Jerarquía de memoria

Tiempo medio de acceso

Katia Leal Algara, URJC
Juan González Gómez, URJC

Objetivos

- Necesidad de una jerarquía de memoria
- Veremos:
  - Mecanismo de acceso de memoria en este tipo de jerarquía
  - Diseño y evaluación del rendimiento de la jerarquía de memoria
  - Aspectos tecnológicos y de diseño de los distintos niveles de la jerarquía de memoria

Diseño de una jerarquía de memoria

- Actualmente se utiliza una jerarquía de memoria organizada en niveles que incluye:
  - **Memoria caché, MC**
  - **Memoria principal, MP**
  - **Memoria virtual, MV**
- Cada una:
  - se ubica físicamente en un lugar distinto
  - se fabrica con una tecnología diferente
  - se gestiona de manera independiente

Diseño de una jerarquía de memoria: MC

- Ubicada en el mismo chip que el procesador
- Fabricada con memoria RAM estática (**SRAM**), *Static Random Access Memory*
  - Tipo de memoria basada en semiconductores es capaz de mantener los datos sin necesidad de circuito de refresco
  - Son memorias volátiles, pierden la información si se les interrumpe la alimentación eléctrica
- Controlada por el **controlador de caché** incluido en el mismo chip
- Las jerarquía actuales suelen tener hasta 3 niveles de MC: L1, L2 y L3

Diseño de una jerarquía de memoria: MP

- Ubicada en un chip diferente al procesador
- Fabricada con memoria RAM dinámica (**DRAM**), *Dynamic Random Access Memory*
  - Para mantener almacenado un dato, hay que revisar el mismo y recargarlo cada cierto tiempo en un ciclo de refresco
  - Ventaja: posibilidad de construir memorias con una gran densidad de posiciones y que funcionen a una velocidad alta
- Controlada por el **controlador de memoria principal** que se encarga de planificar los accesos a la misma
- Hoy en día el controlador puede ubicarse en el mismo chip que el procesador y la memoria caché o en otro chip como el chipset norte o el MCH

Diseño de una jerarquía de memoria: MP

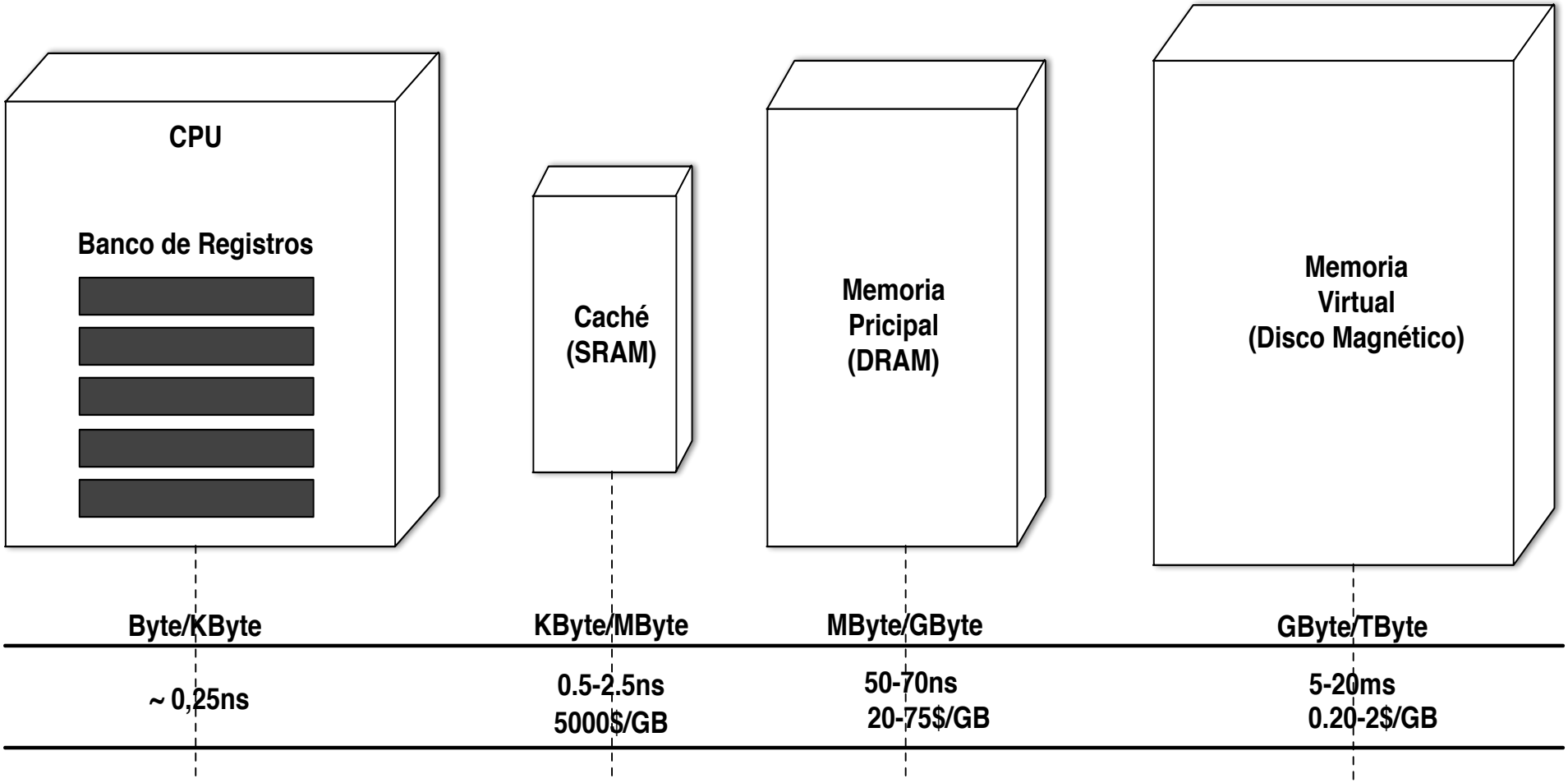
- DRAM

Year	Capacity	$/GB
1980	64Kbit	$1500000
1983	256Kbit	$500000
1985	1Mbit	$200000
1989	4Mbit	$50000
1992	16Mbit	$15000
1996	64Mbit	$10000
1998	128Mbit	$4000
2000	256Mbit	$1000
2004	512Mbit	$250
2007	1Gbit	$50

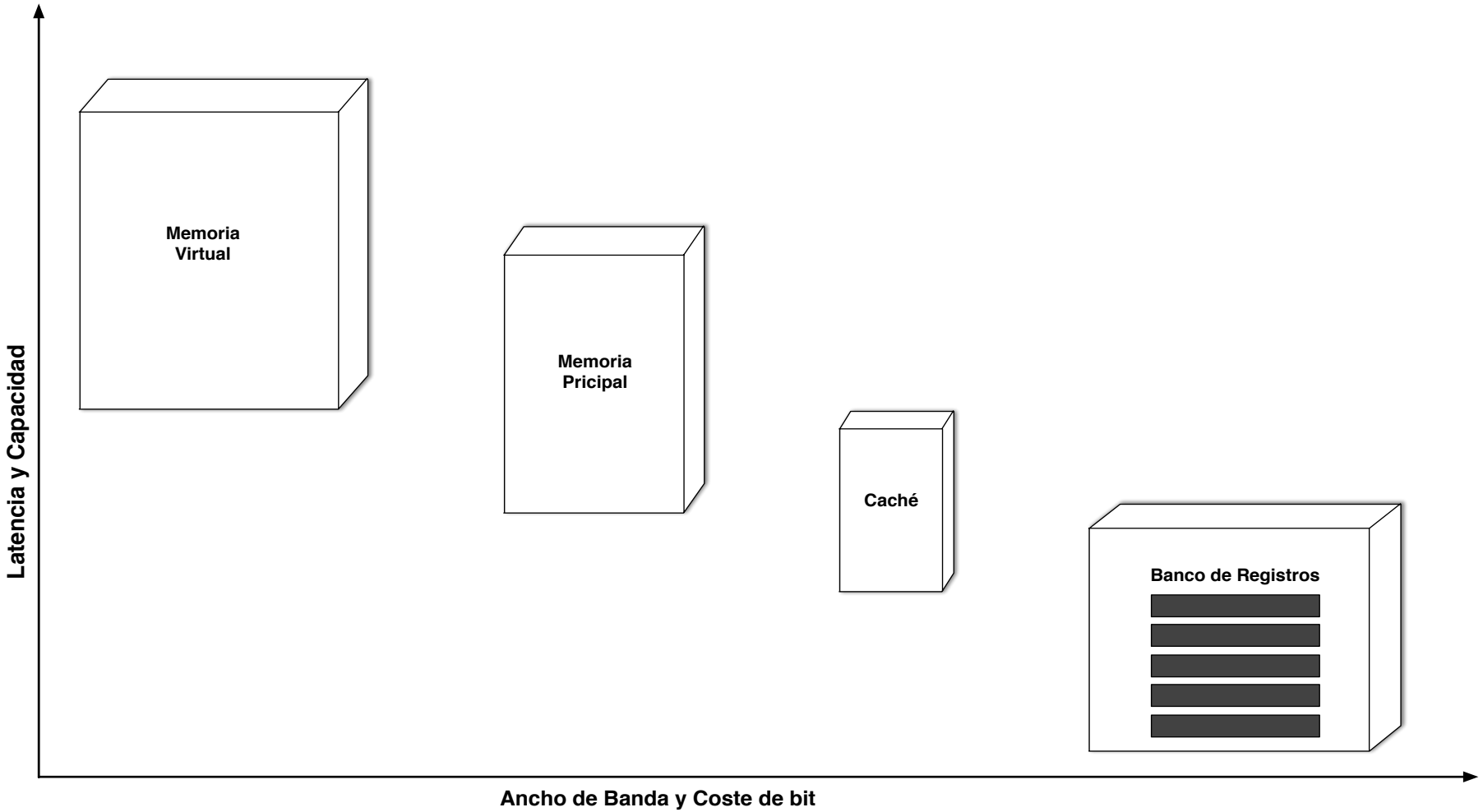
Diseño de una jerarquía de memoria: MV

- Generalmente, ubicada en el disco duro
- Fabricada con tecnología magnética
- Se controla desde el **sistema operativo** a través del controlador de disco duro

Comparativa distintos niveles



Comparativa distintos niveles



- **Latencia**: tiempo que transcurre desde que un acceso a memoria comienza hasta que finaliza
- **Ancho de banda**: cantidad de información por unidad de tiempo que puede transferirse desde/hacia la memoria

Propiedades jerarquía de memoria

- **Inclusión**: cualquier información contenida en un nivel de la jerarquía debe estar también en el resto de niveles
- **Coherencia**: las copias de la misma información en los diferentes niveles son coherentes entre sí, es decir, que almacenan los mismos valores
- Debe haber una **correspondencia de direcciones** entre los distintos niveles de la jerarquía

Principio de localidad

- **Localidad espacial**
  - Si se referencia un elemento, los elementos cercanos a él también tenderán a ser referenciados
  - La jerarquía de memoria mueve bloques con palabras contiguas en memoria a los niveles más altos de la jerarquía
  - Ejemplo: operaciones con matrices y arrays, ejecución secuencial de un programa
- **Localidad temporal**
  - Si se referencia un elemento, este tenderá a ser referenciado pronto
  - La jerarquía de memoria mantiene los datos accedidos recientemente lo más cerca posible del procesador
  - Estructura de los programas: datos y bucles

