

## Problema 13. Continuación anticipada, Transferencia en desorden

Un procesador tiene una cache de primer nivel (que llamaremos L1) con bloques de 32 bytes y está conectado a un segundo nivel de jerarquía de memoria (que llamaremos L2) mediante un bus de 8 bytes de ancho. El primer nivel (L1) tiene un tiempo de acceso de 1 ciclo cuando se produce un acierto. Cuando fallamos en el primer nivel accedemos al segundo nivel (L2) para leer un bloque de datos. Suponemos que solo se realizan lecturas y que nunca hay fallos en L2. El siguiente cronograma ilustra un fallo en L1: se necesitan 5 ciclos de latencia y 4 para transferir los datos (T0-T3). Los datos se cargan en L1 mientras se transfieren (car L1), y una vez tenemos todo el bloque en L1, hay que realizar una lectura en L1 (Lect) para enviar el dato a la CPU (DATO), cosa que ocurre dentro del mismo ciclo.

The diagram illustrates the timing of a CPU access across two levels of cache (L1 and L2) over 16 clock cycles. The vertical axis on the left lists the components: CLK (Clock), Ciclo (Cycle), CPU, L1, and L2. The horizontal axis at the top shows the clock cycles from 1 to 16. The CPU bus (represented by a dashed line) is active during odd-numbered cycles (1, 3, 5, 7, 9, 11, 13, 15). The L1 cache bus (solid line) is active during even-numbered cycles (2, 4, 6, 8, 10, 12, 14). The L2 cache bus (dotted line) is active during odd-numbered cycles (1, 3, 5, 7, 9, 11, 13). A 'MISS' label is shown in the first row of the L1 cache bus. The 'DATO' label is shown in the fourth row of the CPU bus. The 'Latencia' label is positioned below the L1 cache bus. The T0, T1, T2, and T3 labels are located at the bottom right.

Al ejecutar un programa en un simulador, hemos obtenido que, a una frecuencia de 2GHz el programa tardaría 2 segundos en el caso ideal de que no haya fallos de cache, que se han realizado  $10^9$  accesos a memoria y que el 20% de los accesos provocarían un fallo en L1.

- a) Calculad el tiempo de ciclo y los ciclos que tarda el programa en el caso ideal.

$$T_C = \frac{1}{f} = 2 \times 10^{9-1} = 5 \times 10^{-10} \text{ s} \quad , \quad C_{id} = \frac{2s}{5 \times 10^{-10} \text{ s}} = \frac{t_{exec}}{t_c} = 4 \times 10^9 \text{ cycles}$$

- b) Calculad los ciclos de penalización de un fallo de L1 y el tiempo de ejecución

Cálculo los ciclos de penalización de un fallo de L1 y el tiempo de ejecución del programa teniendo en cuenta la penalización debida a los fallos de L1. Son 10 ciclos de penalización.

$$C_F = 10^9 \cdot 2 \cdot 10 = 2 \times 10^9 \text{ ciclos}, C_{TOTAL} = 4 \times 10^9 + 2 \times 10^9 = 6 \times 10^9 \text{ ciclos}$$

$$T_{exe} = \frac{n_{ciclos}}{KC/S} = \frac{6 \times 10^9}{2 \times 10^9} = 3 \text{s}$$

Para mejorar el rendimiento, permitimos que el procesador pueda captar el dato en el mismo ciclo que se transfiere de L2 a L1 (continuación anticipada o *early restart*). Mediante simulación sabemos que cuando se produce un fallo en L1 en nuestro programa, en el 70% de los casos el dato deseado se corresponde al que se transfiere en el ciclo T0, mientras que hay una probabilidad del 10% de que se transfiera en cada uno de los ciclos restantes (T1-T3). Sabemos además que nunca se produce un fallo mientras se acaba de transferir el resto del bloque.

- c) Completad el siguiente cronograma en donde se ilustran las acciones a realizar en caso de fallo en L1, suponiendo que tenemos continuación anticipada y que el dato solicitado se corresponde al byte 12 del bloque.

