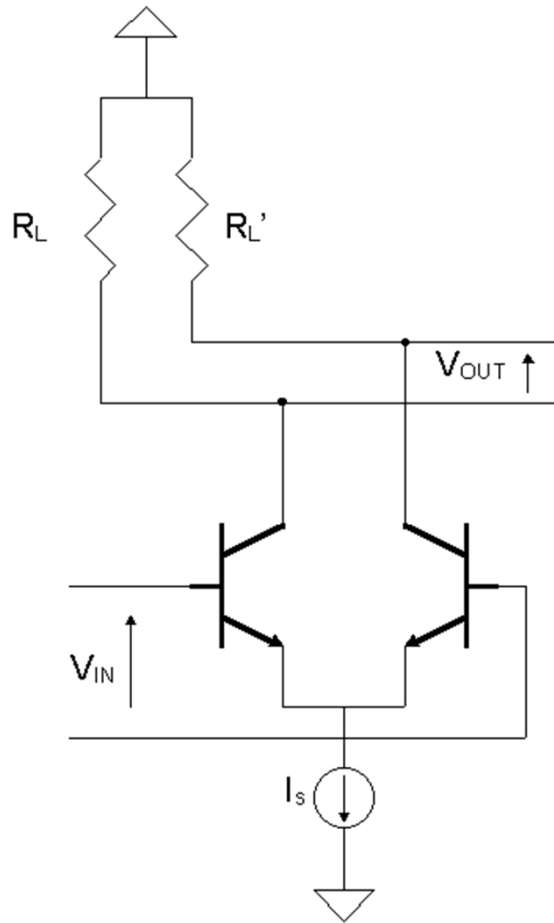
latencia :

- Tiempo de acceso : tiempo que transcurre desde que se pide una lectura hasta que llega la palabra deseada
- Duración del ciclo : tiempo mínimo entre peticiones consecutivas a memoria



Aspectos físicos de las RAM

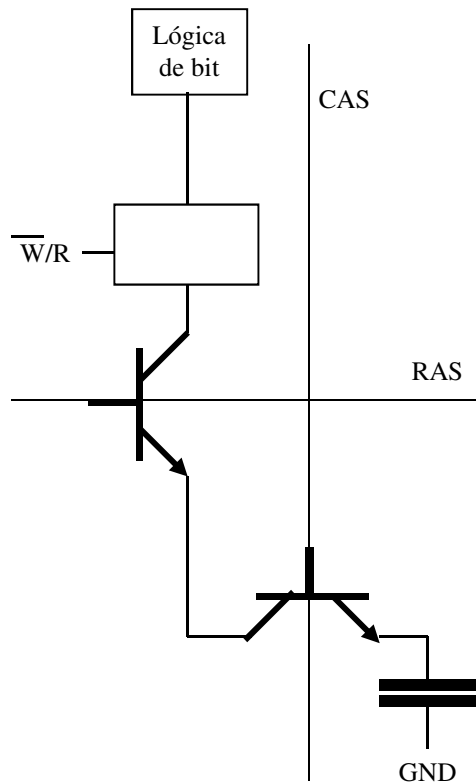
SRAM (Static RAM)



- Construidas a partir de 2 transistores por bit.
- Tecnología más cara
- Menor capacidad
- Mayor velocidad
- Usadas en las memorias cache

