# Relacion3EC.pdf

mxjito

Estructura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Quiero que seas como el powerpoint de 2008 de la Universidad: que nunca cambies ♡

# Relación de Ejercicios 3

## PREGUNTA 1

Enunciado: Un computador tiene un bus de **direcciones de 32 bits** y es direccionable a nivel de byte. La cache tiene **256 KBytes**, con **palabras de 1 byte** y **bloques de 32 bits**, **asignación directa**, **reemplazo LRU** y post-escritura. Se utiliza un **contador de 5 bits para el reemplazo** y **1 bit de Dirty** para los bloques que hayan sido modificados y por tanto tienen que escribirse en memoria principal.

CARACTERÍSTICAS IMPORTANTES

- Tamaño de direcciones: 32 bits
- Tamaño de cache: 256 KBytes
- Tamaño de palabras: 1 byte
- Tamaño de bloques: 32 bits
- Asignación directa

256 KBytes = $2^8 * 2^{10} = 2^{18}$ bytes

Bloques de 32 bits = Bloques de 4 Bytes = $2^2$ bytes

Palabras de 1 byte → 4 palabras por bloque = $2^2$ palabras por bloque

*Apartado a)*

Descompón en los siguientes campos la dirección física e indica su tamaño en bits:

INDICE : $2^{18}$ de tamaño de cache : $2^2$ palabras por bloque = $2^{16}$ bloques → 16 bits

W: $2^2$ palabras por bloque → 2 bits

B: $2^0$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^{14}$ → 14 bits


*Apartado b)*

¿Detectas algún dato en el enunciado que no te parezca consistente?

R) Sí, con una asignación directa no hace falta implementar una política de reemplazo (no necesito el controlador de 5 bits en el directorio).

*Apartado c)*

¿Cuál es el tamaño, en bits, del directorio de la cache? Considera que en el directorio se encuentran todos los bits de control necesarios para el funcionamiento de la memoria cache.

TAMAÑO DEL DIRECTORIO: N° BLOQUES * ( BITS TAG + BITS DIRTY + BIT VALIDEZ)

→ $2^{16}$ * (14 (tag) + 1 (dirty) + 1 (validez) = 1048576 bits

# PREGUNTA 2

Enunciado: Una memoria cache **asociativa por conjuntos de 2 bloques** con d**irecciones 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.

CARACTERÍSTICAS IMPORTANTES

- Tamaño de direcciones: 32 bits
- Tamaño de cache: 128 KBytes
- Tamaño de palabras: 1 byte
- Tamaño de bloques: 4 bytes
- Asociativa por conjuntos de 2 bloques

128 KBytes = $2^7 * 2^{10} = 2^{17}$ bytes

Palabras de 1 byte → 4 palabras por bloque = $2^2$ palabras por bloque

$2^{17}$ (tamaño de cache) : $2^2$ (palabras por bloque) = $2^{15}$ bloques : 2 bloques/conjunto = $2^{14}$ conjuntos

*Apartado a)*

El tamaño de los siguientes campos de la dirección de memoria principal:

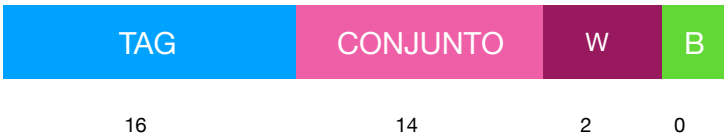
CONJUNTO : $2^{14}$ conjuntos → 14 bits

W: $2^2$ palabras por bloque → 2 bits

B: $2^0$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^{16}$ → 16 bits

DIRECCIÓN DE 32 BITS




CAMPO NÚMERO DE LINEA O BLOQUE = $2^{14}$ (conjunto) + $2^{16}$ (tag) = $2^{30}$ → 30 bits

*Apartado b)*

Indicar el valor de cada campo para las siguientes direcciones de memoria principal. Debes proporcionar los valores en hexadecimal, con el número de dígitos que corresponde al tamaño del campo y usando mayúsculas para las letras:

*Procedimiento: *
1. Pasar la dirección de hexadecimal a binario
2. Dividir bits en sus respectivos campos
3. Pasar cada campo a hexadecimal nuevamente (de derecha a izquierda)

**EJEMPLO:**

0284A482h → 0000 0010 1000 0100 1010 0100 1000 0010

W: 10 = 2h

CONJUNTO = 10 1001 0010 0000 = 2920h

TAG = 0000 0010 1000 0100 = 0284h

| Dirección MP | TAG | CONJUNTO | W |
|---|---|---|---|
| 0284A482h | 0284h | 2920h | 2h |
| 01148C89h | 0114h | 2322h | 1h |
| 0038CF00h | 0038h | 33C0h | 0h |
| 0038CF01h | 0038h | 33C0h | 1h |

msi

# Tu ordenador lo único que necesita programar es su jubilación.

El Stealth 15M es uno de los portátiles gaming más finos y ligeros. Siempre menos es más. Ve a donde quieras llevando siempre el máximo rendimiento.

*Apartado c)*

¿Pueden todos los bloques que incluyen las referencias anteriores estar en caché al mismo tiempo?

R) Si

# PREGUNTA 3

Enunciado: Sea un sistema de memoria de **1 MByte**, con **palabras de 1 byte**, que incorpora una **memoria caché 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 secuencia de direcciones de memoria principal (en hexadecimal) que aparece en la tabla.

