PRINCIPIOS GENERALES DE JERARQUÍA DE MEMORIA

REGULARIDADES EN LOS ACCESOS A MEMORIA

PRINCIPIO DE LOCALIDAD

- ESPACIAL: Si se referencia un elemento, los elementos cercanos a él se volverán a referenciar pronto.
- TEMPORAL: Si se referencia un elemento, probablemente se volverá a referenciar pronto.

CAUSAS:

- A menudo datos e instrucciones se almacenan en memoria en el mismo orden en que se necesitan.
- Bucles de los programas.

PRINCIPIO ESENCIAL DE JERARQUÍA DE MEMORIA

- Copiar bloques con los elementos que probablemente se referenciarán próximamente en una memoria más rápida.
- VENTAJA: Dado el principio de localidad, en una jerarquía de memoria la mayoría de los accesos se harán sobre la memoria más rápida.

FRECUENCIA LECTURAS > FRECUENCIA ESCRITURAS

CAUSAS:

- Todas las instrucciones hay que leerlas.
- Muchas operaciones leen varios operandos y escriben un único resultado.

CONSECUENCIA: Hay que hacer más esfuerzos por mejorar la velocidad de las lecturas.

TERMINOLOGÍA Y CONCEPTOS BÁSICOS

- Niveles:
  - Superior: El más cercano al procesador (más pequeño y rápido).
  - Inferior.

- **Bloque**: Cantidad de información que se transfiere entre dos niveles de memoria

- Acierto (hit) o fallo (miss/fault) en un acceso a la memoria del nivel superior.

- Frecuencia de aciertos (F_a): Nº aciertos/Nº accesos.

- Frecuencia de fallos (F_f): Nº fallos/Nº accesos.

- **Tiempo de acierto (T_a)**: Tiempo para acceder al nivel superior (incluido el tiempo para saber si el acceso es acertado o fallido).

- **Penalización de fallo (P_f)**: Es la suma de:
  - **Tiempo acceso a primera palabra del bloque** en un fallo
  - **Tiempo de transferencia** (tiempo necesario para transferir las restantes palabras del bloque)

- Tiempo de fallo: $T_f = T_a + P_f$.

MEDIDA DEL RENDIMIENTO

Tiempo medio de acceso a memoria =

$$T_a \cdot F_a + T_f \cdot F_f =$$

$$= T_a + F_f \cdot P_f$$

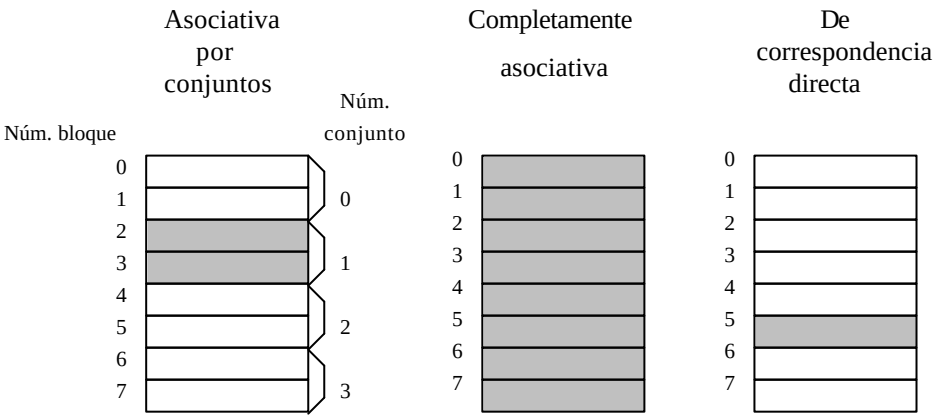
ASPECTOS ESENCIALES EN EL DISEÑO DE UNA JERARQUÍA DE MEMORIA

- Política de emplazamiento de bloques: Correspondencia entre los bloques de los niveles inferior y superior.
- Modo de identificar un bloque en el nivel superior.
- Política de reemplazo de bloques.
- Estrategia de escritura
- Tamaño y naturaleza de los bloques

POLÍTICAS DE EMPLAZAMIENTO

Número de bloque	Desplazamiento

Partes de una dirección de memoria principal.



Ubicación en la caché del bloque número 13 de memoria principal, según la política de emplazamiento.

Asociatividad por conjuntos **de N vías**: N bloques en cada conjunto.

De correspondencia directa: Asociatividad 1.

Completamente asociativa de m bloques: Asociatividad m.

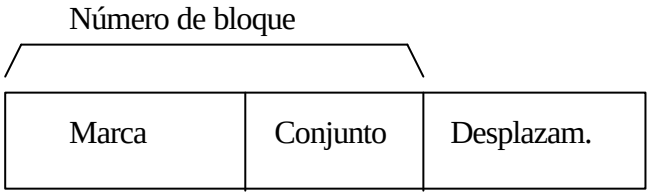
IDENTIFICACIÓN DE BLOQUES

- Problema:

  Un bloque de caché $\longrightarrow$ Varios de mem. principal

- Solución: Colocar en cada bloque de la caché una **marca** que identifique el bloque de memoria principal correspondiente.

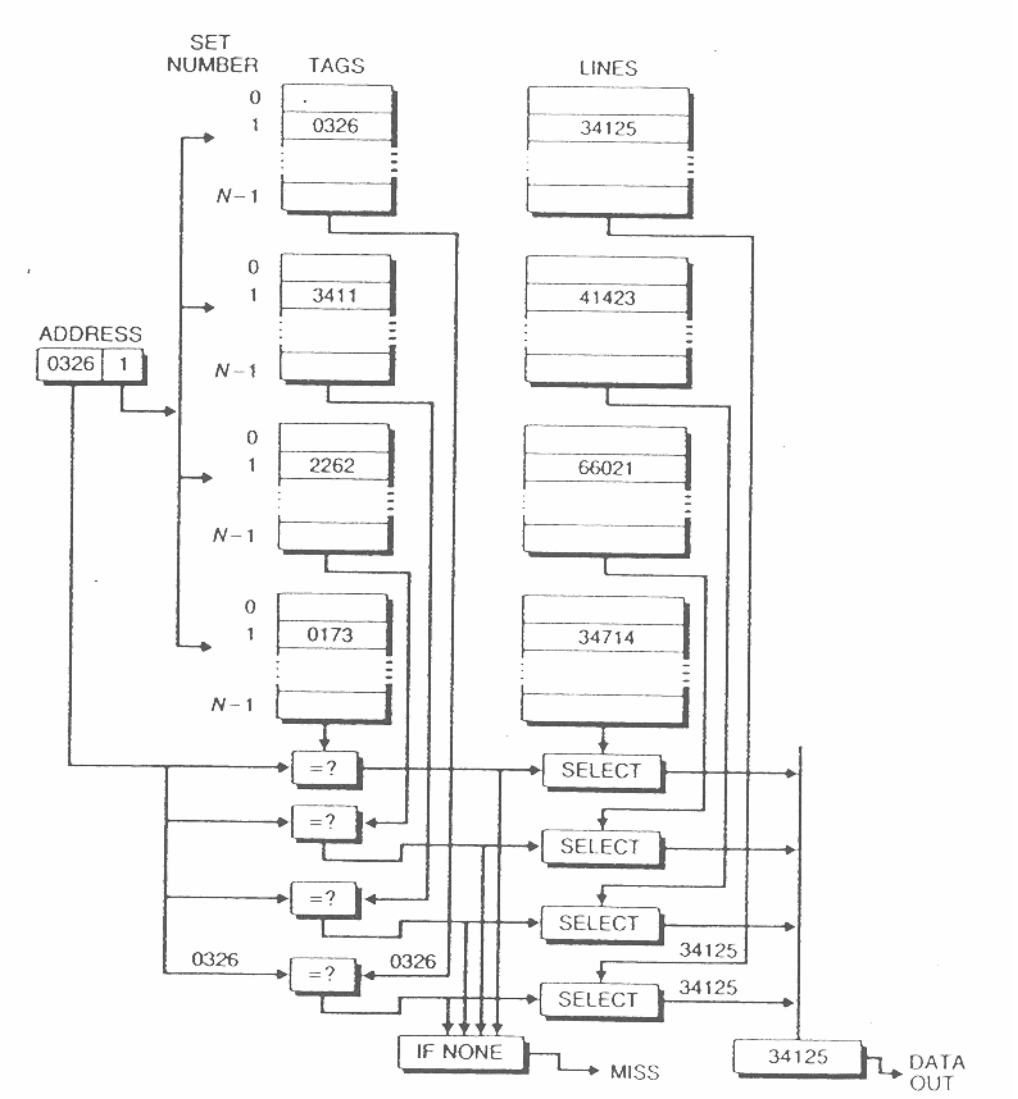
- A esa marca se añade un **bit de validez** para indicar si la información del bloque es válida (=1) o no (=0).