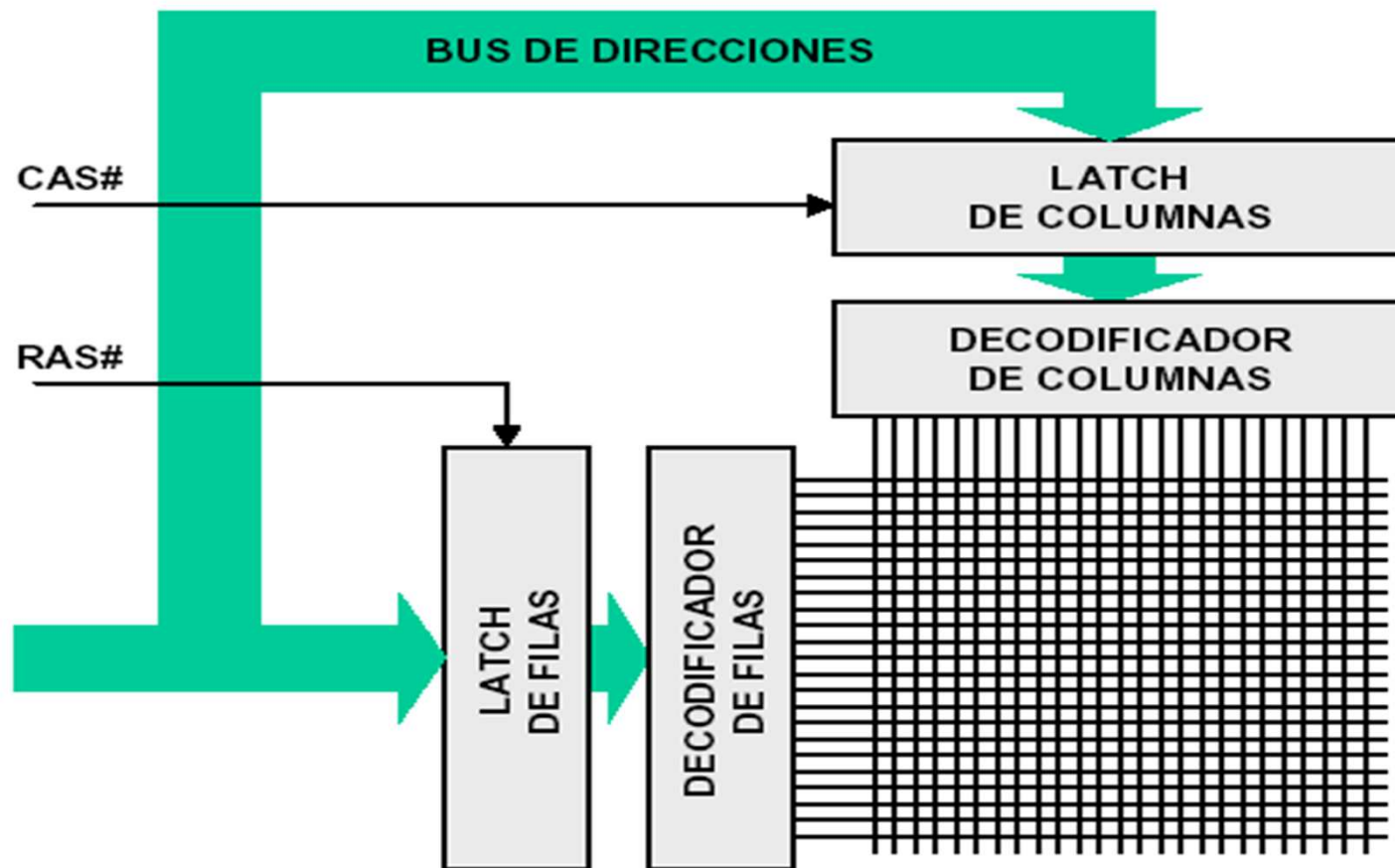
Aspectos físicos de las RAM

DRAM (Dinamic RAM)



- Usan un capacitor como elemento de memoria
- Direccionamiento doble
 - RAS
 - CAS
- Tecnología más barata
- Mayor capacidad
- Más lentas
- Usadas en las memorias de masa