CARACTERÍSTICAS IMPORTANTES

- Tamaño de memoria principal: 1 MByte
- Tamaño de cache: 64 KBytes
- Tamaño de palabras: 1 byte
- Tamaño de bloques: 8 palabras
- Asociativa en 4 conjuntos

Memoria Principal de 1MByte → $2^{20}$ bytes → direcciones de 20 bits

4 conjuntos → $2^2$ bytes

8 palabras por bloque → $2^3$ bytes

*Apartado a)*

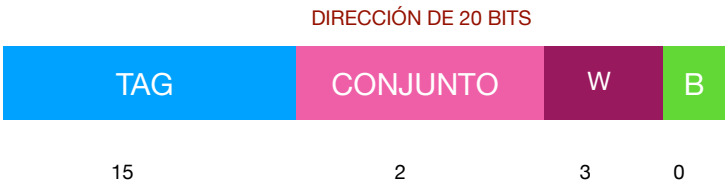
Indica para cada referencia la dirección de bloque en memoria principal (en hexadecimal), el conjunto de la caché asignado y si se produce un acierto o un fallo. Para ello habrás tenido previamente que determinar para tus referencias:

CONJUNTO : $2^2$ conjuntos → 2 bits

W: $2^3$ palabras por bloque → 3 bits

B: $2^0$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^{15}$ → 15 bits




CAMPO NÚMERO DE LINEA O BLOQUE = $2^2$(conjunto) + $2^{15}$(tag) = $2^{17}$→ 17 bits

*Procedimiento: *
1. Pasar la dirección de hexadecimal a binario
2. Dividir bits en sus respectivos campos
3. Pasar cada campo a hexadecimal nuevamente (de derecha a izquierda

**EJEMPLO:**

ABC80h → 1010 1011 1100 1000 0000

W: 000 = 0h

CONJUNTO = 00 = 0h

TAG = 101 0101 1110 0100 = 55E4h

BLOQUE M.P = 1 0101 0111 1001 0000 = 15790h

Calcula el índice de fallos (con dos decimales):

ÍNDICE DE FALLOS = FALLOS / REFERENCIAS

Fallos = 8
Referencias = 9

Índice de fallos = 8/9 = 0,88

| DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| ABC80h | 15790h | 0h | Fallo |
| ABC81h | 15790h | 0h | Acierto |
| ABC88h | 15791h | 1h | Fallo |
| BCD90h | 179B2h | 2h | Fallo |
| BCD9Dh | 179B3h | 3h | Fallo |
| BCDA0h | 179B4h | 0h | Fallo |
| CDE00h | 19BC0h | 0h | Fallo |
| CDE18h | 19BC3h | 3h | Fallo |
| CDE20h | 19BC4h | 0h | Fallo |

Apartado b)

Supón que las referencias anteriores se corresponden con una iteración de un bucle. ¿Cuál será el índice de fallos cuando el número de iteraciones tienda a infinito?

R) 0

# PREGUNTA 4

Enunciado: Disponemos de una memoria cache de **4 Kbytes** 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 Mbyte**. Para la siguiente secuencia de peticiones de direcciones a memoria principal:

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

CARACTERÍSTICAS IMPORTANTES

- Tamaño de memoria principal: 1 MByte
- Tamaño de bloque: 256 bytes
- Tamaño de cache: 4 KBytes
- Tamaño de palabras: 1 byte

Memoria Principal de 1MByte → $2^{20}$ bytes → direcciones de 20 bits

Bloques de 256 bytes y palabras de 1 byte → 256 palabras por bloque : $2^8$ palabras por bloque

Cache de 4KBytes → $2^2 * 2^{10} = 2^{12}$ bytes

*Apartado a)*

## ASOCIATIVIDAD 2 → 2 vías por conjunto

$2^{12}$ (tamaño de cache) : $2^8$ (palabras por bloque) = $2^4$ bloques → $2^4$ bloques : 2 vías por conjunto = $2^3$ conjuntos

CONJUNTO : $2^3$ conjuntos → 3 bits

W: $2^8$ palabras por bloque → 8 bits

B: $2^0$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^9$ → 9 bits

Suponiendo la cache inicialmente vacía, describir la evolución del directorio cache identificando, la etiqueta (TAG, en hexadecimal), el conjunto de la caché asignado (c, en hexadecimal) y si se produce un acierto o un fallo.

**EJEMPLO:**

319F0h → 0011 0001 1001 1111 0000

W: 1111 0000 = F0h

CONJUNTO = 001 = 1h

TAG = 0 0110 0011 = 063h

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 319F0h | 063h | 1h | Fallo |
| 31AF0h | 063h | 2h | Fallo |
| 7013Ch | 0E0h | 1h | Fallo |
| 77777h | 0EEh | 7h | Fallo |
| 44037h | 088h | 0h | Fallo |
| 778DEh | 0EFh | 0h | Fallo |
| A5021h | 14Ah | 0h | Fallo |

Calcula el índice de fallos (con dos decimales):

Fallos = 7
Referencias = 7

Índice de fallos = 7/7 = 1

*Apartado b)*

## ASIGNACIÓN DIRECTA

$2^{12}$ (tamaño de cache) : $2^{8}$ (palabras por bloque) = $2^{4}$ bloques

INDEX : $2^{4}$ bloques → 4 bits

W: $2^{8}$ palabras por bloque → 8 bits

B: $2^{0}$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^{8}$ → 8 bits


Suponiendo la cache inicialmente vacía, describir la evolución del directorio cache identificando, la etiqueta (TAG, en hexadecimal), el conjunto de la caché asignado (c, en hexadecimal) y si se produce un acierto o un fallo.

**EJEMPLO:**

319F0h → 0011 0001 1001 1111 0000

W: 1111 0000 = F0h

CONJUNTO = 1001 = 9h

TAG = 0011 0001 = 31h

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 319F0h | 31h | 9h | Fallo |
| 31AF0h | 31h | Ah | Fallo |
| 7013Ch | 70h | 1h | Fallo |
| 77777h | 77h | 7h | Fallo |
| 44037h | 44h | 0h | Fallo |
| 778DEh | 77h | 8h | Fallo |
| A5021h | A5h | 0h | Fallo |

Calcula el índice de fallos (con dos decimales):

Fallos = 7
Referencias = 7

Índice de fallos = 7/7 = 1

*Apartado c)*

## TOTALMENTE ASOCIATIVA

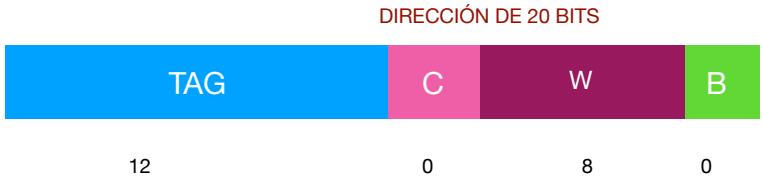
$2^{12}$ (tamaño de cache) : $2^8$ (palabras por bloque) = $2^4$ bloques → $2^4$ bloques : $2^4$ vías por conjunto = 1 conjunto

CONJUNTO : $2^0$ conjuntos → 0 bits

W: $2^8$ palabras por bloque → 8 bits

B: $2^0$ (1) bytes por palabra → 0 bits

TAG: El resto de bytes = $2^{12}$ → 12 bits




Suponiendo la cache inicialmente vacía, describir la evolución del directorio cache identificando, la etiqueta (TAG, en hexadecimal), el conjunto de la caché asignado (c, en hexadecimal) y si se produce un acierto o un fallo.

