Problemas Tema 3: Jerarquía de Memoria

1. Sea un computador que tiene un bus de direcciones de 32 bits y direccionamiento a nivel de byte. La cache tiene 256 Kbytes, con palabras de 1 byte y bloques de 32 bits organizados con asignación directa, reemplazo LRU y post-escritura con 1 bit de dirty para los bloques que hayan sido modificados y por tanto estén pendientes de escribirse en memoria principal. Responde a las siguientes preguntas:
   a) Descompón en campos la dirección física (campo TAG, campo índice, etc.)
   b) ¿Cuál es el tamaño, en bits, del directorio de la cache? Recuerda que en el directorio se encuentran todos los bits de control necesarios para el funcionamiento de la memoria cache.
   c) ¿Son coherentes todos los datos que proporcionan sobre la memoria caché? Justifica tu respuesta.

2. Una memoria cache asociativa por conjuntos de 2 bloques con direcciones de 32 bits y palabras de 1 byte tiene 4 bytes en cada uno de sus bloques y una capacidad total de 128 Kbytes. El direccionamiento es a nivel de byte. Se pide:
   a) El tamaño de una dirección de memoria y su descomposición en los campos número de línea o bloque (M), etiqueta (M-c), conjunto (c) y desplazamiento dentro del bloque (w).
   b) Indicar el valor de cada campo para las siguientes direcciones de memoria principal:
      `0284A482h, 01148C89h, 0038CF00h, 0038CF01h`
   c) ¿Pueden todos los bloques que incluyen las referencias anteriores estar en caché al mismo tiempo?

3. Sea un sistema de memoria de 1 MByte, con palabras de 1 byte, que incorpora una memoria cache de 64 KBytes organizada asociativamente en cuatro conjuntos con 8 palabras por bloque y algoritmo de reemplazo LRU. Se ejecuta un programa que referencia a la siguiente secuencia de direcciones (en hexadecimal):

   `ABC80h, ABC81h, ABC88h, BCD90h, BCD9Dh, BCDA0h, CDE00h, CDE18h, CDE20h`

   a) Muestra la evolución de la cache, indicando para cada referencia si se produce un fallo o un acierto. Calcula el índice de fallos.
   b) Repite el ejercicio anterior calculando el índice de fallos suponiendo que la secuencia de referencias se corresponde al cuerpo de un bucle y el número de iteraciones tiende a infinito.

4. Disponemos de una memoria cache de 4 Kb con tamaño de bloque de 256 bytes, asociatividad 2 y algoritmo de reemplazo LRU. Conectamos la cache entre un procesador que trabaja con palabras de 8 bits y una memoria principal de 1 Mb. Se pide:
   a) Suponiendo la cache inicialmente vacía, describir la evolución del directorio cache para la siguiente secuencia de peticiones de direcciones a memoria principal:

      `319F0h, 31AF0h, 7013Ch, 77777h, 44037h, 778DEh, A5021h`

b) Comparar el sistema anterior con respecto a criterios de rendimiento y coste, frente a una cache organizada de forma directa y a otra totalmente asociativa.

5. Un adicto a los PCs pretende evaluar el comportamiento de la caché L1 de datos del procesador "Pear P3" para un famoso videojuego de moda. Dispone para ello de una arquitectura con 16 Mbytes de memoria principal direccionable a nivel de byte y con caché L1 para datos 8 Kbytes. En cuanto al videojuego (programa a ejecutar), se compone de un bucle de 100 iteraciones, referenciando en cada una de ellas a una secuencia de 4 datos ubicados en las siguientes direcciones de memoria principal (en hexadecimal): `000A40h, 004A40h, 008A40h, 00BA40h`. Nuestro amigo observa que la caché de datos evoluciona de la siguiente manera para la primera iteración del juego (en las 99 restantes, el comportamiento es muy similar):

Al final de la primera referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	000h
	1	-
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la segunda referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	000h
	1	004h
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la tercera referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	008h
	1	004h
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la cuarta referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	008h
	1	00Bh
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

El significado de cada campo es el siguiente: C = Número de conjunto; L = Número de línea; Etiq = Campo etiqueta del directorio caché; "-" = Línea vacía).

a) En función del comportamiento mostrado en los diagramas anteriores, se pide obtener el número de conjuntos de la caché de datos del "Pear P3", tamaño de la etiqueta, tamaño de la línea de caché y todas las estrategias de reemplazo de líneas que pueden dar lugar al resultado anterior (excluyendo RANDOM).

