Tema 5. Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

Rendimiento de la jerarquía de memoria

Objetivos

- Analizar conceptos sobre la jerarquía de memoria
- Reflexionar sobre la optimización de los accesos a memoria

Rendimiento de la jerarquía de memoria

- **Contenido**
  - **5.1. Introducción**
  - **5.2. Jerarquía de memoria**
  - **5.3. Memoria caché**
  - **5.4. Mejoras del rendimiento de la memoria principal**
  - **5.5. Memoria Virtual**

5.1. Introducción

Tema 5 Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

INTRODUCCIÓN

- 5.1.1 Principio de localidad

Introducción

- Los usuarios desean memorias grandes, rápidas y baratas. Sin embargo:
  - a menor tiempo de acceso, mayor coste por bit
  - a mayor capacidad, menor coste por bit
  - a mayor capacidad, mayor tiempo de acceso
- Es necesario equilibrar el gasto, el tamaño y la velocidad de las memorias utilizadas.
- Se utilizan diferentes tipos de memoria.

Principio de localidad

- Principio de localidad:
  - Los programas acceden a una porción relativamente pequeña del espacio de direcciones en cualquier instante de de tiempo.
- Dos dimensiones:
  - **Localidad temporal**: si se referencia un elemento, tenderá a ser referenciado pronto
  - **Localidad espacial:** los elementos cercanos al elemento referenciado tenderán a ser referenciados pronto

5.2. Jerarquía de memoria

Tema 5 Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

JERARQUÍA DE MEMORIA

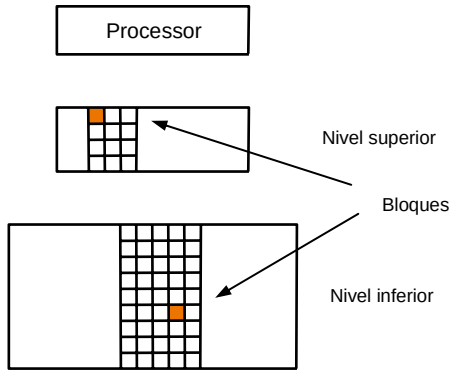
- 5.2.1 Definiciones

Jerarquía de memoria

- Según se desciende en la jerarquía:
  - disminuye el coste por bit
  - aumenta la capacidad
  - aumenta el tiempo de acceso
  - disminuye la frecuencia de accesos por parte de la CPU
- La información se ubica en un nivel dependiendo de su probabilidad de uso.
- Como ésta varía durante la ejecución de un programa, se produce un **tráfico continuo** de información entre los niveles.

Definiciones (I)

- Una jerarquía de memoria consta de varios **niveles**, pero en cada momento se gestiona entre dos niveles.
  - **Nivel superior**: el más cercano a la CPU, corresponde a la memoria más rápida, pequeña y cara
  - **Nivel inferior**: el más alejado a la CPU, corresponde a la memoria más lenta, grande y barata.
- Todos los datos del nivel superior se encuentran también el nivel inferior.
- **Bloque**: la mínima unidad de información que se puede referenciar en la jerarquía de dos niveles.
  - Está compuesto de palabras de memoria



Definiciones (II)

- Un **acierto** (hit) es un acceso con éxito a la memoria del nivel superior, en caso contrario se produce un **fallo** (miss).
- La frecuencia o tasa de aciertos (hit ratio) es el porcentaje de aciertos en accesos a memoria en el nivel superior.
- La tasa de fallos (miss ratio) se define como:

1 - tasa de aciertos

Principal razón de la jerarquía de memoria: rendimiento

La velocidad de aciertos y fallos es importante

Definiciones (III)

- El **tiempo de acierto** (TA) es el tiempo necesario para acceder al nivel superior cuando se produce un acierto:

t. empleado en determinar si la información está en ese nivel

+

t. empleado en acceder a esa información

Definiciones (III)

- La **penalización de fallo** (PF) es el tiempo consumido en obtener la información cuando se produce un fallo:

t. empleado en sustituir un bloque del nivel superior por el bloque correspondiente del nivel inferior

+

tiempo en proporcionar el bloque al dispositivo que lo ha pedido (CPU)

  - Se distingue:
    - **tiempo de acceso**: para acceder a la primera palabra del bloque
    - **tiempo de transferencia**: para transferir el resto del bloque. Depende del ancho de banda entre las memorias.

Definiciones (IV)

- Una **dirección de memoria** especifica, por completo, la localización de un elemento de información dentro de un nivel determinado de la jerarquía de memoria. Consta de:
  - **dirección de la estructura de bloque**: identifica un bloque dentro de ese nivel.
  - **dirección del desplazamiento del bloque**: identifica el elemento dentro de un bloque.

Dirección de la estructura de bloque	Dirección del desplazamiento de bloque
01011010001000001001010	111001110

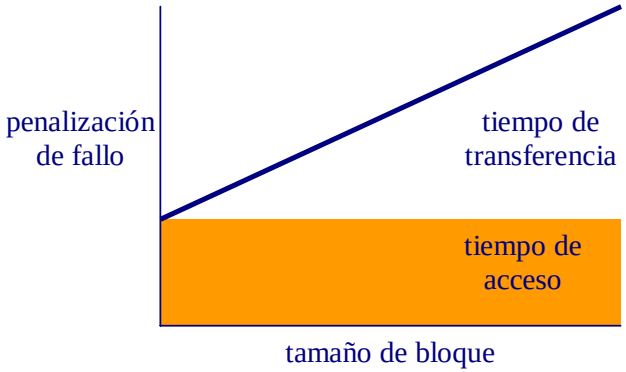
  - Direccionamiento de 32 bits
  - $2^9=512 \Rightarrow$ tamaño del bloque
  - 2^{23} bloques

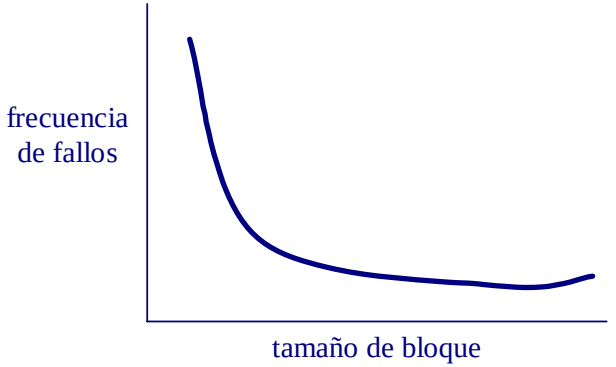
Definiciones (y V)

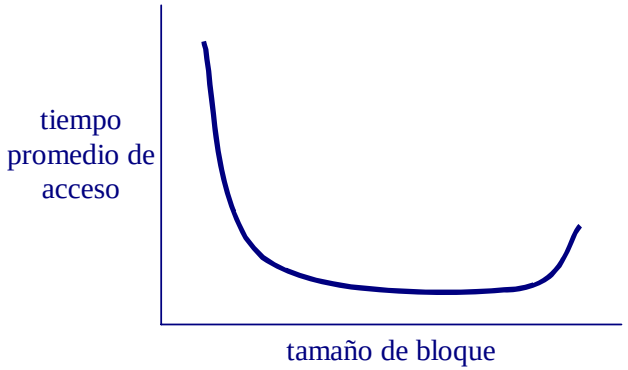
- El **tiempo medio de acceso a memoria** (TMA) se define como:

  tiempo de acierto + tasa de fallos * penalización de fallo

  - PF se puede medir en ciclos de reloj
  - Existe una relación entre el TMA y el tamaño del bloque.







5.3. Memoria caché

Tema 5 Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

MEMORIA CACHÉ

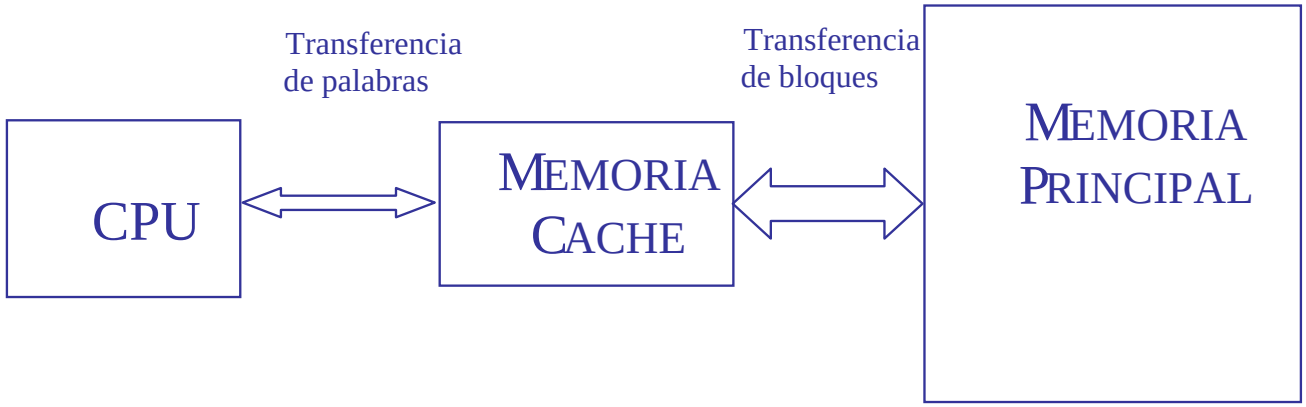
- 5.3.1 Organización de la caché
- 5.3.2 Ubicación de bloque
- 5.3.3 Estrategias de reemplazo
- 5.3.4 Operación de la caché
- 5.3.5 Políticas de escritura
- 5.3.6 Rendimiento

Memoria caché (I)

- La memoria de la CPU es **mucho más rápida** que la memoria principal.
- Lo ideal sería que la memoria principal fuera de la misma tecnología que los registros de la CPU, pero debido a su alto coste, se tiende a **soluciones intermedias**.
- Aprovechando el principio de localidad se coloca una memoria muy rápida (**memoria caché**) entre la CPU y la memoria principal.
- La CPU accede más veces a memoria caché que a memoria principal.
- Contiene una copia de aquellas posiciones de memoria principal que están siendo utilizadas por la CPU.

Memoria caché (y II)

- El funcionamiento de la memoria caché se basa en la **transferencia de bloques** entre memoria principal y caché, y en la **transferencia de palabras** entre memoria caché y CPU.

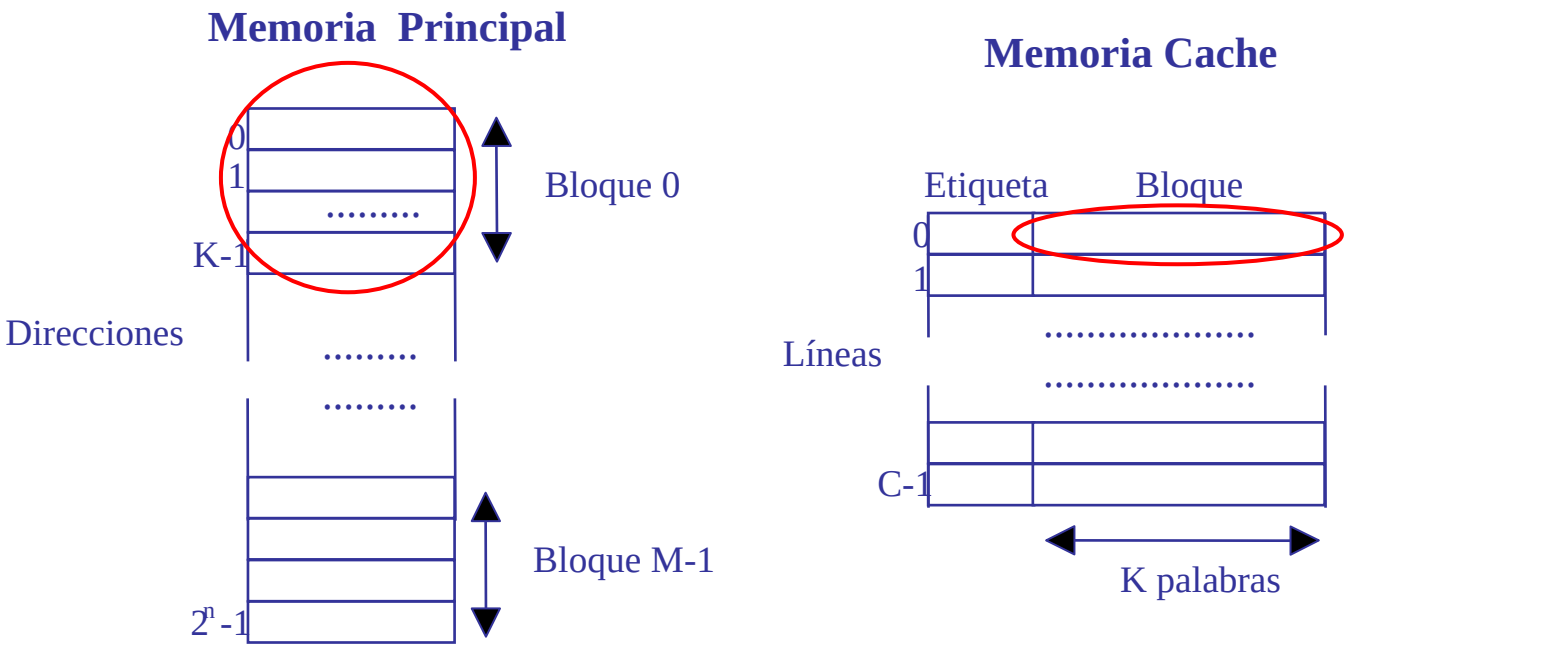


Cuestiones a tener en cuenta:

- ¿Dónde se ubica un bloque en la caché?
- ¿Cómo se encuentra un bloque en la caché?
- ¿Qué bloque debe reemplazarse en caso de fallo?
- ¿Qué ocurre en una escritura?

