Boletín 2 de ejercicios resueltos del tema 6

PROBLEMA 1 Se dispone de un procesador MIPS R2000, con espacio de direccionamiento de 4 GB, que contiene una memoria cache de datos de 8 KB, con una organización por conjuntos de dos vías. El tamaño del bloque es de 16 bytes. La memoria cache utiliza el algoritmo LRU para los reemplazos y posee políticas de escritura directa (*write through*) y de ubicación en escritura (*write allocate*). En dicho sistema se ejecuta el siguiente código:

```
          .data 0x01004000
vector:   .space 12288          # 3072 elementos de una palabra

          .text 0x00400000
          ori $t1, $zero,2      # Contador de bucle externo
bucle1:   lui $t0, 0x0100
          ori $t0, $t0, 0x4000  # $t0 apunta a vector
          ori $t2, $zero, 3072  # Contador del bucle interno
bucle2:   lw $t3, 0($t0)
          add $t3, $t3, $t3
          sw $t3, 0($t0)
          addi $t0, $t0, 4
          addi $t2, $t2, -1
          bne $t2, $zero, bucle2
          addi $t1, $t1, -1
          bne $t1, $zero, bucle1
```

Calcule:

1. Tamaño de la información de control necesaria para gestionar la memoria cache.
2. Número de accesos a la memoria cache de datos una vez se ha ejecutado todo el código.
3. Número de bloques de memoria que contienen el vector y dirección del primer bloque que contiene elementos del vector.
4. Tasa de aciertos de la memoria cache.
5. Suponga que la memoria cache usara una política de reemplazo de tipo aleatorio (el bloque a reemplazar se elige de forma aleatoria). Indique si la tasa de aciertos del programa anterior variaría (aumentando o disminuyendo) respecto a la política de reemplazo LRU. **Justifique la respuesta.**
6. Por último, suponga que la memoria cache implementara una política de no ubicación en escritura. Indique si la tasa de aciertos del programa anterior variaría (aumentando o disminuyendo) respecto a la política de ubicación en escritura. **Justifique la respuesta.**

PROBLEMA 2 Un computador basado en el MIPS R2000 tiene un único nivel de memoria cache. Esta memoria cache es unificada, esto es, contiene tanto datos como código. La política de escritura es posterior (*write back*) y sin ubicación (*write no allocate*). La dirección emitida por el procesador es interpretada por la memoria cache de la siguiente manera: 19 bits como etiqueta, 8 bits como línea y el resto como desplazamiento dentro del bloque. Considere que el computador ejecuta el programa siguiente:

```
          .text 0x00401000
__start:  addi $t0, $zero, 5000
          addi $t1, $zero, 1
          addi $t2, $zero, $zero
bucle:    add  $t2, $t2, $t1
          addi $t1, $t1, 1
          addi $t0, $t0, -1
          bnez $t0, bucle
```

1. Calcule el número total de accesos al sistema de memoria que efectúa el procesador durante la ejecución del programa.
2. Calcule el volumen del directorio (etiquetas y bits de control) de la memoria cache. Indique el resultado en bits.
3. Determine el número de líneas de memoria cache que ocupa el programa.
4. Calcule la tasa de aciertos H obtenida por el programa. Suponga que inicialmente la memoria cache está vacía.

PROBLEMA 3 Se dispone de un procesador MIPS R2000 con un único nivel de memoria caché con las características que se muestran en la tabla:

Capacidad	16 KB + 16 KB
Tamaño de bloque	16 B
Correspondencia	Directa
Política de escritura	Posterior (*write back*), sin ubicación (*non allocate*)

Se ejecuta el siguiente programa en el sistema anterior:

```
          .data 0x10000000
Vec_A:    .word 1,2,3,5,6,8,10,11,12,0,2,6,7,9,1,0
Vec_B:    .space 64
Dato:     .word 12

          .text 0x00400000
          la $t0, Dato          # Puntero a Dato
          lw $t2, 0($t0)        # Cargamos el valor
          la $t0, Vec_A         # Puntero a Vec_A
          la $t1, Vec_B         # Puntero a Vec_B
          li $t4, 16            # Contador de elementos
    bucle:lw $t3, 0($t0)        # Acceso al elemento Vec_A
          add $t3, $t2, $t3
          sw $t3, 0($t1)        # Escribo el elemento Vec_B
          addi $t0, $t0, 4
          addi $t1, $t1, 4
          addi $t4, $t4, -1
          bne $t4, $zero, bucle
          .end
```

1. Indique la estructura de la dirección en la cache, es decir, cuántos bits determinan cada uno de los campos atendiendo a la función de correspondencia.
2. Indique los bloques a los que pertenecen las instrucciones del programa anterior. Rellene una tabla como la que se muestra a continuación, indicando en qué línea y con qué etiqueta se encontrará cada uno de los bloques de instrucciones.
   NOTA: Las pseudointrucciones se traducen por una instrucción de código máquina, y cada instrucción se codifica en 4 Bytes.