Número de conjuntos:	Tamaño de la etiqueta(en bits):	Tamaño de bloque (en bytes):	Reemplazo:

b) Nuestro amigo quiere ahora cambiar el procesador de su equipo, dudando entre adquirir un "Carrot C4"o un "Grape G5". El "Carrot C4" dispone de 16 Kbytes para la caché de datos y las mismas características que en el "Pear P3" original, aumentando únicamente el nivel de asociatividad (esto es, se tiene el mismo número de conjuntos, pero se dobla el número de bloques por conjunto). El "Grape G5", por el contrario, tiene una caché de datos de 32 Kbytes, pero mantiene el nivel de asociatividad y el resto de parámetros del "Pear P3", con la salvedad del número de conjuntos, que es ahora cuatro veces superior. ¿Qué microprocesador recomendarías a nuestro amigo atendiendo exclusivamente al índice de aciertos a caché de datos producido por las referencias a memoria efectuadas desde el mencionado bucle del videojuego?. Ayúdate de una traza del directorio caché para cada caso con el mismo formato que el que hemos utilizado para el "Pear P3".

6. Considérese un procesador con instrucciones de 32 bits y una caché de instrucciones de 32 bytes con líneas de 8 bytes. Para los dos fragmentos de código MIPS siguientes, indicar la organización (y sus parámetros) que da lugar a una ejecución con el mínimo número de fallos (si hay varias que produzcan el mismo rendimiento, ordenar por coste hardware). NOTA: El número a la izquierda de cada instrucción indica la dirección en memoria principal donde se encuentra.

```
0   lw $1, 100($2)
4   add $1, $1, $3
8   sub $4, $5, $1
12  j 24
          ...
24  mul $2, $2, $8
28  sub $2, $2, $9
32  div $8, $1, $4
36  j 0
```

```
0    lw $1, 100($2)
4    add $1, $1, $3
8    sub $4, $5, $1
12   j 32
          ...
24   mul $2, $2, $8
28   sub $2, $2, $3
32   add $8,
36   j 24
```

7. Sea el siguiente programa en MIPS:

```
      addi $1, $0, 1
      j  lab1
lab0: lw $1, 16($0)
      sw $1, 80($0)
      j  lab1
      …
lab1: sw $0, 16($0)
      lw $2, 208($0)
      lw $3, 336($0)
      sw $2, 400($0)
      bne $1, $0, lab0
```

donde la dirección `lab0` es igual a 8 y `lab1` es igual a 140. Implementamos el primer nivel de la jerarquía de memoria con una cache de instrucciones, con asignación directa y otra cache de datos asociativa de 4 conjuntos y reemplazo LRU. Ambas caches tienen bloques de 8 bytes y un tamaño de 128 bytes. Para cada cache se pide:

a) Mostrar la secuencia de referencias que hace el programa durante su ejecución.

b) Describe cada una de las caches, indicando el tamaño tanto de la zona de control como de la de almacenamiento. Muestra la evolución de cada una de las caches, indicando para cada referencia si se produce un fallo o un acierto.
c) Calcula las tasas de fallo.

8. Sea el siguiente programa en MIPS:

```
      ori  $2, $0, 1000h
loop: lw   $1, 2800h($2)
      sub  $4, $1, $0       rotar:    add  $10, $4, $4
      jal  rotar                      muli $7, $10, 2
      sw   $7, 7800h($2)              jr $31
      sw   $1, C800h($2)
      subi $2, $2, 4
      bne  $2, $0, loop
```

Se desea usar una memoria cache de tamaño 4 Kbytes, con asignación directa, para resolver las referencias a datos. Se pide:

a) ¿Cuál es el tamaño de bloque óptimo teniendo en cuenta la tasa de fallos? Prueba distintos tamaños de bloque, empezando con 1 palabra por bloque y calcula sus tasas de fallo.
b) Si ahora fijamos el tamaño de bloque en 256 bytes, ¿es posible reducir la tasa de fallos aumentando el nivel de asociatividad? ¿A partir de qué nivel de asociatividad no se obtiene mejora?
c) Para la configuración elegida en el apartado anterior, ¿Qué política de reemplazo es mejor, LRU o FIFO?

