Jerarquía de memoria y memoria caché
Ejercicios resueltos

Ejercicio 1. Sea un computador de 32 bits con una memoria caché de 256 KB, líneas de 64 bytes y un tiempo de acceso de 5 ns. La caché es asociativa por conjuntos de 4 vías y se emplea la política de reemplazo LRU. Se pide:

a) Indique el número de líneas y de conjuntos de la memoria caché del enunciado.
b) ¿Cuál es el tamaño de los bloques que se transfieren entre la memoria caché y la memoria principal?
c) Si el tiempo para transferir un bloque de memoria principal a caché es de 200 ns, indique la tasa de aciertos a caché necesaria, de forma que el tiempo medio de acceso al sistema de memoria sea de 20 ns.

Solución:

a) La caché tiene un tamaño de 256 KB = 2^{18} bytes. como cada línea tiene 2^6 bytes, el número de líneas es 2^{18} bytes / 2^6 bytes = 2^{12} líneas = 4096 líneas. Como la caché es asociativa por conjuntos de 4 vías, cada conjunto tiene cuatro líneas, por tanto el número de conjuntos es 4096 / 4 = 1024 conjuntos.
b) El tamaño del bloque que se transfiere entre memoria principal y caché coincide con el tamaño de la línea, es decir, con 64 bytes.
c) El tiempo medio de acceso al sistema viene dado por la siguiente expresión:

$$tm = tc \cdot Pa + (1-Pa) \cdot tf$$

donde tc = 5 ns, tm = 20 ns y tf = 205 (200 + 5). Por tanto:

$$20 = 5 \cdot Pa + (1-Pa) \cdot 205$$

Despejando, se obtiene que Pa = 185 / 200 = 0,92. Es decir, la tasa de aciertos debe ser del 92 %.

Ejercicio 2. Se dispone de un computador con direcciones de memoria de 32 bits, que direcciona la memoria por bytes. El computador dispone de una memoria caché asociativa por conjuntos de 4 vías, con un tamaño de línea de 4 palabras. Dicha caché tiene un tamaño de 64 KB. El tiempo de acceso a la memoria caché es de 2 ns y el tiempo necesario para tratar un fallo de caché es de 80 ns. Indique de forma razonada:

a) Tamaño en MB de la memoria que se puede direccionar en este computador.
b) Número de palabras que se pueden almacenar en la memoria caché.
c) Número de líneas que se pueden almacenar en el mismo conjunto.
d) Número de líneas de la caché.
e) Número de conjuntos de la caché.
f) Indique la tasa de aciertos necesaria para que el tiempo medio de acceso al sistema de memoria de este computador sea de 10 ns.

Solución:

a) Si el computador tiene direcciones de memoria de 32 bits, el tamaño de memoria principal será 2^{32} bytes = 2^{12} MB
b) Tamaño de la cache = 64 KB
Como el computador es de 32 bits, las palabras son de 32 bits, 4 bytes, por tanto el número de palabras que puede almacenar la cache es de 64 KB / 4 bytes = 16 * 1024 palabras.
c) En cada conjunto se pueden almacenar 4 líneas.
d) El tamaño de la línea es de 4 palabras = 4 *4 = 16 bytes. El número de líneas = 64 KB / 16 bytes = 2^{16} / 2^4 = 2^{12} = 4096 líneas
e) El número de conjuntos de la caché será el número de líneas caché dividido por el número de vías de cada conjunto = 4096 líneas / 4 vías = 1024 conjuntos
f) Tm = Ta * T_cache + (1-Ta) * T_fallo => 10 = Ta*2 + (1-Ta) * 80 => despejando Ta = 70 / 78, es decir, una tasa de aciertos del 89 %

Ejercicio 3. De acuerdo a un estudio realizado sobre la utilización de las instrucciones de este computador se ha determinado que en media ejecuta 50 millones de instrucciones por segundo y que el porcentaje de utilización de sus instrucciones es el siguiente:

LOAD	un 30 %
STORE	un 10 %
MOVE	un 10 %
Operaciones aritméticas	un 24 %
Operaciones lógicas	un 6 %
Bifurcaciones	un 20 %

Se pide:

a) Determine el número de accesos a memoria por segundo que se realizan en este computador.
b) ¿Cuál será el número de accesos a memoria principal por segundo en caso de utilizar una memoria cache con un tamaño de línea de 8 palabras, política de actualización write-through (escritura inmediata) y una tasa de aciertos del 95 %?

Solución:

a) El computador ejecuta 50 millones de instrucciones por segundo. La ejecución de cada instrucción supone un acceso a memoria de lectura para la lectura de la propia instrucción. Además, de todas estas, el 40 % (correspondientes a las instrucciones LOAD y STORE) implican un acceso adicional a memoria principal. Por lo tanto, en un segundo se realizan 50 millones de accesos a memoria por segundo más 20 millones (un 40% de 50) de accesos para las instrucciones LOAD y STORE.

   El número total de accesos a memoria es de 70 millones de accesos por segundo.

b) De todos los 70 millones de accesos a memoria hay que determinar cuántos son de lectura y cuántos de escritura. 50 millones son lecturas de las instrucciones en sí. De los 20 millones de accesos a memoria correspondientes a las instrucciones LOAD y STORE, el 75% (3 de cada 4) corresponden a lecturas de la instrucción LOAD y el 25 % restante a escrituras correspondientes a instrucciones STORE, es decir, 15 millones de accesos son de lectura y 5 millones de acceso por segundo son de escritura.

   El número total de lecturas por segundo = 50 +15 = 65 millones de accesos de lectura por segundo. El número total de escrituras por segundo = 5 millones de accesos de escritura por segundo.

   Del total de los accesos de lectura, el 5% representan fallos que deberán ser resueltos mediante su correspondiente acceso a memoria principal. Por lo tanto el número de fallos por segundo a memoria principal para las lecturas será de 0,05 * 65 = 3,25 millones. Recuérdese que cada fallo de lectura en memoria cache supone la lectura de un bloque completo, que en este caso es de 8 palabras, por tanto, el número de accesos que se producen a memoria principal como consecuencia de los fallos de lectura = 8 *3,25 = 26 millones de accesos por segundo

   Del total de los accesos de escritura, todos suponen accesos a memoria principal debido a que se utiliza una política de escritura inmediata. Suponiendo que el bloque que provoca el fallo se modifica en el nivel inferior (memoria principal) y no se carga en la cache, todas las escrituras acceden a memoria principal. Por tanto, el número de accesos de escritura será de 5 millones de accesos por segundo.