Bloque de MP	Etiqueta	Línea

3. Calcule el volumen total del directorio de la cache. Expréselo en Kbit.

4. Indique el número total de accesos (datos + instrucciones) que se realiza al ejecutar el código anterior. Observe que el bucle se repite 16 veces.

5. Calcule el número de accesos (lecturas y escrituras) que recibe la memoria cache de instrucciones. Determine la tasa de aciertos resultante.

PROBLEMA 4 Un computador basado en el MIPS R2000 tiene un único nivel de memoria cache. Esta memoria cache es unificada, es decir, contiene tanto datos como código. La capacidad de la memoria cache es de 16 KB, la correspondencia es directa, el tamaño de bloque de 16 bytes y la política de escritura es directa (*write through*) y sin ubicación (*write no allocate*). Considere que el computador ejecuta el programa siguiente:

```
          .data 0x10000000
          .word 1, 2, 3, ..., 512

          .text 0x00401000
__start:  addi $t0, $zero, 512
          lui $t1, 0x1000
bucle:    lw $t2, 0($t1)
          add $t2, $t2, $t2
          sw $t2, 0($t1)
          addi $t1, $t1, 4
          addi $t0, $t0, -1
          bnez $t0, bucle
```

1. Calcule el número de accesos al sistema de memoria que efectúa el procesador durante la ejecución del programa. Nótese que se han de contabilizar tanto los accesos al código como los accesos a los datos.

2. Indique la interpretación de las direcciones (campos y longitud) que lleva a cabo la memoria cache.

3. Calcule el volumen del directorio (etiquetas y bits de control) de la memoria cache. Indique el resultado en bits.

4. Calcule la tasa de aciertos H obtenida por el programa. Suponga que inicialmente la memoria cache está vacía.

5. Durante la cena de fin de año un informático nos dice que si el sistema de memoria cache de nuestro computador estuviera implementado con dos memorias cache separadas (datos y código) la **tasa de aciertos** obtenida por el programa anterior sería más elevada. ¿Tiene razón este ingeniero? Justifique al respuesta.

PROBLEMA 5 Se disponde de un procesador con espacio de direccionamiento de 4 GB y memoria caché L1 de instrucciones de 16 KB de capacidad, correspondencia directa y tamaño de línea de 16 bytes. El procesador tambien dispone de una memoria caché L1 de datos de 16 KB de capacidad, asociativa por conjuntos de 4 vías, tamaño de línea de 16 bytes, escritura directa y ubicación en escritura. Las dos memorias caché están inicialmente vacías. En dicho procesador se ejecuta el siguiente código:

```
          .data 0x10000000
indices:  .word 0x12000020, 0x12000124, 0x12000040, 0x120000F0
vector:   .space 16
suma:     .word 0

          .text 0x00400000
          la $t0, indices       # Puntero a indices
          li $t1, 4             # Cargamos el número de índices
```

```
        la $t2, vector        # Puntero a vector
        add $t3, $0, $0       # Suma parcial
bucle:  lw $t4, 0($t0)
        lw $t4, 0($t4)
        sw $t4, 0($t2)
        add $t3, $t3, $t4
        addi $t0, $t0, 4
        addi $t2, $t2, 4
        addi $t1, $t1, -1
        bne $t1, $zero, bucle
        la $t2, suma          # Puntero a suma
        sw $t3, 0($t2)        # Almacenamos suma final
       .end
```

1. Indique el número de accesos del procesador a la memoria caché de datos al ejecutar por completo el código.

2. Calcule la tasa de aciertos de la memoria caché de instrucciones al ejecutar el código. Suponga que todas las instrucciones listadas en el código ocupan 4 bytes en memoria.

3. Calcule la tasa de aciertos de la memoria caché de datos al ejecutar el código. Para facilitar la resolución se propone rellenar la siguiente tabla con los accesos a datos.

Dirección	Etiqueta	Conjunto	Despl.	Acierto	Fallo	Comentario
0x10000000	..	..	..	..	F	Lectura indice[0]
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
Total				..	..	$H = \ldots$

Tabla 1: Direcciones accedidas de datos y tasa de aciertos de la memoria caché de datos.