Partes de una dirección de memoria principal en una caché asociativa por conjuntos

Eficiencia $\Rightarrow$ Comparación de la marca en paralelo con todas las etiquetas del conjunto.

Mayor asociatividad $\Rightarrow$ Más bits para la marca y menos para el conjunto.



Implementación de una caché asociativa por conjuntos, de 4 vías.

POLÍTICA DE REEMPLAZO

- Gran frecuencia aciertos ⇒ Caché tiene pocos bloques *vacíos*.

- Política de reemplazo. Más importante en las cachés pequeñas.

- Caché de correspondencia directa: Obvio. Hardw. sencillo.

- Caché con asociatividad >1. Estratégias básicas:

    - **Aleatoria**. El bloque a sustituir se selecciona al azar:

        - Ubicación uniforme de bloques dentro de un conjunto.

        - Implementación sencilla.

    - Menos recientemente usado (**LRU**).

        - Registrar accesos a los bloques.

        - Los más recientemente usados tienen más probabilidades de volver a ser usados (localidad temporal).

        - Eficaz.

        - Implementación costosa si el número de bloques es grande.

        - A menudo se usa una aproximación a la estrategia LRU que sea menos costosa.

    - Primero en entrar, primero en salir (FIFO). Menos usada. Peor rendimiento que la aleatoria.

POLÍTICA DE ACTUALIZACIÓN DE LA MEMORIA PRINCIPAL

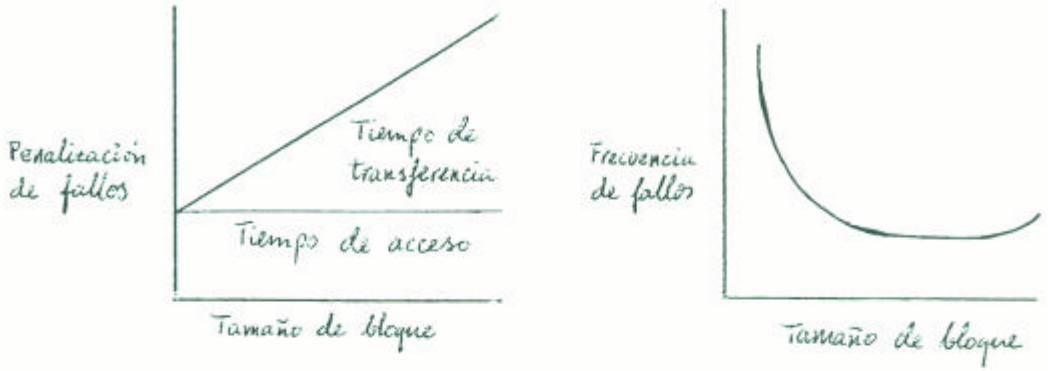
- Postescritura (*write back* o *copy back*)
    - Sólo se escribe en la caché.
    - Bit de modificación o ensuciado (*dirty bit*):
        = 1 si bloque modificado
        = 0 si bloque limpio
    - Al reemplazar el bloque modificado, se escribe en memoria principal.

- Escritura directa (*write through*).
    - Se escribe en la caché y en memoria principal.

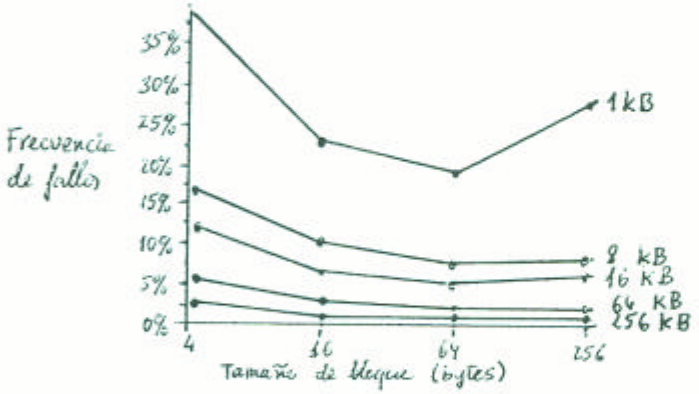
- Comparación:
        - Más fácil de implementar: Escritura directa.
        - Memoria principal siempre actualizada: Escritura directa.
        - Menos accesos a memoria principal: Postescritura.
        - Más velocidad: Postescritura.

- Fallos de escritura. Estrategias:
    - Ubicar en escritura.
        - El bloque se carga y después se realizan las acciones de escritura.
        - Suele usarse con postescritura.
    - No ubicar en escritura.
        - Se escribe en memoria principal.
        - Suele usarse con escritura directa.

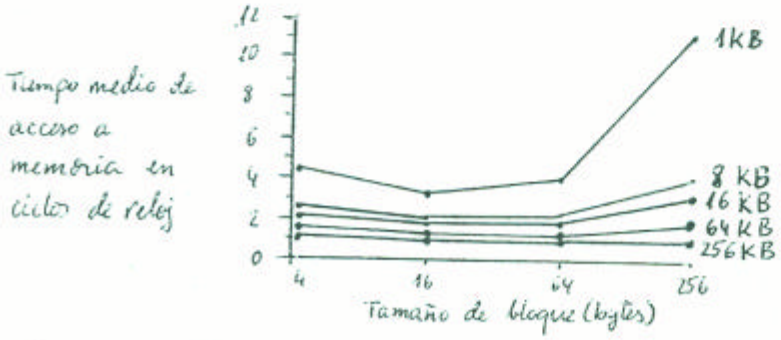
TAMAÑO DE LOS BLOQUES



Tamaño de bloque frente a frecuencia de fallos y penalización de fallos.



Frecuencia de fallos frente a tamaño de bloque para disitntos tamaños de cache.



Tiempo medio de acceso frente a tamaño de bloque para disitntos tamaños de cache (según las frecuencias de fallos de la figura anterior)

NATURALEZA DE LOS BLOQUES

Cachés:

- De datos.
- De instrucciones.
- Unificadas.

Ventajas de cachés separadas para datos e instrucciones:

- Mejora el ancho de banda entre la caché y la UCP.
- Se puede optimizar cada caché separadamente.
- Con UCP encauzada, se evitan las colisiones entre la etapa FETCH y las etapas que acceden a datos de memoria.

Inconvenientes de cachés separadas para datos e instrucciones:

- El tamaño dedicado a datos e instrucciones es inamovible.