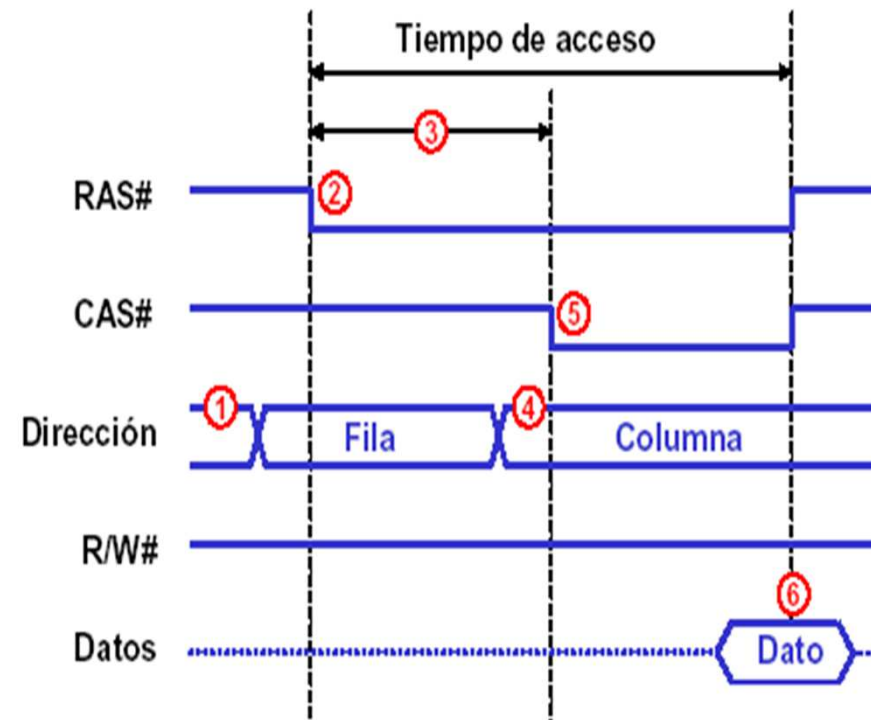
Aspectos físicos de las RAM



Cronograma de acceso a una DRAM

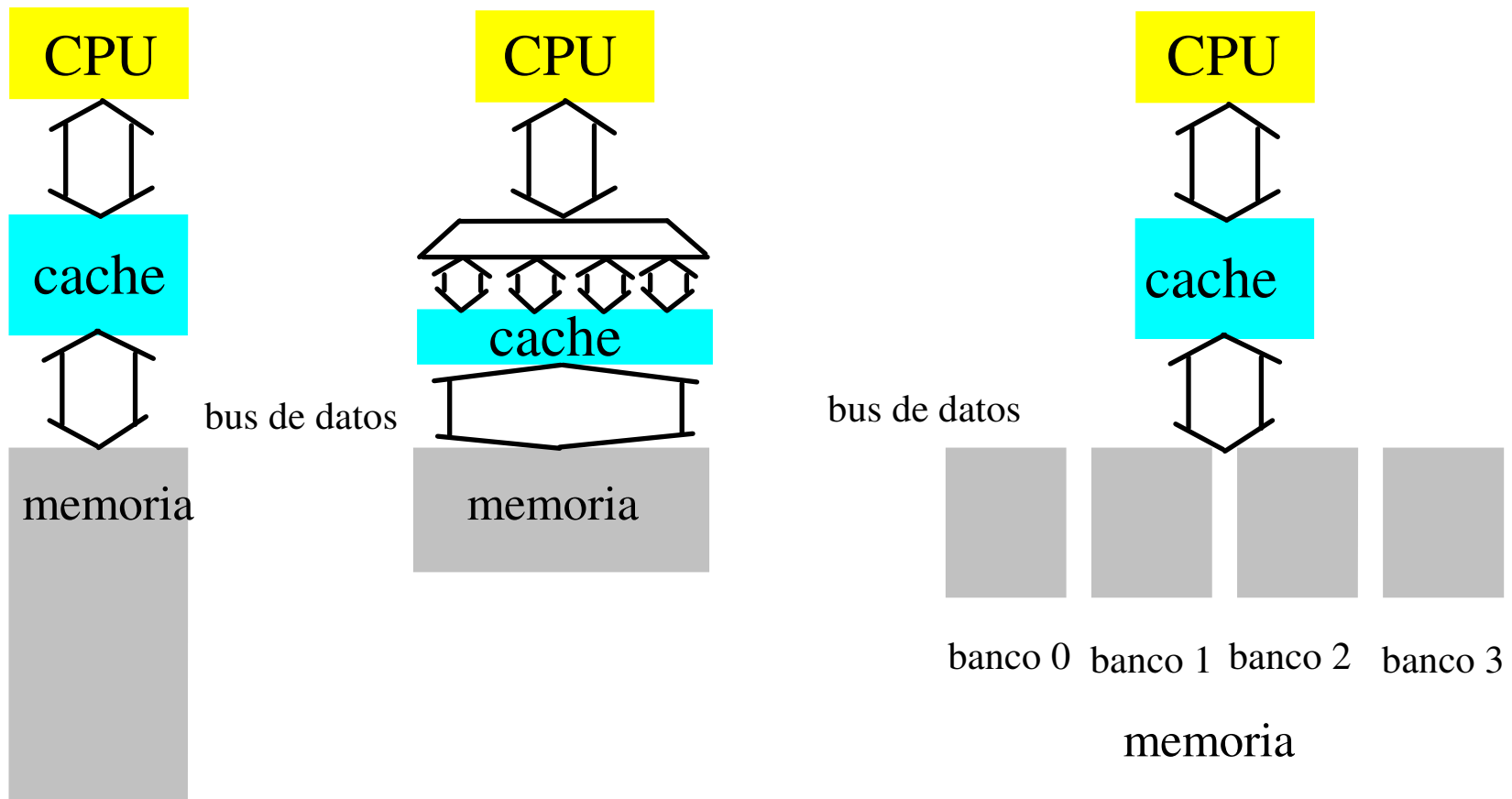
Secuencia de Acceso

- 1 - dirección de fila
- 2 - activación de RAS#
- 3 - retardo RAS/CAS
- 4 - dirección de columna
- 5 - activación CAS#
- 6 - lectura o escritura (R/W#)



Organizaciones de las memorias

DRAM



Ejemplo

1 ciclo de reloj para enviar la dirección.

6 ciclos de reloj para el tiempo de acceso por palabra.