4. Calcule cual sería la tasa de aciertos de la memoria caché de datos si se cambia la política de fallo de escritura por no ubicación en escritura. **Justifique la respuesta.**

PROBLEMA 6 En el ámbito del procesamiento de señales, aparece a menudo la necesidad de transformar un vector de entrada $E[i]$ en otro de salida $S[i]$. El cálculo que se aplica para obtener el elemento $S[n]$ es un producto escalar entre los k elementos $E[n-k+1] \ldots E[n]$ y un vector de k coeficientes $C[1] \ldots C[k]$. El conjunto de k elementos necesarios para el cálculo se llama *ventana*. Algebraicamente, este cálculo se expresa así:

$$S[n] = \sum_{i=0}^{k-1} E[n-i] \cdot C[k-i]$$

Para calcular los sucesivos elementos de S, la ventana de los k elements de E avanza de posición, pero el vector C es constante para todo el proceso. La figura 1 muestra gráficamente los datos implicados.

En un lenguaje de alto nivel, si los vectores E, C y S son del tipo *float* (coma flotante de simple precisión codificado en 32 bits), el cálculo de los elementos de S se especifica así:

```
for(int n = inicial; n <= final; n++){
    float t = 0;
    for(int i = 0; i < k; i++)
        t = t + E[n-i] * C[k-i];
    S[n] = t;
}
```

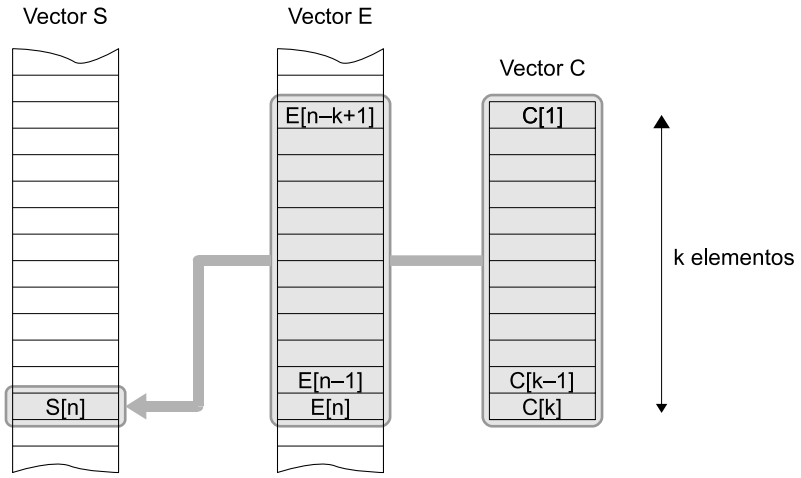

Figura 1: Esquema del cálculo de $S[n]$

Considera ahora que estos cálculos se hacen con un MIPS R2000 dotado de una memoria cache de datos de 32 KB, dos vías y política de escritura directa sin ubicación (es decir, *write-through* i *write-not allocate*). Los bloques son de 32 bytes. El compilador ubica en registros del procesador todas las variables escalares que aparecen en el código (es decir, n, t y i) y coloca los tres vectores en la memoria principal haciendo que cada uno comience al principio de un bloque de memoria (es decir, que $E[0]$ y $C[1]$ están al principio de un bloque). La longitud de la ventana es constante y su valor $k = 6$ también está contenido en un registro.

1. Calcula los parámetros de la memoria cache que faltan: la capacidad de la memoria cache en bloques y el número de conjuntos que contiene. Qué campos se encuentran en la dirección? Cuántos elementos de vector caben en un bloque?

2. Supón que la memoria cache se encuentra vacía. Si se ejecuta el código anterior con $inicial = final = 14$, calcula:

   a) Los elementos del vector E a los que accede el programa.
   b) El número de accesos de lectura y escritura de datos.
   c) El número de bloques de cada vector que se cargarán en la memoria.
   d) Las tasas de aciertos de lectura y de escritura en la memoria cache.

3. Suponiendo el estado de la memoria cache resultante del apartado 2, se ejecuta el código con $inicial = 15$ y $final = 16$. Calcula de nuevo:

   a) Los elementos del vector E a los que accede el programa.
   b) El número de accesos de lectura y escritura de datos.
   c) El número de bloques de cada vector que se cargarán en la memoria.
   d) Las tasas de aciertos de lectura y de escritura en la memoria cache.

SOLUCIÓ:

1. Càlcul dels paràmetres de la memòria cau:

$$\text{Capacitat en blocs} = \frac{32 \text{ KB totals}}{32 \text{ bytes/bloc}} = 1024 \text{ blocs}$$

$$\text{Nombre de conjunts} = \frac{1024 \text{ blocs}}{2 \text{ vies}} = 512 \text{ conjunts}$$

Estructura de l'adreça: 18 bits d'etiqueta, 9 bits de conjunt i 5 de desplaçament

En cada bloc caben 8 elements de vector.