Aciertos y fallos en el acceso a la MC

- La palabra a leer o escribir se busca en MC
  - Si la palabra se encuentra en caché, **acierto**
  - Si no se encuentra en caché, **fallo**. La penalización por fallo dependerá de la latencia de la MP y de su ancho de banda
- Si la palabra no se encuentra en la MC, se trae un **bloque** que contiene dicha **palabra** desde la MP
- ¿Qué pasaría si siempre se produjeran fallos en la caché?
  - **Principio de localidad**

Aciertos y fallos en el acceso a la MP

- Relación entre MP y MV similar a la existente entre MC y MP
- La MP se divide en **páginas** o segmentos
- Cuando se produce un fallo de página o segmento, se debe acceder a la MV
- **La penalización en este caso es mayor** ya que la MV es la más lenta de la jerarquía
- En la gestión de este nivel **interviene el SO**. El procesador realiza un *cambio de contexto* y ejecuta otra tarea hasta que la página o segmento esté disponible en MP

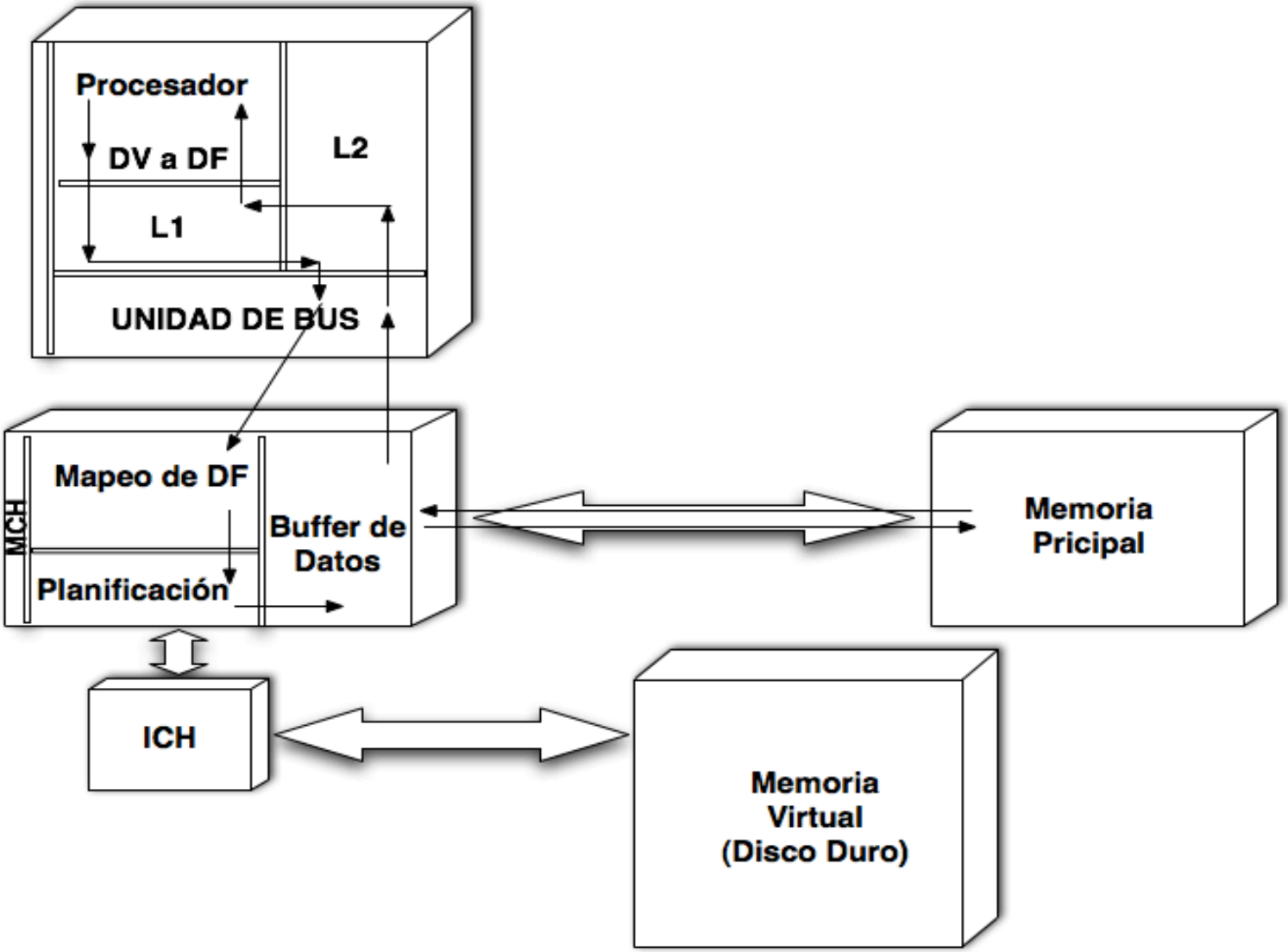
Mecanismo completo de acceso a memoria: con un solo nivel de MC

1. Se traduce la **dirección virtual** a **dirección física:** MMU
2. Si se traduce con éxito, es porque la palabra se encuentra hasta la MP
   - Con esta dirección se accede a la memoria caché
3. Tipos de fallos en la MC:
   - **Iniciales**, cuando se referencia una palabra por primera vez
   - **De capacidad**, cuando se producen reemplazos
   - **De conflicto**, varios bloques tienen asignada la misma ubicación en MC
4. En caso de fallo en la MC **hay que pasar por el controlador de MP**, que mapeará la dirección física buscada a la ubicación física de la palabra dentro de los chips DRAM
5. El controlador planificará el acceso a la MP puesto que otros dispositivos también acceden a la MP
   - El bloque que incluye la palabra se envía a la MC
6. Si no se traduce con éxito, se debe resolver el fallo de página
   - El SO realiza un cambio de contexto y pasa a ejecutar otro proceso
   - Una vez que la página está en MP, se lleva el bloque correspondiente a MC y se reanuda la ejecución de la instrucción que provocó el fallo

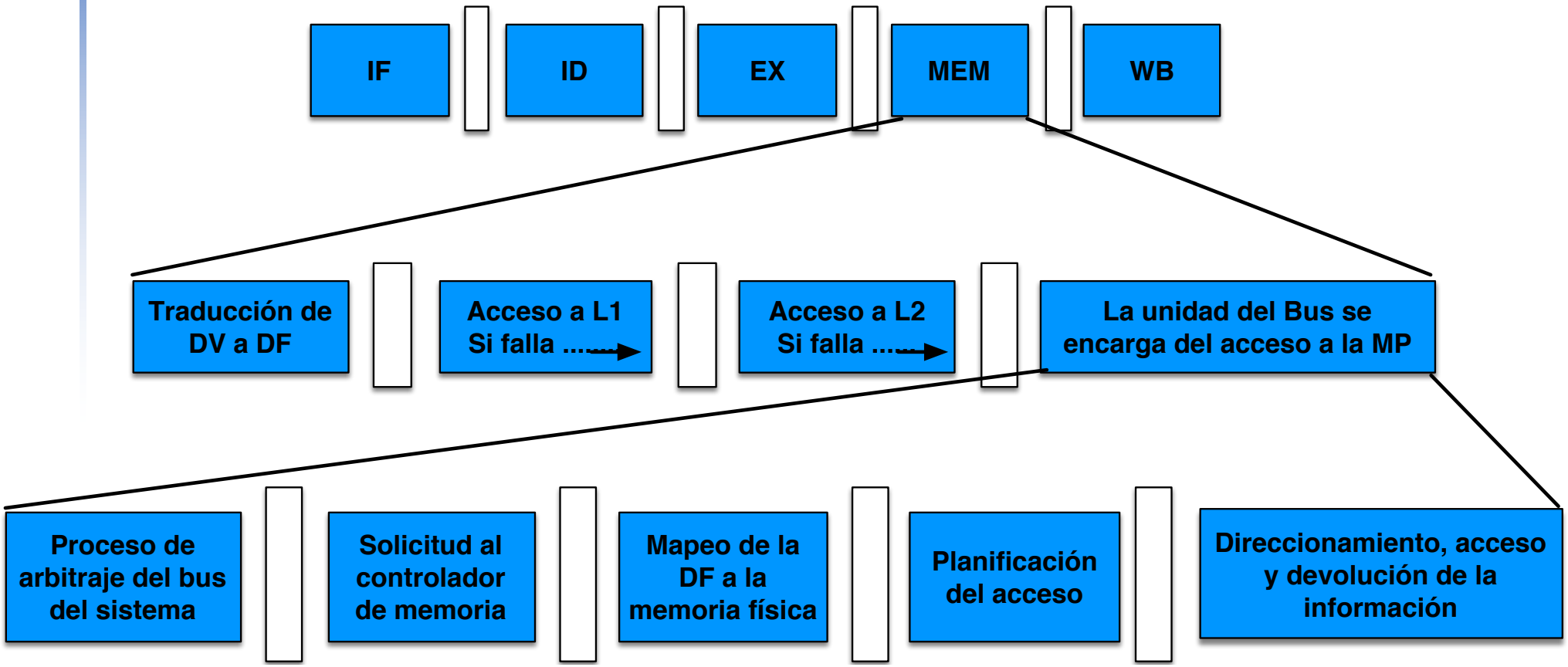
Mecanismo completo de acceso a memoria: con un dos niveles de MC

- El mecanismo de acceso es el mismo, salvo que cuando falla el acceso a L1, se intenta resolver desde L2
- Si la palabra está en L2, se envía el bloque completo a L1, completando el acceso sin salir del chip del procesador
- En caso de fallo, se debe intentar desde MP
- Siguiendo el principio de inclusión, el bloque que provocó el fallo se lleva desde la MP hasta L2, y desde L2 hasta L1
- En el caso de un procesador segmentado, todo este proceso (hasta llegar a la MP) debería completarse en la etapa de acceso a memoria, es decir, en un único ciclo de reloj

Mecanismo completo de acceso a memoria: con un dos niveles de MC



Acceso a memoria del nanoRISC-V en la etapa MEM



Evaluación prestaciones jerarquía de memoria

- Incluir los accesos a memoria completos en la etapa MEM aumenta mucho el valor de periodo de reloj
- Normalmente, en la etapa MEM sólo se tienen en cuenta los tiempo de acceso a la MC
- En caso de fallo, el tiempo extra se suma al tiempo que tarda en ejecutarse la instrucción
  - La técnica de **caché no bloqueante** permite que las instrucciones avancen por la ruta de datos mientras se resuelve el fallo
  - El tiempo extra se solapa con la ejecución de otras instrucciones

Evaluación prestaciones jerarquía de memoria

- **Tiempo medio de acceso a la jerarquía de memoria**

$$t_{MEM} = t_{aciertoMC} + TF \bullet pF$$

- $t_{aciertoMC}$: Tiempo de acierto de la MC
- ***TF***: Tasa de Fallos de la MC

  TF = núm de fallos / núm total accesos a memoria

- ***pF***: Penalización por fallo en MC
- Normalmente, el tiempo invertido en acceder a la memoria se suma al tiempo de CPU

$$t = t_{CPU} + t_{MEM}$$

Ejercicio 1

- Un procesador que funciona a 2 GHz ejecuta 100 instrucciones. El procesador tiene 2 MC, una de instrucciones (MI) y otra de datos (MD) ideales, es decir, tienen un tiempo de acceso despreciable y nunca fallan. Si el CPI es ideal, es decir, 1, ¿cuánto tiempo tardan en ejecutarse las 100 instrucciones?