9. En la figura se proponen dos códigos en alto nivel que inicializan ambos una matriz de 10×10 números reales a cero. El código generado por el compilador almacena la matriz en la memoria en posiciones consecutivas por filas. Las variables i, j son ubicadas en dos registros diferentes del procesador (NO están en memoria).

`i=1`
`While (i <=10)Do`
`  j=1`
`  While (j <=10)Do`
`  A(i,j)=0.0`
`  j=j+1`
`  End While`
`  i=i+1`
`End While`
	`i=1`
	`While (i <=10)Do`
	`  j=1`
	`  While (j <=10)Do`
	`  A(j,i)=0.0`
	`  j=j+1`
	`  End While`
	`  i=i+1`
	`End While`

Código A	*Código B*

Se dispone de un sistema de memoria cuya caché es completamente asociativa y con reemplazo FIFO. El tamaño de la caché es equivalente a diez números reales en la representación usada por el procesador y permite almacenar dos números por bloque. Ambos códigos realizan la misma función, pero desde el punto de vista del sistema de memoria el rendimiento es diferente. Analiza cuál de ellos daría más fallos de caché.

10. Queremos ejecutar un simple bucle como éste:

```
DO i=0,9
  A(i)=0
ENDDO
```

en una procesador con una memoria caché de 8 bytes y tamaño de bloque 2 bytes. Sabiendo que A es un array de 30 números almacenados en memoria a partir de la dirección 0 y que cada elemento del array ocupa 1 byte:

a) ¿Qué tipo de organización de memoria cache (directa, por conjuntos o totalmente asociativa) elegirías en función del porcentaje de fallos? Nota: a igualdad de rendimiento se debe elegir la menos costosa desde el punto de vista HW). Justifica la respuesta.
b) ¿Se podría decir lo mismo independientemente de la posición de comienzo del array A en memoria?. Razonar adecuadamente.

Imaginemos ahora que el bucle que estamos procesando es este otro (en la sentencia `A(i)=B(9-i)` el acceso de lectura ocurre antes que el de escritura):

```
DO i=0,9
  A(i)=B(9-i)
ENDDO
```

donde A es el mismo array anterior y B es otro array de 45 números de 1 byte, almacenado a partir de la posición 60 de memoria.

c) ¿Qué tipo de organización de memoria cache elegirías ahora?. Justifica la respuesta calculando el porcentaje de fallos para cada caso.
d) ¿Es independiente la respuesta anterior de la posición de comienzo del array B en memoria?. Pon ejemplos explicativos.

11. Sea un sistema de memoria de 1 MByte, con palabras de 1 byte, que incorpora una memoria caché de 64 KBytes, con 4096 palabras por bloque y algoritmo de reemplazo FIFO. Se ejecuta un programa que referencia a la siguiente secuencia de direcciones (en hexadecimal):

`00000h, 01000h, 01001h, 0F000h, 10000h, 00000h, 40000h, 20000h, 08000h`

a) Compara las tasas de fallo cuando se considera una organización con asignación directa, totalmente asociativa y asociativa con 4 conjuntos.
b) Calcula el tamaño de cada una de las caches incluyendo la zona de directorio.

12. Considerar una memoria principal de 64 Kbytes direccionable a nivel de byte y con palabras de este mismo tamaño. Sea la siguiente secuencia de direcciones de acceso a memoria principal:

`04F5h, 11E0h, 1500h, 2000h, 241Fh, 16FFh, 1233h, 21F0h`

Mostrar el contenido del directorio cache, así como los fallos que se producen para cada una de las caches descritas en la siguiente tabla:

	Tamaño Cache	Tamaño Bloque	Organización	Reemplazo
C1	4 Kb	256 bytes	Directa	---
C2	2 Kb	256 bytes	Totalmente asociativa	FIFO
C3	2 Kb	256 bytes	Asociativa por conjuntos de 4 vías	LRU

13. En aplicaciones multimedia para la reproducción de audio o vídeo es habitual tener cargas de trabajo denominadas de *streaming*, es decir, cargas de trabajo con una gran cantidad de datos que apenas son reusados. Los datos de audio o de vídeo suelen ser *streams* con datos de tamaño igual a 1 byte que codifican por ejemplo un nivel de gris o de cuantificación de una onda sonora. Considera un *stream* de vídeo que accede a 512 KB de datos secuencialmente:

```
0, 2, 4, 6, 8, 10, 12, 14, 16…
```

a) Considera una cache de 64 Kb con asignación directa y bloques de tamaño 32 bytes. ¿Cuál es la tasa de fallo? ¿depende del tamaño de la cache o del tamaño de los datos a los que se accede? Usando el modelo de las 3C, ¿qué tipo de fallos se producen?

b) Recalcula las tasas de fallo considerando bloques de tamaño 16 bytes, 64 bytes y 128 bytes. ¿Qué tipo de localidad se explota en este tipo de trabajo?