¿Dónde se ubica un bloque en la caché?

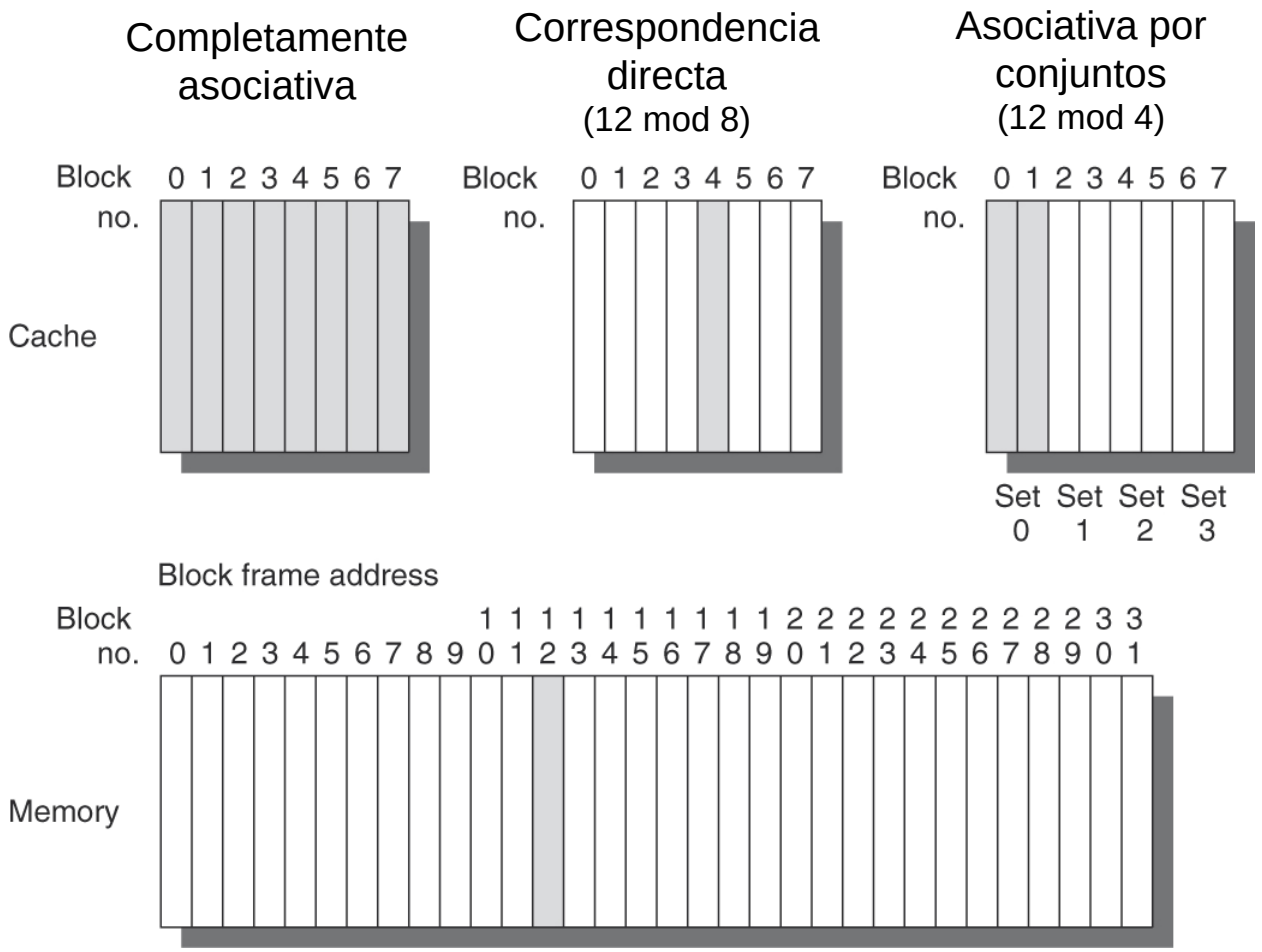
- La memoria caché se divide en **líneas**, que poseen el mismo tamaño que los bloques de memoria principal.



- A partir de la dirección de memoria suministrada por la CPU, se convierte en la dirección de caché correspondiente de acuerdo con el **método de ubicación de bloques**.

¿Dónde se ubica un bloque en la caché?

- Las restricciones de ubicación del bloque dan lugar a tres categorías en la organización de la caché:



¿Dónde se ubica un bloque en la caché?

- Es necesaria una **función de correspondencia** que haga corresponder bloques de memoria principal con líneas de memoria caché.
  - Correspondencia directa: Cada bloque sólo puede aparecer en un lugar en la caché.

    línea = dirección de la estructura de bloque **mod** n° de líneas

  - Correspondencia asociativa por conjuntos: Un bloque se puede colocar en un conjunto restringido de lugares en la caché.

    línea = dirección de la estructura de bloque **mod** n° de conjuntos

  - Correspondencia completamente asociativa: Un bloque se puede colocar en cualquier parte de la caché.

¿Dónde se ubica un bloque en la caché?

Ejemplo:

- El tamaño de la memoria **caché** es de **4Kb**
- **Bloques de 16 bytes** para transferir entre MP y caché
- Esto indica que la **caché** tiene (4096/16) **256 líneas**
- **Memoria principal** tiene **64Kb** por lo que el **bus** de **direcciones** es de **16 bits**. Luego, la memoria principal posee **4096 bloques**.

¿Cómo se encuentra un bloque en la caché?

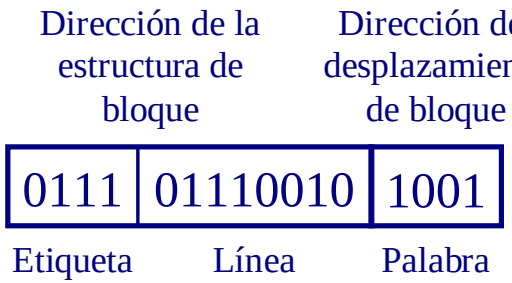
- Las cachés incluyen una etiqueta de dirección en cada línea que identifica la dirección de la estructura del bloque.
- Esta etiqueta se comprueba para ver si coincide con la dirección de la estructura del bloque requerido por la CPU.
- Como la velocidad es esencial, todas las posibles etiquetas se buscan en paralelo
- Se añade un **bit de validez** a la etiqueta para indicar si la línea de la cache contiene información válida.

Correspondencia directa (I)

- Un bloque sólo puede colocarse en una línea de la caché, siguiendo la expresión:

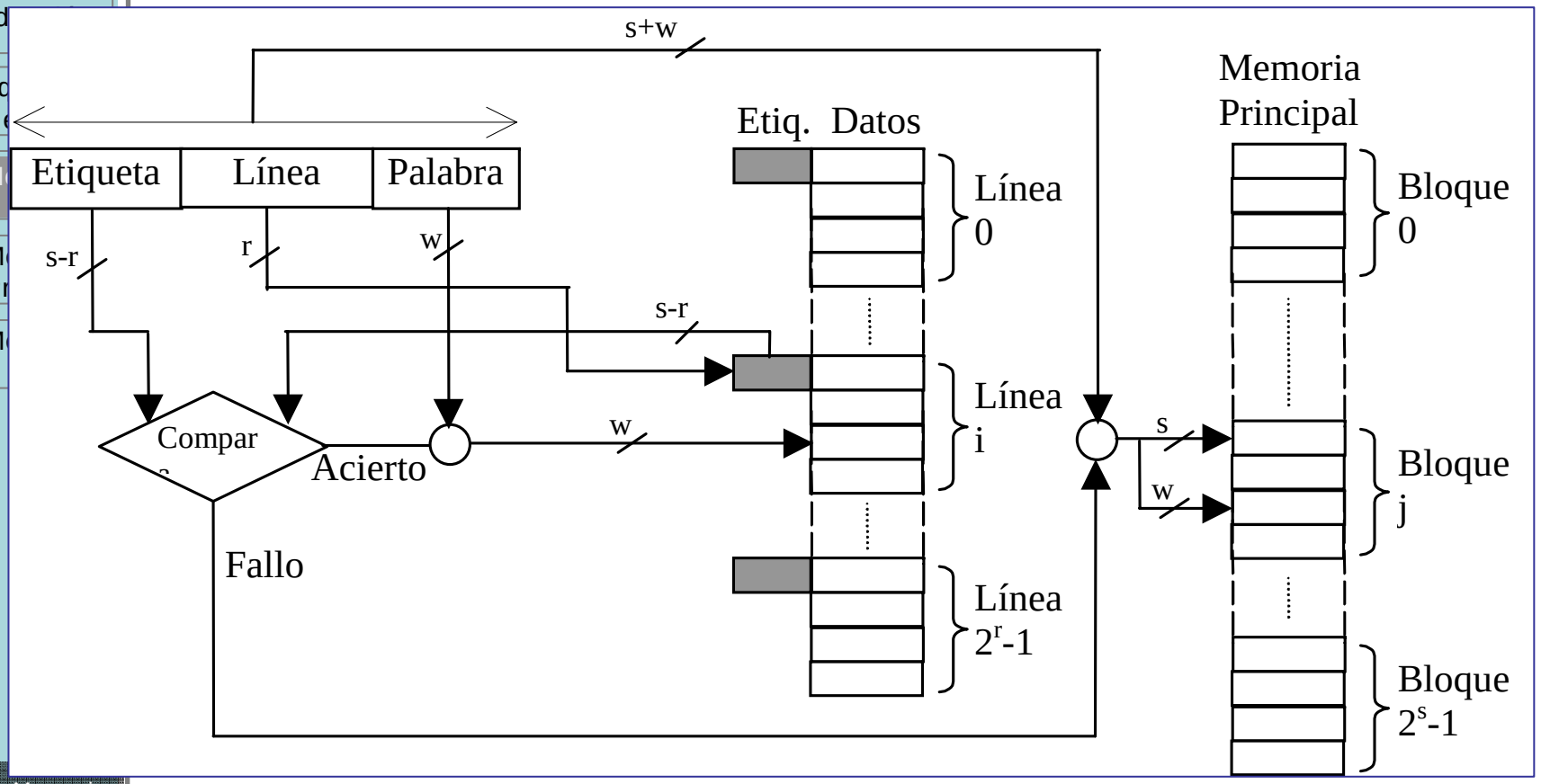
línea = dirección de la estructura de bloque mod **n° de líneas**

- Cada dirección de memoria principal puede verse dividida en 3 campos:

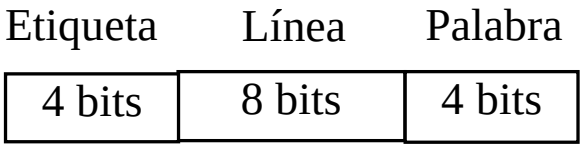


- La dirección de la estructura de bloque se divide en **etiqueta** y **línea**.
  - Con los bits de **línea** se puede averiguar directamente en qué línea de la caché se debe guardar ese bloque.
  - Cuando un bloque se almacena en una línea es necesario **etiquetarla** para diferenciarlo del resto de bloques que podrían estar en dicha línea.