   El número total de accesos a memoria principal por segundo = 5 + 26 = 31 millones de accesos por segundo.

Ejercicio 4. Sea un computador de 32 bits con una memoria caché para datos de 32 KB y líneas de 64 bytes. La caché es asociativa por conjuntos de 2 vías y se emplea la política de reemplazo LRU. Dado el siguiente fragmento de código:

```
int v[262144]

for (i = 0; i < 262144; i = i + 2)
  v[i] = 9;
```

se pide:

a) Indique el número de líneas y de conjuntos de la memoria caché de datos del enunciado.
b) Considerando exclusivamente la caché de datos y los accesos al vector, calcule de forma razonada la tasa de fallos que se obtiene en la ejecución del bucle anterior.

Solución:

c) La caché tiene un tamaño de 32 KB = 215 bytes. como cada línea tiene 26 bytes, el número de líneas es 215 bytes / 26 bytes = 29 líneas = 512 líneas. Como la caché es asociativa por conjuntos de 2 vías, cada conjunto tiene dos líneas, por tanto el número de conjuntos es 512 / 2 = 256 conjuntos
d) El patrón de accesos del vector al bucle es el siguiente:

v[0], v[2], v[4], v[6], v[8] …

es decir, se accede cada 2 elementos. Como la caché está estructurada en líneas de 64 bytes y cada dato de tipo entero (int) ocupa 4 bytes, en cada línea caben 16 elementos del vector. Como el vector se recorre de forma secuencial y de estos 16 solo se acceden realmente a 8, la tasa de fallos es 1/8.

Ejercicio 5. Sea un computador de 32 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 64 bits. Sobre este computador se desea ejecutar el siguiente fragmento de código:

```
int v[200];
int i;
int s;

for (i=0; i < 200; i++)
      s = s + v[i];
```

Se pide:

a) Indique de forma razonada el número de líneas y conjuntos de la caché.
b) Si se considera que la caché se encuentra vacía y que los valores de las variables i y s del código anterior se almacenan en registros, indique, considerando solo los accesos al vector v, la tasa de aciertos que se obtiene al ejecutar el fragmento de código anterior.

Solución:

a) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 64 bits = 8 bytes = 2^3 bytes. El número de línea viene dado por 2^{19} bytes / 2^3 bytes = 2^{16} líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{16} / 2^2 = 2^{14} conjuntos

a) Como cada elemento del vector es un entero de 4 bytes, en cada línea caben dos enteros. Cada vez que se accede a un elemento se produce un fallo, y se traen a caché el elemento que ha producido el fallo y el siguiente, que da lugar a un acierto en la siguiente vuelta del bucle. Por tanto, hay un fallo cada dos accesos y la tasa de aciertos es del 50 %.

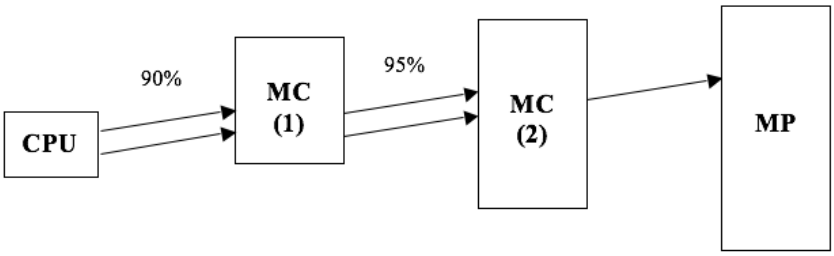
Ejercicio 6. Se dispone de un sistema con una memoria caché de 2 niveles. En la ejecución de una determinada aplicación, la tasa de aciertos de la caché de nivel 1 es del 90% y la tasa de aciertos de la caché de nivel 2 es del 95%. La aplicación genera durante su ejecución un millón de accesos a memoria. Indique de forma razonada:

a) El número de accesos que se genera a la caché de nivel 1.
b) El número de accesos que se genera a la caché de nivel 2.
c) El número de accesos que se genera a memoria principal.

Solución:

a) El número de accesos con acierto en MC(1) es: **10^6**
b) El número de accesos con acierto en MC(2) es: **10^6 * 0,1**
c) El número de accesos con acceso a MP por fallos de MC1 y MC2: **10^6 * 0,1 * 0,05**

La siguiente figura ilustra el funcionamiento.



Ejercicio 7. Sea un computador de 32 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 128 bytes. Sobre este computador se desea ejecutar el siguiente fragmento de código:

```
int a1[200];
int a2[200]:
int i;
int s;

for (i=0; i < 200; i++)
      s = s + a1[i] + a2[i];
```

Se pide:

a) Indique de forma razonada el número de líneas y conjuntos de de la caché.
b) Si se considera que la caché se encuentra vacía y que los valores de las variables `i` y `s` del código anterior se almacenan en registros, indique, considerando solo los accesos al vector `a1 y a2`, la tasa de aciertos que se obtiene al ejecutar el fragmento de código anterior.