$$t = t_{CPU} = I \bullet CPI \bullet T = 100 \bullet 1 \bullet 1/2 \bullet 10^9 = 50\ ns$$

Ejercicio 2

- Si ahora tenemos en cuenta que las cachés sí que fallan, es decir, que hay que acceder a la MP, ¿cuánto tiempo tardarán en ejecutarse las 100 instrucciones?
- La MI tiene una tasa de fallos del 4% y una penalización por fallo de 100 ns. La MD tiene una talla se fallos del 6% y una penalización por fallo de 115 ns
- Para ejecutar las 100 instrucciones hacen falta 100 accesos a la memoria de instrucciones y 25 a la de datos

$$t_{MEM} = nº \text{ de acceso a MI}(t_{aciertoMI} + TF_{MI} \bullet pF_{MI}) + nº \text{ de acceso a MD}(t_{aciertoMD} + TF_{MD} \bullet pF_{MD})$$
$$= 100(0 + 0.04 \bullet 100) + 25(0 + 0.06 \bullet 115) = 572{,}25 \text{ ns}$$

- Tiempo total:

$$t = t_{CPU} + t_{MEM} = 50 \text{ ns} + 572{,}5 \text{ ns} = 622{,}5 \text{ ns}$$

Diseño de la memoria caché

- Almacena bloques de información denominados ***marcos de bloque***
- Para determinar qué bloque está ocupando un determinado marco, se utilizan **etiquetas**
- Las etiquetas se comparan con la del bloque buscado para indicar si se ha producido un acierto o un fallo
- Aspectos básicos de su diseño
  - Organización de la memoria caché
  - Política de ubicación
  - Política de reemplazo
  - Política de escritura

Organización de la memoria caché

- Tamaño de la memoria caché
- Tamaño de marco
- Unificación o división de las instrucciones y los datos
- Implementación de cachés multinivel

Organización de la memoria caché: - *Tamaño total*

- **Demasiado pequeña**
  - Incrementa la tasa de fallos
  - No captura bien la localidad
  - Fallos de capacidad
- **Demasiado grande**
  - No se podrá integrar en el mismo chip que el procesador debido al consumo de área y de potencias
  - Más lenta, más comparaciones de etiquetas

Organización de la memoria caché: - *Tamaño del marco de bloque*

- **Bloques grandes**
  - Se captura mejor la localidad espacial
  - Se reducen los fallos iniciales
  - …
  - Aumenta la penalización por fallo, se necesita más tiempo para traer los bloques del siguiente nivel
- **Se debe llegar a un compromiso** teniendo en cuenta la latencia y el ancho de banda de conexión con el siguiente de la jerarquía

Organización de la memoria caché: - *Unificación o división*

- **La segmentación del procesador obliga a la división** para evitar riesgos estructurales entra las etapas F y M
- Cuando no se trata del primer nivel se puede optar por unificar en bloques de información comunes las instrucciones y los datos

- Todas estas decisiones se toman con ayuda de herramientas de simulación

Organización de la memoria caché: - *Multinivel*

- Es el aspecto que más influye en el rendimiento
- Memoria caché de un único nivel
  - ¿pequeña y rápida o grande y lenta?
  - **Solución**: utilizar más de un nivel de memoria caché, normalmente **dos niveles** (muchas arquitecturas ya utilizan tres niveles)
- Nivel 1 (L1)
  - La más cercana al procesador, pequeña y rápida
- Nivel 2 (L2)
  - Mayor tamaño, aprovecha el principio de localidad, más lenta pero menos fallos de capacidad
- La penalización por fallo es menor, puesto que en lugar de ir a MP irá a L2

Ejercicio 3

- Comparar tiempo medio de acceso a una jerarquía de memoria con un solo nivel de MC y con dos niveles de MC
- Jerarquía con un solo nivel de MC:
  - El tiempo de acceso de L1 es de 1 ns
  - La tasa de fallos de L1 es del 5%
  - La penalización por fallo de L1 es de 90 ns
- ¿Tiempo medio de acceso a memoria para realizar una lectura?

$$t_{MEM}\ (lectura) = t_{aciertoL1} + TF_{L1} \bullet pF_{L1} = 1 + 0{,}05 \bullet 90 = 5{,}5\ ns$$

Ejercicio 4

- Jerarquía con dos niveles de MC:
  - El tiempo de acceso de L1 es de 1 ns
  - La tasa de fallos de L1 es del 5%
  - El tiempo de acceso de L2 es de 12 ns
  - La tasa de fallos de L2 es del 10%
  - La penalización por fallo de L2 es de 90 ns
- ¿Tiempo medio de acceso a memoria para realizar una lectura?

$$t_{MEM}\ (lectura) = t_{aciertoL1} + TF_{L1} \bullet pF_{L1} = 1 + 0{,}05 \bullet pF_{L1}$$

$$pF_{L1} = t_{L2} = t_{aciertoL2} + TF_{L2} \bullet pF_{L2} = 12 + 0{,}1 \bullet 90 = 21\ ns$$

$$t_{MEM}\ (lectura) = 1 + 0{,}05 \bullet 21 = 2{,}05\ ns$$

$$Ganancia = 5{,}5/2{,}05 = 2{,}68$$

Política de ubicación

- En la MC sólo se pueden almacenar unos pocos bloques de información
- ¿En qué marco alojamos el bloque?
- Debemos comparar las política de ubicación
  - **Tasa de Fallos Vs Tiempo de Acceso**
- Políticas de ubicación
  - **Correspondencia directa**
  - **Totalmente asociativa**
  - **Asociativa por conjuntos**

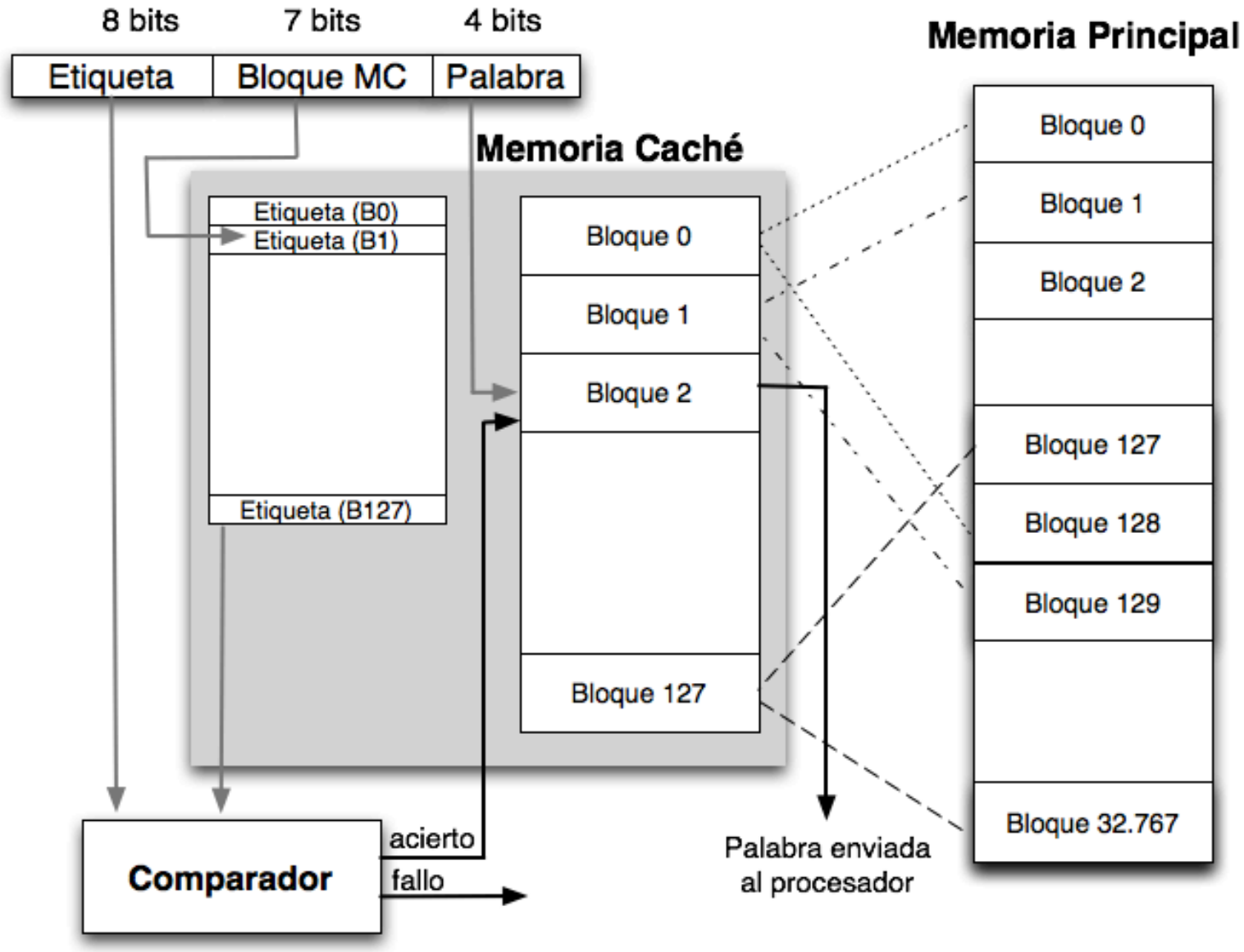
Política de ubicación:
- *Correspondencia directa*

- A cada bloque de memoria principal solo le corresponde un marco de memoria caché
- Correspondencia entre memoria principal y caché
  - Dirección física: 19 bits
  - Tamaño de bloque: 16 bytes (2^4)
  - Capacidad MC: 2KB (2048B/16B = 128 bloques)
  - Capacidad MP: 512KB (512KB/16B = 32.768 bloques)
- **Ejemplo**: para ubicar el bloque 5 de MP en una MC de 128 marcos tendremos que realizar el módulo

Bloque 5 *mod* 128 marcos = marco 5

- Al bloque 5 de MP solo le corresponde el bloque 5 de MC. Es decir, el bloque 5 de MP sólo se puede ubicar en el bloque 5 de MC