Correspondencia directa (II)



Correspondencia directa (III)



Memoria Principal

Dirección Dato

0000	12
0001	34
0002	56
000F	78
F800	AB
F801	0F
F802	FF
F80F	54
FFF0	FF
FFFD	21
FFFE	33
FFFF	55

Bloque 0

Bloque 3968

Bloque 4095

Memoria Cache

Etiqueta	Datos	Nº Línea
0	123456.....78	0 (00)
.....	.....	1 (01)
.....	.....	.....
F	AB0FFF.....54	128 (80)
.....	.....	.....
F	FF.....213355	255 (FF)

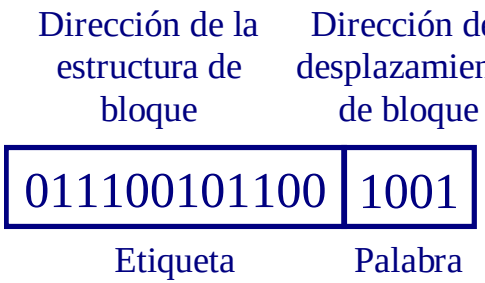
4 bits | 16 bytes | 8 bits

Correspondencia directa (y IV)

- Ventajas:
  - Técnica muy simple
  - Fácil de implementar
- Desventajas:
  - Existe una gran limitación al poder ubicar un bloque de memoria principal en una única línea de caché.

Correspondencia asociativa (I)

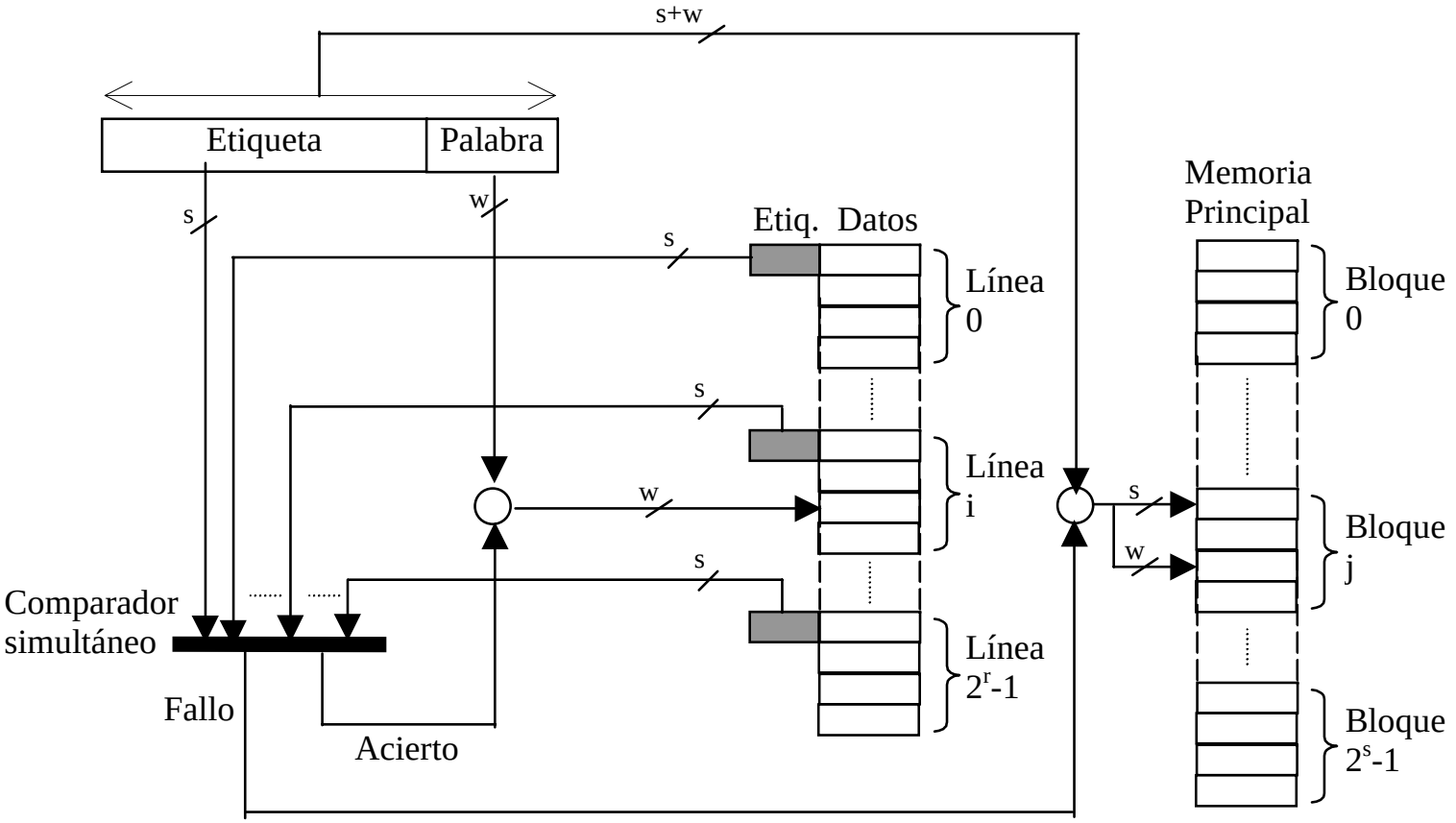
- Un bloque se puede ubicar en **cualquier posición** de la caché.
- Las direcciones de memoria están formadas por una **etiqueta** y una **palabra**.



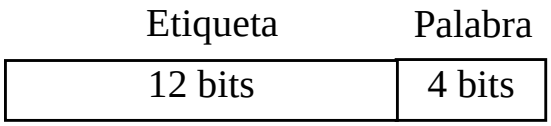
- El campo de **etiqueta** se utiliza para identificar el bloque que se encuentra almacenado en esa línea.

Correspondencia asociativa (II)

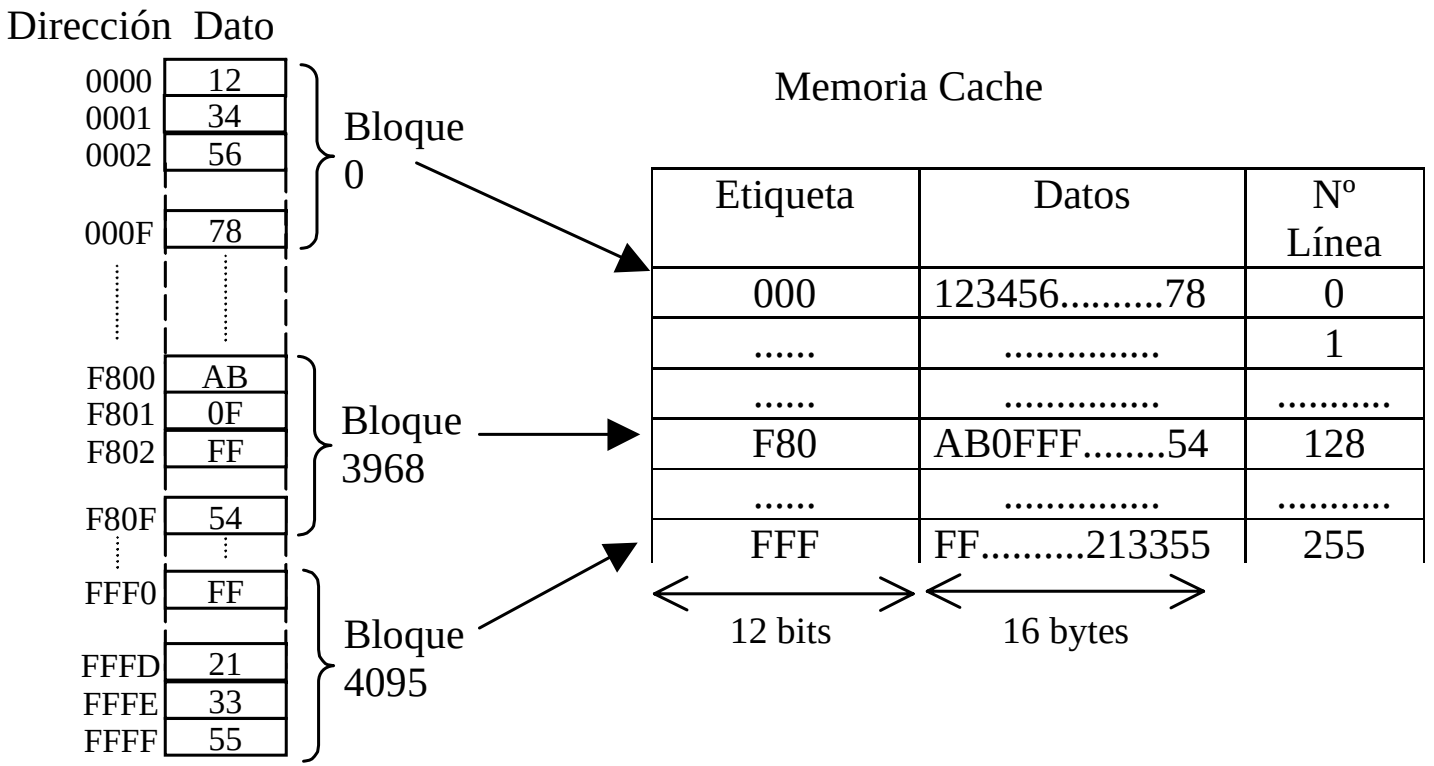
- Para determinar si un bloque está en caché, se examinan en **paralelo** todas las etiquetas de las líneas de memoria caché.



Correspondencia asociativa (III)



Memoria Principal



Correspondencia asociativa (y IV)

- Ventaja:
  - Cualquier combinación de bloques de memoria principal puede estar en la caché en un determinado instante.
- Desventaja:
  - Se necesita una circuitería muy compleja para realizar la búsqueda de un bloque.

Correspondencia asociativa por conjuntos

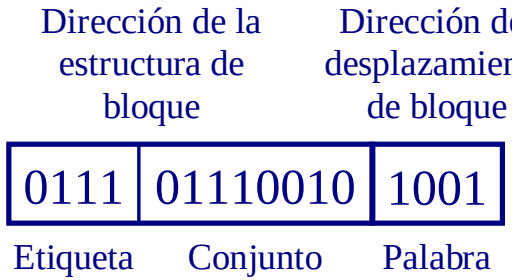
- Combina las 2 técnicas anteriores.
- Un **conjunto** es un grupo de dos o más líneas de caché.
- Al número de líneas que forman un conjunto se le denomina **vías**.
- Se dice que existe correspondencia asociativa por conjuntos de N vías.
- Un **bloque de memoria** principal se corresponde con un **conjunto**, pudiéndose ubicar en cualquiera de las líneas que lo componen.
- El conjunto se escoge de forma similar a la correspondencia directa:

  conjunto = dirección de la estructura de bloque mod **n° de conjuntos**

- Dentro de ese conjunto, se puede situar el bloque en cualquier línea.