**EJEMPLO:**

319F0h → 0011 0001 1001 1111 0000

W: 1111 0000 = F0h

CONJUNTO = 0h

TAG = 0011 0001 1001 = 319h

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 319F0h | 319h | 0h | Fallo |
| 31AF0h | 31Ah | 0h | Fallo |
| 7013Ch | 701h | 0h | Fallo |
| 77777h | 777h | 0h | Fallo |
| 44037h | 440h | 0h | Fallo |
| 778DEh | 778h | 0h | Fallo |
| A5021h | A50h | 0h | Fallo |

Calcula el índice de fallos (con dos decimales):

Fallos = 7
Referencias = 7

Índice de fallos = 7/7 = 1

*Apartado d)*
A la vista de los resultados, teniendo en cuenta criterios de rendimiento y coste, recomendarías:

R) Organización directa

*Para evaluar el rendimiento se tiene que tener en cuenta no solo el índice de fallos sino también el tiempo de acceso. Entre menor asociatividad, menor tiempo de acceso.

# PREGUNTA 5

Enunciado: 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).

CARACTERÍSTICAS IMPORTANTES

- Tamaño de memoria principal: 16 MBytes

Memoria principal de 16 MBytes = $2^4 * 2^{20} = 2^{24}$ bytes → direcciones de 24 bits

*Apartado a)*

## PEAR P3

En función del comportamiento mostrado en los diagramas anteriores, determina: (Tamaño conjunto de los campos que identifican el byte dentro de la palabra y la palabra dentro del bloque, tamaño del campo que asigna el conjunto, tamaño del campo etiqueta (TAG)).

Cache de 8Kbytes → $2^3 * 2^{10} = 2^{13}$ bytes

Del diagrama podemos observar que hay 128 ($2^7$) conjuntos de 2 vías cada uno → 256 bloques → $2^8$ bloques

$2^{13}$ (tamaño de cache) : $2^X$ palabras por bloque = $2^8$ bloques → X = 5 → Hay $2^5$ palabras por bloque

CONJUNTO : $2^7$ conjuntos → 7 bits

W + B : $2^5$ palabras por bloque → 5 bits

TAG: El resto de bytes = $2^{12}$ → 12 bits


**EJEMPLO:**

000A40h → 0000 0000 0000 1010 0100 0000

W: 0 0000 = 00h

CONJUNTO = 101 0010 = 52h

TAG = 0000 0000 0000 = 000h

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 000A40h | 000h | 52h | Fallo |
| 004A40h | 004h | 52h | Fallo |
| 008A40h | 008h | 52h | Fallo |
| 00BA40h | 00Bh | 52h | Fallo |

¿Qué estrategias de reemplazo de líneas pueden dar lugar al resultado anterior (excluyendo RANDOM)?

R) LRU/FIFO

Índice de aciertos total: 0

*Apartado b)*

## CARROT C4

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**).

Cache de 16Kbytes → $2^4 * 2^{10} = 2^{14}$ bytes

Se dobla el nivel de asociatividad → 2 (nivel de asociatividad Pear 3) * 2 = Asociatividad 4 → $2^2$

Se tiene el mismo número de conjuntos pero se doblan los bloques por conjunto

$2^{14}$ (tamaño de cache) : $2^5$ palabras por bloque = $2^9$ bloques

$2^9$ bloques : $2^2$ (asociatividad) = $2^7$ conjuntos

Por lo tanto ahora los tamaños de los campos pasan a ser:

CONJUNTO : $2^7$ conjuntos → 7 bits

W + B : $2^5$ palabras por bloque → 5 bits

TAG: El resto de bytes = $2^{12}$ → 12 bits

Muestra la evolución para las 2 primeras iteraciones en el Carrot C4:

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 000A40h | 000h | 52h | Fallo |
| 004A40h | 004h | 52h | Fallo |
| 008A40h | 008h | 52h | Fallo |
| 00BA40h | 00Bh | 52h | Fallo |
| 000A40h | 000h | 52h | Acierto |
| 004A40h | 004h | 52h | Acierto |
| 008A40h | 008h | 52h | Acierto |
| 00BA40h | 00Bh | 52h | Acierto |

*Explicación: Como tenemos 4 vías por conjunto, cada uno de los bloques de memoria irá a una vía diferente. Así logramos evitar los fallos de conflicto. En la segunda iteración del bucle (al igual que en el resto) obtendremos aciertos ya que tendremos almacenados los bloques a los que queremos acceder.

Total de referencias = 400 (4 referencias por iteración del bucle {100 iteraciones del bucle})

Total de aciertos = 400 - 4 (primera iteración del bucle donde obtenemos fallos) = 396

Indice de aciertos = 396/400 = 0,99

## GRAPE G5

Por otro lado, el "Grape G5" 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**.

