Facultad de Informática
UPV/EHU

Grado en Ingeniería Informática

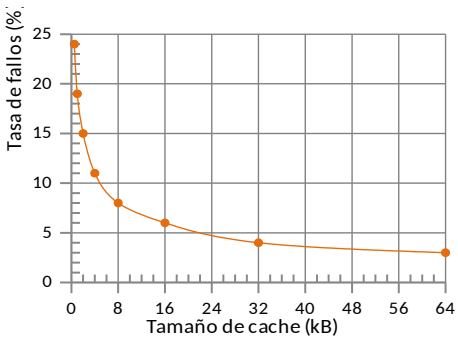
Arquitectura de **C**omputadores

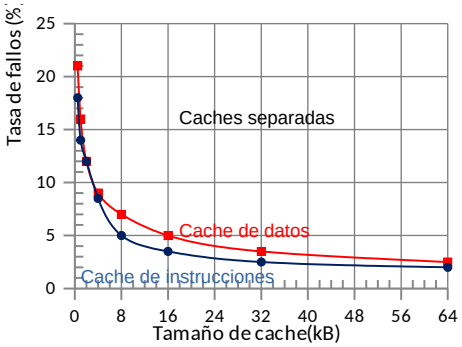
Departamento de Arquitectura y Tecnología de Computadores

Ejercicios Memoria cache

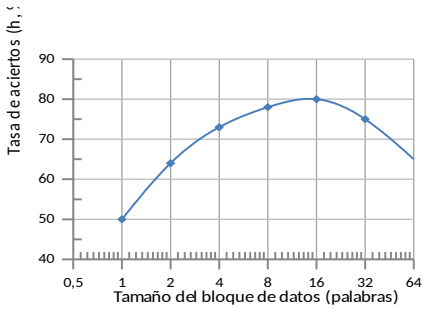
1.1. Las gráficas de este ejercicio muestran la tasa de fallos en función del tamaño de la cache para dos casos diferentes: cache de instrucciones y cache de datos separadas y cache unificada (datos + instrucciones). El porcentaje de referencias a instrucciones ha sido del 60%.

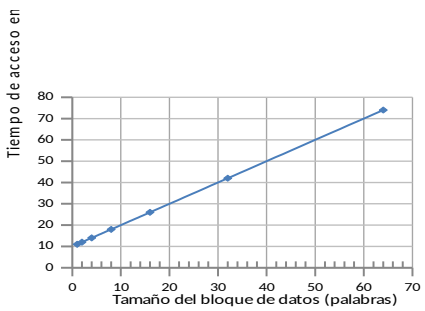
Hay que diseñar un sistema con una memoria cache (total) de 16 KB. De acuerdo a las gráficas, ¿qué opción consideras más adecuada: caches separadas o cache unificada? ¿Por qué?

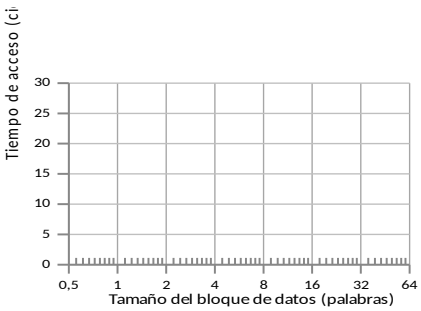




1.2. Las dos gráficas siguientes se han obtenido analizando el comportamiento una MC de 1K. En la primera se muestra la tasa de aciertos en función del tamaño de bloque en la cache (en palabras), mientras que la segunda representa el tiempo de acceso en ciclos al sistema de memoria en caso de fallo en la cache en función del tamaño de bloque (en caso de acierto, consideraremos un ciclo).







a. Justificar el comportamiento de ambas curvas explicando los motivos a los que se debe.

b. En base a ambas curvas, dibuja la curva que representa el tiempo medio de acceso a memoria en función del tamaño de bloque. Calcula de esta manera el tamaño idóneo de bloque.

Datos de las gráficas

Tamaño del bloque de datos:	1	2	4	8	16	32	64	
Tasa de acierto (*h*):	50	64	73	78	80	75	65	(%)
Tiempo de acceso en caso de fallo:	11	12	14	18	26	42	74	(ciclos)
Tiempo medio de acceso:								

1.3. (Ejemplo sencillo para analizar las diferentes estructuras de la memoria cache). Se dispone de una memoria cache con capacidad para 4 bloques de ocho palabras cada uno. Las direcciones de memoria son de 16 bits, se direcciona al byte y las palabras son de 4 bytes. La política de reemplazo es LRU.

a. Indica la estructura en bits de las direcciones físicas para las siguientes configuraciones de memoria cache: correspondencia directa, totalmente asociativa y asociativa por conjuntos con grado de asociatividad 2.

b. Dada la secuencia de referencias físicas indicada (r = read; w = write), rellena la tabla para cada configuración. Indica las acciones a realizar para las dos últimas referencias en los tres casos anteriores suponiendo dos políticas de escritura: a) *write-back* y b) *write-through* (al acertar en escritura no se anula el bloque en cache).

c. Indica una secuencia de referencias (corta, unas 5-10) donde se observe para esta memoria cache

que la política *write-back* ofrece mejores resultados que el caso *write-through* y explica por qué.