Correspondencia asociativa por conjuntos (II)

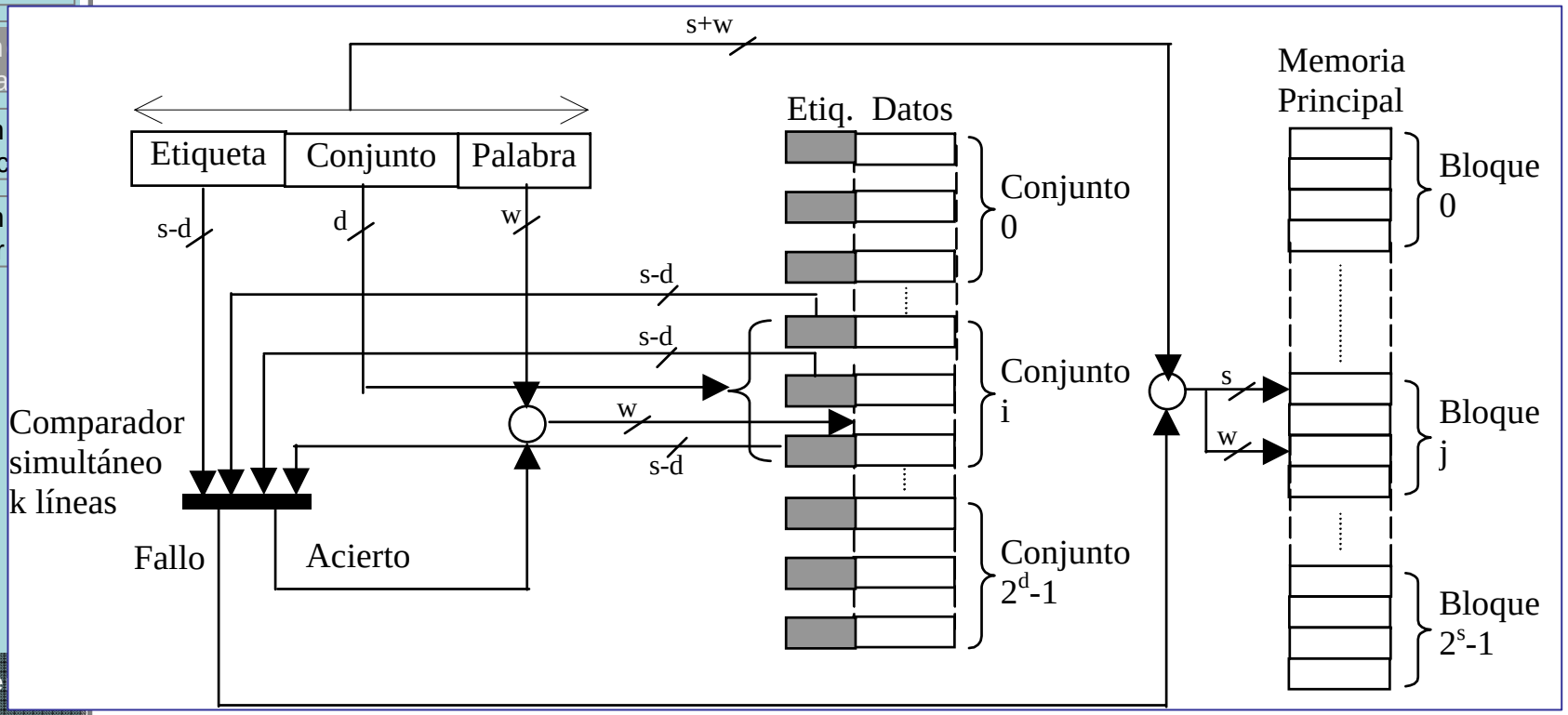
- Cualquier tipo de correspondencia puede expresarse como correspondencia asociativa por conjuntos de 1 vía (directa), P vías (totalmente asociativa) o N vías.
- La dirección de memoria se divide en 3 campos:



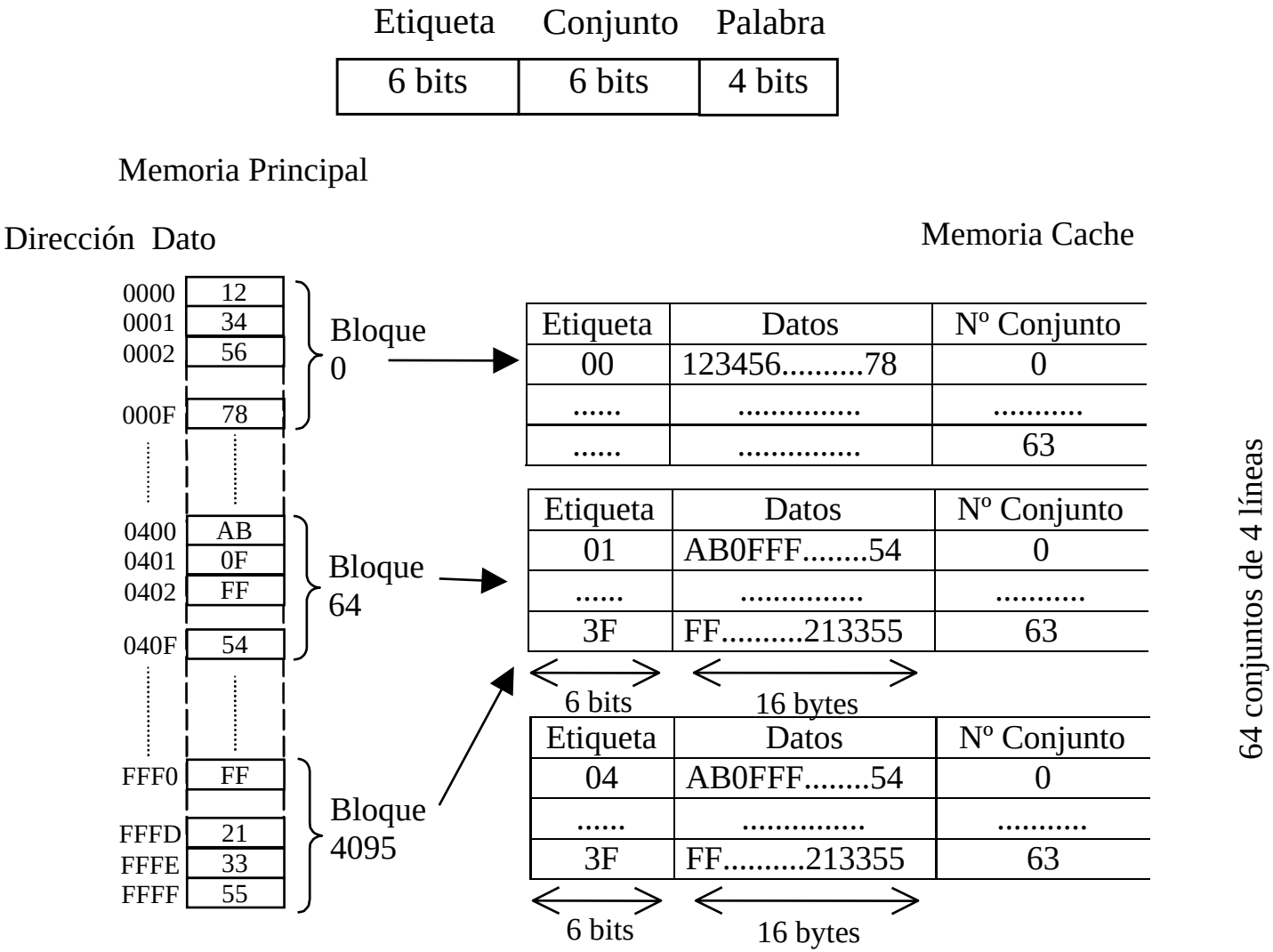
- Si se incrementa el grado de asociatividad (N) vías, aumenta el n° de líneas por conjunto, disminuyendo el número de conjuntos y aumentando la etiqueta.

Correspondencia asociativa por conjuntos (III)

- Para realizar la búsqueda de un bloque, primero se aplica correspondencia directa y posteriormente correspondencia asociativa.



Correspondencia asociativa por conjuntos (IV)



Correspondencia asociativa por conjuntos (y V)

- Ventajas:
  - Aúna ventajas de los otros dos métodos.
  - Usualmente se emplean conjuntos de 2 vías, mejorando las tasa de acierto frente a la correspondencia directa.
  - También se usan las 4 vías, que proporciona una relativa mejora a un coste moderado.
- Desventajas:
  - A partir de 4 vías, el beneficio de mayor número de vías no compensa debido al mayor coste.

¿Qué bloque debe reemplazarse en caso de fallo?

- Cuando se introduce un nuevo bloque en la cache, debe sustituirse uno de los bloques existentes.
- Para el caso de correspondencia directa sólo hay una posible línea para cada bloque.
- Para las técnicas asociativas se requieren **algoritmos de sustitución**. Estos algoritmos deben implementarse por hardware.

¿Qué bloque debe reemplazarse en caso de fallo?

Método aleatorio

- Se escoge aleatoriamente uno de los bloques candidatos.
- Implementación hardware sencilla.

Método LRU (Least recently used)

- Se sustituye el bloque que hace más tiempo que no fue referenciado
- Fácil de implementar para asociativas por conjuntos de dos vías
- Cuando se referencia una línea se pone a uno su bit de uso y a cero el de la otra línea del mismo conjunto

¿Qué bloque debe reemplazarse en caso de fallo?

- **Método FIFO (First in first out):**
  - Se reemplaza el bloque que hace más tiempo que está en caché
  - Puede implementarse mediante una técnica de buffer circular
- **Método LFU:**
  - Se elimina el bloque que ha sido menos referenciado
  - Se debe asociar un contador a cada línea

Operación de la caché (I)

1. Inicialización

- Cuando se enciende el computador:
  - La memoria principal es cargada por el disco con una serie de programas

2. Lectura

- La CPU solicita una palabra de memoria:
  - Se lee primero de caché (se puede leer al mismo tiempo que se compara la etiqueta)
  - Si allí no se encuentra la palabra requerida (fallo de caché) se acude a MP, de donde se lee la palabra, y se emplaza el bloque que contiene esa palabra en la caché.
- Las lecturas dominan los accesos a la caché, por lo que se debe **optimizar la caché para las operaciones de lectura**.

Operación de la caché (y II)

3. Escritura

- La CPU pide reemplazar una porción de bloque (palabra)
- Esto implica la secuencia de:
  - leer el bloque original
  - modificar una parte de él
  - escribir de nuevo el bloque
- La modificación del bloque no puede comenzar hasta que no se haya comprobado la etiqueta → más lenta que lectura

¿Qué ocurre en una escritura?

- **Asegurar coherencia entre caché y memoria**
- Aciertos:
  - Escritura directa (write-through)
  - Postescritura (write-back)
- Fallos
  - no ubicar en escritura (write-non-allocate)
  - ubicar en escritura (wrte-allocate)

Escritura directa (write through)

- **Acierto**: la información se escribe en la caché y en la MP en paralelo.
- **Fallo**: Se escribe en MP. Usualmente el bloque no es cargado en la caché (**no ubicar en escritura**). Menos frecuente es cargar el bloque en caché (**ubicar en escritura**).
- **Ventaja**:
  - Los contenidos de MP y caché están siempre actualizados
- **Desventajas**:
  - Mucho tráfico de datos entre MP y CPU.
  - La **CPU debe detenerse** en las escrituras ya que siempre tiene que acceder a MP.

Postescritura (write back)

- **Acierto**: la información se escribe sólo en el bloque de caché. Este bloque sólo se escribe en MP al ser reemplazado.
- **Fallo**: Se suele utilizar la **ubicación en escritura** (similar a un fallo de lectura). Las escrituras posteriores no provocan fallo de caché.
- Se utiliza un **bit de modificación** para saber si ese bloque está **limpio** o **sucio**.
- **Ventajas:**
  - Las escrituras se realizan a la **velocidad de la memoria caché**.
  - La postescritura utiliza **menos ancho de banda** de memoria ya que varias escrituras requieren una única escritura del bloque en MP. Esto reduce la espera de CPU.
- **Desventaja:**
  - La caché y la MP son inconsistentes.