Cache de 32KBytes → $2^5 * 2^{10} = 2^{15}$ bytes

Nivel de asociatividad = 2 (Igual al del Pear P3)

Conjuntos = 128 (conjuntos del Pear P3) * 4 = 512 → $2^9$ conjuntos

Por lo tanto ahora los tamaños de los campos pasan a ser:

CONJUNTO : $2^9$ conjuntos → 9 bits

W + B : $2^5$ palabras por bloque → 5 bits

TAG: El resto de bytes = $2^{10}$ → 10 bits


Muestra la evolución para las 2 primeras iteraciones en el Grape G5:

**EJEMPLO:**

000A40h → 0000 0000 0000 1010 0100 0000

W: 0 0000 = 00h

CONJUNTO = 0 0101 0010 = 052h

TAG = 0000 0000 0000 = 000h

| DIRECCIÓN MP | TAG | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 000A40h | 000h | 052h | Fallo |
| 004A40h | 004h | 052h | Fallo |
| 008A40h | 008h | 052h | Fallo |
| 00BA40h | 00Bh | 1D2h | Fallo |
| 000A40h | 000h | 052h | Fallo |
| 004A40h | 004h | 052h | Fallo |
| 008A40h | 008h | 052h | Fallo |
| 00BA40h | 00Bh | 1D2h | Acierto |

Total de referencias = 400 (4 referencias por iteración del bucle {100 iteraciones del bucle})

Total de aciertos = 400 (total de iteraciones) - 4 (primera iteración) - (3 fallos * 99 iteraciones restantes) → 99 aciertos

Índice de aciertos = 99/400 = 0,24

¿Cuál de los dos microprocesadores 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?

R) Carrot C4

# PREGUNTA 6

Considérese un procesador con **instrucciones de 32 bits** y una **caché de instrucciones de 32 bytes** con **bloques 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.

Código A:

```
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
```

Código B:

```
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
```

CARACTERÍSTICAS IMPORTANTES

- Tamaño de instrucción/dirección: 32 bits
- Tamaño de cache: 32 bytes
- Tamaño de bloques: 8 bytes

Cache de 32 bytes → $2^5$ bytes

Bloque de 8 bytes = $2^3$ bytes → $2^5$ (tamaño cache) : $2^3$ (tamaño de bloque) = $2^2$ bloques

Código MIPS → Palabras de 4 Bytes ($2^2$ bytes)

2 ($2^1$) palabras por bloque

*Apartado a)*

Indica para una referencia

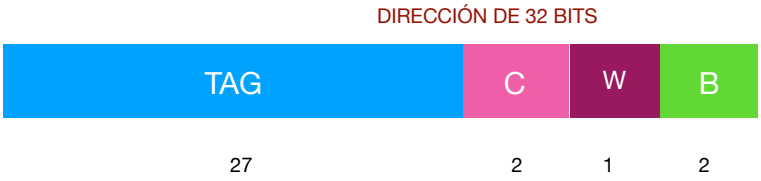
## ASIGNACIÓN/ORGANIZACIÓN DIRECTA

INDEX : $2^2$ bloque → 2 bits

W: $2^1$ palabras por bloque → 1 bits

B: $2^2$ (4) bytes por palabra → 2 bits

TAG: El resto de bytes = $2^{27}$ → 27 bits




## ORGANIZACIÓN TOTALMENTE ASOCIATIVA

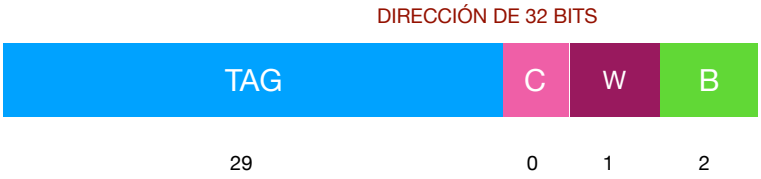
$2^5$ (tamaño de cache) : $2^3$ (bits de palabra totales) = $2^2$ bloques → $2^2$ bloques : $2^2$ asociatividad = 1 conjunto

CONJUNTO : $2^0$ (1) conjunto → 0 bits

W: $2^1$ palabras por bloque → 1 bits

B: $2^2$ (4) bytes por palabra → 2 bits

TAG: El resto de bytes = $2^{29}$ → 29 bits




## ORGANIZACIÓN ASOCIATIVA POR CONJUNTOS

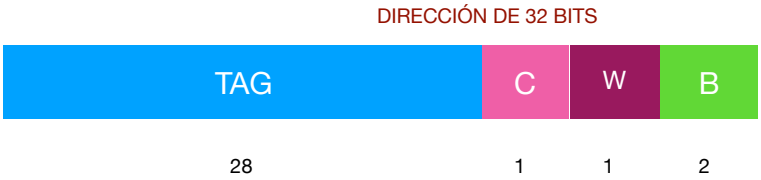
$2^5$ (tamaño de cache) : $2^3$ (bits de palabra totales) = $2^2$ bloques → $2^2$ bloques : 2 asociatividad = 2 conjuntos

CONJUNTO : $2^1$ conjuntos → 1 bits

W: $2^1$ palabras por bloque → 1 bits

B: $2^2$ (4) bytes por palabra → 2 bits

TAG: El resto de bytes = $2^{28}$ → 28 bits




*Apartado b)*

Suponiendo que el código se ejecuta un número de iteraciones muy grande (infinito), calcula para la ejecución completa:

## CODIGO A

*Recordatorio: El número a la izquierda de cada instrucción indica la dirección en memoria principal donde se encuentra.

Referencias a memoria del código A:

0, 4, 8, 12, 24, 28, 32, 36, 0, 4, 8, 12, …

Cache por organización directa

*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 BLOQUE = 00
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 BLOQUE = 00

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 BLOQUE = 01
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 BLOQUE = 01

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 BLOQUE = 11
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 BLOQUE = 11

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 BLOQUE = 00
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 BLOQUE = 00

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 4 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 8 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 12 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 24 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 28 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 32 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 36 [1]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 0 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 4 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 8 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 12 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

la vida es como una carrera de ingeniería:
hay que seguir hacia delante aunque
no sepas lo que estás haciendo :)

Referencia 24 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 28 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 32 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 36 [2]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

El índice de fallos cuando las iteraciones son infinitas será de 2 (fallos) / 8 (referencias) = 0,25

Cache por organización totalmente asociativa

*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 CONJUNTO = 00
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 CONJUNTO = 00