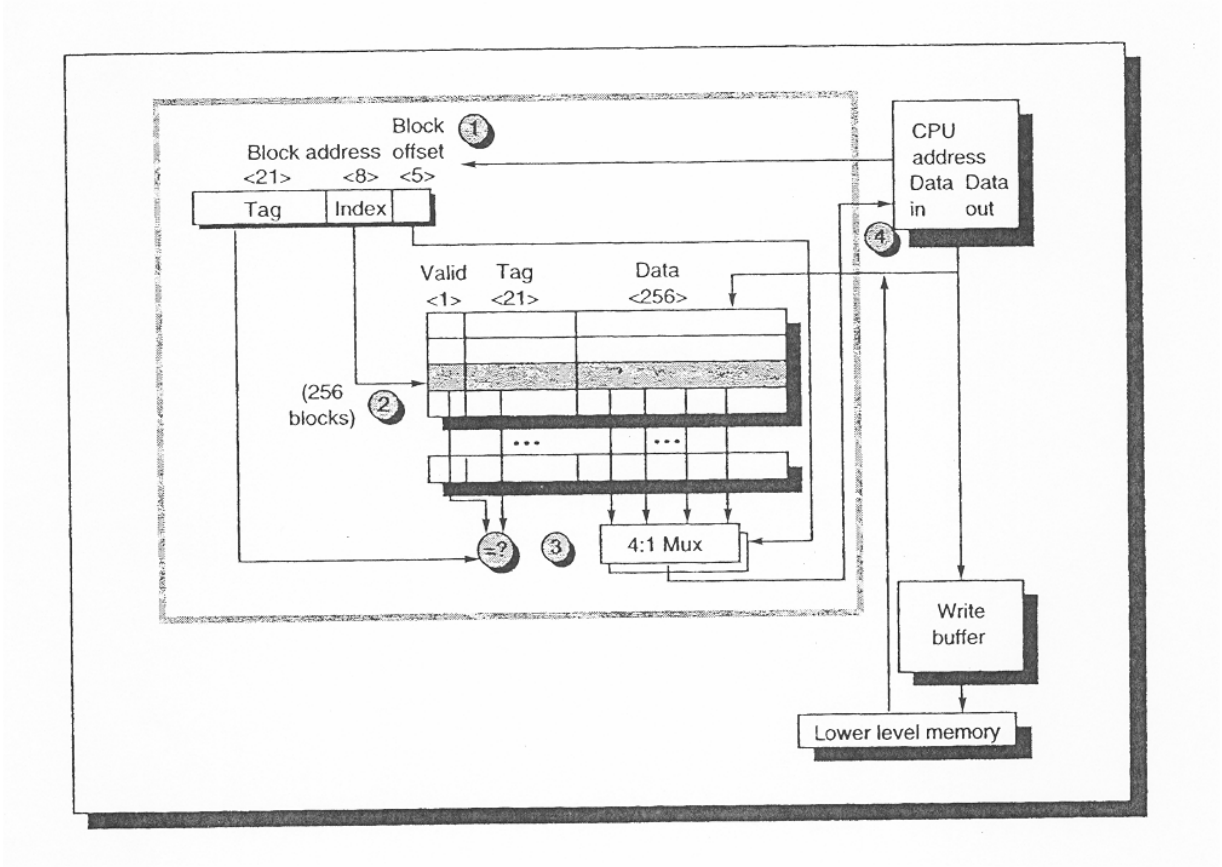
Tamaño	Caché de instrucciones	Caché de datos	Caché unificada
8KB	1,10%	10,19%	4,57%
16KB	0,64%	6,47%	2,87%
32KB	0,39%	4,82%	1,99%
64KB	0,15%	3,77%	1,35%
128KB	0,02%	2,88%	0,95%

Frecuencia de fallos para cachés de distinta naturaleza. Con los programas de prueba utilizados el porcentaje de referencias a instrucciones está en torno al 75%.

Comparación de la frecuencia de fallos de dos cachés separadas de 8KB con una unificada de 16KB, si el porcentaje de referencias a instrucciones es del 75%:

$0{,}75 \cdot 1{,}10 + 0{,}25 \cdot 10{,}19 = 3{,}37\% > 2{,}87\%$

ORGANIZACIÓN DE LA CACHÉ DE DATOS DEL ALPHA AXP 21064



CACHE Y RENDIMIENTO DE LA UCP

- Los ciclos dedicados a la ejecución de un programa, CE, podemos desglosarlos en

  CEM: ciclos de reloj de espera a la memoria.

  CEJ: demás ciclos utilizados por la UCP durante la ejecución.

  $$CE = CEJ+CEM$$

  Por tanto, siendo IE el número de instrucciones ejecutadas, el CPI es

  $$CPI = CE/IE = (CEJ+CEM)/IE$$

- La mayoría de los ciclos de espera a la memoria se deben a los fallos de cache. Despreciando los demás

  $$CEM = N_f \cdot P_f = Na \cdot F_f \cdot P_f$$

  donde N_f: número de fallos

  P_f: penalización de fallos (**¡en ciclos!**)

  N_a: número de accesos

  F_f: frecuencia de fallos => $F_f = N_f/N_a$ => $N_f = Na \cdot F_f$

- Si tomamos**:**

  NAI = N_a/IE, promedio de accesos a memoria por instrucción

  CEPI = CEJ/IE, promedio de ciclos de ejecución por instrucción

  obtenemos

  $$\mathbf{CPI} = (CEJ+CEM)/IE = \mathbf{CEPI + NAI \cdot F_f \cdot P_f}$$

  y, siendo FPI el promedio de fallos por instrucción (N_f/IE),

  $$\mathbf{CPI = CEPI + FPI \cdot P_f}$$

- Finalmente, el TUCP será:

  $$\mathbf{TUCP = IE \cdot (CEPI + FPI \cdot P_f) \cdot T = IE \cdot (CEPI + NAI \cdot F_f \cdot P_f) \cdot T}$$

- **Conclusiones**:
  - Un tiempo medio de acceso a memoria menor puede corresponder a un TUCP mayor (si T es mayor)
  - Los fallos de la cache tienen mayor impacto
    - Cuanto menor CEPI
    - Cuanto menor T. (En ciclos de reloj P_f será mayor)

FUENTES DE FALLOS DE LA CACHÉ

- Fallos **forzosos** o de primera referencia a un bloque (de memoria principal).

- Fallos **de capacidad**. Hay bloques que estarían en la caché si fuese más grande.

- Fallos **de conflicto**. Habría acierto si el bloque no se hubiera reemplazado por falta de suficiente asociativiadad.

- Este modelo:

  - No permite distinguir siempre si un fallo individual es de capacidad o de conflicto:
    podría desaparecer tanto aumentando la asociatividad como aumentando la capacidad.

  - Sí permite dar un número total de fallos de cada tipo:

    - El número de fallos forzosos es independiente de la cache.

    - Los fallos de capacidad son todos los fallos no forzosos al simular la cache en forma completamente asociativa.

    - El resto de fallos, son los de conflicto.

- ¿Cómo evitarlos?

    - Conflicto. Aumentar asociatividad. Pero esto:

        - Encarece el hardware
        - Alarga la duración del ciclo de reloj

    - Capacidad. Aumentar la caché. Pero:

        - Encarece el hardware
        - Aumenta el tiempo de acierto

    - Forzosos. Aumentar tamaño de bloque. Pero esto:

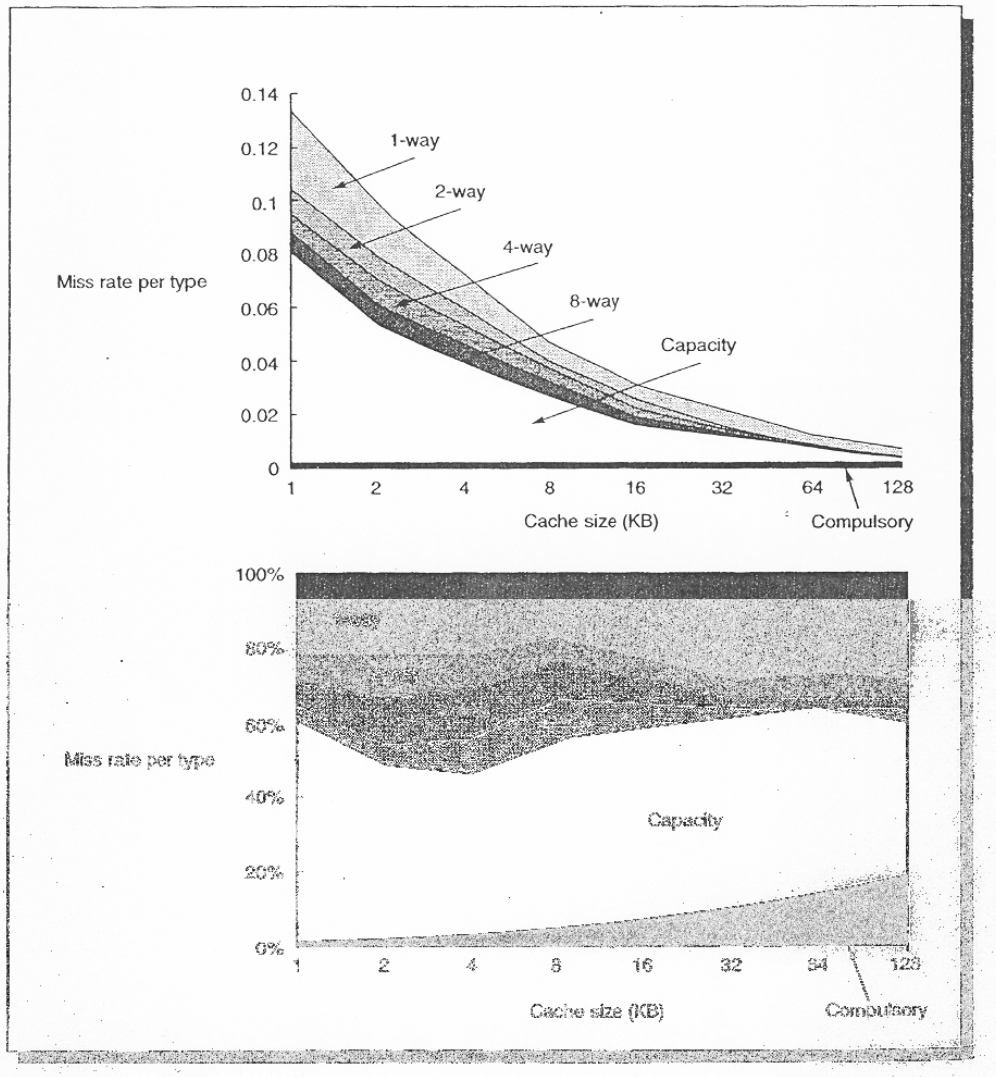
        - Aumenta los fallos de conflicto.
        - Aumenta la penalización de fallos.