2. Execució amb $inicial = final = 14$:

   a) Elements d'E als què accedeix el programa: $E[9] \dots E[14]$

   b) El nombre d'accessos: 6 accessos de lectura a $E + 6$ accessos de lectura a $C + 1$ accés d'escriptura a S.

   c) El programa llig dels vectors E i C. Els 6 elements de C caben en un bloc que es carregarà durant la primera fallada de lectura. Els elements $E[9] \dots E[14]$ també caben en un bloc. El bloc que conté el component $S[14]$ no es carregarà en la memòria a causa de la política *write not-allocate* d'escriptura. En total, el programa carregarà dos blocs.

   d) Taxes d'encerts: Es produiran dues fallades de lectura i una d'escriptura.

$$\text{TE (lectura)} = \frac{10 \text{ encerts}}{12 \text{ accessos}} = 0{,}83$$

$$\text{TE (escriptura)} = \frac{0 \text{ encerts}}{1 \text{ accés}} = 0$$

3. Execució amb $inicial = 15$ i $final = 16$:

   a) Elements d'E als què accedeix el programa: $E[10] \dots E[16]$ (la iteració $n = 15$ accedeix a $E[10] \dots E[15]$ i la iteració $n = 16$ a $E[11] \dots E[16]$.

   b) El nombre d'accessos: 12 accessos de lectura a $E + 12$ accessos de lectura a $C + 2$ accessos d'escriptura a S.

   c) Els 6 elements de C estan en el bloc present en la memòria cau. Dels 7 components d'E llegits, només $E[15]$ i $E[16]$ són nous. $E[15]$ està en el bloc que s'ha carregat en la cau en l'apartat 2, però $E[16]$ està en un altre bloc. En total, el programa carregarà el nou bloc del vector E.

   d) Taxes d'encerts: Es produiran una fallada de lectura i dues d'escriptura.

$$\text{TE (lectura)} = \frac{24 - 1 \text{ encerts}}{24 \text{ accessos}} = 0{,}96$$

$$\text{TE (escriptura)} = \frac{2 - 2 \text{ encerts}}{2 \text{ accessos}} = 0$$

■

PROBLEMA 7 Un computador basado en el MIPS R2000 tiene dos niveles de memoria cache. El primer nivel, L1, es dual (instrucciones y datos) y el segundo, L2, es unificado. He aquí sus parámetros:

	L1	L2
Capacidad	4 KB + 4 KB	128 KB
Tamaño de bloque	16 B	16 B
Correspondencia	Directa	Conjuntos de 2 vías
Reemplazo	—	LRU
Política de escritura	Directa, sin ubicación	Posterior, con ubicación

Cargado en la memoria principal está el programa siguiente:

```
            .data 0x00001230
V:          .space 40
            .data 0x00002230
            .space 40

            .text 0x00011230
__start:    li $t0,10
            la $t1,V
bucle:      lw $t2,0($t1)
            lw $t3,0x1000($t1)
```

```
add $t2,$t2,$t3
sw $t2,0x1000($t1)
addi $t1,$t1,4
addi $t0,$t0,-1
bnez $t0,bucle
```

El programa opera con dos vectores de 10 componentes de tipo *word*, el primero de dirección $V = 0x01230$ y el segundo de dirección $V + 0x1000$. Tenga en cuenta que cada pseudoinstrucción del código genera sólo una instrucción máquina. Suponga que todas las memorias cache están vacías.

1. Calcule la etiqueta de todos los bloques ocupados por las instrucciones del programa en cada nivel de memoria cache. ¿A qué líneas de L1 está asociado cada bloque? ¿A qué conjuntos de L2 está asociado cada bloque?

2. Calcule la etiqueta de todos los bloques ocupados por los datos del programa en cada nivel de la memoria cache y a qué líneas o conjuntos están asociados.

3. Calcule el número de accesos que hace el procesador a la cache L1 de instrucciones al ejecutar este programa. ¿Cuál es la tasa de aciertos resultante? No olvide que el programa contiene un bucle que se ejecuta 10 veces.

4. Calcule el número de accesos (lecturas y escrituras) que recibe la memoria cache L1 de datos y clasifique los fallos entre los tres tipos (de inicio, de capacidad y de conflicto). Calcule la tasa de aciertos resultante cuando se ejecuta este programa.

5. En total, entre instrucciones y datos, ¿cuántos accesos de lectura y escritura recibe la memoria cache L2? Clasifique los fallos y calcule la tasa de aciertos de L2.

SOLUCIÓ:

1. Etiqueta i línia dels blocs ocupats en L1 d'instruccions. Per al control de L1, les adreces s'interpreten segons la figura 2.

Figura 2: Estructura de l'adreça per a L1

Com que cada bloc pot contenir fins a quatre instruccions i la primera està ubicada en l'adreça $0x011230$, les 9 instruccions ocuparan tres blocs: $0x1123$, $0x1124$ i $0x1125$ (vegeu la figura 3).