1 ciclo de reloj para enviar una palabra de datos.

Dado un bloque de 4 palabras (de 2 byte c/u \Rightarrow 64 bits)

Memoria convencional
Datos de 16 bits

Caso I

PF= 32 ciclos

AB = 64/32

= 2 bits/c

Memoria mas ancha
Datos de 32 bits

Caso II

PF= 16 ciclos

AB = 64/16

= 4 bits/c

Memoria entrelazada
16 bits cada una

Caso III

PF= 11 ciclos

AB = 64/11

aprox. 6 bits/c

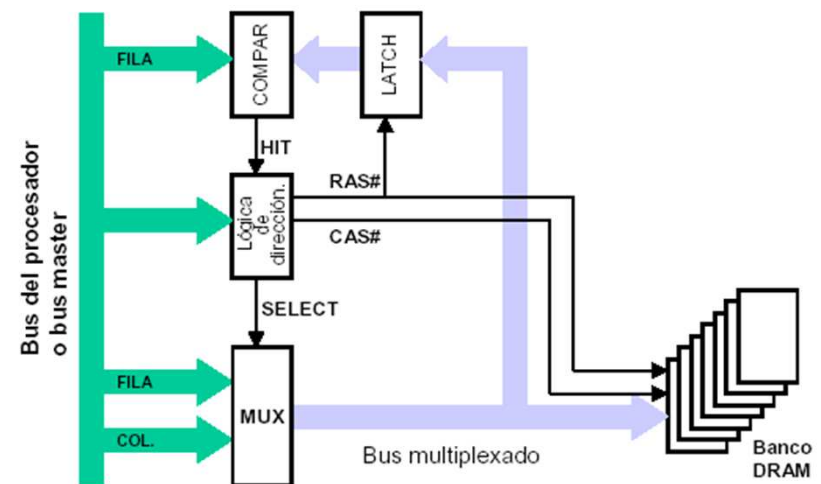
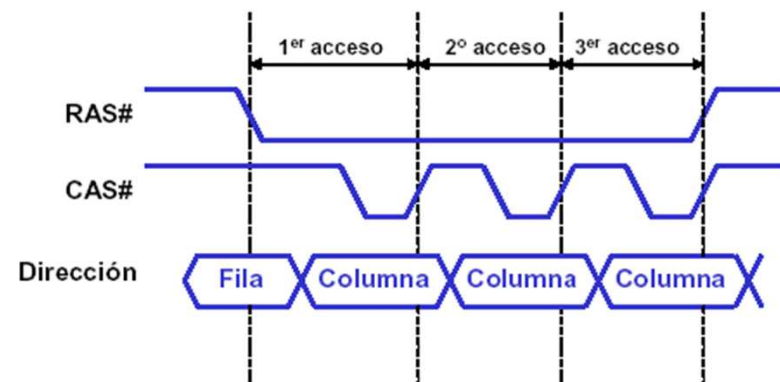
Modos de las DRAM