Solución:

a) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 128 bytes = 2^7 bytes. El número de líneas viene dado por 2^{19} bytes / 2^7 bytes = 2^{12} líneas = 4096 líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{12} / 2^2 = 2^{10} conjuntos = 1024 conjuntos

b) Como cada elemento del vector es un entero de 4 bytes, en cada línea caben 128 / 4 = 32 enteros. Cuando se accede por primera vez a `a1[0]` se produce un fallo y se trae a caché una línea que incluye a los elementos: `a1[0], a1[1], ... , a1[31]`. Lo mismo ocurre para el vector `a2`. Cuando se acceden a los elementos `a1[1], a2[1], ... , a1[31], a2[31]`, se producen aciertos.
Cada 32 accesos hay un fallo. Este proceso se repite 6 veces, hasta llegar al elemento 192. Para los últimos 8 elementos del vector, hay un fallo y 7 aciertos. Por tanto el número de accesos por cada vector es de 200 y el número de fallos es de 7, luego la tasa de aciertos es 193 /200 =96,5%

Ejercicio 8. Sea un computador de 64 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 128 bytes. El tiempo de acceso a la memoria caché es de 2 ns y el tiempo de penalización de fallo es de 120 ns. Sobre este computador se desea ejecutar el siguiente fragmento de código:

```
int m[32][32];          // matriz de enteros de 32 x 32 elementos.
                        // los elementos de m se almacenan por filas.


for (i=0; i < 32; i = i + 1)
      for (j = 0; j < 32; j = j + 2)
            s = s + m[i][j];
```

Se pide:

a) Indique de forma razonada el número de líneas y conjuntos de de la caché.

b) Si se considera que la caché se encuentra vacía y que los valores de las variables `i, j` y `s` del código anterior se almacenan en registros, indique, considerando solo los accesos a la matriz `m,` la tasa de aciertos que se obtiene al ejecutar el bucle anterior.
c) Calcule el tiempo total invertido en el acceso a los datos del bucle anterior.

Solución:

a) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 128 bytes = 2^7 bytes. El número de líneas viene dado por 2^{19} bytes / 2^7 bytes = 2^{12} líneas = 4096 líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{12} / 2^2 = 2^{10} conjuntos = 1024 conjuntos

b) Como cada elemento de la matriz es un entero de 8 bytes (puesto que el computador es de 64 bits), en cada línea caben 128 / 8 = 16 enteros. Cuando se accede por primera vez a `m[0][0]` se produce un fallo y se trae a caché una línea que incluye a los elementos: `m[0][0], m[0][1], ... , m[0][15]`. Es decir que cada 8 accesos se produce 1 fallo. Este proceso se repite (32 x 16) / 8 = 64 veces. Por tanto el número total de fallos es de 64 fallos y el de aciertos es 512 – 64 = 448.

c) El tiempo de acceso es = 64 * 120 + 448 * 2 = 8576 ns.

Ejercicio 9. Considere el siguiente fragmento de código:

```
for (i=0; i<64; i++) {
      for (j=0; j<1024; j=j+2 ){
          v[i][j] = v[i][j] * v[i][j+1] +  b[i];
      }
}
```

Dicho código se ejecuta en una arquitectura con un ancho de palabra de 8 bytes. La arquitectura consta de una caché de tamaño 256 KB, con tamaño de bloque de 64 bytes. La memoria caché es totalmente asociativa y utiliza un algoritmo de reemplazo LRU. En el fragmento de código anterior `v` representa una matriz de números reales de 8 bytes, que se almacena por filas (estilo C) de tamaño 64x1024; y `b` representa un vector de 64 números reales de 8 bytes. Se supondrá que las variables índice se almacenan en registros y que al empezar el código la caché está vacía. Considere que en cada iteración del bucle interior, primero se accede al elemento `v[i][j]` y después al elemento `v[i][j+1]`.

Se pide:
a) Indique el número de fallos de caché que se producen al ejecutar el fragmento de código anterior.
b) La tasa media de fallos.
c) Si el tiempo medio de acceso al sistema de memoria es de 6 ns y el tiempo de acceso a cache de 4 ns, calcule el tiempo necesario para tratar un fallo de caché.

Solución:

a) La caché tiene un tamaño de 256KB = 2^{18} bytes = 2^{12} líneas

En cada bloque de 64 bytes se pueden almacenar 64/8 = 8 elementos de la matriz v y del vector b.

En primer lugar, se van a considerar los accesos a los elementos de v:

- Iteración i = 0;
  - Lectura de v[0][0] Fallo se trae a cache v[0][0]… v[0][7]
  - Lectura de v[0][1] Acierto
  - Escritura de v[0][0] Acierto
  - Lectura de v[0][2] Acierto
  - Lectura de v[0][3] Acierto
  - Escritura de v[0][2] Acierto
  - Lectura de v[0][4] Acierto
  - Lectura de v[0][5] Acierto

- Escritura de v[0][4] Acierto
- Lectura de v[0][6] Acierto
- Lectura de v[0][7] Acierto
- Escritura de v[0][6] Acierto

- Lectura de v[0][8] Fallo se trae a cache v[0][8]… v[0][15]
- Lectura de v[0][9] Acierto
- Lectura de b[0] Acierto
- Escritura de v[0][8] Acierto
- …. Se repite el patrón de accessos

En cada iteración del bucle externo, se realizan 512 * 3 = 1536 accesos a los elementos de v. Cada 12 accesos se produce un fallo. En cada iteración del bucle externo hay, por tanto, 1536/12 =128 fallos y 1408 aciertos a los elementos de v. Como hay 64 iteraciones del bucle externo, habrá 64 * 128 = 8192 fallos y 90112 aciertos en los accesos a la matriz v.

En cuanto a los elementos de b, hay 1 fallo cada 8 iteraciones del buce externo. Se producen fallos en los accesos a los elementos b[0], b[7], b[15]… Por tanto hay 64 / 8 = 8 fallos. El número de accesos a los elementos de b es de 64 * 512 = 32768, de los cuales son 8 son fallos.
Por tanto, el número de accesos totales es: 64 * 512 * 4 = 131072 y el número de fallos es 8192 + 8 = 8200.

b) La tasa media de fallos es 8200 /131072=0,0625 => 6,25%
La tasa media de aciertos es 1 – 0,0625 = 0,9375 => 93,75 %

c) TiempoMedio = h * Tc + (1-h) * Tf
siendo h la tasa de aciertos, Tc el tiempo de cache y Tf el tiempo para tratar un fallo.

6 = 0,9375 * 4 + 0,0625 * Tf
6 = 3,75 + 0,0625 * Tf
Tf = (6-3,75) / 0,0625 = 36 ns, es el tiempo empleado en tratar un fallo.