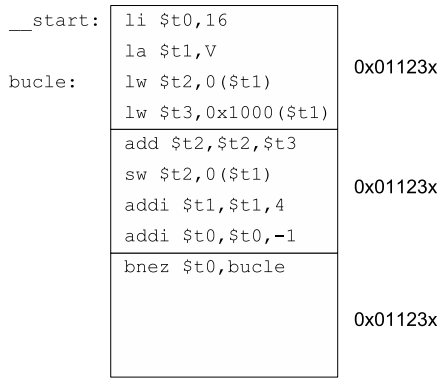

Figura 3: Distribució de les instruccions en blocs

Els blocs tenen tots la mateixa etiqueta $0x00011$ i van associats a les línies $0x23$, $0x24$ i $0x25$ de L1.

2. Etiqueta i conjunts d'instruccions en la cau L2. El control de L2 estructura les adreces com es mostra en la figura 4. En conseqüència, els blocs tindran l'etiqueta $0x0001$ i van associats als conjunts $0x123$, $0x124$ i $0x125$ de L2.

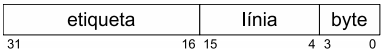

Figura 4: Estructura de l'adreça per a L2

3. Taxa d'encerts en L1 d'instruccions. Atenent el codi, s'executaran $2 + 10 \times 7 = 72$ instruccions. Per tant, L1 patirà 72 accessos.

   Com que la memòria cau està buida, tots tres blocs referenciats produiran fallades d'inici. Com que no hi ha conflictes entre ells i caben fins a 256 blocs en L1, no hi haurà d'altres fallades. Per tant, dels 72 accessos, 3 seran fallades i la resta encerts. És a dir,

$$H_{L1}(\text{instruccions}) = \frac{72-3}{72} = 95{,}8\,\%$$

4. Etiqueta i línia dels blocs en L1 de dades. Cada vector ocupa 3 blocs, perquè en cada bloc caben quatre components. El vector d'adreça V comença en el bloc $0x123$ i acaba en el bloc $0x125$. Per tant, atenent la figura 2 tots tenen la mateixa etiqueta $0x0001$ i estan associats a les línies $0x23$, $0x24$ i $0x25$.

   El vector d'adreça $V + 0x1000$ comença en el bloc $0x223$ i acaba en el bloc $0x225$. L'etiqueta de tots tres blocs és $0x00002$ i estan associats a les línies $0x23$, $0x24$ i $0x25$.

5. Etiqueta i línia dels blocs en L2 de dades. Tots els blocs ocupats pels dos vectors tenen la mateixa etiqueta $0x0000$. El vector V està associat als conjunts 123 a 125 i el vector $V + 0x1000$ als conjunts 223 a 225.

6. Taxa d'encerts en L1 de dades. El programa fa 20 accessos de lectura i 10 d'escriptura sobre els 6 blocs. Els dos vectors entren en conflicte perquè ocupen les mateixes linies de la memòria cau L1 de dades. Com que el programa referencia alternativament ambdós vectors, tots els accessos de lectura seran fallades; com que totes les escriptures van seguint una lectura en la mateixa adreça, seran encerts.

   Només el primer accés a cada bloc és una fallada d'inici; la resta són de conflicte. En conseqüència, hi haurà 6 fallades d'inici i 14 de conflicte. La taxa d'encerts serà

$$H_{L1}(\text{dades}) = \frac{30-20}{30} = \frac{10}{30} = 33{,}3\,\%$$

7. Taxa d'encerts de lectura en L2. L2 rep 3 accessos de lectura per causa de les fallades d'inici en L1 d'instruccions. Pel que fa a les dades, hi ha un total de 20 accessos de lectura per fallades de lectura en L1, més 10 fallades perquè les escriptures en L1 de dades es propaguen a L2 immediatament per raó de la política d'*escriptura directa*. En total, n'hi arriben 33 accessos.

   No hi ha conflictes entre els blocs presents en L2. Així que només hi ha fallades d'inici, una per cada bloc referenciat. En conseqüència,

$$H_{L2} = \frac{33-9}{33} = 72{,}7\,\%$$

■

PROBLEMA 8 Un procesador MIPS R2000 dispone de un primer nivel de memoria cache con dos memorias separadas (datos y código) de 4 KB cada una de ellas. Las características de las mismas son las siguientes:

- Cache de datos: correspondencia directa, línea (bloque) de 16 bytes, escritura posterior (*write back*) y ubicación en escritura (*write allocate*).

- Cache de código: 2 vías, línea (bloque) de 256 bytes y algoritmo de reemplazo LRU.