6 modos principales

- *Modo de página (Page Mode y Fast Page Mode)*
- *Modo EDO Page (Enhanced Data Out Page Mode)*
- *Modo nibble*
- *Columna estática*
- *Modo Síncrono (SDRAM)*
- *Modo DDR (Double Data Rate)*

Modo Página

- Permiten acceder **más rápidamente** posiciones de memoria contenidas en la misma fila.
- Normalmente se desactivan **RAS#** y **CAS#** al final.
- El acceso al primer elemento es normal.
- A continuación se mantiene **RAS#** activa y se direcciona **sólo la columna mediante CAS#**
- Necesitan hardware especial.



Modo EDO (Enhanced Data Out)

- *Enhanced Data Out* o *Hyperpage* consiste en no desactivar los drivers de salida cuando #CAS está a nivel alto.
- Se reduce el tiempo que #CAS debe permanecer a nivel bajo y aumenta el tiempo que los datos permanecen disponibles en el bus
- La actividad de los drivers de salida se controla con #OE

Diagrama de señales en modo Pagina

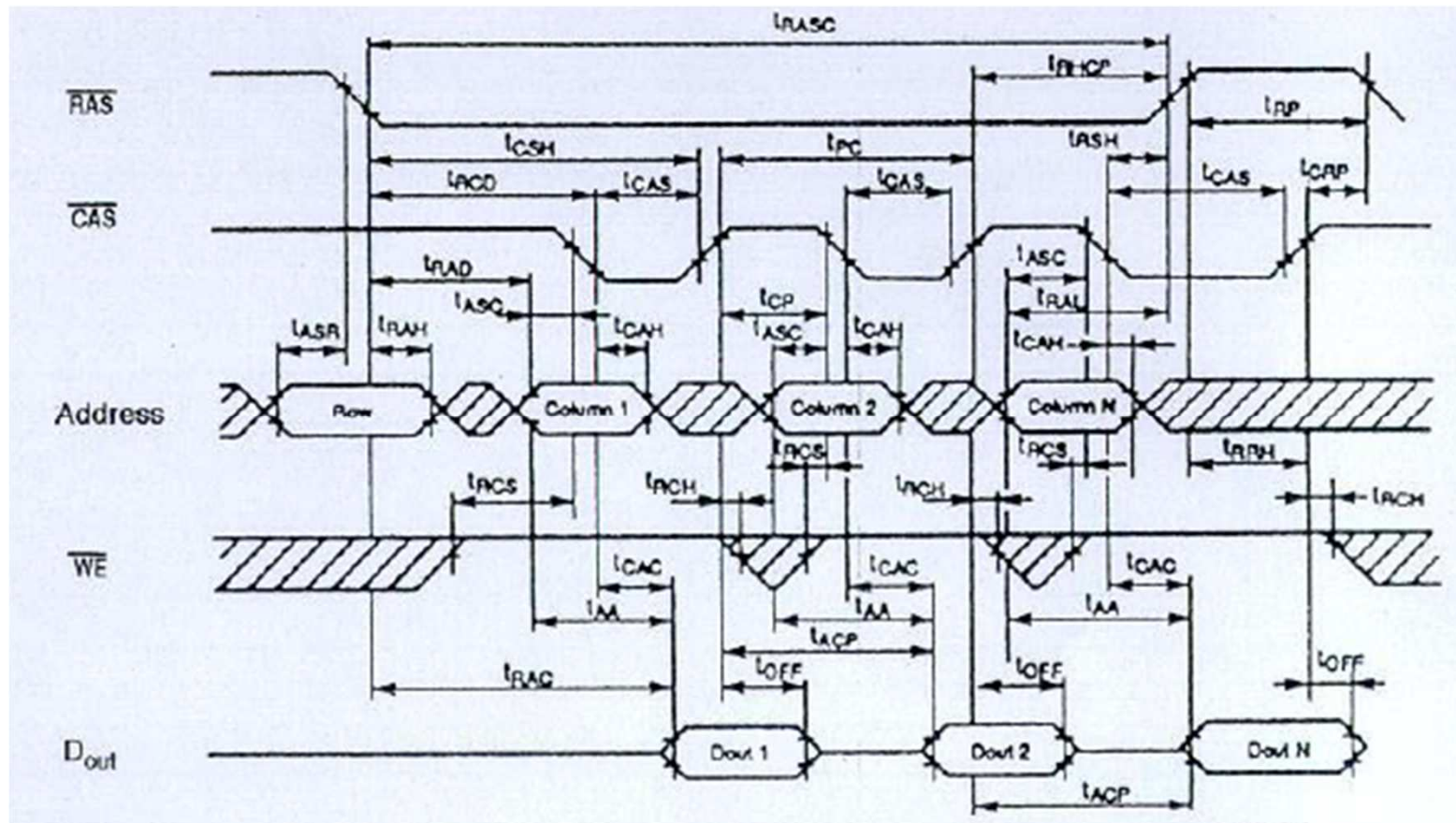
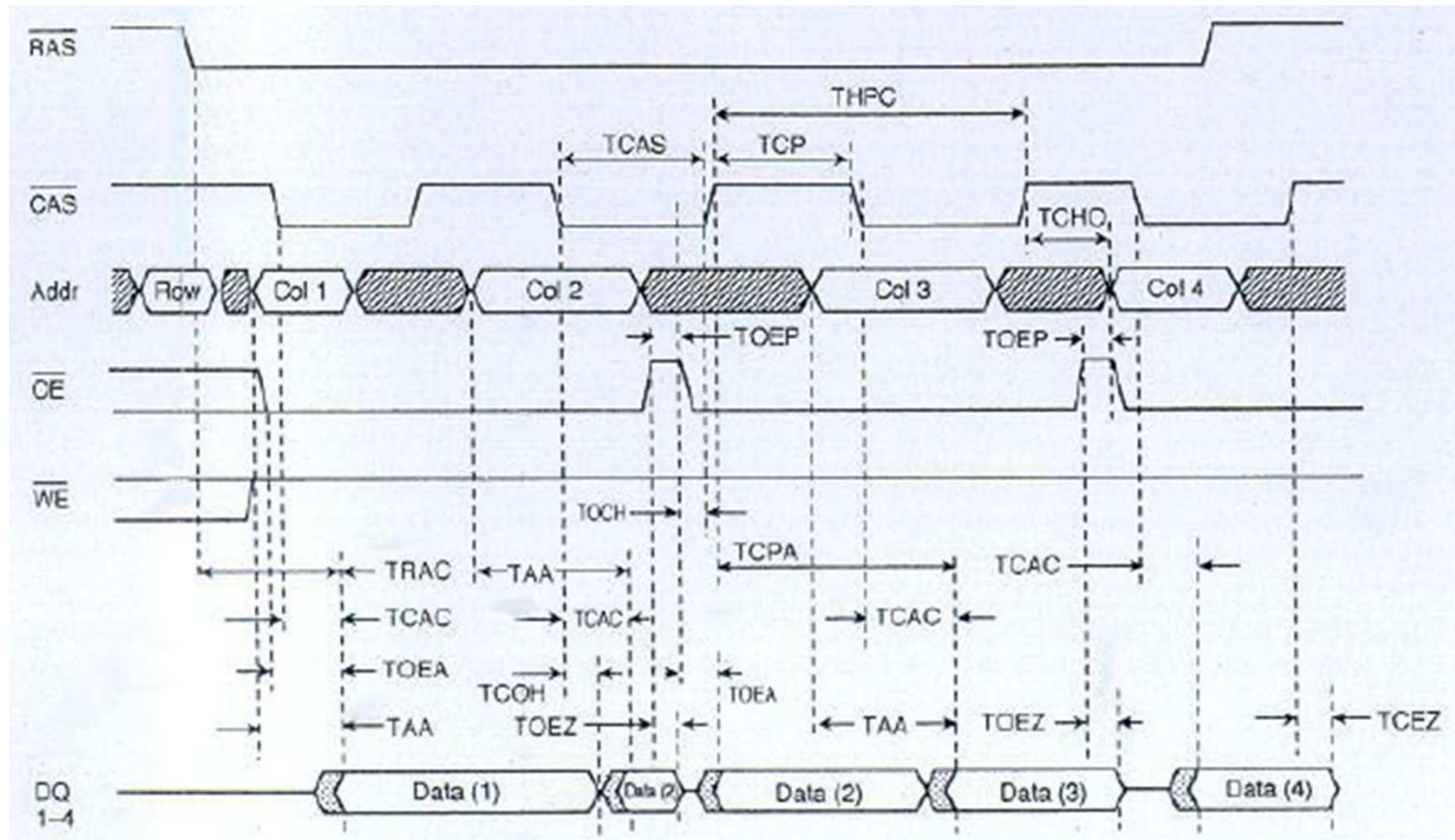


Diagrama de señales en modo EDO



Modo Burst and Nibble

- Idénticas en funcionamiento a las **modo de página**
- Contador de columnas interno
- Incremento automático con la señal **CAS#**
- **Ante una petición de lectura, proporciona el dato solicitado y los tres siguientes.**

El tiempo de acceso de los tres últimos datos es mucho más corto que el del primero.

En escritura hay que proporcionar cuatro datos para ser almacenados en posiciones consecutivas.