Política de ubicación:
- *Correspondencia directa*



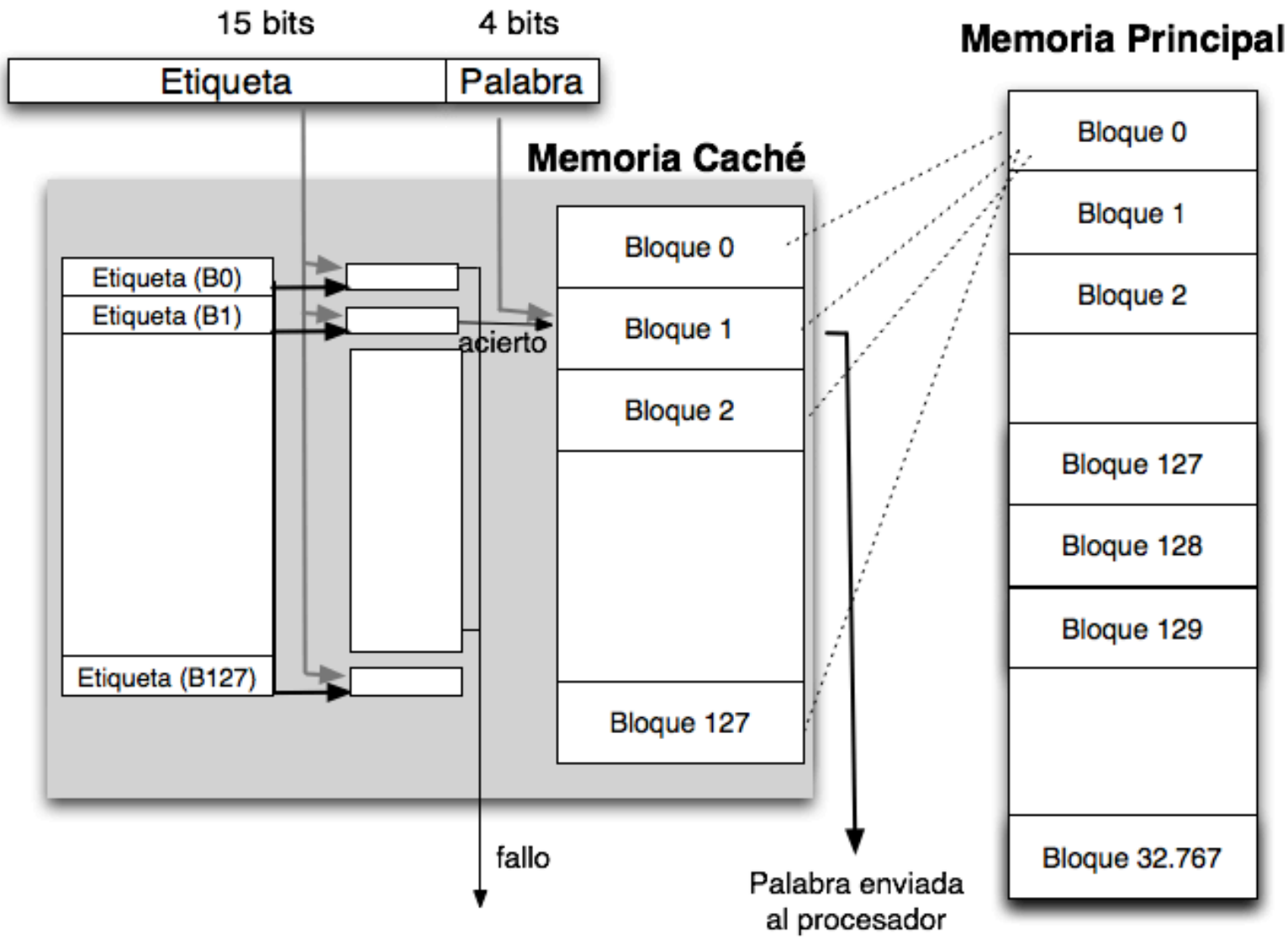
Política de ubicación:
- *Correspondencia directa*

- **Ventajas**
  - La lectura permite el acceso simultáneo al **directorio** y a la palabra dentro del bloque de MC
  - Algoritmo de reemplazo trivial
- **Inconvenientes**
  - Incremento de la tasa de fallos cuando dos bloques de MP correspondientes al mismo bloque de MC son accedidos de forma alternativa

Política de ubicación:
- *Totalmente asociativa*

- Cualquier bloque de MP se puede ubicar en cualquier bloque de MC
- La etiqueta se compara con todas las etiquetas almacenadas en caché
- **Ventajas**
  - Flexibilidad, permite implantar gran variedad de algoritmos de reemplazo
  - Presenta la mayor tasa de aciertos
- **Inconvenientes**
  - Mayor tiempo de acceso
- Al bloque 5 de MP le corresponde cualquier bloque de MC. Es decir, el bloque 5 de MP se ubicará en el bloque de MC que indique el algoritmo implementado

Política de ubicación:
- *Totalmente asociativa*

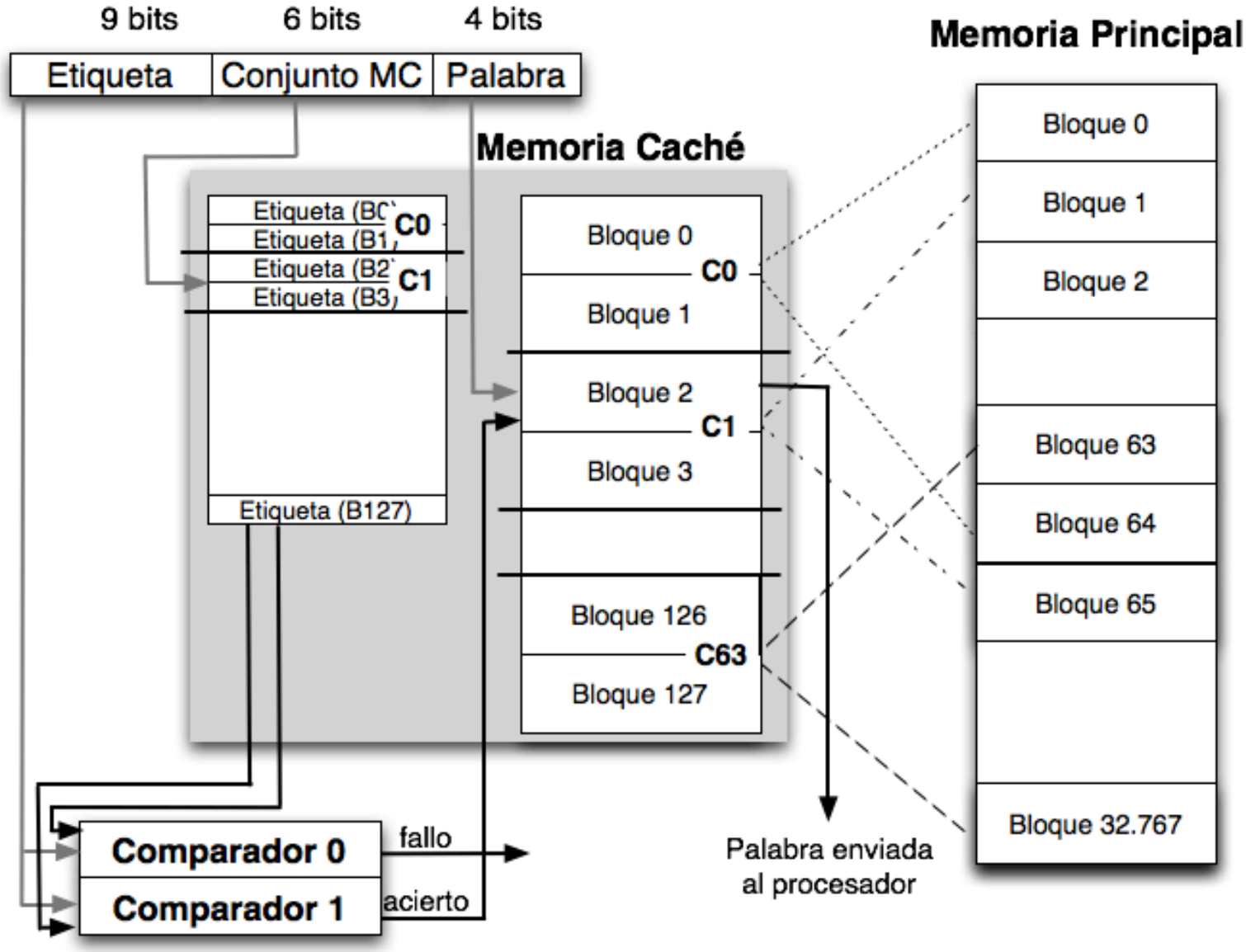


Política de ubicación:
- *Asociativa por conjuntos*

- Dividir la MC en *C* **conjuntos** de *B* **bloques** o *vías*
- Se aplica:
  - Correspondencia directa a nivel de conjuntos
  - Correspondencia asociativa a nivel de bloques
- **Ventajas**
  - Reduce el tiempo de acceso de la memoria totalmente asociativa
  - Incrementa la tasa de aciertos de la directa
- **Ejemplo**: para ubicar el bloque 5 de MP en una MC asociativa de 64 conjuntos de 2 vías, tendremos que realizar el módulo

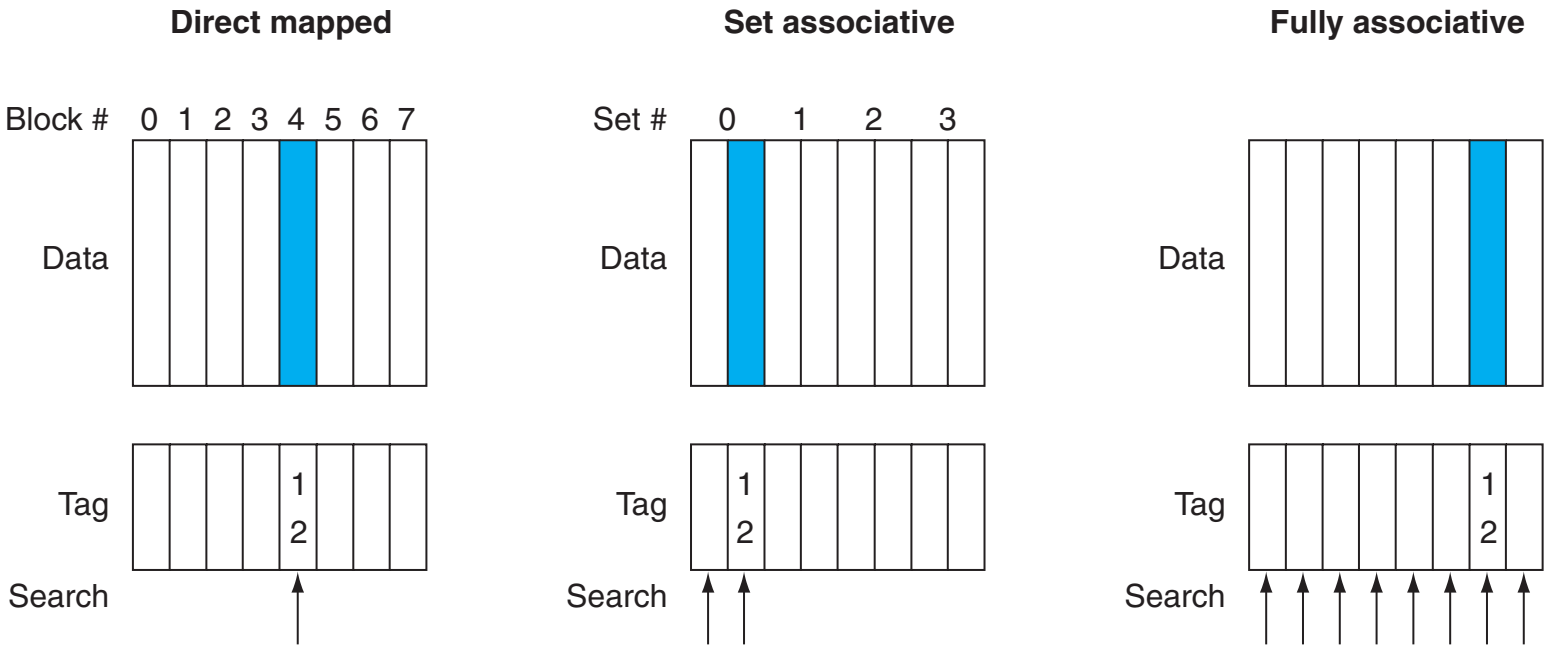
  Bloque 5 *mod* 64 conjuntos = conjunto 5
- Al bloque 5 de MP sólo le corresponde el conjunto 64 de MC. Dentro del conjunto 64 se puede ubicar en cualquiera de los dos bloques siguiendo la política totalmente asociativa
- Ajustar los valores de C y B mediante el uso de simuladores para determinar el diseño de caché que más nos conviene