Múltiples niveles de caché

- El esquema descrito se puede extender a más de un nivel de memoria caché.
- Cuando una memoria caché recibe una petición y el dato no está presente en la memoria, el dato se le pide a la memoria del siguiente nivel
- La memoria de mayor nivel (comenzando por el 1 al lado del procesador) tiene menor tamaño
- Los procesadores suelen tener el primer nivel de memoria dentro del encapsulado
- En ciertos modelos también encontramos un segundo y tercer nivel (L2 Cache, L3 Cache) que puede encontrarse dentro o fuera del chip

Caché Pentium

- **Caché Instrucciones**
  - P y Ppro: 8 ó 16 KB
  - P4: 12000 microinstrucciones
- **Caché de nivel 1**
  - P: 16 KB
  - Ppro: 8 ó 16 KB
  - PIII: 8 KB
  - PIV: 8 KB
- **Caché de nivel 2**
  - P: 256 ó 512 KB
  - Ppro: 128, 256, 512 KB ó 1 MB
  - PIII: 256 KB
  - PIV: 256, 512 KB ó 1 MB
- **Caché de nivel 3**
  - PIV: 2MB

Rendimiento de la caché (I)

- **Frecuencia de fallos de la caché (FF)**
  - Es independiente de la velocidad del hardware.
  - Es engañosa, ya que no tiene en cuenta la penalización de fallos (PF)
  - El objetivo de una jerarquía de memoria es reducir el tiempo de ejecución, no los fallos.
- **Tiempo medio de acceso a memoria (TMA)**
  - Contempla tanto la FF como la PF:

$$TMA = TA + FF * PF$$

- **Tiempo de ejecución (de una tarea o programa) (T_{CPU})**
  - Medida real del rendimiento

$$T_{CPU} = NI * CPI * T_{reloj} = NI * (CPI_{ejec} + CPI_{mem}) * T_{reloj}$$

Rendimiento de la caché (II)

- CPI_{MEM}: ciclos en espera de CPU por referencias a memoria
  - Debidos a fallos de caché.

$$CPI_{mem} = \frac{\text{ciclos de detención debido a fallos}}{NI} = CPI_{mem}(\text{lecturas}) + CPI_{mem}(\text{escrituras})$$

$$CPI_{mem}(\text{lecturas}) = \frac{\text{fallos en lecturas} * PF_{\text{lecturas}}}{NI} = \frac{\text{lecturas} * FF_{\text{lecturas}} * PF_{\text{lecturas}}}{NI}$$

$$CPI_{mem}(\text{escrituras}) = \frac{\text{fallos en escrituras} * PF_{\text{escrituras}}}{NI} = \frac{\text{escrituras} * FF_{\text{escrituras}} * PF_{\text{escrituras}}}{NI}$$

$$CPI_{mem} = \frac{NM * FF * PF}{NI}$$

Rendimiento de la caché (y IV)

$$T_{CPU} = NI * (CPI_{ejec} + CPI_{mem}) * T_{reloj}$$

$$T_{CPU} = NI * \left(CPI_{ejec} + \frac{NM}{NI} * FF * PF \right) * T_{reloj}$$

- Algunos diseñadores prefieren medir los fallos por instrucción en lugar de fallos por acceso a memoria:

$$T_{CPU} = NI * \left(CPI_{ejec} + \frac{\text{fallos}}{\text{instrucción}} * PF \right) * T_{reloj}$$

- Puesto que el CPI global depende tanto de la FF como de la PF, **el impacto de la caché sobre el rendimiento global ha de ser analizado en función de las características de la MP**.
- Los cambios en la FF (diseño de caché) puede causar cambios sustanciales o despreciables, dependiendo de la PF (diseño o velocidad de MP)

Ejemplo

- Suponer que la penalización de fallos de la caché es de 200 ciclos de reloj y que todas las instrucciones normalmente emplean 1.0 ciclos de reloj (ignorando las detenciones de memoria). Suponer que la frecuencia de fallos es de 2%, hay una media de 1.5 referencias a memoria por instrucción y que el número medio de fallos a la caché por 1000 instrucciones es 30. ¿Cuál es el impacto en el rendimiento cuando se incluye el comportamiento de la caché?Calcula el impacto utilizando tanto los fallos por instrucción como la frecuencia de fallos.

$$T_{CPU} = NI * \left(CPI_{ejec} + \frac{NM}{NI} * FF * PF \right) * T_{reloj}$$

$$T_{CPU} = NI * \left(CPI_{ejec} + \frac{fallos}{instrucción} * PF \right) * T_{reloj}$$

Ejemplo

- Suponer que la penalización de fallos de la caché es de 200 ciclos de reloj y que todas las instrucciones normalmente emplean 1.0 ciclos de reloj (ignorando las detenciones de memoria). Suponer que la frecuencia de fallos es de 2%, hay una media de 1.5 referencias a memoria por instrucción y que el número medio de fallos a la caché por 1000 instrucciones es 30. ¿Cuál es el impacto en el rendimiento cuando se incluye el comportamiento de la caché?Calcula el impacto utilizando tanto los fallos por instrucción como la frecuencia de fallos.

$$T_{CPU} = NI \times (1.0 + (30/1000 \times 200)) * T_{reloj} = NI \times 7.0 \times T_{reloj}$$

$$T_{CPU} = NI \times (1.0 + (1.5 \times 2\% \times 200)) * T_{reloj} = NI \times 7.0 \times T_{reloj}$$

T_{CPU} se incrementa en 7 con una caché con fallos.

Sin jerarquía de memoria el CPI se incrementaría en 1.0+200x1.5 o 301

- Cuanto más bajo CPI_{ejec} más pronunciado el impacto
- Una frecuencia mayor de reloj lleva a una PF mayor

Ejemplo

- ¿Cuál es el impacto de dos organizaciones cachés diferentes en el rendimiento de un procesador? Suponer que el CPI con una caché perfecta es 1.6 con una duración del ciclo de reloj es 0.36ns; hay 1.4 referencias a memoria por instrucción, el tamaño de ambas cachés es de 128KB y ambas tienen un tamaño de bloque de 64bytes. Un caché es de correspondencia directa y la otra es asociativa por conjunto de dos vías. Como la velocidad de la CPU está ligada directamente a la velocidad de un acierto en la caché, suponer que la duración del ciclo de reloj debe alargarse 1.35 veces para acomodar la selección en la caché asociativa por conjuntos. Para la primera aproximación, la penalización de fallos es de 65ns para cualquier organización de la caché (En la práctica se debe redondear hacia arriba o hacia abajo un número entero de ciclos de reloj). Primero calcular el tiempo medio de acceso a memoria y después el rendimiento de la CPU. Suponer que el tiempo de acierto es 1 ciclo de reloj, la frecuencia de fallos de la caché de correspondencia directa es de 2.1% y la de la caché asociativa por conjuntos es de 1.9%.

Ejemplo

- Tiempo medio de acceso a memoria:

$$TMA = TA + FF*PF$$

$TMA_{1\text{-vía}} = 0.35 + (0.021 \times 65) = 1.72$ ns

$TMA_{2\text{-vías}} = 0.35 \times 1.35 + (0.019 \times 65) = 1.71$ ns

- Rendimiento del procesador:

$$T_{CPU} = NI * \left(CPI_{ejec} + \frac{NM}{NI} * FF * PF \right) * T_{reloj}$$

Se sustituye 65ns por $PF*T_{CLK}$

$T_{CPU\ 1vía} = NI * (1.65 \times 0.35 + (0.021 \times 1.4 \times 65)) = 2.47 \times NI$

$T_{CPU\ 2vía} = NI * (1.65 \times 0.35 \times 1.35 + (0.019 \times 1.4 \times 65)) = 2.49 \times NI$

El rendimiento relativo:

$$\frac{T_{CPU\ 2vía}}{T_{CPU\ 1vía}} = \frac{2.49 \times NI}{2.47 \times NI} = 1.01$$

Optimizaciones

- Tiempo medio de acceso a memoria:

$$TMA = TA + FF*PF$$

- Optimizaciones:
  - Reducir Frecuencia de fallos: tamaño del bloque más grande, tamaño de la caché más grande y mayor asociatividad.
  - Reducir Penalización de fallos: caché multinivel y dar prioridad a las lecturas sobre las escrituras.
  - Reducir el tiempo de acierto: evitar la traducción de la dirección cuando se indexa a la caché.

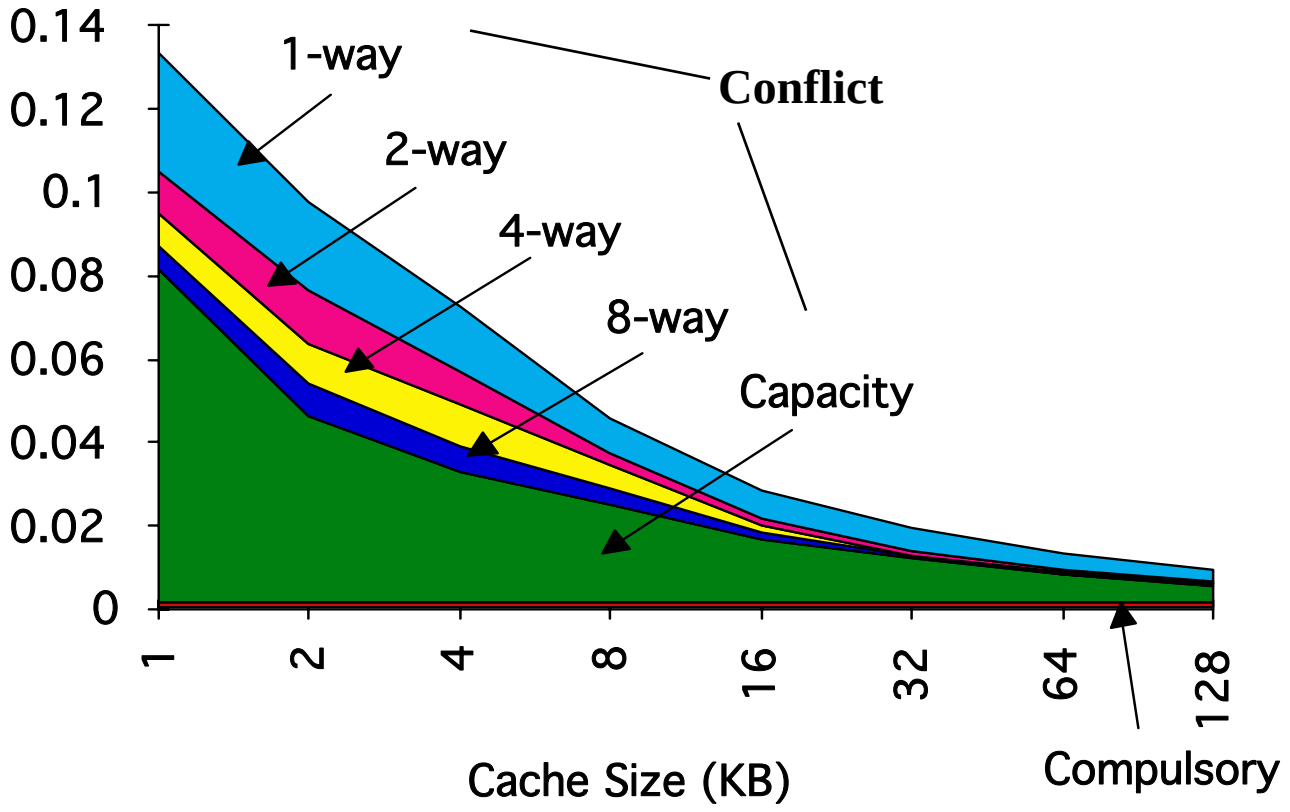
Optimizaciones

- Tipos de fallos
  - Por carga inicial o forzosos (*Compulsory*)
    - Carga inicial de la caché.
    - Fallos incluso en una caché infinita
  - De capacidad (*Capacity*)
    - Si la caché no puede almacenar todos los bloques de un programa en ejecución.
  - Por conflicto (*Conflict*)
    - Aún habiendo líneas libres no se puede cargar un bloque
    - Suele suceder en cachés con un grado de asociatividad (correspondencia directa o asociativa por conjuntos de pocas vías)