TIPOS DE FALLOS DE CACHÉ

Cache size	Degree associative	Total miss rate	Miss rate components (relative percent) (*Sum = 100% of total miss rate*)					
			Compulsory		Capacity		Conflict	
1 KB	1-way	0.133	0.002	1%	0.080	60%	0.052	39%
1 KB	2-way	0.105	0.002	2%	0.080	76%	0.023	22%
1 KB	4-way	0.095	0.002	2%	0.080	84%	0.013	14%
1 KB	8-way	0.087	0.002	2%	0.080	92%	0.005	6%
2 KB	1-way	0.098	0.002	2%	0.044	45%	0.052	53%
2 KB	2-way	0.076	0.002	2%	0.044	58%	0.030	39%
2 KB	4-way	0.064	0.002	3%	0.044	69%	0.018	28%
2 KB	8-way	0.054	0.002	4%	0.044	82%	0.008	14%
4 KB	1-way	0.072	0.002	3%	0.031	43%	0.039	54%
4 KB	2-way	0.057	0.002	3%	0.031	55%	0.024	42%
4 KB	4-way	0.049	0.002	4%	0.031	64%	0.016	32%
4 KB	8-way	0.039	0.002	5%	0.031	80%	0.006	15%
8 KB	1-way	0.046	0.002	4%	0.023	51%	0.021	45%
8 KB	2-way	0.038	0.002	5%	0.023	61%	0.013	34%
8 KB	4-way	0.035	0.002	5%	0.023	66%	0.010	28%
8 KB	8-way	0.029	0.002	6%	0.023	79%	0.004	15%
16 KB	1-way	0.029	0.002	7%	0.015	52%	0.012	42%
16 KB	2-way	0.022	0.002	9%	0.015	68%	0.005	23%
16 KB	4-way	0.020	0.002	10%	0.015	74%	0.003	17%
16 KB	8-way	0.018	0.002	10%	0.015	80%	0.002	9%
32 KB	1-way	0.020	0.002	10%	0.010	52%	0.008	38%
32 KB	2-way	0.014	0.002	14%	0.010	74%	0.002	12%
32 KB	4-way	0.013	0.002	15%	0.010	79%	0.001	6%
32 KB	8-way	0.013	0.002	15%	0.010	81%	0.001	4%
64 KB	1-way	0.014	0.002	14%	0.007	50%	0.005	36%
64 KB	2-way	0.010	0.002	20%	0.007	70%	0.001	10%
64 KB	4-way	0.009	0.002	21%	0.007	75%	0.000	3%
64 KB	8-way	0.009	0.002	22%	0.007	78%	0.000	0%
128 KB	1-way	0.010	0.002	20%	0.004	40%	0.004	40%
128 KB	2-way	0.007	0.002	29%	0.004	58%	0.001	14%
128 KB	4-way	0.006	0.002	31%	0.004	61%	0.001	8%
128 KB	8-way	0.006	0.002	31%	0.004	62%	0.000	7%

Total miss rate for each size cache and percentage of each according to the "three C's." Compulsory misses are independent of cache size, while capacity misses decrease as capacity increases. Gee et al. [1993] calculated the average miss rate for the SPEC92 benchmark suite with 32-byte blocks and LRU replacement on a DECstation 5000. Figure 5.10 shows the same information graphically. The compulsory rate was calculated as the miss rate of a fully associative 1-MB cache. Note that the 2:1 cache rule of thumb (inside front cover) is supported by the statistics in this table: a direct-mapped cache of size *N* has about the same miss rate as a 2-way set-associative cache of size *N*/2.

TIPOS DE FALLOS DE CACHÉ (cont.)

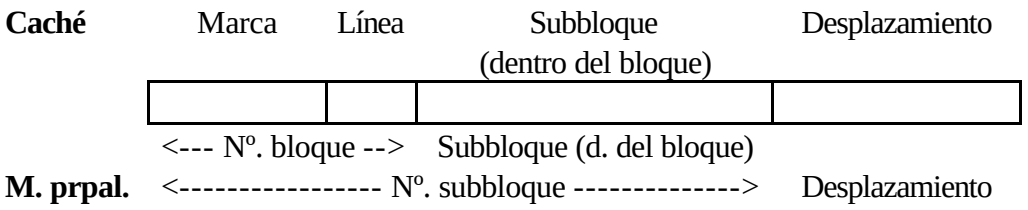


Total miss rate (top) and distribution of miss rate (bottom) for each size cache according to three C's for the data in Figure 5.9. The top diagram is the actual miss rates, while the bottom diagram is scaled to the direct-mapped miss ratios.

CACHÉS CON SUBBLOQUES

- Válida sólo para cachés de correspondencia directa.
- El subbloque es ahora la unidad de transferencia de información.
- Cada bloque tiene una marca. Cada subbloque, un bit de validez.
- Ventajas:
  - Menor penalización de fallos (solo hay que copiar un subbloque) que cachés sin subbloques con bloques de igual tamaño.
  - Menor memoria de marcas (principal objetivo de las cachés con subbloques) que cachés sin subbloques con bloques del tamaño de los subbloques.

- Partes de una dirección:



Etiqueta	V		V		V		V	
100	1		1		1		1	
300	1		1		0		0	
200	0		1		0		1	
204	0		0		0		0	

En este ejemplo hay cuatro subbloques por bloque. En el bloque superior (línea 0) todos los bits de validez están a 1, lo que equivale a que el bit de validez esté a 1 para un bloque en una caché normal. En la línea 3 ocurre lo contrario; ningún bit de validez está a 1. En la línea 1, los subbloques 0 y 1 son válidos y serán aciertos, mientras que los subbloques 2 y 3 serán fallos.

Si en lugar de esta organización, hubiera 16 bloques, se necesitarían 16 etiquetas en lugar de 4.

CACHÉS DE DOS NIVELES

- Mayores memorias principales → Cachés mayores para disminuir la frecuencia de fallos.

- CPU's más rápidas → Cachés pequeñas y rápidas.

Compromiso: Caché de dos niveles:

- Nivel 1, caché pequeña y rápida.

- Nivel 2, caché grande.