Ejercicio 10. Sea un computador de 32 bits cuyo sistema de memoria como el que se describe a continuación:

- Una **memoria principal** de tipo DRAM de 512 Mbytes y un tiempo de acceso de 200 ns, incluida la gestión del fallo de caché.
- Una **memoria caché** con las siguientes características:
  - Tamaño: 16 Kbytes
  - Tamaño de la línea: 64 bytes.
  - Función de correspondencia: asociativa por conjuntos de 4 vías
  - Política de reemplazo: FIFO (First In, First Out)
  - Tiempo de acceso: 10 ns.
  - La probabilidad de acierto en caché (h) es del 95%

Se pide:

a) Describa estructura de la memoria caché descrita, indicando el número de conjuntos y el tamaño de cada conjunto en bytes.
b) Calcule el tiempo medio de acceso a memoria en dicho computador.
c) Indique cual será el formato que utilizará esta caché para interpretar direcciones de memoria, describiendo qué representa cada una de las partes y su tamaño en bits. El tamaño de palabra del computador es de 4 bytes.

Solución:

a) La caché tiene un tamaño de 16 Kbytes = 16 x 2^{10} bytes = 2^{14} bytes

El número de entradas o líneas de caché = 2^{14} bytes / 64 bytes (tamaño del bloque) = 256 líneas de caché

El número de conjuntos = 256 / 4 = 64 conjuntos (2^6 conjuntos)

El tamaño del campo etiqueta, la memoria principal tiene un tamaño de 512 Mb = 512 x 2^{20} bytes, el número de bloques de memoria principal será 2^{29} bytes / 64 bytes = 2^{23} bloques de M.P.

Se precisa etiquetar 2^{23} bloques de M.P. en 2^6 conjuntos => 2^{23} bloques / 2^6 conjuntos = 2^{17}

Por último, si el computador tiene un tamaño de palabra de 4 bytes y hay bloques de 64 bytes, en cada bloque cabrán 16 palabras (2^4)

b) El tiempo medio de acceso a memoria en una caché de un nivel se puede calcular con la siguiente expresión:

Tavg. = h * tc + (1-h) * tmp

Donde:
- h = probabilidad de un acierto en caché (hit)
- 1-h= probabilidad de un fallo de caché (miss)
- tc = tiempo de acceso a caché
- tmp= tiempo de penalización por fallo.

Por lo que el tiempo medio de acceso en este sistema de memoria sería:

Tavg. = 0,95 x 10 + (1-0,95) x200 = 9,5 + 10 = 19,5 ns

c) El formato de una dirección de memoria será el siguiente:
- Etiqueta: 20 bits superiores de la dirección.
- Número de conjunto: los 6 bits siguientes.
- Byte dentro de la línea: 6 bits inferiores.

Ejercicio 11. Sea un computador dotado de una memoria cache con las siguientes características:
A. Tamaño: 16KB con bloques de 32 bytes (8 palabras)
B. Tiempo de acceso: 10ns

Esta memoria está conectada a través de un bus de 32 bits a una memoria principal que tiene un tiempo de latencia de 40 ns y es capaz de transferir 8 bytes cada 10 ns. Calcular la tasa de aciertos que es necesaria para que el tiempo medio de acceso al sistema de memoria sea de 20 ns.

Solución:

El tamaño de la línea de caché es de 32 bytes. El tiempo de acceso a una línea es de 40 ns + (32/8)*10 ns = 80 ns. Para calcular la tasa de acierto (h) se puede utilizar la siguiente expresión:

Tavg. = h * tc + (1-h) * Tmp

20 = h * 10 +(1-h) * 80 de donde se obtiene que h = 6/7 = 0.857, es decir uma tasa de acierto del 85,7%.

Ejercicio 12. Un computador de 32 bits que se direcciona por bytes posee una memoria caché de datos de 4 KB y una caché de instrucciones de 2 KB. Ambas cachés utilizan una función de correspondencia directa y su tamaño de bloque es de 4 palabras. El tiempo de acceso al sistema de memoria en caso de acierto es de 20 ns. En caso de fallo, el tiempo de acceso (incluyendo la actualización de la caché) es de 100 ns. La memoria caché utiliza una política de actualización diferida (post-escritura, *write-back* o *copy-back*). El tiempo para escribir un bloque de memoria caché a memoria principal es también de 100 ns.

En el computador descrito se ejecuta el siguiente fragmento de código, que se almacena a partir de la dirección de memoria 0x00F00000:

```
            move  $t0, $zero
bucle:      lw    $t1, x($t0)
            sw    $t1, y($t0)
            addi  $t0, $t0, 4
            bne   $t0, 10, bucle
fin:
```

Donde **x** se corresponde con la dirección de memoria 0x70000000 e **y** se corresponde con la dirección de memoria 0x00000400. Todas las instrucciones usadas en el fragmento de código ocupan una palabra cada una.

Se pide:

a) Determine cuál es el tiempo medio de acceso a memoria para el programa descrito.
b) Si se sustituye la memoria caché de datos, por otra que utilice una función de correspondencia asociativa por conjuntos de 4 entradas (4 vías) que utiliza una política de sustitución FIFO ¿cuál será el tiempo?

Solución:

a) La caché de instrucciones tiene 2 KB de tamaño. Cada entrada de la caché tiene 4 palabras. Por tanto:

- Tamaño de bloque = 4 palabras = 16 bytes.
- Número de bloques = 2KB / 16 bytes = 128 bloques

Para la traducción de direcciones en la memoria caché de código se utilizará el siguiente formato:

- 2 bits para selección de byte dentro de palabra.
- 2 bits para selección de palabra dentro de entrada.
- 7 bits para selección de una de las 128 entradas.
- 21 bits restantes como etiqueta.