Tasa de Fallo = Nº Fallos / Nº Accesos

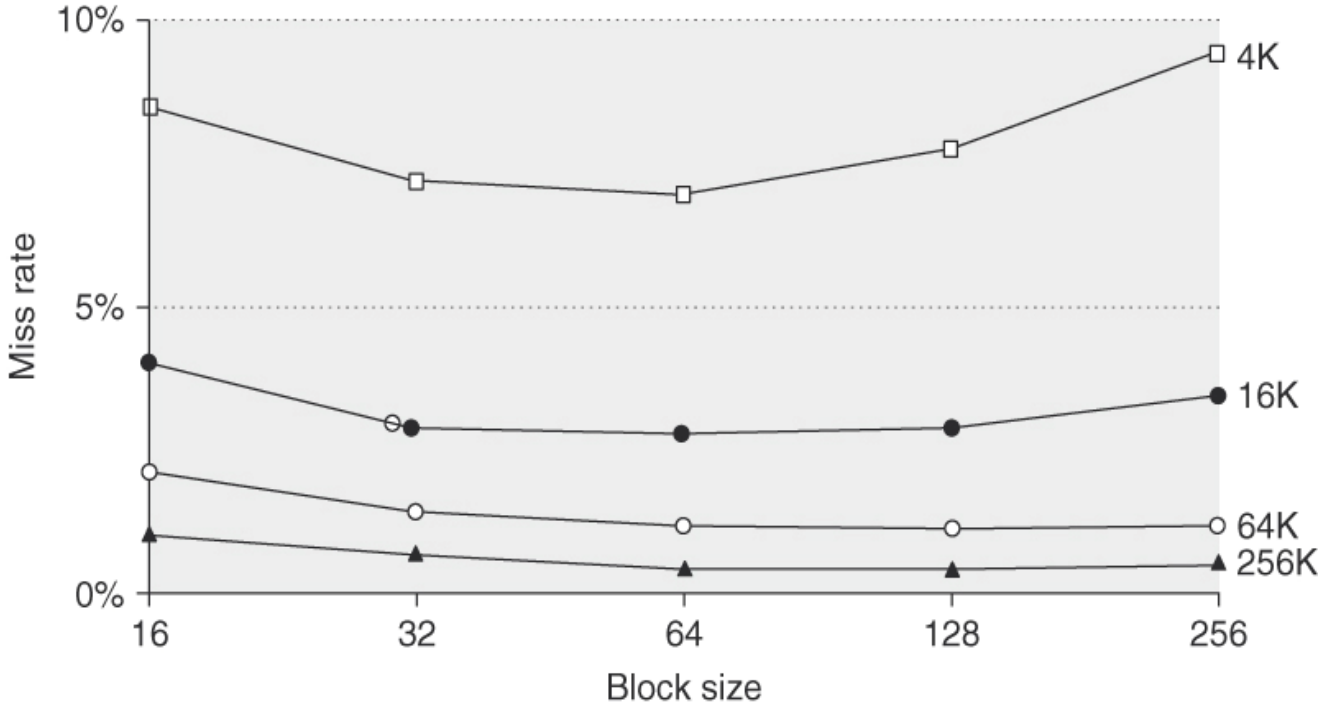
Optimizaciones

- Tipos de fallos



Optimizaciones

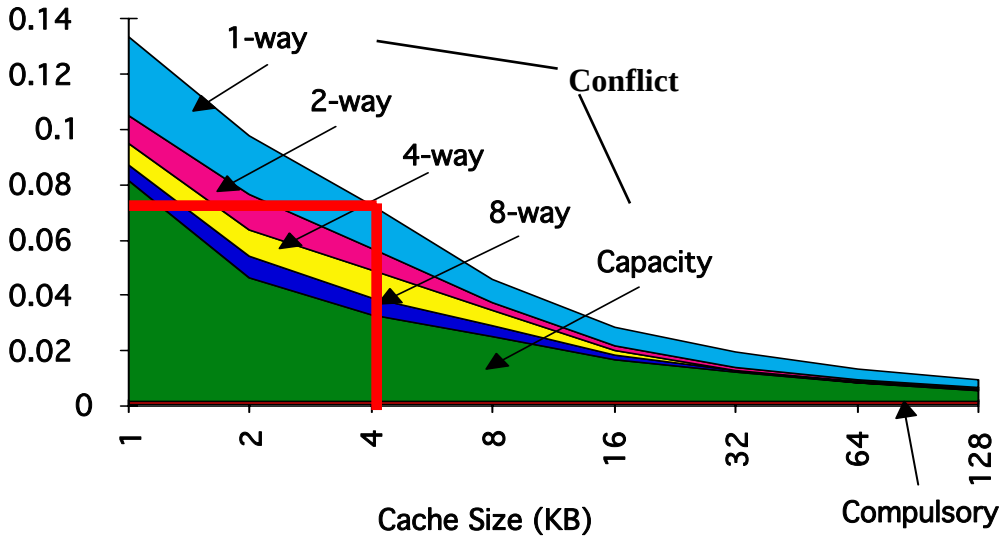
- Reducir frecuencia de fallos aumentando el tamaño del bloque



- Se reducen los fallos por carga inicial pero aumentan los fallos por capacidad y conflicto
- Los fallos aumentan si el tamaño del bloque es demasiado grande en relación al tamaño de la caché.
- Aumentar el tamaño del bloque incrementa la penalización por fallos.

Optimizaciones

- Reducir frecuencia de fallos aumentando el tamaño de la caché
  - Potencialmente se alarga el tiempo de acierto y aumenta el coste.
- Reducir frecuencia de fallos aumentando la asociatividad



- Una caché de correspondencia directa de tamaño N tiene la misma frecuencia de fallos que una asociativa por conjunto de 2 vías de tamaño N/2.
- Aumentar la asociatividad puede aumentar el coste e incrementar el tiempo de acierto.

Optimizaciones

- Reducir la penalización de fallos con cachés multinivel
  - La caché de primer nivel pequeña para para ajustarse al tiempo de ciclo de reloj del procesador rápido.
  - La caché de segundo nivel lo suficientemente grande para que haya pocos accesos a la MP.

$$TMA = TA_{L1} + FF_{L1} * PF_{L1}$$
$$PF_{L1} = TA_{L2} + FF_{L2} * PF_{L2}$$

- Hay que diferenciar:
  - *FF_local = nº de fallos / nº de accesos a la caché*
  - *FF_global = nº de fallos / nº total de accesos realizados por la CPU*
- En general se cumple: $FF\_local \geq FF\_global$
- Y en particular:
  - $FF\_local_{L1} = FF\_global_{L1}$
  - $FF\_local_{L2} > FF\_global_{L2}$

Ejemplo

- Suponed que en 1000 referencias a memoria hay 40 fallos en la caché de primer nivel y 20 fallos en la caché de segundo nivel. ¿Cuáles son los distintas frecuencias de fallos? Asumid que la penalización de fallos de la caché L2 a la memoria es de 200 ciclos de reloj, el tiempo de acierto a la caché L2 es de 10 ciclos de reloj, el tiempo de acierto a la caché L1 es de 1 ciclos de reloj y que hay 1.5 referencias a memoria por instrucción. ¿Cuál es el tiempo medio de acceso?

Ejemplo

- Suponed que en 1000 referencias a memoria hay 40 fallos en la caché de primer nivel y 20 fallos en la caché de segundo nivel. ¿Cuáles son los distintas frecuencias de fallos? Asumid que la penalización de fallos de la caché L2 a la memoria es de 200 ciclos de reloj, el tiempo de acierto a la caché L2 es de 10 ciclos de reloj, el tiempo de acierto a la caché L1 es de 1 ciclos de reloj y que hay 1.5 referencias a memoria por instrucción. ¿Cuál es el tiempo medio de acceso?

$FF\_local_{L1} = FF\_global_{L1} = FF_{L1} = 40/1000 = 0.04$ (4%)

$FF\_local_{L2} = 20/40 = 0{,}5$ (50%)

$FF\_global_{L2} = 20/1000 = 0.02$ (2%)

$$TMA = TA_{L1} + FF_{L1} * (TA_{L2} + FF_{L2} * PF_{L2}) =$$

$$= 1 + 4\% \times (10 + 50\% \times 200) = 5.4 \text{ ciclos de reloj}$$

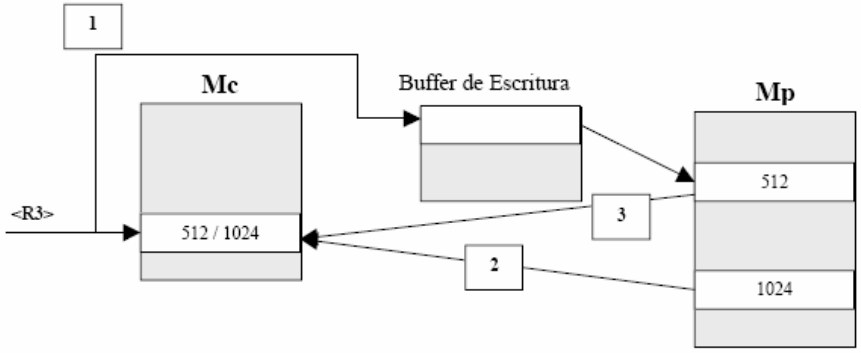
Optimizaciones

- Reducir la penalización de fallos dando prioridad a las lecturas sobre las escrituras
  - Se realiza la lectura antes de que la escritura se haya completado.
  - Ejemplo: Caché de correspondencia directa y escritura directa mejorada con un buffer de escritura de tamaño apropiado. Se hace corresponder los bloques en los que se encuentran las direcciones 512 y 1024 sobre la misma línea de la caché. Supongamos que se ejecuta el siguiente programa:

(1) SW R3, 512(R0) ; M[512] <-- <R3> // escritura sobre M[512]

(2) LW R1, 1024(R0); R1 <-- M[1024] // lectura de M[1024]

(3) LW R2, 512(R0); R2 <-- M[512] // lectura de M[512]



Optimizaciones

- Reducir la penalización de fallos dando prioridad a las lecturas sobre las escrituras

  (1) SW R3, 512(R0) ; M[512] <-- <R3> // escritura sobre M[512]

  (2) LW R1, 1024(R0); R1 <-- M[1024] // lectura de M[1024]

  (3) LW R2, 512(R0); R2 <-- M[512] // lectura de M[512]

- Dos soluciones para evitar que la lectura 3 no lea un dato inconsistente en memoria:
  - Esperar a que se vacíe el buffer. Esto incrementará la penalización de fallos de lectura en un factor de 1.5.
  - En un fallo de lectura chequear el contenido del buffer de escritura y continuar la lectura si no hay conflicto.

5.4. Mejoras del rendimiento de la memoria principal

Tema 5 Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

5.4. MEJORAS DEL RENDIMIENTO DE LA MEMORIA PRINCIPAL

5.4.1 Técnicas de mejora del rendimiento

5.4.2 Anchura de la memoria

5.4.3 Memoria entrelazada

5.4.4 Bancos de memoria independientes

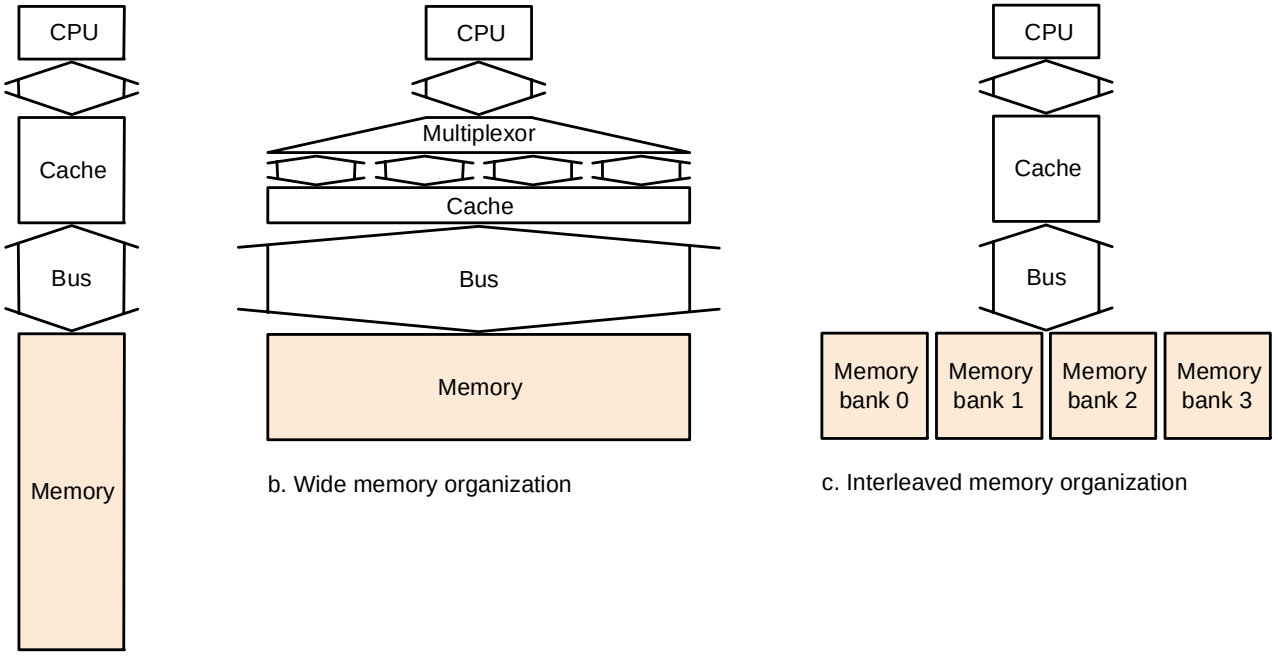
5.4.5 Evitar conflictos entre bancos de memoria

Introducción

- La memoria principal satisface las demandas de las cachés y unidades vectoriales y sirve como interfaz de E/S
- Las medidas de rendimiento en la MP hacen énfasis en la latencia y el ancho de banda:
  - La latencia es la preocupación primordial de la caché (afecta a la penalización de fallos)
  - El ancho de banda es la preocupación primordial de las E/S y de las unidades vectoriales
- La latencia se expresa utilizando:
  - Tiempo de acceso: tiempo desde que se pide una lectura hasta que llega la palabra deseada.
  - Duración del ciclo: tiempo mínimo entre peticiones a memoria.