		referencias							
Dirección física		**136r**	**200w**	**168r**	**146r**	**140w**	**264r**	**146w**	
Bloque de MP									
Bloque de MC (directa)									
Cjto. de MC (aso. cjtos.)									
Contenido de la MC (bloques de datos)									acciones a realizar en las dos últimas referencias
directa	B0								
	B1								
	B2								
	B3								
totalmente asociativa	B0								
	B1								
	B2								
	B3								
asociativa conjuntos	C0 B0								
	C0 B1								
	C1 B0								
	C1 B1								

1.4. En un sistema de memoria, la memoria cache tiene 4 bloques de 8 palabras. La cache es directa, con los siguientes tiempos de acceso: T_{MC} = 1 ciclo; T_{MP} = 9 ciclos (2 ciclos desde el buffer de entrelazado).

a. En un momento de la ejecución de un programa, la cache tiene los 4 bloques que se indican en la tabla (contenido). Partiendo de este punto, completa la tabla para las cuatro referencias indicadas. Calcula los tiempos de acceso para estas dos políticas: WB y WT (actualizando la MC). Indica también el tráfico habido en el bus de datos.

b. Si las próximas 100 referencias se realizan al bloque 9 y de éstas 25 son escrituras, ¿cuál será su tiempo de acceso para ambas escrituras?

			referencias (bloques)			
		contenido	3w	3	4	4w
CM	B0	12w				
	B1	9				
	B2	10				
	B3	7				
	A / F					
	Ciclos / tráfico(WB)					
	Ciclos / tráfico (WT)					

1.5. Las direcciones para acceder al sistema de memoria de un computador tienen la siguiente estructura:

- MP: entrelazada [13 bit – 2 bit – 1 bit]
- MC: 4 bloques, escritura *write-back* y reemplazo LRU [12 bit – 1 bit – 2 bit – 1 bit]

Tiempos de acceso al sistema: MP, 9 ciclos (1 ciclo desde el buffer de entrelazado); MC, un ciclo.

a. ¿Cuál es la organización de la memoria principal y de la memoria cache? ¿Cómo direcciona la memoria, al byte o a la palabra? ¿Cuál es el tamaño de la palabra? ¿Cuál es el tamaño del bloque?

b. Dada la siguiente secuencia de referencias: {16w, 28r, 48r, 18w, 30r, 36r, 38w, 0r, 42w, 30r, 8r, 16r}, calcula el tiempo de acceso al sistema de memoria. Indica también el tráfico habido en el bus de datos.

c. Si la política de escritura fuera *write-through*, actualizando sólo la MP, ¿qué cambiaría en la primera referencia? Indica el tiempo de acceso para dicha referencia.

1.6. El sistema de memoria de un computador tiene las siguientes características:

- Direccionamiento al byte, con palabras de 8 bytes. Tamaño de bloque de 4 palabras.
- Memoria principal de 128 Kbytes, siendo el nivel de entrelazado el tamaño de bloque. El tiempo de acceso a la memoria principal es de 7 ciclos (1 ciclo desde el buffer de entrelazado).
- Memoria cache de 128 bytes con correspondencia asociativa por conjuntos (2 conjuntos), algoritmo de reemplazo LRU y escritura *write-back*. El tiempo de acceso de 1 ciclo.

a. Representa los esquemas de las direcciones de acceso a memoria cache.

b. Dada la siguiente secuencia de referencias: {544r, 10256r, 10264r, 528w, 10264r, 520w, 1312r, 552w, 1696r, 896r, 1312r, 1728w}, calcula el tiempo de acceso al sistema de memoria.

c. Calcula nuevamente el tiempo de acceso para la secuencia anterior para la política de escritura *write-through* (actualizando la memoria cache).

1.7. Los parámetros de diseño de un subsistema de memoria son los siguientes:

- Memoria principal: 1 MB, entrelazada al tamaño del bloque; tiempo de acceso, 10 ciclos (1 ciclo desde el buffer de entrelazado); direccionamiento al byte con palabras de 4 byte; bloques de 8 palabras.
- Memoria cache: capacidad para 4 bloques; correspondencia asociativa por conjuntos, con 2 conjuntos; reemplazo LRU y escritura *write-back*. Tiempo acceso: 1 ciclo

a. Indica la estructura de los bits de las direcciones físicas.

b. Completa la siguiente tabla para las referencias: 36w, 104, 180, 64, 108w, 36.

		36w	104r	180r	64r	108w	36r
@byte							
@palabra							
Bloque de MP							
Conjunto de MC							
Tag (marca @)							
Conjunto 0	B0						
	B1						
Conjunto 1	B0						
	B1						
acierto / fallo							
ciclos							
Tráfico							

1.8. En un sistema de memoria, la memoria principal (MP) es de 256 MB, el direccionamiento es al byte y las palabras son de 4 bytes. La memoria cache (MC) de datos es de 64 bytes, con bloques de 32 bytes. El tiempo de acceso a MC es de 2 ciclos y el tiempo de acceso a MP es de 15 ciclos (1 desde el buffer de entrelazado).

a. Indica la estructura de los bits de las direcciones físicas (MP y MC, totalmente asociativa y directa).

b. Queremos analizar el comportamiento de 2 configuraciones. Para ello, rellena las dos tablas que tienes a continuación con la secuencia de referencias indicada. Calcula la tasa de aciertos y el tiempo de acceso en ambos casos.