Política de ubicación:
- *Asociativa por conjuntos*



Ejemplo

- Ubicación del bloque 12 de MP con las tres políticas de ubicación



Políticas de reemplazo

- Cuando se produce un fallo en MC hay que determinar qué bloque de MC desalojar para traer el bloque que ha provocado el fallo desde MP
- Con *correspondencia directa* solo se puede desalojar un bloque, ¿cuál?
- La selección del bloque puede reducir la tasa de fallos de los accesos posteriores a MC
- A tener en cuenta
  - Probabilidad de uso de una página
  - Coste del intercambio (modificada?)
- Algoritmos más utilizados con política *totalmente asociativa*
  - **Aleatorio**
  - **FIFO**
  - **LRU**

Políticas de reemplazo

- **Aleatorio:** se utiliza un generador de números aleatorios para escoger el bloque a reemplazar
  - Más sencilla de implementar y la menos costosa
- **FIFO** - *First In First Out*, reempleza el bloque que ha permanecido en MC el mayor periodo de tiempo
  - Más costoso a medida que aumenta al número de bloques, se debe mantener una lista ordenada
- **LRU** - *Least Recently Used*, reemplaza el bloque de memoria que lleva más tiempo sin utilizarse
  - Más complejo y costoso en cuanto a recursos, pero es el qué mejores resultados ofrece
- Tasa de aciertos Vs Tamaño caché (número de bloques)

Políticas de escritura

- Las escrituras llevan más tiempo puesto que no se puede realizar trabajo en paralelo
  - En el caso de las lecturas, se recupera la información de un marco determinado al tiempo que se realiza la comparación de las etiquetas
- Políticas
  - **Escritura directa**
  - **Post-escritura**

Políticas de escritura:
- *Escritura directa, Write through*

- Se escribe a la vez en el primer nivel de memoria caché y en el siguiente nivel de la jerarquía
- En caso de fallo, se trae el bloque de MP a MC y una vez en esta, se procede a realizar la escritura
- **Ventajas**
  - Fácil de implementar
  - Asegura la coherencia
- **Inconvenientes**
  - Se genera mucho tráfico con MP
  - El procesador tiene que esperar (diferencia tiempos de escritura en MC y MP) a que se complete la escritura
  - Solución: **buffer de escritura**

Políticas de escritura:
- *Post-escritura, Write back*

- Cuando se modifica una palabra solo se hace en el primer nivel de caché
- Para no actualizar la memoria con cada reemplazo, se activa un bit, el **bit de sucio** o *dirty bit*, que indicará si el bloque ha sido modificado o no
- La escritura del bloque en MP se realiza cuando el bloque sucio en MC se reemplaza
- **Ventajas**
  - Menos tráfico de información
  - Los aciertos en escritura se llevan a cabo a la velocidad de la MC
- **Inconvenientes**
  - Diseño más complejo, requiere más recursos
  - En caso de fallo y de que el bloque a reemplazar esté modificado, hay que volcarlo entero en el siguiente nivel
- Teniendo en cuenta posibles fallos transitorios de la MC, es preferible la escritura directa en L1

Políticas de escritura:
- *Buffer de escritura*

- **Estructura hardware** en la que se realizan las escrituras en primera instancia
- Posteriormente se solapa la escritura con el siguiente nivel de la jerarquía con la ejecución de las siguientes instrucciones
- Estructura pequeña y rápida organizada como una memoria caché
- Las escrituras en el buffer suponen una penalización menor que hacerlas en el siguiente nivel de la jerarquía de memoria
- Si el buffer está lleno, el procesador debe parar hasta que quede una posición vacía

Políticas de escritura:
- *Buffer de escritura con escritura directa*

- Las escrituras se hacen palabra a palabra en caché y en este buffer en lugar de en el siguiente nivel de la jerarquía de memoria
- El contenido del buffer se volcará en el siguiente nivel:
  - Cuando esté **lleno**
  - Cuando se produzca un **fallo de lectura** y sea necesario actualizar el siguiente nivel con el contenido del buffer antes de resolver el fallo

Políticas de escritura:
- *Buffer de escritura con post-escritura*

- El buffer se utiliza para volcar los bloques sucios que van a ser reemplazados en memoria caché
- No hay que esperar a que se escriba el bloque sucio en el siguiente nivel, puesto que se almacena temporalmente en el buffer
- El contenido del buffer se volcará en el siguiente nivel:
  - Cuando esté lleno
  - Cuando se produzca un fallo **de lectura** y sea necesario actualizar el siguiente nivel con el contenido del buffer antes de resolver el fallo

Políticas de escritura

- Si el acceso a MC es para escritura y el bloque no se encuentra, se produce un ***fallo de escritura***
- **Escritura con ubicación** *(Write with allocate)*
  - Se suele asociar con *post-escritura*
  - Se lleva el bloque de MP a MC donde se realiza la escritura
- **Escritura sin ubicación** *(Write with no allocate)*
  - Se suele asociar con *escritura directa*
  - Sólo se escribe sobre la MP, el bloque que contiene la palabra que ha provocado el fallo no se lleva a MC

Ejercicio 5

- Calcular tiempo medio de acceso a una jerarquía de memoria si se usa **escritura directa con ubicación**
- Único nivel de MC unificada
- El tiempo medio de acceso a MC es de 4ns
- Tasa de fallos MC: 12%
- Tamaño de bloque: 16 palabras
- La latencia de acceso a MP es de 85ns
- 70% lecturas, 30% escrituras

Ejercicio 5

Tipo de acceso	Acciones	Tráfico con MP
Acierto de lectura	1. Leer 1 palabra en MC	0 palabras
Fallo de lectura	1. Traer bloque fallo de MP a MC 2. Leer 1 palabra en MC	16 palabras
Acierto de escritura	1. Escribir 1 palabra en MC y en MP	1 palabra
Fallo de escritura	1. Traer bloque fallo de MP a MC 2. Escribir 1 palabra en MC y en MP	16 + 1 palabras

Ejercicio 5

t_{MEM} **(lectura)** = $t_{acierto}$ + $TF \bullet pF$

t_{MEM} **(escritura)** = $t_{acierto}$ + $latencia_{MP}$ + $TF \bullet pF$

$pF = 16 \bullet (latencia_{MP})$ = 16 • (85) = **1360 ns**

t_{MEM} **(lectura)** = $t_{acierto} + TF \bullet pF$ = 4 + 0,12 •1360 = **167,2 ns**

t_{MEM} **(escritura)** = $t_{acierto} + latencia_{MP} + TF \bullet pF$ =

4 + 85 + 0,12 •1360 = **252,2 ns**

- En media:

t_{MEM} **= %lectura •** t_{MEM} **(lectura) + %escritura •** t_{MEM} **(escritura)**

= *0,7* • 167,2 + 0,3 • 252,2 = **192,7 ns**

Ejercicio 6

- Calcular tiempo medio de acceso a una jerarquía de memoria si se usa **post-escritura con ubicación**
- Único nivel de MC unificada
- El tiempo medio de acceso a MC es de 4ns
- Tasa de fallos MC: 12%
- Tamaño de bloque: 16 palabras
- La latencia de acceso a MP es de 85ns
- 70% lecturas, 30% escrituras
- 10% de bloques modificados/sucios

Ejercicio 6

Tipo de acceso	Acciones	Tráfico con MP
Acierto de lectura	1. Leer 1 palabra en MC	0 palabras
Fallo de lectura + reemplazo de bloque limpio	1. Traer bloque fallo de MP a MC 2. Leer 1 palabra en MC	16 palabras
Fallo de lectura + reemplazo de bloque sucio	1. Volcar bloque sucio en MP 2. Traer bloque fallo de MP a MC 3. Leer 1 palabra en MC	16 + 16 palabras
Acierto de escritura	1. Escribir 1 palabra en MC	0 palabras
Fallo de escritura + reemplazo de bloque limpio	1. Traer bloque fallo de MP a MC 2. Escribir 1 palabra en MC	16 palabras
Fallo de escritura + reemplazo de bloque sucio	1. Volcar bloque sucio en MP 2. Traer bloque fallo de MP a MC 3. Escribir 1 palabra en MC	16 + 16 palabras

Ejercicio 6

- Con post-escritura tenemos el mismo comportamiento para lecturas y escrituras:

$$t_{MEM} = t_{acierto} + TF \bullet pF$$

- Pero la penalización por fallo se modifica respecto de la escritura directa

$$pF = 16 \bullet (latencia_{MP}) + \%sucios \bullet 16 \bullet (latencia_{MP}) =$$

$$16 \bullet 85 + 0{,}1 \bullet 16 \bullet 85 = \mathbf{1496\ ns}$$

$$t_{MEM} = t_{acierto} + TF \bullet pF = 4 + 0{,}12 \bullet 1496 = \mathbf{183{,}52\ ns}$$

Ejercicio 7

- Calcular el tiempo medio de acceso a una jerarquía de memoria con dos niveles de caché unificados
  - *El primer nivel de escritura directa*
  - *El segundo nivel de post-escritura con un 26% de bloques modificados*
- L1:
  - tamaño de bloque de 8 palabras
  - tº de acceso de 1 ns
  - tasa de fallos del 5%
- L2
  - tamaño de bloque de 16 palabras
  - tº de acceso de 9 ns
  - tasa de fallos del 9%
- La transferencia de una palabra entre MP y L2 supone 0,5 ns
- La transferencia de una palabra entre L1 y L2 supone 0,1 ns
- Latencia acceso a MP de 85 ns
- El 70% de los accesos son lecturas y el 30% escrituras

Ejercicio 7

Tipo de acceso	Acciones	Tráfico con L2
Acierto de lectura	1. Leer 1 palabra en L1	0 palabras
Fallo de lectura	1. Traer bloque fallo de L2 a L1 2. Leer 1 palabra en L1	8 palabras
Acierto de escritura	1. Escribir 1 palabra en L1 y en L2	1 palabra
Fallo de escritura	1. Traer bloque fallo de L2 a L1 2. Escribir 1 palabra en L1 y en L2	8 + 1 palabras
Tipo de acceso	**Acciones**	**Tráfico con MP**
Acierto de lectura o escritura	1. Leer o escribir 1 palabra en L2	0 palabras
Fallo de lectura o escritura + reemplazo de bloque limpio	1. Traer bloque fallo de MP a L2 2. Leer o escribir 1 palabra en L2	16 palabras
Fallo de lectura o escritura + reemplazo de bloque sucio	1. Volcar bloque sucio en MP 2. Traer bloque fallo de MP a L2 3. Leer o escribir 1 palabra en L2	16 + 16 palabras

Ejercicio 7

t_{MEM} **(lectura)** $= t_{aciertoL1} + TF_{L1} \bullet pF_{L1}$

t_{MEM} **(escritura)** $= t_{aciertoL1} + t_{aciertoL2} + TF_{L1} \bullet pF_{L1}$