La técnica de prefetching (precarga) permite mejorar el acceso a datos cuando el patrón de acceso es predecible. Esta técnica se basa en traer, de forma especulativa, un bloque al que todavía no se ha accedido; por ejemplo, una implementación sencilla consistiría en traer, a un buffer separado, el siguiente bloque adyacente al bloque que se está accediendo. Si el dato que se busca se encuentra en ese buffer se considera un acierto, se introduce en la cache y se precarga el siguiente bloque en el buffer.

c) Considerando que se dispone de un buffer de precarga con capacidad para dos bloques y que la latencia de la cache es menor o igual que el tiempo que necesita nuestro programa para procesar los datos de un bloque de la cache. ¿Cuál sería la tasa de fallos para el *stream* anterior?

14. El siguiente código en alto nivel:

```
for (i=0; i<2; i++)
  for (j=0; j<2; j++)
      {
        ac = C[i+j*2];
        for (k=0; k<2; k++) ac += A[i+k*2]*B[k+j*2];
        C[i+j*2] = ac;
      }
```

se ejecuta en un procesador que dispone de una memoria principal de 1GB con tamaño de palabra de 4 bytes y con tamaño de línea de 8 bytes. El array C se almacena a partir de la dirección 0 de memoria

principal, el array A a partir de la dirección 20 (0x14) y el array B a partir de la dirección 40 (0x28). Se sabe también que cada elemento del array ocupa 4 bytes en memoria. Se pide lo siguiente:

a) Mostrad los accesos a los elementos de los array, junto con su dirección de memoria principal y su número de bloque. Para ello, rellenad una tabla como la de abajo indicando en la columna "dato" el elemento concreto del array que se accede (p.ej, A[3]), la dirección de memoria principal donde se encuentra dicho elemento (dir MP) y número de bloque de memoria principal (bloq MP). Mostradlo en el mismo orden que se solicita en el programa teniendo en cuenta que en cada iteración del tercer bucle FOR, el orden de acceso a los arrays es: `B[k+j*2], A[i+k*2]`.
b) Suponiendo que disponemos de una cache de datos, asociativa por conjuntos de 2 vías (2 bloques por conjunto), reemplazo LRU y tamaño 32 bytes, muestra la evolución del directorio de la cache para dicho código. Para ello, rellena en la tabla la columna "CONJ" con el número del conjunto de la caché donde dicho bloque va a ser alojado, e indica en "A/F" si se produce acierto (A) o fallo (F) de caché y un comentario opcional de lo que ocurre. Muestra también cómo evoluciona el directorio de la cache. Finalmente, calcula el Indice de Fallos (exprésalo en forma fraccionaria)

dato	dir MP	bloq MP	CONJ	A/F

15. El siguiente código en alto nivel:

```
FOR (J=1; J<=2; J++)   A[J]=J;
FOR (J=1; J<=3; J++)   B[J+1]=2*J;
FOR (J=2; J<=4; J++)   A[J+1]=A[J-1]-B[J];
```

se ejecuta en un procesador que dispone de una memoria principal de 1GB con tamaño de palabra de 1 byte y con tamaño de bloque de 2 bytes. El array A se almacena a partir de la dirección 0 de memoria principal y el array B a partir de la dirección 32 (ambos números están expresados en DECIMAL). Se sabe también que cada elemento del array ocupa 1 byte en memoria.

Se pide lo siguiente:

a) Mostrar los accesos a los elementos de los array, junto con su dirección de memoria principal (MP) y su número de línea (bloque). Mostrad los resultados en una tabla similar a la del ejercicio anterior, mostrando el elemento concreto del array que se accede, la dirección de memoria principal donde se encuentra dicho elemento y el número de bloque al que pertenece. Mostradlo en el mismo orden que se solicita en el programa teniendo en cuenta que en cada iteración del tercer bucle FOR, el orden de acceso a los arrays es: `B[J], A[J-1], A[J+1].`
b) Suponiendo que disponemos de una cache de datos, asociativa por conjuntos de 2 vías (2 bloques por conjunto), reemplazo LRU y tamaño 8 bytes, muestra la evolución del directorio de la cache para dicho código. Para ello, incluye en la tabla anterior dos columnas adicionales: una con el número del conjunto de la caché donde dicho bloque va a ser alojado y otra si se produce acierto (A) o fallo (F) de caché. Complétalo con un comentario adicional de lo que va ocurriendo en cada referencia.