Considere la ejecución del siguiente programa:

```
            .data 0x10000000                # Segmento de datos
datos:      .byte 0x45, 0x22, 0x69, 0x83    # Valores de origen
            .byte 0x72, 0x15, 0x39, 0x00    # Valores de origen (acaba en cero)
valor:      .word 0                         # Resultado del cómputo

            .text 0x400000                  # Segmento de código
__start:    lui $t0, 0x1000                 # Dirección de los datos
            add $t2, $zero, $zero           # Cálculos intermedios

bucle:      lb $t1, 0($t0)                  # Lectura de un dato
            add $t2, $t2, $t1               # Cálculo intermedio
            addi $t0, $t0, 1                # Incremento de dirección
            bne $t1, $zero, bucle           # Si no es final ir a bucle

            lui $t1, 0x1000                 # Dirección del resultado (parte alta)
            ori $t1, $t1, 0x0008            # Dirección del resultado (parte baja)
            sw $t2, 0($t1)                  # Almacenamiento del resultado
            .end                            # Final del programa
```

1. Indique la interpretación de las direcciones que efectúa la memoria cache de código.
2. Determine el volumen del directorio (etiquetas y bits de control) de la memoria cache de datos. Indique el resultado en bits.
3. Calcule la tasa de aciertos producida por el programa para cada una de las memorias cache (suponga que inicialmente ambas están vacías).

SOLUCIÓ:

1. Para el desplazamiento dentro del bloque necesitamos 8 bits ($2^8 = 256$). La cache tiene $2^{12}/2^8 = 2^4 = 16$ líneas, y como hay 2 vías, el número de conjuntos será de $16/2 = 8$; así pues, hacen falta 3 bits para indicar el conjunto. Los $32 - 3 - 8 = 21$ bits restantes conformarán la etiqueta. En definitiva, tenemos una disposición de $32 = 21 + 3 + 8$ (de mayor a menor peso).
2. La memoria cache de datos interpreta la dirección como $32 = 20 + 8 + 4$ (los bloques son de $16 = 2^4$ bytes y hay $2^{12}/2^4 = 2^8 = 256$ líneas). Habrá un bits de válido por cada línea; dado que usa una política de escritura posterior (*writeback*) también habrá un bit de modificado (*dirty bit*); no hay bits para el algoritmo de reemplazo porque la correspondencia es directa. En consecuencia el volumen del directorio ocupará $256 \times (20 + 1 + 1) = 5632$ bits.
3. El programa ejecuta ocho veces las instrucciones del bucle. En total hay 9 accesos a datos (ocho accesos de lectura de byte para leer los datos y un acceso de escritura de palabra para almacenar el resultado). Los datos en memoria del programa ocupan $8 + 4 = 12$ bytes ubicados a partir de la dirección $1000\ 0000_{16}$. Atendiendo a las características de la memoria cache de datos, todos caben en un bloque; en particular, se trata del bloque de memoria con etiqueta 10000_{16} que se ubica en la línea 0 de cache. Dado que el primer acceso de lectura produce un fallo, una vez traído el bloque el resto de accesos serán aciertos; así pues, la tasa de aciertos en la cache de datos es de $H = 8/9 = 0{,}889$.

   En cuanto a la memoria cache de código, en total hay $2 + 8 \times 4 + 3 = 37$ accesos al segmento de código. Por otro lado, dado el tamaño del bloque de línea (256 bytes), la totalidad de las instrucciones cabrán en un bloque; en particular, el código se almacenará en el bloque de memoria 004000_{16}, es decir, se ubica en el conjunto 0 con etiqueta 800_{16}. Desde el punto de vista de la tasa de aciertos el comportamiento es similar al ocurrido en la cache de datos: hay un fallo en la lectura de la primera instrucción y el resto de accesos, una vez traído el bloque, serán aciertos. La tasa de aciertos es, por lo tanto, $H = 36/37 = 0{,}973$. ■

PROBLEMA 9 El siguiente programa rellena un vector de 256 enteros con los valores $v_i = v_{i-1} + v_{i-2}$, donde $v_0 = 1$ y $v_1 = 2$.

```
        .data 0x10000000
    V:  .word 1,2,0,0,0...   # 256 valores en total
        ...

        .text
        li $t0,8           # Desplazamiento del tercer elemento del vector
        li $t5,1024        # Final del vector
bucle:  lw $t1,V-8($t0)    # Cargar V(i-2)
        lw $t2,V-4($t0)    # Cargar V(i-1)
        add $t3,$t1,$t2    # $t3 = V(i-1) + V(i-2)
        sw $t3,V($t0)      # Almacenar V(i)
        addi $t0,$t0,4     # Incrementar $t0
        bne $t0,$t5,bucle
```

El programa se ejecuta sobre un sistema que dispone de una memoria cache de datos de 1 KB de capacidad, tamaño de bloque de 16 bytes y correspondencia asociativa por conjuntos de 2 vías.