Tiempo medio de acceso a memoria =

$$= T_{a1} + F_{f1} \cdot P_{f1}$$

donde

$$P_{f1} = T_{a2} + F_{f2} \cdot P_{f2}$$

quedando

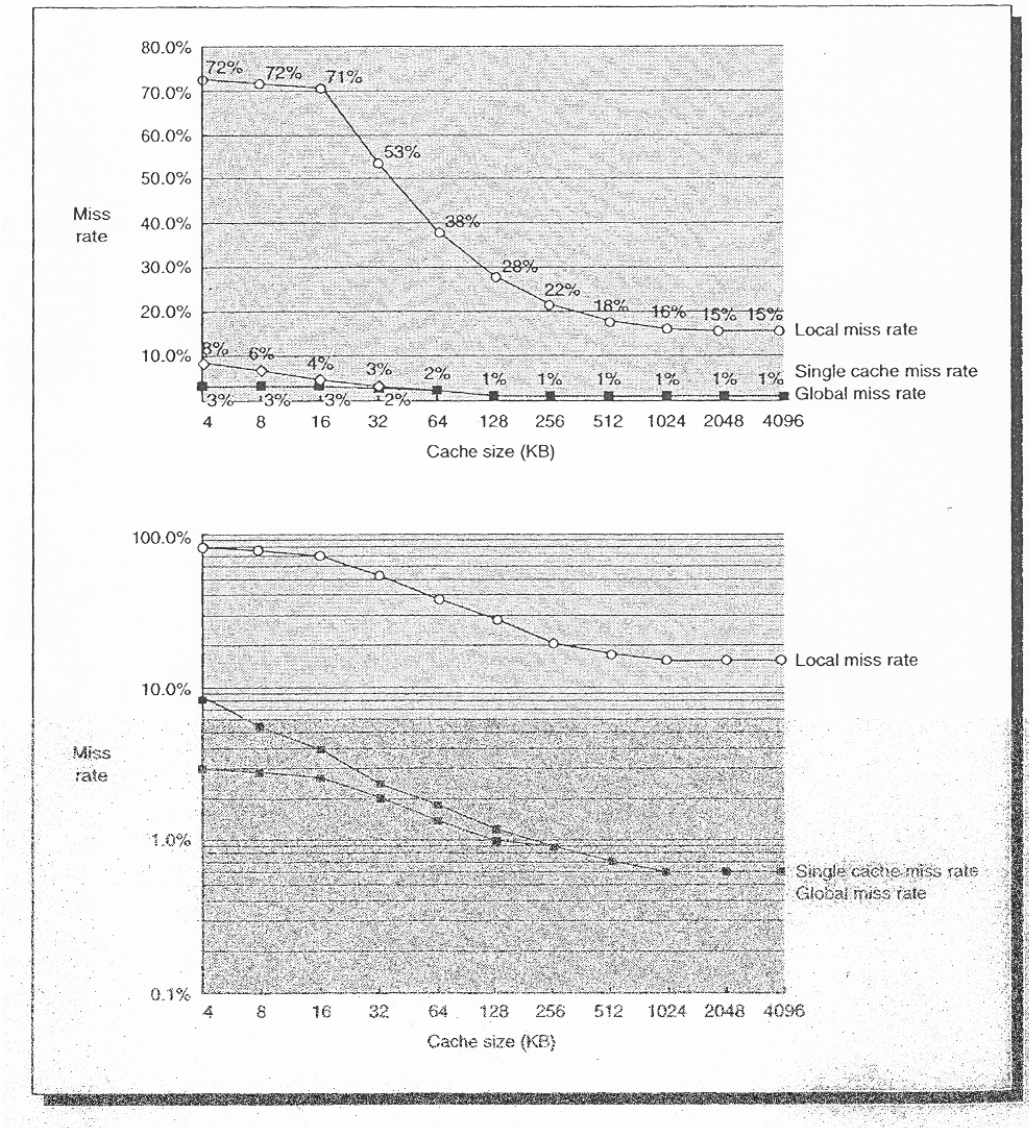
$$T_{a1} + F_{f1} \cdot T_{a2} + F_{f1} \cdot F_{f2} \cdot P_{f2}$$

- Habitualmente:

- Caché nivel 1 $\subset$ caché nivel 2.

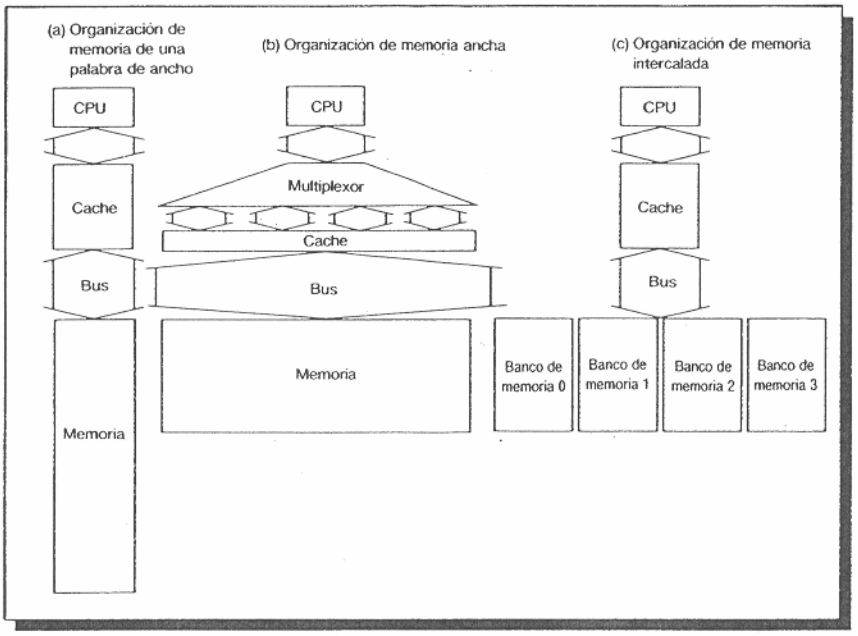
- Caché niveles 1 y 2 integradas en el microprocesador.

CACHÉS DE SEGUNDO NIVEL: FRECUENCIAS DE FALLOS LOCAL Y GLOBAL FRENTE A TAMAÑO DE CACHÉ

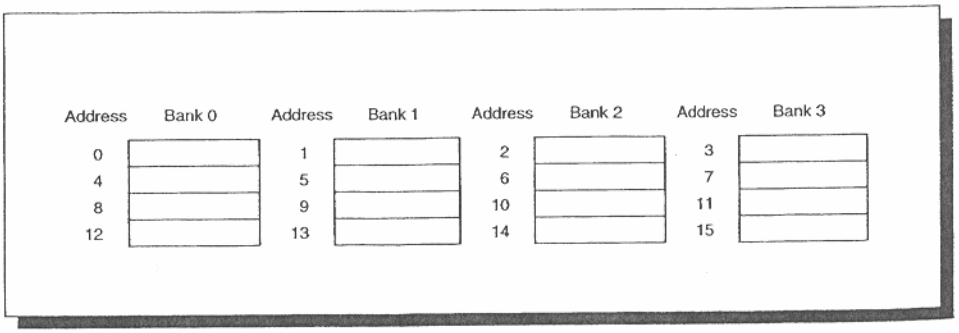


Miss rates versus cache size for reads and writes. The top graph shows the results plotted on a linear scale as we have done with earlier figures, while the bottom graph shows the results plotted on a log scale. As miss rates shrink, the log scale makes the differences easier to follow. The miss rate of a single-level cache versus size is plotted against the local miss rate and global miss rate of a second-level cache using a 32-KB first-level cache. Second-level caches *smaller* than the 32-KB first-level make little sense, as reflected in the high miss rates. After 256 KB the single cache and global miss rates are virtually identical. Przybylski [1990] used four traces from the VAX system and four user programs from the MIPS R2000 that were randomly interleaved to duplicate the effect of process switches.

ORGANIZACIÓN DE MEMORIA Y ANCHO DE BANDA

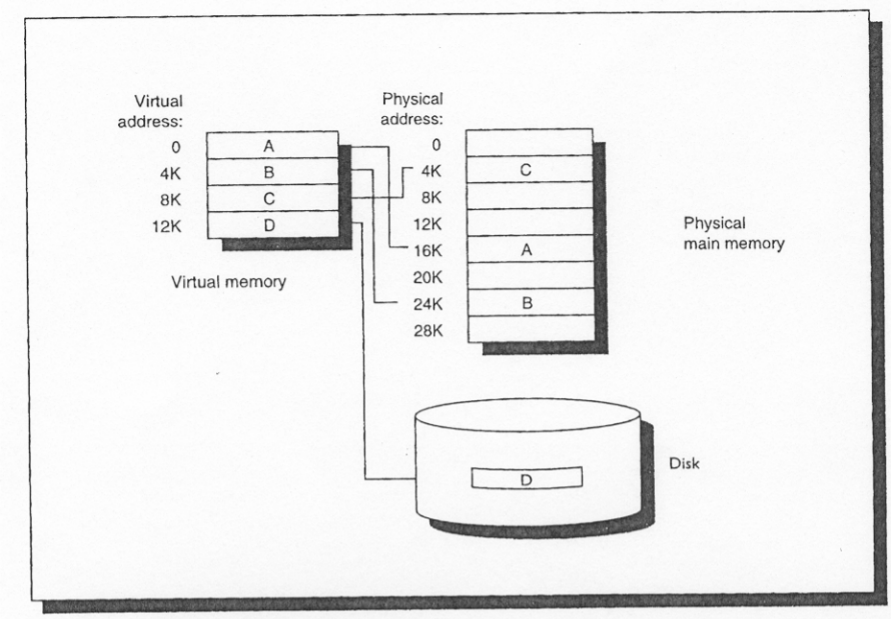


Tres ejemplos de anchura de bus, ancho de memoria, y entrelazado de memoria para lograr mayor ancho de banda de memoria. a) es el diseño más simple, siendo todo de la anchura de una palabra; b) muestra una memoria, bus y cache más anchos; mientras que c) muestra una cache y bus delgados con una memoria entrelazada.