Totalmente asociativa, LRU / WB

byte	**800w**	**400r**	**404w**	**464r**
@ palabra				
Bloque de MP				
Tag (marca @)				
B0				
B1				
A / F				
ciclos				

Directa / WT (actualizar sólo la MP)

@byte	**400w**	**200r**	**204w**	**464r**
@ palabra				
Bloque de MP				
Tag (marca @)				
B0				
B1				
A / F				
ciclos				

1.9. El sistema de memoria de un computador tiene las siguientes características:

- MP: 1 MB, entrelazada al tamaño del bloque; tiempo de acceso: 13 ciclos (2 ciclos desde el buffer de entrelazado); direccionamiento al byte, con palabras de 64 bits; bloque: 32 bytes.
- MC: 128 bytes con correspondencia directa y escritura *write-trhough* (actualizando la MC). El tiempo de acceso a la memoria cache es 1 ciclo.

a. Indica el esquema de bits de la dirección de la memoria principal y de la memoria cache.

b. Después de una serie de referencias, el directorio de la memoria cache tiene las siguientes etiquetas 3, 1, 7 y 4, en el orden de los bloques de la cache. Teniendo en cuenta esta información, rellena la primera columna de la tabla indicando qué bloques se encuentran en la cache y completa la tabla para la secuencia de referencias indicada.

@byte		**400w**	**1008w**	**2024r**	**2032r**
@palabra					
Bloque MP					
Etiqueta (*tag*)					
Bloque de MC					
B0					
B1					
B2					
B3					
A/F					
Ciclos (WT)					
Ciclos (WB)					

c. Si la política de escritura fuera *write-back*, calcula el nuevo tiempo de acceso para la secuencia de referencias anterior, dibuja cómo quedaría el directorio cache al final de esa secuencia indicando el significado de cada componente del mismo

d. La ejecución de un programa, suponiendo memoria cache con política de escritura *write-back*, genera una tasa de fallos del 2% y en estos fallos se reemplaza un bloque modificado un 25% de las veces. ¿Cuál es el tiempo medio de acceso a memoria para dicho programa?

1.10. El sistema de memoria de un computador tiene las siguientes características:

- Direccionamiento al byte, con palabras de 8 bytes; bloques de 4 palabras.
- Memoria principal de 8 MB. El tiempo de acceso a la memoria principal es de 9 ciclos (1 ciclos desde el buffer de entrelazado)
- Memoria cache de datos de 128 bytes y 256 para instrucciones. El tiempo de acceso a la memoria cache es 1 ciclo.

Se quiere ejecutar el siguiente programa, utilizando la matriz `A(2x64)`.

```
for (j=0; j<64; j++)
for (i=0; i<2; i++)
  X = X + A[i][j];
```

La matriz está almacenada por filas en memoria a partir de la dirección 1024, pero la accede por columnas. Cada elemento es de tamaño 1 palabra. Las variables `x, i` y `j` se almacenan en los registros del procesador (no en memoria).

a. Queremos elegir la correspondencia más adecuada para la cache de datos: directa o asociativa por conjuntos. Para ello, simula el comportamiento de las 12 primeras referencias y, a la vista de los resultados obtenidos, calcula la tasa de aciertos para todas las referencias a datos del programa. ¿Qué correspondencia te parece la más adecuada? ¿Por qué?

b. Indica la división en bits de las direcciones físicas para la memoria principal y para la memoria cache elegida en el apartado anterior. Calcula el tiempo de acceso para las primeras 8 referencias a datos.

c. Si el programa hubiera sido la siguiente instrucción `A[i][j]=A[i][j]+5`, ¿qué política de escritura te parece más adecuada para la cache de datos?

1.11. En un sistema de memoria las direcciones físicas son de 8 bits, el direccionamiento al byte y las palabras de 2 bytes. La memoria cache de datos tiene capacidad para 8 bloques y el tamaño de bloque es de 8 bytes. La correspondencia es asociativa por conjuntos y la cache cuenta con 4 conjuntos. La política de escritura es *write-back* y la política de reemplazo LRU (si el bit de control LRU del directorio está a 1, indica que ese bloque ha sido el último en ser accedido).

	libre	modificado	LRU	marca (*tag*)
C0	0	1	0	100
	0	0	1	010
C1	0	1	0	000
	0	0	1	010
C2	0	0	1	010
	0	1	0	111
C3	0	1	1	110
	1	1	0	000

a. Indica los campos que se distinguen en una dirección de memoria y el tamaño en bits de cada uno de ellos.

b. ¿Qué bloques de memoria principal se encuentran en memoria cache?

c. El procesador quiere leer el contenido de la dirección 118. Indica si se trata de un acierto o fallo en memoria cache y justifica tu respuesta. Explica las operaciones que se realizarán debido a esa referencia.

d. Responde a las mismas cuestiones, suponiendo que se quiere escribir en la dirección 78. Indica también cuál será el contenido del directorio tras estas dos referencias.