Modo Columna Estática

- Idénticas en funcionamiento a las modo de página
- No hace falta suministrar la señal CAS#
- Interpreta los cambios del bus como una nueva columna

Modo Síncrono (SDRAM)

- Todas las memorias anteriores son **asíncronas**
- **CAS# y RAS#** se activan con independencia del reloj de la CPU.
- En estas memorias, **RAS# y CAS#** solamente se suministran en sincronía con el **reloj del sistema**.
- **Simplifica el diseño** del subsistema de memoria
- Los sistemas actuales son todos **síncronos**

Modo DDR (Double Data Rate)

- Son memorias **síncronas**
- Permiten un acceso en **flanco de subida** y otro en **flanco de bajada** de la señal del reloj
- La velocidad de transferencia es el doble de una **SDRAM**
- Funcionan internamente con **16 bits (32 bits)** aunque tienen bus de datos de **8 bits (16 bits)**

Memorias comerciales

- **FPMDRAM (Fast Page Mode DRAM)**: DRAM que permite acceso en modo página.
- **EDO DRAM (Extended Data Output DRAM)**: Se empieza a leer el siguiente bloque de memoria sin haber finalizado el anterior.(40-66 MHz)
- **BEDO DRAM (Burst EDO DRAM)**: Puede procesar hasta 4 posiciones de memoria en una ráfaga, pero sólo de forma puntual.(66-75 MHz)
- **SDRAM (Synchronous DRAM)**: Se sincroniza con el bus del sistema para leer/escribir varias posiciones de memoria en una ráfaga, de manera continua. (100 MHz)
- **DDR SDRAM (Double Data Rate SDRAM)**: Transfiere dos datos por ciclo de reloj. Dobra las prestaciones de la SDRAM. Buses AGP.
- **DR DRAM (Direct Rambus DRAM)**: Tecnología distinta de SDRAM. Bus especial de alta velocidad (1.6 Gb/sg).
- **SL DRAM (Synchronous Link DRAM)**: Evolución de las SDRAM. Teóricamente, hasta 3.2Gb/sg.
- **RAM de tarjetas de vídeo: (RDRAM, VRAM, WRAM, SGRAM, MDRAM)**: Múltiples puertos de acceso, puertos de lectura y escritura diferenciados y simultáneos,...
- **RAM alimentada por baterías**: Solución intermedia entre RAM y ROM. RAM “normal” alimentada por una pila botón de forma continua. Ej: BIOS PCs.

Memoria virtual

Implementación de la jerarquía de memoria entre el segundo y tercer nivel de memoria

- Esquema similar al del nivel de cache
- Estrategia de manejo orientada al SO mas que al rendimiento

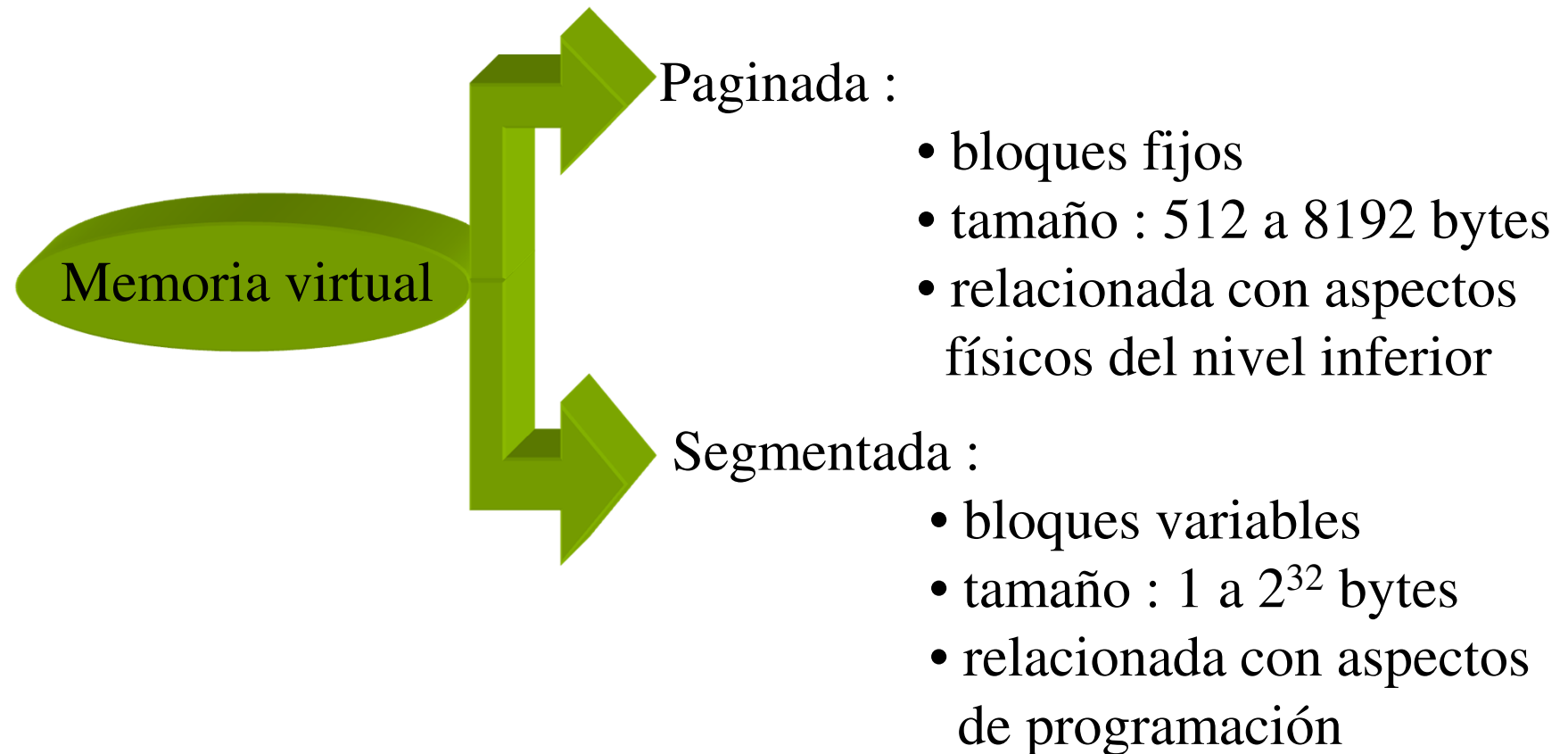
Terminología:

- Segmento o página se refieren a un bloque
- Fallo de segmento o fallo de página se refieren a fallo

Otras diferencias

- El reemplazo de los fallos de cache está controlado por hardware mientras que el reemplazo en memoria virtual se controla por el sistema operativo.
- El tamaño de la dirección del procesador determina el tamaño de la memoria virtual, pero el tamaño de la cache es normalmente independiente de la dirección del procesador.
- El nivel más bajo de la memoria virtual (discos) se comparte con el sistema de archivos del sistema operativo, con lo cual el tamaño o capacidad de la memoria virtual no es siempre la misma.

Tipos de memoria virtual



Ubicación de bloques en memoria principal

Penalización de fallos de memoria virtual involucra el acceso a un dispositivo generalmente lento (discos magnéticos)

Alternativas:

- Reducir la frecuencia de fallos
- Un algoritmo de ubicación de bloques mas sencillo

elección mas adecuada:

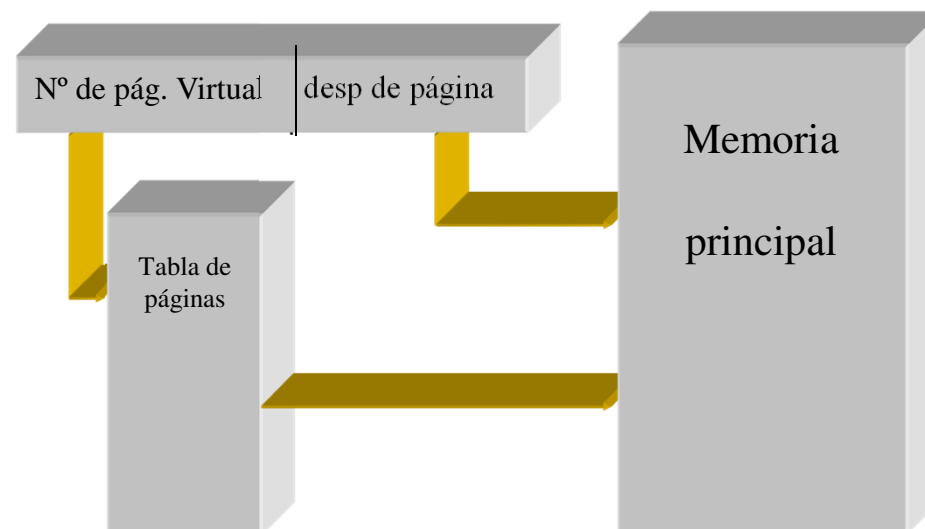
- Disminuir la frecuencia de fallos debido al alto costo de un fallo

Por tanto los sistemas operativos permiten que un bloque se coloque en cualquier parte de la memoria virtual

Identificación de un bloque

la memoria virtual implementa una estructura de datos indexada por el numero de pagina o segmento

La estructura de datos tiene la forma de una tabla de paginas, donde sus entradas se seleccionan mediante una dirección virtual y cada una de las entradas posee datos diversos entre los que se cuentan: dirección física, limite (para segmentación solamente), atributos propios de la pagina/segmento, protecciones, etc.



Estrategias de reemplazo

Meta principal :

minimizar los fallos de página por su coste elevado en tiempo

Algoritmo de sustitución :

elección del bloque menos recientemente usado (LRU)

(es el que menos, probablemente, se necesitará)

Algoritmo LRU rápido

- 1) Para N marcos de página se usa una matriz de $N \times N$ bits inicialmente en cero
- 2) Para una referencia al marco k se activan los bits de la fila k y se desactivan los bits de la columna k
- 3) En todo instante, la fila cuyo valor es mínimo es la de uso menos reciente y la candidata a reemplazo en un fallo

Ejemplo de algoritmo LRU

Suponer el siguiente orden de acceso a marcos: 0 1 2 3 2 1 0 3

	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Candidato: 1, 2, 3

	0	1	2	3
0	0	0	1	1
1	1	0	1	1
2	0	0	0	0
3	0	0	0	0

Candidato: 2, 3

	0	1	2	3
0	0	0	0	1
1	1	0	0	1
2	1	1	0	1
3	0	0	0	0

Candidato: 3

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

Candidato: 0

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	1
3	1	1	0	0

	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	1	0	0	1
3	1	0	0	0

	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

	0	1	2	3
0	0	1	1	0
1	0	0	1	0
2	0	0	0	0
3	1	1	1	0

Estrategia de escritura

Una escritura al nivel mas bajo de memoria virtual (el disco)
es bastante costosa en tiempo

El sistema operativo trata de evitar accesos a disco hasta que
el bloque sea reemplazado haciendo todas las escrituras en
memoria principal