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 CONJUNTO = 00
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 CONJUNTO = 00

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 CONJUNTO = 00
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 CONJUNTO = 00

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 CONJUNTO = 00
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 CONJUNTO = 00

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 4 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 8 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 12 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 24 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | |

Referencia 28 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | |

Referencia 32 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | 32 y 36 |

Referencia 36 (1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | 32 y 36 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 0 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 4 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 8 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 12 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 24 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | |

Referencia 28 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | |

Referencia 32 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | 32 y 36 |

Referencia 36 (2)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 24 y 28 |
| 11 | 32 y 36 |

El índice de fallos cuando las iteraciones son infinitas será de 0 (fallos) / 8 (referencias) = 0

Cache por organización asociativa por conjuntos
*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 CONJUNTO = 0
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 CONJUNTO = 0

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 CONJUNTO = 1
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 CONJUNTO = 1

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 CONJUNTO = 1
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 CONJUNTO = 1

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 CONJUNTO = 0
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 CONJUNTO = 0

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | |
| 1 | | |

Referencia 4 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | |
| 1 | | |

Referencia 8 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | |

Referencia 12 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | |

Referencia 24 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | 24 y 28 |

Referencia 28 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | 24 y 28 |

Referencia 32 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 36 {1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 0 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 4 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 8 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 12 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 24 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 28 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 32 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 36 {2}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

El índice de fallos cuando las iteraciones son infinitas será de 0 (fallos) / 8 (referencias) = 0

A la vista del rendimiento, teniendo en cuenta el coste hardware recomendaría: Org.Asociativa por conjuntos

## CODIGO B

Referencias a memoria del código B:

0, 4, 8, 12, 32, 36, 24, 28, 32, 36, 24, 28 …

Cache por organización directa

*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 BLOQUE = 00
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 BLOQUE = 00

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 BLOQUE = 01
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 BLOQUE = 01

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 BLOQUE = 11
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 BLOQUE = 11

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 BLOQUE = 00
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 BLOQUE = 00

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 4 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 8 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 12 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 32 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 36 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 24 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 28 [1B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 32 [1B1]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 36 [1B1]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 24 [2B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 28 [2B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 32 [2B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

Referencia 36 [2B]

| BLOQUE | DATO |
|---|---|
| 0 | 32 y 36 |
| 1 | 8 y 12 |
| 10 | |
| 11 | 24 y 28 |

El índice de fallos cuando las iteraciones son infinitas será de 0 (fallos) / 4 (referencias) = 0

Cache por organización totalmente asociativa

*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 CONJUNTO = 00
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 CONJUNTO = 00

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 CONJUNTO = 00
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 CONJUNTO = 00

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 CONJUNTO = 00
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 CONJUNTO = 00

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 CONJUNTO = 00
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 CONJUNTO = 00

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 4 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | |
| 10 | |
| 11 | |

Referencia 8 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 12 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | |
| 11 | |

Referencia 32 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | |

Referencia 36 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | |

Referencia 24 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 28 (1B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 32 (1B1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 36 (1B1)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 24 (2B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 28 (2B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 32 (2B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

Referencia 36 (2B)

| VÍA | CONJUNTO 0 |
|---|---|
| 0 | 0 y 4 |
| 1 | 8 y 12 |
| 10 | 32 y 36 |
| 11 | 24 y 28 |

El índice de fallos cuando las iteraciones son infinitas será de 0 (fallos) / 4 (referencias) = 0

Cache por organización asociativa por conjuntos
*La parte subrayada indica el bloque en memoria principal. Cómo se puede observar, cada uno de los siguientes pares consecutivos de referencias forman parte del mismo bloque de M.P.

Referencia 0d = 0000 0000 0000 0000 0000 0000 0000 0000 CONJUNTO = 0
Referencia 4d = 0000 0000 0000 0000 0000 0000 0000 0100 CONJUNTO = 0

Referencia 8d = 0000 0000 0000 0000 0000 0000 0000 1000 CONJUNTO = 1
Referencia 12d = 0000 0000 0000 0000 0000 0000 0000 1100 CONJUNTO = 1

Referencia 24d = 0000 0000 0000 0000 0000 0000 0001 1000 CONJUNTO = 1
Referencia 28d = 0000 0000 0000 0000 0000 0000 0001 1100 CONJUNTO = 1

Referencia 32d = 0000 0000 0000 0000 0000 0000 0010 0000 CONJUNTO = 0
Referencia 36d = 0000 0000 0000 0000 0000 0000 0010 0100 CONJUNTO = 0

Evolución de los datos de la cache a través de las referencias de la primera iteración:

*Rojo indica fallo, verde indica acierto

Referencia 0 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | |
| 1 | | |

Referencia 4 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | |
| 1 | | |

Referencia 8 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | |

Referencia 12 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | | |

Referencia 32 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | |

Referencia 36 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | |

Referencia 24 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 28 {1B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 32 {1B1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 36 {1B1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Evolución de los datos de la cache a través de las referencias de la segunda iteración (que será igual al resto de iteraciones):

Referencia 24 {2B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 28 {2B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 32 {2B}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

Referencia 36 {1B1}

| VÍA | C 0 | C 1 |
|---|---|---|
| 0 | 0 y 4 | 8 y 12 |
| 1 | 32 y 36 | 24 y 28 |

El índice de fallos cuando las iteraciones son infinitas será de 0 (fallos) / 4 (referencias) = 0

A la vista del rendimiento, teniendo en cuenta el coste hardware recomendaría: Org.Asociativa por conjuntos

# PREGUNTA 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 d** y `lab1` **es igual a 140 d**. 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 cada una.