Four-way interleaved memory. This example assumes word addressing: with byte addressing and four bytes per word, each of these addresses would be multiplied by four.

MEMORIA VIRTUAL Y COMPARACIÓN CON CACHÉ

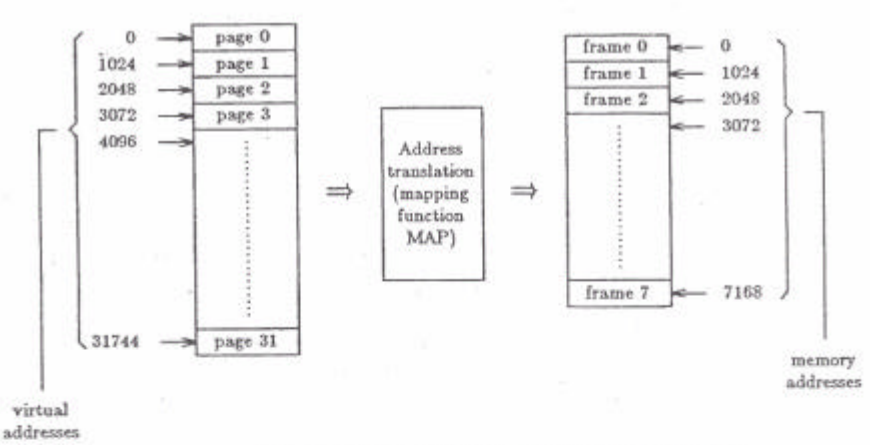


The logical program in its contiguous virtual address space is shown on the left: it consists of four pages A, B, C, and D. The physical location of three of the blocks is physical memory and one is located on disk.

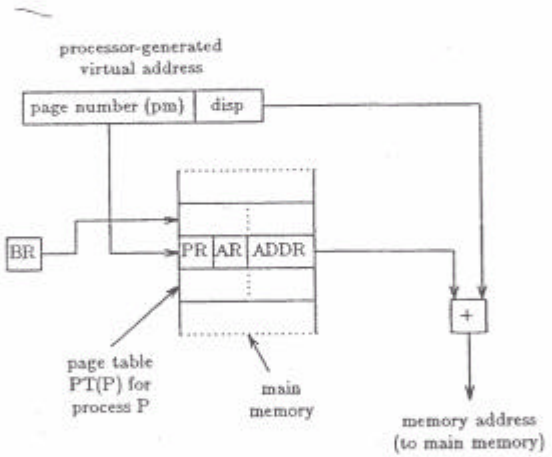
Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–2 clock cycles	40–100 clock cycles
Miss penalty	8–100 clock cycles	700,000–6,000,000 clock cycles
(Access time)	(6–60 clock cycles)	(500,000–4,000,000 clock cycles)
(Transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.5–10%	0.00001–0.001%
Data memory size	0.016–1MB	16–8192 MB

Typical ranges of parameters for caches and virtual memory. Virtual memory parameters represent increases of 10 to 100,000 times over cache parameters.

TRADUCCIÓN DE DIRECCIONES EN MEMORIA VIRTUAL PAGINADA



Paging scheme with 1K page/page-frame size.



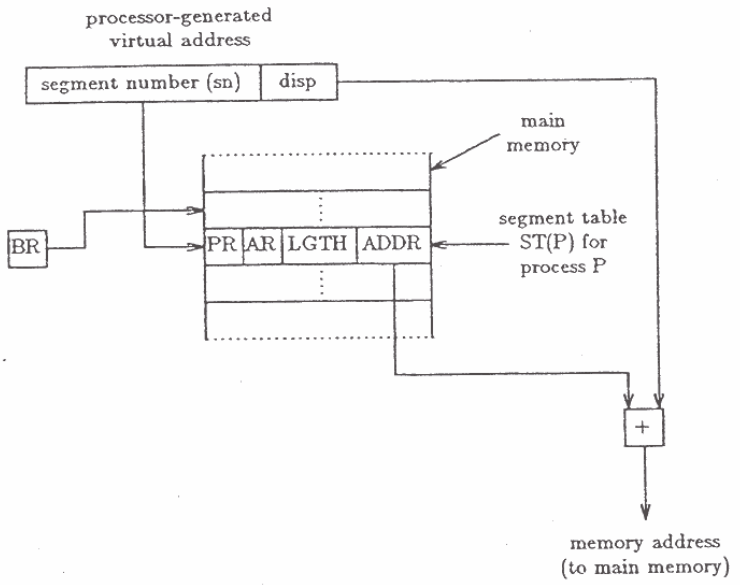
Legend
(a) **if** PR = 1 **then** page is in main memory at frame address ADDR
else page is in secondary memory at address ADDR
(b) AR: access right: R = read only; W = read/write
X = execute only

Address Translation

if is_valid(PT(P).AR)
then if PT(P).PR = 1 **then new** memory address = PT(P).ADDR + disp
else trap page_fault
else trap protection_violation_fault

Address translation in paged virtual memory system.

TRADUCCIÓN DE DIRECCIONES EN MEMORIA VIRTUAL SEGMENTADA



Legend
(a) AR: access right to segment
(b) LGTH: segment length
(c) **if** PR = 1 **then** segment is in main memory at address ADDR
else segment is in secondary memory at address ADDR

Address Translation

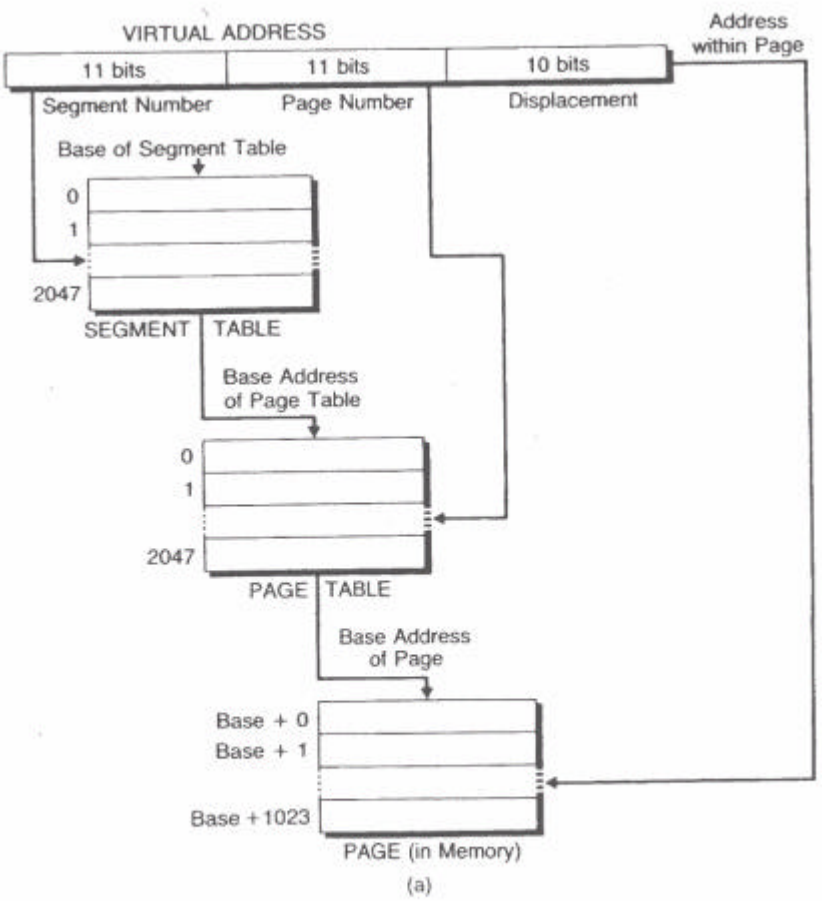
```
if ST(P)[sn].PR = 0 then trap segment_fault
    else if not valid(ST(P)[sn].AR)
            then if ST(P)[sn].LGTH < disp
                    then trap overflow_fault
                    else new memory_address = ST(P)[sn].ADDR + disp
            else trap protection_violation_fault
```

Address translation in segmented virtual memory system.