1.12. En un sistema de memoria, la memoria se direcciona al byte, con palabras de 2 bytes. La memoria principal es de 1024 bytes y está formada por módulos entrelazados. La memoria cache de datos tiene capacidad para 8 bloques, con bloques de 16 bytes. La correspondencia es asociativa por conjuntos y en la cache hay 2 conjuntos. La política de escritura es *write-back* y la política de reemplazo LRU (si los bits LRU del directorio son 00, ese bloque es el menos recientemente referenciado, y si son 11, ese bloque es el último accedido). La siguiente tabla indica el contenido del directorio de la memoria cache.

a. Indica la estructura de los bits de las direcciones físicas (MP y MC).

b. Teniendo en cuenta la información de la tabla, ¿qué bloques de MP están en MC?

c. El procesador quiere escribir en la dirección 182. Indica si es un acierto o un fallo en MC y por qué. Explica las operaciones que se realizarán debido a esa referencia.

d. Responde a las mismas cuestiones, si se quiere leer el contenido de la dirección 684.

e. Por último, tras las dos operaciones anteriores, se quiere escribir en la dirección 4. Indica nuevamente qué sucederá. ¿Cuál será el contenido del directorio tras tratar las 3 referencias anteriores?

	libre	modficado	LRU	*tag*
C0	0	0	10	00101
	0	0	01	00110
	0	0	11	10001
	0	1	00	10011
C1	1	0	00	11111
	0	0	00	00101
	0	1	10	00010
	0	1	01	00001

f. Si el tiempo de acceso a MC es de 1 ciclo y el de MP es de 10 (1 desde el buffer de entrelazado), indica cuánto tiempo requieren las tres referencias anteriores. ¿Y si la política de escritura fuera *write-through* (sin actualizar la MC)?

1.13. En un sistema de memoria, la memoria principal tiene 16 kB, el direccionamiento es al byte y los bloques son de 8 palabras, siendo cada palabra de 4 bytes. La memoria cache tiene capacidad para 8 bloques, divididos en 4 conjuntos. En la tabla se indica el directorio de la memoria cache.

a. ¿Qué bloques están cargados en la memoria cache? ¿Qué política de escritura se está utilizando?

b. El procesador quiere leer en la dirección 3008 de memoria. Indica qué palabra se quiere leer y en qué bloque se encuentra. De acuerdo a la información del directorio, se trata de un acierto o de un fallo? Indica cómo quedará el directorio tras esta referencia.

c. Da una referencia (dirección de memoria) que genere un reemplazo e indica cómo se realizará la operación El reemplazo será aleatorio.

	ocup.	modif.	*tag*
C0	0	1	23 (0010111)
	1	1	18 (0010010)
C1	0	0	0 (0000000)
	1	0	18 (0010010)
C2	0	0	0 (0000000)
	0	0	18 (0010010)
C3	1	1	25 (0011001)
	1	1	6 (0000110)

1.14. Tenemos una cache de 4 bloques, totalmente asociativa. En la tabla se indica el directorio de la memoria cache. Se ejecuta esta siguiente secuencia de referencias: `wr_b8, rd_b44, rd_b8, wr_b6, rd_b100.`

Para cada referencia, indica si se trata de acierto o fallo y cómo va cambiando la información en el directorio. Indica en qué casos se transferirán bloques entre memoria cache y memoria principal.

En relación a la política LRU, 00 indica el bloque menos recientemente referenciado.

ocup.	modif.	LRU	*tag*
1	0	01	6
1	0	10	44
1	1	11	122
0	0	00	18

1.15. En un programa se quieren sumar los elementos de una matriz. El orden de la suma de los elementos es irrelevante, es decir, la suma se puede hacer de cualquiera de estas dos formas:

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
    X = X + A[i][j]
```

```
for (j=0; j<N; j++)
  for (i=0; i<N; i++)
    X = X + A[i][j]
```

En ambos casos, en cada iteración se lee un elemento de la matriz A y se efectúa la suma. La matriz es cuadrada de N×N elementos y está almacenada en memoria por filas.

Teniendo en cuenta:

- MC: 1.024 bloques, totalmente asociativa, acceso 4 ns.
- MP: bloques de 8 palabras, acceso 26 ns (2ns desde el buffer de entrelazado).
- Tiempo de suma 1 ns.

Sin tener en cuenta el resto de los costes, calcula cuantos ciclos se necesitan para ejecutar cada uno de los bucles en estos dos casos: (a) N = 800; y (b) N = 5.000.

1.16. La cache de datos de un sistema de memoria tiene 128 bloques de 4 palabras. La correspondencia es totalmente asociativa, la política de escritura es *write-back* y el reemplazo LRU. El tiempo de acceso a MP es de 12 ciclos (3 desde el buffer de entrelazado) y el tiempo de acceso a memoria cache es de 2 ciclos.

Se quiere ejecutar el siguiente bucle:

```
for (i=0; i<8; i++)
for (j=0; j<200; j++)
  C[i] = A[j] + B[j][i] + C[i];
```

Las matrices y los vectores están almacenadas por filas en memoria a partir de la dirección 0, en el siguiente orden: vector `A[200]`, matriz `B[200][8]` y vector `C[8]`. Cada elemento es del tamaño de una palabra. Calcula el tiempo de acceso a todos los datos del programa y la tasa de aciertos que se consigue.

1.17. La cache de datos de un sistema de memoria es de 2 kB, con bloques de 4 palabras y palabras de 8 bytes. Su correspondencia es totalmente asociativa, la política de escritura es *write-back* y el reemplazo LRU. El tiempo de acceso a memoria principal es de 20 ciclos (3 desde el buffer de entrelazado) y el tiempo de acceso a memoria cache es de 2 ciclos.

Se quiere ejecutar el siguiente bucle:

```
for (i=0; i<64; i++)
for (j=0; j<64; j++) {
  A[i][j] = A[i][j] * B[j][i];
  C[j][i] = A[i][j] * 2;
}
```

Las matrices y los vectores están almacenadas por filas en memoria a partir de la dirección 0, en el siguiente orden: A, B y C. El tamaño de las matrices es de 64x64 elementos (palabras). El código máquina está optimizado para minimizar el número de accesos a memoria.

Calcula el tiempo de acceso a los datos del programa y la tasa de aciertos que se consigue.

1.18. La cache de datos de un sistema de memoria tiene 512 bloques de 8 palabras. Su correspondencia es totalmente asociativa, la política de escritura es *write-through* (actualizando la MC) y el reemplazo LRU. El tiempo de acceso a memoria principal es de 14 ciclos (3 desde el buffer de entrelazado) y el tiempo de acceso a memoria cache es de 2 ciclos.

Se quiere ejecutar el siguiente bucle:

```
for (i=0; i<32; i++)
for (j=0; j<400; j++) {
  C[i] = A[i][j] * B[j];
  C[i] = C[i] + D[j][i];
}
```

Los datos (palabras) están almacenados por filas en memoria a partir de la dirección 0: matriz `A[32][400]`, matriz `D[400][32]` y vectores `B[400]` y `C[32]`.

Calcula el tiempo de acceso a los datos del programa y la tasa de aciertos que se consigue. El compilador genera código optimizado.

1.19. La cache de datos de un sistema de memoria tiene 256 bloques de 8 palabras. Su correspondencia es totalmente asociativa, la política de escritura es *write-back* y el reemplazo LRU. El tiempo de acceso a memoria principal (MP) es de 60 ciclos (2 desde el buffer de entrelazado) y el tiempo de acceso a memoria cache es de 3 ciclos.

Se quiere ejecutar el siguiente bucle.

```
for (i=0; i<200; i++)
for (j=0; j<10; j++)
  A[i][j] = B[j][i] * C[i]
```

Las matrices y los vectores están almacenadas por filas en memoria a partir de la dirección 0, en el siguiente orden: matriz `A[200][10]`, matriz `B[10]200]` y vector `C[200]`. Cada elemento es de tamaño de una palabra. Calcula el tiempo de acceso a los datos del programa y la tasa de aciertos que se consigue. Para calcular el tiempo, ten en cuenta que hay que llevar todos los bloques de A a MP.

Se ha modificado el bucle:

```
for (i=0; i<200; i++) {
  X = C[i];
  for (j=0; j<10; j++)
    A[i][j] = B[j][i] * X;
}
```

Los valores del vector C se van guardando en la variable `X` (supondremos que las variables `X, i` y `j` se almacenan en los registros del procesador). Calcula cuántos ciclos se ahorran en el acceso a los datos ejecutando este nuevo programa.

1.20. Se quiere ejecutar el siguiente programa:

```
for (i=0; i<1024; i++)
{
  C[4*i] = A[i]*A[i] + B[2*i]*B[2*i]
}
```

Valores inciales: R1 = R2 = R3 = 0; R20 = 1024;

```
buc: LD   R8,A(R1)     ;leer A (load)
     MUL  R5,R8,R8     ;R5 := R8 * R8

     LD   R9,B(R2)     ;leer B (load)
     MUL  R6,R9,R9

     ADD  R7,R6,R5
     ST   C(R3),R7     ;escribir C (store)