The diagram illustrates the timing of a CPU access across 16 clock cycles. The CPU sends a request to the L1 cache at cycle 1. A 'MISS' signal is received at the CPU at cycle 2. The L1 cache sends a request to the L2 cache at cycle 3. The L2 cache returns data to the L1 cache at cycle 4. The L1 cache then sends the data to the CPU at cycle 5. The CPU sends a write-back to the L1 cache at cycle 6. The L1 cache sends a response back to the CPU at cycle 7. The CPU continues its execution, with data being read from memory at cycles 8-11 and written back to the L1 cache at cycles 12-15. The word 'DATA' is written vertically along the data bus between cycles 8 and 15. The total latency is labeled as LATENCIA.

- d) Calculad el tiempo medio de penalización (en ciclos) de un fallo en L1 y el tiempo de ejecución del programa para el procesador con continuación anticipada.

$$C_F = 10^9 \cdot 2 \pi \cdot 6.6 = 1'32 \cdot 10^9 \text{ cycles FAULTS}, T_{exe} = \frac{\text{cycles}}{\text{cycles/clock}} \cdot T_{clock} = 4 \times 10^9 + 1'32 \times 10^9 \cdot 5 \times 10^{-10} = 2'66s$$

Para mejorar más el rendimiento hemos modificado también L2 de forma que se transfiera el dato solicitado en el primer ciclo ( $T_0$ ) de la transferencia (transferencia en desorden). Sabemos además que nunca se produce un fallo mientras se acaba de transferir el resto del bloque.

- e) Completad el siguiente cronograma en donde se ilustran las acciones a realizar en caso de fallo en L1, suponiendo que tenemos transferencia en desorden y que el dato solicitado se corresponde al byte 12 del bloque.

The diagram illustrates the timing of a CPU access to two levels of cache (L1 and L2) across 16 clock cycles. The CPU sends an address to both caches. The L1 cache returns a 'MISS' signal during cycles 1-4. The L2 cache returns data during cycles 5-8. Handwritten annotations in blue indicate the latency of each stage: 'T1' for the time from the CPU address to the L2 response, 'T2' for the time from the L2 response to the data on the DATA bus, 'T3' for the time from the L2 response to the end of the cycle, and 'T4' for the time from the end of the cycle to the next one. The CLK signal shows active-low clock edges.

- f) Calculad el tiempo medio de penalización (en ciclos) de un fallo en L1 y el tiempo de ejecución del programa para el procesador con continuación anticipada. Son 6 ciclos penalizados.

$$T_{ave} = (4 \cdot 10^9 + 10^9 \cdot 2 \cdot 6) \cdot \frac{TC}{5 \times 10^{-10}} = 2'6s$$

- g) Calculad la ganancia (speedup) del sistema con continuación anticipada y del sistema con transferencia en desorden respecto al sistema sin ninguna mejora.

$$S_1 = \frac{3}{2'66} = 1'128 \quad S_2 = \frac{3}{2'6} = 1'54$$

#### Problema 14.

##### Prefetch

Disponemos de un procesador de 16 bits con direcciones de 16 bits que tiene una memoria cache de datos con las siguientes características: 3-asociativa, con algoritmo de reemplazo LRU, 12 bloques y 64 bytes por bloque, política de escritura: *copy back + write allocate*

El contenido inicial de la memoria de etiquetas (tags) es el siguiente:

conjunto 0	DB	conjunto 1	DB	conjunto 2	DB	conjunto 3	DB
13	1	13	1	13	0	13	0
43	1	43	1	43	0	43	0
AC	0	AC	0	AC	1	AC	1

El DB=1 indica que el bloque correspondiente ha sido modificado. La información de reemplazo está implícita en la posición. Las posiciones inferiores corresponden a los bloques que llevan más tiempo sin utilizarse. Las posiciones superiores corresponden a los últimos bloques utilizados. Por ejemplo, en el conjunto 3, el bloque con tag 13 es el último utilizado, y el bloque con tag AC el que lleva más tiempo sin ser utilizado.