COMPARACIÓN ENTRE MEMORIA VIRTUAL SEGMENTADA Y PAGINADA

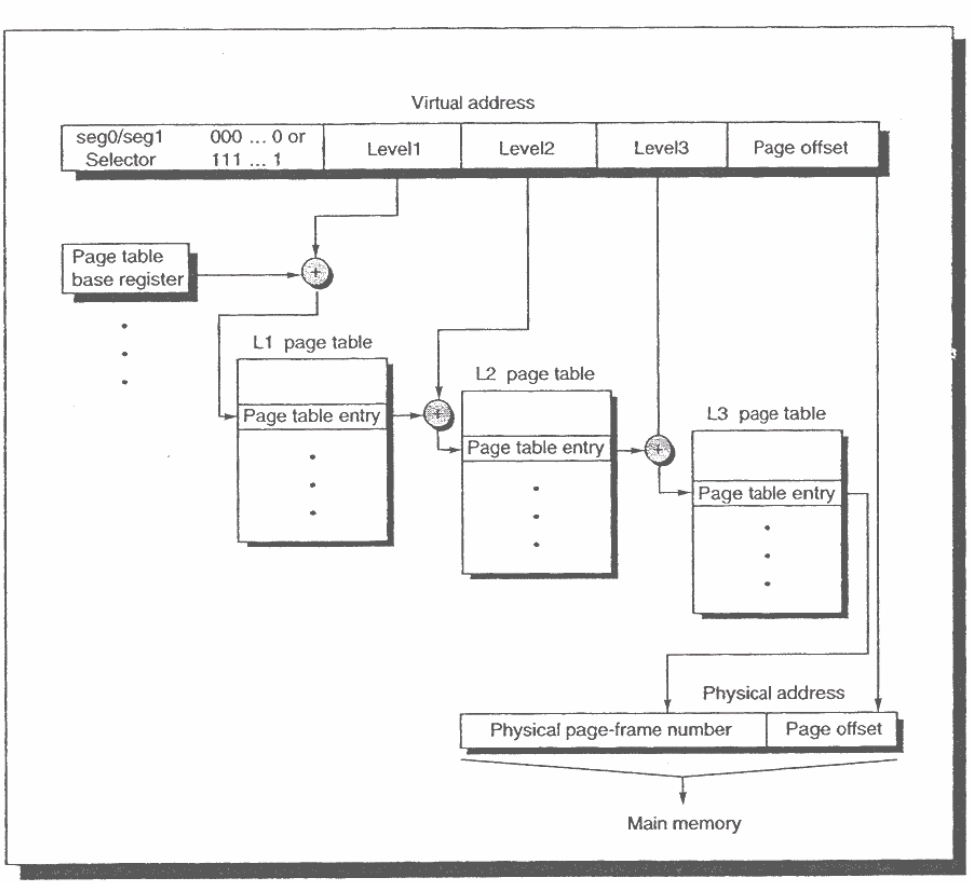
	Segmentada	**Paginada**
¿Visible al programador?	Sí. En las direcciones virtuales debe distinguir el segmento y el desplazamiento.	No
Mecanismos de protección de memoria	Más potentes. Se pueden proteger por separado código y datos.	Menos potentes
Reemplazo de bloque	Complicado por el diferente tamaño de los segmentos	Trivial
Fragmentación	Externa (los segmentos están llenos pero quedan zonas de memoria sin usar) $\Rightarrow$ necesidad de compactación (consume t.)	Interna (una parte de la última página de un programa suele quedar sin usar)
Eficiencia tráfico disco-memoria	Baja con segmentos pequeños	Buena al poder elegir un tamaño de página en que estén equilibrados los tiempos de transferencia y acceso al disco

MEMORIA VIRTUAL CON DOS NIVELES DE PAGINACIÓN



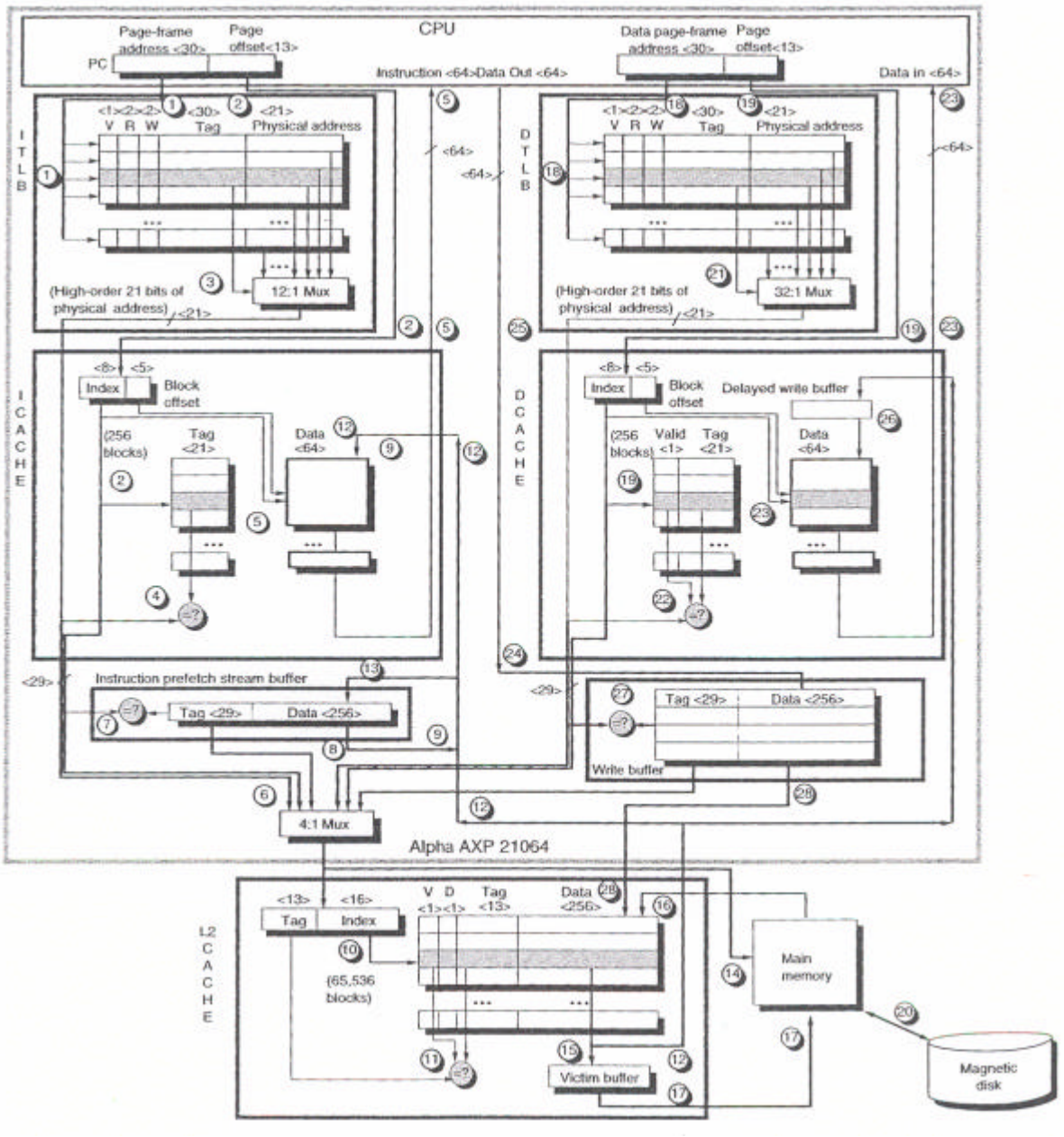
Two-level mappings:
(a) A typical two-level mapping; and
(b) A two-level mapping used in the VAX architecture.