Organizaciones para mejorar el rendimiento

- Generalmente resulta más fácil mejorar el ancho de banda que reducir latencia.
- Una mejora en el ancho de banda permite incrementar tamaño de bloques sin incremento de penalización de fallos.



a. One-word-wide memory organization

b. Wide memory organization

c. Interleaved memory organization

Memoria principal más ancha

- Las cachés están organizadas normalmente con una anchura de una palabra
- La MP tiene la anchura de una palabra para que coincida con anchura de la caché.
- Duplicar o cuadruplicar el ancho de la memoria, duplicará o cuadruplicará el ancho de banda de memoria.

Memoria principal más ancha

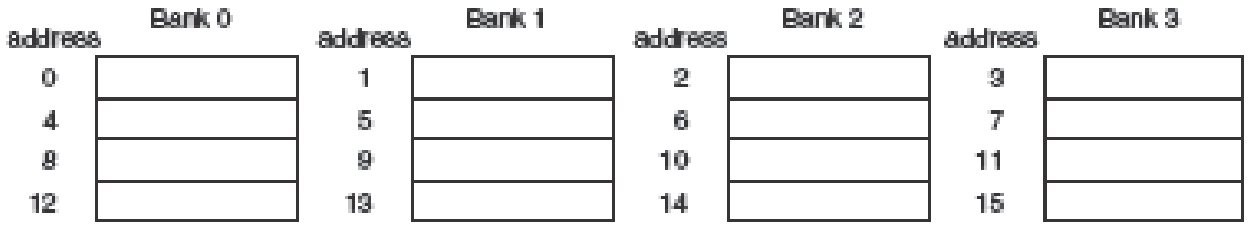
- **Inconvenientes:**
  - Hay coste en el bus más ancho. Se necesita un multiplexor entre caché y CPU ya que se accede a una palabra cada vez. Si la caché es más rápida que el bus se puede colocar el multiplexor entre la caché y el bus.
  - Inconveniente al expandir la memoria por el usuario ya que el incremento mínimo se duplica o cuadruplica.
  - Dificultades en la corrección de errores. Las memorias con corrección de errores tienes dificultades con las escrituras en una parte del bloque protegido. Muchos diseños de memoria más ancha separan la corrección de errores cada 32 bits, ya que la mayor parte de las escrituras tienen ese tamaño.

Memoria entrelazada

- Los chips de memoria se pueden organizar en bancos para leer o escribir múltiples palabras a la vez en lugar de una sola palabra.
- Los bancos son de una palabra de ancho para que la anchura del bus y de la caché no necesiten cambiar pero se envían direcciones a varios bancos lo que les permite a todos leer simultáneamente.
- Los bancos útiles también en las escrituras, permiten un ciclo de reloj para cada escritura, siempre que no estén destinadas al mismo banco. (Si hay muchas escrituras seguidas, normalmente tendrán que esperar que acaben las escrituras anteriores)

Memoria entrelazada

- La correspondencia entre direcciones y bancos afecta al comportamiento del sistema de memoria. Esta correspondencia se llama *factor de entrelazado*.
- El entrelazado optimiza los accesos secuenciales a la memoria
  - Un fallo de lectura de la caché es un caso ideal para una lectura entrelazada a nivel de palabra ya que las palabras se leen secuencialmente.
  - Las cachés de postescritura realizan lecturas y escrituras de forma secuencial.
- En la figura 4 bancos entrelazados a nivel de palabra.



Memoria entrelazada

- **Inconvenientes.**
  - Al aumentar la capacidad por chip de memoria habrá menos chips en un sistema de memoria del mismo tamaño, con lo que tener múltiples bancos será más caro.
  - La dificultad de expansión de la memoria principal. El hardware de control necesitará bancos del mismo tamaño, por lo que el incremento mínimo será probablemente duplicar la memoria.

Ejemplo 1

- Estudiemos estas organizaciones mostrando cómo se satisface un fallo de caché.
  - Supongamos el siguiente rendimiento para una organización básica de memoria:
    - 4 ciclo de reloj para enviar dirección
    - 24 ciclos de reloj para tiempo de acceso por palabra
    - 4 ciclos de reloj para enviar una palabra de datos
    - Bloque de caché de 4 palabras
    - Palabra de 32 bits
  - Para la organización de una palabra de ancho, la penalización de fallos es de $4x(4+24+4)=128$ ciclos de reloj por bloque, con un ancho de banda de memoria de 1/8 de byte (16/128) por ciclo de reloj.

Ejemplo 1

- Supongamos el siguiente rendimiento para una organización básica de memoria:
  - 4 ciclo de reloj para enviar dirección
  - 24 ciclos de reloj para tiempo de acceso por palabra
  - 4 ciclos de reloj para enviar una palabra de datos
  - Bloque de caché de 4 palabras
  - Palabra de 32 bits

- Para la organización de memoria ancha con un ancho de dos palabras, la penalización de fallos es de 2x(4+24+4)=64 ciclos de reloj por bloque con un ancho de banda de 1/4 byte/ciclo de reloj. Con un ancho de 4 palabras, la penalización de fallos será 1X32 ciclos de reloj por bloque con un ancho de banda de 1/2 byte/ciclo de reloj.

Ejemplo 1

- Supongamos el siguiente rendimiento para una organización básica de memoria:
  - 4 ciclo de reloj para enviar dirección
  - 24 ciclos de reloj para tiempo de acceso por palabra
  - 4 ciclos de reloj para enviar una palabra de datos
  - Bloque de caché de 4 palabras
  - Palabra de 32 bits

- Para la organización de memoria entrazada con cuatro bancos, enviando una dirección a los cuatro bancos a la vez, la penalización de fallos es de 4+24+4x4=44 ciclos de reloj por bloque con un ancho de banda de unos 0,4byte/ciclo de reloj.

Ejemplo 2

Considerar la siguiente descripción de una máquina y su rendimiento:

- Tamaño del bloque: 1 palabra
- Anchura del bus de memoria: 1 palabra
- Frecuencia de fallos: 3%
- Accesos a memoria por instrucción: 1.2
- Penalización de fallos de caché: 32 ciclos
- Ciclos medios por instrucción (ignorando los fallos de la caché): 2

Si cambiamos el tamaño de bloque a dos palabras, la frecuencia de fallos cae al 2%, y un bloque de cuatro palabras tiene una frecuencia de fallos del 1%. ¿Cuál es la mejora del rendimiento al entrelazar dos y cuatro vías frente a duplicar el ancho de memoria y del bus, suponiendo los tiempos de acceso del ejemplo anterior?

Solución ejemplo 2

- EL CPI para la máquina base utilizando bloques de una palabra es:

$$CPI = CPI_{ejec} + \frac{\text{Accesos a memoria } (NM)}{\text{Instrucción } (NI)} \cdot FF * PF$$

$$CPI = 2 + (1.2 \cdot 3\% \cdot 32) = 3.15$$

- Como la duración del ciclo de reloj y el recuento de instrucciones n puede cambiar en este ejemplo, podemos calcular la mejora de rendimiento comparando el CPI.
- Incrementar el tamaño de bloque a dos palabras:
  - Bus y memoria de 32 bits, sin entrelazado→ $CPI = 2 + [1.2 \cdot 10\% \cdot (2 \cdot 32)] = 3.54$
  - Bus y memoria de 32 bits, con entrelazado→ $CPI = 2 + [1.2 \cdot 2\% \cdot (4 + 24 + 8)] = 2.86$
  - Bus y memoria de 64 bits, sin entrelazado→ $CPI = 2 + [1.2 \cdot 2\% \cdot (1 \cdot 32)] = 2.77$
- Duplicar el tamaño del bloque ralentiza la implementación, mientras que el entrelazado o la memoria ancha lo hace más rápido, 1.10 y 1.14 veces más rápido respectivamente.

Solución ejemplo 2

- EL CPI para la máquina base utilizando bloques de una palabra es:

$$CPI = CPI_{ejec} + \frac{\text{Accesos a memoria } (NM)}{\text{Instrucción } (NI)} \cdot FF * PF$$

$$CPI = 2 + (1.2 \cdot 3\% \cdot 32) = 3.15$$

- Incrementar el tamaño de bloque a 4 palabras:
  - Bus y memoria de 32 bits, sin entrelazado→ $CPI = 2 + [1.2 \cdot 1\% \cdot (4 \cdot 32)] = 3.54$
  - Bus y memoria de 32 bits, con entrelazado→ $CPI = 2 + [1.2 \cdot 1\% \cdot (4 + 24 + 16)] = 2.53$
  - Bus y memoria de 64 bits, sin entrelazado→ $CPI = 2 + [1.2 \cdot 1\% \cdot (2 \cdot 32)] = 2.77$

- De nuevo, un tamaño del bloque más grande disminuye el rendimiento del caso simple, aunque el el entrazado de 32 bits es ahora el más rápido, 1.25 veces más rápido frente a 1.14 veces más rápido de la memoria y bus más ancho.

Bancos de memoria independientes

- Una generalización del entrelazado es permitir múltiples accesos independientes, donde múltiples controladores de memoria permitan a los bancos operar independientemente.
- Cada banco necesita líneas de dirección separadas y posiblemente un bus de datos separado.
- Por ejemplo, un dispositivo de entrada puede utilizar un controlador y un banco, la lectura en caché puede utilizar otro banco y la escritura en caché un tercer banco.
- Se utiliza el término *superbanco* para referirse a toda la memoria activa en una transferencia de un bloque y el término banco para referirse a la porción dentro de un superbanco que está entrelazada a nivel de palabra.

Evitar conflictos en bancos de memoria

- Si el sistema de memoria se diseña para soportar múltiples peticiones independientes, la efectividad del sistema va a depender de la frecuencia con que las peticiones independientes vayan a distintos bancos.
- Para evitar problemas de conflicto al acceder al mismo banco hay dos posibles soluciones:
  - Solución Software: El compilador puede realizar optimizaciones. Puede por ejemplo expandir el tamaño de los arrays para que no sean potencias de 2, forzando a que las direcciones pertenezcan a distintos bancos.
  - Solución Hardware: Una solución es tener un número primo de bancos. Esto puede complicar el cálculo de la dirección del módulo pero hay muchas técnicas rápidas para calcular el módulo.

5.5. Memoria Virtual

Tema 5 Rendimiento de la jerarquía de memoria

Arquitectura de los Computadores

5.5. MEMORIA VIRTUAL

5.3.1 Organización de la caché

5.3.2 Ubicación de bloque

5.3.3 Estrategias de reemplazo

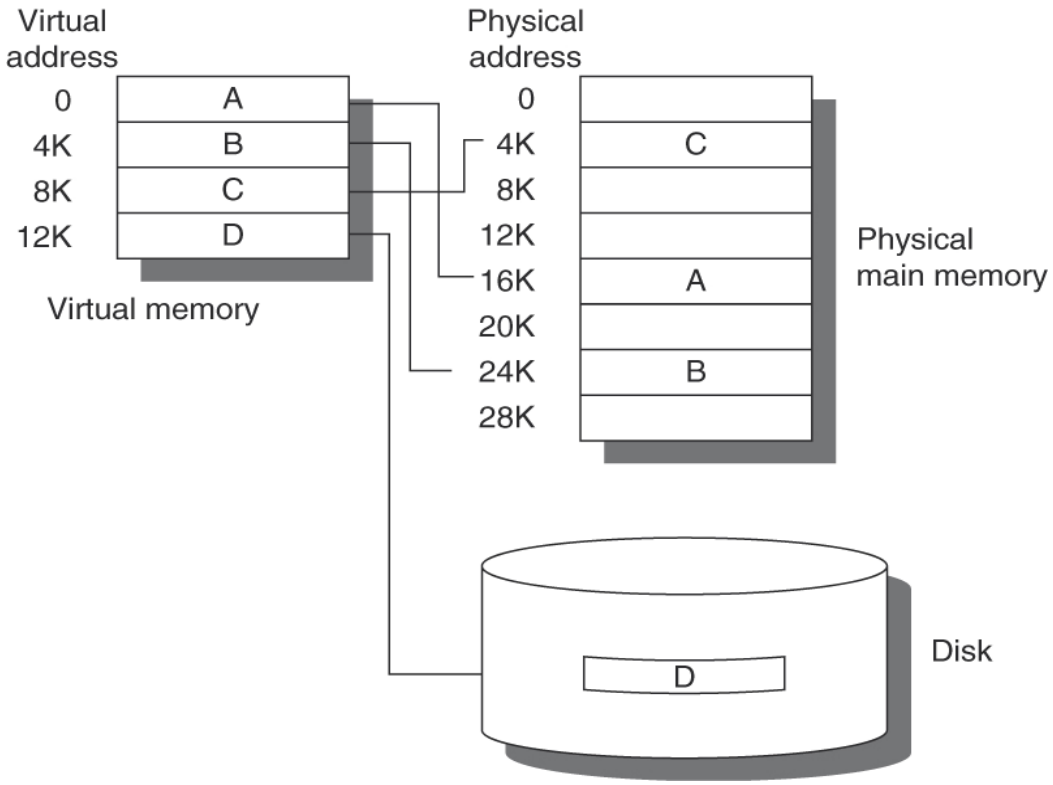
5.3.4 Operación de la caché

5.3.5 Políticas de escritura

5.3.6 Rendimiento

Introducción

- La memoria virtual gestiona automáticamente los niveles de la jerarquía de memoria representada por la memoria principal y la secundaria.