$pF_{L1} = 8 \bullet (t_{L2} + t_{busL1L2})$

$t_{L2} = t_{aciertoL2} + TF_{L2} \bullet pF_{L2}$

$pF_{L2} = 16 \bullet (\text{latencia}_{MP} + t_{busL2MP}) + 16 \bullet \%\text{sucios} \bullet (\text{latencia}_{MP} + t_{busL2MP})$

$pF_{L2} = 16 \bullet (85+0{,}5) + 16 \bullet 0{,}26 \bullet (85+0{,}5) = 1723{,}68$ ns

$t_{L2} = 9 + 0{,}09 \bullet 1723{,}68 = 164{,}13$ ns

$pF_{L1} = 8\ (164{,}13 + 0{,}1) = 1313{,}85$ ns

t_{MEM} (lectura) $= 1 + 0{,}05 \bullet 1313{,}85 =$ **66,69 ns**

t_{MEM} (escritura) $= 1 + 9 + 0{,}05 \bullet 1313{,}85 =$ **75,69 ns**

$\mathbf{t_{MEM}} = \%\text{lectura} \bullet t_{MEM}$ (lectura) $+ \%\text{escritura} \bullet t_{MEM}$ (escritura)

$= 0{,}7 \bullet 66{,}69 + 0{,}3 \bullet 75{,}69 =$ **69,39 ns**

Ejercicio 8

- Calcular el tiempo medio de acceso a una jerarquía de memoria con dos niveles de caché unificados
  - *El primer nivel de escritura directa **con buffer de escritura** (fallo buffer del 8%)*
  - *El segundo nivel de post-escritura **con buffer de escritura** (fallo buffer del 10%) con un 26% de bloques modificados*
- L1:
  - tamaño de bloque de 8 palabras
  - tº de acceso de 1 ns
  - tasa de fallos del 5%
- L2
  - tamaño de bloque de 16 palabras
  - tº de acceso de 9 ns
  - tasa de fallos del 9%
- La transferencia de una palabra entre MP y L2 supone 0,5 ns
- La transferencia de una palabra entre L1 y L2 supone 0,1 ns
- Latencia acceso a MP de 85 ns
- El 70% de los accesos son lecturas y el 30% escrituras

Ejercicio 8: tabla L1-L2

Tipo de acceso	Acciones	Tráfico con L2
Acierto de lectura	1. Leer 1 palabra en L1	0 palabras
Fallo de lectura	1. Traer bloque fallo de L2 a L1 2. Leer 1 palabra en L1	8 palabras
Acierto de escritura + acierto buffer	1. Escribir 1 palabra en L1 y en buffer	0 palabra
Acierto de escritura + fallo buffer	1. Escribir 1 palabra en L1 y en L2	1 palabra
Fallo de escritura + acierto buffer	1. Traer bloque fallo de L2 a L1 2. Escribir 1 palabra en L1 y en buffer	8 palabras
Fallo de escritura + fallo buffer	1. Traer bloque fallo de L2 a L1 2. Escribir 1 palabra en L1 y L2	8 + 1 palabras

Ejercicio 8: tabla L2-MP

Tipo de acceso	Acciones	Tráfico con MP
Acierto de lectura o escritura	1. Leer o escribir 1 palabra en L2	0 palabras
Fallo de lectura o escritura + reemplazo de bloque limpio	1. Traer bloque fallo de MP a L2 2. Leer o escribir 1 palabra en L2	16 palabras
Fallo de lectura o escritura + reemplazo de bloque sucio + acierto buffer	1. Volcar bloque sucio en buffer 2. Traer bloque fallo de MP a L2 3. Leer o escribir 1 palabra en L2	16 palabras
Fallo de lectura o escritura + reemplazo de bloque sucio + fallo buffer	1. Volcar bloque sucio en MP 2. Traer bloque fallo de MP a L2 3. Leer o escribir 1 palabra en L2	16 + 16 palabras

Ejercicio 8

t_{MEM} **(lectura)** = $t_{aciertoL1}$ + $TF_{L1} \bullet pF_{L1}$

t_{MEM} **(escritura)** = $t_{aciertoL1}$ + %fallo_buffer$\bullet t_{aciertoL2}$ + $TF_{L1} \bullet pF_{L1}$

$pF_{L1} = 8 \bullet (t_{L2} + t_{busL1L2})$

$t_{L2} = t_{aciertoL2} + TF_{L2} \bullet pF_{L2}$

pF_{L2} = 16•(latencia$_{MP}$ + $t_{busL2MP}$)+ 16•%sucios•%fallo_buffer•(latencia$_{MP}$ + $t_{busL2MP}$)

pF_{L2} = 16•(85+0,5) + 16•0,26•0,1•(85+0,5) = 1403,57 ns

t_{L2} = 9 + 0,09•1403,57 = 135,32 ns

pF_{L1} = 8•(135,32 + 0,1) = 1083,36 ns

t_{MEM} (lectura) = 1 + 0,05•1083,36 = **55,17 ns**

t_{MEM} (escritura) = 1 + 0,08•9 + 0,05•1083,36 = **55,89 ns**

t_{MEM} = %lectura•t_{MEM} (lectura) + %escritura•t_{MEM} (escritura)

= 0,7•55,17 + 0,3•55,89 = **55,39 ns**

Diseño de la Memoria Virtual

- Permite la **multiprogramación**
  - Por lo que se deben implementar mecanismos de protección
- Permite ejecutar procesos más grandes que la MP
- Permite independencia de las referencias con respecto a la localización de los procesos en MP
- La MV no se controla exclusivamente por hardware
- La tecnología de la MV es el **almacenamiento magnético**
- La unidad de información no es el bloque, sino el **segmento o la página**, mucho mayores que el bloque que se maneja entre MC y MP
- El alojamiento es siempre **asociativo**
- La política de escritura es siempre **post-escritura**

Diseño de la Memoria Virtual

- Más compleja y difícil de gestionar
- Latencia del orden de *ms* y no de *ns*
- Resolver un **fallo de página** implica un **cambio de contexto** para evitar la enorme penalización por fallo
- Decisiones de diseño de un sistema de memoria virtual:
  - El **tamaño de la página** debe ser lo suficientemente grande para amortizar el elevado tiempo de acceso
    - Tamaño típico entre 4KB y 16KB
    - Servidores y desktops soportar páginas de 32KB y 64KB
    - Los sistemas empotrados utilizan páginas de 1KB
  - Priman las **políticas que reducen el número de fallos de pagina**, por eso la técnica de alojamiento empleada es totalmente asociativa
  - **Los fallos de página se manejan por software**, puesto que la sobrecarga introducida por estos algoritmos es mucho menor que el tiempo de acceso a disco
  - La *escritura-directa* queda descartada en MV. En su lugar se emplea ***post-escritura***

Organización de la Memoria Virtual

- **Dirección virtual:** dirección generada por el procesador. Se corresponde con el espacio de direcciones lógicas de un proceso. Puede ser la dirección de una instrucción o de un un dato
- **Dirección física:** dirección que maneja la unidad de memoria. Dirección real de memoria en la que se almacena la instrucción o el dato referenciado
- ¿Cómo podemos sabemos cuál es la dirección física correspondiente a una dirección virtual?
  - **Mecanismo de traducción**

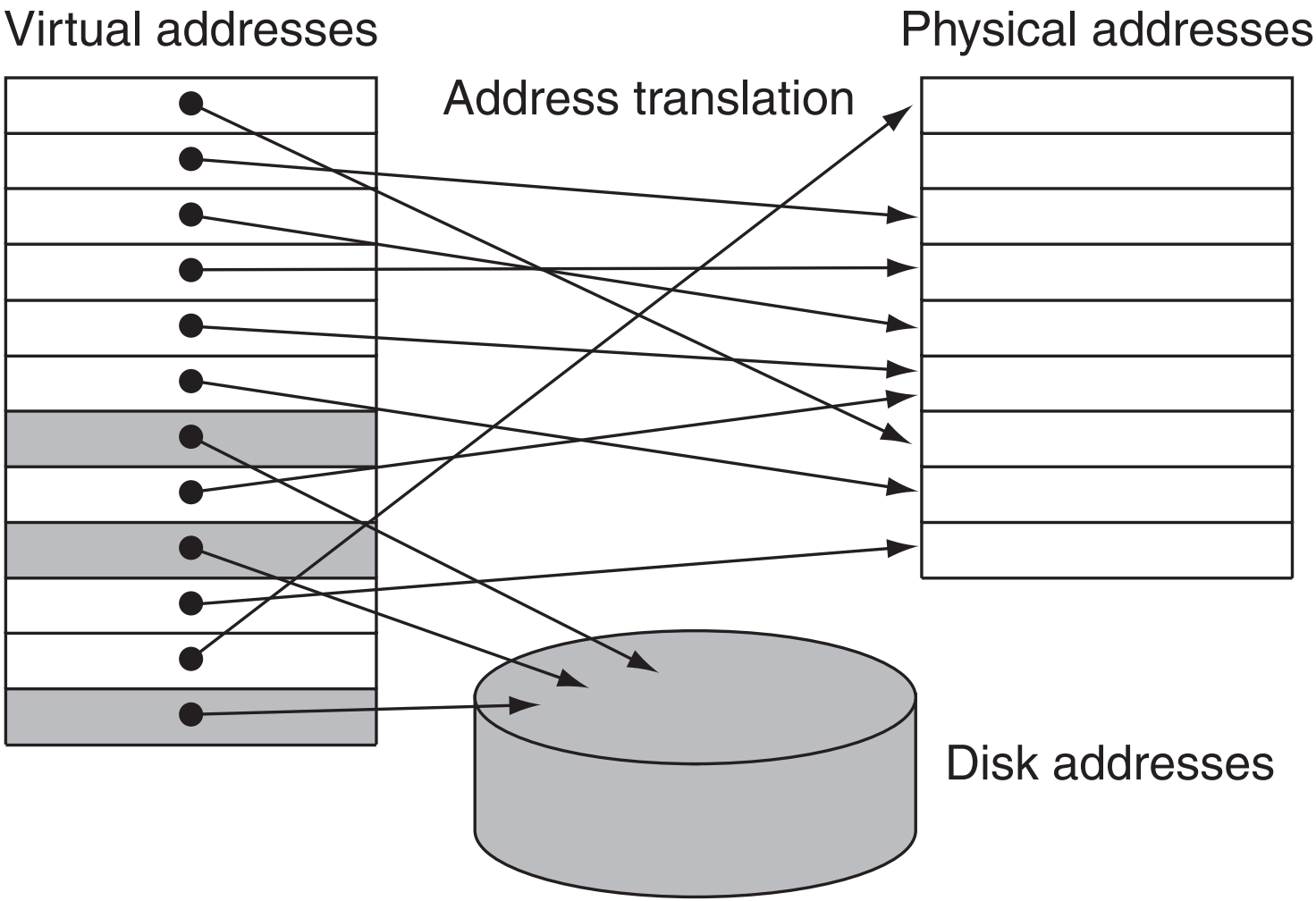
Mecanismo de traducción de direcciones

- *Memory Management Unit (**MMU**)*:
  - Dispositivo hardware por el que pasan todas las referencias a memoria
  - Trabaja en tiempo de ejecución de manera transparente a la CPU
  - Gestiona al mismo tiempo:
    - La traducción de direcciones virtuales a direcciones físicas
    - La protección de la memoria: evita que los programas accedan a porciones de memoria prohibidas
    - El control de la caché