El fragmento de código propuesto consiste en un bucle en el que la primera instrucción se ejecuta una única vez, y el resto de instrucciones se ejecutan 4 veces. Las instrucciones se almacenan a partir de la dirección de memoria 0x00F00000. Por tanto, el contenido de la memoria en esa zona será:

Dirección de memoria	Instrucción
0x00F00000	
0x00F00004	**lw $t1, x($t0)**
0x00F00008	**sw $t1, y($t0)**
0x00F0000C	**addi $t0, $t0, 4**
0x00F00010	**bne $t0, 16, bucle**

Las direcciones utilizadas, se descomponen, por tanto:

Dirección	Etiqueta	Entrada	Palabra	Byte
0x00F00000	0000 0000 1111 0000 0000 0	0000000	00	00
0x00F00004	0000 0000 1111 0000 0000 0	0000000	01	00
0x00F00008	0000 0000 1111 0000 0000 0	0000000	10	00
0x00F0000C	0000 0000 1111 0000 0000 0	0000000	11	00
0x00F00010	0000 0000 1111 0000 0000 0	0000001	00	00

Se observa que, en todas las referencias, la etiqueta siempre es la misma. La secuencia de referencias es:

Etiqueta	Entrada	Palabra	Acierto/Fallo
0x001700	0	0	Fallo
0x001700	0	1	Acierto
0x001700	0	2	Acierto
0x001700	0	3	Acierto

0x001700	1	0	Fallo
0x001700	0	1	Acierto
0x001700	0	2	Acierto
0x001700	0	3	Acierto
0x001700	1	0	Acierto
...	...	...	...
0x001700	0	1	Acierto
0x001700	0	2	Acierto
0x001700	0	3	Acierto
0x001700	1	0	Acierto

El número total de accesos a memoria para instrucciones es de 17 (2 fallos y 15 aciertos).

La caché de datos tiene 4 KB de tamaño. Cada entrada de la caché tiene 4 palabras. Por tanto:

- Tamaño de bloque = 4 palabras = 16 bytes.
- Número de bloques = 4 KB / 16 bytes = 256 bloques

Para la traducción de direcciones en la memoria caché de datos se utilizará el siguiente formato:

- 2 bits para la selección de byte dentro de palabra.
- 2 bits para selección de palabra dentro de entrada.
- 8 bits para selección de una de las 256 entradas.
- 20 bits restantes como eitqueta.

El fragmento de código realiza 8 accesos a datos. Las direcciones de estos accesos son:

Dirección	**Etiqueta**	**Entrada**	**Palabra**	**Byte**	**Fallo/acierto**	**Actualización**
0x70000000	0x70000	0x00	00	00	Fallo	No
0x00004000	0x00004	0x00	00	00	Fallo	No
0x70000004	0x70000	0x00	01	00	Fallo	Si
0x00004004	0x00004	0x00	01	00	Fallo	No
0x70000008	0x70000	0x00	10	00	Fallo	Si
0x00004008	0x00004	0x00	10	00	Fallo	No
0x7000000C	0x70000	0x00	11	00	Fallo	Si
0x0000400C	0x00004	0x00	11	00	Fallo	No

El número total de accesos es de 8 (0 aciertos y 8 fallos). Además se producen 3 actualizaciones de la memoria principal.

Si se tienen en cuenta los accesos a instrucciones y a datos se tiene:

- Número total de accesos: 17 a instrucciones + 8 a datos = 25 accesos
- Número total de aciertos: 15
- Número total de fallos: 2 + 8 = 10
- Número total de actualizaciones: 3

El tiempo medio de acceso será:

$$T_{acceso} = \frac{15*20+10*100+3*100}{25} = \frac{300+1000+300}{25} = \frac{1600}{25} = 64$$

b) La memoria caché de instrucciones es la misma, por lo que los resultados de esta parte no varían.

La memoria caché de datos tiene conjuntos de 4 entradas. Cada entrada tiene 4 palabras

- Tamaño de entrada: 4 palabras = 16 bytes.
- Tamaño de conjunto: 4 entradas = 16 palabras = 64 bytes.
- Número de conjuntos = 4 KB / 64 bytes = 64 conjuntos.

El formato de una dirección de datos será:

- 2 bits para la selección de byte.

- 2 bits para la selección de palabra en entrada.
- 6 bits para la selección de conjunto.
- 22 bits para la etiqueta.

El fragmento de código realiza 20 accesos a datos. Las direcciones de estos accesos son:

Dirección	Etiqueta	Conjunto	Palabra	Byte	Fallo/acierto	Actualización
0x70000000	0x70000,00	00 0000	00	00	Fallo	No
0x00004000	0x00004,00	00 0000	00	00	Fallo	No
0x70000004	0x70000,00	00 0000	01	00	Acierto	No
0x00004004	0x00004,00	00 0000	01	00	Acierto	No
0x70000008	0x70000,00	00 0000	10	00	Acierto	No
0x00004008	0x00004,00	00 0000	10	00	Acierto	No
0x7000000C	0x70000,00	00 0000	11	00	Acierto	No
0x0000400C	0x00004,00	00 0000	11	00	Acierto	No

De los 8 accesos, 2 son fallos y los 6 restantes son aciertos.

Si se tienen en cuenta los accesos a instrucciones y a datos se tiene:

- Número total de accesos: 17 a instrucciones + 8 a datos = 25 accesos
- Número total de aciertos: 15 + 6 = 21
- Número total de fallos: 2 + 2 = 4

El tiempo medio de acceso será:

$$T_{acceso} = \frac{21*20+4*100}{25} = \frac{420+400}{25} = \frac{820}{25} = 32{,}8$$

Ejercicio 13. Se dispone de un computador con las siguientes características:

- Tamaño de palabra de 64 bits.
- Memoria principal con un tiempo de acceso de 60ns.
- Memoria cache interna de datos:
  - Capacidad: 8KB.
  - Bloques de 16 bytes.
  - Tiempo de acceso de 10ns.
  - Organización totalmente asociativa, con política de reemplazo FIFO (primero en entrar, primero en salir)

Sea el siguiente programa:

```
n = 0;
for (i = 0; i < 500; i++)
      n = n + b[i];
for (i = 0; i < 500; i++)
      a = a + b[i] * c[i];
for (i = 0; i < 500; i++)
      r  = r + b[i] + d[i];
```

La variable de control del bucle (`i`) y las variables `a`, `r` y `n` están asignadas a registros del procesador. Todas las variables que se manejan son enteros de 64 bits.

Sabiendo que inicialmente la cache de datos está vacía. Se pide:

a) Calcule la tasa de fallos de cache que genera el programa cuando se ejecuta sobre el computador descrito.
b) Calcule el tiempo total invertido en el acceso a los datos.

Solución:

Como cada elemento del vector es un entero de 64 bits (8 bytes) y puesto que la memoria cache es de 8KB, ésta tiene capacidad para almacenar 8KB ÷ 8B por elemento = 1024 elementos.

