ESTRUCTURA DE COMPUTADORES

Control de clase Tema 6

Apellidos y Nombres

Un sistema basado en procesador MIPS R2000 posee una cache L1 dual configurada como sigue:

- **Cache de Instrucciones**: 1 KB, correspondencia asociativa por conjuntos de 4 vías
- **Cache de Datos**: 4 KB, correspondencia directa, política de NO ubicación en escritura (***write-no allocate***) con actualización posterior (*write-back*)
  Ambas cache poseen un tamaño de bloque de **8 bytes** y usan LRU para los reemplazos

El procesador ejecuta el siguiente código en alto nivel y ensamblador, respectivamente:

```
byte A[60];
...
for (int i=0; i<60; i=i+22)
    A[i]=A[i] & 0xF0;
for (int i=0; i<60; i=i+15)
  -   A[i]=0;
- - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
        .data 0x20180000
A:      .space 60
        ...
        .text0x01400000
        la $t0,A          # $t0 apunta al principio del vector
        addi $t1,$t0,60   # $t1 apunta al final del vector
for1:   lb $t2,0($t0)
        andi $t2,$t2,0xF0
        sb $t2,0($t0)
        addi $t0,$t0,22
        blt $t0,$t1,for1
        la $t0,A          # $t0 apunta de nuevo al principio del vector
for2:   sb $zero,0($t0)
        addi $t0,$t0,15
        blt $t0,$t1,for2
```

Nótese que la pseudo-instrucción *blt $t0,$t1,eti* (aparece dos veces en el programa) se ensamblará **en dos instrucciones máquina**.
Supóngase que todas las líneas de ambas caches son inválidas inicialmente. Analice el comportamiento de las cache de instrucciones y de datos durante la ejecución del código anterior

Vector A:

0/0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

Cache de Instrucciones

¿Cuántos bloques ocupan las instrucciones del programa?	7
Calcule el número de accesos a instrucciones	bucle *for1* itera 3 veces (i=0, 22, 44); bucle *for2* itera 4 veces (i=0, 15, 30, 45); en total: 2 + 3x6 + 1 + 4x4 = 37 accesos
¿Cuál es la tasa de aciertos de la cache de instrucciones al ejecutar este código?	Al haber 7 fallos, el total de aciertos será de 30 30/37= 0,81

Cache de Datos

¿Cuántos bloques ocupa el vector A?	8
¿Con qué etiqueta se almacenarán estos bloques en la memoria cache?	0x20180
¿Cuál es el número de accesos a la cache de datos?	bucle *for1*: 3x2 accesos bucle *for2*: 4 accesos Total 10 accesos
¿Cuántos fallos se producirán en la cache de datos al ejecutar el bucle *for1* del código?	3 fallos:
¿Qué componentes del vector A fallan?	componentes 0, 22, y 44
¿Cuántos fallos se producirán en la cache de datos al ejecutar el bucle *for2* del código?	2 fallos:
¿Qué componentes del vector A fallan?	componentes 15 y 30. componentes 0 y 45 aciertan
¿Cuántas escrituras se harán sobre la memoria cache de datos?	3 escrituras durante *for1* 2 escrituras durante *for2*
¿A qué componentes del vector A afectan?	*for1*: 0, 22, 44; *for2*: 0, 45
¿Cuántas escrituras se harán sobre la memoria principal?	2 escrituras durante for2
¿A qué componentes del vector A afectan?	for2: 15 y 30
¿Cuál es la tasa de fallos de la memoria cache de datos?	(3+2)/10= 0.5
¿Cuántos bloques quedarán como válidos en la cache al término de la ejecución del código?	Los tres bloques que fallan en *for1*
¿Cuántos bloques quedarán modificados en la cache al término de la ejecución del código?	Los tres bloques que quedan válidos en la cache
¿Cuál es el tamaño en bits de la memoria de control? (*Indíquese el tamaño en bits de cada línea, justificándolo, y el total de líneas que la componen)*	[1(V) + 1(M) + 20(etiqueta)]bits x 512 líneas= 11264 bits

Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

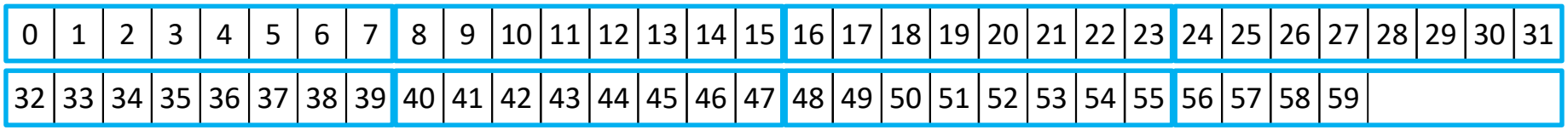
Vector A:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59				

```
byte A[60];
   ...
   for (int i=0; i<60; i=i+22)
        A[i]=A[i] & 0xF0;
   for (int i=0; i<60; i=i+15)
     -   A[i]=0;
```