1. Calcule la tasa de aciertos del programa anterior, suponiendo que la cache sigue una política de no ubicación en escritura.
2. Calcule ahora la tasa de aciertos suponiendo que la cache sigue una política de ubicación en escritura.
3. Suponiendo que la tasa de aciertos fuera $H = 0{,}9$ calcule el tiempo medio de acceso a los datos, teniendo en cuenta que las lecturas o escrituras en cache cuestan 5 ns, mientras que las lecturas o escrituras en memoria principal cuestan 100 ns.

SOLUCIÓ:

1. Tasa de aciertos: El programa va calculando cada uno de los valores del vector de la siguiente forma:

Iteración	Operación
1	$v_2 = v_1 + v_0$
2	$v_3 = v_2 + v_1$
3	$v_4 = v_3 + v_2$
4	$v_5 = v_4 + v_3$
5	$v_6 = v_5 + v_4$
6	$v_7 = v_6 + v_5$
7	$v_8 = v_7 + v_6$
...	...
252	$v_{253} = v_{252} + v_{251}$
253	$v_{254} = v_{253} + v_{252}$
254	$v_{255} = v_{254} + v_{253}$

   En cada línea de cache caben 4 enteros. En la primera iteración se produce un fallo al acceder a v_0. Este fallo provoca la carga en cache de los elementos v_0 hasta v_3, por lo que no hay más fallos en las iteraciones 1 y 2.

   El acceso a v_4 en la tercera iteración resulta en un fallo de escritura que, como indica el enunciado, no carga el bloque que lo contiene. En la cuarta iteración, el acceso a v_4 es en lectura, por lo que se ubica el bloque que contiene a v_4, v_5, v_6 y v_7, con lo que no hay más fallos hasta la iteración 7, donde se produce un fallo en escritura (que no ubica), lo que producirá un fallo de lectura en la iteración 8. Y así sucesivamente.

   El número de bloques de cache a los que se accede es 64, ya que el vector ocupa 1024 bytes y cada bloque es de 16 bytes ($1024/16 = 64$). En el primer bloque se produce únicamente un fallo en lectura, mientras que en cada uno de los 63 restantes se producen dos fallos: uno de escritura seguido de otro de lectura. Así pues, el número de fallos es de $63 \times 2 + 1 = 127$.

El número de accesos a datos que realiza es de 3 accesos $\times$ 254 iteraciones $=$ 762 accesos. Por lo que el número de aciertos es $762 - 127 = 635$ y la tasa de aciertos resultante es:

$$H = \frac{635 \text{ aciertos}}{762 \text{ accesos}} = 0{,}83$$

2. Tasa de aciertos: En este caso, los fallos de escritura producen ubicación en cache, por lo que no se produce el fallo de lectura que siempre se producía en el caso anterior. Por ello, el número de fallos es 64, uno por bloque. Así pues, el número de aciertos es de $762 - 64 = 698$, con lo que la tasa de aciertos es de:

$$H = \frac{698 \text{ aciertos}}{762 \text{ accesos}} = 0{,}92$$

3. Tiempo medio de acceso a memoria: $0{,}9 \times 5 \text{ ns} + 0{,}1 \times 100 \text{ ns} = 14{,}5 \text{ ns}$ ■

PROBLEMA 10 Un computador basado en el MIPS R2000 tiene un único nivel de memoria cache. Esta memoria cache es unificada, es decir, contiene tanto datos como código. La capacidad de la memoria cache es de 16 KB, la correspondencia es asociativa por conjuntos de 4 vías, el tamaño de bloque es de 16 bytes y la política de escritura es posterior (*write back*) y sin ubicación (*write no allocate*). El algoritmo de reemplazo utilizado es LRU. Considere que el computador ejecuta el programa siguiente:

```
          .data 0x108000A0
A:        .word 1, 2, 3, ..., 512

          .text 0x00040000
__start:  addi $t0, $zero, 512
          lui $t1, 0x1080
          ori $t1, $t1, 0xA0
bucle:    lw $t2, 0($t1)
          srl $t2, $t2, 1
          sw $t2, 0($t1)
          addi $t1, $t1, 4
          addi $t0, $t0, -1
          bnez $t0, bucle
```

1. Indique la interpretación de las direcciones (campos y longitud) que lleva a cabo la memoria cache.
2. Calcule el volumen del directorio (etiquetas y bits de control) de la memoria cache. Indique el resultado en bits.
3. Indique el número de bloques que ocupa el programa y el vector `A`, especificando en cada caso en qué conjuntos de la memoria caché irán ubicados.
4. Calcule la tasa de aciertos H obtenida por el programa. Suponga que inicialmente la memoria cache está vacía.
5. Indique qué le ocurrirá a la tasa de aciertos H si el vector `A` se hubiera ubicado en la dirección de memoria `0x10800000`. Justifique la respuesta.