El número de elementos de cada uno de los vectores manejados por el programa es 500, por lo que la memoria cache tiene capacidad para ubicar dos vectores y 24 enteros adicionales.

Por otra parte, puesto que el tamaño del bloque es de 16 bytes, cada bloque da cabida a 2 elementos de un vector.

a) Cálculo de la tasa de fallos

El primer bucle realiza 500 accesos; el segundo bucle realiza 1000 accesos y el tercer bucle realiza 1000 accesos. En **total** se realizan ***2.500 accesos*** a los elementos de los vectores b, c y d.

En el primer bucle se producen 250 fallos:

i = 0 : b[0] no está en el cache, fallo. Se llevan b[0] y b[1] al bloque 0 del cache
i = 1: b[1] está en el cache, acierto.
i = 2: b[2] no está en el cache, fallo. Se llevan b[2] y b[3] al bloque 1 del cache
i = 3: b[3] está en el cache, acierto.
…
Se produce un fallo cada 2 iteraciones (500 ÷ 2 = 250)

En el segundo bucle, aunque se vuelve a hacer referencia al vector b, los elementos de b se encuentran en la cache, por lo tanto, **no** se producen fallos por uso de b. Se produce un fallo por cada dos elementos de c, en total se producen 250 fallos.

Cuando comienza el tercer bucle, solo hay espacio para 24 nuevos valores de c, cuando se requiera el elemento c[24] se debe escoger una víctima que abandone la cache, como la política escogida es FIFO esa víctima corresponde al bloque que tiene los valores b[0] y b[1], que afortunadamente ya han sido utilizados y no se necesitan para cálculos posteriores. El efecto continuará en cascada y por lo tanto, se producirá el mismo número de fallos que en el bucle anterior, esto es 250 fallos.

El número **total de fallos** es de 250 + 250 + 250 = **750. La tasa de fallos** es 100 x (750 fallos ÷ 2.500 accesos) = **30%**

b) Tiempo total invertido en el acceso a los datos:

$$T = TotalAciertos \times T_{acceso\_cache} + TotalFallos \times (T_{acceso\_cache} + T_{acceso\_memoria})$$

$$= 1.750 \times 10ns + 750 \times (10ns + 60ns) = \mathbf{70.000ns = 70\mu s = 0.07\ ms}$$

Ejercicio 14. Se dispone de un computador que direcciona la memoria por bytes y que dispone de una memoria caché con un tamaño de 32 KB para guardar instrucciones o datos de los procesos. El tamaño de la línea es de 64 bytes. Los procesos pueden direccionar 1MB de memoria principal y ésta se direcciona por bytes. Un acceso a memoria RAM consume 80 ns y a la caché 30 ns. Asuma que la caché está inicialmente vacía

Se tienen dos alternativas de diseño para la caché:
1. Correspondencia directa.
2. Correspondencia asociativa por conjuntos de 8 vías con algoritmo de reemplazo LRU.

Para cada una de estas dos alternativas de diseño indique:
a) Dada la siguiente secuencia de direcciones:
0x7B042 0x7D042 0x7D074
identifique la etiqueta, la línea (o conjunto) y el desplazamiento en el que se encuentra el dato asociado a cada dirección y cuáles de estas direcciones ocasionan un fallo de caché.
b) El tiempo de acceso de la secuencia.

Solución:

En primer lugar, se analizará el caso de correspondencia directa. El tamaño de la caché es de 32KB (2^{15} bytes), la línea es de 64B (2^6 bytes), el número de entradas de la caché es: 2^{15} bytes ÷ 2^6 bytes = 2^9 entradas.

El número de bits para guardar datos o instrucciones: 64B x 8 = 2^6 x 2^3 = 2^9 bits.
Bits para el campo etiqueta. Tamaño de la Memoria principal ÷ Tamaño de la caché = 1MB ÷ 32KB = $2^{20} \div 2^{15} = 2^5$, i.e. 5 bits.

Es decir, se utilizarán los 5 bits superiores para la etiqueta, los 9 intermedios para la línea y los 6 bits inferiores para el indicar el byte dentro de la línea.

Para la secuencia: 0x7B042 0x7D042 0x7D074 se obtiene lo siguiente:

7B042 es: 01111 011000001 000010 –fallo, caché vacía-
7D042 es: 01111 101000001 000010 –fallo, líneas diferentes -
7D074 es: 01111 101000001 110100 –acierto con 0x0007D042 igual etiqueta, igual línea-

Un acierto se realiza en 30ns y un fallo en 30ns + 80ns = 110ns. El tiempo de acceso viene dado por 2 x 110ns + 1 x 30ns = 250ns

A continuación, se analiza el caso caso de la correspondencia asociativa por conjuntos de 8 vias. El tamaño de la caché es de 32KB (2^{15} bytes) y 8 vías, la línea es de 64B (2^6 bytes), el número de entradas de la caché es: 2^{15} bytes ÷ (2^6 bytes x 2^3) = 2^6 conjuntos de 8 líneas cada uno. Se utilizan los 6 bits inferiores para identificar el byte. Lo siguientes 8 bits para identificar el conjunto y los 20-6-8 = 6 bits superiores para la etiqueta.

Para la secuencia: 0x7B042 0x7D042 0x7D074 se obtiene:

7B042 es: 011110 11000001 000010 –fallo, caché vacía-
7D042 es: 011111 01000001 000010 –fallo
7D074 es: 011111 01000001 110100 –acierto, el mismo conjunto que 0x7D042 y coincide con la etiqueta de 0x7D042 por ello se trata del mismo bloque de memoria-

Un acierto se realiza en 30ns y un fallo en 30ns + 80ns = 110ns. El tiempo de acceso viene dado por 2 x 110ns + 1 x 30ns = 250ns

Ejercicio 15. Sea un computador de 32 bits que dispone de una memoria caché para instrucciones de 64 KB y una memoria caché de datos de 128 KB. Ambas memorias disponen de líneas de 32 bytes, política de correspondencia directa y política de escritura diferida o retrasada. La memoria se direcciona por bytes. Considere el siguiente fragmento de código:

```
for (i = 0; i< 512; i++)
      for (j = 0; j < 512; j++)
            c[i][j] = a[i][j] + b[i][j]
```

Donde `a`, `b` y `c` representan matrices cuadradas de 512 x 512 números enteros, que se almacenan en memoria de forma consecutiva. Cada matriz a su vez se almacena por filas. La matriz `a` comienza en la dirección hexadecimal 0x00100000. Se pide:

a) Calcular la tasa de aciertos que se produce en la memoria caché de datos para el fragmento de código anterior, suponiendo que dicha memoria está inicialmente vacía.
b) Proponga un diseño de memoria caché que mejore la tasa de aciertos para el fragmento de programa anterior sin que ello suponga un incremento excesivo en el tiempo de búsqueda en la memoria caché. Calcule la tasa de aciertos para este diseño.