     ADDI R1,R1,#1
     ADDI R2,R2,#2
     ADDI R3,R3,#4

     SUBI R20,R20,#1
     BNZ  R20,buc      ;branch not zero
```

Supondremos que el direccionamiento se realiza a nivel de byte, los bloques son de 8 palabras y los vectores A, B y C están almacenados desde el comienzo de un bloque. Teniendo en cuenta sólo los accesos a datos, calcula cuál será la tasa de aciertos que se consigue al ejecutar el bucle.

1.21.

1. En un programa se procesa un vector de 2M de elementos. En un primer bucle se suma el primer millón de elementos del vector —A[0]+A[1]+...+A[999999]—, y en un segundo bucle se suman los elementos de las posiciones pares del vector —A[0]+A[2]+...+A[1999998]—. El vector está alineado con los módulos de memoria (el primer elemento está al comienzo de un bloque).

   Teniendo en cuenta estos datos: bloques de 8 palabras; tiempo de suma: 1 ciclo; tiempo de acceso a MC: 3 ciclos y tiempo de transferencia de bloque de MP a MC: 24 ciclos, ¿Cuál será el tiempo de ejecución de esos dos bucles? ¿Será el mismo o distinto? ¿Por qué?

2. Un sistema de memoria de un procesador tiene el nivel L1 separado: 32 kB para datos y 32 kB para instrucciones y en ambos casos la correspondencia es asociativa por conjuntos (asociatividad, 8), las palabras son de 64 bits y los bloques de 64 bytes. Además, tiene una MC de nivel L2.

   ¿En cuántos conjuntos está dividida la cache de datos? En el mejor de los casos, ¿cuántos bloques de datos se pueden cargar en la cache sin tener que reemplazar ninguno? ¿Y en el peor de los casos? ¿Cuál sería la mejor política de escritura para la cache de instrucciones? ¿Cuántas copias de un bloque puede haber en el sistema de memoria y en qué estado?

   El tiempo de acceso a la cache L1, es de 8 ns y la tasa de acierto de 0,95. El tiempo de acceso a la cache L2, es de 25 ns y la tasa de acierto de 0,85. Por último, el tiempo de acceso a la MP es de 80 ns. ¿Cuál es el tiempo de acceso medio al sistema de memoria?

3. En la ejecución de una iteración de un bucle dado, se efectúan 10 accesos a memoria y se realizan varias operaciones con los datos. En los accesos a memoria se necesitan 10 ns en el caso de acierto y 80 ns en el caso de fallo. Por otro lado, para realizar todas las operaciones aritméticas, 70 ns.

Si la tasa de acierto es de 0,9, ¿cuánto tiempo se necesita para ejecutar 1.000 iteraciones del bucle? ¿Si conseguimos mejorar la tasa de acierto en un 5%, cuánto se reducirá (%) el tiempo de ejecución?

Repite este ejercicio suponiendo que para el cálculo de las operaciones se necesitan 700 ns. Extrae conclusiones.

4. Una MC con correspondencia asociativa por conjuntos tiene 32 conjuntos de 10 bloques. Los bloques son de 8 palabras. En la cache se ha cargado un vector de 1.024 palabras. ¿Cuántos bloques de datos del vector habrá en cada conjunto? ¿Qué elementos del vector estarán en el conjunto C5? Supón que la dirección del primer bloque de datos del vector corresponde al conjunto C0.

5. En un programa dado la tasa de acierto de la memoria cache es *h*. Se ha revisado el programa, se ha optimizado el código, y en consecuencia, se ha reducido el tiempo de acceso aunque también se haya reducido la tasa de aciertos *h*' < *h*. ¿Es posible esta situación? ¿Por qué?

6. El sistema de memoria de un computador tiene los siguientes tiempos de acceso:

T_{cache}: 5 ns T_{MP} (palabra): 15 ns T_{MP} (transferencia de un bloque): 29 ns

Queremos comparar dos opciones a la hora de ejecutar un programa:

a. No se utiliza memoria cache, por lo que todas las operaciones se hacen en la memoria principal.
b. Se utiliza memoria cache. La política de escritura es *write-back*. Para simplificar, no tendremos en cuenta el tiempo de reemplazo de bloques modificados para actualizar la memoria principal.

Calcula la tasa de acierto mínima para que sea rentable el uso de la MC, es decir, que el tiempo medio de acceso al sistema de memoria sea menor utilizando MC frente al caso de utilizar sólo la MP. ¿Cuál será el tiempo de acceso medio si la tasa de acierto es 0,95?

7. Ten en cuenta una memoria cache con correspondencia directa. En las siguientes tablas se indica la evolución del directorio de la memoria cache, tras la ejecución de una secuencia de referencias. Indica qué bloques están en memoria cache al comienzo de la ejecución, y qué operación se está ejecutando en cada paso para ser coherente con la evolución del directorio.

ocup.	mod.	*tag*
1	0	32
0	0	1
1	1	10
1	0	43

comienzo

ocup.	mod.	*tag*
1	0	32
0	0	1
1	1	10
1	1	43

1ª operación

ocup.	mod.	*tag*
1	0	32
0	0	1
1	0	14
1	1	43

2ª operación

ocup.	mod.	*tag*
1	0	32
0	0	1
1	0	14
1	1	43

3ª operación

8. El rendimiento de las memorias cache viene ligado al principio de localidad, que en un programa se puede dar de dos formas diferentes. Al ejecutar un programa dado se produce esta secuencia de referencias (direcciones) a memoria (con palabras de 4 bytes): 104, 108, 112, 116, 120, 550, 554, 558, 108, 112, 116... Explica brevemente cada uno de los dos tipos de localidad y, teniendo en cuenta la secuencia anterior, da un ejemplo de cada tipo que se pueda ver en la secuencia.

9. La memoria cache de un procesador es de 32 kB. El tamaño del bloque es de 64 bytes, la cache es asociativa por conjuntos con grado de asociatividad 4. En la dirección de memoria, la dirección de bloque es de 16 bits. Teniendo en cuenta estos datos, ¿qué bits de la dirección de bloque representan el campo *tag* que debe guardarse en el directorio de la cache? Utiliza a modo de ejemplo el bloque b11377.

10. El sistema de memoria de un procesador utiliza bloques de 16 palabras. Antes de reemplazar un bloque, se realizan 16 lecturas y 8 escrituras a dicho bloque. Los tiempos de acceso al sistema de memoria son los siguientes: T_{MC} = 2 ciclos; T_{MP} = 16 ciclos (2 ciclos desde el buffer de entrelazado).

Ten en cuenta estas dos políticas de escritura: *write-through* (WT, actualizando la MC) y *write-back* (WB). En el ejemplo anterior, ¿cuántas veces más rápido será el acceso al sistema de memoria si la escritura es WB en lugar de WT?

11. POWER9 (2019) es un procesador de alta gama, que se utiliza generalmente en sistemas paralelos (por ejemplo, en el supercomputador Mare Nostrum). El chip tiene 8.000 millones de transistores y, en algunas versiones (hay varias) tiene 24 núcleos (cores), con una memoria cache de 3 niveles:

L1: 32 kB (datos) + 32 kB (instrucciones); asociativa por conjuntos, con 8 bloques por conjunto (8 *way*); una cache L1 por núcleo.

L2: 256 kB (datos e instrucciones); asociativa por conjuntos, con 8 bloques por conjunto; una cache L2 por núcleo.

L3: 120 MB (datos e instrucciones); asociativa por conjuntos, con 20 bloques por conjunto; una cache compartida por todos los núcleos.

Para simplificar, supongamos que se utilizan 4 transistores para implementar un bit de cache. Calcula el porcentaje de transistores del chip ocupados para implementar el sistema de cache.

12. La memoria cache de un procesador es asociativa por conjuntos, con conjuntos de 4 bloaues. La escritura es *write-back* y el reemplazo LRU. La siguiente secuencia de referencias (bloque) van al mismo conjunto de cache:

`b16r`, `b48r`, `b16w`, `b144r`, `b48r`, `b16w`, `b80r`, `b48r`, `b80r`, `b80r`, `b144r`, `b80r`, `b80r`

A continuación se accede al bloque `b336`. Explica si hay que reemplazar o no un bloque de MC; en caso de reemplazo, qué bloque de MC se reemplaza y por qué.

13. Unas preguntas sobre cuestiones más teóricas.

- ¿Qué es una memoria asociativa? ¿Para qué y cómo se utiliza en las memorias cache? Pon un ejemplo.
- En un computador los bloques son de 8 palabras. El procesador lee una palabra y se produce fallo en la memoria cache. Esa palabra es la primera de un bloque. Más adelante, se produce fallo al leer una nueva palabra, que resulta ser la última palabra de un bloque. En ambos casos, el procesador se queda pendiente de que se soluciones el fallo. Desde el punto de vista del procesador, ¿en qué situación debe esperar más tiempo para continuar con la ejecución del programa? ¿Por qué? ¿Cuánto tiempo? Pon un ejemplo numérico.
- En el contexto de memoria cache, ¿qué es un buffer de escritura? ¿Para qué se utiliza?

14. Un sistema de memoria tiene las siguientes latencias: MP, 20 ciclos (2 ciclos desde el buffer de entrelazado); MC: 1 ciclo. Los bloques son de 8 palabras y el rejol del computador es de 2 GHz. Desde el punto de vista del sistema de memoria, indica cuánto tiempo (ns) es necesario para realizar estas operaciones de lectura: (a) acierto en cache; (b) fallo en cache; (c) fallo con reemplazo de un bloque en la cache (política de escritura *write-back*).

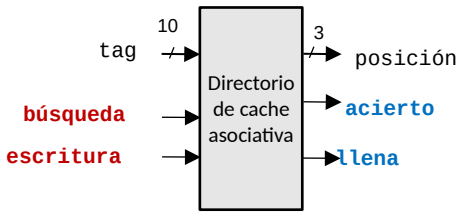
15. Una memoria cache tiene un tamaño de 4 bloques, repartidos en 2 conjuntos. La escritura es *write-back* y el reemplazo LRU. Al ejecutar un programa se han utilizado los siguientes bloques (en el orden indicado): `b155r`, `b124w`, `b1000r`.

Indica el contenido del directorio de la memoria cache tras utilizar los tres bloques anteriores con estos campos: ocup. - mod. - LRU - *tag*. A continuación, indica una referencia a un bloque que produzca reemplazo en la cache, a qué bloque reemplaza e indica los pasos a seguir para completar dicha referencia.

16. En el directorio de una MC se guardan *tags* o etiquetas, es decir, información para identificar los bloques de datos que están en la cache. ¿Es posible que en un momento dado haya dos *tags* iguales en el directorio? En caso negativo, di el por qué y en caso afirmativo indica en qué casos.

17. Una cache totalmente asociativa tiene 8 bloques; el directorio guarda esta información por cada bloque de datos: *tag* (10 bits) y ocup. (1 bit).

Diseña el directorio con estos componentes digitales básicos: 8 registros de 10 bits (*tags*) y 8 biestables JK (ocup.); 2



codificadores (8:3) y un descodificador (3:8); comparadores, y puertas lógicas.
Hay que realizar estas operaciones:

a. Búsqueda. Dado un tag de 10 bits, hay que responder si el bloque está o no en la cache, `acierto`, y dónde, `posición`.
b. Escritura. Hay que escribir un *tag* en una posición libre; si no hay sitio, hay que activar la salida `llena`.

18. El sistema de memoria de un procesador está organizado en 2 niveles, en función de estos parámetros en promedio. Calcula el tiempo promedio para acceder a memoria.

- tiempo de acceso: MC: 8 ns MP: 50 ns
- tasa de acierto: MC: 0,95 MP: 1

Para reducir el tiempo de acceso a memoria, se quiere añadir otro nivel de cache "más cercano" al procesador, más rápida. Se estima que la tasa de aciertos en ese nuevo primer nivel será 0,85. Calcula la latencia de la nueva cache L1, si queremos llegar a la mitad de tiempo de acceso del sistema anterior.

19. En un sistema de memoria cache de dos niveles (L1 y L2), las tasas de acierto de un programa dado son 0,9 y 0,95, en ese orden. ¿Qué porcentajes de accesos a memoria se cubren en la cache L1, la L2 y la MP? En promedio, el costo de los accesos a memoria ha sido de 3 ns. Los accesos de la cache L1 son de 2 ns y los de la cache L2, de 10 ns. ¿Cuál es el tiempo de acceso a MP?

20. En un procesador dado, los bloques de cache son de 128 bytes. En dicho procesador ejecutamos estos dos bucles; en el primer caso los datos son `float` de 4 bytes y en el segundo, `double` de 8 bytes. Calcula en cada caso las tasas de acierto en los accesos a cache.