PROBLEMA 11 Considere la operación de invertir el contenido de un vector V de 20 elementos de tipo word. Expresada en alto nivel, la operación se escribe

$i = 0;\ j = n - 1;$
iterar
    $t = V[i];\ V[i] = V[j];\ V[j] = t;$
    $i = i + 1;\ j = j - 1;$
mentre $i < j$

En ensamblador, el vector V está ubicado en la dirección `0x10010000` y el fragmento de programa está ubicado a partir de `0x00400150`.

```
invertir:
       lui $t0,4097    # la $t0,V
       addi $t1,$t0,76 # dirección del último elemento
bucle: lw $t2,0($t0)
       lw $t3,0($t1)
       sw $t3,0($t0)
       sw $t2,0($t1)
       addi $t0,$t0,4
       addi $t1,$t1,-4
       slt $t4,$t0,$t1
       bnez $t4,bucle
```

Este programa se ejecuta en un procesador dotado de la memoria cache dual siguiente:

Capacidad de 16 KB para instrucciones y 16 KB para datos

Bloque de 32 bytes

Asociatividad de 4 vías, remplazo LRU

Política de escritura de la memoria cache de datos: directa *(write-through)* sin ubicación *(write-not allocate)*

1. Considere que todos los bloques de las dos memorias están marcadaos como inválidos cuando el procesador comienza su ciclo de instrucción con CP = `0x00400150`. ¿En qué conjunto se carga el primer bloque traído de la memoria de instrucciones, y con qué etiqueta? ¿En qué conjunto se carga el primer bloque traído de la memoria de datos, y con qué etiqueta?

2. ¿Cuántos fallos de memoria cache de código se habrán producido al acabar la primera iteración del bucle? ¿Y cuántos fallos de memoria cache de datos? ¿Cuántas escrituras se han hecho en la memoria principal? ¿Qué cambiaría si la caché de datos fuera de escritura posterior *(write-back)* con ubicación *(write-allocate)*?

3. ¿Cuántos accesos a instrucción hace el procesador al ejecutar el código `invertir`? Y cuántos accesos a datos? ¿Cuál es la tasa de aciertos de cada memoria caché? Y cuál es la tasa de aciertos global?

4. Justo al acabar la ejecución del código `invertir` un cambio de contexto hace que todos los bloques de la memoria caché sean remplazados. ¿Cuántas operaciones de escritura en la memoria principal se producirán si la memoria cache de datos fuera de escritura directa sin ubicación? ¿Y si fuera de escritura posterior con ubicación?

PROBLEMA 12 Un sistema basado en el procesador MIPS R2000 posee una memoria cache de datos de 1 MB de capacidad, con tamaño de línea de 16 bytes, 2 vías, política de escritura directa, con ubicación en escritura, y algoritmo de reemplazo LRU.

En dicho sistema se ejecuta el siguiente código, el cual realiza una suma de tres vectores, de tal forma que el elemento del vector resultado (R) es la suma de los tres elementos de los vectores operandos (A, B, C), es decir $R[i] = A[i] + B[i] + C[i]$. El tamaño de los vectores es de 512 elementos enteros de tipo word.

```
    .data 0x10000000
A:  .space 2048
    .data 0x20000000
B:  .space 2048
    .data 0x30000000
C:  .space 2048
    .data 0x40000000
R:  .space 2048
```

```
        .text 0x00400000
        la $t0, A
        la $t1, B
        la $t2, C
        la $t3, R
        li $t4, 512
bucle:  lw $t5, 0($t0)
        lw $t6, 0($t1)
        lw $t7, 0($t2)
        add $t5, $t5, $t6
        add $t5, $t5, $t7
        sw $t5, 0($t3)
        addi $t0, $t0, 4
        addi $t1, $t0, 4
        addi $t2, $t0, 4
        addi $t3, $t0, 4
        addi $t4, $t4, -1
        bne $t4, $0, bucle
        .end
```

1. Determine la interpretación de las direcciones por parte de la memoria cache.
2. Calcule el número de accesos al sistema de memoria que se realizarán por el acceso a datos.
3. Calcule el número de accesos que se realizarán a la memoria principal por el acceso a datos.
4. Indique la tasa de aciertos H de la memoria cache.
5. Calcule de nuevo el número de accesos a memoria principal y la tasa de aciertos si ampliamos a 4 el número de vías de la memoria cache sin aumentar su capacidad.