Solución:

a) La memoria caché de datos tiene 128 KB / 32 bytes = $2^{17}/2^5 = 2^{12}$ = 4096 líneas o bloques de 32 bytes. Cada matriz consta de 512 x 512 x 4 = 1 MB = 2^{20} bytes. Los tres vectores se almacenan de forma consecutiva, primero `a`, después `b` y a continuación `c`. Las direcciones de comienzo de los tres vectores son las siguientes:

- La matriz a comienza en la posición 0x00100000 y termina en la posición 0x001FFFFF
- La matriz b comienza en la posición 0x00200000 y termina en la posición 0x002FFFFF

- La matriz c comienza en la posición 0x00300000 y termina en la posición 0x003FFFFF

Para determinar en qué línea de caché se almacena una dirección de memoria hay que recurrir a la siguiente distribución de los bits de una dirección:

- Los 15 bits superiores constituyen la etiqueta de la línea.
- Los 12 bits siguientes determinan el número de bloque y por tanto la línea de caché en la que hay que almacenar el bloque.
- Los 5 bits inferiores determinan el byte dentro de la línea.

En el fragmento de programa anterior se producen 512 * 512 * 3 = 786432 accesos a memoria, de los cuales las 2/3 son lecturas (matrices `a` y `b`) y 1/3 son escrituras (la matriz `c`). A continuación, se indican los accesos que se producen a la memoria caché:

Elemento	Dirección de memoria	Tipo de acceso	Línea de caché	Fallo/acierto
`a[0][0]`	0x00100000	Lectura	0	Fallo
`b[0][0]`	0x00200000	Lectura	0	Fallo (se expulsa a)
`c[0][0]`	0x00300000	Escritura	0	Fallo (se expulsa b)
`a[0][1]`	0x00100004	Lectura	0	Fallo (se expulsa c)
`b[0][1]`	0x00200004	Lectura	0	Fallo (se expulsa a)
`c[0][1]`	0x00300004	Escritura	0	Fallo (se expulsa b)

Este proceso se repite hasta el final, es decir todos los accesos son fallos. La tasa de aciertos por tanto es 0.

b) El hecho de que todos los accesos a la caché sean fallos se debe a que en cada vuelta del bucle los elementos `a[i][j]`, `b[i][j]` y `c[i][j]` se almacenan en la misma línea de caché. Para resolver este problema se podría utilizar una memoria caché asociativa por conjuntos de 4 vías. De esta forma en cada vuelta del bucle se podrían almacenar en el mismo conjunto de la memoria caché tres bloques correspondientes a los vectores a, b y c. Como en cada línea caben 8 elementos, se produciría un fallo por cada 8 accesos, con lo que se obtendría una tasa de aciertos de 7/8.

Ejercicio 16. Sea un computador de 32 bits que direcciona la memoria por bytes y que dispone de una memoria caché para instrucciones y una memoria caché para datos. Ambas memorias disponen de líneas de 16 bytes y una política de correspondencia asociativa por conjuntos de **n** vías con algoritmo de reemplazo LRU.

Considere el siguiente fragmento de código:

```
for (i = 0; i< 2^18; i++)
      a[i] = b[i] + c[i] + d[i];
```

Donde `a, b, c y d` representan vectores de 2^{18} números enteros, que se almacenan en memoria de forma consecutiva. El vector `a` comienza en la dirección hexadecimal 0x00500000. A continuación, se almacena `b`, `c` y finalmente `d`. Considerando que la caché se encuentra inicialmente vacía. Se pide:

a) Para un tamaño de caché de datos de 256KB, se barajan dos grados de asociatividad: **n=2 o n=4.** Calcule razonadamente cuál es la tasa de aciertos en cada uno de los dos casos y compare ambos escenarios.
b) Considerando los dos escenarios anteriores, ¿duplicar el tamaño de memoria caché a 512KB aumentaría la tasa de aciertos?

Solución

a) Cada vector consta de 2^{18} x 4 bytes (cada entero) = 2^{20} bytes. Los cuatro vectores se almacenan de forma consecutiva. Las direcciones de comienzo de los tres vectores son las siguientes:

- El vector a comienza en la posición 0x00500000 y termina en la posición 0x005FFFFF
- El vector b comienza en la posición 0x00600000 y termina en la posición 0x006FFFFF
- El vector c comienza en la posición 0x00700000 y termina en la posición 0x007FFFFF
- El vector c comienza en la posición 0x00800000 y termina en la posición 0x008FFFFF

Grado de asociatividad n=2. La memoria caché de datos está dividida en conjuntos de 2 líneas. El tamaño del conjunto es por tanto 2 x 16 bytes = 32 bytes = 2^5 bytes. El número de conjuntos de la caches es des 256 KB / 32 bytes = 2^{18} / 2^5 = 2^{13} conjuntos.

Para determinar en qué conjunto de caché se almacena una dirección de memoria hay que recurrir a la siguiente distribución de los bits de una dirección:

- Los 15 bits superiores constituyen la etiqueta de la línea
- Los 13 bits siguientes determinan el conjunto en el que se almacena el bloque. Dentro de este conjunto el bloque se almacenará en cualquiera de las dos líneas siguiendo una política de reemplazo LRU.
- Los 4 bits inferiores determinan el byte dentro de la línea.