CARACTERÍSTICAS IMPORTANTES

- Tamaño de ambas caches = 128 bytes
- Tamaño de bloques = 8 bytes

Tamaño de cache → 128 bytes = $2^7$ bytes

Programa MIPS → Palabras de 4 Bytes → Bloques de 8 Bytes → $2^1$ palabras por bloque

$2^7$ bytes (tamaño de cache) : $2^1$ palabras por bloque * $2^2$ bytes por palabra = $2^4$ bloques

*Apartado a)*

Mostrar la secuencia de referencias (direcciones. a nivel de byte) que hace el programa durante su ejecución:

*a.1)* Secuencia de referencias a instrucciones de MP (en decimal): 0d, 4d, 140d, 144d, 148d, 152d, 156d, 8d, 12d, 16d,140d, 144d, 148d, 152d, 156d

*Explicación: Ya que es un programa MIPS, los datos se almacenan en palabras de 4 bytes. Para las instrucciones, sabemos que la dirección lab0 = 8d, y por lo tanto la anterior es 4d, y la que le sigue es 16d. De este mismo modo, al saber que lab1=140d, podemos deducir que las que le siguen son 144d, 148d, etc. Importante observar los saltos en el programa.

*a.2)* Secuencia de referencias a datos de MP (en decimal): 16d, 80d, 16d, 208d, 336d, 400d, 16d, 80d, 16d, 208d, 336d, 400d

*Apartado b)*

Muestra la evolución de cada una de las caches, indicando para cada referencia a un bloque de MP el número de conjunto de caché asignado y si se produce un fallo o un acierto.

b.1) **Caché de instrucciones (Asignación directa)**

Indica para una referencia:

INDEX : $2^4$ bloques → 4 bits

W: $2^1$ palabras por bloque → 1 bits

B: $2^2$ (4) bytes por palabra → 2 bits

TAG: El resto de bytes = $2^{25}$ → 25 bits

Para las referencias en el apartado a.1 indica:

| DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 0d | 0d | 0d | Fallo |
| 4d | 0d | 0d | Acierto |
| 140d | 17d | 1d | Fallo |
| 144d | 18d | 2d | Fallo |
| 148d | 18d | 2d | Acierto |
| 152d | 19d | 3d | Fallo |
| 156d | 19d | 3d | Acierto |
| 8d | 1d | 1d | Fallo |
| 12d | 1d | 1d | Acierto |
| 16d | 2d | 2d | Fallo |
| 140d | 17d | 1d | Fallo |
| 144d | 18d | 2d | Fallo |
| 148d | 18d | 2d | Acierto |
| 152d | 19d | 3d | Acierto |
| 156d | 19d | 3d | Acierto |

**EJEMPLO:**

4d → 0000 0000 0000 0000 0000 0000 0000 0100

W: 100= 4d

CONJUNTO = 0000 = 0d

TAG = 0000 0000 0000 = 0d

BLOQUE EN M.P = 0 0000 0000 0000 0000 0000 0000 0000 = 0d

b.1) **Caché de datos (Asociativa 4 conjuntos)**

$2^7$ (tamaño de cache) : $2^1$ palabras por bloque * $2^2$ bytes por palabra = $2^4$ bloques → $2^4$ bloques : $2^X$ asociatividad = $2^2$ conjuntos → X = 2 → Asociatividad $2^2$

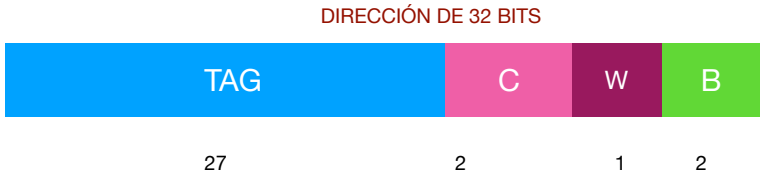
Indica para una referencia:

INDEX : $2^2$ conjuntos → 2 bits

W: $2^1$ palabras por bloque → 1 bits

B: $2^2$ (4) bytes por palabra → 2 bits

TAG: El resto de bytes = $2^{27}$ → 27 bits




Para las referencias en el apartado a.2 indica:

**EJEMPLO:**

16d → 0000 0000 0000 0000 0000 0000 0001 0000

W: 000 = 0d

CONJUNTO = 10 = 2d

TAG = 0000 0000 0000 = 0d

BLOQUE EN M.P = 0 0000 0000 0000 0000 0000 0000 0010 = 2d

| DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|
| 16d | 2d | 2d | Fallo |
| 208d | 26d | 2d | Fallo |
| 336d | 42d | 2d | Fallo |
| 400d | 50d | 2d | Fallo |
| 16d | 2d | 2d | Fallo |
| 80d | 10d | 2d | Fallo |
| 16d | 2d | 2d | Fallo |
| 208d | 26d | 2d | Fallo |
| 336d | 42d | 2d | Fallo |
| 400d | 50d | 2d | Fallo |

*Apartado c)*

Calcula las tasas de fallo (deja indicado numerador y denominador):

Tasa de fallos para la cache de instrucciones: 8 (fallos) / 15 (referencias)

Tasa de fallos para la cache de datos: 8 (fallos) / 10 (referencias)

# PREGUNTA 8

Sea el siguiente programa en MIPS:

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

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

Tamaño de cache = 4 Kbytes = $2^2 * 2^{10}$ bytes = $2^{12}$ bytes

Código MIPS → Palabras de 4 Bytes → $2^2$ tamaño de palabra

*Apartado a)*

Vamos a analizar primero cual es el tamaño de bloque óptimo teniendo en cuenta la tasa de fallos. Prueba distintos tamaños de bloque, empezando con 1 palabra por bloque hasta llegar al máximo número de palabras por bloque, y calcula sus tasas de fallo.