Asignación de memoria

- Tres métodos de asignación de memoria
  - **Paginación**: tamaño fijo de bloque de información
    - Fragmentación interna
  - **Segmentación**: tamaño variable de bloque de información
    - Fragmentación externa
  - **Segmentación paginada**: es una técnica híbrida por la que los segmentos de memoria de un proceso se dividen en un número entero de páginas

Asignación de memoria



Paginación

- La paginación se gestiona además de con el hardware, con la colaboración del SO
- Urgente: *reducir los fallos de página*
  - Los diseñadores se centran en desarrollar algoritmos de ubicación de páginas lo más óptimos posible
- Problema: la ubicación totalmente asociativa radica en localizar una entrada
  - Una búsqueda completa es impracticable!
- En su lugar, se localizan las páginas por medio de unas tablas que indexan la memoria
- A esta estructura se la denomina **tabla de páginas** y reside en memoria

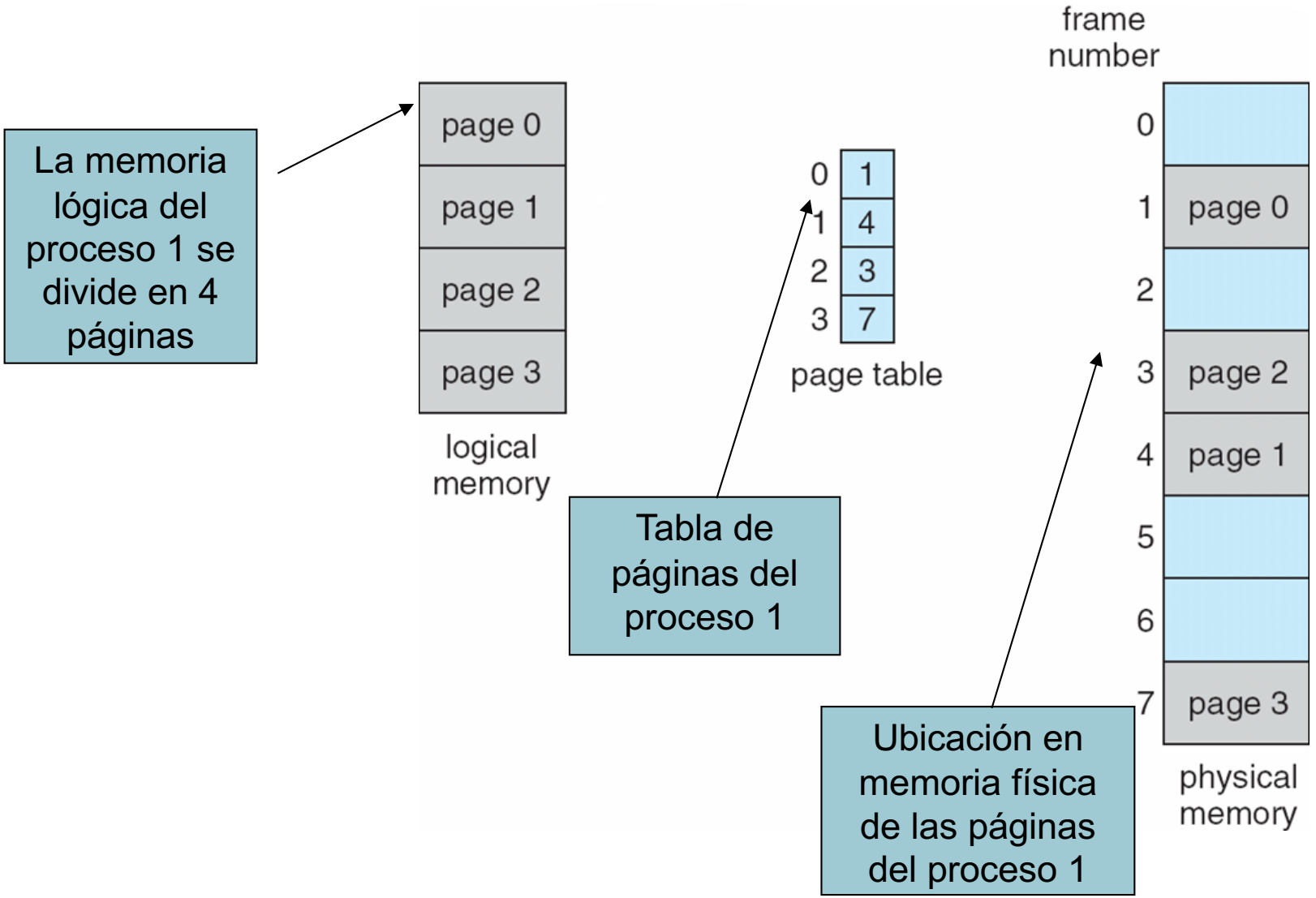
Paginación

- El espacio de direcciones virtuales de un proceso no es contiguo, pudiendo alojarse en cualquier posición de memoria física
- La memoria física se divide en bloques de tamaño fijo denominados **marcos** de página
- La memoria virtual se divide en bloques del mismo tamaño denominados **páginas**
- Se mantiene una lista de todos los marcos libres
- Para ejecutar un programa de *n* páginas, hay que encontrar *n* marcos libres y cargar el programa
- Después, se debe inicializar la **tabla de páginas** para traducir direcciones virtuales a físicas

Ubicación y búsqueda de páginas

- La política de ubicación totalmente asociativa permite al SO implementar los algoritmos de reemplazo más inteligentes para reducir los fallos de página (**LRU**)
  - Encontrar una entrada con una búsqueda exhaustiva es impracticable
- Se usa una **tabla de páginas** que reside en memoria:
  - Indexada con el número de página de la dirección virtual
  - Permite obtener la página física en la que se ubica la página virtual buscada
- Cada proceso del sistema tiene su tabla de páginas
- Para indicar la localización de la tabla de páginas en MP, el hardware incluye un registro que apunta al comienzo de la tabla de páginas, el ***Supervisor Page Table Base Register (*SPTBR*)***

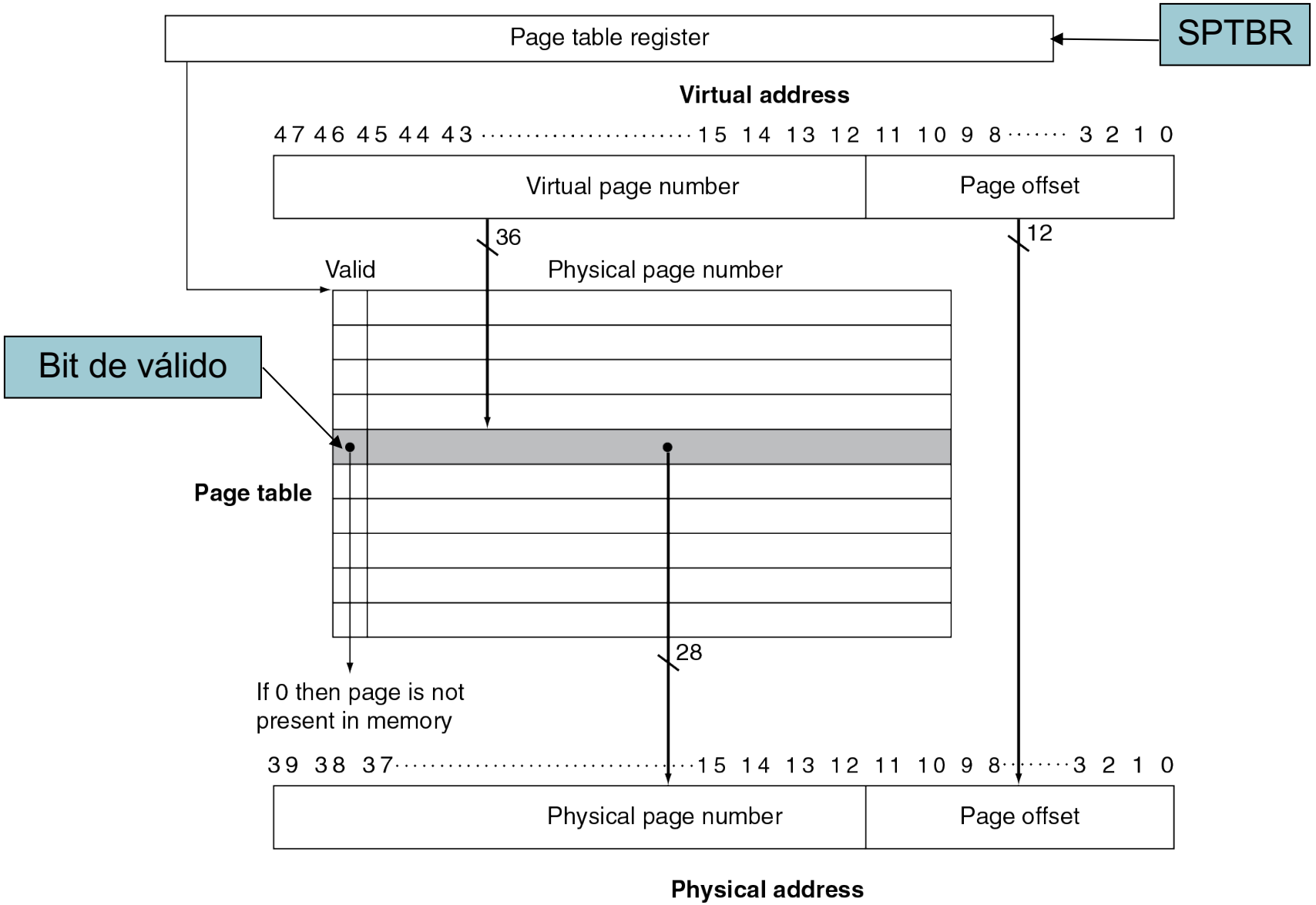
Ubicación y búsqueda de páginas



Ubicación y búsqueda de páginas

- Para evitar colisiones entre el espacio de direcciones virtuales de los distintos procesos, el SO se encarga de:
  - La asignación de la memoria física
  - Actualizar las tablas de páginas. Protección
- Un **bit de válido** se utiliza en cada entrada de la tabla de páginas para indicar si la página está o no presente en la MP:
  - ***Off***: la página no está presente en MP, fallo de página
  - ***On:*** la página está presente en MP, acceso a la MC