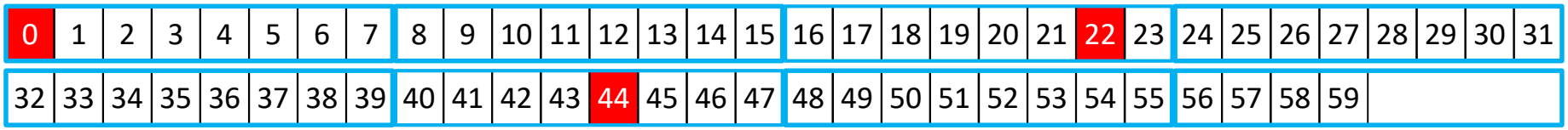
Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



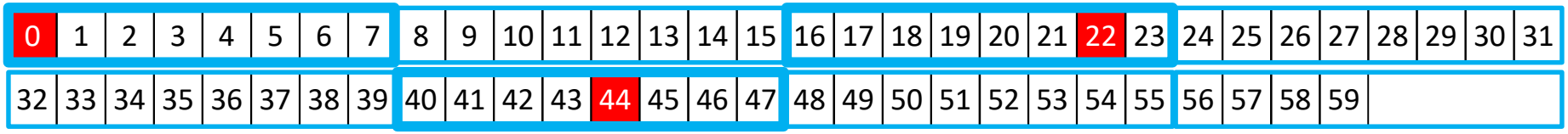
Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



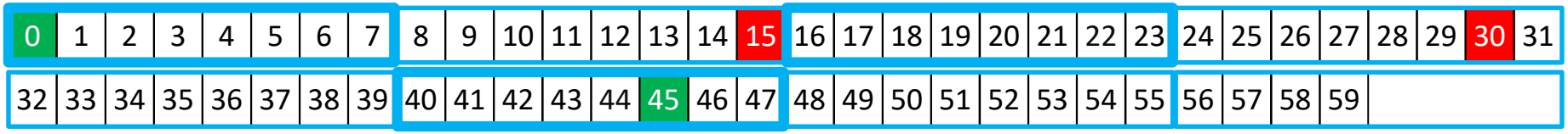
Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



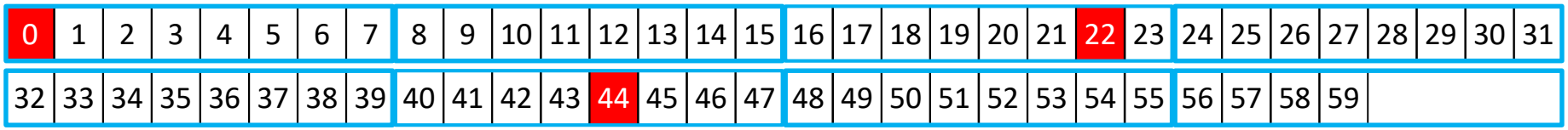
Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:

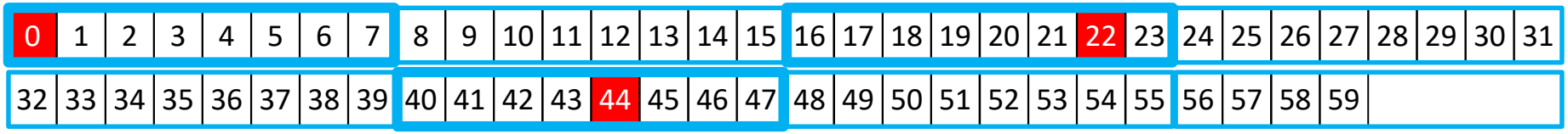


Bucle *for1*: 0, 22, 44 (se leen y escriben)

3 fallos al leer; 3 aciertos al escribir;

Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



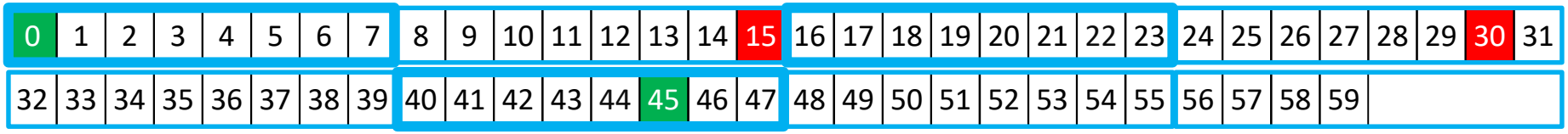
Bucle *for1*: 0, 22, 44 (se leen y escriben)

3 fallos al leer; 3 aciertos al escribir;

Los bloques que fallan quedan almacenados en la cache tras su lectura

Cache Datos: 4KB, correspondencia directa, 8 bytes, *write-no allocate*

Vector A:



Bucle *for1*: 0, 22, 44 (se leen y escriben)

3 fallos al leer; 3 aciertos al escribir;
Los bloques que fallan quedan almacenados en la cache tras su lectura

Bucle *for2*: 0, 15, 30, 45 (solo escritura)

2 fallos y 2 aciertos al escribir;
Los bloques que fallan NO se almacenan en la cache (*write-no allocate*)
Los elementos 15 y 30 se escriben directamente a memoria principal