Posibles tamaños de bloques:

- 1 PALABRA = $2^{12} : 2^0$ (1 palabra) * $2^2$ (tamaño de palabra) = $2^{10}$ bloques
- 2 PALABRAS = $2^{12} : 2^1$ (2 palabras) * $2^2$ (tamaño de palabra) = $2^9$ bloques
- 4 PALABRAS = $2^{12} : 2^2$ (4 palabras) * $2^2$ (tamaño de palabra) = $2^8$ bloques
- 8 PALABRAS = $2^{12} : 2^3$ (8 palabras) * $2^2$ (tamaño de palabra) = $2^7$ bloques
- 16 PALABRAS = $2^{12} : 2^4$ (16 palabras) * $2^2$ (tamaño de palabra) = $2^6$ bloques
- 32 PALABRAS = $2^{12} : 2^5$ (32 palabras) * $2^2$ (tamaño de palabra) = $2^5$ bloques
- 64 PALABRAS = $2^{12} : 2^6$ (64 palabras) * $2^2$ (tamaño de palabra) = $2^4$ bloques
- 128 PALABRAS = $2^{12} : 2^7$ (128 palabras) * $2^2$ (tamaño de palabra) = $2^3$ bloques
- 256 PALABRAS = $2^{12} : 2^8$ (256 palabras) * $2^2$ (tamaño de palabra) = $2^2$ bloques
- 512 PALABRAS = $2^{12} : 2^9$ (512 palabras) * $2^2$ (tamaño de palabra) = $2^1$ bloques
- 1024 PALABRAS = $2^{12} : 2^{10}$ (1024 palabras) * $2^2$ (tamaño de palabra) = $2^0$ (1) bloque

Para cada uno de esos tamaños de bloque hay que probar las siguientes referencias del código y obtener sus respectivas tasas de fallo para saber cuál es menor:

3800h, 8800h, D800h, 37FCh, 87FCh, D7FCh…

EJEMPLO:

Para bloques de 1 palabra

3800h → 0000 0000 0000 0000 0011 1000 0000 0000 BLOQUE = 200
8800h → 0000 0000 0000 0000 1000 1000 0000 0000 BLOQUE = 200
D800h → 0000 0000 0000 0000 1101 1000 0000 0000 BLOQUE = 200
37FCh → 0000 0000 0000 0000 0011 0111 1111 1100 BLOQUE = IFF
87FCh → 0000 0000 0000 0000 1000 0111 1111 1100 BLOQUE = IFF
D7FCh → 0000 0000 0000 0000 1101 0111 1111 1100 BLOQUE = IFF
…

Debido a fallos obligatorios y de conflicto, obtendríamos una tasa de fallos de 1.

Algo muy similar ocurre con todos los otros tamaños de bloque. Aunque reduzcamos el tamaño de bloque, cada 3 direcciones consecutivas estarán almacenadas en el mismo bloque. Como tenemos conflictos de grado 3, necesitaríamos asociatividad por 4 conjuntos para solucionarlo.

Por lo tanto, para todo tamaño de bloque, con asociatividad directa, siempre obtendremos una tasa de fallos de 1.

¿Cuál es la mejor tasa de fallos que obtienes? R) 1

*Apartado 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?

256 bytes por bloque = 64 palabras (4 bytes cada una) por bloque

$2^{12}$ (tamaño de cache) : $2^6$ (palabras por bloque) * $2^2$ (tamaño de palabra) = $2^4$ bloques
$2^4$ bloques → $2^2$ (asociatividad) = $2^2$ conjuntos

Como hemos comentado en el apartado anterior, como tenemos conflictos de grado 3, necesitamos asociatividad 4 para resolverlos.

Para calcular el menor número de fallos obtenidos:

3800h → 0000 0000 0000 0000 0011 1000 0000 0000 CONJUNTO = 00 F
8800h → 0000 0000 0000 0000 1000 1000 0000 0000 CONJUNTO = 00 F
D800h → 0000 0000 0000 0000 1101 1000 0000 0000 CONJUNTO = 00 F
37FCh → 0000 0000 0000 0000 0011 0111 1111 1100 CONJUNTO = 11 F
87FCh → 0000 0000 0000 0000 1000 0111 1111 1100 CONJUNTO = 11 F

Las referencias subrayadas pertenecen al mismo bloque de memoria principal, por lo tanto habrá un acierto.

D7FCh → 0000 0000 0000 0000 1101 0111 1111 1100 CONJUNTO = 11 F
37F8h → 0000 0000 0000 0000 0011 0111 1111 1000 CONJUNTO = 11 A
87F8h → 0000 0000 0000 0000 1000 0111 1111 1000 CONJUNTO = 11 A
D7F8h → 0000 0000 0000 0000 1101 0111 1111 1000 CONJUNTO = 11 A
37F4h → 0000 0000 0000 0000 0011 0111 1111 0100 CONJUNTO = 11 A

Hemos accedido al principio de 3 diferentes bloques y estamos accediendo a todas sus palabras de manera consecutiva. De forma que tendremos 63 * 3 aciertos consecutivos + 3 fallos hasta que se terminen las iteraciones.

Para cada 400h iteraciones, tendremos por cada matriz, 1 Fallo + 16 (1 Fallo + 63 Aciertos). Esto es 17 fallos.

Como tenemos 3 matrices, tendremos 17 * 3 fallos en total.

Menor número de fallos totales obtenidos = 51 fallos

Apartado c)

Para la configuración elegida en el apartado anterior, ¿Qué política de reemplazo es mejor, LRU o FIFO?

Política de reemplazo óptima = Da igual

# PREGUNTA 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).

CÓDIGO A
```
i=1;
while (i <=10) {
  j=1;
  while (j <=10) {
     A[i][j]=0.0;
     j=j+1;
  }
  i=i+1;
}
```

CÓDIGO B
```
i=1;
while (i <=10) {
  j=1;
  while (j <=10) {
     A[j][i]=0.0;
     j=j+1;
  }
  i=i+1;
}
```

Se dispone de un sistema de memoria cuya **caché de datos 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. Para ello contesta a las siguientes preguntas:

El enunciado nos dice que tenemos una cache equivalente a 10 números reales. Cada posición [i] [j] de la matriz referencia a un número real. Y en cada bloque podemos almacenar 2 números reales.

Por lo tanto la cache se vería así

| BLOQUE | DATO |
|---|---|
| 0 | [1] [1] Y [1] [2] |
| 1 | [1] [3] Y [1] [4] |
| 10 | [1] [5] Y [1] [6] |
| 11 | [1] [7] Y [1] [8] |
| 101 | [1] [9] Y [1] [10] |

## CÓDIGO A

En el código A, la secuencia de elementos referenciados es:

A(1,1), A(1,2), A(1,3)…A(10,10)

Tendríamos un índice de fallos del 50% ya que estamos accediendo a posiciones de filas consecutivas que están almacenadas de manera consecutiva.