☑ Estrategia de postescritura

Inconveniente :

Falta de coherencia entre RAM y discos

Rendimiento del sistema global

Hasta ahora:

$$T_{\text{CPU}} = IC * \left(\text{CPI}_{\text{ejecución}} + \frac{\text{accesos-memoria}}{\text{instrucción}} * FF * PF \right) * T_{\text{ciclo}}$$

Al incorporar el comportamiento de la memoria virtual queda:

Penalización de fallos de cache =

$$\begin{aligned} (1) \quad &= FF_{\text{TLB}} * \text{tiempo_acceso_RAM} * \text{Longitud_entrada_tabla_páginas} + \\ (2) \quad &+ FA_{\text{RAM}} * \text{tiempo_acceso_RAM} + \\ (3) \quad &+ (1 - FA_{\text{RAM}}) * \text{Tiempo_acceso_disco} * \text{Tamaño_página} \end{aligned}$$

FF_{TLB} : Frecuencia de fallos de la TLB (Traslation Lookaside Buffer).
 tiempo_acceso_RAM : Tiempo de acceso a una palabra de la RAM.
 FA_{RAM} : Frecuencia de aciertos a la RAM

Problema de ejemplo

Una CPU (A) ejecuta un programa de 250000 instrucciones en 10,5 segundos. El promedio de ciclos de reloj por instrucción es de 2,5 ciclos para aquellas instrucciones que no acceden a memoria y de 5,5 ciclos en promedio para las que acceden a operandos en memoria -que en el programa de testeo ascienden a 30% del total de instrucciones-.

En estas condiciones se agrega al sistema un subsistema de administración de memoria con una caché de mapeo directo que impone una penalización de fallos de 12 ciclos y está organizada de manera que el 35% de los accesos son fallos. (CPU B)

Bajo estas condiciones de trabajo calcular:

1. Frecuencia de funcionamiento del sistema de CPU.
2. Tiempo en segundos que la CPU con caché tarda en ejecutar el programa.
3. Calcule la mejora de rendimiento de la nueva configuración respecto de la antigua
4. Analice y explique a que se debe el bajo rendimiento encontrado.

NOTA: Para algunos de los incisos será necesario asumir ciertas condiciones de operación

Problema: item 1

Frecuencia de funcionamiento del sistema de CPU.

$$T_{cpu\ A} = RI * CPI_{sc} * T_c$$

$$T_{cpu} = 10,5 \text{ seg.}$$

$$RI = 250.000$$

$$CPI_{sc} = 5,5 \text{ ciclos}$$

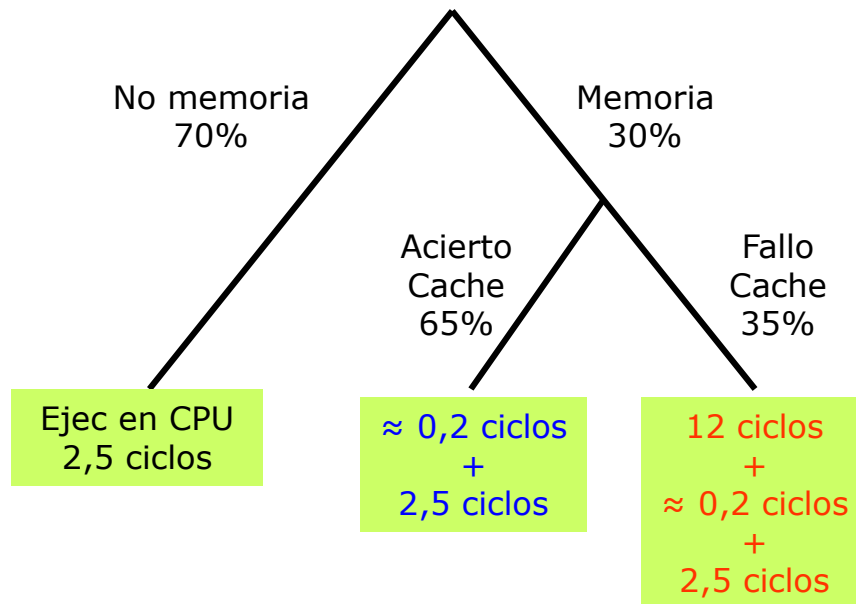
$$T_c = ??$$

$$t_c = \frac{T_{cpuA}}{RI * CPI_{sc}} = \frac{10,5}{250000 * (0,7 * 2,5 + 0,3 * 5,5)} = 12,35 \mu s$$

Problema: item 2

Tiempo en seg. que la CPU con caché tarda en ejecutar el prog.

Genéricamente : $T_{cpu} = RI * (CPI_{ejecución} + CPI_{acc\ memoria}) * T_c$



$$\begin{aligned} CPI_{mem} &= 0,7 * 2,5 + \\ &\quad 0,3 * 0,65 * 2,7 + \\ &\quad 0,3 * 0,35 * 14,7 = \\ &= 3,82 \end{aligned}$$

Entonces

$$T_{cpuB} = 11,79$$

Problema: item 3 - 4

Calcule la mejora de rendimiento de la nueva configuración respecto de la antigua

$$A_{global} = \frac{T_{CPU_A}}{T_{CPU_B}} = \frac{10,5}{11,79} = 0,8905$$



Analice y explique a que se debe el bajo rendimiento encontrado.

???