A continuación, se indican los accesos que se producen a la memoria caché:

Elemento	Dirección de memoria	Tipo de acceso	Conjunto de caché	Línea dentro del conjunto	Fallo/acierto
b[0]	0x00600000	Lectura	0	0	Fallo
c[0]	0x00700000	Lectura	0	1	Fallo
d[0]	0x00800000	Lectura	0	0	Fallo (se expulsa b)
a[0]	0x00500000	Escritura	0	1	Fallo (se expulsa c)
b[1]	0x00600004	Lectura	0	0	Fallo (se expulsa d)
c[1]	0x00700004	Lectura	0	1	Fallo (se expulsa a)
d[1]	0x00800004	Lectura	0	0	Fallo (se expulsa b)
a[1]	0x00500004	Escritura	0	1	Fallo (se expulsa c)
…	…	…	…	…	…
b[4]	0x00600010	Lectura	1	0	Fallo
c[4]	0x00700010	Lectura	1	1	Fallo
d[4]	0x00800010	Lectura	1	0	Fallo (se expulsa b)
a[4]	0x00500010	Escritura	1	1	Fallo (se expulsa c)

Este proceso se repite hasta el final, es decir todos los accesos son fallos. La tasa de aciertos por tanto es 0.

Grado de asociatividad n=4: La memoria caché de datos está dividida en conjuntos de cuatro líneas. En este caso el número de conjuntos es 256 KB / 64 bytes = 2^{18} / 2^6 = 2^{12} conjuntos

Para determinar en qué conjunto de caché se almacena una dirección de memoria hay que recurrir a la siguiente distribución de los bits de una dirección:

- Los 16 bits superiores constituyen la etiqueta de la línea
- Los 12 bits siguientes determinan el conjunto en el que se almacena el bloque. Dentro de este conjunto el bloque se almacenará en cualquiera de las dos líneas siguiendo una política de reemplazo LRU.
- Los 4 bits inferiores determinan el byte dentro de la línea.

A continuación, se indican los accesos que se producen a la memoria caché:

Elemento	Dirección de memoria	Tipo de acceso	Conjunto de caché	Línea dentro del conjunto	Fallo/acierto
b[0]	0x00600000	Lectura	0	0	Fallo
c[0]	0x00700000	Lectura	0	1	Fallo
d[0]	0x00800000	Lectura	0	2	Fallo
a[0]	0x00500000	Escritura	0	3	Fallo
b[1]	0x00600004	Lectura	0	0	Acierto
c[1]	0x00700004	Lectura	0	1	Acierto
d[1]	0x00800004	Lectura	0	2	Acierto
a[1]	0x00500004	Escritura	0	3	Acierto
b[2]	0x00600008	Lectura	0	0	Acierto
c[2]	0x00700008	Lectura	0	1	Acierto
d[2]	0x00800008	Lectura	0	2	Acierto
a[3]	0x00500008	Escritura	0	3	Acierto
b[3]	0x0060000C	Lectura	0	0	Acierto
c[3]	0x0070000C	Lectura	0	1	Acierto
d[3]	0x0080000C	Lectura	0	2	Acierto
a[3]	0x0050000C	Escritura	0	3	Acierto
b[4]	0x00600010	Lectura	1	0	Fallo
c[4]	0x00700010	Lectura	1	1	Fallo
d[4]	0x00800010	Lectura	1	2	Fallo
a[4]	0x0050000C	Escritura	1	3	Fallo
…	…	…	…	…	…

Este proceso se repite hasta que se llena toda la caché y también hasta que acaban de procesar los vectores. Así pues, para cada conjunto de la caché habrá 4 fallos y 12 aciertos, por lo que la tasa de aciertos será de (12/16)x100 = 75%.

b) El aumento del tamaño de la caché no reduce la tasa de aciertos dado que no existe localidad temporal en el acceso a los datos.

Ejercicio 17. Dado el siguiente código que se ejecuta en una arquitectura de 32 bits, donde la matriz A es una matriz de enteros (de tamaño 4 bytes) que se almacena por filas. Es decir, las entradas se almacenan en memoria de acuerdo con el siguiente orden: `A[1][1], A[1][2], A[1][3]…. A[1][1024], A[2][1], …. La` matriz tiene la siguientes dimensiones: A[1024][8]. Dado el siguiente lazo:

```
for(j=0;j<8;j++)
      for(i=0;i<1024;i++)
            A[i][j]=8
```

La arquitectura implementa una memoria caché con correspondencia directa. Esta memoria tiene una capacidad de 16KB y un tamaño de línea de 32 bytes. Las variables i y j se almacenan en registros.

Se pide:

a) Describe el comportamiento del programa sobre la memoria caché de datos.

b) Calcular la tasa de aciertos en la caché.
c) Si el tiempo de acceso a memoria caché es de 5 ciclos de reloj y el tiempo de acceso a memoria es de 15 ciclos, calcular el tiempo medio (en ciclos) de ejecución del lazo.
d) Indica de forma justificada qué transformaciones se pueden realizar en el código con el fin de mejorar su rendimiento.

Solución:

Debido a que la matriz `A` se accede por columnas existirá una pequeña localidad espacial en los accesos. De cada línea cargada en la memoria caché, sólo se leerá una palabra (debido a que la siguiente palabra está a 8 palabras de distancia, es decir, en la siguiente línea caché.).

a) Cuando el programa se ejecuta, se accede a una palabra de cada línea, accediendo a líneas caché consecutivas. La caché tiene capacidad para 16KB/32=4K líneas, mientras que la matriz tiene un tamaño de 1024*8*4=32Kb=8K líneas. Por este motivo, cuando se vuelve a acceder a una línea (después de realizar 1024 accesos del bucle i) la memoria ya no la contiene, originando un fallo caché.
b) La tasa de aciertos caché es del 0%, debido a que un dato nunca es reusado en esta memoria.
c) Debido a que la tasa de aciertos es del 0%, todos los accesos se realizan a memoria, por lo que el tiempo total de ejecución (en ciclos) será: (Ciclos_Mem + Ciclos_Cache)*Num_accesos=(15 + 5)*8K.
d) Intercambiando los índices del buble se consigue maximizar la localidad espacial en los accesos. Ahora el contenido completo de la línea caché es reutilizado, por lo que la tasa de aciertos será de 7 aciertos por cada fallo y el tiempo total de ejecución, en ciclos, será de (5*7+*(15+5)*1)*1K.