- a) Rellenad la siguiente tabla, indicando para cada referencia, el número de bloque de memoria que le corresponde, la etiqueta (TAG), a qué conjunto de MC va a parar, si es acierto o fallo (A/F), el bloque reemplazado cuando proceda, el número de bytes leídos de MP (si se lee de MP) y el número de bytes escritos en MP (si se escribe en MP)

tipo	dirección (hex)	bloque de memoria (hex)	TAG (hex)	conjunto MC	¿acierto o fallo? (A/F)	bloque reemplazado	bytes escritura MP	bytes lectura MP
LECT	B12B	2C4	B1	0	M	AC		64
LECT	B145	2C5	B1	1	M	AC		64
LECT	B1AF	2C6	B1	2	M	AC	64	64
LECT	B1C4	2C7	B1	3	M	AC	64	64
ESCR	4387	10E	43	2	H			
LECT	1108	044	11	0	M	43	64	64
ESCR	1199	046	11	2	M	13		64
LECT	11AA	046	11	2	H			

A la cache anterior le añadimos un *buffer de prefetch* de una entrada. En este buffer se hace prebúsqueda hardware del bloque *i+1* cuando se accede (tanto en acierto como en fallo) al bloque *i*, siempre que el *i+1* no esté ya en la cache o en el buffer. En este último caso, no se realiza *prefetch*.

- b) Rellenad la siguiente tabla (mismas referencias que la anterior) indicando, para cada referencia, el número de bloque de memoria que le corresponde, la etiqueta (TAG), a qué conjunto de MC va a parar, si se produce acierto o fallo en la cache (A/F), el número de bytes leídos de MP (si se lee de MP), el número de bytes escritos en MP (si se escribe en MP), el bloque de MP que se encuentra en el *buffer* (si procede), si se produce acierto o fallo (A/F) en el *buffer* y el bloque que se prebusca de MP (si procede).

tipo	dirección (hex)	bloque de memoria (hex)	TAG (hex)	conjunto MC	Cache ¿acierto o fallo?	bytes escritura MP	bytes lectura MP	bloque actual buffer	Buffer ¿acierto o fallo?	bloque prefetch buffer
LECT	B12B	2C9	B4	0	M		64.2		M	2C5
LECT	B145	2C5	B4	1	M		64	2C5	H	2C6
LECT	B1AF	2C6	B4	2	M	64	64	2C6	H	2C7
LECT	B1C4	2C7	B4	3	M	64	64	2C7	H	2C8
ESCR	4387	10E	43	2	H			2C8	M	
LECT	1108	044	11	0	M	64	64+64		M	045
ESCR	1199	046	11	2	M	64+64	64S	M	047	
LECT	11AA	046	11	2	H		047	M		

# Apéndice: Cronogramas Tema 3

५८

## Problema 16

## Cronograma 1: Buffer de 1 entrada

identisch als 2 Primus ...

CPI = ... 24 c./inst.

$$\text{Ancho de banda} = \frac{\text{bytes/ciclos}}{cycles} = \frac{8b}{42\text{cycles}} = 0'6 \text{ bytes/ciclo}$$

$$CPI = \frac{n_{cycles}}{n_{inst}} = \frac{42 \text{ c/iter} \cdot 10^6}{5 \text{ inst/iter} \cdot 10^6} =$$

## Cronograma 2: Buffer de 2 entradas

$$CPI = \frac{cycles}{inst}$$

Ancho de banda = ..... 8b per 10 cycles  $\Rightarrow$  0.8 bytes/cycle

$$CPI = \frac{10 \text{ cl/liter} \cdot 10^6}{5 \text{ ml/liter} \cdot 10^6} = \dots$$

## Cronograma 3: Buffer de 3 entradas

$$CPI = \frac{\text{cycles}}{\text{instructions}}$$

$$CPI = \frac{10 \text{ cicl/iter} \cdot 10^6}{5 \text{ fasi/iter} \cdot 10^6} = \dots$$

Ancho de banda = 8 bytes/cada 10 ciclos  $\Rightarrow$  0.8 Bytes/ciclo

TOT i que s'ha Afegit una tucera entroada  
Al Buffer, l'ample de banda no ha augmentat  
; É per això que no és una millora substancial.

**Cronograma 4: Merge buffer de 3 entradas**

$$CPT = \frac{\text{Sicks/liter} \cdot 10^6}{\text{Sintraf/liter} \cdot 10^6} = 1 \text{ cicle/instr.}$$

Amplio BandPi: 8bytes (da Scicles) => 1'6 bytes / ciclo