Conceptos generales

- La memoria virtual permite:
  - En un instante de tiempo los computadores estén corriendo múltiples procesos, cada un de ellos con su propio espacio de direcciones.
  - La existencia de un esquema de protección de la memoria para cada proceso.
  - Las necesidades de memoria de todos los programas de un sistema sean superiores a la memoria física disponible.
  - Compartir una cantidad de memoria física pequeña entre muchos procesos.
  - Simplifica la carga del programa para su ejecución (reubicación).

Conceptos generales

- La memoria virtual divide la memoria física en bloques (**páginas**) y los asigna a diferentes procesos.
- Cada proceso tiene un espacio virtual propio dividido en páginas.
- Las páginas pueden estar en memoria principal o en disco.
- La CPU genera **direcciones virtuales**
- Un combinación de hardware y software las traduce a direcciones físicas.
- Cuando una página no está en memoria principal se produce una **fallo** de página (o segmento) y debe traerse desde disco.
- El reemplazo de páginas está controlado principalmente por el sistema operativo.

Conceptos generales

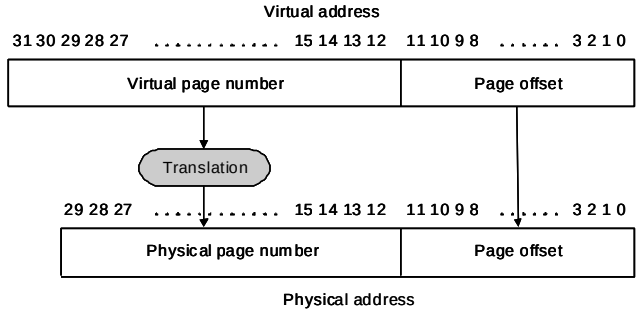
- Dos clases de memoria virtual:
  - Bloques de tamaño fijo denominados **páginas**. Típicamente entre 4096 bytes y 8192 bytes.
  - Bloques de tamaño variable denominados **segmentos**. El segmento mayor varía entre 2^{16} bytes hasta 2^{32} bytes; el más pequeño e de 1 byte.
- Debido a que el reemplazo es difícil pocas máquinas utilizan la segmentación pura.
- Algunas máquinas utilizan un enfoque híbrido llamado de segmentos paginados.

¿Dónde puede ubicarse un bloque en MP?

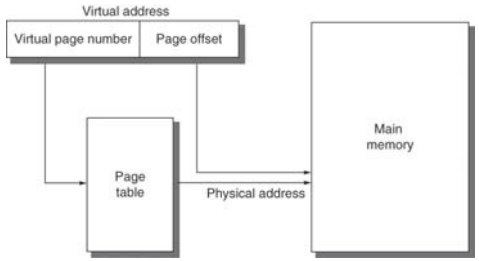
- La penalización de fallos es bastante alta puesto que acceden a disco.
- Es muy importante reducir los fallos de página.
- Los SO permiten la asignación completamente asociativa.

¿Cómo se encuentra un bloque si está en MP?

- La dirección virtual se descompone en dos campos: número de página virtual y desplazamiento de página.
- La dirección física se obtiene concatenando la dirección física de la página con el desplazamiento de página.



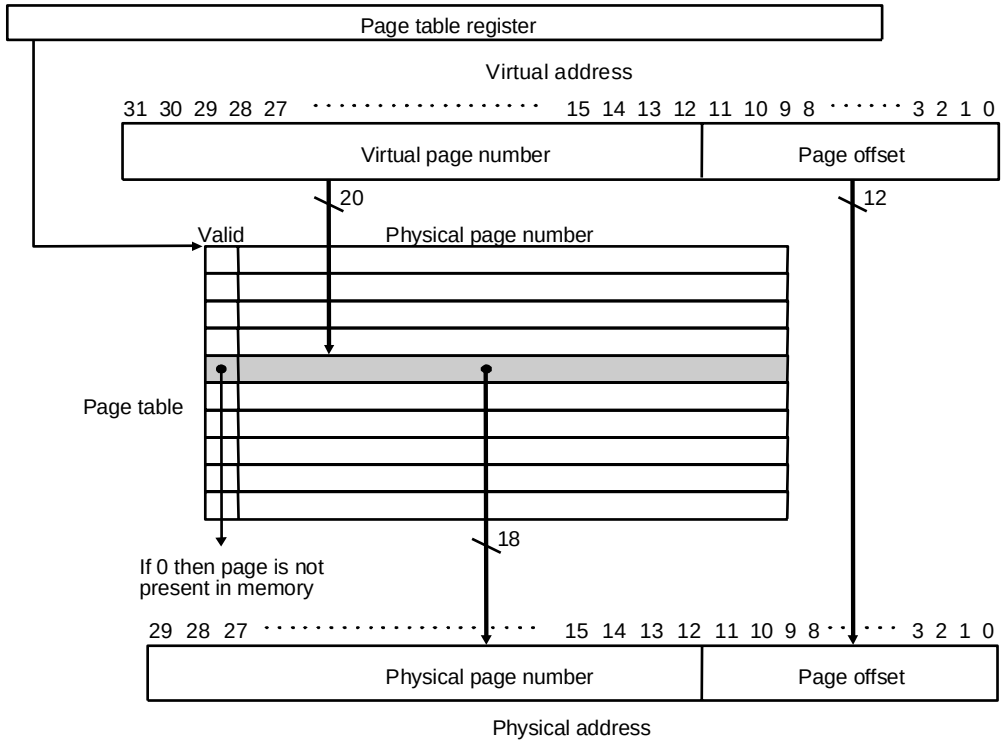
- Se utiliza una estructura de datos (tabla de páginas) que contiene la dirección física de la página y es indexada por la dirección virtual.



- La tabla de páginas reside en memoria principal.

Tabla de páginas

- Correspondencia entre dirección virtual y física



¿Qué bloque debe sustituirse en un fallo de memoria virtual?

- Se pretende minimizar los fallos de página.
- Casi todos los SO intentan sustituir el bloque LRU (menos recientemente utilizado)
- Muchas máquinas proporcionan un bit de uso o referencia para este propósito.

¿Qué ocurre en una escritura?

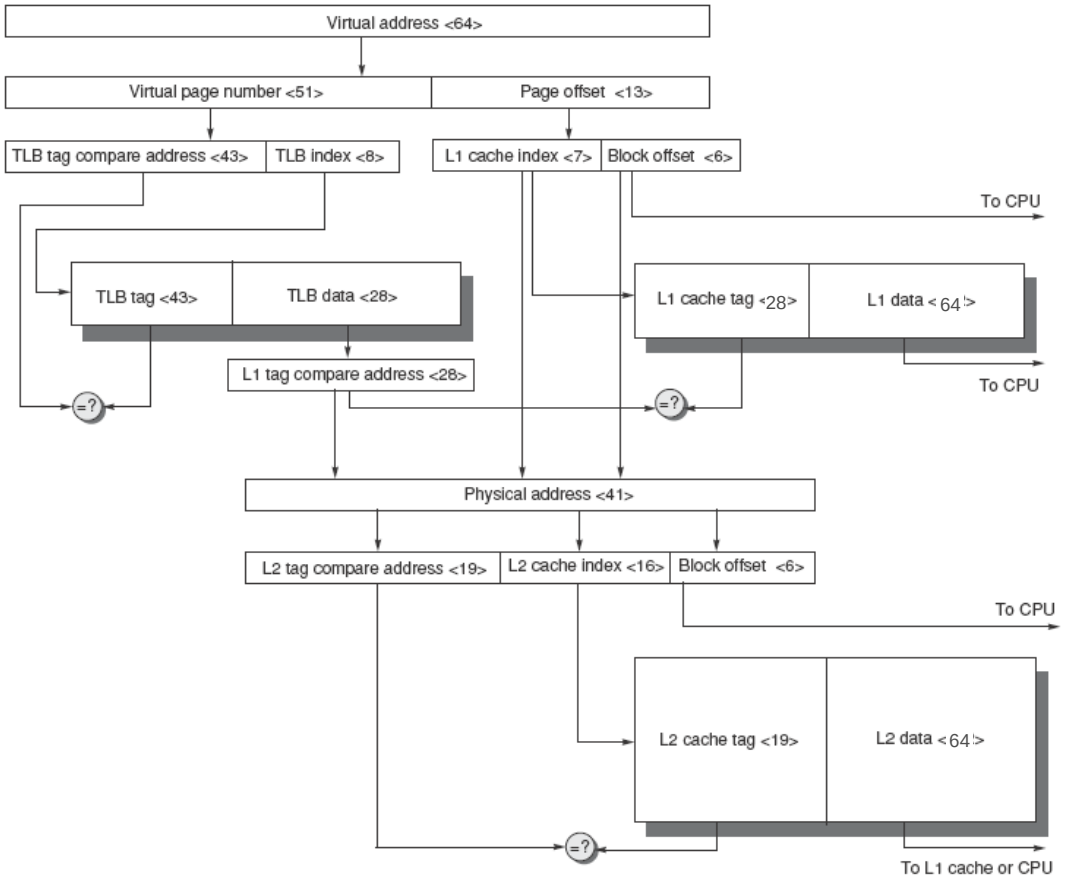
- Usar escritura directa (write-through) es demasiado caro por tanto se usará postescritura (writeback).
- Los sistemas de memoria virtual incluyen un bit de modificación (dirty).

Traducción rápida de direcciones

- Disponer de una caché para la traducción de direcciones: translation lookaside buffer (TLB)
- Una entrada a la TLB es como una entrada de la caché donde la etiqueta contiene parte de la dirección virtual y la parte del dato contiene un número de estructura de página, campo de protección, bit de uso y bit de modificación.
- La dirección virtual debe ir primero a la TLB antes que la dirección física pueda acceder la caché. Lo que significa que el tiempo de acierto se alarga.
- Se puede reducir el tiempo de acierto al acceder a la caché con el desplazamiento de página. La comparación de direcciones se realiza entre la dirección física del TLB y la etiqueta de la caché.
  - El inconveniente es que una caché con correspondencia directa no puede ser mayor de una página.

Ejemplo de jerarquía de memoria

- Tamaño página: 8KB
- TLB: correspondencia directa con 256 entradas
- Caché L1 y L2: correspondencia directa de 8KB y 4MB respectiv.
- Dirección virtual: 64 bits Dirección física: 41 bits



Ejemplos: Pentium Pro y PowerPC 604

Características	Pentium Pro	PowerPc 604
Dirección virtual	32 bits	52 bits
Dirección física	32 bits	32 bits
Tamaño de página	4KB, 4MB	4KB, 256MB
Organización TLB	TLB para datos TLB para instrucciones Asociativas de 4 vías Reemplazo pseudo LRU TLB-i 32 entradas TLB-d 64 entradas Fallos TLB tratados por Hardware	TLB para datos TLB para instrucciones Asociativas de 2 vías Reemplazo LRU TLB-i 128 entradas TLB-d 128 entradas Fallos TLB tratados por Hardware
Organización caché	Cachés de datos e instrucciones separadas	Caches de datos e instrucciones separadas
Tamaño caché	8KB cada una	16KB cada una
Asociatividad	Asociativa de 4 vías	Asociativas de 4 vías
Reemplazo	LRU aproximado	LRU
Tamaño bloque	32 byte	32 byte
Actualización	Aplazada	Aplazada o inmediata