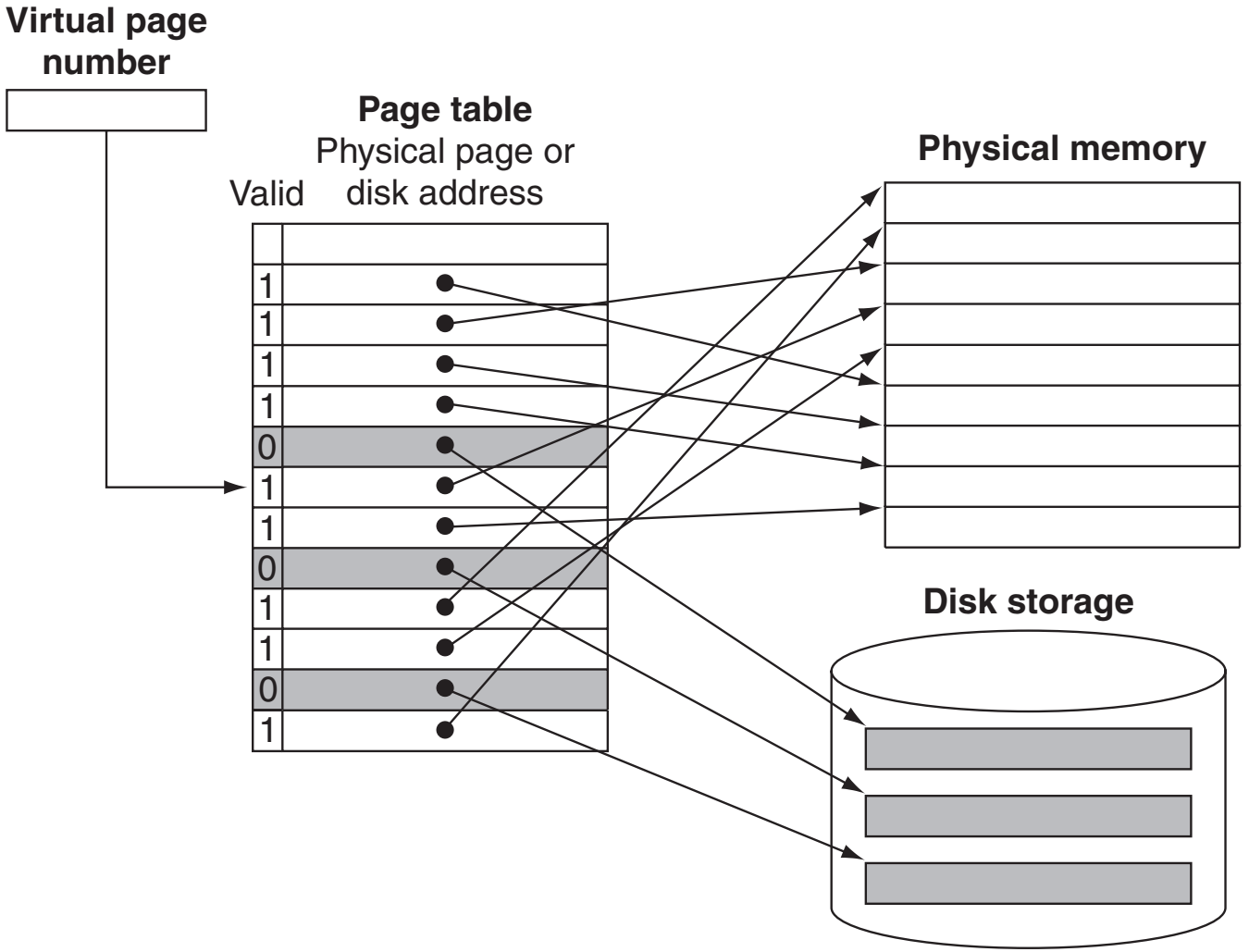
Traducción mediante tabla de páginas



Fallo de página

- Si el bit de válido está a *off* se produce un fallo de página, una excepción
- Se debe transferir el control al SO
  - Mecanismo de excepción
- El SO debe:
  - Buscar la página en el siguiente nivel de la jerarquía
  - Decidir en qué marco de MP alojar la página
    - **Algoritmo de reemplazo**
  - Transferir la página a memoria y actualizar la tabla de páginas
  - Restaurar el proceso y continuar la ejecución por la instrucción que provocó el fallo de página

Fallo de página



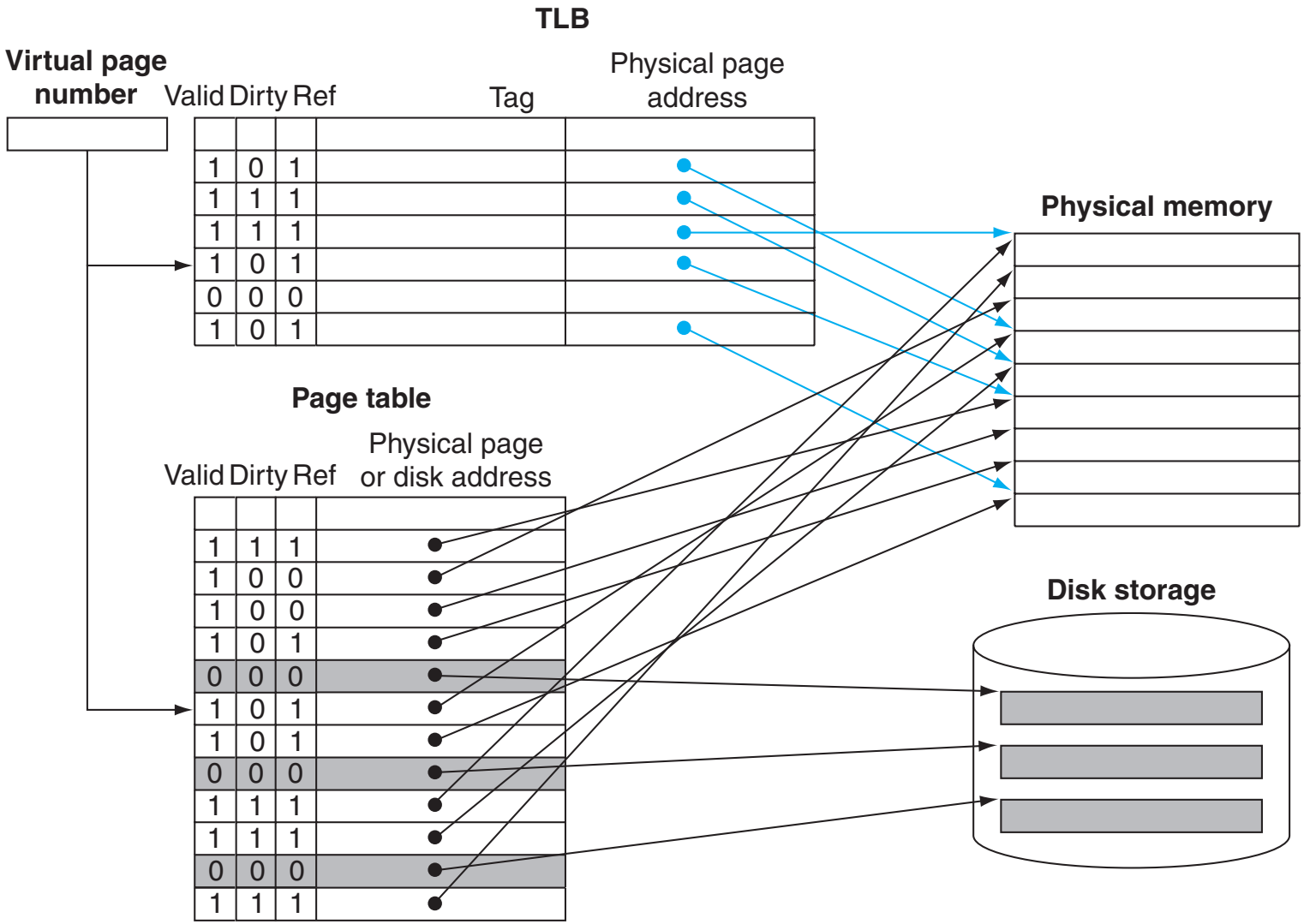
Fallo de página

- La mayoría de los SSOO emplean una aproximación del algoritmo LRU (*Least Recently Used*) para reemplazar una página que lleva cierto tiempo sin ser referenciada
- Para ayudar al SO, algunos ordenadores proporcionan el **bit de referencia** o bit de uso, que se pone a 1 cuando se accede a la página
- Si la página a reemplazar ha sido modificada, en los sistemas de MV se emplea ***write-back***, de tal manera que la página se modifica en cada escritura en MP y solo se copia en el disco duro cuando es sustituida
- Para saber si una página ha sido modificada, a la tabla de páginas se le añade el ***dirty bit***

Traducción más rápida con el TLB

- Las tablas de páginas están almacenadas en MP
  - Cada referencia a memoria por un programa implica, al menos, 2 accesos a memoria
- Debido al principio de localidad, sabemos que las palabras de una página serán referenciadas pronto
  - La traducción de una página virtual a física debería conservarse
- Los procesadores actuales incluyen una caché especial con un número de entradas limitadas para las traducciones de DV a DF más recientemente utilizadas
- A esta caché de traducción de direcciones se la conoce comúnmente como ***translation-lookaside buffer*** (TLB)

Traducción más rápida con el TLB



Traducción más rápida con el TLB

- **TLB *Hit***: la traducción *sí* está en el TLB
  - Nos ahorramos el acceso a memoria para consultar la tabla de páginas
  - Si el bit de válido está a 0, ***page fault***, la página no está en memoria, excepción
  - Si el bit de válido está a 1, se procede a actualizar el bit de referencia y el de sucio (sólo en caso de escritura)
- **TLB *Miss***: la traducción *no* está en el TLB, se genera una excepción
  - No nos ahorramos el acceso a memoria
  - Traemos la traducción de la tabla de páginas sin comprobar el bit de válido
  - Se re-ejecuta la instrucción, pero en este caso la traducción sí que estará
  - El TLB miss se puede atender tanto por hardware como por software
- En el TLB se utiliza ***write-back***, los bits con información solo se copian en la tabla de páginas cuando la entrada en el TLB es reemplazada
- Los diseñadores de TLBs utilizan diferentes combinaciones:
  - LRU por hardware es demasiado caro, tampoco por software
  - Muchos sistemas dan soporte para selección de una entrada aleatoria (***random***)

TLB Miss Handler - *Ligero*

- El ***TLB miss*** indica que la entrada con la traducción de dirección virtual a dirección física no está presente
- Se genera una excepción para invocar al SO
- El SO trata el fallo vía software
- Se transfiere el control a la dirección de la rutina software (***handler***, manejador) encargada de tratar el TLB miss
- El *TLB miss* tiene un punto de entrada especial para reducir la penalización por fallo del TLB
  - No se realiza un cambio de contexto

Page Fault Handler - *Pesado*

- Los fallos de página en el acceso a datos (MEM) son más complicados:
  1. Ocurren en mitad de la ejecución de la instrucción
  2. La instrucción no se puede completar antes de resolver la excepción
  3. Después de tratar la excepción, la instrucción debe comenzar su ejecución de nuevo como si nada hubiera ocurrido
- Las instrucciones deben ser rearrancables, algo sencillo en la arquitectura RISC-V
- Cuando se produce un fallo de página:
  - Se transfiere el control a una dirección general para una excepción
  - El SO utiliza el registro SCAUSE para determinar la causa de la excepción
  - El SO salva el estado del proceso activo: registros de propósito general, de coma flotante, registro de dirección de tabla de páginas y los registros SEPC y SCAUSE

Protección

- Evitar lecturas o escrituras en determinadas zonas de memoria
- El hardware debe proporcionar al SO los siguientes mecanismos:
  - Soportar dos modos de ejecución: supervisor/kernel y usuario
  - Proporcionar una parte del estado del procesador en modo sólo lectura de tal manera que el usuario no la pueda escribir:
    - Bit de modo usuario/supervisor
    - Puntero de tabla de páginas
    - TLB
- Para modificar este hardware, el SO utiliza instrucciones especiales sólo disponibles en modo supervisor
- Proporcionar mecanismos para cambiar de modo usuario a modo supervisor, como las llamadas al sistema, *syscall exception (**ecall**):*
  - Se salva el PC en el registro SEPC
  - Se pone el procesador en modo supervisor
  - Para volver en modo usuario a la dirección almacenada en el EPC, se utiliza la instrucción **sret**
- Las tablas de páginas se almacenan en el espacio de direcciones del SO