PAGINACIÓN DE MEMORIA VIRTUAL EN LA ARQUITECTURA ALPHA AXP



The mapping of an Alpha virtual address. Each page table is exactly one page long, so each level field is n bits wide where 2^n = page size/8. The Alpha AXP architecture document allows the page size to grow from 8 KB in the current implementations to 16 KB, 32 KB, or 64 KB in the future. The virtual address for each page size grows from the current 43 bits to 47, 51, or 55 bits and the maximum physical address size grows from the current 41 bits to 45, 47, or 48 bits. The 21064 uses 8-KB pages, but it implements just 34 bits of the maximum 41-bit physical address possible in this scheme.

JERARQUÍA DE MEMORIA EN EL ALPHA AXP 21064



ACCESOS A MEMORIA A TRAVÉS DE LA JERARQUÍA DEL ALPHA AXP 21064 (1 de 4)

1. La parte de número de página del PC se envía al ITLB (TLB de instrucciones).

2. Al mismo tiempo, los 8 bits más significativos del desplazamiento (*index*) se envían a la caché de instrucciones (de 8 KB y correspondencia directa).

3. La ITLB, que es completamente asociativa, compara simultáneamente sus 12 marcas (tiene 12 entradas) con el número de página enviado desde el PC y, si alguna marca coincide, se selecciona la entrada correspondiente. Además el ITLB comprueba si el acceso viola la protección de la página o si la página no está en memoria principal. En cualquiera de estos casos se produce una excepción.

4. Si en el paso 3 no se produce excepción se compara la dirección física obtenida (los 21 bits) con la marca seleccionada en el paso 2.

5. Si hay acierto, se selecciona la palabra de 8 B, entre las 4 que tiene un bloque, utilizando el índice y los dos bits menos significativos del desplazamiento de página. La palabra seleccionada se envía a la UCP.

6 y 7. Si hay fallo, se inicia un acceso a la caché de segundo nivel y (7) se comprueba el buffer de prebúsqueda de instrucciones.

8 y 9. Si la instrucción se encuentra en el buffer, la palabra buscada de 8 bytes se envía a la UCP [no indicado en el dibujo], (9) el bloque de 32 bytes se escribe en la caché de instrucciones y la solicitud a la caché de segundo nivel se anula. Los pasos 6 a 9 sólo requieren un ciclo de reloj.

10. Si la instrucción no estaba en el buffer se busca el bloque en la caché de segundo nivel (de correspondencia directa). Suponemos una configuración en que esta caché tiene 2 MB, con bloques de 32 B. Por tanto la dirección de bloque de 29 bits se divide en 16 bits para el conjunto (*index*) y el resto para la marca.

ACCESOS A MEMORIA A TRAVÉS DE LA JERARQUÍA DEL ALPHA AXP 21064 (2 de 4)

11. La marca se compara con la seleccionada en la caché mediante el índice.

12. Si hay acierto la caché de segundo nivel envía a la de primer nivel el bloque. En los primeros 5 ciclos de reloj envía los 16 B críticos, y en los 5 siguientes los 16 B restantes. (El camino que une ambas cachés tiene 128 bits de anchura (=16 bytes).

13. Al mismo tiempo, se solicita el siguiente bloque en orden secuencial y se lleva al buffer de prebúsqueda de instrucciones. (10 ciclos de reloj). Para acceder al siguiente bloque no se pasa por el TLB, simplemente la dirección física se incrementa en 32 bytes (el tamaño de bloque) y se comprueba que la nueva dirección está dentro de la misma página. Si no fuera así no se haría la prebúsqueda.

14. Si la instrucción no se encuentra en la caché secundaria la dirección física se envía a memoria principal.

15. La caché de segundo nivel es de postescritura por lo que, si el bloque a reemplazar ha sido modificado, hay que llevarlo a memoria. El 21064 coloca dicho bloque ("víctima") en un buffer víctima para dar prioridad a la transmisión del nuevo bloque. El buffer sólo tiene capacidad para un bloque por lo que, cuando hay varios fallos seguidos, se encontrará ocupado y habrá que esperar a que se vacie.

16. El nuevo bloque se lleva de memoria principal a caché de segundo nivel. El tiempo medio para transferir un bloque de memoria principal a la caché de segundo nivel es de 36 ciclos de reloj desde que el procesador hace la petición. Para transferir un bloque hay que hacer dos transferencias de 16 bytes.

17. El bloque víctima se lleva del buffer a memoria principal.

Supongamos que la instrucción encontrada es de carga.

18. El número de página de la dirección del dato a cargar se envía al DTLB.

ACCESOS A MEMORIA A TRAVÉS DE LA JERARQUÍA DEL ALPHA AXP 21064 (3 de 4)

19. Al mismo tiempo, los 8 bits más significativos del desplazamiento (*index*) se envían a la caché de datos (de 8 KB y correspondencia directa).

20. La DTLB, que es completamente asociativa, compara simultáneamente sus 32 marcas (tiene 32 entradas) con el número de página enviado. Si hay un fallo se produce un salto a una rutina PAL (*Priviliged Architecture Library*). Estas rutinas se ejecutan con las excepciones inhabilitadas. La rutina cargará una entrada válida para el número de página enviado. En el peor caso, habrá que traer la página del disco.

21. Bien mediante un acierto en la DTLB, bien mediante la rituna PAL, la entrada adecuada de la DTLB nos proporciona la dirección física del marco de página.

22. En la caché de datos se compara esa dirección con la marca seleccionada en 19.

23. Si hay acierto, la palabra de 8 bytes se selecciona del bloque de 32 bytes (el mismo mecanismo que en la caché de instrucciones) y se envía a la UCP.

En caso de fallo se va a la caché de segundo nivel y se actúa como en un fallo de instrucciones.

Supongamos ahora que la instrucción es de almacenamiento.

Los pasos 18 a 22 son similares. La caché de datos usa una política de escritura directa por lo que:

24 y 25. El dato a escribir es enviado simultáneamente al buffer de escritura (paso 24) y a la caché de datos (paso 25 [no está dibujado el camino]).

ACCESOS A MEMORIA A TRAVÉS DE LA JERARQUÍA DEL ALPHA AXP 21064 (4 de 4)

26. Las escrituras se solapan. Por tanto, se comprueba si la dirección de la instrucción actual de almacenamiento corresponde a un acierto y, al mismo tiempo, si el buffer de escrituras retardadas no está vacío, se escribe la palabra que contiene en la caché de datos (el buffer también contiene la dirección para saber donde escribir). A continuación, si la dirección correspondió a un acierto, el dato a escribir se sitúa en el buffer. Si correspondió a un fallo no hay que hacer nada en esta caché porque se sigue una política de no ubicación en fallos.

27. El buffer de escritura (no confundir con el anterior) tiene cuatro entradas, cada una con capacidad para un bloque. Si está lleno, la UCP debe esperar hasta que se pase un bloque del buffer a la caché de segundo nivel. Si no, la UCP continúa y la dirección de la palabra se lleva al buffer el cual comprueba si la palabra se corresponde con alguno de los bloques que contiene. Esto sucede cuando se escriben varias palabras de direcciones consecutivas. Entonces el buffer puede completar los bloques (recuérdese que cada bloque tiene 4 palabras) y así optimizar el ancho de banda entre el buffer y la caché de nivel 2.

28. Las escrituras que son aciertos dan lugar a que se copien datos del buffer en la caché de nivel 2. Al copiar un bloque completo (32 bytes) se requieren 5 ciclos para comprobar la dirección y 10 para escribir los datos. Si se escriben 16 bytes o menos, se requieren 5 ciclos para comprobar la dirección y 5 para escribir los datos. En cualquier caso el bit de ensuciado del bloque se pone a 1.

Si el acceso a la caché de segundo nivel es un fallo se actúa con el bloque a reemplazar como en el paso 15. Si los nuevos datos constituyen un bloque completo, se escriben en la caché de segundo nivel y el bloque se marca como modificado. Si no, hay que acceder a memoria principal porque la política de la caché de segundo nivel es ubicar en un fallo de escritura.