```
for (i=0; i<N; i++)              for (i=0; i<N; i=i+2)
  A[i] = A[i] + B[2*i];            B[i] = B[i] + A[i] + C[2*i];
```

21. Para ver la influencia de las estructuras de las caches en la tasa de aciertos, toma en consideración este ejemplo simplificado. En una aplicación dada, se usan estos bloques en el orden indicado: b0, b0, b3, b3, b4, b4, b0, b0, b3, b3, b8, b8, b4, b4, b3, b3, b0, b0. La cache es pequeña, de 4 bloques, y organizada en 3 maneras: cuatro conjuntos de un bloque, dos conjuntos de dos bloques y un conjunto de cuatro bloques. Calcula la tasa de aciertos de esa secuencia de referencias para cada caso.

Los fallos se suelen clasificar de tres tipos: carga (*compulsory*), capacidad (*capacity*) y corerspondencia (*conflict*). Para el segundo caso anterior (dos conjuntos de dos bloques), indica un ejemplo de cada tipo de fallos.

1.22. La utilización apropiada de la cache puede tener una gran consecuencia en el tiempo de ejecución de los programas, tal y como se puede apreciar en estos tres ejemplos. Se han ejecutado en una máquina con las características que se indican, con el fin de conseguir los tiempo de ejecución. Explica las razones de los tiempos de ejecuión que se logran.

dif-cluster.si.ehu.es — PowerEdge R740 (DELL) — 2 Intel Xeon Gold 6130, de 16 núeos (Skylake), 2,1 GHz.
MP: 32 GB RAM (RDIMM - 2666 MT/s).
MC (bloque: 64 B, *writeback*): L1, 32+32 kB, 8 *way* (x16); L2, 1 MB, 16 *way* (x16); L3, 22 MB (16x1,375 MB), 11 *way*.
Compilador: `gcc`, 7.2.0, con optimización `-O2`. SO: Linux 7.4.1708.

ejemplo 1