Muestra también el estado final en el que queda el directorio de la cache. Finalmente, calcula el Indice de Fallos.

16. Supongamos un sistema de memoria compuesto por 2 bancos de memoria principal entrelazada donde el tiempo de direccionamiento es de 1 ciclo de reloj y un tiempo de transferencia de memoria (tiempo de bus) de 2 ciclos. El tiempo de acceso a los bancos de memoria DRAM es de 20 ciclos para el primer acceso y 8 ciclos para los siguientes. Calcula la penalización por fallo y el ancho de banda de la memoria para los siguientes casos:

    a) Tamaño de bus: 8 bytes y tamaño de bloque: 32 bytes.
    b) Tamaño de bus: 8 bytes y tamaño de bloque: 64 bytes.

17. Calcula el tiempo medio de acceso a memoria (AMAT) en nseg. de un sistema cache donde la frecuencia de reloj es de 500 Mhz, el tiempo de acierto es de 1 ciclo de reloj, la penalización por fallo de 20 ciclos, y donde el índice de aciertos es del 95%.

18. Considera un benchmark que ejecuta 1000 instrucciones en un procesador a 1.2 GHz, del cual 330 instrucciones son load/store. En este programa, el CPI para una cache perfecta es 1.8 (cache perfecta: no hay ningún fallo). Ahora consideremos una cache real con política write-through, en la que la tasa de fallos para la cache de instrucciones (I$) es del 2% y la de la cache de datos (D$) es del 8%. La penalización por fallo es de 10 ciclos para I$ y de 15 ciclos para D$. Calcula:
    a) Los ciclos de detención (stalls) debidos a los fallos
    b) CPI efectivo
    c) Tiempo de ejecución del programa
    d) Tiempo medio de acceso a memoria (AMAT) (hit time: 1 cc)

19. Considera un procesador con política de write-through cuya I$ tiene una tasa de fallos del 0.5%, mientras que para la D$ es del 1.5%, la penalización por fallo de I$ es de 10 ciclos y de 15 para la D$. El CPI base (el de una cache perfecta) es 1.7 y en el código hay un 32% de load&stores. Calcula el CPI efectivo y el AMAT.

20. Considera un programa que ejecuta 2150 instrucciones en un MIPS a 1.5 GHz en que se han implementado todos los cortocircuitos posibles. Sabemos que el 20% de las instrucciones son saltos condicionales (el 30% de los mismos se toman), el 5% son saltos incondicionales, el 15% son una instrucción de carga seguida de una instrucción con una dependencia de datos verdadera con ella, el 10% son loads sin esa dependencia y el 8% son stores (con política write-through). Adicionalmente la tasa de aciertos para I$ es del 99,4% (con 12 cc de penalización por falloy) y para la D$ del 98% (con una penalización por fallo de 15 cc).

    El procesador implementa la técnica de predicción estática de salto no tomado, actualizándose el PC en la etapa MEM para los altos condicionales y en la etapa ID para los incondicionales. Calcula el CPI resultante.

21. Considera un sistema cache con dos niveles L1, L2. La tasa de fallos para L1 es del 3% y para L2 es del 5%. Calcula:
    a) Tasa de fallos global del Sistema de memoria
    b) Si el número total de referencias es de 12.000, calcula el número de
        i) Fallos en L1

ii) Referencias a L2
iii) Fallos en L2

c) Asumiendo, como usualmente, que el hit time para L1 es de 1 cc, el hit time para L2 es de 10 cc y la penalizacón por fallos de L2 es de 50 cc, calcula el AMAT

22. Considera un procesador que trabaja a 1,7 GHz y tiene un sistema cache de dos niveles para el que la tasa de aciertos para L1 es del 90% y para L2 es del 75%. Teniendo en cuenta que el AMAT es de 5 cc y el tiempo de acceso para L2 es de 15 cc, calcula la penalización por fallo de L1 y de L2 (tanto en ciclos como en ns.)