## CÓDIGO B

En el código B, la secuencia de elementos referenciados es:

A(1,1), A(2,1), A(3,1)…A(10,10)

Tendríamos un índice de fallos del 100% ya que estamos accediendo a posiciones de columnas consecutivas, y en nuestra cache tenemos a los datos guardados por filas consecutivas.

# PREGUNTA 10

Queremos ejecutar un bucle simple como éste:

```
for (i=0; i<=9; i++)
   A[i] = 0;
```

en un procesador con una **memoria caché de datos 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 000h y que **cada elemento del array ocupa 1 byte**, muestra la evolución en la caché de datos de ese código para las siguientes organizaciones:

CARACTERÍSTICAS IMPORTANTES

- Tamaño de cache = 8 bytes
- Tamaño de bloques = 2 bytes

Cache de 8 = $2^3$ bytes
3 Bloques en cache → 4 bits para direccionarlos → $2^2$ bits
Cada elemento ocupa 1 byte ($2^0$)

## ORGANIZACIÓN DIRECTA

ÍNDEX: 2 bits
W: 1 bit

EJEMPLO:

A(0) = 000h = 0000 0000 000 CONJUNTO = 00 BLOQUE M.P = 000 0000 0000 = 000h

| DATO | DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|---|
| A[0] | 000h | 000h | 0h | Fallo |
| A[1] | 001h | 000h | 0h | Acierto |
| A[2] | 002h | 001h | 1h | Fallo |
| A[3] | 003h | 001h | 1h | Acierto |
| A[4] | 004h | 002h | 2h | Fallo |
| A[5] | 005h | 002h | 2h | Acierto |
| A[6] | 006h | 003h | 3h | Fallo |
| A[7] | 007h | 003h | 3h | Acierto |
| A[8] | 008h | 004h | 0h | Fallo |
| A[9] | 009h | 004h | 0h | Acierto |

Calcula el índice de aciertos (con dos decimales) : 0,50

## ORGANIZACIÓN TOTALMENTE ASOCIATIVA

CONJUNTO: 0 bits
W: 1 bit

EJEMPLO:
A(1) = 001h = 0000 0000 0001 CONJUNTO = 00 BLOQUE M.P = 000 0000 0000 = 000h

| DATO | DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|---|
| A[0] | 000h | 000h | 0h | Fallo |
| A[1] | 001h | 000h | 0h | Acierto |
| A[2] | 002h | 001h | 0h | Fallo |
| A[3] | 003h | 001h | 0h | Acierto |
| A[4] | 004h | 002h | 0h | Fallo |
| A[5] | 005h | 002h | 0h | Acierto |
| A[6] | 006h | 003h | 0h | Fallo |
| A[7] | 007h | 003h | 0h | Acierto |
| A[8] | 008h | 004h | 0h | Fallo |
| A[9] | 009h | 004h | 0h | Acierto |

Calcula el índice de aciertos (con dos decimales) : 0,50

# ORGANIZACION ASOCIATIVA POR CONJUNTOS

CONJUNTO: 1 bit
W: 1 bit

EJEMPLO:
A(1) = 001h = 0000 0000 0001 CONJUNTO = 00 BLOQUE M.P = 000 0000 0000 = 000h

| DATO | DIRECCIÓN MP | BLOQUE EN MP | CONJUNTO | ACIERTO/ FALLO |
|---|---|---|---|---|
| A[0] | 000h | 000h | 0h | Fallo |
| A[1] | 001h | 000h | 0h | Acierto |
| A[2] | 002h | 001h | 1h | Fallo |
| A[3] | 003h | 001h | 1h | Acierto |
| A[4] | 004h | 002h | 0h | Fallo |
| A[5] | 005h | 002h | 0h | Acierto |
| A[6] | 006h | 003h | 1h | Fallo |
| A[7] | 007h | 003h | 1h | Acierto |
| A[8] | 008h | 004h | 0h | Fallo |
| A[9] | 009h | 004h | 0h | Acierto |

Calcula el índice de aciertos (con dos decimales) : 0,50

*Apartado a)*

¿Qué tipo de organización de memoria cache (directa, por conjuntos o totalmente asociativa) elegirías en función del porcentaje de aciertos? Nota: a igualdad de rendimiento se debe elegir la menos costosa desde el punto de vista HW): Org. Directa

*Apartado b)*

¿Se podría decir lo mismo independientemente de la posición de comienzo del array A en memoria?: Si, porque todos los fallos son obligatorios

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):

```
for (i=0; i<=9; i++)
   A[i] = B[9-i];
```

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. Realiza un esquema de la evolución en la caché de datos de este código, similar al código anterior, y responde a las siguientes preguntas:

Para este debemos hacer lo mismo que en el anterior y además tener en cuenta los accesos al Array B.

Tenemos que acceder a las direcciones:

B(9) = 69,
A(0) = 0,
B(8) = 68,
A(1) = 1,
B(7) = 67,
A(2) = 2,
B(6) = 66,
A(3) = 3,
B(5) = 65,
A(4) = 4,
B(4) = 64,
A(5) = 5,
B(3) = 63,
A(6) = 6,
B(2) = 62,
A(7) = 7,
B(1) = 61,
A(8) = 8,
B(0) = 60

Para obtener el índice de aciertos, hay que descomponer las direcciones en sus respectivos campos y observar sí ocurren fallos y aciertos.

Índice de aciertos para organización directa = 0,30

Índice de aciertos para organización totalmente asociativa = 0,50

Índice de aciertos para organización asociativa por conjuntos = 0,50

*Apartado c)*

¿Qué tipo de organización de memoria cache elegirías ahora?: Org. Asociativa por conjuntos

*Apartado d)*

¿Es independiente la respuesta anterior de la posición de comienzo del array B en memoria?

No, porque se puede escoger una posición inicial de B con la que se evitan los conflictos en la asignación directa