```
/*
    AC, Arquitectura de Computadores – Ingeniería Informática, curso 2.

    cache1.c       utilización de cache
    ejemplo 1: efecto del stride o paso de un vector
*****************************************************************/

#include <stdio.h>
#include <time.h>

#define N1 16000000
#define N2 1000000

double  A[N1];

void TiempoEjec (...)
{
  ...
}


void main ()
{
 int  i;
 struct timespec  t0, t1, t2, t3, t4;

 for (i=0; i<N1; i++) A[i] = 1.0;

clock_gettime (CLOCK_REALTIME, &t0);

  for (i=0; i<N2; i++)
    A[i] = A[i] + 1.0;            // stride o paso = 1 (uno a uno)

clock_gettime (CLOCK_REALTIME, &t1);

  for (i=0; i<N2; i++)
    A[2*i] = A[2*i] + 1.0;        // s = 2

clock_gettime (CLOCK_REALTIME, &t2);

  for (i=0; i<N2; i++)
    A[4*i] = A[4*i] + 1.0;        // s = 4

clock_gettime (CLOCK_REALTIME, &t3);

  for (i=0; i<N2; i++)
    A[8*i] = A[8*i] + 1.0;        // s = 8

clock_gettime (CLOCK_REALTIME, &t4);

  TiempoEjec (" Tex (s=1)", &t0, &t1);
  TiempoEjec (" Tex (s=2)", &t1, &t2);
  TiempoEjec (" Tex (s=4)", &t2, &t3);
  TiempoEjec (" Tex (s=8)", &t3, &t4);
}
```

```
$ cache1

Tex (s=1) =  1.1 ms
Tex (s=2) =  2.1 ms
Tex (s=4) =  3.5 ms
```

Tex (s=8) = 7.1 ms

ejemplo 2

```
/*
    AC, Arquitectura de Computadores – Ingeniería Informática, curso 2.

    cache2.c       utilización de cache
    ejemplo 2: reutilizar datos (localidad)
*****************************************************************/

#define N1 1000000

double  A[N1], B[N1], C[N1];

...

void main ()
{
  int  i;
  struct timespec  t0, t1, t2;


clock_gettime (CLOCK_REALTIME, &t0);

  for (i=0; i<N1; i++)
    A[i] = B[i] * B[i];

  for (i=0; i<N1; i++)
    C[i] = B[i] + A[i];

clock_gettime (CLOCK_REALTIME, &t1);

  for (i=0; i<N1; i++)
  {
    A[i] = B[i] * B[i];
    C[i] = B[i] + A[i];
  }

clock_gettime (CLOCK_REALTIME, &t2);

  TiempoEjec (" T1 – dos bucles ", &t0, &t1);
  TiempoEjec (" T2 – un bucle   ", &t1, &t2);
}
```

```
$ cache2

T1 - dos bucles  = 6.2 ms
T2 - un bucle    = 1.9 ms
```

ejemplo 3

```
/*
    AC, Arquitectura de Computadores – Ingeniería Informática, curso 2.

    cache3.c       utilización de cache
    ejemplo 3: minimizar accesos a memoria
*****************************************************************/

#define N1 4000

double  B[N1], A[N1][N1], x;

...

void main ()
{
  int  i, j;
  struct timespec  t0, t1, t2;

clock_gettime (CLOCK_REALTIME, &t0);

  for (i=0; i<N1; i++)
  for (j=0; j<N1; j++)
    A[i][j] = 1.0 + B[i];

clock_gettime (CLOCK_REALTIME, &t1);
```

```
  for (i=0; i<N1; i++)
  {
    x = B[i];
    for (j=0; j<N1; j++)
      A[i][j] = 1.0 + x;
  }

clock_gettime (CLOCK_REALTIME, &t2);

  TiempoEjec (" T1   ", &t0, &t1);
  TiempoEjec (" T2(x)", &t1, &t2);
}
```

```
$ cache3

T1    = 36.7 ms
T2(x) = 16.5 ms
```